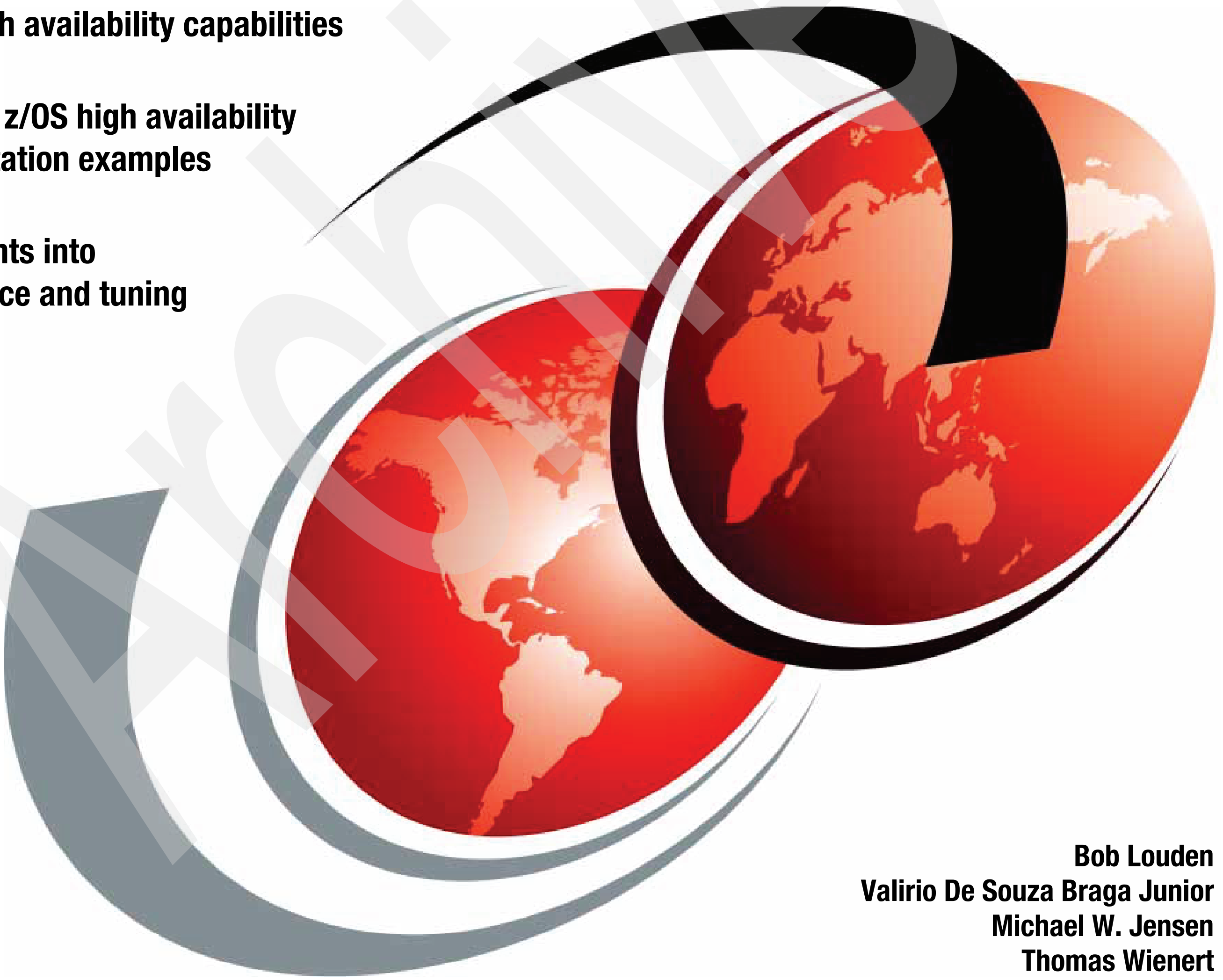
IBM

Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 3 High Availability, Scalability, and Performance

Understand important CS for z/OS TCP/IP high availability capabilities

See CS for z/OS high availability implementation examples

Gain insights into performance and tuning



Bob Louden
Valirio De Souza Braga Junior
Michael W. Jensen
Thomas Wienert

Redbooks

ibm.com/redbooks

International Technical Support Organization

Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 3 - High Availability, Scalability, and Performance

April 2006

SG24-7171-00

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

First Edition (April 2006)

This edition applies to version 1, release 7, of the z/OS Communications Server.

Contents

Notices vii
Trademarks viii

Preface ix
Our implementation environment ix
The environment used for all four books x
Our focus for this book xi
The team that wrote this redbook xi
Become a published author xiii
Comments welcome xiii

Chapter 1. Introduction 1
1.1 Fundamental technologies for z/OS TCP/IP availability 3
1.1.1 Single z/OS system availability 3
1.1.2 z/OS Parallel Sysplex availability 3
1.1.3 Virtual IP Addressing 4
1.1.4 z/OS network connectivity and dynamic routing 5
1.1.5 Single-instance and multiple-instance applications 7
1.1.6 Balancing workload across multiple application instances 8
1.2 Quick start table 11

Chapter 2. Virtual IP Addressing 13
2.1 Basic concepts 14
2.1.1 Application availability modes 16
2.1.2 DVIPA and distributed DVIPA 16
2.1.3 VIPARoute function 25
2.1.4 New DVIPA commands 26
2.2 Importance of VIPA 30
2.3 Dynamic VIPA example 31
2.3.1 Implementation 32
2.3.2 Automatic VIPA takeover and takeback 34

Chapter 3. VIPA without dynamic routing 39
3.1 Basic concepts 40
3.2 Design example 41
3.2.1 Implementation 41
3.2.2 Verification 44
3.3 High-availability scenarios 46
3.3.1 Adapter interface failure 46
3.3.2 Application movement 47
3.3.3 Stack failure 49
3.3.4 Local network failure 50
3.4 Debugging tips 52
3.5 ARP takeover 53

Chapter 4. VIPA with dynamic routing 59
4.1 Basic concepts 60
4.1.1 Dynamic routing and OMPROUTE 60
4.1.2 Advertisement of VIPA addresses 60

4.1.3 Multiple links between IP nodes and LPARs. 63
4.1.4 Generic OSPF address definitions 67
4.2 Design example for dynamic routing. 68
4.2.1 PROFILE statements 69
4.2.2 OMPROUTE definitions 70
4.2.3 Router definitions 72
4.2.4 Verification 72

Chapter 5. Internal application workload balancing 83
5.1 Basic concepts 84
5.1.1 Sysplex Distributor - Principles of operation 85
5.1.2 Sysplex Distributor configured as a Cisco Service Manager 86
5.1.3 Sysplex Distributor and Quality of Service (QoS) policy 88
5.1.4 Portsharing 96
5.1.5 Optimized routing 97
5.1.6 Improved workload distribution 100
5.2 Importance of internal application workload balancing 106
5.3 Design examples. 106
5.3.1 Sysplex Distributor using BASEWLM 107
5.3.2 Sysplex Distributor using round-robin 109
5.3.3 Sysplex Distributor using server-specific WLM. 110
5.3.4 Sysplex Distributor configured as Cisco Service Manager 111
5.3.5 Portsharing using SHAREPORTWLM 113
5.4 Implementations 114
5.4.1 Sysplex Distributor using SERVERWLM and VIPAROUTE 114
5.4.2 Sysplex Distributor configured as Cisco Service Manager 122
5.4.3 Multiple servers sharing a port using SHAREPORTWLM. 130
5.4.4 Problem determination 138

Chapter 6. External application workload balancing 139
6.1 Basic concepts 140
6.1.1 z/OS Load Balancer Advisor. 142
6.1.2 Server/Application State Protocol (SASP) 144
6.1.3 External load balancer without LBA/SASP 145
6.1.4 External load balancer with LBA/SASP. 146
6.2 Importance of external application workload balancing 147
6.3 Design examples. 148
6.3.1 External load balancer without LBA/SASP 148
6.3.2 External load balancer with LBA/SASP. 149
6.3.3 Recommendations 150
6.4 Implementations 150
6.4.1 External load balancer without LBA/SASP 150
6.4.2 External load balancer with LBA/SASP. 159

Chapter 7. Performance and tuning 177
7.1 General performance considerations 178
7.2 TCP/IP configuration files 179
7.2.1 Appropriate MTU for devices 179
7.2.2 Tracing 181
7.3 z/OS UNIX System Services tuning 181
7.4 Storage requirements 182
7.4.1 TCP and UDP buffer sizes 182
7.4.2 CSM storage usage 182
7.4.3 VTAM buffer settings 183

7.5 Application performance and capacity 184
7.6 Communication Server for z/OS performance highlights 187
7.6.1 Communications Server V1R5 performance highlights 187
7.6.2 Communications Server V1R6 performance highlights 187
7.6.3 Communications Server V1R7 performance highlights 188
7.7 TCP/IP performance quick checklist 188

Related publications 191
IBM Redbooks 191
Other publications 191
Online resources 193
How to get IBM Redbooks 193
Help from IBM 193

Index 195

Notices

Trademarks

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

eserver®	ESCON®	RMF™
eserver®	FICON®	RS/6000®
Redbooks (logo) ™	HiperSockets™	S/390®
pSeries®	IBM®	System z9™
z/OS®	Lotus®	System/360™
zSeries®	MVS™	S/390®
z9™	Parallel Sysplex®	VTAM®
CICS®	Redbooks™	WebSphere®
DB2®	RACF®	

The following terms are trademarks of other companies:

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 z9™, the latest generation of the IBM distinguished family of mainframe systems, has come a long way from its IBM System/360™ heritage. Likewise, its 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.

This new and improved *Communications Server (CS) for z/OS TCP/IP Implementation* series provides easy-to-understand step-by-step how-to guidance on enabling the most commonly used and important functions of CS for z/OS TCP/IP.

In this IBM Redbook™ we begin with a discussion of Virtual IP Addresing (VIPA), a TCP/IP high-availability approach introduced by the z/OS Communications Server. We then proceed to show how VIPA can be used for high availability—both with and without using a dynamic routing protocol. Then we discuss a number of different workload balancing approaches that can be used with the z/OS Communications Server. 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 CS for z/OS base functions, standard applications, and security, please reference the other volumes in the series. These are:

- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 1 - Base Functions, Connectivity, and Routing*, SG24-7169
- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 2 - Standard Applications*, SG24-7170
- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 4 - Policy-Based Network Security*, SG24-7172

Our implementation environment

We wrote the four books in the Communications Server for z/OS V1R7 Implementation guides at the same time. Given the complexity of our test environment, we needed to be somewhat creative in organizing the environment so that each team could work with minimal coordination with (and interference from) the other teams.

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 1.

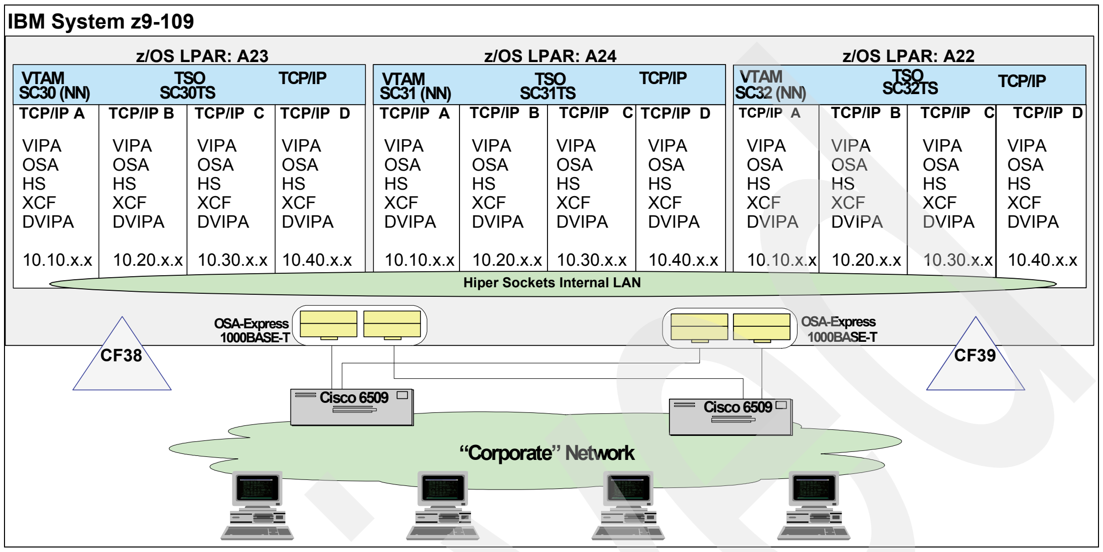


Figure 1 Our implementation environment

Our books were written (and implementation scenarios executed) using three logical partitions (LPARs) on an IBM System z9-109 (referred to as A22, A23, and A24). Because we worked on four books at the same time, we implemented *four* TCP/IP stacks on each LPAR (admittedly, a configuration not recommended for a production environment, but convenient for our purposes). Each LPAR shared:

- HiperSockets™ connectivity
- Coupling Facility connectivity (CF38 and CF39) for parallel sysplex scenarios
- Four OSA-Express2 1000BASE-T Ethernet ports cross-connected to a pair of Cisco 6509 switches

Finally, we shared ten workstations, representing corporate network access to the z/OS networking environment, for scenario verification (using applications such as TN3270 and FTP).

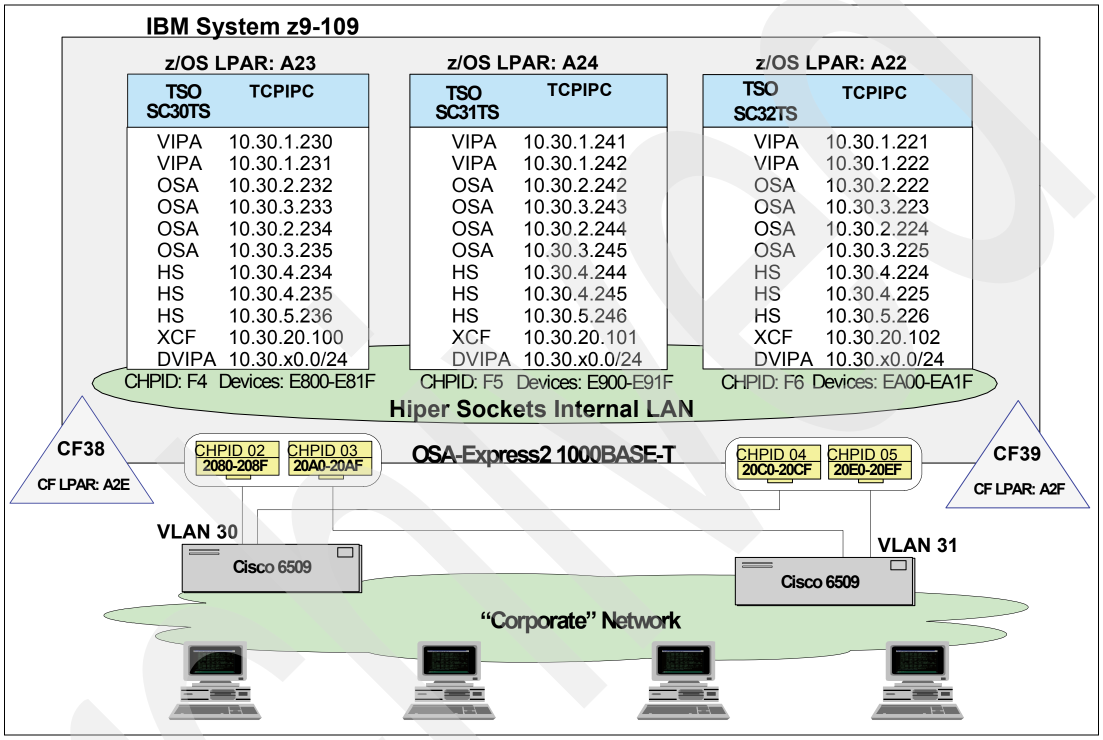
Our IP addressing convention is as follows:

- The first octet is the network (always 10 for our environment).
- The second octet is the VLAN (10,20,30,40) assigned to the stack. (Essentially, except when required by a specific implementation scenario, each team's stacks shared a common VLAN.)
- The third octet refers to the device:
  - The addresses with the third octet of 2 or 3 are defined to the OSA devices.
  - The addresses with the third octet of 4, 5, or 6 are defined to the HiperSocket devices.
- The last octet is made up as follows:
  - The first two digits are the LPAR number.
  - The last digit is the CHPID number.

Important: Our use of multiple TCP/IP stacks on each LPAR and our TCP/IP addressing were set up for our convenience and are not recommended approaches for your environment.

Our focus for this book

Given the above actual environment, the (simplified) environment that we had to work with for this book is illustrated in Figure 2.



TSO SC30TS	TCPIPC
VIPA	10.30.1.230
VIPA	10.30.1.231
OSA	10.30.2.232
OSA	10.30.3.233
OSA	10.30.2.234
OSA	10.30.3.235
HS	10.30.4.234
HS	10.30.4.235
HS	10.30.5.236
XCF	10.30.20.100
DVIPA	10.30.x0.0/24

TSO SC31TS	TCPIPC
VIPA	10.30.1.241
VIPA	10.30.1.242
OSA	10.30.2.242
OSA	10.30.3.243
OSA	10.30.2.244
OSA	10.30.3.245
HS	10.30.4.244
HS	10.30.4.245
HS	10.30.5.246
XCF	10.30.20.101
DVIPA	10.30.x0.0/24

TSO SC32TS	TCPIPC
VIPA	10.30.1.221
VIPA	10.30.1.222
OSA	10.30.2.222
OSA	10.30.3.223
OSA	10.30.2.224
OSA	10.30.3.225
HS	10.30.4.224
HS	10.30.4.225
HS	10.30.5.226
XCF	10.30.20.102
DVIPA	10.30.x0.0/24

Figure 2 Our environment for this book

For the purposes of this book we viewed the environment as three LPARs leveraging coupling facilities, HiperSockets, and OSA connectivity as required for our implementation scenarios.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Each of the books in our four-volume *Communications Server (CS) for z/OS TCP/IP Implementation* series was the result of very close cooperation and coordination across the entire team—a team with representation from seven different countries and with more than 200 years of combined information technology experience. The following authors worked on the books in this series. The main authors who wrote this particular book are Bob Louden, Valirio De Souza Braga Junior, Michael W. Jensen, and Thomas Wienert.

Valirio De Souza Braga Junior is a Senior IT Specialist in Brazil working for the IBM Support Center. He has eight years of experience in networking with expertise in VTAM®, TCP/IP, CS for z/OS, and OSA. He has written the *OSA-Express Implementation Guide*, SG24-5948-04, and is a Cisco Certified Network Associate (CCNA).

Michael W. Jensen is a Senior IT Specialist with IBM Global Services, Strategic Outsourcing in Denmark. Michael has 23 years of experience with networking (OSA, SNA, APPN, TCP/IP, Cisco SNASw, SNA-to-APPN migration, and SNA/IP Integration) and IBM Mainframe systems primarily focusing on the z/OS Communications Server. His areas of expertise include design and implementation of Cisco Content Switching and load balancing solutions for the z/OS sysplex environment using Cisco products.

Thomas Wienert is a Senior IT Specialist working for IBM z9 Field Technical Sales Support (FTSS) in Germany. He has over 20 years of experience with IBM networking. Thomas has been with IBM for 16 years and worked as a S/390® Systems Engineer in technical support and marketing. His areas of expertise include Communications Server for z/OS, Communication Controller for Linux®, OSA-Express, z/OS, Parallel Sysplex®, and zSeries®-related hardware. He co-authored the *OSA-Express Implementation Guide*, SG24-5948-04, and is a Cisco certified Associate (CCNA).

Bob Louden is a Consultant in the IBM Global Services, IT Services, Network Services Delivery practice. His twenty-three years as a networking professional with IBM have enabled him to develop strong professional and consulting skills, including leadership, project management, problem solving, and decision analysis. Bob's technical expertise includes wide area and local area networking and SNA and TCP/IP protocols. More important, however, is his ability to leverage technology understanding to develop business solutions.

Rama Ayyar is a Senior IT Specialist with the IBM Support Center in Sydney, Australia. Rama has over 22 years of experience with the MVS™ Operating System. His areas of expertise include TCP/IP, RACF®, DFSMS, z/OS Operating System, Configuration Management, Dump Analysis, and Disaster Recovery, and he has written six IBM Redbooks. Rama holds a master's degree in Computer Science from the Indian Institute of Technology, Kanpur, and has been in the computer industry for 34 years.

Octavio Ferreira is a Senior IT Specialist in IBM Brazil. He has 26 years of experience in IBM software support. His areas of expertise include z/OS Communications Server, SNA and TCP/IP, and the Communications Server in all platforms. For the last eight years he has worked at the Area Program Support Group providing guidance and support to clients and designing networking solutions such as SNA/TCP/IP integration, z/OS Connectivity, Enterprise Extender design and implementation, and SNA-to-APPN migration.

Sherwin Lake is an IT Specialist with IBM Global Services in Research Triangle Park, North Carolina. Sherwin has worked over the past 30 years in VSE/SP, VM, and MVS systems environments. During that time he was an Online and Batch Applications Developer, Product Support Specialist, Network Analyst, IT Specialist, and Manager. Before joining IBM, Sherwin managed the Technical Support group at the Trinidad and Tobago Telephone Company in Trinidad. Sherwin was a member of the team that migrated IBM from Office Vision to Lotus® Notes. He currently works with the Delivery part of IBM Global Services providing SNA and TCP/IP support to internal and external accounts. Sherwin holds a Bachelor of Science Degree in Computer Science/Math from the University of Miami.

Garth Madella is an Information Technology Specialist with IBM South Africa. He has 20 years of experience in the S/390® networking software field. He has worked with IBM for nine years. His areas of expertise include VTAM, SNA, TCP/IP, and sysplex. He has written extensively on TCP/IP and Enterprise Extender issues.

Joel Porterie is a Senior IT Specialist who has been with IBM France for 28 years. He works for Network and Channel Connectivity Services in the EMEA Product Support Group. His areas of expertise include z/OS, TCP/IP, VTAM, OSA-Express, and Parallel Sysplex for zSeries. He has taught OSA-Express and FICON® problem determination classes and provided on-site assistance in these areas in numerous countries. He also co-authored the IBM Redbooks *Using the IBM S/390 Application StarterPak*, SG24-2095; *OSA-Express Gigabit Ethernet Implementation Guide*, SG24-5443; *OSA-Express Implementation Guide*, SG24-5948; and *Introduction to the New Mainframe: Networking*, SG24-6772.

Marc Price is a Staff Software Engineer with IBM Software Group in Raleigh, North Carolina. Marc has over six years of experience with the Communications Server for z/OS as member of the organization that develops and services the Communications Server for z/OS. His areas of expertise include TCP/IP, security, z/OS dump analysis, and a variety of operating systems. Marc holds a bachelor's degree in Computer Science from Purdue University in Indiana, USA. Marc has 10 years of experience in the computer industry.

Larry Templeton is a Network Architect with IBM Global Services, Network Outsourcing. He has 36 years of experience in IBM mainframe and networking systems, consulting with clients throughout the United States. His current responsibilities include architecting mainframe IP connectivity solutions, designing inter-company Enterprise Extender configurations, and assisting clients with high-availability data center implementations.

Thanks to **Alfred Christensen**, Programming Consultant, Enterprise Networking Solutions, for his vision and drive to make these Redbooks possible.

As is always the case with any complex technical effort such as this *Communications Server (CS) for z/OS TCP/IP Implementation* series, success would not have been possible without the advice, support, and review of many outstanding technical professionals from within IBM. We are especially grateful for the significant expertise and contributions of content to these books from the Communications Server for z/OS Development team, especially from: Jeannie Kristufek, Jeremy Geddes, Barry Mosakowski, Andy Tracy, Mike Fox, and Gary McAfee.

Thanks also to the International Technical Support Organization, Raleigh and Poughkeepsie, for their invaluable support in this project, particularly Margaret Ticknor, Bob Haimowitz, David Bennin, Denice Sharpe, Linda Robinson, Eran Yona, and, most importantly, Bill White—our ITSO mentor and guide.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or clients.

Your efforts will help increase product acceptance and client satisfaction. As a bonus, you'll 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 Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

  `ibm.com/redbooks`

- Send your comments in an email to:

  `redbook@us.ibm.com`

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYJ Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

Introduction

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. 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 z9 and zSeries mainframe hosts, critical mass can be achieved wherein investments to provide higher availability for all applications are more easily justified.

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 z9 and zSeries mainframes) and understanding and avoiding *single points of failure*: 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 may be too expensive to justify either of the following:

- Providing redundant high-speed wide area network 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).

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), we address those topics in this book 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 upon to scale application services to serve larger numbers of users and higher transaction volumes than can be supported by a single application instance.

1.1 Fundamental technologies for z/OS TCP/IP availability

In this section we introduce several important technologies for z/OS TCP/IP availability. Several of these technologies are discussed further, including implementation examples, in the subsequent chapters of this book.

1.1.1 Single z/OS system availability

Given the remarkable reliability of the IBM System z9 and zSeries 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 systems. While most of the technology solutions discussed in this book leverage clusters of z/OS systems, some also apply in single-system environments. They include:

- *Virtual IP Addressing (VIPA)* (discussed briefly in 1.1.3, "Virtual IP Addressing" on page 4, and to greater depth in Chapter 2, "Virtual IP Addressing" on page 13) provides physical interface independence for the TCP/IP stack (the part of your z/OS Communications Server software that provides TCP/IP protocol support) and applications so that interface failures will not impact application availability.
- *ARP takeover* (discussed in Chapter 3, "VIPA without dynamic routing" on page 39) enables your system to transparently exploit redundant physical interfaces without implementing a dynamic routing protocol in your mainframe.
- *Dynamic routing* (discussed in Chapter 4, "VIPA with dynamic routing" on page 59) 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).
- *Portsharing* (discussed in *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 1 - Base Functions, Connectivity, and Routing*, SG24-7169, and in 5.1.4, "Portsharing" on page 96) enables you to run multiple instances of an application for higher availability and scalability (to the extent possible in a single system).

1.1.2 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 up to 32 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 where:

- 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 via ESCON®, FICON CTCs, or HiperSockets. XCF communications is discussed in detail (complete with implementation examples) in the “Connectivity” chapter of *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 1 - Base Functions, Connectivity, and Routing*, SG24-7169.

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*).

1.1.3 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 in order 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). VIPA technology is the focus of Chapter 2, “Virtual IP Addressing” on page 13; consequently, we only introduce 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.

1.1.4 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 below with more in-depth discussion (and implementation examples) in Chapter 3, "VIPA without dynamic routing" on page 39, and Chapter 4, "VIPA with dynamic routing" on page 59.

z/OS availability without dynamic routing

Why should we bear the cost and complexity of implementing a dynamic routing protocol in our mainframe environment when we do not do so in our other server environments? Why can we not just leave dynamic routing to the network people and keep servers out of it? These are, indeed, very good 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: You can do so by leveraging z/OS ARP takeover support. To understand how ARP takeover works, you need to understand a little bit about how the z/OS Communications Server and the OSA-Express adapter (in QDIO mode) work together to support TCP/IP communication:

- 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 dynamically maintain OSA address tables (OATs) containing information such as that shown in Table 1-1 below.

Table 1-1 Example of OSA address table information

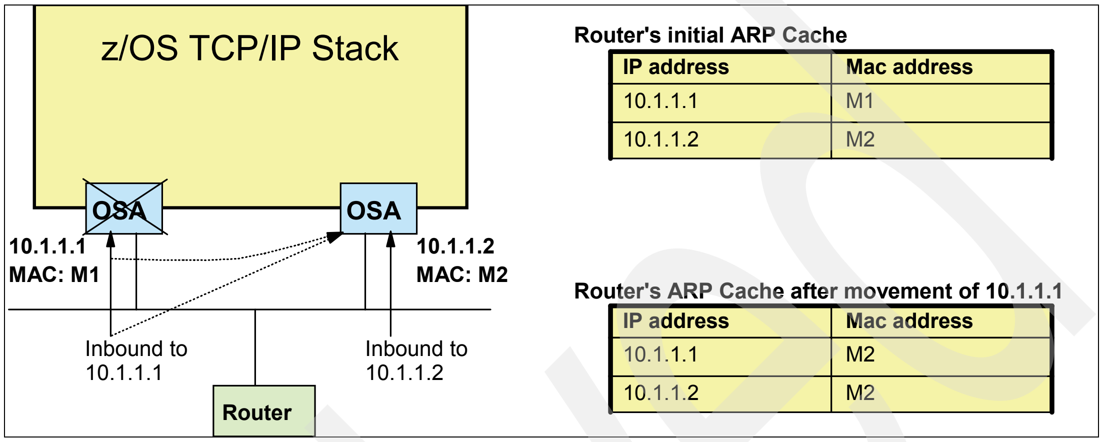
IP-Dest=	Give to...
10.0.1.1	LPAR-1
10.0.1.4	LPAR-2
10.0.1.5	LPAR-2
10.0.1.6	LPAR-2

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 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 3.5, “ARP takeover” on page 53.

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 loose 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/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 should you ever consider implementing one? One reason is because a sysplex, a cluster of System z9 and zSeries mainframes coupled together with dynamic XCF connectivity, is a network in and of 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:

- Should a system loose 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 VRRP or 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.

1.1.5 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 may care about which server they reach, while others may not.

Single-instance applications

The relationship of a client to a particular server instance may require that only a single-instance of an application be available at a time. In other cases, it may be that the server application needs exclusive access to certain resources that 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 1.1.2, “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 via their own means (shared files or structures in the Coupling Facility)

1.1.6 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 three different approaches for workload balancing: what we refer to as internal, external, and hybrid workload balancing.

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 will be made by *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 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: Prior to the z/OS V1R7.0 Communications Server, all distributed DVIPA traffic was forwarded from the routing stack to target stacks only across XCF interfaces. Now, however, with the VIPARoute function you can use any available interface for this traffic forwarding. Using alternative interfaces (such as HiperSockets or OSA-Express connectivity) can improve performance while reducing the utilization of XCF interfaces.

Note: 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 83.

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—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.

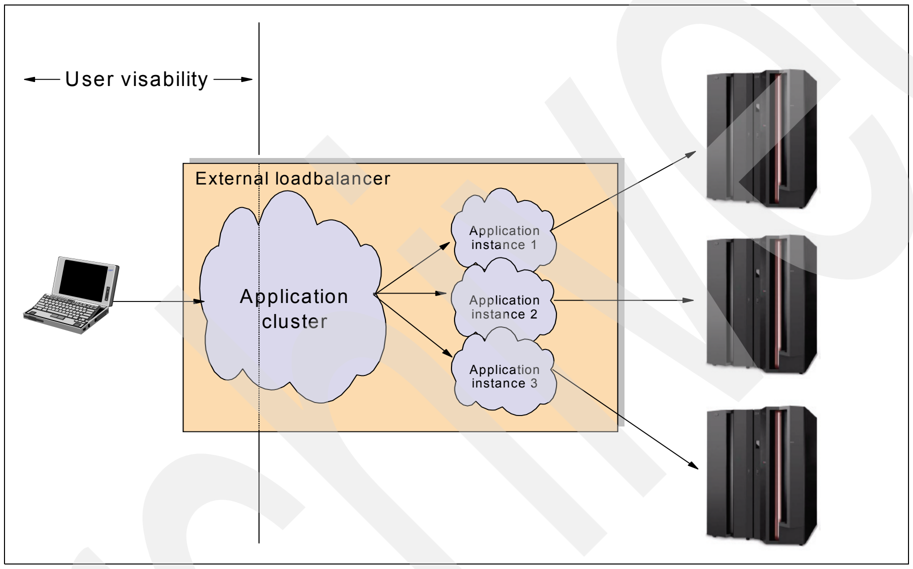


Figure 1-2 Relationship between application cluster and application instances

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 example of an external application workload balancing solution is Cisco System’s Content Services Switch (CSS). Like Sysplex Distributor, CSS distributes connection requests across a set of destination target servers; however, with CSS, traffic is forwarded *directly* to the appropriate application server from the network switches rather than through a z/OS routing stack.

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 may 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 may 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/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 lots of 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 an 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 139.

Hybrid internal/external approaches

In many cases, your application load balancing requirements may be such that purely internal or external workload balancing will be satisfactory. However, in some situations you may need elements of both approaches. For example, you may 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 have been developed to address this requirement: Sysplex Distributor’s Multi-Node Load Balancing (MNLB) Service Manager support, and the new 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* and 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 Cisco switches perform the Forwarding Agent role. Note that the Sysplex Distributor makes the actual workload distribution decision. Then, acting as a Cisco Service Manager, Sysplex Distributor relays that decision to the Cisco 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 83.

The z/OS Load Balancer Advisor (LBA) 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 LBA, a weight metric is calculated (based upon 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 139.

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

1.2 Quick start table

Table 1-2 on page 12 is intended to 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 may be helpful for you and where they are discussed in the 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 39)		► SHAREPORT and SHAREPORTWLM (Chapter 5, “Internal application workload balancing” on page 83) ► External WLB (Chapter 6, “External application workload balancing” on page 139) ► LBA/SASP (Chapter 6, “External application workload balancing” on page 139)	► General ► TCP/IP ► z/OS UNIX® ► Storage ► Applications (Chapter 7, “Performance and tuning” on page 177)
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 59)			
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 39)	DVIPA and ARP takeover (Chapter 2, “Virtual IP Addressing” on page 13, and Chapter 3, “VIPA without dynamic routing” on page 39	► Distributed DVIPA and Sysplex Distributor (Chapters 2 and 6) ► External WLB (Chapter 6) ► LBA/SASP (Chapter 6)	
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 59)	DVIPA and OSPF (Chapter 2, “Virtual IP Addressing” on page 13, and Chapter 4, “VIPA with dynamic routing” on page 59)		

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 in order 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" on page 14"	Covers the basic concepts, commands, and functions associated with a VIPA
2.2, "Importance of VIPA" on page 30	Describes the reasons VIPA is important
2.3, "Dynamic VIPA example" on page 31	Details the implementation of a basic VIPA environment

2.1 Basic concepts

The z/OS Communications Server supports two types of VIPA: static and 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. An example configuration for a static VIPA can be seen in Example 4-2 on page 70.

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, 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* 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 may 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 (via IOCTL macro) -when one instance of an application takes over primary function from another

- By using a utility program (MODDVIPA) from the operator console

A simple example of DVIPA is shown in Figure 2-1, illustrating a case where a TCP/IP stack (and associated applications) has failed.

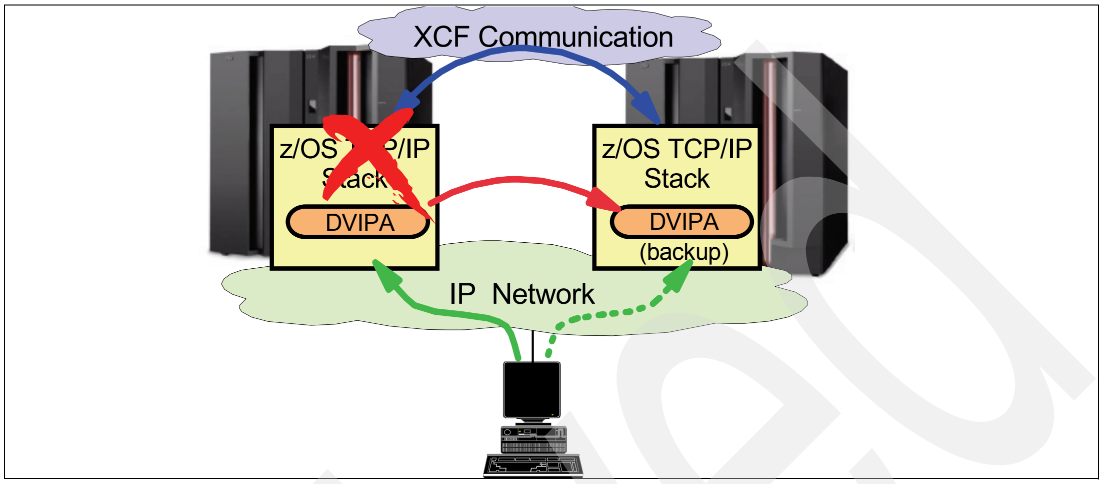


Figure 2-1 DVIPA address movement

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 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 dynamically activated by an application program (via 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 on page 16) 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.

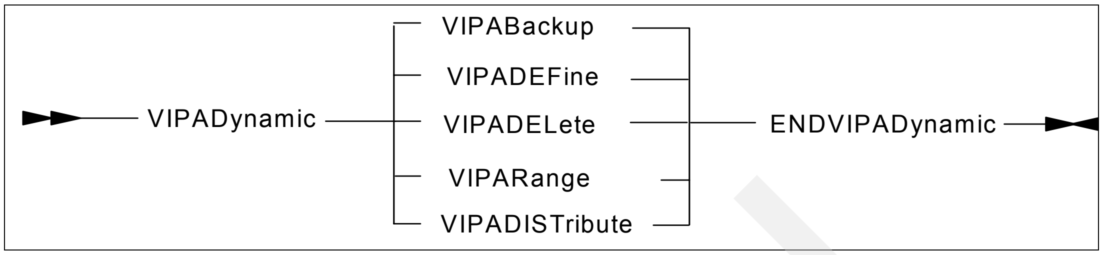


Figure 2-2 DVIPA configuration

See “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 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; however, leveraging automation utilities (such as Automatic Restart Manager, or 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; however, 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)

While multiple application instances will provide the highest availability (as well as higher scalability), 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.
- There may be 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 may need to implement several of the above 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) will dictate how you set up and use DVIPAs and distributed DVIPAs. Table 2-1 on page 17 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 may 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 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 via 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, provided 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. This implementation is described in 5.4.1, "Scenario A - Single-instance server - event-managed" on page 110.

Since the VIPA is specified by the application, it need not 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 dynamically activate, as shown in Figure 2-3.

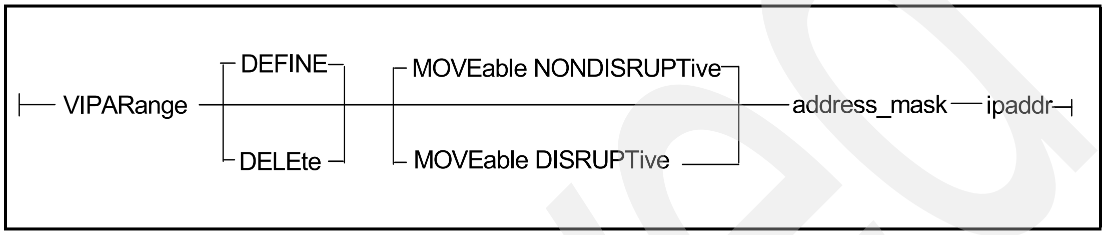


Figure 2-3 Definition format for VIPARange

The VIPARange statement defines an IP subnetwork using a network address (prefix) and a subnet mask. Since the same VIPA address may not be activated by IOCTL/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) may cause a Dynamic VIPA to be created and activated for a specific IP address. The same range may have DVIPAs defined via VIPADEFINE or VIPABACKUP, provided no attempt is made to use the same IP address for both VIPADEFINE and activation via program BIND.

For our operation we used the following definition:

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

Once activated on a stack via bind() or IOCTL, a dynamic VIPA IP address remains active until the VIPA IP address is moved to another stack or it is deleted. The system operator may 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 may be used. Deleting a VIPARANGE does not affect existing DVIPAs that are already activated within the range, but prevents 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 may be automatic or manual:

- Automatic

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

- 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 (such as resulting from some sort of failure).

  A reactivate command:

  – Causes takeback for a VIPADEFINE DVIPA

  – Re-enables VIPABACKUP DVIPA to do takeover in case a stack leaves the XCF group

Applications that bind() to INADDR_ANY

An application may 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 via JCL, from the OMVS command line, or from a shell script. This utility is in EZB.MODDVIPA.sysname.tcpname. More information about the MODDVIPA utility may be found in *z/OS V1R7.0 CS: IP Configuration Reference,* SC31-8775.

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

Using 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 may 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 via VIPADEFine or VIPABackup in a VIPADynamic block.

MODDVIPA must run with APF authorization. For additional information about APF authorization, see the “RACF Demystified” chapter in *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 4 - Policy-Based Network Security*, SG24-7172.

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 us 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

Let us consider the environment depicted in Figure 2-4 on page 21. This 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.

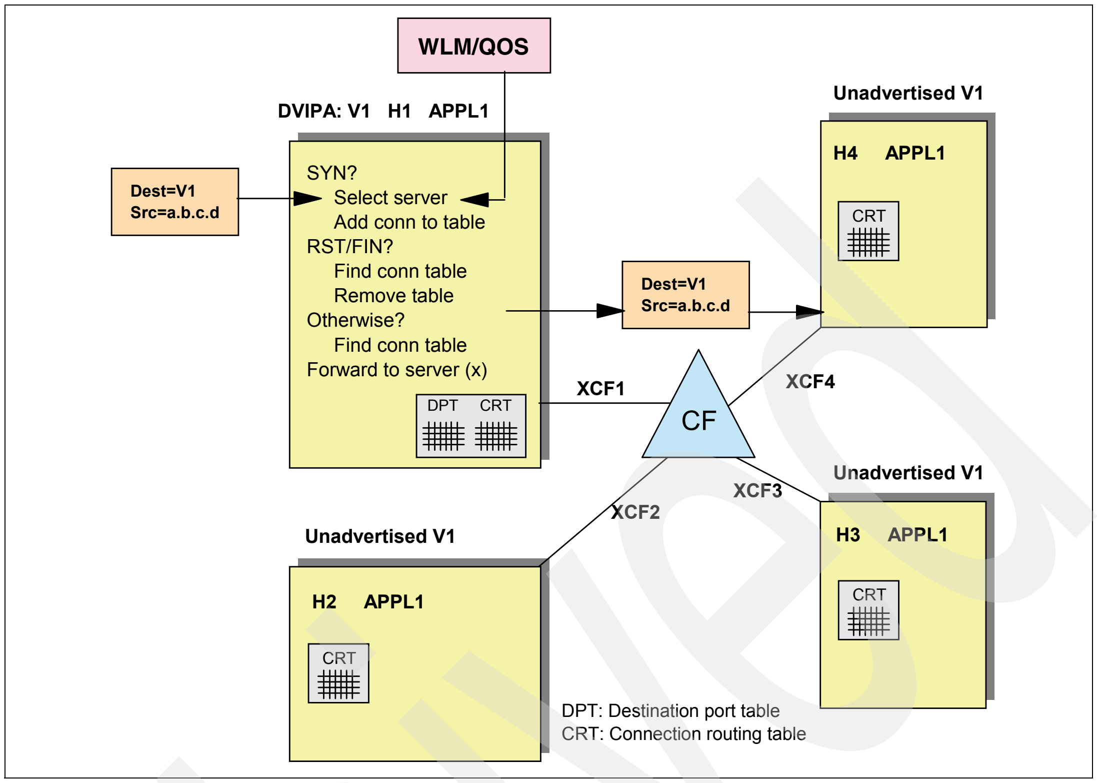


Figure 2-4 Sysplex Distributor functionality

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 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 indicated Figure 2-5.

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. Say 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.

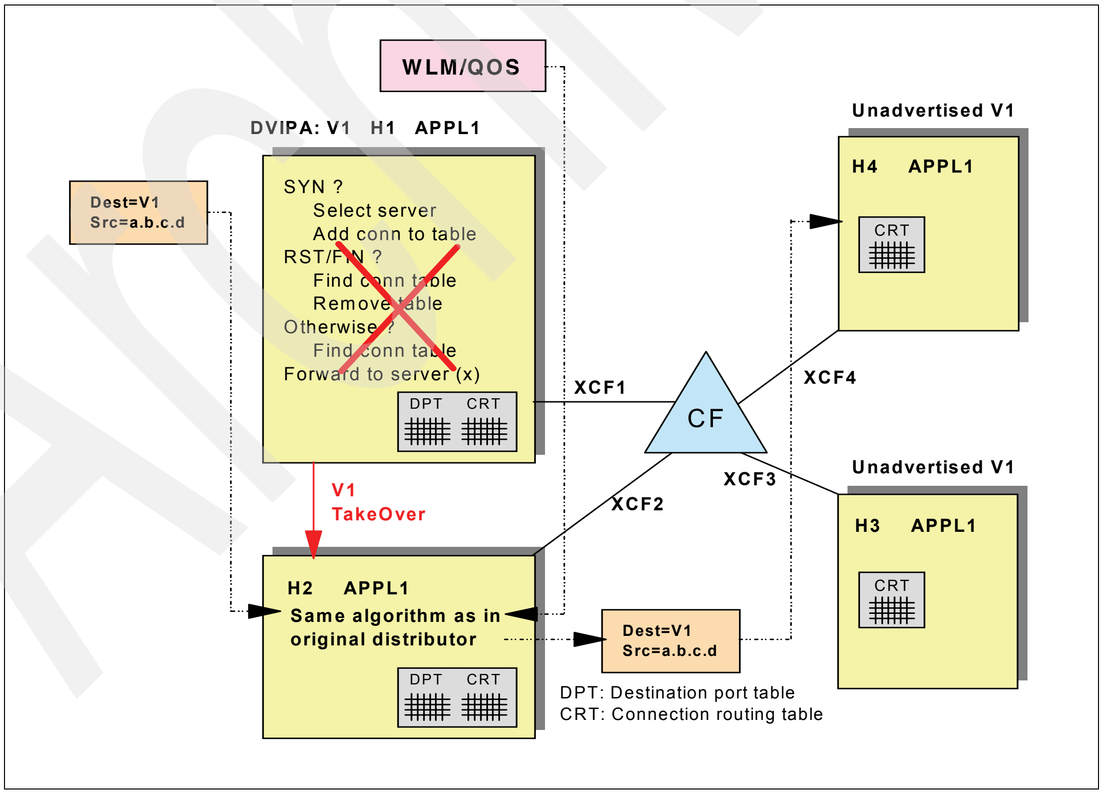


Figure 2-5 DVIPA Backup capability

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 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 were lost. All other connections remain allocated and function properly.

Recovery

Once 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 re-activated, H2 sends its connection information to H1. This gives H1 the information it needs to once again distribute packets 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 on page 24 illustrates this situation.

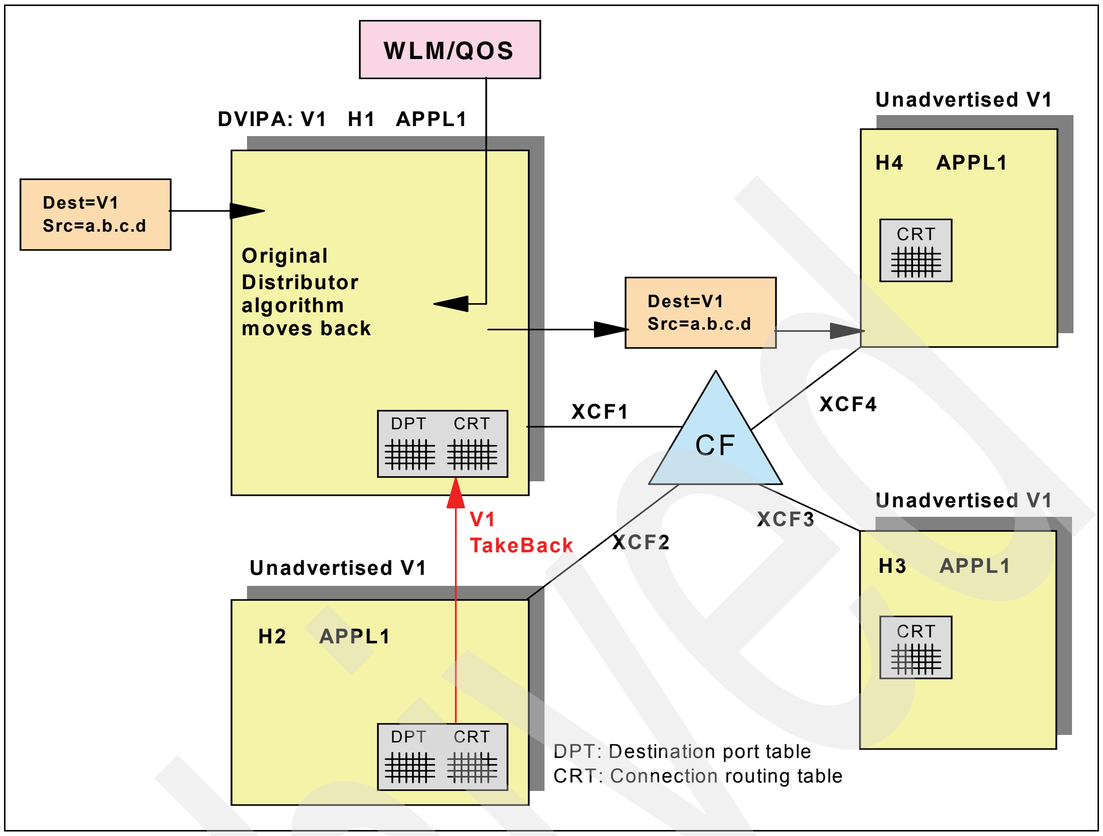


Figure 2-6 Sysplex Distributor and VIPA takeback

Dynamic routing with Sysplex Distributor

Routing IP packets for Sysplex Distributor can be divided into two cases: routing inside the sysplex cluster and routing through the IP network.

Routing *inside* the sysplex cluster is accomplished by the distributing host, as just described. 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. (This is described in 2.1.3, "VIPARoute function" on page 25.) With release 1.7, OSA-Express connections may 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 (such as resulting from some sort of failure).

When using OMPROUTE, you should 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 may 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 V1R7.0 Communications Server. Previously, all DVIPA IP traffic was forwarded to target stacks only by using Dynamic XCF interfaces. This new 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.

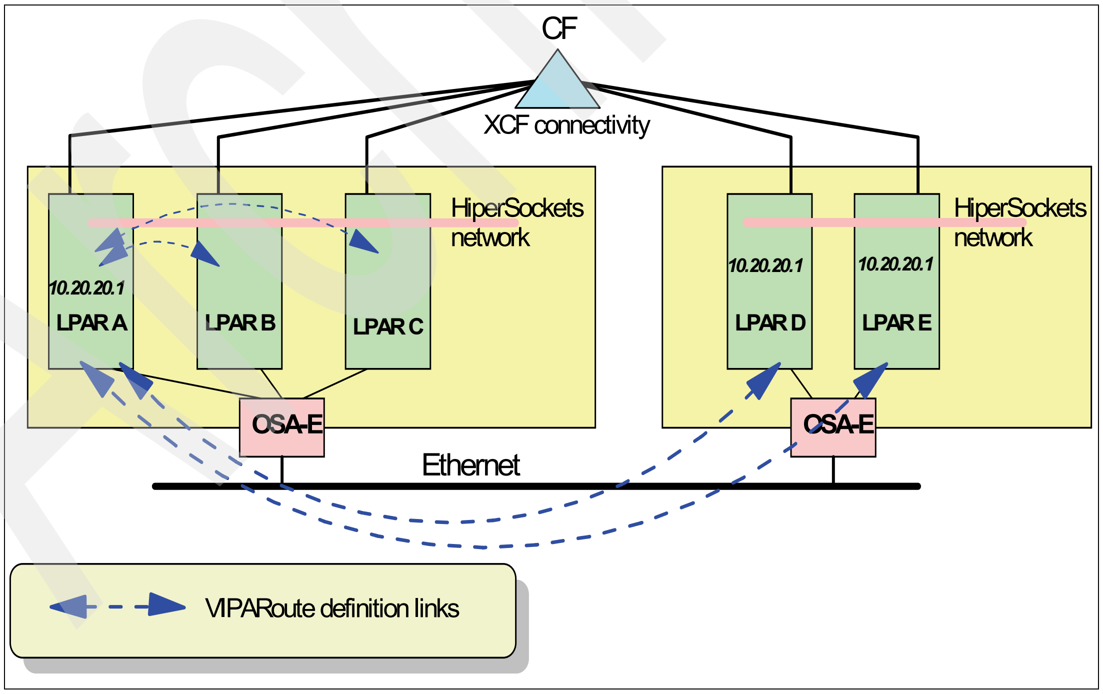


Figure 2-7 VIPARoute function

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. The command **NETSTAT VIPADYN /-v** may be used to query the status of a dynamic VIPA, including the usage of VIPAROUTE. For example:

```
MVS TCP/IP NETSTAT CS V1R7 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.

2.1.4 New DVIPA commands

New commands have been added in the z/OS V1R7.0 Communications Server to assist with management, problem detection, and recovery of DVIPA address. These commands are used as extensions to the VARY TCPIP operator command:

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

Joingroup and leavegroup

These two commands are not unique to DVIPA, but apply to a complete TCP/IP stack. We discuss them here because they complement the other new 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.

Starting with the z/OS V1R7.0 Communications Server, 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. Figure 2-8 on page 27 illustrates part of this operation.

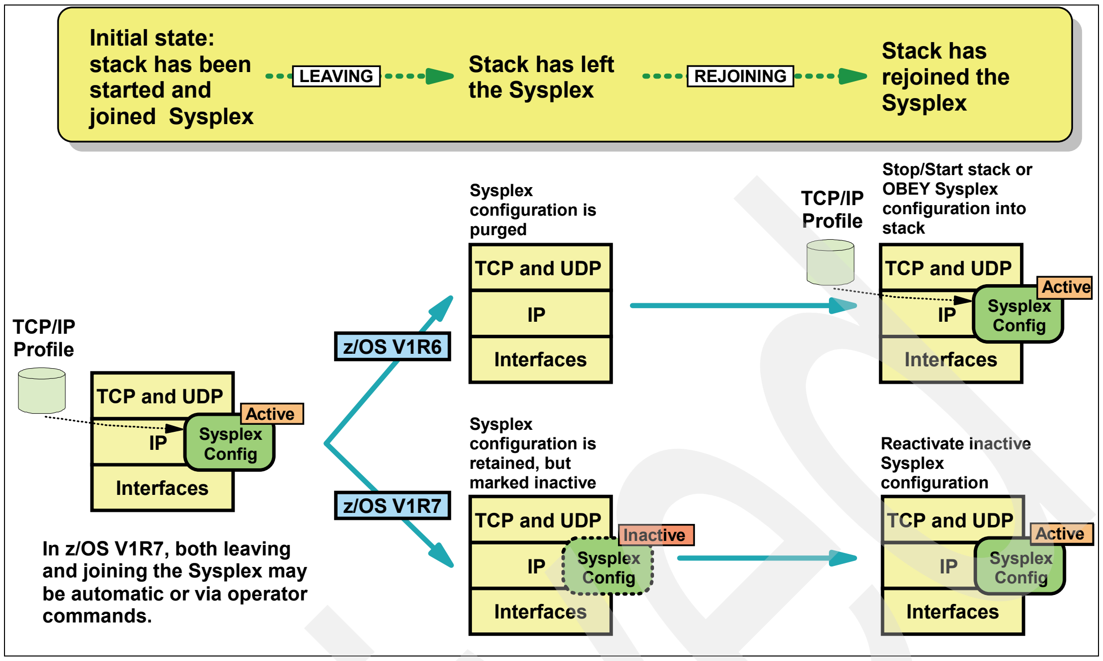

Initial state: stack has been started and joined Sysplex
LEAVING
Stack has left the Sysplex
REJOINING
Stack has rejoined the Sysplex
TCP/IP Profile
TCP and UDP
IP
Sysplex Config
Active
Interfaces
In z/OS V1R7, both leaving and joining the Sysplex may be automatic or via operator commands.
z/OS V1R6
z/OS V1R7
Sysplex configuration is purged
Sysplex configuration is retained, but marked inactive
Inactive
TCP/IP Profile
Stop/Start stack or OBEY Sysplex configuration into stack
Reactivate inactive Sysplex configuration

Figure 2-8 Join and leaving function

The LEAVEGROUP or JOINGROUP function can be automatic or through an operator command.

Automatic control is configured by a parameter in the GLOBALCONFIG statement in the TCP/IP PROFILE:

```
GLOBALCONFIG SYSPLEXMONITOR AUTOREJOIN
```

This has the following effects and controls:

- The stack will rejoin the sysplex when the problem that caused it to automatically leave the sysplex has been relieved.
- AUTOREJOIN is only supported 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 back up, 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 are in the following format:

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

These allow full operator control over when a stack should leave or rejoin 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:

```
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. This OBEY format provides compatibility for operations procedures that were established prior to z/OS V1R7.

Certain dynamic stack configuration definitions are not saved when a stack leaves a sysplex. These are:

- Target DVIPAs. These will be automatically re-created when the stack rejoins the group if this stack is still a target for another (distributing) stack.
- BIND-created or IOCTL-created DVIPAs. These must be re-created by the applications or by the MODDVIPA utility after the stack has rejoined the group.

Deactivate and reactivate

These commands (new with z/OS V1R7) 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.

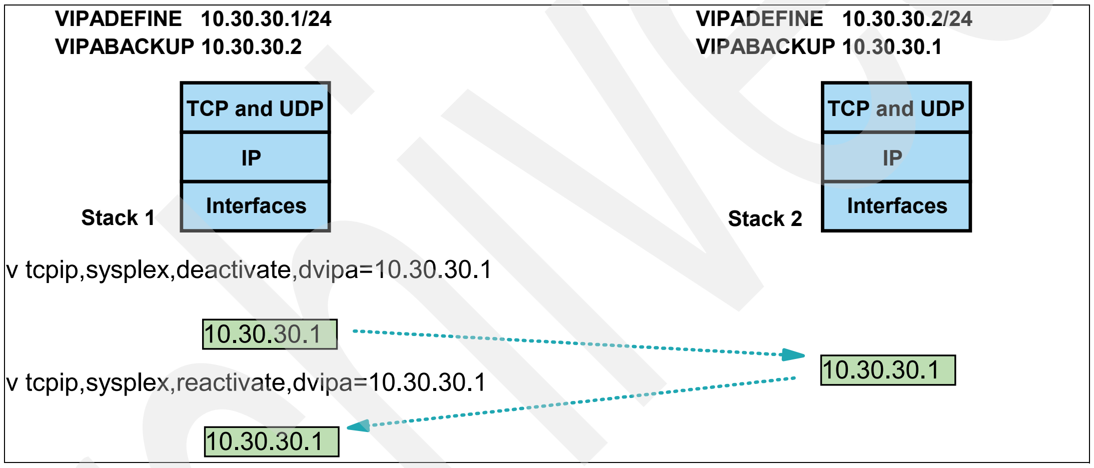


Figure 2-9 VIPA commands deactivate and reactivate

The format of the DEACTIVATE command is:

```
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) will take over the DVIPA. A backup DVIPA can be deactivated. This removes its eligibility as a backup.

The format of the REACTIVATE command is:

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

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

Restriction: These commands cannot be used on a DVIPA that was created from VIPARANGE with bind, ioctl(), or the MODDVIPA utility.

Quiesce and resume

Those new 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 the following:

- Planned maintenance of the system or application
- Allows existing systems or applications to drain their work queue prior to shutdown
- Relieve temporary constraints of resources on target system
- Temporary - Does not affect Sysplex Distributor's permanent configuration
- Issued on target system being affected
- Can also be used 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:

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

The options field may be one or several of the following:

- TARGET - Quiesces all applications on the target stack
- PORT=xxx - Quiesces all applications bound to the specified port on this stack
- JOBNAME=jobname - Allows quiesce of a single application in SHAREPORT group
- ASID=asid - Further qualifys jobs being quiesced (such as when dealing with duplicate jobnames)

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:

```
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 will be 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 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 2-1 on page 30.

Example 2-1 Nestat ALL/-A result

```
MVS TCP/IP NETSTAT CS V2R7      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 see *z/OS V1R7.0 CS: IP Configuration Reference,* SC31-8775.

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, another device or link can automatically provide alternate paths to the destination.
- Recovery when a z/OS TCP/IP stack leaves the XCF group (such as 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 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.

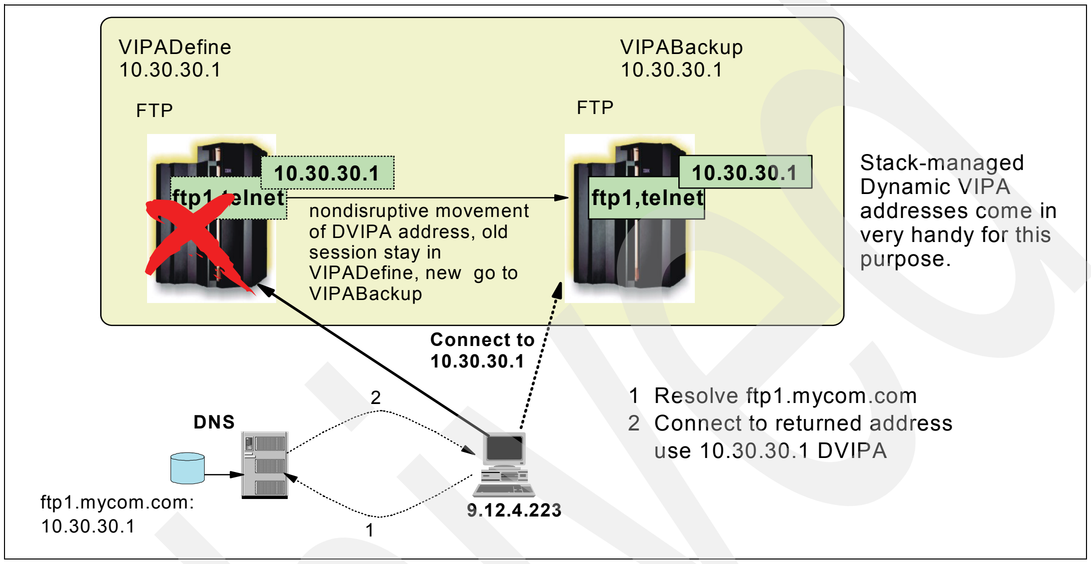

VIPADefine
10.30.30.1
FTP
10.30.30.1
ftp1,telnet
VIPABackup
10.30.30.1
FTP
10.30.30.1
ftp1,telnet
nondisruptive movement of DVIPA address, old session stay in VIPADefine, new go to VIPABackup
Stack-managed Dynamic VIPA addresses come in very handy for this purpose.
Connect to 10.30.30.1
DNS
2
1
ftp1.mycom.com: 10.30.30.1
9.12.4.223
1 Resolve ftp1.mycom.com
2 Connect to returned address use 10.30.30.1 DVIPA

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

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 automatically moved to another stack without the need for human intervention. The new stack will have received information regarding the connections from the original stack and will accept new connections for the DVIPA. The routers are automatically informed about the change. This 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 may be distributed across the sysplex rather than on a single node.

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. Lastly, we decided to use SYSPLEXROUTING, although this is optional.

2.3.1 Implementation

Our goal was to perform the following:

- 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 used.

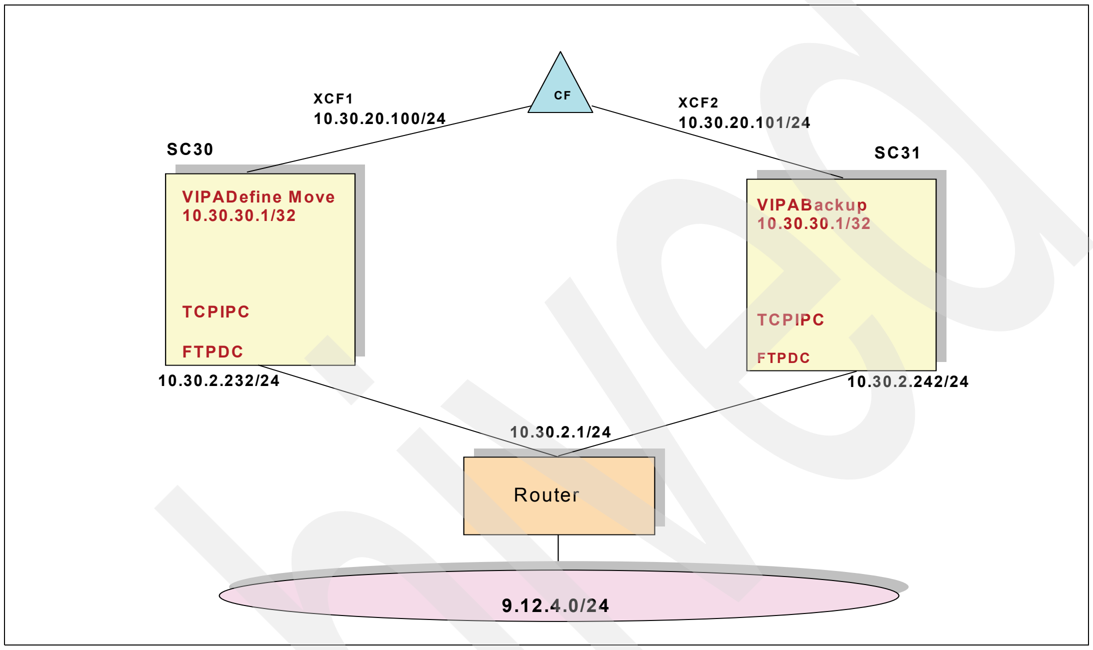


Figure 2-11 Base network diagram used for stack-managed implementation

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 following:

```
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
```

Our PROFILE for TCPIPC on SC31 included the following:

```
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
```

Our OMPRoute definitions included the following:

```
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 2-2 shows the commands and results from the two LPARs.

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

```
D TCPIP,TCPIPC,N,VIPADCFG
RESPONSE=SC30
EZD0101I NETSTAT CS V1R7 TCPIPC 217
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 V1R7 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 V1R7 584
 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
```

We also displayed the OSPF routing table on TCPIPC, as shown in Example 2-3. Note the presence of the local dynamic VIPA 1 with its stack-generated link name.

Example 2-3 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     VIPL0A1E1E01
```

```
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 Figure 2-4.

Example 2-4 Display sessions using DVIPA address

```
D TCPIP,TCPIPC,N,CON
RESPONSE=SC30
 EZD0101I NETSTAT CS V1R7 TCPIPC 231
 USER ID  CONN     STATE
FTPDC1   00000724 ESTBLSH
   LOCAL SOCKET:   ::FFFF:10.30.30.1..21
   FOREIGN SOCKET: ::FFFF:10.12.4.223..2222
```

2.3.2 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 may 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 shows the result of this operation. The network is informed, via 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 (indeed, the only stack) on the backup list for 10.30.30.1 was TCPIPC on SC31.

Example 2-5 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 V1R7 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
EZD0101I NETSTAT CS V1R7 TCPIPC 516
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. Most TCP/IP applications recover from this automatically. Movements during *takeack* 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 2-6 shows the console messages associated with this action.

Example 2-6 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 2-7 shows the existing to connections to SC31. However, TCPIPC in SC30 is now receiving all new connections. Example 2-8 on page 36 shows the creation of a new FTP connection.

Example 2-7 Display of the connections to SC31 after the takeback to SC30

```
D TCPIP,TCPIPC,N,VCRT
EZD0101I NETSTAT CS V1R7 TCPIPC 592
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 2-8 shows the VIPA connection routing table (VCRT) in both stacks, which include the old and new connections.

Example 2-8 Display of the older session in SC31 and new session in SC30 after the takeback

```
RESPONSE=SC30
 EZD0101I NETSTAT CS V1R7 TCPIPC 121
 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
EZD0101I NETSTAT CS V1R7 TCPIPC 123
FTPDC1   0000003A ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.30.30.1..21
  FOREIGN SOCKET: ::FFFF:10.12.4.223..2256
```

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 2-9 shows a new FTP connection to DVIPA address 10.30.30.1 and shows that this connection was established with TCPIPC in SC31, since the DVIPA has not yet been taken back.

Example 2-9 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 2-10 on page 36 shows this result.

Example 2-10 VIPA active in SC31

```
RESPONSE=SC30
 EZZ8260I SYSPLEX CS V1R7 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 2-11 shows the result of this operation.

Example 2-11 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 may be retired in a future release, since 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 2-12 shows the results of this manual operation.

Example 2-12 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
EZD0101I NETSTAT CS V1R7 TCPIPC 571
DYNAMIC VIPA:
  IPADDR/PREFIXLEN: 10.30.30.1/32
    STATUS: QUIESCING  ORIGIN: VIPADEFINE
```

Example 2-13 shows that SC31 now has the DVIPA active.

Example 2-13 System SC31 assumes the VIPA address after the deactivate

```
RO SC31,D TCPIP,TCPIPC,N,VIPAD
RESPONSE=SC31
 EZD0101I NETSTAT CS V1R7 TCPIPC 105
 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 2-12 on page 37 shows the results of this manual operation.

Example 2-14 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 had started two FTP sessions to TCPIPC in SC31 while that system had the active DVIPA address. After the REACTIVATE we started more two FTP sessions. Example 2-15 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 2-15 Display VCRT shows the old and new sessions SC31 with moving status

```
D TCPIP,TCPIPC,SYSPLEX,VIPADYN
EZZ8260I SYSPLEX CS V1R7 794
RESPONSE=SC30
 EZZ8260I SYSPLEX CS V1R7 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
EZD0101I NETSTAT CS V1R7 TCPIPC 805
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
```

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

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. This chapter explains how to design and implement high availability for your z/OS Communications Server environment without dynamic routing.

This chapter discusses the following.

Section	Topic
3.1, "Basic concepts" on page 40"	Discusses background and why this solution may be important in typical environments
3.2, "Design example" on page 41	Discusses key design points
3.3, "High-availability scenarios" on page 46	Steps through detailed implementation examples
3.4, "Debugging tips" on page 52	Summarizes debugging approaches useful when working with layer two definitions
3.5, "ARP takeover" on page 53	Discusses what is ARP Takeover, the basic concepts, implementations and test

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 to be able 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 requires OSA-Express adapters (in QDIO mode).

Note: The way ARP takeover and OSA-Express QDIO mode work together is introduced in 1.1.4, “z/OS network connectivity and dynamic routing” on page 5. An ARP takeover implementation example is provided in 3.5, “ARP takeover” on page 53.

In addition, if you wish to move an IP address *from one adapter to another within a system,* you must also set up Static Virtual IP Addresses (VIPA). And, 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, including implementation scenarios in Chapter 2, “Virtual IP Addressing” on page 13.

ARP takeover uses LAN broadcast protocols; therefore, in order to use ARP takeover, all of the systems in your sysplex must be connected to the same LAN (or virtual LAN, VLAN), and, hence, 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 two 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 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. Hence, a dynamic routing protocol is not required.

3.2 Design example

We used two OSA adapters in QDIO mode, connected to two Cisco 6509 switches. We created a unique VLAN (VLAN 130), 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 failover implementation moves the active IP address from one LPAR to the other when needed.

Our design had the following dependencies:

- We need 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 automatically. 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 all broadcast frames.
- The IP address for the DVIPA must be in the same subnet used for the LAN segment.
- If the subnet we use has many hosts 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 a high-availability network environments.

This design has the following advantages:

- It is simple, easy to configure, and 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.

Note: If you are not using dynamic routing, you should not use z/OS multipathing because the z/OS system will have no visibility beyond its OSA interfaces and could end up sending connection requests and traffic over unavailable network paths (resulting in performance problems and data packets being discarded in the network).

3.2.1 Implementation

Figure 3-1 on page 42 illustrates our working environment. The OSA-Express2 CHPIDs were defined as OSD types, causing them to work in QDIO mode. To increase the availability of our network, we installed two CISCO 6509 switches and used Hot Standby Routing Protocol (HSRP) between them.

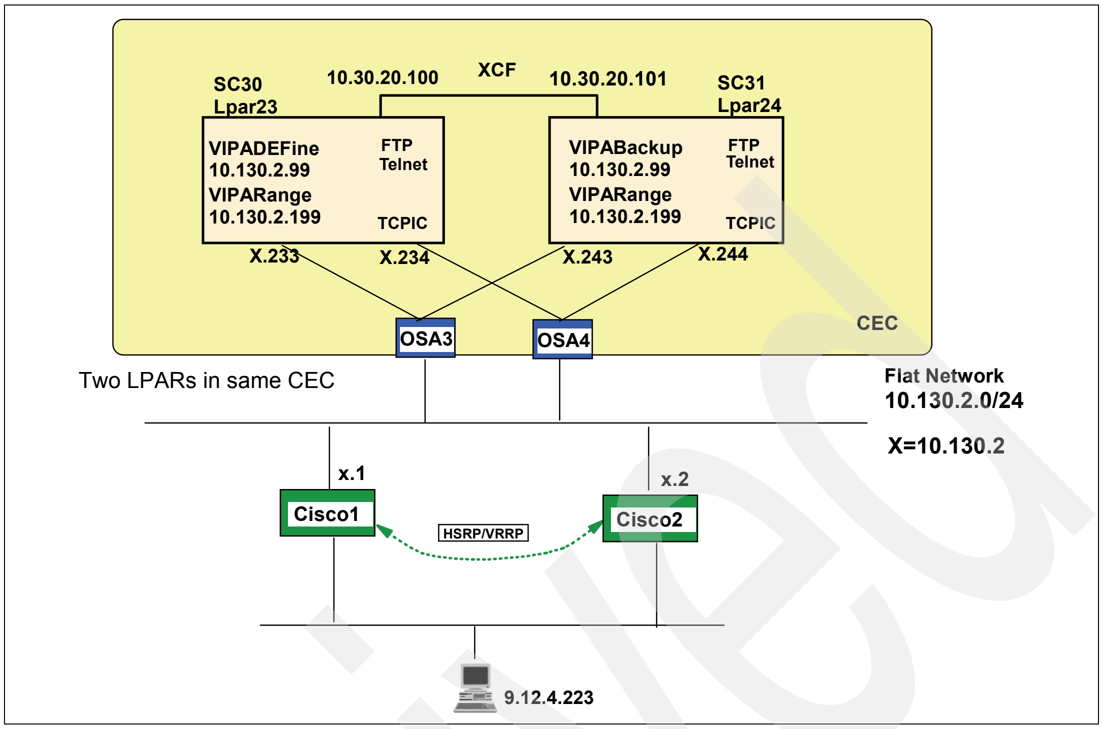


Figure 3-1 Diagram of our working environment

Example 3-1 and Example 3-2 on page 43 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 Adapters to use the same VLAN (VLAN 130) **1**.
- We created static route and default route definitions to both OSA adapters that belong to the new VLANID. Doing this we have high availability in the interface. One is a backup of another **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 instead of INADDR_ANY by using the server bind control that is implemented through the BIND keyword on the PORT statement **5**.

Example 3-1 TCP/IP profile in SC30

```
IPCONFIG MULTIPATH IQDIOR DATAGRAMFWD
IPCONFIG SOURCEVIPA
DEVICE OSA20A0  MPCIPA
LINK   OSA20A0LNK  IPAQENET     OSA20A0 VLANID 130 1
DEVICE OSA20C0  MPCIPA
LINK   OSA20C0LNK  IPAQENET     OSA20C0 VLANID 130 1

HOME
10.130.2.233    OSA20A0LNK
10.130.2.234    OSA20C0LNK
BEGINROUTES
ROUTE 10.130.2.0       255.255.255.0 =       OSA20C0LNK    mtu defaultsize 2
ROUTE 10.130.2.0       255.255.255.0 =       OSA20A0LNK    mtu defaultsize
ROUTE DEFAULT         10.130.2.1    OSA20C0LNK  mtu defaultsize
ROUTE DEFAULT         10.130.2.2    OSA20A0LNK  mtu defaultsize
ENDROUTES
```

```
;Test flat network start
VIPADynamic
VIPARange DEFINE MOVEable NONDISRUPTive 255.255.255.255 10.130.2.199 4
VIPADEFINE MOVE IMMED 255.255.255.255 10.130.2.99 3
ENDVIPADYNAMIC
;Test flat network end
;Bind to Dynamic VIPA SC30 and SC31:
PORT
20 TCP *  NOAUTOLOG          ; FTP Server
21 TCP OMVS BIND 10.130.2.199 ; control port 5
```

Example 3-2 TCP/IP profile in SC31

```
IPCONFIG MULTIPATH IQDIOR DATAGRAMFWD
IPCONFIG SOURCEVIPA

DEVICE OSA20A0  MPCIPA
LINK   OSA20A0LNK  IPAQENET      OSA20A0 VLANID 130 1
DEVICE OSA20C0  MPCIPA
LINK   OSA20C0LNK  IPAQENET      OSA20C0 VLANID 130 1

HOME
10.130.2.243    OSA20A0LNK
10.130.2.244    OSA20C0LNK
BEGINROUTES 2
ROUTE 10.130.2.0       255.255.255.0 =        OSA20C0LNK    mtu defaultsize
ROUTE 10.130.2.0       255.255.255.0 =        OSA20A0LNK    mtu defaultsize
ROUTE DEFAULT         10.130.2.1    OSA20C0LNK  mtu defaultsize
ROUTE DEFAULT         10.130.2.2    OSA20A0LNK  mtu defaultsize
ENDROUTES
;Test flat network start
VIPADynamic
VIPARange DEFINE MOVEable NONDISRUPTive 255.255.255.255 10.130.2.199 4
VIPABackup 1 MOVEable IMMEDiate 255.255.255.255 10.130.2.99 3
ENDVIPADYNAMIC
;Test flat network end
20 TCP *  NOAUTOLOG          ; FTP Server
21 TCP OMVS BIND 10.130.2.199 ; control port 5
```

We defined our VLAN in both Cisco 6509 switches, as shown in Example 3-3 and Example 3-4 on page 44. 6 These two examples also illustrate **ping** operations between the routers and the OSA adapters.

Example 3-3 Cisco SW 1 setup

```
Router(config)#interface vlan 130 6
Router(config-if)#ip address 10.130.2.1 255.255.255.0
Router(config-if)#no shut
Router(config-if)#end

Router# ping 10.130.2.234
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.130.2.234, 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 3-4 Cisco SW 2 setup

```
Router(config)#interface vlan 130 6
Router(config-if)#ip address 10.130.2.2 255.255.255.0
Router(config-if)#no shut
Router(config-if)#end

Router# ping 10.130.2.233
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.130.2.233, 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 3-5.

Example 3-5 Flat network result display routes in normal operation of SC30 and SC31

```
D TCPIP,TCPIPC,N,ROUTE
RESPONSE=SC30
EZD0101I NETSTAT CS V1R7 TCPIPC 944
IPV4 DESTINATIONS
DESTINATION         GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT             10.130.2.1       UGS       000001  OSA20C0LNK 1
DEFAULT             10.130.2.2       UGS       000000  OSA20A0LNK
10.130.2.0/24       0.0.0.0          UZ        000000  OSA20C0LNK
10.130.2.0/24       0.0.0.0          UZ        000000  OSA20A0LNK
10.130.2.99/32      0.0.0.0          UH        000000  VIPL0A820263
10.130.2.233/32     0.0.0.0          UH        000000  OSA20A0LNK
10.130.2.234/32     0.0.0.0          UH        000000  OSA20C0LNK
RESPONSE=SC31
EZD0101I NETSTAT CS V1R7 TCPIPC 928
IPV4 DESTINATIONS
DESTINATION         GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT             10.130.2.1       UGS       000000  OSA20C0LNK 1
DEFAULT             10.130.2.2       UGS       000000  OSA20A0LNK
10.130.2.0/24       0.0.0.0          UZ        000000  OSA20C0LNK
10.130.2.0/24       0.0.0.0          UZ        000000  OSA20A0LNK
10.130.2.99/32      0.0.0.0          UH        000000  VIPL0A820263
10.130.2.233/32     0.0.0.0          UH        000000  OSA20A0LNK
10.130.2.234/32     0.0.0.0          UH        000000  OSA20C0LNK
```

Note the two default gateways to routers 10.130.2.1 and 10.130.2.2 (1) in Example 3-5.

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 3-6.

Example 3-6 Display NETSTAT Dvipa config in SC30 and SC31

```
D TCPIP,TCPIPC,N,VIPADCFG
RESPONSE=SC30
EZD0101I NETSTAT CS V1R7 TCPIPC 947
DYNAMIC VIPA INFORMATION:
VIPA DEFINE:
IPADDR/PREFIXLEN: 10.130.2.99/32
    MOVEABLE: IMMEDIATE  SRVMGR: NO
```

```
VIPA RANGE:
IPADDR/PREFIXLEN: 10.130.2.199/32
    MOVEABLE: NONDISR
RESPONSE=SC31
 EZD0101I NETSTAT CS V1R7 TCPIPC 938
 DYNAMIC VIPA INFORMATION:
   VIPA BACKUP:
IPADDR/PREFIXLEN: 10.130.2.99
       RANK: 001  MOVEABLE:            SRVMGR:
   VIPA RANGE:
IPADDR/PREFIXLEN: 10.130.2.199/32
MOVEABLE: NONDISR
```

We used two TCP/IP applications, TELNET and FTP. Both were used from a client workstation at address 9.12.4.223. The Telnet client was configured to access the VIPADEFine **1** 10.130.2.99. The FTP client was started to access the VIPARange 10.130.2.199 **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 3-7.

Example 3-7 Display NETSTAT Dvipa status in SC30 and SC31

```
D TCPIP,TCPIPC,N,VIPADYN
RESPONSE=SC30
EZD0101I NETSTAT CS V1R7 TCPIPC 956
DYNAMIC VIPA:
IPADDR/PREFIXLEN: 10.130.2.99/32     1
    STATUS: ACTIVE     ORIGIN: VIPADEFINE       DISTSTAT:
after we start the FTP the VIPARange address appears
LINKNAME: VIPL0A8202C7
IPADDR/PREFIXLEN: 10.130.2.199/32    2
  ORIGIN: VIPARANGE BIND
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPC   SC30     ACTIVE

RESPONSE=SC31
 EZD0101I NETSTAT CS V1R7 TCPIPC 941
 DYNAMIC VIPA:
IPADDR/PREFIXLEN: 10.130.2.99/32
     STATUS: BACKUP     ORIGIN: VIPABACKUP       DISTSTAT:
```

A NETSTAT with a CONN operand, shown in Example 3-8, provides connection information.

Example 3-8 Telnet and FTP to DVIPA address in use

```
D TCPIP,TCPIPC,N,CONN
EZD0101I NETSTAT CS V1R7 TCPIPC 964
USER ID  CONN     STATE
TCPIPC   00000288 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.130.2.99..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..3640
FTPDC1   000000BF ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.130.2.199..21
  FOREIGN SOCKET: ::FFFF:9.12.4.223..3820
```

3.3 High-availability scenarios

The examples discussed here are not designed to be indicative of typical failures. Instead, they help us understand various recovery and transfer processes. We examine four areas of interest:

- Adapter interface failure
- Application movement
- Stack failure
- Local network failure

3.3.1 Adapter interface failure

We stopped the OSA adapter (OSA3) 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 3-2.

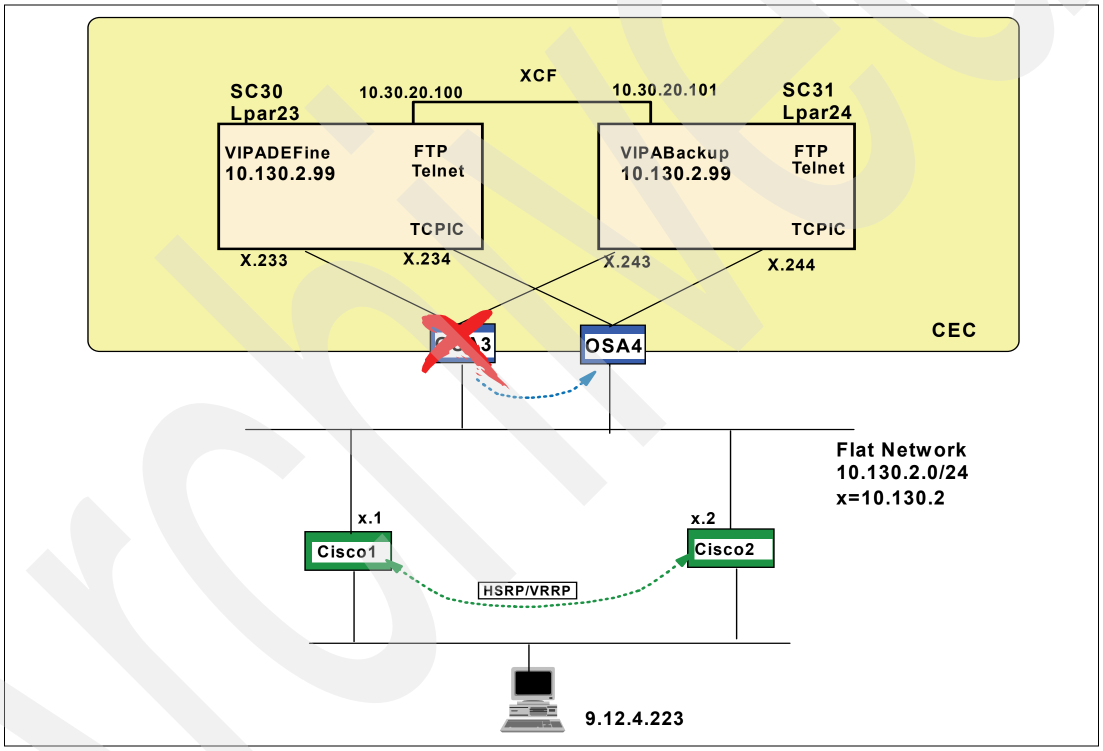


Figure 3-2 High availability for an adapter failure

Our command to stop the primary interface, and the resulting messages, is shown in Example 3-9.

Example 3-9 High availability in fail of interface

```
V TCPIP,TCPIPC,STOP,OSA20A0
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,STOP,OSA20A0
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4329I LINK OSA20COLNK HAS TAKEN OVER ARP RESPONSIBILITY FOR INACTIVE LINK OSA20AOLNK
```

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 3.5, “ARP takeover” on page 53.

Note that this function requires that both LAN adapters involved (OSA3 and OSA4 in our example) 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 3-10 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 the example.

Example 3-10 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,TCPIPC,N,CONN
EZD0101I NETSTAT CS V1R7 TCPIPC 964
USER ID  CONN     STATE
TCPIPC   00000288 ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.130.2.99..23
  FOREIGN SOCKET: ::FFFF:9.12.4.223..3640
FTPDC1   000000BF ESTBLSH
  LOCAL SOCKET:   ::FFFF:10.130.2.199..21
  FOREIGN SOCKET: ::FFFF:9.12.4.223..3820
```

After observing these results, we restarted the primary OSA interface (OSA3 in the illustrations). This resulted in a take back function that was transparent to the client.

3.3.2 Application movement

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 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 10.130.2.199. 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 may be able to have ARM restart the application automatically. If your application cannot bind to a DVIPA, you may be able to use the bind() on the port statement or the MODDVIPA utility.

The situation is illustrated in Figure 3-3.

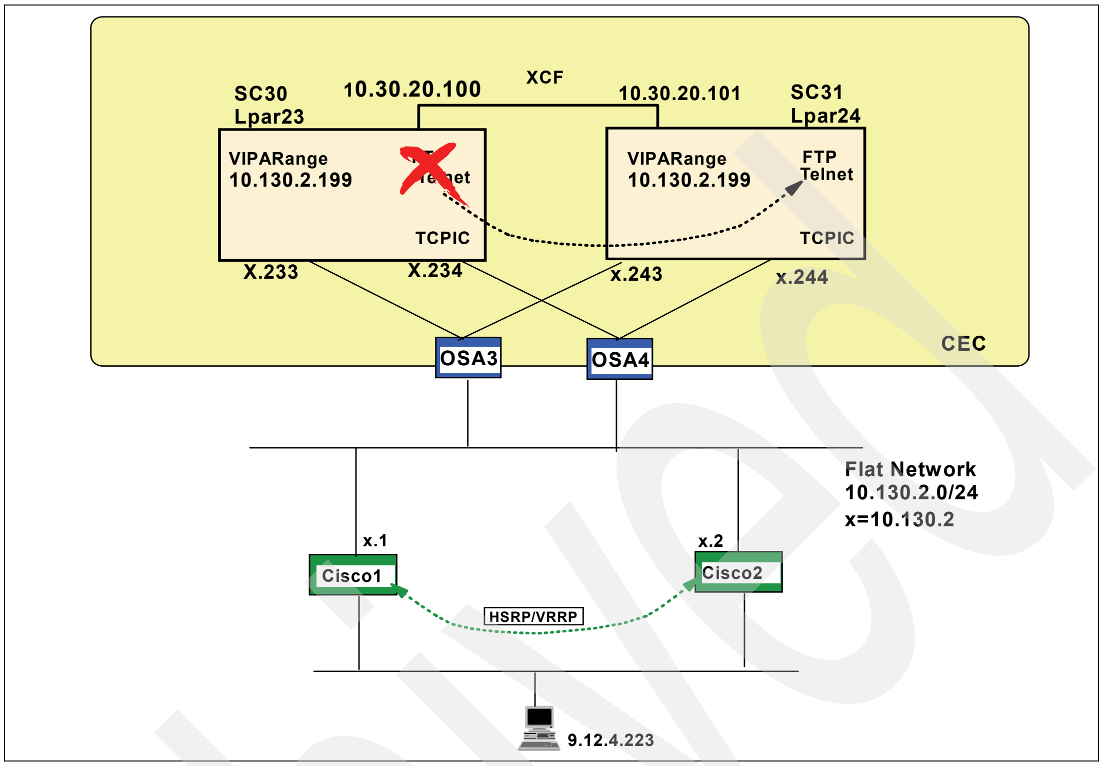


Figure 3-3 High availability for an application failure

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 examples below.

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 3-11.

Example 3-11 Vipa active in SC30

```
LINKNAME: VIPL0A8202C7
IPADDR/PREFIXLEN: 10.130.2.199/32
  ORIGIN: VIPARANGE BIND
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPC   SC30     ACTIVE
```

We then started an FTP application in SC31 and observed this stack take over the VIPA address, as shown in Example 3-12.

Example 3-12 The DVIPA address moves to SC31

```
S FTPDC
$HASP100 FTPDC    ON STCINRDR
IEF695I START FTPDC    WITH JOBNAME FTPDC    IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 FTPDC    STARTED
EZZ8302I VIPA 10.130.2.199 TAKEN FROM TCPIPC ON SC30
EZZ8303I VIPA 10.130.2.199 GIVEN TO TCPIPC ON SC31
+EZY2702I Server-FTP: Initialization completed at 15:33:05
```

After starting FTP in SC31 we can see the DVIPA status has automatically changed, and the DVIPA address is now active in SC31. Example 3-13 illustrates this new status.

Example 3-13 DVIPA active in SC31

```
IPADDR: 10.130.2.199
  ORIGIN: VIPARANGE BIND
  TCPNAME  MVSNAME  STATUS RANK DIST
  -------- -------- ------ ---- ----
  TCPIPC   SC31     ACTIVE
  TCPIPC   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 a whole TCP/IP stack. (We simulate this by simply stopping the stack.) The failure situation is illustrated in Figure 3-4.

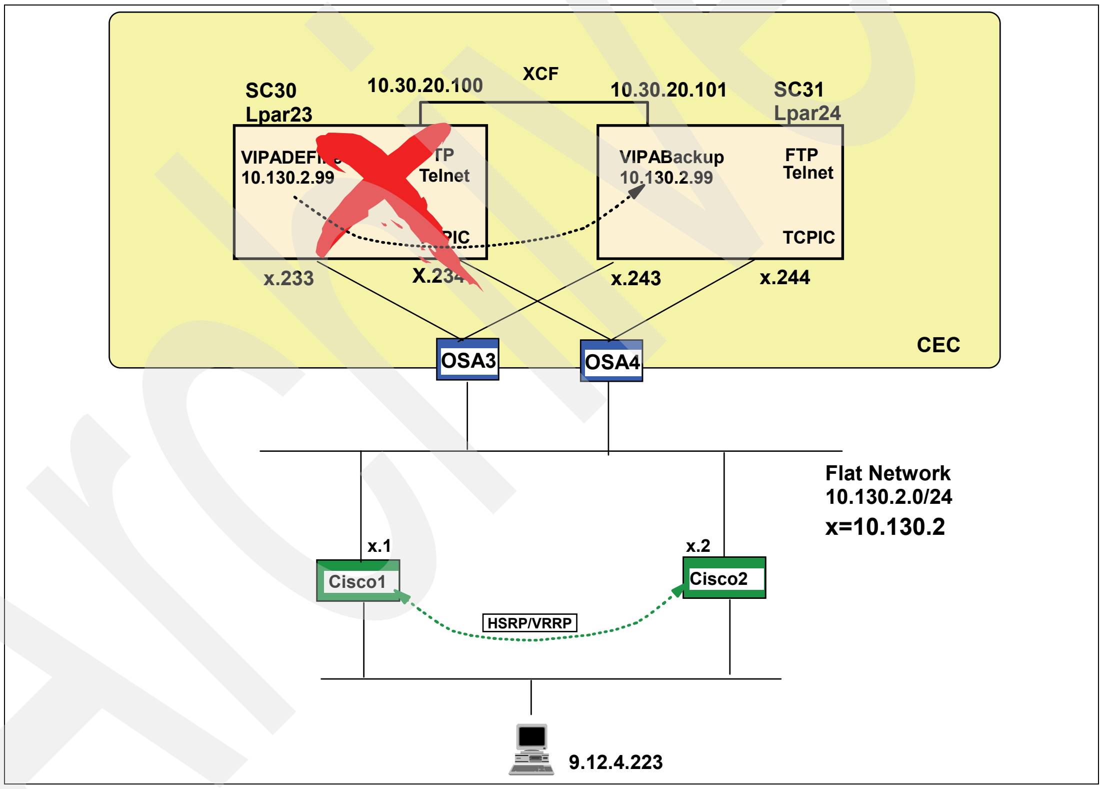


Figure 3-4 High availability for a stack failure

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 3-14 on page 49.

Example 3-14 High availability in fail of the stack

```
P TCPIPC
EZZ3205I SNMP SUBAGENT: SHUTDOWN IN PROGRESS
EZZ0673I AUTOLOG TASK: SHUTDOWN IN PROGRESS
EZZ6008I TELNET STOPPING
```

```
EZZ8301I VIPA 10.130.2.99 TAKEN OVER FROM TCPIPC ON SC30
RO SC31,D TCPIP,TCPIPC,N,VIPADYN
EZD0101I NETSTAT CS V1R7 TCPIPC 549
DYNAMIC VIPA:
IPADDR/PREFIXLEN: 10.130.2.99/32
    STATUS: ACTIVE     ORIGIN: VIPABACKUP       DISTSTAT:
```

This is a disruptive failure mode. Depending on the applications involved, clients may 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 10.130.2.99 (by advertising the DVIPA address with a gratuitous ARP, discussed in 3.5, "ARP takeover" on page 53). We then started new Telnet connections to the DVIPA address 10.130.2.99, which is now running in SC31. We can use NETSTAT operand N,VCRT to verify that these connections are going to DESTXCF 10.30.20.101 (the Dynamic XCF address of the SC31 stack).

Example 3-15 illustrates this situation.

Example 3-15 New connections go to SC31

```
D TCPIP,TCPIPC,N,VCRT
EZD0101I NETSTAT CS V1R7 TCPIPC 554
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.130.2.99..23
  SOURCE:  9.12.4.223..1202
  DESTXCF: 10.30.20.101            1
DEST:      10.130.2.99..23
  SOURCE:  9.12.4.223..1203
  DESTXCF: 10.30.20.101
DEST:      10.130.2.99..23
  SOURCE:  9.12.4.223..3149
  DESTXCF: 10.30.20.101
```

1 DESTXCF 10.30.20.101 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 3-16.

Example 3-16 Start TCPIPC in SC30 take back function

```
S TCPIPC
$HASP100 TCPIPC   ON STCINRDR
IEF695I START TCPIPC   WITH JOBNAME TCPIPC   IS ASSIGNED TO USER TCPIP   , GROUP TCPGRP
EZZ8302I VIPA 10.130.2.99 TAKEN FROM TCPIPC ON SC31
EZZ8303I VIPA 10.130.2.99 GIVEN TO TCPIPC ON SC30
```

3.3.4 Local network failure

We used the Hot Standby Router Protocol (HSRP) of the Cisco routers to implement a high-availability function in our local network. Our local network has two default routes to the outside world, via two Cisco routers. HSRP causes one router to take over both routes if the other router fails. This situation is shown in Figure 3-5.

The TCP/IP environment in our example does not use dynamic routing. It uses two static routes as its default routes and is unable to dynamically change these static routes in response to external routing updates. We used addresses 10.130.2.1 and 10.130.2.2 as our default gateways, and these are the addresses of the two Cisco routers. We also defined each Cisco router to provide the HSRP backup for the other.

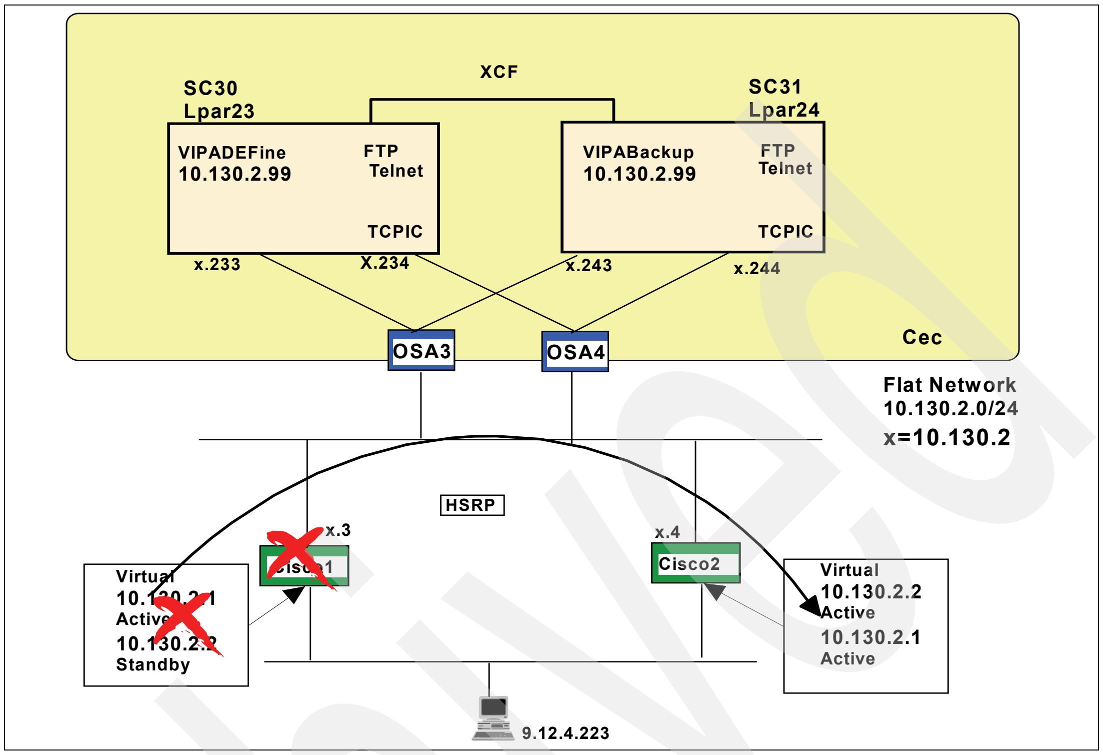


Figure 3-5 High availability out of the flat network

Example 3-17 and Example 3-18 show the two Cisco switch configurations.

Example 3-17 Cisco 1 6509 configuration for HSRP

```
interface Vlan130
 ip address 10.130.2.3 255.255.255.0
 standby 1 ip 10.130.2.1
 standby 1 priority 110
 standby 1 preempt
 standby 2 ip 10.130.2.2
 standby 2 preempt
```

Example 3-18 Cisco 2 6509 configuration for HSRP

```
interface Vlan130
 ip address 10.130.2.4 255.255.255.0
 standby 1 ip 10.130.2.1
 standby 1 preempt
 standby 2 ip 10.130.2.2
 standby 2 priority 110
 standby 2 preempt
```

We issued the command **show standby** to verify the HSRP status. We can see that one switch is backup of the other Virtual IP address. Example 3-19 illustrates this command result.

Example 3-19 Show standby results inside the Cisco SW

```
Cisco 1
Router# sh standby
```

```
Vlan130 - Group 1
  Local state is Active, priority 110, may preempt
Virtual IP address is 10.130.2.1 configured
  Active router is local
  Standby router is 10.130.2.4 expires in 9.008
  Virtual mac address is 0000.0c07.ac01
IP redundancy name is "hsrp-Vl130-1" (default)
Vlan130 - Group 2
  Local state is Standby, priority 100, may preempt
Virtual IP address is 10.130.2.2 configured
  Active router is 10.130.2.4, priority 110 expires in 8.684
  Standby router is local
IP redundancy name is "hsrp-Vl130-2" (default)
Cisco 2
Router# show standby
Vlan130 - Group 1
  Local state is Standby, priority 100, may preempt
Virtual IP address is 10.130.2.1 configured
  Active router is 10.130.2.3, priority 110 expires in 9.248
  Standby router is local
IP redundancy name is "hsrp-Vl130-1" (default)
Vlan130 - Group 2
  Local state is Active, priority 110, may preempt
Virtual IP address is 10.130.2.2 configured
  Active router is local
  Standby router is 10.130.2.3 expires in 9.488
  Virtual mac address is 0000.0c07.ac02
IP redundancy name is "hsrp-Vl130-2" (default)
```

We then simulated a failure on Cisco 1 and verified that IP address 10.130.2.1 was assumed by Cisco 2. Client connections to SC30 and SC31 were not disrupted.

3.4 Debugging tips

When 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. (UNIX System Services commands, such as **onetstat -r**, may also be used.)

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 (with a SOURCEVIPA parameter) may 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 may be useful. This trace consists of IP packets flowing to and from a TCP/IP stack and provides a detailed view of local routing and packet flow.

The following additional steps may be helpful:

- Use a **netstat display vipadcfg** command to verify the DVIPA definition.

- Use a **`netstat display sysplex vipadyn`** command 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.

More debugging information may be found in *z/OS V1R7.0 CS: IP Configuration Reference*, SC31-8775.

3.5 ARP takeover

ARP takeover is a function that allows traffic to be redirected from a failing OSA-Express connection to another OSA-Express connection. This section provides a more-detailed description of the function.

When TCP/IP is started, all the IP addresses connected to a OSA-Express feature in QDIO mode (which is device type MPCIPA in the TCP/IP profile) are dynamically downloaded to the OSA-Express card.

Note: If you want to use the ARP takeover function, you should use your OSA-Express in QDIO mode. If your OSA-Express card is used 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-Express adapter fails while there is a backup OSA-Express adapter 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. Once set up correctly, the fault tolerance provided by the ARP takeover function is automatic.

For our example we used two OSA adapters (in QDIO mode) connected to one Cisco 6509 switch. Figure 3-6 on page 54 illustrates our environment for ARP takeover. We defined CHPID 02 and 03 as OSD-type CHPDs.

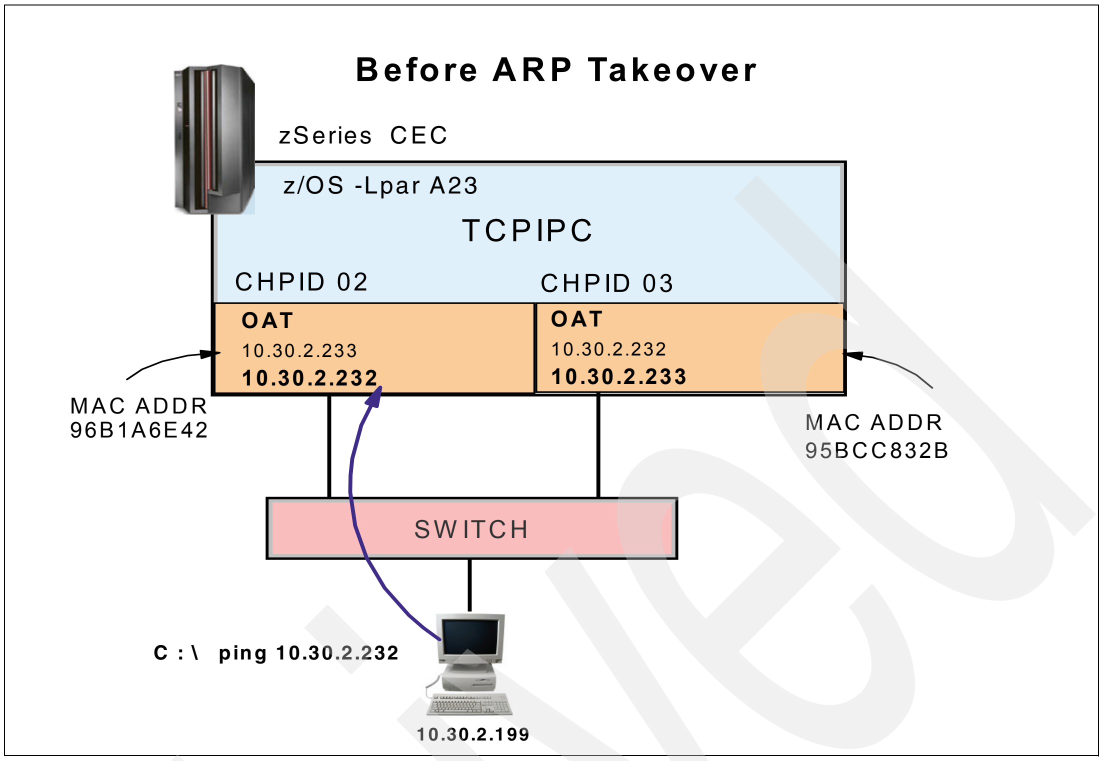


Figure 3-6 ARP takeover scenario

The relevant lines in our TCP/IP PROFILE were:

```
IPCONFIG IQDIOR DATAGRAMFWD
DEVICE OSA2040 MPCIPA; OSD Fast Ethernet Devices on CHPID 02
LINK OSA2040LINK IPAQENET OSA2040
DEVICE OSA2080 MPCIPA ; OSD Fast Ethernet Devices on CHPID 03
LINK OSA2080LINK IPAQENET OSA2080
HOME
10.30.2.232 OSA2040LINK
10.30.2.233 OSA2080LINK
BEGINROUTES
ROUTE 10.30.2.0/24 = OSA2040LINK MTU 1492
ROUTE 10.30.2.0/24 = OSA2080LINK MTU 1492
ROUTE DEFAULT 10.30.2.1 OSA2040LINK MTU 1492
ROUTE DEFAULT 10.30.2.1 OSA2080LINK MTU 1492
ENDROUTES
```

We used the Cisco **set port host** command to configure the router:

```
6500-top> (enable) set port host 1/1
2005 Oct 18:04:01:32 %SYS-6-CFG_CHG:Module 1 block changed by Console//
Port(s) 1/1 channel mode set to off.
Warning: Spantree port fast start should only be enabled on ports connected to
a single host. Connecting hubs, concentrators, switches, bridges, etc. to a
fast start port can cause temporary spanning tree loops. Use with caution.
Spantree port 1/1 fast start enabled.
Port(s) 1/1 trunk mode set to off
```

We verified that ARP takeover worked by performing two different tasks:

- Pulling the CAT5 cable from the OSA-Express feature
- Stopping the device in the TCP/IP stack

Figure 3-7 shows our environment after pulling the CAT5 cable, and the route that a **ping** takes as a result of this.

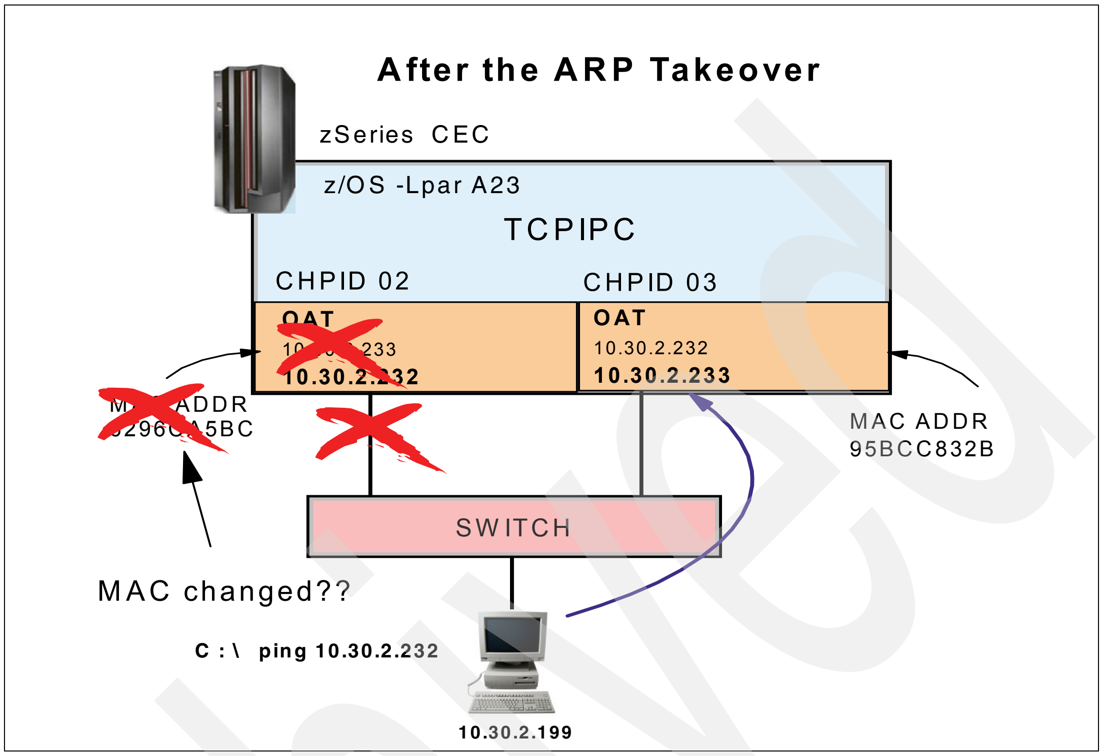


Figure 3-7 ARP takeover after fail in OSA 02

We displayed the contents of the ARP cache on a workstation after pinging each IP address defined in TCPIPC before any error condition was introduced:

```
C:\>arp -a
Interface: 10.30.2.199 on Interface 0x3000004
Internet Address    Physical     Address Type
  10.30.2.233      95BCC832B      dynamic
  10.30.2.232      96B1A6E42      dynamic
```

Notice that 10.30.2.232 is currently assigned to the MAC address associated with CHPID 02. Also, 10.30.2.233 is currently assigned to the MAC address associated with CHPID 03.

We displayed the contents of the ARP table from the OSA-Express cards before any error condition was introduced:

```
MVS TCP/IP NETSTAT CS V1R7 TCPIP Name: TCPIPC 15:34:44
Querying ARP cache for address 10.30.2.233
Link: OSA2080LINK ETHERNET: 00095BCC832B
Querying ARP cache for address 10.30.2.232
Link: OSA2040LINK ETHERNET: 00096B1A6E42
Querying ARP cache for address 10.30.2.199
Link: OSA2040LINK ETHERNET: 0004AC1D18E9
```

We then pulled the CAT5 cable connected to the OSA-Express adapter at CHPID 02. This produced the following messages on the z/OS console:

```
EZZ4329I LINK OSA2080LINK HAS TAKEN OVER ARP RESPONSIBILITY FOR
INACTIVE LINK OSA2040LINKLINK
EZZ4311I LINK OSA2040LINK HAS FAILED ON DEVICE OSA2040
```

A new display of the contents of the ARP cache of the workstation (after pulling the cable) was:

```
C:\>arp -a
Interface: 10.30.2.199 on Interface 0x3000004
Internet Address    Physical Address    Type
  10.30.2.232      00-09-5B-0C-83-2B  dynamic
  10.30.2.233      00-09-5B-0C-83-2B  dynamic
```

Notice that both IP addresses now point to the same MAC address, which is associated with CHPID 03. Nothing had to be done at the workstation to update the ARP cache. The TCP/IP running on z/OS initiated a gratuitous ARP to all hosts on the LAN when it was notified that the connection on CHPID 02 was lost.

The ARP status in z/OS (immediately after pulling the cable) was:

```
MVS TCP/IP NETSTAT CS V1R7 TCPIP Name: TCPIPC 15:37:16
Querying ARP cache for address 10.30.2.232
Link: OSA2040LINK ETHERNET: 00096B1A6E42
Querying ARP cache for address 10.30.2.199
Link: OSA2080LINK ETHERNET: 0004AC1D18E9
Querying ARP cache for address 10.30.2.233
Link: OSA2080LINK ETHERNET: 00095BCC832B
```

Notice that there is no change yet in the MAC addresses in this ARP table. However, after a few minutes the ARP table contained entries associating the same IP address with both MAC addresses:

```
MVS TCP/IP NETSTAT CS V1R7 TCPIP Name: TCPIPC 15:39:23
Querying ARP cache for address 10.30.2.232
Link: OSA2040LINK ETHERNET: 0006296CA5BC
Querying ARP cache for address 10.30.2.199
Link: OSA2080LINK ETHERNET: 0004AC1D18E9
Querying ARP cache for address 10.30.2.232
Link: OSA2040LINK ETHERNET: 00095BCC832B
Querying ARP cache for address 10.30.2.233
Link: OSA2080LINK ETHERNET: 00095BCC832B
```

We cleaned up the ARP table by using the **purgecache** command available with z/OS:

```
V TCPIP,TCPIPC,PURGECACHE,OSA2080LINK
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,PURGECACHE,OSA2080LINK
EZZ9786I PURGECACHE PROCESSED FOR LINK OSA2080LINK
EZZ0053I COMMAND PURGECACHE COMPLETED SUCCESSFULLY
```

When the CAT5 cable was plugged back into the switch, we received the following z/OS messages:

```
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE OSA2040
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE OSA2040
```

Another gratuitous ARP was issued by TCPIPC to the hosts on the LAN that updated the ARP cache with the correct MAC addresses. On our workstation, the gratuitous ARP updated only existing entries in ARP cache. It did not create entries for IP addresses that were not currently in the ARP cache.

As our second exercise, we created an error condition by stopping the device in the TCP/IP stack using this command:

```
V TCPIP,TCPIPC,STOP,OSA2040
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,STOP,OSA2040
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
```

```
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE OSA2040
EZZ4329I LINK OSA2080LINK HAS TAKEN OVER ARP RESPONSIBILITY FOR
INACTIVE LINK OSA2040LINK
```

A gratuitous ARP was sent, and the changes to the ARP tables were identical to those shown previously. We started the device again with the following command:

```
V TCPIP,TCPIPC,START,OSA2040
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,START,OSA2040
EZZ0053I COMMAND VARY START COMPLETED SUCCESSFULLY
EZZ4313I INITIALIZATION COMPLETE FOR DEVICE OSA2040
```

Once again, a gratuitous ARP was sent to update existing ARP cache entries to the correct MAC addresses.

4

VIPA with dynamic routing

This chapter introduces and provides implementation examples for using dynamic routing in your z/OS Communications Server environment. The dynamic routing protocols supported in z/OS are OSPF and RIP. These dynamic routing protocols are implemented and maintained by OMPROUTE. This chapter is focused on OSPF as the recommended dynamic routing protocol for z/OS. For more details regarding implementation and concepts of OMPROUTE, please refer to *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 1 - Base Functions, Connectivity, and Routing*, SG24-7169.

This chapter discusses the following.

Section	Topic
4.1, "Basic concepts" on page 60	Discusses the basic concepts of high availability using dynamic routing
4.2, "Design example for dynamic routing" on page 68	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

4.1 Basic concepts

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 cost effectively be achieved. An important benefit of using a dynamic routing protocol in the 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.

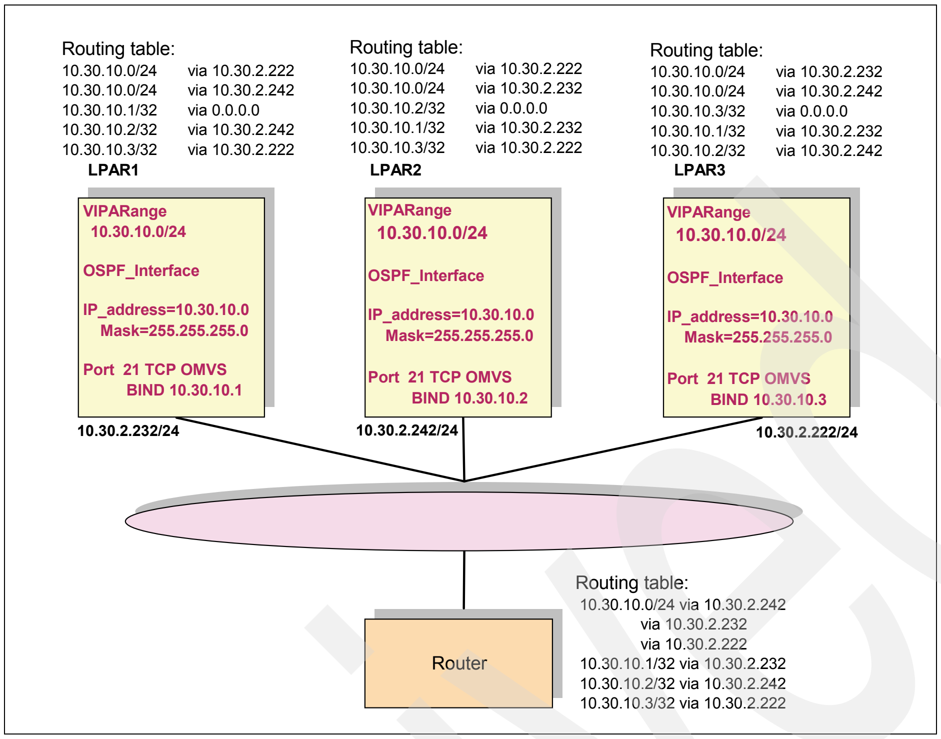
Open Shortest Path First (OSPF) is a link-state routing protocol. Every router has a full map of the entire network topology. Changes to the network topology (due to link-state changes or outages) is propagated throughout the network. Every router that receives updates must then recompute its shortest 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/unreachable destination.

For further details regarding dynamic routing protocols, please refer to *z/OS 1.7 Implementation Guide - Base functions, connectivity and routing.*

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 as a generic entry by the OSPF_Interface statement. 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 the VIPA host address and the IP subnet that belongs to the VIPA address.



Routing table:
10.30.10.0/24 via 10.30.2.222
10.30.10.0/24 via 10.30.2.242
10.30.10.1/32 via 0.0.0.0
10.30.10.2/32 via 10.30.2.242
10.30.10.3/32 via 10.30.2.222
LPAR1
VIPARange 10.30.10.0/24
OSPF_Interface
IP_address=10.30.10.0 Mask=255.255.255.0
Port 21 TCP OMVS BIND 10.30.10.1
10.30.2.232/24
Routing table:
10.30.10.0/24 via 10.30.2.222
10.30.10.0/24 via 10.30.2.232
10.30.10.2/32 via 0.0.0.0
10.30.10.1/32 via 10.30.2.232
10.30.10.3/32 via 10.30.2.222
LPAR2
VIPARange 10.30.10.0/24
OSPF_Interface
IP_address=10.30.10.0 Mask=255.255.255.0
Port 21 TCP OMVS BIND 10.30.10.2
10.30.2.242/24
Routing table:
10.30.10.0/24 via 10.30.2.232
10.30.10.0/24 via 10.30.2.242
10.30.10.3/32 via 0.0.0.0
10.30.10.1/32 via 10.30.2.232
10.30.10.2/32 via 10.30.2.242
LPAR3
VIPARange 10.30.10.0/24
OSPF_Interface
IP_address=10.30.10.0 Mask=255.255.255.0
Port 21 TCP OMVS BIND 10.30.10.3
10.30.2.222/24
Router
Routing table:
10.30.10.0/24 via 10.30.2.242
via 10.30.2.232
via 10.30.2.222
10.30.10.1/32 via 10.30.2.232
10.30.10.2/32 via 10.30.2.242
10.30.10.3/32 via 10.30.2.222

Figure 4-1 VIPA subnet across LPARs

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.

What happens if one of the FTP servers in this setup is taken down and the VIPA address deleted?

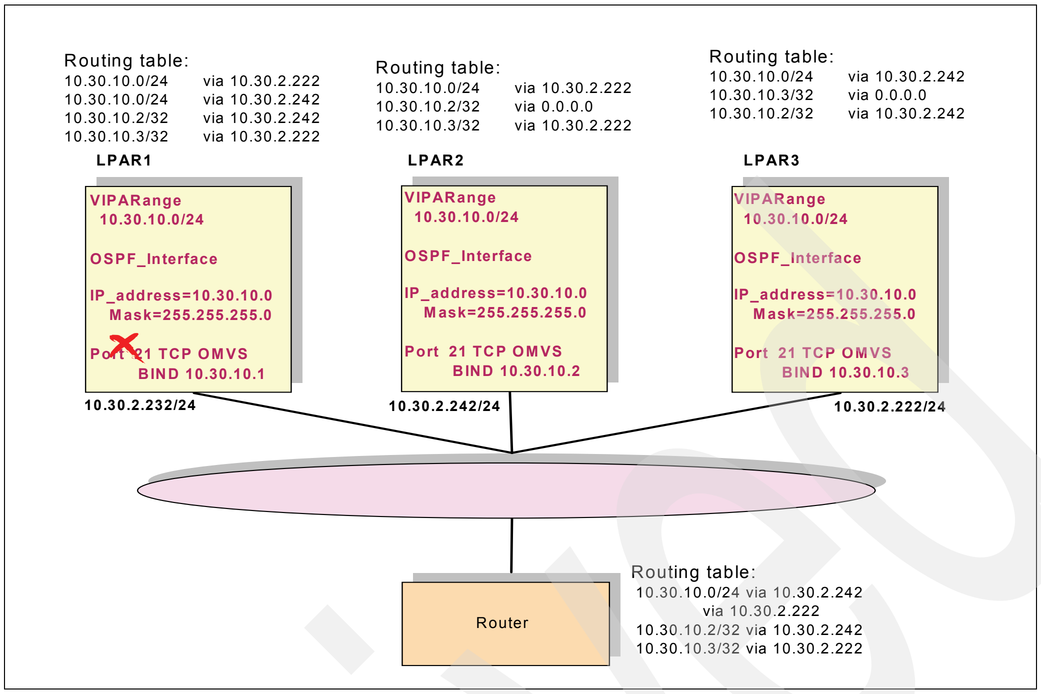

Routing table:
10.30.10.0/24 via 10.30.2.222
10.30.10.0/24 via 10.30.2.242
10.30.10.2/32 via 10.30.2.242
10.30.10.3/32 via 10.30.2.222
LPAR1
VIPARange 10.30.10.0/24
OSPF_Interface
IP_address=10.30.10.0 Mask=255.255.255.0
Port 21 TCP OMVS BIND 10.30.10.1
10.30.2.232/24
Routing table:
10.30.10.0/24 via 10.30.2.222
10.30.10.2/32 via 0.0.0.0
10.30.10.3/32 via 10.30.2.222
LPAR2
VIPARange 10.30.10.0/24
OSPF_Interface
IP_address=10.30.10.0 Mask=255.255.255.0
Port 21 TCP OMVS BIND 10.30.10.2
10.30.2.242/24
Routing table:
10.30.10.0/24 via 10.30.2.242
10.30.10.3/32 via 0.0.0.0
10.30.10.2/32 via 10.30.2.242
LPAR3
VIPARange 10.30.10.0/24
OSPF_Interface
IP_address=10.30.10.0 Mask=255.255.255.0
Port 21 TCP OMVS BIND 10.30.10.3
10.30.2.222/24
Router
Routing table:
10.30.10.0/24 via 10.30.2.242
via 10.30.2.222
10.30.10.2/32 via 10.30.2.242
10.30.10.3/32 via 10.30.2.222

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 10.30.10.1. 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 (10.30.10.1). The request arrives at the router, but the router does not have a route to 10.30.10.1. The next best match in the router is an equal cost entry (10.30.10.0/24) pointing to LPAR2 and LPAR3. The routing tables in LPAR2 and LPAR3 do not have a route to 10.30.10.1 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 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. In our example the OSPF_Interface statement looks like this:

```
OSPF_Interface IP_address=10.30.10.*
           subnet_mask=255.255.255.0
           Attaches_To_Area=0.0.0.0
           Advertise_VIPA_Routes=HOST_ONLY
           Cost0=10
           MTU=65535;
```

By not advertising the VIPA subnet the routing tables now appear as shown in Figure 4-3 on page 63.

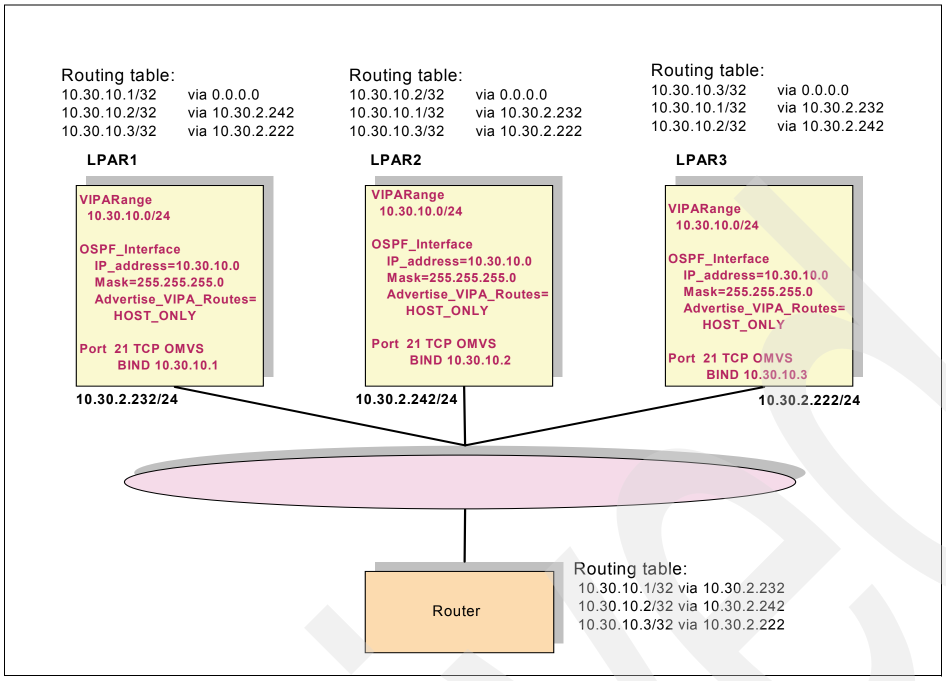

Routing table:
10.30.10.1/32 via 0.0.0.0
10.30.10.2/32 via 10.30.2.242
10.30.10.3/32 via 10.30.2.222
Routing table:
10.30.10.2/32 via 0.0.0.0
10.30.10.1/32 via 10.30.2.232
10.30.10.3/32 via 10.30.2.222
Routing table:
10.30.10.3/32 via 0.0.0.0
10.30.10.1/32 via 10.30.2.232
10.30.10.2/32 via 10.30.2.242
LPAR1
LPAR2
LPAR3
VIPARange 10.30.10.0/24
OSPF_Interface IP_address=10.30.10.0 Mask=255.255.255.0 Advertise_VIPA_Routes= HOST_ONLY
Port 21 TCP OMVS BIND 10.30.10.1
Port 21 TCP OMVS BIND 10.30.10.2
Port 21 TCP OMVS BIND 10.30.10.3
10.30.2.232/24
10.30.2.242/24
10.30.2.222/24
Router
Routing table:
10.30.10.1/32 via 10.30.2.232
10.30.10.2/32 via 10.30.2.242
10.30.10.3/32 via 10.30.2.222

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.

Attention: Pre z/OS V1R7: PQ82792 prevents OMPROUTE from including the VIPA subnets in OSPF advertisements. The function is controlled by `Subnet=NoVipaSubnet`.

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. An example is shown in Figure 4-4 on page 64.

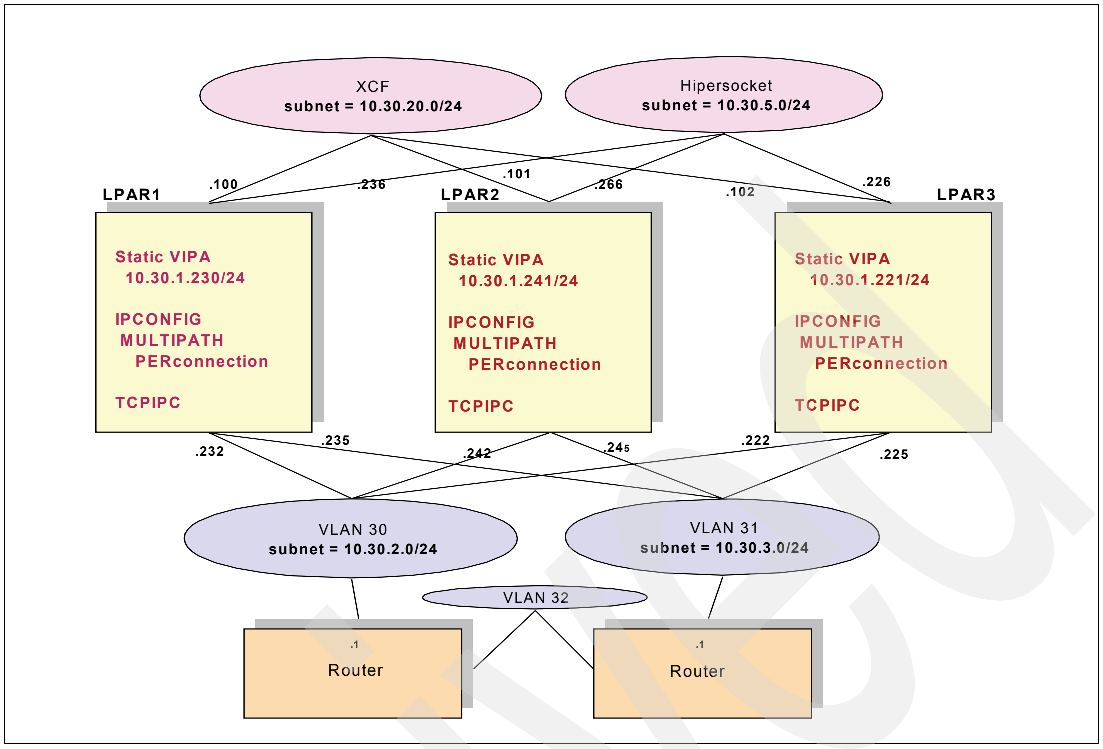


Figure 4-4 Sample network with multiple paths between LPARs

From a routing perspective, all links with 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** to display the routing information:

```
SC30:/u/cs05>netstat -p tcpipc -r
MVS TCP/IP NETSTAT CS V1R7       TCPIP Name: TCPIPC         13:37:44
IPv4 Destinations
Destination        Gateway          Flags     Refcnt  Interface
-----------        -------          -----     ------  ---------
10.30.1.221/32     10.30.2.222      UGHO      000001  OSA2080LNK 1
10.30.1.221/32     10.30.3.225      UGHO      000000  OSA20E0LNK 1
10.30.1.221/32     10.30.5.226      UGHO      000000  IUTIQDF6LNK 1
10.30.1.221/32     10.30.20.102     UGHO      000000  IQDIOLNKOA1E1464 1
10.30.1.230/32     0.0.0.0          UH        000000  STAVIPA1LNK
10.30.1.241/32     10.30.2.242      UGHO      000001  OSA2080LNK 2
10.30.1.241/32     10.30.3.245      UGHO      000000  OSA20E0LNK 2
10.30.1.241/32     10.30.5.246      UGHO      000000  IUTIQDF6LNK 2
10.30.1.241/32     10.30.20.101     UGHO      000000  IQDIOLNKOA1E1464 2
```

Note the 1 four equal cost links to VIPA 10.30.1.221 and the 2 four equal cost links to VIPA 10.30.1.241. There might be situations where we require more control over which links are used. 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:

- OSA as the primary path for traffic between LPARs in different CECs
- HiperSockets for the primary path between LPARs in the same CEC
- XCF as the backup and last path available

This is achieved 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 within...	**Primary**	**Secondary**	**Third**
same LPAR	HiperSockets	OSA	XCF
same CEC	HiperSockets	OSA	XCF
across CECs	OSA	XCF	

Installation requirements differ, of course, and this example is intended only to illustrate a general technique. We set our OSPF configuration as follows:

```
OSPF_Interface IP_address=10.30.2.232
           Subnet_mask=255.255.255.0
           Name=OSA2080LNK 1
           Router_Priority=0
           Attaches_To_Area=0.0.0.0
           Cost0=10
           MTU=1492;

OSPF_Interface IP_address=10.30.3.235
           Subnet_mask=255.255.255.0
           Name=OSA20E0LNK 2
           Router_Priority=0
           Attaches_To_Area=0.0.0.0
           Cost0=10
           MTU=1492;

OSPF_Interface IP_address=10.30.5.236
           Subnet_mask=255.255.255.0
           Name=IUTIQDF6LNK 3
           Attaches_To_Area=0.0.0.0
           Cost0=5
           MTU=8192;

OSPF_Interface IP_address=10.30.20.* 4
           Subnet_mask=255.255.255.0
           Attaches_To_Area=0.0.0.0
           Cost0=15
           MTU=8192;
```

Where 1 is the OSA link in VLAN 30, 2 is the OSA link in VLAN 31, 3 is the HiperSocket link, and 4 is the XCF Link. The routing table is different after the new OSPF cost values have been implemented:

```
SC30:/u/cs05>netstat -p tcpipc -r
MVS TCP/IP NETSTAT CS V1R7       TCPIP Name: TCPIPC         16:47:11
IPv4 Destinations
Destination         Gateway          Flags     Refcnt  Interface
-----------         -------          -----     ------  ---------
10.30.1.221/32      10.30.5.226      UGHO      000001  IUTIQDF6LNK 1
10.30.1.230/32      0.0.0.0          UH        000000  STAVIPA1LNK
10.30.1.241/32      10.30.5.246      UGHO      000001  IUTIQDF6LNK 2
```

In this case, 1 HiperSocket is the preferred path towards VIPA 10.30.1.221, and 2 HiperSocket is preferred path towards VIPA 10.30.1.241. If the HiperSocket path becomes unavailable this affects the network topology and causes OSPF to recalculate the routing table. We disabled the HiperSocket path as follows:

```
V TCPIP,TCPIPC,STO,IUTIQDF6
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,STO,IUTIQDF6
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE IUTIQDF6
EZZ7921I OSPF ADJACENCY FAILURE, NEIGHBOR 10.30.5.246, OLD STATE 128, NEW STATE 1, EVENT 11
EZZ7921I OSPF ADJACENCY FAILURE, NEIGHBOR 10.30.5.226, OLD STATE 128, NEW STATE 1, EVENT 11
```

The routing table now looked as follows:

```
SC30:/u/cs05>netstat -p tcpipc -r
MVS TCP/IP NETSTAT CS V1R7       TCPIP Name: TCPIPC         17:37:45
IPv4 Destinations
Destination        Gateway         Flags     Refcnt  Interface
-----------        -------         -----     ------  ---------
10.30.1.221/32     10.30.2.222     UGHO      000001  OSA2080LNK 1
10.30.1.221/32     10.30.3.225     UGHO      000000  OSA20E0LNK 1
10.30.1.230/32     0.0.0.0         UH        000000  STAVIPA1LNK
10.30.1.241/32     10.30.2.242     UGHO      000001  OSA2080LNK 2
10.30.1.241/32     10.30.3.245     UGHO      000000  OSA20E0LNK 2
```

Now 1 two equal cost OSA links are the preferred paths towards VIPA 10.30.1.221, and 2 two equal cost OSA links are the preferred paths towards VIPA 10.30.1.241. There are two OSA adapters available and datagrams will be forwarded round-robin.

If the OSA adapter path becomes unavailable, it again affects the network topology and OSPF again recalculates the routing table. We removed the OSA adapters as follows:

```
V TCPIP,TCPIPC,STO,OSA2080
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,STO,OSA2080
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE OSA2080
EZZ7921I OSPF ADJACENCY FAILURE, NEIGHBOR 10.30.2.1, OLD STATE 128, NEW STATE 1, EVENT 11
V TCPIP,TCPIPC,STO,OSA20E0
EZZ0060I PROCESSING COMMAND: VARY TCPIP,TCPIPC,STO,OSA20E0
EZZ0053I COMMAND VARY STOP COMPLETED SUCCESSFULLY
EZZ4315I DEACTIVATION COMPLETE FOR DEVICE OSA20E0
EZZ7921I OSPF ADJACENCY FAILURE, NEIGHBOR 10.30.3.1, OLD STATE 128, NEW STATE 1, EVENT 11
```

This, in turn, caused the routine table to look as follows:

```
SC30:/u/cs05>netstat -p tcpipc -r
MVS TCP/IP NETSTAT CS V1R7       TCPIP Name: TCPIPC         17:45:34
IPv4 Destinations
Destination        Gateway         Flags     Refcnt  Interface
-----------        -------         -----     ------  ---------
10.30.1.221/32     10.30.20.102    UGHO      000001  IQDIOLNK0A1E1464 1
10.30.1.230/32     0.0.0.0         UH        000000  STAVIPA1LNK
10.30.1.241/32     10.30.20.101    UGHO      000001  IQDIOLNK0A1E1464 2
```

We now have 1 XCF (HiperSocket) as the preferred path towards VIPA 10.30.1.221, and 2 XCF (HiperSocket) as the preferred path towards VIPA 10.30.1.241.

Because we are in an z/OS V1R7 environment and because we have our LPARs running in the same CEC, the link chosen for the XCF path becomes the HiperSocket path. For pre z/OS V1R6 environments this path can only be the DYNAMICXCF path itself.

Note: Note that the HiperSocket that is used for the XCF path is not necessarily the same HiperSocket path as used as our primary path. The path chosen for XCF depends of the VTAM start option IQDCHPID.

This has been an example of using dynamic routing to automatically provide several levels of backup for IP traffic within a single CEC. The only significant cost involved is the time to plan and test the required definitions.

4.1.4 Generic OSPF address definitions

As seen in the previous examples, we can define a link to OSPF by specifying the IP address associated with the link. When the IP address is activated, TCP/IP sends information about the IP address to OMPROUTE. OMPROUTE then looks in its configuration file to see if there is a statement describing how this IP address is to be treated by OSPF.

If the IP address matches an OSPF_Interface statement, then the options associated with this definition are used by OSPF to advertise information about the IP address to its neighbors. As an example, consider two DVIPA addresses in a subnet. One way to configure them to OPSF is by having each DVIPA address be represented by an OSPF_Interface statement:

```
OSPF_Interface IP_address=10.30.10.20
           Subnet_mask=255.255.255.0
           Name=VIPA1 1
           Attaches_To_Area=0.0.0.0
           Advertise_VIPA_Routes=HOST_ONLY
           Cost0=10
           MTU=65535;

OSPF_Interface IP_address=10.30.10.21
           Subnet_mask=255.255.255.0
           Name=VIPA2 1
           Attaches_To_Area=0.0.0.0
           Advertise_VIPA_Routes=HOST_ONLY
           Cost0=10
           MTU=65535;
```

Note that 1 link-names are updated with valid link-names when the DVIPA is created. The name parameter is not validated by OMPROUTE because the real name is defined when the DVIPA is created within the TCP/IP stack. The name parameter is then overwritten with the correct name of the DVIPA link name.

If we dedicate one DVIPA per application we can quickly use a large number of IP addresses, and this would require many OSPF_Interface statements. Maintaining the definitions becomes an administrative burden that is subject to simple errors. Another way to configure the DVIPAs is by using a wildcard as part of the IP address. One of the benefits of using wildcards is that the same configuration can be used across TCP/IP stacks to minimize maintenance of OMPROUTE. The OMPROUTE configuration then looks like this:

```
OSPF_Interface IP_address=10.30.10.*
           Subnet_mask=255.255.255.0
           Attaches_To_Area=0.0.0.0
           Advertise_VIPA_Routes=HOST_ONLY
           Cost0=10
           MTU=65535;
```

All DVIPAs in subnet 10.30.10.0/24 match this OMPROUTE configuration. We did not include the name parameter because this parameter is added automatically when the DVIPA is created.

Note: To ensure availability of your environment it is crucial that your OSPF definitions match the intention of your IP infrastructure; otherwise you will 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 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. 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 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.

Figure 4-5 illustrates a totally stubby area setup.

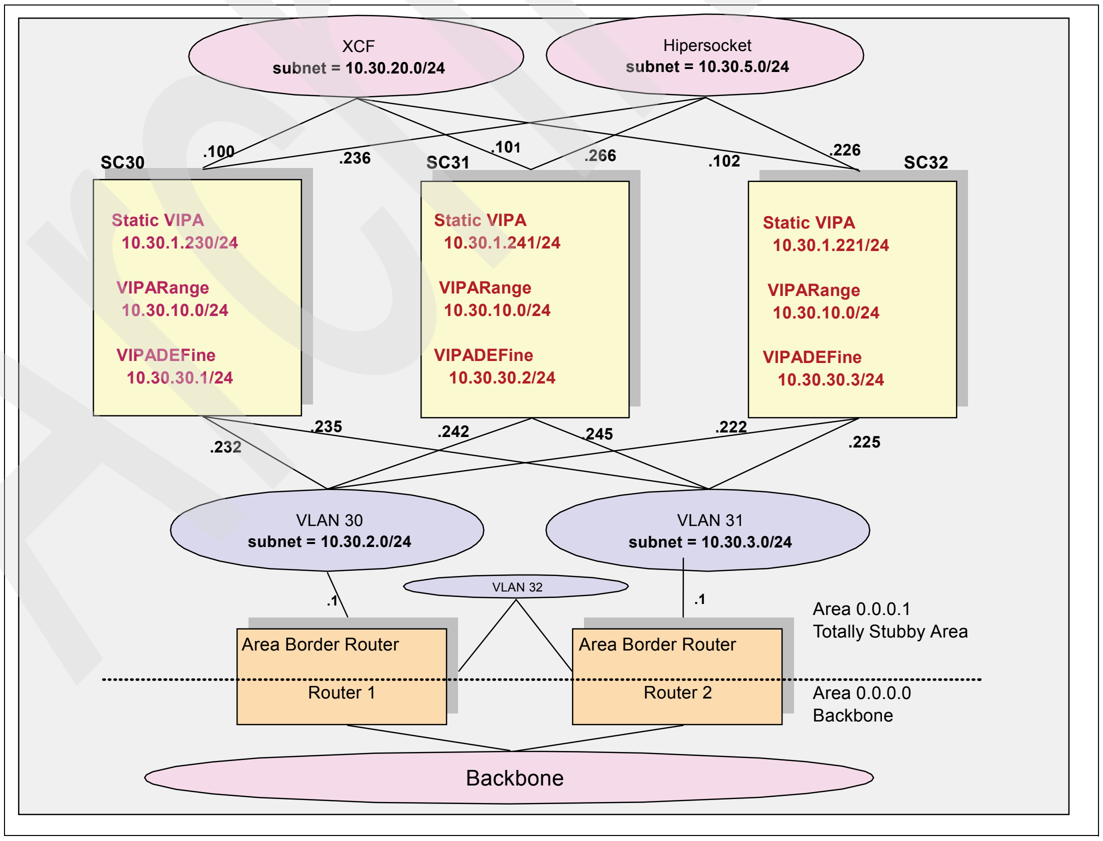


Figure 4-5 Sample network with z/OS being defined as a totally stubby area

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

The tasks used to implement z/OS sysplex as a totally stubby area are:

1. Create PROFILE statements for the stacks.
2. Create OMPROUTE configuration files.
3. Create definitions for the border routers.

4.2.1 PROFILE statements

Example 4-1 lists the relevant TCPIP PROFILE statements for system SC30. Below it, Table 4-2 on page 70 summarizes the differences for SC31 and SC32.

Example 4-1 TCP/IP profile for SC30

```
GLOBALCONFIG NOTCPIPSTATISTICS
;
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
;
DYNAMICXCF 10.30.20.100 255.255.255.0 8
;
SOMAXCONN 10
;
TCPCONFIG TCPSENDBFRSIZE 16K TCPRCVBUFRSIZE 16K SENDGARBAGE FALSE
TCPCONFIG RESTRICTLOWPORTS
;
UDPCONFIG RESTRICTLOWPORTS
;
 DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 30
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 31
;
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6LNK  IPAQIDIO      IUTIQDF6
;
 DEVICE STAVIPA1    VIRTUAL 0
 LINK   STAVIPA1LNK VIRTUAL 0     STAVIPA1
;
VIPADynamic
 VIPARange  DEFINE MOVEable NONDISRUPTive 255.255.255.0 10.30.10.0
 VIPADEFine MOVEable IMMEDiate 255.255.255.255 10.30.30.1
ENDVIPADYNAMIC
;
HOME
   10.30.1.230   STAVIPA1LNK
   10.30.2.232   OSA2080LNK
   10.30.3.235   OSA20E0LNK
   10.30.5.236   IUTIQDF6LNK
;
AUTOLOG 5
   FTPDC  JOBNAME FTPDC1
   OMPC
ENDAUTOLOG
;
PORT
```

```
    20 TCP *  NOAUTOLOG         ; FTP Server
    21 TCP OMVS BIND 10.30.10.1  ; control port
;
SACONFIG ENABLED COMMUNITY public AGENT 161
;
START IUTIQDF6
START OSA2080
START OSA20E0
```

Table 4-2 summarizes the differences between the TCPIP PROFILE for SC30 (shown above) and those of SC31 and SC32.

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

Statement	SC30	SC31	SC32
DYNAMICXCF	10.30.20.100	10.30.20.101	10.30.20.102
VIPADEFine	10.30.30.1	10.30.30.2	10.30.30.3
HOME			
STAVIPA1LNK	10.30.1.230	10.30.1.240	10.30.1.220
OSA2080LNK	10.30.2.232	10.30.2.242	10.30.2.222
OSA20E0LNK	10.30.3.235	10.30.3.245	10.30.3.225
IUTIQDF6LNK	10.30.5.236	10.30.5.246	10.30.5.226
PORT			
21 TCP OMVS BIND	10.30.10.1	10.30.10.2	10.30.10.3

4.2.2 OMPROUTE definitions

The OMPROUTE configuration file for SC30 is shown in Example 4-2. Table 4-3 on page 71 summarizes the differences for SC31 and SC32.

Example 4-2 OMPROUTE configuration file for SC30

```
Area Area_Number=0.0.0.1
     Stub_Area=YES 1
     Authentication_type=None
     Import_Summaries=No; 2
;
OSPF RouterID=10.30.1.230;
;
Routesa_Config Enabled=No;
;
OSPF_Interface IP_address=10.30.1.230 3
          Subnet_mask=255.255.255.252
          Name=STAVIPA1LNK
          Attaches_To_Area=0.0.0.1
          Advertise_VIPA_Routes=HOST_ONLY
          Cost0=10
          MTU=65535;
;
OSPF_Interface IP_address=10.30.10.* 4
           Subnet_mask=255.255.255.0
           Attaches_To_Area=0.0.0.1
           Advertise_VIPA_Routes=HOST_ONLY
           Cost0=10
           MTU=65535;
;
OSPF_Interface IP_address=10.30.30.* 5
```

```
          Subnet_mask=255.255.255.0
          Attaches_To_Area=0.0.0.1
          Advertise_VIPA_Routes=HOST_ONLY
          Cost0=10
          MTU=65535;
;
OSPF_Interface IP_address=10.30.2.* 6
          Subnet_mask=255.255.255.0
          Router_Priority=0
          Attaches_To_Area=0.0.0.1
          Cost0=10
          MTU=1492;
;
OSPF_Interface IP_address=10.30.3.* 7
          Subnet_mask=255.255.255.0
          Router_Priority=0
          Attaches_To_Area=0.0.0.1
          Cost0=10
          MTU=1492;
;
OSPF_Interface IP_address=10.30.5.* 8
          Subnet_mask=255.255.255.0
          Attaches_To_Area=0.0.0.1
          Cost0=5
          MTU=8192;
;
OSPF_Interface IP_address=10.30.20.* 9
          Subnet_mask=255.255.255.0
          Attaches_To_Area=0.0.0.1
          Cost0=15
          MTU=8192;
;
AS_Boundary_routing
  Import_Direct_Routes=yes;
```

Key elements in these definitions include the following:

- 1 Indicates this is an OSPF Stub Area
- 2 No summaries indicates totally stubby area
- 3 Static VIPA
- 4 DVIPA - VIPARange
- 5 DVIPA - VIPADefine/VIPABackup
- 6 OSA adapter in VLAN 30
- 7 OSA adapter in VLAN 31
- 8 HiperSocket link
- 9 XCF link

Table 4-3 summarizes the differences between the OMPROUTE configurations for SC30 (shown above) 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	10.30.1.230	10.30.1.241	10.30.1.221
OSPF_Interface IP_address	10.30.1.230	10.30.1.241	10.30.1.221

4.2.3 Router definitions

The router configurations are critical, of course. The definitions for router 1 are shown in Example 4-3, Example 4-4, and Example 4-5.

Example 4-3 OSPF configuration in router 1

```
router ospf 300
 router-id 10.30.2.1
 log-adjacency-changes
 area 1 stub no-summary 1
 network 10.30.0.0 0.0.255.255 area 1
 network 10.200.1.0 0.0.0.255 area 0
```

1 The no-summary parameter indicates that this is a totally stubby area.

Example 4-4 Configuration of VLAN30 in router 1

```
interface Vlan30
 ip address 10.30.2.1 255.255.255.0
```

Example 4-5 Configuration of VLAN32 in router 1

```
interface Vlan32
 ip address 10.30.32.1 255.255.255.0
 ip ospf cost 1
```

The configuration of router 2 is shown in Example 4-6, Example 4-7, and Example 4-8.

Example 4-6 OSPF configuration in router 2

```
router ospf 300
 router-id 10.30.3.1
 log-adjacency-changes
 area 1 stub no-summary 1
 network 10.30.0.0 0.0.255.255 area 1
 network 10.200.1.0 0.0.0.255 area 0
```

1 The no-summary parameter makes the stub area a totally stubby area.

Example 4-7 Configuration of VLAN31 in router 2

```
interface Vlan31
 ip address 10.30.3.1 255.255.255.0
```

Example 4-8 Configuration of VLAN32 in router 2

```
interface Vlan32
 ip address 10.30.32.2 255.255.255.0
 ip ospf cost 1
```

4.2.4 Verification

We used the following steps to verify that our configuration worked correctly:

1. Display the interfaces belonging to the TCP/IP stacks.
2. List the global OSPF configurations in the three systems.
3. List the OSPF neighbors for each of the three systems.

4. List the routing tables in each of the three systems.
5. List the OSPF neighbors for the two routers.

Display interfaces

Example 4-9, Example 4-10, and Example 4-11 on page 74 illustrate displaying the interfaces for the three systems.

Example 4-9 IP homelist in SC30

```
D TCPIP,TCPIPC,N,HOME
EZD0101I NETSTAT CS V1R7 TCPIPC 335
HOME ADDRESS LIST:
LINKNAME:   STAVIPA1LNK
  ADDRESS:  10.30.1.230
    FLAGS:  PRIMARY
LINKNAME:   OSA2080LNK
  ADDRESS:  10.30.2.232
    FLAGS:
LINKNAME:   OSA20E0LNK
  ADDRESS:  10.30.3.235
    FLAGS:
LINKNAME:   IUTIQDF6LNK
  ADDRESS:  10.30.5.236
    FLAGS:
LINKNAME:   EZASAMEMVS
  ADDRESS:  10.30.20.100
    FLAGS:
LINKNAME:   IQDIOLNK0A1E1464
  ADDRESS:  10.30.20.100
    FLAGS:
LINKNAME:   VIPL0A1E1E01
  ADDRESS:  10.30.30.1
    FLAGS:
LINKNAME:   VIPL0A1E0A01
  ADDRESS:  10.30.10.1
    FLAGS:
LINKNAME:   LOOPBACK
  ADDRESS:  127.0.0.1
    FLAGS:
INTFNAME:   LOOPBACK6
  ADDRESS:  ::1
    TYPE:   LOOPBACK
    FLAGS:
10 OF 10 RECORDS DISPLAYED
```

Example 4-10 IP homelist in SC31

```
D TCPIP,TCPIPC,N,HOME
EZD0101I NETSTAT CS V1R7 TCPIPC 994
HOME ADDRESS LIST:
LINKNAME:   STAVIPA1LNK
  ADDRESS:  10.30.1.241
    FLAGS:  PRIMARY
LINKNAME:   OSA2080LNK
  ADDRESS:  10.30.2.242
    FLAGS:
LINKNAME:   OSA20E0LNK
  ADDRESS:  10.30.3.245
    FLAGS:
LINKNAME:   IUTIQDF6LNK
```

```
  ADDRESS:  10.30.5.246
    FLAGS:
LINKNAME:   EZASAMEMVS
  ADDRESS:  10.30.20.101
    FLAGS:
LINKNAME:   IQDIOLNK0A1E1465
  ADDRESS:  10.30.20.101
    FLAGS:
LINKNAME:   VIPL0A1E1E02
  ADDRESS:  10.30.30.2
    FLAGS:
LINKNAME:   VIPL0A1E0A02
  ADDRESS:  10.30.10.2
    FLAGS:
LINKNAME:   LOOPBACK
  ADDRESS:  127.0.0.1
    FLAGS:
INTFNAME:   LOOPBACK6
  ADDRESS:  ::1
    TYPE:   LOOPBACK
    FLAGS:
10 OF 10 RECORDS DISPLAYED
```

Example 4-11 IP homelist in SC32

```
D TCPIP,TCPIPC,N,HOME
EZD0101I NETSTAT CS V1R7 TCPIPC 434
HOME ADDRESS LIST:
LINKNAME:   STAVIPA1LNK
  ADDRESS:  10.30.1.221
    FLAGS:  PRIMARY
LINKNAME:   OSA2080LNK
  ADDRESS:  10.30.2.222
    FLAGS:
LINKNAME:   OSA20E0LNK
  ADDRESS:  10.30.3.225
    FLAGS:
LINKNAME:   IUTIQDF6LNK
  ADDRESS:  10.30.5.226
    FLAGS:
LINKNAME:   EZASAMEMVS
  ADDRESS:  10.30.20.102
    FLAGS:
LINKNAME:   IQDIOLNK0A1E1466
  ADDRESS:  10.30.20.102
    FLAGS:
LINKNAME:   VIPL0A1E1E03
  ADDRESS:  10.30.30.3
    FLAGS:
LINKNAME:   VIPL0A1E0A03
  ADDRESS:  10.30.10.3
    FLAGS:
LINKNAME:   LOOPBACK
  ADDRESS:  127.0.0.1
    FLAGS:
INTFNAME:   LOOPBACK6
  ADDRESS:  ::1
    TYPE:   LOOPBACK
    FLAGS: 10 OF 10 RECORDS DISPLAYED
```

List OSPF configurations

Example 4-12, Example 4-13, and Example 4-14 on page 76 illustrate listing the OSPF configurations for the three systems.

Example 4-12 OSPF global configuration in SC30

```
D TCPIP,TCPIPC,OMPR,OSPF,LIST,ALL
EZZ7831I GLOBAL CONFIGURATION 333
    TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
    STACK AFFINITY:          TCPIPC
    OSPF PROTOCOL:           ENABLED
    EXTERNAL COMPARISON:     TYPE 2
    AS BOUNDARY CAPABILITY:  ENABLED
    IMPORT EXTERNAL ROUTES:  DIR SUB
    ORIG. DEFAULT ROUTE:     NO
    DEMAND CIRCUITS:         ENABLED

EZZ7832I AREA CONFIGURATION 1
AREA ID          AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
0.0.0.1          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*
10.30.10.1      0.0.0.1            10   N/A   N/A N/A   N/A   N/A   N/A
10.30.30.1      0.0.0.1            10   N/A   N/A N/A   N/A   N/A   N/A
10.30.20.100    0.0.0.1            15     5     1   1    10    40    40
10.30.20.100    0.0.0.1            15     5     1   1    10    40    40
10.30.5.236     0.0.0.1             5     5     1   1    10    40    40
10.30.3.235     0.0.0.1            10     5     1   0    10    40    40
10.30.2.232     0.0.0.1            10     5     1   0    10    40    40
10.30.1.230     0.0.0.1            10   N/A   N/A N/A   N/A   N/A   N/A

                 DEMAND CIRCUIT PARAMETERS
IP ADDRESS        DONOTAGE    HELLO SUPPRESSION   POLL INTERVAL
10.30.20.100      OFF         N/A                   N/A
10.30.20.100      OFF         ALLOW                  60
10.30.5.236       OFF         N/A                   N/A
10.30.3.235       OFF         N/A                   N/A
10.30.2.232       OFF         N/A                   N/A

             ADVERTISED VIPA ROUTES 3
10.30.10.1      /255.255.255.255   10.30.30.1     /255.255.255.255
10.30.1.230     /255.255.255.255
```

Example 4-13 OSPF global configuration in SC31

```
D TCPIP,TCPIPC,OMPR,OSPF,LIST,ALL
EZZ7831I GLOBAL CONFIGURATION 124
    TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
    STACK AFFINITY:          TCPIPC
    OSPF PROTOCOL:           ENABLED
    EXTERNAL COMPARISON:     TYPE 2
    AS BOUNDARY CAPABILITY:  ENABLED
    IMPORT EXTERNAL ROUTES:  DIR SUB
    ORIG. DEFAULT ROUTE:     NO
    DEMAND CIRCUITS:         ENABLED

EZZ7832I AREA CONFIGURATION 1
AREA ID          AUTYPE          STUB? DEFAULT-COST IMPORT-SUMMARIES?
```

```
0.0.0.1          O=NONE           YES         1             NO
0.0.0.0          O=NONE           NO          N/A           N/A

EZZ7833I INTERFACE CONFIGURATION 2
IP ADDRESS      AREA             COST RTRNS TRDLY PRI HELLO  DEAD   DB_E*
10.30.10.2      0.0.0.1            10   N/A   N/A N/A   N/A   N/A     N/A
10.30.20.101    0.0.0.1            15     5     1   1    10    40      40
10.30.20.101    0.0.0.1            15     5     1   1    10    40      40
10.30.5.246     0.0.0.1             5     5     1   1    10    40      40
10.30.3.245     0.0.0.1            10     5     1   0    10    40      40
10.30.2.242     0.0.0.1            10     5     1   0    10    40      40
10.30.1.241     0.0.0.1            10   N/A   N/A N/A   N/A   N/A     N/A

                 DEMAND CIRCUIT PARAMETERS
IP ADDRESS        DONOTAGE    HELLO SUPPRESSION   POLL INTERVAL
10.30.20.101      OFF         N/A                   N/A
10.30.20.101      OFF         ALLOW                  60
10.30.5.246       OFF         N/A                   N/A
10.30.3.245       OFF         N/A                   N/A
10.30.2.242       OFF         N/A                   N/A

             ADVERTISED VIPA ROUTES 3
10.30.10.2     /255.255.255.255   10.30.30.2     /255.255.255.255
10.30.1.241    /255.255.255.255
```

Example 4-14   OSPF global configuration in SC32

```
D TCPIP,TCPIPC,OMPR,OSPF,LIST,ALL
EZZ7831I GLOBAL CONFIGURATION 613
    TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
    STACK AFFINITY:           TCPIPC
    OSPF PROTOCOL:            ENABLED
    EXTERNAL COMPARISON:      TYPE 2
    AS BOUNDARY CAPABILITY:   ENABLED
    IMPORT EXTERNAL ROUTES:   DIR SUB
    ORIG. DEFAULT ROUTE:      NO
    DEMAND CIRCUITS:          ENABLED

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

EZZ7833I INTERFACE CONFIGURATION 2
IP ADDRESS      AREA             COST RTRNS TRDLY PRI HELLO  DEAD     DB_E*
10.30.10.3      0.0.0.1            10   N/A   N/A N/A   N/A   N/A     N/A
10.30.30.3      0.0.0.1            10   N/A   N/A N/A   N/A   N/A     N/A
10.30.20.102    0.0.0.1            15     5     1   1    10    40      40
10.30.20.102    0.0.0.1            15     5     1   1    10    40      40
10.30.5.226     0.0.0.1             5     5     1   1    10    40      40
10.30.3.225     0.0.0.1            10     5     1   0    10    40      40
10.30.2.222     0.0.0.1            10     5     1   0    10    40      40
10.30.1.221     0.0.0.1            10   N/A   N/A N/A   N/A   N/A     N/A

                 DEMAND CIRCUIT PARAMETERS
IP ADDRESS        DONOTAGE    HELLO SUPPRESSION   POLL INTERVAL
10.30.20.102      OFF         N/A                   N/A
10.30.20.102      OFF         ALLOW                  60
10.30.5.226       OFF         N/A                   N/A
10.30.3.225       OFF         N/A                   N/A
```

```
10.30.2.222       OFF         N/A                  N/A

             ADVERTISED VIPA ROUTES 3
10.30.10.3     /255.255.255.255   10.30.30.3     /255.255.255.255
10.30.1.221    /255.255.255.255
```

The following are key elements of the OSPF definitions:

- 1 OSPF stub area with no summaries (totally stubby area)
- 2 OSPF cost values reflects configuration (wildcards)
- 3 VIPA routes being advertised

List OSPF neighbors

Example 4-15, Example 4-16, and Example 4-17 illustrate listing the OSPF neighbors for each system.

Example 4-15 OSPF neighbors as seen from SC30

```
D TCPIP,TCPIPC,OMPR,OSPF,NBRS
EZZ7851I NEIGHBOR SUMMARY 209
NEIGHBOR ADDR   NEIGHBOR ID     STATE  LSRXL DBSUM LSREQ HSUP IFC
10.30.20.101    10.30.1.241       128      0     0     0  OFF IQDIOLNK*
10.30.20.102    10.30.1.221       128      0     0     0  OFF IQDIOLNK*
10.30.5.246     10.30.1.241       128      0     0     0  OFF IUTIQDF6*
10.30.5.226     10.30.1.221       128      0     0     0  OFF IUTIQDF6*
10.30.3.245     10.30.1.241         8      0     0     0  OFF OSA20E0L*
10.30.3.225     10.30.1.221         8      0     0     0  OFF OSA20E0L*
10.30.3.1       10.30.3.1         128      0     0     0  OFF OSA20E0L*
10.30.2.1       10.30.2.1         128      0     0     0  OFF OSA2080L*
10.30.2.242     10.30.1.241         8      0     0     0  OFF OSA2080L*
10.30.2.222     10.30.1.221         8      0     0     0  OFF OSA2080L*
* -- LINK NAME TRUNCATED
```

Example 4-16 OSPF neighbors as seen from SC31

```
D TCPIP,TCPIPC,OMPR,OSPF,NBRS
EZZ7851I NEIGHBOR SUMMARY 266
NEIGHBOR ADDR   NEIGHBOR ID     STATE  LSRXL DBSUM LSREQ HSUP IFC
10.30.20.100    10.30.1.230       128      0     0     0  OFF IQDIOLNK*
10.30.20.102    10.30.1.221       128      0     0     0  OFF IQDIOLNK*
10.30.5.226     10.30.1.221       128      0     0     0  OFF IUTIQDF6*
10.30.5.236     10.30.1.230       128      0     0     0  OFF IUTIQDF6*
10.30.3.1       10.30.3.1         128      0     0     0  OFF OSA20E0L*
10.30.3.225     10.30.1.221         8      0     0     0  OFF OSA20E0L*
10.30.3.235     10.30.1.230         8      0     0     0  OFF OSA20E0L*
10.30.2.222     10.30.1.221         8      0     0     0  OFF OSA2080L*
10.30.2.232     10.30.1.230         8      0     0     0  OFF OSA2080L*
10.30.2.1       10.30.2.1         128      0     0     0  OFF OSA2080L*
* -- LINK NAME TRUNCATED
```

Example 4-17 OSPF neighbors as seen from SC32

```
D TCPIP,TCPIPC,OMPR,OSPF,NBRS
EZZ7851I NEIGHBOR SUMMARY 621
NEIGHBOR ADDR   NEIGHBOR ID     STATE  LSRXL DBSUM LSREQ HSUP IFC
10.30.20.101    10.30.1.241       128      0     0     0  OFF IQDIOLNK*
10.30.20.100    10.30.1.230       128      0     0     0  OFF IQDIOLNK*
10.30.5.246     10.30.1.241       128      0     0     0  OFF IUTIQDF6*
10.30.5.236     10.30.1.230       128      0     0     0  OFF IUTIQDF6*
```

```
10.30.3.245     10.30.1.241        8      0     0     0  OFF OSA20E0L*
10.30.3.235     10.30.1.230        8      0     0     0  OFF OSA20E0L*
10.30.3.1       10.30.3.1        128      0     0     0  OFF OSA20E0L*
10.30.2.242     10.30.1.241        8      0     0     0  OFF OSA2080L*
10.30.2.232     10.30.1.230        8      0     0     0  OFF OSA2080L*
10.30.2.1       10.30.2.1        128      0     0     0  OFF OSA2080L*
* -- LINK NAME TRUNCATED
```

List routing tables

Example 4-18, Example 4-19, and Example 4-20 on page 79 illustrate listing the routing tables for the three systems.

Example 4-18 IP route table in SC30

```
D TCPIP,TCPIPC,N,ROUTE
EZD0101I NETSTAT CS V1R7 TCPIPC 423
IPV4 DESTINATIONS
DESTINATION        GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT            10.30.2.1        UGO       000000  OSA2080LNK
DEFAULT            10.30.3.1        UGO       000001  OSA20E0LNK
10.30.1.221/32     10.30.5.226      UGHO      000000  IUTIQDF6LNK 1
10.30.1.230/32     0.0.0.0          UH        000000  STAVIPA1LNK 1
10.30.1.241/32     10.30.5.246      UGHO      000000  IUTIQDF6LNK 1
10.30.2.0/24       0.0.0.0          UO        000000  OSA2080LNK
10.30.2.232/32     0.0.0.0          UH        000001  OSA2080LNK
10.30.3.0/24       0.0.0.0          UO        000000  OSA20E0LNK
10.30.3.235/32     0.0.0.0          UH        000000  OSA20E0LNK
10.30.5.0/24       0.0.0.0          UO        000000  IUTIQDF6LNK
10.30.5.236/32     0.0.0.0          UH        000000  IUTIQDF6LNK
10.30.10.1/32      0.0.0.0          UH        000000  VIPL0A1E0A01
10.30.10.2/32      10.30.5.246      UGHO      000000  IUTIQDF6LNK
10.30.10.3/32      10.30.5.226      UGHO      000000  IUTIQDF6LNK
10.30.20.0/24      0.0.0.0          US        000000  IQDIOLNK0A1E1464
10.30.20.100/32    0.0.0.0          UH        000000  EZASAMEMVS
10.30.20.100/32    0.0.0.0          UH        000000  IQDIOLNK0A1E1464
10.30.20.101/32    0.0.0.0          UHS       000000  IQDIOLNK0A1E1464
10.30.20.102/32    0.0.0.0          UHS       000000  IQDIOLNK0A1E1464
10.30.30.1/32      0.0.0.0          UH        000000  VIPL0A1E1E01
10.30.30.2/32      10.30.5.246      UGHO      000000  IUTIQDF6LNK
10.30.30.3/32      10.30.5.226      UGHO      000000  IUTIQDF6LNK
10.30.32.0/24      10.30.2.1        UGO       000000  OSA2080LNK
10.30.32.0/24      10.30.3.1        UGO       000000  OSA20E0LNK
127.0.0.1/32       0.0.0.0          UH        000004  LOOPBACK
IPV6 DESTINATIONS
DESTIP:   ::1/128
  GW:     ::
  INTF:   LOOPBACK6         REFCNT:  000000
  FLGS:   UH                MTU: 65535
26 OF 26 RECORDS DISPLAYED
```

Example 4-19 IP route table in SC31

```
D TCPIP,TCPIPC,N,ROUTE
EZD0101I NETSTAT CS V1R7 TCPIPC 305
IPV4 DESTINATIONS
DESTINATION        GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT            10.30.2.1        UGO       000000  OSA2080LNK
DEFAULT            10.30.3.1        UGO       000001  OSA20E0LNK
10.30.1.221/32     10.30.5.226      UGHO      000000  IUTIQDF6LNK 1
```

```
10.30.1.230/32      10.30.5.236      UGHO      000000  IUTIQDF6LNK 1
10.30.1.241/32      0.0.0.0          UH        000000  STAVIPA1LNK 1
10.30.2.0/24        0.0.0.0          UO        000000  OSA2080LNK
10.30.2.242/32      0.0.0.0          UH        000000  OSA2080LNK
10.30.3.0/24        0.0.0.0          UO        000000  OSA20E0LNK
10.30.3.245/32      0.0.0.0          UH        000000  OSA20E0LNK
10.30.5.0/24        0.0.0.0          UO        000000  IUTIQDF6LNK
10.30.5.246/32      0.0.0.0          UH        000001  IUTIQDF6LNK
10.30.10.1/32       10.30.5.236      UGHO      000000  IUTIQDF6LNK
10.30.10.2/32       0.0.0.0          UH        000000  VIPL0A1E0A02
10.30.10.3/32       10.30.5.226      UGHO      000000  IUTIQDF6LNK
10.30.20.0/24       0.0.0.0          US        000000  IQDIOLNK0A1E1465
10.30.20.100/32     0.0.0.0          UHS       000000  IQDIOLNK0A1E1465
10.30.20.101/32     0.0.0.0          UH        000000  EZASAMEMVS
10.30.20.101/32     0.0.0.0          UH        000000  IQDIOLNK0A1E1465
10.30.20.102/32     0.0.0.0          UHS       000000  IQDIOLNK0A1E1465
10.30.30.1/32       10.30.5.236      UGHO      000000  IUTIQDF6LNK
10.30.30.2/32       0.0.0.0          UH        000000  VIPL0A1E1E02
10.30.30.3/32       10.30.5.226      UGHO      000000  IUTIQDF6LNK
10.30.32.0/24       10.30.2.1        UGO       000000  OSA2080LNK
10.30.32.0/24       10.30.3.1        UGO       000000  OSA20E0LNK
127.0.0.1/32        0.0.0.0          UH        000004  LOOPBACK
IPV6 DESTINATIONS
DESTIP:   ::1/128
  GW:     ::
  INTF:   LOOPBACK6         REFCNT:  000000
  FLGS:   UH                MTU: 65535
26 OF 26 RECORDS DISPLAYED
```

Example 4-20 IP route table in SC32

```
D TCPIP,TCPIPC,N,ROUTE
EZD0101I NETSTAT CS V1R7 TCPIPC 628
IPV4 DESTINATIONS
DESTINATION         GATEWAY          FLAGS     REFCNT  INTERFACE
DEFAULT             10.30.2.1        UGO       000001  OSA2080LNK
DEFAULT             10.30.3.1        UGO       000000  OSA20E0LNK
10.30.1.221/32      0.0.0.0          UH        000000  STAVIPA1LNK 1
10.30.1.230/32      10.30.5.236      UGHO      000000  IUTIQDF6LNK 1
10.30.1.241/32      10.30.5.246      UGHO      000000  IUTIQDF6LNK 1
10.30.2.0/24        0.0.0.0          UO        000000  OSA2080LNK
10.30.2.222/32      0.0.0.0          UH        000000  OSA2080LNK
10.30.3.0/24        0.0.0.0          UO        000000  OSA20E0LNK
10.30.3.225/32      0.0.0.0          UH        000000  OSA20E0LNK
10.30.5.0/24        0.0.0.0          UO        000000  IUTIQDF6LNK
10.30.5.226/32      0.0.0.0          UH        000001  IUTIQDF6LNK
10.30.10.1/32       10.30.5.236      UGHO      000000  IUTIQDF6LNK
10.30.10.2/32       10.30.5.246      UGHO      000000  IUTIQDF6LNK
10.30.10.3/32       0.0.0.0          UH        000000  VIPL0A1E0A03
10.30.20.0/24       0.0.0.0          US        000000  IQDIOLNK0A1E1466
10.30.20.100/32     0.0.0.0          UHS       000000  IQDIOLNK0A1E1466
10.30.20.101/32     0.0.0.0          UHS       000000  IQDIOLNK0A1E1466
10.30.20.102/32     0.0.0.0          UH        000000  EZASAMEMVS
10.30.20.102/32     0.0.0.0          UH        000000  IQDIOLNK0A1E1466
10.30.30.1/32       10.30.5.236      UGHO      000000  IUTIQDF6LNK
10.30.30.2/32       10.30.5.246      UGHO      000000  IUTIQDF6LNK
10.30.30.3/32       0.0.0.0          UH        000000  VIPL0A1E1E03
10.30.32.0/24       10.30.2.1        UGO       000000  OSA2080LNK
10.30.32.0/24       10.30.3.1        UGO       000000  OSA20E0LNK
127.0.0.1/32        0.0.0.0          UH        000005  LOOPBACK
```

```
IPV6 DESTINATIONS
DESTIP:   ::1/128
  GW:     ::
  INTF:   LOOPBACK6         REFCNT:  000000
  FLGS:   UH                MTU: 65535
26 OF 26 RECORDS DISPLAYED
```

A key element is that 1 VIPA addresses are advertised only as host address (/32) due to the Advertise_VIPA_Routes=HOST_ONLY parameter.

List neighbors for routers

Example 4-21 and Example 4-22 illustrate listings of the neighbors for the two routers.

Example 4-21 Display of OSPF neighbors in router 1

```
Router# sh ip ospf 300 neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.30.3.1         1   FULL/BDR        00:00:33    10.30.32.2      Vlan32 1
10.30.1.221       0   FULL/DROTHER    00:00:32    10.30.2.222     Vlan30 3
10.30.1.230       0   FULL/DROTHER    00:00:32    10.30.2.232     Vlan30 4
10.30.1.241       0   FULL/DROTHER    00:00:33    10.30.2.242     Vlan30 5
```

Example 4-22 Display of OSPF neighbors in router 2

```
Router# sh ip ospf 300 neighbor

Neighbor ID     Pri   State           Dead Time   Address         Interface
10.30.2.1         1   FULL/DR         00:00:31    10.30.32.1      Vlan32 2
10.30.1.221       0   FULL/DROTHER    00:00:30    10.30.3.225     Vlan31 3
10.30.1.230       0   FULL/DROTHER    00:00:39    10.30.3.235     Vlan31 4
10.30.1.241       0   FULL/DROTHER    00:00:38    10.30.3.245     Vlan31 5
```

Key points here include the following:

- 1 Full neighbor ship with router 2
- 2 Full neighbor ship with router 1
- 3 Full neighbor ship with SC32
- 4 Full neighbor ship with SC30
- 5 Full neighbor ship with SC31

Routing tables in routers

Example 4-23 and Example 4-24 on page 81 illustrate listing the routing tables in the two routers.

Example 4-23 Display of IP route table in router 1

```
Router# sh ip route ospf 300
O       10.30.20.0/24 [110/16] via 10.30.2.242, 05:08:52, Vlan30
                      [110/16] via 10.30.2.232, 05:08:52, Vlan30
                      [110/16] via 10.30.2.222, 05:08:52, Vlan30
O       10.30.30.2/32 [110/11] via 10.30.2.242, 05:08:52, Vlan30
O       10.30.30.3/32 [110/11] via 10.30.2.222, 05:08:52, Vlan30
O       10.30.30.1/32 [110/11] via 10.30.2.232, 05:08:52, Vlan30
O       10.30.5.0/24 [110/6] via 10.30.2.242, 05:08:52, Vlan30
                     [110/6] via 10.30.2.232, 05:08:52, Vlan30
                     [110/6] via 10.30.2.222, 05:08:52, Vlan30
```

```
O       10.30.3.0/24 [110/2] via 10.30.32.2, 00:25:08, Vlan32
O       10.30.10.2/32 [110/11] via 10.30.2.242, 05:08:52, Vlan30
O       10.30.10.3/32 [110/11] via 10.30.2.222, 05:08:52, Vlan30
O       10.30.10.1/32 [110/11] via 10.30.2.232, 05:08:52, Vlan30
O       10.30.20.100/32 [110/1] via 10.30.2.232, 05:08:52, Vlan30
O       10.30.20.101/32 [110/1] via 10.30.2.242, 05:08:52, Vlan30
O       10.30.20.102/32 [110/1] via 10.30.2.222, 05:08:52, Vlan30
O       10.30.1.221/32 [110/11] via 10.30.2.222, 05:08:52, Vlan30 1
O       10.30.1.241/32 [110/11] via 10.30.2.242, 05:08:52, Vlan30 1
O       10.30.1.230/32 [110/11] via 10.30.2.232, 05:08:52, Vlan30 1
```

Example 4-24 Display of IP route table in router 2

```
Router# sh ip route ospf 300
O       10.30.20.0/24 [110/16] via 10.30.3.245, 05:10:15, Vlan31
                      [110/16] via 10.30.3.235, 05:10:15, Vlan31
                      [110/16] via 10.30.3.225, 05:10:15, Vlan31
O       10.30.30.2/32 [110/11] via 10.30.3.245, 05:10:15, Vlan31
O       10.30.30.3/32 [110/11] via 10.30.3.225, 05:10:15, Vlan31
O       10.30.30.1/32 [110/11] via 10.30.3.235, 05:10:15, Vlan31
O       10.30.5.0/24 [110/6] via 10.30.3.245, 05:10:15, Vlan31
                     [110/6] via 10.30.3.235, 05:10:15, Vlan31
                     [110/6] via 10.30.3.225, 05:10:15, Vlan31
O       10.30.2.0/24 [110/2] via 10.30.32.1, 00:25:49, Vlan32
O       10.30.10.2/32 [110/11] via 10.30.3.245, 05:10:15, Vlan31
O       10.30.10.3/32 [110/11] via 10.30.3.225, 05:10:15, Vlan31
O       10.30.10.1/32 [110/11] via 10.30.3.235, 05:10:15, Vlan31
O       10.30.20.100/32 [110/1] via 10.30.3.235, 05:10:15, Vlan31
O       10.30.20.101/32 [110/1] via 10.30.3.245, 05:10:16, Vlan31
O       10.30.20.102/32 [110/1] via 10.30.3.225, 05:10:16, Vlan31
O       10.30.1.221/32 [110/11] via 10.30.3.225, 05:10:16, Vlan31 1
O       10.30.1.241/32 [110/11] via 10.30.3.245, 05:10:16, Vlan31 1
O       10.30.1.230/32 [110/11] via 10.30.3.235, 05:10:16, Vlan31 1
```

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

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

The application workload balancing decision maker provided with the Communications Server is the Sysplex Distributor (SD). Another option, the use of Shareport, is a method workload distribution done by the stack itself within a single z/OS system. This chapter describes internal application workload balancing, primarily focusing on the Sysplex Distributor function and the benefits it provides. However, we also discuss and explain the usage of Shareports.

This chapter discusses the following.

Section	Topic
5.1, "Basic concepts" on page 84	Discusses the basic concepts of Internal application workload balancing
5.2, "Importance of internal application workload balancing" on page 106	Discusses key characteristics of Internal application workload balancing and why it may be important in your environment
5.3, "Design examples" on page 106	Presents commonly implemented internal application workload balancing design scenarios, their dependencies, advantages, considerations, and our recommendations
5.4, "Implementations" on page 114	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

5.1 Basic concepts

The basic 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 WLM and QoS data from the Service Policy Agent. Sysplex Distributor also measures the responsiveness of target servers in accepting new TCP connection setup requests, favoring those servers that are more successfully accepting new requests.

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, on a very high level, where the decision-maker for internal application workload balancing resides.

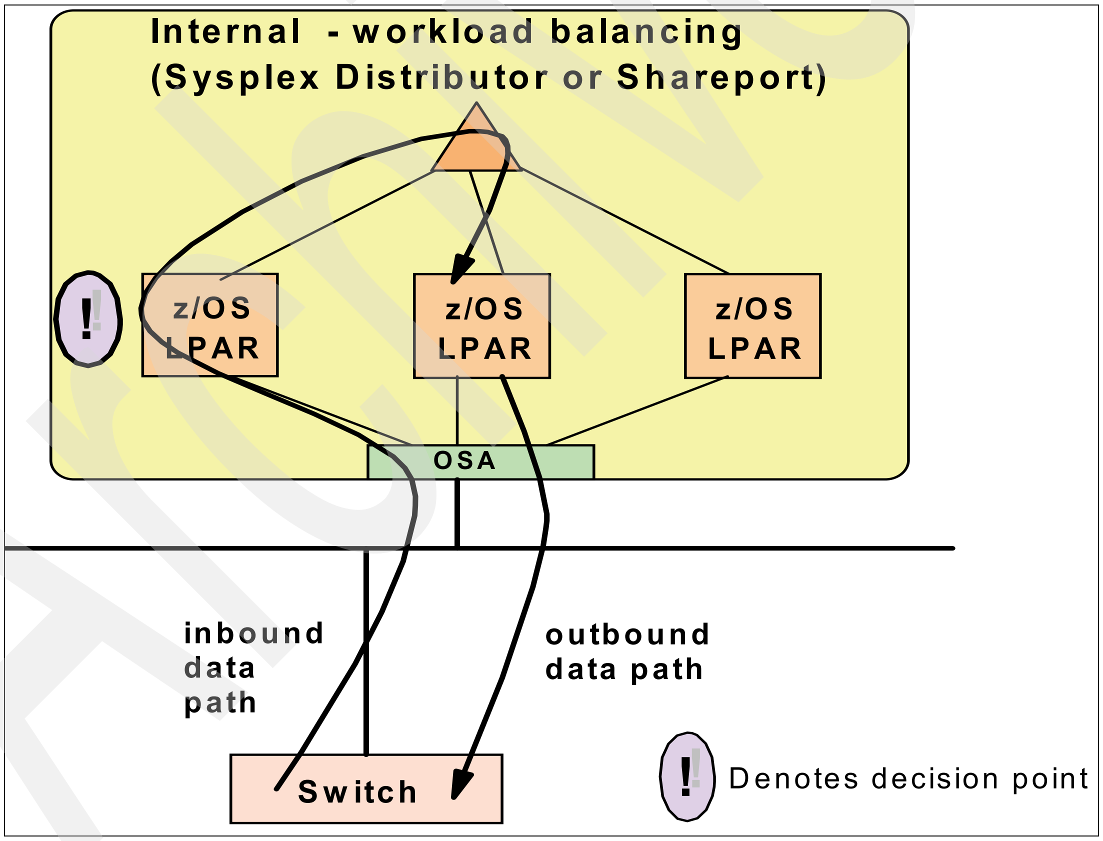


Figure 5-1 Decision point within the sysplex: internal workload balancing

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. CS for z/OS allows us 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 more background information describing the usage of Distributed DVIPA in a Sysplex Distributor environment including failover and recovery scenarios, please refer to “Distributed DVIPA (Sysplex Distributor)” on page 20.

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 involved communicate via XCF; this 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 can be based on real-time consultation with WLM (or round-robin), z/OS QoS Policy Agent, and the target stack for application availability. When the selection of the target stack is done, the connection information is stored in the Sysplex Distributor stack to route future IP packets belonging to the same connection to the selected target stack. Routing is based on the connection (or“session) to which IP packets belong, and this is known as *connection-based routing*. The hot standby 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 (such as resulting from some sort of failure). This takeover is nondisruptive to all existing connections.

Figure 5-2 on page 86 provides a conceptual overview of Sysplex Distributor operation.

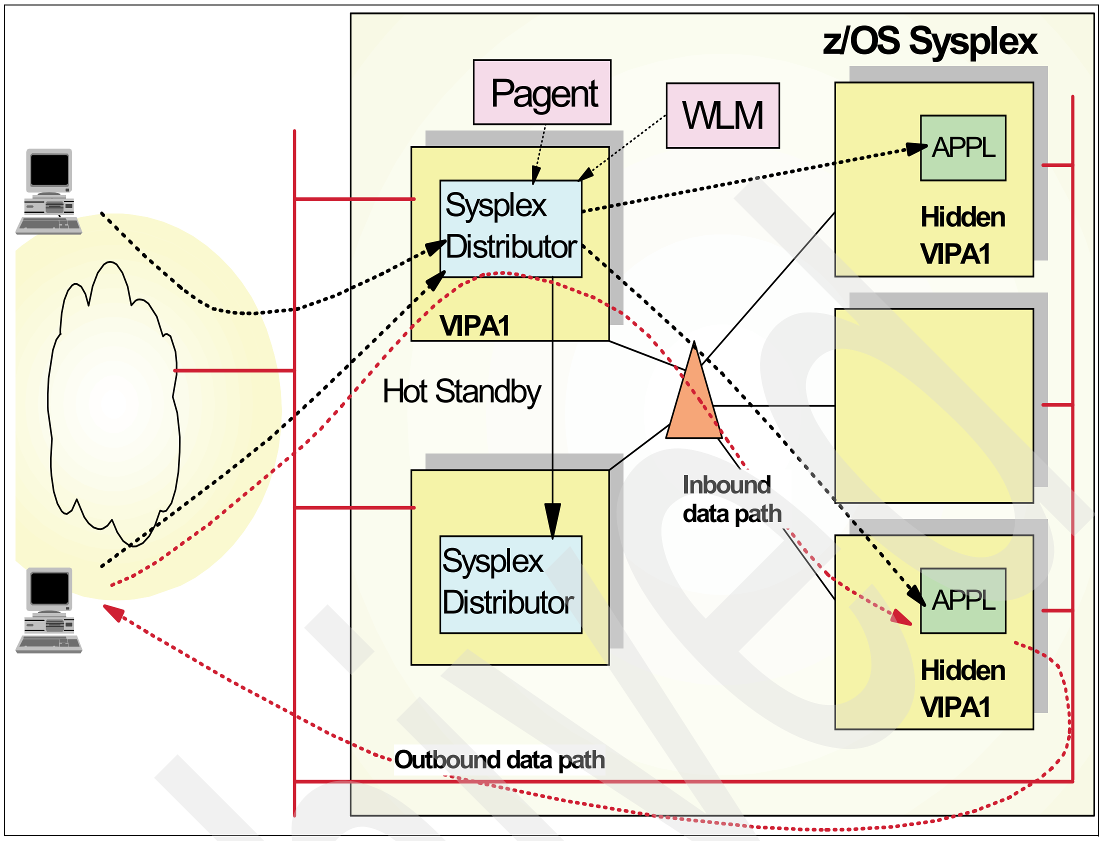


Figure 5-2 Sysplex Distributor for z/OS-integrated intra-sysplex workload balancing

Tip: Prior to z/OS V1R6, the Sysplex Distributor function required the stack to have IP forwarding enabled. z/OS V1R6 and later releases do not require DATAGRAMFWD to be enabled. Sysplex Distributor can now be deployed without any risk of using a z/OS stack as a general intermediate routing node.

5.1.2 Sysplex Distributor configured as a Cisco Service Manager

Cisco's MultiNode Load Balancing (MNLB) is designed to provide functionality similar to that of the Sysplex Distributor. In essence, MNLB distributes connection requests across a set of destination target servers. With MNLB, traffic is forwarded directly to the appropriate application server from the network switches rather than through a z/OS routing stack (such as the Sysplex Distributor). The disadvantage, however, is that (without Sysplex Distributor's understanding real-time application status and performance) MNLB must make its workload balancing decisions based upon limited information.

Beginning with the Communications Server for z/OS V1R2, the MNLB architecture can cooperate with the Sysplex Distributor function. In this paradigm, the connection dispatching function once performed by Sysplex Distributor is divided into two parts: the distribution of connections and the forwarding of data. What results is a *best of both worlds* situation. The Sysplex Distributor leverages its strength in using up-to-date information to select the most appropriate target stack to receive a connection. The MNLB router is informed of the selection and directly forwards data to the target server. This removes the overhead of forwarding all packets from the Sysplex Distributor stack.

In the MNLB architecture, the entity that decides which target server is to receive a given connections is called the *Service Manager* and 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 Cisco switches perform the Forwarding Agent role. Figure 5-3 illustrates the general concepts of a *hybrid SD/MNLB* configuration.

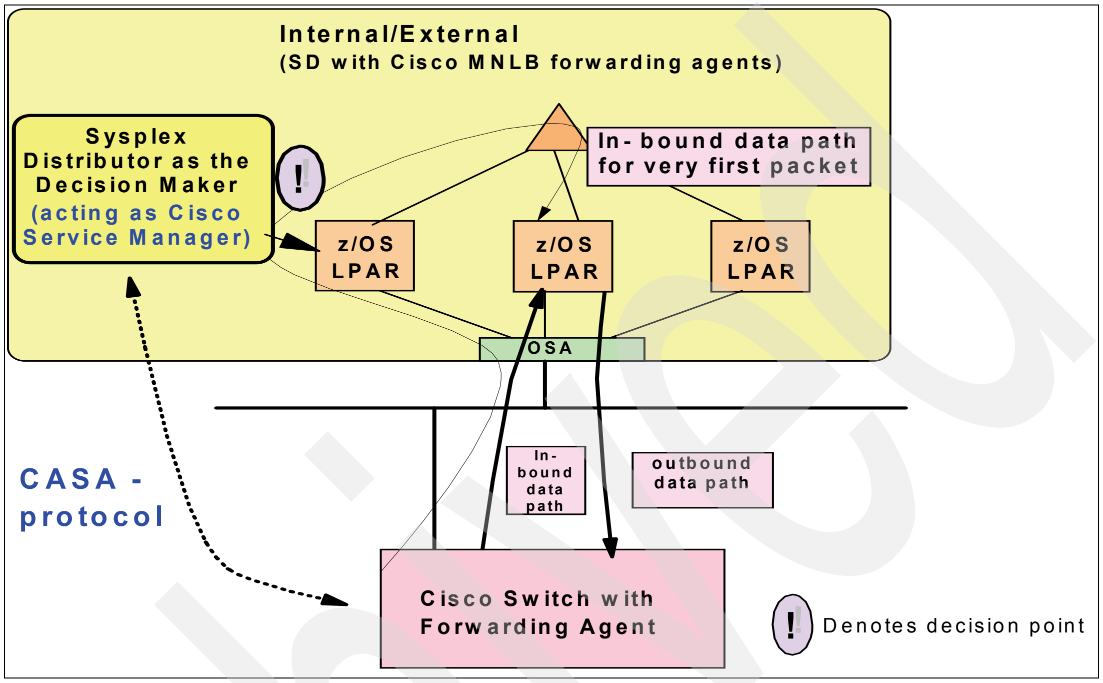

Internal/External
(SD with Cisco MNLB forwarding agents)
Sysplex Distributor as the Decision Maker
(acting as Cisco Service Manager)
!
In-bound data path for very first packet
z/OS LPAR
z/OS LPAR
z/OS LPAR
OSA
CASA - protocol
In-bound data path
outbound data path
Cisco Switch with Forwarding Agent
! Denotes decision point

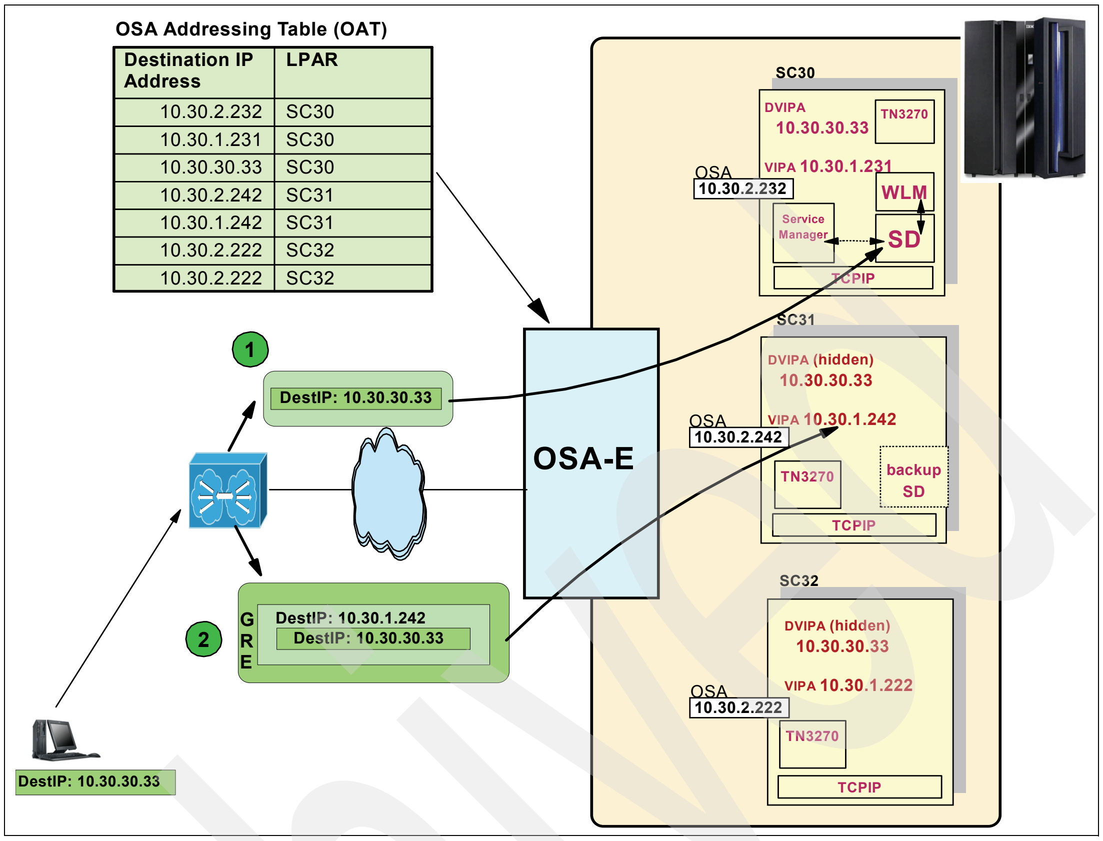
Figure 5-3 Decision point still within the sysplex: SD acting as Cisco Service Manager

In the hybrid implementation, the Sysplex Distributor makes the actual workload distribution decision and, acting as a Cisco Service Manager, relays that decision to the Cisco Forwarding Agent within the switch using the Cisco Appliance Services Architecture (CASA) protocol. As a result, the Forwarding Agents are able to directly send future packets for that connection directly to the target server, bypassing the distribution stack.

This hybrid solution combines the advantages of internal decision-making with routing efficiency of an external control point. It is especially useful for applications with large amounts of inbound data.

Considerations

Remember that in a SD/MNLB environment only the distribution stack advertises the DVIPA, using a dynamic routing protocol. All connection requests from clients to that DVIPA will go to the LPAR that is the current owner of the distributed dynamic VIPA. In Figure 5-4 on page 88 this is LPAR SC30. Since only one LPAR can own a DVIPA at any one time, we must use other means (such as Generic Routing Encapsulation, GRE, tunneling) to pass work to throughout the LPAR servers in an SD/MNLB environment.


OSA Addressing Table (OAT)

Destination IP Address	LPAR
10.30.2.232	SC30
10.30.1.231	SC30
10.30.30.33	SC30
10.30.2.242	SC31
10.30.1.242	SC31
10.30.2.222	SC32
10.30.2.222	SC32

Figure 5-4 SD/MNLB - Dispatch mode use of GRE tunneling with a shared OSA

In the illustration in Figure 5-4, the routing works as follows:

1. When an initial connection request arrives at the router/switch, it is forwarded to the distribution stack in LPAR SC30. When a server decision is made the Sysplex Distributor informs the router/switch of the selected target stack; this is SC31 in the illustration.
2. When the next IP packet for this connection arrives at the router/switch, it is wrapped, using GRE encapsulation, and sent directly to the target stack.

Note: Generic Routing Encapsulation (GRE) is a standard protocol described by RFC1701. GRE allows a wrapper to be placed around a packet during transmission of the data. A receiving stack that supporting GRE removes the GRE wrapper, allowing the original packet to be processed by the receiving stack. This is often used to deliver a packet to a stack using an alternate destination IP address. For more information regarding GRE, please refer to RFC1701.

5.1.3 Sysplex Distributor and Quality of Service (QoS) 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 dynamically monitor server performance and affect sysplex workload distribution is an important part of the overall QoS mechanism.

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

- Policies can be defined to control which stacks the Sysplex Distributor may 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 time-outs 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.

In earlier releases, the QoS performance data was collected by the Policy Agent on the target for each DVIPA and port or application. After collecting the QoS information, the Policy Agent on the target stack pushed this information down to the stack sysplex function, which then forwarded it to the stack sysplex function on the distributing stack. The z/OS V1R7.0 Communications Server has two significant additions to Policy Agent and sysplex interaction:

- The Policy Agent at each target now collects information with an additional level of granularity; 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.

For our examples, we decided to configure our Sysplex Distributor so that the Policy Agent extracts the needed policy information from a statically configured flat file (PAGENT file) rather than using an LDAP server. This had the following advantages:

- Policies are easy to define and change.
- Definitions are local. No connection is needed.
- Requires very little overhead when policies do not change.

If we wanted to create a more complex or dynamic Sysplex Distributor policy, we would use the zQoS Manager rather than a flat file. An example of using the zQoS Manager to define Sysplex Distributor policies is provided in Chapter 6, "Quality of Service," in *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 4 - Policy-Based Network Security*, SG24-7172.

Our sysplex consists of three z/OS V1R7 LPARs: SC30, SC31, and SC32. The TCP/IP stack on SC30 has the distributing stack, and the stacks on SC31 and SC32 act as the primary and secondary backup, respectively, for the distributing function. A TN3270 server has been configured and runs on each LPAR. We have defined a Sysplex Distributor policy for TN3270 connections have defined an Sysplex Distributor performance monitoring policy for our target stacks on SC31 and SC32.

Figure 5-5 on page 90 illustrates our QoS system environment.

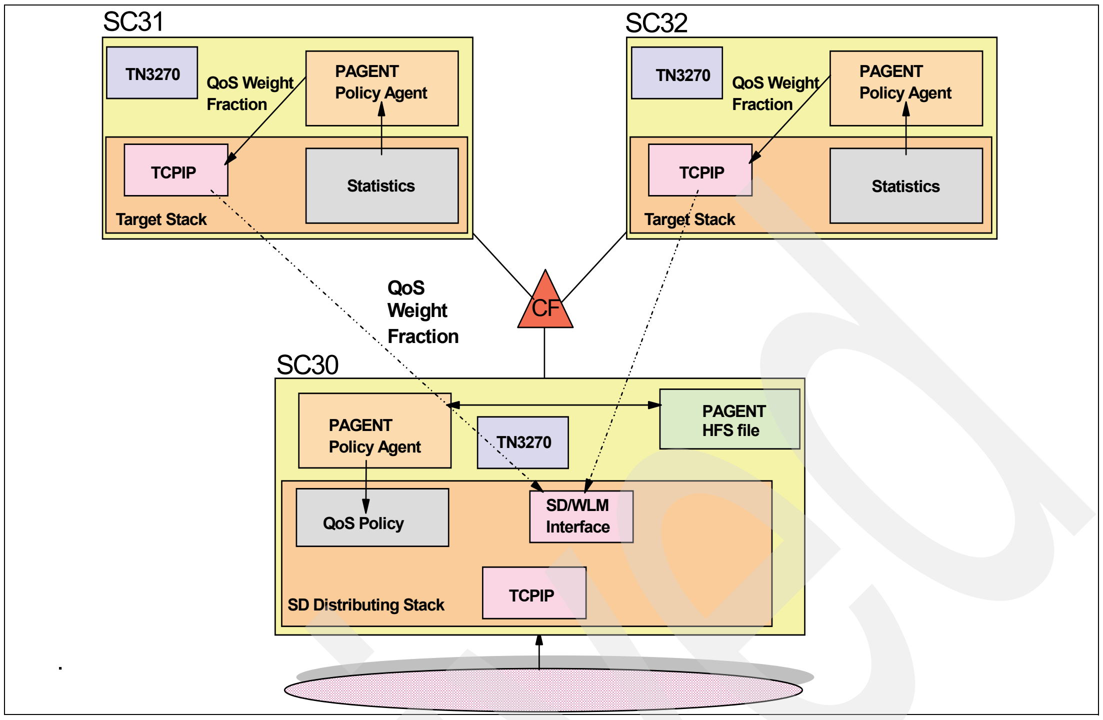


Figure 5-5 Our Sysplex Distributor policy implementation

We defined a Sysplex Distributor policy that limits the number of Sysplex Distributor target stacks for inbound traffic, and a SD performance monitoring policy on all the participating stacks. For SD performance monitoring, the traffic to be monitored must be represented by at least one Differentiated Services policy defined for the target application.

All policies were configured in the image configuration file, which is the second-level PAGENT configuration file.

Attention: Be very careful when copying policy samples from other sources. We started with a copied policy and accidentally included some *invalid* “curly braces” into our policy definition. Consequently, we received some very misleading error messages when we started PAGENT. While these braces looked fine when browsing or editing our policy definition file, when we finally suspected them and turned ’HEX ON’, they proved wrong. The EBCDIC codes for the correct curly braces are: x’C0’ corresponding to ’{’ and x’D0’ corresponding to ’}’.

Example 5-1 lists the SD policy we configured on SC30.

Example 5-1 Our SD policy for the distribution stack on SC30

```
# LogLevel Statement
# Loglevel 511
# PolicyPerfMonitorForSDR Statement
PolicyPerfMonitorForSDR enable
{
 samplinginterval 60
 LossRatioAndWeightFr 20 25
 TimeoutRatioAndWeightFr 50 50
 LossMaxWeightFr 95
 TimeoutMaxWeightFr 100
 MaxConnWeightFr 50 65 80
}
# Policy Action statement
policyAction tnaction
{
 policyScope DataTraffic
 MinRate 500
 OutgoingTOS 10100000
 outboundinterface 10.30.20.100   1
 outboundinterface 10.30.20.101   1
}
# Policy Rule statement for distributing stack
policyRule sdtnrule
{
 ProtocolNumberRange 6
 DestinationPortRange 23
 SourceAddressRange 9.12.4.0 9.12.4.255   2
 policyactionreference tnaction    3
}
```

These are identified as SD policies by the presence of the Outboundinterface 1 attribute in the PolicyAction statement. The target stacks are identified by the IP address of the dynamic XCF link.

We had TN3270 servers running on all three systems (SC30, SC31, and SC32). The distribution stack on SC30 was an eligible target server for workload distribution of TN3270 connections. We defined no outboundinterface pointing to system SC32, so therefore all the incoming TN3270 connection requests will be forwarded only to either SC30 or SC31 for inbound traffic from a given subnet 2, even though there is a TN3270 server running on SC32. Without this SD policy activated, an incoming TN3270 connection would be forwarded to either of our three systems.

A policyRule must be linked to the policyAction by a policyActionReference 3 statement.

An additional possibility would be to activate this policy only at certain times, for example, during normal working hours from Monday to Friday. Two additional statements in the policyRule definition would do this:

```
DayOfWeekMask      0111110
TimeOfDayRange     08:00-17:00
```

Example 5-2 contains the SD policy for our target stacks, SC31 and SC32.

Example 5-2 SD policy for the target stacks on SC31 and SC32

```
# LogLevel Statement
Loglevel 511
# PolicyPerfMonitorForSDR Statement
PolicyPerfMonitorForSDR enable    1
{
 samplinginterval 60              2
 LossRatioAndWeightFr 20 25       3
 TimeoutRatioAndWeightFr 50 50    4
 LossMaxWeightFr 95               5
 TimeoutMaxWeightFr 100           6
}
# Policy Action statement
policyAction tnaction            7
{
  policyScope DataTraffic       8
  Maxconnections 10
  MinRate 400
  OutgoingTOS 01000000
}
# Policy Rule statement for target stack
policyRule tgtnrule              9
{
 ProtocolNumberRange 6
 DestinationPortRange 23
 SourceAddressRange 9.12.4.0 9.12.4.255
 policyactionreference tnaction
}
```

Note the following important elements in this example:

- 1 Enables the policy performance monitor function. This assigns a weight fraction to the monitored policy performance data and sends it to the SD distributing stack when the monitored data crosses defined thresholds. The SD distributing stack uses this weight fraction for its routing decisions for incoming connection requests.
- 2 With the samplingInterval we specify how often we want to sample the policy information for changes.
- 3 The LossRatioAndWeightFr has two values: the ratio of retransmitted bytes to transmitted bytes in tenths of a percent, and the weight fraction to be assigned as a percentage. In our implementation, if a loss rate of 2% occurs, the loss fraction will be 25%. If a loss ratio rate above 4% occurs, the loss weight fraction will be 50%. A a loss rate of 3% means a loss weight fraction of 25%.
- 4 The TimeoutRatioAndWeightFr has two values: the timeout ratio in tenths of a percent, and the weight fraction to be assigned in percentage. In our implementation, if a timeout rate of 5% occurs, the timeout fraction weight would be 50%. If the timeout ratio is above 10%, the timeout weight fraction would be 100%.
- The two weight fractions LossRatioAndWeightFr and TimeoutRatioWeightFr are added. As an example, we have a QoS weight fraction value of 75%, which this target stack sends to the Sysplex Distributor. This value is used by the Sysplex Distributor stack to reduce the WLM weight given to this stack. If the WLM weight assigned to this target stack is 50, the QoS weight fraction of 75% will give an effective WLM weight fraction of 37.5.
- 5 LossMaxWeightFr defines the maximum loss weight fraction.
- 6 TimeoutMaxWeightFr defines the maximum timeout weight fraction.
- The Policy Agent monitors the traffic for which one or more service policy statements have been defined 7, 9. The policyScope attribute for the monitored policy must be either DataTraffic 8 or both. We monitored the TN3270 traffic originated from the 9.12.4.0 IP subnetwork.

Starting and stopping PAGENT

The Policy Agent can be started from the UNIX System Services shell or as a started task. We used a started task procedure to start the Policy Agent.

Although the /etc/pagent.conf file is the default configuration file, a specific search order is used when starting the Policy Agent. The following order is used:

1. File or data set specified with the -c startup option
2. File or data set specified with the PAGENT_CONF_FILE environment variable
3. /etc/pagent.conf
4. hlq.PAGENT.CONF

A security product authority is required to start the Policy Agent. When using RACF, the following commands can be used to create the necessary profile and permit users to use it:

```
RDEFINE OPERCMDS (MVS.SERVMGR.PAGENT) UACC(NONE)
PERMIT MVS.SERVMGR.PAGENT ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)SETROPTS
RACLIST(OPERCMDS) REFRESH i
```

Example 5-3 is the PAGENT started task procedure that we used in our system.

Example 5-3 Procedure for the started task PAGENT

```
//PAGENT   PROC SYS=&SYSCLONE
//PAGENT   EXEC PGM=PAGENT,REGION=OK,TIME=NOLIMIT,
//     PARM='POSIX(ON) ALL31(ON) ENVAR("_CEE_ENVFILE=DD:STDENV")/-d 1'
//STDENV 1 DD PATH='/etc/pagent.sc&SYS..env',PATHOPTS=(ORDONLY)  2
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

We can use environment variables, from an MVS data set or an HFS file, specified by the STDENV DD 1, to run with the desired configuration. We have configured environment variables in HFS file 2 /etc/pagent.sc&SYS..env, as shown in Example 5-4.

Example 5-4 Environment variables for our policy agent

```
PAGENT_CONFIG_FILE=/SC30/etc/pagent.sc30.conf        1
PAGENT_LOG_FILE=/SC30/tmp/pagent.sc30.log            2
PAGENT_LOG_FILE_CONTROL=999,3                        3
_BPXK_SETIBMOPT_TRANSPORT=TCPIPC
TZ=EST5EDT                                           4
```

The PAGENT_CONFIG_FILE specifies the PAGENT configuration file to use 1. The PAGENT_LOG_FILE specifies the log file name used by PAGENT 2, and the PAGENT_LOG_FILE_CONTROL 3 defines how many PAGENT log files are used (round-robin) and the size of the file; we used the default values.

If we define PAGENT to use syslogd to log messages (done by defining PAGENT_LOG_FILE=SYSLOGD), then the PAGENT_LOG_FILE_CONTROL has no meaning.

For the Policy Agent to run in our local time zone, we specified our time zone using the TZ environment variable 4.

Note: Be aware that most z/OS UNIX applications that start as MVS started tasks cannot use environment variables that have been configured in /etc/profile.

Note that while we do not have the RESOLVER_CONFIG variable configured, PAGENT can establish an affinity to a proper TCP/IP stack. The Policy Agent uses the TCP/IP image name configured in the TcpImage statement in the Policy Agent configuration file to determine to which TCP/IP it uses to install the policies. We used the following statement in /SC30/etc/pagent.sc30.conf:

```
TcpImage TCPIPC /etc/sc30.tcpipc_image.conf  FLUSH PURGE 600
```

The Pagent server can be stopped by:

- Using the **cancel** command, for example, C PAGENT
- Using the **kill** command in the z/OS shell

- Using the operator command STOP

The following command with the TERM signal enables PAGENT to clean up resources properly before terminating:

```
kill -s TERM pid
```

The PAGENT process ID can be obtained using the following z/OS UNIX command:

```
ps -A
```

Verification of our SD QoS policy implementation

We verified our QoS policy setup with the command shown in Example 5-5.

Example 5-5 Using the OMVS pasearch command on SC30

```
CS02 § SC30:/u/cs02> pasearch -p tcpipc -n

TCP/IP pasearch CS V1R7                  TCP/IP Image:      TCPIPC
  Date:                 11/01/2005       Time:  19:46:36
  QoS Instance Id:      1130892279

policyRule:             sdtnrule
  Qos Action:           tnaction
CS02 § SC30:/u/cs02>
```

The output of the **pasearch** command verified that our PolicyRule sdtnrule is active on our distribution stack on SC30. The **pasearch** command in Example 5-6 verified that our PolicyRule tgtnrule was active on our target stack on SC31.

Example 5-6 Using the OMVS pasearch command on SC31

```
CS02 § SC31:/u/cs02> pasearch -n

TCP/IP pasearch CS V1R7                  TCP/IP Image:      TCPIPC
  Date:                 11/01/2005       Time:  19:48:06
  QoS Instance Id:      1130891171

policyRule:             tgtnrule
  Qos Action:           tnaction
CS02 § SC31:/u/cs02>
```

Note: More detailed output, including the policy definitions, can be obtained by using **pasearch** without any parameters.

We started five TN3270 connections and used the NETSTAT,VDPT,DETAIL command shown in Example 5-7.

Example 5-7 NETSTAT,VDPT,DETAIL command on SC30

```
D TCPIP,TCPIPC,N,VDPT,DETAIL
 EZD0101I NETSTAT CS V1R7 TCPIPC 288
 DYNAMIC VIPA DESTINATION PORT TABLE:
 DEST:         10.30.30.33..23
   DESTXCF:    10.30.20.100
   TOTALCONN: 0000000005  RDY: 003  WLM: 15  TSR: 100
   FLG: SERVERWLM
     TCSR: 100  CER: 100 SEF: 100
     QOSPLCACT: *DEFAULT*
       W/Q: 15
     QOSPLCACT: TNACTION                       1
       W/Q: 15
 DEST:        10.30.30.33..23
   DESTXCF:   10.30.20.101
   TOTALCONN: 0000000000  RDY: 001  WLM: 16  TSR: 100
   FLG: SERVERWLM
     TCSR: 100  CER: 100 SEF: 100
     QOSPLCACT: *DEFAULT*
       W/Q: 16
    QOSPLCACT: TNACTION
       W/Q: 16
 DEST:        10.30.30.33..23
   DESTXCF:   10.30.20.102
   TOTALCONN: 0000000000  RDY: 001  WLM: 15  TSR: 100
   FLG: SERVERWLM
     TCSR: 100  CER: 100 SEF: 100
     QOSPLCACT: *DEFAULT*
       W/Q: 15
     QOSPLCACT: TNACTION
       W/Q: 15
 3 OF 3 RECORDS DISPLAYED
 END OF THE REPORT
```

The output showed that our QoS Policy for incoming TN3270 connections was in place and active 1. We then issued a **netstat -j** command on SC30 to verify that our PolicyRule was working, as shown in Example 5-8.

Example 5-8 OMVS output of the netstat -j command on SC30

```
MVS TCP/IP NETSTAT CS V1R7       TCPIP Name: TCPIPC          20:09:27
PolicyRuleName:  sdtnrule
  FirstActTime:     11/01/2005 19:44:40
  LastMapTime:      11/01/2005 20:07:51
  TotalBytesIn:     760
  TotalBytesOut:    0
  TotalInPackets:   30
  TotalOutPackets:  0
  OutBytesInProf:   0
  OutPacksInProf:   0
  TotalBytesReTrn:  0
  TotalPacksReTrn:  0
  ReTrnTimeouts:    0
  AcceptConn:       5     1
  DeniedConn:       0
  ActConnMap:       5            Status:            Active
  SmoothRTTAvg:     0            SmoothRTTMdev:     0
  SmoothConnDlyAvg: 0            SmoothConnDlyMdev: 0
  AcceptQDelayAvg:  0            AcceptQDelayMdev:  0
CS02 § SC30:/u/cs02>
```

All five TN3270s were recognized 1 by our policy and distributed correctly by the Sysplex Distributor.

Note: More information about the SD policy and the SD performance monitor policy can be found in *z/OS V1R7.0 CS: IP Configuration Guide*, SC31-8775, and *z/OS V1R7.0 CS: IP Configuration Reference*, SC31-8776. More detailed information can also being found in *Communications Server for z/OS V1R7 TCP/IP Implementation Guide Volume 4: SAF and Policy based network security*, SG24-7172.

5.1.4 Portsharing

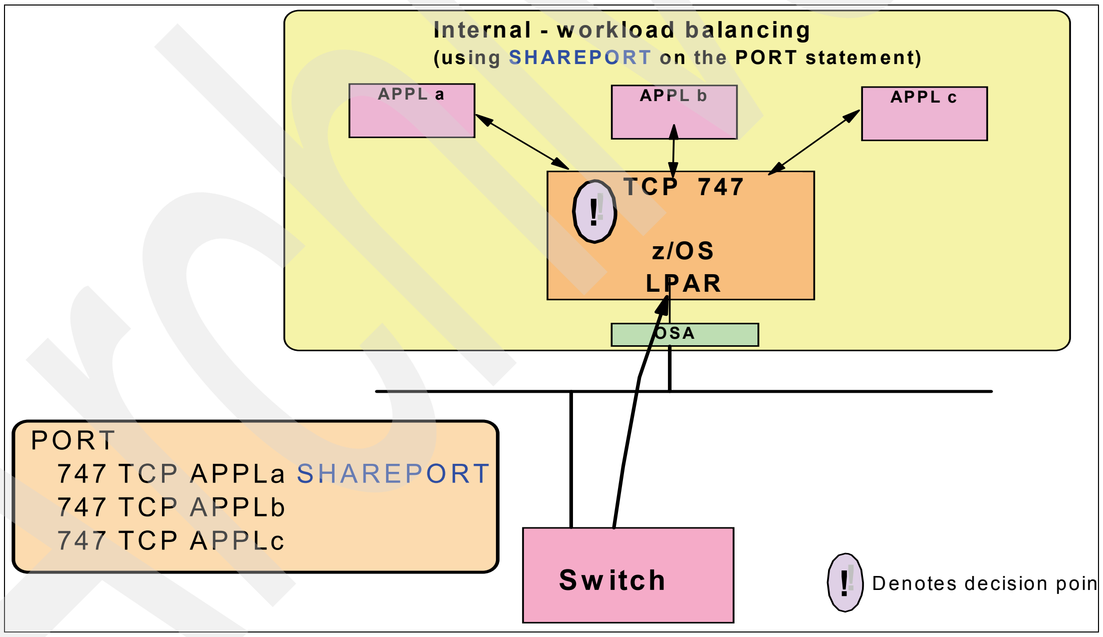
For a TCP server application to support a large number of client connections on a single system it might be necessary to run more than one instance of the server application. (This assumes the application design allows this usage, of course.) *Portsharing* is a method to distribute workload for IP applications *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 SD. However, portsharing may be used in addition to SD operation. z/OS currently supports two forms of portsharing, SHAREPORT and 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 servers' Efficiency Fractions (SEFs) of the listeners sharing the port.

  The 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.

  Figure 5-6 depicts a simple configuration where SHAREPORT is used for internal workload balancing.


Internal - workload balancing
(using SHAREPORT on the PORT statement)
APPL a
APPL b
APPL c
TCP 747
!
z/OS
LPAR
OSA
Switch
!
Denotes decision poin

```
PORT
  747 TCP APPLa SHAREPORT
  747 TCP APPLb
  747 TCP APPLc
```

Figure 5-6 The TCP/IP stack is the decision maker using a weighted round-robin method

- SHAREPORTWLM

  New with CS for z/OS V1R7, the SHAREPORTWLM option can be used instead of SHAREPORT. Like 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 reflect the listener's capacity to handle additional work.

Figure 5-7 depicts a simple configuration where SHAREPORTWLM is used for internal workload balancing.

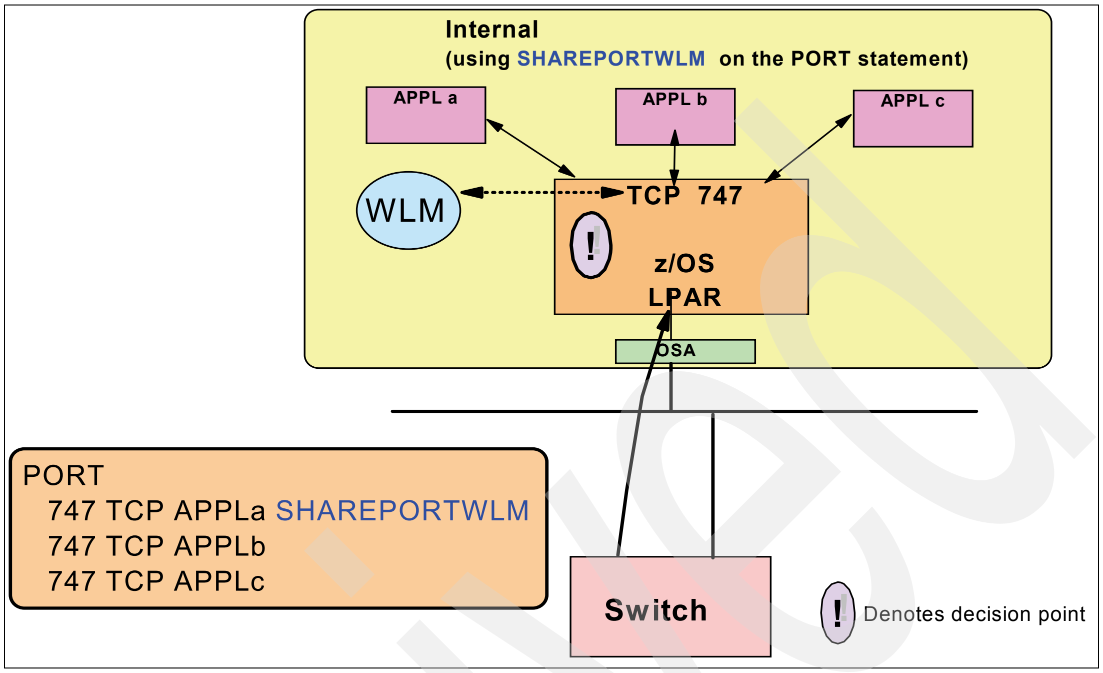


Figure 5-7 TCP/IP stack is the decision maker using a server-specific WLM data

Figure 5-8 shows the syntax of the PORT statement, including the new parameter SHAREPORTWLM.

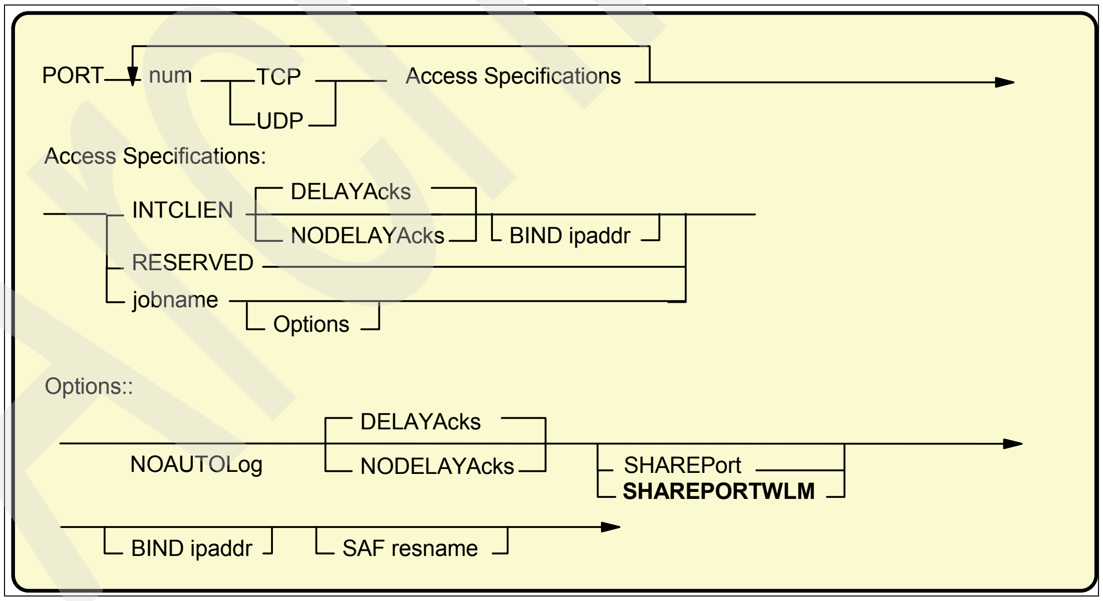


Figure 5-8 Configure the new parameter, SHAREPORTWLM, on the PORT statement

5.1.5 Optimized routing

Multiple instance applications, when running in a Sysplex Distributor parallel sysplex environment, cannot directly influence when and how many connections they will get. They rely on the Sysplex Distributor for those decisions. Prior to z/OS 1.7, after the target application server was chosen, all distributed DVIPA traffic was forwarded by the Sysplex Distributor routing stack to the target stacks using Dynamic XCF Interfaces. While leveraging our existing sysplex XCF communication links is simple because the communication paths between all systems are defined automatically, the XCF path may not be the optimal path for forwarding all traffic from the routing stack. The z/OS V1R7.0 Communications Server introduced 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.

With the z/OS V1R7.0 Communications Server, we can control the path to be taken to the target application server by using the new VIPAROUTE statement. The VIPAROUTE statement indicates which IP address on the target stack is to 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 specifies the Dynamic XCF address of the target stack. The second parameter of the VIPAROUTE statement 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 10.1.200.1 PORT DESTIP ALL
VIPAROUTE DEFINE 10.1.1.2 192.168.1.1
```

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 CEC
- OSA-Express cards when the target server is in another CEC

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

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

Figure 5-9 on page 99 illustrates the use of DVIPA IP forwarding using OSA-Express adapters.

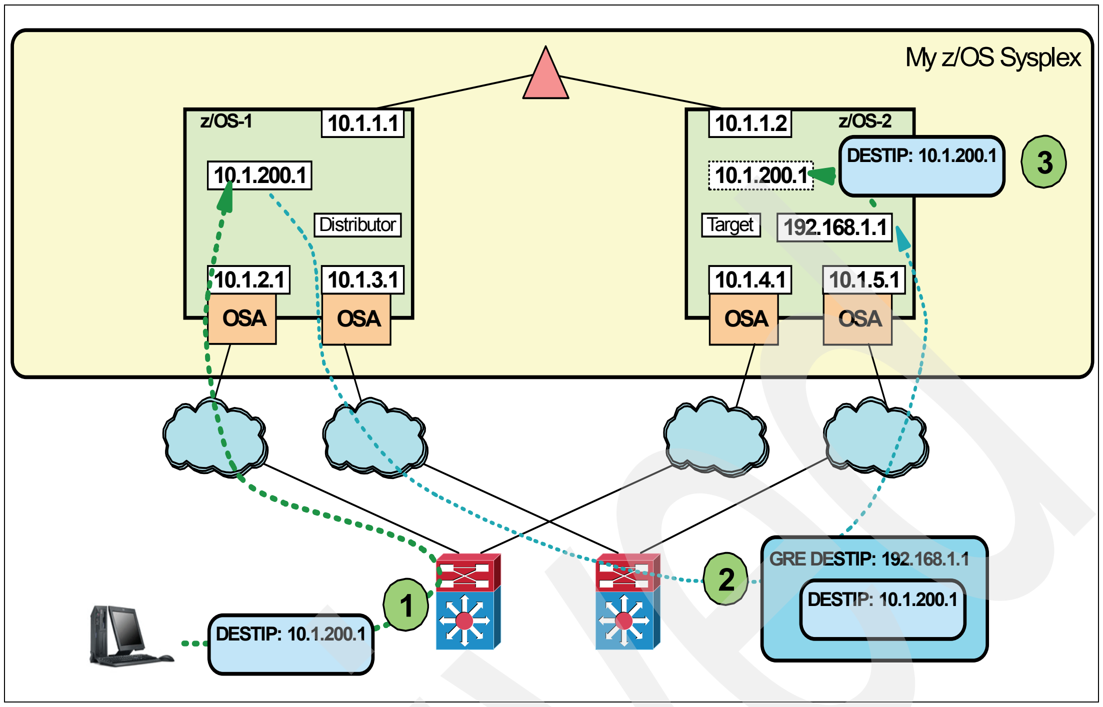


Figure 5-9 Intra-sysplex IP communication using OSA-Express instead of DynamicXCF

When a connection from the client is processed by Sysplex Distributor, it determines if a matching VIPAROUTE statement has been specified. If it has, the best available route will be 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, packets are sent from the distributor to the target encapsulated in a GRE wrapper. The outer IP header contains the VIPAROUTE target IP address as its destination IP address and the distributor's dynamic XCF address as the source IP address.

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 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:

- For a pre-V1R7 target stack all DVIPA packets distributed from the routing stack must be sent over the dynamic XCF interfaces.
- 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

Note: The new VIPAROUTE statement also influences the IP address that is returned to Cisco routers when the Sysplex Distributor is configured as a Service Manager for the Cisco Multi-Node Load Balancing (MNLB) function (discussed in 5.3.4, "Sysplex Distributor configured as Cisco Service Manager" on page 111).

We customized our distribution stack on system SC30 to use this new function. For an implementation example please refer to 5.4.1, "Sysplex Distributor using SERVERWLM and VIPAROUTE" on page 114.

5.1.6 Improved workload distribution

With z/OS V1R7 we have a new sysplex distribution feature providing server-specific WLM recommendations for load balancing. The distribution of new connection requests can now be based on the actual workload of a target server. This new interface (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 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.

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.

Figure 5-10 on page 101 illustrates the BASEWLM distribution method using QoS data.

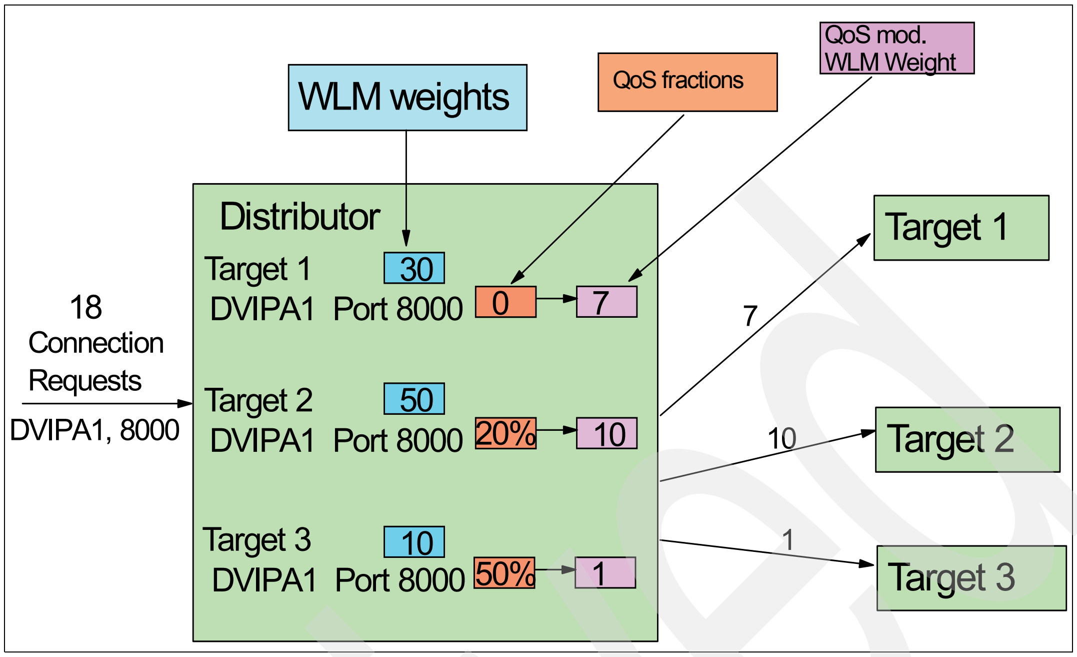


Figure 5-10 BASEWLM distribution method

In the example illustrated in Figure 5-10, BASEWLM system weights of 50, 30, and 10 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. 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 on page 102.

Table 5-1 BASEWLM weight calculations

	Example WLM weights	**Example QoS fractions**	**QoS-modified weights**	**Normalized weights**
Target 1	30	0%	30 = (30 - (30 * 0%))	7
Target 2	50	20%	40 = (50 - (50 * 20%))	10
Target 3	10	50%	5 = (10 - (10 * 50%))	1

Note: The normalization algorithm used by the z/OS V1R7.0 Communications Server is:

> 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.

Please note, however, that this approach may be changed in future releases of the z/OS Communications Server.

Continuing with our example, let us assume that 18 connection requests arrive for DVIPA1, port 8000. Based on the QoS-modified WLM weights for this DVIPA, port, and service level, 7 connections are distributed to target 1, 10 connections to target 2, and one connection 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). But if the service class of the server is of low importance then it may 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 may appear lightly loaded since applications are not processing any new work.
- QoS fractions monitor target/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

New in z/OS V1R7, WLM can assign a weight based on how well the server is meeting the goal of its service class the displaceable capacity for the new work based on the importance of its service class. Figure 5-11 illustrates this new server-specific WLM distribution method.

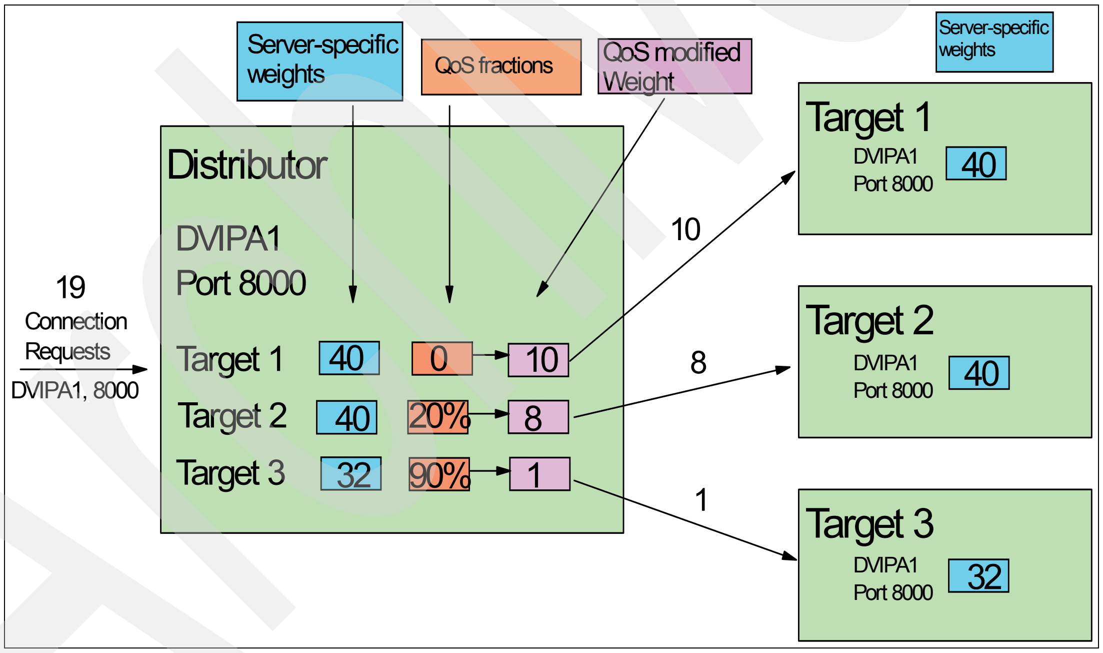


Figure 5-11 Server-specific WLM

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 follows in Table 5-2.

Table 5-2 Server-specific WLM weight calculation

	Server-specific weight x QOS fraction	Normalized weight
Target 1	40 = (40 - (40 * 0%))	10
Target 2	32 = (40 - (40 * 20%))	8

	Server-specific weight x QOS fraction	Normalized weight
Target 3	4 = (32 - (32 * 90%))	1

Note: The normalization algorithm used by the is:

> If any server-specific WLM weight is greater than 16, then all weights are normalized by dividing by 4 (and ignoring any remainder) after applying QoS fractions.

Please note, however, that this approach may be changed in future release of z/OS Communications Server.

WLM depends on the server receiving work to change server weights. Even if a server weight is zero, 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, 10 connections are distributed to target 1, 8 connections to target 2, and one to target 3.

The following restrictions should be noted:

- 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 new 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 will continue to use the new WLM server recommendations.

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

Figure 5-12 on page 104 illustrates the VIPADISTRIBUTE DISTMethod statement, including the new parameter.

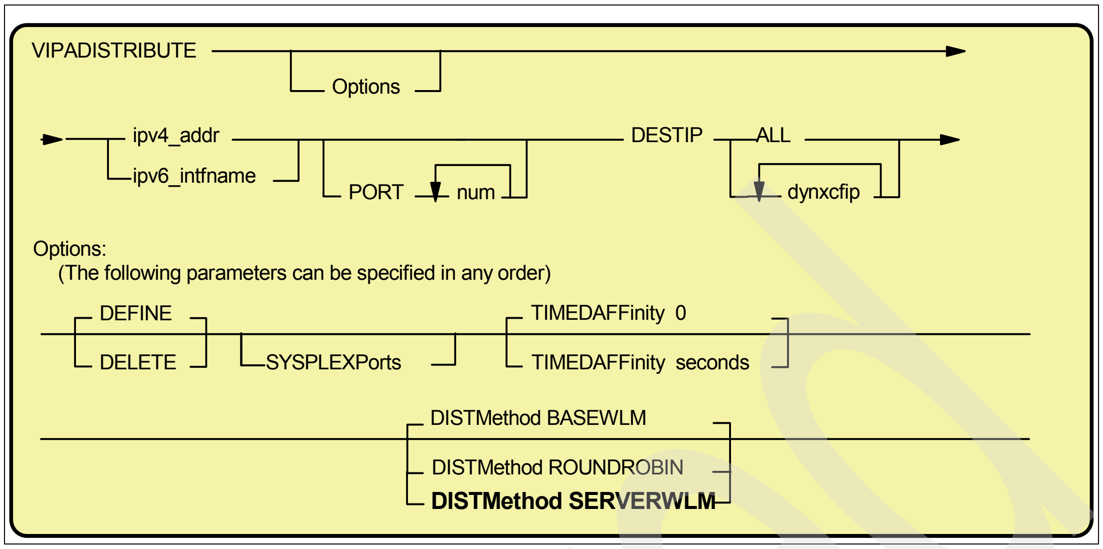


Figure 5-12 New VIPADISTRIBUTE parameter: DISTMethod SERVERWLM

Sysplex autonomics health monitor for target stacks

When the Sysplex Distributor distributes connections to servers on a set of target stacks, there may be cases where a server's ability to process new connections is slow or bad due to external issues.

With z/OS V1R7 we have a new function that allows the SD to monitor the target server's responsiveness at intervals of approximately one minute. If a target server experiences response problems SD 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 5-13 illustrates the use of the enhanced sysplex autonomics health monitor function.

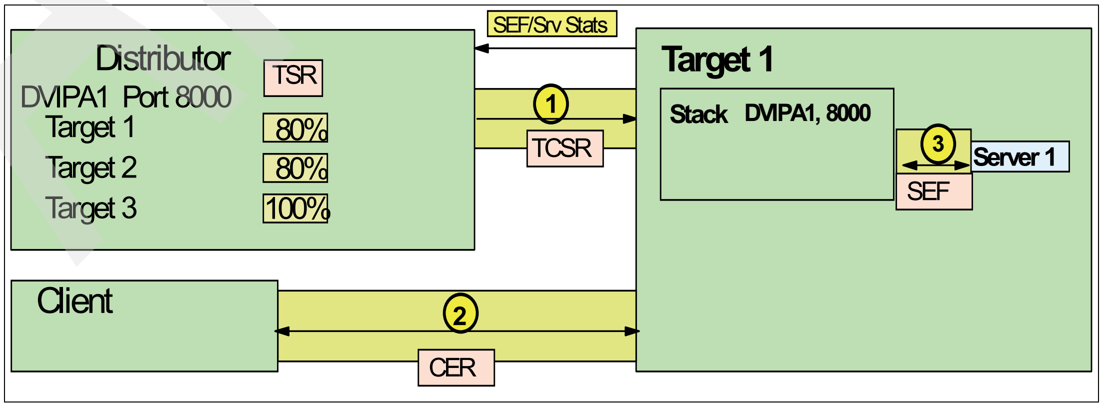


Figure 5-13 Possible weak points, monitored automatically by the sysplex autonomics health monitor

We can monitor the three potential weak point in this example:

- 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.

- 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.
- 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 SD then creates a Target Server Responsiveness fraction (TSR). This, in turn, influences the decisions made by the Sysplex Distributor. The higher the value, the healthier the condition of the target server.

Figure 5-14 illustrates the influence of the sysplex autonomics health monitor based on the sample we used for server-specific WLM.

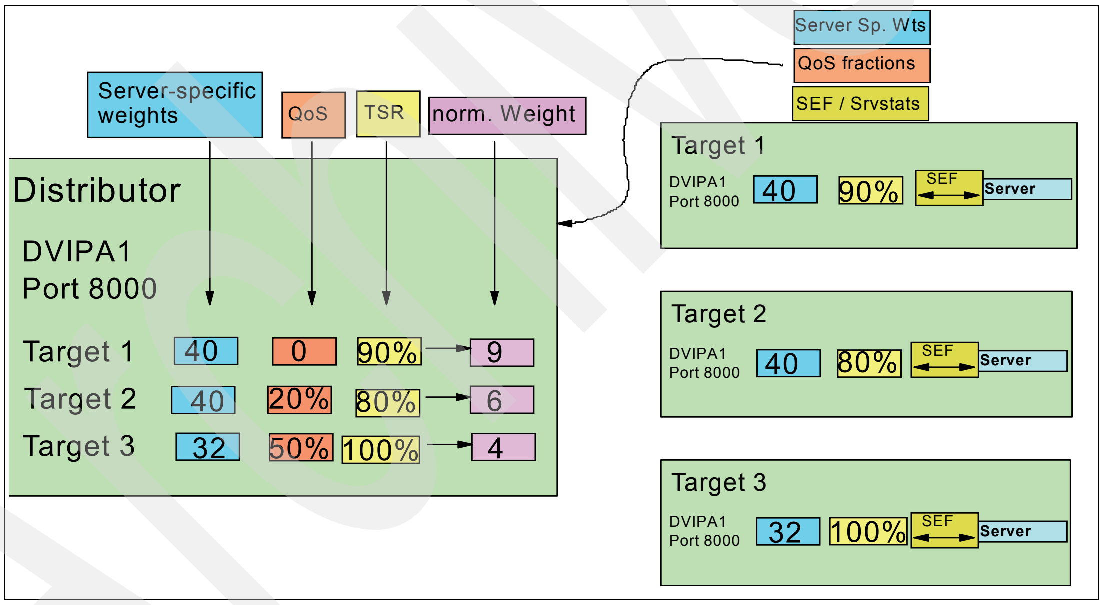


Figure 5-14 Sysplex autonomics health monitor for target stacks

Table 5-3 Server-specific WLM weight calculation with autonomics

	WLM weight x QOS fraction	**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

The weight for each target server is calculated as follows:

1. First the QoS service level is applied against the raw WLM server weight.

2. Then the TSR fraction is calculated from the SEF value and server statistics that are received from the target and from information that the distributor keeps for each server. If the QoS modified WLM weight is 32 and the TSR fraction is 80%, the new modified weight is 25 (32 * 80%).
3. Finally, the weights are normalized.

Note: The normalization algorithm use by the z/OS V1R7.0 Communications Server is:

> If any server-specific WLM weight is greater than 16, then all weights are normalized by dividing by 4 (and ignoring any remainder) after applying QoS and TSR fractions.

Please note, however, that this approach may be changed in future releases of the z/OS Communications Server.

Note: This function is automatically enabled if SERVERWLM distribution is being used and the Sysplex Distributor and all target stacks are started at the V1R7 release level.

5.2 Importance of internal application workload balancing

Workload balancing becomes important when system workloads are nearing capacity. In such situations, if your systems are all fairly homogeneous in terms of relative capacities, and the typical transactions are fairly homogenious in terms of system resource requirements, a simple round-robin workload distribution approach may be quite adequate to meet your needs. If, however, either your servers or your transactions are not homogeneous, significant utilization imbalances may occur over time—reducing the overall capacity and performance of your distributed system. Additionally, if you need to provide different levels of services for different user constituencies, you will need to use a more powerful workload distribution approach than simple round-robin. For such complex situations, the internal application workload balancing provided by Sysplex Distributor may prove to be a critical part of your distributed application solution. Based on real-time information about potential target servers, it can make the optimal decisions for workload distribution. The decision-making mechanism includes detection of failing or blocked servers, and this improves the high availability of the IP application as well.

5.3 Design examples

We implemented several examples of internal application workload balancing. In this section we describe the examples, including the respective advantages and disadvantages of each. Our general goals were as follows:

- No interruption of application service should occur for planned or unplanned outages of LPARs, network interfaces, application instances, and switches/routers.
- A high degree of scalability is important. This applies to additions of LPARs, network interfaces, applications, switches, and routers.
- A well-balanced distribution of workload among the application server instances is important.
- We want to establish a single system image for the application users. Users should not be aware of the exact techniques used to provide their service.
- It must be possible to control the use of network resources, so that more important applications can obtain performance when needed.

Where practical in our test environment, we tried to minimize single points-of-failure. The specific environment that we used for our design examples (shown in Figure 5-15 on page 108) includes the following:

- Three LPARs (in a single system) named: SC30, SC31, and SC32. SC30 is our Sysplex Distributor stack while SC31 is defined as the backup SD.
- We used two different OSA-Express GbE adapters (one port used per physical OSA Card) shared in QDIO mode.
- We ran TN3270 in each LPAR (using DVIPA for our TN3270 servers).
- We used two Cisco Catalyst 6500 Series Switches.
- We implemented the OSPF dynamic routing protocol to provide rerouting around failures.

Note: We used a single CEC, which is a single point-of-failure. A major production sysplex should include more than one CEC.

The following design examples are discussed:

- Sysplex Distributor using BASEWLM
- Sysplex Distributor using round-robin
- Sysplex Distributor using server-specific WLM
- Sysplex Distributor configured as Cisco Service Manager
- Portsharing using SHAREPORTWLM

5.3.1 Sysplex Distributor using BASEWLM

We started with a typical SD situation where three application instances of the TN3270 server are to be available in a z/OS sysplex environment. Figure 5-15 on page 108 shows how TN3270 connections are load balanced across all three LPARS, based on the standard WLM distribution method (DISTMETHOD BASEWLM).

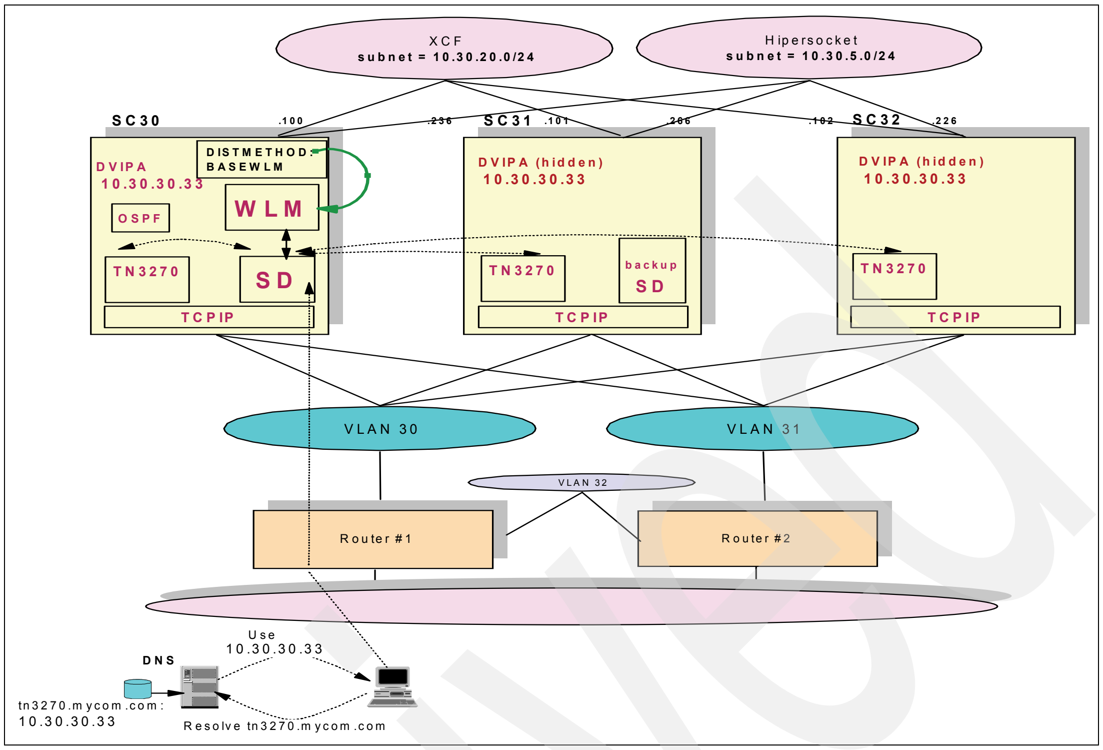


Figure 5-15 Workload balancing of TN3270 using Sysplex Distributor with DISTMETHOD BASEWLM

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.

Dependencies

You must implement either a dynamic routing protocol using OMPROUTE (as discussed in Chapter 4, “VIPA with dynamic routing” on page 59) or use ARP takeover (as discussed in Chapter 3, “VIPA without dynamic routing” on page 39) 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).

Advantages

This implementation provides several advantages:

- Ease of configuration. The Sysplex Distributor greatly eases configuration complexity. Additional target applications can be added simply.
- More accurate measure of server load. The Sysplex Distributor uses WLM-provided server load information. Additionally, the set of potential target stacks can be different depending on which client is requesting the connection. This allows for the reservation of particular stacks to a subset of clients in a straightforward manner.
- High availability. This configuration uses DVIPA takeover and takeback support. The distributing function can be backed up and taken over.

- Easy integration. End users are not aware of the distribution being performed by the Sysplex Distributor. They simply connect to a server IP address (or host name).
- Independence from DNS. Sysplex Distributor operation does not depend on host name resolution for load balancing. Rather, the workload distribution occurs at TCP connection setup. As a result, the Sysplex Distributor is not susceptible to host name caching effects or increased network utilization resulting from low TTL settings of DNS resource records, as is the case with DNS/WLM.
- No special external hardware required. Because all the Sysplex Distributor function is contained within the sysplex, no external hardware is necessary to take advantage of this function.
- Performance. The Sysplex Distributor can take advantage of the homogeneity within the cluster. That is, forwarding connections and data through the distributing stack is fast and efficient.
- Real-time workload balancing for TCP/IP applications. This is true even if clients cache the IP address of the server (a common problem for DNS/WLM).
- Enhances Dynamic VIPA support for nondisruptive application server instance movement.
- Immediate visibility of new server instances or loss of a server instance. Target stacks are aware when an application instance has a listening socket that could accept connections via a distributed DVIPA, or when such a socket is closed (or the application instance terminates for any reason). The target stacks notify the routing stack proactively, eliminating the need for application advisors on the routing stack.

Considerations

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 z9 or zSeries servers and resident within the same sysplex. This may impose some limitations on flexibility.

5.3.2 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. (The round-robin method was introduced with z/OS V1R5.) The key PROFILE statements are:

```
VIPADynamic
   VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.33
   VIPADISTribute DISTMethod ROUNDROBIN 10.30.30.33
    PORT 23 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
```

Note: Be aware that when round-robin distribution is chosen it supersedes any defined policies.

Figure 5-16 on page 110 illustrates three incoming TN3270 connections being evenly distributed across three LPARs.

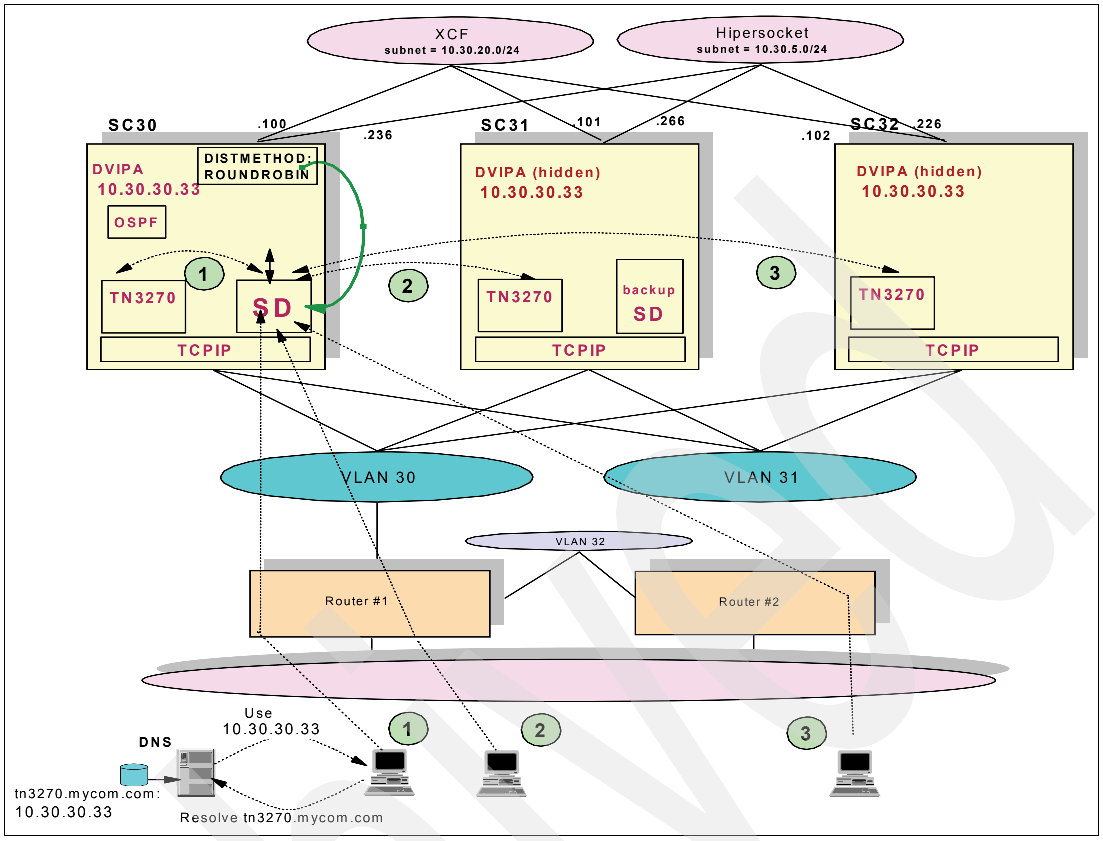


Figure 5-16 Even (round-robin) workload distribution

Dependencies

The routing stack and all backup routing stacks for a Distributed DVIPA should be at z/OS V1R4 with the enabling PTF or later for this environment to work properly for takeover situations.

Advantages

In an environment where too many connections are sent the LPAR with the largest capacity, we could have a larger risk if an outage occurs for this LPAR. A more even distribution among servers might be appropriate, as provided by using round-robin as the internal application workload distribution method.

Considerations

The round-robin distribution method does not take into account server capacity when distributing new connection requests.

5.3.3 Sysplex Distributor using server-specific WLM

Server-specific Workload Manager (WLM) controls for load balancing is a new sysplex distribution feature in z/OS V1R7. The key PROFILE statements are:

```
VIPADynamic
   VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.33
   VIPADISTribute DISTMethod SERVERWLM 10.30.30.33
    PORT 23 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
```

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 5-17 illustrates how incoming TN3270 connections are load balanced using server-specific WLM weights.

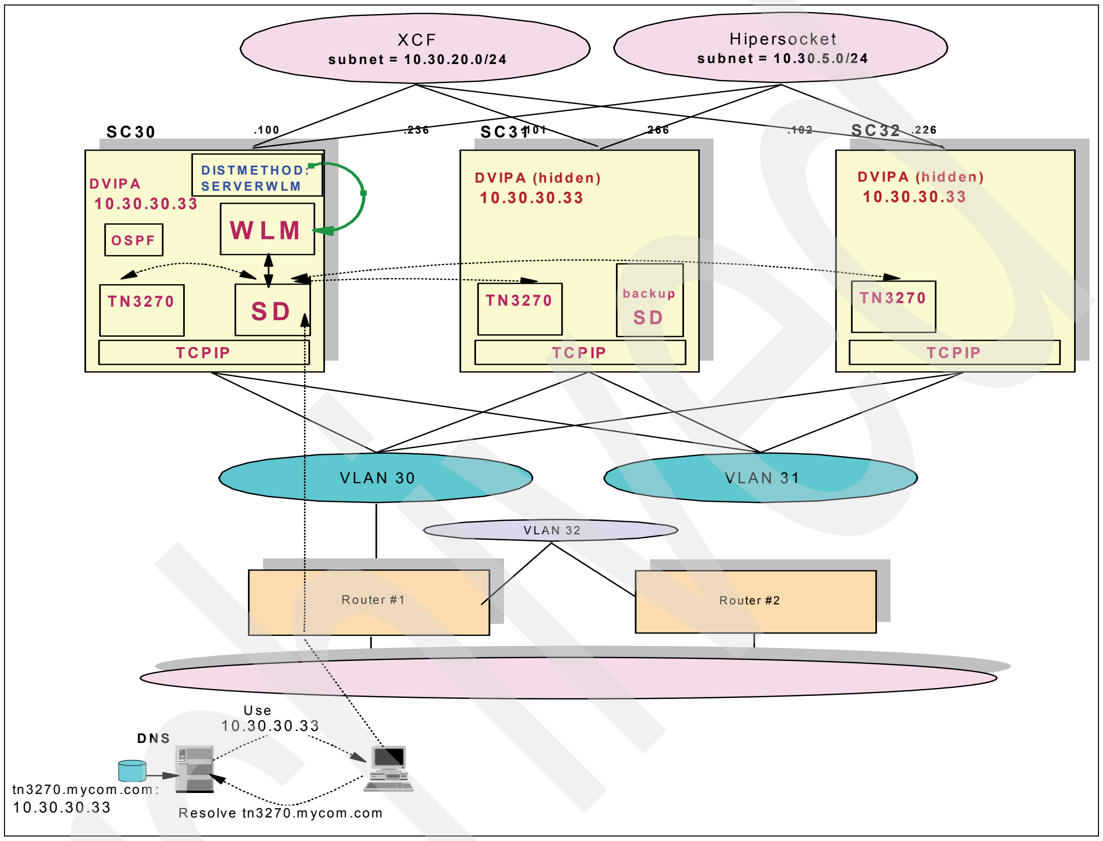


Figure 5-17 Server-specific WLM Workload balancing of TN3270 connections

Dependencies

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.

Advantages

With server-specific WLM we can determine how well a specific server on a stack is meeting the goals of its WLM service class for the transactions that it is servicing. This additional granularity enables the distributor to balance the workload more effectively.

Considerations

The older BASEWLM system weight recommendations are not compatible with the new WLM server recommendations. Therefore, the new WLM server recommendations support requires the z/OS V1R7.0 Communications Server on all stacks.

5.3.4 Sysplex Distributor configured as Cisco Service Manager

In this example we use the Sysplex Distributor to determine which TN3270 instance is best suited for a TN3270 connection request. However, when a decision has been made, we use the CISCO Service Manager to inform the Forwarding Agent running on the external router/switch about the chosen target stack. This is done through the Cisco CASA protocol. With this information the router/switch forwards the follow-up packets for this connection directly to the target stack. The distribution stack is completely bypassed with this method, providing a more direct path and saving System z9 or zSeries CPU cycles.

The relevant PROFILE statements are:

```
VIPADynamic
VIPADEFine MOVEable IMMEDiate SERVICEMGR 255.255.255.0 10.30.30.33
   VIPASMPARMS SMMCAST 224.0.1.2 SMPORT 1637
   VIPADISTribute DISTMethod SERVERWLM 10.30.30.33
    PORT 23 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
```

In addition to the PROFILE statements, we need to create appropriate Cisco router definitions. These are given in “Sysplex Distributor configured as Cisco Service Manager” on page 122.

OSPF is used as the dynamic routing protocol for advertising of the VIPA addresses and to ensure connectivity by rerouting connections around failures.

A diagram of this example is given in Figure 5-18.

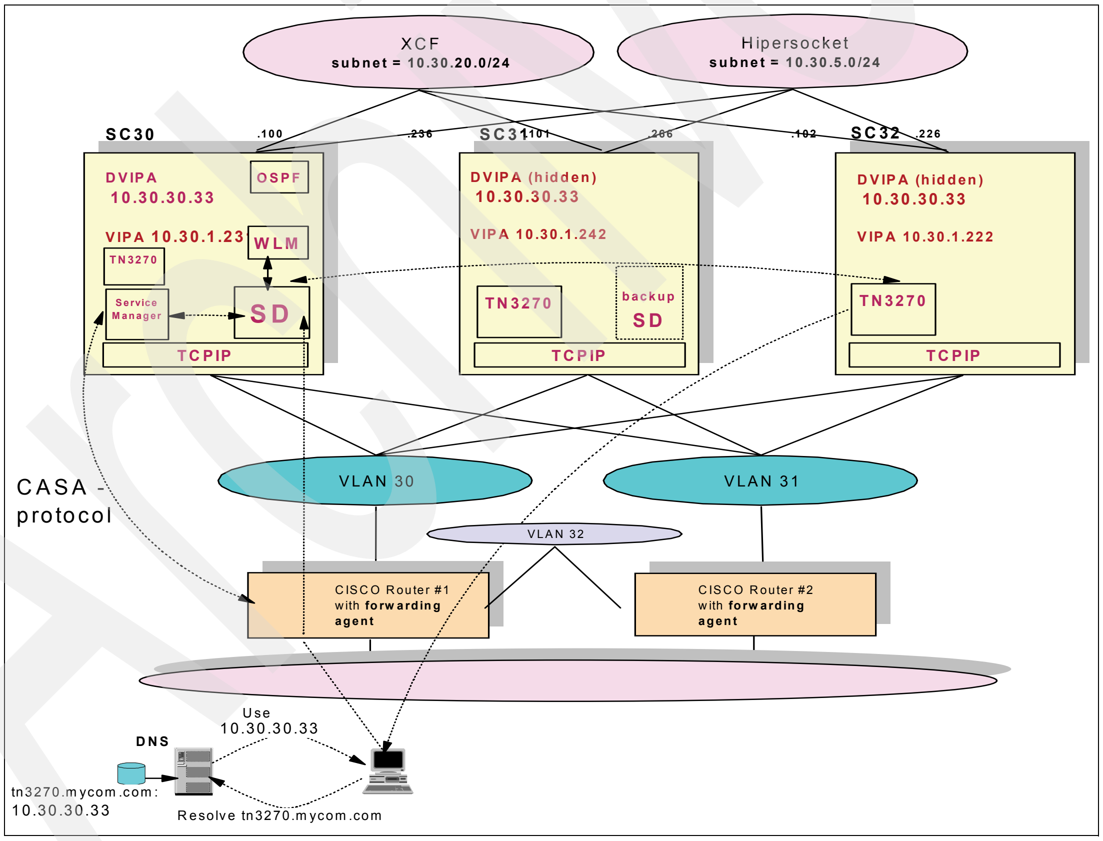


Figure 5-18 Workload balancing of TN3270 using Sysplex Distributor configured as Service Manager

In this case, our Sysplex Distributor stack is configured as Cisco Service Manager, running in LPAR SC30, and we use static VIPA for GRE tunnel between the potential target stacks and the CISCO switches. Our two Cisco Catalyst 6500 Series Switches are configured to run as CISCO MNLB forwarding agents.

Dependencies

This *hybrid* SD/MNLB implementation is only supported with Cisco equipment.

Advantages

There are significant advantages to implementing the Sysplex Distributor with the Service Manager function in an MNLB environment. These include:

- The inbound path for all TCP data to the target server now goes directly from the Forwarding Agent to the target server, avoiding potential bottlenecks caused by routing all inbound traffic through the Sysplex Distributor routing stack.
- The Service Manager in the Sysplex Distributor provides load-balancing decisions based on Quality of Service (QoS) from the Policy Agent (PAGENT) definitions in the z/OS system, in addition to WLM information.
- A backup distributing stack can provide for failure recovery of the Service Manager.

Considerations

The Cisco Service Manager communicates with Cisco Forwarding Agents, located in routers, using the Cisco propriety CASA protocol. Therefore this *hybrid* SD/MNLB solution only works with Cisco routers.

5.3.5 Portsharing using SHAREPORTWLM

In an environment where we must 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.

Figure 5-19 on page 114 illustrates this situation.

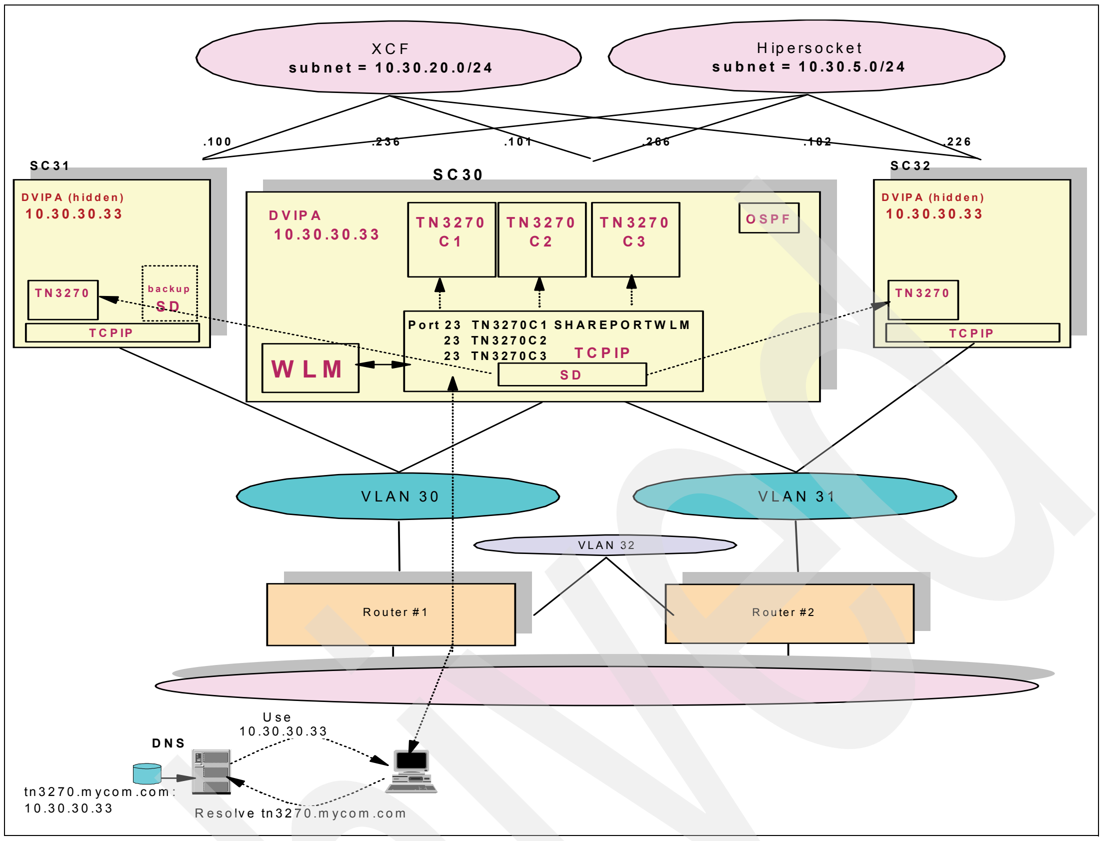


Figure 5-19 Internal workload balancing using SHAREPORTWLM

In this case, we added three TN3270 instances running in the same LPAR (SC30).

5.4 Implementations

This section walks through the implementation of three examples:

- Sysplex Distributor using SERVERWLM and VIPAROUTE
- Sysplex Distributor configured as Cisco Service Manager
- Multiple servers sharing a port using SHAREPORTWLM

5.4.1 Sysplex Distributor using SERVERWLM and VIPAROUTE

To implement the Sysplex Distributor function, we 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.

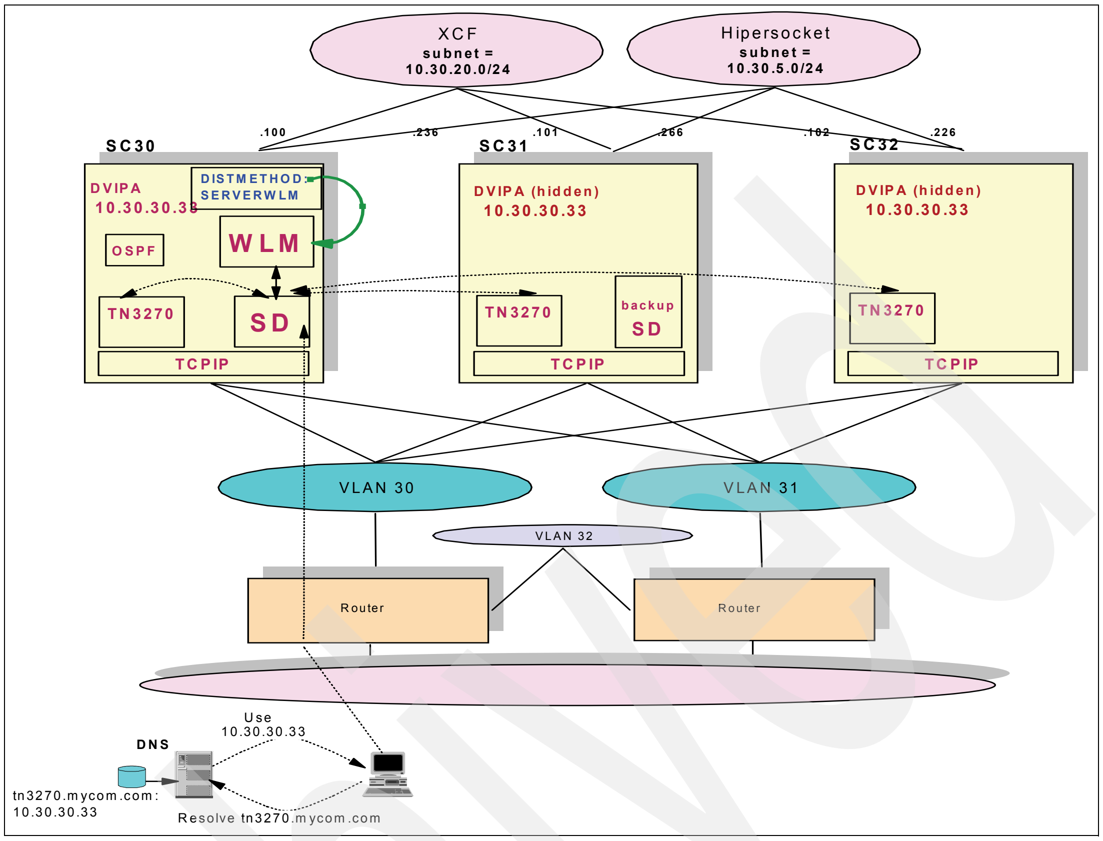


Figure 5-20 Server-specific WLM Workload balancing of TN3270 connections

Implementation tasks

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 those IP stacks that are active under a multi-stack 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) 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 10.30.30.33 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.

Configuration

Example 5-9 contains the most important parameters of our TCP/IP profile that were necessary to define, to enable the distribution of TN3270 connections.

Example 5-9 Profile TCPIPC SC30 Sysplex Distributor

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING                  3
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
DYNAMICXCF 10.30.20.100 255.255.255.0 8
VIPADynamic
 VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.33 1
 VIPADISTribute DISTMethod SERVERWLM 10.30.30.33         2
  PORT 23 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
 VIPAROUTE 10.30.20.100 10.30.1.230 4
 VIPAROUTE 10.30.20.101 10.30.1.241 4
 VIPAROUTE 10.30.20.102 10.30.1.221 4
ENDVIPADYNAMIC
```

The following are important considerations for this PROFILE:

- 1 We used the IP address 10.30.30.33 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 is new with z/OS V1R7 and 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 5-10 and Example 5-11 on page 117. These define SC31 and SC32 to act as backup Sysplex Distributor in case the primary Sysplex Distributor on SC30 fails.

Example 5-10 Profile TCPIPC SC31 target and backup

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
DYNAMICXCF 10.30.20.101 255.255.255.0
VIPADynamic
```

```
 VIPABACKUP 1 MOVEABLE IMMEDIATE 255.255.255.0 10.30.30.33 1
ENDVIPADynamic
```

Example 5-11 Profile TCPIPC SC32 target and backup

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
DYNAMICXCF 10.30.20.102 255.255.255.0
 VIPADynamic
 VIPABACKUP 2 MOVEABLE IMMEDIATE 255.255.255.0 10.30.30.33 2
 ENDVIPADynamic
```

Important points here are:

- 1 If the Sysplex Distributor on SC30 fails, SC31 will take over as the first backup distributor.
- 2 In case the Sysplex Distributor on SC30 fails *and SC31 is not active*, SC32 will take over as the second backup distributor.

Verification

We used these steps to verify that our implementation was correct:

1. Verification of the general dynamic VIPA setup including the backup SD
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

Once all of our IP stacks were active and running, we started our verification steps.

Issuing a NETSTAT,VIPADCFG command, as shown in Example 5-12, on our distribution stack (SC30) provided much information that helped us verify our Sysplex Distributor configuration.

Example 5-12 Display results of the VIPADCFG command issued on our distribution stack SC30

```
SC30
D TCPIP,TCPIPC,N,VIPADCFG
EZD0101I NETSTAT CS V1R7 TCPIPC 406
DYNAMIC VIPA INFORMATION:
   VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.30.30.33/24 1
      MOVEABLE: IMMEDIATE  SRVMGR: NO
   VIPA DISTRIBUTE:
   DEST:        10.30.30.33..23      2 3
     DESTXCF:   10.30.20.100        4
       SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM 5
   DEST:        10.30.30.33..23
     DESTXCF:   10.30.20.101        4
       SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM 5
   DEST:        10.30.30.33..23
     DESTXCF:   10.30.20.102        4
       SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM 5
   VIPA ROUTE:                 6
     DESTXCF:     10.30.20.100
       TARGETIP:  10.30.1.230
     DESTXCF:     10.30.20.101
       TARGETIP:  10.30.1.241
     DESTXCF:     10.30.20.102
       TARGETIP:  10.30.1.221
 END OF THE REPORT
```

Key information in this display includes the following:

- 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 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.

The results of VIPADCFG commands for SC31 and SC32 are shown in Example 5-13.

Example 5-13 Display results of the VIPADCFG command issued on SC31 and SC32

```
SC31
  EZD0101I NETSTAT CS V1R7 TCPIPC 368
  DYNAMIC VIPA INFORMATION:
    VIPA BACKUP:
      IPADDR/PREFIXLEN: 10.30.30.33
     1 RANK: 001  MOVEABLE:           SRVMGR:
END OF THE REPORT

SC32
 EZD0101I NETSTAT CS V1R7 TCPIPC 227
 DYNAMIC VIPA INFORMATION:
   VIPA BACKUP:
     IPADDR/PREFIXLEN: 10.30.30.33
    2 RANK: 002  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 1, and the output from the IP stack on SC32, that it the secondary backup 2 with rank 2.

A SYSPLEX VIPADYN command, as shown in Example 5-15 on page 119, provides concise information about all the stacks participating in the sysplex.

Example 5-14 Display of SYSPLEX VIPADYN issued on our distribution stack SC30

```
-D TCPIP,TCPIPC,SYSPLEX,VIPADYN
 EZZ8260I SYSPLEX CS V1R7 502
 VIPA DYNAMIC DISPLAY FROM TCPIPC   AT SC30

LINKNAME: VIPL0A1E1E21
IPADDR/PREFIXLEN: 10.30.30.33/24
   ORIGIN: VIPADEFINE
   TCPNAME  MVSNAME  STATUS  RANK   DIST
   -------- -------- ------ ---- ----
   TCPIPC   SC30     ACTIVE 1       BOTH  3
   TCPIPC   SC32     BACKUP 1 002 2 DEST  3
   TCPIPC   SC31     BACKUP 1 001 2 DEST  3
 10 OF 10 RECORDS DISPLAYED
```

Important information includes the following:

- 1 Actual Status of the TCP/IP stack.
- 2 Rank value determines the order on which a backup distributor will take over in case the primary Sysplex Distributor fails.
- 3 indicates whether a stack is defined as the distributor or as a destination or both.

We issued NETSTAT,HOME commands on all our stacks to obtain a list of all known IP addresses. Example 5-15 contains the output of these commands and 1 marks the DVIPA address representing the TN3270 servers.

Example 5-15 Display results of Netstat Home in all stacks

```
SC30
D TCPIP,TCPIPC,N,HOME
EZD0101I NETSTAT CS V1R7 TCPIPC 016
HOME ADDRESS LIST:
LINKNAME:   STAVIPA1LNK
  ADDRESS:  10.30.1.230
    FLAGS:  PRIMARY
LINKNAME:   OSA2080LNK
  ADDRESS:  10.30.2.232
    FLAGS:
LINKNAME:   OSA20E0LNK
  ADDRESS:  10.30.3.235
    FLAGS:
LINKNAME:   IUTIQDF6LNK
  ADDRESS:  10.30.5.236
    FLAGS:
LINKNAME:   VIPL0A1E1E21
ADDRESS:  10.30.30.33 1
     FLAGS
12 OF 12 RECORDS DISPLAYED
 END OF THE REPORT

SC31
D TCPIP,TCPIPC,N,HOME
 EZD0101I NETSTAT CS V1R7 TCPIPC 982
 HOME ADDRESS LIST:
 LINKNAME:   STAVIPA1LNK
   ADDRESS:  10.30.1.241
     FLAGS:  PRIMARY
 LINKNAME:   OSA2080LNK
   ADDRESS:  10.30.2.242
     FLAGS:
 LINKNAME:   OSA20E0LNK
   ADDRESS:  10.30.3.245
     FLAGS:
 LINKNAME:   IUTIQDF6LNK
   ADDRESS:  10.30.5.246
     FLAGS:
.LINKNAME:   VIPL0A1E1E21
   ADDRESS:  10.30.30.33 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
EZD0101I NETSTAT CS V1R7 TCPIPC 257
HOME ADDRESS LIST:
LINKNAME:   STAVIPA1LNK
  ADDRESS:  10.30.1.221
    FLAGS:  PRIMARY
LINKNAME:   OSA2080LNK
  ADDRESS:  10.30.2.222
    FLAGS:
LINKNAME:   OSA20E0LNK
  ADDRESS:  10.30.3.225
    FLAGS:
LINKNAME:   IUTIQDF6LNK
  ADDRESS:  10.30.5.226
    FLAGS:
.LINKNAME:   VIPL0A1E1E21
   ADDRESS:  10.30.30.33 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 5-16, to determine the distribution of these sessions.

Example 5-16 NETSTAT VDPT,DETAIL command at our distribution stack SC30

```
SC30
/D TCPIP,TCPIPC,N,VDPT,DETAIL
RESPONSE=SC30
  EZD0101I NETSTAT CS V1R7 TCPIPC 498
  DYNAMIC VIPA DESTINATION PORT TABLE:
  DEST:        10.30.30.33..23    1 2
    DESTXCF:   10.30.20.100       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:        10.30.30.33..23
    DESTXCF:   10.30.20.101
    TOTALCONN: 0000000004 5 RDY: 001  WLM: 07  TSR: 093
    FLG: SERVERWLM
      TCSR: 093  CER: 100 SEF: 100
      QOSPLCACT: *DEFAULT*
        W/Q: 07

 DEST:        10.30.30.33..23
   DESTXCF:   10.30.20.102
   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 key information displayed included:

- **1** is the destination IP address, which is the Sysplex Distributor IP address.
- **2** is the port number to which connections are being distributed.
- **3** is the destination XCF IP address.
- **4** is the number of applications listening on the port number selected.
- **5** is the total number of connections that have been forwarded by the Sysplex Distributor.

In our case eight of twelve TN3270 connection went to SC30. The remaining four connections went to SC31. To determine if 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: Note that 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 5-17. 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 5-17 Display of NETSTAT VCRT from SC30 IP stack

```
SC30
D TCPIP,TCPIPC,N,VCRT,DETAIL
EZD0101I NETSTAT CS V1R7 TCPIPC 896
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.30.30.33..23
  SOURCE:  9.12.4.222..1091
  DESTXCF: 10.30.20.101
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
    INTF:  IUTIQDF6LNK 2
      VIPAROUTE:  YES 1 GW:  10.30.5.246 3
DEST:      10.30.30.33..23
  SOURCE:  9.12.4.222..1094
  DESTXCF: 10.30.20.102
```

```
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
    INTF:  IUTIQDF6LNK
      VIPAROUTE:  YES       GW:  10.30.5.226
```

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: IUTIQDF6 represents our HiperSockets interface, which was used to forward the connection request within our CEC.
- **3** GW: 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.

While 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 5-18 shows VCRT list on SC32.

Example 5-18 Display of NETSTAT VCRT from our target IP stack on SC32

```
SC32
 EZD0101I NETSTAT CS V1R7 TCPIPC 287
 DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.30.30.33..23
   SOURCE:  9.12.4.222..1094
   DESTXCF: 10.30.20.102
     POLICYRULE:    *NONE*
     POLICYACTION:  *NONE*
 1 OF 1 RECORDS DISPLAYED
 END OF THE REPORT
```

5.4.2 Sysplex Distributor configured as Cisco Service Manager

The Sysplex Distributor provides a workload balancing function within a Parallel Sysplex. The Sysplex Distributor consists of a primary distributor stack and a set of target stacks. An inbound packet flows through the primary distributor stack, which then forwards the packet to the selected target stack.

The Cisco Multi-Node Load Balancer (MNLB) provides a workload balancing function, which distributes traffic through Cisco switches across multiple destination TCP/IP stacks. MNLB consists of a Service Manager (the Cisco local director, which is denoted by a cluster IP address) and a set of Forwarding Agents (Cisco switches). For a TCP connection to the cluster IP address, the Forwarding Agent forwards an incoming connection request (SYN packet) along to the service manager, which then selects a target stack and notifies the forwarding agent of this decision. The forwarding agent then sends all future packets for that TCP connection directly to the target stack.

Running our Sysplex Distributor configured as a Cisco Service Manager, as illustrated in Figure 5-21 on page 123, enabled us to use a combination of the Sysplex Distributor and the MNLB to provide workload balancing.

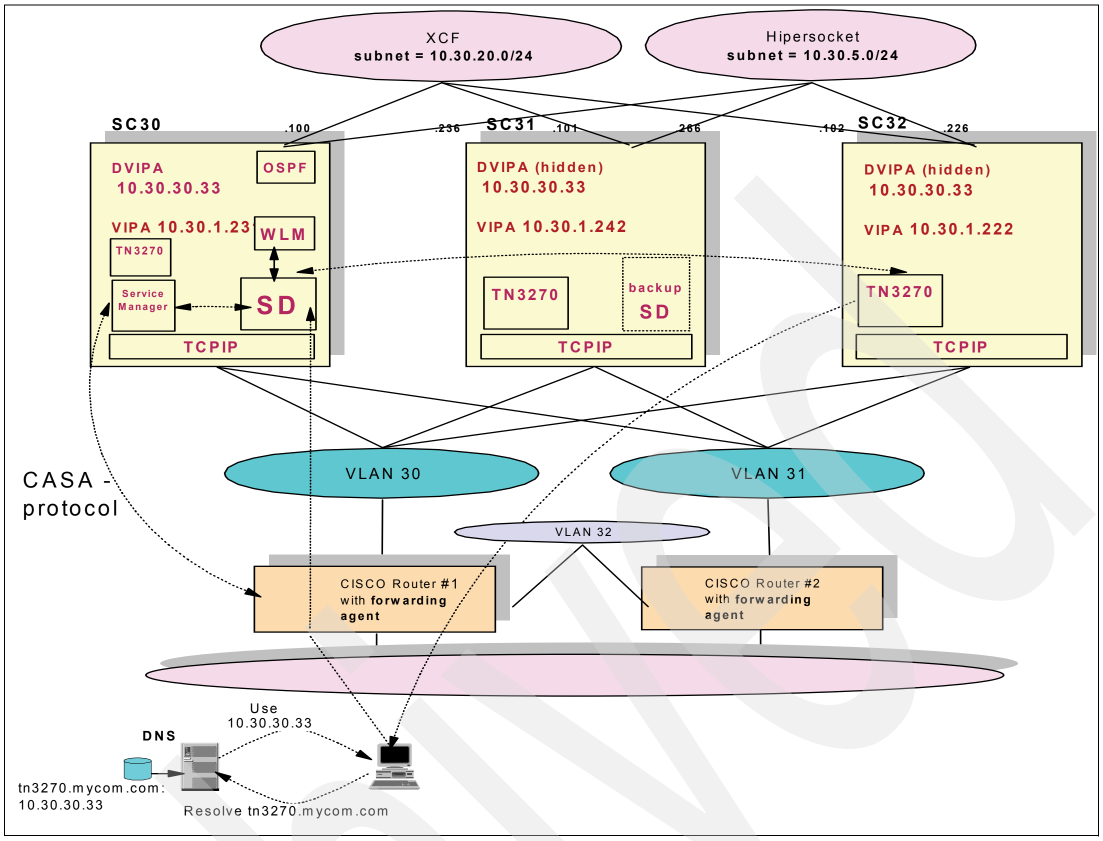


Figure 5-21 Workload balancing with SD acting as Cisco service manager

Implementation tasks

The following list shows what is needed to implement Sysplex Distributor acting as Cisco service manager:

1. Configure the Cisco router as a Forwarding Agent.
2. Define an additional static VIPA, used for our Generic Routing Encapsulation (GRE) tunnels in our MNLB environment.
3. Specify the SERVICEMGR keyword on the VIPADEFine statement in the TCP/IP profile.
4. Specify the VIPASMparms statement in the TCP/IP profile.
5. Configure GRE tunnels on the Cisco routers.
6. Configure the VIPAROUTE statement.

Our first task was to configure the Cisco router as a forwarding agent, as shown in Example 5-19.

Example 5-19 Enable forwarding agents in our Cisco router 1

```
ip multicast-routing
ip casa 8.8.8.8 224.0.1.2
 forwarding-agent 1637

interface Vlan30
 ip address 10.30.2.1 255.255.255.0
 ip igmp join-group 224.0.1.2 1
```

Note: This parameter setting 1 is important for the correct flow of the wildcards between the service manager and the forwarding agents. You must specify it either at the interface level or, like we did, as part of our VLAN definitions.

We defined an additional static VIPA in the PROFILE for each system participating in our SD/MNLB environment, as shown in Example 5-20, Example 5-21, and Example 5-22.

Example 5-20 New static VIPA for MNLB environment on SC30

```
DEVICE STAVIPA2    VIRTUAL 0
LINK   STAVIPA2LNK VIRTUAL 0     STAVIPA2
;
HOME
   10.30.1.231   STAVIPA2LNK
   10.30.1.230   STAVIPA1LNK
   10.30.2.232   OSA2080LNK
   10.30.3.235   OSA20E0LNK
   10.30.5.236   IUTIQDF6LNK
;
PRIMARYINTERFACE STAVIPA1LNK
```

Example 5-21 New static VIPA for MNLB environment on SC31

```
DEVICE STAVIPA2    VIRTUAL 0
LINK   STAVIPA2LNK VIRTUAL 0     STAVIPA2

HOME
    10.30.1.242   STAVIPA2LNK
    10.30.1.241   STAVIPA1LNK
    10.30.2.242    OSA2080LNK
    10.30.3.245    OSA20E0LNK
    10.30.5.246   IUTIQDF6LNK
;
 PRIMARYINTERFACE STAVIPA1LNK
```

Example 5-22 New static VIPA for MNLB environment on SC32

```
DEVICE STAVIPA2    VIRTUAL 0
LINK   STAVIPA2LNK VIRTUAL 0     STAVIPA2
;
HOME
    10.30.1.222   STAVIPA2LNK
    10.30.1.221   STAVIPA1LNK
    10.30.2.222    OSA2080LNK
    10.30.3.225    OSA20E0LNK
    10.30.5.226   IUTIQDF6LNK

 PRIMARYINTERFACE STAVIPA1LNK
```

Note: By specifying the PRIMARYINTERFACE statement we want to make certain that this link is designated as the default local host. If PRIMARYINTERFACE is not used, the first IP address in the HOME list becomes the default local host address.

We specified the SERVICEMGR keyword on the VIPADEFine statement in the TCP/IP profile for our Sysplex Distributor stack (SC30), as shown in Example 5-23 on page 125.

Example 5-23   Define the Cisco service manager

```
VIPADEFine MOVEable IMMEDiate SERVICEMGR 255.255.255.0 10.30.30.33
```

We specified the VIPASMparms statement in the TCP/IP profile for our Sysplex Distributor stack (SC30), as shown in Example 5-24.

Example 5-24   Specify Cisco service manager parameters: multicast address and port

```
VIPASMPARMS SMMCAST 224.0.1.2 SMPORT 1637
```

Note: Specify the same multicast group and UDP port on the VIPASMparms statement in the TCP/IP profile as are configured in the MNLB for the Cisco forwarding agents.

When using the Cisco MNLB in a configuration where there is an OSA adapter between a Cisco router, and the destination TCP/IP stacks are sharing the OSA, then we must configure GRE tunnels on the Cisco routers, as shown in Example 5-25. For more background information regarding the GRE tunnel, please refer to 5.1.2, “Sysplex Distributor configured as a Cisco Service Manager” on page 86.

Example 5-25   GRE tunnel and tunnel-related definitions we used for our Cisco router 1

```
interface Tunnel2
 ip address 2.2.2.1 255.255.255.252
 no ip mroute-cache
 tunnel source 10.30.2.1
 tunnel destination 10.30.1.231
!
interface Tunnel4
 ip address 4.4.4.1 255.255.255.252
 no ip mroute-cache
 tunnel source 10.30.2.1
 tunnel destination 10.30.1.242
!
interface Tunnel6
 ip address 6.6.6.1 255.255.255.252
 no ip mroute-cache
 tunnel source 10.30.2.1
 tunnel destination 10.30.1.222

ip route 10.30.1.222 255.255.255.255 Tunnel6
ip route 10.30.1.231 255.255.255.255 Tunnel2
ip route 10.30.1.242 255.255.255.255 Tunnel4

router ospf 300
 router-id 10.30.2.1
 log-adjacency-changes
 area 1 stub no-summary
 redistribute connected
 redistribute static
 network 2.2.2.0 0.0.0.3 area 1
 network 4.4.4.0 0.0.0.3 area 1
 network 6.6.6.0 0.0.0.3 area 1
 network 8.8.8.8 0.0.0.3 area 1
 network 10.30.0.0 0.0.255.255 area 1
 network 10.200.1.0 0.0.0.255 area 0
```

Note: The Sysplex Distributor stack is the only stack that registers the DVIPA to OSA. Therefore, if GRE tunnels are not configured, OSA will send all packets destined for the DVIPA to the Sysplex Distributor stack (or to the default router stack if the OSA is not shared with the Sysplex Distributor stack).

Packets destined for a DVIPA that are routed by the Forwarding Agents directly to TCP/IP target stacks are encapsulated to the target stack's dynamic XCF address or, when running the z/OS V1R7.0 Communications Server or later, to the VIPAROUTE target IP address, which is a dedicated static VIPA for MNLB usage.

The VIPAROUTE statement influences the IP address that is returned to Cisco routers when Sysplex Distributor is configured as a service manager for the Cisco Multi-Node Load Balancing (MNLB) function. In Example 5-26 we define three VIPAROUTE statements, pointing to the respective static VIPA in our LPARs SC30, SC31, and SC32.

Note: VIPAROUTE is only supported and used when running z/OS V1R7 or later. If your z/OS operating system does not support VIPAROUTE yet, the XCF addresses are used to identify the target stacks.

Example 5-26 VIPAROUTE: Define the XCF address and the respective IP interface

```
VIPAROUTE 10.30.20.100 10.30.1.231
VIPAROUTE 10.30.20.101 10.30.1.242
VIPAROUTE 10.30.20.102 10.30.1.222
```

Configuration example

Example 5-27 lists a part of our TCP/IP profile from our distribution stack SC30, containing the parameter and statements that are important for our SD/MNLB Service Manager installation.

Example 5-27 TCP/IP profile of our distribution stack SC30

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
DYNAMICXCF 10.30.20.100 255.255.255.0 8
;Osa definitions
DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 30
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 31
 ;
 ;HiperSockets definitions dynamically created on vtam
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6LNK  IPAQIDIO      IUTIQDF6
;
; Static VIPA definitions
  DEVICE STAVIPA1    VIRTUAL 0
  LINK   STAVIPA1LNK VIRTUAL 0     STAVIPA1
 ;
 ; Static VIPA used for mnlb
  DEVICE STAVIPA2    VIRTUAL 0
  LINK   STAVIPA2LNK VIRTUAL 0     STAVIPA2
 ;
 VIPADynamic
    VIPADEFine MOVEable IMMEDiate SERVICEMGR 255.255.255.0 10.30.30.33
    VIPASMPARMS SMMCAST 224.0.1.2 SMPORT 1637
    VIPADISTribute DISTMethod SERVERWLM 10.30.30.33
     PORT 23 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
    VIPAROUTE 10.30.20.100 10.30.1.231
    VIPAROUTE 10.30.20.101 10.30.1.242
    VIPAROUTE 10.30.20.102 10.30.1.222
 ENDVIPADYNAMIC
 ;
 HOME
   10.30.1.231   STAVIPA2LNK
   10.30.1.230   STAVIPA1LNK
   10.30.2.232   OSA2080LNK
   10.30.3.235   OSA20E0LNK
   10.30.5.236   IUTIQDF6LNK
;
PRIMARYINTERFACE STAVIPA1LNK
;
 START IUTIQDF6
 START OSA2080
 START OSA20E0
```

Verification

We used the following steps to verify our SD/MNLB implementation for the TN3270 application:

1. Verify the TCP/IP configuration of the distributing stack.
2. Verify the configuration of the Cisco router.
3. Verify that our TN3270 connections are being distributed using SD/MNLB.
4. Verify that our VIPAROUTE definition works.

To verify the TCP/IP configuration of the distributing stack we used a NETSTAT,VIPADCFG command on SC30, as shown in Example 5-28. This provided key information to help us verify our SD/MNLB configuration.

Example 5-28 Display NETSTAT,VIPADCFG on our distribution LPAR SC30

```
D TCPIP,TCPIPC,N,VIPADCFG
EZD0101I NETSTAT CS V1R7 TCPIPC 035
DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
   IPADDR/PREFIXLEN: 10.30.30.33/24
      MOVEABLE: IMMEDIATE  SRVMGR: YES 1

VIPA DISTRIBUTE:
    DEST:         10.30.30.33..23
      DESTXCF:    10.30.20.100
        SYSPT:    NO   TIMAFF: NO    FLG: SERVERWLM  2
    DEST:         10.30.30.33..23
       DESTXCF:    10.30.20.101
         SYSPT:    NO   TIMAFF: NO    FLG: SERVERWLM 2
     DEST:         10.30.30.33..23
       DESTXCF:    10.30.20.102
         SYSPT:    NO   TIMAFF: NO    FLG: SERVERWLM 2

   VIPA SERVICE MANAGER:
     MCASTGROUP: 224.0.1.2 3
     PORT: 01637 4 PWD: NO
   VIPA ROUTE: 5
     DESTXCF:     10.30.20.100
       TARGETIP:  10.30.1.231
     DESTXCF:     10.30.20.101
       TARGETIP:  10.30.1.242
     DESTXCF:     10.30.20.102
       TARGETIP:  10.30.1.222
 END OF THE REPORT
```

Important fields are:

- 1 SRVMGR: YES indicates that the Cisco Service manager is active and will be used.
- 2 FLG: SERVERWLM indicates that server-specific WLM is active and will be used.
- 3 Multicast group used in our MNLB environment.
- 4 UDP port used in our MNLB environment.
- 5 VIPAROUTE indicates that we do not use XCF links, but any available IP network interface for forwarding SD-related packets.

To verify the correctness of the Cisco router MNLB configuration, we used four SHOW IP,CASA commands, as illustrated in Example 5-29, Example 5-30, Example 5-31 on page 129, and Example 5-32 on page 129.

Example 5-29 Display SHOW IP CASA OPER

```
Router# show ip casa oper
CASA Active 1
  Casa control address is 8.8.8.8/32 2
  Casa multicast address is 224.0.1.2 3
  Listening on ports:
    Forwarding Agent Listen Port: 1637 4
      Current passwd: <none> Pending passwd: <none>
      Passwd timeout: 180 sec (Default)

Router#
```

This command provided the following data:

- 1 indicates that CASA is operational in our Cisco router.
- 2 CASA control address, used for unicasts to the Service Manager and vice versa.
- 3 indicates the multicast group address.
- 4 indicates the UDP port number used.

The next command, in Example 5-30, checks that the Service Manager propagates wildcard affinities.

Example 5-30 Display SHOW IP CASA AFFINITIES

```
Router# show ip casa affinities

Source Address  Port  Dest Address    Port  Prot
9.12.4.222      1906  10.30.30.33     23    TCP
9.12.4.222      1904  10.30.30.33     23    TCP
9.12.4.222      1905  10.30.30.33     23    TCP
9.12.4.222      1898  10.30.30.33     23    TCP
9.12.4.222      1897  10.30.30.33     23    TCP
Router#
```

The next command, in Example 5-31, shows the wildcard affinities multicasted by the Service Manager to the Forwarding Agents.

Example 5-31 Display SHOW IP CASA WILDCARD

```
Router# show ip casa wildcard

Source Address  Source Mask     Port  Dest Address    Dest Mask       Port  Prot
0.0.0.0         0.0.0.0         0     10.30.30.33  255.255.255.255  23    TCP
Router#
```

The source IP address of 0.0.0.0 allows the Forwarding Agent to accept any incoming IP packet with any port number of the protocol TCP. The destination IP address represents our cluster address for the TN3270 application service followed by the corresponding port number. This wildcard affinity block is sent to the Forwarding Agents from the Service Manager every 30 seconds over each interface of the Sysplex Distributor.

We use the SHOW CASA IP AFFINITIES DETAIL command to check if our TN3270 connections are being distributed as wanted. Example 5-32 shows an excerpt of this command output, including detailed information about our TN3270 connections.

Example 5-32 Display SHOW CASA IP AFFINITIES DETAIL

```
Router# show ip casa affinities detail

Source Address  Port  Dest Address    Port  Prot
9.12.4.222      1971  10.30.30.33     23    TCP
  Action Details:
    Interest Addr:          10.30.20.100 1 Interest Port: 1637 2
    Interest Packet: 0x0002 SYN
    Interest Tickle: 0x0000
    Dispatch (Layer 2):     YES             Dispatch Address: 10.30.1.242 3

Source Address  Port  Dest Address    Port  Prot
9.12.4.222      1974  10.30.30.33     23    TCP
  Action Details:
    Interest Addr:          10.30.20.100 1 Interest Port: 1637 2
    Interest Packet: 0x0002 SYN
    Interest Tickle: 0x0000
    Dispatch (Layer 2):     YES             Dispatch Address: 10.30.1.242 3
```

Key elements here include:

- 1 The interest address is the IP address of the Sysplex Distributor's XCF link address.
- 2 The port 1637 is the CASA listening port of the Sysplex Distributor.
- 3 The dispatch address is the IP address for a target server running a TN3270 instance.

We then needed to verify that our VIPAROUTE definition worked correctly. In our SD/MNLB design only the TCP SYN packets are potential candidates that could benefit from using VIPAROUTE. We defined VIPAROUTE in our SD/MNLB implementation as well, relieving the XCF links of forwarding SD TCP SYN packets. We verified that by checking the dynamic VIPA connection routing table, as shown in Example 5-33.

Example 5-33 Display of the VCRT at SC30

```
D TCPIP,TCPIPC,N,VCRT,DETAIL
EZD0101I NETSTAT CS V1R7 TCPIPC 800
DYNAMIC VIPA CONNECTION ROUTING TABLE:
DEST:      10.30.30.33..23
  SOURCE:  9.12.4.222..1971
  DESTXCF: 10.30.20.101
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
    INTF:  IUTIQDF6LNK
      VIPAROUTE:  YES      GW:  10.30.5.246
DEST:      10.30.30.33..23
  SOURCE:  9.12.4.222..1974
  DESTXCF: 10.30.20.101
    POLICYRULE:    *NONE*
    POLICYACTION:  *NONE*
    INTF:  IUTIQDF6LNK
      VIPAROUTE:  YES      GW:  10.30.5.246
   2 OF 2 RECORDS DISPLAYED
 END OF THE REPORT
```

Both TN3270 connections were distributed via the HiperSockets interface and not via an XCF link, which is what we wanted.

5.4.3 Multiple servers sharing a port using SHAREPORTWLM

If we need to support a large number of client connections on a single system then having multiple servers sharing the same port could help with performance and reliability. With SHAREPORTWLM we can run more than one instance of an application without changing the application. (This assumes that the basic design of the application allows multiple instances to be used.) To use this function, each instance of the server application binds to the same server IP address and port. The SHAREPORTWLM keyword in the PORT profile statements indicates that multiple application instances may 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*.

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 is our goal to distribute incoming TN3270 connections among our five TN3270 servers, based on the individual server weight (server-specific WLM). This is a combined use of the Sysplex Distributor and the shareport function, and is illustrated in Figure 5-22 on page 131.

Note: SHAREPORTWLM does not work with the internal Telnet server INTCLIENT. Therefore we set up our TN3270 server to run in a separate address space, which is supported with z/OS V1R6 and later.

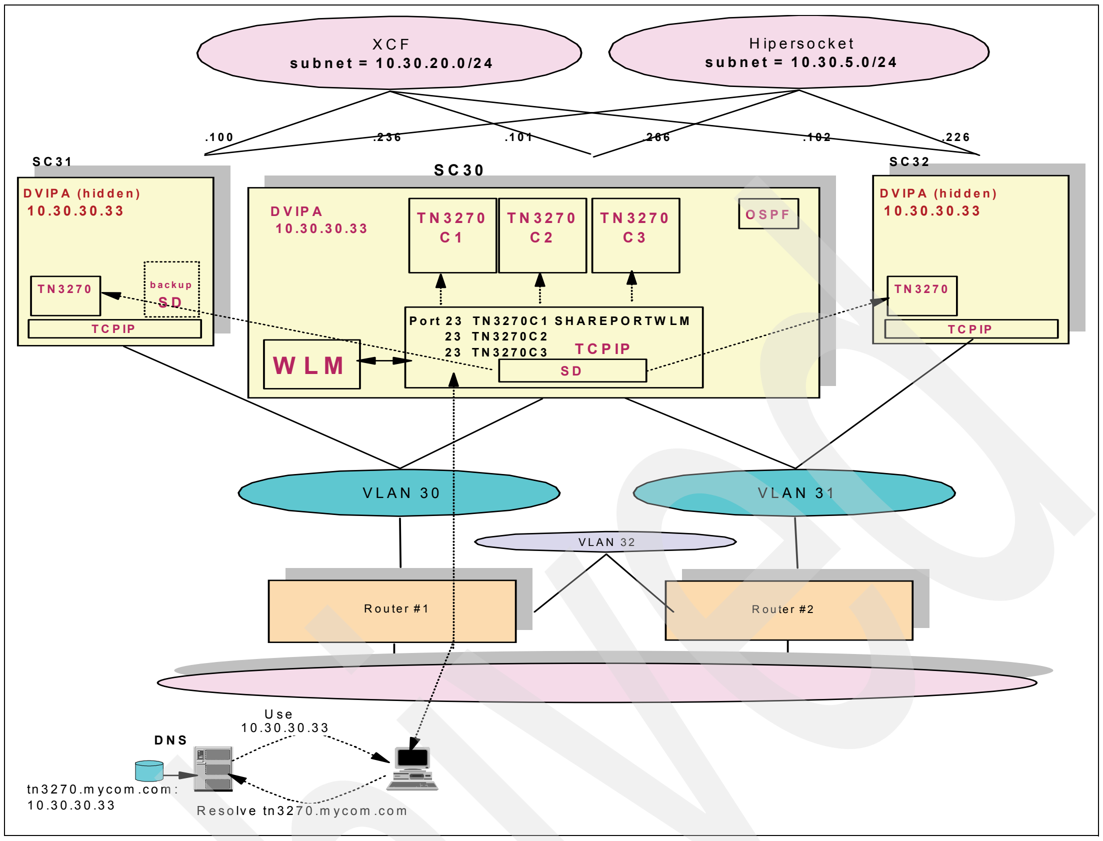


Figure 5-22 Three TN3270 servers building a shareport group on system SC30

Implementation tasks

The following list shows what is needed 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 in Example 5-34, TCP/IP allows our three TN3270 servers to listen on the same combination of port and interface.

Note: We could specify SHAREPORT instead of the new SHAREPORTWLM. With SHAREPORT, incoming connection requests would be distributed in a weighted round-robin fashion. We choose SHAREPORTWLM, which is new with z/OS V1R7.0 Communications Server, because we wanted to use the more accurate server-specific WLM weights.

Example 5-34 Define the TN3270 shareport group on SC30

```
PORT
    20 TCP *  NOAUTOLOG          ; FTP Server
    21 TCP OMVS                  ; control port
    23 TCP TN3270C1 SHAREPORTWLM      ; MVS Telnet Server sep.addrspace
    23 TCP TN3270C2                  ; MVS Telnet Server sep.addrspace
    23 TCP TN3270C3                  ; MVS Telnet Server sep.addrspace
```

Example 5-35 lists 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 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 5-35   Define a separate TN3270 profile member: TNSC30C1

```
TelnetGlobals
  TCPIPJOBNAME TCPIPC 2
 EndTelnetGlobals
 ;
 TelnetParms
   Port 23                        ; Port number 23 (std.)
   TELNETDEVICE 3278-3-E NSX32703 ; 32 line screen -
                                  ; default of NSX32702 is 24
   TELNETDEVICE 3279-3-E NSX32703 ; 32 line screen -
                                  ; default of NSX32702 is 24
   TELNETDEVICE 3278-4-E NSX32704 ; 48 line screen -
                                  ; default of NSX32702 is 24
   TELNETDEVICE 3279-4-E NSX32704 ; 48 line screen -
                                  ; default of NSX32702 is 24
   TELNETDEVICE 3278-5-E NSX32705 ; 132 column screen-
                                  ; default of NSX32702 is 80
   TELNETDEVICE 3279-5-E NSX32705 ; 132 column screen -
                                  ; default of NSX32702 is 80
  LUSESSIONPEND     ; On termination of a Telnet server connection,
                    ; the user will revert to the DEFAULTAPPL
                    ; instead of having the connection dropped

  MSG07             ; Sends a USS error message to the client if an
                    ; error occurs during session establishment
                    ; instead of dropping the connection
  CodePage ISO8859-1 IBM-1047   ; Linemode ASCII, EBCDIC code pages
  Inactive 0                    ; Let connections stay around
  PrtInactive 0                 ; Let connections stay around
  TimeMark 600
  ScanInterval 120
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
               ; applications.
              ; TSO is multiple applications all beginning with TSO,
              ; so use the * to get them all.  If a session is closed,
            ; disconnect the user rather than log off the user.

  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 shown in Example 5-36, Example 5-37, and Example 5-38.

Example 5-36 Define a TN3270 procedure: TN3270C1

```
SYS1.PROCLIB(TN3270C1) - 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=TCPIPC.TCPPARMS(TNSC30C1)
//SYSTCPD  DD DSN=TCPIPC.TCPPARMS(DATAC&SYSCLONE),DISP=SHR
=======================================================================
```

Example 5-37 Define a TN3270 procedure: TN3270C2

```
SYS1.PROCLIB(TN3270C2) - 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=TCPIPC.TCPPARMS(TNSC30C2)
//SYSTCPD  DD DSN=TCPIPC.TCPPARMS(DATAC&SYSCLONE),DISP=SHR
=======================================================================
```

Example 5-38 Define a TN3270 procedure: TN3270C3

```
SYS1.PROCLIB(TN3270C3) - 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=TCPIPC.TCPPARMS(TNSC30C3)
//SYSTCPD  DD DSN=TCPIPC.TCPPARMS(DATAC&SYSCLONE),DISP=SHR
=======================================================================
```

We started our three TN3270 server instances within the same LPAR, as shown in Example 5-39 on page 134, Example 5-40 on page 134, and Example 5-41 on page 134.

Example 5-39 Start TN3270C1

```
S TN3270C1
$HASP100 TN3270C1 ON STCINRDR
IEF695I START TN3270C1 WITH JOBNAME TN3270C1 IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 TN3270C1 STARTED
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
EZZ6001I TELNET SERVER STARTED
EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 538
            TCPIPC.TCPPARMS(TNSC30C1)
EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 539
            TCPIPC.TCPPARMS(TNSC30C1)
EZZ6003I TELNET LISTENING ON PORT    23
```

Example 5-40 Start TN3270C2

```
S TN3270C2
$HASP100 TN3270C2 ON STCINRDR
IEF695I START TN3270C2 WITH JOBNAME TN3270C2 IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 TN3270C2 STARTED
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
EZZ6001I TELNET SERVER STARTED
EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 547
            TCPIPC.TCPPARMS(TNSC30C2)
EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 548
            TCPIPC.TCPPARMS(TNSC30C2)
EZZ6003I TELNET LISTENING ON PORT    23
```

Example 5-41 Start TN3270C3

```
S TN3270C3
$HASP100 TN3270C3 ON STCINRDR
IEF695I START TN3270C3 WITH JOBNAME TN3270C3 IS ASSIGNED TO USER
TCPIP   , GROUP TCPGRP
$HASP373 TN3270C3 STARTED
IEE252I MEMBER CTIEZBTN FOUND IN SYS1.IBM.PARMLIB
EZZ6001I TELNET SERVER STARTED
EZZ6044I TELNET PROFILE PROCESSING BEGINNING FOR FILE 556
            TCPIPC.TCPPARMS(TNSC30C3)
EZZ6045I TELNET PROFILE PROCESSING COMPLETE FOR FILE 557
            TCPIPC.TCPPARMS(TNSC30C3)
EZZ6003I TELNET LISTENING ON PORT    23
```

Configuration example

Example 5-42 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 5-42 TCP/IP profile of our distribution stack SC30

```
ARPAGE 20
GLOBALCONFIG NOTCPIPSTATISTICS
IPCONFIG NODATAGRAMFWD SYSPLEXROUTING
IPCONFIG MULTIPATH PERCONNECTION
IPCONFIG SOURCEVIPA
;
DYNAMICXCF 10.30.20.100 255.255.255.0 8
;
 DEVICE OSA2080  MPCIPA
 LINK   OSA2080LNK  IPAQENET      OSA2080 VLANID 30
 DEVICE OSA20E0  MPCIPA
 LINK   OSA20E0LNK  IPAQENET      OSA20E0 VLANID 31
 ;
 DEVICE IUTIQDF6 MPCIPA
 LINK   IUTIQDF6LNK  IPAQIDIO      IUTIQDF6
;
 DEVICE STAVIPA1    VIRTUAL 0
 LINK   STAVIPA1LNK VIRTUAL 0     STAVIPA1
 ;
  DEVICE STAVIPA2    VIRTUAL 0
  LINK   STAVIPA2LNK VIRTUAL 0     STAVIPA2
 ;
 VIPADynamic
    VIPADEFine MOVEable IMMEDiate 255.255.255.0 10.30.30.33
    VIPADISTribute DISTMethod SERVERWLM 10.30.30.33
     PORT 23 DESTIP 10.30.20.100 10.30.20.101 10.30.20.102
    VIPAROUTE 10.30.20.100 10.30.1.230
    VIPAROUTE 10.30.20.101 10.30.1.241
    VIPAROUTE 10.30.20.102 10.30.1.221
 ENDVIPADYNAMIC
 ;
 HOME
    10.30.1.231   STAVIPA2LNK
    10.30.1.230   STAVIPA1LNK
    10.30.2.232   OSA2080LNK
    10.30.3.235   OSA20E0LNK
    10.30.5.236   IUTIQDF6LNK
 ;
 PRIMARYINTERFACE STAVIPA1LNK
 ;
 AUTOLOG 5
    FTPDC  JOBNAME FTPDC1
    OMPC
 ENDAUTOLOG
 ;
 PORT
     20 TCP *  NOAUTOLOG         ; FTP Server
     21 TCP OMVS                 ; control port
     23 TCP TN3270C1 SHAREPORTWLM ; MVS Telnet Server sep.addrspace
     23 TCP TN3270C2                 ; MVS Telnet Server sep.addrspace
     23 TCP TN3270C3                 ; MVS Telnet Server sep.addrspace
 ;
  START IUTIQDF6
  START OSA2080
   START OSA20E0
```

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 5-43.

Example 5-43 tDisplay NETSTAT,CONN on SC30

```
D TCPIP,TCPIPC,N,CONN
```

```
 EZD0101I NETSTAT CS V1R7 TCPIPC 560
 USER ID  CONN     STATE

TN3270C1 0000003C LISTEN 1
  LOCAL SOCKET:   ::..23 2
  FOREIGN SOCKET: ::..0
TN3270C2 00000041 LISTEN 1
  LOCAL SOCKET:   ::..23 2
  FOREIGN SOCKET: ::..0
TN3270C3 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 5-44.

Example 5-44 Display NETSTAT,VIPADCFG on SC30

```
D TCPIP,TCPIPC,N,VIPADCFG
EZD0101I NETSTAT CS V1R7 TCPIPC 403
DYNAMIC VIPA INFORMATION:
  VIPA DEFINE:
    IPADDR/PREFIXLEN: 10.30.30.33/24
       MOVEABLE: IMMEDIATE  SRVMGR: NO
VIPA DISTRIBUTE:
    DEST:        10.30.30.33..23
      DESTXCF:   10.30.20.100
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM  1
    DEST:        10.30.30.33..23
      DESTXCF:   10.30.20.101
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM  1
    DEST:        10.30.30.33..23
      DESTXCF:   10.30.20.102
        SYSPT:   NO   TIMAFF: NO    FLG: SERVERWLM  1
  VIPA ROUTE:
    DESTXCF:     10.30.20.100
      TARGETIP:  10.30.1.230
    DESTXCF:     10.30.20.101
      TARGETIP:  10.30.1.241
    DESTXCF:     10.30.20.102
      TARGETIP:  10.30.1.221
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 5-45 on page 137.

Example 5-45 Display NETSTAT,VDPT on SC30

```
D TCPIP,TCPIPC,N,VDPT
 EZD0101I NETSTAT CS V1R7 TCPIPC 146
 DYNAMIC VIPA DESTINATION PORT TABLE:
 DEST:        10.30.30.33..23
   DESTXCF:   10.30.20.100
   TOTALCONN: 0000000013 1 RDY: 003 2 WLM: 05  TSR: 070
   FLG: SERVERWLM
 DEST:        10.30.30.33..23
   DESTXCF:   10.30.20.101
   TOTALCONN: 0000000007 1 RDY: 001 2 WLM: 08  TSR: 100
   FLG: SERVERWLM
 DEST:        10.30.30.33..23
   DESTXCF:   10.30.20.102
   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, which RDY: 003 indicates. 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 5-46, Example 5-47 on page 138, and Example 5-48 on page 138.

Example 5-46 Display TELNET,CONN for our first TN3270 server TN3270C1 on SC30

```
D TCPIP,TN3270C1,T,CONN
 EZZ6064I TELNET CONNECTION DISPLAY 234
          EN                                      TSP
 CONN     TY IPADDR..PORT           LUNAME   APPLID   PTR LOGMODE
 -------- -- ---------------------- -------- -------- --- --------
 0000005C    ::FFFF:9.12.4.222..2683
                                   TCP30307           TPE
 0000005A    ::FFFF:9.12.4.222..2682
                                   TCP30306           TPE
 00000057    ::FFFF:9.12.4.222..2681
                                   TCP30305           TPE
 0000004B    ::FFFF:9.12.4.222..2677
                                   TCP30304           TPE
 00000045    ::FFFF:9.12.4.222..2676
                                   TCP30303           TPE
 00000043    ::FFFF:9.12.4.222..2675
                                   TCP30302           TPE
 00000041    ::FFFF:9.12.4.222..2674
                                   TCP30301           TPE
 ----- PORT:    23  ACTIVE              PROF: CURR CONNS:      7
 ------------------------------------------------------------
 16 OF 16 RECORDS DISPLAYED
```

Example 5-47 Display TELNET,CONN for our second TN3270 server TN3270C2 on SC30

```
D TCPIP,TN3270C2,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 342
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- ---------------------- -------- --------  --- --------
00000052    ::FFFF:9.12.4.222..2680
                                   TCP30313          TPE
00000050    ::FFFF:9.12.4.222..2679
                                   TCP30312          TPE
0000004D    ::FFFF:9.12.4.222..2678
                                   TCP30311          TPE
----- PORT:    23  ACTIVE               PROF: CURR CONNS:      3
------------------------------------------------------------
8 OF 8 RECORDS DISPLAYED
```

Example 5-48 Display TELNET,CONN for our third TN3270 server TN3270C3 on SC30

```
D TCPIP,TN3270C3,T,CONN
EZZ6064I TELNET CONNECTION DISPLAY 392
         EN                                          TSP
CONN     TY IPADDR..PORT            LUNAME   APPLID   PTR LOGMODE
-------- -- ---------------------- -------- --------  --- --------
00000065    ::FFFF:9.12.4.222..2686
                                   TCP30323          TPE
00000063    ::FFFF:9.12.4.222..2685
                                   TCP30322          TPE
00000061    ::FFFF:9.12.4.222..2684
                                   TCP30321          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.

5.4.4 Problem determination

We found that drawing out a configuration, including all the IP addresses for all the paths involved, was very useful. NETSTAT commands (including the operator D TCPIP,xxx commands that implicitly use NETSTAT commands) provided almost all of the problem determination functions that we needed. The examples in this chapter include many of these NETSTAT commands. We strongly recommend that anyone implementing these types of configurations should take the time to become familiar with many of the NETSTAT options.

In addition, we recommend using *Communications Server V1R7: IP Configuration Reference,* SC31-8776, for additional information.

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 lots 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) Uniform Resource Locators (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 done by implementing the new z/OS Load Balancing Advisor (LBA) solution that uses the Server/Application State Protocol (SASP).

The combination of z/OS sysplex and external load balancing provides a very strong architecture for the base networking infrastructure and is prepared for future growth demands.

This chapter discusses the following.

Section	Topic
6.1, “Basic concepts” on page 140	Discusses the basic concepts of external application workload balancing
6.2, “Importance of external application workload balancing” on page 147	Discusses key characteristics of external application workload balancing and why it may be important in your environment
6.3, “Design examples” on page 148	Presents commonly implemented external application workload balancing design scenarios, their dependencies, advantages, considerations, and our recommendations
6.4, “Implementations” on page 150	Presents selected implementation scenarios, tasks, configuration examples, and problem determination suggestions

6.1 Basic concepts

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 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 Cisco Workload Distribution Technologies with the use of Cisco Catalyst 6500 Series Switch and Cisco Content Switching Module (CSM) as the load balancing device. The CSM is one of the external load balancers that supports SASP.

Understanding directed mode load balancing

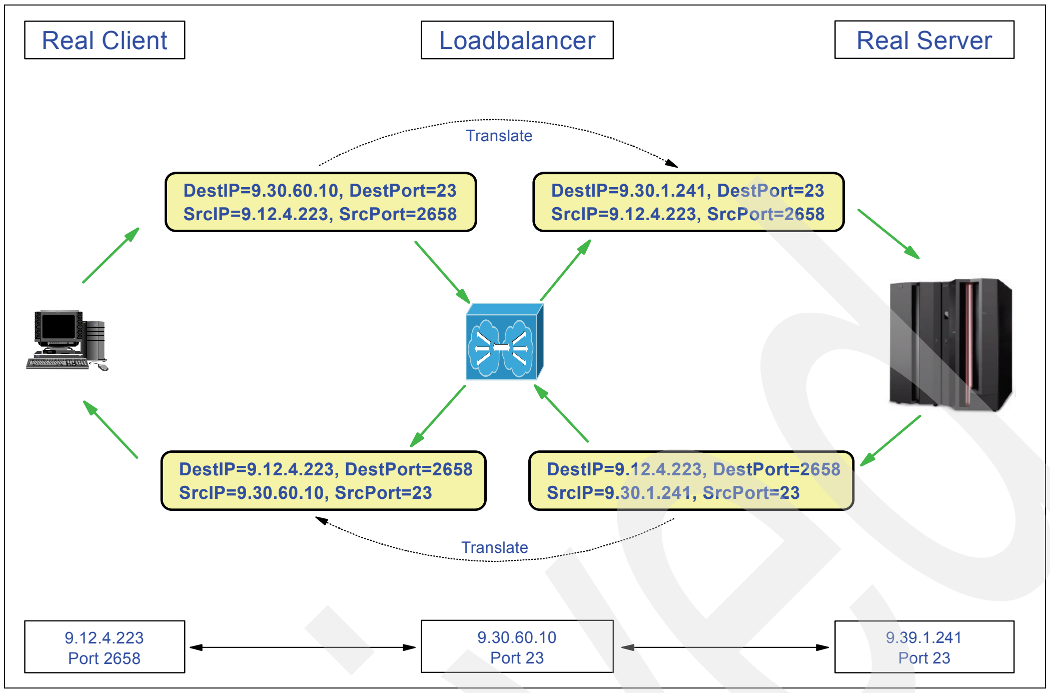
Directed mode load balancing works by changing the destination IP address of the inbound IP packets to the IP address of the chosen target stack. One of the advantages of directed mode is that there can be any number of router hops in between the load balancer and the target stacks.

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 since 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 via 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 that outbound routing back through the load balancer occurs:

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 via the load balancer (via 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 via the load balancer, while outbound packets for workload that were not balanced are routed directly back to the clients.

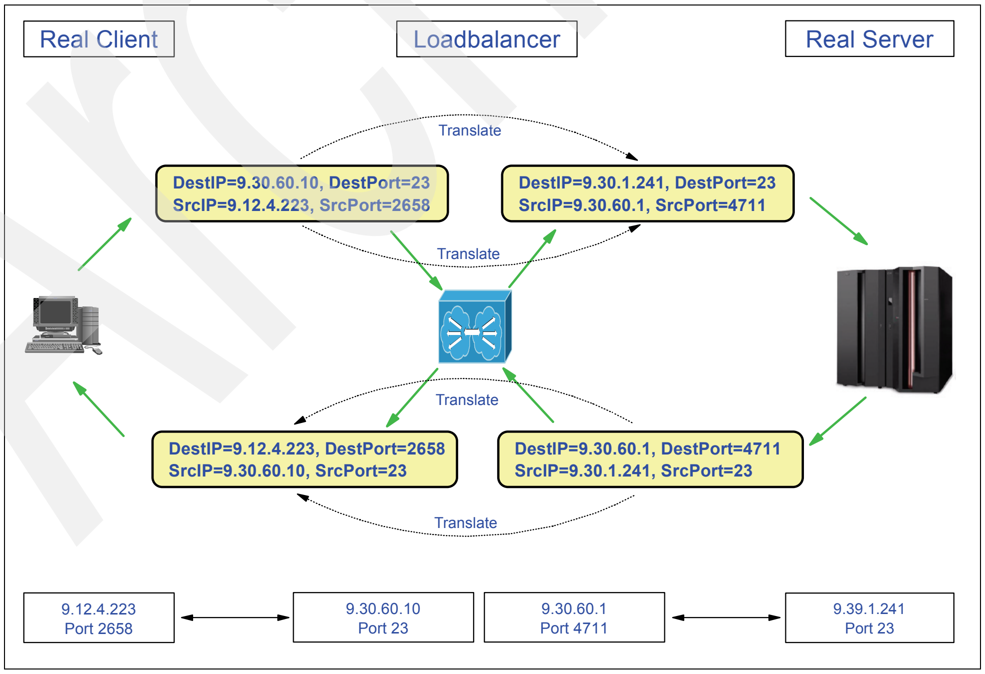
   This is known as server NAT operation and is illustrated in Figure 6-1 on page 141.



Real Client
Loadbalancer
Real Server
Translate
DestIP=9.30.60.10, DestPort=23
SrcIP=9.12.4.223, SrcPort=2658
DestIP=9.30.1.241, DestPort=23
SrcIP=9.12.4.223, SrcPort=2658
DestIP=9.12.4.223, DestPort=2658
SrcIP=9.30.60.10, SrcPort=23
DestIP=9.12.4.223, DestPort=2658
SrcIP=9.30.1.241, SrcPort=23
Translate
9.12.4.223
Port 2658
9.30.60.10
Port 23
9.39.1.241
Port 23

Figure 6-1 Server NAT with policy-based routing

2. The load balancer may be configured to change both the destination IP address and the source IP address in the inbound IP packets. This is known as server NAT and client NAT. The advantage of this 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.



Real Client
Loadbalancer
Real Server
Translate
DestIP=9.30.60.10, DestPort=23
SrcIP=9.12.4.223, SrcPort=2658
DestIP=9.30.1.241, DestPort=23
SrcIP=9.30.60.1, SrcPort=4711
Translate
Translate
DestIP=9.12.4.223, DestPort=2658
SrcIP=9.30.60.10, SrcPort=23
DestIP=9.30.60.1, DestPort=4711
SrcIP=9.30.1.241, SrcPort=23
Translate
9.12.4.223
Port 2658
9.30.60.10
Port 23
9.30.60.1
Port 4711
9.39.1.241
Port 23

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 may complicate diagnosing network-related problems.

Client NATing may have an impact on z/OS networking policy functions. Networking policies on z/OS may apply network QoS differentiated services, selection of Intrusion Detection Service (IDS) actions, and Traffic Regulation (TR) limitations. z/OS networking policies are specified via 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 NETACCESS rules. These 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 may 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, since it may 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.

6.1.1 z/OS Load Balancer Advisor

The z/OS Load Balancing Advisor is a z/OS Communications Server technology made available in 4Q2004 via APARs PQ90032 (V1R4) andPQ96293 (V1R5/V1R6). It is fully integrated into z/OS V1R7. The z/OS Load Balancer Advisor (LBA) 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 LBA flow is illustrated in Figure 6-3 on page 143.

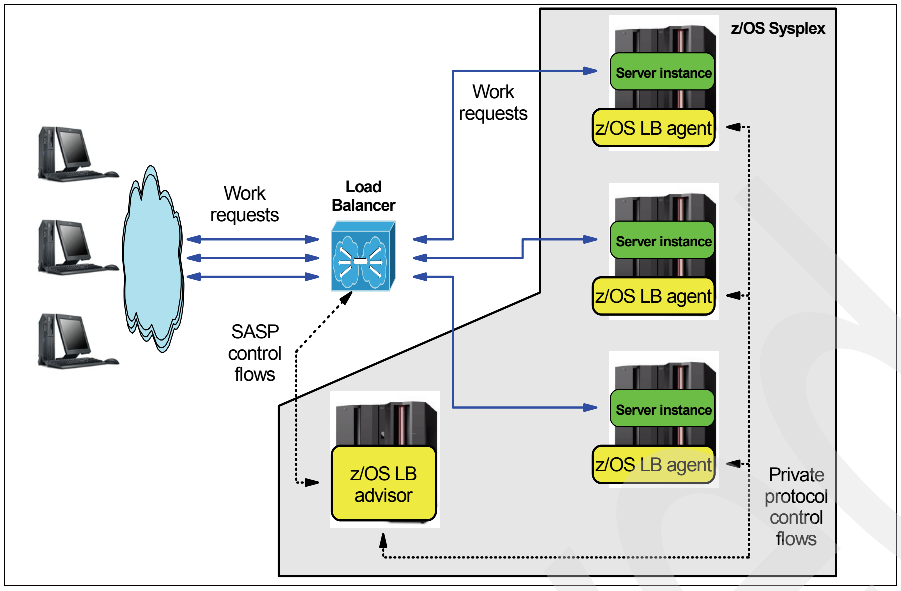


Figure 6-3 z/OS Load Balancing Advisor overview

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 via 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 will 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 Communication 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

  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 (CS) weight

  The CS 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 CS weight has a numeric value between 0 and 100.

  There are several purposes of the calculation of the CS weight. One purpose is to prevent a stalled application instance from being sent more work. Another purpose is to proactively react to an application instance that is becoming overloaded. It may 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 CS weights into a single metric:

```
Final weight = (WLM weight * CS weight) / 100
```

6.1.2 Server/Application State Protocol (SASP)

The Server/Application State Protocol (SASP) is an open protocol currently described in an individual Internet draft RFC and submitted to the IETF: Alan Bivens, Sever/Application State Protocol version 1 (SASPv1), *Individual Internet Draft*, August 2005 (work in progress):

http://www.ietf.org/internet-drafts/draft-bivens-sasp-02.txt

The SASP protocol is used for communication between external load balancers and the advisor. Information 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

  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 LPAR. The CS weight indicates whether the IP address is active in the sysplex. A value of 0 means quiesced and 100 means not quiesced. LBA 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. CS weights indicate how well the server instances are performing. LBA 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

  With this 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

  With this 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 LBA and agents, and LBA and external load balancers, as well as cycles used in LBA and load balancers.

6.1.3 External load balancer without LBA/SASP

There are several products available today that do load balancing of IP traffic. The example in this chapter was built using the Cisco Content Switching Module (CSM) and has the general structure shown in Figure 6-4 on page 146.

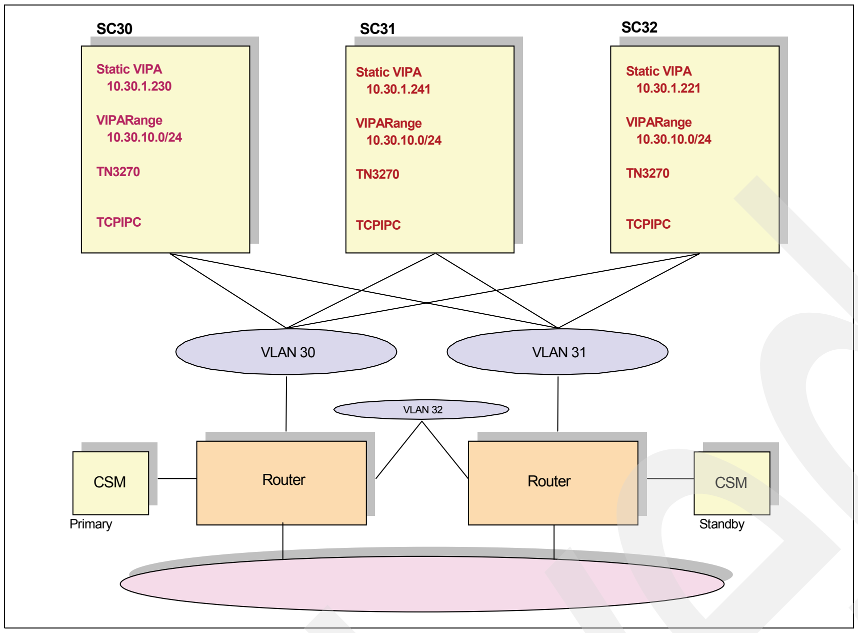


Figure 6-4 External load balancing without LBA/SASP

The external load balancer function is implemented as a active/standby pair for fault tolerance. The CSMs are used to exchange state information so that existing connections continue nondisruptively in case of a failure in the active CSM. The external load balancer receives end-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 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.

6.1.4 External load balancer with LBA/SASP

The example for this mode was built using the Cisco Content Switching Module (CSM), which is a product that supports SASP. The general structure of the example is shown in Figure 6-5 on page 147.

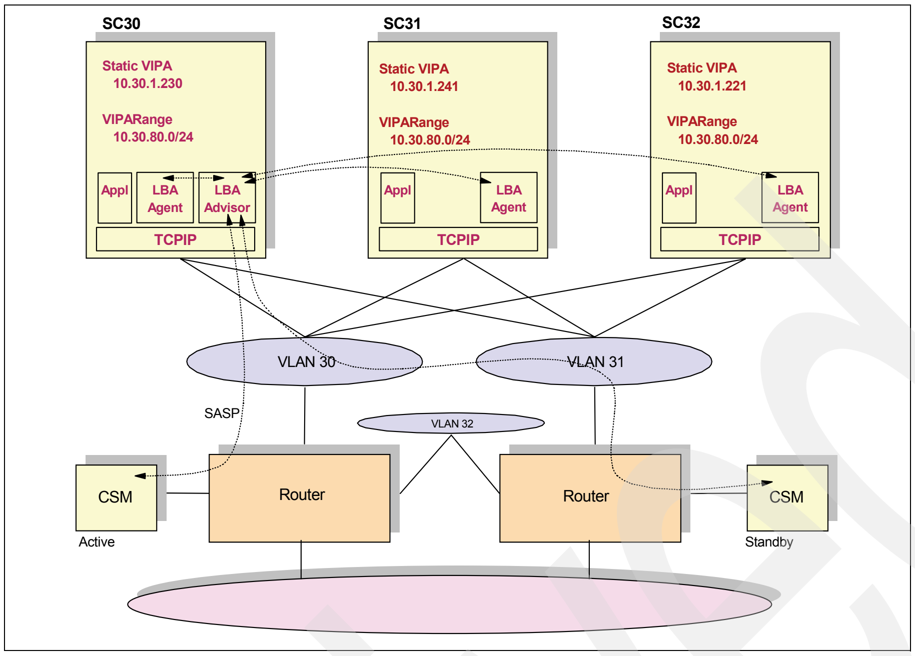


Figure 6-5 External load balancing with LBA/SASP

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 (LBA) solution that communicates the recommendations to the external load balancer via the new SASP protocol. The load balancer must support the new SASP protocol and include the recommendations from LBA when it decides which application instance to use for the next request. The external load balancer function is implemented as a active/standby pair for fault tolerance. The CSMs exchange state information so that existing connections continue nondisruptively in case of a failure in the active CSM.

The external load balancer receives end-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 may combine the recommendations from LBA with its own techniques, such as ICMP polling.

6.2 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:

- We may want 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.
- There may be a requirement of content-based load balancing using higher-layer information buried deep inside of datagrams rather than just using IP addresses—including application-specific information such as an HTTP (or FTP) Uniform Resource Locators (URLs).
- We may want to minimize processor usage on our System z9 and zSeries mainframes.

External workload balancing offers a different dimension to the z/OS sysplex environment, especially when the load balancer becomes z/OS sysplex aware. It may 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 Cisco Content Switching.

6.3 Design examples

We describe two examples in the remainder of this chapter. These examples are:

- External load balancer without LBA/SASP
- External load balancer with LBA/SASP

6.3.1 External load balancer without LBA/SASP

This example describes the external load balancing of three application instances of TN3270 in a z/OS sysplex environment without LBA/SASP.

The infrastructure eliminates single network point-of-failures by using the following:

- Two OSA-Express Gigabit Ethernet adapters (one port used per physical OSA Card)
- Two Cisco Catalyst 6500 Series Switches
- Two CSMs with stateful failover
- OSPF to ensure rerouting around failures

Our example uses a single server, which presents a single point-of-failure. A real production environment might use multiple servers. Our general structure is shown in Figure 6-6.

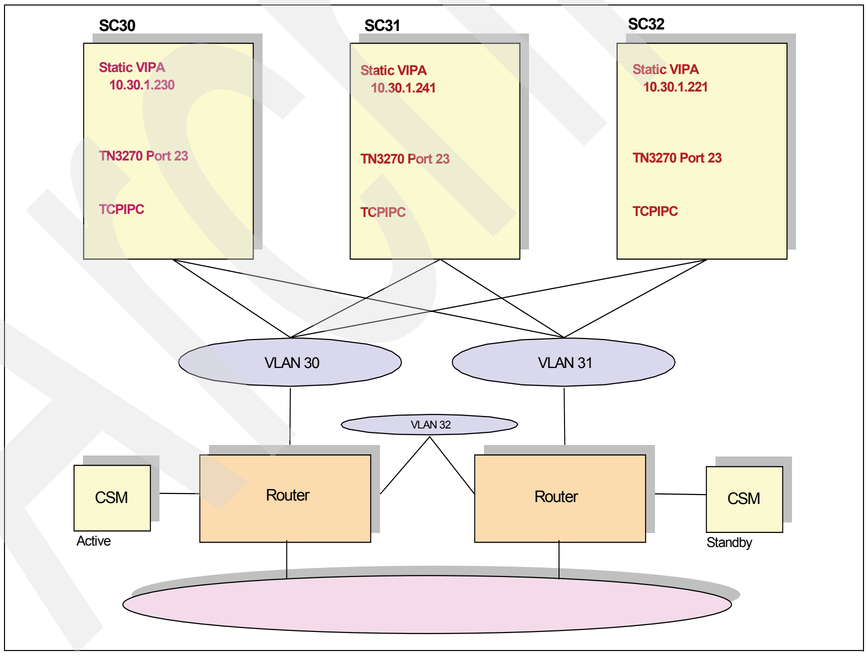


Figure 6-6 External load balancing of TN3270 without LBA/SASP

Environment

Our specific environment contains the following:

- Three LPARS in the same server

- 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 Cisco Catalyst 6500 Series Switches
- Two Cisco Content Switching Modules (CSM) in active/standby mode

A static VIPA is used to address each instance of a TN3270 server. The CSM can sense the presence of the VIPA check whether the TN3270 server is available.

Dependencies

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 points of failure exist in the network elements of this design.

Considerations

If application instances are intended to move between LPARs we should use a dynamic VIPA. We did not include this function in out example.

6.3.2 External load balancer with LBA/SASP

Our second example has external load balancing for three application instances of TN3270 servers in a z/OS sysplex environment using LBA/SASP. It has the same advantages as the previous example for eliminating single points of failure in the network elements. The general design is shown in Figure 6-7.

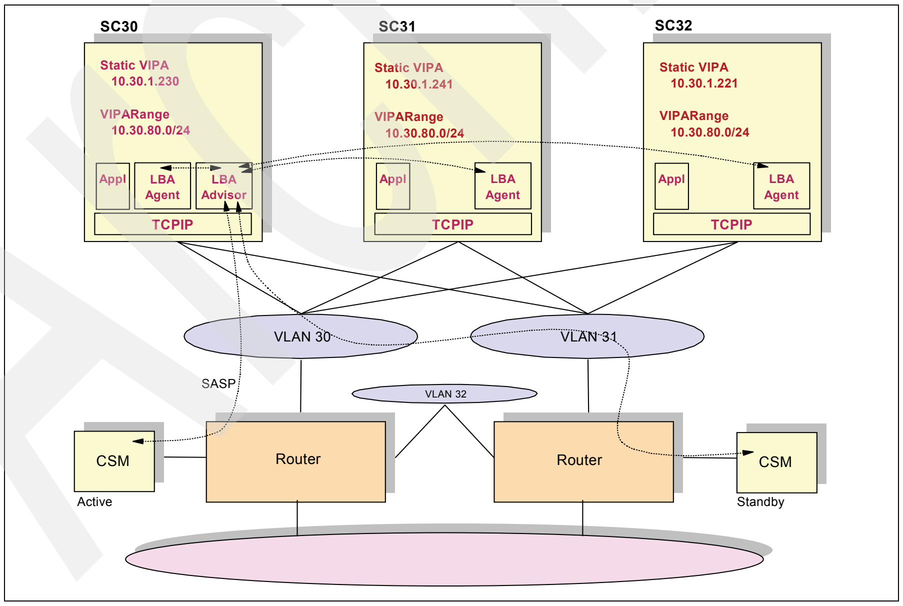


Figure 6-7 External load balancing with LBA/SASP

Environment

The specific environment for this example includes the following:

- Three LPARS in the same server

- 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
- Policy-based routing (PBR) used in the routers
- Two Cisco Catalyst 6500 Series Switches
- Two Cisco Content Switching Modules (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 should use a dynamic VIPA. We did not include this function in out example.

6.3.3 Recommendations

If you elect to use an external load balancing solution for the z/OS environment we strongly recommend the z/OS Load Balancer Advisor (LBA) and Server/Application State Protocol (SASP) approach.

6.4 Implementations

The following sections provide detailed listings for key elements of our two examples.

6.4.1 External load balancer without LBA/SASP

Our work plan started with a list of implementation tasks.

Implementation tasks
Our key tasks were the following:

- Plan the IP addressing needed for our configuration.
- Configure the TN3270 server IP and port addresses.
- Define the CSM active and standby configurations.

We used the IP addressing plan shown in Table 6-1 on page 151.

Table 6-1 IP addressing scheme

Application	Application cluster address	Application instance IP address
TN3270 - Port 23	10.30.60.11	SC30: 10.30.1.230 SC31: 10.30.1.241 SC32: 10.30.1.221

The key TCP/IP profile statements for our three LPARs are shown in Example 6-1, Example 6-2, and Example 6-3.

Example 6-1 TCP/IP profile configuration in SC30

```
PORT
    23 TCP INTCLIEN                  ; MVS Telnet Server 1
HOME
   10.30.1.230   STAVIPA1LNK 2
   10.30.2.232    OSA2080LNK
   10.30.3.235    OSA20E0LNK
```

Example 6-2 TCP/IP profile configuration in SC31

```
PORT
    23 TCP INTCLIEN                  ; MVS Telnet Server 1

HOME
   10.30.1.241   STAVIPA1LNK 2
   10.30.2.242    OSA2080LNK
   10.30.3.245    OSA20E0LNK
```

Example 6-3 TCP/IP profile configuration in SC32

```
PORT
    23 TCP INTCLIEN                  ; MVS Telnet Server 1

HOME
   10.30.1.221   STAVIPA1LNK 2
   10.30.2.222    OSA2080LNK
   10.30.3.225    OSA20E0LNK
```

Note that 1 indicates TN3270 binding to INADDR_ANY, and 2 indicates static VIPA.

Our active CSM definitions are shown in Example 6-4.

Example 6-4 Active CSM definitions

```
module ContentSwitchingModule 3 1
 variable ROUTE_UNKNOWN_FLOW_PKTS 1
!
 ft group 61 vlan 61 2
  priority 101
!
 vlan 60 server 3
  ip address 10.30.60.2 255.255.255.0
  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 10.30.60.1 255.255.255.255
!
 probe PING-30S icmp 4
```

```
  interval 30
  retries 2
  failed 30
!
 real SC30-TN3270 5
  address 10.30.1.230
  inservice
 real SC31-TN3270
  address 10.30.1.241
  inservice
 real SC32-TN3270
  address 10.30.1.221
  inservice
!
 serverfarm TN3270-BASIC 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-BASIC 7
  virtual 10.30.60.11 tcp telnet
  serverfarm TN3270-BASIC
  replicate csrp connection
  persistent rebalance
  inservice
!
interface Vlan30 8
 ip address 10.30.2.1 255.255.255.0
 ip policy route-map pbr-to-csm
!
interface Vlan32 9
 description intra-router vlan
 ip address 10.30.32.1 255.255.255.0
 ip ospf cost 1
!
interface Vlan60 10
 description VLAN 60 for CSM
 ip address 10.30.60.4 255.255.255.0
 ip ospf cost 5
!
interface Vlan61 11
 description CSM Failover
 no ip address
!
router ospf 300 12
 router-id 10.30.2.1
 log-adjacency-changes
 area 1 stub no-summary
 network 10.30.0.0 0.0.255.255 area 1
!
ip access-list extended tn3270 13
 permit tcp host 10.30.1.230 eq telnet any
 permit tcp host 10.30.1.241 eq telnet any
 permit tcp host 10.30.1.221 eq telnet any
```

```
!
!
route-map pbr-to-csm permit 10 14
 match ip address tn3270
 set ip next-hop 10.30.60.1
!
```

Note the following key elements in the definitions:

- 1 CSM is in module 3 of this Cisco Switch 6509.
- 2 Fault tolerance uses dedicated VLAN 61.
- 3 CSM server VLAN. Subnet 10.30.60.0/24. Subnet for CSM and application clusters. There is no routing within the CSM, only static routes pointing to the routers.
- 4 ICMP probe used for polling a given application instance for availability.
- 5 Real server definition pointing to the static VIPA per LPAR.
- 6 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 Virtual server representing the application cluster. *Replicate csrp connection* implements stateful failover between the CSMs without connection loss of existing connections.
- 8 VLAN 30. Subnet 10.30.2.0/24. OSA-Express Gigabit Ethernet attachment of z/OS. Policy-based routing is enabled in this VLAN.
- 9 VLAN 32. Inter-router communication if the path through VLAN 30 becomes unavailable.
- 10 VLAN 60. Subnet 10.30.60.0/24. Subnet for CSM and application clusters.
- 11 VLAN 61. Dedicated VLAN for CSM failover services.
- 12 OSPF process 300 includes subnets matching 10.30.0.0/16.
- 13 Extended access list used by policy-based routing.
- 14 Route Map used by policy-based routing. 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 2 1
 variable ROUTE_UNKNOWN_FLOW_PKTS 1
!
 ft group 61 vlan 61 2
  priority 100
!
 vlan 60 server 3
  ip address 10.30.60.3 255.255.255.0
  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 10.30.60.1 255.255.255.255
!
 probe PING-30S icmp 4
  interval 30
  retries 2
  failed 30
!
 real SC30-TN3270 5
  address 10.30.1.230
  inservice
 real SC31-TN3270
  address 10.30.1.241
  inservice
 real SC32-TN3270
  address 10.30.1.221
  inservice
 real TN3270-SC30
  inservice
!
 serverfarm TN3270-BASIC 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-BASIC 7
  virtual 10.30.60.11 tcp telnet
  serverfarm TN3270-BASIC
  replicate csrp connection
  persistent rebalance
  inservice
!
interface Vlan31 8
 ip address 10.30.3.1 255.255.255.0
 ip policy route-map pbr-to-csm
!
interface Vlan32 9
 description intra router vlan
 ip address 10.30.32.2 255.255.255.0
 ip ospf cost 1
!
interface Vlan60 10
 description VLAN 60 for CSM
 ip address 10.30.60.5 255.255.255.0
 ip ospf cost 5
!
interface Vlan61 11
 description CSM Failover
 no ip address
!
router ospf 300 12
 router-id 10.30.3.1
 log-adjacency-changes
 area 1 stub no-summary
 network 10.30.0.0 0.0.255.255 area 1
!
ip access-list extended tn3270 13
 permit tcp host 10.30.1.230 eq telnet any
 permit tcp host 10.30.1.241 eq telnet any
 permit tcp host 10.30.1.221 eq telnet any
!
!
route-map pbr-to-csm permit 10 14
 match ip address tn3270
 set ip next-hop 10.30.60.1
```

```
!
```

The key points in the standby definitions include the following:

- 1 CSM is in module 2 of this Cisco Switch 6509.
- 2 Fault tolerance use dedicated VLAN 61.
- 3 CSM server VLAN. Subnet 10.30.60.0/24. Subnet for CSM and application clusters. There is no routing within the CSM, only static routes pointing to the routers.
- 4 ICMP probe used for *polling* a given application instance for availability.
- 5 Real server definition pointing to the static VIPA per LPAR.
- 6 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 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 10.30.3.0/24. OSA-Express Gigabit Ethernet attachment of z/OS. Policy-based routing is enabled in this VLAN.
- 9 VLAN 32. Inter-router communication if the path through VLAN 31 becomes unavailable.
- 10 VLAN 60. Subnet 10.30.60.0/24. Subnet for CSM and application clusters.
- 11 VLAN 61. Dedicated VLAN for CSM failover services.
- 12 OSPF process 300 includes subnets matching 10.30.0.0/16.
- 13 Extended access list used by policy-based routing.
- 14 Route Map used by policy-based routing. 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
EZD0101I NETSTAT CS V1R7 TCPIPC 552
USER ID  CONN     STATE
TCPIPC   00000021 LISTEN
  LOCAL SOCKET:   ::..23
  FOREIGN SOCKET: ::..0
1 OF 1 RECORDS DISPLAYED
```

We used the commands in Example 6-7 and Example 6-8 on page 156 to verify that the active CSM and standby CSM were ready.

Example 6-7 Command issued in active CSM

```
Router# sh mod csm 3 ft 1
FT group 61, vlan 61
 This box is active
 priority 101, heartbeat 1, failover 3, preemption is off

Router#
```

Example 6-8 Command issued in standby CSM

```
Router# sh mod csm 2 ft 1
FT group 61, vlan 61
 This box is in standby state
 priority 100, heartbeat 1, failover 3, preemption is off

Router#
```

The command in Example 6-9 displayed the active vserver configuration.

Example 6-9 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 6-10 displayed the active server farm configuration.

Example 6-10 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 6-11 displays the real servers being used.

Example 6-11 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 6-12.

Example 6-12 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 **1** 12 connections appear to be distributed round-robin.

We verified the TN3270 server usage by displaying the connections in each LPAR, as shown in Example 6-13, Example 6-14 on page 158, and Example 6-15 on page 158.

Example 6-13 TN3270 connections in SC30

```
D TCPIP,TCPIPC,N,CONN,PORT=23
EZD0101I NETSTAT CS V1R7 TCPIPC 733
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 6-14 TN3270 connections in SC31

```
D TCPIP,TCPIPC,N,CONN,PORT=23
EZD0101I NETSTAT CS V1R7 TCPIPC 667
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 6-15 TN3270 connections in SC32

```
D TCPIP,TCPIPC,N,CONN,PORT=23
EZD0101I NETSTAT CS V1R7 TCPIPC 059
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 did our problem determination in the following order:

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 analyser.
10. Run debug in the external networking devices.

6.4.2 External load balancer with LBA/SASP

Our work started with listing the implementation tasks we needed.

Implementation tasks

The tasks for implementing external load balancing with LBA/SASP are:

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.

Our IP addressing plan in shown in Table 6-2.

Table 6-2 IP addressing scheme

Application	Application cluster address	Application instance IP address
TN3270 - Port 23	10.30.60.10	SC30: 10.30.1.230 SC31: 10.30.1.241 SC32: 10.30.1.221

We provided the TCP/IP profile definitions for the z/OS Load Balancing Advisor and Agents for the three LPARs, as shown in Example 6-16, Example 6-17 on page 160, and Example 6-18 on page 160.

Example 6-16 TCP/IP profile definitions for z/OS Load Balancing Advisor and z/OS Load Balancing Agent in SC30

```
VIPADynamic
 VIPARange  DEFINE MOVEable NONDISRUPTive 255.255.255.0 10.30.80.0
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 6-17 TCP/IP profile definitions for z/OS Load Balancing Advisor and z/OS Load Balancing Agent in SC31

```
VIPADynamic
 VIPARange  DEFINE MOVEable NONDISRUPTive 255.255.255.0 10.30.80.0
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)
```

Example 6-18 TCP/IP profile definitions for z/OS Load Balancing Advisor and z/OS Load Balancing Agent in SC32

```
VIPADynamic
 VIPARange  DEFINE MOVEable NONDISRUPTive 255.255.255.0 10.30.80.0
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 6-19. (The z/OS Load Balancing Advisor configuration file is specified on the CONFIG DD statement in the advisor start proc.)

Example 6-19 Configuration file for z/OS Load Balancing Advisor

```
debug_level             7

update_interval         60

agent_connection_port   8100 1

agent_id_list
{
  10.30.1.221..8000 2
  10.30.1.230..8000
  10.30.1.241..8000
# ::1..8000
}
```

```
lb_connection_v4          10.30.80.10..3860 3

lb_id_list
{
  10.30.60.2 4
  10.30.60.3 5
# ::1
}
wlm serverwlm 6

port_list
{
  7000
  {
    wlm serverwlm
  }
  7100
  {
    wlm serverwlm
  }
}
```

The following key elements are in this configuration are:

- 1 Port 8100 is used to receive connections from LBAGENTs.
- 2 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 IP address of active load balancer.
- 5 IP address of standby load balancer.
- 6 Serverwlm weights will be the default.

We configured the z/OS Load Balancing Agents in each LPAR, as shown in Example 6-20, Example 6-21, and Example 6-22. (The z/OS Load Balancing Agent configuration file is specified on the CONFIG DD statement in the agent start proc.)

Example 6-20 Configuration file for z/OS Load Balancing Agent in SC30

```
debug_level               7

advisor_id                10.30.80.10..8100 1

host_connection           10.30.1.230..8000 2
```

Example 6-21 Configuration file for z/OS Load Balancing Agent in SC31

```
debug_level               7

advisor_id                10.30.80.10..8100 1

host_connection           10.30.1.241..8000 2
```

Example 6-22 Configuration file for z/OS Load Balancing Agent in SC32

```
debug_level               7

advisor_id                10.30.80.10..8100 1

host_connection           10.30.1.221..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 6-23.

Example 6-23 Configuration for active CSM

```
module ContentSwitchingModule 3 1
 variable ROUTE_UNKNOWN_FLOW_PKTS 1
 variable SASP_CSM_UNIQUE_ID Cisco-CSM-6509A 2

 ft group 61 vlan 61 3
  priority 101

 vlan 60 server 4
  ip address 10.30.60.2 255.255.255.0
  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 10.30.60.1 255.255.255.255

 probe PING-30S icmp 5
  interval 30
  retries 2
  failed 30

 real SC30-TN3270 6
  address 10.30.1.230
  inservice
 real SC31-TN3270
  address 10.30.1.241
  inservice
 real SC32-TN3270
  address 10.30.1.221
  inservice

 serverfarm TN3270 7
  nat server
  no nat client
  bindid 65520
  real name SC30-TN3270
   inservice
  real name SC31-TN3270
   inservice
  real name SC32-TN3270
   inservice

 vserver TN3270 8
  virtual 10.30.60.10 tcp telnet
  serverfarm TN3270
  replicate csrp connection
  persistent rebalance
  inservice

 dfp 9
  agent 10.30.80.10 3860 65520

interface Vlan30 10
 ip address 10.30.2.1 255.255.255.0
 ip policy route-map pbr-to-csm
```

```
interface Vlan32 11
 description intra-router vlan
 ip address 10.30.32.1 255.255.255.0
 ip ospf cost 1

interface Vlan60 12
 description VLAN 60 for CSM
 ip address 10.30.60.4 255.255.255.0
 ip ospf cost 5

interface Vlan61 13
 description CSM Failover
 no ip address

router ospf 300 14
 router-id 10.30.2.1
 log-adjacency-changes
 network 10.30.0.0 0.0.255.255 area 0
 default-information originate always

ip access-list extended tn3270 15
 permit tcp host 10.30.1.221 eq telnet any
 permit tcp host 10.30.1.230 eq telnet any
 permit tcp host 10.30.1.241 eq telnet any

route-map pbr-to-csm permit 10 16
 match ip address tn3270
 set ip next-hop 10.30.60.1
```

The following elements are important:

- 1 CSM is in module 3 of this Cisco Switch 6509.
- 2 Unique ID for active CSM required.
- 3 Fault tolerance use dedicated VLAN 61.
- 4 CSM server VLAN. Subnet 10.30.60.0/24. Subnet for CSM and application clusters. There is no routing within the CSM, only static routes pointing to the routers.
- 5 ICMP probe used for polling a given application instance for availability.
- 6 Real server definition pointing to the static VIPA per LPAR.
- 7 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 Virtual server representing the application cluster. *Replicate csrp connection* implements stateful failover between the CSMs without connection loss of existing connections.
- 9 DFP agent pointing to z/OS Load Balancing Advisor.
- 10 VLAN 30. Subnet 10.30.2.0/24. OSA-Express Gigabit Ethernet attachment of z/OS. Policy-based routing is enabled in this VLAN.
- 11 VLAN 32. Inter-router communication if the path through VLAN 30 becomes unavailable.
- 12 VLAN 60. Subnet 10.30.60.0/24. Subnet for CSM and application clusters.
- 13 VLAN 61. Dedicated VLAN for CSM failover services.
- 14 OSPF process 300 includes subnets matching 10.30.0.0/16.
- 15 Extended access list used by policy-based routing.

- **16** Route Map used by policy-based routing. Ensures that datagrams from extended access list TN3270 are being forwarded back to the CSM.

We configured the standby CSM as shown in Example 6-24.

Example 6-24 Configuration for standby CSM

```
module ContentSwitchingModule 2 1
 variable ROUTE_UNKNOWN_FLOW_PKTS 1
 variable SASP_CSM_UNIQUE_ID Cisco-CSM-6509B 2

 ft group 61 vlan 61 3
  priority 100

 vlan 60 server 4
  ip address 10.30.60.3 255.255.255.0
  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 10.30.60.1 255.255.255.255

 probe PING-30S icmp 5
  interval 30
  retries 2
  failed 30

 real SC30-TN3270 6
  address 10.30.1.230
  inservice
 real SC31-TN3270
  address 10.30.1.241
  inservice
 real SC32-TN3270
  address 10.30.1.221
  inservice

 serverfarm TN3270 7
  nat server
  no nat client
  bindid 65520
  real name SC30-TN3270
   inservice
  real name SC31-TN3270
   inservice
  real name SC32-TN3270
   inservice

 vserver TN3270 8
  virtual 10.30.60.10 tcp telnet
  serverfarm TN3270
  replicate csrp connection
  persistent rebalance
  inservice

 dfp 9
 agent 10.30.80.10 3860 65520

interface Vlan31 10
 ip address 10.30.3.1 255.255.255.0
 ip policy route-map pbr-to-csm

interface Vlan32 11
```

```
 description intra router vlan
 ip address 10.30.32.2 255.255.255.0
 ip ospf cost 1

interface Vlan60 12
 description VLAN 60 for CSM
 ip address 10.30.60.5 255.255.255.0
 ip ospf cost 5

interface Vlan61 13
 description CSM Failover
 no ip address

router ospf 300 14
 router-id 10.30.3.1
 log-adjacency-changes
 network 10.30.0.0 0.0.255.255 area 0
 default-information originate always

ip access-list extended tn3270 15
 permit tcp host 10.30.1.221 eq telnet any
 permit tcp host 10.30.1.230 eq telnet any
 permit tcp host 10.30.1.241 eq telnet any

route-map pbr-to-csm permit 10 16
 match ip address tn3270
 set ip next-hop 10.30.60.1
```

Some elements are slightly different than used with the active CSM:

- 1 CSM is in module 3 of this Cisco Switch 6509.
- 2 Unique ID for active CSM required.
- 3 Fault tolerance use dedicated VLAN 61.
- 4 CSM server VLAN. Subnet 10.30.60.0/24. Subnet for CSM and application clusters. There is no routing within the CSM, only static routes pointing to the routers.
- 5 ICMP probe used for polling a given application instance for availability.
- 6 Real server definition pointing to the static VIPA per LPAR.
- 7 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 Virtual server representing the application cluster. *Replicate csrp connection* implements stateful failover between the CSMs without connection loss of existing connections.
- 9 DFP agent pointing to z/OS Load Balancing Advisor.
- 10 VLAN 31. Subnet 10.30.3.0/24. OSA-Express Gigabit Ethernet attachment of z/OS. Policy-based routing is enabled in this VLAN.
- 11 VLAN 32. Inter-router communication if the path through VLAN 30 becomes unavailable.
- 12 VLAN 60. Subnet 10.30.60.0/24. Subnet for CSM and application clusters.
- 13 VLAN 61. Dedicated VLAN for CSM failover services.
- 14 OSPF process 300 includes subnets matching 10.30.0.0/16.
- 15 Extended access list used by policy-based routing.

- 16 Route Map used by policy-based routing. Ensures that datagrams from extended access list TN3270 are being forwarded back to the CSM.

Verification

We verified correct operation by using the following steps:

1. We started the LB advisor.
2. We started the LB agent in all three LPARs.
3. We checked the connections between the agents and the Cisco units.
4. We checked the status of the DFP agents in CSM.
5. We verified that the 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, as shown in Example 6-25 and Example 6-26.

Example 6-25 LB Advisor is listening ready and listening

```
D TCPIP,TCPIPC,N,CONN,PORT=3860
EZD0101I NETSTAT CS V1R7 TCPIPC 212
USER ID  CONN     STATE
LBADV    000001C7 LISTEN
  LOCAL SOCKET:   10.30.80.10..3860
  FOREIGN SOCKET: 0.0.0.0..0
1 OF 1 RECORDS DISPLAYED
```

Example 6-26 z/OS Load Balancing Advisor joblog

```
$HASP373 LBADV    STARTED
IEF188I PROBLEM PROGRAM ATTRIBUTES ASSIGNED
EZD1231I LBADV STARTING
EZD1232I LBADV INITIALIZATION COMPLETE
EZD1263I LBADV LOAD BALANCER CONNECTED FROM 10.30.60.3
EZD1263I LBADV LOAD BALANCER CONNECTED FROM 10.30.60.2
EZD1261I LBADV AGENT CONNECTED FROM ::FFFF:10.30.1.221
EZD1261I LBADV AGENT CONNECTED FROM ::FFFF:10.30.1.230
EZD1261I LBADV AGENT CONNECTED FROM ::FFFF:10.30.1.241
```

The agents started in all three LPARs, as shown in Example 6-27.

Example 6-27 LB Agent joblog in SC30, SC31, and SC32

```
$HASP373 LBAGENT  STARTED
IEF188I PROBLEM PROGRAM ATTRIBUTES ASSIGNED
EZD1231I LBAGENT STARTING
EZD1232I LBAGENT INITIALIZATION COMPLETE
EZD1259I LBAGENT CONNECTED TO ADVISOR 10.30.80.10
```

We verified that the agents were connected to the advisor, as shown in Example 6-28, Example 6-29, and Example 6-30.

Example 6-28   z/OS Load Balancing Advisor connection in SC30

```
D TCPIP,TCPIPC,N,CONN,PORT=8000
EZD0101I NETSTAT CS V1R7 TCPIPC 413
USER ID  CONN     STATE
LBAGENT  000001CF ESTBLSH
  LOCAL SOCKET:   10.30.1.230..8000
  FOREIGN SOCKET: 10.30.80.10..8100
1 OF 1 RECORDS DISPLAYED
```

Example 6-29   z/OS Load Balancing Advisor connection in SC31

```
D TCPIP,TCPIPC,N,CONN,PORT=8000
EZD0101I NETSTAT CS V1R7 TCPIPC 335
USER ID  CONN     STATE
LBAGENT  00000242 ESTBLSH
  LOCAL SOCKET:   10.30.1.241..8000
  FOREIGN SOCKET: 10.30.80.10..8100
1 OF 1 RECORDS DISPLAYED
```

Example 6-30   z/OS Load Balancing Advisor connection in SC32

```
D TCPIP,TCPIPC,N,CONN,PORT=8000
EZD0101I NETSTAT CS V1R7 TCPIPC 303
USER ID  CONN     STATE
LBAGENT  000001F0 ESTBLSH
  LOCAL SOCKET:   10.30.1.221..8000
  FOREIGN SOCKET: 10.30.80.10..8100
1 OF 1 RECORDS DISPLAYED
```

We checked the connections from the LBA connection on the LPAR running the advisor, as shown in Example 6-31.

Example 6-31   .z/OS Load Balancing Advisor connections in SC30

```
LBADV    000001D2 Establsh
  Local Socket:   ::ffff:10.30.80.10..8100
  Foreign Socket: ::ffff:10.30.1.230..8000 1
LBADV    000001D4 Establsh
  Local Socket:   ::ffff:10.30.80.10..8100
  Foreign Socket: ::ffff:10.30.1.241..8000 2
LBADV    000001D0 Establsh
  Local Socket:   ::ffff:10.30.80.10..8100
  Foreign Socket: ::ffff:10.30.1.221..8000 3
LBADV    000001C7 Listen
  Local Socket:   10.30.80.10..3860 4
  Foreign Socket: 0.0.0.0..0
LBADV    000001CA Establsh
  Local Socket:   10.30.80.10..3860
  Foreign Socket: 10.30.60.3..1030 5
LBADV    000001CA Establsh
  Local Socket:   10.30.80.10..3860
  Foreign Socket: 10.30.60.2..1030 6
LBADV    000001C9 Listen
  Local Socket:   ::..8100 7
  Foreign Socket: ::..0
```

```
LBAGENT  000001CF Establsh
  Local Socket:   10.30.1.230..8000 8
  Foreign Socket: 10.30.80.10..8100
```

Relevant points in this display are:

- **1** z/OS Load Balancing Agent from SC30
- **2** z/OS Load Balancing Agent from SC31
- **3** z/OS Load Balancing Agent from SC32
- **4** z/OS Load Balancing Advisor listener socket - SASP clients
- **5** z/OS Load Balancing Advisor connection from standby CSM
- **6** z/OS Load Balancing Advisor connection from active CSM
- **7** z/OS Load Balancing Advisor listener socket - z/OS Load Balancing Agents
- **8** 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 6-32.

Example 6-32 DFP Agent display in active CSM

```
Router# sh mod csm 3 dfp detail
DFP Agent 10.30.80.10:3860  Connection state: Connected 1
   Keepalive = 65520  Retry Count = 0      Interval = 180   (Default)
   Security errors = 0
   Last message received: 19:39:18 Local 10/24/05
   Last reported Real weights for Protocol TCP, Port telnet
      Host 10.30.1.221      Bind ID 65520  Weight 2
      Host 10.30.1.230      Bind ID 65520  Weight 1
      Host 10.30.1.241      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 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 6-33.

Example 6-33 DFP Agent display in standby CSM

```
Router#sh mod csm 2 dfp detail
DFP Agent 10.30.80.10:3860  Connection state: Connected 1
   Keepalive = 65520  Retry Count = 0      Interval = 180   (Default)
   Security errors = 0
   Last message received: 19:46:13 Local 10/24/05
   Last reported Real weights for Protocol TCP, Port telnet
      Host 10.30.1.221      Bind ID 65520  Weight 2
      Host 10.30.1.230      Bind ID 65520  Weight 1
      Host 10.30.1.241      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 6-34 on page 169.

Example 6-34 Display a load balancer summary

```
F LBADV,DISP,LB
EZD1242I LOAD BALANCER SUMMARY 845
LB INDEX      : 00       UUID      : 436973636F2D43534D2D3635303942 1
 IPADDR..PORT : 10.30.60.3..1030
 HEALTH       : 7E       FLAGS     : NOCHANGE PUSH
LB INDEX      : 01       UUID      : 436973636F2D43534D2D3635303941 2
 IPADDR..PORT : 10.30.60.2..1030
 HEALTH       : 7E       FLAGS     : NOCHANGE PUSH
2 OF 2 RECORDS DISPLAYED
```

Note that 1 LB INDEX 00 shows the DFP Agent from the standby CSM and 2 LB INDEX 01 shows the DFP Agent from the active CSM.

We displayed a detailed list of the data registered with the active CSM under index 01, as shown in Example 6-35. (The LB index numbers were shown in Example 6-34.)

Example 6-35 Display load balancer details about active CSM

```
F LBADV,DISP,LB,I=01
EZD1243I LOAD BALANCER DETAILS 870
LB INDEX      : 01       UUID      : 436973636F2D43534D2D3635303941
 IPADDR..PORT : 10.30.60.2..1030
 HEALTH       : 7E       FLAGS     : NOCHANGE PUSH
 GROUP NAME   : TN3270
  GROUP FLAGS : SERVERWLM
  IPADDR..PORT: 10.30.1.241..23
   SYSTEM NAME: SC31      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00016     CS WEIGHT : 100  NET WEIGHT: 00001 1
   FLAGS      :
  IPADDR..PORT: 10.30.1.230..23
   SYSTEM NAME: SC30      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00016     CS WEIGHT : 100  NET WEIGHT: 00001 1
   FLAGS      :
  IPADDR..PORT: 10.30.1.221..23
   SYSTEM NAME: SC32      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00030     CS WEIGHT : 100  NET WEIGHT: 00002 1
   FLAGS      :
3 OF 3 RECORDS DISPLAYED
```

Example 6-36 displays the registered data from the standby CSM (index 00).

Example 6-36 Display load balancer details about standby CSM

```
F LBADV,DISP,LB,I=00
EZD1243I LOAD BALANCER DETAILS 936
LB INDEX      : 00       UUID      : 436973636F2D43534D2D3635303942
 IPADDR..PORT : 10.30.60.3..1030
 HEALTH       : 7E       FLAGS     : NOCHANGE PUSH
 GROUP NAME   : TN3270
  GROUP FLAGS : SERVERWLM
  IPADDR..PORT: 10.30.1.241..23
   SYSTEM NAME: SC31      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00016     CS WEIGHT : 100  NET WEIGHT: 00001 1
   FLAGS      :
  IPADDR..PORT: 10.30.1.230..23
   SYSTEM NAME: SC30      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00016     CS WEIGHT : 100  NET WEIGHT: 00001 1
   FLAGS      :
```

```
  IPADDR..PORT: 10.30.1.221..23
   SYSTEM NAME: SC32      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00030     CS WEIGHT : 100  NET WEIGHT: 00002 1
   FLAGS      :
3 OF 3 RECORDS DISPLAYED
```

Note the 1 WLM weight, Communications Server (CS) weight, and NET weight (the weight that is forwarded to the external load balancer).

The load balancer details displayed in Example 6-36 on page 169 are also available from the CSM, as shown in Example 6-37 and Example 6-38. Remember that the z/OS Load Balancing Advisor sends the NET WEIGHT to the CSM and to the standby CSM.

Example 6-37 Display dfp weights in active CSM

```
Router# sh mod csm 3 dfp weights

   Real IP : 10.30.1.221 Protocol: TCP Port: telnet Bind_ID: 65520 Weight: 2 1
   Set by Agent 10.30.80.10:3860 at 19:27:10 Local 10/24/05

   Real IP : 10.30.1.230 Protocol: TCP Port: telnet Bind_ID: 65520 Weight: 1 1
   Set by Agent 10.30.80.10:3860 at 19:27:10 Local 10/24/05

   Real IP : 10.30.1.241 Protocol: TCP Port: telnet Bind_ID: 65520 Weight: 1 1
   Set by Agent 10.30.80.10:3860 at 19:27:10 Local 10/24/05
Router#
```

Example 6-38 Display dfp weights in standby CSM

```
Router# sh mod csm 2 dfp weights

   Real IP : 10.30.1.221 Protocol: TCP Port: telnet Bind_ID: 65520 Weight: 2 1
   Set by Agent 10.30.80.10:3860 at 19:34:05 Local 10/24/05

   Real IP : 10.30.1.230 Protocol: TCP Port: telnet Bind_ID: 65520 Weight: 1 1
   Set by Agent 10.30.80.10:3860 at 19:34:05 Local 10/24/05

   Real IP : 10.30.1.241 Protocol: TCP Port: telnet Bind_ID: 65520 Weight: 1 1
   Set by Agent 10.30.80.10:3860 at 19:34:05 Local 10/24/05
Router#
```

Note 1 the NET weight received from the z/OS Load Balancing Advisor.

We used the command shown in Example 6-39, Example 6-40 on page 171, and Example 6-41 on page 171 to verify the members registered in the agents in each LPAR.

Example 6-39 LBAGENT member details in SC30

```
F LBAGENT,DISP,MEM,DET
EZD1245I MEMBER DETAILS 094
LB INDEX      : 00        UUID      : 436973636F2D43534D2D3635303942
 GROUP NAME   : TN3270
  IPADDR..PORT: 10.30.1.230..23 1
   TCPNAME    : TCPIPC    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TCPIPC    ASID      : 00DE RESOURCE  : 00000022
LB INDEX      : 01        UUID      : 436973636F2D43534D2D3635303941
 GROUP NAME   : TN3270
```

```
  IPADDR..PORT: 10.30.1.230..23 2
   TCPNAME    : TCPIPC    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TCPIPC    ASID      : 00DE RESOURCE  : 00000022
2 OF 2 RECORDS DISPLAYED
```

Example 6-40 LBAGENT member details in SC31

```
F LBAGENT,DISP,MEM,DET
EZD1245I MEMBER DETAILS 960
LB INDEX      : 00        UUID      : 436973636F2D43534D2D3635303942
 GROUP NAME   : TN3270
  IPADDR..PORT: 10.30.1.241..23 1
   TCPNAME    : TCPIPC    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TCPIPC    ASID      : 0083 RESOURCE  : 00000023
LB INDEX      : 01        UUID      : 436973636F2D43534D2D3635303941
 GROUP NAME   : TN3270
  IPADDR..PORT: 10.30.1.241..23 2
   TCPNAME    : TCPIPC    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TCPIPC    ASID      : 0083 RESOURCE  : 00000023
2 OF 2 RECORDS DISPLAYED
```

Example 6-41 LBAGENT member details in SC32

```
F LBAGENT,DISP,MEM,DET
EZD1245I MEMBER DETAILS 312
LB INDEX      : 00        UUID      : 436973636F2D43534D2D3635303942
 GROUP NAME   : TN3270
  IPADDR..PORT: 10.30.1.221..23 1
   TCPNAME    : TCPIPC    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TCPIPC    ASID      : 00C0 RESOURCE  : 00000021
LB INDEX      : 01        UUID      : 436973636F2D43534D2D3635303941
 GROUP NAME   : TN3270
  IPADDR..PORT: 10.30.1.221..23 2
   TCPNAME    : TCPIPC    MATCHES   : 001  PROTOCOL  : TCP
   FLAGS      : ANY
   JOBNAME    : TCPIPC    ASID      : 00C0 RESOURCE  : 00000021
2 OF 2 RECORDS DISPLAYED
```

Note key elements in these displays:

- 1 Member details registered by standby CSM. UUID points to standby CSM.
- 2 Member details registered by active CSM. UUID points to active CSM.

We connected 12 TN3270 clients using address 10.30.60.10 and verified the distribution, as shown in Example 6-42. As can be seen, the weights are changed based on updates from the z/OS Load Balancing Advisor. (The weight for SC32-TN3270 changed from 2 to 3 since we made a similar display a few minutes earlier.)

Example 6-42 Display of the server farm in the CSM shows how the 12 connections are distributed

```
Router# sh mod csm 3 serverfarms name tn3270 detail
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65520, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING-30S, type = icmp
  Real servers:
    SC30-TN3270, weight = 1, OPERATIONAL, conns = 2 1
    SC31-TN3270, weight = 1, OPERATIONAL, conns = 2 1
    SC32-TN3270, weight = 3, OPERATIONAL, conns = 8 1
  Total connections = 12

Router#
```

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 eight TN3270 sessions. When we reconnected the sessions they were distributed according to the weights in the remaining TN3270 servers, as shown in Example 6-43. The weighting recommendations may change rapidly during a recovery situation such as this and slightly different reconnection timings could produce slightly different connection distributions.

Example 6-43 Display of the server farm in the CSM shows how the 12 connections are distributed

```
Router# sh mod csm 3 serverfarms name tn3270 detail
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65520, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING-30S, type = icmp
  Real servers:
    SC30-TN3270, weight = 4, OPERATIONAL, conns = 7
    SC31-TN3270, weight = 3, 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 6-44. The zero weight means “Do not forward requests to this server instance.”

Example 6-44 Display load balancer details about active CSM

```
F LBADV,DISP,LB,I=01
EZD1243I LOAD BALANCER DETAILS 635
LB INDEX      : 01        UUID      : 436973636F2D43534D2D3635303941
 IPADDR..PORT : 10.30.60.2..1030
 HEALTH       : 7E        FLAGS     : NOCHANGE PUSH
 GROUP NAME   : TN3270
  GROUP FLAGS : SERVERWLM
  IPADDR..PORT: 10.30.1.241..23
   SYSTEM NAME: SC31      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00013     CS WEIGHT : 100  NET WEIGHT: 00001
   FLAGS      :
  IPADDR..PORT: 10.30.1.230..23
   SYSTEM NAME: SC30      PROTOCOL  : TCP  AVAIL     : YES
   WLM WEIGHT : 00013     CS WEIGHT : 100  NET WEIGHT: 00001
   FLAGS      :
  IPADDR..PORT: 10.30.1.221..23
   SYSTEM NAME: N/A       PROTOCOL  : TCP  AVAIL     : NO
   WLM WEIGHT : 00000     CS WEIGHT : 000  NET WEIGHT: 00000 1
   FLAGS      : LBQ NOTARGETSYS 2
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 6-45 we stopped the z/OS Load Balancing Agent in SC32.

Example 6-45 Display of the server farm in the CSM shows how the 12 connections are distributed

```
Router# sh mod csm 3 serverfarms name tn3270 detail
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65520, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING-30S, type = icmp
  Real servers:
    SC30-TN3270, weight = 1, OPERATIONAL, conns = 2
    SC31-TN3270, weight = 1, OPERATIONAL, conns = 2
    SC32-TN3270, weight = 0, DFP_THROTTLED, conns = 8 1
  Total connections = 12

Router#
```

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 6-46 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 6-46 Connection from DFP fails

```
Router#sh mod csm 3 dfp detail
DFP Agent 10.30.80.10:3860  Connection state: Failed  Retries: 3 1
   Keepalive = 65520  Retry Count = 0      Interval = 180   (Default)
   Security errors = 0
   Last message received: 22:00:12 Local 10/24/05
   Last reported Real weights for Protocol TCP, Port telnet
      Host 10.30.1.221      Bind ID 65520  Weight 3
      Host 10.30.1.230      Bind ID 65520  Weight 1
      Host 10.30.1.241      Bind ID 65520  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 6-47. The equal weights of 8 indicate that the CSM is doing its round-robin balancing.

Example 6-47 Display of the server farm in the CSM shows how the 12 connections are distributed

```
Router#sh mod csm 3 serverfarms name tn3270 detail
TN3270, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 3, bind id = 65520, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING-30S, type = icmp
  Real servers:
    SC30-TN3270, weight = 8, OPERATIONAL, conns = 2 1
    SC31-TN3270, weight = 8, OPERATIONAL, conns = 2 1
    SC32-TN3270, weight = 8, OPERATIONAL, conns = 8 1
  Total connections = 12

Router#
```

1 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 each vserver utilizes separate serverfarms; 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 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 analyser.
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 may 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

Performance and tuning

A system delivering poor response times to the end user may be perceived as unavailable from the end 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 Communications Server for z/OS 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 suggest parameter values that have been observed to maximize performance in many client installations.

7.1 General performance considerations

When discussing performance and response times, we must remember 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 and 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) may 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.

- 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 above. 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).

- OSA-Express/OSA-Express2 adapter

  The OSA-Express/OSA-Express2 adapters contain hardware and firmware that enables the flow of traffic between the Ethernet link and host memory. 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% could 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 may be more of a concern than utilization, especially when running interactive traffic.

7.2 TCP/IP configuration files

The most important configuration files for z/OS TCP/IP are the following:

- 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: One important 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.

7.2.1 Appropriate MTU for devices

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 also applies to the finding of the parameter setting of the largest possible MTU size in your installation. Table 8-1 provides an overview of the largest MTU sizes supported by different interfaces.

Table 7-1 Maximum MTU sizes of interfaces by LINK type

Link type	Connectivity	max.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	57377

The MTU size used for a given outbound frame depends on several factors:

- interface_MTU

This is either a hard-coded size based on the physical device or a value 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 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 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 for sending outbound packets.

- path_MTU

  This is the value determined by the path MTU discovery function. When path MTU discovery is in effect, TCP/IP uses path_MTU for sending outbound packets. We 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 7-1 illustrates this function.

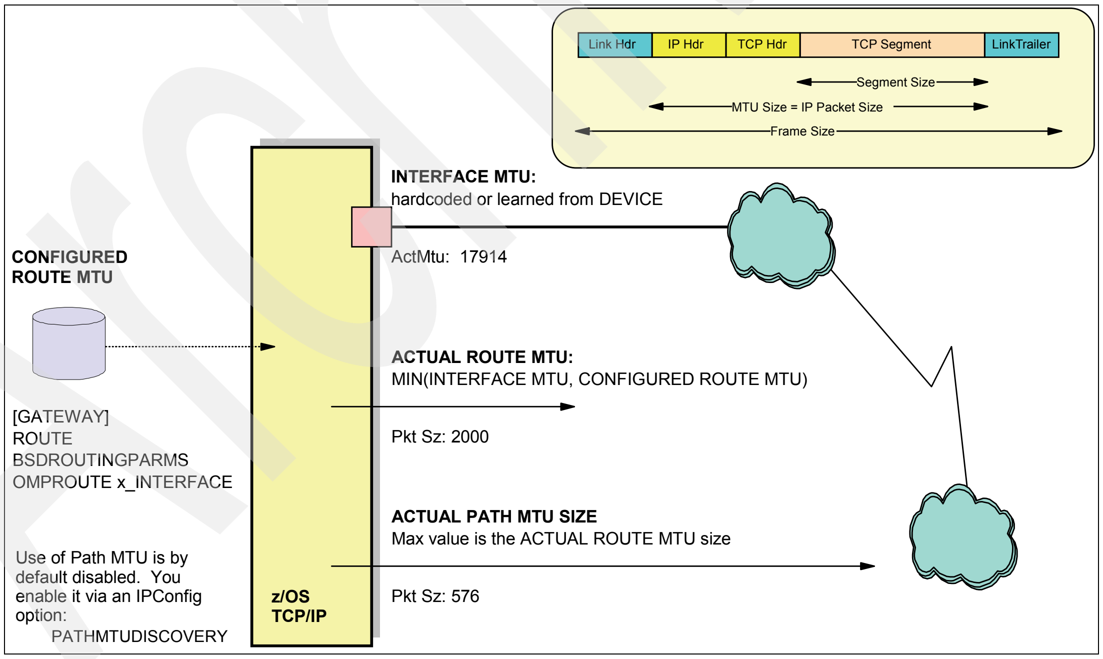


Figure 7-1 Large MTU sizes have a very positive impact on file transfer type of workloads

Recommendations and guidelines are:

- 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.

7.2.2 Tracing

From a performance perspective, all tracing should be disabled. Tracing activity can have a significant impact on system performance. To disable tracing, include the following in the TCPIP.PROFILE:

```
ITRACE OFF
```

To to turn off ctrace for TCP/IP, issue the following command:

```
TRACE  CT,OFF,COMP=SYSTCPIP,SUB=(tcp_proc_name)
```

If tracing is to be used, 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 the ITRACE command above, “CONFIG 1” specifies “tracing level 1” for the configuration component. Documentation for ITRACE can be found in *z/OS V1R7.0 Communications Server: IP Configuration Reference*, SC31-8776.

7.3 z/OS UNIX System Services tuning

The *z/OS V1R7 UNIX System Services Planning* manual, GA22-7800, provides useful tuning information. The following are among the most important 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 System Services 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. Once 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, since this adds functionality (sharing), but with some performance expense. File system tuning is a large topic that is outside the scope of this networking document.
- Monitor z/OS UNIX resources usage with RMF™ or system commands (DISPLAY ACTIVE, DISPLAY OMVS, and so forth).

7.4 Storage requirements

For a system with significant network activity we should 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 may serve as a starting point for estimating CSA, SQA, and CSM storage requirements for these and other applications.

7.4.1 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 a maximum of 65535.

We can specify the send/recv buffer sizes on the TCPCONFIG and UDPCONFIG statements, as shown in Example 7-1.

Example 7-1 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 by 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.

7.4.2 CSM storage usage

Communications Storage Manager (CSM) storage consists of the buffer pools that are shared between SNA and TCP/IP for each z/OS system image. You should use the default (100 MB) unless you are storage constrained and need to reduce the size of your estimated usage. Table 7-2 provides a summary of CSM storage usage, based on an internal performance benchmark using a previous version of the Communications Server for z/OS. It may help provide a starting point for tuning.

Table 7-2 CSM usage

Workload	# Users/clients	Workload	MAX CSM (ECSA)	MAX CSM (DataSpace)	Max CSM (FIXED)
Web server	80	5425 c/s	2.97 MB	8.0 MB	11.2 MB
CICS Sockets	84	409 c/s	0.736 MB	1.96 MB	3.8 MB
TN3270 (Echo's)	4000 8000 16000 32000 64000	399.1 tr/sec 798.5 tr/sec 1591.5 tr/sec 3115.0 tr/sec 5732.5 tr/sec	0.85 MB 1.40 MB 1.34 MB 8.10 MB 17.90 MB	5.1 MB 11.1 MB 8.1 MB 12.6 MB 27.4 MB	12.5 MB 13.6 MB 17.5 MB 24.2 MB 50.0 MB

Workload	# Users/clients	Workload	MAX CSM (ECSA)	MAX CSM (DataSpace)	Max CSM (FIXED)
FTP Server	16 Inbound 16 Outbound	49.8 MBps 68.9 MBps	4.5 MB 0.54 MB	6.8 MB 4.4 MB	9.6 MB 6.4 MB

Based on this data and workload, we suggest the following definitions in the IVTPRM00 member of PARMLIB:

- FIXED MAX(60M), which includes 23 MB of additional storage for expected growth in the workloads
- ECSA MAX(40M), which includes 18 MB of additional storage for expected growth in the workloads

With a different application mix, CSM requirements may change. We 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
```

7.4.3 VTAM buffer settings

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 7-3 provides guidelines for initial settings.

Table 7-3 VTAM buffer max usage

Workload	# Users/ clients	Workload	VTAM Buffer (IO00)	VTAM Buffer (LF00)	VTAM Buffer (CRPL)	VTAM Buffer (TI00)	VTAM Buffer (CRA4)
Web server	80	5425 c/s	6	4	2	4	4
CICS Sockets	84	409 c/s	26	5	54	29	6
TN3270 (Echo's)	4000 8000 16000 32000 64000	399.1 tr/sec 798.5 tr/sec 1591.5 tr/sec 3115.0 tr/sec 5732.5 tr/sec	112 112 345 2005 11000	4005 8005 16005 32003 64005	8007 16012 32019 64018 128018	5 5 5 5 5	25 41 49 65 170
FTP server	16 Inbound 16 Outbound	49.8 MBps 68.9 MBps	5 5	4 4	2 2	3 3	4 4

We can use the same commands mentioned earlier to help verify our settings:

```
D NET,BFRUSE,BUFFER=SHORT
D NET,STORUSE
Telnet (TN3270) storage utilization
```

You can check your Telnet storge utilization by running it in a separate address space and using the normal RMF montior functions to see the storage usage.

7.5 Application performance and capacity

The following parameters should be considered for the TELNET section of the profile parameters:

```
TELNETPARMS
   INACTIVE 3600
   TIMEMARK 1200
   SCANINTERVAL 30
   DISABLESGA
ENDTELNETPARMS
```

Where:

- SCANINTERVAL specifies the period (in seconds) for the Telnet server to scan the entire list of sessions. The default is 120 seconds.
- TIMEMARK specifies how often the Telnet server sends a heartbeat to clients. Clients that do not respond to three consecutive probes are labeled inactive.
- INACTIVE specifies how long a client remains in inactive state (without communication) until it is completely disconnected.
- DISABLESGA permits the transmission of GO AHEAD by the Telnet server. This is negotiated by the client and server. Using this parameter increases the overhead for a full-duplex terminal using a full-duplex connection. The recommended action is to disable GO AHEAD.

Telnet (TN3270) capacity planing

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 (3 CP LPARs).

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 (3 CP LPAR) we can use the formula in Figure 7-2. 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 sec
 ------------------------------------- =  0.126   --------
           60  elap. sec                           elap sec
```

Figure 7-2 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 7-3.

```
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 %
     3 processors
```

Figure 7-3 TN3270 CPU utilization

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 V1R7 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 end-user commands.

The use of preallocated data sets for FTP transfers to z/OS is usually recommended. If new data sets allocation is required, then using BLKSIZE=0 usually allows the system to determine the best block size.

FTP capacity planing

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 working with z/OS V1R7 using a OSA-Express Gigabit Ethernet port as network gateway on a z990 2084-332 (2 CP LPAR).

Our total (TCP/IP + VTAM + FTP) CPU requirements for FTP transfer are given by the formula in Figure 7-4. 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 7-4 General formula to calculate FTP CPU requirements

If we consider a load of 69,529.6 KB/Sec 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 7-5.

```
69529.6 KB       .00000656           .456 CPU secs
---------   *  ------------   =        ----------
Elap secs         KB                    Elap  secs
```

Figure 7-5 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 7-6.

```
 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 7-6 FTP CPU requirements example

Thus, the total zOS CPU (TCP/IP + VTAM + FTP) requirements for driving 69,529 KB/Sec throughput would require an average CPU utilzation of 22.8% of a two-processor z990 (2084-332, 2 CP LPAR).

7.6 Communication Server for z/OS performance highlights

During the last few years a number of improvements related to performance were delivered in each Communications Server for z/OS release. The following is a summary of these improvements, starting with V1R5.

7.6.1 Communications Server V1R5 performance highlights

Some highlights are:

- DVIPA limit enhancements.

  The limit for Dynamic Virtual IP Addresses (DVIPAs) was increased from 256 to 1024. In connection with this, some TCP/IP control blocks associated with DVIPAs were moved from common storage to TCP/IP private storage. This provides more flexibility in defining a network configuration. An additional benefit is reducing requirements for common storage.

- Sysplexports performance enhancements.

  The use of the SYSPLEXPORTS function introduced in the z/OS V1R4 Communications Server caused performance degradation for short-lived connections. The z/OS V1R5 Communications Server provides significant performance improvement for short-lived connections by having the stack obtain a group of ephemeral ports instead of requesting one port for each bind. In a Parallel Sysplex environment, port management involves CF structures, and this improvement reduces the need to communicate with the CF.

- OSA performance enhancements (checksum offload).

  The checksum offload function provides improved performance for IPv4 by offloading checksum processing from the z990 to an OSA-Express in QDIO mode that supports the checksum offload function. This saves z990 CPU cycles.

- Improve performance for TN3270 definite response sessions.

  The TN3270E server can turn off DELAYACKS for TN3270E clients that have negotiated a definite response. The throughput will be slightly better while the CPU costs per transaction will decrease a little.

7.6.2 Communications Server V1R6 performance highlights

The performance highlights are:

- TN3270E server address space option

  The TN3270E server can now run in a separate address space from the TCP/IP stack address space. While you can continue to run the TN3270E server as part of the TCP/IP address space, you may want to consider running the TN3270E server separately from TCP/IP for the following reasons:

  - The TN3270E server could run at a different priority than the stack.
  - A second instance of the TN3270E server could be started.
  - The TN3270E server could be stopped without stopping the stack. This makes it easier to reset the server or apply maintenance.

- Large send/segmentation offload

This provides improved performance for outbound IPv4 TCP traffic by offloading outbound TCP segmentation processing to an OSA-Express2 Ethernet feature in QDIO mode, which supports the TCP segmentation offload function. The segmentation offload function is also known as the large send function.

7.6.3 Communications Server V1R7 performance highlights

The highlights are:

- Sysplex optimized routing

  The z/OS V1R7 Communications Server introduces support to enable the Sysplex Distributor to distribute incoming packets to target stacks using the best available IP route. By configuring routes using the VIPAROUTE statement, you enable the system to select the optimal interface for forwarding DVIPA packets to each target. This can reduce use of XCF links. Using high speed IP interfaces like HiperSockets or OSA-Express instead will improve your throughput and save valuable System z9 or zSeries CPU cycles.
- Application Transparent Transport Layer Security (AT-TLS)

  The z/OS V1R7 Communications Server introduces a new Application Transparent Transport Layer Security (AT-TLS) function in the TCP/IP stack to provide TLS for TCP/IP sockets applications that require secure connections. AT-TLS performs TLS on behalf of the application by invoking the z/OS System Secure Socket Layer (SSL) in the TCP transport layer of the stack. The benefit from a performance point of view is that for bulk data workload AT-TLS provides a significantly higher throughput than IPSEC or TLS implemented within the application.

7.7 TCP/IP performance quick checklist

The following is a quick checklist of performance items that you may want to consider:

- 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/IP send and receive buffers and window sizes equal, if possible. For large bulk data transfer workloads, we suggest large TCP/IP send and receive socket buffers (at least 64 KB) and window size of 65535 for both clients and servers.
- If routers are used, ensure that sufficient buffers are allocated in order to avoid excessive numbers of dropped packets.
- When using FTP, use large data set block sizes for traditional z/OS data sets.
- When using the TN3270 server, review your settings for the TIMEMARK, SCANINTERVAL, and INACTIVE parameters in the TCP/IP profile:
  - In order to minimize potential traffic, the default for the TIMEMARK parameter has been increased to: 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 typically low-activity times such as during lunch breaks.
  - In order to avoid CPU issues, the default for the SCANINTERVAL parameter has been increased to: 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). If the INACTIVE time is smaller than the typical lunch break time, significant numbers of sessions may be terminated during lunch.

Note: TIMEMARK and SCANINTERVAL are intended to help with cleaning up idle connections on the TCP/IP side of a TN3270 session while INACTIVE is intended to do the same for the SNA side of things. Hence, if a user is not currently logged onto any SNA application (for example, he has a USS Message 10 screen up), INACTIVE will not terminate the TCP/IP side of his TN3270 connection. Rather, 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.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see "How to get IBM Redbooks" on page 193. Note that some of the documents referenced here may be available in softcopy only.

- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 1 - Base Functions, Connectivity, and Routing*, SG24-7169
- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 2 - Standard Applications*, SG24-7170
- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 3 - High Availability, Scalability, and Performance*, SG24-7171
- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 4 - Security*, SG24-7172
- *z/OS Communications Server: SNA Network Implementation Guide*, *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender,* SG24-5957
- *TCP/IP Tutorial and Technical Overview*, GG24-3376
- *The Basics of IP Network Design*, SG24-2580
- *OSA-Express Implementation Guide*, SG24-5948
- *zSeries HiperSockets*, SG24-6816
- *SNA in a Parallel Sysplex Environment,* SG24-2113
- *z/OS Infoprint Server Implementation*, SG24- 6234
- *z/OS Security Services Update,* SG24-6448-00
- *Deploying a Public Key Infrastructure*, SG24-5512

Other publications

These publications are also relevant as further information sources:

- *z/OS V1R7.0 XL C/C++ Run-Time Library Reference*, SA22-7821
- *z/OS V1R7.0 Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS V1R7.0 MVS IPCS Commands*, SA22-7594
- *z/OS V1R7.0 MVS System Commands*, SA22-7627
- *z/OS V1R7.0 Communications Server: SNA Operation*, SC31-8779
- *z/OS V1R7.0 TSO/E Command Reference*, SA22-7782
- *z/OS V1R7.0 UNIX System Services Command Reference*, SA22-7802
- *z/OS V1R2.0 Communications Server: CSM Guide*, SC31-8808

- *z/OS V1R7.0 Communications Server: New Function Summary*, GC31-8771
- *z/OS V1R7.0 Communications Server: Quick Reference*, SX75-0124
- *z/OS V1R7.0 Communications Server: IP and SNA Codes*, SC31-8791
- *z/OS V1R7.0 Communications Server: IP System Administrator's Commands*, SC31-8781
- *z/OS V1R7.0 MVS IPCS Commands*, SA22-7594
- *z/OS V1R7.0 Communications Server: IP Diagnosis Guide*, GC31-8782
- *z/OS V1R7.0 Communications Server: IP Configuration Guide*, SC31-8775
- *z/OS V1R7.0 Communications Server: IP Messages Volume 1 (EZA)*, SC31-8783
- *z/OS V1R7.0 Communications Server: IP Messages Volume 2 (EZB, EZD)*, SC31-8784
- *z/OS V1R7.0 Communications Server: IP Messages Volume 3 (EZY)*, SC31-8785
- *z/OS V1R7.0 Communications Server: IP Messages Volume 4 (EZZ, SNM)*, SC31-8786
- *z/OS V1R7.0 Communications Server: IP Programmer's Guide and Reference*, SC31-8787
- *z/OS V1R7.0 Communications Server: IP Configuration Reference*, SC31-8776
- *z/OS V1R7.0 Communications Server: IP Sockets Application Programming Interface Guide and Reference*, SC31-8788
- *z/OS V1R7.0 Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS V1R7.0 Communications Server: IP User's Guide and Commands*, SC31-8780
- *z/OS V1R7.0 Communications Server: IPv6 Network and Application Design Guide*, SC31-8885
- *z/OS V1R7.0 Migration*, GA22-7499
- *z/OS V1R7.0 MVS System Commands*, SA22-7627
- *OSA-Express Customer's Guide and Reference*, SA22-7935
- *z/OS V1R7.0 Communications Server: SNA Operation*, SC31-8779
- *z/OS V1R7.0 TSO/E Command Reference*, SA22-7782
- *z/OS V1R7.0 UNIX System Services Programming: Assembler Callable Services Reference*, SA22-7803
- *z/OS V1R7.0 UNIX System Services Command Reference*, SA22-7802
- *z/OS V1R7.0 UNIX System Services File System Interface Reference*, SA22-7808
- *z/OS V1R7.0 UNIX System Services Messages and Codes*, SA22-7807
- *z/OS V1R7.0 UNIX System Services Parallel Environment: Operation and Use*, SA22-7810
- *z/OS V1R7.0 UNIX System Services Programming Tools*, SA22-7805
- *z/OS V1R7.0 UNIX System Services Planning*, GA22-7800
- *z/OS V1R7.0 UNIX System Services User's Guide*, SA22-7801
- *z/OS V1R7.0 UNIX System Services User's Guide*, SA22-7801

Online resources

These Web sites and URLs are also relevant as further information sources:

- 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/servers/eserver/zseries/zos/bkserv/r7pdf/commserv.html`

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks 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

A
active CSM 146
  2 Unique ID 163
  DFP agent 168
  DFP agents 168
active DVIPA
  address 37
  return 37
Address Resolution Protocol (ARP) 4, 13
application cluster 8–9, 146–147
  application instance 146
  end-user requests 146
  IP address 9
application instance 2–3, 17, 29, 84, 106, 140, 143–144
  current workload 150
  external load balancing 148
  status data 140
Application Transparent Transport Layer Security (AT-TLS) 188
ARP table 55–57
  MAC addresses 56
ARP takeover 3, 14
  implementation example 6
  z/OS support 5, 39
AUTOLOG 5 69, 135, 159
Automatic Restart Manager (ARM) 4, 16, 47

B
basic concept 4, 13, 39, 59, 83, 139

C
checksum processing 187
Cisco Appliance Services Architecture (CASA) 87
Cisco Forwarding Agent 10, 87
  Cisco Service Manager communicates 113
Cisco Router 100, 122
Cisco router
  1 123
  GRE tunnels 123
  MNLB configuration 128
  OSA adapter 125
Cisco Service Manager 10, 86, 111
client NAT 141
commands see (CS) 26
Communications Server
  previous version 182
Communications Server (CS) 83, 142, 177
Communications Storage Manager (CSM) 179
configuration example 59, 83, 126, 139
CONN (CS) 44
Connection Establishment Rate (CER) 105
connection request 8, 14, 84, 87
  best suited target server 84
connection routing table (CRT) 22
connection setup (CS) 20, 84
Content Switching Module (CSM) 140
CRA4BUF 183
CRPLBUF 183
CS weight 144
CSA 182
CSM
  storage requirements 182
  storage usage 182
csrp connection 152

D
DEFAULTSIZE 179
DEVICE IUTIQDF6 66
DFP agent 163
DISABLESGA 184
distributed DVIPA
  backup routing stacks 110
distributed dynamic VIPA
  current owner 87
distributing IP
  dynamic VIPA 116
  VIPADYNAMIC/ENDVIPADYNAMIC block 116
DVIPA 4, 14, 85
DVIPA address 5, 14, 42, 67, 116, 118
  10.30.30.1 36
  move 48
DVIPA/PORT 101
dynamic routing 1, 5, 7, 13–14, 39–40, 59, 87
  Single z/OS System 12
  z/OS availability 5
  z/OS Parallel Sysplex 12
dynamic routing protocol 59
DYNAMIC VIPA
  Connection 35–36, 50, 121
  DESTINATION PORT Table 94, 120
  Information 33, 44, 117
Dynamic VIPA 4, 15, 18, 43, 84–85, 149–150
Dynamic XCF
  address 50, 98–99
  interface 25–26, 98

E
Existing connection 23, 47, 85, 146
External application workload balancing 139
external load
  balancer 9, 142, 145
  balancer function 146–147
  balancers available today 140
  balancing 10, 139, 142
external load balancer 9, 142–143
EZZ0053I Command 46, 66
EZZ4315I Deactivation 57, 66

F
failing IP
  IP connections 23
first z/OS
  IP address 15
Foreign Socket 30, 45, 136, 155
Forwarding Agent 10, 87
FTP
  capacity 186
  tuning 185
FTP server 16, 43, 61, 131, 179
  TCPCONFIG statement 182
FTP session
  initial configuration options 185
  ls command 47
ftp session 35, 47, 181
FTP.DATA 179

G
Generic Routing Encapsulation (GRE) 88
GLOBALCONFIG NOTCPIPSTATISTICS 69, 116
GRE tunnel 88
GRE tunneling
  Dispatch Mode use 88

H
Hot Standby Router Protocol (HSRP) 50

I
immediately picks (IP) 13
implementation example 3, 100
Implementation task 115, 150
INACTIVE 184
INADDR_ANY 17
include polling (IP) 2
Internal application workload balancing 8–9, 83
  key characteristics 83
  several examples 106
Intrusion Detection Service (IDS) 142
IOBUF 183
IP ADDRESS
  DONOTAGE 75–76
IP address 4, 13, 15, 40, 85, 98, 139–140
  10.30.30.1 20
  10.30.30.33 116
  connection requests 19
  full knowledge 85
  LBA Advisor 162
  location information 5
IP network 4, 14, 60, 85
IP traffic 67, 145
  load balancing 145
IPV4 Destination 44, 64

K
KB 182

L
LB Index 169
LBA Adviser 143
  8 LBA Agent connection 168
  TCP/IP portion 159
  TCP/IP profile definitions 159
LBA agent 143
LFBUF 183
link H1 21
load balancer
  destination IP address 141
  detail 169
  different source port number 142
  has quiesced 173
  request 145
  routing infrastructure 140
  summary 169
load balancer (LB) 9, 139–140, 148
load balancer has quiesced (LBQ) 173
Load Balancing Advisor (LBA) 10, 139
Local Socket 30, 45, 136, 155
LOCSITE 185
LPAR 17, 87, 145, 181

M
MAC address 5, 31, 40
  ARP cache 56
  same IP address 56
  traditional LAN interconnection 31
MAXFILEPROC 181
maximum transmission unit (MTU) 178–179
MAXPROCUSER 181
MAXPTYS 181
MAXTHREADS 181
MAXTHREADTASKS 181
MAXUIDS 181
MNLB environment 87
MODDVIPA utility 17, 47
MTU
  settings 179
MTU size 178
Multi-Level Security (MLS) 100
Multi-Node Load Balancing (MNLB) 86
MVS.SERVMGR.PAGENT RACF facility class 93

N
NET Weight 169
New DVIPA 30

O
OMPROUTE configuration
  file 69, 180
  statement 14
OS/390 UNIX System Services Tuning 181
OSA Express 85
OSPF neighbor 72, 77
ospf_interface ip_address 33, 62
OSPF_Interface statement 24, 60

Advertise_VIPA_Routes parameter 62
generic entry 60
IP address 25

P
PAGENT 113
Parallel Sysplex
environment 3, 26, 187
Performance checklist 188
Policy Agent 20, 83
performance data collection 100
Policy Based
Routing 141
policyScope 91
PRIMARYINTERFACE STAVIPA1LNK 124

Q
QDIO mode 5, 40–41, 178
OSA adapters 41
OSA-Express2 adapters 41
OSA-Express2 Ethernet feature 188
QoS weight
fraction 89
fraction value 92
Quality of Service (QOS) 20, 83
Quality of Service (QoS) 20

R
RACF
MVS.SERVMGR.PAGENT facility class 93
RDY 94
Real IP 170
real name
SC30-TN3270 152
SC31-TN3270 152
SC32-TN3270 152
real server 153
Redbooks Web site 193
Contact us xiv
retcode map 156
routing table 5, 14, 25, 60

S
same CEC 64, 98
samplingInterval 90
SCANINTERVAL 184
Secure Socket Layer (SSL) 188
Server accept Efficiency Fraction (SEF) 94
Server/Application State Protocol (SASP) 10, 139
Server-specific WLM
recommendation 100
Service Manager 10, 86
failure recovery 113
Forwarding Agents 129
setsockopt 182
show ip
casa affinities detail 129
casa affinity 128
casa oper 128
single network-visible IP address 20
single point 1, 149
SITE 185
SO_SNDBUF 182
source IP address 14, 99, 129, 140
SQA 182
standby CSM
registered data 169
static route 42, 153, 180
Static VIPA 4, 14, 98, 149
static VIPA
example configuration 14
storage requirements 182
sysplex 3, 14, 83, 139
Sysplex Distributor
available IP interface 116
base 84
basic design 84
capability 4
configuration 117
DATAGRAMFWD 20
DYNAMICXCF 20
environment 85
function 83, 86
functionality 22
IP address 129
IP packets 24
operation 20, 85
optimized routing 98
performance monitoring policy 89
policy 84, 89
Policy Agent interacts 88
policy implementation 90
Service Manager 113
SYSPLEXROUTING 20
target stack 89
VIPADISTRIBUTE 21
WLM 20
Sysplex Distributor (SD) 3, 15, 111, 144, 188

T
Target Connectivity Success Rate (TCSR) 105
Target Server
Responsiveness 94
Responsiveness Fraction 105
Responsiveness value 121
target server 10, 84
actual workload 100
distribution method 104
IP address 129
Workload Manager weight value 121
target stack 21, 89, 104, 140
default route definition 140
home list entry 21
outbound IP packets 141
Policy Agents 89
stale data 89
Sysplex autonomics health monitor 104
total set 89

TCP 40, 131, 177
 buffer sizes 182
 window size 182
TCP connection
 request 8
 setup 7, 109
TCP/IP 2, 14, 85, 142, 177
 performance checklist 188
TCP/IP profile 17, 42, 100, 178
 device type LCS 53
 device type MPCIPA 53
 GATEWAY statement 180
 GLOBALCONFIG statement 27
 PORT statement 100
 SCANINTERVAL INACTIVE parameters 188
 VIPARANGE parameter 17
 VIPARange statement 18
TCP/IP protocol
 support 3
TCP/IP stack 4, 14, 62, 145
 available IP network connectivity 98
TCPCONFIG 182
TCPCONFIG RESTRICTLOWPORTS 32, 69
TCPIP 28, 44, 66, 89, 155
TCPIPC.TCPP Arm 133
Telnet
 capacity 184
telnet Bind_ID 170
Three LPARs 107, 148
TIBUF 183
TIMAFF 117
time zone (TZ) 93
TIMEMARK 184
TN3270 183
TN3270 connection 89, 91, 172
 option 142
 request 111
 Server-specific WLM Workload balancing 115
 Sysplex Distributor policy 89
TN3270 server 8, 16, 89, 130, 142
 application instances 107
TN3270 session 120
tr/sec 182
TRACE RESOLVER 179
tracing
 disabling 181
Transport Layer Security (TLS) 188

U
UDP
 buffer sizes 182
UDPCONFIG 182
UNIXMAP RACF 181
UUID 169

V
VIPA 2, 13, 149
VIPA address 7, 19, 48, 60, 85, 108
 Advertisement 60
VIPA Connection Routing Table (VCRT) 35, 121
VIPA subnet 61
 VIPA addresses 61
VIPADEFINE Move 43
VIPADISTRIBUTE DISTMethod
 statement 103
VIPADISTribute DISTMethod 103
VIPARoute function 8, 24–25
VIPAROUTE statement 26, 98, 188
 second parameter 98
Virtual IP 3–4, 13
VLAN 30 65, 153
 OSA adapter 71
 OSA link 65
vlan 60
 server 3 151
 server 4 162
VLAN 61 151
VTAM Buffer Settings 183

W
WLM 20
WLM policy
 goal 144
WLM weight 89, 144
workload distribution 20, 83, 88, 139–140
 eligible target server 91

X
XCF communication 4, 14
XCF link 24, 65, 118, 188

Z
z/OS availability 7
z/OS Sysplex 139
z/OS system 3, 29–30, 40, 52, 140, 150
 book leverage clusters 3
z/OS TCP/IP
 availability 3
 stack 5
z/OS V1R7 28, 89, 142, 181
 Communications Server 188
 environment 66
 IP Configuration Reference 185
 new function 102
 new sysplex distribution feature 110
 TCP/IP Implementation 95

IBM Redbooks

Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 3 - High Availability, Scalability, and Performance

Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 3 - High Availability, Scalability, and Performance

Understand important CS for z/OS TCP/IP high availability capabilities

See CS for z/OS high availability implementation examples

Gain insights into performance and tuning

This new and improved Communications Server (CS) for z/OS TCP/IP Implementation series provides easy-to-understand, step-by-step, how-to guidance on enabling the most commonly used and important functions of CS for z/OS TCP/IP.

For more specific information on CS for z/OS base functions, standard applications, and security, please reference the other volumes in the series. These are:

- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 1 - Base Functions, Connectivity, and Routing*, SG24-7169
- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 2 - Standard Applications*, SG24-7170
- *Communications Server for z/OS V1R7 TCP/IP Implementation, Volume 4 - Policy-Based Network SecurityDefault*, SG24-7172

SG24-7171-00 ISBN 0738497339