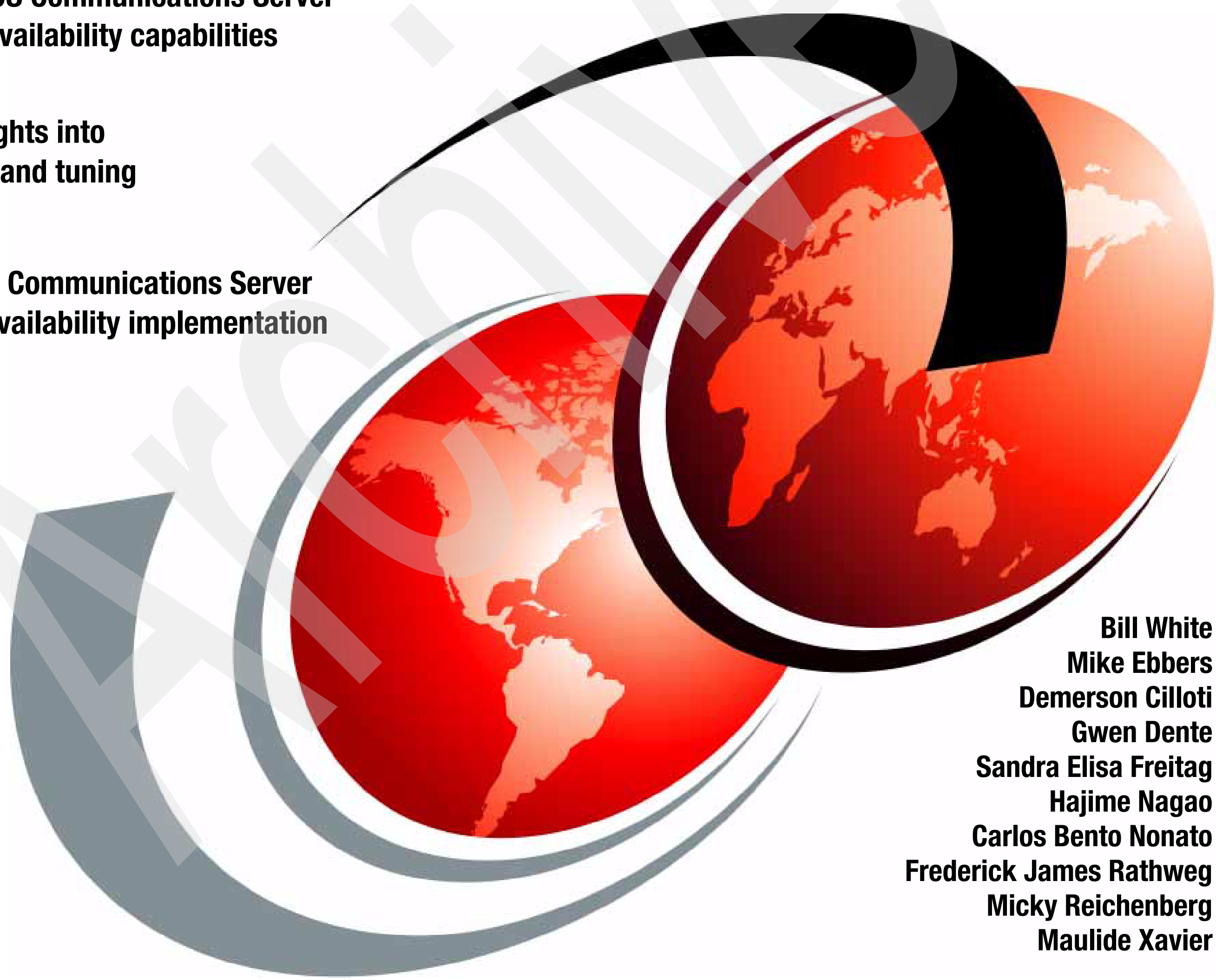
IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 3:
High Availability, Scalability, and Performance

Discusses z/OS Communications Server TCP/IP high availability capabilities

Provides insights into performance and tuning

Includes z/OS Communications Server TCP/IP high availability implementation examples



Bill White
Mike Ebbers
Demerson Cilloti
Gwen Dente
Sandra Elisa Freitag
Hajime Nagao
Carlos Bento Nonato
Frederick James Rathweg
Micky Reichenberg
Maulide Xavier

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance

February 2010

SG24-7800-00

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

First Edition (February 2010)

This edition applies to Version 1, Release 11, of z/OS Communications Server.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

CICS®
DataPower®
DB2®
ESCON®
FICON®
HiperSockets™
IBM®
IMS™
OMEGAMON®
Parallel Sysplex®
RACF®
Redbooks®
Redbooks (logo) ®
System z10™
System z9®
System z®
Tivoli®
VTAM®
WebSphere®
z/OS®
z9®

The following terms are trademarks of other companies:

Catalyst, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

For more than 40 years, IBM® mainframes have supported an extraordinary portion of the world's computing work, providing centralized corporate databases and mission-critical enterprise-wide applications. The IBM System z®, the latest generation of the IBM distinguished family of mainframe systems, has come a long way from its IBM System/360 heritage. Likewise, its IBM z/OS® operating system is far superior to its predecessors, providing, among many other capabilities, world-class, state-of-the-art, support for the TCP/IP Internet protocol suite.

TCP/IP is a large and evolving collection of communication protocols managed by the Internet Engineering Task Force (IETF), an open, volunteer, organization. Because of its openness, the TCP/IP protocol suite has become the foundation for the set of technologies that form the basis of the Internet. The convergence of IBM mainframe capabilities with Internet technology, connectivity, and standards (particularly TCP/IP) is dramatically changing the face of information technology and driving requirements for ever more secure, scalable, and highly available mainframe TCP/IP implementations.

The *IBM z/OS Communications Server TCP/IP Implementation* series provides understandable, step-by-step guidance about how to enable the most commonly used and important functions of z/OS Communications Server TCP/IP.

In this IBM Redbooks® publication, we begin with a discussion of Virtual IP Addressing (VIPA), a TCP/IP high-availability approach that was introduced by the z/OS Communications Server. We then show how to use VIPA for high availability, both with and without a dynamic routing protocol. We also discuss a number of different workload balancing approaches that you can use with the z/OS Communications Server. We also explain the optimized Sysplex Distributor intra-sysplex load balancing. This function represents improved multitier application support using optimized local connections together with weight values from extended Workload Manager (WLM) interfaces. Finally, we highlight the most important tuning parameters and suggest parameter values that we observed to maximize performance in many client installations.

> **Note:** In this book, we use the terms *internal* and *external* application workload balancing to refer to approaches where the decision as to which application instance should received a given connection request is made within the sysplex environment (such as by Sysplex Distributor) or outside of it (using a separate, external, workload balancing solution), respectively.

For more specific information about z/OS Communications Server base functions, standard applications, and security, refer to the other volumes in the series:

- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798
- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7799
- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7801

For comprehensive descriptions of the individual parameters for setting up and using the functions described in this book, along with step-by-step checklists and supporting examples, refer to the following publications:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780

This book does not duplicate the information in those publications. Instead, it complements them with practical implementation scenarios that can be useful in your environment. To determine at what level a specific function was introduced, refer to *z/OS Communications Server: New Function Summary*, GC31-8771. For complete details, we encourage you to review the documents referred to in "Related publications" on page 307.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Bill White is a Project Leader and Senior Networking Specialist at the ITSO, Poughkeepsie Center.

Mike Ebbers is a Project Leader and Consulting IT Specialist at the ITSO, Poughkeepsie Center. He has worked for IBM since 1974 on mainframe projects, including time as an SNA Specialist.

Demerson Cilloti is a System Programmer at Banco do Brasil. He has 5 years of experience in mainframe networking and holds a Bachelor's degree in Information Systems. His areas of expertise include cross-platform integration, network design, and System z networking.

Gwen Dente is a Consulting IT Specialist with IBM Advanced Technical Support at the Washington Systems Center in Gaithersburg, Maryland, U. S. She focuses on System z networking and security, assisting customers and the IBM technical community with cross-platform integration and network design. Gwen presents frequently at SHARE and other IBM technical conferences and has shared authorship of multiple IBM Redbooks publications.

Sandra Elisa Freitag is an IT Specialist in IBM Brazil. She has 19 years of experience in IBM software support first as an IBM customer and for the last 10 years with IBM. Her areas of expertise include z/OS Communications Server, SNA and TCP/IP, and WebSphere® MQ and Message Broker on all platforms. She provides support to clients on z/OS network, SNA/TCP/IP Integration, and z/OS Connectivity,

Hajime Nagao is an IT Specialist in IBM Japan. He has 13 years experience in designing, implementing, and maintaining network infrastructure for customers. Recently, he has focused on z/OS networking, answering product-specific question, and providing consultation about customer systems as a member of a technical support team for z/OS Communications Server.

Carlos Bento Nonato is an Information Technology Analyst in the Management of Network and Telecommunication at Banco do Brasil S/A. He earned an MBA in Management Information Systems from the Catholic University of Brasília. In 2004, he published his thesis on IT Management by Projects: a case study. In 1999, he built an application on Artificial Intelligence (Expert Systems) for the implementation of the PCC—Work and Career Plan at the State University of Mato Grosso do Sul, Brazil.

Frederick James Rathweg is an independent consultant. His many years of technical experience as an IBM Systems Programmer, Project Leader, Network Communications Designer and Architect have led to his current passion: B2B E-commerce, integrating diverse businesses to build successful enterprises. His current technical focus is z/OS system migration, cross platform integration, and migrating SNA networks to OSA Enterprise Extender. Fred is a retired Infantry Officer (US Army 1973-1993). He holds a Bachelor's degree (BA) in Business Management from St. Leo College, an MBA from Webster University, and is a part time Sivananda YOGA teacher.

Micky Reichenberg is an independent consultant with more than 35 years of experience in mainframe networking. He specializes in mainframe TCP/IP, SNA, open systems connectivity to the mainframe, Enterprise Extender design and implementation. Prior to becoming a consultant, Micky was a systems programmer with IBM in Israel for 17 years. During his assignment with the ITSO at the Raleigh Center, he published five networking-related IBM Redbooks publications. He holds a Bachelor's degree in Aeronautical Engineering from the Techion Israel Institute of Technology. Micky has also co-authored other IBM Redbooks publications.

Maulide Xavier is an IT Specialist in IBM Portugal. He has 11 years of experience in IBM mainframe and networking systems. His current responsibilities include network-related problem solving in System z and supporting clients to implement a network infrastructure in complex environments. His areas of expertise include z/OS Communications Server, SNA, and TCP/IP networking. He holds a diploma in Business and Economics Applied Math from Instituto Superior de Economia and Gestão and a post graduation in Management of Information Systems from the same school. He is also a member of the IBM Networking Software Support in Europe and a Cisco Certified Network Associate (CCNA).

Thanks to the following people from the ITSO, Poughkeepsie Center for their contributions to this project: David Bennin, Emma Jacobs, Rich Conway, and Bob Haimowitz.

Thanks to the following people for their technical advice and support

As is always the case with any complex technical effort, success would not have been possible without the advice, support, and review of many outstanding technical professionals. We are especially grateful for the significant expertise and contributions of content to this book from the z/OS Communications Server development team and other Communications Server experts:

David Herr, Communications Server development, IBM Raleigh CS

Finally, we want to thank the authors of the previous *z/OS Communications Server TCP/IP Implementation* series for creating the groundwork for this series: Rama Ayyar, Valirio Braga, WenHong Chen, Gwen Dente, Gilson Cesar de Oliveira, Mike Ebbers, Octavio Ferreira, Marco Giudici, Adi Horowitz, Michael Jensen, Shizuka Katoh, Sherwin Lake, Bob Louden, Garth Madella, Yukihiko Miyamoto, Shuo Ni, Yohko Ojima, Roland Peschke, Joel Porterie, Marc Price, Micky Reichenberg, Larry Templeton, Rudi van Niekerk, Bill White, Thomas Wienert, and Andi Wijaya.

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks
- Send your comments in an e-mail to:

  redbooks@us.ibm.com
- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Introduction to z/OS Communications Server high availability technologies

Organizations invest in information systems because the value they receive (through automation of business processes) is greater than the cost. Depending upon the business processes supported, certain availability, response times, and throughput must be provided for an application in order for it to be useful. For example, an unreliable credit card authorization system would be useless. Meeting the service levels required for each application, however, comes at a potentially significant cost in terms of additional redundancy, capacity, and complexity. By consolidating services onto System z mainframe hosts, critical mass can be achieved wherein investments to provide higher availability for all applications are more easily justified.

This book provides guidance on implementing the most important high-availability capabilities of z/OS Communications Server for TCP/IP applications. Because scalability and performance concerns often manifest themselves as availability issues (such as when application response times degrade to the point where the application is unusable), the book addresses those topics as well.

In the case of scalability, the same architectural solutions that are used for high availability (running multiple instances of an application and balancing load across them) are extended to scale application services to serve larger numbers of users and higher transaction volumes than can be supported by a single application instance.

This chapter discusses the following topics.

Section	Topic
1.1, "Overview of high availability" on page 2	An overview of high availability
1.2, "Fundamental technologies for z/OS TCP/IP availability" on page 2	Technologies that are important for z/OS TCP/IP availability
1.3, "Quick start table" on page 11	Cross-reference to technologies that we discuss in this book

1.1 Overview of high availability

For the purposes of this book, *high availability* is the ability of an application to continue to provide its services in spite of component failures. High availability is achieved through choosing highly reliable system components (such as System z mainframes), and by understanding and avoiding *single points of failure*, which are places in the underlying infrastructure where the failure of a single component can impact the availability of an entire application (or set of applications).

> **Note:** In some cases, the cost of eliminating a single point of failure exceeds the expected cost of outages (estimated as the cost of a single outage multiplied by the probability of occurrence). For example, it might be too expensive to justify either of the following:
>
> - Providing redundant high-speed wide area network (WAN) connectivity for certain small locations
> - Rewriting an application so that it can be distributed across multiple hosts (to prevent a host failure from impacting application availability)
>
> In such cases, the organization chooses to *assume* the risk of outage.

Redundancy, by itself, does not necessarily provide higher availability. It is also necessary to design and implement the system using technologies that can take advantage of the redundancy. For example:

- Dynamic routing protocols (such as OSPF) enable networks to take advantage of redundant connectivity to route traffic around failed links or nodes.
- Virtual IP Addressing (VIPA) technology (discussed in Chapter 2, "Virtual IP addressing" on page 13) can be used to support having multiple instances of an application in a z/OS environment to provide continuing application services in spite of the failure of individual instances (or mainframe systems).

1.2 Fundamental technologies for z/OS TCP/IP availability

This section introduces several important technologies for z/OS TCP/IP availability. Several of these technologies, along with implementation examples, are discussed in more detail in subsequent chapters.

1.2.1 Single z/OS system availability

Based on the remarkable reliability of IBM System z mainframe technology, many organizations have found that they can cost-effectively achieve higher application availability and scalability with a single mainframe system than would be possible with many smaller *distributed* systems. Although most of the technology solutions discussed in this book take advantage of clusters of z/OS systems, some also apply in single-system environments. They include:

- Virtual IP Addressing (VIPA)

  VIPA provides physical interface independence for the TCP/IP stack (the part of a z/OS Communications Server software that provides TCP/IP protocol support) and applications so that interface failures will not impact application availability. Refer to 1.2.3, "Virtual IP addressing" on page 4, and Chapter 2, "Virtual IP addressing" on page 13 for more information about this topic.

- Address Resolution Protocol (ARP) takeover

  ARP enables your system to transparently exploit redundant physical interfaces without implementing a dynamic routing protocol in your mainframe. Refer to Chapter 3, "VIPA without dynamic routing" on page 45 for more information about this topic.

- Dynamic routing

  Dynamic routing leverages network-based routing protocols (such as OSPF) in the mainframe environment to exploit redundant network connectivity for higher availability (when used in conjunction with VIPA). Refer to Chapter 4, "VIPA with dynamic routing" on page 65 for more information about this topic.

- Portsharing

  Portsharing enables you to run multiple instances of an application for higher availability and scalability (to the extent possible in a single system). Refer to 5.1.3, "Portsharing" on page 99 for more information about this topic.

1.2.2 z/OS Parallel Sysplex availability

z/OS Parallel Sysplex® combines parallel processing with data sharing across multiple systems to enable you to harness the power of plural z/OS mainframe systems, yet make these systems behave like a single, logical computing facility. This combination gives the z/OS Parallel Sysplex unique availability and scalability capabilities.

The overall objective of designing a Parallel Sysplex environment for high availability is to create an environment where the loss of any single component does not affect the availability of the application. This is achieved by designing a fault-tolerant systems environment in which:

- Hardware and software failures are masked by redundant design.
- Hardware and software systems are configured such that all systems have access to all devices, thereby enabling workloads to run anywhere in the sysplex.
- Sysplex Distributor, the strategic IBM solution for connection workload balancing within a sysplex, balances workloads and logons among systems that implement multiple concurrent application instances, each sharing access to data.
- Recovery events are automated so that when a failure does occur, end-user impact is minimized by fast-restart mechanisms.

> **Note:** For applications that cannot support multiple concurrent instances, the best that you can do (short of reengineering the application) is to minimize the duration of outages by leveraging automation, such as:
>
> - Automatic Restart Manager (ARM) for quick application server restart
> - Dynamic VIPA takeover for transferring application services to a standby server

Nodes in a sysplex use XCF Communication Services (XCF) to perform coordination among Sysplex TCP/IP stacks, to discover when new TCP/IP stacks are started, and to learn when a TCP/IP stack is stopped or leaves the XCF group (such as resulting from some sort of failure). That information is essential for automating the movement of applications and for enabling Sysplex Distributor to make intelligent workload balancing decisions. XCF communication messages can be transported either through a Coupling Facility or directly through ESCON® or FICON® CTCs.

Note: Dynamic VIPA and Sysplex Distributor capabilities do not rely on data stored in structures in the Coupling Facility. Therefore, they can be implemented using XCF communication without a Coupling Facility (also called *Basic Sysplex connectivity*).

1.2.3 Virtual IP addressing

In TCP/IP networking, Internet Protocol (IP) addresses are typically assigned to physical network interfaces. If a server has two physical interfaces, a separate IP address is assigned to each of them. IBM introduced the concept of Virtual IP Addressing (VIPA) for its z/OS environment to support the use of IP addresses, representing TCP/IP stacks, applications, or clusters of applications, that are not tied to any specific physical interface. The association between a VIPA and an actual physical interface is subsequently accomplished using either the Address Resolution Protocol (ARP) or dynamic routing protocols (such as OSPF).

We discuss VIPA technology in Chapter 2, “Virtual IP addressing” on page 13. Consequently, we introduce only the basic concepts here.

Static VIPA

A *static* VIPA is an IP address that is associated with a particular TCP/IP stack. Using either ARP takeover or a dynamic routing protocol (such as OSPF), static VIPAs can enable mainframe application communications to continue unaffected by network interface failures. As long as a single network interface is operational on a host, communication with applications on the host will persist.

Note: Static VIPA does not require sysplex (XCF communications) because it does not require coordination between TCP/IP stacks.

Dynamic VIPA (DVIPA)

Dynamic VIPAs (DVIPAs) can be defined on multiple stacks and moved from one TCP/IP stack in the sysplex to another automatically. One stack is defined as the primary or owning stack, and the others are defined as backup stacks. Only the primary stack is made known to the IP network.

TCP/IP stacks in a sysplex exchange information about DVIPAs and their existence and current location, and the stacks are continuously aware of whether the partner stacks are still functioning. If the owning stack leaves the XCF group (such as resulting from some sort of failure), then one of the backup stacks automatically takes its place and assumes ownership of the DVIPA. The network simply sees a change in the routing tables (or in the adapter that responds to ARP requests).

In this case, applications associated with these DVIPAs are active on the backup systems, thereby providing a hot standby and high availability for the services. DVIPA addresses identify applications independently of which images in the sysplex the server applications execute on and allow an application to retain its identity when moved between images in a sysplex.

When used with a Sysplex Distributor *routing* stack, DVIPAs (in such cases referred to as *distributed* DVIPAs) allow a cluster of hosts, running the same application service, to be perceived as a single large server node.

1.2.4 z/OS network connectivity and dynamic routing

Ever since the earliest support of TCP/IP in mainframe environments, if you wanted high availability, we recommended that you use a dynamic routing protocol in your mainframes. Recently, however, innovative designs have leveraged z/OS support of ARP takeover to achieve comparable availability without using a dynamic routing protocol. The basic concepts of each of these approaches is discussed in the following section. For more detailed information and implementation examples, refer to Chapter 3, “VIPA without dynamic routing” on page 45, and Chapter 4, “VIPA with dynamic routing” on page 65.

z/OS availability without dynamic routing

Why bear the cost and complexity of implementing a dynamic routing protocol in a mainframe environment, when not doing so in other server environments? Why not simply leave dynamic routing to the network people, and keep servers out of it? These are important questions.

But without dynamic routing, how can your mainframe environment take advantage of redundant network connectivity, and reroute around a network interface failure? The answer is, by leveraging z/OS ARP takeover support.

To understand how ARP takeover works, you need to understand something about how the z/OS Communications Server and the OSA-Express adapter (in QDIO mode) work together to support TCP/IP communication, as explained here:

- The OSA adapter depends on the z/OS Communications Server TCP/IP stacks to maintain IP-layer routing tables and handle IP routing issues.
- The z/OS TCP/IP stacks pass IP address location information to the OSAs, allowing them to maintain dynamically OSA address tables (OATs) containing information such as that shown in Table 1-1.

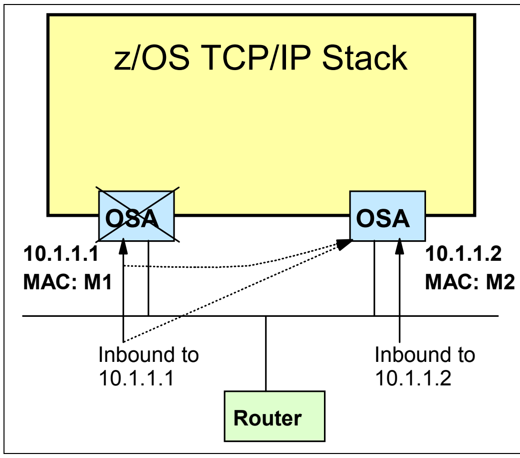
Table 1-1 Example of OSA address table information

IP-Dest=	Give to...
10.0.1.1	LPAR-1
10.0.1.4	LPAR-2
10.0.1.5	LPAR-2
10.0.1.6	LPAR-2

> **Note:** The OAT is maintained and updated automatically when using OSA-Express in QDIO mode.

- When an OSA adapter is shared by multiple LPARs, all IP packets to those LPARs are sent to the same local area network (Ethernet) MAC address and the OSA adapter looks into the IP header to determine to which LPAR an IP packet should be sent.
- Whenever a QDIO device is activated or the home list is modified (through OBEYFILE command processing or through dynamic changes, such as VIPA takeover), the TCP/IP stack updates the OAT configuration dynamically with the HOME list IP addresses of the stack.

Consider the example of ARP takeover as illustrated in Figure 1-1.



Router's initial ARP Cache

IP address	Mac address
10.1.1.1	M1
10.1.1.2	M2

Router's ARP Cache after movement of 10.1.1.1

IP address	Mac address
10.1.1.1	M2
10.1.1.2	M2

Figure 1-1 Example of ARP takeover

Figure 1-1 shows a z/OS TCP/IP stack with two OSA interfaces into the same Ethernet subnet. Initially, the stack assigns address 10.1.1.1 to one of the interfaces and 10.1.1.2 to the other.

However, when the 10.1.1.1 interface fails, the stack will *automatically* move that address over to the second interface. Further, when the second OSA interface is assigned address 10.1.1.1, it will broadcast its newly-added address to the subnetwork (using a protocol known as *gratuitous* ARP) so that every host on that subnet (the router, in this example) is informed of the change and can update its ARP cache appropriately. ARP takeover thereby allows TCP/IP traffic to be rerouted, in most cases nondisruptively, around a failed OSA interface.

The ARP takeover function shown in Figure 1-1 is an important availability tool in and of itself because it provides for high availability in the event of an interface failure. However, that is really just the beginning. Using the same underlying ARP takeover function in conjunction with Sysplex XCF communications, the z/OS Communications Server can also move DVIPAs from one TCP/IP stack to another (including on a completely different system), providing high availability even for application failures. An implementation example of ARP takeover is shown in 3.1.1, “ARP takeover” on page 47.

Using ARP takeover, you can achieve comparable high availability to what can be provided using dynamic routing, with the following *limitations*:

- Given a well-designed OSA connectivity environment, it is extremely unlikely that one of your systems could lose all OSA connectivity. However, should such a loss of connectivity occur, you cannot configure your other mainframe-to-mainframe connectivity (such as HiperSockets™, CTCs, or XCF communications) to be dynamically used for alternative connectivity out of the system without a dynamic routing protocol (though you can still use such back-end connectivity, with static routing and separate subnets, for host-to-host traffic).
- All of your systems must be attached to the same layer-2 subnetwork infrastructure (such as an Ethernet LAN), and you must be satisfied with the availability that can be achieved by you router or switch vendors' technology within a subnet:
  - Redundant switches, trunked together, and using their VLAN technology to create highly available subnetworks
  - Gateway router redundancy capabilities (such as VRRP or HSRP)

- *All* of your IP addresses (interfaces, static VIPAs, and DVIPAs) should be allocated out of the same subnet to which all of the sysplex hosts are attached in order to be reachable from the network (though you could use a separate, isolated, subnet for your HiperSockets connectivity for high-speed connectivity between LPARs).
- While it is possible to configure and use multiple parallel paths to increase available bandwidth (called *multipathing*) without a dynamic routing protocol, if there is a failure in the network beyond a point where the OSA adapter is able to detect it, TCP connection setups that are directed across the failed path will time out and UDP and RAW traffic will be lost.

z/OS availability using dynamic routing

With all of the function available without implementing a dynamic routing protocol in the z/OS Communications Server, why ever consider implementing one? One reason is because a sysplex (a cluster of System z mainframes coupled together with dynamic XCF connectivity), is a network in itself. So, when you connect a sysplex to a network, you are really connecting two networks together—a situation in which one would normally want to use a dynamic routing protocol. (Specifically, IBM recommends that you use OSPF and configure the sysplex as a *totally stubby area* to give the sysplex and the network full awareness of each other yet minimize the amount of routing information that has to be shared.)

If you do not use a dynamic routing protocol, you are essentially bridging together two dynamic TCP/IP networks with static, layer-2 based, network connectivity. To summarize the advantages of using a dynamic routing protocol, we need only take the limitations inherent in not using one and turn them around:

- If a system should lose all OSA connectivity, dynamic routing provides the ability to automatically use back-end connectivity (such as HiperSockets, CTCs, or XCF) as alternative pathing for your OSA-based network connectivity. And you can use OSPF link weights to ensure that traffic is routed across the optimal links (HiperSockets first, then OSAs, then XCF).
- You can design your layer-2 connectivity for higher availability, using isolated and redundant Ethernet switches (each its own subnet and not interconnected) and dynamic routing to recover from any subnetwork failure. You also do not need router redundancy technologies (such as Virtual Router Redundancy Protocol (VRRP) or Hot-Standby Router Protocol (HSRP)) because, through dynamic routing protocols, your sysplex will understand which routing paths are (or are not) available.
- You have much greater flexibility in terms of allocation of IP addresses. Indeed, your VIPA addresses should come from unique (and easily identifiable) subnets.
- Multipathing for bandwidth scalability is a built-in and fully supported capability of OSPF.

1.2.5 Single-instance and multiple-instance applications

The basic approach to availability is avoiding single points of failure. For server applications, that means executing multiple instances of the same application on different operating system images and systems. However, some clients might care about which server they reach, although others might not.

Single-instance applications

The relationship of a client to a particular server instance might require that only a single-instance of an application be available at a time. In other cases, it might be that the server application needs exclusive access to certain resources, which prevents the use of multiple-concurrent instances of an application. Such application environments, then, are necessarily single-instance applications—meaning that only one instance of an application is available at a time.

As mentioned in 1.2.2, “z/OS Parallel Sysplex availability” on page 3, the best that can be done in terms of availability for such applications is to leverage automation to either quickly restart them, or to redirect requests to another instance that is standing by. One example of a necessarily single-instance application is a TN3270 server that must support mapping to specific LU names (because the same LU names cannot be active on more than one VTAM® at a time).

Multiple-instance applications

For multiple-instance applications (also sometimes called “application clusters”), it really does not matter to which server instance a particular connection gets directed, as long as some server is there to field the connection request. These applications are characterized by doing all their work over a single connection, and either not saving state between connections (possibly including state information with the work request), or having the server applications share saved state among the instances. Examples of such applications include:

- Web servers (WebSphere), either for simple Web requests that do not create persistent client state, or where the Web servers share state in a database (such as can be done with WebSphere and DB2®)
- LDAP and MQ, which share their state among the application instances through their own means (shared files or structures in the Coupling Facility)

1.2.6 Balancing workload across multiple application instances

In general, application workload balancing refers to the ability to utilize different systems within the cluster simultaneously, thereby taking advantage of the additional computational function of each. In the sections that follow, we discuss the following different approaches for workload balancing:

- Internal application
- External application
- Hybrid

Internal application workload balancing

Using internal workload balancing in a z/OS environment, when a TCP connection request is received for a given *distributed* DVIPA address, the decision as to which instance of the application will serve that particular request is made by the Sysplex Distributor running in the TCP/IP stack that is configured to be the *routing* stack.

The Sysplex Distributor has realtime capacity information available (from the sysplex workload manager) and can leverage QoS information from the Service Policy Agent. Consequently, internal workload balancing requires no special external hardware. However, all in-bound messages to the distributed DVIPA must first transit the routing stack before being forwarded along to the appropriate application instance.

> **Note:** With the VIPARoute function, you can use any available interface for DVIPA traffic forwarding. Using alternative interfaces (such as HiperSockets or OSA-Express connectivity) can improve performance while reducing the utilization of XCF interfaces.
>
> Sysplex Distributor only supports the balancing of TCP (not UDP) traffic.

Internal application workload balancing is discussed in detail (including implementation examples) in Chapter 5, “Internal application workload balancing” on page 95.

External application workload balancing

External workload balancing uses switching hardware outside of the mainframe environment to distribute connection requests across available z/OS servers. Such hardware usually supports both UDP and TCP traffic, and can be shared with other (non-mainframe) servers, thus making external workload balancing a potentially more cost-effective alternative.

An external load balancer represents multi-instance servers as one application to the outside world. Application users know the application cluster only as an IP address within the external load balancer. How many application instances are available, the relationship between the IP address of the application cluster and the application instances, and port assignments are configured and managed in the external load balancer. This is illustrated in Figure 1-2.

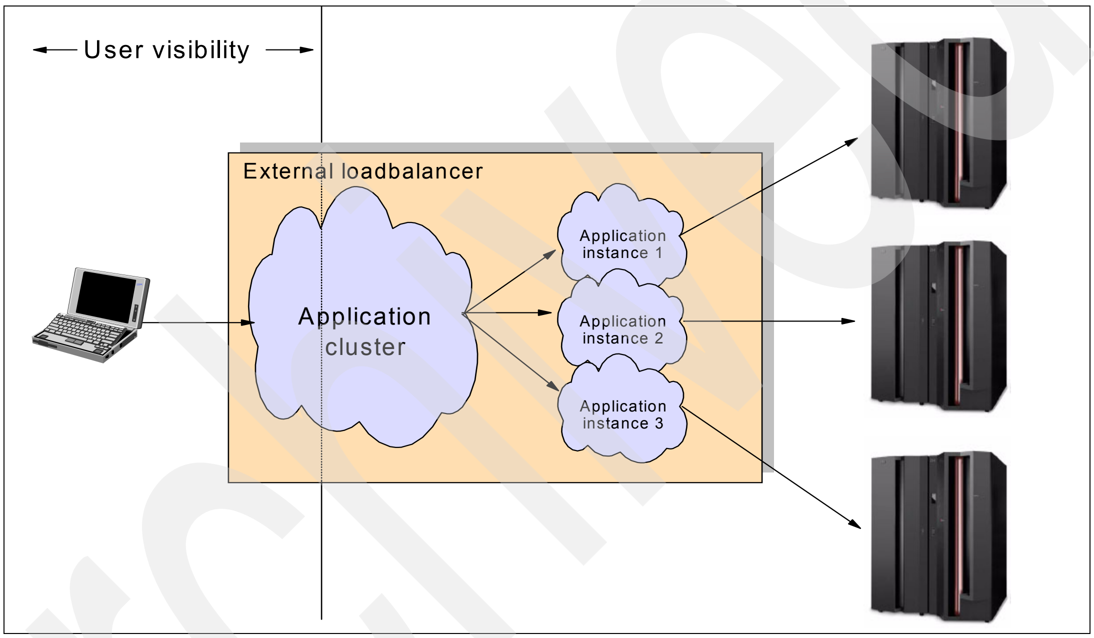


Figure 1-2 Relationship between application cluster and application instances

The external load balancer receives datagrams destined for the IP address representing the application cluster and then forwards the datagrams to application instances belonging to the application cluster.

One challenge for external load balancers involves a lack of awareness about the status of the target application instances. The environment includes three levels that affect the usability of the application instances:

- The networking infrastructure
- The operating environment
- The application instance and any multi-instance locking that might exist

External load balancers have developed various techniques to sense the environmental status. The most common techniques include polling of the application, pinging the IP address used by the application, measurement of turnaround times for datagrams, execution of a predefined dummy transaction, or analysis of transaction behavior. These techniques provide only part of the information needed to make the best decision for server assignment.

A disadvantage of these techniques is that there can be some delay between the time an environmental problem occurs and when the load balancer detects it. In a highly loaded application environment, this can result in many connections being forwarded to an unhealthy application instance.

The reverse problem arises when the application instance becomes fully operational again. Hybrid internal/external workload balancing approaches (discussed in "Hybrid internal or external approaches" on page 10) can give external load balancers (such as CSS) real-time awareness of the status of the sysplex environment and applications.

The strength of external load balancers is that they are dedicated devices performing a fixed set of tasks. They typically have significant processing capacity, giving them the ability to find and make decisions using higher-layer information buried deep inside datagrams rather than just using IP addresses, including application-specific information such as HTTP (or FTP) Uniform Resource Locators (URLs). External load balancers can also be implemented in redundant modes for availability, and more devices can usually be added in parallel to increase capacity. External load balancers are especially attractive (as compared to internal workload balancing) when the workload includes large amounts of inbound data traffic, because of their more-efficient inbound data path.

External application workload balancing is discussed in detail (including implementation examples) in Chapter 6, "External application workload balancing" on page 171.

Hybrid internal or external approaches

In many cases, your application load balancing requirements might be such that purely internal or external workload balancing will be satisfactory. However, in some situations you might need elements of both approaches.

For example, you might need the UDP workload balancing capability of an external load balancer but want to make z/OS application load balancing decisions based upon real-time awareness of current application workloads and performance. Two technology solutions address this requirement:

- Sysplex Distributor's Multi-Node Load Balancing (MNLB) Service Manager support
- z/OS Load Balancing Advisor (LBA) that uses the Server Application State Protocol (SASP)

In the MNLB architecture, the entity that decides which target server is to receive a given connections is called the Service Manager. The entity that forwards data to target servers is called the *Forwarding Agent*. In a hybrid environment, the Sysplex Distributor acts as the Service Manager and switches perform the Forwarding Agent role.

Note that the Sysplex Distributor makes the actual workload distribution decision. Then, acting as a Service Manager, Sysplex Distributor relays that decision to the Forwarding Agents, which thereafter directly send traffic for that connection to the target server, bypassing the distribution stack (thereby saving the mainframe CPU cycles associated with routing messages through the distribution stack).

> **Note:** Because Sysplex Distributor makes the actual workload distribution decision, we chose to include Sysplex Distributor's MNLB Service Manager support (including implementation examples) in Chapter 5, "Internal application workload balancing" on page 95.

The z/OS Load Balancing Advisor is a z/OS Communications Server component that allows any external load balancing solution to become *sysplex-aware*. The external load balancer must support the Server Application State Protocol (SASP) to obtain sysplex information through this function. Sysplex awareness information is collected by the Load Balancing Advisor, a weight metric is calculated (based upon Workload Manager (WLM) and z/OS Communications Server information), and then the weight is sent to the external load balancer. Ultimately, however, the external load balancer decides which application instance is best for the next request, which makes the LBA/SASP approach slightly different from the MNLB Service Manager approach.

Note: Because the external load balancer makes the actual workload distribution decision, we chose to include the detailed discussion of the LBA/SASP approach (including implementation examples) in Chapter 6, “External application workload balancing” on page 171.

Hybrid solutions, though more complex, combine the information advantages of internal workload balancing with the routing efficiency and functionality of external load balancing.

1.3 Quick start table

The information in Table 1-2 can help you quickly find your way to the parts of this book that are of interest to you. Start by finding the row in the table that most closely corresponds to your environment: single z/OS system or z/OS Parallel Sysplex, and with or without the use of dynamic routing in your z/OS environment. Then find the column with availability, scalability, or performance functionality that interests you. In the table cells, we have identified technologies that might be helpful for you and where we discuss these technologies in this book.

Table 1-2 Quick start for high availability, scalability, and performance

	High availability		**Scalability**	**Performance**
	Network connectivity	Application	Workload balancing	
Single z/OS system without dynamic routing	Static VIPA and ARP takeover (Chapter 2, “Virtual IP addressing” on page 13, and Chapter 3, “VIPA without dynamic routing” on page 45)		► SHAREPORT and SHAREPORTWLM (Chapter 5, “Internal application workload balancing” on page 95) ► External WLB (Chapter 6, “External application workload balancing” on page 171) ► LBA/SASP (Chapter 6, “External application workload balancing” on page 171)	► General ► TCP/IP ► z/OS UNIX® ► Storage ► Applications (Chapter 8, “Performance and tuning” on page 261)
Single z/OS system with dynamic routing	Static VIPA and OSPF (Chapter 2, “Virtual IP addressing” on page 13, and Chapter 4, “VIPA with dynamic routing” on page 65)			
z/OS Parallel Sysplex without dynamic routing	Static VIPA and ARP takeover (Chapter 2, “Virtual IP addressing” on page 13, and Chapter 3, “VIPA without dynamic routing” on page 45)	DVIPA and ARP takeover (Chapter 2, “Virtual IP addressing” on page 13, and Chapter 3, “VIPA without dynamic routing” on page 45	► Distributed DVIPA and Sysplex Distributor (Chapter 2, “Virtual IP addressing” on page 13 and Chapter 6, “External application workload balancing” on page 171) ► External WLB (Chapter 6, “External application workload balancing” on page 171) ► LBA/SASP (Chapter 6, “External application workload balancing” on page 171) ► Sysplex Distributor (Chapter 5, “Internal application workload balancing” on page 95 and Chapter 7, “Intra-sysplex workload balancing” on page 213)	
z/OS Parallel Sysplex with dynamic routing	Static VIPA and OSPF (Chapter 2, “Virtual IP addressing” on page 13, and Chapter 4, “VIPA with dynamic routing” on page 65)	DVIPA and OSPF (Chapter 2, “Virtual IP addressing” on page 13, and Chapter 4, “VIPA with dynamic routing” on page 65)		
Non-z/OS DataPower® box		DVIPA and OSPF (Chapter 2, “Virtual IP addressing” on page 13, and Chapter 4, “VIPA with dynamic routing” on page 65)	► Sysplex Distributor (Chapter 7, “Intra-sysplex workload balancing” on page 213)	

2

Virtual IP addressing

In TCP/IP networking, Internet Protocol (IP) addresses are typically assigned to physical network interfaces. If a server has two physical interfaces, a separate IP address is assigned to each of them.

IBM introduced the concept of Virtual IP Addressing (VIPA) for its z/OS environment to support the use of IP addresses, representing TCP/IP stacks, applications, or clusters of applications, that are not tied to any specific physical interface. The association between a VIPA and an actual physical interface is subsequently accomplished using either the Address Resolution Protocol (ARP) or dynamic routing protocols (such as OSPF).

This chapter discusses the following topics.

Section	Topic
2.1, "Basic concepts of Virtual IP Addressing" on page 14	Basic concepts, commands, and functions associated with a VIPA
2.2, "Importance of VIPA" on page 26	Why VIPA is important
2.3, "Timing of DVIPA activation" on page 27	Controlling the timing of when a stack joins the TCP/IP sysplex group
2.4, "Dynamic VIPA example" on page 34	Implementation of a basic VIPA environment

2.1 Basic concepts of Virtual IP Addressing

The z/OS Communications Server supports two types of Virtual IP Addressing (VIPA):

- Static
- Dynamic

A *static* VIPA is an IP address that is associated with a particular TCP/IP stack. Using either ARP takeover or a dynamic routing protocol (such as OSPF), static VIPAs can enable mainframe application communications to continue unaffected by network interface failures. As long as a single network interface is operational on a host, communication with applications on the host will persist. Example 4-7 on page 77 shows an example configuration for a static VIPA.

Static VIPAs have the following characteristics:

- They can be activated during TCP/IP initialization or VARY TCPIP, OBEYFILE command processing, and are configured using an appropriate set of DEVICE, LINK, HOME, and, optionally, OMPROUTE configuration statements or BSDROUTINGPARMS statements for IPv4 Static VIPAs (or INTERFACE statements for IPv6 Static VIPAs).
- Using the SOURCEVIPA configuration option, or SOURCEVIPA parameter on the INTERFACE configuration statement, static VIPAs can be used as the source IP address for outbound datagrams for TCP, RAW, UDP (except routing protocols), and ICMP requests.
- They can be specified as the source IP address for outbound TCP connection requests for *all* applications using this stack with TCPSTACKSOURCEVIPA, or just a specific *job* and a specific *destination* through the use of the SRCIP profile statement block.
- The number of static VIPAs on a stack is limited only by the range of host IP addresses that are available for that host.
- They can be moved to a backup stack after the original owning stack has left the XCF group (such as resulting from some sort of failure), by using VARY TCPIP,,OBEYFILE command processing to configure the VIPA on the backup stack.

> **Note:** Static VIPA does not require sysplex (XCF communications) because it does not require coordination between TCP/IP stacks.

Dynamic VIPAs (DVIPAs) can be defined on multiple stacks and moved from one TCP/IP stack in the sysplex to another automatically. One stack is defined as the primary or owning stack, and the others are defined as backup stacks. Only the primary stack is made known to the IP network.

TCP/IP stacks in a sysplex exchange information about DVIPAs, their existence, and their current location, and the stacks are continuously aware of whether the partner stacks are still functioning. If the owning stack leaves the XCF group (such as resulting from some sort of failure), then one of the backup stacks automatically takes its place and assumes ownership of the DVIPA. The network simply sees a change in the routing tables (or in the adapter that responds to ARP requests).

In this case, applications associated with these DVIPAs are active on the backup systems, thereby providing a hot standby and high availability for the services. DVIPA addresses identify applications independently of which images in the sysplex the server applications execute on and allow an application to retain its identity when moved between images in a sysplex.

DVIPAs can be moved among any TCP/IP stack in the sysplex (that is configured to allow it) through any of the following mechanisms:

- Automatically, for example, in response to the owning stack leaving the XCF group (such as resulting from some sort of failure)
- Through an application program (using an IOCTL macro), when one instance of an application takes over primary function from another
- By using a utility program (MODDVIPA) from the operator console

Figure 2-1 shows a simple example of DVIPA that illustrates a case where a TCP/IP stack (and associated applications) has failed.

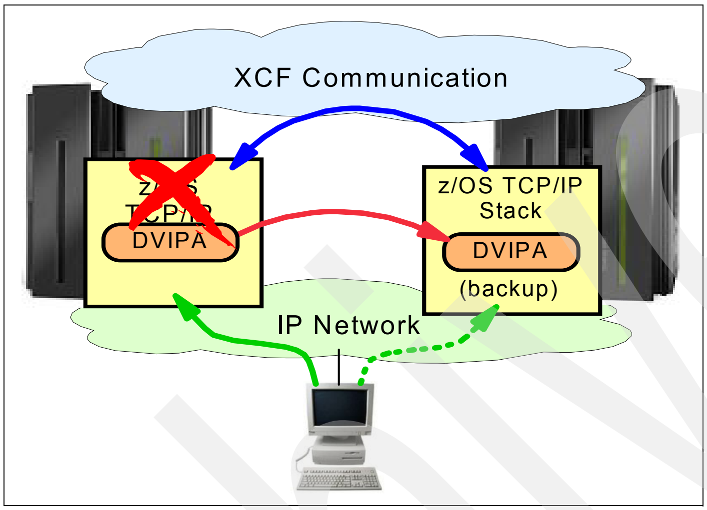


Figure 2-1 DVIPA address movement

DVIPA automatically moves operation of the failed IP address to another z/OS image. With proper planning, the same application is available on the second z/OS image and it immediately picks up the workload that is directed to the IP address. External users see only workload directed to an IP address; the shift of this IP address from the first z/OS to the second z/OS is largely transparent.

Dynamic VIPAs are defined using the VIPADynamic statement (as shown in Figure 2-2 on page 16) and have the following characteristics:

- They can be configured to be moved dynamically from a failing stack to a backup stack within the same sysplex without operator intervention or external automation. This is a *stack-managed* configuration and is defined using the VIPADefine and VIPABackup parameters.
- They can be activated dynamically by an application program (using IOCTL) or by a supplied utility (MODDVIPA). This is *event activation* or *command activation* and is managed using the VIPARange parameter.
- They can be specified on a TCPSTACKSOURCEVIPA statement. This allows a user to specify one Dynamic VIPA to be used as the source IP address for outbound datagrams for TCP-only requests.

When used with Sysplex Distributor, DVIPAs (in such cases referred to as *distributed* DVIPAs) allow a cluster of hosts, running the same application service, to be perceived as a single large server node. Distributed DVIPAs are defined using the VIPADistribute parameter with the VIPADynamic statement (as shown in Figure 2-2) and have the following characteristics:

- They have all the characteristics of DVIPAs, but cannot be dynamically activated by an application program.
- One stack (called the *routing stack*) defines a DVIPA and advertises its existence to the network. Stacks in the target distribution list activate the DVIPA and accept connection requests.
- Connection workload can be spread across several stacks.

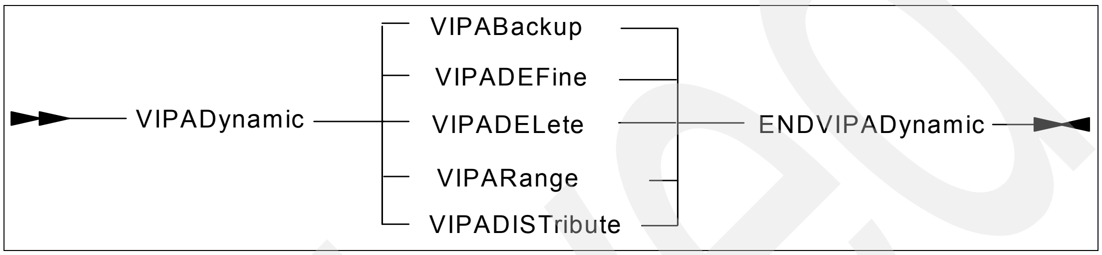


Figure 2-2 DVIPA configuration

Refer to "Distributed DVIPA (Sysplex Distributor)" on page 20 for a more detailed description of this function.

2.1.1 Application availability modes

Consider the following *availability modes* for an application (arranged in order from least available to most highly available):

- Single-instance

  Running just one instance of an application
- Single-instance with automated restart

  Running just one instance of an application, but leveraging automation utilities such as Automatic Restart Manager (ARM) in the case of an application failure to either immediately restart the application in the same place or start it on another stack
- Single-instance with hot standby

  Running just one instance of an application, but having standby instances of the same application running and ready to take over in the event of a failure (or termination) of the first instance (this is still essentially single-instance because there is only one copy of the application that is available at any given time)
- Multiple-instance

  By actually running multiple, concurrently available instances of the same application (and balancing workload across them)

Multiple application instances will provide the highest availability (as well as higher scalability), but not all applications can support multiple concurrent instances; for example:

- An FTP server might require exclusive access to certain data sets.
- A TN3270 server might require use of specific LU names.
- Software license limitations.

- Application implementation specific requirements that limit the application to a single instance, such as lack of locking or data sharing in the basic program design, need for exclusive access to certain resources, architecturally-related dependencies, or other issues
- Possible client dependencies upon a particular server instance

For the purposes of this chapter we assume that such single-instance characteristics are fixed and cannot be changed. Depending upon your applications, therefore, you might need to implement several of these application availability modes.

2.1.2 DVIPA and distributed DVIPA

Ultimately, the availability modes that you need to support (based upon your specific set of applications) dictate how you set up and use DVIPAs and distributed DVIPAs. Table 2-1 summarizes the choices for providing availability that is higher than just running a single instance of an application.

Table 2-1 Matching DVIPA support with application requirements

Availability mode	Type of DVIPA to use	How to define it
Single-instance with automated restart (same stack or different stack)	Event-activated DVIPA (driven by application bind, IOCTL, or MODDVIPA command)	VIPARANGE
Single-instance with hot standby	Stack-managed DVIPA	VIPADEFINE (primary owner), VIPABACKUP (backup stacks)
Multiple-instance	Distributed DVIPA	VIPADEFINE (primary owner) VIPABACKUP (backup stacks) VIPADISTRIBUTE

DVIPAs can be moved between TCP/IP stacks by any of the following methods:

- Automatic movement of a VIPA from a failing TCP/IP stack to a backup stack, known as *automatic takeover* and *takeback*
- Dynamic allocation of a VIPA by an application program application programming interface (API) call, using any of these methods:
  - The application issues a `bind()` to that particular (specific) IP address.
  - The application binds to INADDR_ANY instead of a specific IP address, but the server bind control function changes the generic `bind()` to a specific one.
  - An authorized application issues a SIOCSVIPA IOCTL command. An example of such an application is the MODDVIPA utility.
- Manual movement of a VIPA by deactivating or reactivating it with the VARY TCPIP,,SYSPLEX command

Event-activated DVIPAs

Event-activated DVIPA functions are indicated by the VIPARANGE parameter in the TCP/IP profile. Activation of an application-specific DVIPA IP address associated with a specific application instance occurs only using an application program's API call, in any of the following ways:

- When the application instance issues a `bind()` function call and specifies an IP address that is not active on the stack. The stack will activate the address as a DVIPA, if it meets certain criteria. When the application instance closes the socket, the DVIPA is deleted.
- Some applications cannot be configured to issue `bind()` for a specific IP address, but are unique application-instances. For such applications, a utility is provided (MODDVIPA), which issues SIOCSVIPA or SIOCSVIPA6 `ioctl()` to activate or deactivate the DVIPA.

  This utility can be included in a JCL procedure or OMVS script to activate the DVIPA before initiating the application. As long as the same JCL package or script is used to restart the application instance on another LPAR in the event of a failure, the same DVIPA will be activated on the new stack.
- An alternative for unique application-instance applications that do not contain the logic to bind to a unique IP address is to use the BIND parameter on the PORT reservation statement. It is usually a good practice to reserve a port for the listening socket of a server application.

  If the BIND parameter *and* an IP address are specified on the PORT reservation statement for a server application, and the application binds a socket to that port and specifies INADDR_ANY, then z/OS TCP/IP converts the bind to be specific to the IP address specified on the PORT reservation statement.

  From that point on, it appears as though the application itself had issued the `bind()` specifying the designated IP address, including having the IP address deleted when the server application closes the socket.

Because the VIPA is specified by the application, it does not need to be uniquely defined in the TCP/IP profile. However, we must ensure that the addresses being used by the application correspond to our IP addressing scheme. We use the VIPARange statement in the TCP/IP profile to indicate the range of VIPAs that we are willing to activate dynamically, as shown in Figure 2-3.

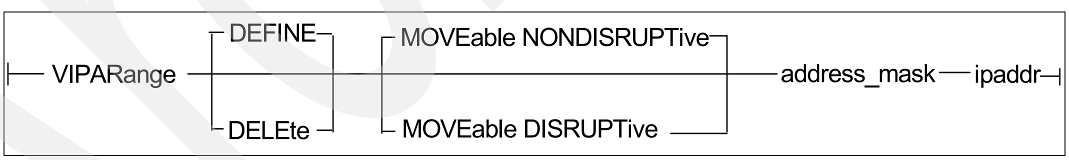


Figure 2-3 Definition format for VIPARange

The VIPARange statement defines an IP subnetwork using a network address (prefix) and a subnet mask. Because the same VIPA address cannot be activated by IOCTL or `bind()` while also participating in automatic takeover as defined by VIPADEFine/VIPABackup, we recommend that subnets for VIPADEFine be different from subnets for VIPARange.

VIPARange in itself does not create, activate, or reserve any dynamic VIPAs. It merely defines an allocated range within which program action (or the BIND parameter or a PORT statement) can cause a dynamic VIPA to be created and activated for a specific IP address. The same range can have DVIPAs defined through VIPADEFINE or VIPABACKUP, provided that no attempt is made to use the same IP address for both VIPADEFINE and activation uses the BIND program.

For our operation, we used the following definition:

```
VIPADYNAMIC
   VIPARANGE  DEFINE MOVEABLE NONDISRUPT 255.255.255.0 10.30.30.1
ENDVIPADYNAMIC
```

After it is activated on a stack using `bind()` or IOCTL, a Dynamic VIPA IP address remains active until the VIPA IP address is moved to another stack or is deleted. The system operator might delete an active application-specific Dynamic VIPA IP address by using the MODDVIPA utility or by stopping the application that issued the `bind()` to activate the VIPA. To remove an active VIPARange statement, the VIPARange DELETE statement can be used.

Deleting a VIPARANGE does not affect existing DVIPAs that are already activated within the range, but it does prevent new ones from being activated within the deleted range until the VIPARANGE is reinstated.

Stack-managed DVIPAs

Stack-managed DVIPAs are defined by VIPADefine and VIPABackup, and are moved between stacks. The switch can be automatic or manual, as explained here.

- Automatic

  If the owning stack leaves the XCF group (for example, resulting from some sort of failure), the stack with an already-configured VIPABACKUP definition for that DVIPA activates and advertises the DVIPA, and this is the TAKEOVER operation. Multiple stacks can back up a DVIPA. The stack with the highest configured rank will take over.

- Manual (planned)

  This method uses **`VARY TCPIP,,SYSPLEX deact/act`** operator commands.

  A deactivate command:

  – If the DVIPA is active, deactivate deletes the DVIPA allowing a backup stack to take over the DVIPA.

  – If the DVIPA is a backup, deactivate removes this stack as a backup, preventing this stack from taking over if the current owner leaves the XCF group (for example, resulting from some sort of failure).

  A reactivate command:

  – Causes takeback for a VIPADEFINE DVIPA.

  – Re-enables VIPABACKUP DVIPA to perform takeover if a stack leaves the XCF group.

Applications that bind() to INADDR_ANY

An application can issue a `bind()` to INADDR_ANY to accept connection requests from any IP address associated with the stack. This is insufficient if the application must be associated with a specific VIPA address. There are three ways to manage this situation:

- Cause the application to `bind()` to a specific address (instead of INADDR_ANY) by using the Server Bind Control function that is implemented by the BIND keyword on a PORT statement.
- Modify the application to `bind()` to a specific address.
- Use the utility MODDVIPA or modify the application to send the appropriate IOCTL.

The most attractive solution is to use the Server Bind Control function because it does not require changing the application. Using this function, a generic server (such as the TN3270 server) can be directed to bind to a specific address instead of INADDR_ANY. When the application attempts binds to INADDR_ANY, the `bind()` is intercepted and converted to the specified IP address. The process then continues as though the server had issued a `bind()` to that specific address. In order to use this function, however, the port used by the application must be known in advance so that it can be added to the PORT statement in the TCP/IP profile.

In situations where the application cannot take advantage of the server bind control function and cannot be otherwise modified, the MODDVIPA utility can be used to create a Dynamic VIPA IP address using the IOCTL call. The utility can be initiated through JCL, from the OMVS command line, or from a shell script. This utility is in EZB.MODDVIPA.sysname.tcpname. For more detailed information about the MODDVIPA utility, refer to *z/OS CS: IP Configuration Guide,* SC31-8775.

In most of our examples, we active the DVIPA address by the BIND keyword on the PORT statement.

Using the MODDVIPA utility

We used the following procedure to run the MODDVIPA utility:

```
//TCPDVP   PROC
//TCPDVP   EXEC PGM=MODDVIPA,REGION=OK,TIME=1440,
//   PARM='-p TCPIPC -c 10.30.30.1' 1
//SYSPRINT DD SYSOUT=A,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSERR   DD SYSOUT=A
//SYSERROR DD SYSOUT=A
//SYSDEBUG DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSABEND DD SYSOUT=A
```

The utility expects a parameter specifying the VIPA address to be activated. It can also be used to delete a VIPA address as an alternative to using a VARY OBEY operator command. The parameter option field can be **-c** for create or **-d** for delete. This example creates a Dynamic VIPA **1** with IP address 10.30.30.1.

Activation of the Dynamic VIPA IP address will succeed as long as the desired IP address is not claimed by any other stack, is not an IP address of a physical interface or a static VIPA, and is not defined using VIPADEFine or VIPABackup in a VIPADynamic block.

MODDVIPA must run with APF authorization. For additional information about APF authorization, refer to *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7801.

Distributed DVIPA (Sysplex Distributor)

Sysplex Distributor is a function that allows an IP workload to be distributed to multiple server instances within the sysplex without requiring changes to clients or networking hardware and without delays in connection setup. It allows you to implement a dynamic VIPA as a single network-visible IP address that is used for a set of hosts belonging to the same sysplex cluster. A client on the IP network sees the sysplex cluster as one IP address, regardless of the number of hosts in the cluster.

With Sysplex Distributor, clients receive the benefits of workload distribution provided by both the Workload Manager (WLM) and the Quality of Service (QoS) Policy Agent. In addition, Sysplex Distributor ensures high availability of the IP applications running on the sysplex cluster by providing continued operation if a LAN fails, or an entire IP stack leaves the XCF group=, or a z/OS image is lost.

Important: Note that Sysplex Distributor only works for TCP applications, not UDP applications.

Sysplex Distributor operation

Consider the environment shown in Figure 2-4. It includes four TCP/IP stacks (in four z/OS images) running in the same sysplex cluster in WLM GOAL. All stacks have SYSPLEXROUTING, DATAGRAMFWD, and DYNAMICXCF configured. Assume that:

- H1 is configured as the distributing IP stack with V1 as the Dynamic VIPA (DVIPA) assigned to the sysplex cluster.
- H2 is configured as backup for V1.
- H3 and H4 are configured as secondary backups for V1.
- APPL1 is running in all the hosts that are members of the same sysplex cluster.

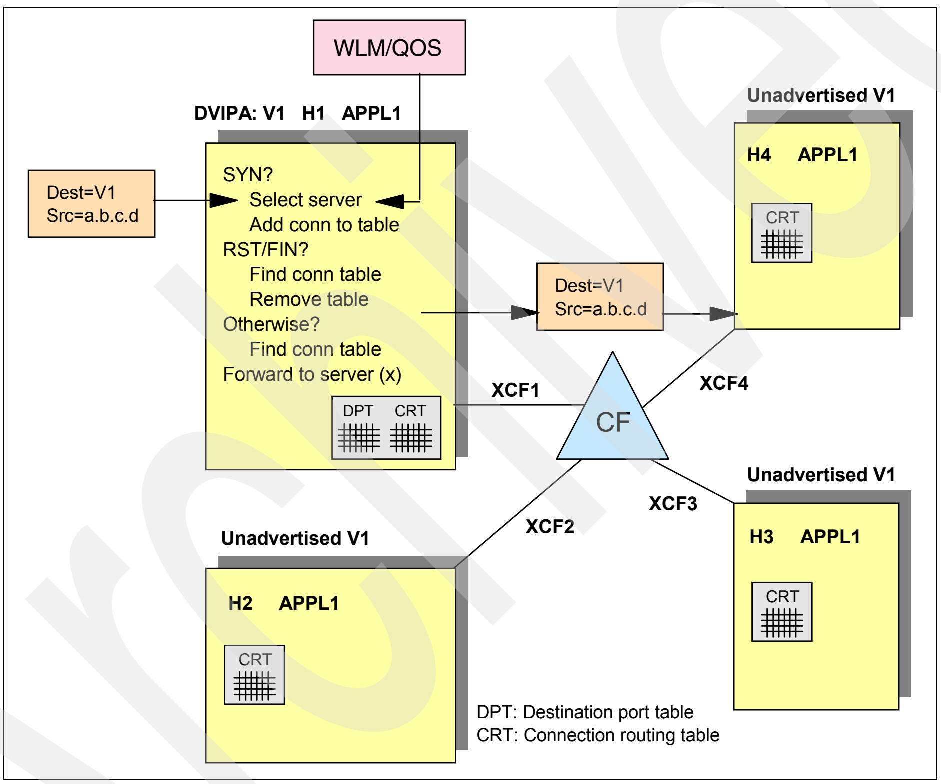


Figure 2-4 Sysplex Distributor functionality

Sysplex Distributor works as follows:

1. When IP stack H1 is activated, the definitions for the local XCF1 link are created dynamically due to DYNAMICXCF being coded in the H1 profile. Through this link, H1 recognizes the other IP stacks that belong to the same sysplex cluster and their XCF associated links: XCF2, XCF3, and XCF4.
2. The DVIPA assigned to the sysplex cluster and the application ports that this DVIPA serves are read from the VIPADISTRIBUTE statement in the profile data set. An entry in the home list for all stacks is added for the distributed IP address. The home list entry on the target stacks is done with a message that H1 sends to all the stacks read from the VIPADISTRIBUTE statement. Only one stack (H1, in this example) advertises the DVIPA through the RIP or OSPF routing protocol.

3. H1 monitors whether there is at least one application (APPL1 in Figure 2-4) with a listening socket for the designated port and DVIPA. H2, H3, and H4 send a message to H1 when a server (APPL1) is bound to either INADDR_ANY or specifically to the DVIPA (and, of course, the designated port). With that information, H1 builds a table with the name of the application and the IP stacks that could serve a connection request for it. The table matches the application server listening port with the target XCF IP address.
4. When a network client requests a service from APPL1, DNS resolves the IP address for the application with the DVIPA address. (This DNS could be any DNS in the IP network and does not need to register with WLM.)
5. When H1 receives the connection request (a TCP segment with the SYN flag), it queries WLM or QoS (or both) to select the best target stack for APPL1 and forwards the SYN segment to the chosen target stack. In our example, it is APPL1 in H4 that best fits the request.
6. An entry is created in the connection routing table (CRT) in H1 for this new connection, with XCF4 as the target IP address. H4 also adds the connection to its connection routing table.

   If a program binds to DVIPA on H4 and initiates a connection, H4 needs to send a message to H1 so that H1 can update its connection routing table accordingly. As an example, this is used when the FTP server on H4 would initiate a data connection (port 20) to a client.
7. The H1 IP stack forwards subsequent incoming data for this connection to the correct target stack.
8. When the H4 IP stack decides that the connection no longer exists, it informs the H1 IP stack with a message so H1 can remove the connection from its connection routing table.

Backup capability

The VIPA takeover functionality supports Sysplex Distributor. The following example illustrates its usage, as depicted in Figure 2-5 on page 23.

Consider the situation where our example has been running for some time without problems. APPL1 connections have been distributed (according to WLM directions) to H1, H2, H3, and H4. Many connections are currently established between several APPL1 server images and clients in the IP network.

Assume we now have a major failure in our distributing IP stack (H1). Automatic dynamic VIPA takeover occurs. This function allows a VIPA address to automatically move from one IP stack where it was defined to another one in the event of the failure of the first. The VIPA address remains active in the IP network, allowing clients to access the services associated with it.

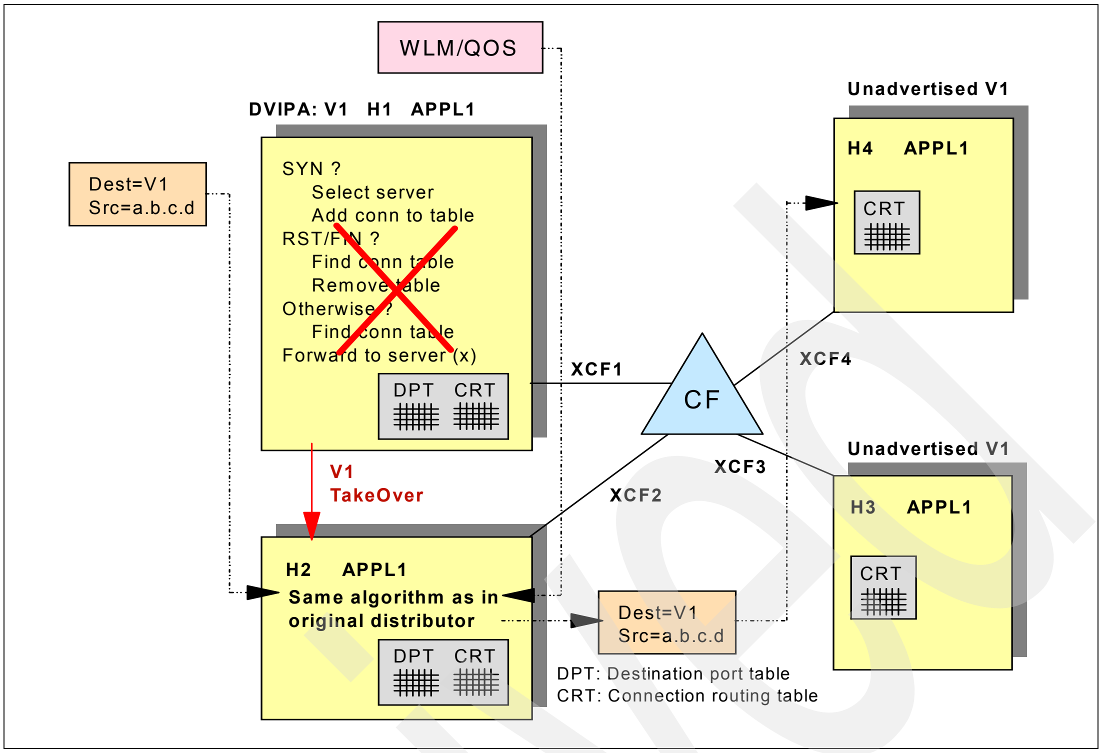


Figure 2-5 DVIPA backup capability

In our example, H1 is the distributing IP stack and H2 is the primary VIPABACKUP IP stack. When H1 fails, the following occurs:

1. All the IP connections terminating at H1 are lost. The Sysplex Distributor connection routing table (CRT) is also lost.
2. H2 detects that H1 is down (because it has left the XCF group) and defines itself as the distributing IP stack for the VIPA.
3. Because H2 saved information about H1, it informs the other target stacks that it knows V1 is distributable.
4. H3 and H4 are informed that H2 is the chosen backup for V1 and they immediately send connection information regarding V1 to IP stack H2.
5. H2 advertises V1 (DVIPA) through the dynamic routing protocol (RIP or OSPF). Retransmitted TCP segments for already existing connections or SYN segments for new connections are hereafter processed by IP stack H2 and routed by H2 to the appropriate target stacks.

Notice that only the IP connections with the failing IP stack are lost. All other connections remain allocated and function properly.

Recovery

After the H1 IP stack is activated again, the process of taking back V1 to H1 is started automatically. This process is nondisruptive for the IP connections already established with V1, regardless of which host is the owner at that time (in our example, H2). In our example, connection information is maintained by H2. When H1 is reactivated, H2 sends its connection information to H1. This gives H1 the information it needs to distribute packets again for existing connections to the correct stacks in the sysplex.

Connections with the backup host are not broken when the V1 address is taken back to H1, and takeback is not delayed until all connections with the backup host have terminated. Figure 2-6 illustrates this situation.

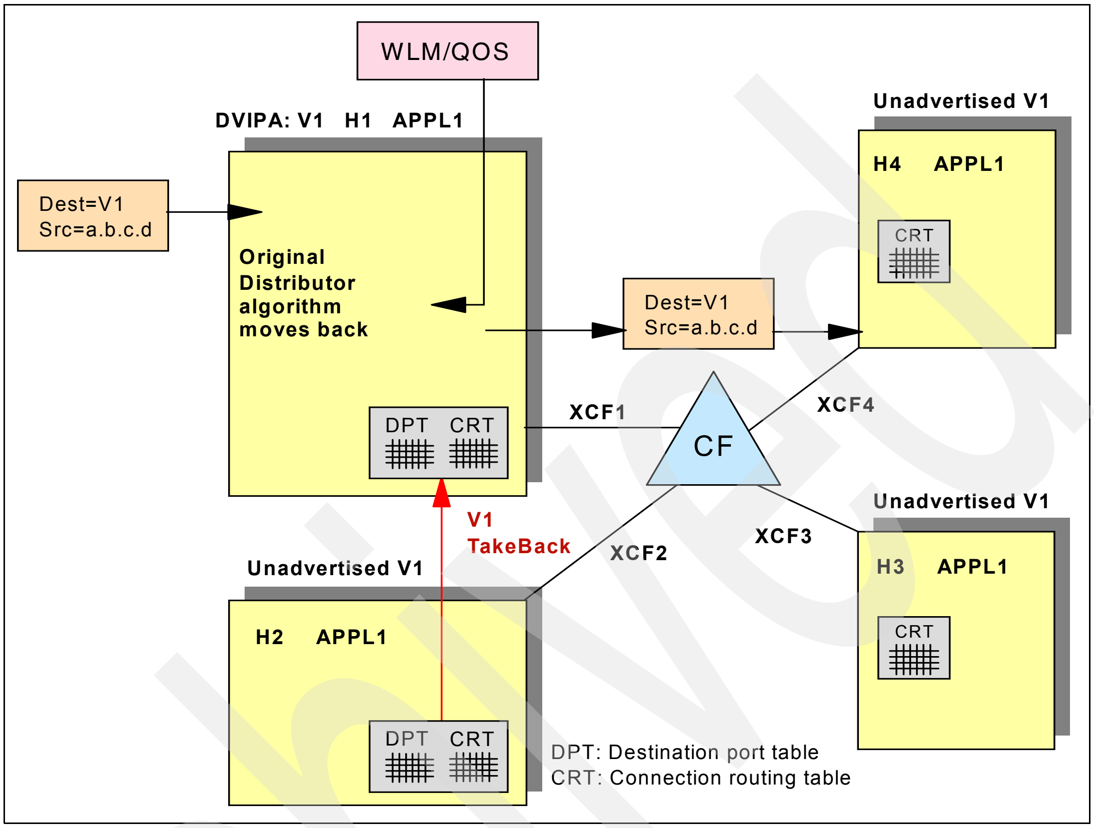


Figure 2-6 Sysplex Distributor and VIPA takeback

Dynamic routing with Sysplex Distributor

Routing IP packets for Sysplex Distributor can be divided into two cases:

- Routing inside the sysplex cluster
- Routing through the IP network

Routing *inside* the sysplex cluster is accomplished by the distributing host. All incoming traffic (new connection requests and connection data) arrives at the distributing stack. This stack forwards the traffic to the target applications in the sysplex cluster, using XCF links or using the VIPARoute function. (See 2.1.3, "VIPARoute function" on page 25 for more information.) OSA-Express connections can be used, as well.

Routing *outside* the sysplex is done by the downstream routers. Those routers learn about the DVIPA assigned to the sysplex dynamically using OSPF or RIP routing protocols. It is usually necessary to implement either one of these routing protocols in all the IP stacks of the sysplex cluster.

The distributing VIPA address is dynamically added to the home list of each IP stack participating in the Sysplex Distributor, but only one IP stack advertises the sysplex VIPA address to the routers: the one defined as the distributing IP stack. The other stacks do not advertise it and only the backup IP stack will do so if the distributing IP stack leaves the XCF group (for example, resulting from some sort of failure).

When using OMPROUTE, consider the following:

- Names of Dynamic VIPA interfaces are assigned dynamically by the stack when they are created. Therefore, the name coded for the OSPF_Interface statement in the Name field is ignored by OMPROUTE.
- We recommend that each OMPROUTE server have an OSPF_Interface defined for each Dynamic VIPA address that the IP stack might own or, if the number of DVIPAs addresses is large, a wildcard should be used.

It is also possible to define ranges of dynamic VIPA interfaces using the subnet mask and the IP address on the OSPF_Interface statement. The range defined includes all the IP addresses that fall within the subnet defined by the mask and the IP address. The following example defines a range of Dynamic VIPA addresses from 10.30.10.1 to 10.30.10.254:

```
OSPF_Interface
   IP_address = 10.30.10.1
   Name = dummy_name
   Subnet_mask = 255.255.255.0
```

For consistency with the VIPARANGE statement in the TCPIP.PROFILE, any value that can fall within this range can be used to define a range of dynamic VIPAs.

2.1.3 VIPARoute function

The VIPARoute function was introduced with the z/OS V1R7 Communications Server. Previously, all DVIPA IP traffic was forwarded to target stacks only by using Dynamic XCF interfaces. The VIPARoute function can use any available interface for this traffic forwarding. Using alternative interfaces can improve performance while reducing the utilization of Sysplex XCF interfaces.

Figure 2-7 illustrates this function where a mixture of XCF, an external Ethernet LAN, and HiperSockets could be used to forward traffic. WLM and QoS weights are factors considered in target selection for new requests. However, *outgoing* traffic generated by the applications is routed according to the routing table in each stack.

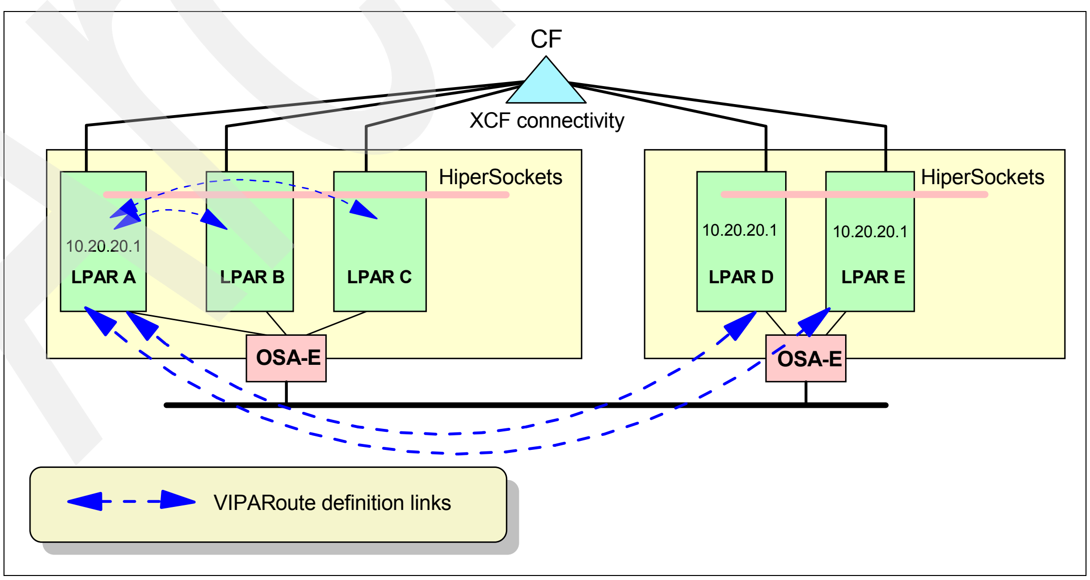


Figure 2-7 VIPARoute function

A VIPAROUTE statement (in the TCP/IP PROFILE) defines a route from a distributing stack or a backup distributing stack to a target stack. For example:

```
VIPAROUTE DEFINE 10.1.1.5 10.50.10.1
```

This statement says that traffic (within the sysplex) from 10.1.1.5 to 10.50.10.1 should be handled by the VIPAROUTE function, which selects the most appropriate route at that instant. You can use the command **NETSTAT VIPADYN /-v** to query the status of a dynamic VIPA, including the usage of VIPAROUTE; for example:

```
MVS TCP/IP NETSTAT CS V1R11 TCPIP Name: TCPIPC 11:24:56
Dynamic VIPA:IpAddr/PrefixLen: 10.20.20.1/28
Status: Active Origin: VIPADefine DistStat: Dist/Dest
VIPA Route:
DestXCF: 10.1.1.5
TargetIp: 10.50.10.1
RtStatus: Active
```

Note that z/OS releases prior to V1R7 cannot process a VIPAROUTE statement and will send distributed traffic using Dynamic XCF interfaces.

2.2 Importance of VIPA

VIPA provides a foundation technology solution for enhancing the availability and scalability of a z/OS system by providing the following capabilities:

- Automatic and transparent recovery from device and adapter failures.

  When a device (for example, a channel-attached router) or an adapter (for example, an OSA-Express adapter) fails, then another device or link can automatically provide alternate paths to the destination.

- Recovery when a z/OS TCP/IP stack leaves the XCF group (for example, resulting from some sort of failure), where an alternate z/OS TCP/IP stack has the necessary redundancy.

  Assuming that an alternate stack is installed to serve as a backup, the use of VIPAs enables the backup stack to activate the VIPA address of the failed stack. Connections on the failed primary stack will be disrupted, but they can be reestablished on the backup using the same IP address as the destination. In addition, the temporarily reassigned VIPA address can be restored to the primary stack after the cause of failure has been removed.

- Limited scope of a stack or application failure.

  If a DVIPA is distributed among several stacks, the failure of only one stack affects only the subset of clients connected to that stack. If the distributing stack experiences the failure, a backup assumes control of the distribution and maintains all existing connections.

- Enhanced workload management through distribution of connection requests.

  With a single DVIPA being serviced by multiple stacks, connection requests and associated workloads can be spread across multiple z/OS images according to Workload Manager (WLM) and Service Level Agreement policies (for example, QOS).

- Allows the nondisruptive movement of an application server to another stack so that workload can be drained from a system in preparation for a planned outage.

2.3 Timing of DVIPA activation

By default, as soon as TCP/IP stack starts and the basic stack function's initialization completes, the stack joins the TCP/IP sysplex group and processes its dynamic VIPA configuration within TCP/IP profile statements. It is possible to control the timing of when a stack joins the TCP/IP sysplex group and activates dynamic VIPAs either by TCP/IP profile definitions or by Vary TCPIP operator commands.

Figure 2-8 illustrates a timeline showing the process of sysplex-related definitions and autologged server binding to a dynamic VIPA when TCP/IP stack starts. The first row shows default processing without DELAYJOIN and DELAYSTART parameters. The second row shows the process with DELAYJOIN. The third row shows the one with both DELAYJOIN and DELAYSTART.

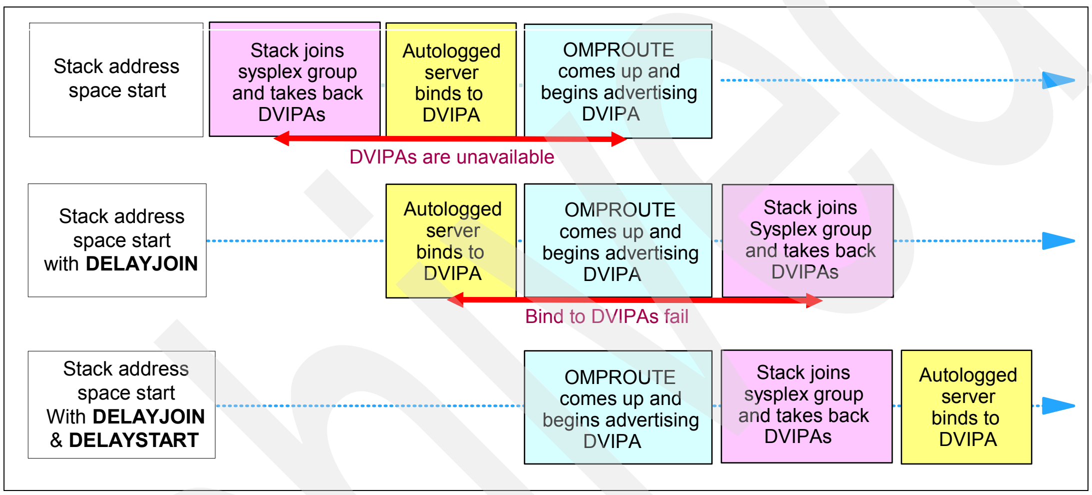


Figure 2-8 Startup sequence of TCP/IP stack

2.3.1 DVIPA definitions

z/OS Communications Server provides two options to control the timing of processing DVIPA-related definitions within the TCP/IP profile statements:

- DELAYJOIN (parameter on GLOBAL statement)
- DELAYSTART (parameter on AUTOLOG statement)

DELAYJOIN

DELAYJOIN specifies that joining the sysplex group and creating dynamic VIPAs is to be delayed until OMPROUTE is active.

When DELAYJOIN is specified, TCP/IP will not join the sysplex group until OMPROUTE is active. Because the stack's dynamic VIPA configuration is not processed until after the stack has joined the sysplex group, this delay in joining the sysplex group ensures that OMPROUTE will be active and ready to advertise dynamic VIPAs when they are created on this stack.

Using AUTOLOG with DELAYJOIN

Procedures specified in the AUTOLOG statement automatically start as soon as TCP/IP initialization is completed.

If DELAYJOIN is configured, TCP/IP will not join the sysplex group and create dynamic VIPAs until OMPROUTE is ready to advertise them. However, procedures specified in the AUTOLOG statement can be started before OMPROUTE is active. Binds to dynamic VIPAs will fail until OMPROUTE is initialized and the stack has joined the sysplex group and created the dynamic VIPAs.

When the TCP/IP stack is started with DELAYJOIN configured, the following steps occur:

1. The stack completes basic initialization, but it is not in the sysplex group yet.
2. At that point, AUTOLOG will start the specified procedures.
3. When OMPROUTE has completed its initialization and is active, it notifies the stack.
4. The stack will join the sysplex group and process its dynamic configuration, which includes creating its dynamic VIPAs.

Until the stack has completed its dynamic configuration processing, any bind request to a dynamic VIPA will fail due to the delay in creating these DVIPAs.

DELAYSTART

z/OS Communications Server provides an optional keyword, DELAYSTART, for procedures specified on the AUTOLOG profile statement. DELAYSTART is used to identify procedures that should not be automatically started until after the stack has joined the sysplex group and its dynamic sysplex configuration is completed. After that occurs, bind requests to dynamic VIPAs can follow.

> **Important:** Do not specify the DELAYSTART parameter for your OMPFine ROUTE procedure if you specify the DELAYJOIN parameter on the GLOBALCONFIG profile statement. Otherwise, the stack will complete its basic initialization, but OMPROUTE will never be started because AUTOLOG is waiting for the stack to join the sysplex group. And the stack will not join the group and create its dynamic VIPAs because the stack is waiting for OMPROUTE to be active.

Example 2-1 shows the AUTOLOG statement specified with DELAYSTART for the procedure name FTPDC in the TCP/IP profile.

Example 2-1 Sample AUTOLOG definition

```
AUTOLOG 5
   FTPDC  JOBNAME FTPDC1 DELAYSTART ; FTP server
   OMPC                             ; OMPROUTE Server
ENDAUTOLOG
```

You can use the NETSTAT CONFIG /-f command to display DELAYJOIN and DELAYSTART definitions, as illustrated in Example 2-2.

Example 2-2 Nestat Config/-f result

```
GLOBAL CONFIGURATION INFORMATION:
TCPIPSTATS: NO   ECSALIMIT: 0000000K  POOLLIMIT: 0000000K
MLSCHKTERM: NO   XCFGRPID:      IQDVLANID:  0
SEGOFFLOAD: NO   SYSPLEXWLMPOLL: 060
EXPLICITBINDPORTRANGE:  00000-00000
SYSPLEX MONITOR:
  TIMERSECS: 0060  RECOVERY: NO   DELAYJOIN: YES  AUTOREJOIN: NO
  MONINTF:   NO    DYNROUTE: NO
ZIIP:
```

```
  IPSECURITY:NO
NETWORK MONITOR CONFIGURATION INFORMATION:
PKTTRCSRV: NO   TCPCNNSRV: NO   SMFSRV: NO
AUTOLOG CONFIGURATION INFORMATION: WAIT TIME: 0300
PROCNAME: FTPDC     JOBNAME: FTPDC1   DELAYSTART: YES
  PARMSTRING:
PROCNAME: OMPC      JOBNAME: OMPC     DELAYSTART: NO
  PARMSTRING:
END OF THE REPORT
```

We used the FTP server FTPDC in the AUTOLOG statement as shown in Example 2-1 on page 28 and added a bind to the DVIPA in the PORT statement as follows:

```
PORT
20 TCP OMVS    NOAUTOLOG     ; FTP Server
21 TCP OMVS    BIND 10.1.8.30; control port
```

Syslog messages will help you understand how the DELAYJOIN and DELAYSTART options work. Example 2-3 shows the startup of TCPIPC with DELAYJOIN:NO and DELAYSTART:NO, which is the default.

Example 2-3 Syslog - DELAYJOIN:NO and DELAYSTART:NO

```
16:11:20.27           -S TCPIPC,PROFILE=PROFC32A
16:11:20.49  STC01790   EZZ7450I FFST SUBSYSTEM IS NOT INSTALLED
16:11:22.67  STC01790   EZZ0300I OPENED PROFILE FILE DD:PROFILE
16:11:22.67  STC01790   EZZ0309I PROFILE PROCESSING BEGINNING FOR DD:PROFILE
16:11:22.67  STC01790   EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE DD:PROFILE
16:11:22.68  STC01790   EZZ0641I IP FORWARDING NOFWDMULTIPATH SUPPORT IS ENABLED
16:11:22.69  STC01790   EZZ0350I SYSPLEX ROUTING SUPPORT IS ENABLED
16:11:22.69  STC01790   EZZ0624I DYNAMIC XCF DEFINITIONS ARE ENABLED
16:11:22.69  STC01790   EZZ0338I TCP PORTS 1 THRU 1023 ARE RESERVED
16:11:22.69  STC01790   EZZ0338I UDP PORTS 1 THRU 1023 ARE RESERVED
16:11:22.69  STC01790   EZZ0613I TCPIPSTATISTICS IS DISABLED
16:11:22.69  STC01790   EZZ4202I Z/OS UNIX - TCP/IP CONNECTION ESTABLISHED FOR
TCPIPC
16:11:22.81  STC01790   EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
16:11:22.81  STC01790   EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPC ARE AVAILABLE.
16:11:24.71  STC01790   EZD1176I TCPIPC HAS SUCCESSFULLY JOINED THE TCP/IP SYSPLEX
GROUP EZBTCPCS
16:11:32.60 STC01790 S FTPDC
16:11:32.60 STC01790 S OMPC
```

Example 2-4 shows the startup of TCPIPC with DELAYJOIN:YES and DELAYSTART:NO.

Example 2-4 Syslog - DELAYJOIN:YES and DELAYSTART:NO

```
16:20:58.77           -S TCPIPC,PROFILE=PROFC32A
16:20:59.05  STC01804   EZZ7450I FFST SUBSYSTEM IS NOT INSTALLED
16:21:01.48  STC01804   EZZ0300I OPENED PROFILE FILE DD:PROFILE
16:21:01.48  STC01804   EZZ0309I PROFILE PROCESSING BEGINNING FOR DD:PROFILE
16:21:01.49  STC01804   EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE DD:PROFILE
16:21:01.49  STC01804  *EZD1166E TCPIPC DELAYING SYSPLEX PROFILE PROCESSING -
OMPROUTE IS NOT ACTIVE
16:21:01.49  STC01804   EZZ0641I IP FORWARDING NOFWDMULTIPATH SUPPORT IS ENABLED
16:21:01.49  STC01804   EZZ0350I SYSPLEX ROUTING SUPPORT IS ENABLED
16:21:01.49  STC01804   EZZ0624I DYNAMIC XCF DEFINITIONS ARE ENABLED
```

```
16:21:01.49  STC01804   EZZ0338I TCP PORTS 1 THRU 1023 ARE RESERVED
16:21:01.49  STC01804   EZZ0338I UDP PORTS 1 THRU 1023 ARE RESERVED
16:21:01.49  STC01804   EZZ0613I TCPIPSTATISTICS IS DISABLED
16:21:01.49  STC01804   EZZ4202I Z/OS UNIX - TCP/IP CONNECTION ESTABLISHED FOR
TCPIPC
16:21:01.61  STC01804   EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
16:21:01.61  STC01804   EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPC ARE AVAILABLE.
16:21:11.52  STC01804  S FTPDC
16:21:11.52  STC01804  S OMPC
16:21:11.61  STC01805  IEF695I START OMPC WITH JOB OMPC IS ASSIGNED TO USER
TCPIP,GROUP TCPGRP
16:21:11.62  STC01806  IEF695I START FTPDC WITH JOB FTPDC IS ASSIGNED TO USER
TCPIP, GROUP TCPGRP
:
16:21:11.86  STC01787  BPXF024I (TCPIP) Sep 11 16:21:11 ftpd›196653: EZYFT13E bind
error : 048  EDC8116I Address not available. (errno2=0x744C7230)
16:21:11.86  STC01787  BPXF024I (TCPIP) Sep 11 16:21:11 ftpd›196653: EZY2714I FTP
server
049  shutdown in progress
16:21:11.86  STC01787  +EZY2714I FTP server shutdown in progress
16:21:11.87  STC01787  BPXF024I (TCPIP) Sep 11 16:21:11 ftpd›196653: EZYFT59I FTP
shutdown
051  complete.
16:21:11.87  STC01787  +EZYFT59I FTP shutdown complete
16:21:18.30  STC01804   EZD1176I TCPIPC HAS SUCCESSFULLY JOINED THE TCP/IP SYSPLEX
GROUP EZBTCPCS
```

Example 2-5 shows the startup of TCPIPC with DELAYJOIN:YES and DELAYSTART:YES.

Example 2-5 Syslog - DELAYJOIN:Yes and DELAYSTART:Yes

```
16:41:29.07            -S TCPIPC,PROFILE=PROFC32A
16:41:29.33  STC01813   EZZ7450I FFST SUBSYSTEM IS NOT INSTALLED
16:41:31.46  STC01813   EZZ0300I OPENED PROFILE FILE DD:PROFILE
16:41:31.46  STC01813   EZZ0309I PROFILE PROCESSING BEGINNING FOR DD:PROFILE
16:41:31.46  STC01813   EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE DD:PROFILE
16:41:31.47  STC01813  *EZD1166E TCPIPC DELAYING SYSPLEX PROFILE PROCESSING -
OMPROUTE IS NOT ACTIVE
16:41:31.47  STC01813   EZZ0641I IP FORWARDING NOFWDMULTIPATH SUPPORT IS ENABLED
16:41:31.47  STC01813   EZZ0350I SYSPLEX ROUTING SUPPORT IS ENABLED
16:41:31.47  STC01813   EZZ0624I DYNAMIC XCF DEFINITIONS ARE ENABLED
16:41:31.47  STC01813   EZZ0338I TCP PORTS 1 THRU 1023 ARE RESERVED
16:41:31.47  STC01813   EZZ0338I UDP PORTS 1 THRU 1023 ARE RESERVED
16:41:31.47  STC01813   EZZ0613I TCPIPSTATISTICS IS DISABLED
16:41:31.47  STC01813   EZZ4202I Z/OS UNIX - TCP/IP CONNECTION ESTABLISHED FOR
TCPIPC
16:41:31.59  STC01813   EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
16:41:31.59  STC01813   EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPC ARE AVAILABLE.
16:41:41.50  STC01813   S OMPC
16:41:41.58  STC01814   IEF695I START OMPC     WITH JOBNAME OMPC     IS ASSIGNED
TO USER
TCPIP   , GROUP TCPGRP
16:41:41.73  STC01814   EZZ7800I OMPC STARTING
16:41:46.87  STC01814   EZZ7898I OMPC INITIALIZATION COMPLETE
16:41:46.87  STC01814   EZZ8100I OMPC SUBAGENT STARTING
```

```
16:41:48.27  STC01813   EZD1176I TCPIPC HAS SUCCESSFULLY JOINED THE TCP/IP SYSPLEX
GROUP EZBTCPCS
16:41:50.36  STC01813   EZD1214I INITIAL DYNAMIC VIPA PROCESSING HAS COMPLETED FOR
TCPIPC
16:41:50.36  STC01813   S FTPDC
```

2.3.2 DVIPA commands

TCP/IP commands can assist with management, problem detection, and recovery of DVIPA address. The Vary TCPIP commands include:

- JOINGROUP and LEAVEGROUP
- DEACTIVATE and REACTIVATE
- QUIESCE and RESUME

JOINGROUP and LEAVEGROUP

These two commands are not unique to DVIPA; they apply to a complete TCP/IP stack. We discuss them here because they complement the other commands that specifically deal with DVIPAs. The JOINGROUP and LEAVEGROUP commands are used to connect or disconnect a whole TCP/IP stack from a Parallel Sysplex environment.

There can be many reasons for removing a TCP/IP stack from sysplex operation, including both normal operational decisions and problem situations. For example, sometimes situations occur where TCP/IP appears unresponsive without terminating. Causes can include the following:

- The downstream network has lost visibility to the distributing stack due to an OMPROUTE outage.
- VTAM is malfunctioning or data link control services are not working properly.
- TCP/IP is in a critical storage-constraint situation.
- XCF IP network connectivity (Dynamic XCF) between the distributing stack and the target stacks is not functioning.
- Abends/errors in the TCP/IP sysplex code components.

A stack's sysplex configuration data is retained in an inactive status when a stack leaves the sysplex, and this saved configuration is recovered when the stack rejoins the sysplex.

The LEAVEGROUP or JOINGROUP function can be automatic or can be initiated through an operator command. Automatic control is configured using a parameter in the GLOBALCONFIG statement in the TCP/IP PROFILE as follows:

```
GLOBALCONFIG SYSPLEXMONITOR AUTOREJOIN
```

This parameter has the following effects and controls:

- The stack rejoins the sysplex when the problem that caused it to leave the sysplex automatically is relieved.
- AUTOREJOIN is supported only in combination with the SYSPLEXMONITOR RECOVERY option (which causes the stack to leave the sysplex automatically if a problem is detected).
- Automatic rejoin is triggered by the events that clear the error condition (XCF links up again, OMPROUTE is restarted, and so forth.)
- Bounce prevention logic is built into the storage condition logic if storage limits are set on GLOBALCONFIG.

The manual operator commands use the following format:

```
VARY TCPIP,[stackname],SYSPLEX,JOINgroup
VARY TCPIP,[stackname],SYSPLEX,LEAVEgroup
```

These commands allow full operator control over when a stack leaves or rejoins the sysplex. The operator can also use the OBEY command to change the operational configuration (PROFILE) of a TCP/IP stack that is not currently in the sysplex and then have it rejoin the sysplex. The command syntax is as follows:

```
VARY TCPIP,[stackname],OBEY,DSN=my.sysplex.conf
```

This command replaces the PROFILE of a currently inactive sysplex TCP/IP stack and then causes it to rejoin the sysplex.

The following dynamic stack configuration definitions are not saved when a stack leaves a sysplex:

- Target DVIPAs

  These definitions are recreated automatically when the stack rejoins the group if this stack is still a target for another (distributing) stack.
- BIND-created or IOCTL-created DVIPAs

  These definitions must be recreated by the applications or by the MODDVIPA utility after the stack rejoins the group.

DEACTIVATE and REACTIVATE

These commands permit the operator to move individual stack-managed DVIPA Dynamic VIPAs. Prior to these commands, VARY OBEY commands and associated files were needed to cause a DVIPA takeover. Figure 2-9 illustrates these commands.

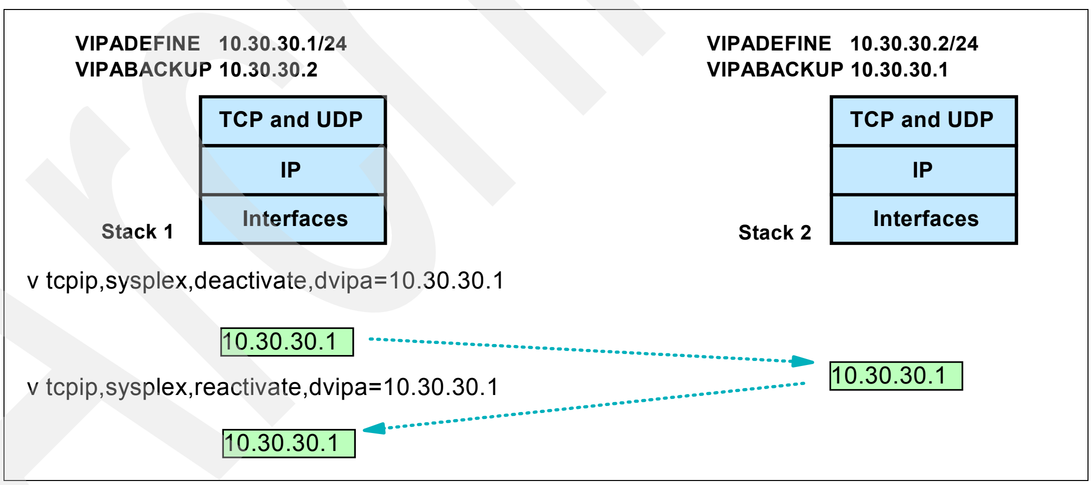


Figure 2-9 VIPA commands deactivate and reactivate

The format of the DEACTIVATE command is as follows:

```
V TCPIP,SYSPLEX,DEACTIVATE,DVIPA= xxx.xxx.xxx.xxx
```

The currently active stack providing the indicated DVIPA address is deactivated. A configured backup stack (if one exists) takes over the DVIPA, and a backup DVIPA can be deactivated, which removes its eligibility as a backup.

The format of the REACTIVATE command is as follows:

```
V TCPIP,SYSPLEX,REACTIVATE,DVIPA= xxx.xxx.xxx.xxx
```

The original stack regains ownership of the DVIPA, and a backup DVIPA can be reactivated, making it eligible to function as a backup.

Restriction: You cannot use these commands on a DVIPA that was created from VIPARANGE with `bind()`, `ioctl()`, or the MODDVIPA utility.

QUIESCE and RESUME

Those commands permit operator-initiated quiesce and resume of individual server applications or full target systems. The ability to quiesce a target system stack or an application instance, without a full system shutdown, is useful for many reasons including:

- For planned maintenance of the system or application
- When you need to drain the work queue from existing systems or applications prior to shutdown
- When you need to relieve temporary constraints of resources on target system
- When the quiesce is temporary and does not affect the Sysplex Distributor's permanent configuration
- When the commands are issued on the target system that is affected
- When you need to control individual server applications in a SHAREPORT group

The only way to achieve similar capability on earlier z/OS systems was through temporary configuration changes based on OBEYFILE commands.

The format for QUIESCE is as follows:

```
VARY TCPIP,,SYSPLEX,QUIESCE,options
```

The *`options`* field can be one or several of the following options:

`TARGET`	Quiesces all applications on the target stack
`PORT=xxx`	Quiesces all applications bound to the specified port on this stack
`JOBNAME=job name`	Allows quiesce of a single application in SHAREPORT group
`ASID=asid`	Further qualifies jobs that are quiesced (such as when dealing with duplicate job names)

After the QUIESCE, no new TCP connections are sent to the quiesced target (stack or application). Existing TCP connections are maintained (that is, the process is nondisruptive).

The format for RESUME is as follows:

```
VARY TCPIP,,SYSPLEX,RESUME,options
```

The *`options`* field is `[TARGET | PORT | JOBNAME | ASID]`, with the same parameter formats used for QUIESCE.

Notes:

- These commands must be issued on the system and TCP/IP stack where the application instance is running.
- The commands apply to a single TCP/IP stack's application instance.
- Any Sysplex Distributor timed affinities with the application instance being quiesced are terminated.
- Existing connections are not affected.
- The QUIESCE state for a TARGET persists for all application instances (existing and new) running on this TCP/IP stack, until the TCP/IP stack is recycled or a V TCPIP,,RESUME,TARGET command is issued.
- RESUME with the TARGET option is the only valid command following a QUIESCE with the TARGET option command.
- When a TCP/IP stack is quiesced, the "ready count" (Rdy) field that appears on the Netstat VDPT display (issued on the Sysplex Distributor routing stack) will be zero (0) for all entries associated with this target TCP/IP stack.

Use the NETSTAT ALL /-A command to display application status and the quiescing state, as illustrated in Example 2-6.

Example 2-6 NESTAT ALL/-A result

```
MVS TCP/IP NETSTAT CS V1R11       TCPIP NAME: TCPCS           17:40:36
Client Name: CICS1                        Client Id: 0000004A
Local Socket: 0.0.0.0..27                 Foreign Socket: 0.0.0.0..0
  Last Touched:       17:09:22              State:              Listen



   CurrentBacklog:     0000000000        MaximumBacklog:     0000000010
   CurrentConnections:    0000000300     SEF:                098
   SharePort: WLM
      RawWeight:        02                 NormalizedWeight:   01
      Quiesced:         Dest
```

The quiesced line indicates whether this server application has been quiesced for DVIPA Sysplex Distributor workload balancing. A `Dest` value indicates that this server will receive no new DVIPA Sysplex Distributor workload connections until the server application has been resumed. When the server application is resumed, the quiesced value changes to `No`.

For additional information about these commands, refer to *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

2.4 Dynamic VIPA example

We implemented a stack-based (VIPADEFINE and BACKUP) configuration. The following material relates the design, implementation, and operational techniques we used. We worked in two LPARs named SC30 and SC31, and we used TCP/IP stacks named TCPIPC. Our goal was to implement and exercise a failover situation (both automatic and operator-directed), as illustrated in Figure 2-10. To keep the example simple, we used FTP as the application.

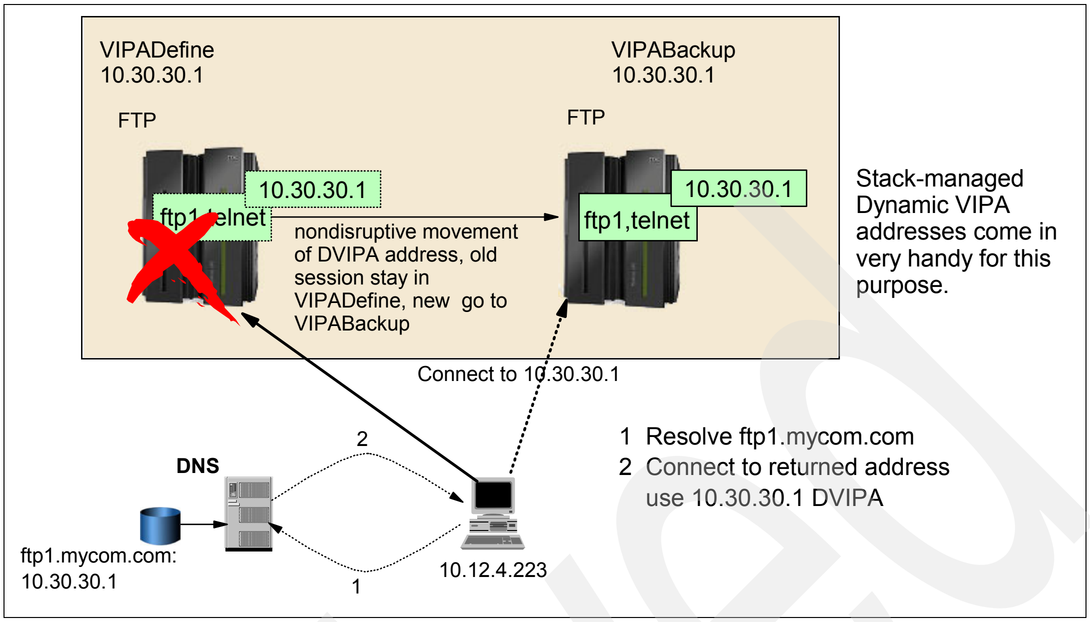


Figure 2-10 Stack-managed DVIPA with nondisruptive movement in TAKEBACK

2.4.1 Advantages

When a stack or its underlying z/OS fails, it is not necessary to restart the stack with the same configuration profile on a different z/OS image. After the stack leaves the XCF group, the VIPA address is moved automatically to another stack without the need for human intervention. The new stack receives information regarding the connections from the original stack and accepts new connections for the DVIPA. The routers are informed automatically about the change. This method increases availability because multiple server instances can be started in different stacks.

VIPA takeover allows complete flexibility in the placement of servers within sysplex nodes, not limited to traditional LAN interconnection with MAC addresses. Building on the VIPA concept means that spare adapters do not have to be provided, as long as the remaining adapters have enough capacity to handle the increased load. Spare processing capacity can be distributed across the sysplex rather than on a single node.

2.4.2 Dependencies

External access to our TCP/IP is through DNS and we needed OMPROUTE active to inform DNS about any changes to the DVIPA location. We also needed our TCP/IP stacks configured to support DYNAMICXCF, which is needed for nondisruptive movement of DVIPAs. Finally, we decided to use SYSPLEXROUTING, although this is optional.

2.4.3 Implementation

Our goal was to perform the following tasks:

- Provide the necessary definitions to use VIPA Define and VIPA Backup
- Use various commands to display the results of our definitions
- Exercise the takeover and takeback function in both automatic and manual operation

Figure 2-11 shows the base network configuration that we used.

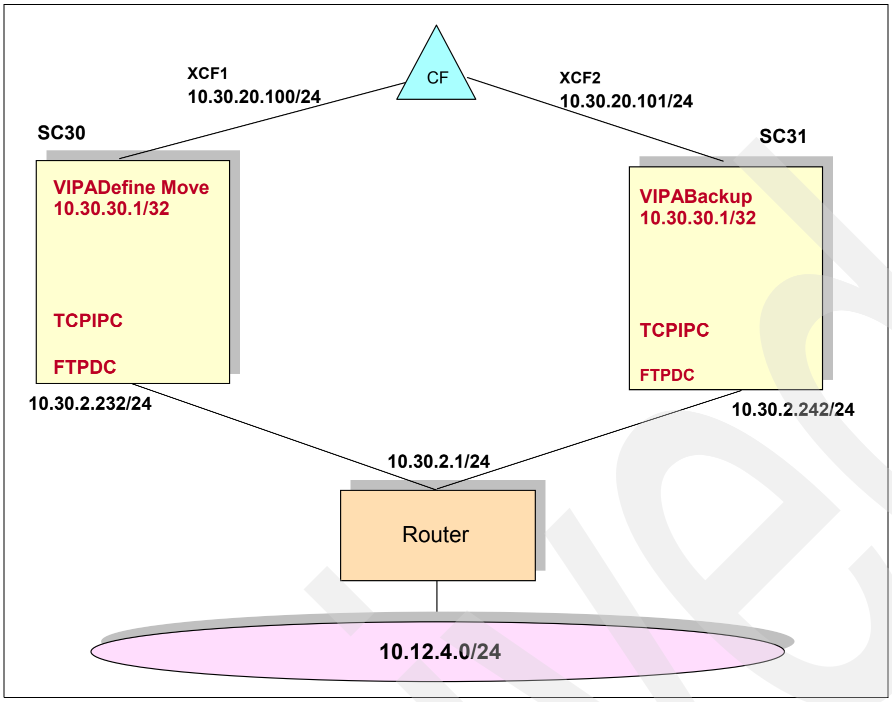


Figure 2-11 Base network diagram used for stack-managed implementation

Definitions

Remember that we are using LPARs SC30 and SC31, and our TCP/IP stack name is TCPIPC in both cases. The following material illustrates only the statements in the PROFILEs (and OMPRoute configuration) that are relevant to our specific task.

Our PROFILE for TCPIPC on SC30 included the statements shown in Example 2-7.

Example 2-7 Our PROFILE for TCPIPC on SC30

```
IPCONFIG DATAGRAMFWD SYSPLEXROUTING
DYNAMICXCF 10.30.20.100 255.255.255.0 8
SOMAXCONN 10
TCPCONFIG TCPSENDBFRSIZE 16K TCPRCVBUFRSIZE 16K SENDGARBAGE FALSE
TCPCONFIG RESTRICTLOWPORTS
UDPCONFIG RESTRICTLOWPORTS
; VIPADEFine/VIPABackup
VIPADynamic
VIPADEFine MOVEable IMMEDiate 255.255.255.255 10.30.30.1
ENDVIPADynamic
```

Example 2-8 shows the statements in our PROFILE for TCPIPC on SC31.

Example 2-8 Our PROFILE for TCPIPC on SC31

```
IPCONFIG DATAGRAMFWD SYSPLEXROUTING
DYNAMICXCF 10.30.20.101 255.255.255.0 8
SOMAXCONN 10
TCPCONFIG TCPSENDBFRSIZE 16K TCPRCVBUFRSIZE 16K SENDGARBAGE FALSE
TCPCONFIG RESTRICTLOWPORTS
; VIPADEFine/VIPABackup
VIPADynamic
VIPABACKUP 1 MOVEable IMMEDiate 255.255.255.255 10.30.30.1
ENDVIPADynamic
```

Example 2-9 shows the statements in our OMPRoute definitions.

Example 2-9 Our OMPRoute definitions

```
OMPROUTE config
; VIPADefine/VIPABackup vipa
ospf_interface ip_address=10.30.30.*
           subnet_mask=255.255.255.0
           attaches_to_area=0.0.0.0
           cost0=10
```

Verification

After activating the TCP/IP stacks with these definitions, we used operator commands to display VIPADCFG and VIPADYM status. Example 2-10 shows the commands and results from the two LPARs.

Example 2-10 Display results VIPA definition for TCPIPC (SC30 and 31)

```
D TCPIP,TCPIPC,N,VIPADCFG
RESPONSE=SC30
DYNAMIC VIPA INFORMATION:
VIPA DEFINE:
IPADDR/PREFIXLEN: 10.30.30.1/32
MOVEABLE: IMMEDIATE  SRVMGR: NO

D TCPIP,TCPIPC,SYSPLEX,VIPADYN
RESPONSE=SC30
 EZZ8260I SYSPLEX CS V1R11 215
 VIPA DYNAMIC DISPLAY FROM TCPIPC   AT SC30
 LINKNAME: VIPL0A1E1E01
 IPADDR/PREFIXLEN: 10.30.30.1/32
   ORIGIN: VIPADEFINE
   TCPNAME  MVSNAME  STATUS RANK DIST
   -------- -------- ------ ---- ----
   TCPIPC   SC30     ACTIVE
   TCPIPC   SC31     BACKUP 001
 2 OF 2 RECORDS DISPLAYED

D TCPIP,TCPIPC,SYSPLEX,VIPADYN
RESPONSE=SC31
 EZZ8260I SYSPLEX CS V1R11 584
 VIPA DYNAMIC DISPLAY FROM TCPIPC   AT SC31
```

```
 LINKNAME: VIPLOA1E1E01
 IPADDR/PREFIXLEN: 10.30.30.1/32
   ORIGIN: VIPABACKUP
   TCPNAME  MVSNAME  STATUS RANK DIST
   -------- -------- ------ ---- ----
   TCPIPC   SC31     ACTIVE
```

We also displayed the OSPF routing table on TCPIPC, as shown in Example 2-11. Note the presence of the local dynamic VIPA 1 with its stack-generated link name.

Example 2-11 Display OMPRoute after DVIPA activation

```
D TCPIP,TCPIPC,OMPROUTE,RTTABLE
000090   DIR*  10.30.30.0       FFFFFF00  1       595     10.30.30.1
000090   DIR*  10.30.30.1       FFFFFFFF  1       595     VIPLOA1E1E01
000090   SPF   10.30.60.0       FFFFFF00  11      595     10.30.2.1
000090  STAT*  10.40.20.100     FFFFFFFF  0       36977   10.30.20.100
000090  STAT*  10.40.20.101     FFFFFFFF  0       36977   10.30.20.100
000090  STAT*  10.40.20.102     FFFFFFFF  0       36977   10.30.20.100
000090
000090  DEFAULT GATEWAY IN USE.
```

We started the FTP service in the client workstation to use this DVIPA address. We then checked the new status of the session, as shown in Example 2-12.

Example 2-12 Display sessions using DVIPA address

```
D TCPIP,TCPIPC,N,CON
RESPONSE=SC30
USER ID  CONN     STATE
FTPDC1   00000724 ESTBLSH
   LOCAL SOCKET:   ::FFFF:10.30.30.1..21
   FOREIGN SOCKET: ::FFFF:10.12.4.223..2222
```

2.4.4 Automatic VIPA takeover and takeback

We tried two different options for this exercise, based on MOVE IMMED and MOVE WHENIDLE parameters. When a stack leaves the XCF group (such as resulting from a failure), it causes immediate movement of a DVIPA to its backup stack. These two parameters determine how the original stack takes back the DVIPA from the backup stack when the problem is resolved.

The MOVE IMMED causes the DVIPA to be moved back to the primary stack immediately after the primary stack is available. This might disturb some IP sessions. The MOVE WHENIDLE parameter causes the takeback to be delayed until all connections are finished on the backup stack.

MOVE IMMED

We defined the primary VIPA IP addresses of 10.30.30.1 on TCPIPC in SC30 and we defined the same VIPA address in TCPIPC on SC31 as the backup. We started our server application (FTP) on both stacks and established connections from a client to the server on TCPIPC in SC30 10.30.30.1.

We stopped the TCPIPC to verify that the automatic takeover function would work. Example 2-13 shows the result of this operation. The network is informed, using dynamic routing of OSPF, that the VIPA has moved and all connection requests are routed to the new stack owning the VIPA now. All the other sysplex stacks were informed of the failure through XCF (because the stack left the XCF group) and took steps to recover the failed VIPA. The stack with the highest rank (the only stack) on the backup list for 10.30.30.1 was TCPIPC on SC31.

Example 2-13 Display after the stopped of TCPIC in SC30

```
EZZ4201I TCP/IP TERMINATION COMPLETE FOR TCPIPC
IEF352I ADDRESS SPACE UNAVAILABLE
$HASP395 TCPIPC   ENDED
EZZ8301I VIPA 10.30.30.1 TAKEN OVER FROM TCPIPC ON SC30

D TCPIP,TCPIPC,SYSPLEX,VIPAD
EZZ8260I SYSPLEX CS V1R11 512
VIPA DYNAMIC DISPLAY FROM TCPIPC   AT SC31
LINKNAME: VIPL0A1E1E01
IPADDR/PREFIXLEN: 10.30.30.1/32
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPC   SC31     ACTIVE
1 OF 1 RECORDS DISPLAYED

RESPONSE=SC31
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IPADDR/PREFIXLEN: 10.30.30.1
      RANK: 001  MOVEABLE:           SRVMGR:
END OF THE REPORT

D TCPIP,TCPIPC,N,CON
(Display of new ftp session using 10.30.30.1 in VIPABACKUP)
FTPDC1   00000831 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.30.1..21
  FOREIGN SOCKET: ::FFFF:10.12.4.223..2234
FTPDC1   0000082D ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.30.1..21
  FOREIGN SOCKET: ::FFFF:10.12.4.223..2233
```

> **Note:** A “failure” like this is disruptive to connections that are active when the takeover occurs. However, most TCP/IP applications recover from this automatically. Movements during takeback are nondisruptive.

We then restarted the original TCPIPC stack on SC30, including the FTP application. The VIPA address 10.30.10.1 returned automatically from SC31 to SC30. Example 2-14 shows the console messages associated with this action.

Example 2-14 Console message after restarting the failed TCP/IP stack in SC30

```
EZZ8302I VIPA 10.30.30.1 TAKEN FROM TCPIPC ON SC31
EZZ8303I VIPA 10.30.30.1 GIVEN TO TCPIPC ON SC30
```

TCPIPC in SC31 had the VIPA 10.30.30.1 active and had some active connections when the address was taken back by SC30. These active connections were not broken. Example 2-15 shows the existing to connections to SC31. However, TCPIPC in SC30 is now receiving all new connections. Example 2-16 shows the creation of a new FTP connection.

Example 2-15 Display of the connections to SC31 after the takeback to SC30

```
D TCPIP,TCPIPC,N,VCRT
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.30.30.1..21
  SOURCE: 10.12.4.223..2252
DEST:      10.30.30.1..21
FTPDC1   00000B05 FINWT2
  LOCAL SOCKET:   ::FFFF:10.30.30.1..21
  FOREIGN SOCKET: ::FFFF:10.12.4.223..2252
OMPC     0000002B ESTBLSH
```

Example 2-16 shows the VIPA connection routing table (VCRT) in both stacks, which include the old and new connections.

Example 2-16 Display of the older session in SC31 and new session in SC30 after the takeback

```
RESPONSE=SC30
DYNAMIC VIPA CONNECTION ROUTING TABLE:
 DEST:      10.30.30.1..21
   SOURCE: 10.12.4.223..2256
   DESTXCF: 10.30.20.100      2
 DEST:      10.30.30.1..21
   SOURCE: 10.12.4.223..2252
   DESTXCF: 10.30.20.101       1
SOURCE: 10.12.4.223..2253
   DESTXCF: 10.30.20.101       1
 3 OF 3 RECORDS DISPLAYED
 END OF THE REPORT

D TCPIP,TCPIPC,N,CONN
FTPDC1   0000003A ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.30.1..21
  FOREIGN SOCKET: ::FFFF:10.12.4.223..2256
```

In this example, the numbers correspond to the following information:

1. DestXCF 10.30.20.101 are connections that are still active on SC31.
2. DestXCF 10.30.20.100 is the new FTP connection to TCPIPC in SC30 after the takeback.

MOVE WHENIDLE

In this case, we used the same definitions and processes as for the last example, except that we changed the parameter MOVE IMMED to MOVE WHENIDLE in VIPA DEFINE statements.

The behavior was exactly the same as in the previous scenario. TCPIPC was informed of the failure through XCF and took steps to recover the failed DVIPA. TCPIPC on SC31 dynamically defined and activated this DVIPA address.

We then restarted TCPIPC in SC30 and started a new connection to DVIPA address 10.30.30.1. Because we defined this DVIPA as MOVE WHENIDLE, the DVIPA was not taken back to SC30 because there were active connections to 10.30.30.1 on SC31.

Example 2-17 shows a new FTP connection to DVIPA address 10.30.30.1 and shows that this connection was established with TCPIPC in SC31, because the DVIPA has not yet been taken back.

Example 2-17 New connection still going to SC31

```
FTPDC1   00000044 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.30.1..21
  FOREIGN SOCKET: ::FFFF:10.12.4.223..2362
FTPDC1   0000012B ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.30.1..21
  FOREIGN SOCKET: ::FFFF:10.12.4.223..2409
```

As long as the sessions exist in SC31, the active DVIPA stays on the stack in SC31. Example 2-18 shows this result.

Example 2-18 VIPA active in SC31

```
RESPONSE=SC30
 EZZ8260I SYSPLEX CS V1R11 378
 VIPA DYNAMIC DISPLAY FROM TCPIPC   AT SC30
 IPADDR: 10.30.30.1
   ORIGIN: VIPADEFINE
   TCPNAME  MVSNAME  STATUS RANK DIST
   -------- -------- ------ ---- ----
   TCPIPC   SC31     ACTIVE
   TCPIPC   SC30     BACKUP 255
 IPADDR: 255.255.255.255
   TCPNAME  MVSNAME  STATUS RANK DIST
   -------- -------- ------ ---- ----
   TCPIPC   SC31     BACKUP 001
```

After all the FTP sessions are closed, the takeback is performed, and the active DVIPA returns to SC30. Example 2-19 shows the result of this operation.

Example 2-19 Takeback DVIPA after close sessions

```
EZZ8303I VIPA 10.30.30.1 GIVEN TO TCPIPC ON SC30
EZZ8301I VIPA 10.30.30.1 TAKEN OVER FROM TCPIPC ON SC31
```

> **Note:** The VIPADEFINE MOVEABLE WHENIDLE parameter might be retired in a future release, because MOVE IMMED is the most common choice.

Deactivate and reactivate operator commands

Using the same configuration, we issued a DEACTIVATE command to force a takeover process. This moved the active DVIPA address from SC30 to SC31 in the same way a failure in SC30 would cause the move. Example 2-20 shows the results of this manual operation.

Example 2-20 Display the result of deactivate command in SC30

```
V TCPIP,TCPIPC,SYSPLEX,DEACTIVATE,DVIPA=10.30.30.1
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,SYSPLEX,DEACTIVATE,
DVIPA=10.30.30.1
EZD1197I THE VARY TCPIP,,SYSPLEX,DEACTIVATE,DVIPA COMMAND COMPLETED

D TCPIP,TCPIPC,N,VIPAD
```

```
DYNAMIC VIPA:
  IPADDR/PREFIXLEN: 10.30.30.1/32
    STATUS: QUIESCING  ORIGIN: VIPADEFINE
```

Example 2-21 shows that SC31 now has the DVIPA active.

Example 2-21 System SC31 assumes the VIPA address after the deactivate

```
RO SC31,D TCPIP,TCPIPC,N,VIPAD
RESPONSE=SC31
DYNAMIC VIPA:
   IPADDR/PREFIXLEN: 10.30.30.1/32
     STATUS: ACTIVE     ORIGIN: VIPABACKUP
   IPADDR/PREFIXLEN: 255.255.255.255
     STATUS: BACKUP     ORIGIN: VIPABACKUP
```

We next used REACTIVATE to start the takeback function, moving the DVIAP address from SC31 to SC30. We issued this command for SC30. Example 2-22 shows the results of this manual operation.

Example 2-22 Result of takeback operation using reactivate command in SC30

```
V TCPIP,TCPIPC,SYSPLEX,REACTIVATE,DVIPA=10.30.30.1
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,SYSPLEX,REACTIVATE,DV
IPA=10.30.30.1
EZZ8302I VIPA 10.30.30.1 TAKEN FROM TCPIPC ON SC31
EZZ8303I VIPA 10.30.30.1 GIVEN TO TCPIPC ON SC30
EZD1189I THE VARY TCPIP,,SYSPLEX,REACTIVATE,DVIPA COMMAND COMPLETED
SUCCESSFULLY
```

We started two FTP sessions to TCPIPC in SC31 while that system had the active DVIPA address. After the REACTIVATE, we started two more FTP sessions. Example 2-23 shows the VIPA Connection Routing Table (VCRT) in both stacks, which include the old and new connections. DestXCF 10.30.20.101 has the old connections that are still active on SC31. DestXCF 10.30.20.100 has the new FTP connection to TCPIPC in SC30.

Example 2-23 Display VCRT shows the old and new sessions SC31 with moving status

```
D TCPIP,TCPIPC,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V1R11 794
RESPONSE=SC30
 EZZ8260I SYSPLEX CS V1R11 794
 VIPA DYNAMIC DISPLAY FROM TCPIPC   AT SC30
 LINKNAME: VIPL0A1E1E01
 IPADDR/PREFIXLEN: 10.30.30.1/32
   ORIGIN: VIPADEFINE
   TCPNAME  MVSNAME  STATUS RANK DIST
   -------- -------- ------ ---- ----
   TCPIPC   SC30     ACTIVE
   TCPIPC   SC31     MOVING

D TCPIP,TCPIPC,N,VCRT
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.30.30.1..20 1
  SOURCE: 10.12.4.223..2648
  DESTXCF: 10.30.20.100
DEST:      10.30.30.1..20
  SOURCE: 10.12.4.223..2639
  DESTXCF: 10.30.20.101
DEST:      10.30.30.1..20 2
  SOURCE: 10.12.4.223..2642
  DESTXCF: 10.30.20.101
DEST:      10.30.30.1..21
  SOURCE: 10.12.4.223..2643
  DESTXCF: 10.30.20.100
```

In this example, the numbers correspond to the following information:

1. DestXCF 10.30.20.100 are the new FTP connections to TCPIPC in SC30 after the takeback.
2. DestXCF 10.30.20.101 are the old connections that are still active on SC31.

> **Note:** The existing connections during takeback remain up. They are not disrupted.
>
> After all sessions to the backup are closed, then new connections go to the primary DVIPA machine. The *moving status* continues until the last connections end, then the status changes to backup. (This assumes that MOVE WHENIDLE is in effect.)

3

VIPA without dynamic routing

As previously mentioned, we formerly recommended using a dynamic routing protocol in your mainframes for high availability. Now, innovative designs take advantage of z/OS support of Address Resolution Protocol (ARP) takeover to achieve comparable availability without using a dynamic routing protocol. In this chapter, we explain how to design and implement high availability for your z/OS Communications Server environment without dynamic routing.

This chapter discusses the following topics.

Section	Topic
3.1, "Basic concepts" on page 46"	Background information and an explanation of why this solution can be important in typical environments
3.2, "High availability without dynamic routing" on page 51	Key design points and the steps through detailed implementation.
3.3, "High availability scenarios" on page 57	Failing scenarios to test and verify the high availability environment we implemented
3.4, "Debugging tips" on page 62	Summary of debugging approaches useful when working with layer two definitions

3.1 Basic concepts

High availability designs that do not use dynamic routing protocols (which are part of TCP/IP) achieve availability based on local area networking (LAN) capabilities instead (such LAN capabilities are also often referred to as layer 2 functions because LANs constitute layers 1 and 2 of the seven-layer OSI networking model).

The most fundamental of those capabilities is the ability to transfer an IP address from a failing (or failed) physical interface or stack to a different physical interface or stack. In an z/OS Communications Server environment, *dynamic* movement of an IP address (without the use of dynamic routing) is called ARP takeover, and it requires OSA-Express adapters (in QDIO mode).

In addition, if you want to move an IP address from one *adapte*r to another within a system, you must also set up Static Virtual IP Addresses (VIPA). If you want to implement dynamic movement of an IP address from one TCP/IP *stack* to another (including a stack on a completely different system), you must set up Dynamic Virtual IP Addressing (DVIPA), which, in turn, requires Sysplex XCF communications.

> **Note:** Virtual IP Addressing is introduced, along with implementation scenarios, in Chapter 2, "Virtual IP addressing" on page 13.

ARP takeover uses LAN broadcast protocols. Therefore, to use ARP takeover, all of the systems in your sysplex must be connected to the same LAN (or virtual LAN, VLAN) and thus, in the same TCP/IP subnet (also called *flat* network connectivity). Also, ARP takeover does not provide protection from failures in routers or switches in the network. Backup and recovery for these elements must be considered separately.

A *flat* network is one in which all stacks can reach each other without going through intermediate bridges or routers that transform IP addresses. Expressed a different way, a flat network is contained in one network segment or subnet. It is one *broadcast domain*.

Stacks within a flat network communicate with each other at layer 2 of the OSI model. Layer 2 frames are switched based on MAC addresses. (Layer 3 frames are switched based on IP addresses.) A flat network does not preclude the incorporation of transparent switches or bridges, and it might support up to a few hundred stations.

In the design discussed here, all z/OS systems are directly attached to the same IP subnet. A primary DVIPA TCP/IP stack and a backup DVIPA stack exist. Each has a unique MAC address.

If the backup DVIPA stack detects a failure of the primary stack (when it leaves the XCF group), it takes over the IP address of the primary stack but retains its unique MAC address. It then responds with its own MAC address to ARP requests that are searching for the host with the required IP address (which has not changed with the move from the primary to the backup stack).

No external routing changes are needed as a result of the move from the primary stack to the backup stack. Therefore, a dynamic routing protocol is not required.

3.1.1 ARP takeover

ARP takeover is a function that allows traffic to be redirected from a failing OSA connection to another OSA connection. This function is supported with IPv4 and IPv6 OSA interfaces.

When an OSA DEVICE defined in a TCP/IP stack is started, all IP addresses associated with that OSA port in QDIO mode (which is device type MPCIPA in the TCP/IP profile) are dynamically downloaded to the OSA card.

> **Note:** If you want to use the ARP takeover function, use your OSA-Express or OSA-Express2 port in QDIO mode. If the port is defined in non-QDIO mode (device type LCS in the TCP/IP profile), then you must use OSA/SF to build and activate a configuration that identifies the multiple IP addresses that *might* be used with the adapter. In a DVIPA environment (or a simple OSA takeover environment), the use of QDIO mode simplifies setup and management.

If an OSA port fails while there is a backup OSA port available on the same subnetwork, then TCP/IP informs the backup adapter as to which IP addresses (real and VIPA) to take over and network connections are maintained. After it is set up correctly, the fault tolerance provided by the ARP takeover function is automatic. Figure 3-1 shows the ARP Takeover process.

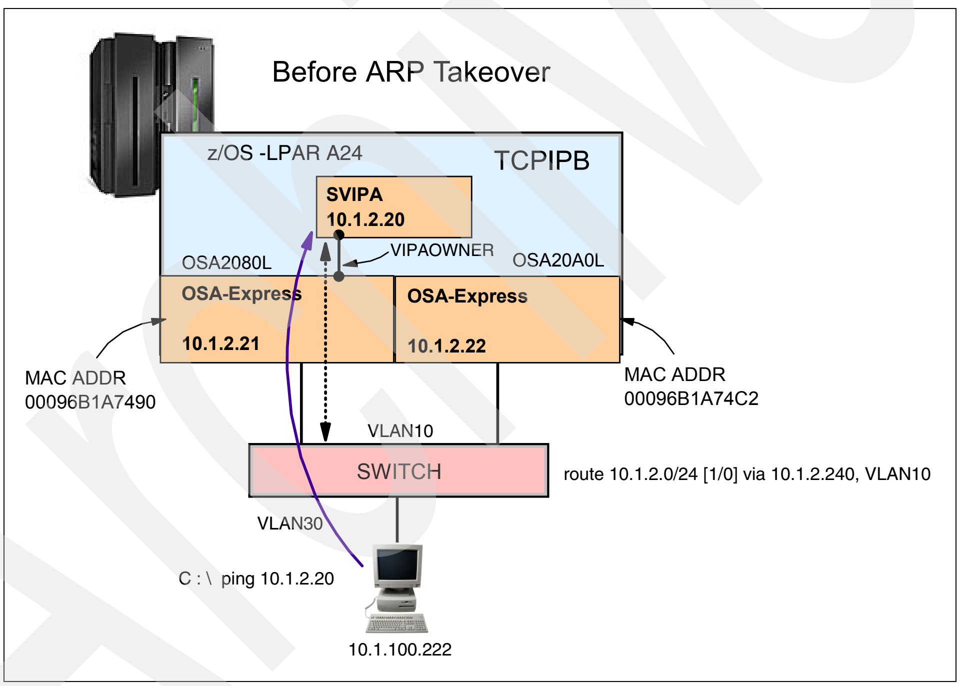


Figure 3-1 ARP takeover environment

> **Note:** For demonstration purposes, we show a single-switch environment in our examples. However, to avoid single points of failure, always separate your OSA connections across multiple switches.

Example 3-1 shows the IP address and MAC address relationship for each active OSA interface using the **`Display Netstat,Arp`** command.

Example 3-1 Display result of a NETSTAT, ARP command before takeover

```
QUERYING ARP CACHE FOR ADDRESS 10.1.2.21
LINK: OSA2080L          ETHERNET: 00096B1A7490
QUERYING ARP CACHE FOR ADDRESS 10.1.2.22
LINK: OSA20A0L          ETHERNET: 00096B1A74C2
```

ARP takeover

Figure 3-2 shows our environment as we simulated an error condition by stopping the device OSA2080 in the TCP/IP stack.

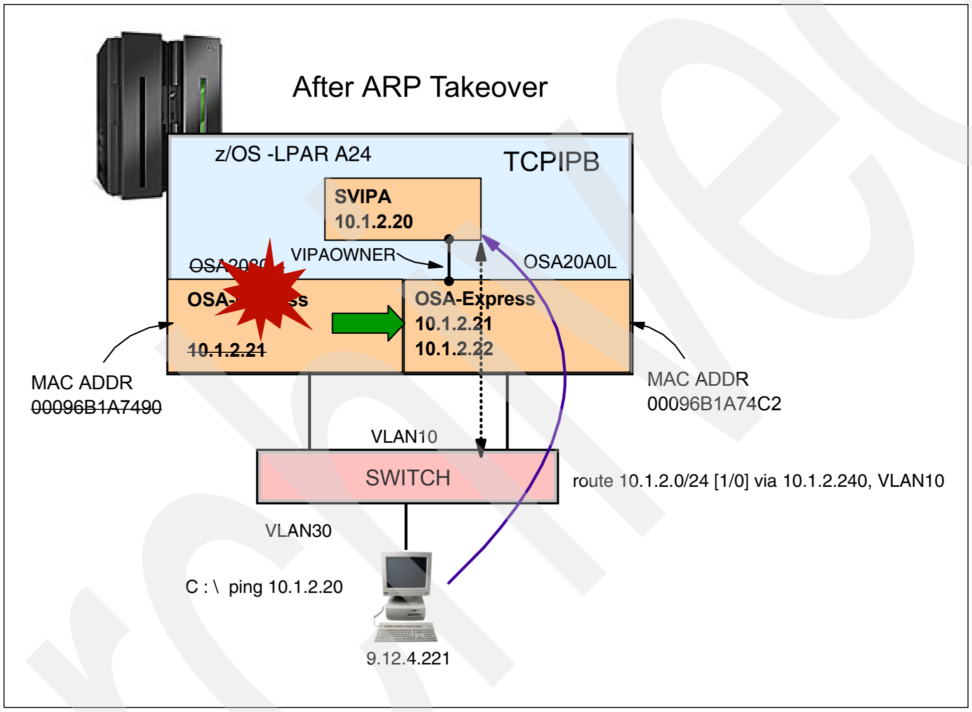


Figure 3-2 ARP takeover after OSA2080 failed

This simulated failure produced messages on the z/OS console, as shown in Example 3-2. The EZZ4329I message marked as 1 shows that OSA20A0L has taken over ARP responsibility for OSA2080L. OSA20A0L will respond to all ARP requests for IP addresses formerly owned by OSA2080L.

Example 3-2 ARP takeover message after stopping OSA2080

```
V TCPIP,TCPIPB,STOP,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPB,STOP,OSA2080
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4329I LINK OSA20A0L HAS TAKEN OVER ARP RESPONSIBILITY FOR
INACTIVE LINK OSA2080L 1
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE OSA2080
```

Example 3-3 shows a display of the ARP cache tables from TCPIPB.

Example 3-3 Display results of a NETSTAT, ARP command after takeover

```
D TCPIP,TCPIPB,N,ARP
QUERYING ARP CACHE FOR ADDRESS 10.1.2.22
LINK: OSA20A0L          ETHERNET: 00096B1A74C2 1
QUERYING ARP CACHE FOR ADDRESS 10.1.2.21
LINK: OSA20A0L          ETHERNET: 00096B1A74C2 1
```

Both IP addresses were pointed to the same MAC address marked as 1, which was associated with the link OSA20A0L. TCPIPB notified all hosts on the LAN by broadcasting gratuitous ARPs when the link (OSA2080L) stopped.

Example 3-4 shows the status of VIPAOWNER after taking over.

Example 3-4 Display result of a NETSTAT, DEV command after takeover

```
D TCPIP,TCPIPB,NETSTAT,DEV
LANGROUP: 00002
   LNKNAME           LNKSTATUS   ARPOWNER          VIPAOWNER
   -------           ---------   --------          ---------
   OSA20A0L          ACTIVE      OSA20A0L          YES 1
   OSA2080L          NOT ACTIVE  OSA20A0L 2        NO
```

In this example, the numbers correspond to the following information:

1. The VIPAOWNER was switched to link OSA20A0L.

2. Link OSA2080L was inactive, and was no longer the ARPOWNER.

ARP takeback

We started the OSA2080 device using a VARY TCPIP,,START command. Figure 3-3 shows our environment after starting the device OSA2080.

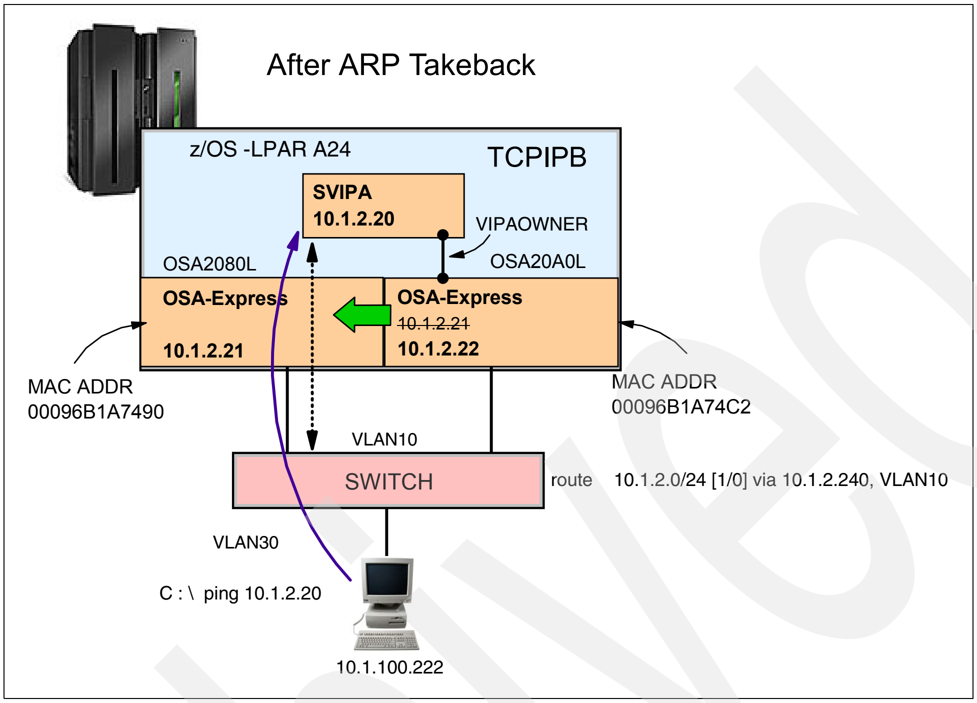


Figure 3-3 ARP takeback after OSA20A0 recovered

When the OSA2080 device was started using a VARY TCPIP,,START command, the messages in Example 3-5 were displayed.

Example 3-5 Display result

```
V TCPIP,TCPIPB,START,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPB,START,OSA2080
EZZ0053I COMMAND VARY START COMPLETED SUCCESSFULLY
IEF196I IEF237I 2081 ALLOCATED TO TP2081
IEF196I IEF237I 2080 ALLOCATED TO TP2080
IEF196I IEF237I 2082 ALLOCATED TO TP2082
EZD0013I LINK OSA2080L HAS TAKEN BACK ARP RESPONSIBILITY FROM LINK 1
OSA20A0L
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE OSA2080
```

The EZD0013I message marked as 1 showed that the link OSA2080L had taken back the ARP responsibility from the link OSA20A0L.

After the link OSA2080L was taken back, we verified that the ARP cache tables were correctly modified, using the **`Display Netstat Arp`** and **`Dev`** commands, as shown in Example 3-6 and Example 3-7.

Example 3-6 Display result of a NETSTAT, ARP command after takeback

```
D TCPIP,TCPIPB,NETSTAT,ARP
QUERYING ARP CACHE FOR ADDRESS 10.1.2.21
LINK: OSA2080L          ETHERNET: 00096B1A7490 1
QUERYING ARP CACHE FOR ADDRESS 10.1.2.22
```

```
LINK: OSA20A0L         ETHERNET: 00096B1A74C2
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

In this example, the number corresponds to the following information:

1. The ARP cache now referenced the real MAC for address 10.1.2.21. A gratuitous ARP was broadcasted on the LAN by the TCPIP stack, and the ARP cache in the switch was updated with the correct MAC addresses.

Example 3-7 Display result of a NETSTAT, DEV command after takeback

```
D TCPIP,TCPIPB,NETSTAT,DEV
LANGROUP: 00002
  LNKNAME          LNKSTATUS   ARPOWNER          VIPAOWNER
  -------          ---------   --------          ---------
  OSA20A0L         ACTIVE      OSA20A0L          YES        2
  OSA2080L         ACTIVE      OSA2080L          NO         1
```

In this example, the number corresponds to the following information:

2. The link OSA2080L was now active and had its own ARPOWNER status back. However, link OSA20A0L kept the VIPAOWNER. This is because a VIPA belongs to a stack and not to a particular interface; therefore, it does not matter which OSA is the VIPAOWNER.

Note: If spanning tree is used in the bridge or switch that is connected to an OSA QDIO port, the Spanning Tree Protocol must be turned ON to allow the routing protocol packets that advertise a VIPA to flow correctly. Without the Spanning Tree Protocol enabled, there is a transition through the Listening and Learning stages (30 seconds in total) of Spanning Tree before the port can actually pass valid traffic.

3.2 High availability without dynamic routing

In our environment, we used two OSA adapters in QDIO mode, connected to two switches. We used a unique VLAN (VLAN 10), spanning our switches, and connected our hosts to that VLAN.

We worked in two LPARs, A23 (system SC30) and A24 (system SC31), using static routing between them. Our DVIPA fail-over implementation moves the active IP address from one LPAR to the other when needed.

Our design had the following dependencies:

- We needed separate LAN adapters for each stack. We used OSA-Express2 adapters in QDIO mode. QDIO mode is recommended because the OSA internal IP tables (OATs) are updated dynamically. Other LAN interfaces, such as channel-attached routers, cannot be used in the high availability configuration described in this chapter.
- All z/OS images must be connected to the same LAN segment, so that all adapters receive broadcast frames.
- This scenario is suitable for a subnet with a small number of hosts. If the subnet had more hosts, then we must consider the potential effects of LAN broadcast (sometimes called *broadcast storms*). This exposure is not directly related to the DVIPA operation we are describing, but should be considered when designing high availability network environments.

- To show a flat network scenario, all resources in VLAN 10, including the VIPA addresses, belong to the same subnet.

This design has the following advantages:

- It is simple and easy to configure, and it avoids the complexities of implementing dynamic routing.
- No additional routers or router paths are required.
- This design supports DVIPA takeover and takeback functions as well as distributed DVIPA.

3.2.1 Implementation

Figure 3-4 illustrates our working environment. The OSA-Express2 CHPIDs were defined as OSD types, causing them to work in QDIO mode.

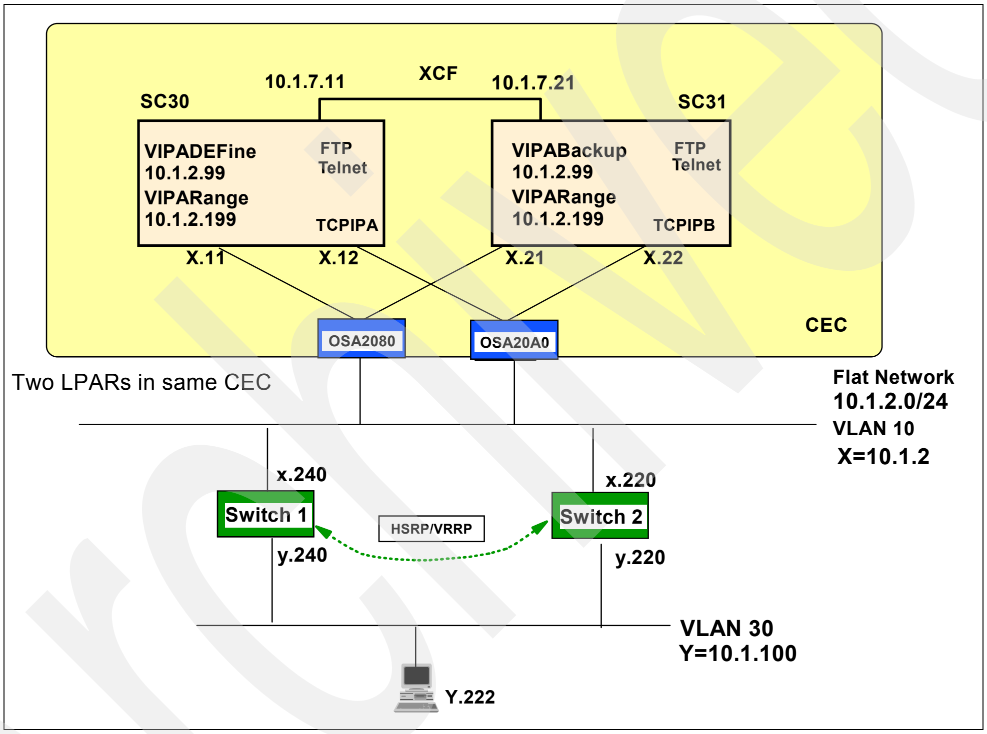


Figure 3-4 Diagram of our working environment

Example 3-8 and Example 3-9 list the relevant statements in our TCP/IP profiles for the two LPARs. The following key items should be noted:

- We configured the two OSA ports to use the same VLAN (VLAN 10) 1.
- We created static route and default route definitions to both OSA ports that belong to the VLANID. By doing this, we had high availability at the interface level. One is a backup of the other 2.
- We defined two DVIPA addresses using VIPADEFine and VIPABackup in the stack 3 and VIPARange at the application level 4.
- We defined the FTP application to `bind()` to a specific address (DVIPA) instead of INADDR_ANY by using the server bind control that is implemented through the BIND keyword on the PORT statement 5.

Example 3-8 TCP/IP profile in SC30

```
IPCONFIG MULTIPATH IQDIOR DATAGRAMFWD
IPCONFIG SOURCEVIPA
DEVICE OSA2080  MPCIPA
LINK   OSA2080L  IPAQENET      OSA2080 VLANID 10 1
DEVICE OSA20A0  MPCIPA
LINK   OSA20A0L  IPAQENET      OSA20A0 VLANID 10 1
HOME
10.1.2.11       OSA2080L
10.1.2.12       OSA20A0L
BEGINROUTES
ROUTE 10.1.2.0       255.255.255.0 =        OSA2080L    mtu defaultsize 2
ROUTE 10.1.2.0       255.255.255.0 =        OSA20A0L    mtu defaultsize
ROUTE DEFAULT         10.1.2.240    OSA2080L  mtu defaultsize
ROUTE DEFAULT         10.1.2.220    OSA20A0L  mtu defaultsize
ENDROUTES
;Test flat network start
VIPADynamic
VIPARange DEFINE MOVEable NONDISRUPTive 255.255.255.255 10.1.2.199 4
VIPADEFINE MOVE IMMED 255.255.255.255 10.1.2.99 3
ENDVIPADYNAMIC
;Test flat network end
;Bind to Dynamic VIPA SC30 and SC31:
PORT
20 TCP *  NOAUTOLOG          ; FTP Server
21 TCP OMVS BIND 10.1.2.199 ; control port 5
```

Example 3-9 shows our TCP/IP profile in SC31.

Example 3-9 TCP/IP profile in SC31

```
IPCONFIG MULTIPATH IQDIOR DATAGRAMFWD
IPCONFIG SOURCEVIPA
DEVICE OSA2080  MPCIPA
LINK   OSA2080L  IPAQENET      OSA2080 VLANID 10 1
DEVICE OSA20A0  MPCIPA
LINK   OSA20A0L  IPAQENET      OSA20A0 VLANID 10 1
HOME
10.1.2.21       OSA2080L
10.1.2.22       OSA20A0L
BEGINROUTES
ROUTE 10.1.2.0       255.255.255.0 =        OSA2080L    mtu defaultsize 2
ROUTE 10.1.2.0       255.255.255.0 =        OSA20A0L    mtu defaultsize
ROUTE DEFAULT         10.1.2.240    OSA2080L  mtu defaultsize
ROUTE DEFAULT         10.1.2.220    OSA20A0L  mtu defaultsize
ENDROUTES
;Test flat network start
VIPADynamic
VIPARange DEFINE MOVEable NONDISRUPTive 255.255.255.255 10.1.2.199 4
VIPABackup 1 MOVEable IMMEDiate 255.255.255.255 10.1.2.99 3
ENDVIPADYNAMIC
;Test flat network end
20 TCP *  NOAUTOLOG          ; FTP Server
21 TCP OMVS BIND 10.1.2.199 ; control port 5
```

We defined our VLAN in both switches, as shown in Example 3-10 and Example 3-11 (6). These two examples also illustrate ping operations between the routers and the OSA ports.

Example 3-10 Switch 1 setup

```
Router(config)#interface vlan 10 6
Router(config-if)#ip address 10.1.2.240 255.255.255.0
Router(config-if)#no shut
Router(config-if)#end

Router# ping 10.1.2.11
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.1.2.11, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Router#wr mem
Building configuration...
```

Example 3-11 illustrates the switch 2 setup.

Example 3-11 Switch 2 setup

```
Router(config)#interface vlan 10 6
Router(config-if)#ip address 10.1.2.220 255.255.255.0
Router(config-if)#no shut
Router(config-if)#end

Router# ping 10.1.2.21
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 10.1.2.21, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
Router#wr mem
Building configuration...
```

3.2.2 Verification

To verify that our network was operational, we first used NETSTAT to check the routing configuration, as shown in Example 3-12.

Example 3-12 Flat network result display routes in normal operation of SC30 and SC31

```
D TCPIP,TCPIPA,N,ROUTE
RESPONSE=SC30
IPV4 DESTINATIONS
DESTINATION         GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT             10.1.2.240       UGS       000001  OSA2080L 1
DEFAULT             10.1.2.220       UGS       000000  OSA20A0L
10.1.2.0/24         0.0.0.0          UZ        000000  OSA2080L
10.1.2.0/24         0.0.0.0          UZ        000000  OSA20A0L
10.1.2.99/32        0.0.0.0          UH        000000  VIPL0A010263
10.1.2.11/32        0.0.0.0          UH        000000  OSA2080L
10.1.2.12/32        0.0.0.0          UH        000000  OSA20A0L
RESPONSE=SC31
IPV4 DESTINATIONS
DESTINATION         GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT             10.1.2.240       UGS       000000  OSA2080L 1
DEFAULT             10.1.2.220       UGS       000000  OSA20A0L
10.1.2.0/24         0.0.0.0          UZ        000000  OSA2080L
10.1.2.0/24        0.0.0.0          UZ        000000  OSA20A0L
10.1.2.99/32       0.0.0.0          UH        000000  VIPL0A010263
10.1.2.21/32       0.0.0.0          UH        000000  OSA20A0L
10.1.2.22/32       0.0.0.0          UH        000000  OSA20C0L
```

Note the two default gateways to routers 10.1.2.240 and 10.1.2.220 (1) in Example 3-12.

A NETSTAT with a VIPADCFG operand provides information about the DVIPA configuration in each stack where the command is issued. The results are illustrated in Example 3-13.

Example 3-13 Display NETSTAT Dvipa config in SC30 and SC31

```
D TCPIP,TCPIPA,N,VIPADCFG
RESPONSE=SC30
DYNAMIC VIPA INFORMATION:
VIPA DEFINE:
IPADDR/PREFIXLEN: 10.1.2.99/32
    MOVEABLE: IMMEDIATE  SRVMGR: NO
VIPA RANGE:
IPADDR/PREFIXLEN: 10.1.2.199/32
    MOVEABLE: NONDISR
RESPONSE=SC31
DYNAMIC VIPA INFORMATION:
   VIPA BACKUP:
IPADDR/PREFIXLEN: 10.1.2.99
```

```
      RANK: 001  MOVEABLE:           SRVMGR:
   VIPA RANGE:
IPADDR/PREFIXLEN: 10.1.2.199/32
MOVEABLE: NONDISR
```

We used two TCP/IP applications, TELNET and FTP. Both were used from a client workstation at address 10.1.100.222. The Telnet client was configured to access the VIPADEFine 1 10.1.2.99. The FTP client was started to access the VIPARange 10.1.2.199 2. This DVIPA address only appears in the home list after the application FTP server is started.

A NETSTAT with a VIPADYN operand provides information about DVIPA status in each stack where the command is issued. The results are illustrated in Example 3-14.

Example 3-14 Display NETSTAT Dvipa status in SC30 and SC31

```
D TCPIP,TCPIPA,N,VIPADYN
RESPONSE=SC30
DYNAMIC VIPA:
IPADDR/PREFIXLEN: 10.1.2.99/32    1
    STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT:
after we start the FTP the VIPARange address appears
LINKNAME: VIPL0A0102C7
IPADDR/PREFIXLEN: 10.1.2.199/32   2
  ORIGIN: VIPARANGE BIND
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPA   SC30     ACTIVE

RESPONSE=SC31
DYNAMIC VIPA:
IPADDR/PREFIXLEN: 10.1.2.99/32
  STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT:
```

A NETSTAT with a CONN operand, shown in Example 3-15, provides connection information.

Example 3-15 Telnet and FTP to DVIPA address in use

```
D TCPIP,TCPIPA,N,CONN
USER ID  CONN     STATE
TCPIPB   00000288 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.2.99..23
  FOREIGN SOCKET: ::FFFF:10.1.100.222..3640
FTPDC1   000000BF ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.2.199..21
  FOREIGN SOCKET: ::FFFF:10.1.100.222..3820
```

3.3 High availability scenarios

The examples that we discuss in this section are not designed to be indicative of typical failures. Instead, they can help you to understand various recovery and transfer processes. We examine the following areas of interest:

- Adapter interface failure
- Application movement
- Stack failure

3.3.1 Adapter interface failure

We stopped the OSA adapter (OSA2080) on our primary system to verify that the connection from the client workstation continued to function during the takeover by the backup system. The failure scenario is illustrated in Figure 3-5.

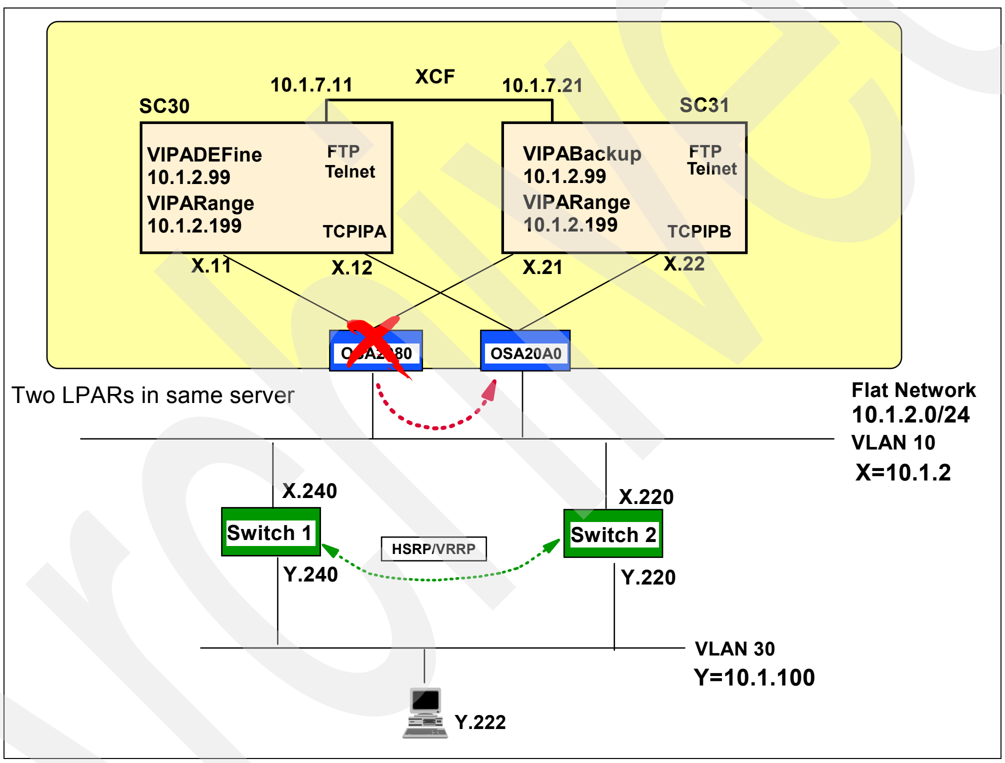


Figure 3-5 High availability for an adapter failure

Our command to stop the primary interface, and the resulting messages, is shown in Example 3-16.

Example 3-16 High availability in fail of interface

```
V TCPIP,TCPIPA,STOP,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,STOP,OSA2080
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4329I LINK OSA20A0L HAS TAKEN OVER ARP RESPONSIBILITY FOR INACTIVE LINK
OSA2080L
```

We observed that the traffic was automatically redirected from the failing interface connection (OSA3) to the backup interface connection (OSA4). This is the ARP takeover function. The takeover operation is described in more detail in 3.1.1, “ARP takeover” on page 47.

Note that this function requires that both LAN adapters involved be accessible to the LPAR running the stack and applications. The FTP and TELNET connections continued to operate in SC30, the primary system. In this scenario, the operational TCP/IP stack and application remain in their original LPAR.

Example 3-17 shows the client user issuing an **ls** command through the FTP session, with the expected results. The adapter takeover was transparent to the client. The result of a NETSTAT command is also shown in Example 3-17.

Example 3-17 FTP and Telnet services keep running OK after interface fail

```
ftp> ls
200 Port request OK.
125 List started OK
BRODCAST
HFS
NTUSER.DAT
NTUSER.DAT.LOG
NTUSER.INI
SC30.ISPF42.ISPPROF
SC30.SPFLOG1.LIST
SC30.SPF1.LIST
TESTE.TXT
TESTE2.TXT
250 List completed successfully.
ftp: 134 bytes received in 0.00Seconds 134000.00Kbytes/sec.

D TCPIP,TCPIPB,N,CONN
USER ID  CONN     STATE
TCPIPB 00000288 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.2.99..23
  FOREIGN SOCKET: ::FFFF:10.1.100.222..3640
FTPDC1   000000BF ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.2.199..21
  FOREIGN SOCKET: ::FFFF:10.1.100.222..3820
```

After observing these results, we restarted the primary OSA interface. This resulted in a takeback function that was transparent to the client.

3.3.2 Application movement

This example illustrates how you can benefit from nondisruptive movement of an application by simply starting another instance of the application in another LPAR that takes over new connections. We defined the FTP services in SC30 and SC31 to be manually started, that is, we did not use *autolog* in the port definition profile. First, we started the FTP server in SC30 and this activated DVIPA address 10.1.2.199. We then connected to the FTP server from the workstation.

When the new instance of the application becomes active, the TCP/IP stack can immediately take over the ownership of the DVIPA while existing connections continue to the original TCP/IP stack (until the connections end or the original application or TCP/IP stack is taken down). This approach can be very useful for planned outage situations (such as software maintenance). In failure situations, if the application supports Automatic Restart Manager (ARM), you can have ARM restart the application automatically. If your application cannot bind to a DVIPA, you can use the `bind()` on the port statement or the MODDVIPA utility. The situation is illustrated in Figure 3-6.

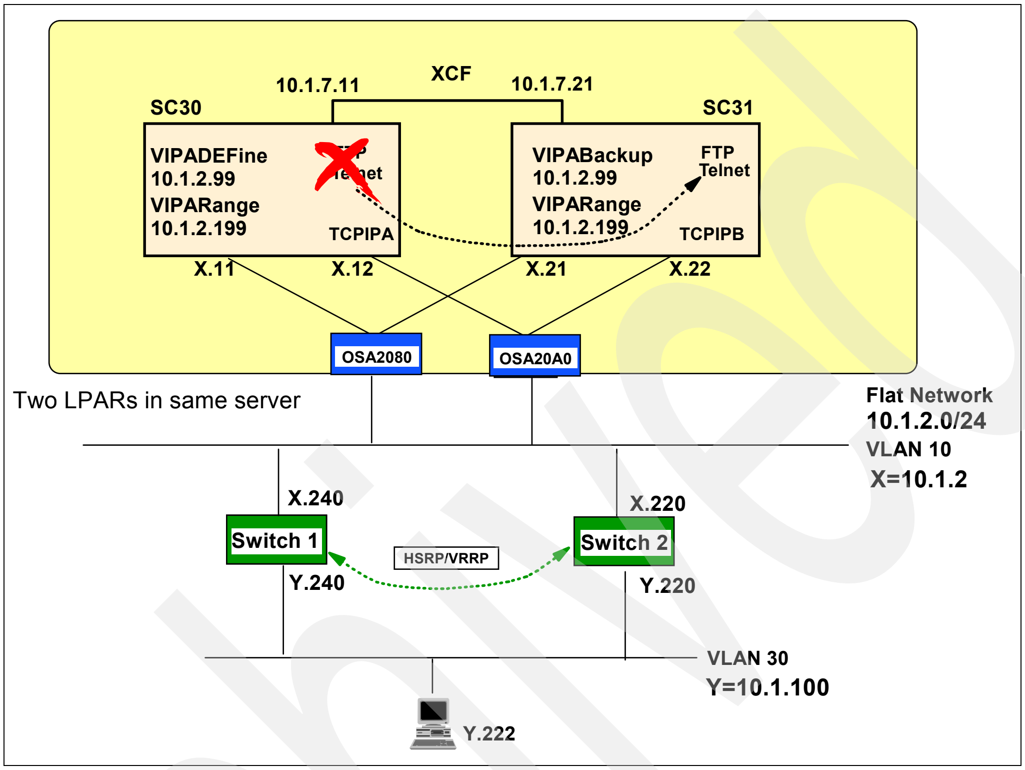


Figure 3-6 High availability for an application failure

We can observe that the DVIPA address defined by VIPARange was redirected from stack SC30 to another stack SC31. This is the DVIPA movement. We can see it in the following examples.

Before starting FTP in SC31, the VIPA address is active in SC30. We can verify this by using NETSTAT with a VIPADYN operand. The results are shown in Example 3-18.

Example 3-18 VIPA active in SC30

```
LINKNAME: VIPL0A0102C7
IPADDR/PREFIXLEN: 10.1.2.199/32
  ORIGIN: VIPARANGE BIND
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPB   SC30     ACTIVE
```

We then started an FTP application in SC31 and observed this stack take over the VIPA address, as shown in Example 3-19.

Example 3-19 The DVIPA address moves to SC31

```
S FTPDA
$HASP100 FTPDA    ON STCINRDR
IEF695I START FTPDA    WITH JOBNAME FTPDA    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 FTPDA    STARTED
EZZ8302I VIPA 10.1.2.199 TAKEN FROM TCPIPA ON SC30
EZZ8303I VIPA 10.1.2.199 GIVEN TO TCPIPB ON SC31
+EZY2702I Server-FTP: Initialization completed at 15:33:05
```

After starting FTP in SC31, the DVIPA status has automatically changed, and the DVIPA address is now active in SC31. Example 3-20 illustrates this new status.

Example 3-20 DVIPA active in SC31

```
IPADDR: 10.1.2.199
  ORIGIN: VIPARANGE BIND
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPB   SC31     ACTIVE
  TCPIPA   SC30     MOVING
```

When a new FTP client now connects, it will connect to SC31. Existing FTP sessions remain with SC30 until they end. At that time the DVIPA address for SC30 disappears.

3.3.3 Stack failure

In this example, we lose the entire TCP/IP stack. (We simulate this by simply stopping the stack.) The failure situation is illustrated in Figure 3-7.

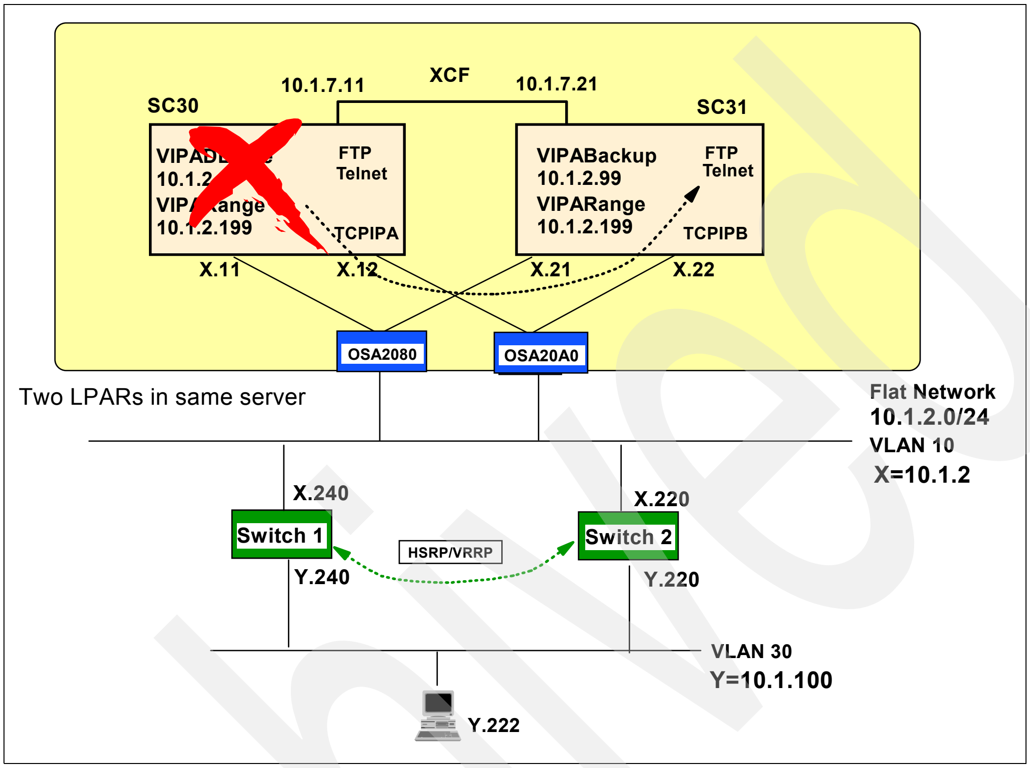


Figure 3-7 High availability for a stack failure

We used VIPADEFINE and VIPABACKUP definitions for the DVIPA to create a takeover and takeback environment. We then stopped the primary TCP/IP (in SC30) and observed the results, as shown in Example 3-21.

Example 3-21 High availability in fail of the stack

```
P TCPIPA
EZZ3205I SNMP SUBAGENT: SHUTDOWN IN PROGRESS
EZZ0673I AUTOLOG TASK: SHUTDOWN IN PROGRESS
EZZ6008I TELNET STOPPING
EZZ8301I VIPA 10.1.2.99 TAKEN OVER FROM TCPIPA ON SC30
RO SC31,D TCPIP,TCPIPB,N,VIPADYN
DYNAMIC VIPA:
IPADDR/PREFIXLEN: 10.1.2.99/32
    STATUS: ACTIVE     ORIGIN: VIPABACKUP       DISTSTAT:
```

This is a disruptive failure mode. Depending on the applications involved, clients might need to restart their sessions. (A subsequent takeback, when the TCP/IP stack in SC30 is restarted, is nondisruptive.)

When system SC31 discovers that SC30 has gone down (when SC30 leaves the XCF group), SC31 will take over the DVIPA address 10.1.2.99 (by advertising the DVIPA address with a gratuitous ARP, as discussed in 3.1.1, “ARP takeover” on page 47). We then started new Telnet connections to the DVIPA address 10.1.2.99, which is now running in SC31. We can use NETSTAT operand N,VCRT to verify that these connections are going to DESTXCF 10.1.7.21 (the Dynamic XCF address of the SC31 stack).

Example 3-22 illustrates this situation.

Example 3-22 New connections go to SC31

```
D TCPIP,TCPIPB,N,VCRT
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.1.2.99..23
  SOURCE:  10.1.100.222..1202
  DESTXCF: 10.1.7.21                1
DEST:      10.130.2.99..23
  SOURCE:  10.1.100.222..1203
  DESTXCF: 10.1.7.21
DEST:      10.1.2.99..23
  SOURCE:  10.1.100.222..3149
  DESTXCF: 10.1.7.21
```

In this example, the number corresponds to the following information:

1. DESTXCF 10.1.7.21 is the Dynamic XCF address of the SC31 stack.

We finally restarted the TCP/IP stack in SC30 and observed the takeback operation. This is shown in Example 3-23.

Example 3-23 Start TCPIPA in SC30 takeback function

```
S TCPIPA
$HASP100 TCPIPA   ON STCINRDR
IEF695I START TCPIPA   WITH JOBNAME TCPIPA   IS ASSIGNED TO USER TCPIP, GROUP
TCPGRP
EZZ8302I VIPA 10.1.2.99 TAKEN FROM TCPIPB ON SC31
EZZ8303I VIPA 10.1.2.99 GIVEN TO TCPIPA ON SC30
```

3.4 Debugging tips

When you are analyzing problems in a flat network, the first step is to verify that the static and default routes are correct. After trying PING and TRACERT commands from a remote host, use NETSTAT ROUTE, NETSTAT GATE, and NETSTAT ARP commands in the z/OS system to verify that the routing parameters are correct. (You can also use UNIX System Services commands such as **onetstat -r**.)

Routing is almost always a two-way function. That is, packets from remote systems must find their way back to z/OS. Problems with *return routing* (back to z/OS) are possibly the most common errors found when implementing TCP/IP in a new host. This return routing is usually under the control of external systems and routers, and there is no easy way to check it from the z/OS end. DVIPA configurations might involve multiple IP addresses for z/OS stacks, and *all* these addresses must be handled correctly by the external network.

If the routing appears correct, then a PKTTRACE in z/OS can be useful. This trace consists of IP packets flowing to and from a TCP/IP stack, and the trace provides a detailed view of local routing and packet flow.

The following additional steps might be helpful:

- Use a **netstat** command with the **vipadcfg** operand to verify the DVIPA definition.
- Use a **netstat** command with the **vipadyn** operand to verify that the DVIPA status is ACTIVE.
- As a last resort, collect a CTRACE with options XCF, INTERNET, TCP, and VTAMDATA. These can be used to:
  - Identify the connection being received by the stack.
  - Determine the stack to which the connection will be forwarded.
  - Verify that the connection is being forwarded.
  - Verify that the expected target stack is receiving and processing the connection.

You can find more debugging information in *z/OS Communications Server: IP Configuration Guide*, SC31-8775.

4

VIPA with dynamic routing

This chapter introduces and provides implementation examples for using dynamic routing in a z/OS Communications Server environment. The dynamic routing protocols supported in z/OS Communications Server are Open Shortest Path First (OSPF) and Routing Information Protocol (RIP), which are implemented in OMPROUTE.

This chapter focuses on OSPF as the recommended dynamic routing protocol for z/OS Communications Server. For more details regarding implementation and concepts of OMPROUTE, refer to *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7800.

This chapter discusses the following topics.

Section	Topic
4.1, “Basic concepts of high availability using dynamic routing” on page 66	Basic concepts of high availability using dynamic routing
4.2, “Design example for dynamic routing” on page 74	Selected implementation scenario tasks, configuration examples and verification steps Problem determination suggestions
4.3, “High availability scenarios” on page 83	Examples of failing scenarios and how they are recovered in a dynamic routing environment

4.1 Basic concepts of high availability using dynamic routing

In its most basic sense, dynamic routing is a process used in an IP network to ensure that the routing tables always reflect the current network topology. Using those routing tables, the routing process is then able to deliver datagrams to the correct destinations. Dynamic routing is implemented in a *routing daemon* in each router. The daemon is responsible for maintaining local routing information, propagating changes to neighbors, and updating its routing table according to changes received from neighbors.

The main goal of designing for high availability is to ensure that users can reach their applications as close to 100% of the time as can be achieved most cost-effectively. An important benefit of using a dynamic routing protocol in z/OS Communications Server is being able to leverage a consistent, TCP/IP-based, end-to-end, approach for dynamically recognizing and responding to changes in the network and z/OS environment.

4.1.1 Dynamic routing and OMPROUTE

OMPROUTE supports both OSPF and RIP as dynamic routing protocols. The main function of both protocols is to exchange routing information with routers in the network.

OSPF is a link-state routing protocol. Every router has a full map of the entire network topology. Changes to the network topology (due to link-state changes or outages) is propagated throughout the network. Every router that receives updates must then recompute its shortest path to all destinations. The convergence time for OSPF is very low and there is virtually no limit to the network design.

RIP (RIPv1 and RIPv2) is a distance vector routing protocol. Every 30 seconds, each router sends its full set of distance vectors to neighboring routers. When each router receives a set of distance vectors, it must recalculate the shortest path to each destination. The convergence time for RIP can be up to 180 seconds and the longest path that can be managed by RIP is 15 hops, which means that a distance of 16 or more hops constitutes an invalid or unreachable destination.

For further details regarding dynamic routing protocols, refer to *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798.

> **Note:** If spanning tree is used in the bridge or switch that is connected to an OSA QDIO port, the Spanning Tree Protocol must be turned ON to allow the routing protocol packets that advertise a VIPA to flow correctly. Without the Spanning Tree Protocol enabled, there is a transition through the Listening and Learning stages (30 seconds in total) of Spanning Tree before the port can actually pass valid traffic.

4.1.2 Advertisement of VIPA addresses

VIPA addresses must be known or recognized by the routing protocol in order to propagate them throughout the network. This means that VIPA addresses (and associated subnet masks) must be defined in all OMPROUTE instances in the sysplex that might own the addresses. Normally, VIPA addresses are defined by the OSPF_Interface statement so that the OSPF protocol will send out Link State Advertisements (LSAs) about them.

When a VIPA address is created in the TCP/IP stack, OMPROUTE receives that information and searches for the OSPF_Interface statement that best matches the VIPA address. OSPF then propagates by default both the VIPA host address and the VIPA IP subnet that belongs to the VIPA address.

Figure 4-1 shows how VIPA addresses are propagated throughout the network. Each VIPA address is represented both by the VIPA address (/32) and by the VIPA subnet (/24). The VIPA address (/32) is more specific than the VIPA subnet (/24), so at first sight the VIPA subnet is never to be used.

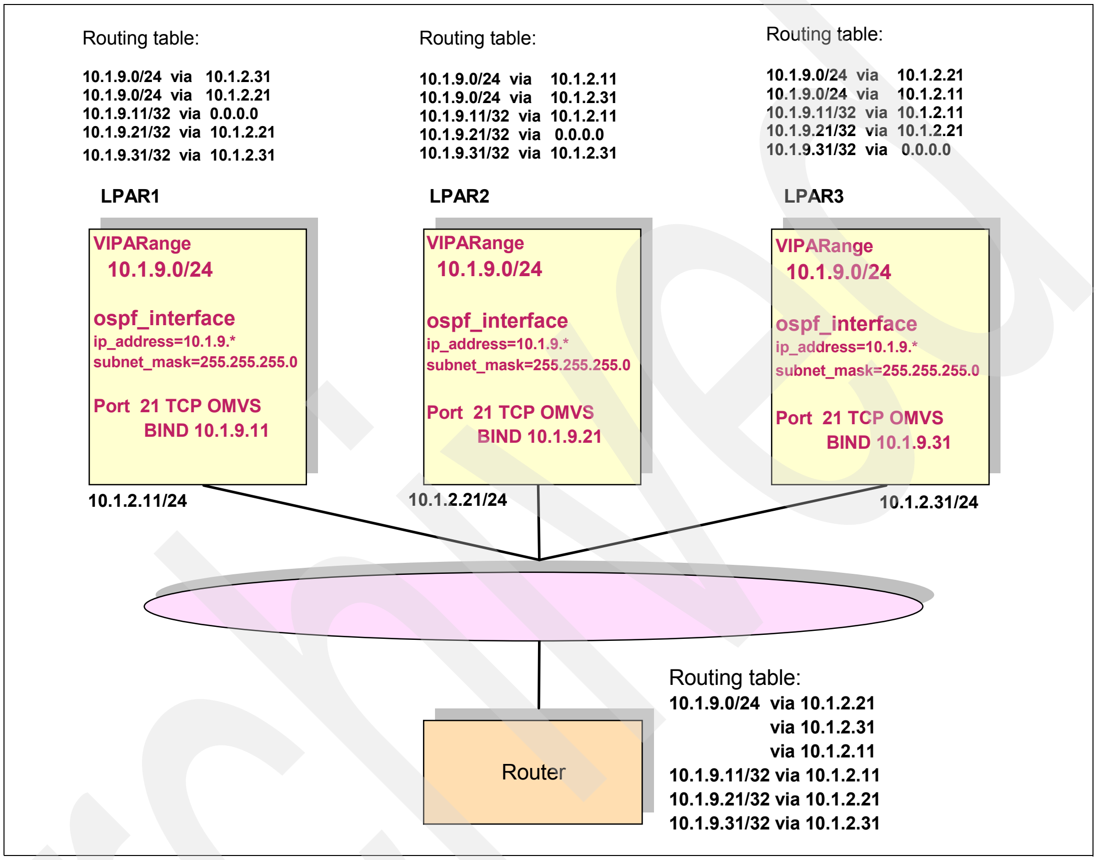


Figure 4-1 VIPA subnet across LPARs

Figure 4-2 shows what happens if one of the FTP servers in this setup is taken down and the VIPA address is deleted.

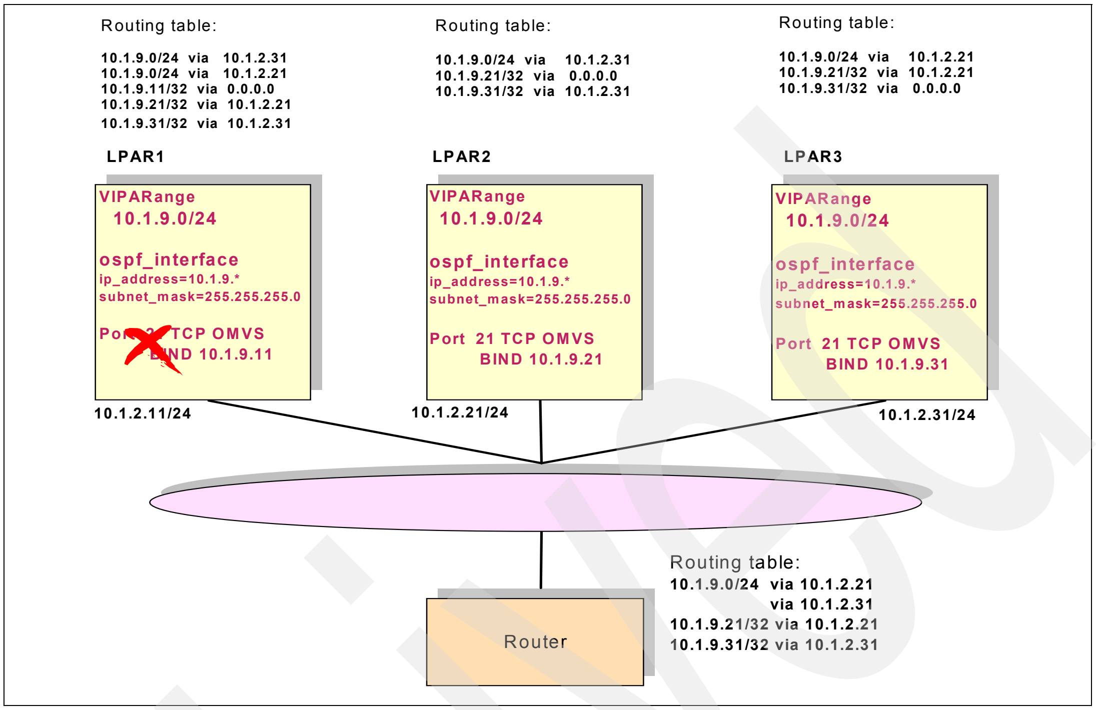


Figure 4-2 VIPA subnet across LPARs after FTP server is taken down in LPAR1

The routing tables now reflect the deletion of VIPA address 10.1.9.11. Both the IP address and the subnet entry have been deleted from all routing tables. The interesting point is that the VIPA subnet entries now come into play.

Consider what happens if an FTP client requests a connection to the FTP server in LPAR 1 (10.1.9.11). The request arrives at the router, but the router does not have a route to 10.1.9.11. The next best match in the router is an equal cost entry (10.1.9.0/24) pointing to LPAR2 and LPAR3. The routing tables in LPAR2 and LPAR3 do not have a route to 10.1.9.11, either. The next best match in LPAR 2 is a route pointing to LPAR3, and the next best match in LPAR 3 is a route pointing to LPAR 2. This looks like a routing loop. The FTP client request continues searching for the IP address until the time-to-live count goes to zero (0) and the packet is discarded.

From an availability point of view, there are two ways to circumvent this situation:

- The first circumvention is to avoid using **IPCONFig DATAGRamfwd**. This prevents datagrams from being forwarded between TCP/IP stacks, but it does not prevent the router from sending the first request to the first LPAR.
- The other circumvention is to avoid advertising the VIPA subnet throughout the network. This will prevent the router from sending the request to any TCP/IP stack not owning the exact VIPA address. You can control this by the Advertise_VIPA_Routes parameter on the OSPF_Interface statement. Our OSPF_Interface statement looks as follows:

```
ospf_interface ip_address=10.1.9.*
           subnet_mask=255.255.255.0
           attaches_to_area=0.0.0.2
           Advertise_VIPA_Routes=HOST_ONLY
           cost0=10
           mtu=65535 ;
```

By not advertising the VIPA subnet, the routing tables now displays as shown in Figure 4-3.

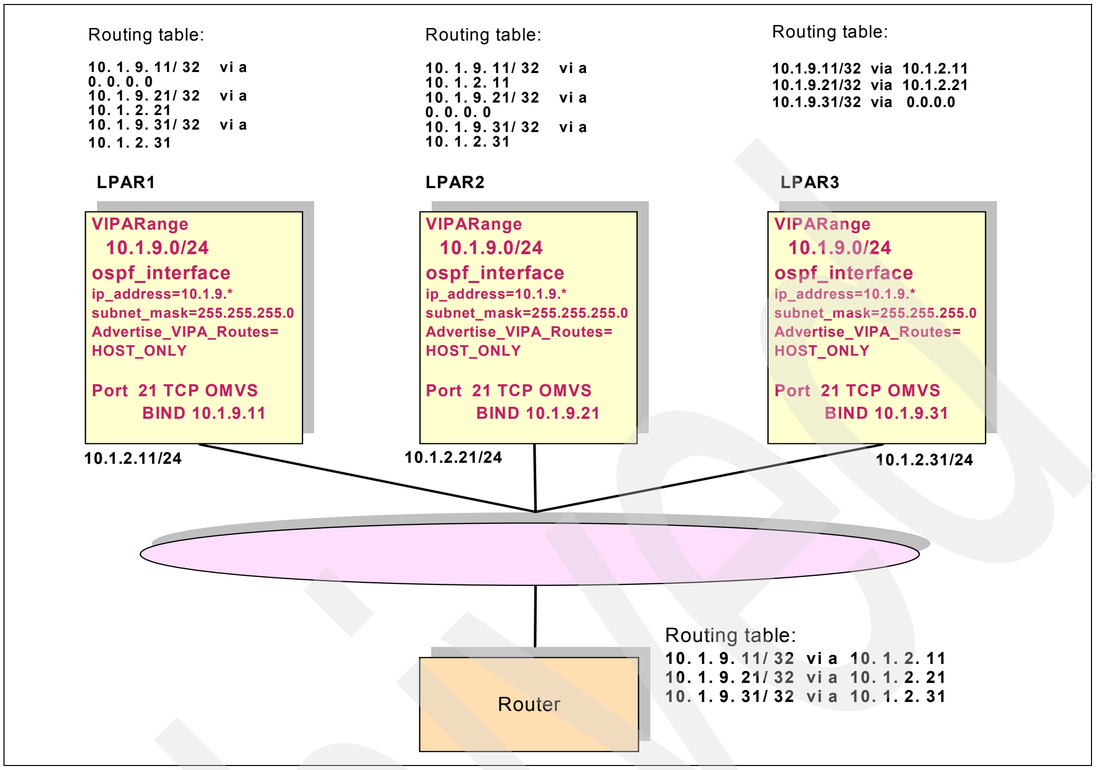


Figure 4-3 VIPA addresses without VIPA subnet

The routing tables are simplified and there is no searching for VIPA addresses not available. Datagrams are forwarded to VIPA addresses in the LPARs only if the specific VIPA address is active and advertised as an IP host address.

4.1.3 Multiple links between IP nodes and LPARs

A system designed for high availability should have more than one link between the various TCP/IP stacks in the system. It should also have more than one router for connections to the external network.

In our test environment, we designed a network with four OSA ports. Two ports connected to one Layer 3 switch (router), VLANs 10 and 11, and the other two ports connected to the second Layer 3 switch (router), also VLANs 10 and 11, thus achieving physical and logical redundancy as shown in Figure 4-4.

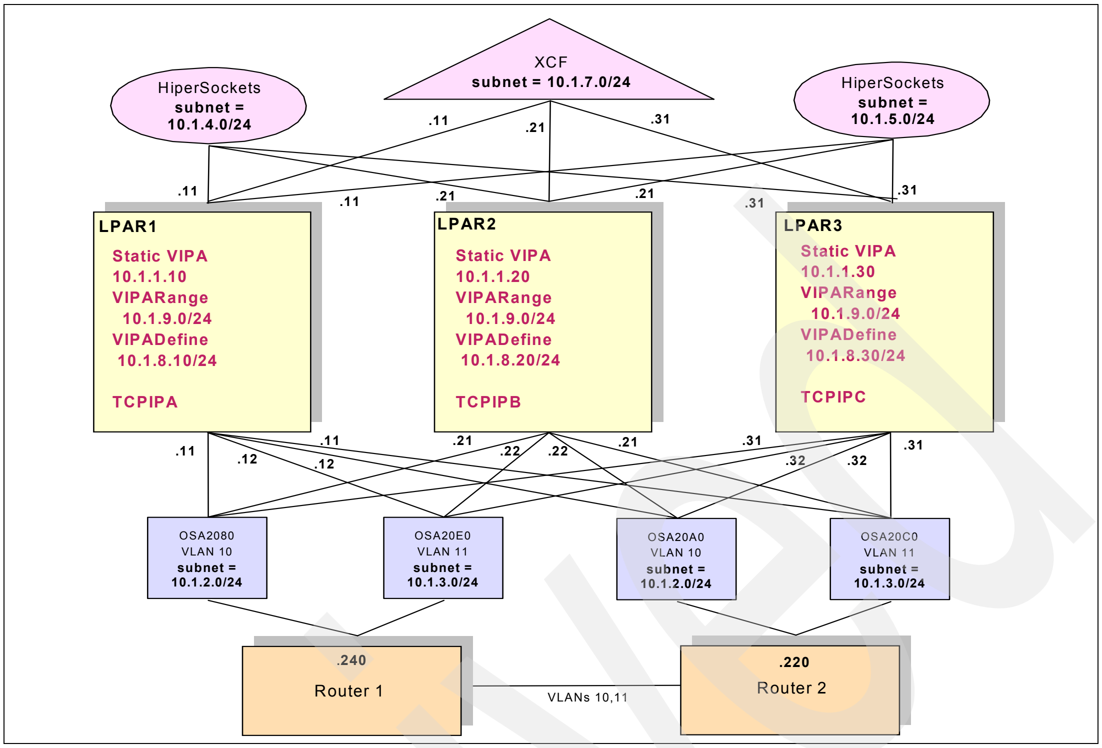


Figure 4-4 Sample network with multiple paths between LPARs

From a routing perspective, by default, all links in this scenario that have the same distance and the same cost are considered equal-cost links, and datagrams will be forwarded in a round-robin fashion according to the IPCONFIG MULTIPATH implementation. In our example, datagrams between LPARs are equally distributed across the XCF, HiperSocket, and OSA links. We can use **netstat at TCPIPA** to display the routing information shown in Example 4-1.

Example 4-1 Routing information displayed using netstat

```
D TCPIP,TCPIPA,N,ROUTE,IPA=10.1.9.*
IPV4 DESTINATIONS
DESTINATION          GATEWAY          FLAGS     REFCNT   INTERFACE
10.1.9.21/32         10.1.4.21        UGHO      000000   IUTIQDF4L
10.1.9.21/32         10.1.5.21        UGHO      000000   IUTIQDF5L
10.1.9.21/32         10.1.7.21        UGHO      000000   IQDIOLNK0A01070B
10.1.9.21/32         10.1.2.22        UGHO      000000   OSA2080L
10.1.9.21/32         10.1.2.21        UGHO      000000   OSA2080L
10.1.9.21/32         10.1.2.22        UGHO      000000   OSA20A0L
10.1.9.21/32         10.1.2.21        UGHO      000000   OSA20A0L
10.1.9.21/32         10.1.3.22        UGHO      000000   OSA20C0L
10.1.9.21/32         10.1.3.21        UGHO      000000   OSA20C0L
10.1.9.21/32         10.1.3.22        UGHO      000000   OSA20E0L
10.1.9.21/32         10.1.3.21        UGHO      000000   OSA20E0L
10.1.9.31/32         10.1.2.32        UGHO      000000   OSA2080L
10.1.9.31/32         10.1.2.31        UGHO      000000   OSA2080L
10.1.9.31/32         10.1.2.32        UGHO      000000   OSA20A0L
10.1.9.31/32         10.1.2.31        UGHO      000000   OSA20A0L
10.1.9.31/32         10.1.3.32        UGHO      000000   OSA20C0L
10.1.9.31/32         10.1.3.31        UGHO      000000   OSA20C0L
10.1.9.31/32         10.1.3.32        UGHO      000000   OSA20E0L
```

```
10.1.9.31/32        10.1.3.31        UGHO       000000  OSA20E0L
10.1.9.31/32        10.1.4.31        UGHO       000000  IUTIQDF4L
10.1.9.31/32        10.1.5.31        UGHO       000000  IUTIQDF5L
10.1.9.31/32        10.1.7.31        UGHO       000000  IQDIOLNK0A01070B
10.1.9.11/32        0.0.0.0          UH         000000  VIPL0A01090B
```

Note that OSPF used all available links with the same cost value, and created routes to destination hosts 10.1.9.21 and 10.1.9.31 using all the equal cost links. As a result there are eleven equivalent ways to reach destination 10.1.9.21; there are also eleven equal cost routes to reach destination 10.1.9.31. There might be situations where we require more control over which links are used. After all, are all these paths truly equivalent routes? Should they really be assigned the same routing cost? Probably not. The links might have different capacities, or might be dedicated for specific traffic, for example. A way to control this is by specifying different OSPF cost values on the OMPROUTE definitions of the OSPF interfaces.

As an example, we want to use the following values:

- OSA as the primary path for traffic between LPARs in different servers and external connections.
- HiperSockets for the primary path between LPARs in the same server
- XCF as the backup and last path available

For the external connections, we also wanted, just as an example, to define the OSA connections with a different cost to provide a preferable path. We achieve this goal by giving XCF the highest cost, OSA a medium cost, and HiperSockets the lowest cost, as shown in Table 4-1.

Table 4-1 Overview of the prioritization of available paths

Path used for communication between TCP/IP stacks	First	Second	Third
Same LPAR	HiperSockets	OSA	XCF
Same server	HiperSockets	OSA	XCF
Across servers	OSA (using VLAN11)	OSA (using VLAN10)	XCF
External connections	OSA (using VLAN11)	OSA (using VLAN10)	

Installation requirements differ, of course, and this example is intended only to illustrate a general technique. We defined the **cost0** parameter of each OSPF_Interface statement in our OSPF configuration as shown in Example 4-2.

Example 4-2 Sample of OSPF_Interface configuration

```
; OSA Qdio 10.1.2.x
ospf_interface ip_address=10.1.2.* 1
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=100
           mtu=1492
;
; OSA Qdio 10.1.3.x
ospf_interface ip_address=10.1.3.* 2
           subnet_mask=255.255.255.0
```

```
            ROUTER_PRIORITY=0
            attaches_to_area=0.0.0.2
            cost0=90
            mtu=1492
;
; Hipersockets 10.1.4.x
ospf_interface ip_address=10.1.4.* 3
            subnet_mask=255.255.255.0
            ROUTER_PRIORITY=1
            attaches_to_area=0.0.0.2
            cost0=80
            mtu=8192
;
; Hipersockets 10.1.5.x
ospf_interface ip_address=10.1.5.* 3
            subnet_mask=255.255.255.0
            ROUTER_PRIORITY=1
            attaches_to_area=0.0.0.2
            cost0=80
            mtu=8192
;
; Dynamic XCF
 ospf_interface ip_address=10.1.7.11 4
             name=IQDIOLNK0A01070B
             subnet_mask=255.255.255.0
             ROUTER_PRIORITY=1
             attaches_to_area=0.0.0.2
             cost0=110
             mtu=65535;
```

In this example, the numbers correspond to the following information:

1. The OSA links in VLAN 10 (10.1.2.0/24) with a cost of 100
2. The OSA links in VLAN 11 (10.1.3.0/24) with a cost of 90
3. The HiperSockets links (10.1.4.0/24 and 10.1.5.0/24) with a cost of 80
4. The XCF link (10.1.7.0/24) with a cost of 110

The lowest cost interfaces in this configuration are now reduced to the HiperSockets interfaces with their cost of 80. After the cost changes are in effect, OSPF sends only the lowest cost routes to the TCP/IP stack's routing table. We verify this by using the Netstat ROUTE command, as shown in Example 4-3.

Example 4-3 SC30 Netstat Route display command

```
D TCPIP,TCPIPA,N,ROUTE,IPA=10.1.9.*
IPV4 DESTINATIONS
DESTINATION         GATEWAY          FLAGS     REFCNT   INTERFACE
10.1.9.11/32        0.0.0.0          UH        000000   VIPL0A01090B
10.1.9.21/32        10.1.4.21        UGHO      000000   IUTIQDF4L 1
10.1.9.21/32        10.1.5.21        UGHO      000000   IUTIQDF5L 1
10.1.9.31/32        10.1.4.31        UGHO      000000   IUTIQDF4L 1
10.1.9.31/32        10.1.5.31        UGHO      000000   IUTIQDF5L 1
5 OF 5 RECORDS DISPLAYED
END OF THE REPORT
```

In this example, the number corresponds to the following information:

1. HiperSockets is the preferred path towards VIPA 10.1.9.21 and VIPA 10.1.9.31

If the HiperSockets path becomes unavailable, this affects the network topology and causes OSPF to recalculate the routing table. We disabled the HiperSockets paths (see Example 4-4).

Example 4-4 Stopping HiperSockets devices (only IUTIQDF4 shown)

```
V TCPIP,TCPIPA,STOP,IUTIQDF4
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,STOP,IUTIQDF4
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE IUTIQDF4
EZZ7921I OSPF ADJACENCY FAILURE, NEIGHBOR 10.1.4.21, OLD STATE 128, NEW STATE 1,
EVENT 11
EZZ7921I OSPF ADJACENCY FAILURE, NEIGHBOR 10.1.4.31, OLD STATE 128, NEW STATE 1,
EVENT 11
```

We expect the routes to be recomputed and to favor the next lowest cost interfaces, which are the OSA interfaces that are attached to VLAN 11. These interfaces were assigned a cost of 90. The routing table now looked as shown in Example 4-5.

Example 4-5 D TCPIP,TCPIPA,ROUTE,IPA=10.1.9. command*

```
D TCPIP,TCPIPA,N,ROUTE,IPA=10.1.9.*
IPV4 DESTINATIONS
DESTINATION        GATEWAY          FLAGS     REFCNT  INTERFACE
10.1.9.11/32       0.0.0.0          UH        000000  VIPL0A01090B
10.1.9.21/32       10.1.3.22        UGHO      000000  OSA20C0L 1
10.1.9.21/32       10.1.3.21        UGHO      000000  OSA20C0L 1
10.1.9.21/32       10.1.3.22        UGHO      000000  OSA20E0L 1
10.1.9.21/32       10.1.3.21        UGHO      000000  OSA20E0L 1
10.1.9.31/32       10.1.3.32        UGHO      000000  OSA20C0L 1
10.1.9.31/32       10.1.3.31        UGHO      000000  OSA20C0L 1
10.1.9.31/32       10.1.3.32        UGHO      000000  OSA20E0L 1
10.1.9.31/32       10.1.3.31        UGHO      000000  OSA20E0L 1
9 OF 9 RECORDS DISPLAYED
END OF THE REPORT
```

Now the routing paths **1** have converged on the OSA links with less cost in our configuration, OSA20C0L and OSA20E0L. These interfaces are now the preferred interfaces to VIPA addresses 10.1.9.21 and 10.1.9.31. There are two OSA adapters available, and datagrams will be forwarded round-robin because we assigned each the same cost.

The examples depicted in this section show how to use dynamic routing to provide several levels of backup automatically for IP traffic within a single server. The only significant cost involved is the time to plan and to test the required definitions.

> **Note:** To ensure availability of your environment, it is crucial that your OSPF definitions match the intention of your IP infrastructure; otherwise, you can experience unpredictable results.

4.2 Design example for dynamic routing

The best way to use dynamic routing in the z/OS environment is through OSPF, with the z/OS Communications Server environment defined as an OSPF *stub area* or (even better) a *totally stubby area*.

Stub areas minimize storage and CPU utilization at the nodes that are part of the stub area because less knowledge is maintained about the topology of the Autonomous System (AS) compared to being a full node in an OSPF area or OSPF backbone. Note the following points:

- A stub area system maintains knowledge only about intra-area destinations, summaries of inter-area destinations, and default routes needed to reach external destinations.
- A totally stubby area system receives even less routing information than a stub area. It maintains knowledge only about intra-area destinations and default routes in order to reach external destinations.

z/OS Communications Server typically connects to the IP network through routers that act as gateways. A totally stubby area is a good solution because all it needs is a default route pointing to the gateways.

In this section we show the relevant steps to implement an OSPF totally stubby Area using OMPROUTE. For more information about the implementation of OMPROUTE, refer to *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798.

Figure 4-5 illustrates a totally stubby area setup.

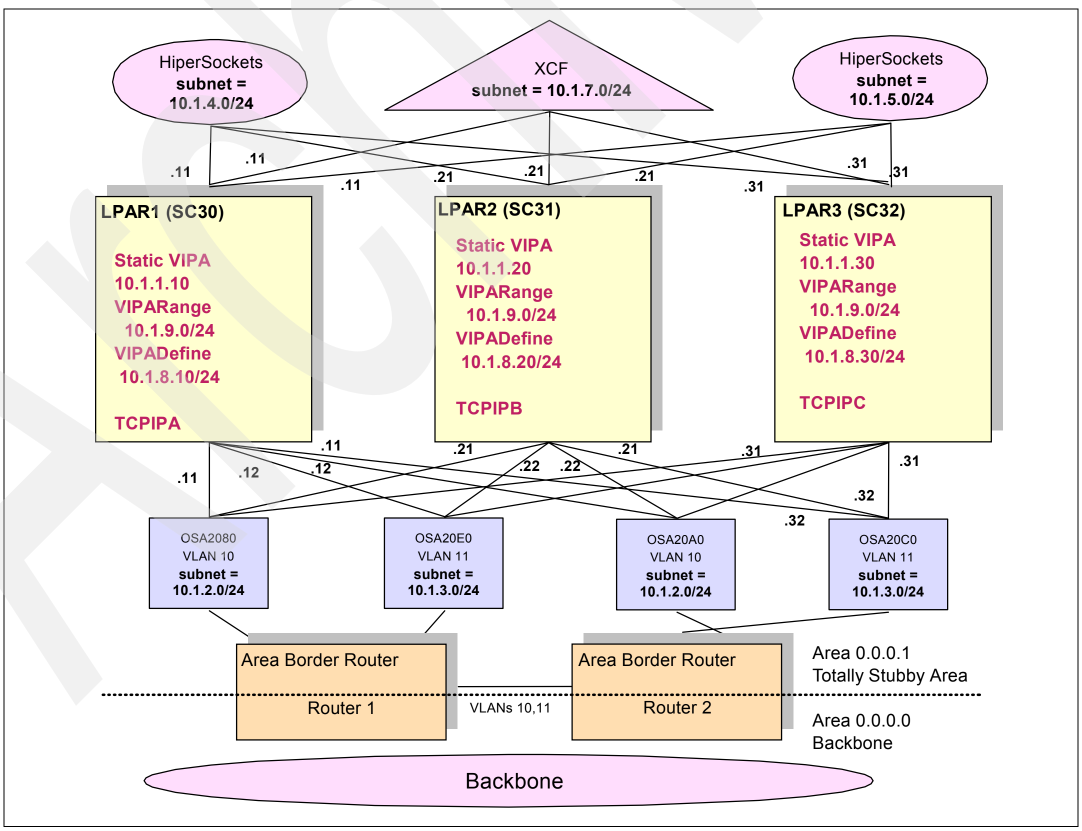


Figure 4-5 Sample network with z/OS being defined as a totally stubby area

In this example, XCF is part of the OSPF design. We could omit XCF as an alternate path for our IP traffic by not configuring XCF links as an OSPF_Interface and by using interface statements instead.

You need to perform the following tasks to implement sysplex environment as a totally stubby area:

1. Define PROFILE statements for the stacks.
2. Create OMPROUTE configuration files.
3. Create definitions for the border routers.
4. Verify that the configuration works as planned.

4.2.1 PROFILE statements

Example 4-6 lists the relevant TCPIP PROFILE statements for system SC30.

Example 4-6 TCP/IP profile for SC30

```
ARPAGE 20
;
GLOBALCONFIG NOTCPIPSTATISTICS
;
IPCONFIG DATAGRAMFWD SYSPLEXROUTING
DYNAMICXCF 10.1.7.11 255.255.255.0 1
...
;
;OSA DEFINITIONS
DEVICE OSA2080    MPCIPA
LINK   OSA2080L   IPAQENET    OSA2080 VLANID 10
DEVICE OSA20C0    MPCIPA
LINK   OSA20C0L   IPAQENET    OSA20C0 VLANID 11
DEVICE OSA20E0    MPCIPA
LINK   OSA20E0L   IPAQENET    OSA20E0 VLANID 11
DEVICE OSA20A0    MPCIPA
LINK   OSA20A0L   IPAQENET    OSA20A0 VLANID 10
;
;HIPERSOCKETS DEFINITIONS
DEVICE IUTIQDF4   MPCIPA
LINK   IUTIQDF4L  IPAQIDIO    IUTIQDF4
DEVICE IUTIQDF5   MPCIPA
LINK   IUTIQDF5L  IPAQIDIO    IUTIQDF5
;
;STATIC VIPA DEFINITIONS
DEVICE VIPA1      VIRTUAL 0
LINK   VIPA1L     VIRTUAL 0   VIPA1
;
;DYNAMIC VIPA DEFINITIONS
VIPADYNAMIC
VIPADEFINE MOVEABLE IMMEDIATE 255.255.255.0 10.1.8.10
;
VIPARANGE 255.255.255.0 10.1.9.0
ENDVIPADYNAMIC
;
HOME
   10.1.1.10     VIPA1L
   10.1.2.11     OSA2080L
```

```
   10.1.3.11     OSA20C0L
   10.1.3.12     OSA20E0L
   10.1.2.12     OSA20A0L
   10.1.4.11     IUTIQDF4L
   10.1.5.11     IUTIQDF5L
;
 PRIMARYINTERFACE VIPA1L
AUTOLOG 5
   FTPDA JOBNAME FTPDA1
   OMPA
ENDAUTOLOG
;
PORT
    20 TCP *  NOAUTOLOG          ; FTP Server
    21 TCP OMVS BIND 10.1.9.11 ; control port
;
SACONFIG ENABLED COMMUNITY public AGENT 161
;
START OSA2080
START OSA20C0
START OSA20E0
START OSA20A0
START IUTIQDF4
START IUTIQDF5
```

Table 4-2 summarizes the differences between the TCPIP PROFILE for SC30 (shown in Example 4-6) and the profiles for SC31 and SC32.

Table 4-2 Summary of TCPIP PROFILE differences between SC30, SC31, and SC32

Statement	SC30 - TCPIPA	SC31 - TCPIPB	SC32 TCPIPC
DYNAMICXCF	10.1.7.11	10.1.7.21	10.1.7.31
VIPADEFine	10.1.8.10	10.1.8.20	10.1.8.30
HOME VIPA1L OSA2080L OSA20C0L OSA20E0L OSA20A0L IUTIQDF4L IUTIQDF5L	 10.1.1.10 10.1.2.11 10.1.3.11 10.1.3.12 10.1.2.12 10.1.4.11 10.1.5.11	 10.1.1.20 10.1.2.21 10.1.3.21 10.1.3.22 10.1.2.22 10.1.4.21 10.1.5.21	 10.1.1.30 10.1.2.31 10.1.3.31 10.1.3.32 10.1.2.32 10.1.4.31 10.1.5.31
PORT 21 TCP OMVS BIND	VIPARANGE 10.1.9.0/24 10.1.9.11	VIPARANGE 10.1.9.0/24 10.1.9.21	VIPARANGE 10.1.9.0/24 10.1.9.31

4.2.2 OMPROUTE definitions

Example 4-7 shows the OMPROUTE configuration file for SC30.

Example 4-7 OMPROUTE configuration file for SC30

```
Area Area_Number=0.0.0.2
   Stub_Area=YES 1
   Authentication_type=None
   Import_Summaries=No ; 2
;
OSPF
 RouterID=10.1.1.10 A
 Comparison=Type2
 Demand_Circuit=YES;
Global_Options
 Ignore_Undefined_Interfaces=YES
;
; Static vipa
ospf_interface ip_address=10.1.1.10 3a
           name=VIPA1L 3b
           subnet_mask=255.255.255.0
           Advertise_VIPA_Routes=HOST_ONLY
           attaches_to_area=0.0.0.2
           cost0=10
           mtu=65535
; OSA Qdio 10.1.2.x
ospf_interface ip_address=10.1.2.* 4
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=100
           mtu=1492
; OSA Qdio 10.1.3.x
ospf_interface ip_address=10.1.3.* 5
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=0
           attaches_to_area=0.0.0.2
           cost0=90
           mtu=1492
; Hipersockets 10.1.4.x
ospf_interface ip_address=10.1.4.* 6
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=1
           attaches_to_area=0.0.0.2
           cost0=80
           mtu=8192
; Hipersockets 10.1.5.x
ospf_interface ip_address=10.1.5.* 6
           subnet_mask=255.255.255.0
           ROUTER_PRIORITY=1
           attaches_to_area=0.0.0.2
           cost0=80
           mtu=8192
; Dynamic XCF
ospf_interface ip_address=10.1.7.11 7a
```

```
          name=IQDIOLNKOA01070B 7b
          subnet_mask=255.255.255.0
          ROUTER_PRIORITY=1
          attaches_to_area=0.0.0.2
          cost0=110
          mtu=65535;
; Dynamic vipa VIPADEFINE
ospf_interface ip_address=10.1.8.* 8
          subnet_mask=255.255.255.0
          Advertise_VIPA_Routes=HOST_ONLY
          attaches_to_area=0.0.0.2
          cost0=10
          mtu=65535
; Dynamic vipa VIPARANGE
ospf_interface ip_address=10.1.9.* 9
          subnet_mask=255.255.255.0
          attaches_to_area=0.0.0.2
          Advertise_VIPA_Routes=HOST_ONLY
          cost0=10
          mtu=65535
```

These definitions include the following key elements:

A Note how we followed best practices and assigned a static VIPA as the OSPF router ID.

1. Indicates this is an OSPF Stub Area.

2. No summaries indicates a totally stubby area.

   a. The entry here on z/OS is in our scenario for documentation purposes only. In fact, the Area Border Router (ABR) decides whether an area is totally stubby. Therefore, the definition in the Area Border Router enforces the totally stubby nature of the connections to z/OS. See 4.2.3, “Router definitions” on page 79 for an example of the router coding.

3. Indicates a static VIPA.

   a. We assigned the full IP address—all four octets—in this definition, which requires us to code the link name of the interface.

   b. We specify the full link name as it is known to the IP stack; this name is found in the stack’s HOME list.

4. Indicates an OSA adapter in VLAN 10.

5. Indicates an OSA adapter in VLAN 11.

6. Indicates a HiperSockets link.

7. Indicates an XCF link.

   a. We assigned the full IP address—all four octets—in this definition, which requires us to code the link name of the interface.

   b. We specify the full link name as it is known to the IP stack; this name is found in the stack’s HOME list.

8. DVIPA - VIPADefine/VIPABackup.

9. DVIPA - VIPARange.

Table 4-3 summarizes the differences between the OMPROUTE configurations for SC30 (shown previously) and those of SC31 and SC32.

Table 4-3 Summary of OMPROUTE configuration differences between SC30, SC31, and SC32

Statement	SC30	SC31	SC32
OSPF RouterID	10.1.1.10	10.1.1.20	10.1.1.30
OSPF_Interface IP_address (Static VIPA)	10.1.1.10	10.1.1.20	10.1.1.30
OSPF_Interface IP_address (XCF)	10.1.7.11	10.1.7.21	10.1.7.31

4.2.3 Router definitions

The router configurations are critical. This section describes the relevant definitions for Router 1. We created an interface for VLAN 10 to include it in the OSPF service configuration, as shown in Example 4-8.

Example 4-8 Configuration of VLAN10 in Router 1

```
interface Vlan10
 ip address 10.1.2.240 255.255.255.0
 ip policy route-map pbr-to-csm
 ip ospf cost 100
 ip ospf priority 100
```

We created an interface for VLAN 11 to include it also in the OSPF service configuration, as shown in Example 4-9.

Example 4-9 Configuration of VLAN11 in Router 1

```
interface Vlan11
 ip address 10.1.3.240 255.255.255.0
 ip policy route-map pbr-to-csm
 ip ospf cost 100
 ip ospf priority 100
```

The physical interfaces that are related to the OSA connections are configured as *Trunk*, so they are not seen as part of this routing configuration.

After we defined the interfaces that participate in the OSPF environment, we defined the OSPF service, as shown in Example 4-10.

Example 4-10 OSPF configuration in Router 1

```
router ospf 100
router-id 10.1.3.240
log-adjacency-changes
area 2 stub no-summary 1
network 10.1.2.0 0.0.0.255 area 2
network 10.1.3.0 0.0.0.255 area 2
network 10.1.0.0 0.0.255.255 area 0
network 10.200.1.0 0.0.0.255 area 0
default-information originate always metric-type 1
```

In this example, the number corresponds to the following information:

1. The `no-summary` parameter indicates that this is a totally stubby area. As shown in Example 4-7 on page 77, the z/OS coding of `Import_Summaries=No` does not influence the behavior of the connections to this router, because z/OS is not acting as an Area Border Router.

The definitions are the same for Router 2.

4.2.4 Verification

To verify that the configuration works as planned, we executed the following steps:

1. List the OSPF configuration in use by OMPROUTE in SC30.
2. List the OSPF neighbors for each of the three systems.
3. List the routing tables in each of the three systems.
4. List the OSPF neighbors for the two routers.
5. List the routing table in the routers.

For details about other commands that you can use to obtain more information about the routing environment, refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

> **Note:** The route table that we show here is only related to the Dynamic VIPA subnet, as an example. All other VIPA subnets have the same routes.

List OSPF configurations

To verify that we had implemented the correct configuration, we used the **`Display OMPRoute,OSPF, List, ALL`** command. We executed the command on SC30. The resulting output is shown in Example 4-11.

Example 4-11 OSPF global configuration in SC30

```
D TCPIP,TCPIPC,OMPR,OSPF,LIST,ALL
EZZ7831I GLOBAL CONFIGURATION 191
    TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
    STACK AFFINITY:           TCPIPA
    OSPF PROTOCOL:            ENABLED
    EXTERNAL COMPARISON:      TYPE 2
    AS BOUNDARY CAPABILITY:   DISABLED
    DEMAND CIRCUITS:          ENABLED

EZZ7832I AREA CONFIGURATION  1
AREA ID           AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
0.0.0.2           0=NONE           YES         1             NO
0.0.0.0           0=NONE           NO          N/A           N/A

EZZ7833I INTERFACE CONFIGURATION     2
IP ADDRESS     AREA               COST   RTRNS   TRDLY   PRI    HELLO   DEAD
DB_E*
10.1.9.11     0.0.0.2       10        N/A      N/A     N/A    N/A      N/A
N/A
10.1.8.10   0.0.0.2    10    N/A   N/A   N/A  N/A   N/A    N/A
10.1.8.20   0.0.0.2    10    N/A   N/A   N/A  N/A   N/A    N/A
10.1.8.30   0.0.0.2    10    N/A   N/A   N/A  N/A   N/A    N/A
10.1.8.40     0.0.0.2       10        N/A      N/A      N/A    N/A      N/A
N/A
```

```
10.1.5.11      0.0.0.2        80      5         1        1        10        40
40
10.1.4.11      0.0.0.2        80      5         1        1        10        40
40
10.1.2.12      0.0.0.2      100      5         1        0        10        40
40
10.1.3.12      0.0.0.2        90      5         1        0        10        40
40
10.1.3.11      0.0.0.2        90      5         1        0        10        40
40
10.1.2.11      0.0.0.2      100      5         1        0        10        40
40
10.1.7.11      0.0.0.2      110      5         1        1        10        40
40
10.1.1.10       0.0.0.2         10      N/A       N/A     N/A    N/A     N/A
N/A

              ADVERTISED VIPA ROUTES 3
 10.1.9.11       /255.255.255.255   10.1.8.40       /255.255.255.255
 10.1.8.30       /255.255.255.255   10.1.8.20       /255.255.255.255
 10.1.8.10       /255.255.255.255   10.1.1.10       /255.255.255.255
```

The following are key elements of the OSPF definitions:

1. The OSPF stub area with no summaries (totally stubby area).

2. The OSPF cost values reflect the configuration (wildcards).

3. The VIPA routes being advertised.

List OSPF neighbors

To verify that SC30 is part of the OSPF Area we implemented, we used the Display OMPR,OSPF,NBRS command; the resulting output is shown in Example 4-12.

Example 4-12 OSPF neighbors as seen from SC30

```
D TCPIP,TCPIPA,OMPR,OSPF,NBRS
RESPONSE=SC30
 EZZ7851I NEIGHBOR SUMMARY 427
 NEIGHBOR ADDR   NEIGHBOR ID     STATE  LSRXL DBSUM LSREQ HSUP IFC
 10.1.5.31       10.1.1.30        128      0     0     0  OFF IUTIQDF5L
 10.1.5.21       10.1.1.20        128      0     0     0  OFF IUTIQDF5L
 10.1.4.31       10.1.1.30        128      0     0     0  OFF IUTIQDF4L
 10.1.4.21       10.1.1.20        128      0     0     0  OFF IUTIQDF4L
 10.1.2.32       10.1.1.30          8      0     0     0  OFF OSA20A0L
 10.1.2.22       10.1.1.20          8      0     0     0  OFF OSA20A0L
 10.1.2.240      10.1.3.240       128      0     0     0  OFF OSA20A0L
 10.1.2.220      10.1.3.220       128      0     0     0  OFF OSA20A0L
 10.1.3.32       10.1.1.30          8      0     0     0  OFF OSA20E0L
 10.1.3.22       10.1.1.20          8      0     0     0  OFF OSA20E0L
 10.1.3.240      10.1.3.240       128      0     0     0  OFF OSA20E0L
 10.1.3.220      10.1.3.220       128      0     0     0  OFF OSA20E0L
 10.1.7.31       10.1.1.30        128      0     0     0  OFF IQDIOLNK*
 10.1.7.21       10.1.1.20        128      0     0     0  OFF IQDIOLNK*
 * -- LINK NAME TRUNCATED
```

List routing tables

To verify that the expected routing table has been created, we used the Netstat,ROUTE command to list the generated routes for subnet 10.1.9.0 and the resulting output for SC30 is shown in Example 4-13.

Example 4-13 IP route table for subnet 10.1.9. in SC30*

```
D TCPIP,TCPIPA,N,ROUTE,IPA=10.1.9.*
IPV4 DESTINATIONS
DESTINATION         GATEWAY          FLAGS     REFCNT   INTERFACE
10.1.9.11/32        0.0.0.0          UH        000000   VIPL0A01090B
10.1.9.21/32        10.1.4.21        UGHO      000000   IUTIQDF4L
10.1.9.21/32        10.1.5.21        UGHO      000000   IUTIQDF5L
10.1.9.31/32        10.1.4.31        UGHO      000000   IUTIQDF4L
10.1.9.31/32        10.1.5.31        UGHO      000000   IUTIQDF5L
5 OF 5 RECORDS DISPLAYED
END OF THE REPORT
```

Notice the VIPA addresses are advertised only as host address (/32) due to the `Advertise_VIPA_Routes=HOST_ONLY` parameter. It is also important to notice the routes are created using the links that we defined with the lowest cost (the HiperSockets links).

List neighbors for routers

Example 4-14 lists the neighbors in Router 1.

Example 4-14 Display of OSPF neighbors in Router 1

```
Router1#sh ip ospf 100 neighbor

Neighbor ID      Pri   State           Dead Time   Address         Interface
10.1.1.10          0   FULL/DROTHER    00:00:31    10.1.2.12       Vlan10 3
10.1.1.20          0   FULL/DROTHER    00:00:36    10.1.2.22       Vlan10 3
10.1.1.30          0   FULL/DROTHER    00:00:39    10.1.2.32       Vlan10 3
10.1.3.220       101   FULL/DR         00:00:31    10.1.2.220      Vlan10 1
10.1.1.10          0   FULL/DROTHER    00:00:30    10.1.3.12       Vlan11 4
10.1.1.20          0   FULL/DROTHER    00:00:35    10.1.3.22       Vlan11 4
10.1.1.30          0   FULL/DROTHER    00:00:38    10.1.3.32       Vlan11 4
10.1.3.220       101   FULL/DR         00:00:31    10.1.3.220      Vlan11 2
```

Key points include the following:

1. Router 2 is the Designated Router for Subnet 10.1.2.0.
2. Router 2 is the Designated Router for Subnet 10.1.3.0.
3. All other neighbors in subnet 10.1.2.0 cannot act as Designated Router (Pri=0).
4. All other neighbors in subnet 10.1.2.0 cannot act as Designated Router (Pri=0).

Routing tables in routers

Example 4-15 shows the routing tables in Router 1.

Example 4-15 Display of IP route table in Router 1

```
Router1#sh ip route ospf 100
     10.0.0.0/8 is variably subnetted, 21 subnets, 2 masks
O   1   10.1.9.11/32 [110/110] via 10.1.2.12, 00:00:29, Vlan10
                     [110/110] via 10.1.2.11, 00:00:29, Vlan10
                     [110/110] via 10.1.3.12, 00:00:29, Vlan11
                     [110/110] via 10.1.3.11, 00:00:29, Vlan11
O   1   10.1.9.21/32 [110/110] via 10.1.2.22, 00:00:50, Vlan10
                     [110/110] via 10.1.2.21, 00:00:50, Vlan10
                     [110/110] via 10.1.3.22, 00:00:50, Vlan11
                     [110/110] via 10.1.3.21, 00:00:50, Vlan11
O   1   10.1.9.31/32 [110/110] via 10.1.2.32, 00:00:00, Vlan10
                     [110/110] via 10.1.2.31, 00:00:00, Vlan10
                     [110/110] via 10.1.3.32, 00:00:00, Vlan11
                     [110/110] via 10.1.3.31, 00:00:00, Vlan11
```

Note that 1 VIPA addresses are advertised as host address (/32) due to the Advertise_VIPA_Routes=HOST_ONLY parameter.

4.3 High availability scenarios

In this section, we describe how some failing scenarios are recovered using a dynamic routing environment. The examples discussed here are the same as used in Chapter 3, “VIPA without dynamic routing” on page 45 and are not designed to be indicative of typical failures. Instead, they help you to understand various recovery and transfer processes. Our scenarios cover the following situations:

- Adapter interface failure
- Application movement using VIPADEFINE
- Stack failure scenario using VIPADEFINE and VIPABACKUP

To simplify these scenarios, we used one OSA port for each subnet with different costs and only one router.

4.3.1 Adapter interface failure

We stopped the OSA adapter (OSA2080) on our primary system to verify that the connection from the client workstation continued to function through an alternate route the during the takeover by the backup system. The failure scenario is illustrated in Figure 4-6.

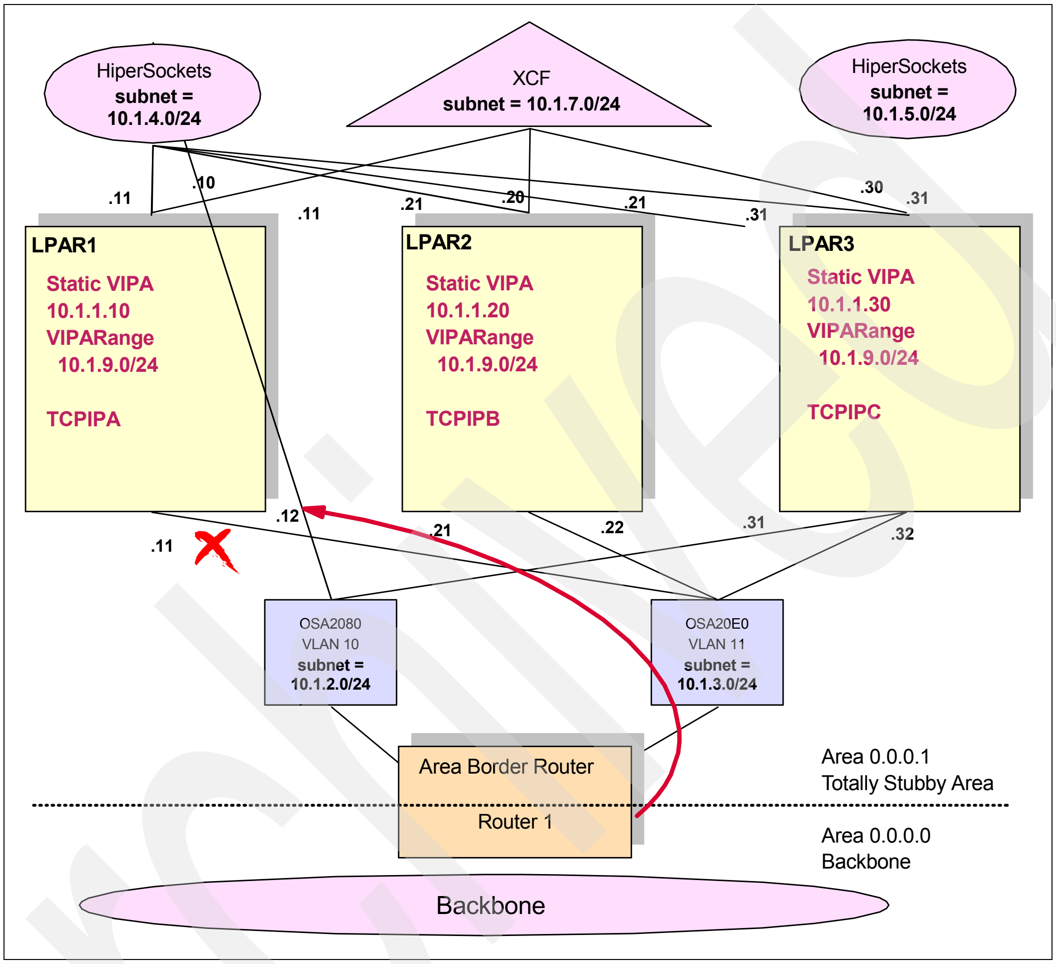


Figure 4-6 High availability for an adapter failure

Example 4-16 shows a display of the route table in Router 1 with the primary route to VIPA host 10.1.9.11 before we stop device OSA2080.

Example 4-16 Show IP route ospf 100 in Router one before we lost device OSA2080

```
O       10.1.9.11/32 [110/110] via 10.1.2.11, 00:01:28, Vlan10
                     [110/110] via 10.1.3.12, 00:01:28, Vlan11
```

We executed a **tracert** command in a workstation located in the backbone network to verify that we are able to reach host 10.1.9.11 on SC30, as shown in Example 4-17.

Example 4-17 TRACERT 10.1.9.11 command in the workstation

```
C:\Documents and Settings\RESIDENT>tracert 10.1.9.11
Tracing route to 10.1.9.11 over a maximum of 30 hops
  1    <1 ms    <1 ms    <1 ms  10.1.100.240
  2    <1 ms    <1 ms    <1 ms  10.1.9.11
```

Then we stopped OSA2080, which caused OSPF to lose adjacency to the neighbor at 10.1.2.240, as shown in Example 4-18.

Example 4-18 Show IP route ospf 100 in Router 1 after we lost device OSA2080

```
V TCPIP,TCPIPA,STOP,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPA,STOP,OSA2080
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE OSA2080
EZZ7921I OSPF ADJACENCY FAILURE, NEIGHBOR 10.1.2.240, OLD STATE 128,
NEW STATE 1, EVENT 11
```

Next, OSPF recovers through OSA20E0, which can be verified using the OSPF neighbor summary as shown in Example 4-19.

Example 4-19 Neighbor summary in SC30 showing OSPF has recovered its neighbors through OSA20E0

```
D TCPIP,TCPIPA,OMPR,OSPF,NBRS
RESPONSE=SC30
 EZZ7851I NEIGHBOR SUMMARY 524
 NEIGHBOR ADDR   NEIGHBOR ID     STATE  LSRXL DBSUM LSREQ HSUP IFC
 10.1.5.31       10.1.1.30         128      0     0     0  OFF IUTIQDF5L
 10.1.5.21       10.1.1.20         128      0     0     0  OFF IUTIQDF5L
 10.1.4.31       10.1.1.30         128      0     0     0  OFF IUTIQDF4L
 10.1.4.21       10.1.1.20         128      0     0     0  OFF IUTIQDF4L
 10.1.3.240      10.1.3.240        128      0     0     0  OFF OSA20E0L
 10.1.3.32       10.1.1.30           8      0     0     0  OFF OSA20E0L
 10.1.3.22       10.1.1.20           8      0     0     0  OFF OSA20E0L
 10.1.7.31       10.1.1.30         128      0     0     0  OFF IQDIOLNK*
 10.1.7.21       10.1.1.20         128      0     0     0  OFF IQDIOLNK*
 * -- LINK NAME TRUNCATED
```

Using command **sh ip route ospf 100**, we can confirm that we still have a route to reach host 10.1.9.11, as shown in Example 4-20.

Example 4-20 Show ip route ospf 100 command (only host address 10.1.9.11 shown here)

```
O       10.1.9.11/32 [110/110] via 10.1.3.12, 00:00:12, Vlan11
```

These displays confirm we were still able to reach VIPA host 10.1.9.11 through the alternate OSA path.

Next, we see what happens if we lose this OSA path. Refer to Figure 4-7.

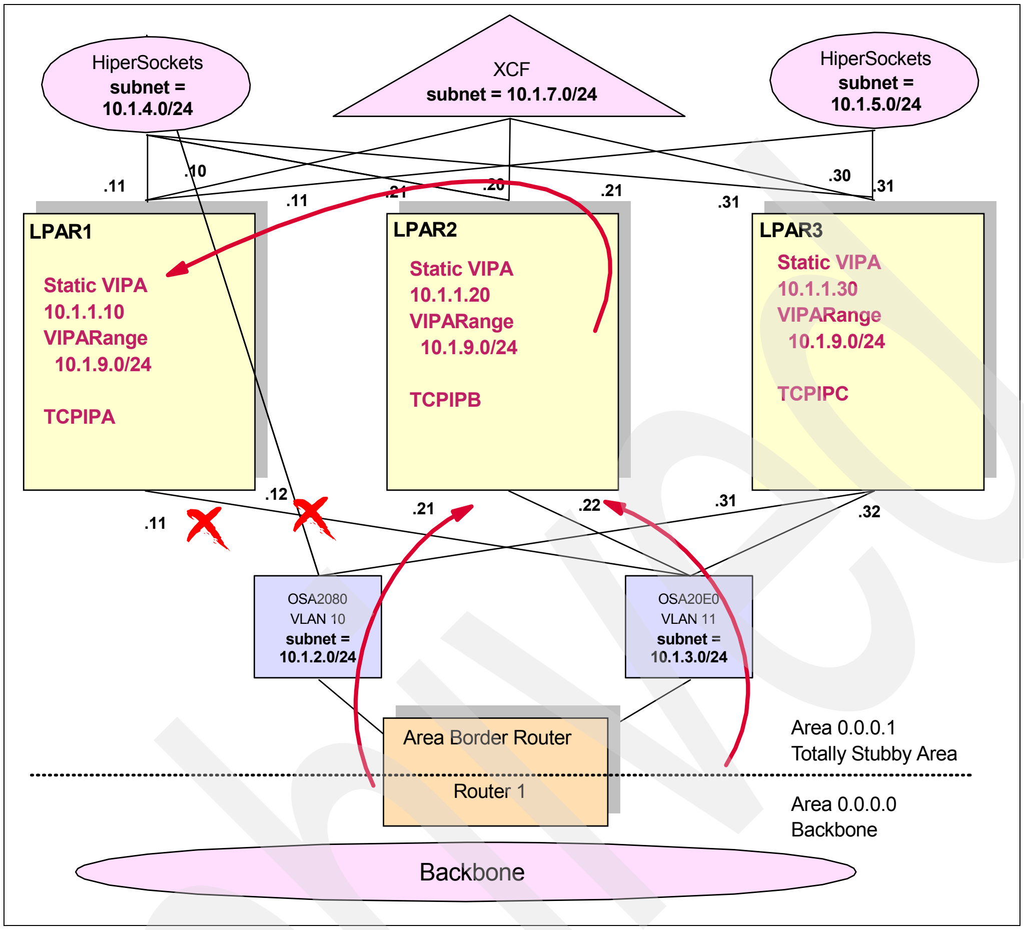


Figure 4-7 Losing contact with SC30 through OSA paths

We observed that OSPF in Router 1, as soon as it discovered that the route to VIPA 10.1.9.11 had been lost, calculated a new route, this time through links OSA2080 and OSA20E0 in stack TCPIPB. TCPIPB routes the traffic to VIPA 10.9.1.11 through HiperSockets. The new route can be verified by displaying the routing table to 10.1.9.11 in Router 1, as shown in Example 4-21.

Example 4-21 The sh ip route ospf 100 command results

```
Router1#sh ip route ospf 100
     10.0.0.0/8 is variably subnetted, 22 subnets, 2 masks
O       10.1.9.11/32 [110/190] via 10.1.3.22, 00:01:28, Vlan11
                     [110/190] via 10.1.2.21, 00:01:28, Vlan10
                     [110/190] via 10.1.3.32, 00:01:28, Vlan11
                     [110/190] via 10.1.2.31, 00:01:28, Vlan10
```

Example 4-22 shows a user issuing an **ls** command through the FTP session, with the expected results. The route convergence was transparent to the client. The result of a NETSTAT command is also shown in Example 4-22.

Example 4-22 FTP to VIPA 10.9.1.11 is still running OK after losing the OSA interface connections

```
ftp> ls
200 Port request OK.
125 List started OK
BRODCAST
HFS
NTUSER.DAT
NTUSER.DAT.LOG
NTUSER.INI
SC30.ISPF42.ISPPROF
SC30.SPFLOG1.LIST
SC30.SPF1.LIST
TESTE.TXT
TESTE2.TXT
250 List completed successfully.
ftp: 134 bytes received in 0.00Seconds 134000.00Kbytes/sec.
```

To recover the original route through OSA20A0, all we have to do is to restart the device. OSPF will recalculate the best route based on cost definitions.

4.3.2 Application movement using VIPADEFINE

This example illustrates how we can benefit from nondisruptive movement of an application by simply starting another instance of the application in another LPAR that takes over new connections. We defined the FTP services in SC30 and SC32 to be manually started; that is, we did not use *autolog* in the port definition profile. First, we started the FTP server in SC30 and this activated DVIPA address 10.1.9.11. Next, we connected to the FTP server from the workstation.

When the new instance of the application becomes active, the TCP/IP stack can immediately take over the ownership of the DVIPA while existing connections continue to the original TCP/IP stack (until the connections end or the original application or TCP/IP stack is taken down). This approach can be very useful for planned outage situations (such as software maintenance).

In failure situations, if the application supports Automatic Restart Manager (ARM), you might be able to have ARM restart the application automatically. If your application cannot bind to a DVIPA, you might be able to use the `bind()` on the port statement or the MODDVIPA utility. The situation is illustrated in Figure 4-8.

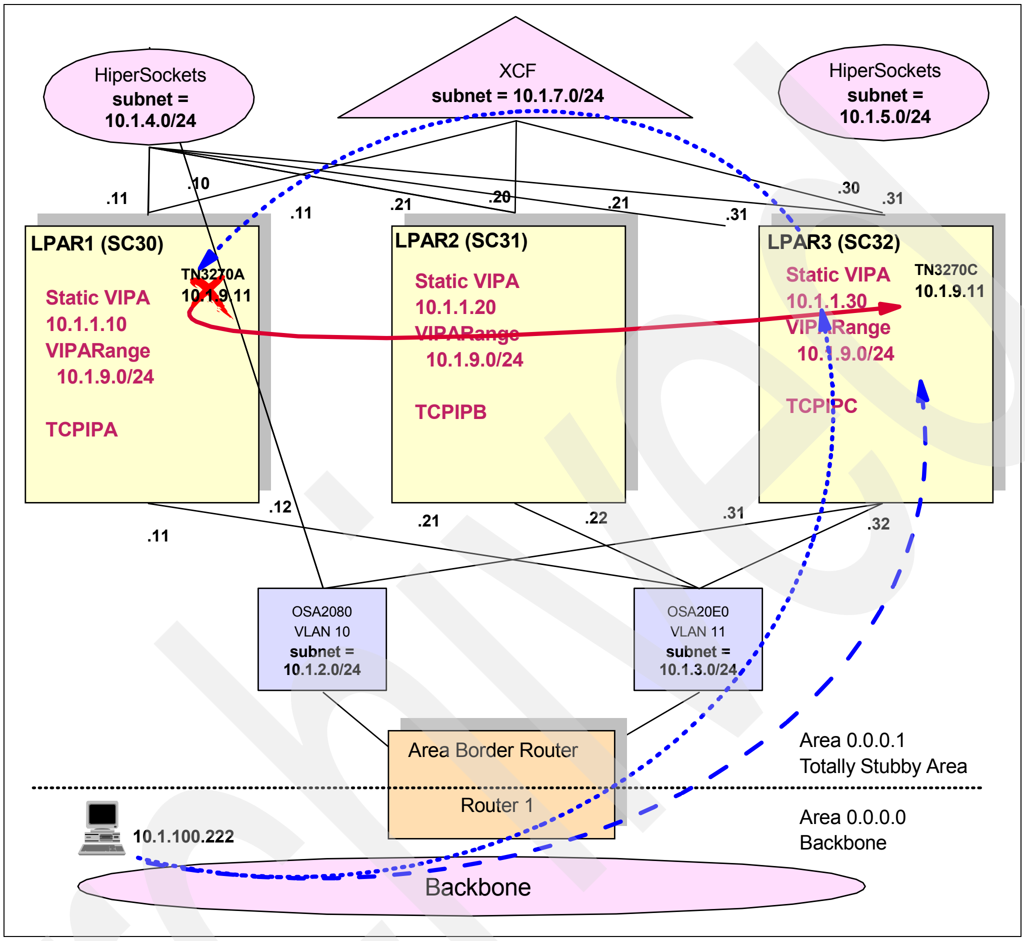


Figure 4-8 High availability for an application movement

We can observe that the DVIPA address defined by VIPARange was redirected from stack TCPIPA in SC30 to stack TCPIPC in SC32. This is the DVIPA movement. We can verify that the application movement process is working as expected by performing the following steps.

First, we started the TN3270A server using VIPA address 10.1.9.11, which is in the VIPARANGE definition in our test environment. We can verify this using the display command D TCPIP,TCPIPA,SYS,VIPADYN. The results are shown in Example 4-23.

Example 4-23 Verifying VIPA address 10.1.9.11 is active in SC30

```
D TCPIP,TCPIPA,SYS,VIPADYN,IPA=10.1.9.11
EZZ8260I SYSPLEX CS V1R11 054
VIPA DYNAMIC DISPLAY FROM TCPIPA   AT SC30
LINKNAME: VIPLOA01090B
IPADDR/PREFIXLEN: 10.1.9.11/24
  ORIGIN: VIPARANGE BIND
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPA   SC30     ACTIVE
```

The routing table in Router 1 shows that the route to VIPA address 10.1.9.11 is using SC30 OSA path, as shown in Example 4-24.

Example 4-24 The sh ip route 10.1.9.11 command results

```
Router1#sh ip route 10.1.9.11
Routing entry for 10.1.9.11/32
  Known via "ospf 100", distance 110, metric 110, type intra area
  Last update from 10.1.3.12 on Vlan11, 00:12:35 ago
  Routing Descriptor Blocks:
  * 10.1.2.11, from 10.1.1.10, 00:12:35 ago, via Vlan10
      Route metric is 110, traffic share count is 1
    10.1.3.12, from 10.1.1.10, 00:12:35 ago, via Vlan11
      Route metric is 110, traffic share count is 1
```

We started a new TN3270 Server in SC32 (TN3270C), and observed this stack take over the VIPA address, as shown in Example 4-25.

Example 4-25 The DVIPA address moves to SC32

```
$HASP373 TN3270C  STARTED
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
EZZ6001I TELNET SERVER STARTED
EZZ6003I TELNET LISTENING ON PORT  4992
EZZ6003I TELNET LISTENING ON PORT   992
EZZ8302I VIPA 10.1.9.11 TAKEN FROM TCPIPA ON SC30
EZD1205I DYNAMIC VIPA 10.1.9.11 WAS CREATED USING BIND BY TN3270C ON
TCPIPC
EZZ6003I TELNET LISTENING ON PORT    23
EZZ8303I VIPA 10.1.9.11 GIVEN TO TCPIPC ON SC32
```

The new VIPA location is advertised by OSPF changing the routes; see Example 4-26.

Example 4-26 The sh ip route 10.9.1.11 command to verify the new routing to reach VIPA 10.9.1.11

```
Router1#sh ip route 10.1.9.11
Routing entry for 10.1.9.11/32
  Known via "ospf 100", distance 110, metric 110, type intra area
  Last update from 10.1.3.31 on Vlan11, 00:08:38 ago
  Routing Descriptor Blocks:
  * 10.1.2.31, from 10.1.1.30, 00:08:38 ago, via Vlan10
      Route metric is 110, traffic share count is 1
    10.1.3.32, from 10.1.1.30, 00:08:38 ago, via Vlan11
      Route metric is 110, traffic share count is 1
```

We can now verify the DVIPA status, confirming it has moved to SC32. Example 4-27 illustrates this new status.

Example 4-27 DVIPA active in SC32

```
D TCPIP,TCPIPC,SYS,VIPADYN,IPA=10.1.9.11
EZZ8260I SYSPLEX CS V1R11 737
VIPA DYNAMIC DISPLAY FROM TCPIPC   AT SC32
LINKNAME: VIPL0A01090B
IPADDR/PREFIXLEN: 10.1.9.11/24
  ORIGIN: VIPARANGE BIND
  TCPNAME  MVSNAME  STATUS RANK DIST
```

```
  -------- -------- ------ ---- ----
  TCPIPC   SC32     ACTIVE
  TCPIPA   SC30     MOVING
2 OF 2 RECORDS DISPLAYED
```

The active connections with TN3270A server in SC30 will continue to work and the traffic related to it will be sent through the new route and redirected to SC30 using the XCF connection. When a new TN3270 client connects using address 10.1.9.11, it will connect to TN3270C Server in SC32. This situation can be verified using the display command D TCPIP,TCPIPC,N,VCRT as shown in Example 4-28.

Example 4-28 D TCPIP,TCPIPC,N,VCRT

```
D TCPIP,TCPIPC,N,VCRT
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.1.9.11..23
  SOURCE:  10.1.100.222..2009
  DESTXCF: 10.1.7.11
DEST:      10.1.9.11..23
  SOURCE:  10.1.100.222..2012
  DESTXCF: 10.1.7.31
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

The existing sessions remain with SC30 until they end. At that time the DVIPA address for SC30 disappears.

4.3.3 Stack failure scenario using VIPADEFINE and VIPABACKUP

In this example, we lose the entire TCP/IP stack. (We simulate this by simply stopping the stack.) The failure situation is illustrated in Figure 4-9.

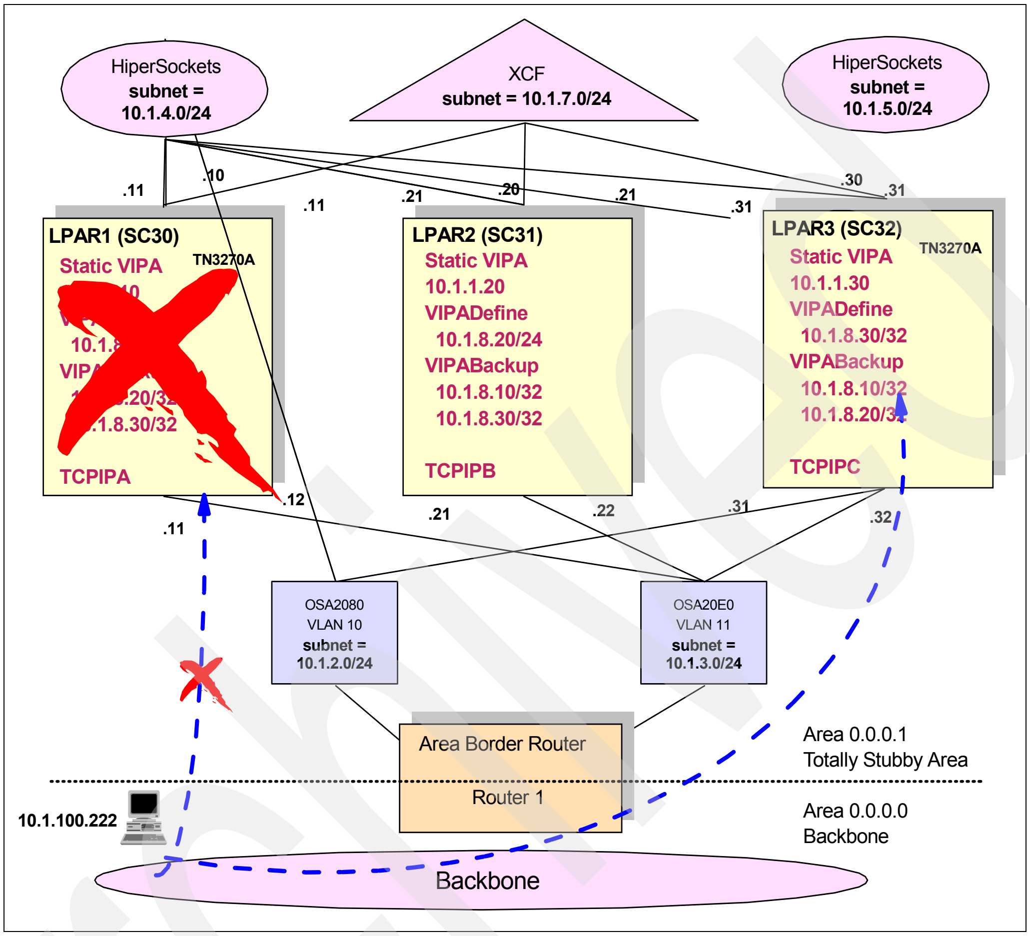


Figure 4-9 High availability for a stack failure

We used VIPADEFINE and VIPABACKUP definitions for the DVIPA to create a takeover and takeback environment. In our TCPIPA configuration, we created a VIPADEFINE with address 10.1.8.10, which is activated in SC30 as soon as the stack comes up. We also created, in the TCPIPC configuration, a VIPABACKUP for 10.1.8.10 with a rank of 4, stating this stack is a candidate to take over this address if something happens with the TCPIPA stack. We can verify this configuration is in place using command **`display tcpip,sys,vipadyn`**, as shown in Example 4-29.

Example 4-29 The d tcpip,tcpipa,vipadyn,ipa=10.1.8.10 command

```
D TCPIP,TCPIPA,SYS,VIPADYN,IPA=10.1.8.10
EZZ8260I SYSPLEX CS V1R11 845
VIPA DYNAMIC DISPLAY FROM TCPIPA   AT SC30
LINKNAME: VIPL0A01080A
IPADDR/PREFIXLEN: 10.1.8.10/24
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPA   SC30     ACTIVE
```

```
  TCPIPC   SC32     BACKUP 004
2 OF 2 RECORDS DISPLAYED
```

We established a telnet session using address 10.1.8.10, then stopped the primary TCP/IP (in SC30) and observed the results; see Example 4-30.

Example 4-30 High availability in failure of the stack

```
P TCPIPA
EZZ3205I SNMP SUBAGENT: SHUTDOWN IN PROGRESS
EZZ0673I AUTOLOG TASK: SHUTDOWN IN PROGRESS
EZZ6008I TELNET STOPPING
EZZ8301I VIPA 10.1.8.10 TAKEN OVER FROM TCPIPA ON SC30
RO SC31,D TCPIP,TCPIPC,N,VIPADYN
DYNAMIC VIPA:
IPADDR/PREFIXLEN: 10.1.8.10/32
    STATUS: ACTIVE     ORIGIN: VIPABACKUP       DISTSTAT:
```

This is a disruptive failure mode. Depending on the applications involved, clients might need to restart their sessions. (A subsequent takeback, when the TCP/IP stack in SC30 is restarted, is nondisruptive.)

When system SC32 discovers that SC30 has gone down (when SC30 leaves the XCF group), SC32 will take over the DVIPA address 10.1.8.10, advertising its new address through OSPF, as shown in Example 4-31.

Example 4-31 The sh ip route 10.1.8.10 in Router 1

```
Router1#sh ip route 10.1.8.10
Routing entry for 10.1.8.10/32
  Known via "ospf 100", distance 110, metric 110, type intra area
  Last update from 10.1.3.31 on Vlan11, 00:00:32 ago
  Routing Descriptor Blocks:
  * 10.1.2.31, from 10.1.1.30, 00:00:32 ago, via Vlan10
      Route metric is 110, traffic share count is 1
    10.1.3.32, from 10.1.1.30, 00:00:32 ago, via Vlan11
      Route metric is 110, traffic share count is 1
```

After that we started new Telnet connections to the DVIPA address 10.1.8.10, which was now running on SC32. We used the NETSTAT operand N,VCRT to verify that these connections were going to DESTXCF 10.1.7.31 (the Dynamic XCF address of the SC32 stack) as shown in Example 4-32.

Example 4-32 New connections go to SC31

```
D TCPIP,TCPIPC,N,VCRT
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.1.8.10..23
  SOURCE:  10.1.100.222..1998
  DESTXCF: 10.1.7.31          1
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

In this example, the number corresponds to the following information:

1. DESTXCF 10.1.7.31 is the Dynamic XCF address of the SC32 stack.

We restarted the TCPIPA stack in SC30 and observed the takeback operation; see Example 4-33.

Example 4-33 Start TCPIPA in SC30 takeback function

```
S TCPIPA
...
$HASP373 TCPIPA   STARTED
...
EZB6473I TCP/IP STACK FUNCTIONS INITIALIZATION COMPLETE.
EZAIN11I ALL TCPIP SERVICES FOR PROC TCPIPA ARE AVAILABLE.
EZD1176I TCPIPA HAS SUCCESSFULLY JOINED THE TCP/IP SYSPLEX GROUP EZBTCPCS
EZZ8302I VIPA 10.1.8.10 TAKEN FROM TCPIPC ON SC32
EZZ8303I VIPA 10.1.8.10 GIVEN TO TCPIPA ON SC30
EZD1214I INITIAL DYNAMIC VIPA PROCESSING HAS COMPLETED FOR TCPIPA
```

We verified that OSPF advertised the relocation of the VIPA address 10.1.8.10 to SC30 by displaying the route table in Router 1; see Example 4-34.

Example 4-34 The sh ip route 10.1.8.10

```
Router1#sh ip route 10.1.8.10
Routing entry for 10.1.8.10/32
  Known via "ospf 100", distance 110, metric 110, type intra area
  Last update from 10.1.3.11 on Vlan11, 00:12:34 ago
  Routing Descriptor Blocks:
  * 10.1.2.11, from 10.1.1.10, 00:12:34 ago, via Vlan10
      Route metric is 110, traffic share count is 1
    10.1.3.12, from 10.1.1.10, 00:12:34 ago, via Vlan11
      Route metric is 110, traffic share count is 1
```

Finally, we started a new TN3270 session using the same address, and confirmed that we connected to the server located in SC30. We also confirmed our TN3270 session was still up as expected, using command D TCPIP,TCPIPA,N,VCRT; see Example 4-35.

Example 4-35 D TCPIP,TCPIPA,N,VCRT

```
RESPONSE=SC30
DYNAMIC VIPA CONNECTION ROUTING TABLE:
 DEST:      10.1.8.10..23
   SOURCE:  10.1.100.222..2001
   DESTXCF: 10.1.7.11
 DEST:      10.1.8.10..23
   SOURCE:  10.1.100.222..1998
   DESTXCF: 10.1.7.31
 2 OF 2 RECORDS DISPLAYED
 END OF THE REPORT
```

5

Internal application workload balancing

With internal application workload distribution, decisions for load balancing are made within the z/OS environment. Decisions can be based on:

- Application availability information received through Sysplex XCF communications
- Real-time server capacity information from Workload Manager (WLM)
- Quality of Service (QoS) data provided by the Service Policy Agent
- A simple round-robin mechanism
- Distribution method based on target weight using WEIGHTEDActive connections
- Distribution controlled using weights that are received from non-z/OS targets servers.

The application workload balancing decision maker provided with the Communications Server is the Sysplex Distributor. Another option, the use of Shareport, is a method of workload distribution performed by the stack itself within a single z/OS system. This chapter describes internal application workload balancing and focuses primarily on the Sysplex Distributor function and the benefits that it provides. However, it also explains the use of Shareport.

This chapter discusses the following topics.

Section	Topic
5.1, “Basic concepts of internal application workload balancing” on page 96	Basic concepts of internal application workload balancing
5.2, “Design and implementation examples” on page 132	Commonly implemented internal application workload balancing design scenarios, their dependencies, advantages, considerations, and recommendations
5.3, “Problem determination” on page 168	Summary of useful NETSTAT commands and debugging approaches

5.1 Basic concepts of internal application workload balancing

The design of the Sysplex Distributor provides an advisory mechanism that checks the availability of applications running on different z/OS servers in the same sysplex, and then selects the best-suited target server for a new connection request. The Sysplex Distributor bases its selections on real-time information from sources such as Workload Manager (WLM), QoS data from the Service Policy Agent, and weights received from non-z/OS targets servers (such as WebSphere DataPower). Sysplex Distributor also measures the responsiveness of target servers in accepting new TCP connection setup requests, favoring those servers that are accepting new requests more successfully.

Internal workload balancing within the sysplex ensures that a group or cluster of application server instances can maintain optimum performance by serving client requests simultaneously. High availability considerations suggest at least two application server instances should exist, both providing the same services to their clients. If one application instance fails, the other carries on providing service. Multiple application instances minimize the number of users affected by the failure of a single application server instance. Thus, load balancing and availability are closely linked.

Figure 5-1 depicts, at a very high level, where the decision-maker for internal application workload balancing resides.

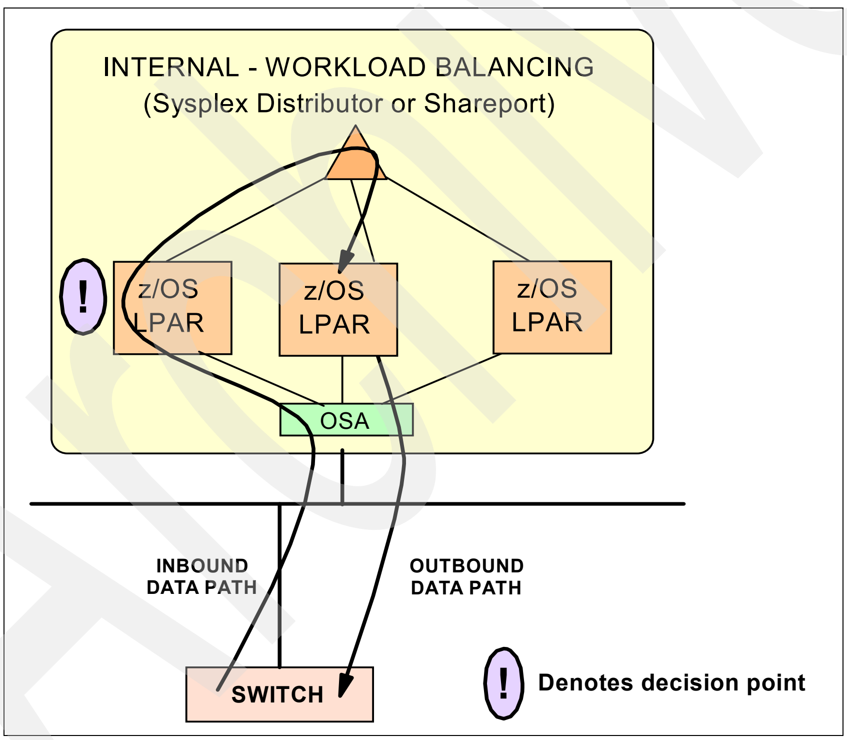


Figure 5-1 Decision point within the sysplex: internal workload balancing

The Sysplex Distributor is the internal decision-maker for application workload balancing. The Sysplex Distributor, together with XCF dynamics and dynamic VIPA, creates an advanced level of availability and workload balancing in a sysplex. Workload can be distributed to multiple server application instances without requiring changes to clients or networking hardware, and without delays in connection setup.

z/OS Communications Server allows you to implement a dynamic VIPA as a single network-visible IP address for a set of hosts that belong to the same sysplex cluster. A client located anywhere in the IP network can see the sysplex cluster as one IP address, regardless of the number of hosts that it includes.

For background information that describes how to use Distributed DVIPA in a Sysplex Distributor environment, including failover and recovery scenarios, refer to “Distributed DVIPA (Sysplex Distributor)” on page 20.

If you are planning to implement multitier optimized intra-sysplex workload balancing, refer to Chapter 7, “Intra-sysplex workload balancing” on page 213 for more information.

5.1.1 Sysplex Distributor—principles of operation

The Sysplex Distributor is a network-connected stack that owns a specific VIPA address and acts as the distributor for connection requests to the VIPA address. It is independent of network attachment technology, and works with both direct LAN connections (including OSA Express) and channel-attached router network connections.

All z/OS images that are involved communicate through XCF, which permits each TCP/IP stack to have full knowledge of IP addresses and server availability in all stacks. As mentioned earlier, the distribution of new connection requests for a z/OS application or non-z/OS application can be based on real-time consultation with WLM (or round-robin), z/OS QoS Policy Agent, and the target stack for application weight and availability.

When the selection of the target stack is complete, the connection information is stored in the Sysplex Distributor stack to route future IP packets that belong to the same connection as the selected target stack. Routing is based on the connection (or session) to which IP packets belong, which is known as *connection-based routing*. The backup stack is free to do other work but takes over the ownership of the VIPA address and the distributor function if the primary stack leaves the XCF group due to a failure or a planned outage. This takeover is nondisruptive to all existing connections that do not terminate on a failed node.

Figure 5-2 provides a conceptual overview of Sysplex Distributor operation.

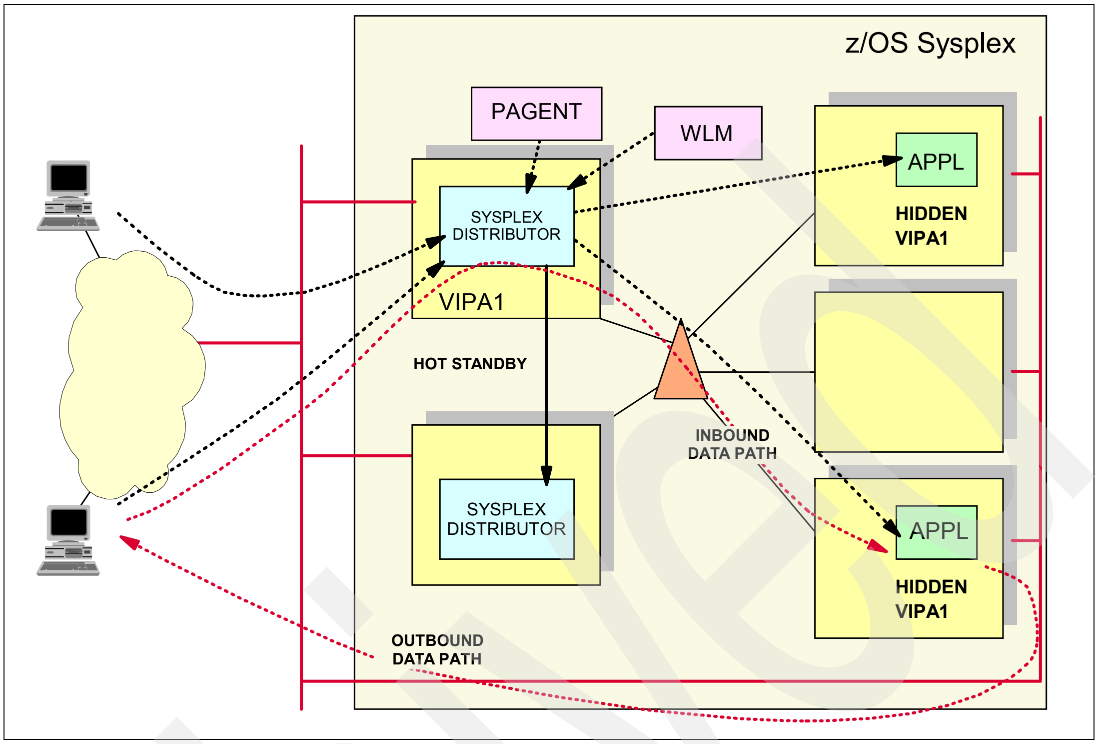


Figure 5-2 Sysplex Distributor for z/OS-integrated intra-sysplex workload balancing

To configure the distribution method used by Sysplex Distributor, use the VIPADISTRIBUTE statement on the VIPADYNAMIC / ENDVIPADYNAMIC block. After defining a DVIPA with VIPADEFINE or VIPABACKUP (as described in 3.2.1, “Implementation” on page 52), you can configure your VIPADISTRIBUTE statement.

```
>>-VIPADYNAMIC--------------------------------------------------------------->

>>-VIPADEFINE-----+-Define parameters-+-----+-address mask--+-----+-ipv4 addr-+--------->
                                            +-ipv6 intfname-+     +-ipv6 addr-+

|--VIPADISTRIBUTE-----+-Dist parameters-+----+-ipv4 addr-+-----+-PORT num-+---+-DestIP-+--->
                                             +-ipV6 intf-+

|--ENDVIPADynamic--|
```

Figure 5-3 VIPADISTRIBUTE statement

5.1.2 Sysplex Distributor and Quality of Service policy

The use of internal application workload balancing and distribution is closely related to Quality of Service (QoS) objectives. The Policy Agent interacts with the Sysplex Distributor to assist with workload balancing. The ability to monitor server performance dynamically and affect sysplex workload distribution is an important part of the overall QoS mechanism.

The Policy Agent performs the following distinct functions to assist the Sysplex Distributor:

- Policies can be defined to control which stacks the Sysplex Distributor can select. The definition of the *outbound interface* on the PolicyAction statement can limit the potential target stacks to a subset of those defined on the VIPADISTRIBUTE statement in the TCPIP.PROFILE. The stacks to which work is distributed can vary, for example, based on time periods in the policy. Another possibility is to limit the number of the Sysplex Distributor target stacks for inbound traffic from a given subnet. In this way, the total set of target stacks can be partitioned among different groups of users or applications requesting connections to distributed applications.
- The PolicyPerfMonitorForSDR statement in the `pagent.conf` file activates the Policy Agent QoS performance monitor function. When activated, the Policy Agent uses data about packet loss and timeouts that exceed defined thresholds and derives a QoS weight fraction for that target stack. This weight fraction is then used to reduce the WLM weight assigned to the target stacks, so that the Sysplex Distributor stack can use this information to better direct workload to where network traffic is best being handled. In this way, processor performance, server performance, and application network performance are taken into account when a decision is made for work distribution. This policy is activated on the Sysplex Distributor and the target stacks.
- The Policy Agent at each target now collects information with an additional level of granularity, and the QoS performance data is collected for each service level that a target's DVIPA port or application supports.
- The Policy Agent on the distributing stack drives the collection of this information by pulling it from the Policy Agents on the target stacks.

To exclude stale data from target stacks where the Policy Agent has terminated, the Policy Agent sends a *heartbeat* to the distributing stack at certain intervals. The distributing stack deletes QoS weight fraction data from a target stack when the heartbeat has not been received within a certain amount of time.

5.1.3 Portsharing

For a TCP server application to support a large number of client connections on a single system, it might be necessary to run more than one instance of the server application. (This assumes the application design allows this usage.) *Portsharing* is a method to distribute workload for IP applications *within* a z/OS LPAR. TCP/IP allows multiple listeners to listen on the same combination of port and interface. Workload destined for this application can be distributed among the group of servers that listen on the same port. Portsharing does not rely on an active Sysplex Distributor implementation; it works without Sysplex Distributor. However, you can use portsharing in addition to Sysplex Distributor operation.

z/OS currently supports the following forms of portsharing:

- SHAREPORT
- SHAREPORTWLM

SHAREPORT

When specified on the PORT statement, incoming client connections for this port and interface are distributed by the TCP/IP stack across the listeners, using a weighted round-robin distribution method based on the Server Efficiency Fractions (SEFs) of the listeners sharing the port. The SEF is a measure of the efficiency of the server application, calculated at intervals of approximately one minute, in accepting new connection requests and managing its backlog queue.

Figure 5-4 depicts a simple configuration where SHAREPORT is used for internal workload balancing.

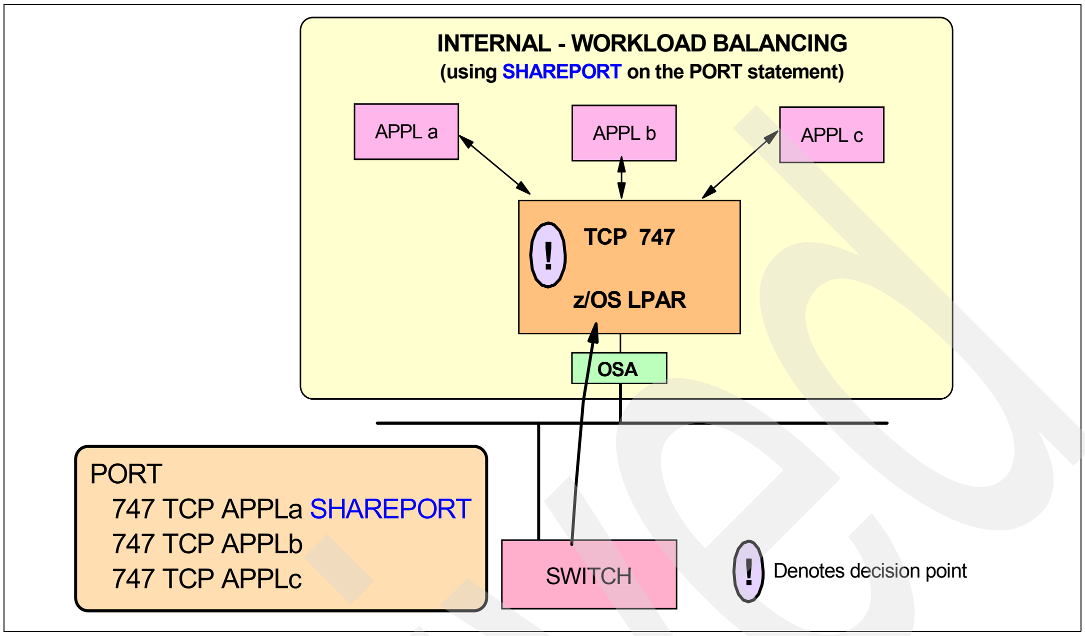


Figure 5-4 The TCP/IP stack is the decision-maker using a weighted round-robin method

SHAREPORTWLM

You can use the SHAREPORTWLM option instead of SHAREPORT. Similar to SHAREPORT, SHAREPORTWLM causes incoming connections to be distributed among a set of TCP listeners. However, unlike SHAREPORT, the listener selection is based on WLM server-specific recommendations, modified by the SEF values for each listener. WLM server-specific recommendations are acquired at intervals of approximately one minute from WLM, and they reflect the listener's capacity to handle additional work.

Figure 5-5 depicts a simple configuration where SHAREPORTWLM is used for internal workload balancing.

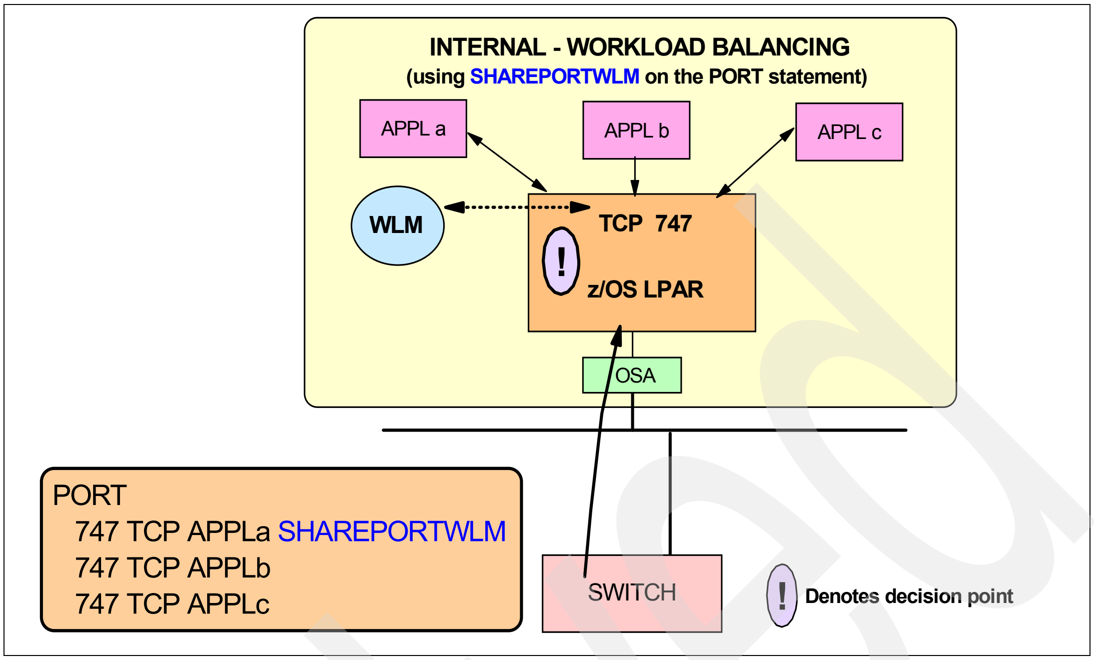


Figure 5-5 TCP/IP stack is the decision-maker using server-specific WLM data

Setting up SHAREPORTWLM

For a TCP server application to support a large number of client connections on a single system while still providing good performance for those connections, it might be necessary to run more than one instance of the server application to service the connection requests. For all instances of the server application to receive client connection requests without changing the client applications, each server must issue a bind request to the same server IP address and port. To enable this binding in the TCP/IP stack, you must add the SHAREPORT or SHAREPORTWLM keyword to the PORT profile statement that reserves the TCP port for the server instances. The SHAREPORT statement can use also zAAP and zIIP co-processors.

We added the statement to the TCP/IP profile, as shown in Example 5-1.

Example 5-1 TELNET SHAREPORT

```
PORT
    23 TCP TN3270C SHAREPORTWLM
```

Then, we started a Telnet session to our stack and issued the following command:

```
NETSTAT ALL TCP TCPIPC (PO 23
```

Example 5-2 shows the output of this command.

Example 5-2 Output of NETSTAT ALL

```
ReceiveBufferSize:   0000065536          SendBufferSize:     0000065536
ConnectionsIn:       0000000001          ConnectionsDropped: 0000000000
CurrentBacklog:      0000000000          MaximumBacklog:     0000000010
CurrentConnections:  0000000001          SEF:                100
Quiesced:            No
SharePort:           WLM
  RawWeight:         064                 NormalizedWeight:   016
  Abnorm:            0000                Health:             100
```

```
RawCP:  064       RawzAAP:  000     RawzIIP:  000
PropCP: 064       PropzAAP: 000     PropzIIP: 000
```

We were not using IPSECURITY, and zAAP and zIIP show 000.

For more information about SHAREPORT, refer to 5.2.5, “Portsharing using SHAREPORTWLM” on page 149.

5.1.4 Optimized routing

Multiple instance applications, when running in a Sysplex Distributor Parallel Sysplex environment, cannot directly influence how many connections they will get or when they will get them. They rely on the Sysplex Distributor for those decisions. z/OS Communications Server uses optimized routing for the Sysplex Distributor and the DVIPA IP forwarding function so that the z/OS Communications Server can use any available IP network connectivity between TCP/IP stacks.

VIPAROUTE

This function allows you to control the path taken to the target application server using the VIPAROUTE statement. The VIPAROUTE statement indicates which IP address on the target stack will be used as the destination IP address during route lookup selection. VIPAROUTE is used to select a route from a distribution stack to a target stack.

The first parameter of the VIPAROUTE statement specifies the Dynamic XCF address of the target stack. The second parameter specifies any fully qualified IP address in the HOME list of the target server (preferably a static VIPA address so that interface failures will not affect the availability of the target stack). An example of this usage is:

```
VIPADISTRIBUTE DEFINE 10.1.8.10 PORT 23 DESTIP ALL
VIPAROUTE DEFINE 10.1.7.21 10.1.1.20
```

Optimal routes to target servers might be:

- IUTSAMEH, when the target server is within the same z/OS image
- HiperSockets, when the target server is within the same CED
- OSA-Express cards, when the target server is in another CEC

You are not required to use these specific routes, of course. Other considerations might dictate using (or not using) certain routes.

> **Tip**: With this function, you can also use your dynamic XCF interfaces as backup routes, or preclude the use of dynamic XCF interfaces completely when routing DVIPA traffic.

Figure 5-6 illustrates the use of DVIPA IP forwarding using OSA-Express adapters.

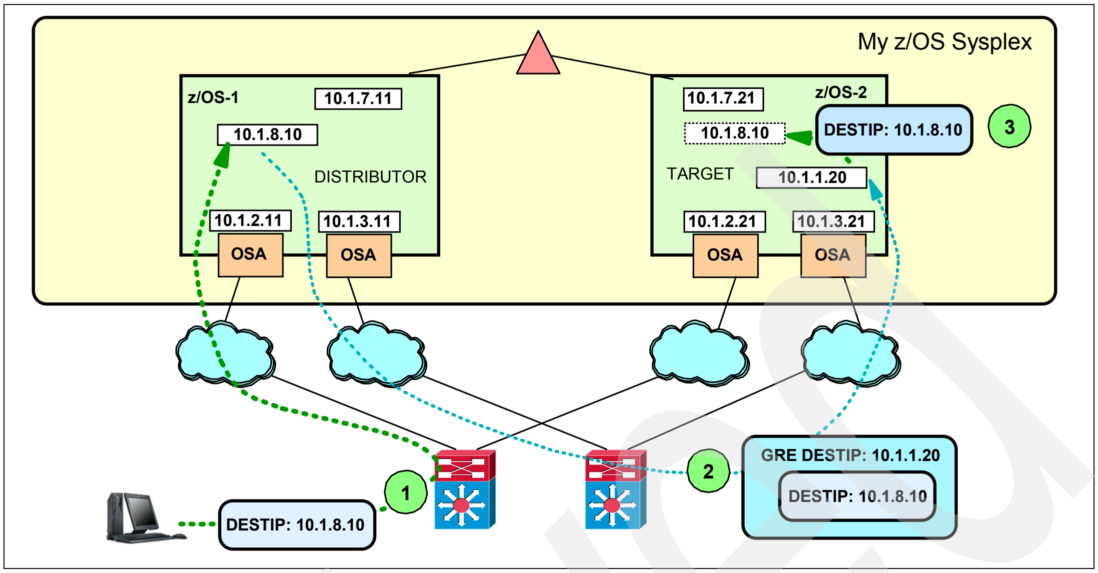


Figure 5-6 Intra-sysplex IP communication using OSA-Express instead of DynamicXCF

When a connection from the client is processed by Sysplex Distributor, it determines whether a matching VIPAROUTE statement has been specified. If it has, the best available route is determined using the normal IP routing tables. If no matching VIPAROUTE statement exists for that target, the Sysplex Distributor uses Dynamic XCF interfaces.

When a VIPAROUTE statement is in effect, the packet is encapsulated using GRE prior to being forwarded. This encapsulation enables the packet to be forwarded through the network to the target stack while preserving the original packet's destination IP address (that is, the DVIPA address).

The GRE encapsulation process increases the size of the forwarded packet by 28 bytes. As a result, if the size of the encapsulated GRE packets is larger than the maximum transmission unit (MTU) of the network interface that will be used for forwarding the packet, the TCP/IP stack might need to perform fragmentation, creating two or more packets that are forwarded to the target stack. The target stack then reassembles the fragmented packets.

Recommendations

We make the following recommendations for using this function:

- Define a HiperSockets network in each CEC (through DEVICE/LINK definitions that interconnect all LPARs in that same CEC) and use a low routing cost for this network, so that it is the first-choice routing path.
- Define Gigabit Ethernet connectivity from all LPARs and use a low routing cost, but higher than what was used for the HiperSockets network, so that it is the second-choice routing path.
- Define the dynamic XCF network with a rather high routing cost so it will not be used for normal IP routing unless it is the only available interface (or define it as a non-OSPF interface so that it is not used for distributed DVIPA routing at all). That will save XCF communications for important sysplex data sharing traffic that must use it.

Restrictions

Note the following restrictions:

- Internal indicators, requested using the SO_CLUSTERCONNTYPE option of the GETSOCKOPT API and used to determine if the partner application is running in the same system or in the same sysplex, are no longer set if the destination IP address is a Dynamic VIPA or distributed Dynamic VIPA residing in the sysplex. The reason for this is that traffic destined to these IP addresses can now be forwarded to the target TCP/IP stacks over links or interfaces that are external to the sysplex.
- You must still configure the dynamic XCF address when using VIPAROUTE support because the Sysplex Distributor still uses dynamic XCF addresses to identify target TCP/IP stacks in the sysplex. In addition, there are several functions that continue to depend on dynamic XCF connectivity for intra-sysplex communications:
  - Sysplex Wide Security Associations (SWSA)
  - Multi-Level Security (MLS) packets
  - Quality of Service (QoS) performance data collection of Policy Agent

Sysplex Distributor connection routing accelerator

With Sysplex Distributor, the distributor stack receives packets and forwards them to a target stack. QDIO Accelerator can provide accelerated forwarding of packets that Sysplex Distributor forwards to a target stack when the packets are flowing over one of the following inbound and outbound DLC combinations:

- Inbound packets that are routed over HiperSockets and that are forwarded outbound over OSA-Express QDIO or HiperSockets
- Inbound packets that are routed over OSA-Express QDIO and that are being forwarded outbound over OSA-Express QDIO or HiperSockets

Figure 5-7 shows Sysplex Distributor forwarding with QDIO Accelerator.

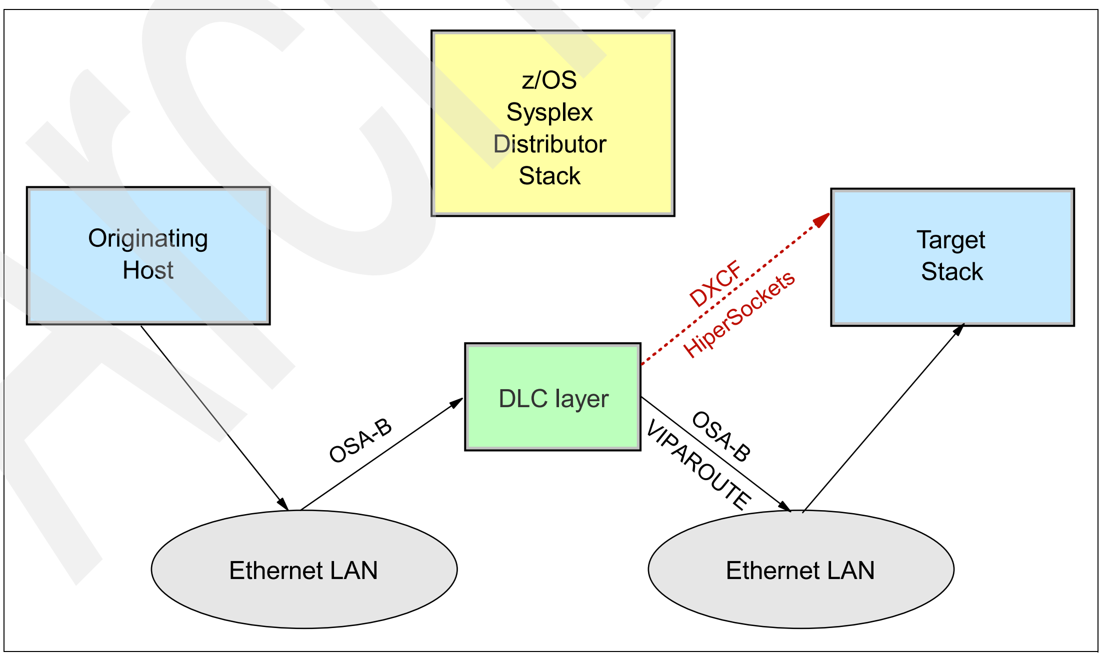


Figure 5-7 Sysplex Distributor forwarding with QDIO Accelerator

In this example, Sysplex Distributor is using a VIPAROUTE over an OSA-Express QDIO interface to get to the target stack (as shown by the solid arrows). Elements of the IP routing table are “pushed” into the DLC layer. This function enables the DLC layer to make routing decisions between the supported interfaces without passing IP packets up through the IP layer, thus consuming fewer processor resources and providing lower latency while routing the packets.

This function also provides Sysplex Distributor acceleration when the Sysplex Distributor reaches the target stack using the Dynamic XCF connectivity over HiperSockets (as shown by the dotted arrow). In addition, although the example shows inbound OSA-Express QDIO, the Sysplex Distributor acceleration function also applies to data that is received inbound over HiperSockets.

To enable QDIO Accelerator, configure QDIOACCELERATOR on the IPCONFIG statement in the TCPIP profile, as shown in Example 5-3. This parameter is mutually exclusive with IQDIOROUTING. You can turn these accelerator functions off and on using the VARY TCPIP,,OBEYFILE command, but you cannot enable either accelerator function unless one was configured in the initial profile.

Example 5-3 Routing accelerator option on TCPIP profile

```
IPCONFIG DATAGRAMFWD SYSPLEXROUTING QDIOACCELERATOR
```

When the QDIOACCELERATOR function is enabled, qualified traffic takes advantage of the improved path. The TCP/IP stack keeps track of the accelerated QDIO Accelerator routing table in a routing table that is a subset of the stack’s own routing table. This routing table does not include Sysplex Distributor routes. You can display that routing table using the NETSTAT ROUTE QDIOACCE command.

Table 5-1 shows the types of tasks and information that are available for operating with the QDIOACCELERATOR function.

Table 5-1 QDIOACCELERATOR tasks

Task	Procedure
Enable the QDIO Accelerator function to provide accelerated forwarding of packets	Specify QDIOACCELERATOR parameter on the IPCONFIG statement in the TCP/IP profile
Display whether QDIO Accelerator is enabled	Issue the Netstat CONFIG/-f command
Display the routing table that has been pushed into the DLC layer	Issue the Netstat ROUTE QDIOACCE command
Display whether a Sysplex Distributor connection is eligible for acceleration	Issue the Netstat VCRT/-V command with DETAIL modifier
Display information about the number of packets and bytes that are accelerated after being received over a specific interface	Initiate VTAM tuning statistics for the OSA-Express QDIO or HiperSockets interface
View information similar to packet trace for packets that are accelerated either from or to an OSA-Express QDIO interface	Use OSAENTA function

The following restrictions apply to this function:

- QDIOACCELERATOR is mutually exclusive with IPSECURITY; it cannot be enabled if IPSECURITY is enabled.
- The function applies only to IPv4.
- This function provides acceleration for VIPAROUTE over OSA, but not VIPAROUTE over HiperSockets. This is because of differences in how acceleration works over HiperSockets compared to OSA-Express QDIO.
- If the traffic that is forwarded requires fragmentation based on the MTU of the outbound route, then those packets are not accelerated but instead are forwarded by the stack.
- Incoming fragments for a Sysplex Distributor connection are not accelerated but instead sent to the stack, because the Sysplex Distributor requires the fragments to be reassembled before forwarding can occur. Similarly, the distributor stack needs to process SYN packets in order to select a target stack so SYN packets for Sysplex Distributor are not accelerated.
- Acceleration is not allowed to or from interfaces that use optimized latency mode (OLM). For more information, see *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798.

5.1.5 Improved workload distribution

With z/OS, there is a sysplex distribution feature providing server-specific WLM recommendations for load balancing. The distribution of new connection requests can be based on the actual workload of a target server. This method (SERVERWLM) indicates how well a specific server on a stack is meeting the goals of the WLM service class for the transactions that it is servicing (server-specific weights). This additional granularity enables the distributor to balance the workload more effectively.

Server-specific WLM recommendations can also be used by a stack to more evenly balance workload across a group of servers sharing the same port in a single system by specifying SHAREPORTWLM (instead of just SHAREPORT) on the PORT statement in the TCP/IP profile. When using this option, connections are distributed in a weighted round-robin fashion based on the WLM server-specific recommendations for the server running on that system.

BASEWLM background

BASEWLM provides capacity recommendations in the form of weights for each *system*. WLM assigns a relative weight to each system in the sysplex, with the highest weight going to the system with the most available CPU capacity. The weights range between 0 and 64. You can use BASEWLM parameters on the VIPADISTRIBUTE statement to configure its options as illustrated in Figure 5-8.

```
>>-VIPADYNAMIC------------------------------------------------------------------------->

>>-VIPADEFINE-----+-Define parameters-+------+-address mask--+------+-ipv4 addr-+---------->
                                             +-ipv6 intfname-+      +-ipv6 addr-+

|--VIPADISTRIBUTE-----+-Dist parameters-+----DISTMethod BASEWLM-----+-BaseWLMOptions-+--->

|---------------------+-ipv4 addr-+----------+-PORT num-+-----------+-DestIP-+---------->
                      +-ipV6 intf-+

|--ENDVIPADynamic--|
```

Figure 5-8 BASEWLM parameter on VIPADISTRIBUTE statement

In the example illustrated in Figure 5-9, BASEWLM system weights of 15, 25, and 5 are returned by WLM for the targets for DVIPA1 port 8000. Then a QoS service level fraction is received from the target for each group of connections that map to a DVIPA/PORT for that service level. The QoS fraction percentage represents the performance of this group of connections, and is used to reduce the WLM weight.

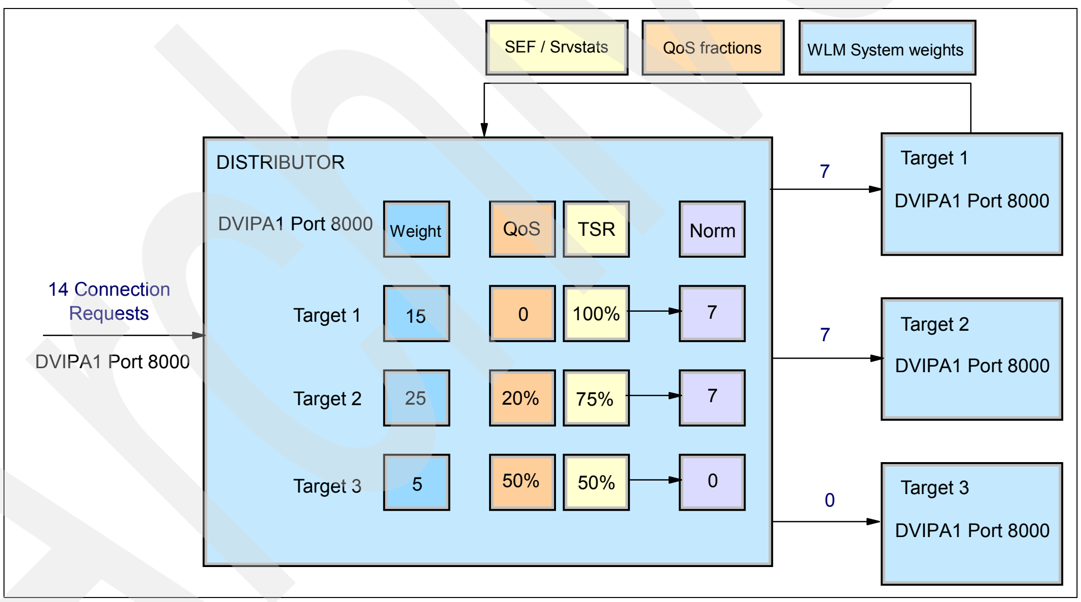


Figure 5-9 BASEWLM distribution method using QoS data

The weights are then *normalized* (reduced proportionally), if necessary, to keep the number of successive connections directed to any given target small. For example, the weights for target 1, target 2, and target 3 are calculated as shown in Table 5-2.

Table 5-2 BASEWLM weight calculations

	WLM weights	QoS fractions	TSR fractions	QoS-modified weights	Normalized weights
Target 1	15	0%	100	15 = (15 - (15 * 0%)) * 100%	7
Target 2	25	20%	75	15 = (25 - (25 * 20%)) * 75%	7
Target 3	5	50%	50	1.25 = (5 - (5 * 50%)) * 50%	0

> **Note:** The z/OS Communications Server uses the following normalization algorithm:
>
> If any system weight provided by WLM is greater than 16, then all weights are normalized by dividing by 4 (and ignoring any remainder) after applying QoS fractions.
>
> However, this approach might be changed in future releases of z/OS Communications Server.

Continuing with our example, assume that 14 connection requests arrive for DVIPA1, port 8000. Based on the QoS-modified WLM weights for this DVIPA, port, and service level, then seven connections are distributed to target 1, seven connections are distributed to target 2, and no connection is distributed to target 3.

Note the following considerations for this process:

- The WLM weight is based on a comparison of the available capacity for new work on each system, not how well each server is meeting the goals of its service class.
- If all systems are using close to 100% of capacity, then the WLM weight is based on a comparison of displaceable capacity (the amount of lower importance work on each system). However, if the service class of the server is of low importance, then it might not be able to displace this work.
- If a target stack or a server on the target stack is not responsive, new connection requests continue to be routed to the target. The stack might appear lightly loaded because applications are not processing any new work.
- QoS fractions monitor target or client performance but do not monitor the path to the target.

> **Note:** QoS fractions are only available if the Policy Agent is enabled on the Sysplex Distributor and all target systems and appropriate policies have been configured.

Server-specific WLM

WLM can assign a weight based on how well the *server* is meeting the goal of its service class and the displaceable capacity for the new work in relation to the actual used service. The displaceable capacity is based on the importance level of its service class for the work that is to be executed.

You can use SERVERWLM parameters with the VIPADISTRIBUTE statement to configure options, as shown in Figure 5-10.

```
>>-VIPADYNAMIC-------------------------------------------------------------->

>>-VIPADEFINE-----+-Define parameters-+------+-address mask--+------+-ipv4 addr-+---------->
                                             +-ipv6 intfname-+      +-ipv6 addr-+

|--VIPADISTRIBUTE----+-Dist parameters-+----DISTMethod SERVERWLM----+-ServerWLMOptions-+--->

|-------------------+-ipv4 addr-+----------+-PORT num-+-----------+-DestIP-+----------->
                    +-ipV6 intf-+

|--ENDVIPADynamic--|
```

Figure 5-10 SERVERWLM parameter on VIPADISTRIBUTE statement

Figure 5-11 illustrates the server-specific WLM distribution method.

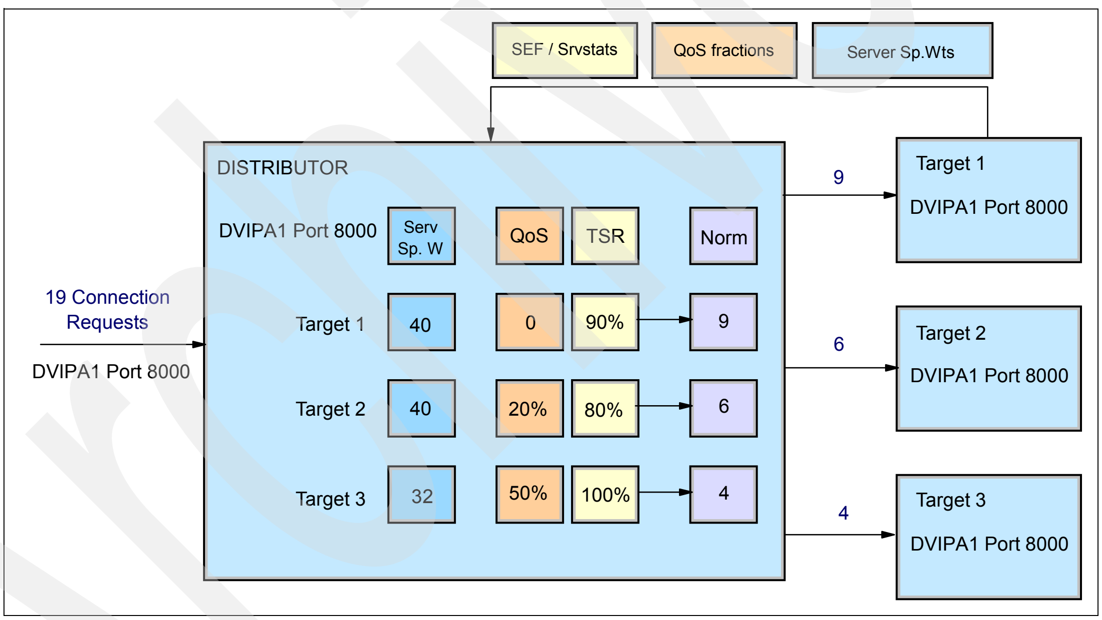


Figure 5-11 Server-specific WLM

In this example, a *server-specific* weight is received for each of the targets for DVIPA1, port 8000. The weights for target 1, target 2, and target 3 are calculated as shown in Table 5-3.

Table 5-3 Server-specific WLM weight calculation

	Server-specific weight * QOS * TSR	**Normalized weight**
Target 1	36 = (40 - (40 * 0%)) * 90%	9
Target 2	25 = (40 - (40 * 20%)) * 80%	6
Target 3	16 = (32 - (32 * 50%)) * 100%	4

WLM depends on the server receiving work to change server weights. Even if a server weight is zero (0), a connection request is forwarded infrequently to that server to generate new WLM values.

In our example, if 19 connections arrive for DVIPA1, port 8000 based on the QoS-modified WLM weights for this DVIPA, port, and service level, then nine connections are distributed to target 1, six connections are distributed to target 2, and four connections are distributed to target 3.

Note the following differences between the BASEWLM and SERVERWLM methods:

- BASEWLM uses the total available system capacity as the base for the weight, while SERVERWLM uses the capacity relative to the importance level of the specific workload that the server is running.
- SERVERWLM bases its calculation on the goal achievement of the work that the server is executing.
- SERVERWLM includes server health and abnormal termination information from WLM to mitigate the "storm drain problem."

> **Note:** To enable WLM server-specific recommendations, you must add the SERVERWLM parameter to an existing VIPADISTRIBUTE DISTMETHOD statement on the distribution stack.

Note the following restrictions when using SERVERWLM:

- WLM server recommendations can be used only if the Sysplex Distributor and all target stacks for a distributed DVIPA port are V1R7 or later. The distributing stack relies on receiving the WLM server recommendations from each target stack. The WLM system weight recommendations in prior releases are not comparable to the latest WLM server recommendations; therefore, if one of the target stacks is *not* z/OS V1R7 or later, then the BASEWLM distribution method must be used.
- The backup stack must be V1R7 or later so that, after a takeover, distribution continues to use the latest WLM server recommendations.

WEIGHTEDActive distribution

WEIGHTEDActive Distribution will provide a more granular control over workload distribution based on predetermined active connection count proportions for each target (fixed weights). Using this method, you can improve your control over the distribution, but also lose the benefit of allowing this distribution to be done based on WLM recommendations dynamically.

Figure 5-13 shows the WEIGHTEDACTIVE parameter with the VIPADISTRIBUTE statement.

```
>>-VIPADYNAMIC--------------------------------------------------------------->

>>-VIPADEFINE-----+-Define parameters-+------+-address mask--+------+-ipv4 addr-+---->
                                             +-ipv6 intfname-+      +-ipv6 addr-+

|--VIPADISTRIBUTE------DISTMethod WEITHEDActive------------------------------------>

|------------------+-ipv4 addr-+---------+-PORT num-+-----------+-DestIP-+---->
                   +-ipV6 intf-+

|--ENDVIPADynamic--|
```

Figure 5-12 WEIGHTEDACTIVE parameter on VIPADISTRIBUTE statement

The configured weight is used with other indicators to achieve the real connection weight on each target. These indicators are the server-specific abnormal completion information, the general health indicator, and the TSR value.

These values are used to reduce the active connection weight when these indicators are not optimal. If weighted active connections are used, study and determine the comparative workload that you want on each system so that you can configure appropriate connection weights.

If weighted active connections are configured with default proportions for each target, connections are evenly distributed across all target servers; the goal is to have an equal number of active connections on each server. This is similar to the round-robin distribution method; the difference is that round-robin distribution distributes connections evenly to all target servers without consideration for the active number of connections on each server. Round-robin distribution bypasses a target if the TSR value is 0. For weighted active forwarding, the TSR value and application health indicators are be used to reduce the active connection weight.

WEIGHTEDActive distribution provides more granular control over workload distribution as shown in Figure 5-13.

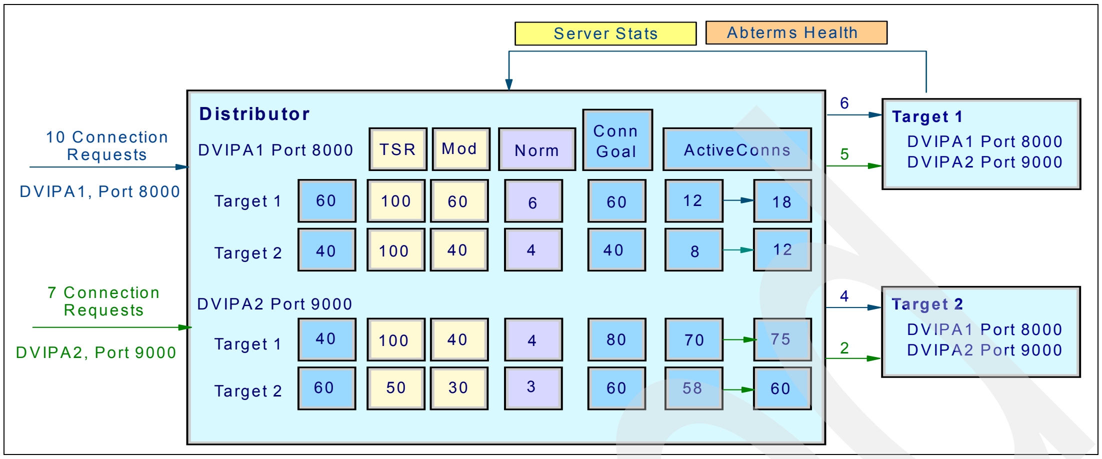


Figure 5-13 WEIGHTEDActive distribution

This distribution is as follows:

- The comparative workload that is desired on each system must be understood so that appropriate connection weights can be configured (fixed weights). Looking at DVIPA2 Port 9000:
  - 40% of the workload is configured to go to Target 1
  - 60% of the workload is configured to go to Target 2
- TSR values and Health Metrics (Abterms, Health) are applied to create a modified weight.

  In the example the TSR of 50% results in a modified weight of 30 for Target 2. (Health Metrics are not shown in this example.)
- Active connection count goals are determined based on the modified connection weights and the active connections on each target (multiple of modified weight > active connections).

  In the example, Target 2's modified weight is 30. The connection goal is 60 (a multiple of 30 and greater than 58). Target 1's weight must be 80 so that the proportions remain the same. New connection goals are determined when the active connection counts for all targets equals the connection goals
- The modified weights are normalized or reduced by dividing by 10.
- Distribution is still weighted round-robin based on the normalized weight, but a target is skipped if a connection goal is reached.

  Because 10 connection requests are received for DVIPA 1 port 8000, six are routed to Target 1, and four are routed to Target 2

  Because seven connection requests are received for DVIPA 2 port 9000, only two requests (instead of three) are routed to Target 2 because the connection goal of 60 is reached. Thus, the other five connection requests are routed to Target 1.

The use of weighted active connections (WEIGHTEDActive) is recommended in the following cases:

- Application scaling concerns

  Target systems vary significantly in terms of capacity (small systems and larger systems). WLM recommendations might favor the larger systems significantly. However, a target application might not scale well to larger systems; because of its design, it might not be able to take full advantage of the additional CPU capacity on the larger systems. This can result in these types of servers getting inflated WLM recommendations when running on larger systems, and getting overloaded with work.

- Unequal number of SHAREPORT servers

  If SHAREPORT is being used, but not all systems have the same number of SHAREPORT server instances (for example, one system has two instances and the other has three). The current round-robin or WLM recommendations do not change distribution based on the number of server instances on each target. Round-robin distribution distributes one connection per target stack regardless of the number of SHAREPORT server instances on that stack. WLM server-specific weights from a target stack with multiple server instances reflect the average weight.

- You need to control the amount of capacity that specific workloads can consume on each system.

  For example, you might need to reserve some capacity on certain systems for batch workloads that are added into selected systems during specific time periods and have specific time window completion requirements. If those systems are also a target for long-running distributed DVIPA connections, WLM recommendations allow that available capacity to be consumed. This can potentially impact the completion times of the batch jobs when they begin to run, if they are not able to displace the existing non-batch workloads. Similarly, the existing connections on that system can experience performance problems if the batch jobs displace those workloads.

WLM routing service enhancements for zAAP and zIIP

The System z platform introduced specialty processors that can be deployed and exploited by qualified workloads on z/OS. This includes support for the following processors:

- System z Application Assist Processor (zAAP)

  These processors can be used for Java™ application workloads on z/OS (including workloads running under z/OS WebSphere Application Server).

- System z Integrated Information Processor (zIIP)

  These processors can be used by qualified z/OS DB2-related workloads and IPSec processing.

When the Sysplex Distributor routes incoming connections based on Workload Manager (WLM) system weights, it returns raw central processor (CP), zAAP, and zIIP weights, considering the available zAAP CPU capacity, the available zIIP CPU capacity, or both. In this case, Sysplex Distributor uses a composite WLM weight that is based on a comparison of displaceable capacity for each processor type, given the importance level and goals of the target servers' service class. WLM returns proportional weights based on the actual usage pattern of the targeted servers for each processor type.

If you need to consider zAAP and zIIP CPU capacity, evaluate whether you can use SERVERWLM distribution as an alternative to BASEWLM distribution for this application:

- SERVERWLM distribution has the advantage that processor proportions are automatically determined and dynamically updated by WLM based on the actual CPU usage of the application.
- BASEWLM distribution for the processor proportions is determined by studying the workload usage of the assist processors (zAAPs and zIIPs). This can be done by analyzing SMF records and using performance monitors reports such as RMF.

WLM recommendations do consider the *cost* of specialty versus conventional processors or zAAp and zIIP processors versus general CPs. Server WLM uses server-specific weights that are based in part on a comparison of displaceable capacity that is available for each processor type, reduced by the actual percentage used of each processor. On previous releases, *crossover cost*—the cost of running zIIP or zAAP intended workloads on the general CP—is not considered.

The solution is to implement a crossover cost parameter, PROCXCOST, on the VIPADISTribute statement. The crossover cost is used to penalize service units that run on the CP instead of the specialty processor. The crossover cost range is between 1 (no crossover penalty) and 100 (heavy crossover penalty).

This configuration parameter can be used to cause WLM to direct more workload targeted for zIIP or zAAP specialty processors to systems that have these more affordable processor cycles available, reducing the overall cost of running those workloads. Similarly, the configuration parameter directs workload that does not require zIIP or zAAP specialty processors to systems with the least amount of zIIP or zAAP processing crossover.

Figure 5-14 shows how to configure the PROCXCOST parameter on the VIPADistribute statement.

```
>>-VIPADYNAMIC---------------------------------------------------------------------->

>>-VIPADEFINE-----+-Define parameters-+------+-address mask--+------+-ipv4 addr-+----------->
                                             +-ipv6 intfname-+      +-ipv6 addr-+

|--VIPADISTRIBUTE----+-Dist parameters-+----DISTMethod SERVERWLM----+-ServerWLMOptions-+---->

|-----------------------+-ipv4 addr-+---------+-PORT num-+------------+-DestIP-+----------->
                        +-ipv6 intf-+

|--ENDVIPADynamic--|

ServerWLMoptions:
  +-PROCXCOST---- ZAAP 1 ZIIP 1-+
|--+                             +---------------------------------------------------------->
  |           +-ZAAP 1-+       |
  +-PROCXCOST---+       ++----+
              | +-ZAAP x-+ |
              |            |
              | +-ZIIP 1-+ |
              ++         ++
                +-ZIIP y-+
```

Figure 5-14 PROCXCOST parameter on VIPADISTRIBUTE statement

> **Warning:** A high crossover cost can cause systems that have constrained specialty processor service units to receive a significantly lower percentage of the overall workload. This lower percentage can have an adverse effect on performance. Run the RMF Workload Activity Report before and after changing this parameter to understand how this change can affect performance.

This support is based on the METHOD=EQUICPU function in the IWM4SRSC WLM service.

> **Restriction:** These new composite weights are returned by WLM only if all systems in the sysplex are V1R11 or later, regardless of whether the systems participate in workload distribution or whether systems are members of different subplexes.
>
> In a mixed-release environment, CP weights are used only to determine the WLM System or server-specific weight because there is no information available about zAAP and zIIP capacity from older releases.
>
> To make use of the solution implemented by the PROCXCOST parameter on the VIPADISTribute statement, all systems must be running on V1R11.
>
> Load Balancing Advisor (LBA) considers zAAP and zIIP weight recommendations for server members but not system members. WLM recommendations for system members consider only CP capacity.

ILWEIGHTING parameter

WLM supports a method to calculate relative weights of the target servers to which Sysplex Distributor distributes incoming connections. WLM considers the relative importance of existing work that is displaced on each target as new work is distributed. Target systems with more lower importance displaceable work are favored over systems with higher importance displaceable work. The ILWEIGHTING parameter is configured on a VIPADISTRIBUTE statement, as shown in Figure 5-15.

```
>>-VIPADYNAMIC---------------------------------------------------------------------->

>>-VIPADEFINE-----+-Define parameters-+------+-address mask--+-----+-ipv4 addr-+---------->
                                             +-ipv6 intfname-+     +-ipv6 addr-+

|--VIPADISTRIBUTE----+-Dist parameters-+----DISTMethod SERVERWLM----+-ServerWLMOptions-+---->

|-------------------+-ipv4 addr-+----------+-PORT num-+------------+-DestIP-+----------->
                    +-ipv6 intf-+

|--ENDVIPADynamic--|

ServerWLMoptions:
  +-PROCXCOST---- ZAAP 1 ZIIP 1-+           +-ILWEIGHTING 0-+
|--+                           +-------------+           +---------------------->
  |          +-ZAAP 1-+       |           +-ILWEIGHTING 1-+
  +-PROCXCOST---+      ++----+                          2-+
             | +-ZAAP x-+ |                          3-+
             |           |
             | +-ZIIP 1-+ |
             ++      ++
               +-ZIIP y-+
```

Figure 5-15 ILWEIGHTING parameter on VIPADISTRIBUTE statement

The ILWEIGHTING parameter value indicates how aggressively WLM should favor systems with displaceable capacity at low importance levels over systems with displaceable capacity at high importance levels. The higher the value specified for ILWEIGHTING, the more a stack with displaceable capacity at lower importance levels is favored.

The following importance level (IL) weighting factors are supported:

IL 0 Ignore importance levels when comparing system displaceable capacity, which is the default.

IL 1 Moderate, specifies that WLM weigh displaceable capacity at each successively lower importance level slightly higher than the capacity at the preceding higher importance level. Adjusts the service units by the square root of the importance level difference of the new work and the Displaceable Service Units + 1.

IL 2 Aggressive, specifies that WLM weigh displaceable capacity at each successively lower importance level significantly higher than the capacity at the preceding higher importance level. Adjusts the service units by the importance level difference of the new work and the Displaceable Service Units + 1.

IL 3 Exceptionally Aggressive, adjusts the service units by the square of the importance level difference of the new work and the Displaceable Service Units + 1.

Consider the following notes:

- This parameter is available only for the SERVERWLM distribution method with METHOD=EQUICPU.
- It can be useful in environments where the target system workload mix varies significantly.
- The impact can be larger when LPARs differ significantly in their capacities.
- Some level of differentiation of the importance levels is generally desirable.
- The levels that can be specified are 0 (not used), 1(moderate), 2 (aggressive), and 3 (very aggressive). We recommend starting with the moderate level of 1 and then increasing gradually if you find that the result is beneficial.

> **Warning:** Using an IL value of 2 or greater can cause systems with comparatively higher importance displaceable workloads to receive a significantly lower percentage of the new workload, which can have an adverse effect on performance. As a guideline, use a Moderate (IL 1) value initially. Run the RMF Workload Activity Report before and after a change to understand how this setting affects performance.

WLM polling interval

The timings used in TCP/IP and WLM can influence the behavior of workload balancing by the Sysplex Distributor. Using the default settings, the TCP/IP stacks poll WLM every 60 seconds for weight information as follows:

- For BASEWLM, the distributor polls WLM for *system* weights
- For SERVERWLM, each target polls WLM for *server-specific* weights

WLM calculates new weights as follows:

- Calculates based on a comparison of the last 10 seconds of CPU utilization on registered sysplex systems or servers.
- Maintains a 3-minute rolling average of these calculations. The 3-minute rolling average is used to smooth the result.
- The average is returned by WLM in response to the request from the TCP/IP stack.

The default TCP/IP polling interval assumes weights will not change significantly from minute to minute, which is not always true, especially with many short living connections or high CPU utilization. Therefore, you can change the WLM polling interval using the SYSPLEXWLMPOLL option on the GLOBALCONFIG parameter in the TCP/IP profile.

Keep in mind that WLM makes its policy adjustment calculations every 10 seconds. Therefore, you should not specify the value too small. In addition, because it is a global parameter, it influences all specified Sysplex Distributors that operate on the stack.

For detailed information about the WLM services used, see *z/OS MVS Programming: Workload management services*, SA22-7619.

Setting up these functions

This section explains how to set up the SERVERWLM, BASEWLM, and SHAREPORT functions.

SERVERWLM

Enable the Sysplex Distributor to consider zIIP or zAAP processor capacity for target servers. Apply a crossover cost to penalize specialty processor service units that run on the CP processor and its work importance level weight. Example 5-4 shows an extract of the profile TCP/IP.

Example 5-4 TCP/IP definition for SERVERWLM

```
VIPADYNAMIC

   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.23
   VIPADISTribute DISTMethod SERVERWLM
   PROCxCOST ZIIP 20 ZAAP 100 1 ilweighting 1
                          10.1.8.23 PORT 8080 8081 8082
                          DESTIP all

ENDVIPADYNAMIC
```

In the example, a crossover cost of 100 **1** is applied on the CP service units that run as zAAP intended workload runs on the CP.

We have a JAVA workload running at an importance level of 2. Starting with z/OS V1R9, WLM also takes into account the importance level of the new transaction. If you need more information about z/OS WLM, see *z/OS V1R11 MVS Planning Workload Management*, SA22-7602.

Our JAVA workload is designed to consume CP SUs 10% and zAAP SUs 90%, and both LPARs (SC30 and SC31) have the same displaceable capacity of 1000 SUs available.[1]

[1] A service unit (SU) is the amount of processor capacity that is consumed to execute a certain amount of work. You can use SU as a common comparison metric throughout all of our systems, regardless of release or capacity.

Figure 5-16 shows the results of our test runs on SC30 and SC31.

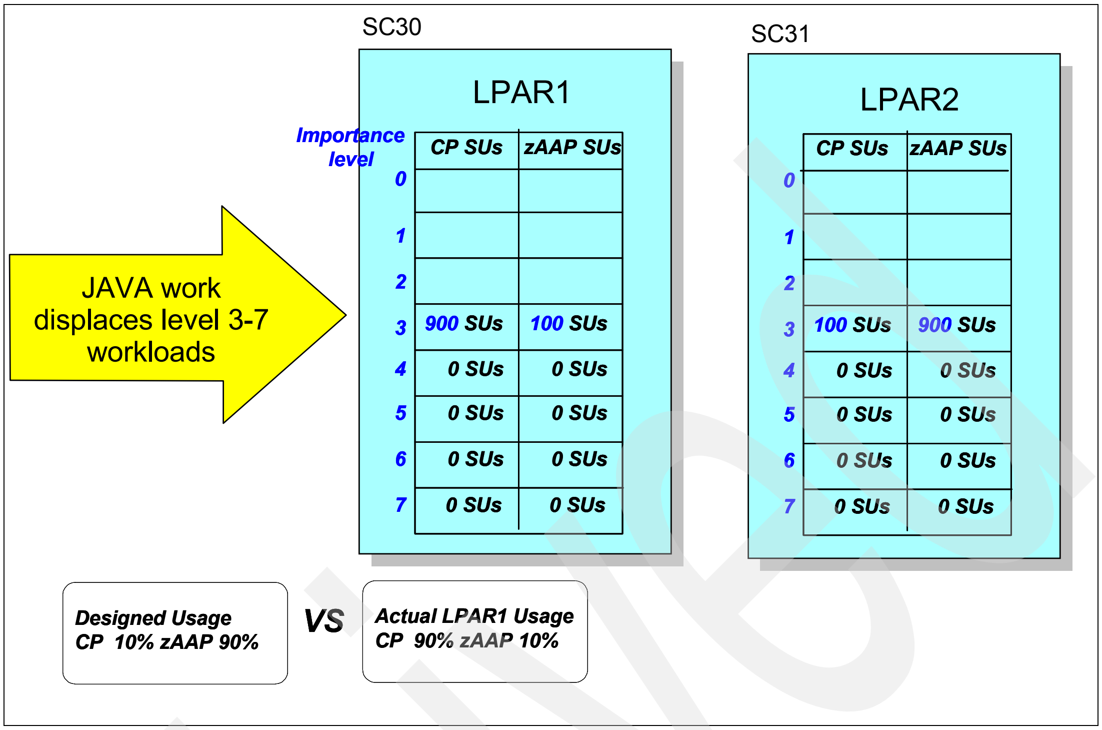


Figure 5-16 Cost of specialty versus conventional processors

In our example, SC30 has 1000 general CP SUs available and 100 zAAP SUs available. The application is designed to run consuming 10% CP SUs as it consumes 90% zAAP SUs.

- Step 1: Consume all of the zAAP SUs, using the designed 10/90 split.

  Because 111 SUs are consumed by the application, 10% of those 111 SUs, or 11 SUs, should run on the CP and 90% of those 111, or 100 SUs, should run on the zAAP, thus consuming all of the zAAP SUs.

  So, after step 1, the equivalent CP weight is 11.

- Step 2: Compute the remaining equivalent CP weight.

  We are left with 889 CP SUs, because the application consumes the remaining 889 SUs. Of those CP SUs consumed, 10% of the 889 CP SUs are intended to run on the CP (89 SUs), but the remaining 90% of the 889 CP SUs (800 SUs) were intended to run on the zAAP. The remaining 89 CP SUs are reduced by the crossover cost, which is the penalty imposed because all 89 CP SUs run at the expense of the remaining 800 SUs, crossing over and running on the CP instead of running on the zAAP processor.

Thus, if the crossover cost is *1*, then there is no penalty applied against the 89 SUs. The equivalent CP weight is as follows:

```
11 + ( 89 / (10% + crossover cost of 1 *  90%)
11 + 89 / (.1 + 1 * .9)
11 + 89/1
100
```

However, if the crossover cost is 100, the highest penalty possible, then the 89 SUs are reduced to 1 as follows:

```
11 + 89/(10% + 100 * 90%)
11 + 1
12
```

See Figure 5-17.

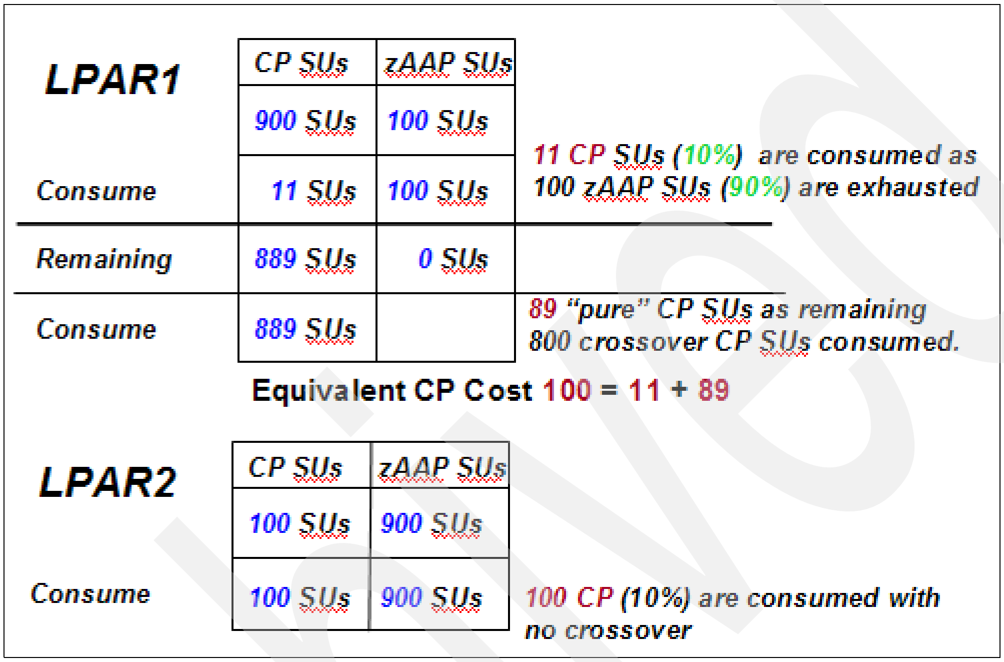


Figure 5-17 Using crossover cost of 100

For LPAR2, as 100 CP SUs run, the designed percentage of 900 zAAP SUs run, so there is no penalty for the 100 CP SUs.

The following examples show the configuration without our JAVA transaction running.

Example 5-5 shows the output of the D TCPIP,TCPIPC,N,VIPADCFG,DETAIL command.

Example 5-5   Output of NETSTAT VIPADCFG command

```
D TCPIP,TCPIPA,N,VIPADCFG,DETAIL
DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.23/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG:
  VIPA DISTRIBUTE:
    DEST:      10.1.8.23..8080
      DESTXCF: ALL
      SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
      OPTLOC:  NO
      PROCXCOST:
        ZAAP: 100  ZIIP: 020
      ILWEIGHTING: 1
    DEST:      10.1.8.23..8081
      DESTXCF: ALL
      SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
      OPTLOC:  NO
      PROCXCOST:
        ZAAP: 100  ZIIP: 020
      ILWEIGHTING: 1
    DEST:      10.1.8.23..8082
      DESTXCF: ALL
      SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
      OPTLOC:  NO
      PROCXCOST:
        ZAAP: 100  ZIIP: 020
      ILWEIGHTING: 1
END OF THE REPORT NG: 1
```

Example 5-6 shows the output of the D TCPIP,TCPIPC,N,VDPT,DETAIL command.

Example 5-6 Output of NETSTAT VDPT (SERVERWLM) command

```
D TCPIP,TCPIPA,N,VDPT,DETAIL
DYNAMIC VIPA DESTINATION PORT TABLE FOR TCP/IP STACKS:
DEST:        10.1.8.23..8080
  DESTXCF:   10.1.7.11
  TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 00
      RAW         CP: 00 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 00 ZAAP: 00 ZIIP: 00
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 00
DEST:        10.1.8.23..8081
  DESTXCF:   10.1.7.11
  TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 00
      RAW         CP: 00 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 00 ZAAP: 00 ZIIP: 00
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 00
DEST:        10.1.8.23..8082
  DESTXCF:   10.1.7.11
  TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 00
      RAW         CP: 00 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 00 ZAAP: 00 ZIIP: 00
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 00
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

In this case we are using CP only; there is no application using zAAP or zIIP processor.

For more information about the NETSTAT command, refer to *IP System Administrator's Commands*, SC31-8781.

BASEWLM

Enable the Sysplex Distributor to consider zIIP or zAAP processor capacity for target servers. Example 5-7 shows an extract of the profile TCT/IP.

Example 5-7 TCP/IP definition for BASEWLM

```
VIPADYNAMIC

   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.21
   VIPADISTRIBUTE DISTMETHOD BASEWLM
   PROCTYPE CP 20 ZAAP 40 ZIIP 40 10.1.8.21
   PORT 20 21
   DESTIP ALL

   VIPADEFINE     MOVE IMMED 255.255.255.0 10.1.8.22
   VIPADISTRIBUTE DISTMETHOD BASEWLM
   PROCTYPE CP 20 ZAAP 40 ZIIP 40 10.1.8.22
   PORT 28538 28518 28510 28502 28503
   DESTIP ALL

ENDVIPADYNAMIC
```

The PROCTYPE parameter is used in conjunction with BASEWLM and sets the weight for each kind of processor.

Example 5-8 shows the output of the D TCPIP,TCPIPC,N,VIPADCFG,DETAIL command.

Example 5-8 Output of NETSTAT VIPADCFG (BASEWLM) command

```
DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.21/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG:
    IPADDR/PREFIXLEN: 10.1.8.22/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG:
  VIPA DISTRIBUTE:
    DEST:        10.1.8.21..20
      DESTXCF:   ALL
      SYSPT:     NO   TIMAFF: NO    FLG: BASEWLM
        OPTLOC:  NO
        PROCTYPE:
          CP: 20  ZAAP: 40  ZIIP: 40
    DEST:        10.1.8.21..21
      DESTXCF:   ALL
      SYSPT:     NO   TIMAFF: NO    FLG: BASEWLM
        OPTLOC:  NO
        PROCTYPE:
          CP: 20  ZAAP: 40  ZIIP: 40
    DEST:        10.1.8.22..28502
      DESTXCF:   ALL
      SYSPT:     NO   TIMAFF: NO    FLG: BASEWLM
        OPTLOC:  NO
        PROCTYPE:
          CP: 20  ZAAP: 40  ZIIP: 40
    DEST:        10.1.8.22..28503
      DESTXCF:   ALL
      SYSPT:     NO   TIMAFF: NO    FLG: BASEWLM
        OPTLOC:  NO
          CP: 20  ZAAP: 40  ZIIP: 40
    DEST:        10.1.8.22..28510
      DESTXCF:   ALL
      SYSPT:     NO   TIMAFF: NO    FLG: BASEWLM
        OPTLOC:  NO
        PROCTYPE:
          CP: 20  ZAAP: 40  ZIIP: 40
    DEST:        10.1.8.22..28518
      DESTXCF:   ALL
      SYSPT:     NO   TIMAFF: NO    FLG: BASEWLM
        OPTLOC:  NO
        PROCTYPE:
          CP: 20  ZAAP: 40  ZIIP: 40
    DEST:        10.1.8.22..28538
      DESTXCF:   ALL
      SYSPT:     NO   TIMAFF: NO    FLG: BASEWLM
        OPTLOC:  NO
        PROCTYPE:
          CP: 20  ZAAP: 40  ZIIP: 40
END OF THE REPORT
```

Example 5-9 shows the output of the D TCPIP,TCPIPC,N,VDPT,DETAIL command.

Example 5-9 Output of NETSTAT VDPT (BASEWLM) command

```
DYNAMIC VIPA DESTINATION PORT TABLE FOR TCP/IP STACKS:
DEST:         10.1.8.21..20
  DESTXCF:    10.1.7.12
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW         CP: 05 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:         10.1.8.21..20
  DESTXCF:    10.1.7.31
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
   FLG: BASEWLM
     TCSR: 100  CER: 100 SEF: 100
     WEIGHT: 01
       RAW         CP: 08 ZAAP: 00 ZIIP: 00
       PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
     ACTCONN:   0000000000
     QOSPLCACT: *DEFAULT*
       W/Q: 01
 DEST:         10.1.8.21..20
   DESTXCF:    10.30.20.101
   TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
   FLG: BASEWLM
     TCSR: 100  CER: 100 SEF: 100
     WEIGHT: 01
       RAW         CP: 05 ZAAP: 00 ZIIP: 00
       PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
     ACTCONN:   0000000000
     QOSPLCACT: *DEFAULT*
       W/Q: 01
 DEST:         10.1.8.21..21
   DESTXCF:    10.1.7.12
   TOTALCONN: 0000000000  RDY: 001  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW         CP: 05 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:         10.1.8.21..21
  DESTXCF:    10.1.7.31
  TOTALCONN: 0000000000  RDY: 001  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW         CP: 08 ZAAP: 00 ZIIP: 00
```

```
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.21..21
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000000000  RDY: 001  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 05 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28502
  DESTXCF:   10.1.7.12
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 05 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28502
  DESTXCF:   10.1.7.31
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 08 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28502
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 05 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28503
  DESTXCF:   10.1.7.12
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 05 ZAAP: 00 ZIIP: 00
```

```
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28503
  DESTXCF:   10.1.7.31
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 08 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28503
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 05 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28510
  DESTXCF:   10.1.7.12
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 05 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28510
  DESTXCF:   10.1.7.31
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 08 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:        10.1.8.22..28510
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 05 ZAAP: 00 ZIIP: 00
```

```
        PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
      ACTCONN:   0000000000
      QOSPLCACT: *DEFAULT*
        W/Q: 01
  DEST:        10.1.8.22..28518
    DESTXCF:   10.1.7.12
    TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
    FLG: BASEWLM
      TCSR: 100  CER: 100 SEF: 100
      WEIGHT: 01
        RAW          CP: 05 ZAAP: 00 ZIIP: 00
        PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
      ACTCONN:   0000000000
      QOSPLCACT: *DEFAULT*
        W/Q: 01
  DEST:        10.1.8.22..28518
    DESTXCF:   10.1.7.31
    TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
    FLG: BASEWLM
      TCSR: 100  CER: 100 SEF: 100
      WEIGHT: 01
        RAW          CP: 08 ZAAP: 00 ZIIP: 00
        PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
      ACTCONN:   0000000000
      QOSPLCACT: *DEFAULT*
        W/Q: 01
  DEST:        10.1.8.22..28518
    DESTXCF:   10.30.20.101
    TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
    FLG: BASEWLM
      TCSR: 100  CER: 100 SEF: 100
      WEIGHT: 01
        RAW          CP: 05 ZAAP: 00 ZIIP: 00
        PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
      ACTCONN:   0000000000
      QOSPLCACT: *DEFAULT*
        W/Q: 01
  DEST:        10.1.8.22..28538
    DESTXCF:   10.1.7.12
    TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
    FLG: BASEWLM
      TCSR: 100  CER: 100 SEF: 100
      WEIGHT: 01
        RAW          CP: 05 ZAAP: 00 ZIIP: 00
        PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
      ACTCONN:   0000000000
      QOSPLCACT: *DEFAULT*
        W/Q: 01
  DEST:        10.1.8.22..28538
    DESTXCF:   10.1.7.31
    TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
     FLG: BASEWLM
       TCSR: 100  CER: 100 SEF: 100
       WEIGHT: 01
         RAW          CP: 08 ZAAP: 00 ZIIP: 00
```

```
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
DEST:         10.1.8.22..28538
  DESTXCF:    10.30.20.101
  TOTALCONN: 0000000000  RDY: 000  WLM: 01  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 01
      RAW          CP: 05 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 01 ZAAP: 00 ZIIP: 00
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 01
```

Each port can be reached using the destination address through the XCF address.

Sysplex autonomics health monitor for target stacks

When the Sysplex Distributor distributes connections to servers on a set of target stacks, there can be cases where a server's ability to process new connections is slow due to external issues.

z/OS has a function that allows the Sysplex Distributor to monitor the target server's responsiveness at intervals of approximately one minute. If a target server experiences response problems, Sysplex Distributor diverts new connection requests away from that server. When the distribution method for a target server is BASEWLM or SERVERWLM, the weights to the target server are modified to reflect its ability to handle new connection requests. Target servers whose weights are lowered because of this are less favored by the distribution algorithm.

> **Note:** When the distribution method for a target server is ROUNDROBIN and the target server appears to have lost its ability to process new connection setup requests, it will not be selected as part of the round-robin distribution.

Figure 5-18 illustrates the use of the enhanced sysplex autonomics health monitor function.

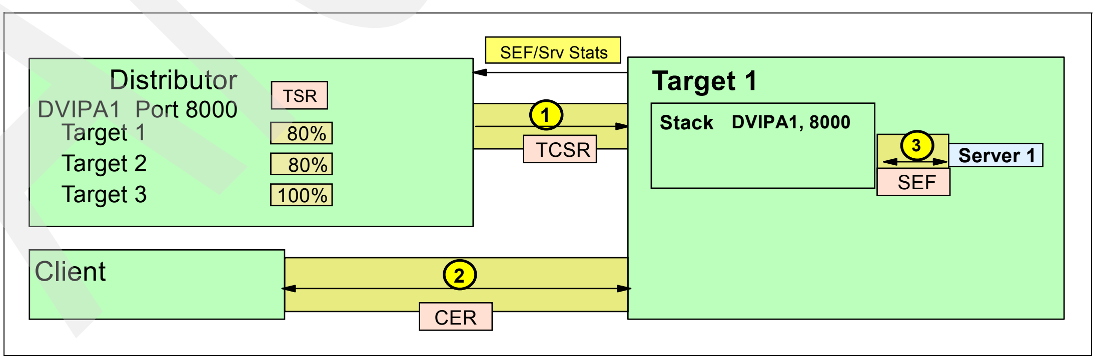


Figure 5-18 Possible weak points, monitored automatically by the sysplex autonomics health monitor

We can monitor the three potential weak points shown in Figure 5-18:

1. Target Connectivity Success Rate (TCSR)

   We monitor the connectivity between the distributing stack and the target stack, to ensure that all new connection requests reach the target. A value of 100 for this function indicates there are no problems in this area.

   > **Note:** If the TCSR drops to 25 or lower, the optimized routing function is triggered to perform a new route lookup.

2. Connection Establishment Rate (CER)

   We monitor the network connectivity between the server and the client, to make sure that all new connections are established successfully. A returned value of 100 indicates that there is no problem.

3. Server accept Efficiency Fraction (SEF)

   This value provides information about the responsiveness of the target server (for example, is the server accepting new work?).

All these values are sent to the distributor. The Sysplex Distributor then creates a Target Server Responsiveness fraction (TSR). This fraction, in turn, influences the decisions made by the Sysplex Distributor. The higher the value, the healthier the condition of the target server.

The Sysplex Autonomic Health indicators are shown in various display messages issued by TCP/IP. All have a value ranging from 0 to 100 (with 0 being the worst and 100 being the best, except for the QoS, which is the other way around).

We can check Sysplex Autonomic Health indicators in an extract of the output from the command D TCPIP,TCIPIP,VPD,DETAIL command, as shown in Example 5-10. The fields are self-explanatory, except the QoS fraction which is shown as `W/Q` in the message display.

Example 5-10 Part of output of NETSTAT VDPT

```
DEST:        10.1.8.30..23
  DESTXCF:   10.1.7.21
  TOTALCONN: 0000000000  RDY: 001  WLM: 16  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100  SEF: 100
    WEIGHT: 64
      RAW          CP: 64 ZAAP: 00 ZIIP: 00
      PROPORTIONAL CP: 64 ZAAP: 00 ZIIP: 00
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 16
```

5.1.6 Unreachable DVIPA detection and recovery

Sysplex autonomics have been enhanced to enable you to specify key network interfaces that should be monitored by the TCP/IP stacks. You can monitor interfaces to ensure that they are functioning and that dynamic routes are known over these interfaces, as shown in Figure 5-19. If a failure occurs on all specified interfaces, sysplex autonomics will trigger other TCP/IP stacks in the sysplex to take over responsibilities for the DVIPAs. This function can be very useful for TCP/IP stacks that own and advertise DVIPAs, and also for TCP/IP stacks that act as backup owners for these DVIPAs.

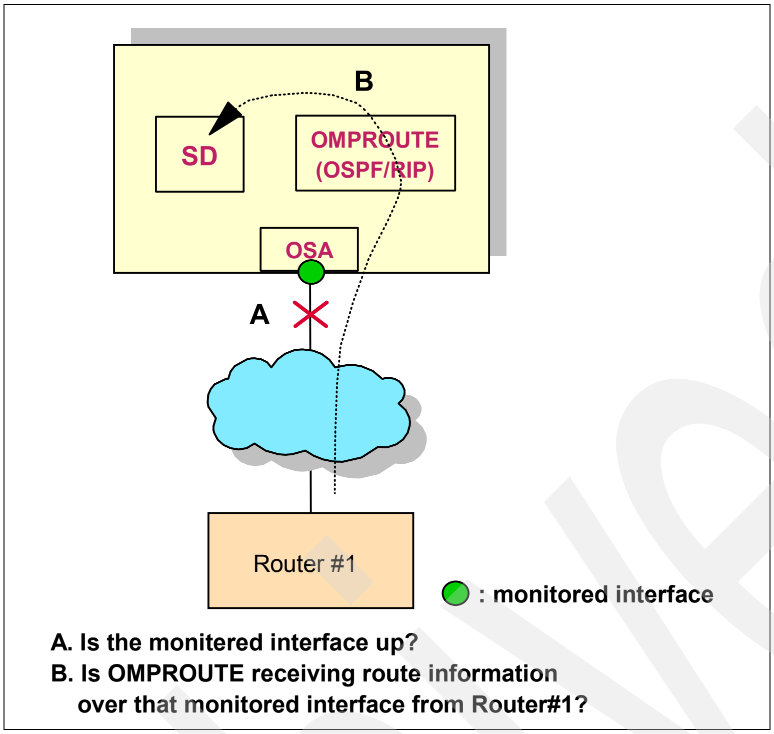


Figure 5-19 Sysplex Autonomics detection of unreachable DVIPA

The SYSPLEMONITOR statement and the LINK statement include parameters which are explained here:

- MONINTERFACE

  Request the TCP/IP stack to monitor the status of specified interfaces. The interfaces being monitored are those that are configured with the MONSYSPLEX parameter on the LINK statement.

- NOMONINTERFACE

  Request the TCP/IP stack to not monitor the status of any interface. This is the default.

- DYNROUTE

  Request the TCP/IP stack to monitor the presence of dynamic routes over monitored interfaces. This level of monitoring is useful in detecting problems that OMPROUTE is having in communicating with other routing daemons on the selected interfaces. If no dynamic routes are present in the TCP/IP stack, even though the monitored interface is active, this provides an indicator that client requests will not be able to reach this TCP/IP stack over that interface. This is the default when MONINTERFACE is configured.

- NODYNROUTE

  Request the TCP/IP stack not to monitor the presence of dynamic routes over the monitored interfaces. This is the default when the MONINTERFACE parameter is not configured.

- MONSYSPLEX

  This indicates that the status of this link is monitored by the sysplex autonomics. This parameter is not in effect unless the MONINTERFACE parameter is coded on the GLOBALCONFIG SYSPLEXMONITOR profile statement. This parameter is dynamically modifiable.

- NOMONSYSPLEX

  This indicates that the status of this link is not monitored by sysplex autonomics. This is the default and is dynamically modifiable.

- TIMERSECS

  The TIMERSECS value which is specified on the GLOBAL CONFIG SYSPLEXMONITOR statement is used to determine how quickly sysplex autonomics will react when a problem is detected. The default value is 60 seconds.

For configuration and verification examples, refer to 5.2.6, “Sysplex Distributor using Unreachable DVIPA Detection and Recovery” on page 155.

5.2 Design and implementation examples

This section contains the following scenarios, which demonstrate Sysplex Distributor usage:

- Sysplex Distributor using server-specific WLM
- Sysplex Distributor using BASEWLM
- Sysplex Distributor using round-robin
- Sysplex Distributor using WEIGHTEDActive method
- Portsharing using SHAREPORTWLM
- Sysplex Distributor using Unreachable DVIPA Detection and Recovery

5.2.1 Sysplex Distributor using server-specific WLM

Server-specific Workload Manager (WLM) controls for load balancing is a sysplex distribution feature in z/OS. The key PROFILE statements are shown in Example 5-11.

Example 5-11 Using ServerWLM distributed method

```
VIPADynamic
   VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.1.8.10
   VIPADISTribute DISTMethod SERVERWLM 10.1.8.10
    PORT 23 DESTIP 10.1.7.11 10.1.7.21 10.1.7.31
```

Using this function, WLM assigns a weight based on:

- How well a server is meeting the goals of its service class
- The displaceable capacity for new work based on the importance of its service class

Figure 5-20 illustrates how incoming TN3270 connections are load balanced using server-specific WLM weights.

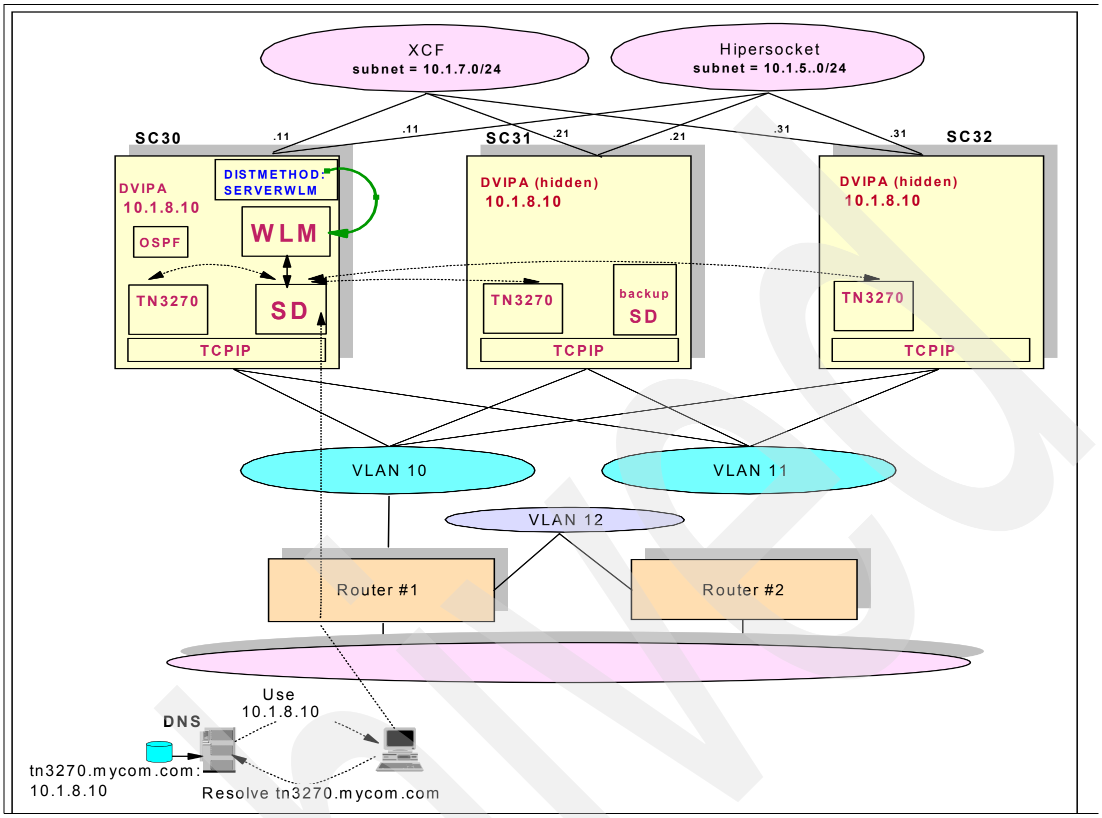


Figure 5-20 Server-specific WLM Workload balancing of TN3270 connections

Considerations when using server-specific WLM method

WLM server-specific recommendations can be used only if the Sysplex Distributor and all target stacks for a distributed DVIPA port are V1R7 or later.

The older BASEWLM system weight recommendations are not compatible with the latest WLM server recommendations.

Implementing the server-specific WLM method

To implement the Sysplex Distributor function, you must define a matching set of VIPADEFINE, VIPABACKUP, and VIPADISTRIBUTE statements. The TCP/IP configuration required is minimal compared to other dispatching mechanisms. For the most part, the standard sysplex environment enables the dynamic establishment of links and the dynamic coordination between stacks in the cluster.

The following specific implementation tasks are involved:

- Choose which IP stack will execute the Sysplex Distributor distributing function. We chose LPAR SC30.
- Select which IP stack will be the backup stack for the Sysplex Distributor stack, and in which order. We selected SC31 to be the first backup and SC32 to be second backup.
- Ensure that WLM GOAL mode is enabled in all the LPARs participating in the Sysplex Distributor. We used the command D WLM,SYSTEMS to ensure that all the LPARs participating in the Sysplex Distributor were running in GOAL mode.

- Enable sysplex routing in all the IP stacks participating in the Sysplex Distributor using the SYSPLEXROUTING statement.
- For IP stacks that are active under a multistack environment, the SAMEHOST links must be created dynamically. In general, code DYNAMICXCF in all the IP stacks participating in the Sysplex Distributor.

> **Note:** For Sysplex Distributor, you cannot specify the XCF address using the IUTSAMEH DEVICE, LINK, and HOME statements. XCF addresses must be allowed to be dynamically created by coding the IPCONFIG DYNAMICXCF statement.

- Select, by port numbers, the applications that are to be managed by the Sysplex Distributor function. Note that if the application chosen requires data and control ports (like FTP), then both ports must be considered. We selected port 23 to distribute TN3270 services.
- Code the VIPADYNAMIC/ENDVIPADYNAMIC block for the distributing IP stack.
- Define the dynamic VIPA for the distributing IP stack with the VIPADEFINE statement. We selected 10.1.8.10 as the port address.
- Associate the sysplex Dynamic VIPA with the application's port number by using the VIPADISTRIBUTE statement.
- Code the VIPADYNAMIC/ENDVIPADYNAMIC block for the distributor's backup IP stack.
- Define the IP stack as backup for the sysplex DVIPA address with the VIPABACKUP statement. We defined SC31 and SC32 to be the VIPABACKUP.

Configuring the server-specific WLM method

Example 5-12 shows the most important parameters of our TCP/IP profile that we needed to define in order to enable the distribution of TN3270 connections.

Example 5-12 Profile TCPIPA SC30 Sysplex Distributor

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING                  3
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
DYNAMICXCF 10.1.7.11 255.255.255.0 8
VIPADynamic
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.1.8.10 1
 VIPADISTribute DISTMethod SERVERWLM 10.1.8.10 2
  PORT 23 DESTIP 10.1.7.11 10.1.7.21 10.1.7.31
 VIPAROUTE 10.1.7.11 10.1.1.10 4
 VIPAROUTE 10.1.7.21 10.1.1.20 4
 VIPAROUTE 10.1.7.31 10.1.1.30 4
ENDVIPADYNAMIC
```

The following are important considerations for this PROFILE:

1. We used the IP address 10.1.8.10 as DVIPA for the TN3270 servers.
2. We used the VIPADIStribute parameter DISTMethod SERVERWLM.
3. We coded NODATAGRAMFWD to disable general IP forwarding of this stack.
4. We defined three VIPAROUTE statements, each pointing to one of our target systems.

Note: VIPAROUTE allows us to relieve our XCF infrastructure by using any available IP interface for Sysplex Distributor-related forwarding traffic.

The relevant PROFILE statements for SC31 and SC32 are shown in Example 5-13 and Example 5-14. These define SC31 and SC32 to act as backup Sysplex Distributor in case the primary Sysplex Distributor on SC30 fails. Example 5-13 shows the SC31 target and backup.

Example 5-13 Profile TCPIPB SC31 target and backup

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
DYNAMICXCF 10.1.7.21 255.255.255.0
VIPADynamic
 VIPABACKUP 2 MOVEABLE IMMEDIATE 255.255.255.0 10.1.8.10 1
ENDVIPADynamic
```

Example 5-14 shows the SC32 target and backup.

Example 5-14 Profile TCPIPC SC32 target and backup

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
DYNAMICXCF 10.1.7.30 255.255.255.0
 VIPADynamic
 VIPABACKUP 1 MOVEABLE IMMEDIATE 255.255.255.0 10.1.8.10 2
 ENDVIPADynamic
```

Important points here are:

1. If the Sysplex Distributor on SC30 fails, SC31 will take over as the first backup distributor.

2. If the Sysplex Distributor on SC30 fails and SC31 is *not* active, SC32 will take over as the second backup distributor.

Verifying the server-specific WLM method implementation

We followed these steps to verify that our implementation was correct:

1. Verification of the general Dynamic VIPA setup, including the backup Sysplex Distributor
2. Verification that server-specific WLM is set up correctly and is used
3. Verification that optimized routing using VIPAROUTE is set up correctly and is used

After all of our IP stacks were active and running, we started our verification steps.

Issuing a NETSTAT,VIPADCFG command, as shown in Example 5-15, on our distribution stack (SC30) provided important information that helped us to verify our Sysplex Distributor configuration.

Example 5-15 Display results of the VIPADCFG command issued on distribution stack SC30

```
SC30
D TCPIP,TCPIPA,N,VIPADCFG
DYNAMIC VIPA INFORMATION:
   VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.10/24                1
      MOVEABLE: IMMEDIATE  SRVMGR: NO
   VIPA DISTRIBUTE:
   DEST:        10.1.8.10..23                   2 3
     DESTXCF:   10.1.7.11                         4
       SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM 5
   DEST:        10.1.8.10..23
     DESTXCF:   10.1.7.21                         4
       SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM 5
   DEST:        10.1.8.10
     DESTXCF:   10.1.7.31                         4
       SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM 5
   VIPA ROUTE:                   6
     DESTXCF:     10.1.7.11
       TARGETIP:  10.1.1.10
     DESTXCF:     10.1.7.21
       TARGETIP:  10.1.1.20
     DESTXCF:     10.1.7.31
       TARGETIP:  10.1.1.30
 END OF THE REPORT
```

This display includes the following key information:

1. This is the DVIPA address representing the TN3270 servers.
2. TN3270 connections are being distributed by the Sysplex Distributor.
3. We use port number 23 for TN3270.
4. This is the XCF IP address assigned and used.
5. FLG: SERVERWLM indicates that server-specific WLM is active and will be used.
6. VIPAROUTE indicates that we do not use XCF links but use any available IP network interface for forwarding Sysplex Distributor-related packets. Three VIPAROUTE definitions point to the static VIPA in each of our stacks.

Example 5-16 shows the results of VIPADCFG commands for SC31 and SC32.

Example 5-16 Display results of the VIPADCFG command issued on SC31 and SC32

```
SC31
DYNAMIC VIPA INFORMATION:
    VIPA BACKUP:
      IPADDR/PREFIXLEN: 10.1.8.10
     1 RANK: 002  MOVEABLE:           SRVMGR:
END OF THE REPORT

SC32
DYNAMIC VIPA INFORMATION:
   VIPA BACKUP:
     IPADDR/PREFIXLEN: 10.1.8.10
    2 RANK: 001  MOVEABLE:           SRVMGR:
END OF THE REPORT
```

The output from our IP stack on SC31 shows that this stack is the primary backup 1 with rank 2. The output from the IP stack on SC32 shows that it is the secondary backup 2 with rank 1.

A SYSPLEX,VIPADYN command, as shown in Example 5-17, provides concise information about all the stacks participating in the sysplex.

Example 5-17 Display of SYSPLEX,VIPADYN issued on distribution stack SC30

```
-D TCPIP,TCPIPA,SYSPLEX,VIPADYN
 EZZ8260I SYSPLEX CS V1R11 502
 VIPA DYNAMIC DISPLAY FROM TCPIPA   AT SC30

LINKNAME: VIPL0A01080A
IPADDR/PREFIXLEN: 10.1.8.10/24
   ORIGIN: VIPADEFINE
   TCPNAME  MVSNAME  STATUS  RANK  DIST
   -------- -------- ------ ---- ----
   TCPIPA   SC30     ACTIVE 1    BOTH  3
   TCPIPB   SC31     BACKUP 1 002 2 DEST  3
   TCPIPC   SC32     BACKUP 1 001 2 DEST  3

 10 OF 10 RECORDS DISPLAYED
```

This output provides the following important information:

1. Actual status of the TCP/IP stack.
2. Rank value determines the order on which a backup distributor will take over if the primary Sysplex Distributor fails.
3. This indicates whether a stack is defined as the distributor, a destination, or both.

We issued NETSTAT,HOME commands on all our stacks to obtain a list of all known IP addresses.

Example 5-18 contains the output of these commands and 1 marks the DVIPA address representing the TN3270 servers.

Example 5-18 Display results of Netstat,Home in all stacks

```
SC30
D TCPIP,TCPIPA,N,HOME
HOME ADDRESS LIST:
LINKNAME:   VIPA1L
  ADDRESS:  10.1.1.10
.
.
.
LINKNAME:   VIPLOA01080A
ADDRESS:  10.1.8.10 1
     FLAGS
12 OF 12 RECORDS DISPLAYED
 END OF THE REPORT

SC31
D TCPIP,TCPIPB,N,HOME
HOME ADDRESS LIST:
 LINKNAME:   VIPA1L
   ADDRESS:  10.1.1.20
     FLAGS:  PRIMARY
.
.
.
LINKNAME:   VIPLOA01080A
   ADDRESS:  10.1.8.10 1
     FLAGS:  INTERNAL
10 OF 10 RECORDS DISPLAYED
END OF THE REPORT

SC32
D TCPIP,TCPIPC,N,HOME
RO SC32,D TCPIP,TCPIPC,N,HOME
HOME ADDRESS LIST:
LINKNAME:   VIPA1L
  ADDRESS:  10.1.1.30
    FLAGS:  PRIMARY
.
.
.
LINKNAME:   VIPLOA01080A
   ADDRESS:  10.1.8.10 1
     FLAGS:  INTERNAL
```

We next needed to verify that server-specific WLM function worked correctly. We started 18 TN3270 sessions and used a NETSTAT,VDPT,DETAIL command, shown in Example 5-19, to determine the distribution of these sessions.

Example 5-19 NETSTAT,VDPT,DETAIL command at our distribution stack SC30

```
SC30
/D TCPIP,TCPIPA,N,VDPT,DETAIL
RESPONSE=SC30
DYNAMIC VIPA DESTINATION PORT TABLE:
  DEST:        10.1.8.10..23   1 2
    DESTXCF:   10.1.7.11      3
    TOTALCONN: 0000000008 5 RDY: 001 4 WLM: 08 7 TSR: 100 8
    FLG: SERVERWLM 6
      TCSR: 100 9 CER: 100 10 SEF: 100 11
      QOSPLCACT: *DEFAULT*
        W/Q: 08 12
  DEST:        10.1.8.10..23
    DESTXCF:   10.1.7.21
    TOTALCONN: 0000000004 5 RDY: 001  WLM: 07  TSR: 093
    FLG: SERVERWLM
      TCSR: 093  CER: 100 SEF: 100
      QOSPLCACT: *DEFAULT*
        W/Q: 07

 DEST:        10.1.8.10..23
   DESTXCF:   10.1.7.31
   TOTALCONN: 0000000000  RDY: 001  WLM: 16  TSR: 100
   FLG: SERVERWLM
     TCSR: 100  CER: 100 SEF: 100
     QOSPLCACT: *DEFAULT*
       W/Q: 16
 3 OF 3 RECORDS DISPLAYED
 END OF THE REPORT
```

The output includes the following key information:

1. The destination IP address, which is the Sysplex Distributor IP address.

2. The port number to which connections are being distributed.

3. The destination XCF IP address.

4. The number of applications listening on the port number selected.

5. The total number of connections that have been forwarded by the Sysplex Distributor.

In our case, 8 of 12 TN3270 connection went to SC30. The remaining 4 connections went to SC31. To determine whether we were actually using server-specific WLM, we checked the FLAG **6** field. SERVERWLM indicates that we are using server-specific WLM when distributing our TN3270 connections.

This command provides additional information regarding the WLM weights and related values as part of the sysplex autonomics function: TCSR, CER, SEF, and TSR. These values are used by the distributing the IP stack to determine the best-suited target server for the TN3270 connection request:

- WLM: The Workload Manager weight value for the target listener **7**.
- TSR: The target server responsiveness value for the target server **8**.
- TCSR: The Target Connectivity Success Rate (TCSR) is a measure of the percentage of connection setup requests routed from the distributor that are successfully received by the target for this server **9**.
- CER: The Connection Establishment Rate (CER) is a measure of the percentage of the connection setup requests received at the target that achieve completion with the client **10**.
- SEF: The Server accept Efficiency Fraction (SEF) is a measure, calculated at intervals of approximately one minute, of the efficiency of the server application in accepting new connection requests and managing its backlog queue **11**.
- W/Q: The Workload Manager weight value for the target server after modification using QoS information provided by the Policy Agent **12**.

> **Note:** In this example, no Policy Agent was used.

We then verified that the optimized routing (with VIPAROUTE) was set up correctly and in use. We used the NETSTAT,VCRT,DETAIL command at our distribution IP stack SC30, as shown in Example 5-20. The output of this command displays the dynamic VIPA Connection Routing Table for SC30. Because this is the distributing IP stack, it shows all the connections between the clients and the participating IP stacks.

Example 5-20 Display of NETSTAT,VCRT from SC30 IP stack

```
SC30
D TCPIP,TCPIPA,N,VCRT,DETAIL
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.1.8.10..23
  SOURCE:  10.1.100.222..1091
  DESTXCF: 10.1.7.21
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
    INTF:  IUTIQDF5LNK 2
      VIPAROUTE:  YES 1 GW:  10.1.5.21 3
DEST:      10.1.8.10..23
  SOURCE:  10.1.100.222..1094
  DESTXCF: 10.1.7.31
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
    INTF:  IUTIQDF5LNK
      VIPAROUTE:  YES      GW:  10.1.5.31
```

This shows that the TN3270 connections were being forwarded to the target stacks using HiperSockets and not XCF. The key points are:

1. VIPAROUTE: YES indicates that an IP interface has been used by the Sysplex Distributor for forwarding this connection request to the chosen target server.

2. INTF: IUTIQDF5 represents our HiperSockets interface, which was used to forward the connection request within our CEC.

3. GW: This is the IP address of our HiperSockets gateway.

> **Note:** In our OSPF setup for dynamic routing, we wanted to prefer HiperSockets, if available, before using our shared OSA-Express ports. This can be simply achieved by setting the appropriate COST0 settings.

Although NETSTAT,VCRT,DETAIL on the distributing IP stack displays all the distributed connections, the same command issued on our target stacks shows only those connections established with the local server instance. Example 5-21 shows VCRT list on SC32.

Example 5-21 Display of NETSTAT,VCRT from our target IP stack on SC32

```
SC32
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.1.8.10..23
   SOURCE:  10.1.100.222..1094
   DESTXCF: 10.1.7.31
     POLICYRULE:    *NONE*
     POLICYACTION:  *NONE*
 1 OF 1 RECORDS DISPLAYED
 END OF THE REPORT
```

5.2.2 Sysplex Distributor using BASEWLM

We started with a typical Sysplex Distributor situation where three application instances of the TN3270 server are to be available in a z/OS sysplex environment. Figure 5-21 shows how TN3270 connections are load balanced across all three LPARS, based on the standard WLM distribution method (DISTMETHOD BASEWLM).

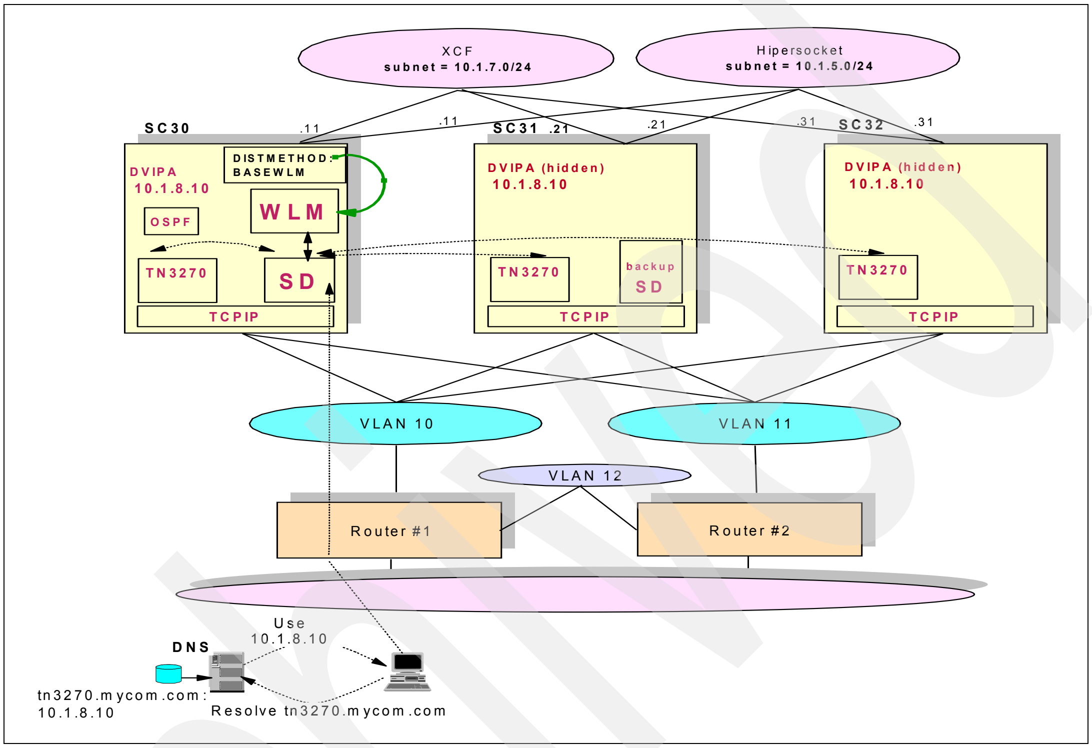


Figure 5-21 Workload balancing of TN3270 using Sysplex Distributor with DISTMETHOD BASEWLM

In this example, the Sysplex Distributor uses the default workload distribution method, BASELWLM, to determine which TN3270 instance is best suitable for aTN3270 connection request. The WLM weight given to the Sysplex Distributor is based on a comparison of the available capacity for new work on each system. OSPF is used as the dynamic routing protocol for advertising of the VIPA addresses, and to ensure connectivity by rerouting connections around failures.

Requirements to implement BasedWLM Method

You must implement either a dynamic routing protocol using OMPROUTE (as discussed in Chapter 4, “VIPA with dynamic routing” on page 65), or use ARP takeover (as discussed in Chapter 3, “VIPA without dynamic routing” on page 45) to propagate the VIPA address.

Furthermore, the TCP/IP stack needs to be configured to support DYNAMICXCF (which must be enabled for nondisruptive movement of DVIPAs) and SYSPLEXROUTING (which is required for WLM-based balancing).

Considerations about BasedWLM method

The Sysplex Distributor only works for TCP applications, not for UDP.

A sysplex is required for the Sysplex Distributor. All target servers must be System z servers and resident within the same sysplex. This might impose some limitations on flexibility.

Implementing the BasedWLM Method

This example has a similar implementation process to that shown in 5.2.1, “Sysplex Distributor using server-specific WLM” on page 132. The only difference is the method used to distribute the sessions. To implement this method we changed, in the TCPIPA profile, only the DISTMethod parameter in the VIPADEFine statement, as shown in Example 5-22.

Example 5-22 Using the distribution method BASEWLM

```
VIPADynamic
VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.1.8.10
VIPADISTribute DISTMethod BASEWLM 10.1.8.10
 PORT 23 DESTIP 10.1.7.11 10.1.7.21 10.1.7.31
```

Verifying the BasedWLM method implementation

To verify that the Basewlm method is implemented, the same commands can be used as those shown in “Sysplex Distributor using server-specific WLM” on page 132. To confirm we were using the expected method, we executed the command NETSTAT,VIPADCFG, as shown in Example 5-25, on our distribution stack (SC30).

Example 5-23 Display results of the VIPADCFG command issued on our distribution stack SC30

```
SC30
D TCPIP,TCPIPA,N,VIPADCFG
DYNAMIC VIPA INFORMATION:
   VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.10/24              1
      MOVEABLE: IMMEDIATE  SRVMGR: NO
   VIPA DISTRIBUTE:
   DEST:        10.1.8.10..23                  2 3
     DESTXCF:   10.1.7.11                        4
       SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM 5
   DEST:        10.1.8.10..23
     DESTXCF:   10.1.7.21                        4
       SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM 5
   DEST:        10.1.8.10
     DESTXCF:   10.1.7.31                        4
       SYSPT:   NO   TIMAFF: NO    FLG: BASEWLM 5
   VIPA ROUTE:                6
     DESTXCF:     10.1.7.11
       TARGETIP:  10.1.1.10
     DESTXCF:     10.1.7.21
       TARGETIP:  10.1.1.20
     DESTXCF:     10.1.7.31
       TARGETIP:  10.1.1.30
 END OF THE REPORT
```

Key information in this display includes:

1. This is the DVIPA address representing the TN3270 servers.
2. TN3270 connections are being distributed by the Sysplex Distributor.
3. We use port number 23 for TN3270.
4. This is the XCF IP address assigned and used.

5. FLG: BASEWLM indicates that server-specific WLM is active and will be used.

6. VIPAROUTE indicates that we do not use XCF links but use any available IP network interface for forwarding Sysplex Distributor-related packets. Three VIPAROUTE definitions point to the static VIPA in each of our stacks.

5.2.3 Sysplex Distributor using round-robin

This example is similar to the last one, but uses a round-robin distribution method to distribute TN3270 workload across three LPARs.

> **Important:** Be aware that when round-robin distribution is chosen, it supersedes any defined policies.

Figure 5-22 illustrates three incoming TN3270 connections being evenly distributed across three LPARs.

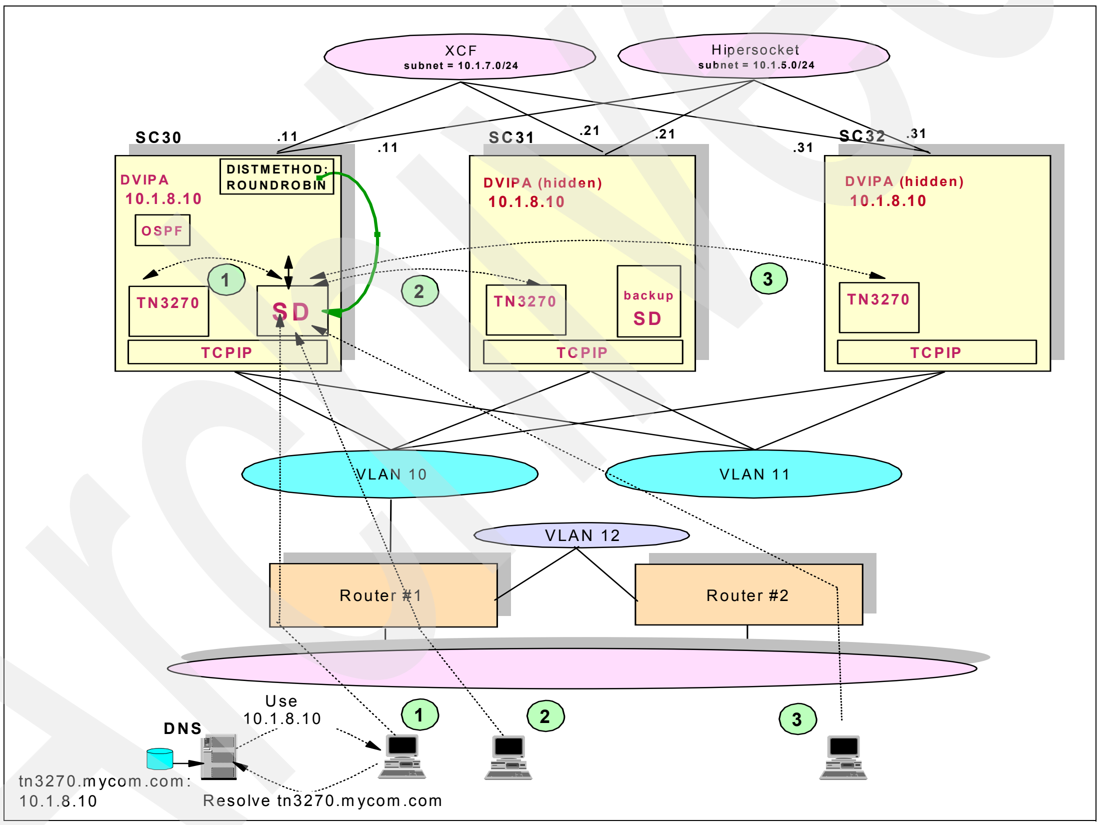


Figure 5-22 Round-robin workload distribution

Considerations about the round-robin method

The round-robin distribution method does not take into account server capacity when distributing new connection requests.

Implementing the round-robin distribution method

To implement this method, we followed the same steps as shown in 5.2.1, "Sysplex Distributor using server-specific WLM" on page 132 and in the TCPIPA profile, but changed the DISTMethod parameter in the VIPADEFine statement, as shown in Example 5-24.

Example 5-24 Using the round-robin distribution method

```
VIPADynamic
   VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.1.8.10
   VIPADISTribute DISTMethod ROUNDROBIN 10.1.8.10
    PORT 23 DESTIP 10.1.7.11 10.1.7.21 10.1.7.31
```

Verifying the round-robin distribution method implementation

To verify that the Round-robin method was implemented, the same commands as shown in 5.2.1, "Sysplex Distributor using server-specific WLM" on page 132 can be used. To confirm we were using the expected method we executed the command NETSTAT,VIPADCFG, as shown in Example 5-25, on our distribution stack (SC30).

Example 5-25 Display results of the VIPADCFG command issued on our distribution stack SC30

```
SC30
D TCPIP,TCPIPA,N,VIPADCFG
DYNAMIC VIPA INFORMATION:
   VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.10/24                     1
      MOVEABLE: IMMEDIATE  SRVMGR: NO
   VIPA DISTRIBUTE:
   DEST:         10.1.8.10..23                       2 3
     DESTXCF:    10.1.7.11                             4
       SYSPT:    NO   TIMAFF: NO    FLG: ROUNDROBIN 5
   DEST:         10.1.8.10..23
     DESTXCF:    10.1.7.21                             4
       SYSPT:    NO   TIMAFF: NO    FLG: ROUNDROBIN 5
   DEST:         10.1.8.10
     DESTXCF:    10.1.7.31                             4
       SYSPT:    NO   TIMAFF: NO    FLG: ROUNDROBIN 5
   VIPA ROUTE:                    6
     DESTXCF:      10.1.7.11
       TARGETIP:  10.1.1.10
     DESTXCF:      10.1.7.21
       TARGETIP:  10.1.1.20
     DESTXCF:      10.1.7.31
       TARGETIP:  10.1.1.30
 END OF THE REPORT
```

Key information in this display includes:

1. This is the DVIPA address representing the TN3270 servers.

2. TN3270 connections are being distributed by the Sysplex Distributor.

3. We use port number 23 for TN3270.

4. This is the XCF IP address assigned and used.

5. FLG: ROUNDROBIN indicates that the round-robin method is active and will be used.

6. VIPAROUTE indicates that we do not use XCF links but use any available IP network interface for forwarding Sysplex Distributor-related packets. Three VIPAROUTE definitions point to the static VIPA in each of our stacks.

5.2.4 Sysplex Distributor using WEIGHTEDActive method

Using this distribution method we are able to configure, for each target TCP/IP stack, an active connection weight. The distributor balances incoming connection requests across the targets, with a goal of having the number of active connections on each target proportionally equivalent to the configured active connection weight of each target.

Each weight can range in value from 1 to 99, so that the weights can be expressed as percentages if desired. Ideally, each weight should be greater than 10 so that granularity is preserved when autonomic fractions need to be applied to determine a modified weight.

It defaults to 10, so if DESTIP ALL is configured, then the default weight of 10 is assumed which results in a connection distribution goal to have an equal number of active connections on each target.

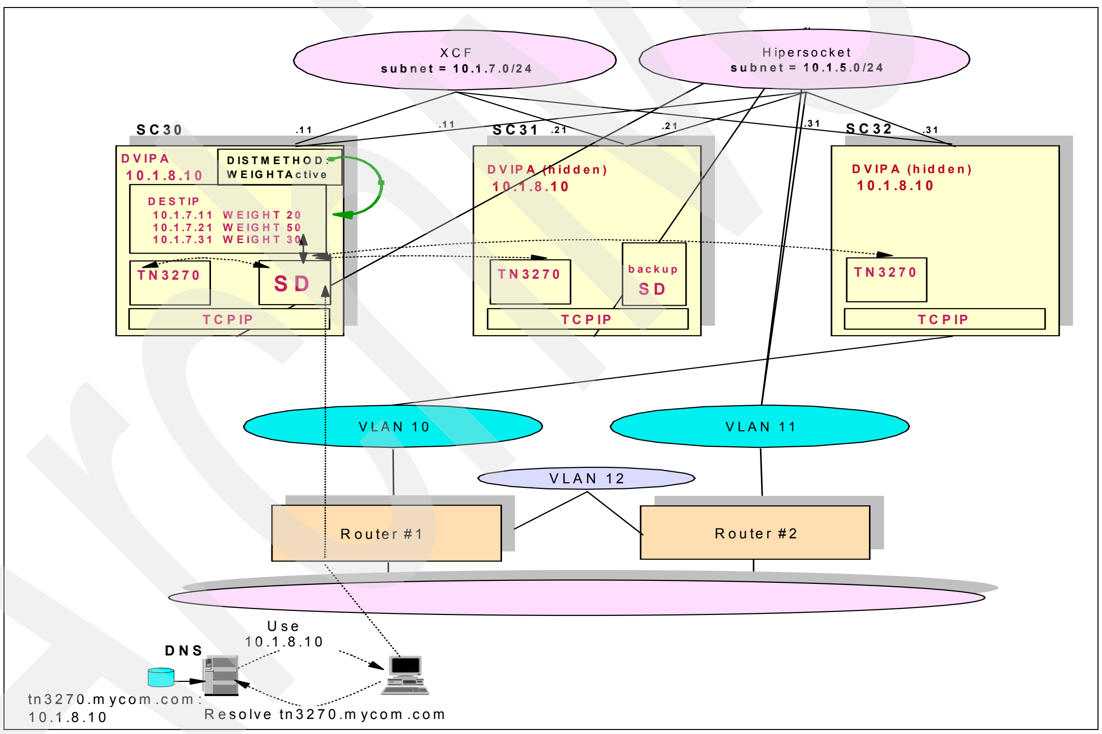


Figure 5-23 Workload balancing of TN3270 using Sysplex Distributor with DISTMethod WEIGHTEDActive

In this example, the Sysplex Distributor uses the WEIGHTEDActive workload distribution method, to determine which TN3270 instance is best suitable for a TN3270 connection request. The Target Server Responsiveness (TSR) fraction, abnormal completion rate fraction, and General Health fraction are applied against the configured Weight on each target to determine a modified weight. Connection goals are established based on the modified weight and the active connection count on each target.

Considerations about WEIGHTEDActive method

When implementing this method in a mixed z/OS environment, the following issues must be taken into consideration:

- If the Sysplex Distributor resides on a z/OS V1R9 system or newer, the WEIGHTEDActive distribution method can be used regardless of the release level of the target stacks.
- Each target stack needs to be on a z/OS V1R7 system or higher so that TSR metrics are reported to the distributor. TSR is considered to be 100% when a target is pre-V1R7.
- Each target stack needs to be on a z/OS V1R8 system or higher so that health metrics (abnormal terminations and health) are reported to the distributor. Health and normal termination rate are considered to be 100% when a target is pre-V1R8.
- Each backup stack needs to be on a z/OS V1R9 system or newer to allow WEIGHTEDActive distribution to be inherited during a takeover; otherwise, BASEWLM will be used.

Implementing the WEIGHTEDActive method

To implement the distribution method WEIGHTEDActive, we followed the same steps as used to implement all other methods, as shown in 5.2.1, “Sysplex Distributor using server-specific WLM” on page 132. In the TCPIPA profile, we changed the DISTMethod parameter in the VIPADEFine statement to use the option WEIGHTEDActive. Using this method, we also had to configure a Weight for each target destination, as shown in Example 5-26.

Example 5-26 Using the WEIGHTEDActive distributed method

```
VIPADYNAMIC
 VIPADEFINE MOVEABLE IMMEDIATE SERVICEMGR 255.255.255.0 10.1.8.10
 VIPADISTRIBUTE DISTMethod WEIGHTEDActive 10.1.8.10
  PORT 23 DESTIP 10.1.7.11 WEIGHT 20
                 10.1.7.21 WEIGHT 50
                 10.1.7.31 WEIGHT 30
```

Verifying the WEIGHTEDActive method implementation

To verify that the WEIGHTEDActive method is implemented, the same commands as shown in 5.2.1, “Sysplex Distributor using server-specific WLM” on page 132 can be used. To confirm that we were using the expected method, we executed the command NETSTAT,VIPADCFG, as shown in Example 5-25, on our distribution stack (SC30).

Example 5-27 Display results of the VIPADCFG command issued on our distribution stack SC30

```
SC30
D TCPIP,TCPIPA,N,VIPADCFG
DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.10/24                1
      MOVEABLE: IMMEDIATE  SRVMGR: NO
  VIPA DISTRIBUTE:
    DEST:        10.1.8.10..23                    2 3
      DESTXCF:   10.1.7.11                        4
        SYSPT:   NO   TIMAFF: NO    FLG: WEIGHTEDACTIVE  5
    DEST:        10.1.8.10..23                    2 3
      DESTXCF:   10.1.7.21                        4
        SYSPT:   NO   TIMAFF: NO    FLG: WEIGHTEDACTIVE  5
     DEST:        10.1.8.10..23                   2 3
      DESTXCF:   10.1.7.31                        4
        SYSPT:   NO   TIMAFF: NO    FLG: WEIGHTEDACTIVE   5
```

Key information in this display includes:

1. This is the DVIPA address representing the TN3270 servers.

2. TN3270 connections are being distributed by the Sysplex Distributor.

3. We use port number 23 for TN3270.

4. This is the XCF IP address assigned and used.

5. FLG: WEIGHTEDACTIVE indicates that WEIGHTEDActive method is active and will be used.

The command Nestat VIPADCFG,Detail shows the same results with more detailed information; see Example 5-28.

Example 5-28 D tcpip,tcpipa,n,vipadcfg,detail response

```
D TCPIP,TCPIPA,N,VIPADCFG,DETAIL
DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.10/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
  VIPA DISTRIBUTE:
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.11
        SYSPT:   NO   TIMAFF: NO    FLG: WEIGHTEDACTIVE
        OPTLOC:  NO   WEIGHT: 20                  1
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.21
        SYSPT:   NO   TIMAFF: NO    FLG: WEIGHTEDACTIVE
        OPTLOC:  NO   WEIGHT: 50                  1
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.31
        SYSPT:   NO   TIMAFF: NO    FLG: WEIGHTEDACTIVE
        OPTLOC:  NO   WEIGHT: 30                 1
END OF THE REPORT
```

Key information in this display includes:

1. This is the configured weight for each target.

The command Netstat,VDPT shows the active distribution method being used and the modified weight of each target; see Example 5-29.

Example 5-29 D tcpip,tcpipa,nestat,vpdt command

```
D TCPIP,TCPIPA,N,VDPT
DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.1.8.10..23
  DESTXCF:   10.1.7.11
  TOTALCONN: 0000000000  RDY: 000  WLM: 20  1 TSR: 100
  FLG: WEIGHTEDACTIVE                       2
DEST:        10.1.8.10..23
  DESTXCF:   10.1.7.21
  TOTALCONN: 0000000000  RDY: 000  WLM: 50  1 TSR: 100
  FLG: WEIGHTEDACTIVE                       2
DEST:        10.1.8.10..23
  DESTXCF:   10.1.7.31
  TOTALCONN: 0000000000  RDY: 001  WLM: 30  1 TSR: 100
```

```
  FLG: WEIGHTEDACTIVE                        2
3 OF 3 RECORDS DISPLAYED
```

Key information in this display includes:

1. The active weight for each target.

2. The active method.

The command Netstat VDPT Detail includes the active connection counts for each target, as shown in Example 5-30.

Example 5-30 D tcpip,tcpipa,nestat,vpdt,detail command

```
D TCPIP,TCPIPA,N,VDPT,DETAIL
DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.1.8.10..23
  DESTXCF:   10.1.7.11
  TOTALCONN: 0000000002 1 RDY: 001  WLM: 20  TSR: 100
  FLG: WEIGHTEDACTIVE
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000002 2
DEST:        10.1.8.10..23
  DESTXCF:   10.1.7.31
  TOTALCONN: 0000000004 1 RDY: 001  WLM: 30  TSR: 100
  FLG: WEIGHTEDACTIVE
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000004  2
DEST:        10.1.8.10..23
  DESTXCF:   10.1.7.21
  TOTALCONN: 0000000005 1 RDY: 001  WLM: 50  TSR: 100
  FLG: WEIGHTEDACTIVE
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000005 2
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

Key information in this display includes:

1. The total number of connections for each target.

2. The total number of active connections on each target.

5.2.5 Portsharing using SHAREPORTWLM

In an environment where you need to support a large number of client connections on a single system while maintaining good performance, the use of SHAREPORT or SHAREPORTWLM might be the best approach. A group of servers that share the same port benefit from using the server-specific WLM recommendations because the workload is more evenly balanced between them.

The SHAREPORTWLM keyword in the PORT profile statements indicates that multiple application instances can use this port. The set of server instances sharing the same TCP port on the same TCP/IP stack is known as a *shareport group*.

In our case, we defined a TN3270 server shareport group, consisting of three TN3270 server instances, on our distribution stack SC30. We did not change the remaining TN3270 servers on systems SC31 and SC32. It was our goal to distribute incoming TN3270 connections among our five TN3270 servers, based on the individual server weight (server-specific WLM) by combining the use of the Sysplex Distributor and the shareport function as illustrated in Figure 5-24.

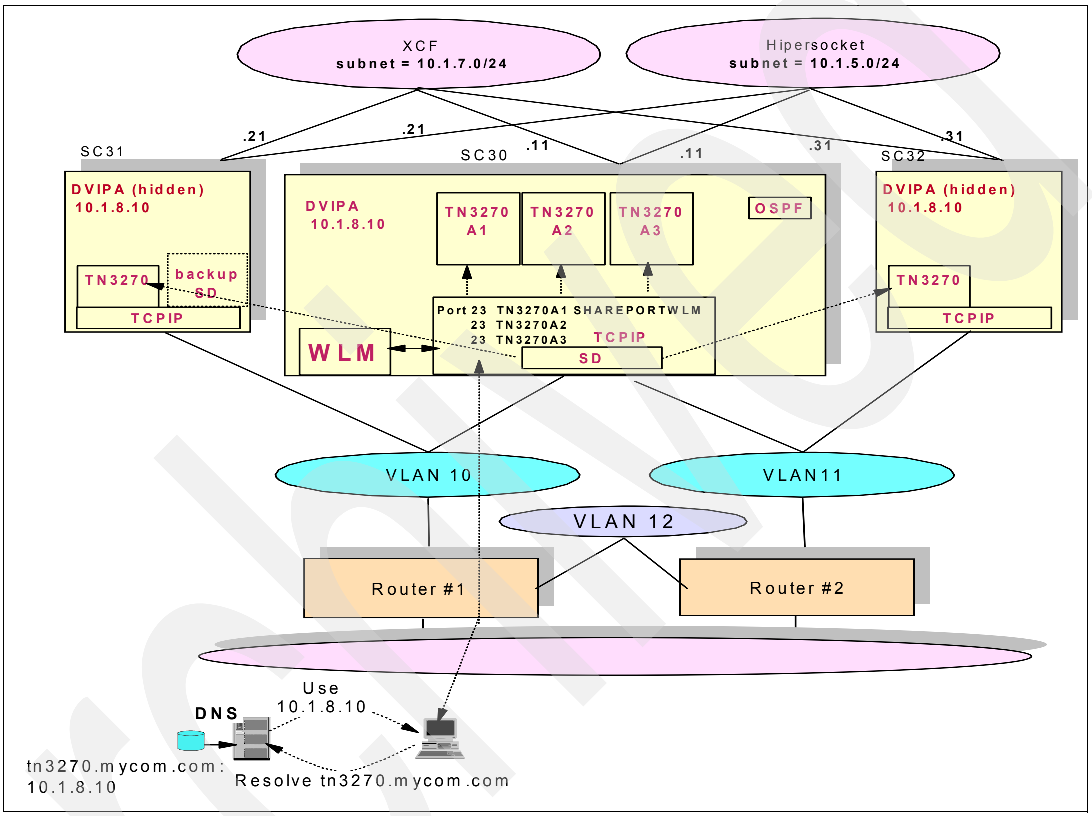


Figure 5-24 Internal workload balancing using SHAREPORTWLM

Implementation tasks

We needed to complete the following tasks in order to implement multiple TN3270 servers sharing a TCP port:

1. Define the TN3270 shareport group in our TCP/IP profile of SC30.
2. Define TN3270 profile statements in a separate input file.
3. Define the TN3270 procedure in SYS1.PROCLIB.
4. Start the TN3270 servers as a separate address space.

The SHAREPORTWLM parameter in the PORT statement is required for reserving a port to be shared across multiple listeners on the same interface. When specified as shown in Example 5-31, TCP/IP allowed our three TN3270 servers to listen on the same combination of port and interface.

> **Note:** We could specify SHAREPORT instead of the SHAREPORTWLM. With SHAREPORT, incoming connection requests would be distributed in a weighted round-robin fashion. We choose SHAREPORTWLM, because we wanted to use the more accurate server-specific WLM weights.

Example 5-31 Define the TN3270 shareport group on SC30

```
PORT
    20 TCP *  NOAUTOLOG          ; FTP Server
    21 TCP OMVS                  ; control port
    23 TCP TN3270A1 SHAREPORTWLM     ; MVS Telnet Server sep.addrspace
    23 TCP TN3270A2                  ; MVS Telnet Server sep.addrspace
    23 TCP TN3270A3                  ; MVS Telnet Server sep.addrspace
```

Example 5-32 lists part of the TN3270 profile statements for the first of our three TN3270 servers. The others are almost identical. The only difference is the definition of the different ranges within the 1 DEFAULTLUS/ENDDEFAULTLUS parameters, so that we could more easily monitor where our TN3270 connections went.

We had more than one TCP/IP stack running in one LPAR and needed TN3270 stack affinity definition with the proper TCP/IP stack used 2.

Example 5-32 Define a separate TN3270 profile member: TNSC30A1

```
TelnetGlobals
  TCPIPJOBNAME TCPIPA 2
 EndTelnetGlobals
 ;
 TelnetParms
 .
 .
 .
 EndTelnetParms
 ;
 BeginVTAM
   Port 23
   ; Define the LUs to be used for general users.
   DEFAULTLUS
     TCP30301..TCP30310 1
   ENDDEFAULTLUS
 ; DEFAULTAPPL TSO   ; Set the default application for all TN3270(E)
                     ; Telnet sessions to TSO

   LINEMODEAPPL TSO  ; Send all line-mode terminals directly to TSO.

   ALLOWAPPL SC*  DISCONNECTABLE ; Allow all users access to TSO
   ALLOWAPPL TSO* DISCONNECTABLE ; Allow all users access to TSO
   ALLOWAPPL *      ; Allow all applications that have not been
                   ; previously specified to be accessed.
  USSTCP USSTEST1
EndVTAM
```

For each of our three TN3270 servers we defined a separate start procedure in SYS1.PROCLIB pointing to the respective TN3270 parameter member located in our TCPPARMS data set. The three procedures are the same except for the name and the profile name, as shown in Example 5-33.

*Example 5-33 Define a TN3270 procedure: TN3270A**

```
SYS1.PROCLIB(TN3270A1) - 01.01                  Columns 00001 00072
===>                                                  Scroll ===> CSR
***************************** Top of Data ******************************
//TN3270C1 PROC PARMS='CTRACE(CTIEZBTN)'
//TN3270C  EXEC PGM=EZBTNINI,REGION=0M,PARM='&PARMS'
//SYSPRINT DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//SYSOUT   DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=132,BLKSIZE=136)
//PROFILE  DD DISP=SHR,DSN=TCPIPA.TCPPARMS(TNSC30A1)
//SYSTCPD  DD DSN=TCPIPA.TCPPARMS(DATAC&SYSCLONE),DISP=SHR
=======================================================================
```

We started our three TN3270 server instances within the same LPAR, as shown in Example 5-34 (we show only one server as an example).

Example 5-34 Start TN3270A1

```
S TN3270A1
$HASP100 TN3270A1 ON STCINRDR
IEF695I START TN3270A1 WITH JOBNAME TN3270A1 IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 TN3270A1 STARTED
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
EZZ6001I TELNET SERVER STARTED
EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 538
            TCPIPA.TCPPARMS(TNSC30A1)
EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 539
            TCPIPA.TCPPARMS(TNSC30A1)
EZZ6003I TELNET LISTENING ON PORT    23
```

Configuration example

Example 5-35 lists the parts of our TCP/IP profile from our distribution stack SC30 containing the parameter and statements that are relevant to our TN3270 shareport group.

Example 5-35 TCP/IP profile of our distribution stack SC30

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
 ;
.
.
.
 PORT
     20 TCP *  NOAUTOLOG              ; FTP Server
     21 TCP OMVS                      ; control port
     23 TCP TN3270A1 SHAREPORTWLM     ; MVS Telnet Server sep.addrspace
```

```
     23 TCP TN3270A2                  ; MVS Telnet Server sep.addrspace
     23 TCP TN3270A3                  ; MVS Telnet Server sep.addrspace
 ;
```

Verification

We used these steps to verify that our TN3270 shareport group implementation was correct:

1. Verify that all three TN3270 servers are running and listening on the same port.
2. Verify that server-specific WLM will be used for distributing TN3270 connections.
3. Start some TN3270 connections and verify that they are distributed among the servers.

We verified that all three TN3270 servers were running and listening on the same port by using a NETSTAT,CONN command in SC30, as shown in Example 5-36.

Example 5-36 Display NETSTAT,CONN on SC30

```
D TCPIP,TCPIPA,N,CONN
USER ID  CONN     STATE

TN3270A1 0000003C LISTEN 1
  LOCAL SOCKET:   ::..23 2
  FOREIGN SOCKET: ::..0
TN3270A2 00000041 LISTEN 1
  LOCAL SOCKET:   ::..23 2
  FOREIGN SOCKET: ::..0
TN3270A3 00000042 LISTEN 1
  LOCAL SOCKET:   ::..23 2
  FOREIGN SOCKET: ::..0

 3 OF 3 RECORDS DISPLAYED
 END OF THE REPORT
```

This output verified that:

1. All three TN3270 servers are running and listening.

2. All were using the same port 23.

We then verified that server-specific WLM was used for distributing TN3270 connections, as shown in Example 5-37.

Example 5-37 Display NETSTAT,VIPADCFG on SC30

```
D TCPIP,TCPIPA,N,VIPADCFG
DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
   IPADDR/PREFIXLEN: 10.1.8.10/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
VIPA DISTRIBUTE:
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.11
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM  1
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.21
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM  1
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.31
```

```
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM  1
  VIPA ROUTE:
    DESTXCF:     10.1.7.11
      TARGETIP:  10.1.1.10
    DESTXCF:     10.1.7.21
      TARGETIP:  10.1.1.20
    DESTXCF:     10.1.7.31
      TARGETIP:  10.1.1.30
END OF THE REPORT
```

In this output **1** FLG: SERVERWLM indicates that server-specific WLM is active and will be used.

We started 21 x TN3270 connections from one of our workstations. To verify the distribution among all TN3270 servers, including our three TN3270 servers belonging to the shareport group on system SC30, we first issued the NETSTAT,VDPT command, as shown in Example 5-38.

Example 5-38 Display NETSTAT,VDPT on SC30

```
D TCPIP,TCPIPA,N,VDPT
DYNAMIC VIPA DESTINATION PORT TABLE:
 DEST:         10.1.8.10..23
   DESTXCF:    10.1.7.11
   TOTALCONN: 0000000013 1 RDY: 003 2 WLM: 05  TSR: 070
   FLG: SERVERWLM
 DEST:         10.1.8.10..23
   DESTXCF:    10.1.7.21
   TOTALCONN: 0000000007 1 RDY: 001 2 WLM: 08  TSR: 100
   FLG: SERVERWLM
 DEST:         10.1.8.10..23
   DESTXCF:    10.1.7.31
   TOTALCONN: 0000000001 1 RDY: 001 2 WLM: 16  TSR: 100
   FLG: SERVERWLM
 3 OF 3 RECORDS DISPLAYED
 END OF THE REPORT
```

The important information in this output includes:

1. TOTALCONN indicates the number of TN3270 connections per LPAR.

2. RDY indicates the number of TN3270 servers listening on the port.

In our implementation we customized our distribution stack, SC30, to be able to run three TN3270 servers as separate address spaces, all belonging to the same shareport group. These TN3270 servers are active and running, as indicated by `RDY: 003`. Of the 21 total TN3270 connections, system SC30 received 13, SC31 received 7, and SC32 received 1.

To determine the distribution of the 13 TN3270 connections within our shareport group on SC30, we used the specific TN3270 commands shown in Example 5-39, Example 5-40, and Example 5-41.

Example 5-39 Display TELNET,CONN for our first TN3270 server TN3270A1 on SC30

```
D TCPIP,TN3270A1,T,CONN
 EZZ6064I TELNET CONNECTION DISPLAY 234
          EN                                      TSP
 CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
```

```
-------- -- --------------------- -------- --------  --- --------
0000005C    ::FFFF:10.1.100.222..2683
                                  TCP30307          TPE
.
.
.
----- PORT:    23  ACTIVE             PROF: CURR CONNS:      7
------------------------------------------------------------
16 OF 16 RECORDS DISPLAYED
```

Example 5-40   Display TELNET,CONN for our second TN3270 server TN3270A2 on SC30

```
D TCPIP,TN3270A2,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 342
         EN                                   TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000052    ::FFFF:10.1.100.222..2680
                                  TCP30313          TPE
----- PORT:    23  ACTIVE             PROF: CURR CONNS:      3
------------------------------------------------------------
8 OF 8 RECORDS DISPLAYED
```

Example 5-41   Display TELNET,CONN for our third TN3270 server TN3270A3 on SC30

```
D TCPIP,TN3270A3,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 392
         EN                                   TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- --------------------- -------- --------  --- --------
00000065    ::FFFF:10.1.100.222..2686
                                  TCP30323          TPE
----- PORT:    23  ACTIVE             PROF: CURR CONNS:      3
------------------------------------------------------------
8 OF 8 RECORDS DISPLAYED
```

When we add the number of connections reported by each TN3270 server on SC30 under CURR CONNS: 1 we obtain 13 connections for the TN3270 portsharing group, which is the expected result.

5.2.6 Sysplex Distributor using Unreachable DVIPA Detection and Recovery

There are configurations in which it is not optimal to configure dynamic XCF interfaces as eligible backup network paths for TCP/IP stacks that own and advertise DVIPAs. In these configurations, incoming network traffic for the DVIPAs is expected to arrive by way of one or more external network interfaces. If these external interfaces fail, or the networks they are attached to experience a failure, the DVIPAs owned by the local stack might become isolated. Client traffic destined for these DVIPAs will not be able to reach the local TCP/IP stack; the DVIPA is unreachable.

To prevent the DVIPAs from becoming isolated, you can use the MONINTERFACE parameter on the GLOBALCONFIG statement and the MONSYSPLEX parameter on the LINK statement. If the MONINTERFACE parameter is defined, you can monitor the status of interfaces set the MONSYSPLEX keyword. The VIPAs can be taken over to the other stacks when an inactive status is detected on all monitored interfaces. Optionally, you can monitor the presence of dynamic routing over the interfaces if the DYNROUTE parameter is specified on the GLOBALCONFIG statement.

Figure 5-25 illustrates our test environment.

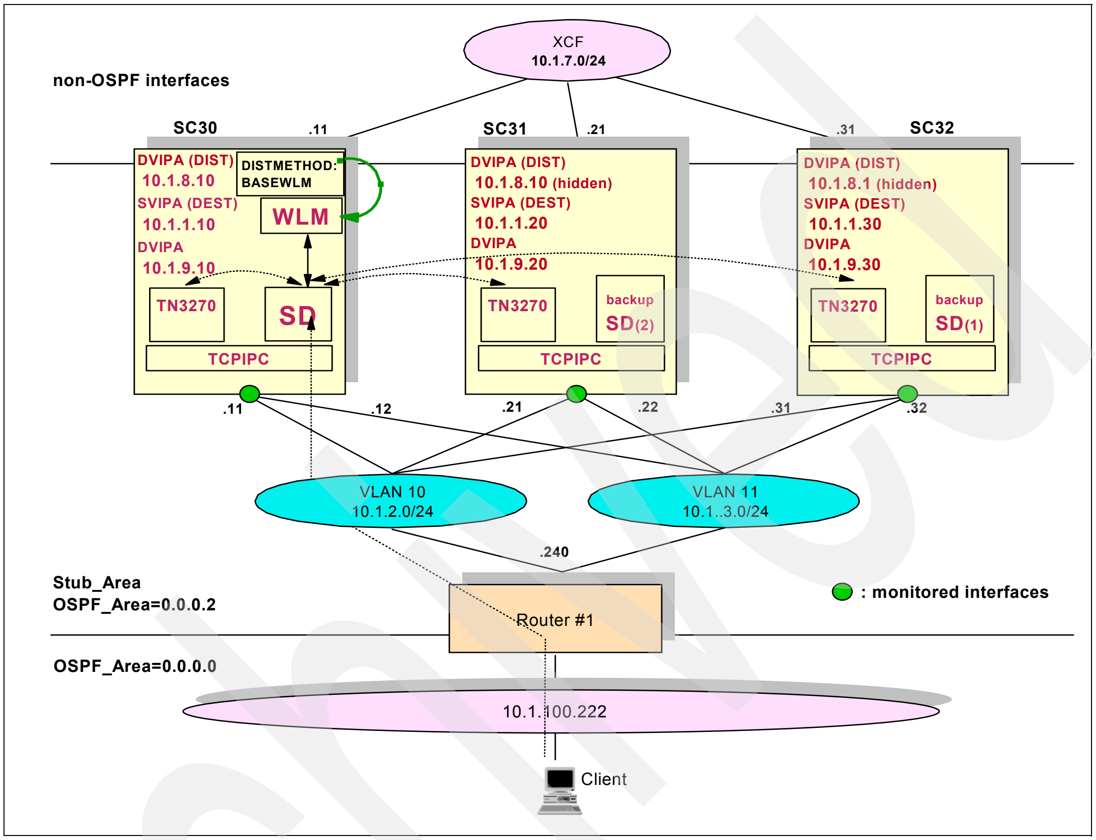


Figure 5-25 Server-specific WLM Workload balancing with monitored interfaces

In our environment:

- SC30 was defined as Sysplex Distributor, with SC31 and SC32 defined as the backup stacks for the DVIPAs.
- All three stacks were the target stacks for our clients.
- Dynamic XCF interfaces were non-OSPF interfaces.
- Two shared OSA adapters on each TCP/IP stack were defined as the OSPF interface, and these interfaces were also defined as monitored.

Although we only used one Layer 3 switch in our test configuration, you should always consider using at least two Layer 3 switch or routers for fault tolerance in a production environment.

Note: The Unreachable DVIPA Detection and Recovery function works with the DVIPA, which can be defined by the VIPADEFINE or VIPARANGE statement. Therefore, we defined DVIPAs on all stacks.

Implementation tasks

We needed to complete the following implementation tasks:

1. Specify the MONINTERFACE and DYNROUTE parameters on the GLOBALCONFIG SYSPLEXMONITOR statement.
2. Specify the RECOVERY parameter on the GLOBALCONFIG SYSPLEXMONITOR statement.
3. Specify the DELAYJOIN and AUTOREJOIN parameters on the GLOBALCONFIG SYSPLEXMONITOR statement.
4. Configure the Sysplex Distributor stack, the target stacks, and the backup stacks with at least one DVIPA. The related statements will be defined in the TCPIP profiles and the OMPROUTE configuration files.
5. Configure the VIPAROUTE statements.
6. Specify the MONSYSPLEX parameter on the LINK statement of the interfaces to be monitored.

Configuration

In Example 5-42, we show the most important parts of our TCP/IP profile, which are necessary to monitor the interfaces and to detect and recover any failures.

Example 5-42 Profile TCPIPA SC30 Sysplex Distributor, Target and Backup

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
SYSPLEXMONitor RECOVERY DYNROUTE MONINTERFACE DELAYJOIN AUTOREJOIN 1
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA PATHMTUDISC
DYNAMICXCF 10.1.7.10 255.255.255.0 8
;
DEVICE OSA2080   MPCIPA
LINK   OSA2080L  IPAQENET        OSA2080  VLANID 10 MONSYSPLEX 2
DEVICE OSA20E0   MPCIPA
LINK   OSA20E0L  IPAQENET        OSA20E0  VLANID 11 MONSYSPLEX 2
DEVICE VIPA1    VIRTUAL 0
LINK   VIPA1L   VIRTUAL 0     VIPA1
;
VIPADynamic
   VIPADEFine MOVEable IMMEDiate 255.255.255.255 10.1.8.10 3
   VIPADEFine MOVEable IMMEDiate 255.255.255.255 10.1.9.10
   VIPADISTribute DISTMethod SERVERWLM 10.1.8.10 3
    PORT 23 DESTIP 10.1.7.11 10.1.7.21 10.1.7.31
    VIPAROUTE 10.1.7.11 10.1.1.10 4
    VIPAROUTE 10.1.7.21 10.1.1.20 4
    VIPAROUTE 10.1.7.31 10.1.1.30 4
ENDVIPADYNAMIC
;
HOME
   10.1.1.10   VIPA1L 4
   10.1.2.11   OSA2080L
   10.1.3.12   OSA20E0L
```

The following are important considerations for this TCPIP profile:

1. We defined the DYNROUTE and MONINTERFACE parameters. Moreover, we defined the RECOVERY parameter so that the recovery action is initiated when the stack detected unreachable DVIPA.

   > **Note:** The TIMERSECS value on the GLOBALCONFIG statement can be used to determine how quickly sysplex autonomics will react when a failure is detected. The default value is 60 seconds.

2. We defined both OSA adapters as monitored interfaces.
3. We used the IP address 10.1.8.10 as the Distributed DVIPA.
4. We defined three VIPAROUTE statements, each pointing to one of our target systems. The target IP address also needs to be correctly defined in the HOME list of the target stack.

The TCPIP profile statements for SC31 and SC32 are shown in Example 5-43 and Example 5-44. Example 5-43 shows the Target and Backup for SC31.

Example 5-43 Profile TCPIPB SC31 Target and Backup

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
SYSPLEXMONITOR RECOVERY DYNROUTE MONINTERFACE DELAYJOIN AUTOREJOIN 1
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA PATHMTUDISC
DYNAMICXCF 10.1.7.21 255.255.255.0 8
;
DEVICE OSA2080  MPCIPA
LINK   OSA2080L  IPAQENET      OSA2080  VLANID 10 MONSYSPLEX   2
DEVICE OSA20E0  MPCIPA
LINK   OSA20E0L  IPAQENET      OSA20E0  VLANID 11 MONSYSPLEX   2
DEVICE VIPA1    VIRTUAL 0
LINK   VIPA1L VIRTUAL 0     VIPA1
;
VIPADynamic
   VIPABackup 2 MOVEable IMMEDiate 255.255.255.255 10.1.8.10  3
   VIPADEFine   MOVEable IMMEDiate 255.255.255.255 10.1.9.20  4
ENDVIPADynamic
;
HOME
   10.1.1.20   VIPA1L 5
   10.1.2.21   OSA2080L
   10.1.3.22   OSA20E0L
```

Example 5-44 shows the Target and Backup for SC32.

Example 5-44 Profile TCPIPC SC32 Target and Backup

```
ARPAGE 20
;
GLOBALCONFIG NOTCPIPSTATISTICS
SYSPLEXMONITOR RECOVERY DYNROUTE MONINTERFACE DELAYJOIN AUTOREJOIN 1
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
```

```
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA PATHMTUDISC
DYNAMICXCF 10.1.7.31 255.255.255.0 8
;
DEVICE OSA2080   MPCIPA
LINK   OSA2080L  IPAQENET      OSA2080  VLANID 10 MONSYSPLEX 2
DEVICE OSA20E0   MPCIPA
LINK   OSA20E0L  IPAQENET      OSA20E0  VLANID 11 MONSYSPLEX 2
DEVICE VIPA1     VIRTUAL 0
LINK   VIPA1L    VIRTUAL 0     VIPA1
;
VIPADynamic
   VIPABackup 1 MOVEable IMMEDiate 255.255.255.0 10.1.8.10 3
   VIPADEFine   MOVEable IMMEDiate 255.255.255.0 10.1.9.30 4
ENDVIPADynamic
;
HOME
   10.1.1.30    VIPA1L 5
   10.1.2.31    OSA2080L
   10.1.3.32    OSA20E0L
```

The important information in this output includes:

1. We defined the same options as SC30.
2. We defined both OSA adapters to be monitored.
3. We defined SC31 as the first backup stack and SC32 as the second backup stack for SC30, by using the VIPABackup statement. Larger numerical rank values move the respective stacks closer to the beginning of the backup list.
4. We defined the non-Distributed DVIPA to enable the monitoring function on each stack.
5. We specified static VIPAs on the VIPAROUTE statement of SC30.

In the OMPROUTE configuration files, we configured the INTERFACE statement for the Dynamic XCF interfaces, because we assumed that our test environment only has the only OSA interfaces as the dynamic routing path.

Verification

We used the following steps to verify our Sysplex Distributor implementation with the Unreachable DVIPA function:

1. Verify the sysplex group (EZBTCPCS) that the stacks should join.
2. Verify the general dynamic VIPA and Sysplex Distributor setup.
3. Verify that our VIPAROUTE definitions work.
4. Verify the monitored interfaces setup.
5. Verify that the Unreachable DVIPA detection and recovery function works.

Normal status

We used a Display XCF, GROUP,*groupname* command to verify that all three stacks 1 were joined into the same XCF group, as shown in Example 5-45.

Example 5-45 Display of XCF Group on SC30

```
SC30
D XCF,GROUP,EZBTCPCS
IXC332I  11.26.51  DISPLAY XCF 955
```

```
 GROUP EZBTCPCS:
               SC30TCPIPA 1
               SC31TCPIPB 1
               SC32TCPIPC 1
```

To verify the general dynamic VIPA and Sysplex Distributor setup, we issued several commands:

- Display NETSTAT,VIPADYN
- Display SYSPLEX,VIPADYN
- Display NETSTAT,HOME

Issuing a NETSTAT,VIPADCFG command on SC30, shown in Example 5-46, provided helpful information about the Sysplex Distributor configuration.

Example 5-46 Display result of the NETSTAT,VIPADYN command on SC30

```
D TCPIP,TCPIPA,NETSTAT,VIPADYN
DYNAMIC VIPA:
  IPADDR/PREFIXLEN: 10.1.9.10/32
    STATUS:  ACTIVE     ORIGIN: VIPADEFINE 1     DISTSTAT:
    ACTTIME: 08/09/2007 15:51:27 2
  IPADDR/PREFIXLEN: 10.1.8.10/32
    STATUS:  ACTIVE     ORIGIN: VIPADEFINE 1     DISTSTAT: DIST/DEST 3
    ACTTIME: 08/09/2007 15:51:27 2
VIPA ROUTE:
  DESTXCF: 10.1.7.11
    TARGETIP: 10.1.1.10
    RTSTATUS: DEFINED 4
  DESTXCF: 10.1.7.21
    TARGETIP: 10.1.1.20
    RTSTATUS: ACTIVE 5
  DESTXCF: 10.1.7.31
    TARGETIP: 10.1.1.30
    RTSTATUS: ACTIVE 5
5 OF 5 RECORDS DISPLAYED
END OF THE REPORT
```

The important information in this output includes:

1. **ORIGIN** represents the original definition of DVIPA.
2. This is the time stamp indicating when the DVIPA was activated on the local stack, specified as Coordinated Universal Time (UTC).
3. This stack is the distributor stack as well as the destination stack for the Distributed DVIPA 10.1.8.10.
4. The target IP address specified is on the local stack.
5. The target IP address specified is reachable from this stack, and it also means that the VIPAROUTE definition works.

Issuing a SYSPLEX,VIPADYN command, shown in Example 5-47, displays the relationship among the members of the sysplex group.

Example 5-47 Display result of the SYSPLEX,VIPADYN command on SC30

```
D TCPIP,TCPIPA,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V1R11 747
VIPA DYNAMIC DISPLAY FROM TCPIPA   AT SC30
IPADDR: 10.1.9.30
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPC   SC32     ACTIVE
LINKNAME: VIPL0A01090A
IPADDR/PREFIXLEN: 10.1.9.10/32
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPA   SC30     ACTIVE
IPADDR: 10.1.9.20
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPB   SC31     ACTIVE
LINKNAME: VIPL0A01080A
IPADDR/PREFIXLEN: 10.1.8.10/32
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPA   SC30     ACTIVE1  BOTH
  TCPIPB   SC31     BACKUP1 0022 DEST
  TCPIPC   SC32     BACKUP1 0012 DEST
6 OF 6 RECORDS DISPLAYED
```

Several fields show important information of members in the sysplex group.

1. **STATUS** represents the actual status of the TCP/IP stacks.
2. **RANK** shows which backup stack will take over the distributor stack when the primary Sysplex Distributor is unreachable or goes down.

A NETSTAT,HOME command, as shown in Example 5-48, can display the DVIPAs that were generated as active.

Example 5-48 Display result of the NETSTAT,HOME command on all stacks

```
SC30
D TCPIP,TCPIPA,NETSTAT,HOME
HOME ADDRESS LIST:
LINKNAME:   VIPA1L
  ADDRESS:  10.1.1.10
.
.
.
LINKNAME:   VIPL0A01080A
  ADDRESS:  10.1.8.10 2
    FLAGS:
LINKNAME:   VIPL0A01090A
  ADDRESS:  10.1.9.20 1
    FLAGS:
```

```
9 OF 9 RECORDS DISPLAYED
END OF THE REPORT

SC31
D TCPIP,TCPIPB,NETSTAT,HOME
HOME ADDRESS LIST:
LINKNAME:   VIPA1L
  ADDRESS:  10.1.1.20
    FLAGS:  PRIMARY
.
.
.
LINKNAME:   VIPL0A010914
  ADDRESS:  10.1.9.20 1
    FLAGS:
LINKNAME:   VIPL0A01080A
  ADDRESS:  10.1.8.10 2
    FLAGS:  INTERNAL 2
LINKNAME:   LOOPBACK
  ADDRESS:  127.0.0.1
    FLAGS:
INTFNAME:   LOOPBACK6
  ADDRESS:  ::1
    TYPE:   LOOPBACK
    FLAGS:
9 OF 9 RECORDS DISPLAYED
END OF THE REPORT...

SC32
D TCPIP,TCPIPC,NETSTAT,HOME
HOME ADDRESS LIST:
LINKNAME:   VIPA1L
  ADDRESS:  10.30.1.221
    FLAGS:  PRIMARY
LINKNAME:   OSA2080L
  ADDRESS:  10.30.2.222
    FLAGS:
LINKNAME:   OSA20E0L
  ADDRESS:  10.30.3.225
    FLAGS:
LINKNAME:   EZASAMEMVS
  ADDRESS:  10.1.7.30
    FLAGS:
LINKNAME:   IQDIOLNK0A1E1466
  ADDRESS:  10.1.7.30
    FLAGS:
LINKNAME:   VIPL0A01091E
  ADDRESS:  10.30.1.222 1
    FLAGS:
LINKNAME:   VIPL0A01080A
  ADDRESS:  10.1.8.10 2
    FLAGS:  INTERNAL 2
LINKNAME:   LOOPBACK
  ADDRESS:  127.0.0.1
    FLAGS:
```

```
INTFNAME:   LOOPBACK6
  ADDRESS:  ::1
    TYPE:   LOOPBACK
    FLAGS:
9 OF 9 RECORDS DISPLAYED
END OF THE REPORT
```

Key information includes:

1. These IP addresses are Non-Distributed DVIPA.

2. This is the Distributed DVIPA.

3. `INTERNAL` represents internally generated VIPAs that are not advertised to routing daemon.

We issued a NETSTAT,CONFIG command and a NETSTAT,DEvlinks command, to verify that the Unreachable DVIPA detection and recovery function was correctly defined.

Example 5-49 shows the output from issuing a NETSTAT,CONFIG command.

Example 5-49 Display result of the NETSTAT,CONFIG command on SC30

```
D TCPIP,TCPIPA,NETSTAT,CONFIG
TCP CONFIGURATION TABLE:
.
.
.
GLOBAL CONFIGURATION INFORMATION:
TCPIPSTATS: NO   ECSALIMIT: 0000000K  POOLLIMIT: 0000000K
MLSCHKTERM: NO   XCFGRPID:      IQDVLANID:  0
SYSPLEXWLMPOLL: 330722956
SYSPLEX MONITOR:
  TIMERSECS: 0060  RECOVERY: YES  DELAYJOIN: YES  AUTOREJOIN: YES
  MONINTF:   YES 1 DYNROUTE: YES 2
```

Important keywords on the SYSPLEXMONITOR parameter are:

1. `MONINTF: YES` indicates that the TCP/IP stack is monitoring the status of network interfaces that have the MONSYSPLEX parameter specified on the LINK statement.

2. `DYNROUTE: YES` indicates that the TCP/IP stack is monitoring the presence of dynamic routes over monitored network interfaces that have the MONSYSPLEX parameter specified on the LINK statement.

In Example 5-50, which is the output of the NETSTAT,DEvlinks command on SC30, `MONSYSPLEX: YES` marked as **1** indicates that the status of this link is being monitored by Sysplex Autonomics.

Example 5-50 Display result of the NETSTAT,DEvlinks on SC30

```
D TCPIP,TCPIPA,NETSTAT,DEV
DEVNAME: OSA2080          DEVTYPE: MPCIPA
  DEVSTATUS: READY
  LNKNAME: OSA2080L       LNKTYPE: IPAQENET   LNKSTATUS: READY
    NETNUM: N/A  QUESIZE: N/A  SPEED: 0000001000
    IPBROADCASTCAPABILITY: NO
    CFGROUTER: NON                  ACTROUTER: NON
    ARPOFFLOAD: YES                 ARPOFFLOADINFO: YES
    ACTMTU: 8992
```

```
    VLANID: 30                      VLANPRIORITY: DISABLED
    DYNVLANREGCFG: NO               DYNVLANREGCAP: YES
    READSTORAGE: GLOBAL (4096K)     INBPERF: BALANCED
    CHECKSUMOFFLOAD: YES            SEGMENTATIONOFFLOAD: YES
    SECCLASS: 255                   MONSYSPLEX: YES 1
.
.
DEVNAME: OSA20E0           DEVTYPE: MPCIPA
  DEVSTATUS: READY
  LNKNAME: OSA20EOL        LNKTYPE: IPAQENET   LNKSTATUS: READY
    NETNUM: N/A  QUESIZE: N/A  SPEED: 0000001000
    IPBROADCASTCAPABILITY: NO
    CFGROUTER: NON                  ACTROUTER: NON
    ARPOFFLOAD: YES                 ARPOFFLOADINFO: YES
    ACTMTU: 8992
    VLANID: 31                      VLANPRIORITY: DISABLED
    DYNVLANREGCFG: NO               DYNVLANREGCAP: YES
    READSTORAGE: GLOBAL (4096K)     INBPERF: BALANCED
    CHECKSUMOFFLOAD: YES            SEGMENTATIONOFFLOAD: YES
    SECCLASS: 255                   MONSYSPLEX: YES 1
.
.
```

You can issue the NETSTAT,VDPT,DETAIL command and the NETSTAT,VCRT,DETAIL command, as shown in Example 5-51, Example 5-52, and Example 5-53, to verify that the Sysplex Distributor works correctly, after TN3270E sessions are started.

For more information about each field of these displays, refer to Example 5-19 on page 139 and Example 5-20 on page 140 and to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Example 5-51 Display result of the NETSTAT,VDPT,DETAIL command on SC30

```
D TCPIP,TCPIPA,NETSTAT,VDPT,DETAIL
DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:         10.1.8.10..23
  DESTXCF:    10.1.7.10
  TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 00
DEST:         10.1.8.10..23
  DESTXCF:    10.1.7.20
  TOTALCONN: 0000000002  RDY: 001  WLM: 15  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000002
    QOSPLCACT: *DEFAULT*
      W/Q: 15
DEST:         10.1.8.10..23
  DESTXCF:    10.1.7.30
  TOTALCONN: 0000000000  RDY: 001  WLM: 16  TSR: 100
```

```
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 16
3 OF 3 RECORDS DISPLAYED
END OF THE REPORT
```

Example 5-52   Display result of the NETSTAT,VCRT,DETAIL command on SC30

```
D TCPIP,TCPIPA,NETSTAT,VCRT,DETAIL
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.1.8.10..23
  SOURCE:  10.1.100.222..1436
  DESTXCF: 10.1.7.20
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
    INTF:  OSA2080L
      VIPAROUTE:  YES       GW:  10.30.2.242
DEST:      10.1.8.10..23
  SOURCE:  10.1.100.222..1437
  DESTXCF: 10.1.7.20
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
    INTF:  OSA2080L
      VIPAROUTE:  YES       GW:  10.30.2.242
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

Example 5-53   Display result of the NETSTAT,VCRT,DETAIL command on a target stack SC31

```
D TCPIP,TCPIPB,NETSTAT,VCRT,DETAIL
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.1.8.10..23
  SOURCE:  10.1.100.222..1436
  DESTXCF: 10.1.7.20
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
DEST:      10.1.8.10..23
  SOURCE:  10.1.100.222..1437
  DESTXCF: 10.1.7.20
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

Detection and recovery

We performed two test scenarios to verify that the unreachable DVIPA detection and recovery function works correctly:

1. Stop all two OSA interfaces of SC30.
2. Stop the relevant routing process of an intermediate router.

Scenario 1

After we certified that all the stacks were joined, we stopped both OSA interfaces (OSA2080 and OSA20E0) on SC30 and the takeover occurred about 60 seconds after, which is the default TIMERSECS value. We saw Syslog messages, as shown in Example 5-54.

Example 5-54 Syslog message of SC30 and SC31 (Scenario 1)

```
SC30 *EZD1209E TCPIPA DETERMINED THAT ALL MONITORED INTERFACES WERE NOT
      ACTIVE FOR AT LEAST       60 SECONDS                                  1
SC30 *EZZ9676E SYSPLEX PROBLEM DETECTION CLEANUP HAS SUCCEEDED FOR TCPIPA 2
SC31  EZZ8301I VIPA 10.1.8.10 TAKEN OVER FROM TCPIPA ON SC30
```

The following detection and recovery messages are involved:

1. **EZD1209E** indicates that sysplex problem detection has determined that all monitored interfaces are inactive.
2. **EZZ9676E** indicates that sysplex problem detection caused the stack to leave the sysplex group and cleared up all DVIPAs.

We checked the status of Sysplex Distributor, using several NETSTAT commands and a SYSPLEX command, as shown in Example 5-55, Example 5-56, Example 5-57, and Example 5-58.

Example 5-55 Display of XCF group of SC30 (Scenario 1)

```
SC30
D XCF,GROUP,EZBTCPCS
IXC332I  13.29.37  DISPLAY XCF 294
 GROUP EZBTCPCS:
               SC30TCPIPA
               SC31TCPIPB
               SC32TCPIPC 1
```

Example 5-56 Display result of the NETSTAT,VIPADCFG command on SC31 (Scenario 1)

```
SC31
D TCPIP,TCPIPB,NETSTAT,VIPADCFG
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IPADDR/PREFIXLEN: 10.1.8.10
      RANK: 002  MOVEABLE:            SRVMGR:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.9.20/32
      MOVEABLE: IMMEDIATE  SRVMGR: NO
  VIPA DISTRIBUTE: 2
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.11
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.21
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
    DEST:        10.1.8.10..23
      DESTXCF:   10.1.7.31
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
END OF THE REPORT
```

Example 5-57 Display result of the SYSPLEX,VIPADYN command on SC31 (Scenario 1)

```
SC31
D TCPIP,TCPIPB,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V1R11 071
VIPA DYNAMIC DISPLAY FROM TCPIPB   AT SC31
IPADDR: 10.1.9.30
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPC   SC32     ACTIVE
LINKNAME: VIPL0A010914
IPADDR/PREFIXLEN: 10.1.9.20/32
  ORIGIN: VIPADEFINE
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPB   SC31     ACTIVE
LINKNAME: VIPL0A01080A
IPADDR/PREFIXLEN: 10.1.8.10/32
  ORIGIN: VIPABACKUP
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPB   SC31     ACTIVE      BOTH 3
  TCPIPC   SC32     BACKUP 001  DEST
4 OF 4 RECORDS DISPLAYED
```

Example 5-58 Display result of the NETSTAT,HOME command on SC31 (Scenario 1)

```
SC31
D TCPIP,TCPIPB,NETSTAT,HOME
HOME ADDRESS LIST:
LLINKNAME:   VIPL0A01080A
  ADDRESS:  10.1.8.10
    FLAGS: 4
```

Important information include:

1. SC30 TCPIPA left the XCF group of EZBTCPCS.

2. SC31 took the VIPA DISTRIBUTE configuration from SC30.

3. SC31 shifted the BOTH status from the DEST status.

4. SC31 FLAGS is not INTERNAL any longer.

Next, we started an OSA interface OSA2080 of SC30 again. A few seconds later the takeback process occurred and syslog messages were displayed, as shown in Example 5-59.

Example 5-59 Syslog messages of SC30 and SC31 (Scenario 1)

```
SC30 *EZD1212E TCPIPA DELAYING SYSPLEX PROFILE PROCESSING - NO DYNAMIC
      ROUTES OVER MONITORED INTERFACES WERE FOUND                     1
      ...
      ...
SC30  EZD1176I TCPIPA HAS SUCCESSFULLY JOINED THE TCP/IP SYSPLEX GROUP
      EZBTCPCS
SC30  EZZ8302I VIPA 10.1.8.10 TAKEN FROM TCPIPB ON SC31
SC30  EZD1192I THE VIPADYNAMIC CONFIGURATION WAS SUCCESSFULLY RESTORED FOR
      TCPIPA
SC30  EZD1214I INITIAL DYNAMIC VIPA PROCESSING HAS COMPLETED FOR TCPIPA
SC31  EZZ8303I VIPA 10.1.8.10 GIVEN TO TCPIPA ON SC30
```

In this example:

1. **EZD1212E** indicates that TCP/IP joined a sysplex group and finished processing the sysplex definitions when at least one dynamic route over monitored interfaces is found.

Scenario 2

We stopped the relevant routing process of an intermediate router. The takeover occurred about 60 seconds after. However, we received a message that was different from Scenario 1, as shown in Example 5-60.

Example 5-60 Syslog messages of SC30 and SC31 (Scenario 2)

```
SC30 *EZD1210E TCPIPA DETERMINED THAT NO DYNAMIC ROUTES OVER MONITORED
      INTERFACES WERE FOUND FOR AT LEAST       60 SECONDS              1
SC30 *EZZ9676E SYSPLEX PROBLEM DETECTION CLEANUP HAS SUCCEEDED FOR TCPIPA
SC31  EZZ8301I VIPA 10.1.8.10 TAKEN OVER FROM TCPIPA ON SC30
```

In this example:

1. **EZD1210E** indicates that sysplex problem detection has determined that no dynamic routes over monitored interfaces were found.

5.3 Problem determination

To diagnose a Sysplex Distributor problem, you can use a combination of the Netstat command from the system console and display sysplex commands to provide a clear picture of the sysplex. Before beginning problem determination, however, you should understand the entire sysplex environment. Creating a diagram of your configuration, including all the IP addresses for all the paths involved, will be useful. Saving and reviewing all related log files can also be helpful in narrowing down where the problem might be located.

When the problem is actually occurring, you can also use certain commands to help with the problem determination. The NETSTAT command has several sysplex-related reports that can be used to verify the status of your environment.

You can use the following commands to provide the necessary diagnosis information:

- Run the **display Netstat CONFIG/-f** command on the distributing stack and all target stacks to confirm that the correct IPCONFIG and IPCONFIG6 options have been specified.
- Run the **display Netstat VIPADYN/-v** command on the distributing stack to verify that the DVIPA status is ACTIVE and the distribution status is DIST or DIST/DEST.
- Run the **display Netstat VIPADYN/-v** command on the target stacks to verify that they have activated the distributed DVIPA and have it designated as a DEST.
- Run the Sysplex VIPADyn command from any stack in the sysplex to get a global view of how and where DVIPAs are defined within the sysplex and what their status is on each stack. Deactivated DVIPA configurations do not appear in this display.
- Run the **Netstat VDPT/-O** command on the distributing stack to confirm that there are target stacks available with server applications ready.
- Examine the READY (RDY) count fields. The READY (RDY) count is the number of servers that are currently listening on the DVIPA and PORT specified in the DEST: field on the target stack that was identified by the DESTXCF address.

- Check the TotalConn count to see the distribution history. This is a cumulative count of the number of connections that have been forwarded by the distributing stack to each target stack.
- Use the **Netstat VCRT/-V** command on the distributing stack to check whether there are any active connections that are being routed by the distributor. If you run the command with the keyword DETAIL (**d tcpip,tcpcs,net,vcrt,detail**), you can see the policy rule and policy action that each connection maps to.
- Go to the target stacks represented by the DESTXCF ADDR field in the VCRT or VDPT display and run the **Netstat ALLCONN(/-a),IPA=destxcf-addr display** command to see the connections on the target stack.

The following steps are suggested to diagnose a sysplex problem on z/OS Communications Server (based on the information presented in *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782):

1. Determine that all the stacks that you expect to be communicating with are in the same sysplex or subplex (if subplexing is being used). Run the D XCF,GROUP MVS command to determine what groups are being used in the sysplex.
2. For problems where the actual DVIPAs defined on a stack are not what you expected, check the current definitions using the **Netstat VIPADCFG/-F** display command on the distributing stack to confirm that it is configured to distribute the DVIPA and determine how it is to be distributed.
3. For Sysplex Distributor workload monitoring, use the commands **Netstat VDPT/-O** and Netstat VCRT/-V on the distributing stack. If the output from these commands is not what you expect, use the command Sysplex VIPADyn to gain an overall picture of all DVIPA activity in your sysplex.
4. If the output from the command Sysplex VIPADyn reveals an expected target stack not listed for a distributed DVIPA, execute the command **Netstat CONFIG/-f** on the target stack in question. This helps to identify configuration problems on that stack. Note what is required of target stacks. Also use the command **Netstat ALLCONN(/-a),IPA=***destxcf-addr* to verify that a server application has indeed been activated and bound to the correct port.
5. To help follow the flow of packets into and throughout the sysplex, use a CTRACE with options XCF, TCP, and SYSTCPDA on participating stacks to:
   - Identify the connection being received by the distributing stack
   - Determine the stack to which the connection is forwarded
   - Verify the connection being forwarded
   - Determine the expected target stack receiving and processing the connection

   After the connection has been established, subsequent packets can be followed in the same manner. When the connection is terminated, CTRACE records record target stacks, cleans up the connection, and notifies the distributing stack.

For further information about Sysplex Distributor diagnosis, refer to *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782.

6

External application workload balancing

With external application workload distribution, decisions for load balancing are made by external devices. Such devices typically have very robust capabilities and are often part of a suite of networking components. They usually have a significant amount of processing capacity, giving them the ability to make decisions using higher-layer information buried deep inside of datagrams rather than just using IP addresses, including application-specific information such as HTTP (or FTP) URLs.

The latest improvement in this area is the ability of external load balancers to be z/OS sysplex-aware, all the way to the application level. This is achieved by implementing the z/OS Load Balancing Advisor (LBA) solution that uses the Server/Application State Protocol (SASP).

This chapter discusses the following topics.

6.1 Basic concepts of external load balancing

From a z/OS viewpoint, there are two types of external load balancers available today. One type bases decisions completely on parameters in the external mechanism, and the other type uses sysplex awareness matrixes for each application and each z/OS system as part of the decision process. Which technique is best depends on many factors, but the best method usually involves knowledge of the health and status of the application-instances and the z/OS systems. Enhancements in Workload Manager (WLM) allow the export of status data for each application instance, as well as the z/OS system itself.

The content in this chapter is based on workload balancing with the use of switches and a Content Switching Module (CSM) as the load balancing device. The CSM is one of the external load balancers that supports SASP.

If optimized multi-tier application load balancing is planned within an intra-sysplex environment in addition to external load balancing, refer to Chapter 7, “Intra-sysplex workload balancing” on page 213 for further information about this topic.

6.1.1 Understanding directed mode load balancing

Directed mode load balancing works by changing the destination IP address of the inbound IP packets to the IP address of the chosen target stack. One of the advantages of directed mode is that there can be any number of router hops in between the load balancer and the target stacks.

A characteristic of directed mode is that outbound packets must be directed back through the load balancer so that the load balancer can change the source IP address of the outbound IP packets from the IP address of the target stack to the cluster IP address. If this is not done, the remote client would reject the response because it would appear to come from an IP address other than the one it sent a request to. Another reason that outbound packets must be directed back using the load balancer is that the load balancer keeps track of the flows passing through it and this information is used for fault tolerance purposes.

There are two ways to ensure outbound routing back through the load balancer:

1. The load balancer can be configured to change only the *destination* IP address in inbound IP packets, and not the source IP address. In this case the target stack responds with IP packets back directly to the client IP address. These packets must be directed to the load balancer for it to change the *source* IP address in the outbound packets. This can be achieved in two ways:
   - By having all outbound packets routed through the load balancer (using a default route definition on the target stacks).
   - By having the routing infrastructure between the load balancer and the target node implement policy-based routing (PBR) where the routers recognize IP packets from the clustered servers (based on source IP address, source port number, or both). In this configuration, the router sends only those packets back through the load balancer, while outbound packets for workload that were not balanced are routed directly back to the clients. This is known as server NAT mode and is illustrated in Figure 6-1.

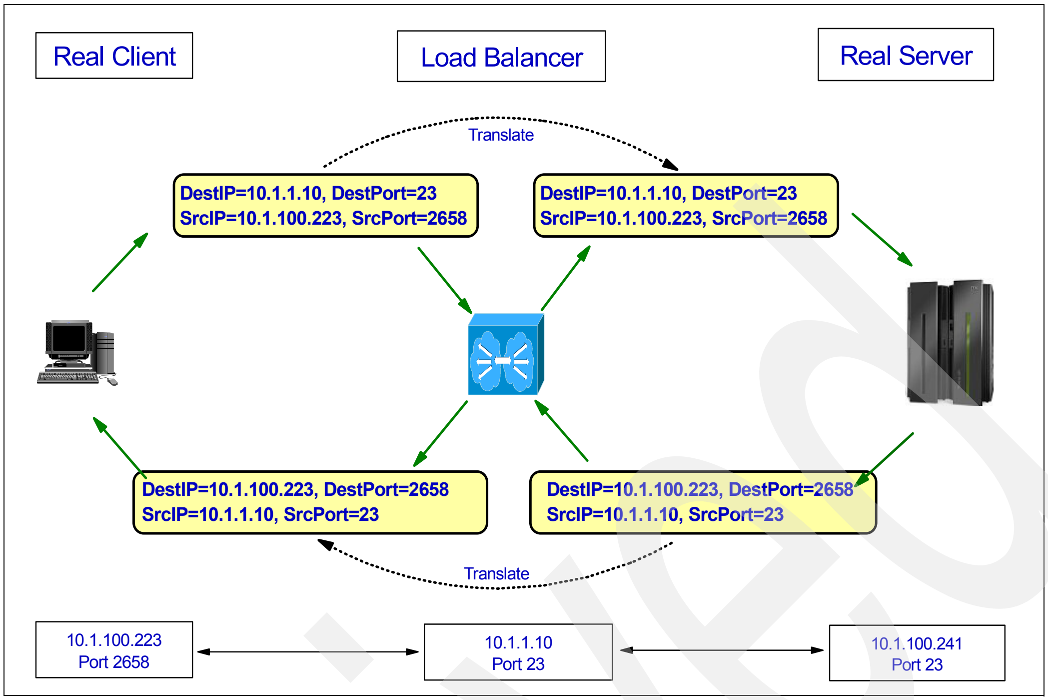


Figure 6-1 Server NAT with policy-based routing

2. You can configure the load balancer to change both the destination IP address and the source IP address in the inbound IP packets, known as server NAT and client NAT mode. The advantage of this method is that outbound IP packets from the target stacks are sent with the destination IP address of the load balancer. The load balancer then changes the packet to match the original cluster IP address as source and client IP address as destination. This is illustrated in Figure 6-2.

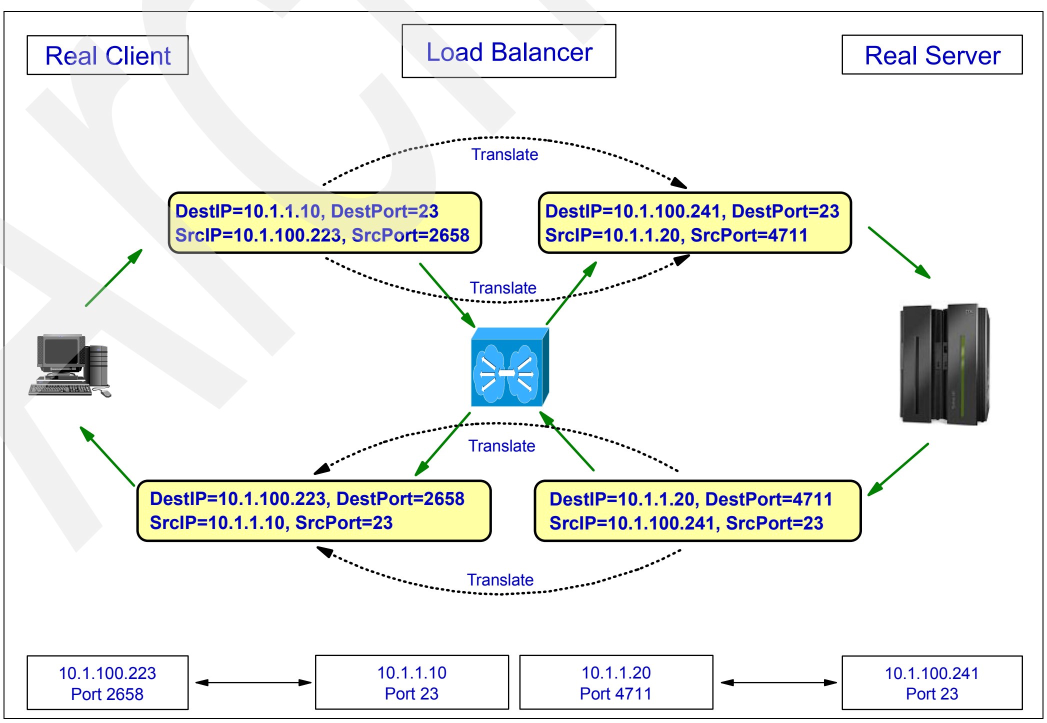


Figure 6-2 Server NAT with client NAT

From a target server view, it appears that all connections come from one client source IP address (that of the load balancer), but each client connection comes from a different source port number on that load balancer. One consequence is that servers cannot see the real client IP address, and this can complicate diagnosing network-related problems.

Client NATing can have an impact on z/OS networking policy functions. Networking policies on z/OS might apply network QoS differentiated services, selection of Intrusion Detection Service (IDS) actions, and Traffic Regulation (TR) limitations. z/OS networking policies are specified using policy conditions and actions. The conditions can be defined in various ways, one of which is to use the client source IP address.

One example is to apply high-priority outbound treatment to traffic that is destined for a specific user community, such as all IP addresses that belong to the client query department. If client NATing is used by the load balancer, all inbound packets appear to come from one IP address (that of the load balancer) and networking policy conditions that were defined based on the real client IP addresses are not applied to that traffic.

Another z/OS function that is impacted by client NATing is a NETACCESS rule. A NETACCESS rule can be used to authorize individual z/OS users to communicate with selected sections of a network identified by prefix definitions in the NETACCESS policies in the z/OS TCP/IP configuration files. NETACCESS rules might also be used to assign Multi Level Security (MLS) labels to inbound IP packets in an MLS-enabled security environment.

If z/OS networking policies are based on client IP addresses, or if NETACCESS rules are in use, client NATing should be used with care, because it might disrupt the operation of those functions.

To complete this discussion, care is also needed when using client NATing to TN3270 servers that use the client source IP address or host name to choose TN3270 connection options, such as selecting an LU name or a primary SNA application for a given connection.

6.1.2 z/OS Load Balancer Advisor

The z/OS Load Balancer Advisor is a component that allows any external load balancing solution to become sysplex-aware. The external load balancer must support the Server Application State Protocol (SASP) to obtain sysplex information through this function. The general Load Balancing Advisor flow is illustrated in Figure 6-3.

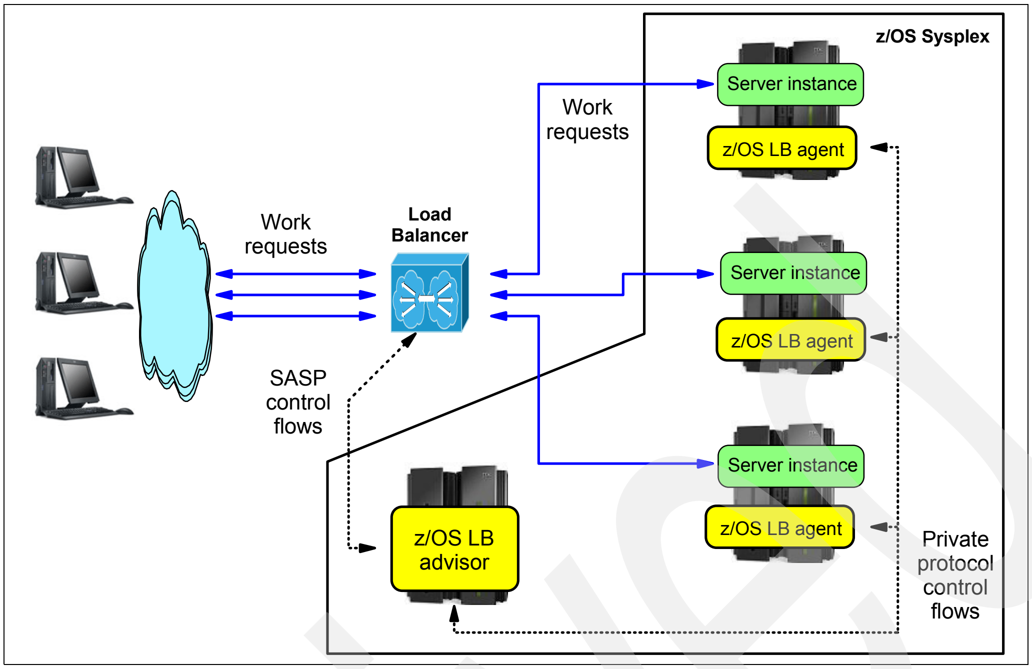


Figure 6-3 z/OS Load Balancing Advisor overview

The z/OS Load Balancing Advisor consists of two z/OS applications:

- z/OS z/OS Load Balancing Advisor

  The advisor is an MVS started task and has one primary instance per sysplex. A backup advisor can be implemented as a secondary instance. The advisor collects information from z/OS Load Balancing Agents in the sysplex using a private protocol through a TCP connection. This z/OS sysplex awareness information is then communicated to the external load balancers using the SASP protocol.

  The advisor provides awareness information from any TCP/UDP server application within the sysplex. It acts as the SASP server using TCP Port 3860 (configurable) and can communicate with multiple external load balancers.

- z/OS z/OS Load Balancing Agent

  An agent is an MVS started task and has one instance per system. The agent collects information about z/OS and applications. The information is then sent to the advisor using the private protocol.

The sysplex awareness information is collected by the advisor and sent to the external load balancer so the load balancer can decide which application instance is best for the next request. The recommendations are derived from the following key components:

- State of the target application and system

  This includes an indication of whether the target application and target system are currently active, enabling the load balancer to exclude systems that are not active or do not have the desired application running.

- z/OS Workload Management (WLM) system-wide recommendations

  WLM recommendations provide a relative measure of a target system's ability to handle new workload, as compared to other target systems in the sysplex. WLM recommendations are derived from many measurements, including each system's available CPU capacity or capacity that can be displaced by higher importance workloads. The latter is important where systems might be 100% utilized, but are running work that has a lower importance (as defined by the WLM policy) and can be displaced by higher importance workload.

- z/OS WLM server-specific recommendations

  These recommendations are similar to the WLM system-wide recommendations, but also contain an indication of how well individual server applications are doing compared to the WLM policy goals that have been specified for that workload. These recommendations can be very useful in helping load balancers avoid selecting application servers that are experiencing performance problems (that is, not meeting the specified WLM policy goals).

- Application server health from a TCP/IP perspective

  TCP/IP statistics for target applications are monitored to determine whether specific server applications are encountering problems keeping up with the current workload. For example, are requests being rejected because the backlog queue is full? In situations such as this, the load balancer can direct fewer connections to any application that is experiencing these problems. Recommendations are provided for both UDP and TCP server applications. These recommendations are referred to as Communications Server weights in this chapter.

The information sent to the load balancer is represented in the form of a weight that is composed of two main elements:

- WLM weight

  This refers to the WLM weight that is known from other WLM-based load balancing solutions, such as Sysplex Distributor. The WLM weight has a numeric value between 0 and 64.

- Communications Server weight

  The Communications Server weight is calculated based on the availability of the actual application instances (are they up and ready to accept workload) and how well TCP/IP and the individual application instance process the workload that is sent to them. The Communications Server weight has a numeric value between 0 and 100.

  There are several reasons for the calculation of the Communications Server weight. One reason is to prevent a stalled application instance from being sent more work. Another is to proactively react to an application instance that is becoming overloaded. It can accept new connections, but the size of the backlog queue increases over time and is approaching the max backlog queue size.

The final weight that is sent to the load balancer is calculated by combining the WLM and Communications Server weights into a single metric:

```
Final weight = (WLM weight * CS weight) / 100
```

6.1.3 Server/Application State Protocol (SASP)

The Server/Application State Protocol (SASP) is an open protocol. The SASP protocol is used for communication between external load balancers and the advisor. The information that is exchanged includes:

- Registration of members from external load balancers that are interested in load balancing
- Requests from external load balancers
- Recommendations to external load balancers
- Deregistration of members from external load balancers that are no longer interested in load balancing

The registration process covers two types of groups, system-level groups and application-level groups, as explained here:

- System-level groups

  This group is only represented by a list of IP addresses that are matched to TCP/IP stacks in the sysplex. The group includes WLM weights for the logical partition (LPAR). The Communications Server weight indicates whether the IP address is active in the sysplex. A value of 0 means quiesced and a value of 100 means not quiesced. Load Balancing Advisor displays show a protocol value of 0 for system-level groups.

- Application-level groups

  This group is represented by a list of IP addresses, protocols (TCP and UDP), and ports that are matched to server address spaces. The group includes WLM weights for the server address spaces. Communications Server weights indicate how well the server instances are performing. Load Balancing Advisor displays show protocols as TCP or UDP with the registered port numbers.

When the external load balancer connects to the advisor, it indicates how it wants the information exchanged. This gives the load balancer the ability to select the best fit for itself. The choices available are pull model and push model:

- Pull model

  The advisor suggests a frequency interval, but it is up to the load balancer to choose the frequency with which it wants to receive the weights.

- Push model

  The load balancer requests the advisor to push weights down at certain intervals or when the weights change.

For both models, the recommendations (weights) can be sent for all registered members or only for those with changes to their weights.

With more frequent updates, the workload is distributed more accurately. The cost of more frequent updates is more flows in the network, both between Load Balancing Advisor and agents, and Load Balancing Advisor and external load balancers, as well as cycles used in Load Balancing Advisor and load balancers.

6.1.4 External load balancer without LBA/SASP

There are several products available today that perform load balancing of IP traffic. The example in this chapter was built using the Content Switching Module (CSM) and has the general structure shown in Figure 6-4.

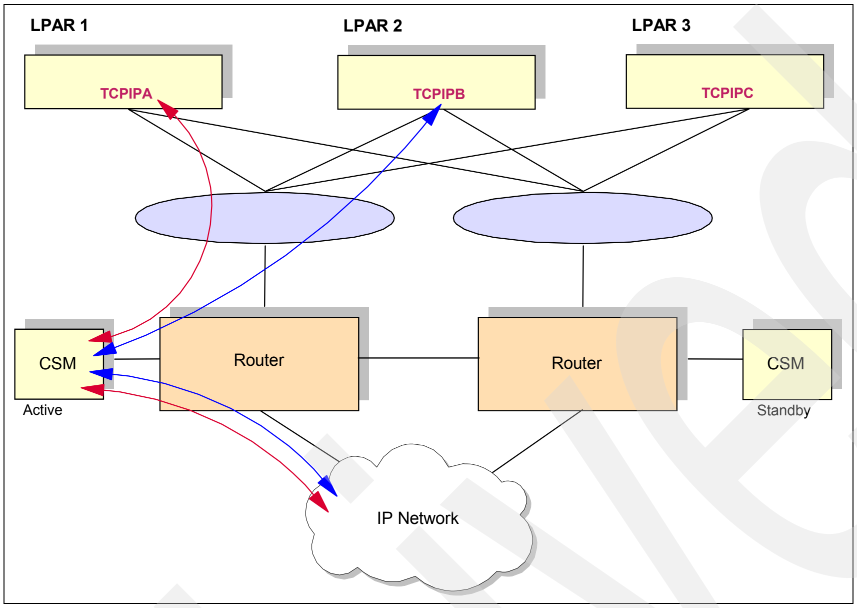


Figure 6-4 External load balancing without LBA/SASP

The external load balancer function is implemented as an active or standby pair for fault tolerance. The CSMs are used to exchange state information so that the existing connections continue nondisruptively in case of a failure in the active CSM.

The external load balancer receives user requests for the application cluster and forwards the requests to an application instance within the application cluster, according to the configuration in the load balancer. The load balancer is responsible for deciding which application instance to use for each new connection. Remember that, in this design, the decision is made without any knowledge of the current workload inside z/OS and the application.

A way for the load balancer to keep track of the availability of each application instance is to poll each application instance. This polling technique is handled in several ways and is different for different vendors. An example of a simple polling technique is Internet Control Message Protocol (ICMP), where the load balancer assumes that the application instance is operating correctly if it receives an ICMP reply from the application instance IP address.

6.1.5 External load balancer with LBA/SASP

The example for this mode was built using a product that supports SASP. The general structure of the example is shown in Figure 6-5.

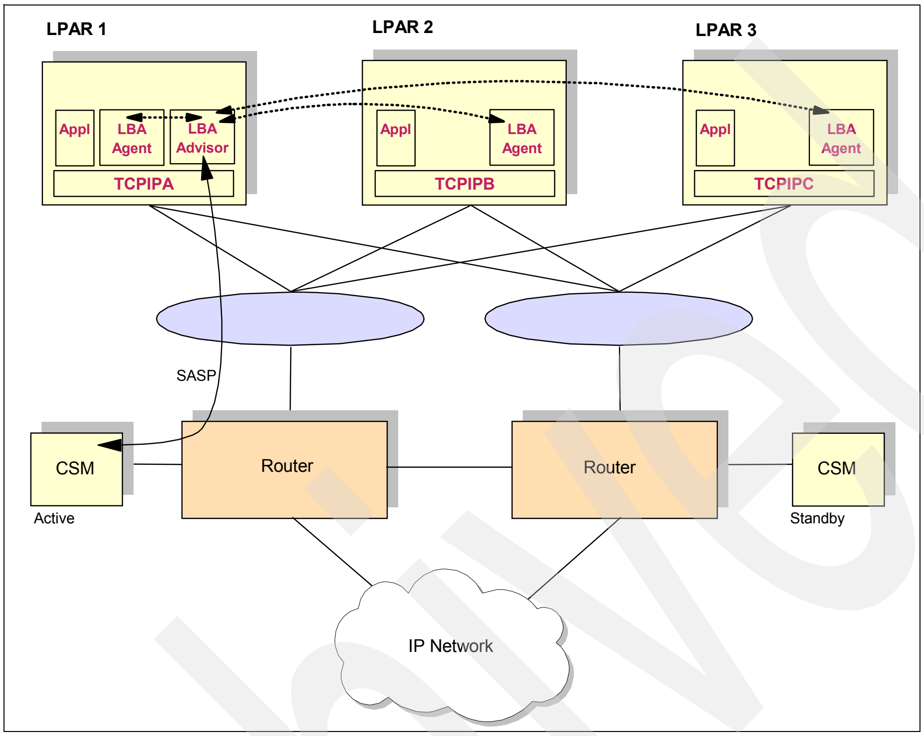


Figure 6-5 External load balancing with LBA/SASP

z/OS sysplex assists the external load balancer with recommendations as to which application instance is the best candidate for the next request. This is done by implementing the z/OS Load Balancing Advisor solution that communicates the recommendations to the external load balancer using the SASP protocol. The load balancer must support the SASP protocol and include the recommendations from Load Balancing Advisor when it decides which application instance to use for the next request. The external load balancer function is implemented as an active or standby pair for fault tolerance. The external load balancer exchange state information so that existing connections continue nondisruptively in case of a failure in the active external load balancer.

The external load balancer receives user requests for the application cluster and forwards the requests to an application instance within the application cluster according to the configuration in the load balancer. The load balancer can combine the recommendations from Load Balancing Advisor with its own techniques, such as ICMP polling.

6.1.6 Importance of external application workload balancing

There are many reasons for choosing an external device to be responsible for load balancing for a z/OS sysplex environment. These include:

- A requirement for a single load balancing solution that is used across multiple platforms.
- The administration of the load balancing functions can be done without the need for specific z/OS skills.

- A requirement for content-based load balancing using higher-layer information, including application-specific information such as an HTTP (or FTP) URLs, rather than just using IP addresses.
- A need to minimize processor usage on System z9® servers.

External workload balancing offers a different dimension to the z/OS sysplex environment, especially when the load balancer becomes z/OS sysplex-aware. It might be desirable to transform a normal z/OS sysplex environment into z/OS application server farms and to move the responsibility for load balancing of the z/OS server farms to an external environment that is responsible for Content Switching.

In the remainder of this chapter, we will describe two scenarios with implementation examples. The scenarios include:

- External load balancer without LBA/SASP
- External load balancer with LBA/SASP

6.2 External load balancer without LBA/SASP

This section describes the implementation of external load balancing for three application instances of TN3270 in a z/OS sysplex environment without LBA/SASP.

The infrastructure eliminates single network point of failure, by using the following:

- Two OSA-Express Gigabit Ethernet adapters (one port used per physical OSA Card)
- Two switches
- Two external load balancer with stateful failover
- OSPF to ensure rerouting around failures

Our example uses a single server, which presents a single point of failure (SPOF). However, a real production environment would use multiple servers. Figure 6-6 shows our general structure.

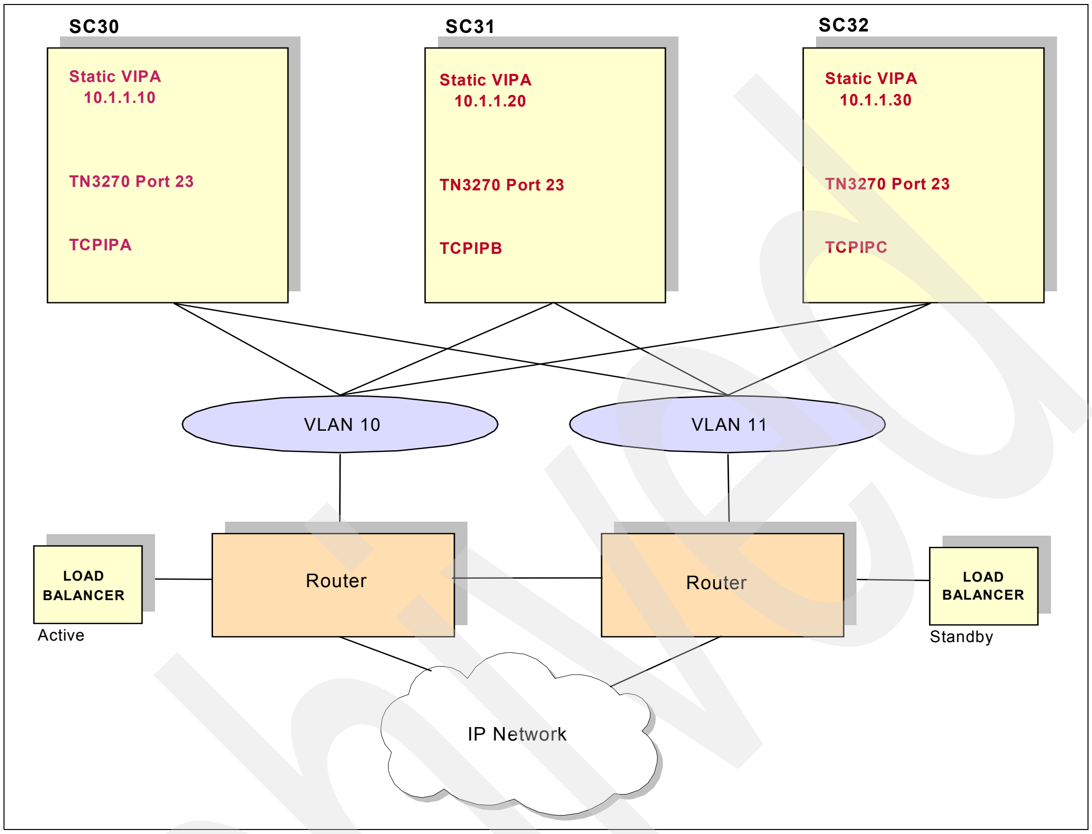


Figure 6-6 External load balancing of TN3270 without LBA/SASP

Environment

Our specific environment contains the following components:

- Three LPARS
- Two OSA-Express Gigabit Ethernet adapters, with shared usage
- A TN3270 server in each LPAR
- TN3270 servers using static VIPAs
- OSPF as the dynamic routing protocol
- Policy-based routing (PBR) used in the routers
- Two switches
- Two external load balancer in active/standby mode

A static VIPA is used to address each instance of a TN3270 server. The load balancer can sense the presence of the VIPA and check whether the TN3270 server is available.

Dependencies

Available application instances must be configured in the external load balancer. Our example uses server NAT and policy-based routing.

Advantages

No single points of failure exist in the network elements of this design.

Considerations

If application instances are intended to move between LPARs, we would use a dynamic VIPA. We did not include this function in out example.

6.2.1 External load balancer without LBA/SASP implementation

Our work plan started with a list of implementation tasks.

Implementation tasks

Our key tasks were as follows:

1. Plan the IP addressing needed for our configuration.
2. Configure the TN3270 server IP and port addresses.
3. Define the load balancer active and standby configurations.

Table 6-1 shows the IP addressing plan that we used.

Table 6-1 IP addressing scheme

Application	Application cluster address	Application instance IP address
TN3270 - Port 23	10.1.60.10	SC30: 10.1.1.10 SC31: 10.1.1.20 SC32: 10.1.1.30

The key TCP/IP profile statements for our three LPARs are shown here. Example 6-1 shows the profile configuration in SC30.

Example 6-1 TCP/IP profile configuration in SC30

```
PORT
23 TCP TN3270A NOAUTOLOG            ; MVS Telnet Server 1
HOME

   10.1.1.10     VIPA1L 2
   10.1.2.11     OSA2080L
   10.1.3.12     OSA20E0L
```

Example 6-2 shows the profile configuration in SC31.

Example 6-2 TCP/IP profile configuration in SC31

```
PORT
23 TCP TN3270B NOAUTOLOG            ; MVS Telnet Server 1
HOME

   10.1.1.20     VIPA1L 2
   10.1.2.21      OSA2080L
   10.1.3.22      OSA20E0L
```

Example 6-3 shows the profile configuration in SC32.

Example 6-3 TCP/IP profile configuration in SC32

```
PORT
23 TCP TN3270C NOAUTOLOG             ; MVS Telnet Server 1

HOME
   10.1.1.30     VIPA1L 2
   10.1.2.31     OSA2080L
   10.1.3.32     OSA20E0L
```

Our active CSM definitions are shown in Example 6-4. Note that 1 indicates TN3270 binding to INADDR_ANY, and 2 indicates static VIPA.

Example 6-4 Active CSM definitions

```
module ContentSwitchingModule 2 1
 variable ROUTE_UNKNOWN_FLOW_PKTS 1
 variable SASP_CSM_UNIQUE_ID CSM-6509A
 variable SASP_GWM_BIND_ID_MAX 5
!
 ft group 61 vlan 61 2
  priority 101
!
 vlan 60 server 3
ip address 10.1.60.2 255.255.255.0
  route 0.0.0.0 0.0.0.0 gateway 10.1.60.240
  alias 10.1.60.1 255.255.255.255
!
 probe PING-30S icmp 4
  interval 30
  retries 2
  failed 30
!
 real SC30-TN3270 5
  address 10.1.1.10
  inservice
 real SC31-TN3270
  address 10.1.1.20
  inservice
 real SC32-TN3270
  address 10.1.1.30
  inservice
!
 serverfarm TN3270 6
  nat server
  no nat client
  bindid 65521
  real name SC30-TN3270
   inservice
  real name SC31-TN3270
   inservice
  real name SC32-TN3270
   inservice
!
```

```
 vserver TN3270 7
  virtual 10.1.60.10 tcp telnet
  serverfarm TN3270
  replicate csrp connection
  persistent rebalance
  inservice
!
 dfp
 agent 10.1.9.30 3860 65521
!
interface Vlan10 8
 ip address 10.1.2.240 255.255.255.0
 ip policy route-map pbr-to-csm
!
interface Vlan60 9
 description VLAN 60 for CSM
 ip address 10.1.60.240 255.255.255.0
 ip ospf cost 5
!
interface Vlan61 10
 description CSM Failover
 no ip address
!
router ospf 100 11
 router-id 10.1.3.240
 log-adjacency-changes
 area 2 stub no-summary
 network 10.1.2.0 0.0.0.255 area 2
 network 10.1.3.0 0.0.0.255 area 2
 network 10.1.100.0 0.0.0.255 area 2
 network 10.1.0.0 0.0.255.255 area 0
 network 10.200.1.0 0.0.0.255 area 0
 default-information originate always metric-type 1
!
ip access-list extended tn3270 12
 permit tcp host 10.1.1.30 eq telnet any
 permit tcp host 10.1.1.20 eq telnet any
 permit tcp host 10.1.1.10 eq telnet any
!
route-map pbr-to-csm permit 10 13
 match ip address tn3270
 set ip next-hop 10.1.60.1
```

Note the following key elements in the definitions:

1. CSM is in module 2 of this switch.

2. Fault tolerance uses dedicated virtual local area network (VLAN) 61.

3. CSM server VLAN. Subnet 10.1.60.0/24. This is the subnet for CSM and application clusters. There is no routing within the CSM, only static routes pointing to the routers.

4. This is the ICMP probe used for polling a given application instance for availability.

5. This is the real server definition pointing to the static VIPA per LPAR.

6. This is the server farm definition grouping real servers together and probe indicating *polling* of each real server. If a real server refuses to answer, the real server is marked unavailable.

7. This is the virtual server representing the application cluster. *Replicate csrp connection* implements stateful failover between the CSMs without connection loss of existing connections.

8. VLAN 10. Subnet 10.1.2.0/24. This is the OSA-Express Gigabit Ethernet attachment of z/OS. Policy-based routing is enabled in this VLAN.

9. VLAN 60. Subnet 10.1.60.0/24. This is the subnet for CSM and application clusters.

10. VLAN 61. This is the dedicated VLAN for CSM failover services.

11. The OSPF process 100 includes subnets matching 10.1.0.0/16.

12. This is the extended access list used by policy-based routing.

13. This is the Route Map used by policy-based routing. This ensures that datagrams from extended access list tn3270 are being forwarded back to the CSM.

The standby definitions, shown in Example 6-5, are slightly different.

Example 6-5 Standby CSM definitions

```
module ContentSwitchingModule 3 1
 variable ROUTE_UNKNOWN_FLOW_PKTS 1
!
 ft group 61 vlan 61 2
  priority 100
!
 vlan 60 server 3
  ip address 10.1.60.3 255.255.255.0
  route 0.0.0.0 0.0.0.0 gateway 10.30.60.4
  route 0.0.0.0 0.0.0.0 gateway 10.30.60.5
  alias 10.1.60.1 255.255.255.255
!
 probe PING-30S icmp 4
  interval 30
  retries 2
  failed 30
!
 real SC30-TN3270 5
  address 10.1.1.10
  inservice
 real SC31-TN3270
  address 10.1.1.20
  inservice
 real SC32-TN3270
  address 10.1.1.30
  inservice
 real TN3270-SC30
  inservice
!
 serverfarm TN3270 6
  nat server
  no nat client
  real name SC30-TN3270
   inservice
```

```
  real name SC31-TN3270
   inservice
  real name SC32-TN3270
   inservice
  probe PING-30S
!
 vserver TN3270 7
  virtual 10.1.60.10 tcp telnet
  serverfarm TN3270
  replicate csrp connection
  persistent rebalance
  inservice
!
interface Vlan31 8
 ip address 10.1.3.11 255.255.255.0
 ip policy route-map pbr-to-csm
!
interface Vlan60 9
 description VLAN 60 for CSM
 ip address 10.1.60.241 255.255.255.0
 ip ospf cost 5
!
interface Vlan61 10
 description CSM Failover
 no ip address
!
router ospf 100 11
 router-id 10.1.2.220
 log-adjacency-changes
 area 1 stub no-summary
 network 10.30.0.0 0.0.255.255 area 1
!
ip access-list extended tn3270 12
 permit tcp host 10.1.1.30 eq telnet any
 permit tcp host 10.1.1.20 eq telnet any
 permit tcp host 10.1.1.10 eq telnet any!
!
route-map pbr-to-csm permit 10 13
 match ip address tn3270
 set ip next-hop 10.1.60.1
!
```

The standby definitions include the following key points:

1. CSM is in module 2 of this switch.
2. Fault tolerance uses dedicated VLAN 61.
3. CSM server VLAN. Subnet 10.1.60.0/24. This is the subnet for CSM and application clusters. There is no routing within the CSM, only static routes pointing to the routers.
4. This is the ICMP probe used for *polling* a given application instance for availability.
5. This is the real server definition pointing to the static VIPA per LPAR.
6. This is the server farm definition grouping real servers together and probe indicating polling of each real server. If a real server refuses to answer, the real server is marked unavailable.

7. This is the virtual server representing the application cluster. *Replicate csrp connection* implements stateful failover between the CSMs without connection loss of existing connections.

8. VLAN 31. Subnet 10.1.3.0/24. This is the OSA-Express Gigabit Ethernet attachment of z/OS. Policy-based routing is enabled in this VLAN.

9. VLAN 60. Subnet 10.1.60.0/24. This is the subnet for CSM and application clusters.

10. VLAN 61.This is the dedicated VLAN for CSM failover services.

11. The OSPF process 100 includes subnets matching 10.1.0.0/16.

12. This is the extended access list used by policy-based routing.

13. This is the Route Map used by policy-based routing. It ensures that datagrams from extended access list tn3270 are being forwarded back to the CSM.

Verification

We verified our setup using the following steps:

1. Verify that TN3270 servers are started in each LPAR.
2. Check the CSM load-balancing environment.
3. Start 12 TN3270 clients and observe the results.

We verified that TN3270 servers were running, as shown in Example 6-6. (The command in this example was repeated in all three LPARs.)

Example 6-6 TN3270 is ready

```
D TCPIP,TCPIPC,N,CONN,PORT=23
USER ID  CONN     STATE
TN3270C  00000030 LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
1 OF 1 RECORDS DISPLAYED
```

The command in Example 6-7 displayed the active vserver configuration.

Example 6-7 List active vserver configuration

```
Router# sh mod csm 3 vservers name tn3270-basic config
TN3270-BASIC, type = SLB, state = OPERATIONAL, v_index = 11
  virtual = 10.30.60.11/32:23 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = connection, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 0
  Policy <default>
    serverfarm TN3270-BASIC, type = SLB, predictor = RoundRobin 1
      nat = SERVER
      bind id = 0, fail action = none
      inband health config: <none>
      retcode map = <none>
      Probes:
        PING-30S, type = icmp
      Real servers: 2
        SC30-TN3270, weight = 8, OPERATIONAL
        SC31-TN3270, weight = 8, OPERATIONAL
        SC32-TN3270, weight = 8, OPERATIONAL
Router#
```

Note that 1 distribution is round-robin, and 2 all three application instances have the same weight and are operational.

The command in Example 6-8 displays the active server farm configuration.

Example 6-8 List the active server farm configuration

```
Router# sh mod csm 3 serverfarms name tn3270-basic detail
TN3270-BASIC, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 0, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING-30S, type = icmp
  Real servers:
    SC30-TN3270, weight = 8, OPERATIONAL, conns = 0
    SC31-TN3270, weight = 8, OPERATIONAL, conns = 0
    SC32-TN3270, weight = 8, OPERATIONAL, conns = 0
  Total connections = 0

Router#
```

The command in Example 6-9 displays the real servers being used.

Example 6-9 List active real servers

```
Router# sh mod csm 3 reals sfarm tn3270-basic detail
SC30-TN3270, TN3270-BASIC, state = OPERATIONAL
  address = 10.30.1.230, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0
SC31-TN3270, TN3270-BASIC, state = OPERATIONAL
  address = 10.30.1.241, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0
SC32-TN3270, TN3270-BASIC, state = OPERATIONAL
  address = 10.30.1.221, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0
Router#
```

We then started 12 TN3270 clients (to application cluster address 10.30.60.11) and displayed the distribution using the command in Example 6-10.

Example 6-10 Display of the vserver in the CSM shows how the 12 connections are distributed

```
Router# sh mod csm 3 serverfarms name tn3270-basic detail
TN3270-BASIC, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 0, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING-30S, type = icmp
  Real servers: 1
    SC30-TN3270, weight = 8, OPERATIONAL, conns = 4
    SC31-TN3270, weight = 8, OPERATIONAL, conns = 4
    SC32-TN3270, weight = 8, OPERATIONAL, conns = 4
  Total connections = 12

Router#
```

Note that 12 connections 1 appear to be distributed round-robin.

We verified the TN3270 server usage by displaying the connections in each LPAR. Example 6-11 shows the connections in SC30.

Example 6-11 TN3270 connections in SC30

```
D TCPIP,TCPIPC,N,CONN,PORT=23

USER ID  CONN     STATE
TCPIPC   000000C8 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.230..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2526
TCPIPC   00000021 LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
TCPIPC   000000CC ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.230..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2529
TCPIPC   000000D0 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.230..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2535
TCPIPC   000000CE ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.230..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2532
5 OF 5 RECORDS DISPLAYED
```

Example 6-12 shows the connections in SC31.

Example 6-12 TN3270 connections in SC31

```
D TCPIP,TCPIPC,N,CONN,PORT=23

USER ID  CONN     STATE
TCPIPC   000000DC ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.241..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2527
TCPIPC   00000023 LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
TCPIPC   000000DE ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.241..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2530
TCPIPC   000000E2 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.241..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2536
TCPIPC   000000E0 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.241..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2533
5 OF 5 RECORDS DISPLAYED
```

Example 6-13 shows the connections in SC32.

Example 6-13 TN3270 connections in SC32

```
D TCPIP,TCPIPC,N,CONN,PORT=23

USER ID  CONN     STATE
TCPIPC   000000C0 ESTBLSH
```

```
  LOCAL SOCKET:   ::FFFF:10.30.1.221..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2525
TCPIPC   00000020 LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
TCPIPC   000000C6 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.221..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2528
TCPIPC   000000CA ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.221..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2534
TCPIPC   000000C8 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.1.221..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..2531
5 OF 5 RECORDS DISPLAYED
```

Problem determination

When we ran into problems, we performed our problem determination using the following steps:

1. Check that application instances are running in z/OS (active listener.)
2. Check that there is connectivity between router and z/OS.
3. Check that the real server is operational.
4. Check that the virtual server is operational.
5. Check for connectivity between client and CSM (application cluster address = vserver address).
6. Check that policy-based routing definitions are in place.
7. Run packet trace.
8. Run CTRACE.
9. Use a network analyzer.
10. Run debug in the external networking devices.

6.3 External load balancer with LBA/SASP

This section describes the implementation of external load balancing for three application instances of TN3270 servers in a z/OS sysplex environment, using LBA/SASP. However, a real production environment would use multiple servers. Figure 6-7 shows the general design.

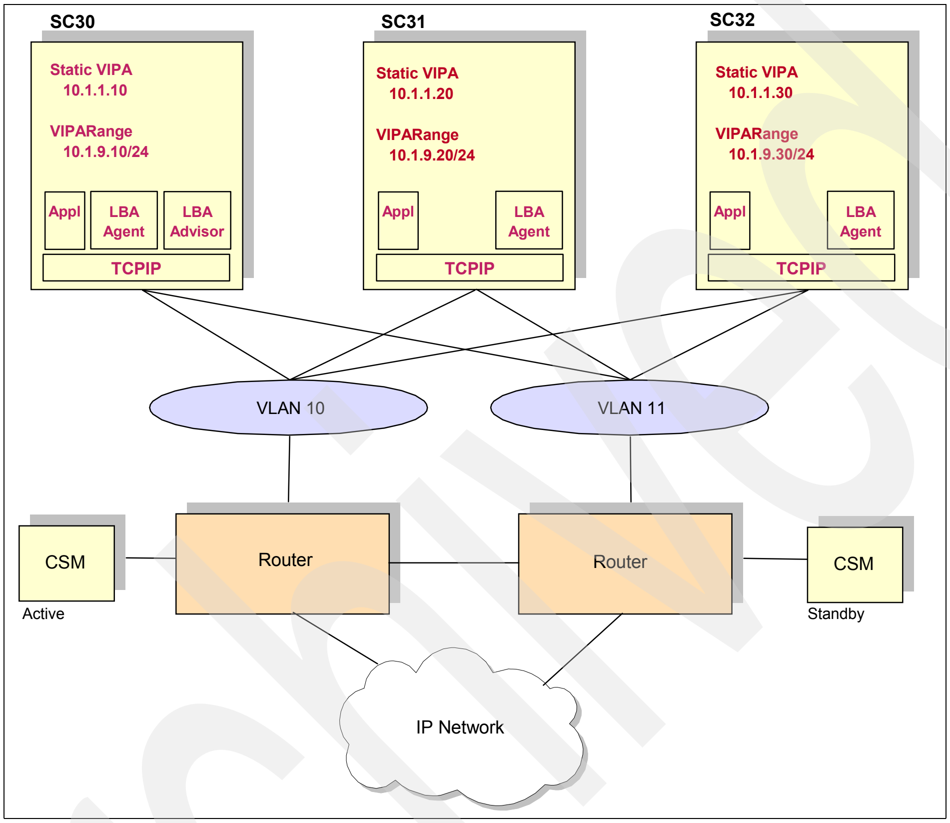


Figure 6-7 External load balancing with LBA/SASP

Environment

The specific environment for this example includes the following components:

- Three LPARS
- Two OSA-Express Gigabit Ethernet adapters, with shared usage
- z/OS Load Balancing Advisor running in one LPAR
- Backup z/OS Load Balancing Advisor can start in another LPAR
- z/OS Load Balancing Agent running in each LPAR
- TN3270 server running each LPAR
- TN3270 uses static VIPA
- OSPF used as the dynamic routing protocol
- PBR used in the routers
- Two switches
- Two CSM in active/standby mode

In this scenario, LBA/SASP is used to send load balancing recommendations to the external load balancer. A static VIPA is used by each instance of a TN3270 server. The CSM can sense the presence of the VIPA to check whether TN3270 is available.

Dependencies

The external load balancer must support SASP for this design. Available application instances must be configured in the external load balancer. Our example uses CSM with server NAT and policy-based routing.

Advantages

No single network points of failure are present in this design. The workload is distributed in a more accurate manner according to current workload of the application instances and the z/OS system.

Considerations

If application instances are intended to move between LPARs, we would use a dynamic VIPA. We did not include this function in our example.

6.3.1 TLS/SSL for z/OS Load Balancing Advisor

The Advisor, Agents, and ADNR are authorized programs which must be started from a start procedure. The ability to establish a connection to the Advisor needs to be restricted to “authorized parties” as sensitive interfaces can be exploited after a connection is accepted by the Load Balancing Advisor. You need to ensure that only Load Balancing Agents that IBM provides are allowed to connect to the Agent listening port. Agents are responsible for providing sensitive information that indicates server application availability, health and performance. You need to ensure that only authorized load balancers are allowed to connect to the Advisor on the external load balancer SASP port. The Advisor to load balancer interface can be used to obtain sensitive information regarding TCP/IP applications deployed in a sysplex, processor utilization information for each system, and so on.

The data flowing on the Advisor’s connections, which includes server application availability, health, and performance, might need to be encrypted. An AT-TLS policy can specify encryption for data flowing outside of the TCP/IP stack.

Using TLS/SSL technologies one can secure and control access to all communications with the Load Balancing Advisor. The TLS/SSL support for the Load Balancing Advisor, Agents and the ADNR function is provided using the AT-TLS feature of the Communications Server. You have the ability to perform access control checks using SAF-compliant security product profiles.

Client certificates can authenticate external load balancers, z/OS Load Balancing Agents and ADNR clients connecting to the Load Balancing Advisor.

You can use a combination of TLS/SSL and non-TLS/SSL connections to the Advisor. Your availability of the advisor and agents can be improved by removing some configuration statements and in some cases, a recycle of the Advisor was required, because dynamic reconfiguration is not supported.

Using AT-TLS, an Agent or load balancer instance can be added without impacting the Advisor.

Implementing Load Balancing Advisor AT-TLS

When implementing load balancing advisor using AT-TLS, complete the following tasks:

1. Define the necessary SERVAUTH profiles and permit authorized users to these profiles. If the appropriate SAF profile is defined, then only authorized Agents, external load balancers, and ADNR are allowed to connect to the Advisor.

   ```
   EZB.LBA.LBACCESS.sysname.tcpsysplexgroupname
   EZB.LBA.AGENTACCESS.sysname.tcpsysplexgroupname
   ```

   The user ID that is associated with each external load balancer or agent must have READ access to either of the SERVAUTH profile.

2. Remove the following Advisor configuration file statements:

   ```
   agent_id_list
   lb_id_list
   ```

3. Remove agent configuration statement:

   ```
   host_connection
   ```

Recommendations

If you choose to use an external load balancing solution for the z/OS environment, we recommend the z/OS Load Balancing Advisor and SASP approach.

6.3.2 External load balancer with LBA/SASP implementation

Our work started with listing the implementation tasks.

Implementation tasks

The tasks for implementing external load balancing with LBA/SASP are as follows:

1. Create an IP addressing plan.
2. Configure the TCP/IP portion of the z/OS Load Balancing Advisor and Agents.
3. Configure the z/OS Load Balancing Advisor.
4. Configure the z/OS Load Balancing Agents.
5. Configure the active CSM.
6. Configure the standby CSM.

Table 6-2 shows out IP addressing plan.

Table 6-2 IP addressing scheme

Application	Application cluster address	Application instance IP address
TN3270 - Port 23	10.1.60.10	SC30: 10.1.1.10 SC31: 10.1.1.20 SC32: 10.1.1.30

We provided the TCP/IP profile definitions for the z/OS Load Balancing Advisor and Agents for the three LPARs, as shown in the following examples. Example 6-14 shows the profile definitions in SC30.

Example 6-14 TCP/IP profile definitions for z/OS Load Balancing Advisor and Agent in SC30

```
;
;DYNAMIC VIPA DEFINITIONS
;
```

```
VIPADYNAMIC
 VIPADEFINE MOVEABLE IMMEDIATE 255.255.255.0 10.1.8.10
 VIPADISTribute DISTMethod SERVERWLM 10.1.8.10
   PORT 23
   DESTIP 10.1.7.31 10.1.7.21 10.1.7.11
 VIPAROUTE 10.1.7.31    10.1.1.30
 VIPAROUTE 10.1.7.21    10.1.1.20
 VIPAROUTE 10.1.7.11    10.1.1.10
 VIPARANGE 255.255.255.0 10.1.9.0
ENDVIPADYNAMIC

AUTOLOG 5
   LBADV                    ; SASP Load Balancing Advisor
   LBAGENT                  ; SASP Load Balancing Agent

PORT
  3860 TCP LBADV            ; SASP Workload Advisor (LB connections)
  8100 TCP LBADV            ; SASP Workload Advisor (Agent connections)
  8000 TCP LBAGENT NOAUTOLOG 1 ; SASP Workload Agent (Advisor connection)
```

Example 6-15 shows the profile definitions for SC31.

Example 6-15 TCP/IP profile definitions for z/OS Load Balancing Advisor and Agent in SC31

```
;
;DYNAMIC VIPA DEFINITIONS
;
VIPADYNAMIC
VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.1.8.20
 VIPADISTRIBUTE DISTMETHOD SERVERWLM 10.1.8.20
   PORT 23
   DESTIP 10.1.7.31 10.1.7.21 10.1.7.11
 VIPAROUTE 10.1.7.31    10.1.1.30
 VIPAROUTE 10.1.7.21    10.1.1.20
 VIPAROUTE 10.1.7.11    10.1.1.10
 VIPARANGE 255.255.255.0 10.1.9.0
ENDVIPADYNAMIC

AUTOLOG 5
   LBAGENT

PORT
  3860 TCP LBADV            ; SASP Workload Advisor (LB connections)
  8100 TCP LBADV            ; SASP Workload Advisor (Agent connections)
  8000 TCP LBAGENT NOAUTOLOG 1 ; SASP Workload Agent (Advisor connection)
```

Example 6-16 shows the profile definitions for SC32.

Example 6-16 TCP/IP profile definitions for z/OS Load Balancing Advisor and Agent in SC32

```
VIPADYNAMIC
VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.1.8.30
   VIPADISTribute DISTMethod SERVERWLM 10.1.8.30
   PORT 80 23
   DESTIP 10.1.7.31 10.1.7.21 10.1.7.11
VIPARANGE 255.255.255.0 10.1.9.0
```

```
ENDVIPADYNAMIC
AUTOLOG 5
   LBAGENT                  ; SASP Load Balancing Agent

PORT
  3860 TCP LBADV            ; SASP Workload Advisor (LB connections)
  8100 TCP LBADV            ; SASP Workload Advisor (Agent connections)
  8000 TCP LBAGENT NOAUTOLOG 1 ; SASP Workload Agent (Advisor connection)
```

Note that 1 NOAUTOLOG must be coded on the port definition if we use AUTOLOG to start the LBAGENT.

We configured the z/OS Load Balancing Advisor as shown in Example 6-17. (The z/OS Load Balancing Advisor configuration file is specified on the CONFIG DD statement in the advisor start procedure.)

Example 6-17 Configuration file for z/OS Load Balancing Advisor

```
debug_level               7

update_interval           60

agent_connection_port     8100 1

agent_id_list
{
  10.1.1.10..8000 2
  10.1.1.20..8000
  10.1.1.30..8000
# ::1..8000
}

lb_connection_v4          10.1.9.30..3860 3

lb_id_list
lb_id_list
{
  10.1.60.2 4
  10.1.60.3 5
# ::1
}
wlm serverwlm 6

port_list
{
  23
  {
    wlm serverwlm
  }
  8000
  {
    wlm serverwlm
  }
 8100
 {
   wlm serverwlm
```

```
 }
}
```

This configuration includes the following key elements:

1. Port 8100 is used to receive connections from LBAGENTs.

2. This is the list of valid LBAGENTs that can connect to this z/OS Load Balancing Advisor.

3. z/OS Load Balancing Advisor binds to this IP address and listens on specified port.

4. This is the IP address of the active load balancer.

5. This is the IP address of the standby load balancer.

6. Serverwlm weights will be the default.

We configured the z/OS Load Balancing Agents in each LPAR. Example 6-18 shows the configuration file for SC30. (The z/OS Load Balancing Agent configuration file is specified on the CONFIG DD statement in the start procedure.)

Example 6-18 Configuration file for z/OS Load Balancing Agent in SC30

```
debug_level                7

advisor_id                 10.1.9.30..8100 1

host_connection            10.1.1.10..8000 2
```

Example 6-19 shows the configuration file for SC31.

Example 6-19 Configuration file for z/OS Load Balancing Agent in SC31

```
debug_level                7

advisor_id                 10.1.9.30..8100 1

host_connection            10.1.1.20..8000 2
```

Example 6-20 shows the configuration file for SC32.

Example 6-20 Configuration file for z/OS Load Balancing Agent in SC32

```
debug_level                7

advisor_id                 10.1.9.30..8100 1

host_connection            10.1.1.30..8000 2
```

Note that **1** LBAGENT connects to the z/OS Load Balancing Advisor by this IP address and port, and **2** LBAGENT uses this as the source IP address and port.

We configured the active CSM as shown in Example 6-21.

Example 6-21 Configuration for active CSM

```
module ContentSwitchingModule 2 1
 variable ROUTE_UNKNOWN_FLOW_PKTS 1
 variable SASP_CSM_UNIQUE_ID CSM-6509A 2
!
```

```
 ft group 61 vlan 61 3
  priority 101
!
 vlan 60 server 4
ip address 10.1.60.2 255.255.255.0
  route 0.0.0.0 0.0.0.0 gateway 10.1.60.240
  alias 10.1.60.1 255.255.255.255
!
 probe PING-30S icmp 5
  interval 30
  retries 2
  failed 30
!
 real SC30-TN3270 6
  address 10.1.1.10
  inservice
 real SC31-TN3270
  address 10.1.1.20
  inservice
 real SC32-TN3270
  address 10.1.1.30
  inservice
!
 serverfarm TN3270 7
  nat server
  no nat client
  bindid 65521
  real name SC30-TN3270
   inservice
  real name SC31-TN3270
   inservice
  real name SC32-TN3270
   inservice
!
 vserver TN3270 8
  virtual 10.1.60.10 tcp telnet
  serverfarm TN3270
  replicate csrp connection
  persistent rebalance
  inservice
!
 dfp
 agent 10.1.9.30 3860 65521 9
!
interface Vlan10 10
 ip address 10.1.2.240 255.255.255.0
 ip policy route-map pbr-to-csm
!
interface Vlan60 11
 description VLAN 60 for CSM
 ip address 10.1.60.240 255.255.255.0
 ip ospf cost 5
!
interface Vlan61 12
 description CSM Failover
```

```
 no ip address
!
router ospf 100 13
 router-id 10.1.3.240
 log-adjacency-changes
 area 2 stub no-summary
 network 10.1.2.0 0.0.0.255 area 2
 network 10.1.3.0 0.0.0.255 area 2
 network 10.1.100.0 0.0.0.255 area 2
 network 10.1.0.0 0.0.255.255 area 0
 network 10.200.1.0 0.0.0.255 area 0
 default-information originate always metric-type 1
!
ip access-list extended tn3270 14
 permit tcp host 10.1.1.30 eq telnet any
 permit tcp host 10.1.1.20 eq telnet any
 permit tcp host 10.1.1.10 eq telnet any
!
route-map pbr-to-csm permit 10 15
 match ip address tn3270
 set ip next-hop 10.1.60.1
!
```

The following elements are important:

1. CSM is in module 2 of this switch.
2. Unique ID for active CSM required.
3. Fault tolerance uses dedicated VLAN 61.
4. CSM server VLAN. Subnet 10.1.60.0/24. This is the subnet for CSM and application clusters. There is no routing within the CSM, only static routes pointing to the routers.
5. This is the ICMP probe used for polling a given application instance for availability.
6. This is the real server definition pointing to the static VIPA per LPAR.
7. This is the server farm definition grouping real servers together and probe indicating polling of each real server. If a real server refuses to answer, the real server is marked unavailable.
8. This is the virtual server representing the application cluster. *Replicate csrp connection* implements stateful failover between the CSMs without connection loss of existing connections.
9. This is the DFP agent pointing to z/OS Load Balancing Advisor.
10. VLAN 10. Subnet 10.1.2.0/24. This is the OSA-Express Gigabit Ethernet attachment of z/OS. Policy-based routing is enabled in this VLAN.
11. VLAN 60. Subnet 10.1.60.0/24.This is the subnet for CSM and application clusters.
12. VLAN 61.This is the dedicated VLAN for CSM failover services.
13. OSPF process 100 includes subnets matching 10.1.0.0/16.
14. This is the extended access list used by policy-based routing.
15. This is the Route Map used by policy-based routing. It ensures that datagrams from extended access list TN3270 are being forwarded back to the CSM.

We configured the standby CSM as shown in Example 6-22.

Example 6-22 Configuration for standby CSM

```
module ContentSwitchingModule 3 1
 variable ROUTE_UNKNOWN_FLOW_PKTS 1
 variable SASP_CSM_UNIQUE_ID CSM-6509B 2
!
 ft group 61 vlan 61 3
  priority 100
!
 vlan 60 server 4
  ip address 10.1.60.3 255.255.255.0
  route 0.0.0.0 0.0.0.0 gateway 10.30.60.4
  route 0.0.0.0 0.0.0.0 gateway 10.30.60.5
  alias 10.1.60.1 255.255.255.255
!
 probe PING-30S icmp 5
  interval 30
  retries 2
  failed 30
!
 real SC30-TN3270 6
  address 10.1.1.10
  inservice
 real SC31-TN3270
  address 10.1.1.20
  inservice
 real SC32-TN3270
  address 10.1.1.30
  inservice
 real TN3270-SC30
  inservice
!
 serverfarm TN3270 7
  nat server
  no nat client
  real name SC30-TN3270
   inservice
  real name SC31-TN3270
   inservice
  real name SC32-TN3270
   inservice
  probe PING-30S
!
 vserver TN3270 8
  virtual 10.1.60.10 tcp telnet
  serverfarm TN3270
  replicate csrp connection
  persistent rebalance
  inservice
!
dfp 9
 agent 10.1.9.31 3860 65520
!
interface Vlan11 10
```

```
 ip address 10.1.3.11 255.255.255.0
 ip policy route-map pbr-to-csm
!
interface Vlan60 11
 description VLAN 60 for CSM
 ip address 10.1.60.241 255.255.255.0
 ip ospf cost 5
!
interface Vlan61 12
 description CSM Failover
 no ip address
!
router ospf 100 13
 router-id 10.1.2.220
 log-adjacency-changes
 area 1 stub no-summary
 network 10.30.0.0 0.0.255.255 area 1
!
ip access-list extended tn3270 14
 permit tcp host 10.1.1.30 eq telnet any
 permit tcp host 10.1.1.20 eq telnet any
 permit tcp host 10.1.1.10 eq telnet any!
!
route-map pbr-to-csm permit 10 15
 match ip address tn3270
 set ip next-hop 10.1.60.1
!
```

The following elements are slightly different than those used with the active CSM:

1. CSM is in module 3 of this switch.
2. Unique ID for active CSM required.
3. Fault tolerance uses dedicated VLAN 61.
4. CSM server VLAN. Subnet 10.1.60.0/24. This is the subnet for CSM and application clusters. There is no routing within the CSM, only static routes pointing to the routers.
5. This is the ICMP probe used for polling a given application instance for availability.
6. This is the real server definition pointing to the static VIPA per LPAR.
7. This is the server farm definition grouping real servers together and probe indicating polling of each real server. If a real server refuses to answer the real server is marked unavailable.
8. This is the virtual server representing the application cluster. *Replicate csrp connection* implements stateful failover between the CSMs without connection loss of existing connections.
9. This is the DFP agent pointing to z/OS Load Balancing Advisor.
10. VLAN 11. Subnet 10.1.3.0/24. This is the OSA-Express Gigabit Ethernet attachment of z/OS. Policy-based routing is enabled in this VLAN.
11. VLAN 60. Subnet 10.1.60.0/24. This is the subnet for CSM and application clusters.
12. VLAN 61.This is the dedicated VLAN for CSM failover services.
13. OSPF process 100 includes subnets matching 10.30.0.0/16.

14. This is the extended access list used by policy-based routing.

15. This is the Route Map used by policy-based routing. It ensures that datagrams from extended access list TN3270 are being forwarded back to the CSM.

Verification

We verified correct operation using the following steps:

1. We started the Load Balancing Advisor.
2. We started the Load Balancing Agent in all three LPARs.
3. We checked the connections between the agents and the switch units.
4. We checked the status of the DFP agents in CSM.
5. We verified that the Data Facility Product (DFP) agents in CSM had registered with an z/OS Load Balancing Advisor.
6. We verified that application members registered with the agents.
7. We connected 12 TN3270 clients.
8. We observed what happens if one TCP/IP stack is lost.
9. We observed what happens if an agent is lost.
10. We observed what happens if the z/OS Load Balancing Advisor is lost.

We started the z/OS Load Balancing Advisor in SC30. Example 6-23 shows the LB Advisor is ready.

Example 6-23 LB Advisor is listening ready and listening

```
D TCPIP,TCPIPA,N,CONN,PORT=3860
USER ID  CONN     STATE
LBADV    00000022 LISTEN
  LOCAL SOCKET:   10.1.9.30..3860
  FOREIGN SOCKET: 0.0.0.0..0
LBADV    00000036 ESTBLSH
  LOCAL SOCKET:   10.1.9.30..3860
  FOREIGN SOCKET: 10.1.60.2..4364
2 OF 2 RECORDS DISPLAYED
```

Example 6-24 shows the joblog.

Example 6-24 z/OS Load Balancing Advisor joblog

```
LBADV    STARTED
LBADV STARTING
LBADV INITIALIZATION COMPLETE
LBADV AGENT CONNECTED FROM ::FFFF:10.1.1.10
LBADV LOAD BALANCER CONNECTED FROM 10.1.60.2
LBADV AGENT CONNECTED FROM ::FFFF:10.1.1.30
LBADV AGENT CONNECTED FROM ::FFFF:10.1.1.20
```

The agents started in all three LPARs, as shown in Example 6-25.

Example 6-25 LB Agent joblog in SC30, SC31, and SC32

```
$HASP373 LBAGENT  STARTED
EZD1231I LBAGENT STARTING
EZD1232I LBAGENT INITIALIZATION COMPLETE
EZD1259I LBAGENT CONNECTED TO ADVISOR 10.1.9.30
```

We verified that the agents were connected to the advisor. Example 6-26 shows the connection in SC30.

Example 6-26 z/OS Load Balancing Advisor connection in SC30

```
D TCPIP,TCPIPA,N,CONN,PORT=8000
USER ID  CONN     STATE
LBADV    000001CD ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.9.30..8100
  FOREIGN SOCKET: ::FFFF:10.1.1.20..8000
LBADV    00000033 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.9.30..8100
  FOREIGN SOCKET: ::FFFF:10.1.1.10..8000
LBADV    0000004B ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.9.30..8100
  FOREIGN SOCKET: ::FFFF:10.1.1.30..8000
LBAGENT  00000032 ESTBLSH
  LOCAL SOCKET:   10.1.1.10..8000
  FOREIGN SOCKET: 10.1.9.30..8100
4 OF 4 RECORDS DISPLAYED
```

Example 6-27 shows the connection in SC31.

Example 6-27 z/OS Load Balancing Advisor connection in SC31

```
D TCPIP,TCPIPB,N,CONN,PORT=8000
USER ID  CONN     STATE
LBAGENT  00000052 ESTBLSH
  LOCAL SOCKET:   10.1.1.20..8000
  FOREIGN SOCKET: 10.1.9.30..8100
1 OF 1 RECORDS DISPLAYED
```

Example 6-28 shows the connection in SC32.

Example 6-28 z/OS Load Balancing Advisor connection in SC32

```
D TCPIP,TCPIPC,N,CONN,PORT=8000
USER ID  CONN     STATE
LBAGENT  00000054 ESTBLSH
  LOCAL SOCKET:   10.1.1.30..8000
  FOREIGN SOCKET: 10.1.9.30..8100
1 OF 1 RECORDS DISPLAYED
```

We checked the connections from the Load Balancing Advisor connection on the LPAR running the advisor, as shown in Example 6-29.

Example 6-29 z/OS Load Balancing Advisor connections in SC30

```
LBADV    000001CD ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.9.30..8100
  FOREIGN SOCKET: ::FFFF:10.1.1.20..8000 1
LBADV    00000033 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.9.30..8100
  FOREIGN SOCKET: ::FFFF:10.1.1.10..8000 2
LBADV    0000004B ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.1.9.30..8100
  FOREIGN SOCKET: ::FFFF:10.1.1.30..8000 3
LBADV    00000022 LISTEN
  LOCAL SOCKET:   10.1.9.30..3860 4
  FOREIGN SOCKET: 0.0.0.0..0
LBADV    00000036 ESTBLSH
  LOCAL SOCKET:   10.1.9.30..3860
  FOREIGN SOCKET: 10.1.60.2..4364 5
LBAGENT  00000032 ESTBLSH
  LOCAL SOCKET:   10.1.1.10..8000  6
  FOREIGN SOCKET: 10.1.9.30..8100
LBADV    00000024 LISTEN
  LOCAL SOCKET:   ::..8100  7
  FOREIGN SOCKET: ::..0
```

This display includes the following relevant points:

1. The z/OS Load Balancing Agent from SC31.
2. The z/OS Load Balancing Agent from SC30.
3. The z/OS Load Balancing Agent from SC32.
4. The z/OS Load Balancing Advisor listener socket - SASP clients.
5. The z/OS Load Balancing Advisor connection from active CSM.
6. The z/OS Load Balancing Advisor listener socket - z/OS Load Balancing Agents.
7. The z/OS Load Balancing Agent connection with z/OS Load Balancing Advisor.

We checked the status of the DFP agents in the active CSM, as shown in Example 6-30.

Example 6-30 DFP Agent display in active CSM

```
Router# sh mod csm 2 dfp detail
Router1>sh mod csm 2 dfp detail
DFP Agent 10.1.9.30:3860  Connection state: Connected 1
   Keepalive = 65521  Retry Count = 0      Interval = 180   (Default)
   Security errors = 0
   Last message received: 14:59:44 est 09/21/07
   Last reported Real weights for Protocol TCP, Port telnet
      Host 10.1.1.10       Bind ID 65521  Weight 1
      Host 10.1.1.20       Bind ID 65521  Weight 1
      Host 10.1.1.30       Bind ID 65521  Weight 1

DFP manager listen port not configured.
No weights to report to managers.
Router#
```

Note that 1 the DFP agent in active CSM is connected with the z/OS Load Balancing Advisor.

We also checked the status of the standby CSM agent, as shown in Example 6-31.

Example 6-31 DFP Agent display in standby CSM

```
Router#sh mod csm 3 dfp detail
DFP Agent 10.1.9.31:3860  Connection state: Connected 1
   Keepalive = 65520  Retry Count = 0      Interval = 180   (Default)
   Security errors = 0
   Last message received: 14:59:55 est 09/21/07
   Last reported Real weights for Protocol TCP, Port telnet
      Host 10.1.1.10        Bind ID 65520  Weight 2
      Host 10.1.1.20        Bind ID 65520  Weight 1
      Host 10.1.1.30        Bind ID 65520  Weight 1

DFP manager listen port not configured.
No weights to report to managers.
Router#
```

Note that 1 the DFP agent in the standby CSM is connected with the z/OS Load Balancing Advisor.

We verified that the DFP agents in CSM had registered with the z/OS Load Balancing Advisor in SC30, as shown in Example 6-32.

Example 6-32 Display a load balancer summary

```
F LBADV,DISP,LB
EZD1242I LOAD BALANCER SUMMARY 994
LB INDEX      : 00        UUID      : 436973636F2D43534D2D3635303941
 IPADDR..PORT : 10.1.60.2..4364
 HEALTH       : 7E        FLAGS     : NOCHANGE PUSH
1 OF 1 RECORDS DISPLAYED
```

We displayed a detailed list of the data registered with the active CSM under index 00, as shown in Example 6-33. (The LB index numbers are shown in Example 6-32.)

Example 6-33 Display load balancer details about active CSM

```
F LBADV,DISP,LB,I=00
EZD1243I LOAD BALANCER DETAILS 999
LB INDEX      : 00        UUID      : 436973636F2D43534D2D3635303941
 IPADDR..PORT : 10.1.60.2..4364
 HEALTH       : 7E        FLAGS     : NOCHANGE PUSH
 GROUP NAME   : TN3270
  GROUP FLAGS : SERVERWLM
  IPADDR..PORT: 10.1.1.30..23
   SYSTEM NAME: SC32      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00064     CS WEIGHT : 100  NET WEIGHT: 00001        1
   RAW           CP: 64  ZAAP: 00  ZIIP: 00
   PROPORTIONAL  CP: 64  ZAAP: 00  ZIIP: 00
   ABNORM     : 00000     HEALTH    : 100
   FLAGS      :
  IPADDR..PORT: 10.1.1.20..23
   SYSTEM NAME: SC31      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00064     CS WEIGHT : 100  NET WEIGHT: 00001        1
```

```
   RAW           CP: 64  ZAAP: 00  ZIIP: 00
   PROPORTIONAL  CP: 64  ZAAP: 00  ZIIP: 00
   ABNORM     : 00000     HEALTH    : 100
   FLAGS      :
  IPADDR..PORT: 10.1.1.10..23
   SYSTEM NAME: SC30      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00064     CS WEIGHT : 100  NET WEIGHT: 00001       1
   RAW           CP: 64  ZAAP: 00  ZIIP: 00
   PROPORTIONAL  CP: 64  ZAAP: 00  ZIIP: 00
   ABNORM     : 00000     HEALTH    : 100
    FLAGS      :
 3 OF 3 RECORDS DISPLAYED
```

Note the 1 WLM weight, Communications Server weight, and NET weight (the weight that is forwarded to the external load balancer).

The load balancer details displayed in Example 6-34 are also available from the CSM. Remember that the z/OS Load Balancing Advisor sends the NET WEIGHT to the CSM and to the standby CSM.

Example 6-34 Display dfp weights in active CSM

```
Router# sh mod csm 2 dfp weights
Real IP : 10.1.1.10 Protocol: TCP Port: telnet Bind_ID: 65521 Weight: 1 1
Set by Agent 10.1.9.30:3860 at 01:13:47 est 09/21/07

Real IP : 10.1.1.20 Protocol: TCP Port: telnet Bind_ID: 65521 Weight: 1 1
Set by Agent 10.1.9.30:3860 at 02:15:20 est 09/21/07

Real IP : 10.1.1.30 Protocol: TCP Port: telnet Bind_ID: 65521 Weight: 1 1
Set by Agent 10.1.9.30:3860 at 01:15:30 est 09/21/07
Router#
```

Note 1 the NET weight received from the z/OS Load Balancing Advisor.

We used the command shown in the following examples to verify the members registered in the agents in each LPAR. Example 6-35 shows the member details in SC30.

Example 6-35 LBAGENT member details in SC30

```
F LBAGENT,DISP,MEM,DET
EZD1245I MEMBER DETAILS 009
LB INDEX      : 00        UUID      : 436973636F2D43534D2D3635303941
 GROUP NAME   : TN3270
  IPADDR..PORT: 10.1.1.10..23
   TCPNAME    : TCPIPA    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TN3270A   ASID      : 00A0 RESOURCE  : 00000028
1 OF 1 RECORDS DISPLAYED
```

Example 6-36 shows the member details in SC31.

Example 6-36 LBAGENT member details in SC31

```
F LBAGENT,DISP,MEM,DET
EZD1245I MEMBER DETAILS 646
LB INDEX      : 00       UUID      : 436973636F2D43534D2D3635303941
 GROUP NAME   : TN3270
  IPADDR..PORT: 10.1.1.20..23
   TCPNAME    : TCPIPB    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TN3270B   ASID      : 006A RESOURCE  : 00000033
1 OF 1 RECORDS DISPLAYED
```

Example 6-37 shows the member details for SC32.

Example 6-37 LBAGENT member details in SC32

```
F LBAGENT,DISP,MEM,DET
EZD1245I MEMBER DETAILS 881
LB INDEX      : 00       UUID      : 436973636F2D43534D2D3635303941
 GROUP NAME   : TN3270
  IPADDR..PORT: 10.1.1.30..23
   TCPNAME    : TCPIPC    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TN3270C   ASID      : 00AA RESOURCE  : 00000035
1 OF 1 RECORDS DISPLAYED
```

We connected up to 12 TN3270 clients using address 10.1.60.10 and verified the distribution, as shown in Example 6-38. As shown in the example, the weights are changed based on updates from the z/OS Load Balancing Advisor. (The weight for SC32-TN3270 changed from 2 to 3 because we made a similar display a few minutes earlier.) This display shows how the 12 connections are distributed across three systems.

Example 6-38 Display of the server farm in the CSM

```
Router# sh mod csm 2 serverfarms name tn3270 detail
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65521, fail action = none
  inband health config: <none>
  retcode map = <none>
  Real servers:
    SC30-TN3270, weight = 1, OPERATIONAL, conns = 2
    SC31-TN3270, weight = 1, OPERATIONAL, conns = 1
    SC32-TN3270, weight = 1, OPERATIONAL, conns = 0
  Total connections = 3
*
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65521, fail action = none
  inband health config: <none>
  retcode map = <none>
  Real servers:
    SC30-TN3270, weight = 8, OPERATIONAL, conns = 5
    SC31-TN3270, weight = 8, OPERATIONAL, conns = 3
    SC32-TN3270, weight = 3, OPERATIONAL, conns = 0
```

```
  Total connections = 8
Router#
*
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65521, fail action = none
  inband health config: <none>
  retcode map = <none>
  Real servers:
    SC30-TN3270, weight = 8, OPERATIONAL, conns = 5
    SC31-TN3270, weight = 8, OPERATIONAL, conns = 4
    SC32-TN3270, weight = 3, OPERATIONAL, conns = 1
  Total connections = 10
*
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65521, fail action = none
  inband health config: <none>
  retcode map = <none>
  Real servers:
    SC30-TN3270, weight = 8, OPERATIONAL, conns = 6 1
    SC31-TN3270, weight = 8, OPERATIONAL, conns = 5 1
    SC32-TN3270, weight = 3, OPERATIONAL, conns = 1 1
  Total connections = 12
```

Note that **1** the 12 TN3270 connections are now distributed according to the weights send by the agents.

We then stopped the TCP/IP stack in SC32, making the TN3270 server in SC32 unavailable. This broke TN3270 sessions. When we reconnected the sessions, they were distributed according to the weights in the remaining TN3270 servers, as shown in Example 6-39. This display shows how the 12 connections are distributed across two systems.

Note that the weighting recommendations can change rapidly during a recovery situation such as this and slightly different reconnection timings can produce slightly different connection distributions.

Example 6-39 Display of the server farm in the CSM

```
Router# sh mod csm 2 serverfarms name tn3270 detail
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65521, fail action = none
  inband health config: <none>
  retcode map = <none>
  Real servers:
    SC30-TN3270, weight = 1, OPERATIONAL, conns = 7
    SC31-TN3270, weight = 1, OPERATIONAL, conns = 5
    SC32-TN3270, weight = 0, DFP_THROTTLED, conns = 0 1
  Total connections = 12
Router#
```

Note that **1** real server SC32-TN3270 has the status DFP_THROTTLED.

After stopping the TCP/IP stack (and the TN3270 server) on SC32, the advisor assigns a weight of zero, as shown in Example 6-40. The zero weight means “Do not forward requests to this server instance.”

Example 6-40 Display load balancer details about active CSM

```
F LBADV,DISP,LB,I=00
EZD1243I LOAD BALANCER DETAILS 303
LB INDEX      : 00       UUID      : 436973636F2D43534D2D3635303941
 IPADDR..PORT : 10.1.60.2..4364
 HEALTH       : 7E       FLAGS     : NOCHANGE PUSH
 GROUP NAME   : TN3270
  GROUP FLAGS : SERVERWLM
  IPADDR..PORT: 10.1.1.30..23
   SYSTEM NAME: N/A       PROTOCOL  : TCP  AVAIL     : NO
   WLM WEIGHT : 00000     CS WEIGHT : 000  NET WEIGHT: 00000 1
   RAW           CP: 64  ZAAP: 00  ZIIP: 00
   PROPORTIONAL  CP: 64  ZAAP: 00  ZIIP: 00
   ABNORM     : 00000     HEALTH    : 100
   FLAGS      : NOTARGETSYS 2
  IPADDR..PORT: 10.1.1.20..23
   SYSTEM NAME: SC31      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00064     CS WEIGHT : 100  NET WEIGHT: 00001
   RAW           CP: 64  ZAAP: 00  ZIIP: 00
   PROPORTIONAL  CP: 64  ZAAP: 00  ZIIP: 00
   ABNORM     : 00000     HEALTH    : 100
   FLAGS      :
  IPADDR..PORT: 10.1.1.10..23
   SYSTEM NAME: SC30      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00064     CS WEIGHT : 100  NET WEIGHT: 00001
   RAW           CP: 64  ZAAP: 00  ZIIP: 00
   PROPORTIONAL  CP: 64  ZAAP: 00  ZIIP: 00
   ABNORM     : 00000     HEALTH    : 100
   FLAGS      :
3 OF 3 RECORDS DISPLAYED
```

Note that 1 weights changed to zero, indicating that the application instance is not ready for connections. Also note that 2 LBQ says that the load balancer has quiesced the member. NOTARGETSYS says that the z/OS Load Balancing Advisor will advise the load balancer that the application instance should not receive new workload.

If an agent is down, existing sessions to the LPAR continue, but new sessions are not sent to that LPAR. In Example 6-41, we stopped the z/OS Load Balancing Agent in SC32.

Example 6-41 Display of the server farm in the CSM shows how the 12 connections are distributed

```
Router# sh mod csm 2 serverfarms name tn3270 detail
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65521,
  inband health config: <none>
  retcode map = <none>
  Real servers:
    SC30-TN3270, weight = 1, OPERATIONAL, conns = 7
    SC31-TN3270, weight = 1, OPERATIONAL, conns = 5
```

```
    SC32-TN3270, weight = 0, DFP_THROTTLED, conns = 0 1
  Total connections = 12
Router#
```

In this command, the number corresponds to the following information:

1. Real server SC32-TN3270 has the status DFP_THROTTLED and weight zero, which means that new connections will not be sent.

We experimented by taking down the z/OS Load Balancing Advisor. The display in Example 6-42 illustrates this state. When the retry limit (10) is reached, the CSM returns to its internal load-balancing algorithm (round-robin) until the z/OS Load Balancing Advisor is back. Existing connections continue without disruption.

Example 6-42 Connection from DFP fails

```
Router#sh mod csm 2 dfp detail
DFP Agent 10.1.9.30:3860  Connection state: Failed  Retries: 2 1
   Keepalive = 65521  Retry Count = 0      Interval = 180   (Default)
   Security errors = 0
   Last message received: 18:37:20 est 09/21/07
   Last reported Real weights for Protocol TCP, Port telnet
      Host 10.1.1.10       Bind ID 65521  Weight 1
      Host 10.1.1.20       Bind ID 65521  Weight 1
      Host 10.1.1.30       Bind ID 65521  Weight 1

DFP manager listen port not configured.
No weights to report to managers.
Router#
```

Note that the 1 DFP connection to the z/OS Load Balancing Advisor has failed. It will be retried 10 times.

After all the retries fail, the CSM uses its default internal balancing, which is round-robin, as shown in Example 6-43. The equal weights of 8 indicate that the CSM is doing its round-robin balancing.

Example 6-43 Display of the server farm in the CSM shows how the 12 connections are distributed

```
Router#sh mod csm 2 serverfarms name tn3270 detail
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65521, fail action = none
  inband health config: <none>
  retcode map = <none>
  Real servers:
    SC30-TN3270, weight = 1, OPERATIONAL, conns = 6 1
    SC31-TN3270, weight = 1, OPERATIONAL, conns = 5 1
    SC32-TN3270, weight = 1, OPERATIONAL, conns = 0 1
  Total connections = 11

Router#
```

In this command, the number corresponds to the following information:

1. The external load balancer returns to internal decision-making (round-robin).

Note: Server (application)-specific WLM weights are supported in z/OS V1R7 (and later). The CSM registers application-specific members, provided that each vserver utilizes separate server farms; otherwise, the CSM only registers system members.

In the last example, the z/OS Load Balancing Advisor used SERVERWLM. Using SERVERWLM usually results in better workload distribution.

Be aware that TN3270 is part of the TCP/IP stack in our example. By using SERVERWLM for TN3270, we actually receive WLM recommendations for the TCP/IP stack and not TN3270 itself. We recommend that you use SERVERWLM for servers running in their own address spaces, such as an HTTP server.

Problem determination

We approached problem determination for this example in the following order:

1. Check that application instances are running in z/OS (active listener).
2. Check that there is connectivity between the router and z/OS.
3. Check that the real server is operational.
4. Check that the virtual server is operational.
5. Check for connectivity between client and CSM (application cluster address=vserver address).
6. Check that policy-based routing definitions are in place.
7. Run packet trace.
8. Run CTRACE.
9. Use a network analyzer.
10. Run debug in the external networking devices.
11. Run debug in z/OS Load Balancing Advisor, z/OS Load Balancing Agents, or both.

Both the z/OS Load Balancing Advisor and the z/OS Load Balancing Agents can write logs to the syslogd daemon facility. To use this, syslogd must be started before starting the z/OS Load Balancing Advisor and the z/OS Load Balancing Agent. The debug level (7) should normally be used unless problem documentation needs to be gathered. Increased levels of debug can result in excessive amounts of information.

Logging levels for the advisor and agent are:

0 None. No debug messages are logged.

1 Error-level messages are logged.

2 Warning-level messages are logged.

4 Event-level messages are logged.

8 Info-level messages are logged.

16 Message-level messages are logged. These are details of the messages (packets) sent between the advisor and LB, and the advisor and agent.

32 Collection-level messages are logged. These are details of the collection and manipulation of data supporting the calculated weights. This level only has meaning to the Agent. The Advisor does not log any data at this level.

64 Debug-level messages are logged. These are internal debug messages intended for development and service.

128 Trace-level messages are logged. These are function entry and exit traces that show the path through the code.

To log a combination of debug levels, add the debug level numbers. The default debug level is 7, which captures all error, warning, and event messages.

7

Intra-sysplex workload balancing

Previous chapters described internal and external application workload balancing methods. These techniques are based on accessing an application server through one cluster IP address by the client.

In a multitier application environment, however, the path between the different application servers within the sysplex might become long and can result in a performance impact, for example, when an HTTP server, a WebSphere Application Server, or an IBM DataPower appliance and an Enterprise Information System (EIS), such as IMS™ or CICS®, are involved in a single client connection request. In such situations, optimized paths within the sysplex can provide better performance.

This chapter includes several examples that illustrate how multitier application support can improve existing workload balancing solutions.

This chapter discusses the following topics.

Section	Topic
7.1, "Optimizing Sysplex Distributor intra-sysplex load balancing" on page 214"	Basic goals needed to optimize intra-sysplex application server workload balancing in a multitier environment
7.2, "Optimized multitier z/OS Sysplex Distributor load balancing" on page 217	Implementation of several optimized multitier load balancing solutions with Sysplex Distributor and external load balancer
7.3, "WLM reporting abnormal conditions" on page 253	Basics of WLM weight calculation and the use of abnormal conditions detected by WLM to reduce weight recommendations

7.1 Optimizing Sysplex Distributor intra-sysplex load balancing

In a multitier application environment, situations exist where connection requests can cross logical partitions (LPARs) before reaching the appropriate application instance. The selection of a server that is on a different LPAR than the client is initiated through workload balancing decision units such as Sysplex Distributor.

Figure 7-1 illustrates a typical configuration where an LPAR crossing occurs.

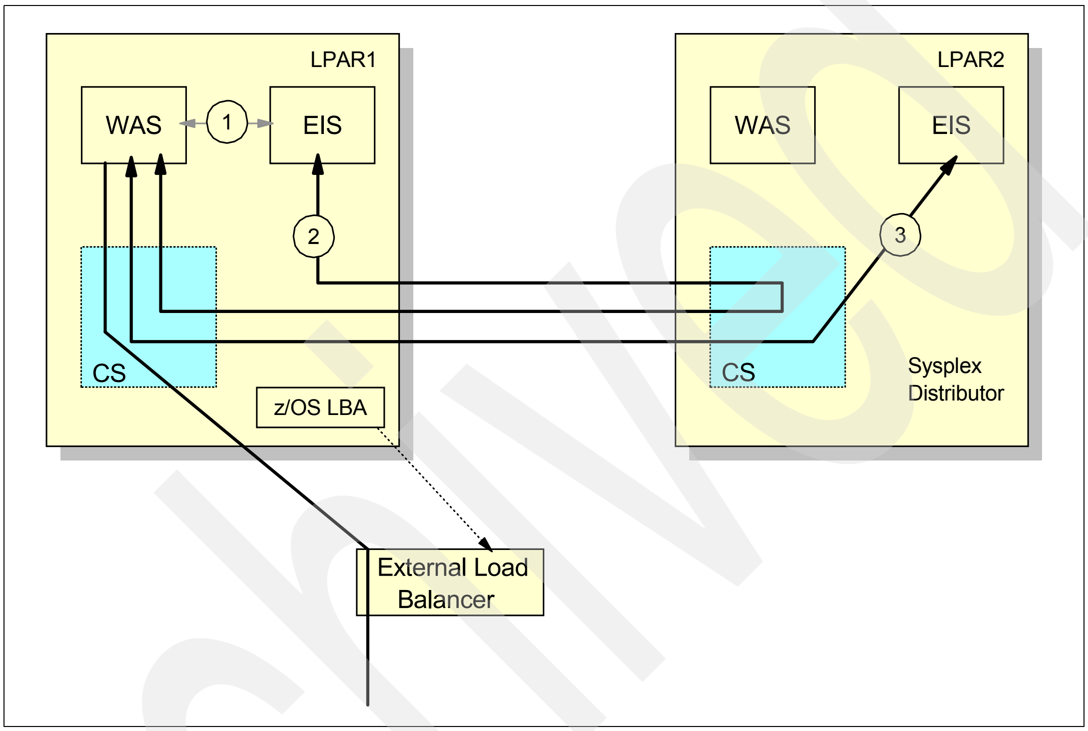


Figure 7-1 Local and remote connections

This configuration shows two load balancing decision points:

- An external load balancer, which uses Workload Manager (WLM) recommendations from the z/OS Load Balancing Advisor (LBA)
- A Sysplex Distributor

This example is also valid if you use Sysplex Distributor only. A backup Sysplex Distributor is not shown in this figure.

Note: This example is not unique to a WebSphere Application Server environment. Any z/OS sysplex multitier application environment might exhibit similar behavior and have similar issues.

7.1.1 Current connections from WebSphere Application Server to EIS

There are two types of connectors for WebSphere Application Server-to-EIS communication, shown in Figure 7-1 as a *local connection* (1) and as *remote connections* (2) and (3).

Local connection (1)

Connection requests coming from a client through the external load balancer arrive at the TCP/IP stack and are then forwarded directly to the WebSphere Application Server. From there, the client requests are forwarded to Enterprise Information Systems (EIS) through a connector. An EIS can be a system such as CICS, IMS, or DB2.

Local connections provide an optimized, high-speed path, see (1) in Figure 7-1, based on local services such as cross-memory services.

Problems will occur if the local target (for example, the EIS) is not available. Because local connections are used, there is no way to switch to an alternate target in a different LPAR. Also, because the WebSphere Application Server transactions are failing, they will appear to complete quickly with little CPU usage, thus causing the Workload Manager (WLM) to prefer that LPAR (in our example, LPAR1) for increased workload. This situation is known as a *storm drain issue*.

Remote connection (2) and (3)

WebSphere Application Server uses remote connections for a connection to the EIS. These are TCP connections. A load balancer decides which address to use as the connection endpoint for the EIS (in our example, in LPAR1 or LPAR2), and this decision is based on WLM recommendations. In our case, it is the Sysplex Distributor that selects a target among available targets defined in the sysplex.

If the target is local and the Sysplex Distributor is remote, as shown in Figure 7-1 in LPAR2 using the (2) path, or if the target is remote (in LPAR2 using the (3) path), the communication path is not as efficient as when it remains in one LPAR.

The solution to this problem needs to include the following aspects:

- Automatically locating a target server in the same LPAR as the client, or at least in the same CED or CEC
- Favoring that local target as long as it is available and has sufficient capacity for new work based on WLM recommendations
- No longer favoring the local target if these conditions are not met

The overall goal of the improved multitier application support should be an optimized path to the application servers with better performance and without losing high availability.

7.1.2 Optimized multitier application workload balancing

Figure 7-2 illustrates a multitier application environment.

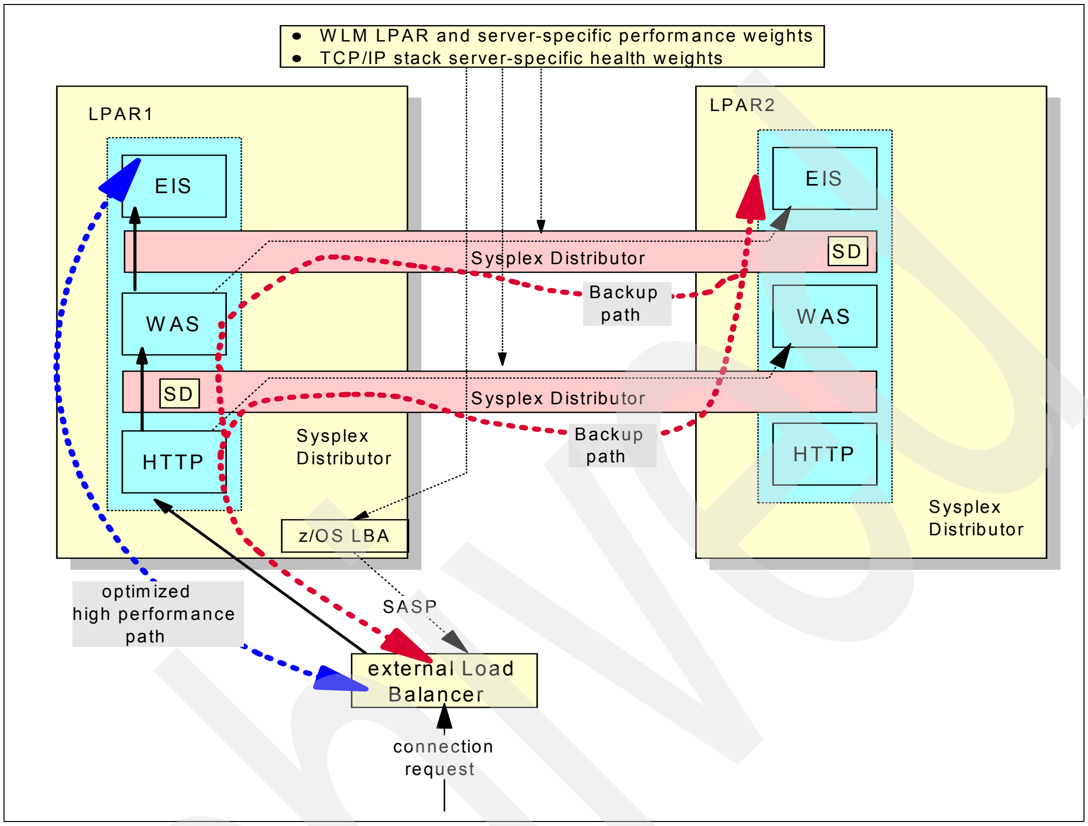


Figure 7-2 Multitier application configuration

In Figure 7-2, two LPARs have the same server applications, and they consist of the following elements:

- An HTTP proxy server that receives TCP requests from clients' browsers through an external load balancer

  The external load balancer has WLM information provided by the z/OS Load Balancing Advisor regarding which HTTP server in LPAR1 or in LPAR2 would best perform. This information is used by the load balancer to select the HTTP server in LPAR1 or LPAR2. In our example, the HTTP request was sent to the TCP/IP stack in LPAR1.

  If no external load balancer is used, then the Sysplex Distributor is the first address of the client request on LPAR1 that can be used to select the HTTP server instead.

- The WebSphere Application Server, which receives TCP requests from the HTTP server

  This TCP request is controlled by the Sysplex Distributor only. The Sysplex Distributor controls the selection of the HTTP server by defining distributed dynamic VIPAs. However, if the Sysplex Distributor decides to send the HTTP request to a WebSphere Application Server in another LPAR, then the communication overhead will rise.

  The best performance can be achieved by using a *fast direct local sockets* path to the WebSphere Application Server, which belongs to the same TCP/IP stack as the HTTP server. With this function, the selection of a local server and connection setup can stay within the local LPAR even if the distributor is remote. When both client and selected server are local, the data path will use fast direct local sockets.

By using application endpoint awareness with the enhanced sysplex sockets API, security functions such as authentication overhead and data conversions can be avoided because the connection's setup and datapath is local.

When configuring the Sysplex Distributor to use optimized load balancing, you can determine the level of preference for local targets, as explained here:

- Always choose a local target or a target running on the same CEC if that target is available and healthy.
- Compare WLM weights after applying a multiplier to the local target weight to determine if the local target should be preferred.

The user controls this function by defining the OPTLOCAL option and CPCSCOPE option on the VIPADISTRIBUTE statement.

7.2 Optimized multitier z/OS Sysplex Distributor load balancing

This section explains some concepts of the Sysplex Distributor functions used on a multitier environment and then describes the configuration that we used in our examples. It also explains the flow of client requests to the WebSphere Application Server and EIS, and examines our test cases, which consisted of different multitier solutions.

Multitier applications can sometimes choose between local and remote connection, depending on the location of the different parts of the application. Running within the same system, a local connection is possible, which gives the highest performance but which has an availability drawback. Using remote connection has a performance cost, but can provide better availability by running on different systems.

Sysplex Distributor provides support for distributing workload (TCP connections) to target systems other than z/OS. Doing so also means that Sysplex Distributor cannot rely on XCF communication with the target systems, and it cannot rely on WLM to provide server weights for the targets. Sysplex Distributor solves this issue by implementing support for an Sysplex Distributor-specific agent on the target systems that uses an Sysplex Distributor-specific protocol to communicate with Sysplex Distributor. The agent provides metrics back to Sysplex Distributor, which Sysplex Distributor uses to determine availability and capacity of the target servers. In addition, the agent also provides Sysplex Distributor with connection state information so that Sysplex Distributor can maintain its normal connection routing table information and can allow nondisruptive takeover of non-z/OS targets by a backup Sysplex Distributor. Incoming IP packets to connections that are distributed to a non-z/OS target are forwarded to the target using generic routing encapsulation (GRE). The initial target platform is IBM DataPower.

You can achieve optimized multitier z/OS load balancing in the following ways:

- Using sysplex external load balancing (see Chapter 6, “External application workload balancing” on page 171), plus sysplex internal load balancing such as Sysplex Distributor (see Chapter 5, “Internal application workload balancing” on page 95)
- Using sysplex internal load balancing only through Sysplex Distributor (see Chapter 5, “Internal application workload balancing” on page 95)

Sysplex Distributor improves the already existing optimized local processing by enabling the distributing stack to factor in metrics for the tier 1 server on each target TCP/IP stack, as well as the tier 2 server that is used as a target by that tier 1 server on the same TCP/IP stack. The Sysplex Distributor can combine the weights of a WebSphere Application Server (tier 1) and the corresponding EIS (tier 2).

In our test environment, we used sysplex internal load balancing only through Sysplex Distributor. This solution differs only in that the Sysplex Distributor (instead of the external load balancer) decides which HTTP server in LPAR1 (SC30) or LPAR2 (SC31) receives the TCP SYN request, and that decision is based on the best WLM value.

Note that in order to switch on optimized load balancing, you need to define additional parameters on the VIPADISTRIBUTE statement, as illustrated in Figure 7-3.

```
>>-VIPADynamic------------------------------------------------------------------>

               +-TIER 1-+------------+
|-VIPADEFINE--+                     +--------------------+--Options--+------------->
               +-TIER 2-+----------+-+
               |        +-CPCSCOPE-+ |
               +-CPCSCOPE-----------+

|--VIPADISTRIBUTE------+--Baseoptions-------+----------+--ipv4 addr--+-------------->
                       +--Tier1 parameters--+          +--ipV6 intf--+
                       +--Tier2 parameters--+

Options:
   +-DEFINE-+  +-DISTMethod BASEWLM--(BaseWLMoptions)-------+
|--+        +--+                                            +----------------------->
   +-DELETE-+  +-DISTMethod + ROUNDROBIN-------------------+
                            + SERVERWLM--(ServerWLMoptions)-+
                            + WEIGHTEDactive----------------+

ServerWLMoptions:
   +-PROCXCOST---- ZAAP 1 ZIIP 1-+      +-ILWEIGHTING 0-+    +-NOOPTLOCAL--------+
|--+                            +-------+              +-----+                 +-->
   |             +-ZAAP 1-+      |      +-ILWEIGHTING 1-+    |         +-1-----+ |
   +-PROCXCOST---+        +-+-----+                    2-+    +OPTLOCAL-+       +--+
               | +-ZAAP x-+ |                          3-+              +-value-+
               |            |
               | +-ZIIP 1-+ |
               +-+        +-+
                 +-ZIIP y-+

|--ENDVIPADynamic--|
```

Figure 7-3 VIPADISTRIBUTE statement

7.2.1 Tier 1 and tier 2 options

The VIPADEFINE statement has parameters to indicate that the DVIPA is used for tier 1 and tier 2 applications. The VIPADISTRIBUTE statement has the same tier parameter followed by the name of the group, which indicates that these tiers work as a group.

Tier 1 and tier 2 servers combined weights are calculated *only* when both distribution methods use WLM weights (that is, either BASEWLM or SERVERWLM). Use SERVERWLM when possible because this method is recommended based on a server's capacity for new work instead of the LPARs' capacity for new work.

Sysplex Distributor includes availability, health, and performance metrics for both the tier 1 z/OS server and the tier 2 z/OS server on a target LPAR when determining to which LPAR the tier 1 connection goes. These metrics increase the likelihood of such connections gaining the performance benefits of optimized local support, such as use of fast local sockets.

In Figure 7-4, a connection request comes into SD1 on SC30. SD1 consults with WLM and the TCP/IP stacks to determine availability, health, performance, and capacity of the target systems for both the HTTP tier server instances and the EIS tier server instances, which is CICS in our example, on each LPAR.

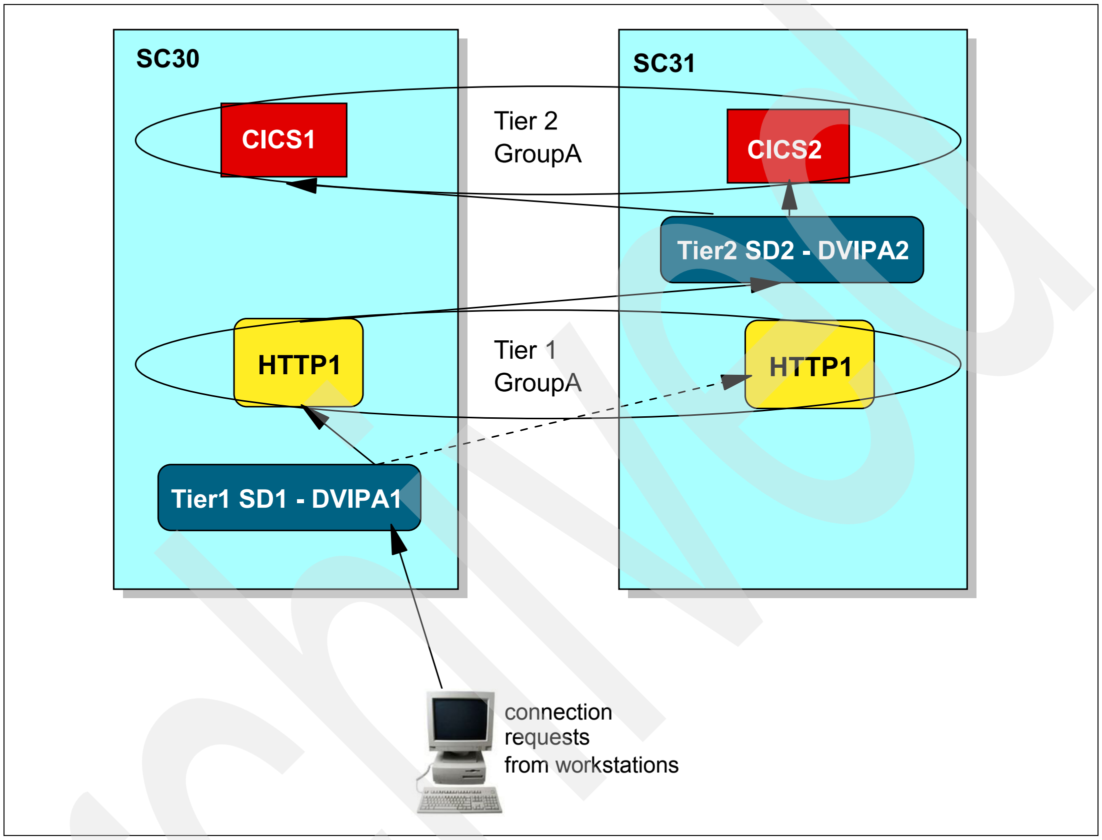


Figure 7-4 Tier 1 and tier 2 applications

When the chosen HTTP server connects to the tier 2 server and optimized local is in effect, that second connection remains on the chosen LPAR. When optimized local is configured with a default value of 1, the second connection remains local if the WLM weight is greater than 0 and the server is healthy. For more information, see 7.2.2, “OPTLOCAL option” on page 220.

Example 7-1 and Example 7-2 show the Dynamic VIPA configurations.

Example 7-1 VIPAD Configuration on SC30

```
VIPADEFINE TIER1 MOVE IMMED 255.255.255.0 10.1.8.24  ; TIER 1
VIPADISTRIBUTE TIER1 GROUP1 DISTMETHOD SERVERWLM
                     10.1.8.24 PORT 10002
                     DESTIP ALL

VIPABACKUP 100 TIER2 10.1.8.23
```

Example 7-2 VIPAD Configuration on SC31

```
VIPADEFINE TIER2 MOVE IMMED 255.255.255.0 10.1.8.23  ; TIER 2
```

```
VIPADISTRIBUTE TIER2 GROUP1 optlocal 1 DISTMETHOD SERVERWLM
                     10.1.8.23 PORT 10000
                     DESTIP ALL

VIPABACKUP 100 TIER1 10.1.8.24
```

When using tier 1 and tier 2 options, consider the following recommendations:

- For every VIPADISTRIBUTE DVIPA, you need to define a corresponding VIPADEFINE and VIPABACKUP.
- On the VIPADEFINE and VIPABACKUP, you can specify the following parameters:
  - TIER1 indicates that this a DVIPA that is used to distribute connections to a tier 1 target and a TIER1 VIPADISTRIBUTE statement follows.
  - TIER2 indicates that this a DVIPA that is used to distribute connections to a tier 2 target and a TIER2 VIPADISTRIBUTE statement follows.
- On the VIPADISTRIBUTE statement for tier 1, the TIER1 group name indicates that this is a distribute statement for a tier 1 DVIPA and port. The group name is used to correlate this group of applications with a corresponding group of tier 2 applications.
- On the VIPADISTRIBUTE statement for tier 2, the TIER2 group name indicates that this is a distribute statement for a tier 2 DVIPA and port. The group name is used to correlate this group of applications with a corresponding group of tier 1 applications.
- Specify OPTLOCAL with a value as required. For more information, see the next section.

7.2.2 OPTLOCAL option

The OPTLOCAL keyword allows the client to bypass sending the connection request to the Sysplex Distributor.

The following values influence the conditions in which the connection remains local:

- A value of 0 indicates that the connection should always remain local.
- A value of 1 indicates that the connection should remain local unless the server's WLM weight is zero (0).
- Values of 2 through 16 are used as multipliers to increase the local WLM weight to favor the local stack.

Regardless of the value specified, the connection request is always sent to the Sysplex Distributor if *any* of the following situations occur:

- No server application is available on the local stack.
- The Server Efficiency Fraction (SEF) value on the local stack is less than 75. A SEF value of 100 is the best value.

  SEF is influenced through monitoring the target server by the stack. This value is based on whether new connections are being established—and after being established, whether the server is accepting new connections.
- The health indicator for the local stack is less than 75 (available only for applications that provide this information to WLM using the IWM4HLTH or IWMSRSRG services).

  The health indicator provides a general health indication for an application or subsystem. Under normal circumstances, the value of this field is 100, meaning the server is 100% healthy. Any value less than 100 indicates that the server is experiencing problem conditions that are not allowing it to process new work requests successfully. A value of less than 100 also causes the WLM to reduce the recommendation provided to the Sysplex Distributor for this server instance.

- The abnormal transactions count for the local stack is greater than 250.

  A non-zero value indicates that the server application has reported abnormal transaction completions to WLM, and that WLM has reduced the server-specific recommendation for this server instance. The higher the value of this field, the greater the reduction in the recommendation provided by WLM.

  For more information regarding the conditions leading to abnormal transaction completions for a given server application, refer to the documentation provided by the server application.

OPTLOCAL is intended to be used mainly in context with DISTMETHOD SERVERWLM or DISTMETHOD BASEWLM. If ROUNDROBIN is used together with OPTLOCAL, then the OPTLOCAL value must be zero (0), because WLM weights are not being collected. If another value is defined, then the EZD1231I message will indicate that the OPTLOCAL value is set to zero (0).

7.2.3 CPCSCOPE option parameter

Dynamic VIPA (DVIPA) addresses can be defined with a scope of a CPC, meaning that these DVIPAs are specific to the central processor complex (CPC) where they are defined. These DVIPAs are never activated on LPARs that reside in other CPCs.

A DVIPA with this characteristic can be a z/OS or non-z/OS target system. In scenarios where Linux® on System z acts as the first tier server, you can use CPCSCOPE DVIPAs to keep tier 2 connections and traffic on the same CPC where Linux is running.

For example, the HTTP tier on CPC1 as shown in Figure 7-5 on page 222 is configured to connect to SD3 for the EIS tier, while the HTTP tier on CPC2 is configured to connect to SD2 for the EIS tier. In our example, LAPR# can be a z/OS or a Linux system, and this topology keeps traffic between the Linux systems and the z/OS systems on high speed networks, such as HiperSockets interfaces. The aim is to Improve overall response times due to reduced cross-CPC traffic in mixed Linux on System z and z/OS workload scenarios.

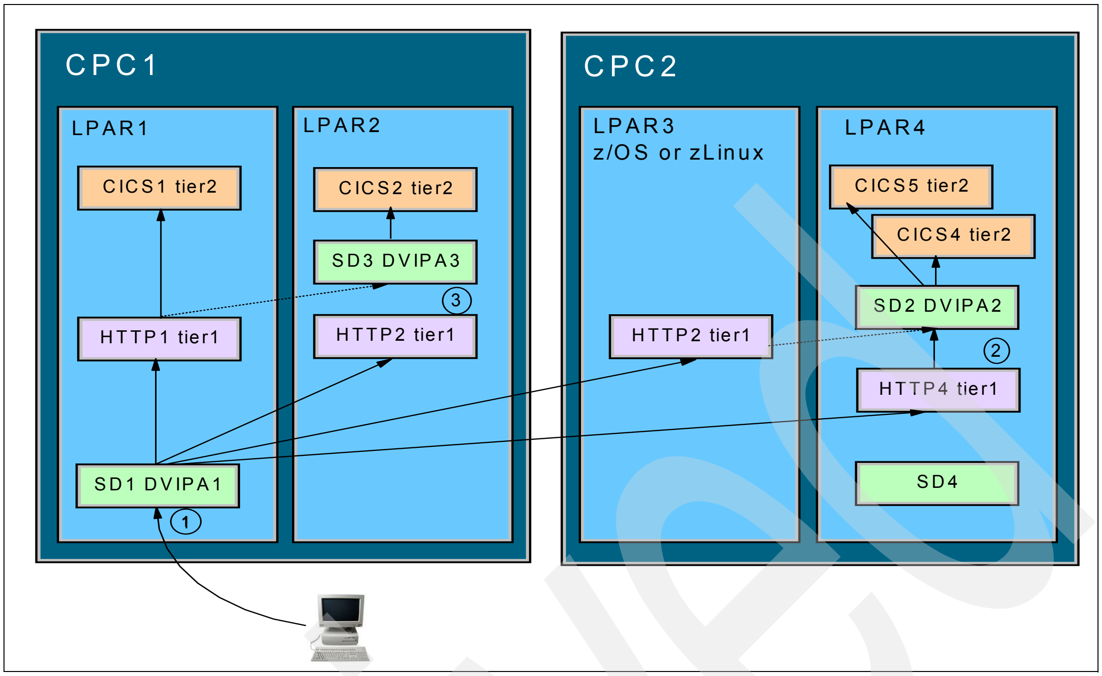


Figure 7-5 Multi-tier applications and CPC optimization

Example 7-3 through Example 7-6 list the definitions that we used to create the configuration shown in Figure 7-5.

Example 7-3 SD1 VIPAD configuration on LPAR1 (SC30)

```
VIPADEFINE TIER1 MOVE IMMED 255.255.255.0 10.1.8.24
VIPADISTRIBUTE TIER1 GROUP2 DISTMETHOD  SERVERWLM
                     10.1.8.24 PORT 8085
                     DESTIP ALL

VIPABACKUP 100 TIER2 10.1.8.26.25
```

Example 7-4 SD3 VIPAD Configuration on LPAR2 (SC31)

```
VIPADEFINE TIER2 CPCSCOPE MOVE IMMED 255.255.255.0 10.1.8.25
VIPADISTRIBUTE OPTLOCAL TIER2 GROUP2 DISTMETHOD  SERVERWLM
                        10.1.8.25 PORT 9001
                        DESTIP 10.1.7.11 10.1.7.12

VIPABACKUP 100 TIER1 10.1.8.24
```

Example 7-5 SD2 VIPAD Configuration on LPAR4 (SC33)

```
VIPADEFINE TIER2 MOVE IMMED 255.255.255.0 10.1.8.26
VIPADISTRIBUTE TIER2 GROUP2 DISTMETHOD SERVERWLM
                     10.1.8.26 PORT 9001
                     DESTIP 10.1.7.13 10.1.7.14
```

Example 7-6 SD4 VIPAD Configuration on LPAR4 (SC33)

```
VIPABACKUP 100 TIER1 10.1.8.26
```

When using this option, consider the following recommendations:

- For every VIPADISTRIBUTE DVIPA, you need to define a corresponding VIPADEFINE or VIPABACKUP.
- On the VIPADEFINE and VIPABACKUP, you can specify the following parameters:
  - TIER1 indicates that this a DVIPA that is used to distribute connections to a tier 1 target and a TIER1 VIPADISTRIBUTE statement follows.
  - TIER2 indicates that this a DVIPA that is used to distribute connections to a tier 2 target and a TIER2 VIPADISTRIBUTE statement follows.
  - TIER2 CPCSCOPE indicates that this a DVIPA that is used to distribute connections to a tier 2 target and a TIER2 VIPADISTRIBUTE statement follows. The movement of the DVIPA is restricted to the same CPC as where it is defined.
- On the VIPADISTRIBUTE statement for tier 1, the TIER1 group name indicates that this is a distribute statement for a tier 1 DVIPA and port. The group name is used to correlate this group of applications with a corresponding group of tier 2 applications.
- On the VIPADISTRIBUTE statement for tier 2, the TIER2 group name indicates that this is a distribute statement for a tier 2 DVIPA and port. The group name is used to correlate this group of applications with a corresponding group of tier 1 applications.
- Specify OPTLOCAL with a value as required.
- Define VIPABACKUP definitions for each of the tier 2 CPCSCOPE DVIPA on one of the stacks on an LPAR within the same CPC.
- A key requirement for this configuration is that you need to configure the tier 1 applications on each CPC to use a unique, CPC-specific DVIPA for tier 2 connections.This configuration implies that the tier 1 application server configurations need to be unique based on the CPC on which they reside.

7.2.4 IBM DataPower

DataPower is an IBM division that produces appliances for processing XML messages as well as any-to-any message transformation such as COBOL.

z/OS Communications Server continues to focus on logical integration between DataPower and z/OS by supporting a DataPower feedback technology to z/OS Sysplex Distributor. This integration allows Sysplex Distributor to make much higher quality load balancing decisions than any other existing load balancing technology when using connections to DataPower appliances.

DataPower appliances are often deployed as a front-end processing tier to z/OS applications, which provides for transparent Web services enablement of z/OS applications.

When the DataPower tier finishes handling a request, it typically routes a request to a tier 2 application that is hosted within the z/OS environment, such as a CICS, IMS, or WebSphere. These tier 2 requests might also require load balancing, especially when the applications are deployed in a sysplex environment.

A z/OS Communications Server feature makes Sysplex Distributor load balance requests to both the DataPower and z/OS tiers, eliminating the need to deploy an external, network-based load balancer for the DataPower tier. By deploying the Sysplex Distributor as the load balancing component for both tiers of distribution, z/OS administrators can use a single load balancing solution for the z/OS workload to the DataPower and z/OS application processing tiers, thus simplifying the load balancing administration.

A control connection between Sysplex Distributor and each DataPower appliance is established to exchange information as follows:

- DataPower to Sysplex Distributor: CPU capacity and connection state information as connections are established and terminated.
- Sysplex Distributor to DataPower: Sysplex Distributor sends a list of distributed DVIPA/Ports to the DataPower appliance to bring up their listeners.

Sysplex Distributor does not need to monitor connection traffic to determine when a connection ends, which allows optimized routing for outbound traffic because the packets do not need to traverse the Sysplex Distributor. It uses Generic Resource Encapsulation (GRE) to forward inbound distributed packets to DataPower. Outbound packets can flow directly from DataPower to the client.

The enhancement optimizes the load balancing support in several ways:

- Routing is optimized so that outbound traffic (from the tier 1 DataPower target server towards the client) does not need to traverse the sysplex distributor.
- Connection information provided by the DataPower appliances allows nondisruptive tier 1 takeover of existing connections between clients and DataPower targets.
- CPU usage information provided by the DataPower appliance allows sysplex distributor to optimize its load balancing decisions.

When using DataPower, keep in mind the following considerations:

- All TCP/IP stacks that participate in the Sysplex Distributor distributor support with DataPower must be V1R11 or later.
- Only IPv4 DVIPAs are supported for connections to DataPower targets.

Sysplex Distributor configuration for optimized load balancing with DataPower

This scenario shows how to use Sysplex Distributor with DataPower appliances.

Figure 7-6 shows two DataPower appliances, *DATAP1* and *DATAP2*, acting as tier 1 applications, and two WebSphere Application Servers, *WAS1* and *WAS2*, as tier 2 applications.

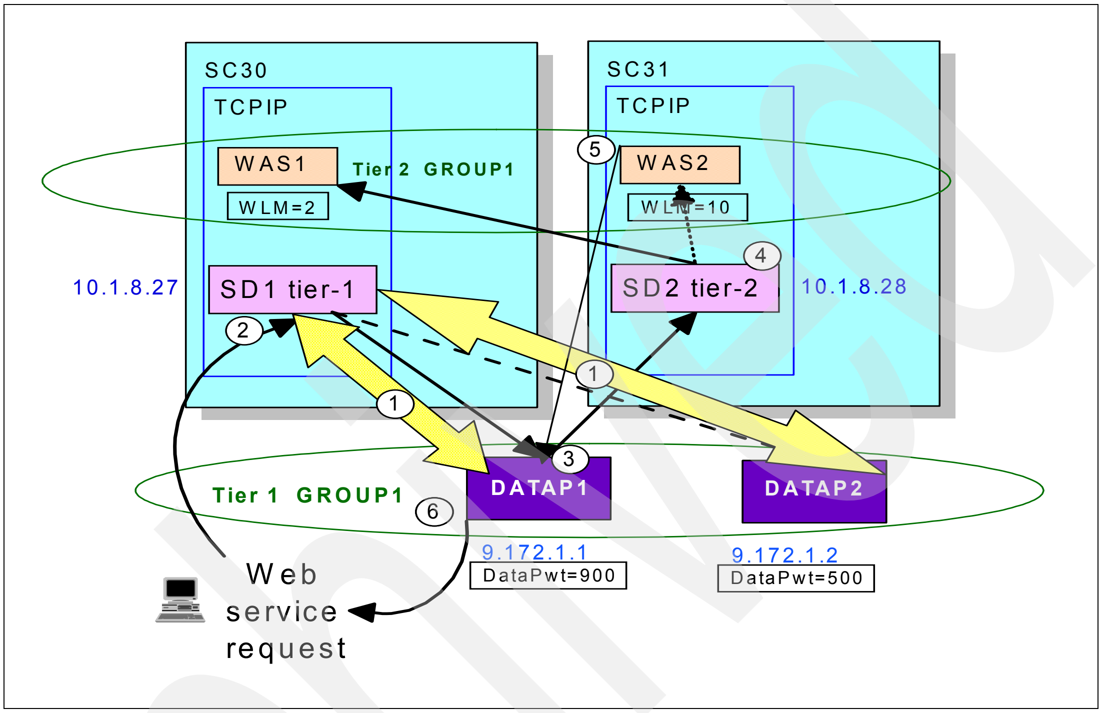


Figure 7-6 Sysplex Distributor distributing to DataPower tier 1and WebSphere Application Server tier2

This scenario has the following steps:

1. Sysplex Distributor and DataPower appliances exchange information over a TCP control connection (port 5601) in our example). The information is used to calculate and decide to which DataPower appliance it is going to distribute the tier 1 inbound connections. Periodically, the DataPower sends weights that are based on the overall CPU usage of the appliances, and connection state information to Sysplex Distributor, so that Sysplex Distributor does not need to monitor whether the connections are established or terminated.
2. A tier 1 distributed DVIPA (10.1.8.27) is defined to represent the cluster of DataPower appliances that can process a client Web service request to TCP (port 8087). For each TCP connection request sent to the tier 1 DVIPA and port, the tier 1 Sysplex Distributor makes a load-balancing decision and routes the request to one of the eligible DataPower target instances.
3. After the DataPower targets complete their processing of the inbound Web service request, they establish a connection and send the work request to Sysplex Distributor tier 2.
4. Sysplex Distributor tier 2 routes the request to the target WebSphere Application Server tier 2 based on WebSphere Application Server tier 2 weights.

5. After WebSphere Application Server tier 2 processes the results of the request are sent to the DataPower instance that originated the request.
6. DataPower can then perform any necessary outbound processing and send a response back to the client that originated the Web service request.

The tier 1 distributor determines a combined weight of each tier 1 target based on the received DataPower weight and the corresponding tier 2 server weight. The combined weights of all the DataPower appliances are then normalized against each other. For example, the combined weight of DATAP1 = (WAS1(2) + WAS2 (10)) * DP Weight 900 = 10800.

Example 7-7 and Example 7-8 show the VIPADistribute configuration on TCP/IP profile.

Example 7-7 VIPADISTRIBUTE configuration on SC30

```
VIPADYNAMIC
VIPADEFINE TIER1 255.255.255.0 10.1.8.27
VIPADISTRIBUTE TIER1 GROUP1 GRE CONTROLPORT 1 5601
                     DISTMETHOD TARGCONTROLLED  2
                     10.1.8.27 PORT 8087
                     DESTIP 9.172.1.1 9.172.1.2
VIPABACKUP 100 TIER2 10.1.8.28
ENDVIPADYNAMIC
```

In this example, the numbers correspond to the following information:

1. CONTROLPORT means that the Sysplex Distributor opens a control connection to the DataPower Sysplex client listening on that port, which is 5601.
2. TARGCONTROLLED means that the distribution is controlled by the target that is using the weights that it sends. The Sysplex Distributor uses a weighted round-robin distribution to the DataPower targets using the normalized DataPower weights received on the control connection. Weighted Active and Round Robin distribution can also be used.

The addresses specified in the DESTIP parameter for Example 7-7 are the addresses of the DataPower boxes and not the normal z/OS target stacks, which is the case whenever DISTMETHOD TARGETCONTROLLED is specified.

Example 7-8 VIPADISTRIBUTE configuration on SC31

```
VIPADYNAMIC
VIPADEFINE TIER2 MOVE IMMED 255.255.255.0 10.1.8.28
VIPADISTRIBUTE TIER2 GROUP1 DISTMETHOD  SERVERWLM
                        10.1.8.28 PORT 9003
                        DESTIP 10.1.7.11 10.1.7.12

VIPABACKUP 100 TIER1 10.1.8.27
ENDVIPADYNAMIC
```

Configuration verification

You can use the VIPADCFG display to see the values of the configured VIPADYNAMIC parameters, as shown in Example 7-9.

Example 7-9 NETSTAT VIPADCFG on SC30

```
D TCPIP,TCPIPA,N,VIPADCFG,DETAIL
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IPADDR/PREFIXLEN: 10.1.8.28
      RANK: 100  MOVEABLE:            SRVMGR:     FLG: 2
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.27/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG: 1 1
  VIPA DISTRIBUTE:
    DEST:      10.1.8.27..8087
      DESTXCF: 9.172.1.1
      SYSPT:   NO   TIMAFF: NO    FLG: TARGCTRL TIER1   4
      OPTLOC:  NO
      GRPNAME: GROUP1  2 RTGTYPE: GRE   CTRLPORT: 5601
    DEST:      10.1.8.27..8087
      DESTXCF: 9.172.1.2
      SYSPT:   NO   TIMAFF: NO    FLG: TARGCTRL TIER1   2
      OPTLOC:  NO
      GRPNAME: GROUP1             RTGTYPE: GRE 5 CTRLPORT: 5601 3
END OF THE REPORT
```

In this example, the numbers correspond to the following information:

1. The display has a flag field (Flg) which indicates if tier 1 was configured.
2. Name of the group to correlate the two-tier distribution.
3. Port 5601 specifies the port number used for the control connection between Sysplex Distributor and DataPower boxes.
4. Indicates which distribution method was configured: TARGECONTRolled.
5. Indicates whether Generic Routing Encapsulation (GRE) is used when routing requests to tier 1 target.

You can use the VDPT display to view the current number of active connections, as shown in Example 7-10 through Example 7-12.

Example 7-10 NETSTAT VDPT on SC30

```
D TCPIP,TCPIPA,N,VDPT,DETAIL
DYNAMIC VIPA DESTINATION PORT TABLE FOR NON-Z/OS TARGETS:
DEST:       10.1.8.27..8087
  TARGET ADDR: 9.172.1.1    1
  TOTALCONN: 0000000000  RDY: 000  WT: 00 CWT: 000  3
  FLG: TARGCTRL
    T1WT: 000  2
    ACTCONN:   0000000000
DEST:       10.1.8.27..8087
  TARGET ADDR: 9.172.1.2    1
  TOTALCONN: 0000000000  RDY: 000  WT: 00 CWT: 000  3
  FLG: TARGCTRL
    T1WT: 000  2
```

```
   ACTCONN:   0000000000
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

In this example, the numbers correspond to the following information:

1. Target IP address for non-z/OS, DataPower appliances.

2. The weight that is received from the tier 1 target.

3. The value that represents the combined weight of tier 2 target servers that are on the same CPC as the tier 1 target.

Example 7-11 NETSTAT VIPADCFG on SC31

```
D TCPIP,TCPIPA,N,VIPADCFG,DETAIL
DYNAMIC VIPA INFORMATION:
  VIPA BACKUP:
    IPADDR/PREFIXLEN: 10.1.8.27
      RANK: 100  MOVEABLE:           SRVMGR:     FLG: 1  2
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.1.8.28/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO  FLG: 2
  VIPA DISTRIBUTE:
    DEST:      10.1.8.28..9003
      DESTXCF: 10.1.7.11
      SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM TIER2 1
      OPTLOC:  NO
      GRPNAME: GROUP2
      PROCXCOST:
        ZAAP: 001  ZIIP: 001
      ILWEIGHTING: 0
    DEST:      10.1.8.28..9003
      DESTXCF: 10.1.7.12
      SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM TIER2
      OPTLOC:  NO
      GRPNAME: GROUP2
      PROCXCOST:
        ZAAP: 001  ZIIP: 001
      ILWEIGHTING: 0
END OF THE REPORT
```

In this example, the numbers correspond to the following information:

1. The display has a flag field (Flg), which indicates whether tier 2 was configured.

2. VIPABACKUP of the tier 1 Sysplex Distributor.

Example 7-12 NETSTAT VDPT on SC31

```
D TCPIP,TCPIPA,N,VDPT,DETAIL
DYNAMIC VIPA DESTINATION PORT TABLE FOR TCP/IP STACKS:
DEST:        10.1.8.28..9003
  DESTXCF:   10.1.7.12
  TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100
  FLG: SERVERWLM, TIER2
    TCSR: 100  CER: 100 SEF: 100
    WEIGHT: 00
      RAW         CP: 00 ZAAP: 00 ZIIP: 00
```

```
      PROPORTIONAL CP: 00 ZAAP: 00 ZIIP: 00
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 00
1 OF 1 RECORDS DISPLAYED
END OF THE REPORT
```

7.2.5 Applied system configuration for optimized load balancing

We used the configuration shown in Figure 7-7 for our test scenarios.

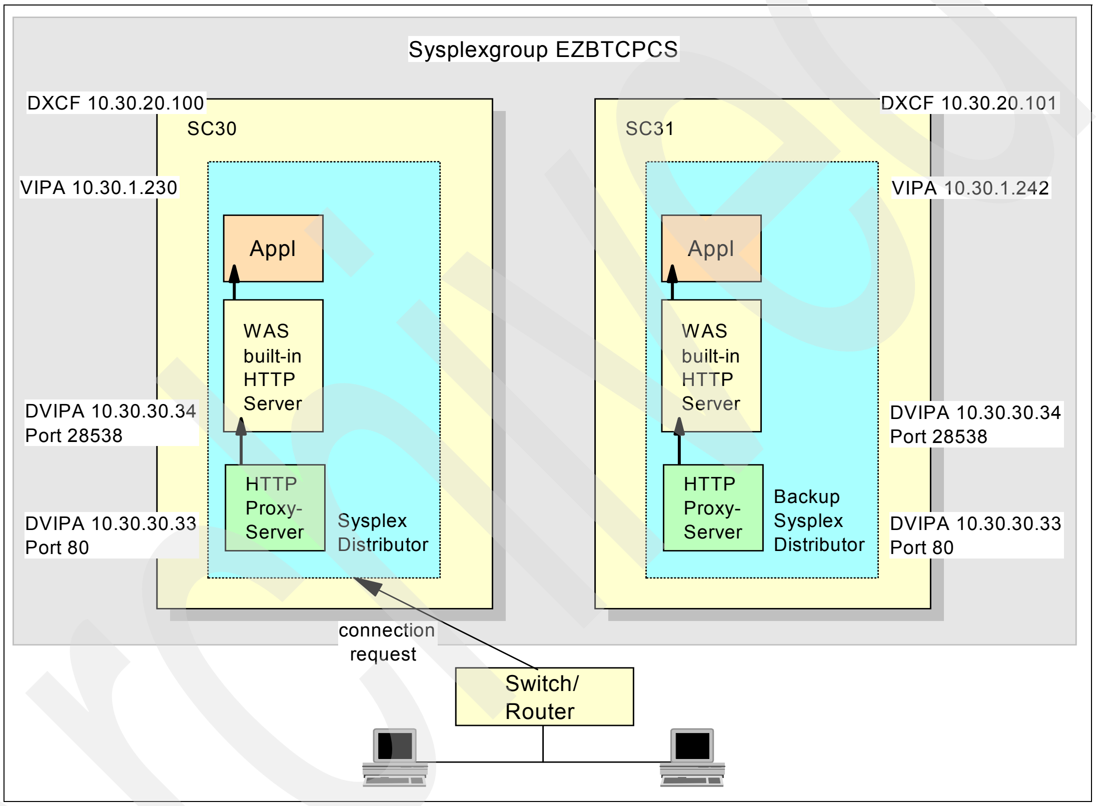


Figure 7-7 Multitier optimized load balancing test environment

This configuration consisted of two LPARs (SC30 and SC31). Each LPAR was configured with one TCP/IP stack. The primary Sysplex Distributor was in SC30, and a backup Sysplex Distributor was in the SC31 stack. In order to build a multitier application scenario testing the VIPADISTRIBUTE OPTLOCAL keyword, we installed on each stack one HTTP proxy server and a WebSphere Application Server with a deployment manager.

The HTTP proxy server was defined as distributed DVIPA with IP cluster address 10.30.30.33 and port 80 for the Sysplex Distributor. The WebSphere Application Server with its built-in HTTP server got the distributed DVIPA cluster address 10.30.30.34 with port 28538. We also had to define additional port numbers for the WebSphere Application Server environment (deployment manager, control region, and so on).

To keep the test scenario simple, we did not use an EIS (such as IMS, DB2 or CICS). Instead, we used a special test application called StockTrade provided by the WebSphere Application Server implementation packet.

We used two kinds of HTTP connection setup requests:

- From a workstation 10.12.4.198 with MS Internet Explorer (see Figure 7-9)
- Through a WebSphere Workload Simulator

TCP/IP profile definitions for SC30 and SC31

The definitions in Example 7-13 and Example 7-14 only show specific VIPADYNAMIC statements. OPTLOCAL 0 is defined for IP address 10.30.30.34, with different ports for WebSphere Application Server applications. This definition forces the selection of the WebSphere Application Server built-in HTTP server in the same LPAR as the HTTP proxy server as long as it is healthy.

The HTTP proxy server gets the client connection requests forwarded from the Sysplex Distributor. It does not need the OPTLOCAL keyword.

Example 7-13 shows the VIPADYNAMIC definitions for the SC30 primary Sysplex Distributor.

Example 7-13 VIPADYNAMIC definitions for SC30 primary Sysplex Distributor

```
VIPADynamic
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.32
   VIPADISTribute DISTMethod BASEWLM 10.30.30.32
   PORT 23 20 21 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.33
   VIPADISTribute DISTMethod SERVERWLM 10.30.30.33
   PORT 80 DESTIP 10.30.20.100 10.30.20.101
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.34
   VIPADISTribute DISTMethod SERVERWLM 10.30.30.34 OPTLOCAL 0
   PORT 28538 28518 28510 28502 28503 DESTIP 10.30.20.100 10.30.20.101
```

Example 7-14 shows the VIPADYNAMIC definitions for the SC31 backup Sysplex Distributor.

Example 7-14 VIPADYNAMIC definitions for SC31 backup Sysplex Distributor

```
VIPADynamic
 VIPABACKUP 100 MOVEABLE IMMEDIATE
   255.255.255.0 10.30.30.32 10.30.30.33 10.30.30.34
VIPADISTribute define DISTMethod BASEWLM 10.30.30.32
  PORT 23 20 21 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
VIPADISTribute define DISTMethod SERVERWLM 10.30.30.33
  PORT 80 DESTIP 10.30.20.100 10.30.20.101
VIPADISTribute define DISTMethod SERVERWLM OPTLOCAL 0 10.30.30.34
  PORT 28538 28518 28510 28502 28503 DESTIP 10.30.20.100 10.30.20.101
 VIPAROUTE 10.30.20.100 10.30.1.230
 VIPAROUTE 10.30.20.101 10.30.1.242
 VIPAROUTE 10.30.20.102 10.30.1.221
ENDVIPADynamic
```

Request flow between client, WebSphere Application Server, and application endpoint

Figure 7-8 shows the data flow from a client to the WebSphere Application Server application with the URL StockTrade/CPUBound.

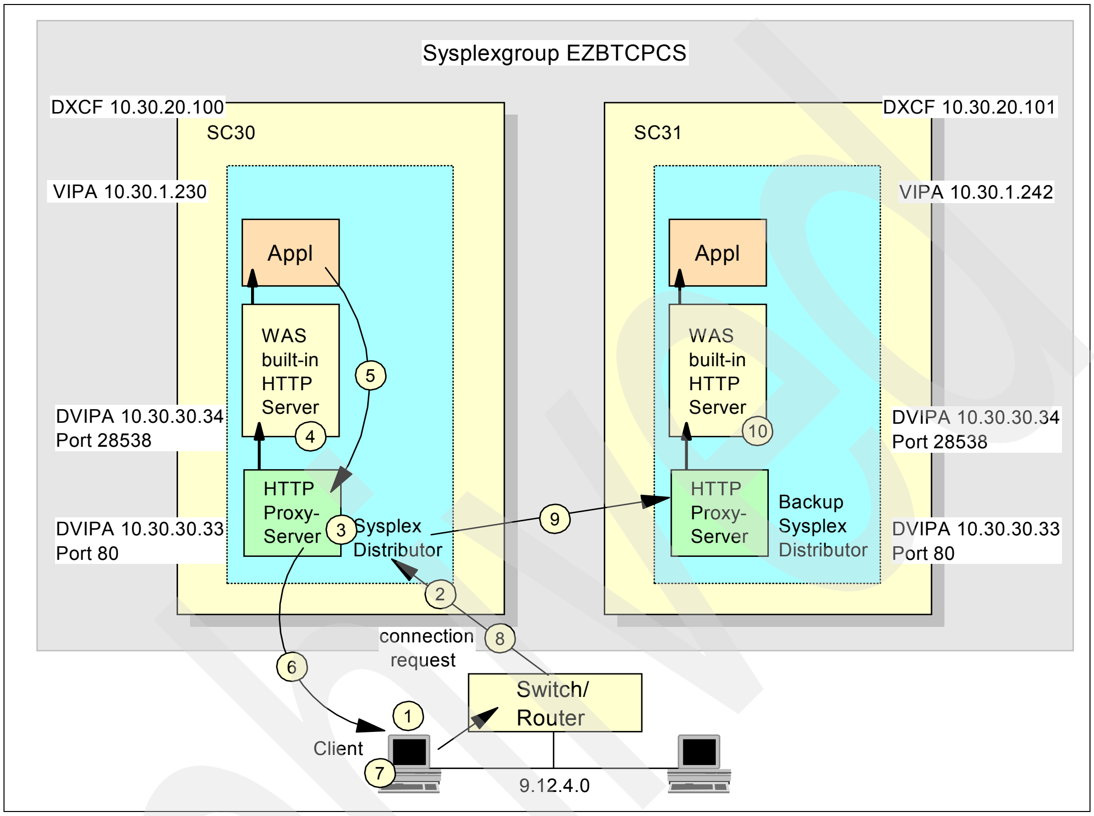


Figure 7-8 HTTP request flow

The steps in the HTTP request flow, as depicted in the figure, are explained here:

1. The user enters an initial request with URL `StockTrade/CPUBound` in the client browser. The request has no session affinity.
2. The request is resolved to a cluster IP address (10.30.30.33) and port address (80), and is sent to the primary Sysplex Distributor running in SC30 (or to the backup Sysplex Distributor SC31, if the primary is down).
3. The Sysplex Distributor makes its load balancing decision based on the best server-specific WLM values for the HTTP proxy servers on SC30 and SC31. It forwards the client request to the selected HTTP server. Both HTTP servers are configured with plug-in configuration file of the WebSphere Application Server (`plug-in-cfg.xml` file).
4. The configuration files describes which URLs are handled by WebSphere Application Server. In our case, it is the application StockTrade, which can be reached through the distributed DVIPA cluster address 10.30.30.34 and port 28538, in WebSphere Application Server on SC30 and SC31.

   The application runs on both target systems SC30 and SC31. The plug-in checks for possible session affinity by scanning the request for session cookies. The first request has no affinity. The plug-in recognizes the URL `/StockTrade/CPUBound`, and routes it to the IP cluster address 10.30.3034 with port 28538, based on:

- The DVIPA address of the Sysplex Distributor forwarding it to WebSphere Application Server on SC30 or SC31 based on best SERVERWLM values if no OPTLOCAL keyword was defined.
- Directly to the WebSphere Application Server in the local system if OPTLOCAL 0 was defined (see Figure 7-3 on page 218).
- Directly to the WebSphere Application Server in the local system if OPTLOCAL 1 (see Figure 7-3 on page 218) was defined and sufficient capacity is provided by the local WebSphere Application Server. If the SERVERWLM values are zero (0), then the Sysplex Distributor is involved to search for a better WebSphere Application Server in the sysplex group to reach the WLM goal.

The HTTP listener on port 28538 in the WebSphere Application Server receives the request. There is no session affinity at this time. The session ID will be created later when the destination application builds a session object with a session ID. Later on, a session cookie is added to the HTTP data stream in the response to the client.

5. The response with the jsessionID cookie in the HTTP header is sent back to the Web server (in our case, to the HTTP proxy server in the system where the request came from).
6. The response is sent back to the client.
7. The browser stores the cookie in its memory. (Session cookies are never stored on a disk.) The cookie stays there as long as the browser is not closed, or until it is deleted (because of expiration) by the server.
8. The cookie is appended to the HTTP request at the next request to the server.
9. The Sysplex Distributor does not consider any session affinity. Therefore, there is no cookie check and the request could be sent to a different HTTP server.
10. The plug-in scans the HTTP header of the request and finds the session ID cookie with an additional CloneID. It compares this CloneID with its definition in the plug-in configuration (plug-in-cfg.xml) file. If there is a match, the request is sent to the IP address and port of the matched application server instance.

For more details about WebSphere Application Server session handling, refer to *Architecting High Availability Using WebSphere V6 on z/OS*, SG24-6850.

Figure 7-9 depicts a browser connection request to an HTTP proxy server, which is sent to the Sysplex Distributor. In this case, the Sysplex Distributor sent the request to the HTTP server on system SC31 where the server with its job name BWSR02B was started.

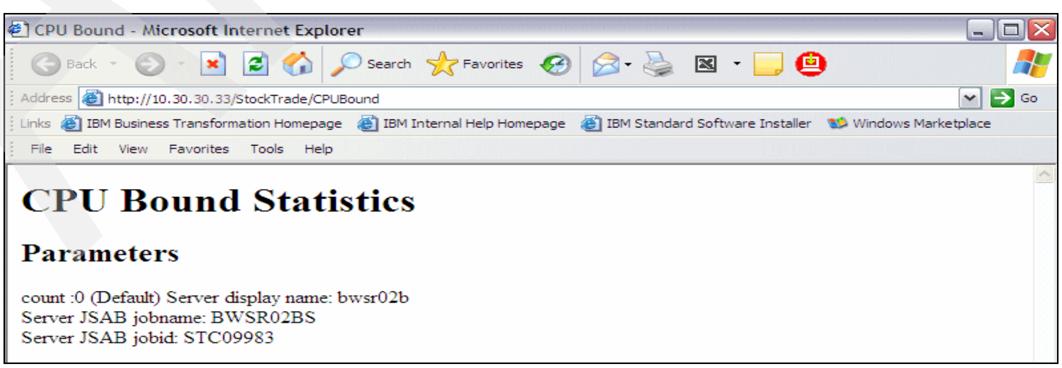


Figure 7-9 Web browser connection to HTTP server in SC31

Another request was directed to the HTTP server on system SC30 with its job name BWSR01A (see Figure 7-10).

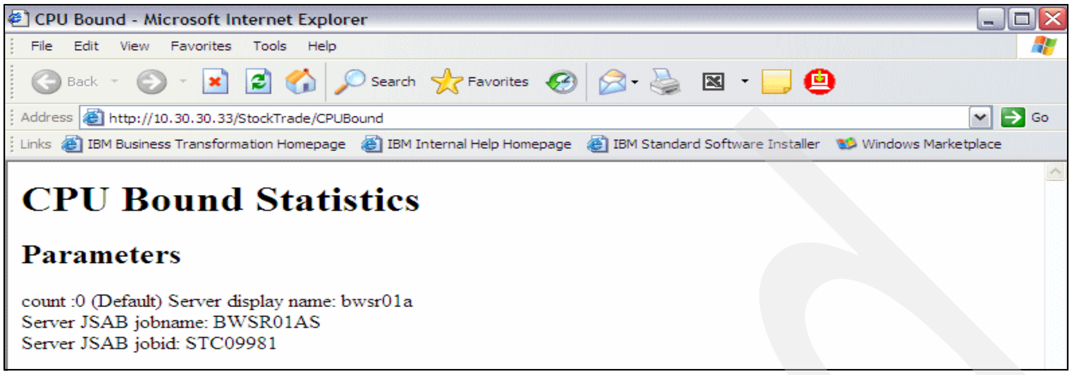


Figure 7-10 Web browser connection to HTTP server in SC30

7.2.6 OPTLOCAL test cases

In this section, we discuss the use of the OPTLOCAL keyword in conjunction with Sysplex Distributor and an external load balancer that supports Server Application State Protocol (SASP). Figure 7-7 on page 229 describes our configuration with the needed IP and port addresses of the components that we used.

Optimized path through Sysplex Distributor, using OPTLOCAL 0

The goal here is to show the impact of the keyword OPTLOCAL 0 in the VIPADISTRIBUTE statement for WebSphere Application Server only. Figure 7-11 shows the optimized path between client and the Web application. Because the Sysplex Distributor has two choices, the optimized path can be created in SC30 or SC31, between the HTTP server and WebSphere Application Server.

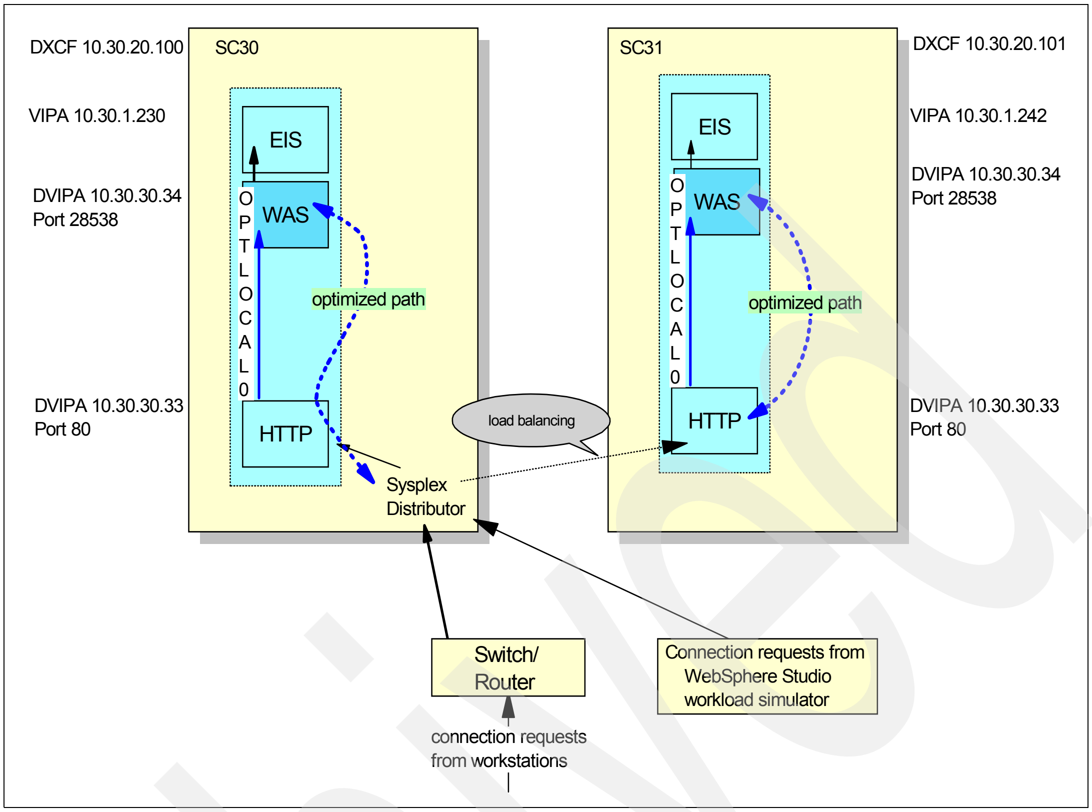


Figure 7-11 Optimized path through OPTLOCAL 0

TCP/IP profile definitions

Example 7-15 shows the TCP/IP profile definitions that we used in our environment.

Example 7-15 VIPADYNAMIC definitions

```
VIPADEFine/VIPADISTRIBUTE
VIPADynamic
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.32
   VIPADISTribute DISTMethod BASEWLM 10.30.30.32
   PORT 23 20 21 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.33
   VIPADISTribute DISTMethod SERVERWLM 10.30.30.33
   PORT 80 DESTIP 10.30.20.100 10.30.20.101
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.34
   VIPADISTribute DISTMethod SERVERWLM OPTLOCAL 0 10.30.30.34
   PORT 28538 28518 28510 28502 28503 DESTIP 10.30.20.100 10.30.20.101
   VIPAROUTE 10.30.20.100 10.30.1.230
   VIPAROUTE 10.30.20.101 10.30.1.242
   VIPAROUTE 10.30.20.102 10.30.1.221
ENDVIPADYNAMIC
```

Example 7-16 points to the flag that displays if OPTLOCAL is switched on for IP address 10.30.30.34 and port address 28538 of WebSphere Application Server. In our case, it is defined as SERVERWLM and OPTLOCAL 0.

Example 7-16 netstat vipadcfg shows the OPTLOCAL definitions

```
D TCPIP,TCPIPC,N,VIPADCFG,DETAIL

DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
IPADDR/PREFIXLEN: 10.30.30.34/24
    MOVEABLE: IMMEDIATE  SRVMGR: NO

DEST:        10.30.30.34..28538
      DESTXCF:   10.30.20.100
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM OPTLOCAL
        OPTLOC:  0
    DEST:        10.30.30.34..28538
      DESTXCF:   10.30.20.101
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM OPTLOCAL
        OPTLOC:  0
```

Client requests from the Web browser are directed to the WebSphere Application Server application (see Figure 7-9 on page 232) on SC30 or SC31. Other client requests are initiated through the WebSphere Studio workload simulator and also directed to the WebSphere Application Server on SC30 or SC31.

The **netstat conn** command shown in Figure 7-17 indicates that the HTTP proxy server on SC31 with the name WEBBW311 received connections from a client with an IP address 10.30.2.50. This is the WebSphere Studio workload simulator we used to produce client connection requests.

Example 7-17 Current connection check of HTTP server on SC31

```
D TCPIP,TCPIPC,N,CONN,IPPORT=10.30.30.33+80

USER ID  CONN     STATE
WEBBW311 0000A6C2 ESTBLSH
  LOCAL SOCKET:   10.30.30.33..80
  FOREIGN SOCKET: 10.30.2.50..3255
WEBBW311 0000A6C4 ESTBLSH
  LOCAL SOCKET:   10.30.30.33..80
  FOREIGN SOCKET: 10.30.2.50..3256
WEBBW311 0000A6C6 ESTBLSH
  LOCAL SOCKET:   10.30.30.33..80
  FOREIGN SOCKET: 10.30.2.50..3257
```

In regard to the current HTTP TCP connections of the HTTP servers, you will notice that those connections will disappear at some point in time. This is due to the HTTP architecture, which allows the server to close the connections because they are no longer needed. The HTTP server (working as a client), starts a new connection to the WebSphere Application Server built-in HTTP server to forward the client request.

Example 7-18 shows the VIPA distribution port table used in the sysplex for SC30 and SC31. It also depicts how many connections the Sysplex Distributor forwarded to the two HTTP servers using a load balancing service. Because OPTLOCAL 0 was defined, we assume that the depicted TOTALCONN number came from the local HTTP server identified through the DESTXCF address, for example, for 10.30.20.100. This is the address of the TCP/IP stack in SC30. Equivalent values are shown for the second HTTP server on SC31.

Thus, 89 connections came to the HTTP proxy server on SC30 and 157 to SC31. For each request, the HTTP server started a new connection to IP address 10.30.30.34. and port address 28538 (of the WebSphere Application Server). The local connections in the same LPAR, however, are separated from each other in the sysplex.

Example 7-18 VIPA distribution port table for the WebSphere Application Server

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

 DYNAMIC VIPA DESTINATION PORT TABLE:
 DEST:        10.30.30.34..28538
   DESTXCF:   10.30.20.100
   TOTALCONN: 0000000089  RDY: 001  WLM: 08  TSR: 100
   FLG: SERVERWLM, LOCAL
     TCSR: 100  CER: 100 SEF: 100
     ABNORM: 0000       HEALTH: 100
     ACTCONN:   0000000005
     QOSPLCACT: *DEFAULT*
       W/Q: 08
 DEST:        10.30.30.34..28538
   DESTXCF:   10.30.20.101
   TOTALCONN: 0000000157  RDY: 001  WLM: 16  TSR: 100
   FLG: SERVERWLM, LOCAL
     TCSR: 100  CER: 100 SEF: 100
     ABNORM: 0000       HEALTH: 100
     ACTCONN:   0000000009
     QOSPLCACT: *DEFAULT*
       W/Q: 16
 2 OF 2 RECORDS DISPLAYED
 END OF THE REPORT
```

You might wonder why the server WLM values in each system are different with 08 and 16. The proportion of the TOTALCONN values are also a 1:2 ratio. If both LPARs would have been defined equally regarding resources such as logical CPU, storage, and so on, then the number of distributed requests would be nearly equal. We later found out that SC31 used four CPUs and SC30 was defined with two CPUs only.

This 1:2 ratio can also be recognized in Figure 7-19 when showing the VIPA distribution table of the HTTP servers with the cluster address of 10.30.30.33 and port address of 80 on both systems. The TOTALCONN values have nearly a 1:2 ratio. **1**

Example 7-19 VIPA distribution port table

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.33+80

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.33..80
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000001496  RDY: 001  WLM: 08  TSR: 100        1
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
```

```
    QOSPLCACT: *DEFAULT*
      W/Q: 08
DEST:         10.30.30.33..80
  DESTXCF:    10.30.20.101
  TOTALCONN: 0000002863  RDY: 001  WLM: 16  TSR: 100        1
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000008
    QOSPLCACT: *DEFAULT*
      W/Q: 16
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

The **netstat conn** command shown in Example 7-17 on page 235 also indicates that the HTTP proxy server on SC31 with the name WEBBW311 receives connections from a client with an IP address 10.30.2.50. This is the WebSphere Studio workload simulator we used to produce client connection requests.

One WebSphere Application Server is down

When the WebSphere Application Server service goes down, shown in Example 7-20 on page 237, the output value from the **netstat vdpt detail** command shows `RDY: 000` (1), instead of RDY: 001(2), whereas the value determines how many servers are ready, active, and listening.

Example 7-20 One WebSphere application server is down

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:         10.30.30.34..28538
  DESTXCF:    10.30.20.100
  TOTALCONN: 0000000707  RDY: 000  WLM: 09  TSR: 100     1  3
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 09
DEST:         10.30.30.34..28538
  DESTXCF:    10.30.20.101
  TOTALCONN: 0000002036  RDY: 001  WLM: 06  TSR: 100     2  3
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000984
    QOSPLCACT: *DEFAULT*
      W/Q: 06
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

Further display commands show that the `TOTALCONN` 3 value does not increase, because the server is no longer being considered for new connections. The actual connection (ACTCONN) value for SC30 is always zero (0). The value for `ACTCONN` for the other member of the same HTTP cluster in SC31 shows different values.

Check whether both HTTP servers are active

Example 7-21 shows that the HTTP server on SC31 (`10.30.30.33` with DESTXCF address `10.30.20.101`) did not start correctly. Refer to **1** (`RDY: 000` and `WLM: 00`).

Example 7-21 HTTP server check id active

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.33+80

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:       10.30.30.33..80
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000000007  RDY: 001  WLM: 15  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 15
DEST:       10.30.30.33..80
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100      1
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 00
2 OF 2 RECORDS DISPLAYED
END OF THE REPORT
```

Extended NETSTAT command and displays

The following additional keywords of the **`netstat`** command allow more convenient operation:

- You can use the **`server`** keyword on the **`netstat ALL SERVER`** command to filter out application servers in listen status, thus reducing the number of display lines; see Example 7-22.

Example 7-22 Command netstat all server in listen state

```
===> netstat all server
Client Name: BWSR01A                   Client Id: 000091C6
  Local Socket: 0.0.0.0..28538
  Foreign Socket: 0.0.0.0..0
    BytesIn:            00000000000000000000
    BytesOut:           00000000000000000000
    SegmentsIn:         00000000000000000000
    SegmentsOut:        00000000000000000000
    Last Touched:       15:29:30          State:              Listen
    RcvNxt:             0000000000        SndNxt:             0000000000
    ClientRcvNxt:       0000000000        ClientSndNxt:       0000000000
    InitRcvSeqNum:      0000000000        InitSndSeqNum:      0000000000
    CongestionWindow:   0000000000        SlowStartThreshold: 0000000000
    IncomingWindowNum:  0000000000        OutgoingWindowNum:  0000000000
    SndWl1:             0000000000        SndWl2:             0000000000
    SndWnd:             0000000000        MaxSndWnd:          0000000000
    SndUna:             0000000000        rtt_seq:            0000000000
```

```
    MaximumSegmentSize: 0000000536        DSField:            00
  Round-trip information:
    Smooth trip time: 0.000               SmoothTripVariance: 1500.000
    ReXmt:              0000000000        ReXmtCount:         0000000000
    DupACKs:            0000000000
    SockOpt:            8800              TcpTimer:           00
    TcpSig:             00                TcpSel:             20
    TcpDet:             C0                TcpPol:             00
    QOSPolicyRuleName:
    ReceiveBufferSize:  0000016384        SendBufferSize:     0000016384
    ConnectionsIn:      0000000001        ConnectionsDropped: 0000000000
    CurrentBacklog:     0000000000        MaximumBacklog:     0000000010
    CurrentConnections: 0000000000        SEF:                100
    Quiesced: No
```

- You can use the keyword **ipport** on **netstat ALL**, **CONN**, **ALLConn**, **SOCKets**, **TELnet**, **VDPT**, and **VCRT** to filter a specific server IP address plus port out of the huge display list.

OPTLOCAL with external load balancer

This configuration differs from the example discussed in "Optimized path through Sysplex Distributor, using OPTLOCAL 0" on page 233 such that in this configuration, the distribution decision is not performed in the sysplex by the Sysplex Distributor. Instead, it is performed by an external load balancer such as Content Switching Module (CSM).

The external load balancer gets workload information for the balancing decision from the z/OS LBAdvisor. The LBAdvisor is informed about active and non-active application servers within the sysplex. System and server WLM information are provided by z/OS LBAgents in each LPAR. For further information about external load balancing, refer to Chapter 5, "Internal application workload balancing" on page 95.

Figure 7-12 depicts the optimized path from the client to the application server.

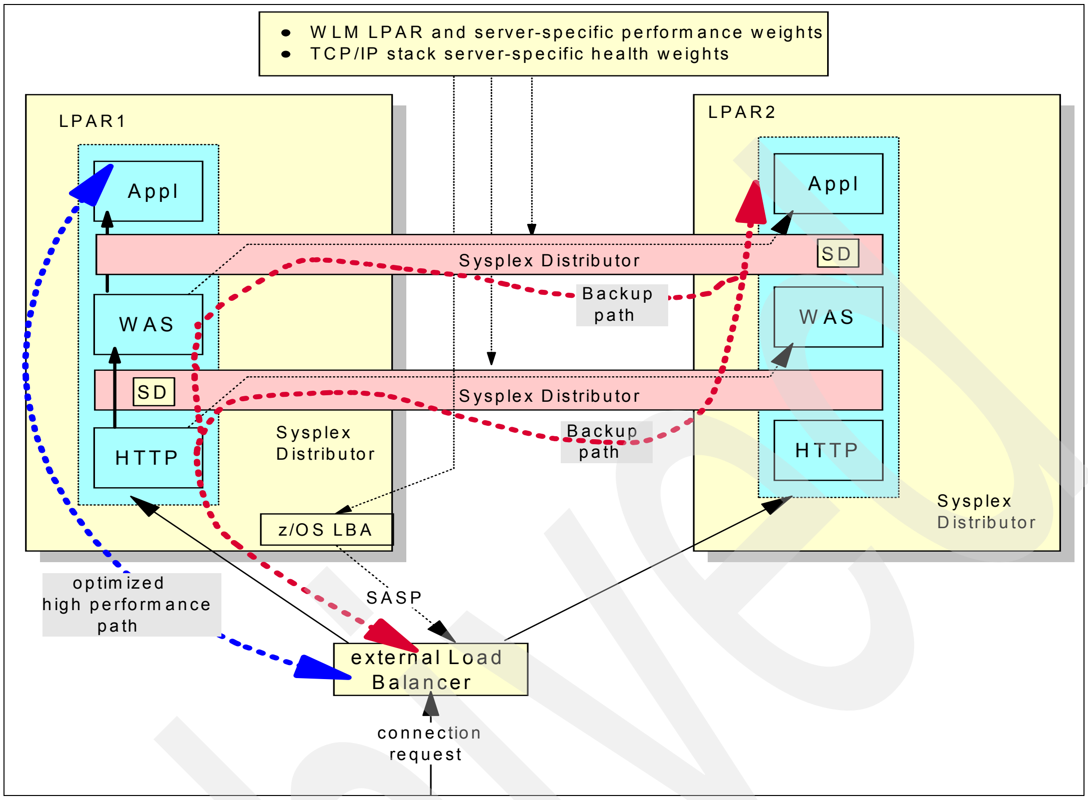


Figure 7-12 Optimized path using external load balancer

As shown in the figure, no Sysplex Distributor is necessary for the distribution of connections to the HTTP server. The optimized high performance path starts at the HTTP server either in SC30 or SC31, regardless of whether an external load balancer or the Sysplex Distributor selected the HTTP server. This means that if the external load balancer decides to send the connection request directly to the HTTP server in SC30, the same TCP/IP profile definitions for the path to the WebSphere Application Server can be used as defined and tested in "Optimized path through Sysplex Distributor, using OPTLOCAL 0" on page 233.

TCP/IP profile definition for OPTLOCAL 0

Example 7-23 shows the definitions that we used previously.

Example 7-23 TCP/IP profile definitions used in context with external load balancer

```
; VIPADEFine/VIPADISTRIBUTE
VIPADynamic
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.34
   VIPADISTribute DISTMethod SERVERWLM OPTLOCAL 0 10.30.30.34
   PORT 28538 28518 28510 28502 28503 DESTIP 10.30.20.100 10.30.20.101
ENDVIPADynamic
```

Verification

Because nothing differs from Figure 7-15 on page 256, we do not repeat the displays for the components in the sysplex here.

OPTLOCAL 1 with Sysplex Distributor only

In the previous test case using OPTLOCAL 0, only local connections should be used for the optimized path between the HTTP servers and the WebSphere Application Server, depending on which HTTP server was selected on SC30 or on SC31.

In this section, we discuss how the high availability path can be achieved automatically, even if the WLM values do not recommend taking the local connection when the value for the local server is zero (0).

Figure 7-13 shows additional paths to WebSphere Application Server server within the sysplex controlled by the Sysplex Distributor. If the local connections cannot be used because the WLM value for the WebSphere Application Server during connection setup indicates 00, then the Sysplex Distributor has to select another remote WebSphere Application Server within the sysplex.

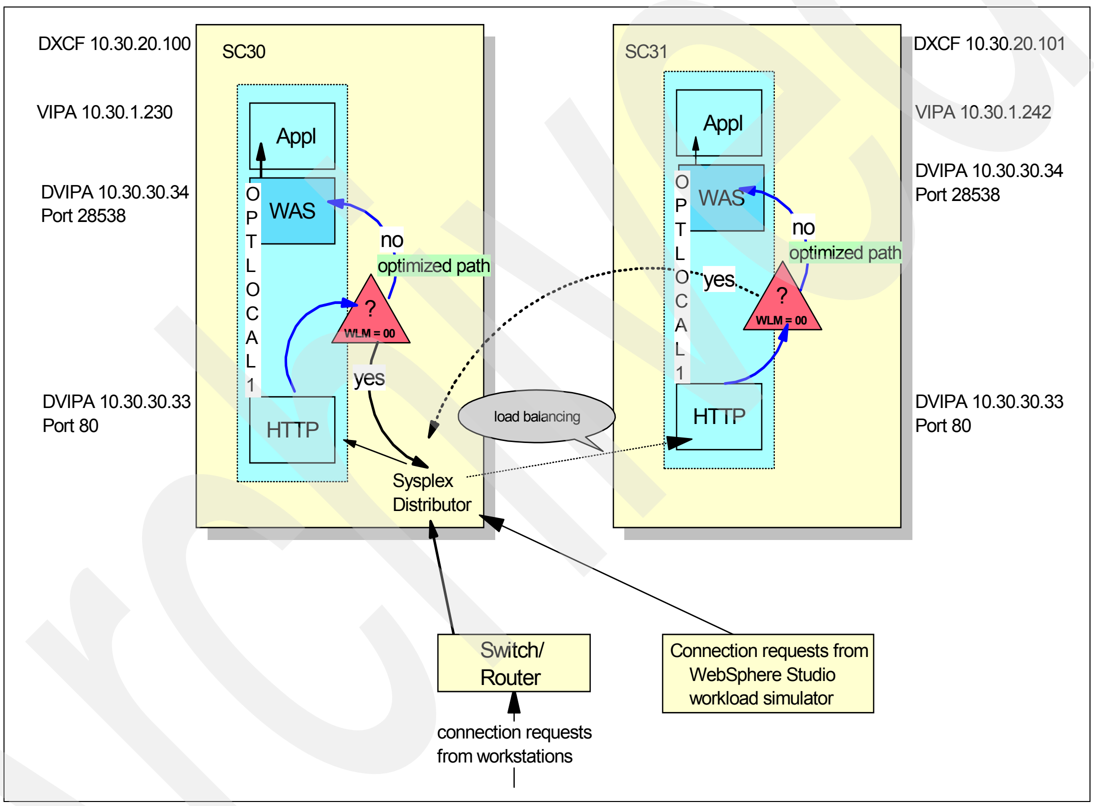


Figure 7-13 Optimized path using OPTLOCAL 1

TCP/IP profile definitions

Example 7-24 shows the OPTLOCAL 1 definitions that we used for this test case.

Example 7-24 TCP/IP profile definitions using OPTLOCAL 1

```
VIPADynamic
VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.33
   VIPADISTribute DISTMethod SERVERWLM 10.30.30.33
   PORT 80 DESTIP 10.30.20.100 10.30.20.101
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.34
   VIPADISTribute DISTMethod SERVERWLM optlocal 1 10.30.30.34
   PORT 28538 28518 28510 28502 28503 DESTIP 10.30.20.100 10.30.20.101
ENDVIPADYNAMIC
```

Verification

We used the NETSTAT VIPADCFG DETAIL command to verify that the OPTLOCAL 1 was active (see Example 7-25).

Example 7-25 Netstat VIPADCFG detail

```
D TCPIP,TCPIPC,N,VIPADCFG,DETAIL

DYNAMIC VIPA INFORMATION:
DEST:        10.30.30.34..28538
    DESTXCF:   10.30.20.100
      SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM OPTLOCAL
      OPTLOC:  1
  DEST:        10.30.30.34..28538
    DESTXCF:   10.30.20.101
      SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM OPTLOCAL
      OPTLOC:  1
```

Example 7-26 shows the start of the WebSphere workload simulator and a check through **netstat vdpt** for the WebSphere Application Server. Notice that both WLM values are equal but that the number of total connections (TOTALCONN) differs slightly.

Example 7-26 Netstat VDPT of WebSphere Application Server using filter ipport

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

 DYNAMIC VIPA DESTINATION PORT TABLE:
 DEST:        10.30.30.34..28538
   DESTXCF:   10.30.20.100
   TOTALCONN: 0000000318  RDY: 001  WLM: 06  TSR: 100
   FLG: SERVERWLM, LOCAL
     TCSR: 100  CER: 100 SEF: 100
     ABNORM: 0000        HEALTH: 100
     ACTCONN:   0000000004
     QOSPLCACT: *DEFAULT*
       W/Q: 06
 DEST:        10.30.30.34..28538
   DESTXCF:   10.30.20.101
   TOTALCONN: 0000000258  RDY: 001  WLM: 06  TSR: 100
   FLG: SERVERWLM, LOCAL
     TCSR: 100  CER: 100 SEF: 100
     ABNORM: 0000        HEALTH: 100
     ACTCONN:   0000000005
     QOSPLCACT: *DEFAULT*
       W/Q: 06
 2 OF 2 RECORDS DISPLAYED
```

After a few minutes, we issued **netstat vdpt** for WebSphere Application Server in order to obtain a second snapshot; see Example 7-27. Notice that the WLM values are no longer equal.

Example 7-27 Netstat VDPT again

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA DESTINATION PORT TABLE:
```

```
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000000428  RDY: 001  WLM: 08  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000006
    QOSPLCACT: *DEFAULT*
      W/Q: 08
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000000359  RDY: 001  WLM: 11  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000008
    QOSPLCACT: *DEFAULT*
      W/Q: 11
```

After additional snapshots, we saw that the WLM values on both systems increased to 16, and that sometimes SC30 (`10.30.20.100`) had better WLM values, and other times SC31 (`10.30.20.101`) had better WLM values.

Takedown of the HTTP server on SC30

Next, we purged the HTTP server address space in SC30 to check whether all connection requests go to HTTP server on SC31.

Example 7-28 shows that the RDY indicator is `000` (**1**), but WLM and other values do not indicate problems. So all connections do go to the HTTP server on SC31. This can be analyzed only through observing the TOTALCONN values in subsequent displays. The TOTALCONN value is frozen on 18880 (**2**) and the active connection count (ACTCONN) remains zero (`0`).

We noticed that the values for WLM, TSR, and so on remained the same, because no new calculations are done for a not ready server.

Example 7-28 After purging HTTP server address space

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.33+80

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.33..80
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000018880 2 RDY: 000 1 WLM: 13  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 13
DEST:        10.30.30.33..80
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000017030 2 RDY: 001  WLM: 08  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
```

```
    ACTCONN:   0000000009
    QOSPLCACT: *DEFAULT*
      W/Q: 08
2 OF 2 RECORDS DISPLAYED
```

Subsequent displays show the frozen TOTALCONN value of 18880 (1) in SC30 and that this value increased in SC31 from 17030 (2) to 21171 (3), and only the SC31 display indicates that there are active connections (ACTCONN); see Example 7-29.

Example 7-29 Frozen TOTALCONN in SC30

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.33+80

 DYNAMIC VIPA DESTINATION PORT TABLE:
 DEST:        10.30.30.33..80
   DESTXCF:   10.30.20.100
   TOTALCONN: 0000018880 1 RDY: 000  WLM: 13  TSR: 100
   FLG: SERVERWLM
     TCSR: 100  CER: 100 SEF: 100
     ABNORM: 0000        HEALTH: 100
     ACTCONN:   0000000000
     QOSPLCACT: *DEFAULT*
       W/Q: 13
 DEST:        10.30.30.33..80
   DESTXCF:   10.30.20.101
   TOTALCONN: 0000021171 3 RDY: 001  WLM: 06  TSR: 100
   FLG: SERVERWLM
     TCSR: 100  CER: 100 SEF: 100
     ABNORM: 0000        HEALTH: 100
     ACTCONN:   0000000010
     QOSPLCACT: *DEFAULT*
       W/Q: 06
 2 OF 2 RECORDS DISPLAYED
```

After a restart of the HTTP server on SC30, TOTALCONN (1) increased again from 18880 to 19772; see Example 7-30.

Example 7-30 Restart of HTTP server in system SC30

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.33+80

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.33..80
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000019772 1 RDY: 001 WLM: 12  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 12
DEST:        10.30.30.33..80
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000024253  RDY: 001  WLM: 13  TSR: 100
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 100
```

```
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000011
    QOSPLCACT: *DEFAULT*
      W/Q: 13
2 OF 2 RECORDS DISPLAYED
```

During takedown of the HTTP server on SC30, the Sysplex Distributor sent all connection requests to the HTTP server on SC31; see Example 7-31.

Example 7-31   Displays for the WebSphere Application Server during takedown phase of HTTP server on SC30

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000001990 1 RDY: 001  WLM: 07  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 07
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000002587  RDY: 001  WLM: 06  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000009
    QOSPLCACT: *DEFAULT*
      W/Q: 06
2 OF 2 RECORDS DISPLAYED
```

Because the optimized local path of the local connections to the WebSphere Application Server takes effect, the TOTALCONN 1990 (1) value of the WebSphere Application Server display sequences on SC30 shows that it is frozen. Only the value for SC31 increased; see Example 7-32.

Example 7-32   Second display in a sequence for the WebSphere Application Server

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

 DYNAMIC VIPA DESTINATION PORT TABLE:
 DEST:        10.30.30.34..28538
   DESTXCF:   10.30.20.100
   TOTALCONN: 0000001990 1 RDY: 001  WLM: 08  TSR: 100
   FLG: SERVERWLM, LOCAL
     TCSR: 100  CER: 100 SEF: 100
     ABNORM: 0000        HEALTH: 100
     ACTCONN:   0000000000
     QOSPLCACT: *DEFAULT*
       W/Q: 08
 DEST:        10.30.30.34..28538
   DESTXCF:   10.30.20.101
```

```
   TOTALCONN: 0000002692  RDY: 001  WLM: 06  TSR: 100
   FLG: SERVERWLM, LOCAL
     TCSR: 100  CER: 100 SEF: 100
     ABNORM: 0000        HEALTH: 100
     ACTCONN:   0000000010
     QOSPLCACT: *DEFAULT*
       W/Q: 06
 2 OF 2 RECORDS DISPLAYED
```

During takedown of the HTTP server on SC30, the optimized path with the local path between the HTTP server and the WebSphere Application Server instance on SC31 is used. No Sysplex Distributor was involved in the connection setup path between the HTTP server and WebSphere Application Server on SC31. This would only be necessary if the WLM value on SC31 would fall to the value of 00.

After a restart of the HTTP on SC30, the TOTALCONN increases; see Example 7-33.

Example 7-33 Restart HTTP server on SC30 - from WebSphere Application Server point of view

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000002028  RDY: 001  WLM: 07  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000003
    QOSPLCACT: *DEFAULT*
      W/Q: 07
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000003070  RDY: 001  WLM: 07  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000        HEALTH: 100
    ACTCONN:   0000000009
    QOSPLCACT: *DEFAULT*
      W/Q: 07
2 OF 2 RECORDS DISPLAYED
```

Overloaded WebSphere Application Server server on SC30

We drove the WebSphere Workload Simulator on SC30 with the WebSphere Application Server weight set to WLM = 000 to test whether the Sysplex Distributor would get control.

As shown in Example 7-34, the WLM value is 00 (1), and TSR (2) and SEF(3) values show 000 for the WebSphere Application Server instance.

Example 7-34 Overloaded WebSphere Application Server server on SC30

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.100
```

```
  TOTALCONN: 0000003491  RDY: 001 4 WLM: 00 1 TSR: 000 2
  FLG: SERVERWLM
    TCSR: 100 5 CER: 100 6 SEF: 000 3
    ABNORM: 0000 8 HEALTH: 100 7
    ACTCONN:   0000000022
    QOSPLCACT: *DEFAULT*
      W/Q: 00
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000004646  RDY: 001  WLM: 06  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000014
    QOSPLCACT: *DEFAULT*
      W/Q: 06
2 OF 2 RECORDS DISPLAYED
```

The server efficiency fraction (SEF) 3 indicated that the WebSphere Application Server was unable to receive connection requests, because the application server queue is overloaded accepting new connection requests. The SEF value of 000 influences also caused the target server responsiveness (TSR) 2 value of zero (0). If the SEF value is under 075, all connection requests will be sent to the Sysplex Distributor to make the routing decision.

Other indicators (such as RDY, TSCR, CER, HEALTH, and abnorm) do not point to any problems.

- RDY = 001 (4) means the server (only one) is still running.
- TCSR (target connectivity success rate) = 100 (5) means that all connection requests sent by the distribution stack are received by the target stack. A low value could indicate a problem with the connection between the distribution and target stack.
- CER (connection establishment rate) = 100 (6) means that all connections can be established between the HTTP proxy server (in this case, working as a client) and the WebSphere Application Server built-in HTTP server. The problems in our case arose in the next step, when the built-in HTTP WebSphere Application Server did not accept the new connections.
- HEALTH (7) is an indicator for the specific server. The field affects the WLM weight and operation of the OPTLOCAL distributor setting. If this value is lower than 75, all connection requests will go to the Sysplex Distributor.
- abnorm (8) counts all abnormal ended transactions.

  For example, a CICS server might work without any problems, but some transactions are in trouble because of very slow database access. Because all indicators for the CICS server show values with a favorable state, the Sysplex Distributor would continue sending new connection requests to the CICS server. Now the abnorm count is also included in the WLM calculation. If the abnorm count for the local stack is greater than 250, then no OPTLOCAL function for transactions is executed. All connection requests would be directed to the Sysplex Distributor.

In the case of the overloaded WebSphere Application Server on SC30, the result of this situation is that the HTTP server in SC30 loses the local connection (the WebSphere Application Server has a WLM weight of zero). The Sysplex Distributor now has to search for another appropriate WebSphere Application Server in the sysplex.

In our case, it is the WebSphere Application Server on SC31. New client HTTP requests, which initiate TCP connection requests, are sent to the Sysplex Distributor and forwarded to the HTTP server with the best WLM value. If the request reaches the HTTP server on SC30, this server will not be able to use the optimized setup path, because the stack signals WLM = 00 for WebSphere Application Server on SC30.

Therefore, the HTTP server sends its connection request to the Sysplex Distributor. The Sysplex Distributor will use the normal distribution path to the WebSphere Application Server instance on SC31 through HiperSockets within the server or using defined VIPAROUTE in the case of a server-to-server path—or, if these paths are not available, through XCF link.

Figure 7-13 on page 241 shows the situation if SERVER WLM = 00 occurs for the WebSphere Application Server on SC30. The figure also shows the same situation if the WebSphere Application Server on SC31 might fall into a SERVER WLM value of 00. In this case, the HTTP server on SC31 has to send the connection request to the Sysplex Distributor in SC30, which will forward the request to the WebSphere Application Server on SC30.

Returning to our first case, as illustrated in Example 7-35, the next display in this sequence shows that the TOTALCONN of `3494` (1) is frozen for WebSphere Application Server on SC30, although the `RDY: 001` (2) shows that the server is still available. The WLM (3), the TSR (4), and the SEF (5) show the actual status through indication of 000 for all measurements.

We noticed that the ACTCONN (6) value for the WebSphere Application Server on SC30 was not zero compared with the previous display. The 17 HTTP requests were still kept, because active connections will continue to be displayed until the HTTP timeout is reached for these connections. The TOTALCONN (7) for the HTTP server on SC31 increased from 4646 to `5006`.

Example 7-35 Second display with frozen connections

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000003494 1 RDY: 001 2 WLM: 00 3 TSR: 000 4
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 000 5
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000017 6
    QOSPLCACT: *DEFAULT*
      W/Q: 00
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000005006 7 RDY: 001  WLM: 06  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000014
    QOSPLCACT: *DEFAULT*
      W/Q: 06
2 OF 2 RECORDS DISPLAYED
```

OPTLOCAL 4 with Sysplex Distributor only

In previous sections, we discussed OPTLOCAL 0 and 1. The values 2 through 16 can also be used. They are applied to the amount of preference for the local server based on a comparison of WLM weights.

To give the local stack a higher preference even if it does not have the best WLM value, use an OPTLOCAL value of 2 to 16 as a multiplier to manipulate the WLM calculation. This allows you to prefer the local server even if the server WLM value for the local stack is much lower than others of the same servers in the server group.

For example, server WLM = 05 was defined on SC30. On the other LPARs, server WLM values were 12, 14, or 15. If OPTLOCAL 4 was defined, then the local WLM value is multiplied by the defined OPTLOCAL value. If OPTLOCAL 4 is defined on SC30, then the local WLM value is multiplied by the defined OPTLOCAL value. This means 20 (5 X4) will be compared with all servers' WLM values of the same server group. Thus, the local server (SC30) will be favored because it has the highest value.

TCP/IP profile definitions

Only OPTLOCAL 4 was defined on the VIPADISTRIBUTE statement, as shown in Example 7-36.

Example 7-36 TCP/IP profile for OPTLOCAL 4

```
VIPADynamic
VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.33
   VIPADISTribute DISTMethod SERVERWLM 10.30.30.33
   PORT 80 DESTIP 10.30.20.100 10.30.20.101
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.34
   VIPADISTribute DISTMethod SERVERWLM optlocal 4 10.30.30.34
   PORT 28538 28518 28510 28502 28503 DESTIP 10.30.20.100 10.30.20.101
ENDVIPADYNAMIC
```

Verification

Use the **netstat vipadcfg** command to show the correct OPTLOCAL 4 definition for the distributed DVIPA on SC30 and SC31 (as shown in Example 7-37):

- SC30 with DXCF IP address 10.30.20.100
- SC31 with DXCF IP address 10.30.20.101

Example 7-37 TCP/IP profile for OPTLOCAL 4

```
D TCPIP,TCPIPC,N,VIPADCFG,DETAIL

DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.30.30.33/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
    IPADDR/PREFIXLEN: 10.30.30.34/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
DEST:        10.30.30.33..80
  DESTXCF:   10.30.20.100
    SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
    OPTLOC:  NO
DEST:        10.30.30.33..80
  DESTXCF:   10.30.20.101
    SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
```

```
    OPTLOC:  NO
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.100
    SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM OPTLOCAL
    OPTLOC:  4
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.101
    SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM OPTLOCAL
        OPTLOC:  4
```

The **netstat vipadyn** command displays the date and time when the DVIPAs are activated, as shown in Example 7-38.

Example 7-38 Date and time of DVPA activation

```
D TCPIP,TCPIPC,N,VIPADYN

DYNAMIC VIPA:
  IPADDR/PREFIXLEN: 10.30.30.32/24
    STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
    ACTTIME: 08/23/2006 17:06:01
  IPADDR/PREFIXLEN: 10.30.30.33/24
    STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
    ACTTIME: 08/23/2006 17:06:01
  IPADDR/PREFIXLEN: 10.30.30.34/24
    STATUS:  ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT: DIST/DEST
    ACTTIME: 08/23/2006 17:06:01
  IPADDR/PREFIXLEN: 10.30.80.10/24
    STATUS:  ACTIVE     ORIGIN: VIPARANGE BIND   DISTSTAT:
    ACTTIME: 08/23/2006 17:10:52                 JOBNAME: LBADV
VIPA ROUTE:
  DESTXCF: 10.30.20.100
    TARGETIP: 10.30.1.230
    RTSTATUS: DEFINED
  DESTXCF: 10.30.20.101
    TARGETIP: 10.30.1.242
    RTSTATUS: ACTIVE
  DESTXCF: 10.30.20.102
    TARGETIP: 10.30.1.221
    RTSTATUS: ACTIVE
7 OF 7 RECORDS DISPLAYED
```

Example 7-39 shows the first display of the VIPA distribution port table after a restart of SC30 with OPTLOCAL option 4. Both WebSphere Application Servers received equal numbers of connections. The load distribution is equal.

Example 7-39 First display of VIPA distribution port table with OPTLOCA 4

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000000038  RDY: 001  WLM: 08  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
```

```
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000005
    QOSPLCACT: *DEFAULT*
      W/Q: 08
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000000038  RDY: 001  WLM: 08  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000007
    QOSPLCACT: *DEFAULT*
      W/Q: 08
```

The VIPA connection routing table for the two WebSphere Application Servers displays the following distributions, as shown in Example 7-40:

- Six connections were distributed from the SOURCEVIPA 10.30.1.230, which used the HTTP server on SC30 to the WebSphere Application Server on SC30 with DXCF IP address 10.30.20.100 using cluster IP address 10.30.30.34 with a port address of 28538.
- Nine connections were distributed from the SOURCEVIPA 10.30.1.242, which used the HTTP server on SC31 to the WebSphere Application Server on SC31 with DXCF IP address 10.30.20.101, using cluster address 10.30.30.34 with a port address of 28538.

Example 7-40 Netstat VCRT of first connections to WebSphere Application Server

```
D TCPIP,TCPIPC,N,VCRT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA CONNECTION ROUTING TABLE:
.......... first two connections from HTTP server on SC30 to WAS on SC30 .........
DEST:       10.30.30.34..28538
  SOURCE:   10.30.1.230..1132
  DESTXCF:  10.30.20.100
    POLICYRULE:     *NONE*
    POLICYACTION:   *NONE*
DEST:       10.30.30.34..28538
  SOURCE:   10.30.1.230..1133
  DESTXCF:  10.30.20.100
    POLICYRULE:     *NONE*
    POLICYACTION:   *NONE*
.......... first two connections from HTTP server on SC31 to WAS on SC31 .........
DEST:       10.30.30.34..28538
  SOURCE:   10.30.1.242..1172
  DESTXCF:  10.30.20.101
    POLICYRULE:     *NONE*
    POLICYACTION:   *NONE*
DEST:       10.30.30.34..28538
  SOURCE:   10.30.1.242..1173
  DESTXCF:  10.30.20.101
    POLICYRULE:     *NONE*
    POLICYACTION:   *NONE*
```

Stopping WebSphere Application Server on SC30

During the test with the workload simulator, we could not increase the load of one system to discover a recalculation of a fictive WLM value. Therefore, we decided to stop the WebSphere Application Server application on SC30 and check whether the distribution worked with the OPTLOCAL 4 option, as shown in Example 7-41.

When the WebSphere Application Server on SC30 was purged, the RDY = 001, the WLM = 0, and the TSR and SEF were 000, as expected. TOTALCONN was frozen with 1328 connections for SC30. TOTALCONN: 1448 on SC31 will increase in the next display.

Example 7-41 WebSphere Application Server on SC30 was stopped, service is still running

```
D TCPIP,TCPIPC,N,VDPT,DETAIL,IPPORT=10.30.30.34+28538

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000001328  RDY: 001  WLM: 00  TSR: 000
  FLG: SERVERWLM
    TCSR: 100  CER: 100 SEF: 000
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000024
    QOSPLCACT: *DEFAULT*
      W/Q: 00
DEST:        10.30.30.34..28538
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000001448  RDY: 001  WLM: 06  TSR: 100
  FLG: SERVERWLM, LOCAL
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000012
    QOSPLCACT: *DEFAULT*
      W/Q: 06
2 OF 2 RECORDS DISPLAYED
```

Although the WebSphere Application Server on SC30 was now down, both HTTP servers distributed to the SC31 WebSphere Application Server, as shown in Example 7-42.

Example 7-42 WebSphere Application Server on SC30 is down, HTTP servers distribute to WebSphere Application Server on SC31

```
D TCPIP,TCPIPC,N,VCRT,DETAIL,IPPORT=10.30.30.33+80

 DYNAMIC VIPA CONNECTION ROUTING TABLE
.............. first two connections from workload simulator to HTTP server on
SC30 ......
DEST:      10.30.30.33..80
   SOURCE:  10.30.2.50..2084
   DESTXCF: 10.30.20.100
     POLICYRULE:    *NONE*
     POLICYACTION:  *NONE*
 DEST:      10.30.30.33..80
   SOURCE:  10.30.2.50..2232
   DESTXCF: 10.30.20.100
............. first two connections from workload simulator to HTTP server on SC31
......
```

```
DEST:      10.30.30.33..80
  SOURCE:  10.30.2.50..2230
  DESTXCF: 10.30.20.101
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
DEST:      10.30.30.33..80
  SOURCE:  10.30.2.50..2231
  DESTXCF: 10.30.20.101
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
```

Example 7-43 shows the relationship between two connections.

The HTTP server WEBBW311 on SC31 (1) established a connection with SOURCEVIPA 10.30.1.242..2665 to the WebSphere Application Server BWSR02B and address 10.30.30.34..28538. This same connection can be viewed with reverse addresses.

Two other connections were established from the WebSphere Application Server on SC30 with address 10.30.1.230 routed to the WebSphere Application Server on SC31 BWSR02B (2). The corresponding partner connection could not be displayed on SC30. They were already closed when the **netstat** command was entered.

Example 7-43 Netstat conn displays HTTP and WebSphere Application Server connections

```
D TCPIP,TCPIPC,N,CONN,IPPORT=10.30.30.34+28538

USER ID  CONN     STATE
BWSR02B  0006539F ESTBLSH
  LOCAL SOCKET:   10.30.30.34..28538 2
  FOREIGN SOCKET: 10.30.1.230..3148
BWSR02B  000653A1 ESTBLSH
  LOCAL SOCKET:   10.30.30.34..28538 2
  FOREIGN SOCKET: 10.30.1.230..3149
.......... further connections .........................................
BWSR02B  0006537F ESTBLSH
  LOCAL SOCKET:   10.30.30.34..28538
  FOREIGN SOCKET: 10.30.1.242..2665 1
.......... further connections .........................................
WEBBW311 0006537E ESTBLSH
   LOCAL SOCKET:   10.30.1.242..2665 1
   FOREIGN SOCKET: 10.30.30.34..28538
```

7.3 WLM reporting abnormal conditions

Workload Manager (WLM) provides system weight and server-specific weight values.

The system weight value is issued for each target LPAR based on:

- Comparison of available capacity
- Comparison of displaceable capacity (lower importance work that can be displaced)

System weights do not reflect how well the server is performing, as explained in 7.3.2, “Calculation of WLM weight and TSR” on page 255.

Server-specific weights are issued for each application server running in an LPAR. They are based on:

- How well each server is meeting the goals for its service class
- Comparison of displaceable capacity on each system based on the importance of the server's work.

The following sections explain the situations and conditions that impact the complexity of workload decisions, and how they are resolved in a design for the Sysplex Distributor environment.

7.3.1 Situation of current workload distribution decisions

WLM is not aware of all problems experienced by load balancing targets; this might occur, for example, when a server application needs a resource such as a database, but the resource is unavailable. Another example is a server application that acts as a transaction router for other back-end applications on other systems, but the path to the back-end application is unavailable.

In each of these scenarios, the server appears to be completing the transactions quickly (using little CPU capacity), although it is actually failing. This situation is known as a *storm drain problem*, with the following results:

- The server is favored by WLM, because it is using very little CPU capacity.
- As workloads increase, the server is favored more and more over other servers.
- All this work goes "down the drain."

Although the Sysplex Distributor is able to reduce the WLM weight values by using the Target Server Responsiveness fraction (TSR), the storm drain problem cannot be fully resolved.

> **Note:** Implementing Sysplex Distribution with multitier applications as described in 7.2, "Optimized multitier z/OS Sysplex Distributor load balancing" on page 217 can mitigate some of the effects of storm drain.

The TSR fraction is built based on the following factors:

- Is the connectivity between the distributing stack and the target stack stable? Are new connections being received by the target stack?

  This factor is the Target Connectivity Success Rate (TCSR).

- Is the network connectivity between server and client in a good state? Can new connections be established?

  This factor is the Connection Establishment Rate (CER).

- Is the target server application accepting new work? What target server responsiveness does the application show? Is the backlog queue overloaded?

  This factor is the Server Efficiency Fraction (SEF).

Figure 7-14 on page 255 illustrates these factors.

7.3.2 Calculation of WLM weight and TSR

The calculation of WLM weights and the TSR determines which target application server gets the work requests, such as the TCP connection request forwarded by the Sysplex Distributor or the external load balancer, as explained here:

- WLM weights

  When determining a system weight, WLM assigns a relative weight to each system in the sysplex, with the highest weight going to the system with the most available CPU capacity. The weights range between 0 & 64.

  If all systems in the sysplex are running at or near 100% utilization, WLM will assign the highest weights to the systems with the largest amounts of lower importance work. In this way, new connection requests will be distributed to the systems with the highest displaceable capacity.

  However, note the following points:

  – This method does not reflect how well the server application is actually meeting the goals of its service class.

  – If all systems are using close to 100% of capacity, then the WLM weight is based on a comparison of displaceable capacity, which is the amount of lower importance work on each system. However, if the service class of the server is of low importance, then it might not be able to displace this work.

  When determining a server-specific weight, WLM assigns a relative weight to each server based on how well each server is meeting the goals of its service class. The weights range between 0 - 64. If all systems in the sysplex are running at or near 100% utilization, WLM will assign the highest weights to the servers running on systems with the largest amounts of work that can be displaced by that server (based on the importance of its service class)

- Calculation of TSR

  – SEF: This value is based on whether the server is processing new connections:

    • New connections are being established—Connection Establishment Rate (CER).
    • The server is accepting the new connections.

  – TCSR: The distributor determines this value from the number SYNs it has sent to the target and the statistics returned from the target.

  – TSR: This is based on the SEF value (which includes CER) and TCSR values.

Figure 7-14 shows that distributed DVIPA1 for port 8000 is defined with DISTMETHOD SERVERWLM to DESTIP target 1 and target 2.

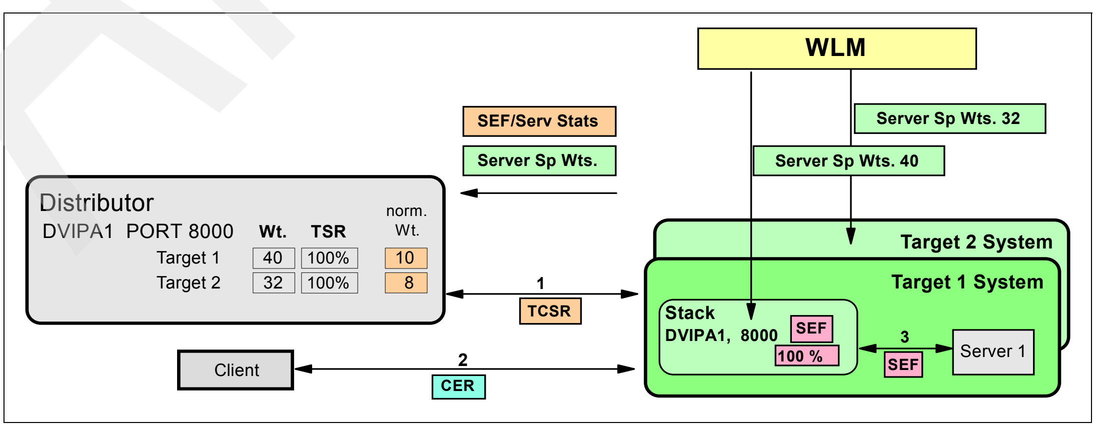


Figure 7-14 Workload distribution weight fractions

The Sysplex Distributor receives a calculated weight value of 40 for Target 1, and 32 for Target 2. Because the TSR for both targets is reported 100%, both weight values are not minimized. At a lower percentage the calculation would be: Weight * TSR% = Weight fraction. The next step is normalizing the weight. This is done by dividing the Weight fraction with the integer 4. This is done only to continue working with smaller values.

- For Target 1, the calculation is: normalized weight is 10 = (40 / 4).
- For Target 2, the calculation is: normalized weight is 8 = (32 / 4).

7.3.3 WLM interface for abnormal transactions and health status

WLM provides an interface that allows a server to pass additional information about its overall health. This overall health information consists of two values:

- Abnormal transaction completion rate

  Applications such as CICS transaction server for z/OS, that act as subsystem work managers, can report an abnormal transaction completion rate to WLM. The value is between 0 and 1000, with 0 meaning no abnormal completions.

- General health of the application

  Applications can report their general health to WLM. The value is between 0 and 100, with 100 meaning that a server has no general health problems (100% healthy).

WLM will reduce the reported weight based on Abnormal Completion Rate and the General Health.

Figure 7-15 illustrates how the abnormal transaction completion rate of 1000 impacts the server-specific weight value, and also how the Sysplex Distributor uses the normalized value of zero (0) instead the previous displayed value of 10 for Target 1.

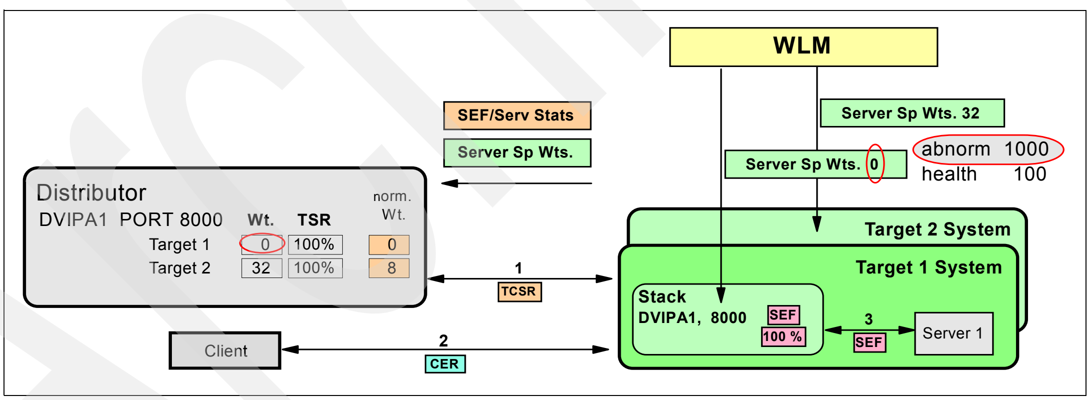


Figure 7-15 Workload distribution with additional weight fractions

Server scenarios

The TCP/IP stack retrieves the information about abnormal transaction termination and health of the application using the IWM4SRSC interface.

- Abnormal termination

  This information solves the problem where the registered server is not the transaction consumer. It does not, however, resolve the problem where another connector is between the TCP/IP stack and the consumer.

- HEALTH

  Address spaces which are not instrumented can use the IWM4HLTH interface to WLM to set a health status, which is also returned by IWM4SRSC.

Figure 7-16 depicts the differences between the information TCP/IP receives through these two interfaces.

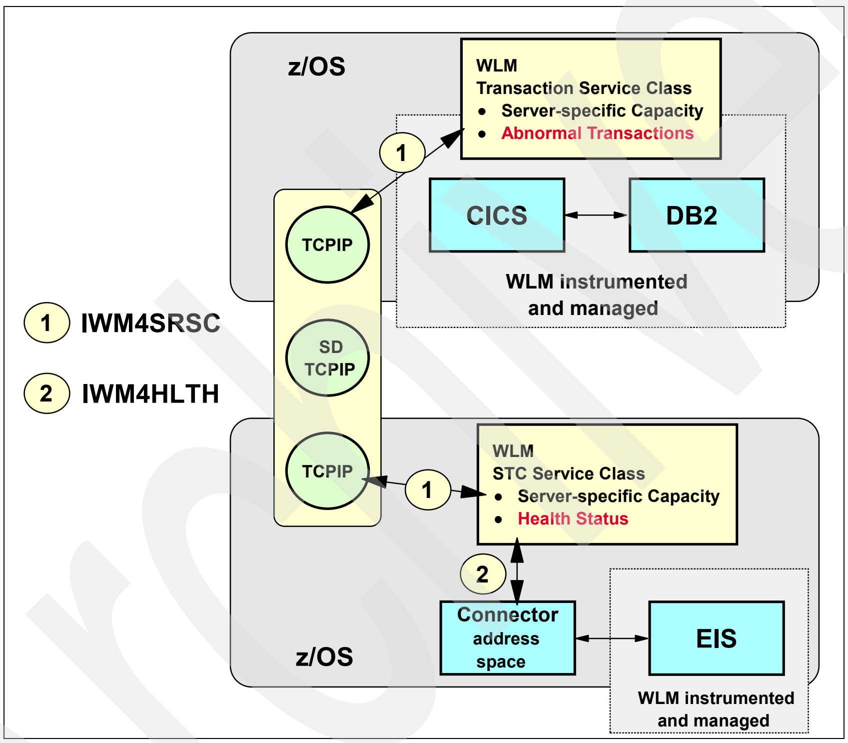


Figure 7-16 WLM target application awareness

Verification

The **netstat** commands in Example 7-44 show samples for the health indicator, active connections, and abnormal termination.

Example 7-44 The netstat command for VIPA distribution port table, short format

```
D TCPIP,TCPIPC,N,VDPT

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:         10.30.30.32..23
  DESTXCF:    10.30.20.102
  TOTALCONN: 0000000000  RDY: 001  WLM: 15  TSR: 100
  FLG: BASEWLM
DEST:         10.30.30.33..80
  DESTXCF:    10.30.20.100
```

```
  TOTALCONN: 0000000000  RDY: 000  WLM: 00  TSR: 100
  FLG: SERVERWLM
```

If you use the **netstat** command with format detail, you will get the desired information about WLM, TSR, TCSR, CER, SEF, ABNORM, and HEALTH values. you will also see the active connections for the distributed server; see Example 7-45.

Example 7-45 netstat command for VIPA distribution port table, long format

```
D TCPIP,TCPIPC,N,VDPT,DETAIL

DYNAMIC VIPA DESTINATION PORT TABLE:
DEST:        10.30.30.32..23
  DESTXCF:   10.30.20.100
  TOTALCONN: 0000000000  RDY: 001  WLM: 05  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 05
DEST:        10.30.30.32..23
  DESTXCF:   10.30.20.101
  TOTALCONN: 0000000000  RDY: 001  WLM: 04  TSR: 100
  FLG: BASEWLM
    TCSR: 100  CER: 100 SEF: 100
    ABNORM: 0000       HEALTH: 100
    ACTCONN:   0000000000
    QOSPLCACT: *DEFAULT*
      W/Q: 04
```

When issuing the **netstat all** command, you will get information indicating how many connections were dropped, if the backlog is overloaded (current backlog and max. backlog).

This is defined through the SMAXCONN statement in the TCP/IP profile (the default is 10); see Example 7-46.

Example 7-46 The netstat all command

```
===> netstat all

Client Name: BWSR02B                  Client Id: 000013AB
  Local Socket: ::..28530
  Foreign Socket: ::..0
    BytesIn:            00000000000000000000
    BytesOut:           00000000000000000000
    SegmentsIn:         00000000000000000000
    SegmentsOut:        00000000000000000000
    Last Touched:       16:03:39         State:              Listen
    RcvNxt:             0000000000       SndNxt:             0000000000
    ClientRcvNxt:       0000000000       ClientSndNxt:       0000000000
    InitRcvSeqNum:      0000000000       InitSndSeqNum:      0000000000
    CongestionWindow:   0000000000       SlowStartThreshold: 0000000000
    IncomingWindowNum:  0000000000       OutgoingWindowNum:  0000000000
    SndWl1:             0000000000       SndWl2:             0000000000
    SndWnd:             0000000000       MaxSndWnd:          0000000000
```

```
  SndUna:              0000000000        rtt_seq:            0000000000
  MaximumSegmentSize: 0000000536        DSField:            00
  Round-trip information:
  Smooth trip time: 0.000               SmoothTripVariance: 1500.000
  ReXmt:              0000000000        ReXmtCount:         0000000000
  DupACKs:            0000000000
  SockOpt:            8000              TcpTimer:           00
  TcpSig:             01                TcpSel:             20
  TcpDet:             C0                TcpPol:             08
  QOSPolicyRuleName:
  ReceiveBufferSize:  0000016384        SendBufferSize:     0000016384
  ConnectionsIn:      0000000000        ConnectionsDropped: 0000000000
  CurrentBacklog:     0000000000        MaximumBacklog:     0000000010
  CurrentConnections: 0000000000        SEF:                100
 Quiesced: No
```

You can check the distribution method and the OPTLOCAL value using **netstat VIPDCFG**, as shown in Example 7-47.

Example 7-47 netstat vipadcfg short format

```
D TCPIP,TCPIPC,N,VIPADCFG

DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.30.30.33/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
    IPADDR/PREFIXLEN: 10.30.30.34/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
  VIPA DISTRIBUTE:
    DEST:        10.30.30.33..80
      DESTXCF:   10.30.20.100
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
    DEST:        10.30.30.33..80
      DESTXCF:   10.30.20.101
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM
```

The **netstat vipadcfg** command shows the defined OPTLOCAL value; see Example 7-48.

Example 7-48 netstat vipdcfg long format

```
D TCPIP,TCPIPC,N,VIPADCFG,DETAIL

DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.30.30.33/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
    IPADDR/PREFIXLEN: 10.30.30.34/24
      MOVEABLE: IMMEDIATE  SRVMGR: NO
     DEST:        10.30.30.34..28538
       DESTXCF:   10.30.20.100
         SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM OPTLOCAL
         OPTLOC:  4
     DEST:        10.30.30.34..28538
       DESTXCF:   10.30.20.101
         SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM OPTLOCAL
```

```
OPTLOC:  4
```

You can also obtain ABNORM and HEALTH information through the LBAdvisor to check if the WLM weight is being influenced by these factors; see Example 7-49.

Example 7-49   F LBADV, DISP

```
F LBADV,DISP,LB,I=0
EZD1243I LOAD BALANCER DETAILS
LB INDEX : 00 UUID : 637FFF175C
...
GROUP NAME : CICS_SERVER
GROUP FLAGS : SERVERWLM
IPADDR..PORT: 201.2.10.11..8000
SYSTEM NAME: MVS209 PROTOCOL : TCP AVAIL : YES
WLM WEIGHT : 00000 CS WEIGHT : 100 NET WEIGHT: 00001
ABNORM : 01000 HEALTH : 100
FLAGS :
```

8

Performance and tuning

A system delivering poor response times to a user can be perceived as unavailable from the user's viewpoint. TCP/IP performance is influenced by a number of parameters that can be tailored for the specific operating environment. This includes not only TCP/IP stack performance that benefits all applications using the stack, but also applications that are part of the z/OS Communications Server shipment, such as TN3270 and FTP.

Because every TCP/IP environment is different, optimum performance can be achieved only when the system is tuned to match its specific environment. In this chapter, we highlight the most important tuning parameters, and also suggest parameter values that have maximized performance in many client installations.

This chapter discusses the following topics.

Section	**Topic**
8.1, "General performance considerations" on page 262	Factors influencing performance
8.2, "TCP/IP configuration files" on page 265	Significant configuration files for z/OS TCP/IP
8.3, "z/OS UNIX System Services tuning" on page 272	Useful tuning information
8.4, "Storage requirements" on page 272	Storage usage summaries for typical applications
8.5, "Application performance and capacity" on page 278	Telnet (TN3270) capacity planning, FTP tuning, and FTP capacity planning information
8.6, "z/OS Communications Server TCP/IP performance highlights" on page 281	z/OS Communications Server TCP/IP performance enhancements (all automatically enabled)
8.7, "TCP/IP performance quick checklist" on page 284	Performance items to consider
8.8, "IBM Health Checker" on page 285	Health Checker checks information

8.1 General performance considerations

When discussing performance and response times, keep in mind that performance can be influenced by many factors.

- Application programs

  Performance is influenced by obvious elements, such as application path lengths, large memory moves, and I/O operations. I/O response is subject to considerable tuning in a variety of ways. Furthermore, the structure of the application itself must be considered, for example, the choice of single-thread or multiple thread design is important.

  An application design that permits multiple instances of itself to be active has obvious advantages. This chapter concentrates on network aspects of performance; application design is largely outside the scope of the current discussion.

- TCP window size

  The TCP *window* governs the amount of data that the sender can transmit to the receiver before an acknowledgement is returned. A larger window can optimize normal traffic, but has higher overhead if frame retransmission is needed.

  The size of the TCP window is adjustable. The maximum standard window size is 65 535 bytes. Optionally, RFC 1323 (window scaling) can be employed. The maximum window with RFC 1323 is 1 073 725 440 bytes.

  With z/OS, the maximum allowable window and the use of window scaling is determined by the TCP send and receive buffer settings. Window scaling is automatically turned on when the TCP send buffer is set above 64 KB.

- HiperSockets MTU size

  HiperSockets TCP/IP devices are configured similar to OSA-Express QDIO devices. Each HiperSockets requires the definition of a channel path identifier (CHPID) similar to any other I/O interface.

  Real LANs have a maximum frame size limit defined by their protocol. The maximum frame size for Ethernet is 1492 bytes. For Gigabit Ethernet, there is the jumbo frame option for a maximum frame size of 9000 bytes.

  The maximum frame size for a HiperSockets is assigned when the HiperSockets CHPID is defined. You can select frame sizes of 16 KB, 24 KB, 40 KB, and 64 KB. The selection depends on the data characteristics transported over a HiperSockets, which is also a trade-off between performance and storage allocation. The MTU size used by the TCP/IP stack for the HiperSockets interface is also determined by the maximum frame size. Table 8-1 lists maximum frame size and MTU size.

Table 8-1 Maximum frame size and MTU size

Maximum frame size	Maximum Transmission Unit (MTU) size
16 KB	8 KB
24 KB	16 KB
40 KB	32 KB
64 KB	56 KB

> **Note:** The default maximum frame size is 16 KB.

- Frame size/MTU size

  Data is transported across the Ethernet in *frames*. Frame size is limited by the type of Ethernet architecture in use, by the equipment in use, and by the settings of the system.

  The Ethernet type determines the maximum payload size, and thus the frame size. For Ethernet-DIX the maximum payload is 1500 bytes. For Ethernet-IEEE 802.3 LAN, the maximum is 1492 bytes (which results in 1500 bytes after including additional headers). In practice, most Ethernet usage involves the DIX format.

  Note: To avoid incompatibilities between DIX and 802.3 ports, keep the MTU size at or below 1492 bytes for standard frames.

  With z/OS, the MTU is set in the TCP/IP profile. Some vendor equipment allows frames to exceed the maximum allowed by the standards cited here. A frame that is larger than the standard is a *jumbo* frame. When operating in QDIO mode, OSA-Express and OSA-Express2 support jumbo frames. The largest jumbo supported holds an MTU of 8992 bytes (9000 bytes for IPv6).

  Use the PMTU option on the **ping** command to determine where fragmentation is necessary in the network. The **PMTU YES** option differs from the **PMTU IGNORE** option in the way the ping completes its echo function. Refer to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781, for a detailed discussion about the PMTU option.

- OSA-Express, OSA-Express2, OSA-Express3 features

  The OSA-Express, OSA-Express2, and OSA-Express3 features contain hardware and firmware that enable the flow of traffic between the Ethernet link and host memory when operating in QDIO mode. Traffic flowing through the OSA is stored in OSA-memory before being forwarded to the host or LAN.

  This design optimizes host memory access. The OSA microprocessor manages the flow of traffic through the OSA. A utilization of over 90% might indicate that the OSA is nearing capacity. In some cases OSA becomes more efficient as traffic increases, even at high utilizations. This is because OSA can transfer more packets per block when moving data between host memory and OSA memory.

  Note: At high loads, response time might be more of a concern than utilization, especially when running interactive traffic.

- Dynamic LAN idle

  Dynamic LAN idle is designed to reduce latency and improve network performance by dynamically adjusting the inbound blocking algorithm. It is exclusive to z9 EC and z9 BC and applicable to the OSA-Express2 QDIO features and higher (CHPID type OSD).

  When enabled, the TCP/IP stack will adjust the inbound blocking algorithm to best match the application requirements. The TCP/IP stack will dynamically determine the best setting for the current running application based on system configuration, inbound workload volume, CPU utilization, and traffic patterns. Refer to "OSA-Express2 LAN idle timer function" on page 269 for configuration examples.

- Nagle algorithm relaxation

  The Nagle Algorithm has been a standard component of TCP send-side flow control since the mid-1990s. Its intent is to preserve bandwidth by limiting the number of small data segments that can be sent-and-unacknowledged on the TCP connection. If an application performs a Socket Send of a small number of bytes and there is already unacknowledged data outstanding on the connection, the Nagle algorithm disallows this new data from being transmitted immediately. Instead, the new data queues and is pushed out some time in the future, in response to receipt of Acknowledgements for the already-outstanding data. The Nagle algorithm is enabled by default and can be disabled on a per-connection basis using the TCP_NODELAY SetSockOpt.

On the receive side, the concept of Delayed Acknowledgements is in place to both conserve bandwidth and minimize CPU consumption. Most TCP stacks employ an ACK-EVERYOTHER packet scheme. If a workload pattern is *inbound data packet-outbound data packet*, it is common for no immediate, stand-alone ACK to be generated for the inbound data packet. Rather the ACK of the inbound packet is piggy-backed on the next outbound data packet, thereby avoiding one flow on the link.

This approach works well to reduce the number of packets flowing, and because there is some CPU consumption involved in generating and processing a stand-alone ACK, this approach also reduces CPU consumption on both sides of the connection.

However, there is a type of traffic pattern that is exposed to traffic stalls due to the Nagle algorithm. The traffic pattern is not of a symmetric inbound packet-outbound packet variety; rather this traffic pattern has multiple small packets heading in one or both directions.

In this example, the first flow from the client is a control packet that simply primes the server for the next flow. The second flow from the client is the actual request portion of the transaction. The server receives the request, does some processing, and generates a response. So conceptually, the transaction is back-to-back send flows from client to server, with a single send flow from server back to client.

The Nagle algorithm causes 200-millisecond delays between the first control flow from the client and the second actual transaction flow from the client if the server has not disabled DELAYACKS. A stall occurs because this new application employs back-to-back socket sends and does not disable the Nagle algorithm.

With the relaxation of the Nagle algorithm, the TCP/IP stack transparently allows exactly two small packets on a connection to be outstanding. Allowing the second packet to flow results in the remote node (which uses a delayed Acknowledgement scheme) to immediately generate an ACK once the second packet arrives. This avoids the 200-millisecond stalls.

Performance testing in the IBM z/OS Communications Server lab has demonstrated the dramatic effect of having removed Nagle stalls: transaction rate for such applications jumped from three transactions per second to 2650 transactions per second.

- `msg-waitall` deadlock relief

  The `msg_waitall` Socket Read flag is used generally by applications that receive large amounts of data in a burst. This read flag instructs the TCP layer to delay completion of a Socket Receive or Read call until the full length of the requested data is available in the TCP receive buffer. If an application issues this socket call for a large amount of data (for example, 32 KB) but the application is operating with a smaller TCP read buffer (for example, 16 KB), a *deadlock* situation can occur. The buffer fills up but the application does not issue a read because it is still waiting for the full 32 KB of data Deadlock relief in the z/OS TCP/IP stack increases the size of the read buffer transparently in such situations to 32 KB that is expected by the application. In fact, the receive buffer can be expanded up to a size less than or equal to the TCPMAXRCVBufrsize specification on TCPCONFIG statement. The default for TCPMAXRCVBufrsize is 256 KB.

> **Note:** This scenario illustrates the importance of following IBM best practices in establishing the send and receive buffer sizes in the z/OS TCP/IP stack and in heeding the warnings from the z/OS Health Checker about an adequate buffer size.

8.2 TCP/IP configuration files

The following TCP/IP confirmation files are important for z/OS TCP/IP:

- PROFILE.TCPIP

  The PROFILE.TCPIP file contains TCP buffer sizes, LAN device definitions, server ports, home IP addresses, gateway and route statements, VTAM LUs for Telnet use, and so forth. Buffers are dynamically allocated by the Communications Storage Manager (CSM) and are not specified in PROFILE.TCPIP.

- FTP.DATA

  The FTP.DATA file is used by the FTP server and FTP client to establish the initial configuration options. FTP.DATA contains items such as default DCB parameters for new data sets, the checkpoint interval, and so forth.

- TCPIP.DATA

  TCPIP.DATA contains host name, domain origin, and name server definitions.

> **Note:** A recommendation is to keep the statement TRACE RESOLVER commented out to avoid complete tracing of all name server queries. This trace should be used for debugging purposes only.

8.2.1 MTU considerations

The maximum transmission unit (MTU) specifies the largest packet that TCP/IP will transmit over a given interface. Be certain to specify a packet size explicitly instead of using DEFAULTSIZE. The DEFAULTSIZE produces a value of 576 bytes, which is unlikely to be optimal.

Every network consists of many components that must be carefully coordinated with each other. This requirement also applies to finding the parameter setting of the largest possible MTU size in your installation.

Table 8-2 provides an overview of the largest MTU sizes supported by different interfaces.

Table 8-2 Maximum MTU sizes of interfaces by Link type

Link type	Connectivity	Maximum MTU size
Ethernet (DIX)	Ethernet	1500
802.3	Ethernet 802.3	1492
CTC	z/OS using CTC	65527
IPAQENET (OSA QDIO)	OSA-Express Gigabit Ethernet 1000BASE-T	1492 8992 (Jumbo)
IPAQIDIO	HiperSockets	57344

The MTU size used for a given outbound frame depends on several factors:

- `interface_MTU`

  This value is either a hardcoded size that is based on the physical device or a value that is obtained from the device during activation. For an active link or interface, TCP/IP reports the `interface_MTU` in the ActMtu field of the **NETSTAT DEVLINKS -d** command.

- `configured_route_MTU`

  This value is the MTU size configured for a route.

  – For static routes, specify `configured_route_MTU` on either a ROUTE statement in a BEGINROUTES block or on a GATEWAY statement in the TCP/IP profile.

  – For dynamic routes, the `configured_route_MTU` value comes from the value of the MTU keyword specified on the RIP_INTERFACE, OSPF_INTERFACE, or INTERFACE statement for that interface in the OMPROUTE configuration file. If you do not specify an MTU for an interface, OMPROUTE uses 576.

- `actual_route_MTU`

  This value is the minimum of the `interface_MTU` and the `configured_route_MTU`. When path MTU discovery is not in effect, TCP/IP uses the `actual_route_MTU` to send outbound packets.

- `path_MTU`

  This value is the value that is determined by the path MTU discovery function. When path MTU discovery is in effect, TCP/IP uses `path_MTU` to send outbound packets. You can enable path MTU discovery for IPv4 using IPCONFIG PATHMTUDISCOVERY.

  Path MTU discovery starts by setting `path_MTU` to the `actual_route_MTU` of the route. If packets require fragmentation to get to the final destination, path MTU discovery finds the `path_MTU` by repeatedly decreasing the value until it can send packets to the final destination without fragmentation. Figure 8-1 illustrates this function.

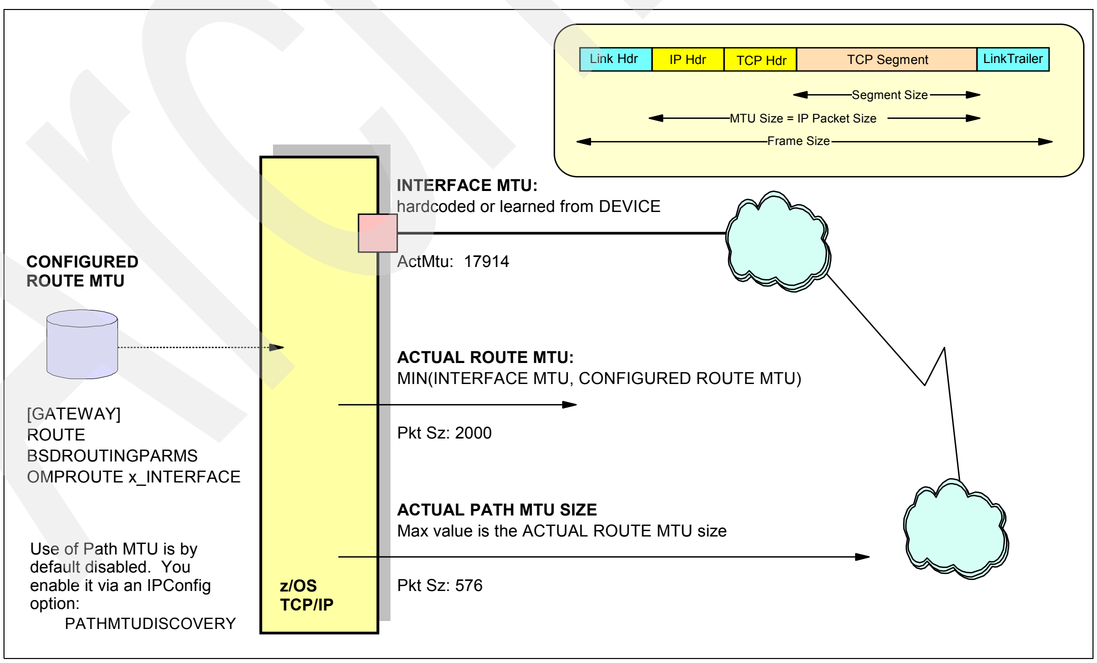


Figure 8-1 Large MTU sizes have a very positive impact on file transfer type of workloads

Use the PMTU option on the **ping** command to determine where fragmentation is necessary in the network. The PMTU YES option differs from the PMTU IGNORE option in the way the ping completes its echo function. Refer to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781, for a detailed discussion about the PMTU option.

Note the following recommendations and guidelines:

- Enable path MTU discovery in configurations where traffic originating in the z/OS TCP/IP stack will traverse multiple hops with different MTU sizes.
- When using OSA-Express Gigabit Ethernet (which supports an interface MTU of 8992), be aware that not all routers and switches support a value this large. Either ensure that all routers and switches in your configuration support 8992, or specify a lower configured_route_MTU.
- When using OMPROUTE, specify the MTU keyword for each IPv4 interface and configure all nodes on a LAN to use the same MTU value. Otherwise, you might encounter problems, such as OSPF adjacency errors.

The ping command detection of network MTU

You can use the **ping** command in z/OS Communications Server as a diagnostic tool to determine MTU and fragmentation problems in the network. Path MTU parameters are added to the TSO and z/OS UNIX versions of the command to prevent the outbound echo request packets from being fragmented and to specify the type of path MTU discovery support for the command.

Example 8-1 shows the output of the TSO command:

“`ping 10.1.1.240 (tcp tcpipa count 4 pmtu ignore length 4096`”

In this command output:

`count 4`:	Sets the number of echo requests that are sent to the host.
`pmtu ignore`:	Specifies that the outbound echo request packets are not fragmented at the local host or in the network, and that any MTU values determined by path MTU discovery for the destination are ignored.
`length 4096`:	Sets the number of data bytes for the echo request.

Example 8-1 Output of TSO command ping PMTU

```
CS V1R11: Pinging host 10.1.1.240
Ping #1 needs fragmentation at: 10.1.3.11 (10.1.3.11)
  Next-hop MTU size is 1492
Ping #2 needs fragmentation at: 10.1.3.11 (10.1.3.11)
  Next-hop MTU size is 1492
Ping #3 needs fragmentation at: 10.1.3.11 (10.1.3.11)
  Next-hop MTU size is 1492
Ping #4 needs fragmentation at: 10.1.3.11 (10.1.3.11)
  Next-hop MTU size is 1492
***
```

For more information about the **ping** command, refer to *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781.

Path MTU discovery for Enterprise Extender

Path MTU discovery for Enterprise Extender (EE) enables VTAM to determine dynamically any MTU size changes that are associated with IPv4 and IPv6 EE connections in the IP network. VTAM can segment the HPR data to avoid IP packet fragmentation.

A VTAM start option, PMTUD, controls whether path MTU discovery is enabled for Enterprise Extender:

- When `PMTUD=NO`, VTAM disable path MTU discovery for IPv4 and IPv6 EE connections, VTAM will only learn of local MTU changes associated with first hop of the IP routes.
- When `PMTUD=TCPVALUE`, VTAM accepts the TCP/IP stack (associated with EE) setting. For IPv6 EE connections, path MTU discovery is enabled by default. For IPv4 EE connections, path MTU discovery is enabled when the PATHMTUDISCOVERY operand is coded on the IPCONFIG profile statement.

When VTAM detects the EE connection MTU size has changed during the transmission of an EE packet, the MTU size is altered. You can check the IST2029I message for the current MTU size using **`DISPLAY NET,EE,ID=name,LIST=DETAIL`**, as shown in Example 8-2.

Example 8-2 Output of command D NET,EE,ID=name,DETAIL

```
D NET,EE,ID=EEPUCS04,DETAIL
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = EE 392
IST2001I ENTERPRISE EXTENDER CONNECTION INFORMATION
IST075I NAME = EEPUCS04, TYPE = PU_T2.1
...
IST2030I PORT PRIORITY = SIGNAL
IST2029I   MTU SIZE =  1433                      1
...
IST2031I PORT PRIORITY = NETWORK
IST2029I   MTU SIZE =  1433
...
IST2032I PORT PRIORITY = HIGH
IST2029I   MTU SIZE =  972                        2
...
IST2033I PORT PRIORITY = MEDIUM
IST2029I   MTU SIZE =  972                        2
...
IST2034I PORT PRIORITY = LOW
IST2029I   MTU SIZE =  1433
IST314I END
```

In this example, the numbers correspond to the following information:

1. This MTU size has already been reduced to account for various header lengths such as the IP, UDP, and LLC headers necessary for EE traffic.
2. EE only learns of MTU changes when HPR data is transmitted across each unique port. Because not all of the EE ports might be transmitting data simultaneously, you might see different MTU sizes for some of the EE ports.

Take into account the following implementation considerations:

- The learned path MTU is considered to be stale information every 20 minutes. By that time, the path MTU is reset to the first hop MTU size. The system will experience the same overhead of learning the path MTU at 20 minutes intervals.
- A firewall must permit ICMP messages when “Path MTU Discovery” is enabled. Otherwise, HPR transmission stall occurs if the path MTU is smaller than that of the “first hop” MTU size.

In Figure 8-2, HPR transmits a large packet base on the path MTU size, and the TCP/IP stack turns on the "do not fragment" bit in the IPv4 header **1**. When the packet arrives at the router with the smaller "next hop" MTU **2**, an ICMP is returned, and the packet is discarded **3**. If the firewall does not allow ICMP package, it drops the ICMP **4**. The TCP/IP stack and VTAM will not learn of the path MTU. It retransmits these large missing packets when the partner reports a packet gap. This process repeats indefinitely and causes the "transmission stall" with message IST2245I.

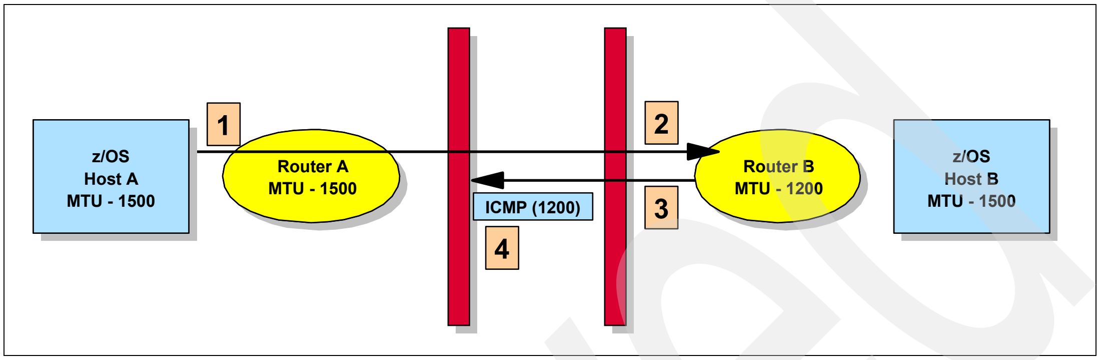


Figure 8-2 ICMP message blocked by firewall

For more information about the path MTU discovery for EE, refer to *z/OS V1R10 Communications Server: SNA network Implementation Guide,* SC31-8777.

Note: We can also use an alternative solution to control Enterprise Extender's MTU size by the VTAM MTU operand in the switch major node's PU statement, EE XCA major node's GROUP statement (Connection Network only) or model major node's PU statement. Refer to *z/OS V1R8.0 Communications Server: SNA Resource Definition*, SC31-8778.

8.2.2 OSA-Express2 LAN idle timer function

OSA-Express supports an inbound "blocking" (or packing) function over the QDIO interface. This function affects how long OSA-Express will hold packets before presenting those packets to the host. In this case, "presenting" means assigning the read buffer to the host, which is a matter of updating the state of the host buffer to host owned. In most cases this same action will result in an interrupt to the host for this QDIO data device. Therefore, this function indirectly affects the QDIO interrupt processing.

The host can pass various time intervals to OSA when the QDIO data device is activated. In the z/OS case, the system administrator can adjust this setting. However, the setting is static and cannot be changed unless the connection to OSA-Express is terminated (device is stopped) and reestablished (restart the device).

In the TCP/IP profile, the user can define a LAN Idle setting for an OSA- Express2 port (in QDIO mode). This is performed by specifying the INBPERF parameter in the TCP/IP profile. INBPERF is an optional parameter indicating how frequently the adapter should interrupt the host for inbound traffic.

There are three supported static settings: MINCPU, MINLATENCY, and BALANCED. The static settings use static interrupt-timing values. The static values are not always optimal for all workload types or traffic patterns, and cannot account for changes in traffic patterns.

There is also one supported dynamic setting. This setting causes the TCP/IP stack to dynamically adjust the timer-interrupt value while the device is active and in use. This function exploits an OSA-Express2 function called Dynamic LAN idle.

The valid sub-parameters to use with the INBPERF parameter are:

- DYNAMIC

  This setting causes the TCP/IP stack to dynamically signal the OSA-Express2 feature to change the timer-interrupt value, based on current inbound workload conditions.

- MINCPU

  This setting uses a static interrupt-timing value, selected to minimize TCP/IP stack interrupts without regard to throughput.

- MINLATENCY

  This setting uses a static interrupt-timing value, selected to minimize latency (delay), by more aggressively presenting received packets to the TCP/IP stack.

- BALANCED

  This setting uses a static interrupt-timing value, selected to achieve reasonably high throughput and reasonably low CPU consumption.

Note: BALANCED mode is the default value for INBPERF statement.

The INBPERF parameter can be specified on the LINK or INTERFACE statement (see Example 8-3).

Example 8-3 TCP/IP profile definition

```
INTERFACE OSA2080I
DEFINE IPAQENET
PORTNAME OSA2080
IPADDR 10.1.2.21/24
INBPERF DYNAMIC
MTU 1492
VLANID 10
VMAC
;
```

Example 8-4 displays the output of the TCPIP command.

```
D TCPIP,TCPIPC,NETSTAT,DEV
```

Example 8-4 Output of Display Netstat command

```
INTFNAME: OSA2080I          INTFTYPE: IPAQENET   INTFSTATUS: READY
    PORTNAME: OSA2080   DATAPATH: 2082     DATAPATHSTATUS: READY
    SPEED: 0000001000
    IPBROADCASTCAPABILITY: NO
    VMACADDR: 020019749925   VMACORIGIN: OSA    VMACROUTER: ALL
    ARPOFFLOAD: YES                   ARPOFFLOADINFO: YES
    CFGMTU: 1492                      ACTMTU: 1492
    IPADDR: 10.1.2.31/24
    VLANID: 10                        VLANPRIORITY: DISABLED
    DYNVLANREGCFG: NO                 DYNVLANREGCAP: YES
    READSTORAGE: GLOBAL (4096K)       INBPERF: DYNAMIC
    CHECKSUMOFFLOAD: YES
    SECCLASS: 255                     MONSYSPLEX: NO
  MULTICAST SPECIFIC:
    MULTICAST CAPABILITY: YES
    GROUP             REFCNT        SRCFLTMD
    -----             ------        --------
      224.0.0.1         0000000001    EXCLUDE
        SRCADDR: NONE
    INTERFACE STATISTICS:
      BYTESIN                           = 0
      INBOUND PACKETS                   = 0
      INBOUND PACKETS IN ERROR          = 0
      INBOUND PACKETS DISCARDED         = 0
      INBOUND PACKETS WITH NO PROTOCOL  = 0
      BYTESOUT                          = 420
      OUTBOUND PACKETS                  = 5
      OUTBOUND PACKETS IN ERROR         = 0
      OUTBOUND PACKETS DISCARDED        = 0
```

Note: When specified for an OSA device that does not support this function, the BALANCED option is used for the INBPERF parameter.

For detailed information about this topic, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

8.2.3 Tracing

From a performance perspective, you need to disable all tracing because tracing activity can have a significant impact on system performance.

To disable tracing, include the following line in the TCPIP.PROFILE:

```
ITRACE OFF
```

To turn off **ctrace** for TCP/IP, issue the following command:

```
TRACE  CT,OFF,COMP=SYSTCPIP,SUB=(tcp_proc_name)
```

If you want to use tracing, use the appropriate parameters on the ITRACE statement. For example, to trace the configuration component, specify the ITRACE parameters as follows:

```
ITRACE ON CONFIG 1
```

In this ITRACE command, `CONFIG 1` specifies “tracing level 1” for the configuration component. Refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776, for more information about ITRACE.

8.3 z/OS UNIX System Services tuning

The publication *z/OS UNIX System Services Planning*, GA22-7800, provides useful tuning information. Note the following points:

- Be certain that the UNIXMAP RACF® class is populated and cached.
- Update PROFILE.TCPIP, TCPIP.DATA, and FTP.DATA files with the applicable recommendations discussed in this chapter.
- Estimate how many z/OS UNIX users, processes, PTYs, sockets, and threads are needed, and update the appropriate BPXPRMxx members in PARMLIB. These parameters include MAXPROCSYS, MAXPROCUSER, MAXUIDS, MAXFILEPROC, MAXPTYS, MAXTHREADTASKS, and MAXTHREADS.
- Set MAXSOCKETS(n) to a high number to avoid a shortage.

  As an example, each z/OS UNIX Telnet session requires one z/OS UNIX socket, and each FTP session requires one z/OS UNIX socket. After the MAXSOCKETS limit is reached, no more Telnet, FTP sessions, or other applications that require z/OS UNIX sockets are allowed to start.
- Optimize HFS and zFS usage. In general, zFS provides better performance. If usage exceeds cache usage, consider spreading HFS and zFS data sets over more DASD volumes. Consider whether shared (multiple LPAR) usage is needed, because this adds functionality (sharing), but with some performance expense. (Information about file system tuning is beyond the scope of this book.)
- Monitor z/OS UNIX resources usage with RMF or system commands (DISPLAY ACTIVE, DISPLAY OMVS, and so forth).

8.4 Storage requirements

For a system with significant network activity, estimate the storage requirements for CSM, CSA, and SQA. We have provided storage usage summaries for typical applications, such as Telnet, FTP, CICS socket, and Web server. This can serve as a starting point for estimating CSA, SQA, and CSM storage requirements for these and other applications.

8.4.1 TCP and UDP buffer sizes

When send/recv buffer sizes are not specified in the PROFILE, a default size of 16 KB is used for send/recv buffers and a default of 32 KB is used for the TCP window size. If send/recv buffer sizes are specified, they are used as specified and the TCP window size is set to twice the TCP recv buffer size, up to the maximum TCPMAXRCVBUFSIZE value (the default is 256 KB).

You can specify the send/recv buffer sizes on the TCPCONFIG and UDPCONFIG statements, as shown in Example 8-5.

Example 8-5 Send/recv buffer sizes definition

```
TCPCONFIG   TCPSENDBFRSIZE    65535
            TCPRCVBUFRSIZE    65535
UDPCONFIG   UDPSENDBFRSIZE    65535
            UDPRCVBUFRSIZE    65535
```

Socket applications can override these values for a specific socket using the `setsockopt` call:

```
setsockopt(SO_SNDBUF)
setsockopt(SO_RCVBUF)
```

Note: The FTP server and client applications override the default settings and use 64 KB, as the TCP window size and 180 KB for send/recv buffers. No changes are required in the TCPCONFIG statement for the FTP server and client.

If the TCPMAXRCVBUFRSIZE parameter is used in the TCP/IP profile, ensure that the value specified is 180 KB or greater. If the value specified is lower than 180 KB, FTP will not be able to use TCP window size of 180 KB for FTP.

TCP throughput for high-latency networks

Streaming workload over large bandwidth and high latency networks (such as satellite links) is in general constrained by the TCP window size. However, it takes time to send data over such a network. At any given point in time, data that fills the full window size is “in transit” and cannot be acknowledged until it arrives at the receiver. The sender can send data up to the window size and then must wait for an ACK to advance the window size before the next chunk of data can be sent.

If it were possible to adjust the window size dynamically to what it takes to fill the network between the sender and the receiver, higher throughput might be achieved.

To help improve performance for inbound streaming workloads over networks with large bandwidth delay, z/OS Communications Server implements Dynamic Right Sizing (DRS). The DRS function keeps the pipe full for inbound streaming TCP connections over networks with a large capacity and high latency and prevents the sender from being constrained by the receiver’s advertised window size. This function improves performance for inbound streaming workloads over such networks. The stack enables this function automatically with no configuration needed.

How DRS works

The stack automatically detects which TCP connections might benefit from dynamic right sizing. Specifically, the stack looks for a high latency inbound streaming TCP connection that is using a receive buffer size of at least 64 KB. For such a connection, the stack attempts to not constrain the sender by increasing the receive buffer size for the connection, which in turn causes the stack to adjust the advertised receive window (up to a max of 2 MB). Note that the intent of DRS is not to buffer 2 MB of data. Instead, the idea is to advertise *up to that size* to allow more data to be in the pipe so the sender can keep the pipe full.

In reality, DRS will, on the receiver side, adjust dynamically the window size upward (beyond 180 KB if needed) in an attempt to fill the pipe between the sender and the receiver. The aim is that as soon as the sender has sent the end of its window, the sender receives an ACK from the receiver. That ACK allows the sender to advance the window and send another chunk of data onto the network.

If the stack detects evidence that the application is not keeping up by reading the data fast enough, then the stack disables DRS for that connection and resets the receive buffer to its original size.

> **Note:** This function allows the stack to grow the TCP receive buffer size up to 2 MB for certain inbound streaming TCP connections regardless of the value specified with the TCPMAXRCVBUFRSIZE parameter on the TCPCONFIG statement. Applications can request a TCP receive buffer size on the SO_RCVBUF socket option on SETSOCKOPT(). This function does not take effect for applications that request a size smaller than 64 KB on this socket option. The TCPRCVBUFRSIZE parameter on TCPCONFIG defines the default receive buffer size for applications which do not use this socket option. Thus, if this value is less than 64 KB, then this function does not take effect for applications that do not use this socket option.

Activation and verification

DRS is activated automatically. You need not do anything to activate it.You can use the **`netstat ALL/-A`** command to see if the stack is using the dynamic right sizing function for a specific TCP connection, as shown in Example 8-6.

Example 8-6 UNIX netstat -A command display

```
CS01 @ SC30:/u/cs01>netstat -p tcpip -A
MVS TCP/IP NETSTAT CS V1R11       TCPIP Name: TCPIP           11:09:18
...
ReXmt:              0000000000       ReXmtCount:         0000000000
DupACKs:            0000000001       RcvWnd:             0002097151    1
SockOpt:            A500             TcpTimer:           00
TcpSig:             01               TcpSel:             C0
TcpDet:             C0               TcpPol:             00
TcpPrf:             F0    2
QOSPolicy:          No
RoutingPolicy:      No
ReceiveBufferSize:  0002097151    3 SendBufferSize:     0000184320
```

In this example, the numbers correspond to the following information:

1. The RcvWnd field shows the receive window size which is currently being advertised.
2. The TcpPrf field shows that DRS function is active for this connection because the x40 bit(11110000) in the TcpPrf byte is *on*.
3. The ReceiveBufferSize field shows the receive buffer size currently in effect (which might be because of an adjustment made by the DRS function).

8.4.2 Communications Storage Manager use of storage

Communications Storage Manager (CSM) storage consists of the buffer pools that are shared between SNA and TCP/IP for each z/OS system image. Use at least the default (120 MB) unless you are storage-constrained and need to reduce the size of your estimated usage (CSM has always adjusted the requested maximum ECSA value when it exceeds 90% of the ECSA available on the z/OS system).

Migration Tip: z/OS Communications Server now exploits CSM more than the previous releases because SCBs have been moved to 64-bit common storage to reduce ECSA usage. Nevertheless, keep in mind that the capacity planning workloads used to derive the numbers in the CSM usage table drive the system much harder than is likely to be the case in a customer shop. Therefore, continue to use common migration techniques of capturing CSM usage pre- and post-migration to determine whether you need to adjust the hlq.PARMLIB(IVTPRM0) values on the new system.

Table 8-3 provides a summary of CSM storage usage based on an internal performance benchmark. This table provides a starting point for tuning.

Table 8-3 z/OS CSM usage table

Application	# Users/clients	Workload	Max. CSM (ECSA)	Max. CSM (DataSpace)	Max. CSM (FIXED)
CICS Sockets (z10, transaction = 200/200)	50 250 500 1000	98.9 Trans/Sec 495.5 989.7 1975.6	780 KB 1.02 MB 1.06 1.00	24.38 MB 30.49 24.54 24.47	32.44 MB 38.44 32.44 32.44
TN3270 (z10, with Think Time, Echo transactions, transaction = 100/800)	8000 16000 32000 64000 128000 256000	266.6 Trans/Sec 533.0 1066.2 2131.1 4255.1 8368.8	900 KB 2.82 MB 2.85 1.18 3.18 7.53	31.60 MB 54.38 54.72 35.86 61.45 63.44	63.56 MB 85.35 85.35 31.90 57.11 84.75
FTP Inbound Data (z10, with and without Think Time, transaction = 2 MB/1)	1 (Binary Put) 2 4 8 16 32 64 128 128 (no TT)	1.38 MBps 2.75 5.50 10.99 21.95 43.79 87.41 164.90 311.58	884 KB 776 800 820 904 800 996 1.18 MB 1.43	32.61 MB 30.24 30.41 31.18 31.55 32.34 34.71 41.01 69.70	41.12 MB 39.96 40.76 41.12 41.16 41.16 42.14 49.52 74.16
FTP Outbound Data (z10, with and without Think Time, transaction = 1/2 MB)	1 (Binary Get) 2 4 8 16 32 64 128 128 (no TT)	1.36 MBps 2.72 5.44 10.83 21.67 43.40 86.09 158.55 239.11	816 KB 1.16 MB 856 KB 1.55 MB 1.96 1.24 2.18 7.94 25.50	30.44 MB 30.40 30.68 31.84 31.95 32.06 33.44 32.38 33.90	39.96 MB 39.96 40.72 42.10 41.88 41.12 41.62 49.00 90.18

If you use the maximum values from these four workloads, we suggest the following definitions in the IVTPRM00 member of PARMLIB:

- `FIXED MAX(433M)`, which includes 288.5 MB (38.84 + 85.35 + 74.16 + 90.18 MB) for the four workloads plus 144 MB (or 50%) of additional storage for expected growth in the workloads.
- ECSA MAX(54M), which includes 35.5 MB (1.06 + 7.53 + 1.43 + 25.50 MB) for the four workloads plus 18 MB (or 50%) of additional storage for expected growth in the workloads.
- `FIXED MAX` and `ECSA MAX` usage should be monitored and adjusted based on one's workload.

With a different application mix, CSM requirements might change. You can monitor CSM and VTAM buffer usage using the following commands:

```
D NET,BFRUSE,BUFFER=SHORT
D NET,STORUSE
D NET,CSM
D NET,CSM,ownerid=all
```

To temporarily change the amount of storage used by CSM, use the following command and specify the parameters you want to change on the ECSA and FIXED operands:

```
F procname,CSM,ECSA=maxecsa,FIXED=maxfix
```

To permanently change the amount of storage used by CSM or tuning parameters for CSM buffer pools, edit the IVTPRM00 member and issue the command without specifying any operands.

Note: Changing a parameter that decreases the specification of a limit might not take effect immediately. Reducing the usage of the resource to comply to the new limit might require users to free buffers to contract a storage pool's size. This type of change could also result in a CSM constraint condition being indicated to users who are monitoring CSM resource usage.

In z/OS Communications Server, CSM issues IVT559*I messages (found in the system log) to indicate its "water level." Table 8-4 lists some useful CSM messages.

Table 8-4 CSM useful messages

Message number	When issued
IVT5590I	At CSM initialization time when the ECSA MAX value in IVTPRM00 is larger than 90% of the ECSA available on the system. During DISPLAY CSM command processing when the ECSA value in effect has been adjusted by CSM. During MODIFY CSM command processing when the maximum ECSA requested is larger than 90% of the ECSA available on the system.
IVT5591I	MODIFY CSM command processing when the maximum ECSA requested is larger than 90% of the ECSA available on the system.
IVT5592I	Fixed storage usage is above 80% of the MAX FIXED value and is approaching 85% of the MAX FIXED value.
IVT5564I	Current ECSA storage usage goes below 80% of the MAX ECSA value.
IVT5565I	Current fixed storage usage goes below 80% of the MAX FIXED value.

CSM activates the Dynamic CSM Monitor function when:

- Current ECSA storage usage reaches 80% or higher of the MAX ECSA value or the current fixed storage usage reaches 80% or higher of the MAX FIXED value.
- Current ECSA storage usage goes below 75% of the MAX ECSA value and the current fixed storage usage goes below 75% of the MAX FIXED value.

The command for changing CSM Monitor funciton is:

```
F procname,CSM,MONITOR=DYNAMIC|YES|NO
```

CSM usage can be specific to a z/OS Communications Server release. Therefore, we recommend reviewing the performance summary reports based on your release level. You can find the reports at:

http://www-1.ibm.com/support/docview.wss?rs=852&context=SSSN3L&dc=DA480&uid=swg27005524

8.4.3 VTAM buffer settings

Typically, VTAM Buffers are important for TN3270 workload. For a large number of Telnet (TN3270) sessions, we recommend that users change the default VTAM buffer settings for IOBUF, LFBUF, CRPLBUF, TIBUF, and CRA4BUF. Table 8-5 provides guidelines for initial settings.

Table 8-5 z/OS VTAM buffer usage table

Application	**Number of users/clients**	**Workload Throughput (Transactions / Sec)**	**VTAM Buffer (IO00)**	**VTAM Buffer (LF00)**	**VTAM Buffer (CRPL)**	**VTAM Buffer (TI00)**	**VTAM Buffer (CRA4)**
CICS Sockets (z10, transaction = 200/200)	50 250 500 1000	98.9 495.5 990.1 1978.6	5 5 5 5	5 5 5 5	54 54 54 54	21 21 21 21	3 4 4 4
TN3270 (z10, with Think Time, Echo transactions, transaction = 100/800)	8000 16000 32000 64000 128000 256000	266.6 533.0 1066.2 2131.1 4255.1 8363.8	201 201 358 663 2256 2256	8004 16004 32004 64004 128004 256004	1636 1636 1636 1636 1636 1636	459 459 473 727 2457 2457	11 19 19 28 59 128
FTP Inbound Data (z10, with and without Think Time, transaction = 2 MB/1)	1 (Binary Put) 2 4 8 16 32 64 128 128 (no TT)	1.38 2.75 5.50 10.99 21.95 43.79 87.41 164.90 311.58	5 5 5 5 5 5 5 5 5	4 4 4 4 4 4 4 4 4	2 2 2 2 2 2 2 2 2	28 28 28 28 28 28 28 28 28	4 4 4 4 4 4 4 4 4

Application	Number of users/clients	Workload Throughput (Transactions / Sec)	VTAM Buffer (IO00)	VTAM Buffer (LF00)	VTAM Buffer (CRPL)	VTAM Buffer (TI00)	VTAM Buffer (CRA4)
FTP Outbound Data	1 (Binary Get)	1.36	4	4	2	28	4
(z10, with Think	2	2.72	5	4	2	28	4
Time	4	5.44	5	4	2	28	4
transaction =	8	10.83	5	4	2	28	4
1/2 MB)	16	21.67	5	4	2	28	4
	32	43.40	5	4	2	28	4
	64	86.09	5	4	2	28	4
	128	158.55	5	4	2	28	4
	128 (no TT)	239.11	5			28	4

The same commands used previously can help verify the settings:

```
D NET,BFRUSE,BUFFER=SHORT
D NET,STORUSE
Telnet (TN3270) storage utilization
```

You can check Telnet storage utilization using the normal RMF monitor functions to see the storage usage in the Telnet address space.

8.5 Application performance and capacity

This section includes tips on capacity planning and performance for Telnet and FTP.

8.5.1 Telnet (TN3270) capacity planning

A key element in determining capacity is to determine the CPU cost of performing a transaction of interest on a specific machine type. For example, we have used a TN3270 workload in which 100 bytes of data is sent from the client to an echo application and 800 bytes of data is sent by the application as a reply to the client. From our benchmarks, we have derived CPU cost in terms of milliseconds per TN3270 transaction using an IBM z990 2084-232 (three-CP LPAR).

For example, if we want to determine the capacity needed for 8000 users, each performing six of these transactions per user per minute on an IBM z990 2084-232 (three-CP LPAR), we can use the formula shown in Figure 8-3. The key factor in this formula is .000157 CPU-seconds per transaction.

```
 # trans/user  x # users x CPU secs/tran      CPU secs
 ---------------------------------------   =  ---------
           # of Elap secs                     Elap secs

Example:   z/OS V1R7 IP,  8000 users,   6 tr/min/user

 6 tr/u x 8000u x 0.000157 CPU secs/tr           cpu se
 ------------------------------------- =  0.126--------
          60  elap. sec                            elap
```

Figure 8-3 TN3270 CPU requirements

If the CPU secs/Elap sec ratio is greater than 1, then more than one CPU processor would be required. This is a very simplistic approach, of course, and must be used with care. For example, the application involved (our echo program) performed no I/O, used no data sets, and did very little processing. This is not a realistic workload, but it can provide a starting point for capacity planning for TN3270 sessions.

Assuming we recognize the limitations of these calculations, we can calculate the percentage use of the processing power (three CPs) available to the LPAR, as shown in Figure 8-4.

```
CPU secs/Elap Sec
-----------------  *  100 %   =  CPU Util %
 # of processors

# of processors:  3 ( This will be equal to number of
                  390 processors used for the LPAR)

Example:   Therefore, Percentage of CPU utilization on three CP
LPAR to drive 6 tran/user/minute for 8000 users will be


0.126 CPU secs/Elap sec
-----------------------  *  100 %  =  4.2 %
```

Figure 8-4   TN3270 CPU utilization

8.5.2 FTP tuning

The FTP.DATA file is used by the FTP server and client to establish the initial configuration options for an FTP session. The search order for FTP.DATA varies, depending on the execution environment. For a detailed description of the search order, refer to *z/OS Communications Server: IP Configuration Reference*, SC31-8776.

Sample specifications in FTP.DATA include the following parameters:

```
PRIMARY   15
SECONDARY 20
LRECL     64
BLKSIZE 27968
RECFM     FB
CHKPTINT  0
DIRECTORY 27
SQLCOL    ANY
INACTIVE  0
```

z/OS data set attributes play a significant role in FTP performance. Normal z/OS advice is relevant. This means using half-track blocking. If large volumes of FTP data are expected, then the normal advice about the use of multiple channels, disk control units, and disk volumes becomes important. For best performance, define CHKPTINT = 0.

Note that several parameters can be changed during an FTP session by use of the SITE or LOCSITE user commands.

The use of preallocated data sets for FTP transfers to z/OS is usually recommended. If new data set allocations are required, then using BLKSIZE=0 usually allows the system to determine the best block size.

8.5.3 FTP capacity planning

Our methodology for estimating FTP capacity planning is similar to our Telnet methodology. In this case the key factor is the capacity needed to transfer 1 KB of data. In our environment, we found this to be .00000656 CPU-seconds per KB. This will vary for different processors. Our measurement included TCP/IP, VTAM, and FTP address space usage, using a OSA-Express Gigabit Ethernet port as a network gateway (two-CP LPAR).

Our total (TCP/IP + VTAM + FTP) CPU requirements for FTP transfer are given by the formula shown in Figure 8-5. Our formula and key factor are probably more realistic for FTP than the equivalent numbers were for TN3270 because there are fewer additional factors involved in FTP usage.

```
Max KB        CPU  secs       CPU secs
---------  *  ---------   =  ----------
Elap secs        KB          Elap  secs
```

Figure 8-5 General formula to calculate FTP CPU requirements

If we consider a load of 69529.6 KBps for transferring data from client workstations to our z990 (using eight parallel binary PUT operations) doing eight 20 MB file transfers, we have the results shown in Figure 8-6.

```
69529.6 KB       .00000656           .456 CPU secs
---------   *  ------------   =        ----------
Elap secs          KB                    Elap  secs
```

Figure 8-6 Formula to calculate FTP CPU requirements

If the CPU secs/Elap sec ratio is 1 or greater, we need more than one processor capacity to drive the desired throughput. The percentage of system capacity is obtained as shown in Figure 8-7.

```
 CPU secs/Elap Sec
 -----------------  *  100 %  =  CPU Util %
  # of processors

 # of processors:  4 ( This will be equal to number of
                   390 processors used for the LPAR)

Example: For our example we are using z990 / 2084-332 2 processor
LPAR.  Therefore, Total CPU utilization will be


 0.456 CPU secs/Elap sec
 -----------------------   *   100 %   = 22.8 %
      2 processor
```

Figure 8-7 FTP CPU requirements example

Thus, the total z/OS CPU (TCP/IP + VTAM + FTP) requirements for driving 69,529 KB/Sec throughput would require an average CPU utilization of 22.8% of a two-processor z990 (2084-332, two-CP LPAR).

8.6 z/OS Communications Server TCP/IP performance highlights

Every release update of z/OS Communications Server delivers a number of improvements that are related to TCP/IP performance. (Not all of these enhancements are enabled automatically.) The following list provides an overview of the enhancements in this release:

- Accept_and_receive API enhancements
- TCP/IP support for system z10 hardware instrumentation
- TCP/IP pathlength improvements
- Relaxation of the Nagle Algorithm
- Expansion of the TCP receive buffer when applications have exploited *msg_waitall*
- TCP throughput improvements for high-latency networks
- Virtual storage constraint relief
- NSS private key and certificate services for XML appliances
- Enterprise Extender IPSec performance improvements
- Resolver DNS cache
- Sysplex autonomics improvements for FRCA
- QDIO routing accelerator
- Sysplex distributor enhancements
- OSA-Express3 optimized latency mode
- Improved responsiveness to storage shortage conditions affecting OMPROUTE, TCP send and receive queues, and device driver handling of inbound packets

Several of these enhancements are described in this general chapter on performance and tuning. Other enhancements are described elsewhere when it makes sense to present them in the context of the application or TCP/IP function that is directly affected by the performance improvement. Consult the following resources for more details about performance improvements that we do not describe in this chapter:

- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS V1R11.0 Communications Server: SNA Network Implementation*, SC31-8777
- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798
- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7799
- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7801

8.6.1 Detailed information about selected performance enhancements

In this brief section, we describe the enhancements that affect the operations of the TCP/IP stack as a whole. As noted, many other performance enhancements are presented in chapters that describe individual TCP/IP implementations and applications, and you should review those chapters for more information.

TCP throughput improvements for high-latency networks

z/OS Communications Server improves performance for inbound streaming TCP connections over networks with large bandwidth and high latency by automatically tuning the ideal window size for such TCP connections.

Virtual storage constraint relief

The single most important ECSA relief comes from moving the socket control blocks (SCB) out of ECSA and into 64-bit common storage.

Each data area that maps a socket, called *socket control blocks* (SCB), requires roughly 384 bytes of ECSA common storage. If you need a large number of socket connections will result in a substantial amount of storage being used. ECSA virtual storage is a limited resource and the maximum amount allowed is 2,031 MB, and using 64-bit common storage this limit goes from two gigabytes to maximum size of one terabyte. For example, a customer who is supporting a large number of TN3270 connections might experience a need for a substantial amount of this storage.

The 64-Bit Common Area size can be specified using the HVCOMMON keyword in IEASYSxx or on system parameter.

Handling storage congestion on TCP send and receive queues

The TCP layer maintains data queues for each connection that is established. The receive queue holds data that is received at the TCP layer from the remote stack but that is not yet read by the local application. The send queue holds data that is sent by the local application but is not sent by the TCP layer to the remote stack or is not acknowledged by the remote stack. An application that is not reading the data that it is sent causes data to remain on the receive queue of its local TCP layer. In addition, after the receive queue of the local TCP layer is full, data remains on the send queue of the remote TCP layer. Although V1R10 reduced the amount of storage that is held due to old data on TCP receive queues, V1R11 reduces the storage that is held due to old data on TCP send queues.

When data is added to the send queue for a local connection (that is, for a connection with both endpoints on *the same* TCP/IP stack), the storage that is obtained to hold the data is marked as *pageable* storage. (Formerly the storage was fixed and non-pageable.) This pageable storage keeps TCP from holding fixed storage for data on the send queue for a local connection.

When new data is added to the send queue for a non-local connection (that is, a connection with endpoints on *different* TCP/IP stacks), TCP determines whether the application is not making progress or whether fixed storage is constrained. If either is true, the storage obtained to hold the data is marked as pageable storage. This pageable storage keeps TCP from holding a large amount of fixed storage for data on the send queue of an application that is not making progress. Also, when fixed storage is constrained, it keeps TCP from requesting additional fixed storage for the send queues of all applications.

When fixed storage becomes constrained, all non-local TCP connections are checked for unsent data on the send queue. Storage holding any unsent data is marked as pageable storage, which helps to resolve the fixed storage constrained state.

Because the Data Link Control (DLC) needs the data it receives in fixed storage, data that TCP marks as pageable is changed back to fixed before it is passed to the DLC for transmission.

It can be difficult to determine which TCP connections have old data on the queues if you have to issue multiple netstat commands to determine which connections are affected.

However, alerts are now issued to syslogd using TRMD to advise you of the connections suffering from storage constraints. Alerts are issued when a TCP send or receive queue enters constrained state, indicating that there is old data on the queue. Alerts are also issued when the queue later exits constrained state. All alerts include the information necessary to identify the connection.

> **Note:** Both syslogd and TRMD must be started in order for the constraint messages to be recorded.

There are two thresholds that are used to indicate old data on a TCP send or receive queue.

- When the quantity of data on the queue equals the queue buffer size and the oldest data on the queue is at least 30 seconds old.
- When there is any amount of data on the queue and the oldest data on the queue is at least 60 seconds old.

An alert indicating entry into constrained state is issued when a queue exceeds 90 percent of either threshold. An alert indicating exit from constrained state is issued when a queue falls back below 80 percent of both thresholds. Example 8-7 shows the text of the alerts that can be issued to indicate this constraint situation.

Example 8-7 Storage constraint alerts if SYSLOGD and TRMD are running

```
EZZ8662I TRMD TCP receive queue constrained entry 1 logged: date time , connid=
connid ,jobname= jobname , lipaddr= lipaddr , lport= lport , ripaddr= ripaddr ,
rport= rport ,correlator= correlator 3, probeid= probeid , sensorhostname=
sensorhostname

EZZ8663I TRMD TCP receive queue constrained exit 2 logged: date time , connid=
connid ,jobname= jobname , lipaddr= lipaddr , lport= lport , ripaddr= ripaddr ,
rport= rport ,correlator= correlator 3, duration= duration , probeid= probeid ,
sensorhostname= sensorhostname

EZZ8664I TRMD TCP send queue constrained entry logged: date time , connid= connid
,jobname= jobname , lipaddr= lipaddr , lport= lport , ripaddr= ripaddr , rport=
rport ,correlator= correlator , probeid= probeid , sensorhostname= sensorhostname

EZZ8665I TRMD TCP send queue constrained exit logged: date time , connid= connid
,jobname= jobname , lipaddr= lipaddr , lport= lport , ripaddr= ripaddr , rport=
rport ,correlator= correlator , duration= duration , probeid= probeid ,
sensorhostname= sensorhostname
```

Messages EZZ8662I and EZZ8663I are issued respectively for constraints and constraint relief on the receive queues. EZZ8662I is issued when the constraint is entered (1) and EZZ8663I is issued when the connection exits (2) from the constraint. The correlator field in the messages (3) identifies which entry and exit alerts are pairs for the same connection.

If there is an entry message with no corresponding exit message, this is an indication of a queue that is still in constrained state.

Throttling service request blocks for the device drivers

Although multi-processing of inbound QDIO is necessary, allowing an unchecked number of threads to run is unacceptable. Therefore, the TCP/IP stack limits the number of execution threads allowed to process inbound QDIO data. For 1 Gigabit Ethernet, there is a maximum of four execution threads per QDIO data device; for 10 Gigabit Ethernet, the maximum number of execution treads per QDIO data device is represented by the following formula:

```
Min(LPAR CPUs + 1, 4) * 2
```

The number of service request blocks (SRBs) on the available queue is hard coded internally and cannot be tailored.

Discarding inbound packets to the device drivers

To avoid storage constraints in ECSA and CSM used to hold inbound data, the stack can start discarding packets. On a Gigabit Ethernet device, the stack imposes a limit of 2 MB of storage if CSM is critical or constrained. If there is no congestion, then the limit is 4 MB. On a 10 Gb Ethernet or HiperSockets device, the stack imposes a limit of 4 MB if CSM has reached a critical or constrained stage. It imposes a 6 MB limit if CSM is not critical or not constrained.

If a congestion situation is encountered, the operator is notified with an unsolicited message, as shown in Example 8-8. The message remains on the console until either cancelled by the operator or cancelled automatically by the stack after congestion is relieved.

Example 8-8 Unsolicited console message IST2273 for read queue congestion

```
IST2273E PACKETS DISCARDED FOR jobname - READ QUEUE CONGESTION
```

The response to the command to display the TRLE (**D NET,TRL,TRLE=<trlename>**) can contain a message that includes the count of SBALs of 64 KB that are discarded. See **1** in Example 8-9.

Example 8-9 Solicited Message IST2305 to indicate discarded packets

```
D NET,TRL,TRLE=TRL001

IST2305I NUMBER OF DISCARDED INBOUND READ BUFFERS = sbalcnt 1
```

The alternative command, **D NET,E,ID=<trlename>** can also produce message IST2305I.

Each SBAL represents a container of packets that have been discarded.

8.7 TCP/IP performance quick checklist

Use the following checklist of performance items when considering TCP/IP performance:

- The z/OS Workload Manager (WLM) definitions for VTAM, TCP/IP, FTP, OMPROUTE, FTPD, INETD, and any other network functions should be set to the SYSSTC level for best performance.
- Make client and server TCP send and receive buffers at least 64 KB. TCP window size should be set to twice the TCP receive buffer size up to a maximum of the TCPMAXRCVBUFRSIZE value (default is 512 KB).
- When using FTP, use large data set block sizes for traditional z/OS data sets.

- Telnet parameters TIMEMARK, SCANINTERVAL, and INACTIVE are set up with default values that in most cases will not require modification. Here is the information about the default settings:
  - In order to minimize potential traffic, the default for the TIMEMARK parameter is TIMEMARK = 10800 (3 hours). Setting the TIMEMARK value too low could cause excessive flooding of the network with TIMEMARK commands or high storage usage, particularly around typical low-activity times such as during lunch breaks.
  - In order to avoid CPU issues, the default for the SCANINTERVAL parameter is SCANINTERVAL = 1800 (30 minutes). Setting a low SCANINTERVAL value can cause a very significant increase in CPU utilization when there are large numbers of TN3270 users.
  - The INACTIVE default is 0, which means that an inactive user will never be disconnected. If you choose to set a non-zero INACTIVE value, we recommend that you set it to INACTIVE = 5400 (90 minutes).

  > **Note:** TIMEMARK and SCANINTERVAL are intended to help with cleaning up idle connections on the TCP/IP side of a TN3270 session. INACTIVE is intended to do the same for the SNA side of things.
  >
  > Therefore, if a user is not currently logged onto any SNA application (for example, the user has a UNIX System Services Message 10 screen up), INACTIVE will not terminate the TCP/IP side of the user's TN3270 connection. Instead, the SCANINTERVAL and TIMEMARK are supposed to handle that job.

- For sockets applications, use large message sizes (> 1 KB) for better performance.
- Ensure that TCP/IP and all other traces are turned off for optimal performance. Trace activity can create significant additional processing overhead.

8.8 IBM Health Checker

The objective of IBM Health Checker for z/OS is to identify potential problems before they impact your availability or, in worst cases, cause outages. It checks the current active z/OS and sysplex settings and definitions for a system, and compares the values to those suggested by IBM or defined by you. It is not meant to be a diagnostic or monitoring tool, but rather a continuously running preventative that finds potential problems.

IBM Health Checker for z/OS produces output in the form of detailed messages to let you know of both potential problems and suggested actions to take. Note that those messages do not mean that IBM Health Checker for z/OS has found problems that you need to report to IBM. IBM Health Checker for z/OS output messages simply inform you of potential problems, so that you can take action on your installation.

There are several parts to IBM Health Checker for z/OS:

- The framework of the IBM Health Checker for z/OS is the interface that allows you to run and manage checks. The framework is a common and open architecture, supporting check development by IBM, independent software vendors (ISVs), and users.
- Individual checks look for component-, element-, or product-specific z/OS settings and definitions, checking for potential problems. The specific component or element owns, delivers, and supports the checks.

  Checks can be either local, and run in the IBM Health Checker for z/OS address space, or remote, and run in the caller's address space.

In the following sections, we discuss only checks that are related to TCP/IP.

8.8.1 What is a check

A *check* is actually a program or routine that identifies potential problems before they impact your availability or, in worst cases, cause outages. A check is owned, delivered, and supported by the component, element, or product that writes it. Checks are separate from the IBM Health Checker for z/OS framework.

A check might analyze a configuration in the following ways:

- Changes in settings or configuration values that occur dynamically over the life of an IPL. Checks that look for changes in these values should run periodically to keep the installation aware of changes.
- Threshold levels approaching the upper limits, especially those that might occur gradually or insidiously.
- Single points of failure in a configuration.
- Unhealthy combinations of configurations or values that an installation might not check.
- Applications or functions that are scheduled to be removed in a future release. (This is known as a *Migration Health Check*.)

8.8.2 Checks owned by TCP/IP

There are several checks that are owned by TCP/IP:

- `CSTCP_SYSTCPIP_CTRACE_`*`tcpipstackname`*

  This checks whether TCP/IP Event Trace (SYSTCPIP) is active with options other than the default options (MINIMUM, INIT, OPCMDS, or OPMSGS). By default, this check is performed once at stack initialization, and then is repeated once every 24 hours. This default can be overridden on either a POLICY statement in the HZSPRMxx parmlib member, or on a MODIFY command.

  The check name is suffixed by *`tcpipstackname`*, which is the job name of each TCP stack that is started, in order to define a separate check for each stack.

- `CSTCP_TCPMAXRCVBUFRSIZE_`*`tcpipstackname`*

  This checks whether the configured TCP maximum receive buffer size is sufficient to provide optimal support to the z/OS Communications Server FTP Server. By default, this check is performed once at stack initialization and whenever a VARY TCPIP,,OBEYFILE command changes the TCPMAXRCVBUFRSIZE parameter.

By default, it checks that TCPMAXRCVBUFRSIZE is at least 180 KB. These defaults can be overridden on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command.

The check name is suffixed by `tcpipstackname`, which is the job name of each TCP stack that is started, in order to define a separate check for each stack.

- `CSTCP_SYSPLEXMON_RECOV_tcpipstackname`

  This checks whether the IPCONFIG DYNAMICXCF or IPCONFIG6 DYNAMICXCF parameters have been specified and the GLOBALCONFIG SYSPLEXMONITOR RECOVERY parameter has been specified. This check produces an exception message if the IPCONFIG DYNAMICXCF or IPCONFIG6 DYNAMICXCF parameters were specified, but the GLOBALCONFIG SYSPLEXMONITOR NORECOVERY parameter is in effect.

  By default, this check is performed once at stack initialization. This default can be overridden on either a POLICY statement in the HZSPRMxx parmlib member or on a MODIFY command.

  The check name is suffixed by `tcpipstackname`, which is the job name of each TCP stack that is started, in order to define a separate check for each stack.

- `CSTCP_CINET_PORTRNG_RSV_tcpipstackname`

  This check is used for z/OS check in a Common INET (CINET) environment. It determines whether the port range specified by the INADDRANYPORT and INADDRANYCOUNT parameters in the BPXPRMxx parmlib member are reserved for OMVS on the TCP/IP stack. Reserving a port range prevents the stack from allocating a port that can later be allocated by CINET. Refer to Example 8-11 on page 289.

8.8.3 Migration Health Check

Support is removed for the following functions:

- Boot Information Negotiation Layer (BINL) server function
- Berkeley Internet Name Domain 4.9.3 (BIND 4.9.3) DNS server, including the Connection Optimization (DNS/WLM) function
- Dynamic Host Configuration Protocol (DHCP) server function
- Network Database (NDB) server function

The Migration Health Checks are available for IBM Health Checker on z/OS V1R9 and z/OS V1R10 to help prepare for a migration. Because you need to run migration checks on the existing system before you migrate to a new release, the checks are provided in the service stream as PTFs.

You must install the PTFs on your current system. Similar to other IBM Health Checker for z/OS checks, you can find migration checks using the functional PSP bucket HCHECKER. Alternatively, you can find all IBM Health Checker for z/OS checks at:

http://www.ibm.com/systems/z/os/zos/hchecker/check_table.html

After you migrate to the current release, you need to obtain any updates to the Migration Health Check function and rerun health checker to verify that you have implemented the necessary migration steps.

For more detailed information about IBM Health Checker for z/OS, refer to *IBM Health Checker for z/OS: User's Guide*, SA22-7994.

8.8.4 Health Monitor checks with commands

Use Health Checker system commands to get a summary display of its checks and their status, as shown in Example 8-10.

Example 8-10 Display Health Checks status using a command

```
F HZSPROC,DISPLAY,CHECKS
HZS0200I 01.32.45 CHECK SUMMARY     334
CHECK OWNER      CHECK NAME                      STATE STATUS
IBMCS            CSTCP_CINET_PORTRNG_RSV_TCPIPB   AE   EXCEPTION-MED
IBMCS            CSTCP_SYSPLEXMON_RECOV_TCPIPB    AE   SUCCESSFUL
IBMCS            CSTCP_TCPMAXRCVBUFRSIZE_TCPIPB   AE   SUCCESSFUL
IBMCS            CSTCP_SYSTCPIP_CTRACE_TCPIPB     AE   SUCCESSFUL
IBMCS            CSTCP_CINET_PORTRNG_RSV_TCPIPA   AE   EXCEPTION-MED
IBMCS            CSTCP_SYSPLEXMON_RECOV_TCPIPA    AE   EXCEPTION-LOW
IBMCS            CSTCP_TCPMAXRCVBUFRSIZE_TCPIPA   AE   SUCCESSFUL
IBMCS            CSTCP_SYSTCPIP_CTRACE_TCPIPA     AE   SUCCESSFUL
IBMUSS           USS_CLIENT_MOUNTS                AE   SUCCESSFUL
IBMUSS           USS_PARMLIB_MOUNTS               AE   SUCCESSFUL
IBMUSS           USS_MAXSOCKETS_MAXFILEPROC       AE   EXCEPTION-LOW
IBMUSS           USS_AUTOMOUNT_DELAY              AE   SUCCESSFUL
IBMUSS           USS_FILESYS_CONFIG               AE   SUCCESSFUL
IBMCS            CSTCP_CINET_PORTRNG_RSV_TCPIP    AE   EXCEPTION-MED
IBMCS            CSTCP_SYSPLEXMON_RECOV_TCPIP     AE   SUCCESSFUL
IBMCS            CSTCP_TCPMAXRCVBUFRSIZE_TCPIP    AE   SUCCESSFUL
IBMCS            CSTCP_SYSTCPIP_CTRACE_TCPIP      AE   SUCCESSFUL
...
IBMCS            CSVTAM_T1BUF_T2BUF_NOEE          AD   ENV N/A
IBMCS            CSVTAM_T1BUF_T2BUF_EE            AE   SUCCESSFUL
IBMCS            CSVTAM_VIT_OPT_ALL               AE   SUCCESSFUL
IBMCS            CSVTAM_VIT_DSPSIZE               AE   SUCCESSFUL
IBMCS            CSVTAM_VIT_OPT_PSSSMS            AE   SUCCESSFUL
IBMCS            CSVTAM_VIT_SIZE                  AE   SUCCESSFUL
IBMCS            CSVTAM_CSM_STG_LIMIT             AE   SUCCESSFUL
...
```

Example 8-10 displays only a partial list of checks. The TCP/IP checks are highlighted.

The letters in the state column can be:

`A` Active
`I` Inactive
`E` Enabled
`D` Disabled

The status field of the display shows the status of the check; that is, whether the check was successful or generated an exception message. If an exception message was generated, it indicates whether the exception severity level is low, medium, or high.

The status field can also indicate if a check was not run because it was not applicable in the current environment or due to an unexpected error during check processing.

In the SDSF panel, type `SDSF CK` and select **CSTCP_CINET_PORTRNG_TCPIPA** to show the message issued when the CSTCP_CINET_PORTRNG_RSV_tcpipstackname check is invoked and finds that the configuration does not agree with the best practice suggested by IBM; see Example 8-11.

Example 8-11 Display health check message in SD.CK panel

```
CHECK(IBMCS,CSTCP_CINET_PORTRNG_RSV_TCPIPA)
START TIME: 11/14/2008 11:06:54.235735
CHECK DATE: 20070901  CHECK SEVERITY: MEDIUM

* Medium Severity Exception *

EZBH008E The port range defined for CINET use has not been reserved for
OMVS on this stack.
.....

END TIME: 11/14/2008 11:06:54.251588  STATUS: EXCEPTION-MED
```

8.8.5 Health Monitor checks with GUI

The OMEGAMON® z/OS Management Console can be used as the Health Checker's graphical user interface (GUI) on your workstation.

The OMEGAMON z/OS Management Console Quick Install Option includes software for z/OS and Windows®. The software is available by download from the z/OS downloads site: (and can also be ordered by tape and DVD).

http://www.ibm.com/servers/eserver/zseries/zos/downloads/

Instructions are in the *Program Directory*. (The software can also be ordered by tape or DVD.) For complete documentation, refer to the IBM Tivoli® Monitoring and OMEGAMON XE Information Center:

http://publib.boulder.ibm.com/infocenter/tivihelp/v15r1/index.jsp?topic=/com.ibm.omegamon_mancon.doc/welcome.htm

Figure 8-8 shows the Health Monitor Status in the OMEGAMON z/OS Management Console.

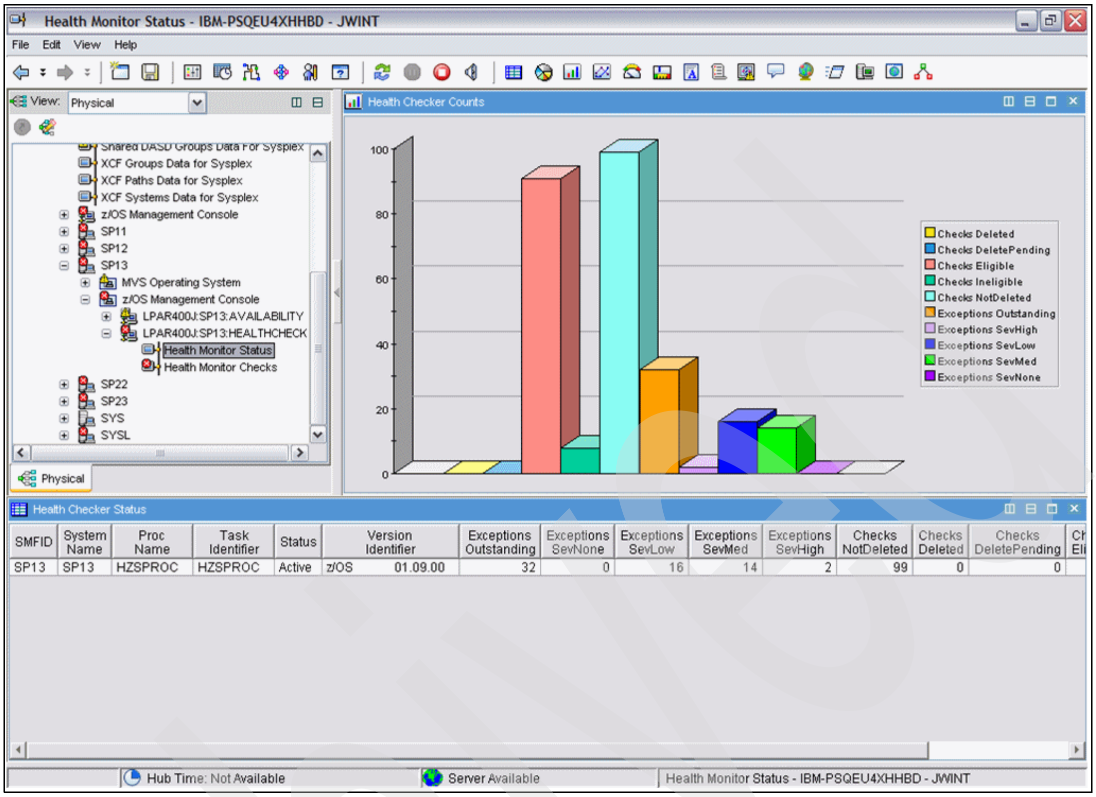


Figure 8-8 Health Checker GUI window with OMEGAMON z/OS MC

When you double-click Health Monitor Checks, the panel shown in Figure 8-9 appears. It presents the current status of all the checks. CSTCP_* checks are the TCP/IP health checks. The OMEGAMON z/OS Management Console will automatically update the check status by the refresh interval values customized in z/OS and the GUI (the default is 5 minutes).

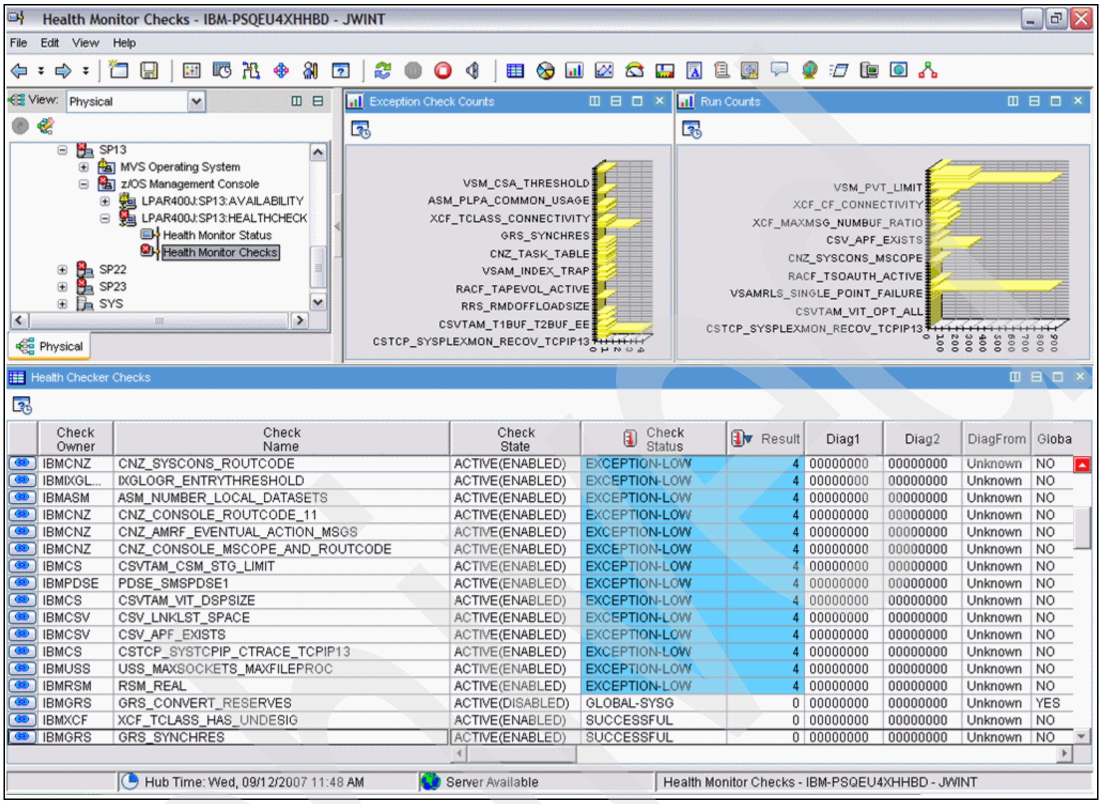


Figure 8-9 Checks Status window

The Checks Status column alerts you by using different colors for the checks in exception status. To see more detailed information about a specific check, double-click the links at the bottom portion of the panel; see Figure 8-10.

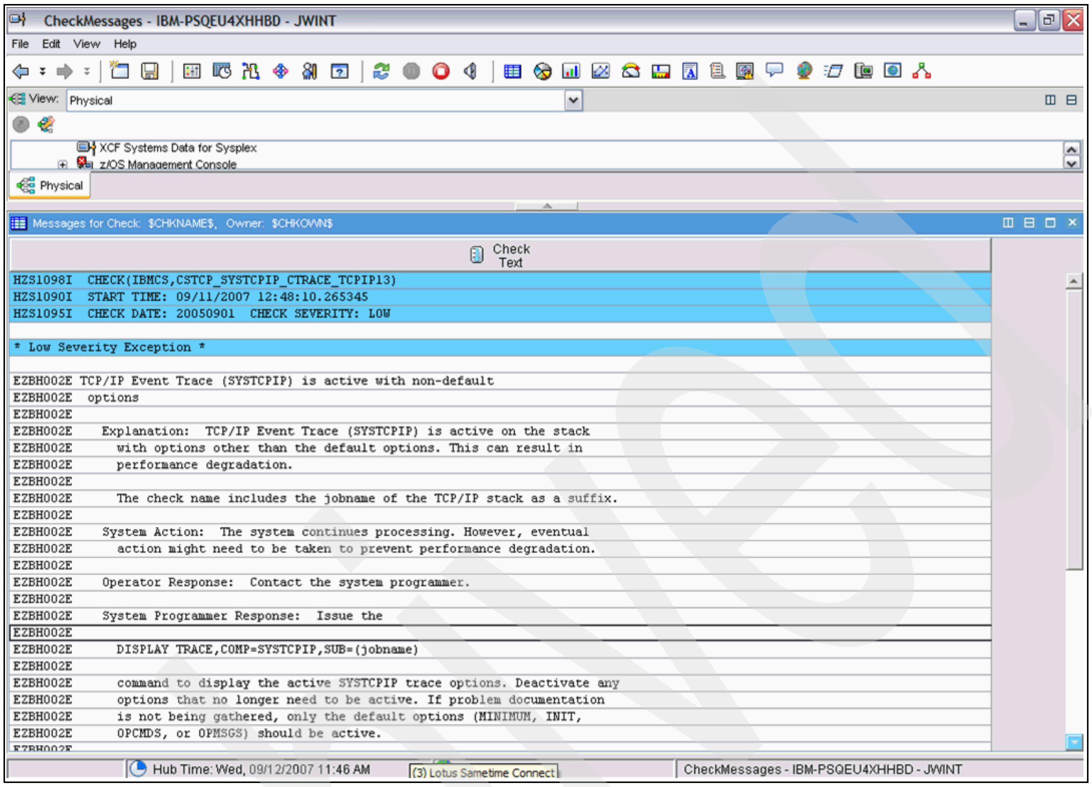


Figure 8-10 TCP/IP Check Message window

A

HiperSockets Multiple Write

In z/OS Communications Server, a HiperSockets interface can move multiple output data buffers in a single write operation. CPU usage can be reduced and there might be a performance improvement for large outbound messages typically generated by traditional streaming workloads such as FTP.

This appendix includes three scenarios to show the CPU utilization reduction using this facility.

In the profile TCP/IP, we used the following statements:

- GLOBALCONFIG NOTCPIPSTATISTICS
- GLOBALCONFIG NOTCPIPSTATISTICS IQDMULTIWRITE
- GLOBALCONFIG NOTCPIPSTATISTICS IQDMULTIWRITE ZIIP IQDIOMULTIWRITE

For a description of HiperSockets Multiple Write, *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798.

The environment used for our tests

We transmitted a large file from A23 to A24 using FTP. We stopped all the other HiperSockets and OSA Express devices. See Figure A-1.

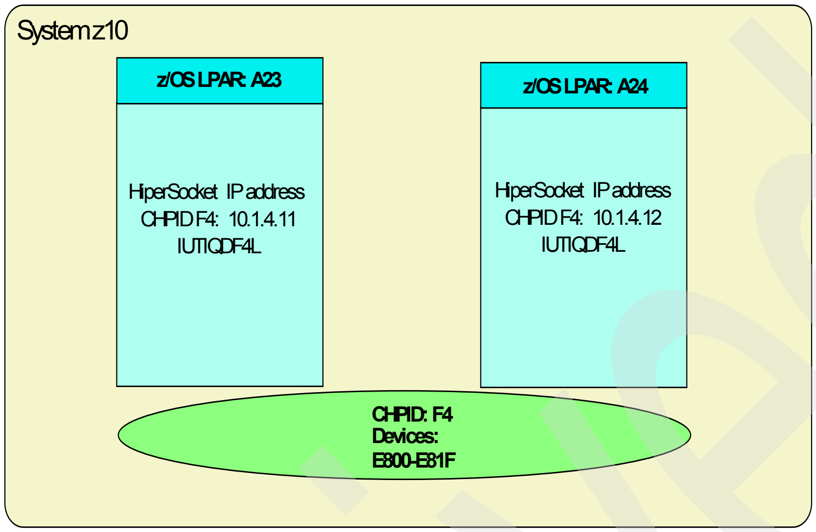


Figure A-1 HiperSockets Multiple Write test environment

The job used to test the HiperSockets Multiple Write is shown in Example A-1.

Example A-1 FTP job with large files

```
//FTPBAT1 JOB (999,POK),'Batch FTP',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=0M
/*JOBPARM L=999,SYSAFF=SC30
//FTP      EXEC PGM=FTP PARM='/-d (EXIT'
//SYSTCPD  DD  DISP=SHR,DSN=TCPIPA.TCPPARMS(DATAA30)
//SYSFTPD  DD  DISP=SHR,DSN=TCPIPA.TCPPARMS(FTPDA30)
//SYSPRINT DD  SYSOUT=*
//OUTPUT   DD  SYSOUT=*,LRECL=160,BLKSIZE=1600,RECFM=FB
//INPUT    DD  *
 10.1.4.12
 cs09
 ******
 ebcdic
 mode b
 site primary=1611 secondary=50
 site recfm=u blksize=27998 cylinders volume=WORK01 unit=3390
 PUT 'RC33.HOM.SEQ1.D070926' SEQ1
 PUT 'RC33.HOM.SEQ1.D070926' SEQ2
 PUT 'RC33.HOM.SEQ1.D070926' SEQ3
 site recfm=u blksize=27998 cylinders volume=WORK02 unit=3390
 PUT 'RC33.HOM.SEQ1.D070926' SEQ4
 PUT 'RC33.HOM.SEQ1.D070926' SEQ5
 PUT 'RC33.HOM.SEQ1.D070926' SEQ6
 site recfm=u blksize=27998 cylinders volume=WORK01 unit=3390
 PUT 'RC33.HOM.SEQ1.D070926' SEQ7
 PUT 'RC33.HOM.SEQ1.D070926' SEQ8
 PUT 'RC33.HOM.SEQ1.D070926' SEQ9
```

```
 site recfm=u blksize=27998 cylinders volume=WORK02 unit=3390
 PUT 'RC33.HOM.SEQ1.D070926' SEQA
 PUT 'RC33.HOM.SEQ1.D070926' SEQB
 PUT 'RC33.HOM.SEQ1.D070926' SEQC
 quit
/*
```

In the first test, we used the parameters shown in Example A-2 in profile TCP/IP for LPAR A23 (SC30) and LPAR A24 (SC31).

Example A-2 Extract of each profile TCP/IP

```
TCPIPA.TCPPARMS(PROFA30)

GLOBALCONFIG NOTCPIPSTATISTICS

DEVICE IUTIQDF4   MPCIPA
LINK   IUTIQDF4L  IPAQIDIO    IUTIQDF4

HOME
   10.1.4.11     IUTIQDF4L

START IUTIQDF4

*********************************************************************************
*********
TCPIPA.TCPPARMS(PROFA31)

GLOBALCONFIG NOTCPIPSTATISTICS

DEVICE IUTIQDF4   MPCIPA
LINK   IUTIQDF4L  IPAQIDIO    IUTIQDF4

HOME
   10.1.4.12     IUTIQDF4L

START IUTIQDF4
```

To verify the options in effect, we issued the following command (as shown in Example A-3):

```
D TCPIP,TCPIPA,N,CONFIG
```

Example A-3 Extract of NETSTAT, CONFIG

```
TCP CONFIGURATION TABLE:
DEFAULTRCVBUFSIZE:  00065536  DEFAULTSNDBUFSIZE: 00065536
DEFLTMAXRCVBUFSIZE: 00262144  SOMAXCONN:         0000000010
MAXRETRANSMITTIME:  120.000   MINRETRANSMITTIME: 0.500
ROUNDTRIPGAIN:      0.125     VARIANCEGAIN:      0.250
VARIANCEMULTIPLIER: 2.000     MAXSEGLIFETIME:    30.000
DEFAULTKEEPALIVE:   00000120  DELAYACK:          YES
RESTRICTLOWPORT:    NO        SENDGARBAGE:       NO
TCPTIMESTAMP:       YES       FINWAIT2TIME:      600
TTLS:               NO
GLOBAL CONFIGURATION INFORMATION:
TCPIPSTATS: NO   ECSALIMIT: 0000000K  POOLLIMIT: 0000000K
```

```
MLSCHKTERM: NO   XCFGRPID:           IQDVLANID: 0
SEGOFFLOAD: YES  SYSPLEXWLMPOLL: 060  MAXRECS:   100
EXPLICITBINDPORTRANGE:  07000-07002   IQDMULTIWRITE: NO  1
SYSPLEX MONITOR:
  TIMERSECS: 0060  RECOVERY: NO   DELAYJOIN: NO   AUTOREJOIN: NO
  MONINTF:   NO    DYNROUTE: NO
ZIIP:
  IPSECURITY: NO   IQDIOMULTIWRITE: NO  2
```

In this example, the numbers correspond to the following information:

1. No IQDMUTIWRITE

2. No zIIP involved

The CPU RMF and WLM RMF report (service class SYSSTC for TCP/IP and FTP and service class BATCHHI for the FTP batch job) are shown in Figure A-2.

```
-CPU  2097   MODEL  714   H/W MODEL  E26  SEQUENCE CODE 000000000001DE50 IPERDISPATCH=NO
0---CPU---    ---------------- TIME % ----------------     LOG PROC   --I/O INTERRUPTS--
 NUM  TYPE    ONLINE    LPAR BUSY    MVS BUSY   PARKED     SHARE %      RATE  % VIA TPI
  0    CP     100.00     5.60         5.72      ------      11.1        531.8     0.34
  1    CP     100.00     5.60         5.70      ------      11.1        557.6     0.28
TOTAL/AVERAGE           5.60         5.71                  22.2        1089     0.31  a
0 6    IIP    100.00     0.00         0.00      ------      50.0
 TOTAL/AVERAGE           0.00         0.00                  50.0

REPORT BY: POLICY=POL01      WORKLOAD=SYSTEM     SERVICE CLASS=SYSSTC     RESOURCE GROUP=*NONE
                                                 CRITICAL     =NONE
-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---   SERVICE TIME  ---APPL %---
AVG      26.00  ACTUAL                 0  SSCHRT   5.1  IOC     15770   CPU     5.820  CP      5.82   b
MPL      26.00  EXECUTION              0  RESP     1.7  CPU     2601K   SRB    27.389  AAPCP   0.00
ENDED        0  QUEUED                 0  CONN     1.4  MSO         0   RCT     0.000  IIPCP   0.00
END/S     0.00  R/S AFFIN              0  DISC     0.1  SRB    12241K   IIT     1.710
#SWAPS       0  INELIGIBLE             0  Q+PEND   0.2  TOT    14858K   HST     0.000  AAP     0.00
EXCTD        0  CONVERSION             0  IOSQ     0.0  /SEC    24763   AAP     0.000  IIP     0.00   c
AVG ENC   0.00  STD DEV                0                                IIP     0.000
REM ENC   0.00                                          ABSRPTN   952
MS ENC    0.00                                          TRX SERV  952

REPORT BY: POLICY=POL01      WORKLOAD=TEST01     SERVICE CLASS=BATCHHI    RESOURCE GROUP=*NONE
                                                 CRITICAL     =NONE
-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---   SERVICE TIME  ---APPL %---
AVG       0.94  ACTUAL           9.22.003  SSCHRT 484.4  IOC      2907K   CPU    14.118  CP      3.02   d
MPL       0.94  EXECUTION        9.21.259  RESP     1.0  CPU      6310K   SRB     3.142  AAPCP   0.00
ENDED        1  QUEUED                744  CONN     0.8  MSO          0   RCT     0.000  IIPCP   0.00
END/S     0.00  R/S AFFIN               0  DISC     0.0  SRB      1404K   IIT     0.877
#SWAPS       0  INELIGIBLE              0  Q+PEND   0.1  TOT     10621K   HST     0.000  AAP     0.00
EXCTD        0  CONVERSION              0  IOSQ     0.0  /SEC     17702   AAP     0.000  IIP     0.00   e
AVG ENC   0.00  STD DEV                 0                                IIP     0.000
REM ENC   0.00                                          ABSRPTN    19K
MS ENC    0.00                                          TRX SERV   19K
```

Figure A-2 Test result for no IDQMULTIWRITE and no ZIIP

The total CPU usage for LPAR is 5.60 a. The APPL% for service class SYSSTC is 5.82 for CP and 0 for IIP (b and c). The APPL% for service class BATCHHI is 3.02 for CP and 0 for IIP d and e.

In the next test, we used the parameters shown in Example A-4 in profile TCP/IP for LPAR A23 (SC30) and LPAR A24 (SC31).

Example A-4 TCP/IP profile extract

```
TCPIPA.TCPPARMS(PROFA30)

GLOBALCONFIG NOTCPIPSTATISTICS IQDMULTIWRITE

DEVICE IUTIQDF4   MPCIPA
LINK   IUTIQDF4L  IPAQIDIO    IUTIQDF4

HOME
   10.1.4.11     IUTIQDF4L

START IUTIQDF4

**********************************************************************************
*********
TCPIPA.TCPPARMS(PROFA31)

GLOBALCONFIG NOTCPIPSTATISTICS IQDMULTIWRITE

DEVICE IUTIQDF4   MPCIPA
LINK   IUTIQDF4L  IPAQIDIO    IUTIQDF4

HOME
   10.1.4.12     IUTIQDF4L

START IUTIQDF4
```

To verify the options in effect, we issued the following command (as shown in Example A-5):

```
D TCPIP,TCPIPA,N,CONFIG
```

Example A-5 Output of NETSTAT CONFIG command

```
TCP CONFIGURATION TABLE:
DEFAULTRCVBUFSIZE:  00262144  DEFAULTSNDBUFSIZE: 00262144
DEFLTMAXRCVBUFSIZE: 00524288  SOMAXCONN:         0000000010
MAXRETRANSMITTIME:  120.000   MINRETRANSMITTIME: 0.500
ROUNDTRIPGAIN:      0.125     VARIANCEGAIN:      0.250
VARIANCEMULTIPLIER: 2.000     MAXSEGLIFETIME:    30.000
DEFAULTKEEPALIVE:   00000120  DELAYACK:          YES
RESTRICTLOWPORT:    NO        SENDGARBAGE:       NO
TCPTIMESTAMP:       YES       FINWAIT2TIME:      600
TTLS:               NO
GLOBAL CONFIGURATION INFORMATION:
TCPIPSTATS: NO   ECSALIMIT: 0000000K  POOLLIMIT: 0000000K
MLSCHKTERM: NO   XCFGRPID:            IQDVLANID: 0
SEGOFFLOAD: YES  SYSPLEXWLMPOLL: 060  MAXRECS:   100
EXPLICITBINDPORTRANGE:  07000-07002   IQDMULTIWRITE: YES 1
SYSPLEX MONITOR:
  TIMERSECS: 0060  RECOVERY: NO   DELAYJOIN: NO   AUTOREJOIN: NO
  MONINTF:   NO    DYNROUTE: NO
ZIIP:
  IPSECURITY: NO   IQDIOMULTIWRITE: NO   2
```

In this example, the numbers correspond to the following information:

1. HiperSockets Multiple Write enabled

2. zIIP not used

The CPU RMF and WLM RMF report (service class SYSSTC for TCP/IP and FTP and service class BATCHHI for the FTP batch job) are shown in Figure A-3.

```
-CPU  2097   MODEL  714   H/W MODEL  E26  SEQUENCE CODE 000000000001DE50    HIPERDISPATCH=NO
0---CPU---    ---------------- TIME % ----------------     LOG PROC      --I/O INTERRUPTS--
 NUM  TYPE    ONLINE    LPAR BUSY    MVS BUSY   PARKED     SHARE %       RATE     % VIA TPI
  0    CP     100.00     5.13         5.26      ------      11.1         282.3     0.61
  1    CP     100.00     5.12         5.23      ------      11.1         286.4     0.43
 TOTAL/AVERAGE           5.12         5.25                  22.2         568.7     0.52          a
0 6    IIP    100.00     0.00         0.00      ------      50.0
 TOTAL/AVERAGE           0.00         0.00                  50.0

REPORT BY: POLICY=POL01      WORKLOAD=SYSTEM      SERVICE CLASS=SYSSTC     RESOURCE GROUP=*NONE
                                                  CRITICAL     =NONE

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---   SERVICE TIME   ---APPL %---
AVG      26.00  ACTUAL                 0  SSCHRT   4.7  IOC     15298   CPU    5.755  CP      4.42   b
MPL      26.00  EXECUTION              0  RESP     1.8  CPU     2572K   SRB   20.716  AAPCP   0.00
ENDED        0  QUEUED                 0  CONN     1.5  MSO         0   RCT    0.000  IIPCP   0.00
END/S     0.00  R/S AFFIN              0  DISC     0.1  SRB     9258K   IIT    0.051
#SWAPS       0  INELIGIBLE             0  Q+PEND   0.2  TOT    11846K   HST    0.000  AAP     0.00
EXCTD        0  CONVERSION             0  IOSQ     0.0  /SEC    19742   AAP    0.000  IIP     0.00    c
AVG ENC   0.00  STD DEV                0                                IIP    0.000
REM ENC   0.00                                          ABSRPTN   759
MS ENC    0.00                                          TRX SERV  759

REPORT BY: POLICY=POL01      WORKLOAD=TEST01      SERVICE CLASS=BATCHHI    RESOURCE GROUP=*NONE
                                                  CRITICAL     =NONE

-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---   SERVICE TIME   ---APPL %---
AVG       0.93  ACTUAL          9.18.587  SSCHRT 484.3  IOC     2907K   CPU   15.033  CP      3.18    d
MPL       0.93  EXECUTION       9.18.304  RESP     1.0  CPU     6719K   SRB    3.169  AAPCP   0.00
ENDED        1  QUEUED               283  CONN     0.8  MSO         0   RCT    0.000  IIPCP   0.00
END/S     0.00  R/S AFFIN              0  DISC     0.0  SRB     1416K   IIT    0.884
#SWAPS       0  INELIGIBLE             0  Q+PEND   0.1  TOT    11042K   HST    0.000  AAP     0.00
EXCTD        0  CONVERSION             0  IOSQ     0.0  /SEC    18403   AAP    0.000  IIP     0.00    e
AVG ENC   0.00  STD DEV                0                                IIP    0.000
REM ENC   0.00                                          ABSRPTN    20K
MS ENC    0.00                                          TRX SERV   20K
```

Figure A-3 Test result for IDQMULTIWRITE and no ZIIP

The total CPU usage for LPAR is 5.12 **a**. APPL% for service class SYSSTC is 4.42 for CP and 0 for IIP (**b** and **c**). APPL% for service class BATCHHI is 3.18 for CP and 0 for IIP (**d** and **e**).

In the third test, we use the parameters shown in Example A-6 in profile TCP/IP for LPAR A23 (SC30) and LPAR A24 (SC31).

Example A-6 Extract of each profile TCP/IP

```
TCPIPA.TCPPARMS(PROFA30)

GLOBALCONFIG NOTCPIPSTATISTICS IQDMULTIWRITE ZIIP IQDIOMULTIWRITE

DEVICE IUTIQDF4   MPCIPA
LINK   IUTIQDF4L  IPAQIDIO    IUTIQDF4

HOME
   10.1.4.11     IUTIQDF4L

START IUTIQDF4

*********************************************************************************
```

```
TCPIPA.TCPPARMS(PROFA31)

GLOBALCONFIG NOTCPIPSTATISTICS IQDMULTIWRITE ZIIP IQDIOMULTIWRITE

DEVICE IUTIQDF4   MPCIPA
LINK   IUTIQDF4L  IPAQIDIO    IUTIQDF4

HOME
   10.1.4.12     IUTIQDF4L

START IUTIQDF4
```

To verify the options in effect, we issued the following command (as shown in Example A-7):

```
D TCPIP,TCPIPA,N,CONFIG
```

Example A-7 Output of NETSTAT CONFIG command

```
TCP CONFIGURATION TABLE:
DEFAULTRCVBUFSIZE:  00262144  DEFAULTSNDBUFSIZE: 00262144
DEFLTMAXRCVBUFSIZE: 00524288  SOMAXCONN:         0000000010
MAXRETRANSMITTIME:  120.000   MINRETRANSMITTIME: 0.500
ROUNDTRIPGAIN:      0.125     VARIANCEGAIN:      0.250
VARIANCEMULTIPLIER: 2.000     MAXSEGLIFETIME:    30.000
DEFAULTKEEPALIVE:   00000120  DELAYACK:          YES
RESTRICTLOWPORT:    NO        SENDGARBAGE:       NO
TCPTIMESTAMP:       YES       FINWAIT2TIME:      600
TTLS:               NO
GLOBAL CONFIGURATION INFORMATION:
TCPIPSTATS: NO   ECSALIMIT: 0000000K  POOLLIMIT: 0000000K
MLSCHKTERM: NO   XCFGRPID:            IQDVLANID: 0
SEGOFFLOAD: YES  SYSPLEXWLMPOLL: 060  MAXRECS:   100
EXPLICITBINDPORTRANGE:  07000-07002   IQDMULTIWRITE: YES  1
SYSPLEX MONITOR:
  TIMERSECS: 0060  RECOVERY: NO   DELAYJOIN: NO   AUTOREJOIN: NO
  MONINTF:   NO    DYNROUTE: NO
ZIIP:
  IPSECURITY: NO   IQDIOMULTIWRITE: YES  2
```

In this example, the numbers correspond to the following information:

1. HiperSockets Multiple Write enabled

2. zIIP Adjunct Processor used

Then we used the following command to see the status of the device used (as shown in Example A-8):

```
D TCPIP,TCPIPA,N,DEV
```

Example A-8 Output of NETSTAT DEVICE command

```
DEVNAME: IUTIQDF4         DEVTYPE: MPCIPA
  DEVSTATUS: READY
  LNKNAME: IUTIQDF4L         LNKTYPE: IPAQIDIO   LNKSTATUS: READY
    IPBROADCASTCAPABILITY: NO
    CFGROUTER: NON                  ACTROUTER: NON
    ARPOFFLOAD: YES                 ARPOFFLOADINFO: YES
```

```
  ACTMTU: 8192
  VLANID: NONE
  READSTORAGE: GLOBAL (2048K)
  SECCLASS: 255                    MONSYSPLEX: NO
  IQDMULTIWRITE: ENABLED (ZIIP)
BSD ROUTING PARAMETERS:
  MTU SIZE: 8192              METRIC: 80
  DESTADDR: 0.0.0.0           SUBNETMASK: 255.255.255.0
```

The CPU RMF and WLM RMF report (service class SYSSTC for TCP/IP and FTP and service class BATCHHI for the FTP batch job) are shown in Figure A-4.

```
-CPU  2097   MODEL  714   H/W MODEL  E26  SEQUENCE CODE 000000000001DE50    HIPERDISPATCH=NO
0---CPU---    ---------------- TIME % ----------------     LOG PROC      --I/O INTERRUPTS--
 NUM  TYPE    ONLINE    LPAR BUSY     MVS BUSY   PARKED     SHARE %      RATE     % VIA TPI
  0    CP     100.00     4.63         4.69      ------       11.1        285.7     0.56
  1    CP     100.00     4.63         4.68      ------       11.1        282.1     0.46
 TOTAL/AVERAGE           4.63         4.68                   22.2        567.9     0.51        a
0 6    IIP    100.00     3.15         3.14      ------       50.0
 TOTAL/AVERAGE           3.15         3.14                   50.0                              f

 REPORT BY: POLICY=POL01      WORKLOAD=SYSTEM     SERVICE CLASS=SYSSTC     RESOURCE GROUP=*NONE
                                                  CRITICAL     =NONE

 -TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---   SERVICE TIME  ---APPL %---
 AVG      26.00  ACTUAL                 0  SSCHRT  4.8  IOC      15346   CPU   24.302  CP      2.92   b
 MPL      26.00  EXECUTION              0  RESP    1.8  CPU     10861K   SRB   11.555  AAPCP   0.00
 ENDED        0  QUEUED                 0  CONN    1.5  MSO          0   RCT    0.000  IIPCP   0.00
 END/S     0.00  R/S AFFIN              0  DISC    0.1  SRB      5164K   IIT    0.051
 #SWAPS       0  INELIGIBLE             0  Q+PEND  0.2  TOT     16041K   HST    0.000  AAP     0.00
 EXCTD        0  CONVERSION             0  IOSQ    0.0  /SEC     26735   AAP    0.000  IIP     3.06   c
 AVG ENC   0.00  STD DEV                0                                IIP   18.374
 REM ENC   0.00                                         ABSRPTN  1028
 MS ENC    0.00                                         TRX SERV 1028

REPORT BY: POLICY=POL01      WORKLOAD=TEST01     SERVICE CLASS=BATCHHI     RESOURCE GROUP=*NONE
                                                 CRITICAL     =NONE
-TRANSACTIONS-  TRANS-TIME HHH.MM.SS.TTT  --DASD I/O--  ---SERVICE---   SERVICE TIME  ---APPL %---
AVG       0.93  ACTUAL           9.21.352  SSCHRT 484.4  IOC      2907K  CPU   15.166  CP      3.18    d
MPL       0.93  EXECUTION        9.20.829  RESP    1.0  CPU      6778K  SRB    3.006  AAPCP   0.00
ENDED        1  QUEUED                523  CONN    0.8  MSO          0  RCT    0.000  IIPCP   0.00
END/S     0.00  R/S AFFIN               0  DISC    0.0  SRB      1344K  IIT    0.880
#SWAPS       0  INELIGIBLE              0  Q+PEND  0.1  TOT     11029K  HST    0.000  AAP     0.00
EXCTD        0  CONVERSION              0  IOSQ    0.0  /SEC     18382  AAP    0.000  IIP     0.00    e
AVG ENC   0.00  STD DEV                 0                               IIP    0.000
REM ENC   0.00                                          ABSRPTN    20K
MS ENC    0.00                                          TRX SERV   20K
```

Figure A-4 Test result for IDQMULTIWRITE and ZIIP

The total CPU usage for LPAR is 4.63 a and 3.15 for ZIIP f. The APPL% for service class SYSSTC is 2.92 for CP and 3.06 for IIP (b and c). The APPL% for service class BATCHHI is 3.18 for CP and 0 for IIP (d and e).

In summary, with the enhancement of HiperSockets Multiple Write and zIIP-Assisted HiperSockets Multiple Write, you can see a performance improvement and reduction in CPU utilization for large outbound messages. See Table A-1 and Table A-2.

Table A-1 CPU utilization

OPTIONS	LAPR BUSY		MVS BUSY	
	CP	zIIP	CP	zIIP
No IQDMULTIWRITE	5.60	0.00	5.71	0.00
IQDMULTIWRITE	5.12	0.00	5.25	0.00
IQDMULTIWRITE + ZIIP	4.63	3.15	4.68	3.14

Table A-2 APPL% in WLM report

OPTIONS	SYSSTC APPL%		BATCHHI APPL%	
	CP	zIIP	CP	zIIP
NO IDQMULTIWRITE	5.82	0.00	3.02	0.00
IQDMULTIWRITE	4.42	0.00	3.18	0.00
IQDMULTIWRITE + ZIIP	2.92	3.06	3.18	0.00

B

Our implementation environment

We wrote the four *z/OS Communications Server TCP/IP Implementation* books at the same time. Given the complexity of this project, we needed to be creative in organizing the test environment so that each team could work with minimal coordination and interference from the other teams. In this appendix, we show the complete environment that we used for the four books as well as the environment that we used for this book.

The environment used for all four books

To enable concurrent work on each of the four books, we set up and shared the test environment illustrated in Figure B-1.

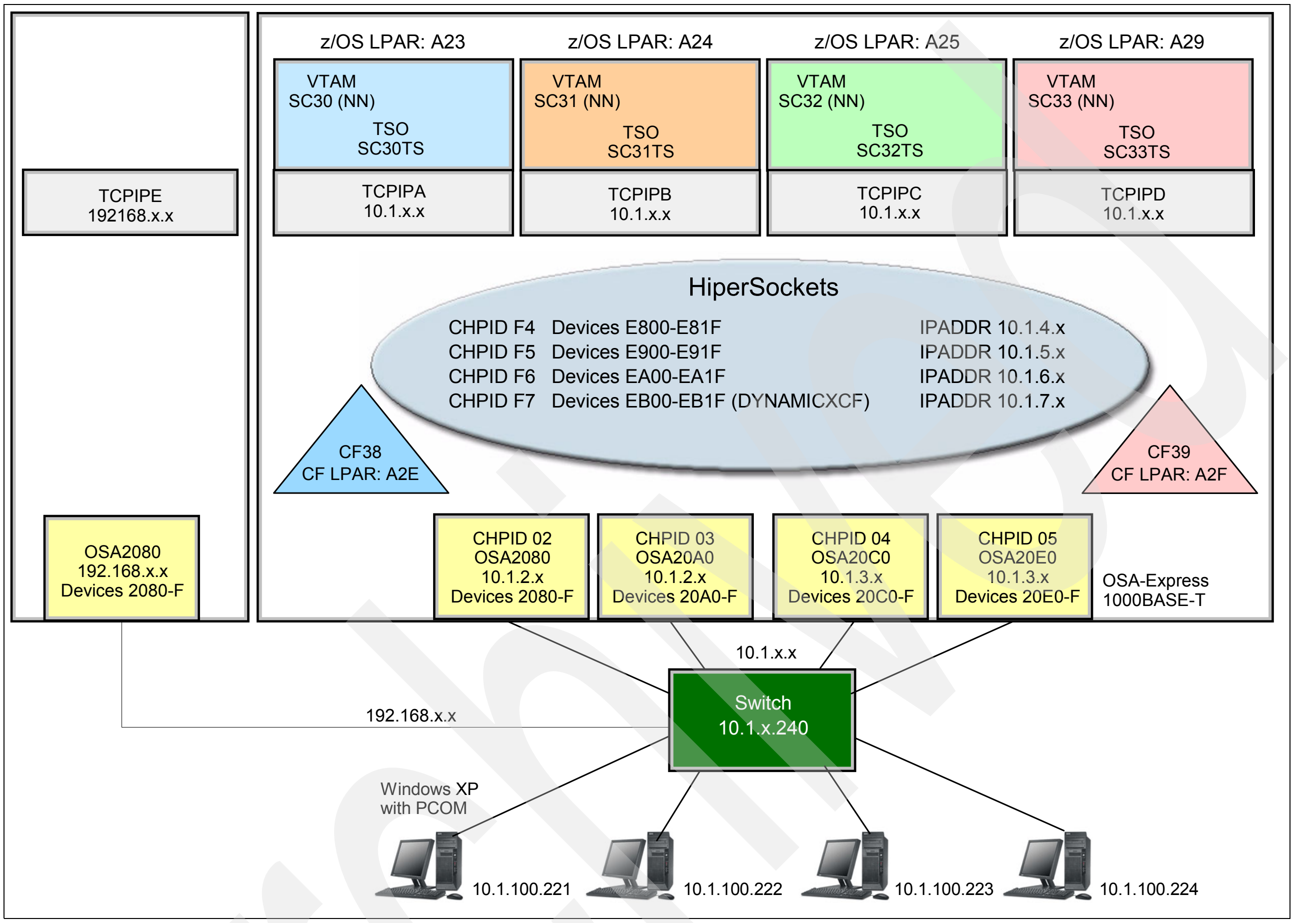


Figure B-1 Our implementation environment

We wrote our books (and ran our implementation scenarios) using four logical partitions (LPARs) on an IBM System z10™ EC (referred to as LPARs A23, A24, A25, and A29). We implemented and started one TCP/IP stack on each LPAR. Each LPAR shared the following resources:

- HiperSockets inter-server connectivity
- Coupling Facility connectivity (CF38 and CF39) for Parallel Sysplex scenarios
- Eight OSA-Express3 1000BASE-T Ethernet ports connected to a switch

Finally, we shared four Windows workstations, representing corporate network access to the z/OS networking environment. The workstations are connected to the switch. To verify our scenarios, we used applications such as TN3270 and FTP.

The IP addressing scheme that we used allowed us to build multiple subnetworks so that we would not impede ongoing activities from other team members.

VLANs were also defined to isolate the TCP/IP stacks and portions of the LAN environment (Figure B-2).

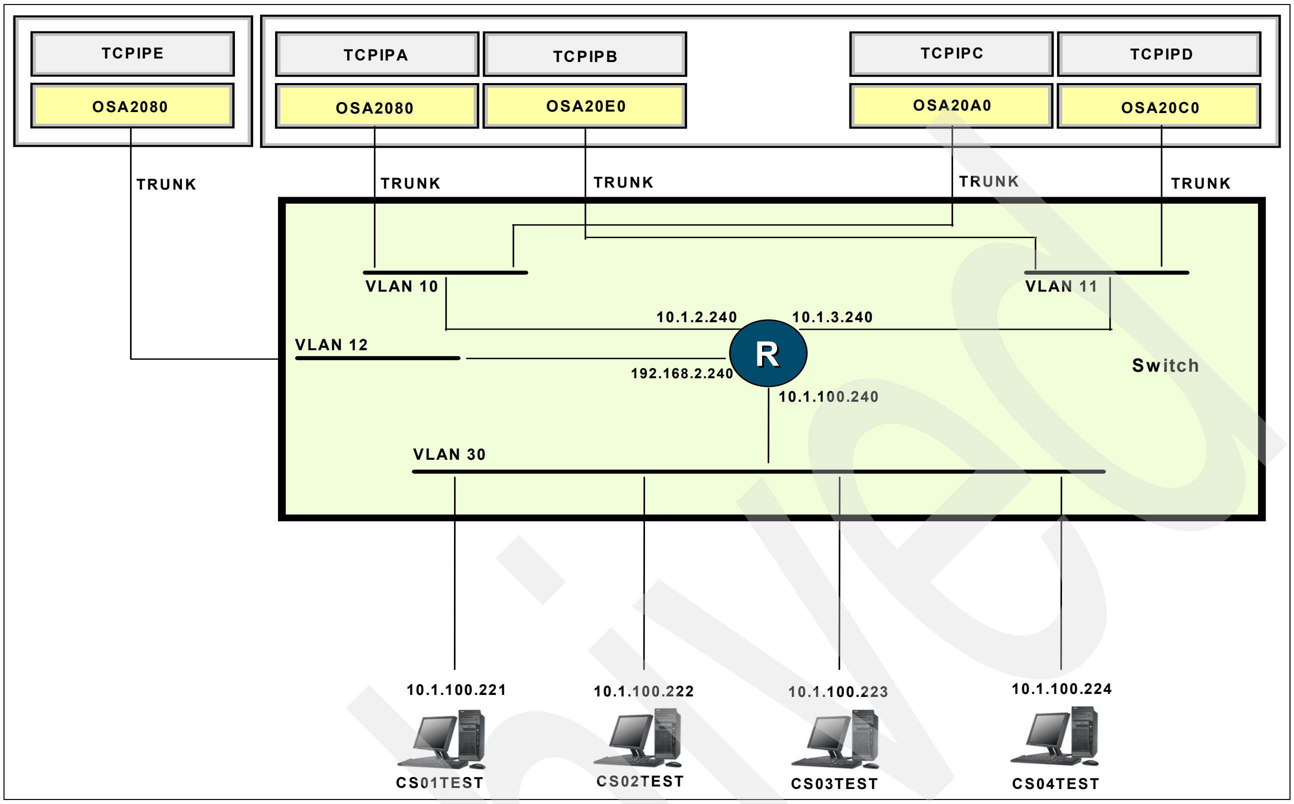


Figure B-2 LAN configuration - VLAN and IP addressing

Our focus for this book

Figure B-3 depicts the environment that we worked with, as required for our basic function implementation scenarios.

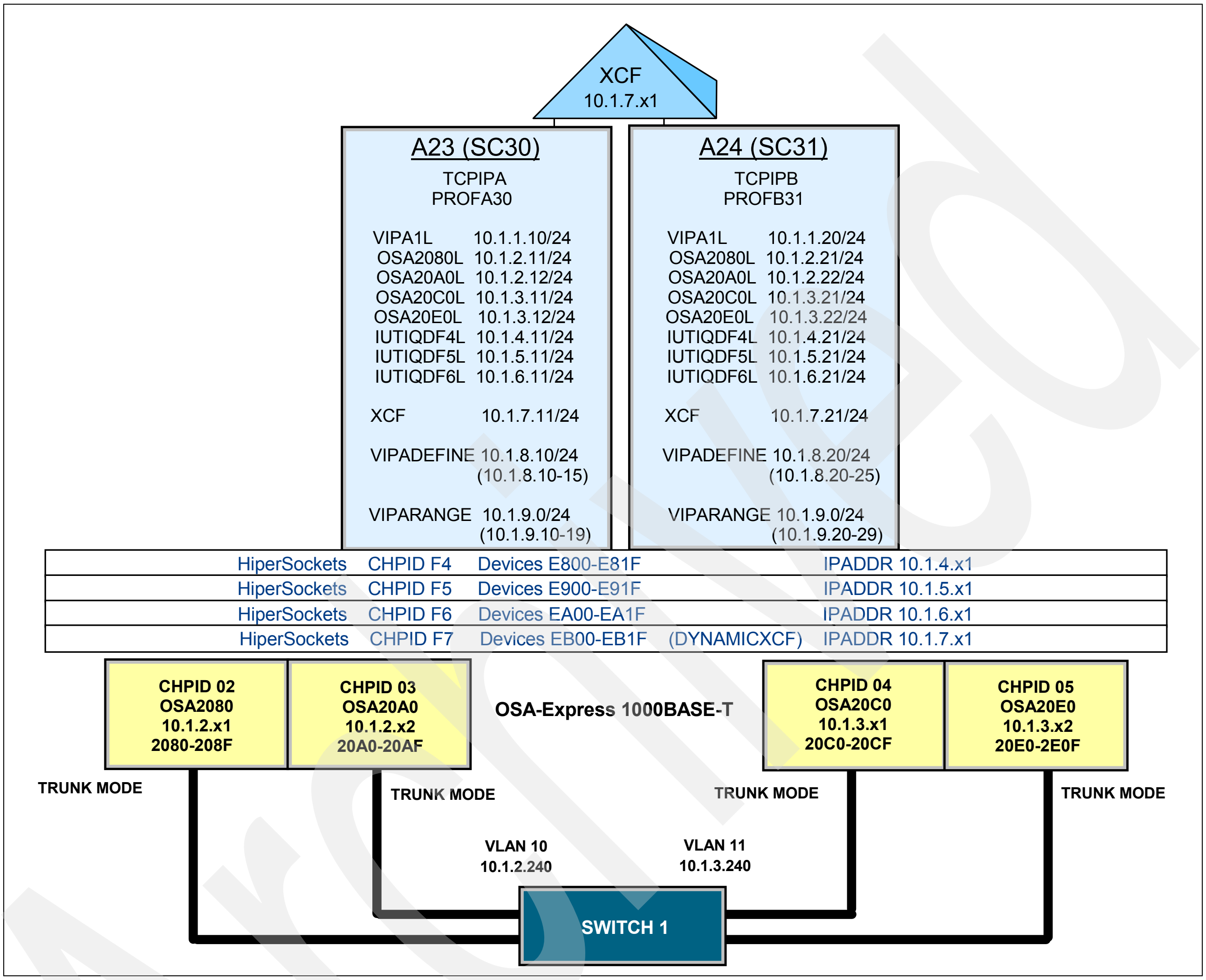


Figure B-3 Our environment for this book

Related publications

We consider the publications that we list in this section particularly suitable for a more detailed discussion of the topics that we cover in this book.

IBM Redbooks publications

For information about ordering these publications, see "How to get IBM Redbooks publications" on page 309. Note that some of the documents referenced here might be available in softcopy only.

- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798
- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7799
- *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7801
- *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *IP Network Design Guide*, SG24-2580
- *OSA-Express Implementation Guide*, SG24-5948
- *HiperSockets Implementation Guide*, SG24-6816
- *SNA in a Parallel Sysplex Environment*, SG24-2113
- *z/OS Infoprint Server Implementation*, SG24-6234
- *z/OS Security Services Update*, SG24-6448
- *Implementing PKI Services on z/OS*, SG24-6968

Other publications

The following publications are also relevant as further information sources:

- *z/OS XL C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS MVS IPCS Commands*, SA22-7594
- *z/OS MVS System Commands*, SA22-7627
- *z/OS TSO/E Command Reference*, SA22-7782
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS Communications Server: CSM Guide*, SC31-8808
- *z/OS Communications Server: New Function Summary*, GC31-8771
- *z/OS Communications Server: Quick Reference*, SX75-0124
- *z/OS Communications Server: IP and SNA Codes*, SC31-8791

- *z/OS Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS MVS IPCS Commands*, SA22-7594
- *z/OS Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS Communications Server: IP Messages Volume 2 (EZB, EZD)*, SC31-8784
- *z/OS Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786
- *z/OS Communications Server: IP Programmer's Guide and Reference*, SC31-8787
- *z/OS Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS Communications Server: IP Sockets Application Programming Interface Guide and Reference*, SC31-8788
- *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS Communications Server: IPv6 Network and Application Design Guide*, SC31-8885
- *z/OS Migration*, GA22-7499
- *z/OS MVS System Commands*, SA22-7627
- *OSA-Express Customer's Guide and Reference*, SA22-7935
- *z/OS Communications Server: SNA Operation*, SC31-8779
- *z/OS TSO/E Command Reference*, SA22-7782
- *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803
- *z/OS UNIX System Services Command Reference*, SA22-7802
- *z/OS UNIX System Services File System Interface Reference*, SA22-7808
- *z/OS UNIX System Services Messages and Codes*, SA22-7807
- *z/OS UNIX System Services Parallel Environment: Operation and Use*, SA22-7810
- *z/OS UNIX System Services Programming Tools*, SA22-7805
- *z/OS UNIX System Services Planning*, GA22-7800
- *z/OS UNIX System Services User's Guide*, SA22-7801

Online resources

The following Web sites are also relevant as further information sources:

- z/OS Communications Server product support

  `http://www-306.ibm.com/software/network/commserver/zos/support/`
- Mainframe networking

  `http://www.ibm.com/servers/eserver/zseries/networking/`
- z/OS Communications Server product overview

  `http://www.ibm.com/software/network/commserver/zos/`
- z/OS Communications Server publications

  `http://www-03.ibm.com/systems/z/os/zos/bkserv/r9pdf/#commserv`

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy books or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

E

F

G

H

I

J

IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 3:

High Availability, Scalability, and Performance

Discusses z/OS Communications Server TCP/IP high availability capabilities

Provides insights into performance and tuning

Includes z/OS Communications Server TCP/IP high availability implementation examples

For more than 40 years, IBM mainframes have supported an extraordinary portion of the world's computing work, providing centralized corporate databases and mission-critical enterprise-wide applications. The IBM System z provides world class and state-of-the-art support for the TCP/IP Internet protocol suite.

TCP/IP is a large and evolving collection of communication protocols managed by the Internet Engineering Task Force (IETF), an open, volunteer, organization. Because of its openness, the TCP/IP protocol suite has become the foundation for the set of technologies that form the basis of the Internet. The convergence of IBM mainframe capabilities with Internet technology, connectivity, and standards (particularly TCP/IP) is dramatically changing the face of information technology and driving requirements for ever more secure, scalable, and highly available mainframe TCP/IP implementations.

The *IBM z/OS Communications Server TCP/IP Implementation* series provides understandable, step-by-step guidance about how to enable the most commonly used and important functions of z/OS Communications Server TCP/IP.

In this IBM Redbooks publication, we discuss Virtual IP Addressing (VIPA), how to use VIPA for high availability, and a number of different workload balancing approaches that you can use with the z/OS Communications Server. We also explain the optimized Sysplex Distributor intra-sysplex load balancing and highlight the most important tuning parameters. For more specific information about z/OS Communications Server standard applications, high availability, and security, refer to the other volumes in the series.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7800-00 ISBN 073843390X