

ABCs of z/OS System Programming Volume 4

SNA, TCP/IP, Hardware interfaces,
Enterprise Extender

Routing, Parallel Sysplex,
Operations

TCP/IP applications, Security



Paul Rogers

Redbooks



International Technical Support Organization

ABCs of z/OS System Programming Volume 4

February 2011

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

First Edition (February 2011)

This edition applies to version 1 release 12 modification 0 of IBM z/OS (product number 5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2011. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
The author who wrote this book	x
Now you can become a published author, too!	x
Comments welcome	x
Stay connected to IBM Redbooks	xi
Chapter 1. Introducing z/OS Communications Server	1
1.1 Communications Server overview	2
1.2 Features and benefits	4
1.3 Communications Server overview	6
1.4 Communications Server protocols	8
1.5 Communications Server functional overview	10
1.6 System z physical interfaces	12
1.7 System z connectivity overview	14
1.8 VTAM and TCP/IP in Communications Server for z/OS	16
Chapter 2. VTAM concepts for SNA networks	19
2.1 SNA implementation and concepts	20
2.2 VTAM nodes types	22
2.3 SNA subarea network	24
2.4 An APPN network	26
2.5 VTAM subarea nodes	28
2.6 Domains in a subarea network	30
2.7 Network node domains in APPN	32
2.8 Starting VTAM SNA	34
2.9 Starting VTAM	36
2.10 VTAM start options	38
2.11 VTAM start options by function	40
2.12 VTAM buffers	43
2.13 VTAM major nodes	45
2.14 Defining LAN attached devices to VTAM	47
2.15 VTAM operator commands	49
Chapter 3. TCP/IP stack	53
3.1 z/OS Communications Server overview	54
3.2 Networking on the mainframe	56
3.3 Hardware network connections	58
3.4 TCP/IP networking	60
3.5 TCP/IP stack - TCP/IP address space	62
3.6 TCP/IP terminology	63
3.7 IPv4 addressing	65
3.8 IPv6 addresses	67
3.9 Internet terms for TCP/IP	68
3.10 TCP/IP suite of protocols	70
3.11 Sending application data to the host	72
3.12 Host configuration tasks	74

3.13	TCP/IP customization	76
3.14	Configuration files used by TCP/IP	78
3.15	TCP/IP data sets search order	80
3.16	DEVICE and LINK statements	82
3.17	The resolver	84
3.18	Resolver basic concepts	86
3.19	Configuring the TCP/IP address space	87
3.20	z/OS UNIX physical file systems	89
3.21	TCP/IP family of protocols	91
3.22	Telnet protocol	93
3.23	Using sockets	95
3.24	Using TCP/IP with IMS and CICS	97
3.25	Using CICS sockets API	99
3.26	Using IMS with SNA or TCP/IP	101
Chapter 4. Network connectivity		103
4.1	System z network connectivity	105
4.2	TCP/IP and network connectivity	107
4.3	OSA Express connectivity	109
4.4	OSA-Express3 or OSA-Express2 Ethernet connectivity	111
4.5	Ethernet protocol	112
4.6	Non-QDIO versus QDIO data paths	114
4.7	Internal Queued Direct Input/Output (iQDIO)	116
4.8	HiperSockets	118
4.9	Open Systems Adapter Support Facility (OSA/SF)	119
4.10	Virtual LAN (VLAN) overview	121
4.11	OSA VLAN support	123
4.12	Dynamic XCF connectivity	125
4.13	Enterprise Extender (EE)	127
4.14	APPN/HPR functions	129
4.15	High-Performance Routing (HPR)	131
4.16	Enterprise Extender and HPR links	133
Chapter 5. IP routing		135
5.1	Network routing terminology	136
5.2	MAC addresses	138
5.3	Static and dynamic VIPA	140
5.4	Virtual IP Address (VIPA)	142
5.5	Virtual media access control (VMAC)	144
5.6	Forwarding packets to VMAC targets	146
5.7	Routing table (static or dynamic)	148
5.8	Routing IP datagrams	150
5.9	Dynamic routing using OMPROUTE	152
5.10	Sysplex distributor	154
5.11	Sysplex distributor workload balancing	156
Chapter 6. z/OS UNIX, FTP, and security		159
6.1	z/OS UNIX OMVS segment	160
6.2	TCP/IP started task user ID	162
6.3	User IDs requiring superuser authority	163
6.4	BPX.DAEMON in the FACILITY class	164
6.5	RACF program control	166
6.6	z/OS UNIX-level security for daemons	168
6.7	z/OS UNIX daemons	169

6.8 TCP/IP and z/OS	171
6.9 FTP as a network protocol	173
6.10 FTPD daemon processing	175
6.11 syslogd daemon	177
6.12 Monitoring the TCP/IP network	179
6.13 z/OS Communications Server security	181
6.14 Cryptography security	183
6.15 IPsec in the IP layer	185
6.16 SSL and TLS	187
6.17 Application transparent transport layer security	189
6.18 Kerberos	191
6.19 OSPF authentication	193
6.20 Secure DNS	195
6.21 SNMPv3	196
6.22 Protection network resources	198
6.23 Network access control	200
Related publications	203
IBM Redbooks	203
Other publications	203
Online resources	203
How to get Redbooks	203
Help from IBM	204

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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


COPYRIGHT LICENSE:

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

Trademarks

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

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

AIX®	Language Environment®	System z9®
AnyNet®	MVS™	System z®
CICS®	OS/390®	VTAM®
DB2®	Parallel Sysplex®	WebSphere®
ESCON®	pSeries®	xSeries®
eServer™	RACF®	z/Architecture®
FICON®	Redbooks®	z/OS®
GDPS®	Redbooks (logo)  ®	z/VM®
Geographically Dispersed Parallel Sysplex™	RMF™	z/VSE™
HiperSockets™	RS/6000®	z10™
IBM®	S/390®	z9®
IMS™	System p®	zSeries®
	System z10®	

The following terms are trademarks of other companies:

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

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

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

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

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

Preface

This IBM® Redbooks® publication describes the functions of z/OS® Communications Server.

z/OS Communications Server provides a set of communications protocols that support peer-to-peer connectivity functions for both local and wide-area networks, including the most popular wide-area network, the Internet. z/OS Communications Server also provides performance enhancements that can benefit a variety of TCP/IP applications.

z/OS Communications Server provides both SNA and TCP/IP networking protocols for z/OS. The SNA protocols are provided by VTAM® and include Subarea, Advanced Peer-to-Peer Networking, and High Performance Routing protocols.

z/OS Communications Server exploits z/OS UNIX® services even for traditional MVS™ environments and applications. Prior to utilizing TCP/IP services, therefore, a full-function mode z/OS UNIX environment including a Data Facility Storage Management Subsystem (DFSMSdfp), a z/OS UNIX file system, and a security product (such as Resource Access Control Facility, or RACF®) must be defined and active before z/OS Communications Server can be started successfully.

The ABCs of z/OS System Programming is a 13-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects. If you want to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful technical tool.

The contents of the volumes are as follows:

Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation

Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, SMP/E, Language Environment®

Volume 3: Introduction to DFSMS, data set basics storage management hardware and software, catalogs, and DFSMStvs

Volume 4: Communication Server, TCP/IP, and VTAM

Volume 5: Base and Parallel Sysplex®, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), Geographically Dispersed Parallel Sysplex™ (GDPS®)

Volume 6: Introduction to security, RACF, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries® firewall technologies, LDAP, and Enterprise identity mapping (EIM)

Volume 7: Printing in a z/OS environment, Infoprint Server and Infoprint Central

Volume 8: An introduction to z/OS problem diagnosis

Volume 9: z/OS UNIX System Services

Volume 10: Introduction to z/Architecture®, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and HMC

Volume 11: Capacity planning, performance management, RMF™, and SMF

Volume 12: WLM

Volume 13: JES3

The author who wrote this book

This book was produced by a technical specialist working at the International Technical Support Organization, Poughkeepsie Center.

Paul Rogers is a Consulting IT Specialist at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on various aspects of z/OS, z/OS UNIX, JES3, and Infoprint Server. Before joining the ITSO 21 years ago, Paul worked in the IBM Installation Support Center (ISC) in Greenford, England for eight years, providing OS/390® and JES support for IBM EMEA. He also worked in the Washington Systems Center for three years. Paul has been with IBM for more than 43 years.

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author - all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

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

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Introducing z/OS Communications Server

The “Communications Server” name is used for products running on various platforms: System z and zSeries, pSeries®, xSeries®, or OEM. Communications Server is currently available for z/OS, Windows®, AIX®, UNIX, and Linux® operating systems.

z/OS Communications Server provides a set of communications protocols that support peer-to-peer connectivity functions for both local and wide-area networks, including the most popular wide-area network, the Internet.

z/OS Communications Server is IBM’s implementation of SNA and standard TCP/IP protocol suites on the System z® platform. SNA is IBM’s proprietary networking protocol, whereas TCP/IP is a component product of the Communications Server for z/OS that provides a multitude of technologies that collectively provide an Open Systems environment for the development, establishment, and maintenance of applications and systems. z/OS Communications Server is a base element, meaning it is part of the z/OS operating system.

This chapter is an introduction to the basics of mainframe networking concepts, including hardware connectivity and z/OS Communications Server TCP/IP and SNA Services.

The topics covered in this chapter are:

- ▶ Basics of mainframe connectivity
- ▶ Hardware and software options
- ▶ Basic elements of a data network
- ▶ System z networking infrastructure used to attach the System z to the network

1.1 Communications Server overview

- ❑ z/OS Communications Server protocol suite supports TCP/IP and VTAM environments:
 - A native z/OS environment in which users can exploit the popular TCP/IP protocols in MVS application environments such as:
 - Batch jobs, started tasks, TSO, CICS applications, and IMS applications
 - A z/OS UNIX System Services environment that lets you create and use applications that conform to the POSIX or XPG4 standard (a UNIX specification)
 - SNA protocols are provided by VTAM and include Subarea, Advanced Peer-to-Peer Networking and High-Performance Routing protocols

Figure 1-1 Communications Server overview

Communications Server overview

z/OS Communications Server provides both SNA and TCP/IP networking protocols for z/OS. The SNA protocols are provided by VTAM and include Subarea, Advanced Peer-to-Peer Networking, and High Performance Routing protocols.

TCP/IP protocol

TCP/IP is the set of communications protocols used both for the Internet and other similar networks. It is named for two of the most important protocols, which were the first two networking protocols defined in this standard as follows:

- ▶ Transmission Control Protocol (TCP)

TCP is a transport layer protocol used by applications that require guaranteed delivery. TCP establishes a full duplex virtual connection between two endpoints. Each endpoint is defined by an IP address and a TCP port number. A byte stream is transferred in segments. A number of bytes of data is sent and an acknowledgement from the receiver is necessary.
- ▶ Internet Protocol (IP)

IP routing is a set of protocols that determine the path that data follows to travel across multiple networks from its source to its destination. Data is routed from its source to its destination through a series of routers, and across multiple networks.

z/OS UNIX System Services

z/OS UNIX System Services is the z/OS Communications Server implementation of UNIX as defined by X/Open in XPG 4.2. z/OS UNIX System Services coexists with traditional MVS functions and traditional MVS file types (partitioned data sets, sequential files, and so on). It concurrently allows access to z/OS UNIX file system files and to z/OS UNIX utilities and commands by means of application programming interfaces and the interactive shell environment.

SNA protocols and VTAM

Systems Network Architecture (SNA) is a data communication architecture established by IBM to specify common conventions for communication among the wide array of IBM hardware and software data communication products and other platforms.

Virtual Telecommunications Access Method (VTAM) controls the network and maintains a table of all the machines and phone links in the network. It selects the routes and the alternate paths that messages can take between different NCP nodes. A subarea is the collection of terminals, workstations, and phone lines managed by an NCP. Generally, the NCP is responsible for managing ordinary traffic flow within the subarea, and VTAM manages the connections and links between subareas. Any subarea network must have a mainframe.

High-Performance Routing (HPR)

HPR allows you to migrate NCP connections to APPN connections without incurring the associated increase in storage and cycles. HPR utilizes a rapid transport protocol (RTP) connection to transport session traffic between session endpoints.

Peer-to-peer networking

Earlier releases of VTAM support type 2.1 casual connection, a way of connecting two VTAMs in a peer-to-peer relationship using low-entry networking (LEN). While type 2.1 casual connection is still supported, IBM recommends that you use APPN to connect two VTAMs in a peer-to-peer relationship.

1.2 Features and benefits

- ❑ The main features of z/OS Communications Server are in the areas of:
 - High bandwidth and high speed networking technologies
 - High availability
 - Enterprise connectivity
 - System and data security
 - Network management

Figure 1-2 Features and benefits of Communications Server

Communications Server features and benefits

z/OS Communications Server provides application programming interfaces (APIs) and networking protocol support to enable SNA and TCP/IP applications running on z/OS to communicate with partner applications or users on the same system, other systems within a single data center, or in distant locations.

High speed connectivity

Communications Server provides support for high bandwidth and high speed networking technologies with:

- ▶ Gigabit Ethernet with OSA Express QDIO for TCP/IP
- ▶ High-speed communication between TCP/IP stacks running in a z/OS logical partition (LPAR) using HiperSockets™

High availability

Communications Server provides high availability to z/OS applications over both SNA and TCP/IP networks. z/OS Parallel Sysplex technology is utilized to enable high availability application support with the following:

- ▶ APPN High Performance Routing (HPR) provides end-to-end session continuity by supporting non-disruptive path switch around failed network components.
- ▶ SNA Generic Resources function provides workload balancing for multi-instance applications within a parallel Sysplex in order to maximize availability and efficiency of application data sharing.

- ▶ Multi-node Persistent Sessions provides session recovery for SNA applications.
- ▶ Dynamic VIPA (Virtual IP Address) provides TCP/IP application availability across z/OS systems in a Sysplex and allows participating TCP/IP stacks on z/OS to provide backup and recovery for each other for both planned and unplanned TCP/IP outages.
- ▶ Sysplex Distributor provides intelligent load balancing for TCP/IP application servers in a Sysplex and, along with Dynamic VIPA, provides a single system image for client applications connecting to these servers.

Enterprise connectivity

Communications Server has many features that support Enterprise Connectivity, including:

- ▶ TN3270 Server provides workstation connectivity over TCP/IP networks to access z/OS and enterprise SNA applications.
- ▶ Enterprise Extender allows SNA Enterprise applications to communicate reliably over an IP network using SNA HPR and UDP transport layer protocols.
- ▶ APPN Extended Border Node (EBN) provides a replacement for 3745/NCP-based SNA Network Interconnect (SNI).

System and data security

Network security protects sensitive data and the operation of the TCP/IP stack on z/OS, as follows:

- ▶ IPSEC/VPN functions enable the secure transfer of data over a network using standards for encryption, authentication, and data integrity.
- ▶ Intrusion Detection Services (IDS) evaluates the stack for attacks that would undermine the integrity of its operation. Events to examine and actions to take at event occurrence (for example, logging) are defined by IDS policy, which is downloaded from an LDAP server.
- ▶ TLS/SSL enablement for key applications ensures data is protected as it is flowing across the networks:
 - FTP client and server
 - TN3270 server
- ▶ Kerberos5 and GSSAPI support is provided for the following applications:
 - FTP client and server
 - UNIX rshd server
 - UNIX System Services telnet server (supports only Kerberos5)

Network management

Network management support collects network topology, status, and performance information, and makes it available to network management tools.

- ▶ VTAM supports the traditional Common Management Information Protocol (CMIP) network management protocol for managing SNA network topology and providing other management interfaces for performance monitoring.
- ▶ Managing TCP/IP, the SNMPv3 protocol is supported.
- ▶ Communications Server provides support for standard SNMP applications and IETF standard TCP/IP protocol MIBs.
- ▶ Additional MIB support is also provided by the IBM MVS TCP/IP enterprise-specific MIB, which supports management data for Communications Server TCP/IP stack-specific functions.

1.3 Communications Server overview

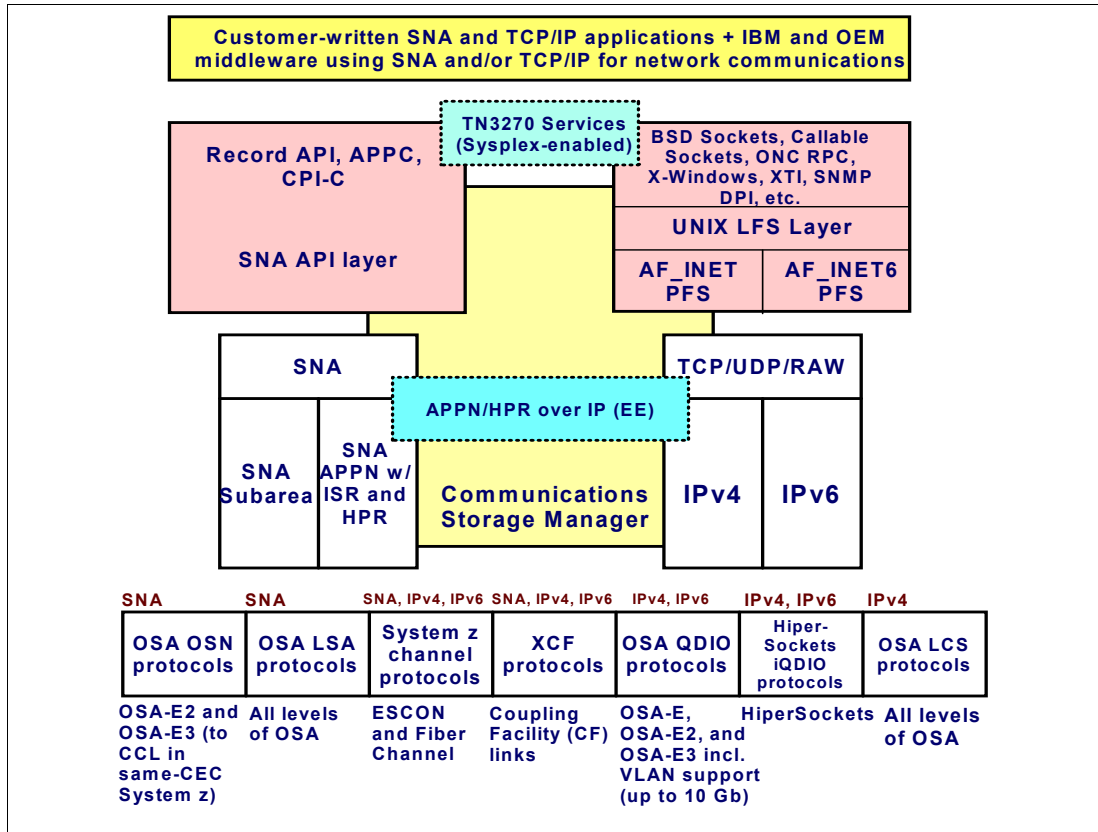


Figure 1-3 Overview of Communications Server functions

Communications Server functions

z/OS Communications Server provides a set of communications protocols that support peer-to-peer connectivity functions for both local and wide-area networks, including the most popular wide-area network, the Internet. z/OS Communications Server also provides performance enhancements that can benefit a variety of TCP/IP applications.

z/OS Communications Server provides both SNA and TCP/IP networking protocols for z/OS. The SNA protocols are provided by VTAM and include Subarea, Advanced Peer-to-Peer Networking, and High Performance Routing protocols.

Telnet TN3270

Telnet 3270, or TN3270, describes either the process of sending and receiving 3270 data streams using the Telnet protocol or the software that emulates a 3270 class terminal which communicates using that process. TN3270 allows a 3270 terminal emulator to communicate over a TCP/IP network instead of a SNA network. Standard telnet clients cannot be used as a substitute for TN3270 clients because they use fundamentally different techniques for exchanging data.

SNA API layer

The IBM LUA access method provides an application programming interface (API) for secondary dependent logical units (LUs). LUA consists of system software and interfaces that supply input/output (I/O) service routines to support communications using LU types 0, 1, 2, and 3 SNA protocols. The RUI and SLI interface of LUA is supported. The services that LUA

provides to application programs include only those that support data communications; LUA does not provide any device emulation facilities. However, LUA does provide a unique subset of presentation services layer functions.

UNIX LFS layer

The file system layer provides the main interface between the APIs and the transport layers. The first component of the file system layer is the z/OS UNIX logical file system (LFS). The LFS provides the API layer with a common interface to access files and sockets. In a POSIX-compliant environment, applications can access both files and sockets in a similar fashion. For example, both files and sockets are represented by a 32-bit integer referred to as a descriptor. Common functions can be used to access both file and socket resources. From a TCP/IP perspective, the AF_INET and the AF_INET6 PFS are of main interest. TCP/IP is enabled for IPv6 by defining an AF_INET6 PFS. Defining the file systems is the responsibility of the installation's z/OS UNIX programmer. The definitions are found in the BPXPRMxx member of SYS1.PARMLIB.

Communications storage manager

The communications storage manager (CSM) is a VTAM component that allows authorized host applications to share data with VTAM and other CSM users without the need to physically copy the data. CSM includes an API that provides a way to obtain and return CSM buffers, change ownership of buffers, copy buffers, and manage CSM buffers.

APPN/HPR over IP (EE)

Enterprise Extender (EE) used in conjunction with the APPN extended border node function can replace SNA Network Interconnection (SNI) functionality in a way that does not require SNA mature hardware, such as the 3745. High performance routing (HPR) is an addition to APPN that improves reliability, increases network performance, and was designed to exploit higher link speed technologies.

SNA protocols

z/OS Communications Server is a network communication access method. It provides both Systems Network Architecture (SNA) and Transmission Control Protocol/Internet Protocol (TCP/IP) networking protocols for z/OS. The SNA protocols are provided by VTAM and include Subarea, Advanced Peer-to-Peer Networking (APPN), and High Performance Routing (HPR) protocols. z/OS Communications Server provides the interface between application programs residing in a host processor, and resources residing in a SNA network; it also links peer users in the network.

TCP, UDP, RAW protocols

The transport layer provides the support for the TCP, UDP, and RAW protocols. All three protocols use IPv4 or IPv6 as the network layer. The TCP protocol provides a connection-oriented, reliable transport layer, whereas UDP provides a simpler, connectionless and unreliable transport layer. The RAW transport layer provides for a more direct interface to the IP layer, which is primarily used by system management type applications.

The Information Management System (IMS™) IPv4 socket interface supports development of client/server applications in which one part of the application executes on a TCP/IP-connected host and the other part executes as an IMS application program. The programming interface used by both application parts is the socket programming interface, and the communication protocols are either TCP, UDP, or RAW.

1.4 Communications Server protocols

SNA	SNA APPN	SNA, IPv4, IPv6	SNA, IPv4, IPv6	IPv4, IPv6	IPv4, IPv6	IPv4
OSA OSN protocols	OSA LSA protocols	System z channel protocols	XCF protocols	OSA QDIO protocols	Hiper-Sockets iQDIO protocols	OSA LCS protocols
OSA-E2 and OSA-E3 (to CCL in same-CEC System z)	All levels of OSA	ESCON and Fiber Channel	Coupling Facility (CF) links	OSA-E, OSA-E2, and OSA-E3 incl. VLAN support (up to 10 Gb)	HiperSockets	All levels of OSA

Figure 1-4 Protocols used with Communications Server

OSA, OSN, and LSA protocols

The Open Systems Adapter-Express3 (OSA-Express3), and OSA-Express2 features comprise a number of integrated hardware features that can be installed in a System z I/O cage, becoming integral components of the server's I/O subsystems. They provide high function, connectivity, bandwidth, data throughput, network availability, reliability, and recovery. Each OSA-Express card is defined to the processor in the Input/Output Configuration Program (IOCP).

The OSN type is only available with OSA-Express2 and requires a z9® mainframe or later model. The primary intention of this type is to free organizations from the constraints of obsolete hardware: device types 3745 and 3746. The 374x device types, as they are called, are no longer manufactured or sold by IBM. A 374x host is required to run the Network Control Program (NCP). NCP is a significant functional component of subarea type SNA networks (more information about SNA networks is covered in the topic on Systems Network Architecture basics and implementation).

The OSN channel type allows an Open Systems Adapter to communicate with an NCP using Channel Data Link Control protocol (CDLC). CDLC cannot be used over an OSD or OSE channel type, and even with channel type OSN it can only communicate to other LPARs within the CPC. Historically, 374x devices were often connected to parallel or ESCON® channels, which support CDLC.

The link-state advertisement (LSA) is a basic communication means of the OSPF routing protocol for IP. It communicates the router's local routing topology to all other local routers in

the same OSPF area. OSPF is designed for scalability, so some LSAs are not flooded out on all interfaces, but only on those that belong to the appropriate area. In this way detailed information can be kept localized, while summary information is flooded to the rest of the network.

ESCON and fiber channels

Access to TCP/IP hosts can be implemented by way of a channel-to-channel (CTC) connection established over a zSeries ESCON channel. There are two connection types: Enterprise Systems Connection (ESCON) Channel-to-Channel adapter and the Fibre Connection (FICON®) Channel-to-Channel adapter. ESCON has the channel-to-channel adapter function implemented and integrated into its licensed internal code. FICON allows a Fiber Channel channel path to function as both a channel and a channel-to-channel (CTC) simultaneously.

Coupling facility links

A sysplex is a set of MVS systems communicating and cooperating with each other, through certain multi-system hardware components and software services, to process customer workloads. The inclusion of a coupling facility within the sysplex allows for high performance data sharing.

VTAM supports the attachment of a sysplex to a network. Sessions into the sysplex can be established from either subarea nodes or APPN nodes. Several VTAM functions are available only in a sysplex environment. If a coupling facility and an APPN or mixed APPN and subarea environment exists in the sysplex, the user can take advantage of the VTAM functions. VTAM also provides support for TCP/IP functions that need access to a coupling facility.

QDIO and OSA protocols

QDIO is a highly efficient data transfer mechanism that is designed to dramatically reduce system overhead and improve throughput by using system memory queues and a signaling protocol to directly exchange data between the OSA microprocessor and network software. QDIO is the interface between the operating system and the OSA hardware. The components that make up QDIO are DMA, data router (OSA-Express3 only), Priority Queuing (z/OS only), dynamic OSA Address Table building, LPAR-to-LPAR communication, and Internet Protocol (IP) Assist functions. QDIO supports IP and non-IP traffic with the OSA-Express3 and OSA-Express2 features. These features support two transport modes: Layer 2 (Link Layer) for IP and non-IP traffic, and Layer 3 (Network Layer) for IP traffic only.

HiperSockets IQDIO protocols

The primary/secondary router function enables an OSA port to forward packets with unknown IP addresses to a TCP/IP stack for routing through another IP network interface, such as HiperSockets or another OSA feature. HiperSockets is a zSeries hardware feature that provides high performance internal communications between LPARs within the same central processor complex (CPC), without the use of any additional or external hardware equipment (for example, channel adapters, LANs, and so on).

LAN Channel Station (LCS)

LCS is the channel protocol which is supported by TCP/IP applications on the host. Each application defines a consecutive pair of sub channels. One sub channel is for TCP/IP to read from and the other one is for TCP/IP to write to. The LCS interface allows LAN MAC frames to be transported over a channel and it provides a command interface to activate, deactivate, and query the LAN interfaces. Each MAC frame has a header which identifies the virtual LAN adapter destination of the frame.

1.5 Communications Server functional overview

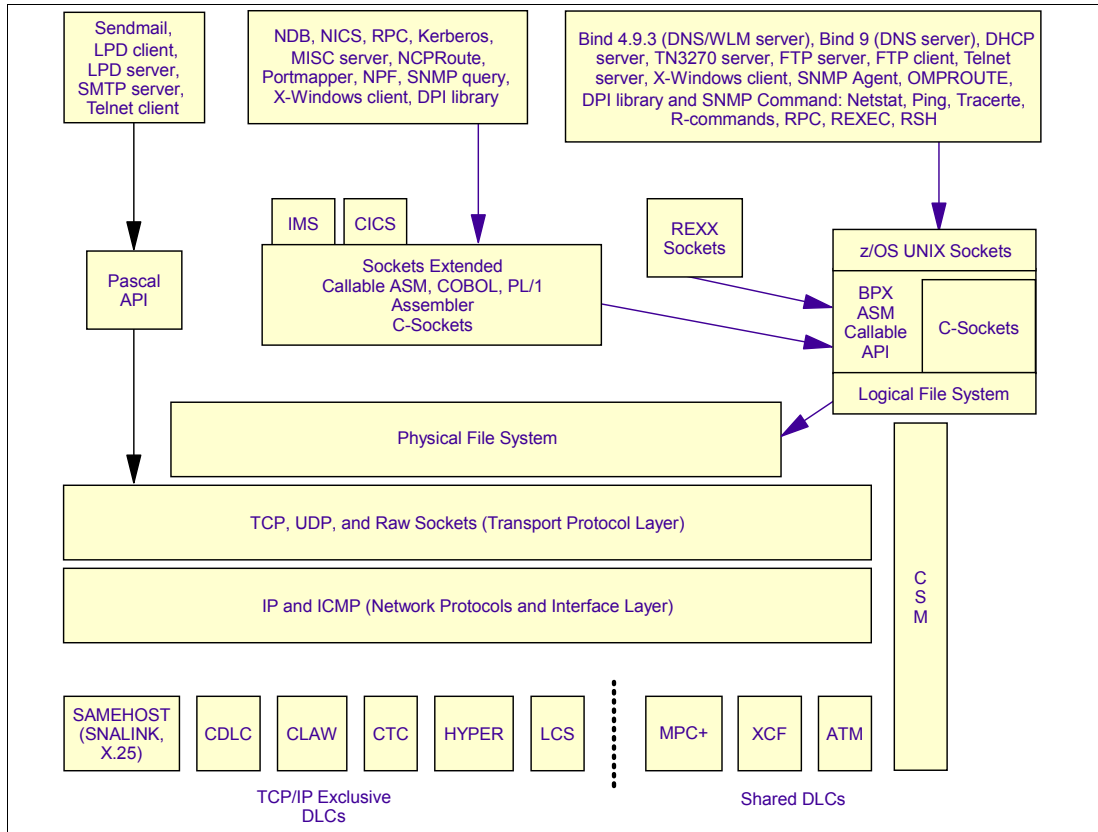


Figure 1-5 Communications Server functional overview

Functional overview

CS for z/OS IP takes advantage of Communications Storage Management (CSM) and of VTAM's Multi-Path Channel (MPC) and Queued Direct I/O (QDIO) capabilities in its TCP/IP protocol implementation. This tight coupling with VTAM provides enhanced performance and serviceability.

Communications Server for z/OS IP provides TCP/IP support for the native MVS and UNIX System Services environment. It is implemented within a z/OS address space and runs within the native MVS environment, and consequently it has RACF, DFSMS, and z/OS UNIX file system dependencies.

As illustrated in Figure 1-5, there are many data link control (DLC) protocols supported the Communications Server. These DLCs are provided by the VTAM component. In CS for z/OS IP, you have access to the z/OS UNIX environment and the traditional MVS environment.

Basic concepts

Network connectivity is handled by the physical and logical interfaces to enable transport of IP or datagrams. Using the OSI model as an example, it is Layer 1 (physical layer) and Layer 2 (Data link control Layer). The z/OS Communications Server has implemented several types of interfaces to connect to different networking environments. These environments vary from point-to-point connections (for example, MPCPTP, CTC, and CLAW) to LAN connections such as LCS and MPCIPA.

The supported interfaces are as follows:

CTC	Provides access to TCP/IP hosts by way of a channel-to-channel (CTC) connection established over a FICON or ESCON channel.
HiperSockets	HiperSockets implementation is based on the OSA-Express queued direct input/output (QDIO) protocol, hence HiperSockets is also called internal QDIO (iQDIO). The communication is through system memory of the server via I/O queues. IP traffic is transferred at memory speeds between LPARs, eliminating the I/O subsystem overhead and external network delays.
LCS	There are a variety of communications adapters that support a protocol called the LAN Channel Station (LCS). One example is the Open Systems Adapter-Express (OSA-Express).
MPCIPA	The OSA-Express features support the Queued Direct I/O (QDIO) architecture. IPv6 is supported on all Ethernet features in QDIO mode. The MPCIPA provides access to TCP/IP hosts using the following: <ul style="list-style-type: none">• For OSA-Express Gigabit Ethernet (GbE): 1000BASE-T Ethernet, Fast Ethernet, HSTR, or ATM in LAN emulation mode.• For OSA-Express2: GbE, 10 GbE, or 1000BASE-T Ethernet• OSA-Express3: GbE or 10 GbE HiperSockets. Using the Internal Queued Direct I/O (iQDIO) provides high-speed, low-latency IP message passing between logical partitions (LPARs) within a single IBM eServer™ zSeries z800, z890, z900, z990, z9, or z10™ server.
MPCPTP	Provides access to TCP/IP hosts through Multipath Channel Point-to-Point (MPCPTP) links. MPCPTP is used to directly connect to other hosts or z/OS LPARs, or by configuring it to utilize Coupling Facility links (if the z/OS LPAR is part of a sysplex).
SAMEHOST	The SAMEHOST Data Link Control (DLC) enables communication between Communications Server for z/OS IP and other servers running on the same MVS image. The configuration and control of these interfaces is provided by VTAM. They are enabled in VTAM as TRLE minor nodes. For most, their definitions are created by VTAM dynamically and are transparent to the application. The SAMEHOST Data Link Control (DLC) provides support for the SNA backbone network (SNALINK LU0 and SNALINK LU6.2) and the X.25 network.
ATM	Enables TCP/IP to send data to an asynchronous transfer mode (ATM) network using an OSA-2 or OSA-Express ATM adapter over an ATM virtual circuit.
CDLC	ESCON attachments can be used to provide native IP transport, using the channel data link control (CDLC), between the 3746 IP and host systems running the z/OS Communications Server. The host systems can be directly attached to the 3746-9x0 or connected via an ESCON director (ESCD).
CLAW	Provides access from IBM RS/6000® workstations directly to TCP/IP hosts over a channel. The CLAW (common link access to workstation) interface can also be used to provide connectivity to the original equipment manufacturer (OEM), such as the Cisco Channel Interface Processor (CIP).
HYPERchannel	Provides access to TCP/IP hosts by way of HYPERchannel series A devices and series DX devices that function as series A devices.

See “System z physical interfaces” on page 12 for a further view of these items and how they can be connected together.

1.6 System z physical interfaces

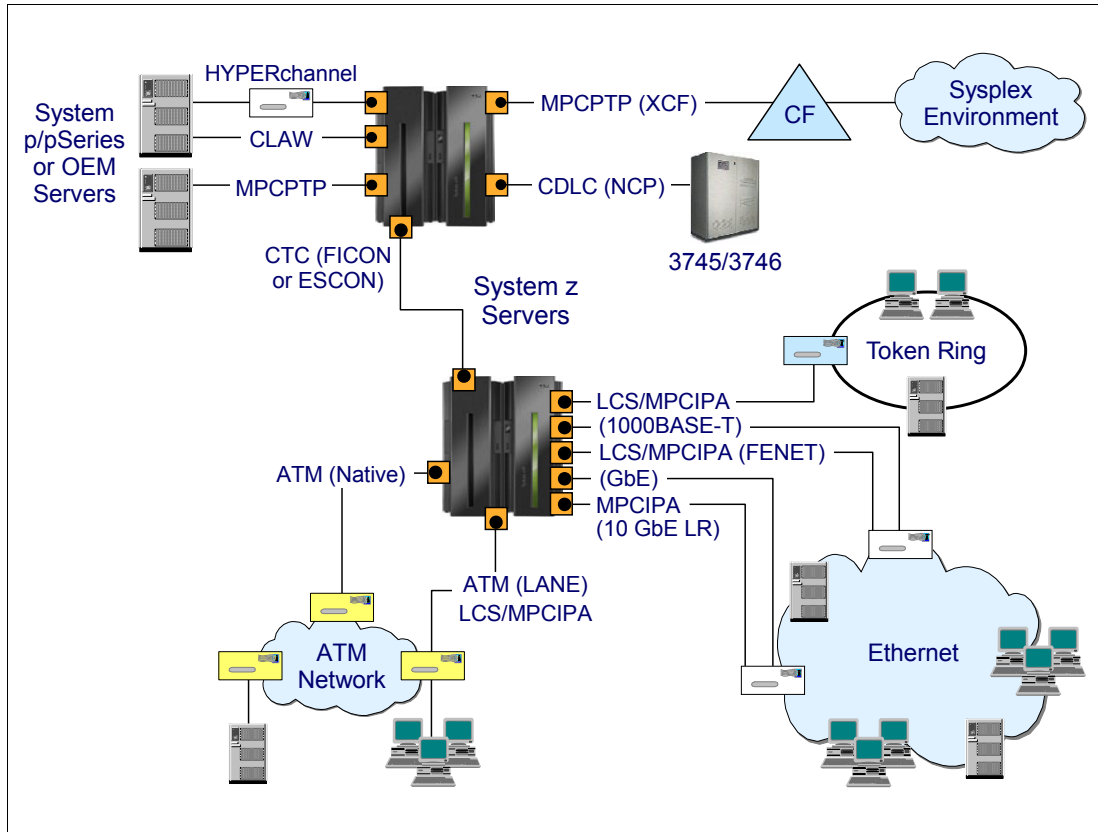


Figure 1-6 System z physical interfaces with Communications Server

System z connectivity with Communications Server

Connectivity is the pipeline through which data is exchanged between clients and servers via physical and logical communication interfaces and the network. The IBM System z servers provide a wide range of interface options for connecting your z/OS system to an IP network or to another IP host. Some interfaces offer point-to-point or point-to-multipoint connectivity, while others support Local Area Network (LAN) connectivity. Figure 1-6 depicts the physical interfaces (and device types) provided by System z servers. The physical network interface is enabled through z/OS Communications Server (TCP/IP) definitions.

Ethernet

Ethernet is a family of frame-based computer networking technologies for local area networks (LANs). It defines a number of wiring and signaling standards for the physical layer of the OSI networking model. Ethernet is standardized as IEEE 802.3. The combination of the twisted pair versions of Ethernet for connecting end systems to the network, along with the fiber optic versions for site backbones, is the most widespread wired LAN technology. It has been in use since 1980, largely replacing competing LAN standards such as token ring, FDDI, and ARCNET.

Note: Ethernet is fast becoming the fabric of choice for local area networks. The 10 Gigabit Ethernet technology with several available physical layers has gained wide acceptance, while higher speed solutions such as 40G and 100G Ethernet are in the works.

ATM connections

ATM is a switching technology that provides fast, reliable, simultaneous transfer of data, voice, and video. VTAM supports ATM technology by enabling communication across ATM networks – both public (wide area networks, or WANs) and private (campus networks) – accessed through:

- ▶ **ATM LAN emulation connections:** ATM networks accessed through LAN emulation are attached to VTAM through an IBM Open Systems Adapter (configured in SNA mode). They appear to VTAM as though they are Ethernet or Ethernet-type LANs or token-ring networks. ATM LAN emulation connections offer the benefits of fast data transfer, but they do not offer the full services of ATM networks, such as guaranteed bandwidth and Quality of Service.

ATM LAN emulation connections through an IBM Open Systems Adapter are defined to VTAM in the same way LAN connections through an IBM 3172 Nways Interconnect Controller are defined. To define ATM LAN emulation connections through the IBM Open Systems Adapter, apply the information about defining LAN connections through the IBM 3172 Nways Interconnect Controller.

- ▶ **ATM native connections:** ATM networks accessed through native ATM are also attached to VTAM through an IBM Open Systems Adapter (configured in HPDT ATM native mode). They offer the full services of ATM networks, such as guaranteed bandwidth and Quality of Service.

Token rings

Token ring local area network technology is a local area network protocol that resides at the data link layer (DLL) of the OSI model. It uses a special three-byte frame called a *token* that travels around the ring. Token ring frames travel completely around the loop.

CTC connections with FICON and ESCON

FICON provides all the strengths of ESCON, plus more. The link data rate is 4 Gbps, with an expected effective data transfer rate of up to 350 MBps (full-duplex data transfer and large sequential read/write mix). The System z servers build on this I/O architecture by offering high speed FICON connectivity.

The Enterprise Systems Connection (ESCON) channel is a high-bandwidth host attachment facility that can be used to connect SNA and non-SNA 3174 controllers, 3172 Nways interconnect controllers, 3746-900 controllers, and channel-attached hosts to VTAM running on MVS/ESA.

ESCON provides bi-directional serial bit transmission, in which the communication protocol is implemented through sequences of special control characters and through formatted frames of characters. ESCON utilizes fiber optic cables for data transmission. The ESCON link data rate is 200 megabits per second (Mbps), which results in a maximum aggregate data rate of up to 17 megabytes per second (MBps). The maximum unrepeated distance of an ESCON link using multimode fiber optic cables is 3 km (1.86 miles) when using 62.5 micron fiber.

1.7 System z connectivity overview

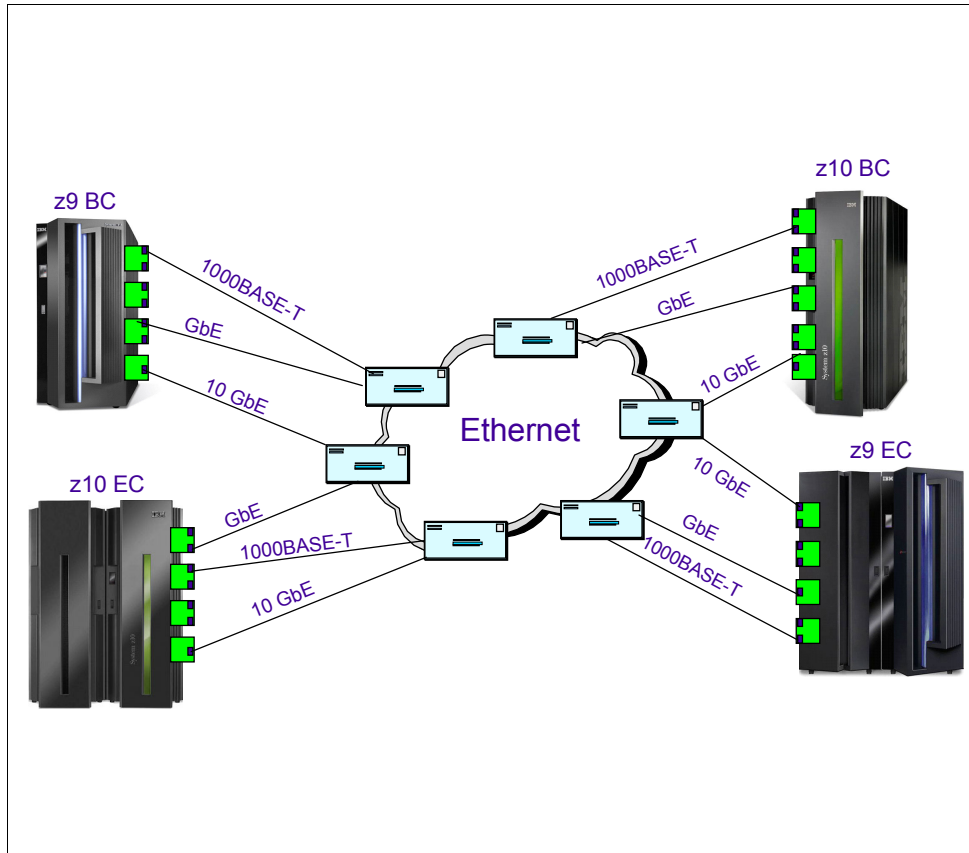


Figure 1-7 System z connectivity overview

System z connectivity overview

Basic elements of a computer network include hardware, software, and protocols. The inter-relationship of these basic elements constitutes the infrastructure of the network. If we think of a network as roads, highways, rails, and other means of transport, the network protocols are the traffic rules.

Connectivity is the pipeline through which data is exchanged between clients and servers via physical and logical communication interfaces and the network. The IBM System z and zSeries servers provide a wide range of interface options for connecting your z/OS system to an IP network or to another IP host. Some interfaces offer point-to-point or point-to-multipoint connectivity, while others support local area network connectivity. Figure 1-7 depicts the physical interfaces (and device types) provided by System z9® and System z10® zSeries servers. The physical network interface is enabled through z/OS Communications Server (TCP/IP) definitions.

A network infrastructure is the topology in which the nodes of a local area network (LAN) or a wide area network (WAN) are connected to each other. These connections involve equipment like routers, switches, bridges, and hubs using cables (copper, fiber, and so on) or wireless technologies (Wi-Fi).

Network protocols

The network protocols define how two devices in the network communicate. The specification of the network protocols starts with the electrical specifications of how a networking device is connected to the infrastructure. For example, line voltage levels, carrier signals, and the designation of which line can be used for what types of signals must all be specified. Building up from there, network protocols include such specifications as the methods that can be used to control congestion in the network and how application programs will communicate and exchange data.

A popular method of documenting network protocols is to use a layered network architecture model. Network architecture models separate specific functions into layers, which collectively form a network stack. While a protocol consists of rules that define characteristics for transporting data between network nodes, the layered model separates the end-to-end communication into specific functions performed within each layer. Ideally, the layers are isolated from each other: each layer doesn't need to know how the layer below it functions. All a layer needs to know is how to interact with the layers adjacent to it.

TCP/IP networking protocol

Today, TCP/IP is by far the most dominant suite of networking protocols. Prior to TCP/IP, SNA was arguably the dominant protocol suite. There is some irony here, because TCP/IP is the older of the two protocols. Even today, SNA powers a majority of the financial transactions that traverse the Web, and most of the airline and hotel reservations booked over the Web rely on SNA. Many networks in larger organizations are using both of these protocol suites. As with most networking protocols, both SNA and TCP/IP are layered protocol stacks.

VTAM protocol

VTAM supports the following types of LANs through the IBM Open Systems Adapter (configured in SNA mode):

- ▶ Ethernet or Ethernet-type LAN
- ▶ Token-ring network
- ▶ Fiber distributed data interface (FDDI)

1.8 VTAM and TCP/IP in Communications Server for z/OS

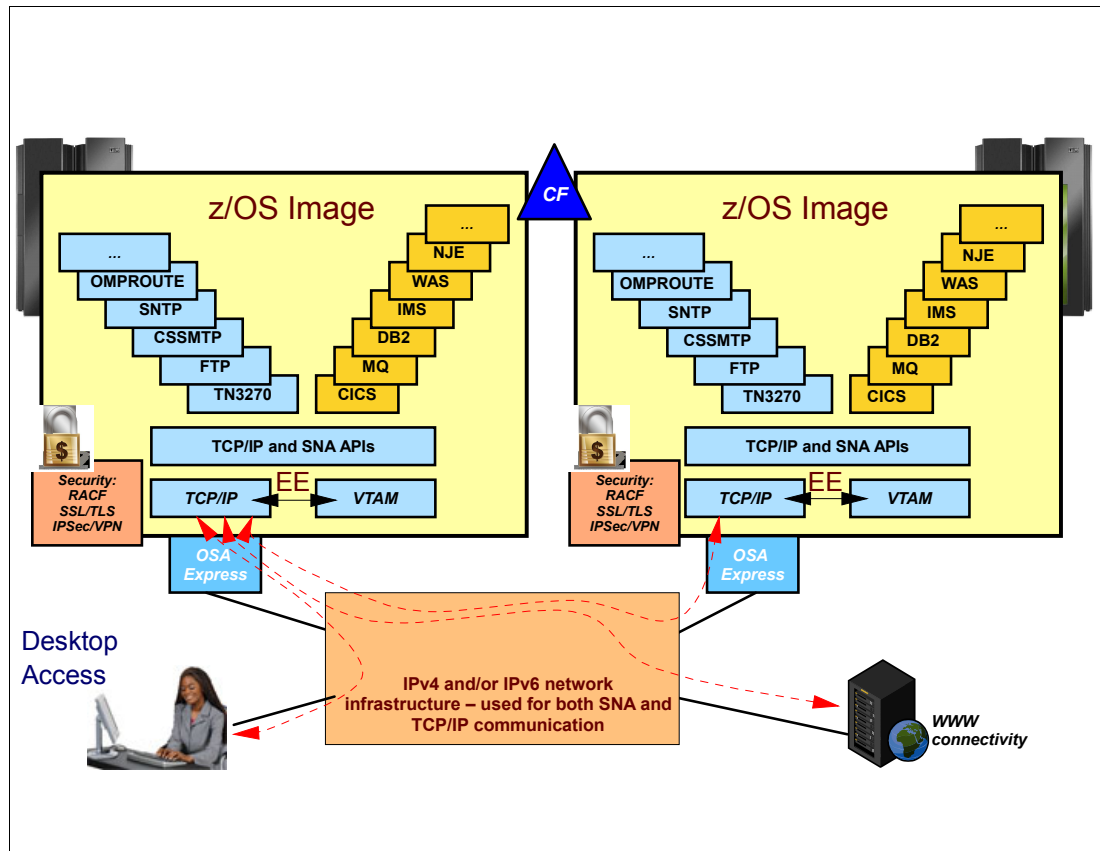


Figure 1-8 Communications Server components

VTAM and TCP/IP in Communications Server for z/OS

VTAM and TCP/IP for MVS were combined into a single product (Communications Server) in 1996. VTAM follows SNA protocols and uses SNA concepts to connect and communicate with elements in a data communication network. Each element in a SNA network to which a data or control message can be sent is assigned a network address. Each element with such an address is known as a network accessible unit (NAU). The network address uniquely identifies the element and contains the information necessary to route a message to its destination.

SNA concepts

In the past, a mainframe backbone network used SNA. With the prevalence of TCP/IP and the introduction of SNA/IP integration technology and additional tools, current mainframe networks are migrating to IP-based networks.

SNA was developed by IBM for the business community. SNA provided industry with a technology that permitted unparalleled business opportunities. What TCP/IP and the Internet were to the public in the 1990s, SNA was to large enterprises in the 1980s. TCP/IP was widely embraced when the Internet came of age because it permitted access to remote data and processing for a relatively small cost. TCP/IP and the Internet resulted in a proliferation of small computers and communications equipment for chat, email, conducting business, and downloading and uploading data.

Large SNA enterprises have recognized the increased business potential of expanding the reach of SNA-hosted data and applications to this proliferation of small computers and communications equipment in the customers' homes and small offices. FTP is one of the most widely used TCP/IP applications on z/OS. Both an FTP client and an FTP server are included as part of the base z/OS Communications Server functions. The FTP functions on z/OS support both traditional MVS data sets and files in the hierarchical file system.

Enterprise Extender (EE)

With Enterprise Extender (EE) you can extend the reach of SNA applications and data to include TCP/IP networks and IP-attached clients with levels of reliability, scalability, and control similar to those that SNA users have used. EE integration uses standard IP technology and does not require new hardware or new software in the IP backbone.

TCP/IP networking

TCP/IP is a set of protocols and applications that enable you to perform certain computer functions in a similar manner independent of the types of computers or networks being used. When you use TCP/IP, you are using a network of computers to communicate with other users, share data with each other, and share the processing resources of the computers connected to the TCP/IP network.

The following functions are part of transmitting information throughout a networking environment:

OMPROUTE	OMPROUTE is a dynamic routing daemon for z/OS.
FTP	The File Transfer Protocol (FTP) allows a user to copy files from one machine to another. The protocol allows for data transfer between the client (the end user) and the server in either direction. In addition to copying files, the client can issue FTP commands to the server to manipulate the underlying file system of the server (for example, to create or delete directories, delete files, rename existing files, and so on.) FTP is the most frequently used TCP/IP application for moving files between computers.
CSSMTP	The Communication Server SMTP (CSSMTP) application sends mail messages from a JES spool data set to an SMTP server.
SNTP	Simple Network Time Protocol (SNTP) is a protocol for synchronizing clocks across a WAN or LAN through a specific formatted message.
TN3270	In a TCP/IP network, remote users that have the Telnet 3270 client function access TSO/E by entering the TN3270 command. Telnet is a terminal emulation protocol. With Telnet, end users can log on to remote host applications as though they were directly attached to that host.
IPv4	Every computer system and device connected to the Internet is located by an IP address. IPv4 is the most widely used version of the Internet Protocol. It defines IP addresses in a 32-bit format, which looks like 123.123.123.123. IPv4 is version 4 of the Internet Protocol (IP). It was the first version of the Internet Protocol to be widely deployed, and forms the basis for the current (as of 2004) Internet.
IPv6	The first system of distributing IP addresses is called IPv4. However, with the growth of computers connected to the Internet, the number of available IP addresses are predicted to run out in only a few years. This is why IPv6 was introduced. IPv6 allows for over 340,000,000,000,000,000,000,000,000,000,000,000,000,000,000 IP addresses.



VTAM concepts for SNA networks

This chapter covers the definition, maintenance, and operation of a SNA System z network using the Virtual Telecommunications Access Method (VTAM) component of the Communications Server for z/OS.

The topics in this chapter will help you to understand:

- ▶ VTAM concepts for a subarea and Advanced Peer-to-Peer Networking (APPN) environments
- ▶ VTAM functions
- ▶ VTAM nodes
- ▶ Network accessible units (NAU)
- ▶ Routing in subarea and APPN networks
- ▶ How to start VTAM
- ▶ How to define the SNA network to VTAM
- ▶ Coding VTAM major nodes
- ▶ How to operate a SNA network
- ▶ Problem determination and traces

2.1 SNA implementation and concepts

- ❑ SNA originally consisted of subarea protocols
- ❑ Advanced Peer-to-Peer networking introduced mid 1980s
- ❑ High Performance Routing introduced in 1990s
- ❑ Enterprise Extender (HPR over UDP) introduced in 1999
- ❑ Subarea SNA is a hierarchical architecture - 2 types of nodes:
 - Subarea nodes - Provide services for and control for peripheral nodes
 - Host nodes (Type 5) - VTAM on System Z
 - Communication Controllers - NCP on 3725/3745/CCL
 - Peripheral nodes - terminals, workstations, cluster controllers

Figure 2-1 SNA concepts

SNA concepts

In 1974, IBM introduced its Systems Network Architecture (SNA), which is a set of protocols and services enabling communication between host computers (IBM mainframes) and peripheral nodes, such as IBM's dedicated hardware boxes, the 3174 controller for 3270 type displays and printers, controllers for the retail and finance industries, and more. The mainframe subsystem that implements SNA was named Virtual Telecommunication Access Method (VTAM). SNA is a data communication architecture established by IBM to specify common conventions for communication among the wide array of IBM hardware and software data communication products and other platforms. Among the platforms that implement SNA in addition to mainframes are IBM's Communications Server on Windows, AIX, and Linux, Microsoft®'s Host Integration Server (HIS) for Windows, and many more.

SNA defines the standards, protocols, and functions used by devices from mainframes to terminals that enable them to communicate with each other in SNA networks. SNA functions are divided into a hierarchical structure of separate layers, each performing a specific set of functions. This division of network functions into layers enables network devices to share information and processing resources without having detailed information about each device on the network. A user at a workstation can communicate with another user without knowing anything about the physical devices on the network or the connections between those devices.

IBM introduced new technologies to help businesses preserve the investment in SNA and use IP as the protocol for connecting SNA computers. The technology is known as SNA/IP ("SNA over IP"). The two endpoints, the SNA application in the mainframe and the SNA application

in the remote location (branch, store, and so forth), remain unchanged, thereby preserving the investment in SNA.

SNA supports the following types of networks:

- ▶ A subarea network is a hierarchically organized network consisting of subarea nodes and peripheral nodes. Subarea nodes, such as hosts and communication controllers, handle general network routing. Peripheral nodes, such as terminals, attach to the network without awareness of general network routing.
- ▶ A peer network is a cooperatively organized network consisting of peer nodes that all participate in general network routing.
- ▶ A mixed network is a network that supports both host-controlled communications and peer communications.

Advanced Peer-to-Peer networking (APPN)

APPN is dynamic in nature and reduces the amount of predefinition required in a SNA subarea network. APPN is the IBM strategic SNA protocol in the mainframe. It is required for sysplex, Enterprise Extender (EE) implementation, and many other technologies. APPN/HPR was introduced in the mid-1990s and supports non-disruptive route switching for failure recovery and connectionless intermediate routing of packets. APPN/HPR still maintains the class of service (CoS) and data flow control using a more advanced pacing mechanism, adaptive session pacing. In contrast to subarea networking, where special hardware and software (VTAM in the mainframe and the network control program in the 3745) are required for intermediate session routing, every node that can act as a network node can perform routing of SNA packets.

High performance routing (HPR)

High performance routing (HPR) allows you to migrate NCP connections to APPN connections without incurring the associated increase in storage and cycles. HPR utilizes a rapid transport protocol (RTP) connection to transport session traffic between session endpoints. HPR routes can also traverse an existing subarea network because HPR support provides for the mapping of HPR routes over virtual route-based transmission groups.

SNA subarea networking

In the mid-1980s, SNA subarea networking was the dominant networking protocol used for data processing. Its robustness, management tools, and predictable response time attracted many organizations to SNA, and they used SNA in their mission-critical applications. The major drawback of SNA subarea network was the requirement to provide static definitions for most SNA resources.

VTAM connections

VTAM attaches to LANs through an IBM 3172 Nways Interconnect Controller, an Open Systems Adapter, or through an IP network for Enterprise Extender. VTAM attaches to ATM networks through LAN emulation access through an Open Systems Adapter. VTAM can communicate with the following types of nodes over LAN and ATM LAN emulation connections:

- ▶ Subarea nodes, which are nodes with a PU type of 4 or 5
- ▶ Peripheral nodes, which are nodes with a PU type of 2 or 2.1

2.2 VTAM nodes types

- APPN only
- Subarea only
- APPN and subarea
 - Intrerchange node
 - Migration data host
 - Composite network node
 - Low Entry Networking (LEN) node

Figure 2-2 VTAM node types

VTAM nodes

A VTAM APPN node is a SNA type 2.1 node that functions in a peer-to-peer environment. APPN nodes use APPN or low-entry network (LEN) connections to communicate with other nodes in an APPN network. Each APPN node has a control point (CP) that manages the node and its resources. Control points provide function for the APPN network similar to the function the SSCP provides for the subarea network. The CP can implement end node or network node functions.

Subarea function only

A VTAM subarea node is a SNA type 5 node that functions in a hierarchical environment. Subarea nodes provide services for and control over peripheral nodes. Peripheral nodes require the services of a VTAM subarea node to communicate with other peripheral nodes and subarea nodes in the network. Peripheral nodes include SNA type 2.1, 2.0, or type 1 nodes and function as distributed processors, cluster controllers, or workstations. Type 2.1 nodes can use LEN or APPN connections to communicate with adjacent type 2.1 nodes; however, when attaching to a VTAM subarea node, they are restricted to using LEN connections.

APPN and subarea function

Nodes that need to be able to directly connect to both subarea and APPN networks require both subarea and APPN function.

These node types include:

- ▶ **Interchange node:** An interchange node resides on the border of an APPN network and a subarea network. It provides protocol conversion between subarea and APPN networks to enable the integration of the two types of networks. Because an interchange node can convert session requests from one protocol to the other and can provide intermediate routing, it can establish sessions from one type of network to the other. An interchange node combines the function of a subarea node and a network node. It controls resources and functions as a network node in the APPN network and as an SSCP and a cross-domain resource manager (CDRM) in the subarea network. All of the characteristics described for network nodes and subarea nodes apply to interchange nodes. The interchange node communicates network control data by using SSCP-SSCP sessions with other subarea nodes and CP-CP sessions with other APPN nodes. To enable it to participate in the subarea network, it is defined with a unique subarea number and requires subarea path definition statements. It can be connected to other APPN nodes, LEN nodes, and subarea nodes.
- ▶ **Migration data host:** A migration data host resides on the periphery of a combined APPN and subarea network. A migration data host combines the function of an end node with the function and role of a subarea data host. The migration data host communicates network control data by using SSCP-SSCP sessions with other subarea nodes and CP-CP sessions with its network node server. It can be connected to other APPN nodes, LEN nodes, and subarea nodes. A migration data host can perform intermediate routing in either a subarea network or an APPN network, but it is not recommended. However, its locally attached LEN nodes and dependent LUs can establish sessions with resources in other APPN or subarea domains.
- ▶ **Composite network node:** A composite network node (CNN) is composed of a VTAM interchange node and any NCPs that it owns. In an APPN network, it functions as a network node and appears to the APPN network as a single node. A composite network node is defined by coding the HOSTSA start option, the NODETYPE start option as NN, and by activating an NCP from that VTAM. The composite network node can be attached to other APPN nodes and also to subarea nodes. In a composite network node where all external connections are APPN connections, the HOSTSA value is not used in communication outside the node. If the composite node is connected to other nodes by subarea connections, the HOSTSA value is used in communication with the subarea network.
- ▶ **Low-entry networking node (LEN):** A low-entry networking node is a type 2.1 node that can function in both a subarea and an APPN environment. VTAM and any NCPs that it owns can present the image of a composite LEN node. It appears as a single node to all LEN or APPN nodes to which it is attached. A LEN node provides peer-to-peer connectivity to subarea nodes, end nodes, or network nodes that are providing a LEN appearance. Unlike end nodes, the LEN node cannot establish CP-CP sessions with a network node. Therefore, a LEN node cannot automatically register its resources with a network node server. A VTAM acting as a LEN node cannot establish SSCP-SSCP sessions across T2.1 connections with a VTAM subarea node. A LEN node can attach to a subarea network as a peripheral node. In a subarea network, LEN nodes can communicate with other LEN nodes without requiring the services of an SSCP. In an APPN network, you can predefine the CP name of a network node adjacent to a LEN node, which enables the LEN node to send a BIND to the adjacent node. The adjacent node determines the actual location of the LU, calculates the route to it, and forwards the BIND.

The type you choose depends on the level of function required for the node and whether the node needs to communicate with other subarea nodes, other APPN nodes, or both.

2.3 SNA subarea network

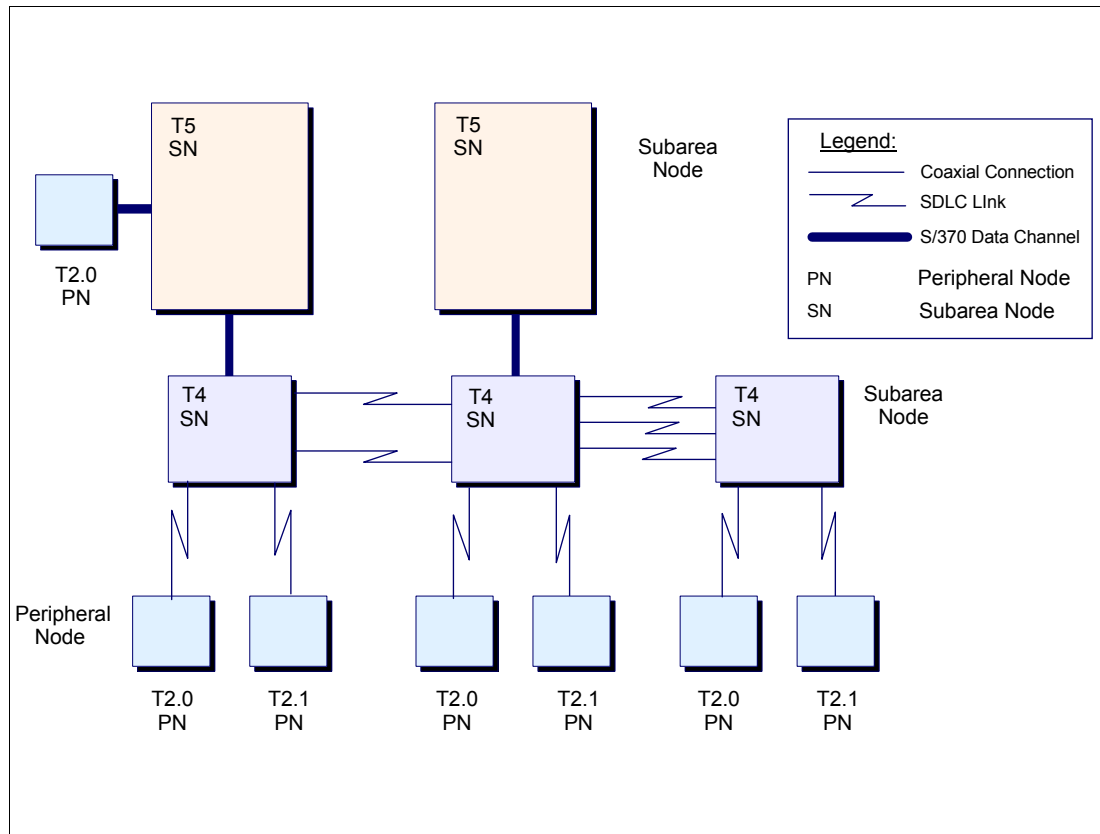


Figure 2-3 Subarea network

Subarea networks

The initial implementation of a SNA network by IBM was a SNA subarea network. This network is a hierarchical network implemented by VTAM in the Series z. VTAM is the software that controls data transfer between channels and OSA LAN-attached devices and performs SNA routing functions. VTAM provides an API that enables the development of application programs that communicate using SNA with remote application programs or devices. VTAM resides in the mainframe and supports a wide variety of network protocols, like SDLC and LAN.

Coaxial cable connection

Coaxial cable is the kind of copper cable widely installed for use in business and corporation Ethernet and other types of local area networks. A coaxial cable is called "coaxial" because it includes one physical channel that carries the signal, surrounded (after a layer of insulation) by another concentric physical channel, both running along the same axis.

SDLC link

Synchronous Data Link Control (SDLC) is a computer communications protocol. It is the layer 2 protocol for SNA. SDLC supports multipoint links as well as error correction. It also runs under the assumption that a SNA header is present after the SDLC header. SDLC was mainly used by IBM mainframe and midrange systems; however, implementations exist on many platforms from many vendors. The use of SDLC (and SNA) is becoming less common, mostly

replaced by IP-based protocols or being tunneled through IP (using AnyNet® or other technologies).

S/370 data channels

Channel Data Link Control (CDLC) is the protocol used today by mainframe host operating systems to communicate with an NCP running in an IBM 3745 Communication Controller over ESCON channel hardware. CCL running on System z hardware uses CDLC emulation to provide a more efficient connectivity alternative to SNA hosts running in the same System z9. No ESCON channel hardware is required; instead, an OSA-Express2 port is used.

Peripheral nodes

Peripheral nodes use local addresses for routing and require boundary function assistance from VTAM or the NCP. Logical units (LUs) are the ports through which end users access the network. The type 1 and type 2 peripheral node architecture supports dependent LUs only. The type 2.1 peripheral node architecture supports independent and dependent LUs. Type 2.1 peripheral nodes can communicate with each other, as well as with application programs in VTAM. Communication between type 2.1 peripheral nodes can occur through the main communication path among the subarea nodes in the network.

Subarea node types

A data communication network can be described as a configuration of nodes and links. Nodes are the network components that send data over, and receive data from, the network. Node implementations include processors, controllers, and workstations. Links are the network components that connect adjacent nodes. Nodes and links work together in transferring data through a network.

A node is a set of hardware and associated software components that implement the functions of the seven architectural layers. Although all seven layers are implemented within a given node, nodes can differ based on their architectural components and the sets of functional capabilities they implement. Nodes with different architectural components represent different node types. Four types of nodes exist: type 5 (T5), type 4 (T4), type 2.0 (T2.0), and type 2.1 (T2.1).

Type 5 (T5) and type 4 (T4) nodes can act as subarea nodes. T5 subarea nodes provide the SNA functions that control network resources, support transaction programs, support network operators, and provide end-user services. Because these functions are provided by host processors, T5 nodes are also referred to as *host nodes*.

T4 subarea nodes provide the SNA functions that route and control the flow of data in a subarea network. Because these functions are provided by communication controllers, T4 nodes are also referred to as *communication controller nodes*.

Type 2.0 (T2.0) and type 2.1 (T2.1) nodes can act as peripheral nodes attached to either T4 or T5 subarea nodes. Peripheral nodes are typically devices such as distributed processors, cluster controllers, or workstations. A T2.1 node differs from a T2.0 node in the T2.1 node's ability to support peer-oriented protocols as well as the hierarchical protocols of a simple T2.0 node. A T2.0 node requires the mediation of a T5 node to communicate with any other node. Subarea nodes to which peripheral nodes are attached perform a boundary function and act as subarea boundary nodes.

2.4 An APPN network

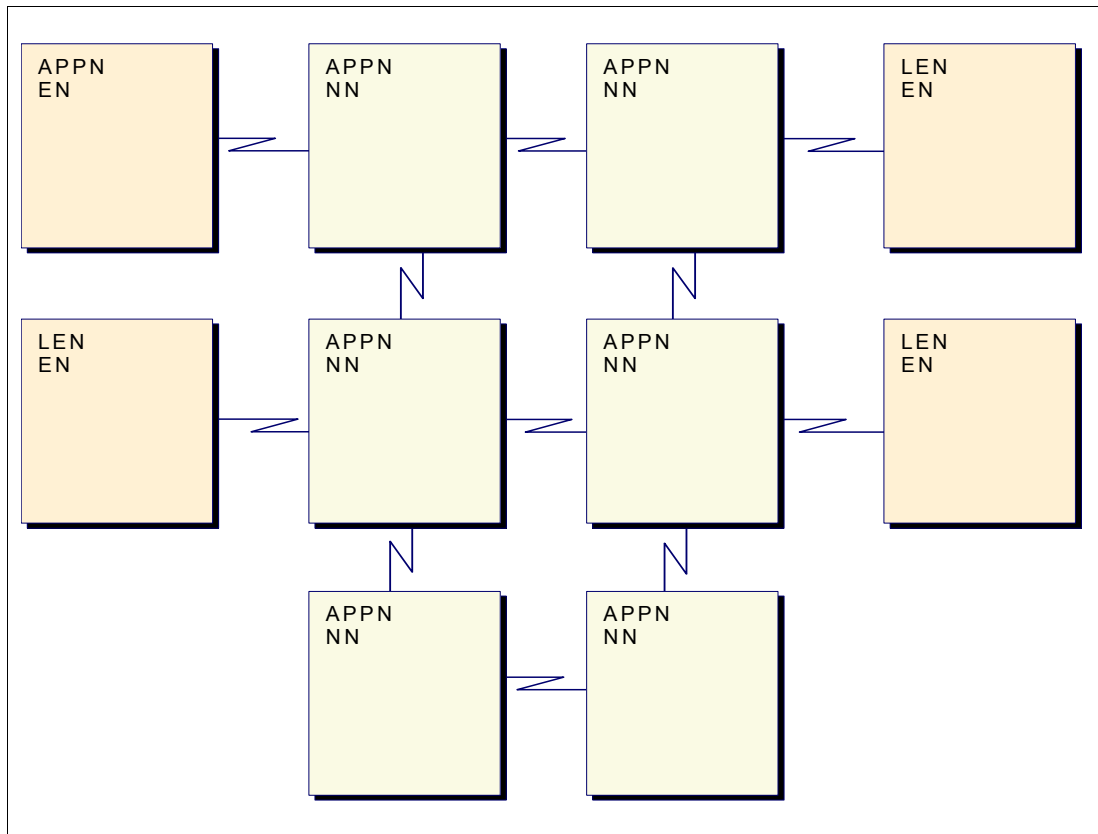


Figure 2-4 APPN network

APPN networking

An Advanced Peer-to-Peer Networking (APPN) network is composed of a group or groups of connected T2.1 nodes. T2.1 nodes provide sessions between LUs and peer-level connectivity using the APPC protocol. Unlike hierarchical SNA subarea networks, sessions between two APPN or low-entry networking nodes can be established without involving a mainframe in the session setup.

APPN nodes:

- ▶ Search and keep track of SNA resource locations within the network
- ▶ Dynamically exchange information about the resource location they own, eliminating the need to pre-define resources owned by other nodes
- ▶ Maintain knowledge of APPN network topology (the APPN nodes and the links between them) and use this information to select the best available path to route sessions between SNA resources, thereby eliminating the need for complex path definitions

An APPN network

APPN extensions allow greater distribution of network control by enhancing the dynamic capabilities of the node. Nodes with these extensions are referred to as APPN nodes, and a network of APPN nodes makes up an APPN network. A low-entry networking (LEN) node can also attach to an APPN network. An APPN node can dynamically find the location of a partner node, place the location information in directories, compute potential routes to the partner, and select the best route from among those computed. These dynamic capabilities

relieve network personnel from having to predefine those locations, directory entries, and routes. APPN nodes can include processors of varying sizes. Peer-oriented protocols enable nodes to communicate without requiring mediation by a T5 node, giving them increased connection flexibility. APPN defines two possible roles for a node in an APPN network, that of an end node and that of a network node.

T2.1 nodes can act as either APPN or LEN nodes. T5 nodes can also act as APPN or LEN nodes, but have additional capability to interconnect subarea and APPN networks by interchanging protocols between them. (In this capacity, they are called *interchange nodes*). Together with its subordinate T4 nodes, a T5 node can also form a composite LEN node or a composite network node. As composite nodes, they appear as single LEN or network nodes to other LEN or APPN nodes to which they are interconnected.

LEN end nodes

LEN end nodes are located on the periphery of an APPN network. An end node obtains full access to the APPN network through one of the network nodes to which it is directly attached – its network node server. The two kinds of end nodes are APPN end nodes and LEN end nodes. An APPN end node supports APPN protocols through explicit interactions with a network node server. Such protocols support dynamic searching for resources and provide resource information for the calculation of routes by network nodes. A LEN end node is a LEN node attached to a network node. Although LEN nodes lack the APPN extensions, they can be supported in APPN networks using the services provided them by network nodes. In an APPN network, when a LEN node is connected to another LEN node, or to an APPN end node, it is referred to simply as a LEN node. When connected to an APPN network node, however, it is referred to as a LEN end node.

Network nodes

Together with the links interconnecting them, network nodes form the intermediate routing network of an APPN network. Network nodes connect end nodes to the network and provide resource location and route selection services for them. Routes used to interconnect network users are selected based on network topology information that can change dynamically. Figure 2-4 represents one possible APPN network configuration and contains LEN end nodes as well as APPN nodes.

Network node server

This is a network node that provides resource location and route selection services to the LUs it serves. These LUs can be in the network node itself, or in the client end nodes. A network node server uses CP-CP sessions to provide network information for session setup to support the LUs on served APPN end nodes. In addition, LEN end nodes can also take advantage of the services of the network node server. A LEN end node, unlike an APPN end node, must be predefined by the network operator as a client end node for which the network node acts as server. Any network node can be a network node server for end nodes that are attached to it. The served end nodes are defined as being in that network node server's domain.

2.5 VTAM subarea nodes

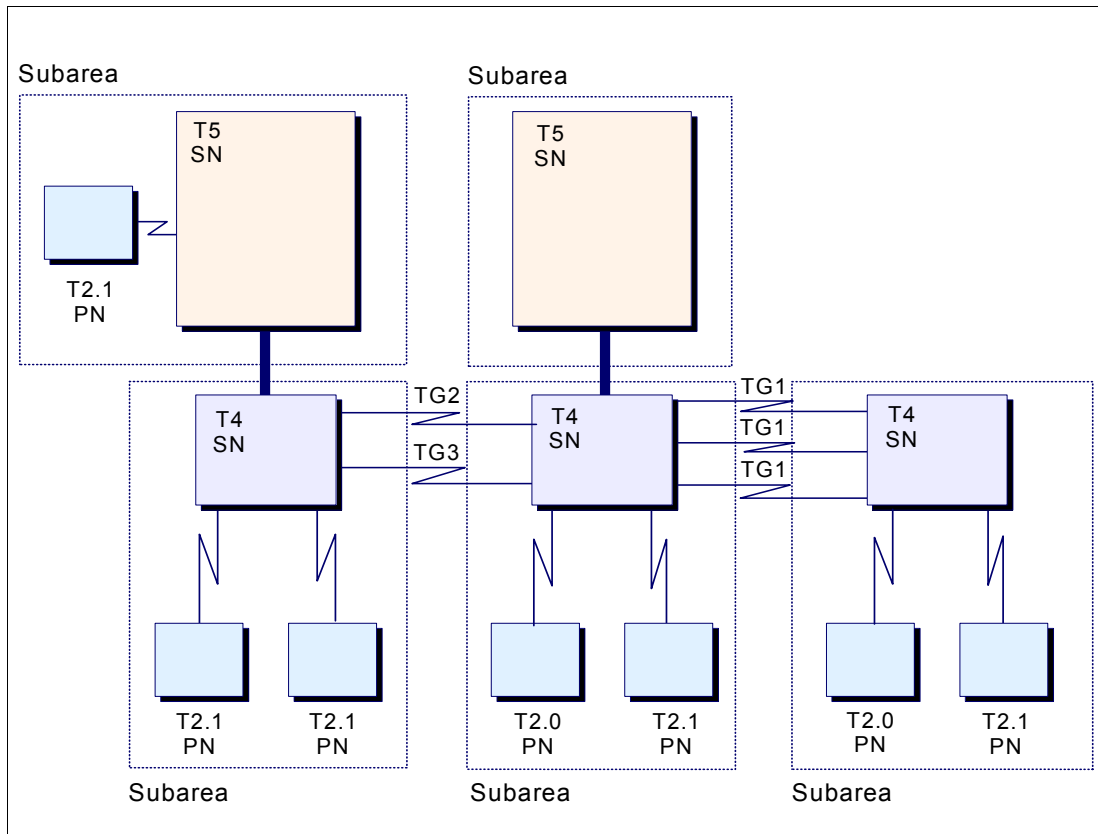


Figure 2-5 Subareas

VTAM subarea nodes

A VTAM subarea node is a SNA type 5 node that functions in a hierarchical environment. Subarea nodes provide services for and control over peripheral nodes. Peripheral nodes require the services of a VTAM subarea node to communicate with other peripheral nodes and subarea nodes in the network. Peripheral nodes include SNA type 2.1, 2.0 or type 1 nodes and function as distributed processors, cluster controllers, or workstations. Type 2.1 nodes can use low-entry networking or APPN connections to communicate with adjacent type 2.1 nodes; however, when attaching to a VTAM subarea node, they are restricted to using LEN connections.

SNA defines the following network configurations:

- ▶ A hierarchical network consisting of subarea nodes and peripheral nodes
- ▶ A peer-oriented network consisting of APPN and LEN nodes
- ▶ A mixed network that combines one or more hierarchical subnets with one or more peer-oriented subnets

Network structure

The organization of a hierarchical network structure is determined by the way control of network services is maintained. Host nodes containing SSCPs are responsible for overall control of communication in the hierarchical network.

A hierarchical network might include LEN or APPN nodes that are attached as peripheral nodes. These nodes can communicate with each other through a subarea network if the boundary nodes to which they are attached support the basic SSCP-independent LU-LU protocols needed for such peer interactions.

Subareas

A subarea consists of one subarea node and the peripheral nodes that are attached to that subarea node. The concept of a subarea applies only to subarea networks and composite networks. The network configuration in Figure 2-5 contains five subarea nodes and seven peripheral nodes. Each subarea node and its attached peripheral nodes constitutes a subarea.

A VTAM subarea node:

- ▶ Requires subarea network routing definitions, such as path, virtual route (VR), and explicit route (ER) definitions to communicate with other subarea nodes
- ▶ Uses SSCP-SSCP, SSCP-PU, SSCP-LU, and LU-LU sessions for network control and end-user data
- ▶ Does not support CP-CP sessions
- ▶ Has a subarea number defined on the HOSTSA start option
- ▶ Does not have the NODETYPE start option defined
- ▶ Can activate NCPs and have SSCP-SSCP sessions with other VTAM subarea, interchange, and migration data host nodes

2.6 Domains in a subarea network

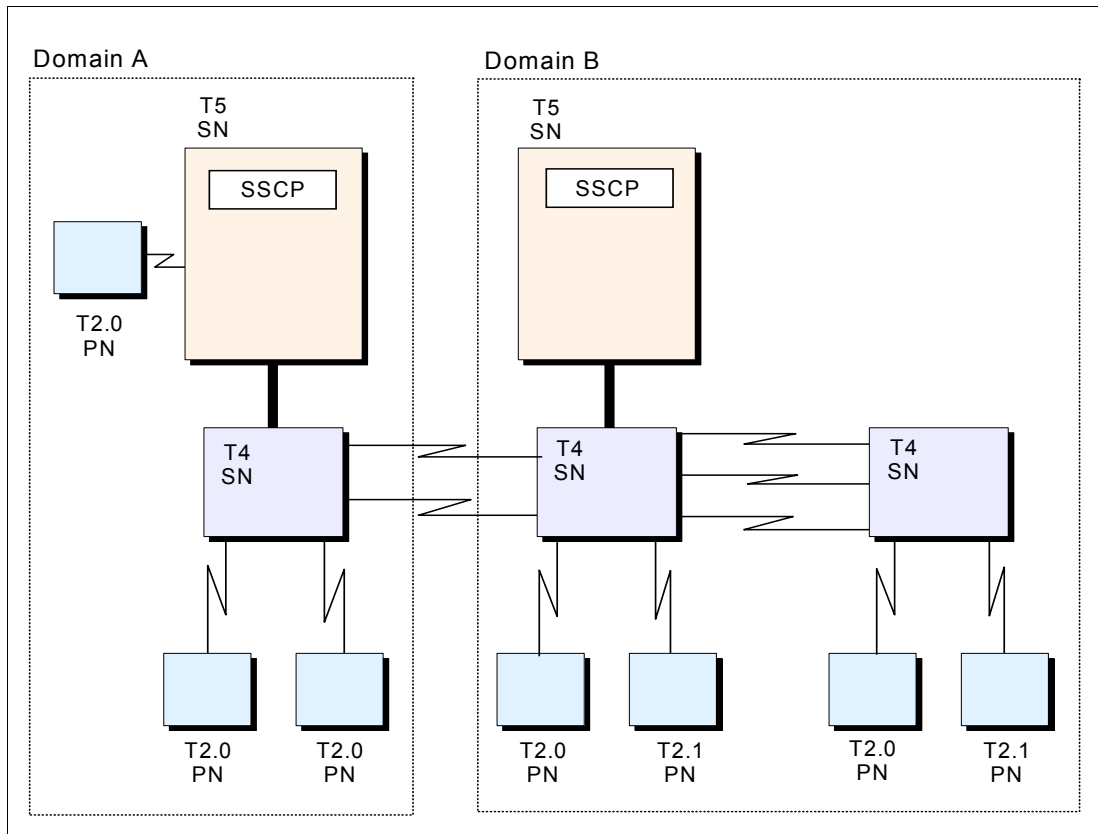


Figure 2-6 SNA subarea cross-domain network

Domains in a subarea network

In a subarea network, every T5 and T4 node is assigned a subarea number. The subarea number has to be unique in the SNA network. The SNA network is assigned a network identifier referred to as a NETID. All the resources in the same subarea network carry the same NETID name. Within one NETID subarea network, you can have more than one z/OS system that implements the SNA protocol.

System services control point (SSCP)

Every z/OS system with VTAM that implements SNA is referred to as a domain, which is an area of control. Within a subarea network, a domain is that portion of the network managed by the system services control point (SSCP) in a T5 subarea node. When a subarea network has only one T5 node, that node must manage all of the network resources.

A subarea network that contains only one T5 node is a single-domain subarea network. When there are multiple T5 nodes in the network, each T5 node can control a portion of the network resources. A subarea network that contains more than one T5 node is a multiple domain subarea network.

For a session between SSCPs to exist, VTAM must know about all cross-domain resource managers with which it can communicate. You must define to VTAM its own cross-domain resource manager and all other cross-domain resource managers in the network.

The cross-domain resource manager that represents the SSCP in your domain is called the host cross-domain resource manager. The cross-domain resource managers that represent the SSCPs in other domains are called external cross-domain resource managers.

2.7 Network node domains in APPN

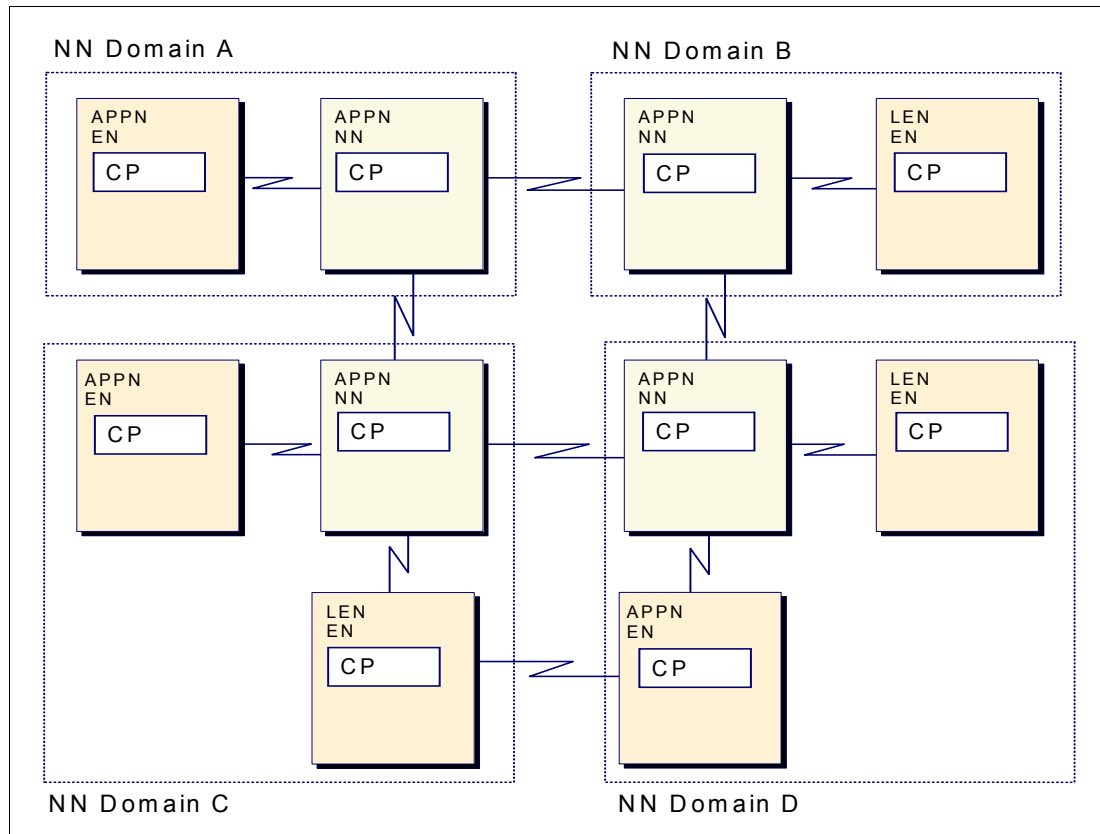


Figure 2-7 Network node domain in APPN

Network node domains in APPN

APPN networks are by definition multiple-domain networks. Figure 2-7 illustrates an APPN network containing four network node domains: A, B, C, and D. The domains of the end nodes are included within the domains of their respective network node servers. An APPN network constitutes a peer-oriented SNA network. All APPN nodes are considered to be peers and do not rely on other nodes to control communication in the network the way a subarea node controls communication between peripheral nodes. There is, however, a measure of hierarchical control because a network node server provides certain network services to its attached end nodes. The difference is that, in APPN networks, hierarchical control is not determined by product or processor type as it is in subarea networks, where only large host processors contain SSCPs and node types generally reflect product types. The domain of a node in an APPN network is that portion of the network served by the control point in the node. The control point in an APPN node is called simply a control point (CP). An end node control point's domain consists solely of its local resources. It is included within the domain of its network node server. A network node control point's domain includes the resources in the network node and in any client end nodes (nodes for which the network node is acting as the network node server) attached directly to the network node.

Defining network nodes

If you specify a value for NN, it is validated when a connection is attempted to the adjacent node. If the adjacent CP is not the type of node that is expected, then connection setup fails. If you do not specify a value for NN, then the APPN capabilities of the adjacent node are identified and accepted when a connection is established. An end node is shown as (EN).

To specify whether an adjacent node is expected to be a network node, specify:

NN=YES | NO

NN=NO - Specifies that the adjacent node is expected to be an end node.

NN=YES - Specifies that the adjacent node is expected to be a network node.

Restriction: In a peripheral subnetwork boundary configuration between a border node and a network node without border node function, the border node appears to the non-border node as an end node. In this case, coding NN=YES on the definition statement that represents the border node will result in a connection failure. Avoid the failure by not coding the NN operand or by coding NN=NO.

Low-entry networking (LEN) nodes

Low-entry networking nodes, also referred to as T2.1 nodes, were introduced in the mid-1980s to address the requirement for peer networking. The low-entry networking node was the first stage of APPN evolution. The T2.1 node allows peer-to-peer connection and provides the physical and session-level connectivity required to support logical unit type 6.2 (LU 6.2). T2.1 nodes use protocols with reduced system definition requirements. For example, link station roles are negotiated as primary or secondary during the connection phase, instead of being predefined, which is the case for subarea networking.

The building block for an APPN node is the T2.1 node. Be aware that T2.1 by itself does not provide any APPN functionality; additional software is required to make a T2.1 node an APPN node.

Depending on the software that implements APPN in T2.1 nodes, the node can be configured in the APPN networks with varying complexity, from the simplest case of an isolated pair of low-entry networking nodes to a large APPN network. Using low-entry networking or APPN protocols, any node can control the establishment and termination of sessions.

The following node types can be implemented by T2.1 nodes:

- ▶ Low-entry networking (LEN)
- ▶ APPN end node (EN)
- ▶ APPN network node (NN)

Low-entry networking nodes do not implement a control point. With low-entry networking, you must predefine every partner resource (if it does not reside on this same node) along with the first hop (link) toward that resource because neither searching nor topology exchanges are supported by low-entry networking nodes. This predefinition requirement is the primary drawback of low-entry networking nodes.

Control point (CP-CP) sessions

To perform directory services and topology and route selection services, adjacent APPN nodes throughout the APPN network use CP-CP sessions to exchange network and control information. CP-CP sessions are always logical unit type 6.2 (LU 6.2) sessions. Using this session type, a contention situation could arise if both session partners attempted to allocate a conversation and exchange data at the same time. This situation is resolved by defining one of the sessions the contention-winner (often called the conwinner) session and the other the contention-loser (or conloser) session. The primary session partner refers to its session as the contention-winner session, and the secondary session partner refers to that same session as the contention-loser session. The contention winner side of the session is the one that initiates the BIND. CP-CP sessions are only established between adjacent APPN nodes and always use the CPSVCMG logon mode name and APPN class of service.

2.8 Starting VTAM SNA

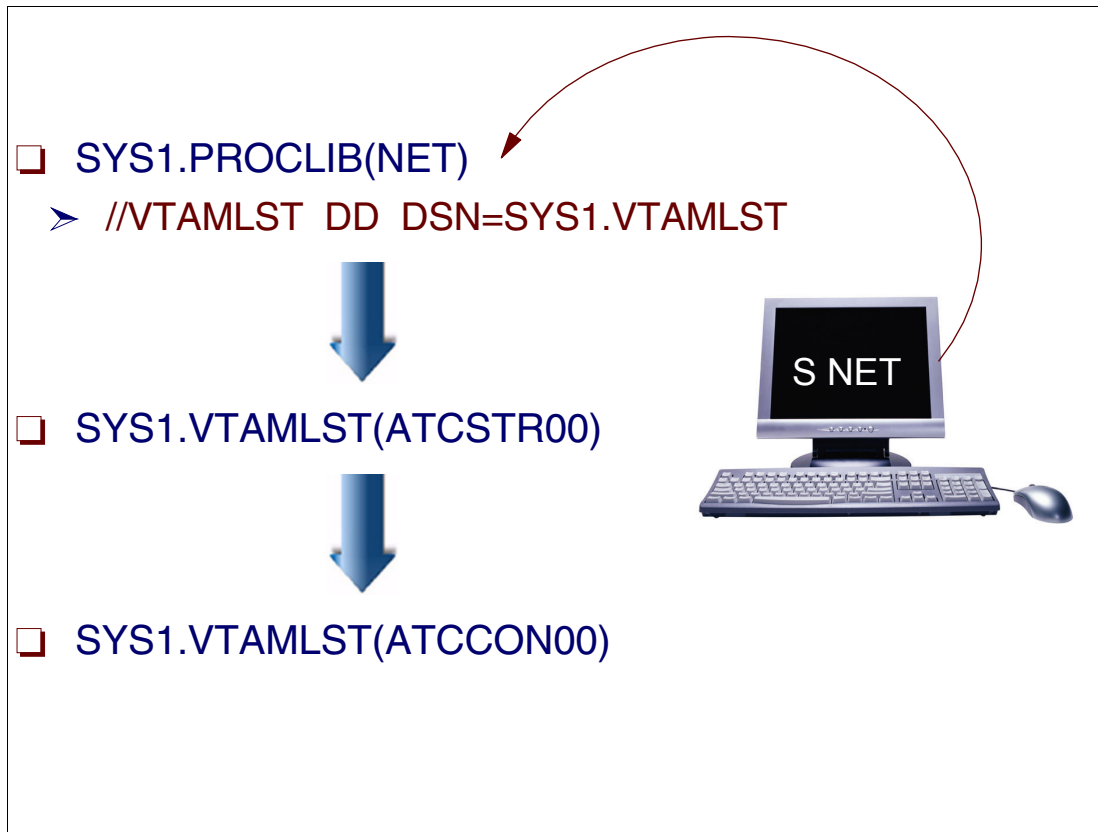


Figure 2-8 VTAM major data sets

Starting Communications Server for z/OS SNA

You use the **START** command to activate VTAM. This command uses some defaults set by the system programmer (as part of the network definition). When you use the **START** command, you might be required to enter changes or additions to the defaults. If this is the case, your system programmer can tell you what commands to enter.

The defaults for the **START** command are contained in the configuration list and start list that are defined by the system programmer. The *configuration list* contains a list of VTAM resources. VTAM uses this list to find out what resources to activate when VTAM is first started. A network can have more than one configuration list; this allows different configurations to be brought up at start time. The *start list* contains VTAM operating system options. For example, it defines how storage is to be used and indicates whether VTAM suppresses or shows messages. The start list also identifies which configuration list is to be used for startup. You should code a z/OS Communications Server start procedure and save it in SYS1.PROCLIB. The system operator specifies the procedure when starting z/OS Communications Server.

VTAM procedure name

The start procedure in Example 2-1 on page 35 is called NET. The name NET is not required but is strongly recommended for consistency in entering the z/OS Communications Server operator commands and to reduce the operator's chances of making a syntax error. The procedure name you specify must be the first operand on the **START** and **MODIFY** operator commands. For **DISPLAY**, **HALT**, and **VARY** the procedure name is always NET.

ATCSTR00 member

When you start VTAM, a list of user-defined default values (ATCSTR00) is read from the VTAM definition library. If VTAM cannot find the ATCSTR00 file, it prompts you, giving you a choice of the following:

- ▶ Using default values
- ▶ Using an alternate start option file (ATCSTRxx, where xx specifies the identifier of the start option file you want to use)
- ▶ Cancelling the start attempt

If you do not want to use the ATCSTR00 member, you can avoid the prompt by coding an ATCSTR00 file that contains only comments. VTAM issues a message that the file is empty, but does not interrupt start processing. The VTAM operator can enter the LIST=xx start option on the **START** command to name another list to supplement ATCSTR00. The supplemental list (ATCSTRxx) can override options in the default ATCSTR00 list, as well as the IBM-supplied internal default values.

ATCCON00 member

A configuration list lets you specify which resources are activated automatically when you start VTAM. You are not required to define a configuration list, but it makes the VTAM operator's job easier because VTAM activates the resources in the configuration list automatically. Configuration lists are coded in files named ATCCONxx, where xx specifies the identifier of a particular configuration list.

Example 2-1 shows job control statements for a typical start procedure.

Example 2-1 Sample VTAM start procedure

```
//NET      PROC
//NET      EXEC PGM=ISTINM01,REGION=3072K,
//          DPRTY=(15,15),TIME=1440
//VTAMLST  DD DSN=SYS1.VTAMLST,DISP=SHR
//VTAMLIB  DD DSN=SYS1.VTAMLIB,DISP=SHR
//SYSABEND DD SYSOUT=*,HOLD=YES
//SISTCLIB DD DSN=SYS1.SISTCLIB,DISP=SHR
//ISTCMIP  DD DSN=SYS1.SISTCMIP,DISP=SHR
//ISTASN1  DD DSN=SYS1.SISTASN1,DISP=SHR
//ACYGDMO  DD DSN=SYS1.SISTGDMO(ACYGDMO),DISP=SHR
```

SYS1.VTAMLST

SYS1.VTAMLST is the z/OS Communications Server definition library, which consists of files containing the definitions for network resources and start options. It is a required partitioned data set, and you need to allocate it on a direct-access volume before you file z/OS Communications Server network definitions.

Installation considerations

To start with, however, basic VTAM configuration can be achieved by tailoring:

- ▶ The VTAM start procedure SYS1.PROCLIB(NET).
- ▶ The VTAM startup member SYS1.VTAMLST(ATCSTRxx).
- ▶ The VTAM major node activation list SYS1.VTAMLST(ATCCONxx).
- ▶ Major node members, usually stored in SYS1.VTAMLST. These members are used to define the SNA network.

2.9 Starting VTAM

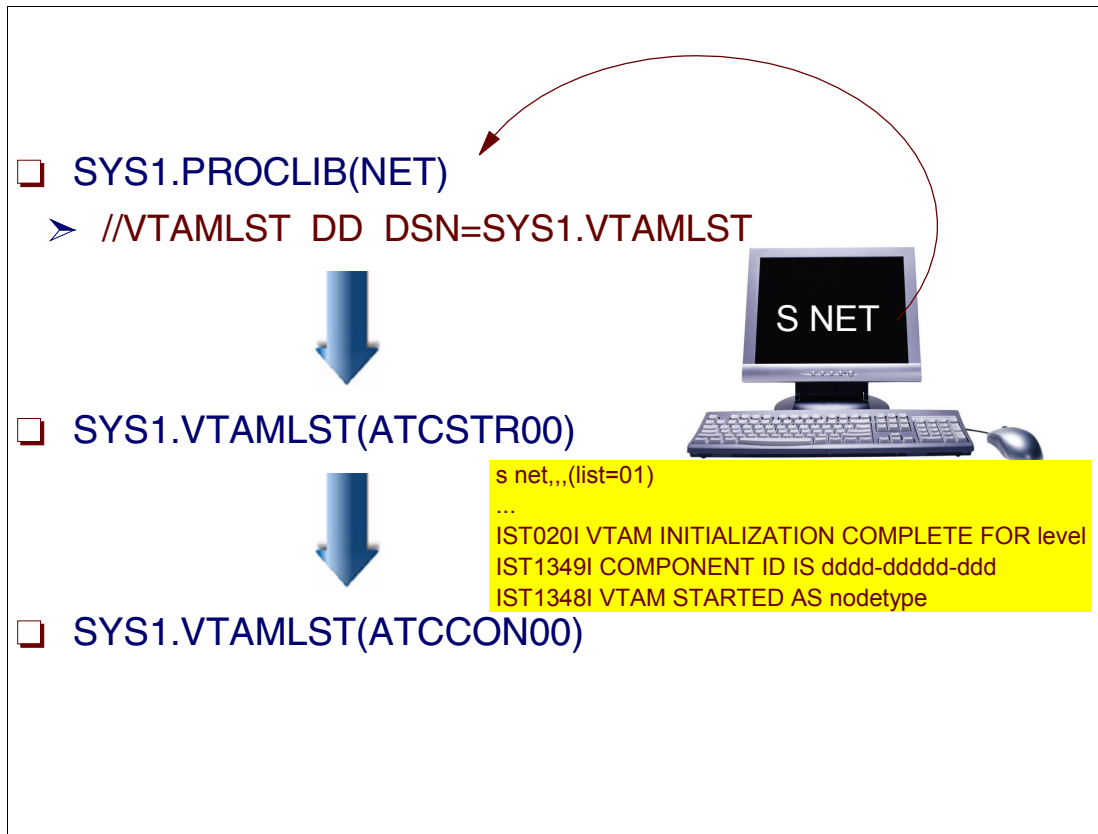


Figure 2-9 VTAM startup

VTAM start list

The defaults for the **START** command are contained in the configuration list and start list that are defined by the system programmer. The configuration list contains a list of VTAM resources. VTAM uses this list to find out what resources to activate when VTAM is first started. A network can have more than one configuration list; this allows different configurations to be brought up at start time.

The start list contains VTAM operating system options. For example, it defines how storage is to be used and indicates whether VTAM suppresses or shows messages. The start list also identifies which configuration list is to be used for startup. In the following example of a basic VTAM procedure, note the VTAMLST DD statement, which specifies the location of the network definition members, and the VTAMLIB DD statement, which specifies the location of VTAM's executable code.

VTAM procedure

The first thing VTAM looks for in the procedure shown in Example 2-1 on page 35 is an ATCSTRxx member in the VTAMLST data set, where xx is the LIST=xx value that is specified on the **START** command and that can be any 2-character alphanumeric value. The ATCSTRxx member contains VTAM start parameters that define VTAM's identity on the network as well as tuning options and a pointer to the major node startup list, ATCCONxx.

If no LIST= parameter is specified on the **S NET** command, the default value of **00** is used.

For example:

The command **S VTAM, , , (LIST=Z9)** will cause VTAM to read startup parameters from ATCSTRZ9.

The command **S VTAM, , , (LIST=EX)** will cause VTAM to read startup parameters from ATCSTREX.

The command **S VTAM, , , (LIST=99)** will cause VTAM to read startup parameters from ATCSTR99.

The command **S VTAM** will cause VTAM to read startup parameters from the default ATCSTR00.

A typical ATCSTRxx member might contain the parameters shown in Example 2-2.

Example 2-2 A typical ATCSTRxx member

```
SSCPID=0061,  
CONFIG=01,  
HOSTSA=130,  
NETID=GBIBMPGX,  
SSCPNAME=PGXYA,  
IOBUF=(100,256,19,,12,30)
```

For a complete description of the parameters and their values refer to *z/OS Communications Server: SNA Resource Definition Reference*, SC31-8778 and *z/OS Communications Server: SNA Resource Definition Samples*, SC31-8736.

Configuration lists

A configuration list specifies the resources that are to be activated when VTAM is started. The member names of the resources you want to have activated when VTAM starts are stored in the ATCCONxx member in the VTAM definition library, where xx is any two alphanumeric characters. The value xx can then be used on the CONFIG operand of the VTAM START command, or on the CONFIG start option in the start option list, to specify which definitions are to be activated at startup. In Example 2-2, to use the configuration list defined in ATCCON01, you can specify CONFIG=01 on the VTAM **START** command.

For example, if ATCSTR00 contains CONFIG=01, as shown in Example 2-2, but the operator enters LIST=ST and ATCSTRST contains the start option CONFIG=02, VTAM activates the configuration defined by ATCCON02.

2.10 VTAM start options

- ❑ Stored in ATCSTRxx member of VTAMLST
- ❑ Default start option member is ATCSTR00 and is mandatory to start VTAM in "quiet" mode.
- ❑ Supplemental start option list.
 - The VTAM operator can enter the LIST=xx start option on the START command
- ❑ Required start options:
 - SSCPID
 - SSCPNAME
 - NETID

Figure 2-10 VTAM start options

Start options

Start options provide information about the conditions under which VTAM runs. They also enable you to tailor VTAM to meet your needs each time VTAM is started. When you start VTAM, you can enter start options on the **START** command, or you can write one or more start lists to assist you in starting VTAM. You can also activate your resources either manually or automatically. To automatically activate some or all of your resources, you can write one or more configuration lists. Writing start lists and configuration lists:

- ▶ Reduces the amount of operator involvement and the chance of entering incorrect information
- ▶ Enables VTAM to initialize the domain faster
- ▶ Can reduce the amount of coding required because many operands can have defaults specified as start options

Many start options can be dynamically modified and also displayed.

You can code multiple start options or configuration lists and use these lists according to your needs. After you specify start options and configuration lists, they remain in effect until you modify them or until VTAM is halted.

SYS1.VTAMLST data set

The start options are stored in the SYS1.VTAMLST partitioned data set (PDS) or any other PDS concatenated to the VTAMLST DD statement. The member name is ATCSTRxx, where

xx specifies the identifier of the start option file. Start options provide information about the conditions under which VTAM runs. They also enable you to tailor VTAM to meet your needs each time VTAM is started. Many operands can have defaults specified as start options, thus reducing the amount of coding required. Many start options can be dynamically modified and also displayed. A complete list of start options is listed in the *z/OS Communications Server: SNA Resource Definition Reference*, SC31-8778.

When VTAM starts, a list of user-defined default values (ATCSTR00) is read from the VTAM definition library. VTAM looks for the default ATCSTR00 member. If VTAM cannot find the ATCSTR00 file, it prompts the operator. You can avoid the prompt by coding an ATCSTR00 file that contains only comments. VTAM issues a message that the file is empty, but does not interrupt start processing.

Required start options

The following three start options are required:

SSCPID The SSCPID start option provides VTAM with a unique numeric identifier. The SSCPID value is used by some physical units to identify the VTAM with which it is in session. If you plan to expand or incorporate a single-domain network into a larger network, be sure that the value of SSCPID is unique for each host. The SSCPID value you specify must be different from the SSCPIDs in other networks that can be in session with this host.

When activating the cross-domain resource manager-cross-domain resource manager (CDRM-CDRM) session, the host with the higher SSCPID value sends the activate cross-domain resource manager (ACTCDRM) and the host with the higher HOSTSA value activates the route. Unpredictable results occur if the route chosen for the ACTCDRM is different from the route activated.

SSCPNAME The SSCPNAME start option provides a unique name for VTAM. This option is required for a single-domain network, but is primarily used in multiple-domain and multiple-network environments to identify a particular VTAM. The SSCPNAME option must be different from the HOSTPU start option that identifies the physical unit within VTAM.

The SSCPNAME option is also used as the CP name for a VTAM that implements APPN. Although the SSCPNAME and CP name are identical, note that the SSCP and CP are distinct resources and can be displayed individually.

Note: The SSCPNAME should match the name that is coded in the CDRM major node for this VTAM.

NETID The NETID start option provides VTAM with the network identifier. If you connect your VTAM to another network, the network identifiers must be unique. The network name should conform to an overall naming standard, such as the following:

- ▶ The first two characters identify the country in which the network is managed and comply with the International Standards Organization (ISO) Standard 3166.
- ▶ The next four characters identify your enterprise.
- ▶ The last two characters identify a particular network within your enterprise.

Note: For start options you are not required to code, VTAM uses the default values. You can override any start option value when VTAM is started or enter new start option values at the operator console.

2.11 VTAM start options by function

- SSCP or CP characteristics
- VTAM initialization
- Storage
- Session control
- Connectivity
- Session security
- Messages
- Recording and statistics
- Traces and dumps
- Buffer pools
- Performance and tuning

Figure 2-11 Optional VTAM start options

VTAM start options by function

Start options are entered by the operator. The VTAM operator can enter additional start options on the **START** command, and also during VTAM startup if prompted. (VTAM prompts for start options if the operator does not enter any start options on the **START** command.) To prevent VTAM from prompting for start options during startup, code the **NOPROMPT** option in **ATCSTR00**.

You can create a start options list. To use a start option list, create a list and put it in a partitioned data set member named **ATCSTRyy** in **SYS1.VTAMLST**. The **yy** value must be the same as the **yy** value in the **LIST=yy** start option entered by the operator. Following is a sample start list **ATCSTRyy**:

```
SSCPID=01
SSCPNAME=SSCP1A
NETID=NETA
DYNASSCP=YES
SSCPDYN=YES
HOSTSA=1
```

Note: For more detailed descriptions of the start options see "Descriptions of start options" in topic 4.3 in *z/OS Communications Server SNA Resource Definition Reference*, SC31-8778.

START options LIST=YY

ATCSTR00 is the default start option list supplied by IBM. It initializes VTAM as a subarea node. For this list to take effect when you start VTAM, you must copy it out of the GENDECK data set in SYS1.ASAMPLIB and put that copy in the SYS1.VTAMLIB, the VTAM definition library.

Tip: As a guideline, the values specified in ATCSTR00 are not necessarily the default values for the start options listed there. If ATCSTR00 is the only start option list you use for a particular node, that node will be initialized as a subarea VTAM node.

The operator can start VTAM with this list using LIST=yy on the **START** command. The following START option lists can be created:

- ▶ IBM-supplied default start option list
- ▶ Subarea node start option list
- ▶ Network node start option list
- ▶ End node start option list
- ▶ Interchange node start option list
- ▶ Composite network node start option list
- ▶ Migration data host start option list
- ▶ Start option list with border node support
- ▶ Central directory server start option list
- ▶ Using MVS system symbols to define start option lists

Note: For more information about start options and configuration lists, see the *z/OS Communications Server: SNA Network Implementation Guide*, SC31-8777 and the *z/OS Communications Server: SNA Resource Definition Reference*, SC31-8778.

Default option list

When VTAM is started, you can provide the options from any combination of the following sources (listed here from the lowest to highest priority):

- ▶ IBM-supplied values internal to VTAM (default values).
- ▶ Default start option list. When you start VTAM, a list of user-defined default values (ATCSTR00) is read from the VTAM definition library. If VTAM cannot find the ATCSTR00 file, it prompts you, giving you a choice of:
 - Using default values
 - Using an alternate start option file (ATCSTRxx, where xx specifies the identifier of the start option file you want to use)
 - Canceling the start attempt

```

* =====> BEGINNING OF DATA SET ATCSTR00
*****
*   ATCSTR00 - VTAM START LIST FOR A SUBAREA NODE - DEFAULT LIST   *
*****
ALSREQ=NO,                ** ADJ LINK STAT IN ALS LIST    **X
ASYDE=TERM,               ** ASYNCH DEVICE SESSION TERM  **X
BSCMDRS=(STATS,INOPS),   ** REPORT BISYNC INOP STATS    **X
CACHETI=8,                ** CACHE TIMER VALUE          **X
CDRSCFI=480,              ** DYNAMIC CDRSC INTERVAL     **X
CMPVTAM=0,                ** MAX HOST APPL COMPRESSION  **X
COLD,                      ** CONFIG RESTART STATUS      **X
CSALIMIT=0,               ** MAXIMUM CSA LIMIT          **X
CSA24=0,                  ** 24 BIT ADDR STORAGE LIMIT  **X
DATEFORM=MDY,             ** AUTO IPL DATE FORM         **X
DLRTCB=32,                ** NCP DUMP/LOAD/RESTART TCBS **X
DYNASSCP=YES,             ** DYNA SESS REQ ROUTE TO ADJS **X
DYNLU=YES,                ** DYNAMIC DLU CAPABILITY     **X
ENCRYPTN=NO,               ** APPL ENCRYPTION CAPABILITY **X
GWSSCP=YES,               ** GATEWAY SSCP CAPABILITY    **X
HOTIOTRM=0,               ** HOT I/O TERMINATION        **X
IOINT=180,                ** OUTSTANDING RESPONSE DISPLAY **X
MAXSUBA=15,               ** HIGHEST SUBAREA VALUE     **X
MSGMOD=NO,                ** VTAM MODULE MESSAGE DISPLAY **X
NCPBUFSZ=512,            ** NCP LOAD/DUMP RU SIZE      **X
NMVTLOG=NPDA,             ** NMVT RECORDING             **X
NODELST=NODEDS1,         ** WARM RESTART NODE LIST     **X
PPOLOG=NO,                ** PPO LOG RECORDING          **X
PROMPT,                   ** START OPTIONS PROMPT       **X
SDLCMDRS=(STATS,INOPS),  ** REPORT SDLC INOP STATS     **X
SONLIM=(60,30),           ** IO BUF % FOR SESS OUT NOTIFY **X
SRCHRED=ON,               ** PERFORM SEARCH REDUCTION   **X
SRCOUNT=10,               ** SEARCH REDUCTION COUNT LIMIT **X
SRTIMER=30,               ** SEARCH REDUCTION TIME LIMIT **X
SSCPDYN=YES,              ** DYNAMIC ADD ENTRY TO ADJSSCP **X
SSCPORD=PRIORITY,        ** ADJSSCP SEARCH ORDER       **X
SUPP=NOSUP,               ** VTAM MESSAGE CLASS SUPPRESS **X
TNSTAT,TIME=60,           ** TUNING STATISTICS          **X
VFYRED=YES,               ** LU 6.2 VERIFICATION REDUCTION**X
XNETALS=YES,              ** NON NATIVE NET CONNECTIVITY **X
TRACE,TYPE=VTAM,MODE=INT,OPT=NONE,SIZE=100,      X
USSTAB=ISTINCNO,         ** VTAM MESSAGE & COMMAND TABLE **X
APBUF=(16,,2,,1,3),      ** 24 BIT CSA BUFFER          **X
BSBUF=(100,,0,,25,60),   ** BOUNDARY LU SESSION BUFFER **X
CRPLBUF=(100,,0,,1,29),  ** APPL REQUEST BUFFER        **X
IOBUF=(100,384,5,,1,30), ** PIU INPUT/OUTPUT BUFFER     **X
LFBUF=(25,,0,,1,1),      ** ACTIVE APPL BUFFER EAS < 30 **X
LPBUF=(70,,0,,5,1),      ** ACTIVE VTAM PROCESS BUFFER **X
SFBUF=(51,,0,,1,1),      ** ACTIVE APPL BUFFER EAS >= 30 **X
SPBUF=(10,,0,,1,1),      ** LARGE MESSAGE REQUEST BUFFER **X
XDBUF=(6,,0,,1,5)        ** XID EXCHANGE PROCESS BUFFER **
END OF DATA SET ATCSTR00

```

Figure 2-12 ATCSTR00 is the default start option list

2.12 VTAM buffers

- ❑ Common storage area (CSA)
- ❑ Buffer pools
 - VTAM control blocks
 - I/O buffers
 - Channel programs
- ❑ Communication storage manager (CSM)

Figure 2-13 VTAM buffers

Common storage area

VTAM uses common service area (CSA) to maintain some buffer pools and control blocks. Common storage areas are also used for other system needs, so you need to monitor VTAM use of common storage areas and you might want to control VTAM use. VTAM uses buffer pools for control blocks, network traffic data, and channel programs. A shortage of buffer pools can have an adverse effect on VTAM CPU time, storage consumption, and the ability to serve DB2® requests.

VTAM buffer pools

VTAM uses buffer pools to control the handling of data. It dynamically allocates and deallocates space in buffer pools for the VTAM control blocks, I/O buffers, and channel programs that control the transmission of data. Specifying large buffer pools can waste storage, but does not require frequent CPU use for expansion. Specifying small buffer pools conserves storage, but requires frequent CPU use for expansion and contraction. With proper tuning, you can achieve a balance between storage use and performance that is suitable for your environment

Buffer pool allocation

VTAM provides two types of buffer pool storage allocations, basic and dynamic, as follows:

Basic allocation The amount of space that you reserve for the buffer pool when VTAM is started.

While basic allocation sets the base size of your buffer pools, dynamic expansion enables a buffer pool to be expanded temporarily during periods of heavy demand. Its use can greatly increase the efficiency with which VTAM uses storage, particularly for I/O buffers.

Dynamic allocation The process by which VTAM temporarily increases the size of a buffer pool when there are heavy demands for space in that pool.

Without dynamic expansion of a pool, basic allocation parameters must be specified large enough to meet the greatest possible demands on the pool. With dynamic expansion, smaller basic allocation values can be specified, and peak demands on the pool can be met with dynamic expansion. Dynamic expansion is strongly recommended for most buffer pools because the peak demand can vary considerably from the normal demand.

Setting buffer pool allocations

To set buffer pool space, use the buffer pool start options:

```
poolname=(baseno,bufsize,slowpt,F,xpanno,xpanpt,xpanlim)
```

The first four operands of the start option (*baseno*, *bufsize*, *slowpt*, and *F*) establish the base size and the characteristics of the buffer pool. The last three operands (*xpanno*, *xpanpt*, and *xpanlim*) specify how dynamic expansion of the buffer pools is to be performed. You enable dynamic expansion by coding the *xpanno* and *xpanpt* operands on the buffer pool start option.

Note: Define dynamic expansion so that pools are expanded one page at a time. To do this, define *xpanno* to be the number of buffers that fit on one page.

For a description of these operands, refer to *z/OS Communications Server: SNA Resource Definition Reference*, SC31-8778. See the guidelines for dynamic expansion section.

Attention: As buffer pools expand, expansion uses virtual storage that can potentially impact an operating system that has virtual storage constraints. VTAM use of virtual storage can be limited by the CSA start options and the virtual storage allocated to VTAM address space.

Communication storage manager (CSM)

The communications storage manager (CSM) is a VTAM component that allows authorized host applications to share data with VTAM and other CSM users without the need to physically copy the data.

CSM is provided as part of the high performance data transfer (HPDT) family of services. HPDT optimizes system performance for the transfer of bulk data. By providing a means for authorized applications to share buffers, CSM improves system performance during the transfer of bulk data by reducing the processing required for data movement. As a result, the following CPU resources are conserved:

- ▶ CPU cycles
- ▶ Memory bus
- ▶ Cache

2.13 VTAM major nodes

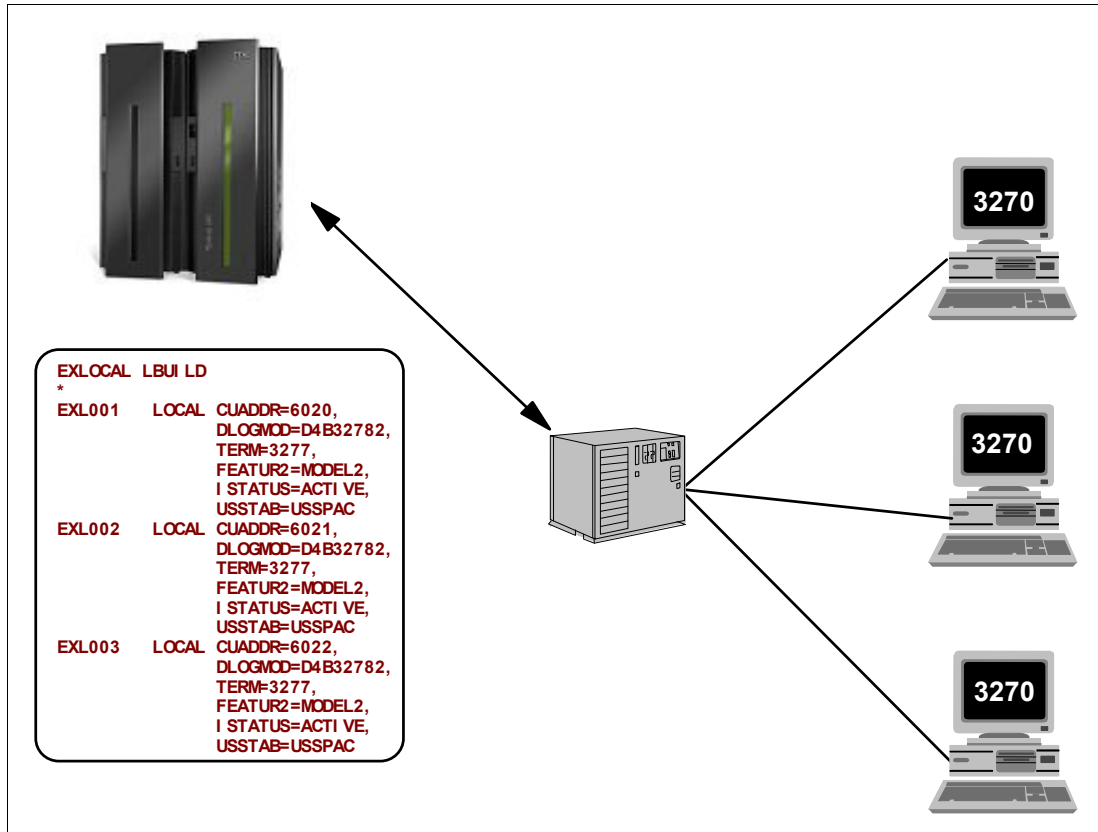


Figure 2-14 VTAM major nodes

VTAM major nodes

Resources in a VTAM network are classified under major and minor nodes. Each major node can contain one or more minor nodes. To define the attachment of any device, set of devices, various tables, or search order of resources to VTAM, you need to code major node members and store them in the VTAMLST partitioned data set.

Figure 2-14 is an example of a local non-SNA major node. In the past this major node was used to define coax-attached 3270 terminals connected to a channel-attached 3174-type controller. Today, the local non-SNA major node is used to define z/OS consoles, whether they use an OSA attachment (OSC) or a 2074 ESCON attached device.

Brief descriptions of the major nodes are as follows:

► **Adjacent control point major node**

Specifies the name and attributes of an adjacent APPN node with which a node can establish type 2.1 connections.

► **Application program major node**

Contains application programs as minor nodes. There can be one or more application programs in each application program major node.

- ▶ **Channel-attachment major node**
 Contains definition statements for one of several types of channel (link) groups. The possible types of link groups are:
 - Channel-to-channel support
 - Multipath channel support
 - Channel-attached NCP support
- ▶ **Cross-domain resource major node**
 A logical set of cross-domain resources (CDRSCs). You can define one or more cross-domain resources, each of which is a minor node, in each major node. You can also use the cross-domain resource major node to predefine independent LUs in a boundary-attached subarea network.
- ▶ **Cross-domain resource manager major node**
 A logical set of cross-domain resource managers (CDRMs). You can define one or more cross-domain resource managers, each of which is a minor node, in each major node.
- ▶ **External communication adapter major node**
 Defines a connection (port) between VTAM and a token-bus network, a token-ring network, a CSMA/CD 802.3 LAN, or an FDDI LAN attached through the IBM 3172 Nways interconnect controller or an IBM Open Systems Adapter. The minor nodes are the line groups, lines, and physical units attached to the port.
- ▶ **Local non-SNA major node**
 Consists of a logical group of channel-attached non-SNA terminals. Each terminal in the group is a minor node. The terminals in a group do not have to all be connected to the same cluster controller.
- ▶ **Local SNA major node**
 Consists of a logical group of channel-attached SNA terminals. The minor nodes are the physical units and their logical units that are contained in the terminals.
- ▶ **LU group major node**
 Defines one or more model LU groups.
- ▶ **Model major node**
 Defines dynamic switched definitions for PUs and LUs.
- ▶ **NCP major node**
 Consists of the resources attached to an NCP. The attached resources (lines, physical units, and logical units) are minor nodes.
- ▶ **Switched major node**
 Contains definitions for potential switched connections to either a subarea node or a peripheral node.
- ▶ **Transport resource list major node**
 Describes the connectivity characteristics of a multipath channel connecting two APPN hosts. The characteristics of each multipath channel are defined by a TRLE definition statement. When an adjacent link station is activated, the PU definition statement identifies which TRLE definition statement VTAM uses to route data over the channel.

2.14 Defining LAN attached devices to VTAM

- LAN attached devices are defined using:
 - XCA major node
 - XCA major nodes defines the OSA adapter
 - Switched major node
 - Defines the PU and the LUs of the LAN attached peripheral device

Figure 2-15 Defining LAN attached devices to VTAM

XCA and switched major nodes

VTAM attaches to LANs through an IBM 3172 Nways Interconnect Controller, an Open Systems Adapter, or through an IP network for Enterprise Extender. VTAM attaches to ATM networks through LAN emulation access through an Open Systems Adapter (OSA). VTAM can communicate with the following types of nodes over LAN and ATM LAN emulation connections:

- ▶ Subarea
 - Subarea nodes are nodes with a PU type of 4 or 5. Connections to subarea nodes are nonswitched, DIAL=NO.
- ▶ Peripheral
 - Peripheral nodes are nodes with a PU type of 2 or 2.1. Connections to peripheral nodes are switched, DIAL=YES. To define a peripheral external communication adapter node, code the VBUILD and PORT definition statements followed by the GROUP definition statement, and LINE and PU definition statements as pairs in the switched line group.

VBUILD definition statement

To define a subarea external communication adapter major node, code the VBUILD and PORT definition statements, followed by the GROUP definition statement, and LINE and PU definition statements as pairs in the nonswitched line group.

PORT definition statement

Code a PORT definition statement for each VBUILD TYPE=XCA definition statement. The PORT definition statement defines the VTAM connection to the Enterprise Extender shared access transport facility (SATF). Note that MEDIUM=HPRIP is required. However, only one XCA Major Node for Enterprise Extender (MEDIUM=HPRIP) can be active at a time.

GROUP definition statement

Code a GROUP definition statement to group the similar characteristics of peer nodes in the IP network. VNNAME and VNTYPE can be used to define a virtual node. Code a separate GROUP definition statement to describe each set of characteristics.

LINE definition statement

Code a LINE definition statement to define a single line to a peer processor or station on a LAN attached through an IBM 3172 Nways interconnect controller or through an IP network for Enterprise Extender. If the device has more than one line connected, code a separate LINE definition statement for each line.

PU definition statement

Code a PU definition statement to define the physical unit associated with a line if the line defines a SNA resource. USER=SNA on the LINE definition statement defines SNA resources. Code only one PU definition statement for each LINE definition statement.

```
LMNBC0  VBUILD  TYPE=XCA
PORT1C2  PORT    MEDIUM=CSMACD,ADAPNO=0,SAPADDR=4,CUADDR=BC0
GP1C21  GROUP   DIAL=YES,ANSWER=ON,ISTATUS=INACTIVE,CALL=INOUT
L1C211  LINE
P1C211  PU
L1C212  LINE
P1C212  PU
L1C213  LINE
P1C213  PU
L1C214  LINE
P1C214  PU
```

Figure 2-16 XCA major node

Note: DIAL=YES is required for Enterprise Extender.

Code a PU definition statement for each peer node in the IP network with which VTAM will communicate over this switched link.

```
SWXCA1A  VBUILD  TYPE=SWNET,MAXNO=256,MAXGRP=256
SW1A2A  PU      MAXPATH=5,MAXDATA=256,ADDR=03,           X
          CPNAME=SSCP2A,CPCP=YES,                       X
          PUTYPE=2
PATH2A   PATH   DIALNO=0104004A11111111,                 X
          GRPNM=GP1A2A
SWLU2A0  LU      LOCADDR=0,ISTATUS=INACTIVE
```

Figure 2-17 Switched major node

2.15 VTAM operator commands

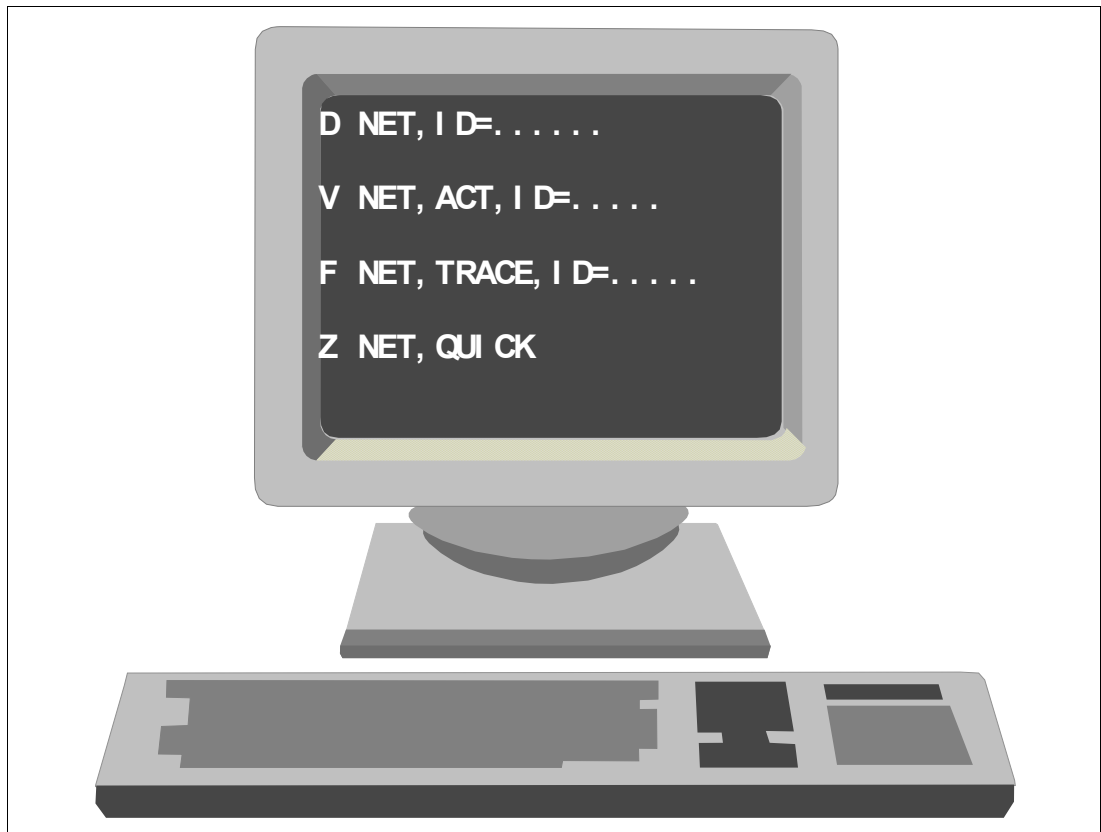


Figure 2-18 VTAM commands

VTAM operator commands

A VTAM operator command consists of the command name or its abbreviation, and various operands that describe the operation to perform. Each VTAM operator command has a procedure name, referred to as *procname* in the command syntax, which tells the operating system to route the command to VTAM for processing. Values for *procname* vary according to the command and the operating system but *procname* must always appear as the first operand of a command.

VTAM operator commands entered by a program operator have the same format and affect as commands entered at the system console. The VTAM operator commands supported are the **VARY (V)**, **DISPLAY (D)**, **MODIFY (F)** and **HALT (Z)** commands. The **REPLY (R)** command is also supported to answer VTAM messages that require a reply.

VTAM command categories

VTAM operator commands fall into four categories:

1. The **DISPLAY** commands
D NET, . . .
2. The **VARY** commands
V NET, . . .
3. The **MODIFY** commands
F NET, . . .

4. The **HALT** commands
Z NET,...

The command descriptions and examples give an overview of the VTAM operator commands available. For a complete list of VTAM operator commands and their syntax and use, refer to *z/OS Communications Server: SNA Operation, SC31-8779*.

Display commands

The basic VTAM Display commands can be used to interrogate VTAM for information about the status of communications links, devices, applications, and even the state of VTAM itself.

The experienced VTAM operator or systems programmer can often successfully diagnose many VTAM-related problems simply by analyzing the output of the various **DISPLAY** commands.

Some of the more useful display commands are defined in Table 2-1.

Table 2-1 VTAM DISPLAY commands

Command	Description
D NET,VTAMOPTS	Display VTAM options, including Subarea number, NETID, SSCPNAME, SSCPID
D NET,BFRUSE	Display information about VTAM's virtual storage and buffer pool usage
D NET,APPLS	Display the applications (APPLIDS) that VTAM knows about
D NET,PATHTAB	Display the status of communications PATHs between subareas in the VTAM network
D NET,MAJNODES	Display details about major nodes
D NET,ID=....	Display information about the specific VTAM resource
D NET,ID=.....,E	Display more information about the specific VTAM resource

VARY commands

These commands can be used to deactivate a resource, thereby making it unavailable on the network, or to reactivate a failed resource. Table 2-2 shows some examples.

Table 2-2 VTAM VARY commands

Command	Description
V NET,ACT,ID=....	Activate the specified resource
V NET,INACT,ID=....	Deactivate the specified resource
V NET,INACT,ID=...,I	Immediately deactivate the specified resource
V NET,INACT,ID=...,F	Force-deactivate the specified resource
V,NET,ACQ,ID=...	Acquire the specified resource (NCP or non-switched PU) without activating it. Used during network recovery whereby this VTAM can take ownership of another VTAM's resource

MODIFY commands

The VTAM Modify commands are used for the more advanced VTAM functions such as replacing an active LOGMODE table, for example, without having to restart VTAM, or to start and stop VTAM tracing. Table 2-3 on page 51 includes some examples.

Table 2-3 VTAM MODIFY commands

Command	Description
F NET,TNSTAT,CNSL=YES	Activate the VTAM tuning statistics display and output to the console
F NET,NOTNSTAT	Deactivate the VTAM tuning statistics display
F NET,TRACE,.....	Initiate or modify VTAM tracing
F NET,NOTRACE,.....	Terminate VTAM tracing

HALT commands

The VTAM Halt commands are used to stop VTAM processing. Table 2-4 includes some common examples.

Table 2-4 VTAM HALT commands

Command	Description
Z NET	Terminate VTAM processing normally
Z NET,QUICK	Terminate VTAM processing immediately without waiting for normal termination of existing sessions - should only be used when normal termination does not complete successfully
Z NET,CANCEL	Abends VTAM - should only be used when both Z NET and Z NET,QUICK are unsuccessful



TCP/IP stack

Communications Server for z/OS IP provides the industry-standard TCP/IP suite, allowing z/OS environments to share data and computing resources with other TCP/IP computing environments when authorized. Communications Server for z/OS IP enables anyone in a non-z/OS TCP/IP environment to access resources in the z/OS environment and perform tasks and functions provided by the TCP/IP suite. It provides the computer platform with the freedom desired by organizations to distribute workload to environments best suited to their needs. Communications Server for z/OS IP, therefore, adds the z/OS environment to the list of environments in which an organization can share data and computer processing resources in a TCP/IP network.

Communications Server for z/OS IP supports two environments:

1. It provides a native MVS (z/OS) environment in which users can exploit the TCP/IP protocols in the z/OS applications environment. This includes batch jobs, started tasks, TSO, CICS® applications, and IMS applications.
2. It also provides native TCP/IP support in the UNIX Systems Services environment in which users can create and use applications that conform to the POSIX or XPG4 standard (a UNIX specification). The UNIX environment and services can also be exploited from the z/OS environment and vice versa.

This chapter will help you to:

- ▶ Configure TCP/IP
- ▶ Understand the role of the Resolver
- ▶ Define the Resolver address space
- ▶ Define the TCP/IP stack
- ▶ Define OSA cards to the TCP/IP stack
- ▶ Define Virtual LANs (VLAN)
- ▶ Assign IP addresses to the networking interfaces
- ▶ Define static routes

3.1 z/OS Communications Server overview

- ❑ z/OS Communications Server TCP/IP protocol suite functions include the following categories:
 - TCP/IP protocol stack
 - Connectivity and gateway functions
 - Network protocol layer
 - Transport layer
 - File systems
 - Application Programming Interfaces

Figure 3-1 z/OS Communications Server overview

TCP/IP protocol stack

In 1983, the new protocol suite TCP/IP was adopted as a standard, and all hosts on the network were required to use it. The Transmission Control Protocol (TCP) and the Internet Protocol (IP) refer to a non-proprietary protocol suite that enables different packet-switched networks to function as a single entity regardless of underlying network topology.

Like SNA, TCP/IP is a set of communication protocols used between physically separated computer systems. Unlike SNA and most other protocols, TCP/IP is not designed for a particular hardware technology. TCP/IP can be implemented on a wide variety of physical networks, and is specifically designed for communicating between systems on different physical networks (local and wide area). This is called *internetworking*.

TCP/IP connectivity and gateway functions

TCP/IP connectivity and gateway functions handle the physical interfaces and the routing of IP data packets called datagrams.

Network protocol layer

The network protocol layer provides the support for the IP protocol. All TCP and User Datagram Protocol (UDP) data goes through the IP layer when entering and leaving the host. TCP and UDP use the IPv4 routing layer or the IPv6 routing layer. The network layer also provides support for the Internet Control Message Protocol (ICMP) and ICMPv6. This is used by the IP layer to exchange information and error messages with IP layers on other hosts and routers. ICMP is used for the IPv4 protocol and ICMPv6 is used for the IPv6 protocol.

Transport layer

The transport layer provides the support for the TCP, UDP, and RAW protocols. All three protocols use IPv4 or IPv6 as the network layer. The TCP protocol provides a connection-oriented, reliable transport layer, whereas UDP provides a simpler, connectionless and unreliable transport layer. The RAW transport layer provides for a more direct interface to the IP layer, which is primarily used by system-management type applications.

File systems

The file system layer provides the main interface between the application programming interfaces (APIs) and the transport layers. The first component of the file system layer is the z/OS UNIX logical file system (LFS). The LFS provides the API layer with a common interface to access files and sockets. In a POSIX-compliant environment, applications can access both files and sockets in a similar fashion. For example, both files and sockets are represented by a 32-bit integer referred to as a descriptor. Common functions can be used to access both file and socket resources.

Application programming interfaces

Each of the application programming interfaces (APIs) can be used to interface with the TCP/IP Services protocol stack provided by z/OS Communications Server. All of the APIs, with the exception of the PASCAL API, interface with the LFS layer.

The APIs are divided into the following two categories:

- ▶ TCP/IP socket APIs provided by z/OS Communications Server
- ▶ z/OS UNIX APIs

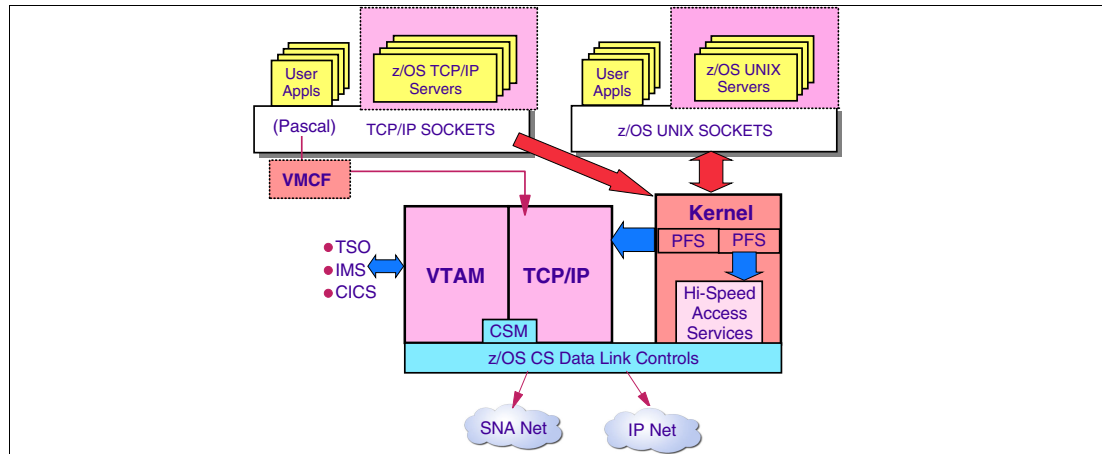


Figure 3-2 z/OS Communications Server APIs

3.2 Networking on the mainframe

- ❑ Networks and mainframes
 - Examples of mainframe-based networks
- ❑ Supporting the network
- ❑ Basic elements of a network
- ❑ Overview of mainframe network capabilities
- ❑ Security in a network

Figure 3-3 Networking on the mainframe concepts

Networks and mainframes

Regardless of which elements comprise a particular network, you—the end user—are the ultimate source and destination of the information that flows through it. Today's online transaction processing increasingly requires support for transactions that span a network and may include more than one company. In today's competitive market, responsiveness to customer or supplier demand is often a decisive factor in the success of an organization. The network is considered one of the most critical resources in an organization, both in the private and public sectors.

Examples of mainframe-based networks

Mainframes are used by large organizations as their central transaction processing system. Transaction processing in this context requires high availability, security, performance, and responsiveness. The design intention of the mainframe, and most of its evolution, is for the mainframe to be a highly available transaction processing server. Obviously, central processing capabilities are evolving to handle more and more transactions. However, in order to be an effective transaction processing server, there must be a proportional capability of moving data in and out of the central processor complex rapidly (CPC, the physical collection of hardware that consists of main storage, one or more central processors, timers, and channels). The result is that the input/output (I/O) options, capabilities and configuration choices of an IBM mainframe are varied, complex, and very performance-oriented.

Supporting the network

Network communications has both a software and a hardware aspect, and a separation of software and hardware administrative duties is common in large enterprises. The network administrator, a skilled software data communication expert, however, needs to understand both aspects.

Basic elements of a network

Basic elements of a computer network include hardware, software, and protocols. The interrelationship of these basic elements constitutes the infrastructure of the network. VTAM follows SNA protocols and uses SNA concepts to connect and communicate with elements in a data communication network. Each element in a SNA network to which a data or control message can be sent is assigned a network address. Each element with such an address is known as a network accessible unit (NAU). The network address uniquely identifies the element and contains the information necessary to route a message to its destination.

Overview of mainframe network capabilities

IBM's current mainframe technology provides significantly large servers with a distinctive strength of handling a high volume of transactions and input/output operations in parallel. The mainframe is capable of serving a large number of network nodes geographically dispersed across the world while simultaneously handling a high volume of input and output operations to disk storage, printers, and other attached computers. TCP/IP applications contact a name server whenever it is necessary to translate a domain name into an IP address, or when information is required about a domain. The name server performs the translation if it has the necessary information.

Security in a network

The z/OS Communications Server, along with other elements of z/OS, provide numerous enterprise-strength security services to protect your mission-critical data. This topic provides an overview of these technologies and how they can be used for a safe and secure z/OS TCP/IP deployment. In the networking environment, the information that is being protected is the data being read and written through sockets. Sockets are opened and used by applications running under user IDs. Today more than ever, businesses depend on the critical data that flows over networks. A large amount of sensitive and confidential data is stored and retrieved from z/OS systems, so the data that moves through z/OS-attached networks must be secured, have high integrity, and be available at all times. The mainframe environment includes both hardware and software tools that meet these goals.

3.3 Hardware network connections

- Network connections
- Channel subsystem (CSS)
- Mainframe channel subsystem and network links
- Hardware channels
- HiperSockets
- I/O cage

Figure 3-4 Hardware network connections

Network connections

Network connections can be made in several different fashions. The mainframe originally relied upon the channel subsystem to offload I/O processing to channel programs. DASD is still accessed using ESCON channels, but for networking connectivity, OSA-Express cards offer better performance and availability. Connectivity is the pipeline through which data is exchanged between clients and servers via physical and logical communication interfaces and the network. The IBM System z servers provide a wide range of interface options for connecting your z/OS system to an IP network or to another IP host. Some interfaces offer point-to-point or point-to-multipoint connectivity, while others support Local Area Network (LAN) connectivity.

The OSA-Express and OSA-Express2 cards provide redundancy capability, as well as throughput improvements when running in QDIO mode. QDIO mode allows direct access to central memory. QDIO mode can be emulated within a CPC by allowing memory-to-memory data transfer among LPARs running z/VM®, Linux, or z/OS.

Channel subsystem (CSS)

I/O channels are components of the System z channel subsystem (CSS) and z/Architecture. They provide a pipeline through which data is exchanged between servers, or between a server and external devices. z/Architecture channel connections are referred to as channel paths. The heart of moving data into and out of a mainframe host is the channel subsystem, or CSS. The CSS is, from a central processor standpoint, independent of the processors of

the mainframe host itself. This means that I/O within a mainframe host can be done asynchronously.

Mainframe channel subsystem and network links

A CHPID no longer directly corresponds to a hardware channel, and CHPID numbers can be arbitrarily assigned. A hardware channel is now identified by a physical channel identifier, or PCHID. End nodes need to predefine a link to their network node server (and usually to their backup network node server) and can also predefine links to other nodes. The predefined link to the network node server is used for CP-CP sessions. Dynamically created connection network links are used only for LU-LU sessions and are typically brought down when the sessions end.

Hardware channels

There are effectively three ways that network traffic can travel between an external network and a z/OS host. They are as follows:

- ▶ **Channel command word (CCW) channels** - CCW-based channels include parallel, ESCON, and FICON channels. A CCW can also be used to talk to an OSA card (this is discussed further in the topic on Open Systems Adaptor, OSA).
- ▶ **Coupling channels** - Communication among LPARs can be facilitated by coupling facility (CF) links. Coupling Facility links are used to support the cross-system coupling facility, or XCF. The XCF component in turn can be used to support the IP protocol.
- ▶ **Open Systems Adapter (OSA)** - The Open Systems Adapter is actually a network controller that you can install in a mainframe I/O cage. The adapter integrates several hardware features and supports many networking transport protocols. The OSA card is the strategic communications device for the mainframe architecture. It has several key features that distinguish it from CCW-based communications.

HiperSockets

Mainframe HiperSockets is a technology that provides high-speed TCP/IP connectivity within a central processor complex. It eliminates the need for any physical cabling or external networking connection between servers running in different LPARs. The HiperSockets hardware device is represented by the IQD CHPID and its associated subchannel devices. All LPARs that are configured (HCD) to use the same IQD CHPID have internal connectivity and therefore have the capability to communicate using HiperSockets. Each HiperSockets LAN uses a Channel Path ID (CHPID) with a channel type IQD. If the system supports multiple channel subsystems, and if HiperSockets connectivity is required across multiple channel subsystems, the IQD CHPID must also be configured (HCD) to span the applicable channel subsystems. The IQD CHPID can be viewed as a logical LAN within the CPC.

I/O cage

The connections to the central processor complex are made in a physical area of the processor frame called an I/O cage. Within the cage, OSA cards and memory modules (and other devices) are physically attached to the central processor complex. Parallel, FICON, and ESCON connections are all made within the cage as well, using an adapter card.

3.4 TCP/IP networking

- ❑ TCP/IP architectural model
 - Build an interconnection of networks
- ❑ Internetworking
 - Collection of individual networks
- ❑ TCP/IP protocol layers
 - Communication software divided into layers
- ❑ TCP/IP applications
 - Communicate with applications on other internet hosts

Figure 3-5 TCP/IP networking concepts

TCP/IP architecture

The TCP/IP protocol suite is so named for two of its most important protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP). A less used name for it is the *Internet Protocol Suite*, which is the phrase used in official Internet standards documents. When computers communicate with one another, certain rules, or protocols, allow them to transmit and receive data in an orderly fashion. Throughout the world, one of the most routinely used sets of protocols is the Transmission Control Protocol/Internet Protocol (TCP/IP).

TCP/IP is a suite of protocols that specify communications standards between computers and detail conventions for routing and interconnecting networks. It is used extensively on the Internet and consequently allows research institutions, colleges and universities, government, and industry to communicate with each other.

TCP/IP allows communication between a number of computers (called hosts) connected on a network. Each network can be connected to another network to communicate with hosts on that network. Although there are many types of network technologies, many of which operate with packet-switching and stream transport, TCP/IP offers one major advantage: hardware independence.

Internetworking

The main design goal of TCP/IP was to build an interconnection of networks, referred to as an *internetwork*, or *internet*, that provided universal communication services over heterogeneous physical networks. The clear benefit of such an internetwork is the enabling of

communication between hosts on different networks, perhaps separated by a large geographical area.

The words internetwork and internet are simply a contraction of the phrase interconnected network. However, when written with a capital “I”, the Internet refers to the worldwide set of interconnected networks. Therefore, the Internet is an internet, but the reverse does not apply. The Internet is sometimes called the *connected Internet*.

TCP/IP protocol layers

Like most networking software, TCP/IP is modeled in layers. This layered representation leads to the term *protocol stack*, which refers to the stack of layers in the protocol suite. It can be used for positioning (but not for functionally comparing) the TCP/IP protocol suite against others, such as Systems Network Architecture (SNA) and the Open System Interconnection (OSI) model. Functional comparisons cannot easily be extracted from this, because there are basic differences in the layered models used by the different protocol suites.

By dividing the communication software into layers, the protocol stack allows for division of labor, ease of implementation and code testing, and the ability to develop alternative layer implementations. Layers communicate with those above and below via concise interfaces. In this regard, a layer provides a service for the layer directly above it and makes use of services provided by the layer directly below it. For example, the IP layer provides the ability to transfer data from one host to another without any guarantee of reliable delivery or duplicate suppression. Transport protocols such as TCP make use of this service to provide applications with reliable, in-order data stream delivery.

TCP/IP applications

The highest level protocols within the TCP/IP protocol stack are application protocols. They communicate with applications on other internet hosts and are the user-visible interface to the TCP/IP protocol suite.

All application protocols have some characteristics in common. They can be user-written applications or applications standardized and shipped with the TCP/IP product. The TCP/IP protocol suite includes application protocols such as:

- ▶ Telnet for interactive terminal access to remote internet hosts
- ▶ File Transfer Protocol (FTP) for high-speed disk-to-disk file transfers
- ▶ Simple Mail Transfer Protocol (SMTP) as an internet mailing system

These are some of the most widely implemented application protocols, but many others exist. Each particular TCP/IP implementation will include a lesser or greater set of application protocols.

3.5 TCP/IP stack - TCP/IP address space

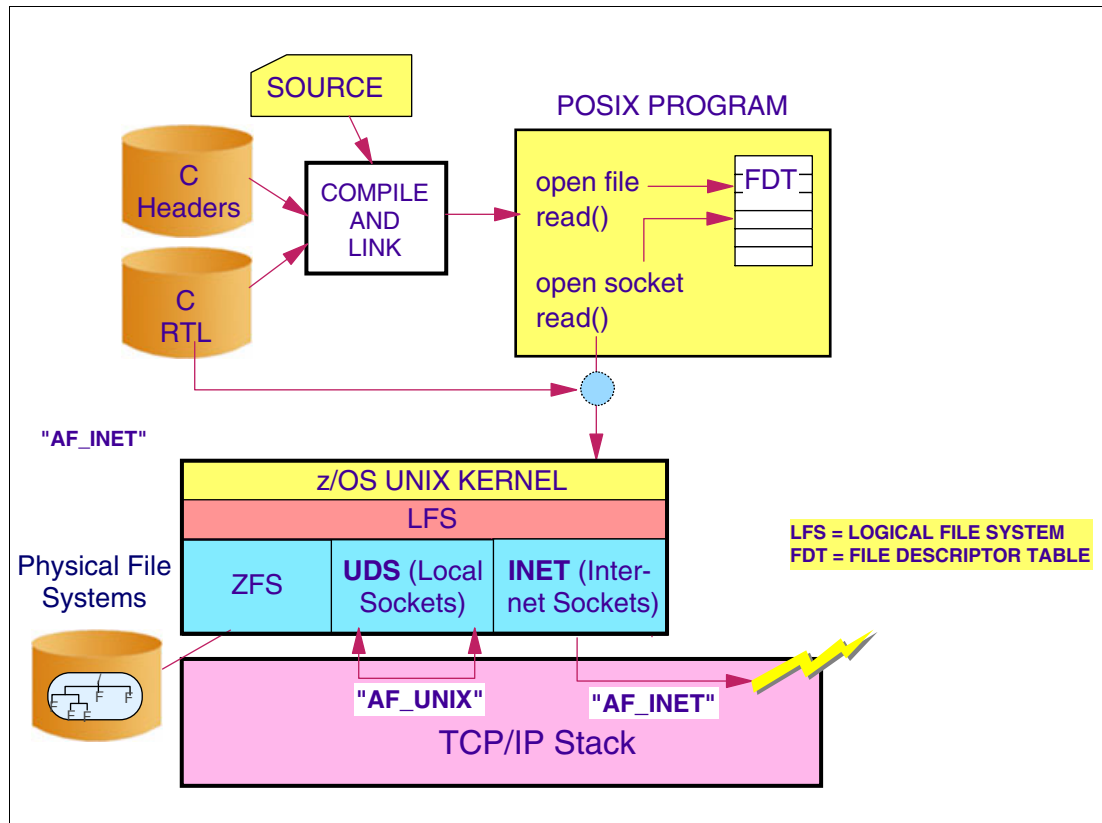


Figure 3-6 TCP/IP address space (TCP/IP stack)

TCP/IP stack

The TCP/IP address space is where the TCP/IP protocol suite is implemented for Communications Server for z/OS TCP/IP. The TCP/IP address space is commonly referred to as a stack. In earlier versions Open Edition (OE) TCP/IP could run either as a stand-alone or in parallel with the TCP/IP for MVS. This was often necessary since the OE stack did not support all the functions and network connections available with TCP/IP for MVS. Communications Server for z/OS IP now has a highly efficient direct communication between the UNIX System Services address space (OMVS) and a TCP/IP stack that was integrated in UNIX System Services. This communication path includes the UNIX System Services Physical File System (PFS) component for AF_INET and AF_INET6 (Addressing Family-Internet) sockets communication.

Note: TCP/IP uses services from other address spaces in its processing. While TCP/IP waits for availability of those services, it is recommended that you verify that the OMVS, resolver, and VTAM address spaces have completed initialization before starting TCP/IP. If using automation to start TCP/IP, have it look for the BPXI004I, EZZ9291I, and IST020I messages, respectively, before issuing the **START** command.

Enter the MVS **START** command from the operator's console to start TCP/IP, specifying the member name of your cataloged procedure. This will start the TCP/IP address space and any of the servers you have defined in PROFILE.TCPIP. For example, if the procedure to start the TCP/IP address space was called TCP1 in your PROCLIB, you would enter:

```
START TCP1
```

3.6 TCP/IP terminology

- ❑ **Host:** Any systems attached to an IP network
- ❑ **Interface:** A logical representation of a network interface card (OSA)
- ❑ **Port:** An entrance to or exit from a network. The part of a socket address that identifies a port within a host.
- ❑ **Socket:** A logical entity used to identify a remote application - **socket = <IP address> + a port number**
- ❑ **Router:** Connects networks and routes packets between them

Figure 3-7 TCP/IP terminology

TCP/IP terminology

The following terminology is commonly used to describe a TCP/IP environment:

- Host** In the Internet suite of protocols, this is an end system. The end system can be any workstation; it does not have to be a mainframe.
- Interface** A functional unit that interconnects two computer networks with different network architectures. An *interface* connects networks or systems of different architectures. A *bridge* interconnects networks or systems with the same or similar architectures. In TCP/IP, this is a synonym for *router*.
- Port** Each process that wants to communicate with another process identifies itself to the TCP/IP protocol suite by one or more ports. A port is a 16-bit number used by the host-to-host protocol to identify to which higher level protocol or application program (process) it must deliver incoming messages. There are two types of ports:
- Well-known** - Well-known ports belong to standard servers; for example, Telnet uses port 23. Well-known port numbers range between 1 and 1023 (prior to 1992, the range between 256 and 1023 was used for UNIX-specific servers). Well-known port numbers are typically odd, because early systems using the port concept required an odd/even pair of ports for duplex operations. Most servers require only a single port. The well-known ports are controlled and assigned by the Internet central authority (IANA) and on most systems can only be used by system processes or by programs executed by privileged users. The reason for

well-known ports is to allow clients to be able to find servers without configuration information.

Ephemeral - Clients do not need well-known port numbers because they initiate communication with servers and the port number they are using is contained in the UDP datagrams sent to the server. Each client process is allocated a port number as long as it needs it by the host it is running on. Ephemeral port numbers have values greater than 1023, normally in the range 1024 to 65535.

- Socket** An endpoint for communication between processes or application programs. A synonym for port.
- Socket address** The address of an application program that uses the socket interface on the network. In Internet format, it consists of the IP address of the socket's host and the port number of the socket. The application program is usually not aware of the structure of the address.
- Socket interface** A Berkeley Software Distribution (BSD) application programming interface (API) that allows users to easily write their own programs.
- Router** A router interconnects networks at the internetwork layer level and routes packets between them. The router must understand the addressing structure associated with the networking protocols it supports and make decisions on whether, or how, to forward packets. Routers are able to select the best transmission paths and optimal packet sizes. The basic routing function is implemented in the IP layer of the TCP/IP protocol stack, so any host or workstation running TCP/IP over more than one interface could, in theory and also with most of today's TCP/IP implementations, forward IP datagrams. However, dedicated routers provide much more sophisticated routing than the minimum functions implemented by IP.

3.7 IPv4 addressing

32 bit format

- 0000 0000 . 0000 0000 . 0000 0000 . 0000 0000
- Example:
 - 9.12.1.43
 - 0000 1001 . 0000 1100 . 0000 0001 . 0010 1011

← Network part of address Host part of address →

IP address classes

Class	Address Range (First Octet)	Default Subnet Mask	Network Host ID Split
A	0 to 127	225.0.0.0	N.h.h.h
B	128 to 191	255.255.0.0	N.N.h.h
C	192 to 223	255.255.255.0	N.N.N.h

Figure 3-8 IPv4 addressing

IP addresses

An Internet Protocol (IP) address is a numerical label that is assigned to devices participating in a computer network that uses the Internet Protocol for communication between its nodes. An IP address serves two principal functions, as follows:

- ▶ Host or network interface identification
- ▶ Location addressing

Its role has been characterized as follows:

- ▶ A name indicates what we seek.
- ▶ An address indicates where it is.
- ▶ A route indicates how to get there.

IP history

The designers of TCP/IP defined an IP address as a 32-bit number and this system, known as Internet Protocol Version 4 or IPv4, is still in use today. However, due to the enormous growth of the Internet and the resulting depletion of available addresses, a new addressing system (IPv6), using 128 bits for the address, was developed in 1995 and last standardized by RFC 2460 in 1998. Although IP addresses are stored as binary numbers, they are usually displayed in human-readable notations, such as:

208.77.188.166 (for IPv4)
2001:db8:0:1234:0:567:1:1 (for IPv6)

IPv4 addresses

An IPv4 address enables data to be routed to the chosen destination. Each destination in your network, as well as any other TCP/IP network you have access to, can be uniquely identified by its assigned address. TCP/IP protocols rely upon a strict system of addressing in order to reach a host in a network. IPv4 addresses are represented in dotted-decimal format. The 32-bit address is divided along 8-bit boundaries. Each set of 8 bits is converted to its decimal equivalent and separated by periods.

An IPv4 TCP/IP address is a 32-bit number written in dotted decimal notation. This scheme is numeric and consists of four groups separated by a period (.). For example, 9.12.1.43 represents a single host on a single network. 193.5.86.9 represents another host on another network.

Subnet mask

Use the subnet_mask to completely define the route. Multiple network routes having an identical destination IP address and address mask can be specified.

Network and host addresses

Each host must have a unique Internet address to communicate with other hosts on the Internet. The network address part of the IP address is centrally administered by the Internet Network Information Center (the InterNIC) and is unique throughout the Internet. Each IP address is made up of two logical addresses:

IP address = <network address> <host address>

The components of the name are:

Network address Represents a specific physical network within the Internet

Host address Specifies an individual host within the physical network identified by the network address.

For example, 9.12.1.43 is an IP address with 9.12 being the network address and 1.43 being the host address.

IP address classes

In the original Internet routing scheme developed in the 1970s, sites were assigned addresses from one of three classes: Class A, Class B, and Class C. The address classes differ in size and number. Class A addresses are the largest, but there are few of them. Class Cs are the smallest, but they are numerous. Classes D and E are also defined, but not used in normal operation.

- ▶ Class A - 0nnnnnnn hhhhhhhh hhhhhhhh hhhhhhhh
First bit 0; 7 network bits; 24 host bits
Initial byte: 0 - 127
126 Class As exist (0 and 127 are reserved)
16,777,214 hosts on each Class A
- ▶ Class B - 10nnnnnn nnnnnnnn hhhhhhhh hhhhhhhh
– First two bits 10; 14 network bits; 16 host bits
– Initial byte: 128 - 191
– 16,384 Class Bs exist
– 65,532 hosts on each Class B
- ▶ Class C - 110nnnnn nnnnnnnn nnnnnnnn hhhhhhhh
– First three bits 110; 21 network bits; 8 host bits
– Initial byte: 192 - 223
– 2,097,152 Class Cs exist
– 254 hosts on each Class C

3.8 IPv6 addresses

- ❑ 128 bit format
 - X:X:X:X:X:X:X:X
 - Examples: Each X representing 16 bits (double IPv4 address) written in Hexadecimal as follows:
 - **FEDC:BA98:7654:3210:FEDC:BA98:7654:3210**
 - **1080:0:0:0:8:800:200C:417A**
- ❑ If you actually try to calculate it, the number of addresses that 128 bits could generate is:
 - 340 billion billion billion billion (340^{38})

Figure 3-9 IPv6 addresses

IPv6 addresses

IPv6 addresses are 128 bits divided along 16-bit boundaries. Each 16-bit block is converted to a 4-digit hexadecimal number and separated by colons. The resulting representation is called colon-hexadecimal. The major drawback of this addressing system is that, for most people, numbers are difficult to remember.

IPv6 TCP/IP address

An IPv6 TCP/IP address is a 128-bit number written in colon hexadecimal notation. This scheme is hexadecimal and consists of eight 16-bit pieces of the address. For example, x:x:x:x:x:x:x:x represents a single host on a single network. Alternate notations described in RFC 2373 are acceptable, for example:

FEDC:BA98:7654:3210:FEDC:BA98:7654:321 or ::1.

Note: Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero bits easier a special syntax is available to compress the zeros. The use of "::" indicates multiple groups of 16-bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress the leading or trailing zeros in an address.

3.9 Internet terms for TCP/IP

- ☐ Internet terms used in relation to TCP/IP
 - Client
 - Host
 - Network
 - Packet
 - Port
 - Process
 - Protocol
 - Server

Figure 3-10 TCP/IP terminology

TCP/IP terminology

Become familiar with the following terms, which are used when discussing TCP/IP:

- | | |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Client | A computer or process that accesses the data, services, or resources of another computer or process on the network. |
| Host | A computer that is attached to an Internet network and can communicate with other Internet hosts. The local host for a particular user is the computer at which that user is working. A foreign host is any other host name on the network. From the point of view of the communications network, hosts are both the source and destination of packets. Any host can be a client, a server, or both. On an Internet network, a host is identified by its Internet name and address. |
| Network | The combination of two or more hosts and the connecting links between them. A physical network is the hardware that makes up the network. A logical network is the abstract organization overlaid on all or part of one or more physical networks. The Internet network is an example of a logical network. The interface program handles translation of logical network operations into physical network operations. |
| Packet | The block of control information and data for one transaction between a host and its network. Packets are the exchange medium used by processes to send and receive data through Internet networks. A packet is sent from a source to a destination. |

Port	A logical connecting point for a process. Data is transmitted between processes through ports (or sockets). Each port provides queues for sending and receiving data. In an interface program network, each port has an Internet port number based on how it is being used. A particular port is identified with an Internet socket address, which is the combination of an Internet host address and a port number.
Process	A program that is running. A process is the active element in a computer. Terminals, files, and other I/O devices communicate with each other through processes. Thus, network communications is interprocess communications (that is, communication between processes).
Protocol	A set of rules for handling communications at the physical or logical level. Protocols often use other protocols to provide services. For example, a connection-level protocol uses a transport-level protocol to transport packets that maintain a connection between two hosts.
Server	A computer or process that provides data, services, or resources that can be accessed by other computers or processes on the network.

3.10 TCP/IP suite of protocols

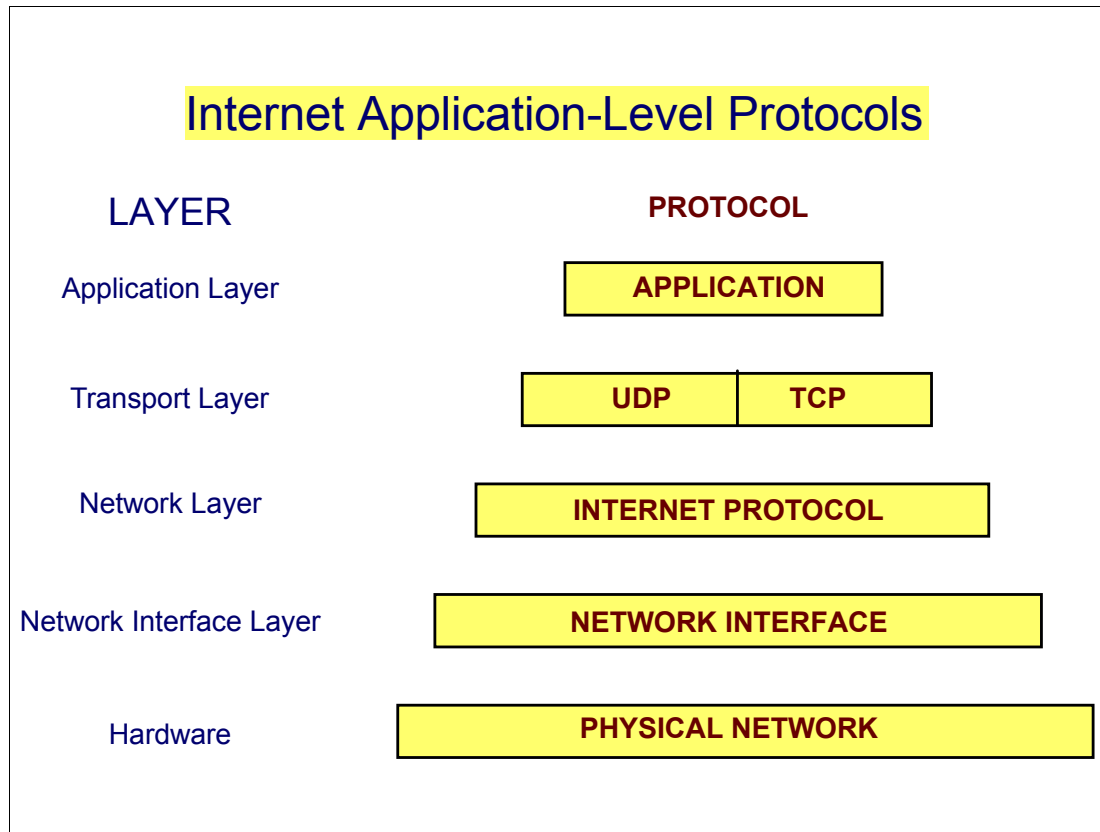


Figure 3-11 TCP/IP suite of protocols

TCP/IP protocols

TCP/IP is modeled in layers that have the following characteristics:

- Application layer** When an application needs to send data to another application on another host, the applications send the information down to the transport level protocols to prepare the information for transmission. TCP/IP implements higher-level Internet protocols at the application program level.
- Transport layer** The transport layer provides the support for the TCP, User Datagram Protocol (UDP), and RAW protocols. All three protocols use IPv4 or IPv6 as the network layer. The TCP protocol provides a connection-oriented, reliable transport layer, whereas UDP provides a simpler, connectionless and unreliable transport layer. The RAW transport layer provides for a more direct interface to the IP layer, which is primarily used by system management type applications.
- Network layer** The network protocol layer provides the support for the IP protocol. All TCP and UDP data goes through the IP layer when entering and leaving the host. TCP and UDP use the IPv4 routing layer or the IPv6 routing layer.
- The network layer also provides support for the Internet Control Message Protocol (ICMP) and ICMPv6. This is used by the IP layer to exchange information and error messages with IP layers on other

hosts and routers. ICMP is used for the IPv4 protocol and ICMPv6 is used for the IPv6 protocol.

Network interface A network interface is the physical connection between a node and a network. Each network will have one or more network interfaces on each cluster node.

Hardware The physical network consists of the cables (coaxial cable, twisted pair, fiber optic, and telephone lines) that connect the different hardware residing on the network, the adapter cards used on computers connected to the network (hosts), and any concentrators, repeaters, routers, or bridges used in the network.

Physical networks vary both in size and in the type of hardware used. The two common kinds of networks are local area networks (LANs) and wide area networks (WANs). A LAN is a network where communications are limited to a moderately sized geographic area of 1 to 10 km (1 to 6 miles), such as a single office building, warehouse, or campus. A WAN is a network providing data communications capability throughout geographic areas larger than those serviced by LANs, such as across a country or across continents. An intermediate class of networks exists also, called metropolitan area networks (MANs). This guide does not generally distinguish MANs; they are grouped with WANs.

LANs commonly use Standard Ethernet, IEEE 802.3 Ethernet, or token-ring hardware for the physical network, while WANs and asynchronous networks use communications networks provided by common carrier companies. Operation of the physical network in both cases is usually controlled by networking standards from organizations such as the Electronics Industry Association (EIA) or the International Telecommunication Union (ITU).

3.11 Sending application data to the host

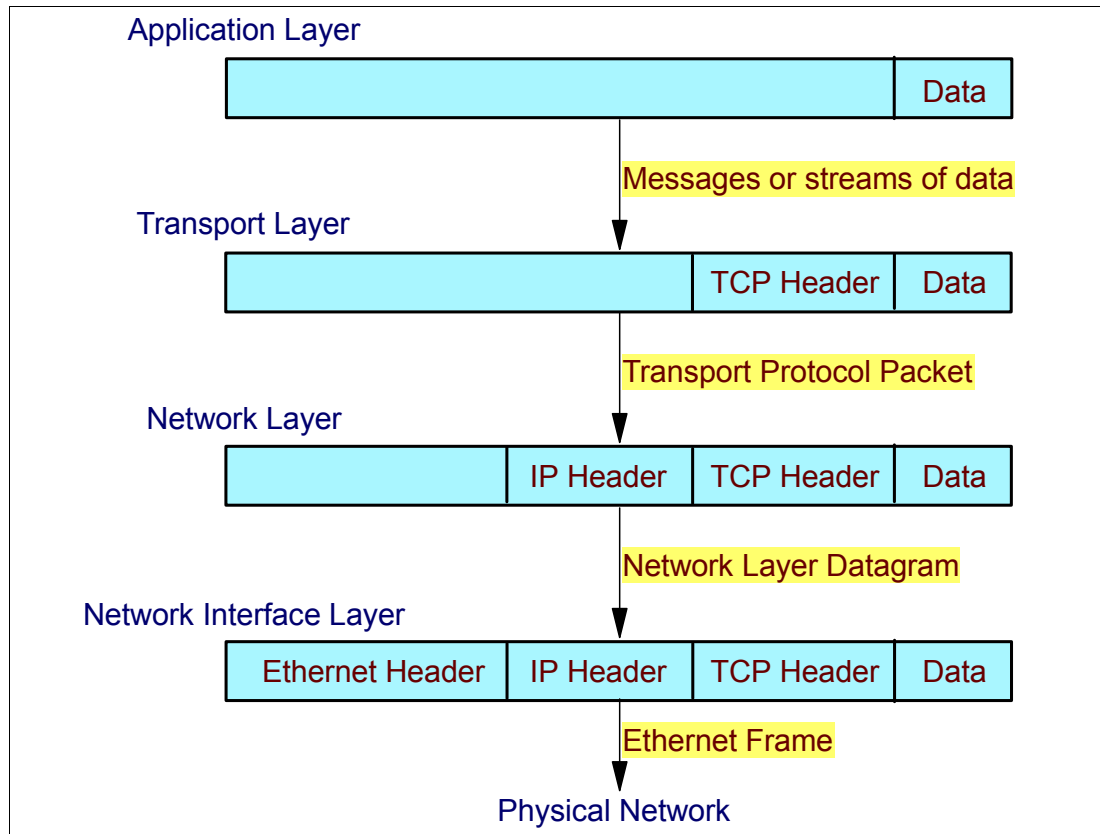


Figure 3-12 Transmission of information from the sender application to the receiver host

Transmission of application data

TCP/IP carefully defines how information moves from sender to receiver. The layers pass data only to the layers immediately above and below. Each layer adds its own header information (and footer information, in the case of the Data Link), effectively encapsulating the information received from the higher layers.

Application layer

The application layer consists of the application. When an application needs to send data to another application on another host, the applications send the information down to the transport level protocols to prepare the information for transmission. The application programs send messages or streams of data to one of the Internet Transport Layer Protocols, either the User Datagram Protocol (UDP) or the Transmission Control Protocol (TCP).

Transport layer

The transport layer contains UDP and TCP. These protocols receive the data from the application, divide it into smaller pieces called packets, add a destination address, and then pass the packets along to the next protocol layer, the Internet Network layer.

Network layer

The network layer contains the network (hardware) interface. The Internet network layer encloses the packet in an Internet Protocol (IP) datagram, puts in the datagram header and

trailer, decides where to send the datagram (either directly to a destination or else to a gateway), and passes the datagram on to the Network Interface layer.

Network interface layer

Finally, the hardware layer contains the physical network. The Network Interface layer accepts IP datagrams and transmits them as frames over specific network hardware, such as Ethernet or Token-Ring networks.

Note: The type of hardware most widely used throughout LANs is what is commonly known as Ethernet. It consists of a single cable with hosts being attached to it through connectors, taps, or transceivers. Simple Ethernets are quite inexpensive to install, which, together with a net transfer rate of 10 Megabits per second accounts for much of their popularity.

Information from host to application

Frames received by a host go through the protocol layers in reverse. Each layer strips off the corresponding header information, until the data is back at the application layer.

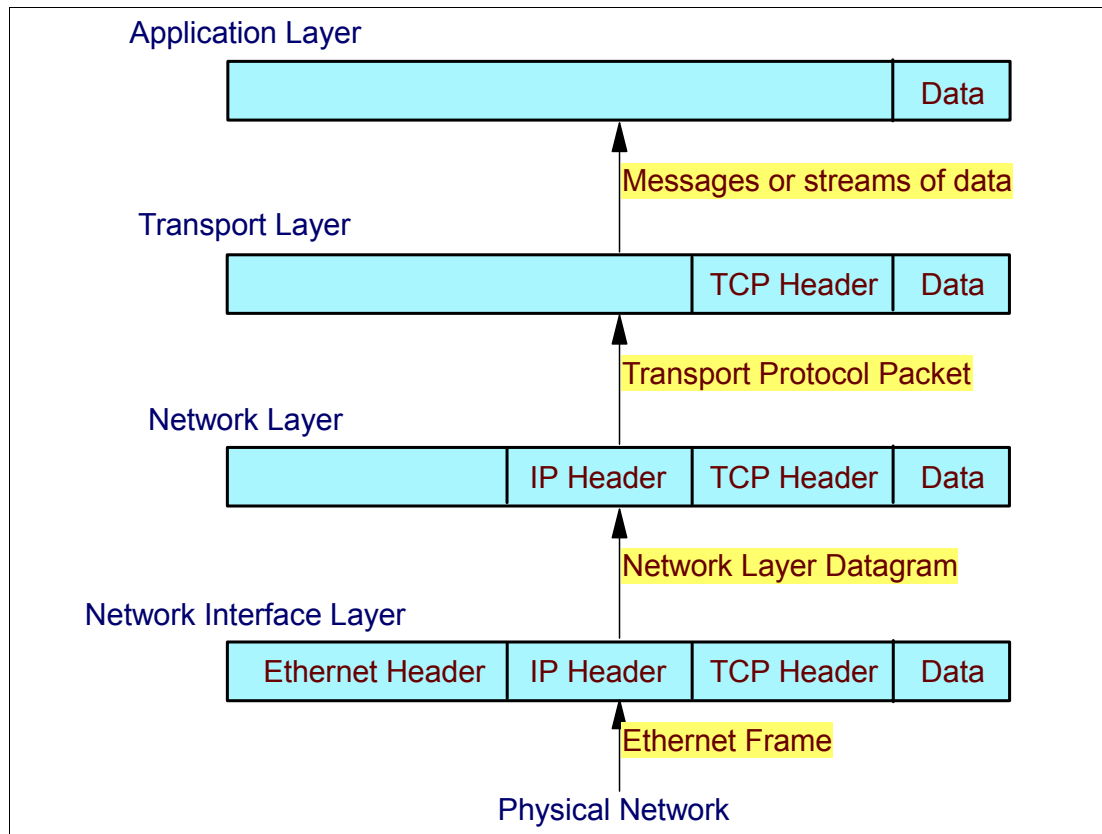


Figure 3-13 Sending information from the host to the application

3.12 Host configuration tasks

- **Setting up the host configuration tasks**
 - **The network interface**
 - **Set the Internet addresses**
 - **Set the host names**
 - **Set up static routes to gateways or other hosts**
 - **Specify daemons to be started by default**
 - **Set up the /etc/hosts file for name resolution (or set up the host to use a name server for name resolution)**

Figure 3-14 Host configuration tasks

Host configuration tasks

Each host machine on your network must be configured to function according to the needs of the end users and the network as a whole. For each host on the network, you must configure the following components:

- ▶ Network interface
- ▶ Internet address
- ▶ Host name
- ▶ Static routes to gateways or other hosts
- ▶ Daemons to be started by default
- ▶ The /etc/hosts file for name resolution (or set up the host to use a name server for name resolution)

The network interface

A network interface is the physical connection between a node and a network. As a system administrator, you may need to perform several diverse tasks relating to cluster network interfaces. The network interface is the network-specific software that communicates with the network-specific device driver and the IP layer in order to provide the IP layer with a consistent interface to all network adapters that might be present.

The IP layer selects the appropriate network interface based on the destination address of the packet to be transmitted. Each network interface has a network address. The Network Interface layer is responsible for adding or removing any link layer protocol header required to

deliver a message to its destination. The network adapter device driver controls the network adapter card.

Set the Internet address

An address enables data to be routed to the chosen destination. Each destination in your network, as well as any other TCP/IP network you have access to, can be uniquely identified by its assigned address (either a 32-bit IPv4 address in dotted decimal notation, or a 128-bit IPv6 address in colon hexadecimal notation). A HOME statement defines the IP addresses of each link in the host as specified on the TCP/IP address space configuration statements.

Set the host name

The purpose of using names for hosts is to provide a quick, easy, and unambiguous way to refer to the computers in your network. Internet system administrators have discovered that there are good, as well as poor, choices for host names. Use the HOSTNAME statement to specify the TCP host name of this z/OS Communications Server server TCPIP.DATA configuration statements.

Set up static routes to gateways or other hosts

Figure 3-15 shows three network clouds numbered one, two, and three. Networks one and two are connected with gateway A. Networks two and three are connected with gateway B. To connect Network 1 directly to Network 2, you would use a single gateway (Router A). To connect Network 2 directly to Network 3, you would use another gateway (Router B). Now, assuming the proper routes are defined, all the users on all three networks can communicate.

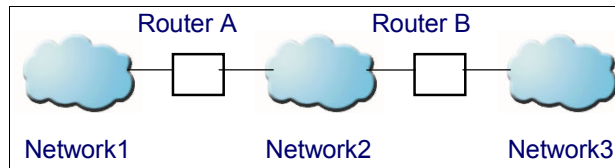


Figure 3-15 Simple gateway configuration

Specify daemons to be started by default

The `/etc/inetd.conf` file is the default configuration file for the `inetd` daemon. This file enables you to specify the daemons to start by default and supply the arguments that correspond to the desired style of functioning for each daemon. This file is part of TCP/IP in Network Support Facilities. The following daemons are controlled by the `inetd` daemon: `comsat`, `ftpd`, `telnetd`, `rshd`, `rlogind`, `rexecd`, `fingerd`, `tftpd`, `talkd`, and `uucpd`.

Set up the `/etc/hosts` file for name resolution

Even if you are using a hierarchical (or domain) naming scheme with name servers, you might want to configure the `/etc/hosts` file to identify hosts that are not known by the name servers. The `/etc/hosts` file contains the Internet Protocol (IP) host names and addresses for the local host and other hosts in the Internet network. This file is used to resolve a name into an address (that is, to translate a host name into its Internet address). When your system is using a name server, the file is accessed only if the name server cannot resolve the host name.

When the local host is using the DOMAIN protocol, the resolver routines query a remote DOMAIN name server before searching this file. In a flat network with no name server, the resolver routines search this file for host name and address data.

3.13 TCP/IP customization

- ❑ Four main configuration files to customize are:
 - TCPIP.DATA
 - PROFILE.TCPIP
 - HOSTS.LOCAL
 - ETC.IPNODES
- ❑ Use the following commands to verify customization:
 - TSO PING
 - TSO NETSTAT
 - TSO HOMETEST
 - TSO TRACERTE

Figure 3-16 TCP/IP customization tasks

TCP/IP customization

Before you begin customizing, it is assumed that you know what configuration data sets are used by the TCP/IP address space, their search order, and considerations for what type of TCP/IP stack you will be running in your environment (for example, Enterprise Extender (EE) and multiple stacks).

You should understand the relationships of TCP/IP configuration files as they apply to the TCP/IP address space. The four main configuration files that you will be working with are:

- ▶ TCPIP.DATA
- ▶ PROFILE.TCPIP
- ▶ HOSTS.LOCAL
- ▶ ETC.IPNODES

Verify TCP/IP customization

You should be able to use the following commands to verify customization:

- ▶ TSO PING and z/OS UNIX ping
 - Sends IP datagrams to a specified destination host, requesting a reply, and measures the round trip time. This helps you to verify the interfaces defined to the TCP/IP address space.

- ▶ TSO NETSTAT, z/OS UNIX netstat, and DISPLAY TCPIP,NETSTAT

Queries TCP/IP about the network status of the local host or the contents of the resolver cache. With Netstat, you can verify most TCP/IP customization values that can be set from PROFILE.TCPIP. You can also display detailed information about the contents of the system-wide resolver cache, or statistical information such as the number of cache queries.

- ▶ TSO HOMETEST

Verifies your host name and address configuration.

- ▶ TSO TRACERTE and z/OS UNIX traceroute

Displays the route that a packet takes to reach a requested destination.

Note: TCP/IP is UNIX. When UNIX became part of MVS, the TCP/IP UNIX files suddenly had equivalent MVS data sets.

3.14 Configuration files used by TCP/IP

- ❑ Files used by the TCP/IP stack
 - **PROFILE.TCPIP** - used only for the stack
 - System operation and configuration parameters
 - A sample data set, hlq.SEZAINST(SAMPPROF), can be copied and modified for use as your default configuration
 - **TCPIP.DATA** - used by stack and applications
 - Configuration information used by TCP/IP clients
 - Create a TCPIP.DATA file by copying the sample provided in SEZAINST(TCPDATA) and modifying it to suit your local conditions

Figure 3-17 Configuration files used by TCP/IP

Configuration files for TCP/IP

Two configuration files are used by the TCP/IP stack, PROFILE.TCPIP and TCPIP.DATA. PROFILE.TCPIP is used only for the configuration of the TCP/IP stack. TCPIP.DATA is used during configuration of both the TCP/IP stack and applications; the search order used to find TCPIP.DATA is the same for both the TCP/IP stack and applications.

PROFILE.TCPIP During initialization of the TCP/IP stack, system operation and configuration parameters for the TCP/IP stack are read from the configuration file PROFILE.TCPIP.

TCPIP.DATA The TCP/IP stack's configuration component uses the TCPIP.DATA configuration file during TCP/IP stack initialization to determine the stack's host name. To find the TCPIP.DATA information, the z/OS UNIX environment search order is used.

PROFILE.TCPIP data set

During TCP/IP address space initialization, a configuration profile data set (PROFILE.TCPIP) is read that contains system operation and configuration parameters. A sample data set, SEZAINST(SAMPPROF), can be copied and modified for use as your default configuration profile.

The PROFILE data set contains the following major groups of configuration parameters:

- ▶ TCP/IP operating characteristics
- ▶ TCP/IP physical characteristics
- ▶ TCP/IP reserved port number definitions (application configuration)
- ▶ TCP/IP network routing definitions
- ▶ TCP/IP diagnostic data statements

Note: Once this data set is defined, if you attempt to edit PROFILE.TCPIP while TCP/IP is active, and PROFILE.TCPIP is defined in the TCPIP PROC as a sequential data set (for example, //PROFILE DD DISP=SHR,DSNAME=TCPIP.PROFILE.TCPIP), the data set in use message might be displayed. To avoid this, specify FREE=CLOSE, as follows:

```
//PROFILE DD DISP=SHR,DSNAME=TCPIP.PROFILE.TCPIP,FREE=CLOSE
```

This allows you to edit the profile while TCP/IP is active. Typically, when TCP/IP starts, it keeps the PROFILE allocated and does not release the allocation until the end of the step (in this case, the end of the job). If you specify FREE=CLOSE, the release occurs once the data set is read. MVS releases the enqueue on the PROFILE, which allows you to edit it. If the PROFILE is a member of a PDS (for example, SYS1.TCPPARMS(PROFILE)), FREE=CLOSE is not needed.

TCPIP.DATA data set

The TCPIP.DATA configuration data set is the anchor configuration data set for the TCP/IP stack and all TCP/IP servers and clients running in z/OS. With a z/OS TCP/IP stack, you can define the TCPIP.DATA parameters in a z/OS UNIX file or in an MVS data set. The TCPIP.DATA configuration data set is read during initialization of all TCP/IP server and client functions. All functions must access this data set in order to find basic configuration information, such as the name of the TCP/IP address space, the TCP/IP host name, and the data set prefix to use when searching for other configuration data sets. The TCPIP.DATA file contains the following major groups of configuration parameters:

- ▶ Application operating characteristics
- ▶ Resolver operating characteristics
- ▶ Socket library diagnostic data statements
- ▶ Resolver diagnostic data statements

The TCPIP.DATA data set is also known as one of the resolver configuration data sets. In fact, this name is now more commonly used to refer to this important file in the UNIX System Services environment because the socket library contains a component called the resolver. In a UNIX system, you use the /etc/resolv.conf file for the same purpose as you use TCPIP.DATA for in your MVS system.

TCPIP.DATA specifies the name of the TCP/IP address space. Because the data set search order can vary, your installation will determine which data set you can use.

Create a TCPIP.DATA file by copying the sample provided in SEZAINST(TCPDATA) and modifying it to suit your local conditions.

3.15 TCP/IP data sets search order

- During initialization of the TCP/IP stack configuration parameters for the TCP/IP stack are read from the following data sets:
 - Search order for PROFILE.TCPIP data set
 - //PROFILE DD DSN=aaa.bbb.ccc(anyname)
 - jobname.nodename.TCPIP
 - hlq.nodename.TCPIP
 - jobname.PROFILE.TCPIP
 - TCPIP.PROFILE.TCPIP
 - Search order for TCPIP.DATA data set
 - DD statement //SYSTCPD
 - jobname.TCPIP.DATA data set
 - SYS1.TCPPARMS(TCPDATA) data set
 - hlq.TCPIP.DATA data set

Figure 3-18 TCP/IP data sets search order

Search order for TCPIP.PROFILE data set

During initialization of the TCP/IP stack, system operation and configuration parameters for the TCP/IP stack are read from the configuration file PROFILE.TCPIP. The search order used by the TCP/IP stack to find PROFILE.TCPIP involves both explicit and dynamic data set allocation as shown in Figure 3-18. The search order is:

1. PROFILE DD

If no PROFILE DD exists, search continues. Because the PROFILE DD is the first step in the search order, TCP/IP uses the data set as the PROFILE.TCPIP configuration file.

2. jobname.nodename.TCPIP

If jobname.nodename.TCPIP is found, the search stops here.

3. hlq.nodename.TCPIP

If hlq.nodename.TCPIP is found, the search stops here.

4. jobname.PROFILE.TCPIP

If jobname.PROFILE.TCPIP is found, the search stops here.

5. TCPIP.PROFILE.TCPIP

TCPIP.PROFILE.TCPIP is searched last if necessary.

Note: Explicitly specifying the PROFILE DD statement in the TCPIPROC JCL is the recommended way to specify PROFILE.TCPIP. If this DD statement is present, the data set it defines is explicitly allocated by MVS and no dynamic allocation is done. If this statement is not present, the search order continues to use dynamic allocation for the PROFILE.TCPIP.

Search order for TCPIP.DATA data set

To find TCPIP.DATA, NPF follows a standard search sequence. It reads each instance of TCPIP.DATA it can find, in order, until it finds values for both the TCP/IP configuration statements. It uses the first value that it finds for each statement. If an allocation fails, the data set does not exist, or the data set is not available, NPF goes to the next data set in the sequence. The search ends when it finds values for both the TCP/IP configuration statements or when all data sets have been checked. The search order is:

1. The data set pointed to by the DD statement //SYSTCPD.
2. A data set with the name of jobname.TCPIP.DATA, where jobname is the jobname for the NPF capture point application or the TSO user ID for the TSO session (panels).
3. A data set with the name of SYS1.TCPPARMS(TCPDATA).
4. A data set with the name of hlq.TCPIP.DATA, where hlq is the value set by a DATASETPREFIX statement found in a prior TCPIP.DATA, the value set by the EZAPPRFX installation job, or if this job was not run, the system default value of TCPIP.

3.16 DEVICE and LINK statements

- ❑ Must update the hlq.PROFILE.TCPIP data set to include:
 - A DEVICE and LINK statement for each DLC connection to be established between the main TCPIP address space and the SNALINK LU6.2 address space

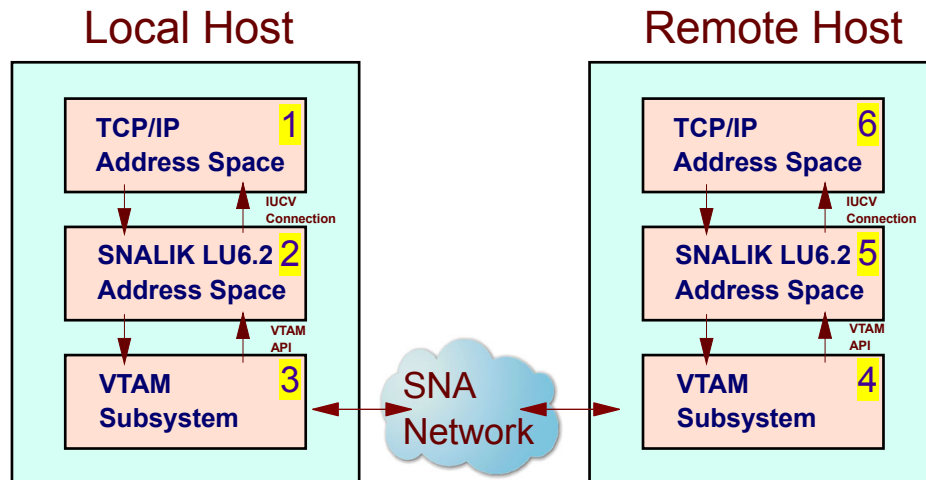


Figure 3-19 DEVICE and LINK statements for LU6.2 in configuration data set

DEVICE and LINK statements

z/OS Communications Server allows a single TCP/IP address space to drive multiple instances of any supported device. To configure your devices, add the appropriate DEVICE and LINK statements to the configuration data set. The LINK statements show how to define a network interface link associated with the device and are included with the DEVICE statement for that device type.

Note: You must update the hlq.PROFILE.TCPIP data set to include a DEVICE and LINK statement for each DLC connection to be established between the main TCP/IP address space and the SNALINK LU6.2 address space.

Update the SNALINK LU6.2 cataloged procedure by copying the sample in SEZAINST(LU62PROC) to your system or to a recognized PROCLIB and modifying it to suit your local conditions. No system parameters are required for the SNALINK LU6.2 address space.

The SNALINK LU6.2 interface uses the LU type 6.2 protocol to establish a point-to-point connection across a SNA network. SNALINK LU6.2 is capable of establishing a connection with any system that runs TCP/IP and uses the LU type 6.2 protocol. The SNALINK LU6.2

interface is similar to the SNALINK LU0 and X.25 NPSI interfaces with the connection involving several subsystems. The components of the SNALINK LU6.2 network are shown in Figure 3-19, as follows:

1. Data is generated and encapsulated on the TCP/IP address space and is passed to the SNALINK LU6.2 address space through a DLC connection.
2. The SNALINK LU6.2 address space handles all establishment, aging, and termination of SNA network connections in a manner transparent to the TCP/IP address space. The data is then sent to the local system SNA subsystem. In the case of MVS hosts, this subsystem is VTAM.
3. VTAM APPC routines are used to pass the data to the SNA network.
4. VTAM routines on the destination system receive the data and pass it through to the SNALINK LU6.2 address space.
5. The SNALINK LU6.2 address space sends the data to the TCP/IP address space using a DLC connection.
6. The data is unencapsulated and processed by the TCP/IP address space.

Configuring SNALINK LU6.2

The steps to configure SNALINK LU6.2 are:

1. Specify DEVICE and LINK statements in hlq.PROFILE.TCPIP.
2. Update the SNALINK LU6.2 cataloged procedure.
3. Define the SNALINK LU6.2 application to VTAM.
4. Update the SNALINK LU6.2 configuration data set.

Components of a SNALINK LU6.2 connection

SNALINK LU6.2 opens two SNA LU type 6.2 sessions with each destination node, one for sending and one for receiving. If a destination node supports parallel SNA LU type 6.2 sessions (PARSESS=YES), the two sessions use the same remote logical unit; otherwise, two remote logical units are used. In either case, SNALINK LU6.2 uses a single local logical unit that must support parallel sessions.

The SNALINK LU6.2 interface uses the LU type 6.2 protocol to establish a point-to-point connection across a SNA network. SNALINK LU6.2 is capable of establishing a connection with any system that runs TCP/IP and uses the LU type 6.2 protocol.

The SNALINK LU6.2 interface is similar to the SNALINK LU0 and X.25 NPSI interfaces with the connection involving several subsystems.

SNALINK LU6.2 address space

The SNALINK LU6.2 address space must be defined to VTAM as a SNA LU type 6.2 application program.

3.17 The resolver

- ❑ The resolver address space must be started before any application or TCP/IP stack resolver calls can occur
 - The resolver is a set of routines that act as a client on behalf of an application to read a local host file or to access one or more domain name system (DNS) for name-to-IP-address or IP-address-to-name resolution
 - The resolver acts on behalf of programs as a client that accesses name servers for name-to-address or address-to-name resolution
 - The resolver configuration file is where the resolver looks for name-address resolution and how it handles requests
 - The resolver can exist on any system that supports sockets programs, using name resolution

Figure 3-20 The resolver

The resolver

The resolver acts on behalf of programs as a client that accesses name servers for name-to-address or address-to-name resolution. The resolver can also be used to provide protocol and services information. To resolve the query for the requesting program, the resolver can use information obtained from any of the following sources:

- ▶ Available name servers
- ▶ Its system-wide resolver cache of name server data
- ▶ Local definitions, such as `/etc/resolv.conf`, `/etc/hosts`, `/etc/ipnodes`, `HOSTS.SITEINFO`, `HOSTS.ADDRINFO`, or `ETC.IPNODES`

Whether the resolver uses name servers, and how, is controlled by `TCPIP.DATA` statements (resolver directives). When system-wide caching is enabled, the resolver can also use DNS response information that has been cached locally.

Starting the resolver address space

There are two ways to start the resolver address space:

- ▶ The resolver starts when z/OS UNIX is initialized, unless there are specific instructions that it should not be started. This method of starting the resolver is best because it ensures that the resolver is available before any applications can make a resolution request.

- ▶ Use the BPXPRMxx statement, RESOLVER_PROC, to specify the procedure name, if any, to be used to start the resolver address space.

Using the resolver address space

Calls from the resolver to the TCP/IP stack can occur. When the resolver address space starts, it reads an optional resolver setup data set pointed to by the SETUP DD card in the resolver JCL procedure. A sample of the JCL for starting the resolver configuration file is shown in Example 3-1.

Example 3-1 Sample JCL for starting resolver address space

```
//RESOLVER PROC PARMS='CTRACE(CTIRES00)'  
//EZBREINI EXEC PGM=EZBREINI,REGION=OM,TIME=1440,PARM=&PARMS  
//SETUP DD PATH='/etc/setup.resolver',PATHOPTS=(ORDONLY)
```

When an application needs to access services identified within the resolver configuration file, it is accomplished using the resolver started task (address space). The resolver address space is normally started when z/OS is started. The significance of this resolver address space is that the address space itself can be configured with resolver configuration file statements. These statements, shown in Example 3-2, can be used to provide resolver configuration file settings for both z/OS UNIX applications and MVS applications.

Example 3-2 Sample resolver configuration file

```
SEARCH ITSO.IBM.COM IBM.COM  
NSINTERADDR 10.12.6.7  
NSPORTADDR 53  
RESOLVEVIA UDP  
RESOLVERTIMEOUT 10  
RESOLVERUDPRETRIES  
LOOKUP LOCAL DNS
```

Resolver statements

For those familiar with TCP/IP implementation on UNIX platforms, you might have experience with a file called `/etc/resolv.conf`. The `resolv.conf` file is referred to as a resolver configuration file or the `TCPIP.DATA` file.

Why two names? It all depends on whether the TCP or UDP application is a z/OS UNIX application or a native MVS application.

3.18 Resolver basic concepts

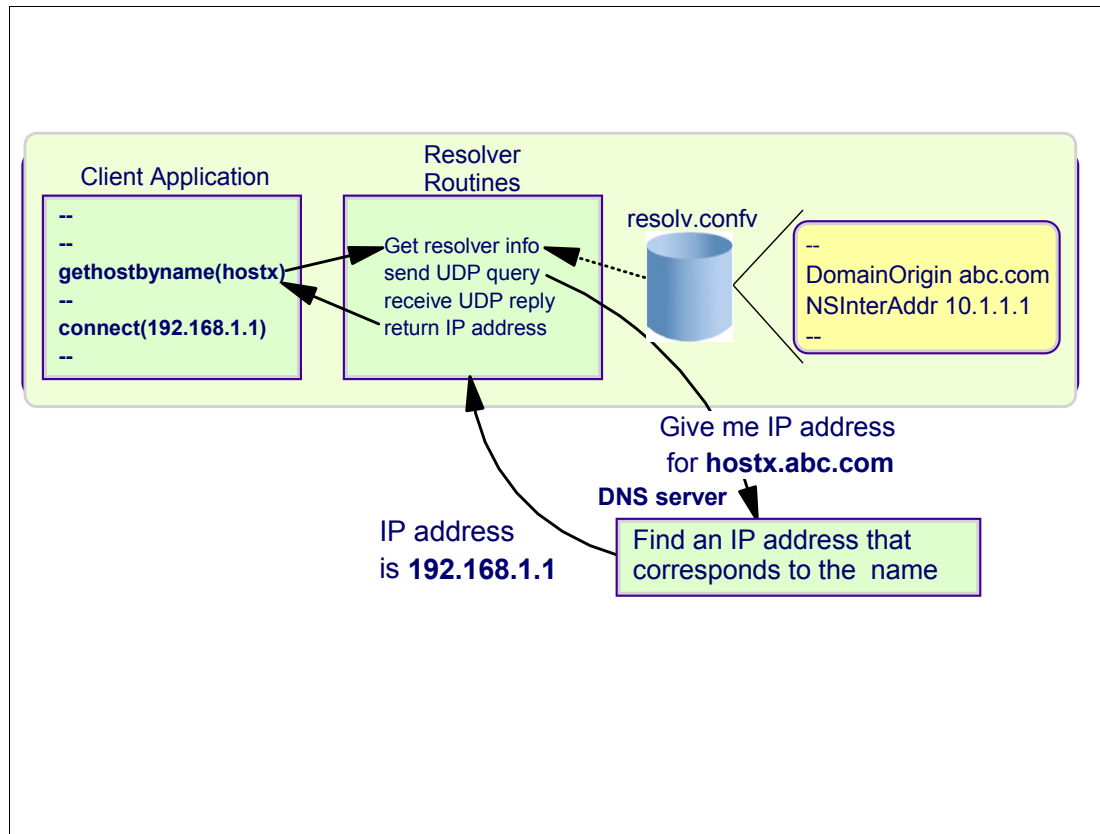


Figure 3-21 How the resolver works

Basic concepts

In most systems, for the application to reach a remote partner, it uses two commands to ask the resolver what the IP address is for a host name, or vice versa. The commands are **gethostbyname(xxx)**. The resolver code and other services that execute as part of the socket program address space to service calls such as `gethostbyname()`, `getservbyname()` and `getprotobyname()` allocate one or more resolver configuration files to service these calls. All socket programs, including standard servers and clients and homegrown socket programs, need access to resolver configuration files.

Resolver flows

Resolver flows show the information request and response flows as the resolver gets a request, and based on its own configuration file either looks at a local hosts file or sends a request to a DNS server. Once the relationship between the host name and IP address is made, the Resolver returns the response to the application. In a z/OS system, this task is more complex, because the applications can be built using one of three main groups of sockets API environments:

- ▶ Native TCP/IP sockets
- ▶ UNIX System Services callable sockets (BPX1xxxx calls)
- ▶ Language Environment C/C++ sockets

To initiate a request to the resolver in z/OS, an application executes a set of commands based on the Sockets API Library the application used to generate the socket to connect to the TCP/IP stack.

3.19 Configuring the TCP/IP address space

- ❑ Configure BPXPRMxx parmlib member
 - AF_UNIX and AF_INET
 - Resolver address space
 - Additional parameters that are crucial to the proper operation of TCP/IP

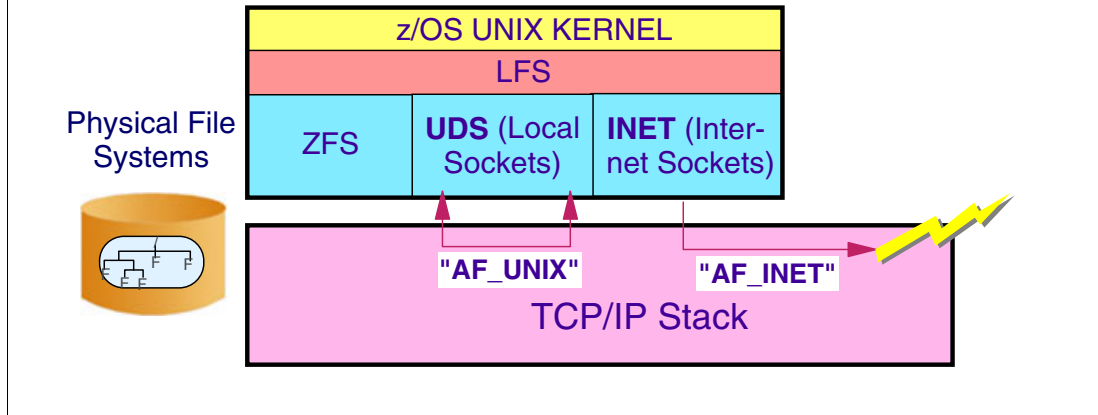


Figure 3-22 TCP/IP configuration data sets

Configuring the TCP/IP address space

The TCP/IP services stack in z/OS Communications Server must be defined as a z/OS Communications Server UNIX System Services physical file system (PFS) before it can be started. This involves updating the BPXPRMxx parmlib member. The following sample definition in BPXPRMxx defines TCP/IP as a z/OS Communications Server UNIX System Services PFS, where the network layer is IP Version 4 (IPv4) and communication at the sockets layer is through the AF_INET family.

Example 3-3 AF_INET definition

```
FILESYSTYPE TYPE(INET) ENTRYPOINT(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
          DOMAINNUMBER(2)
          MAXSOCKETS(60000)
          TYPE(INET)
```

The sample definition in Example 3-3 shows how to define a single TCP/IP stack as IPv4 only. To define a single TCP/IP stack as both IPv4 and IPv6, add an additional NETWORK statement in the BPXPRMxx member. Example 3-4 on page 88 shows a definition in BPXPRMxx that defines TCP/IP as a z/OS Communications Server UNIX System Services PFS, where the network layer is IP Version 6 (IPv6) and communication at the sockets layer is through the AF_INET6 family.

Example 3-4 NETWORK statement defining AF_INET6

```
NETWORK DOMAINNAME(AF_INET6)
        DOMAINNUMBER(19)
        MAXSOCKETS(60000)
        TYPE(INET)
```

BPXPRMxx parmlib member

The BPXPRMxx member contains additional parameters that are crucial to the proper operation of TCP/IP. Carefully examine and specify these parameters:

MAXPROCSYS	Maximum number of z/OS UNIX processes that the system allows.
MAXPROCUSER	Maximum number of processes that a single z/OS UNIX user ID can have concurrently active, regardless of how the processes were created.
MAXUIDS	Maximum number of z/OS UNIX user IDs that can operate concurrently.
MAXFILEPROC	<p>Maximum number of descriptors for files, sockets, directories, and any other file system objects that a single z/OS UNIX process can have concurrently allocated. This includes access to both z/OS UNIX files and socket descriptors. In z/OS Communications Server, most TCP/IP applications access TCP/IP sockets, either directly or indirectly, using the TCP/IP socket APIs. You should set the MAXFILEPROC value high enough to accommodate the largest number of sockets that a single TCP/IP application (or z/OS UNIX process) can allocate.</p> <p>Be aware that the TN3270E Telnet server is exempt from the limit specified in this parameter. The TN3270E Telnet server can obtain the maximum number of socket connections for a single z/OS Communications Server UNIX System Services process.</p>
MAXPTYs	Maximum number of pseudo-terminals for the system.
MAXTHREADTASKS	Maximum number of MVS tasks that a single process can have concurrently active.
MAXTHREADS	Maximum number of threads that a single process can have concurrently active.
MAXQUEUEDSIGS	<p>The sum of MAXQUEUEDSIGS and MAXFILEPROC multiplied by 2 is the system-wide maximum for the total number of asynchronous z/OS UNIX socket calls that can be outstanding. When specifying this number, consider the following:</p> <ul style="list-style-type: none">▶ For every TCP/IP connection that the TN3270E Telnet server has, there is an asynchronous z/OS UNIX socket call outstanding. This is true for both TN3270 and TN3270E clients.▶ Any TCP/IP application, IBM or vendor supplied, that uses either the z/OS UNIX asyncio callable service or the TCP/IP-provided Sockets Extended asynchronous API could have one or more outstanding asynchronous socket calls.

3.20 z/OS UNIX physical file systems

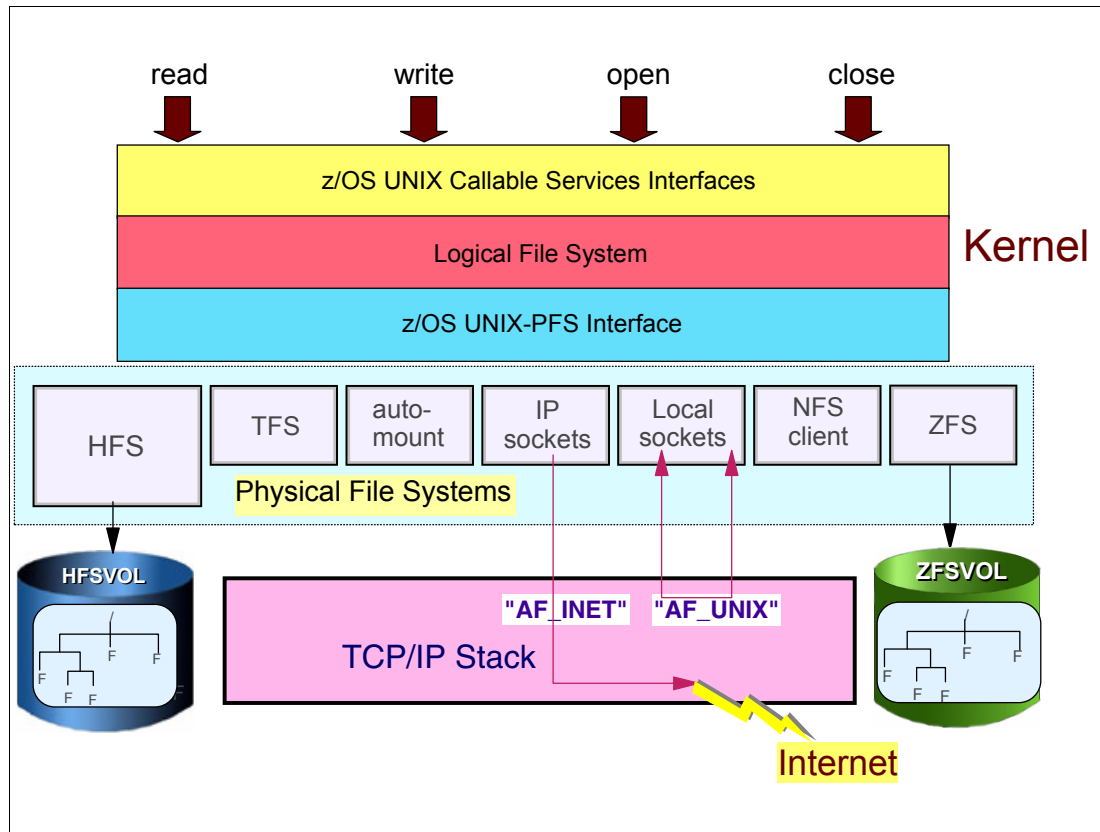


Figure 3-23 TCP/IP and z/OS UNIX

z/OS UNIX file systems

The layer beneath the LFS is the physical file system (PFS). The PFS layer is where the distinction between files, sockets, and other resources is made. Based on the resource type, the LFS passes the incoming function requests to the appropriate PFS, which handles requests related to resources in the z/OS UNIX file system. For more information about these physical file systems, see *z/OS UNIX System Services Planning*, GA22-7800.

From a TCP/IP perspective, the AF_INET and the AF_INET6 PFS are of main interest. TCP/IP is enabled for IPv6 by defining an AF_INET6 PFS. Defining the file systems is the responsibility of the installation's z/OS UNIX programmer. The definitions are found in the BPXPRMxx member of SYS1.PARMLIB. Currently, only two domains are supported:

- ▶ AF_UNIX, domain number 1, and entry point (BPXTUINT)
- ▶ AF_INET, domain number 2, and entry point (BPXTIINT) or entry point BPXTCINT, if Converged Sockets is used

BPXPRMxx parmlib statements

The NETWORK statement defines which domain the specified file system supports and some socket and port limits in that domain by specifying:

- ▶ The address family type
- ▶ Its associated domain number
- ▶ The maximum number of sockets the address family will support
- ▶ The ports to be reserved for use with port zero, INADDR_ANY binds

As shown in Example 3-5, there must be a previous FILESYSTYPE statement that has a TYPE operand that matches the TYPE operand on the NETWORK statement, and:

- ▶ For TYPE(UDS):
 - ENTRYPOINT is BPXTUINT
 - DOMAINNAME is 'AF_UNIX'
 - DOMAINNUMBER is 1
 - MAXSOCKETS is 64
 - The TYPE of the file system is 'UDS'
 - No port reservations are required for AF_UNIX
- ▶ For TYPE(INET):
 - ENTRYPOINT is BPXTIINT
 - DOMAINNAME is 'AF_INET'
 - DOMAINNUMBER is 2
 - MAXSOCKETS is 64
 - The TYPE of the file system is 'INET'
 - No port reservations are required

Example 3-5 AF_UNIX and AF_INET definitions in the BPXPRMxx parmlib member

```
FILESYSTYPE TYPE(UDS) ENTRYPOINT(BPXTUINT)
NETWORK DOMAINNAME(AF_UNIX)
        DOMAINNUMBER(1)
        MAXSOCKETS(10000)
        TYPE(UDS)
```

```
FILESYSTYPE TYPE(CINET)
        ENTRYPOINT(BPXCINT)
NETWORK DOMAINNAME(AF_INET)
        DOMAINNUMBER(2)
        MAXSOCKETS(65535)
        TYPE(CINET)
        INADDRANYPORT(10000)
        INADDRANYCOUNT(2000)
```

3.21 TCP/IP family of protocols

- TCP/IP services Telnet support
- The socket API communication functions
- CICS TCP/IP client and server processing
- TCP/IP TCP, UDP, and IP protocols
- Server TCP/IP calls
- Client TCP/IP calls
- Other socket calls used for servers
- FTP server

Figure 3-24 TCP/IP family of protocols

TCP/IP services Telnet support

TCP/IP services supports traditional 3270 mainframe interactive (MFI) applications with an emulator function called Telnet (TN3270). For these applications, all program logic is housed in the mainframe, and the remote host uses only that amount of logic necessary to provide basic communication services. Thus, if your requirement is simply to provide access from a remote TCP/IP host to existing CICS MFI applications, you should probably consider Telnet rather than CICS TCP/IP as the communications vehicle. Telnet 3270 emulation functions allow your TCP/IP host to communicate with traditional applications without modification.

CICS TCP/IP client and server processing

With CICS TCP/IP, remote client systems can initiate communications with CICS and cause a CICS transaction to start. It is anticipated that this is the most common mode of operation. Alternatively, the remote system can act as a server with CICS initiating the conversation. TCP/IP also supports client and server processing, where processes are either:

- ▶ Servers that provide a particular service and respond to requests for that service
- ▶ Clients that initiate the requests to the servers

TCP/IP TCP, UDP, and IP protocols

TCP/IP is a large family of protocols that is named after its two most important members.

The protocols implemented by TCP/IP Services and used by CICS TCP/IP are as follows:

- TCP** Transmission Control Protocol (TCP) is a transport-layer protocol. It provides a reliable virtual-circuit connection between applications; that is, a connection is established before data transmission begins. Data is sent without errors or duplication and is received in the same order as it is sent. No boundaries are imposed on the data; TCP treats the data as a stream of bytes.
- UDP** User Datagram Protocol (UDP) UDP is also a transport-layer protocol and is an alternative to TCP. It provides an unreliable datagram connection between applications. Data is transmitted link by link; there is no end-to-end connection. The service provides no guarantees. Data can be lost or duplicated, and datagrams can arrive out of order.
- IP** Internet Protocol (IP) is a network-layer protocol. It provides a datagram service between applications, supporting both TCP and UDP.

The socket API communication functions

The socket API is a collection of socket calls that enables you to perform the following primary communication functions between application programs, as follows:

- ▶ Set up and establish connections to other users on the network
- ▶ Send and receive data to and from other users
- ▶ Close down connections

Server TCP/IP calls

To understand Socket programming, the client program and the server program must be considered separately. This is the logical presentation sequence because the server is usually already in running before the client is started. The types of TCP/IP calls are:

- ▶ SOCKET server TCP/IP call - BIND server TCP/IP call - LISTEN server TCP/IP call - ACCEPT server TCP/IP call - GIVESOCKET and TAKESOCKET server TCP/IP call - READ and WRITE server TCP/IP call

Client TCP/IP calls

Stream-oriented socket programs generally follow a prescribed sequence. A concurrent server typically starts before the client does, and waits for the client to request connection. It then continues to wait for additional client requests after the client connection is closed. An iterative server handles both the connection request and the transaction involved in the call itself. Iterative servers are fairly simple and are suitable for transactions that do not last long. However, if the transaction takes more time, queues can build up quickly. So, for lengthy transactions, a different sort of server is needed, the concurrent server, CICS listener.

Other socket calls used for servers

The TCP/IP call sequence for a client is simpler than the one for a concurrent server. A client only has to support one connection and one conversation. A concurrent server obtains a socket upon which it can listen for connection requests, and then creates a new socket for each new connection. In the same manner as the server, the first call issued by the client is the SOCKET call. This call causes allocation of the socket on which the client communicates. After the SOCKET call has allocated a socket to the client, the client can then request connection on that socket with the server through use of the CONNECT call.

FTP server

FTP is the protocol of choice to transfer data over the Internet. A wide range of command lines and GUI clients are available. Most operating systems have an FTP client installed by default.

3.22 Telnet protocol

- ❑ Telnet is a terminal emulation protocol
 - Telnet client connects to Telnet server
- ❑ Types of Telnet servers
 - TN3270E Telnet server
 - z/OS UNIX Telnet server
- ❑ z/OS UNIX System Services
 - otelnetd daemon

Figure 3-25 Telnet protocol and z/OS UNIX

Using Telnet

Telnet is a terminal emulation protocol. With Telnet, end users can log on to remote host applications as though they were directly attached to that host. Telnet protocol requires that the end user have a Telnet client that emulates a type of terminal that the host application can understand. The client connects to a Telnet server, which communicates with the host application. The Telnet server acts as an interface between the client and host application. A PC can support several clients simultaneously, each with its own connection to any Telnet server.

The original basic Telnet protocol was defined in RFC 854. This RFC effectively defined all that was needed to support the 3270 data stream, since the 3270 data stream is just part of the Telnet data payload. In other words, the 3270 portion was implemented outside of (above) the Telnet protocol. Specific options could be negotiated (beyond basic Telnet) using the Telnet option standard of RFC 855. Option negotiation in turn allowed for device type negotiations (later formally defined in RFC 1091) to be completed as part of the Telnet session setup.

TN3270E

Initially, there was no formal standard for TN3270 (the E came along later), but it was clarified in an early RFC titled “TN3270 Current Practices” (RFC 1576). TN3270 itself began to take shape more formally with RFCs 1646 and 1647. RFC 1647 was a significant RFC because it was the first formalization of the TN3270 Enhanced protocol, known as TN3270E. TN3270E improved upon the TN3270 protocol to include control of LU name selection, as well as full

support of Attention Identifier such as SYSREQ (to talk directly to VTAM) and the attention key.

Types of Telnet servers

The types of Telnet servers are:

- ▶ TN3270E Telnet server provides access to z/OS VTAM SNA applications on the MVS host using Telnet TN3270E, TN3270, or linemode protocol.

A TN3270E client uses the TN3270E protocol to access the resources on a TN3270E server. However, the TN3270E client cannot complete the connection all the way to the target application because the TN3270E client communicates according to the TN3270E protocol, while the target application expects communication to be SNA protocol.

- ▶ z/OS UNIX Telnet server provides access to z/OS UNIX shell applications on the MVS host using Telnet linemode protocol.

Note: You can use the same port for both Telnet servers.

otelnetd server

The otelnetd server is a functional equivalent to the telnetd daemons found on other UNIX-style servers. Since it is a z/OS UNIX daemon, it is run within the z/OS UNIX environment and provides the user with a shell only (no access to the MVS file system is possible). All of the functionality is there that would be expected when connecting to a UNIX telnetd server: the vi editor, environment variables, profile processing, and more.

3.23 Using sockets

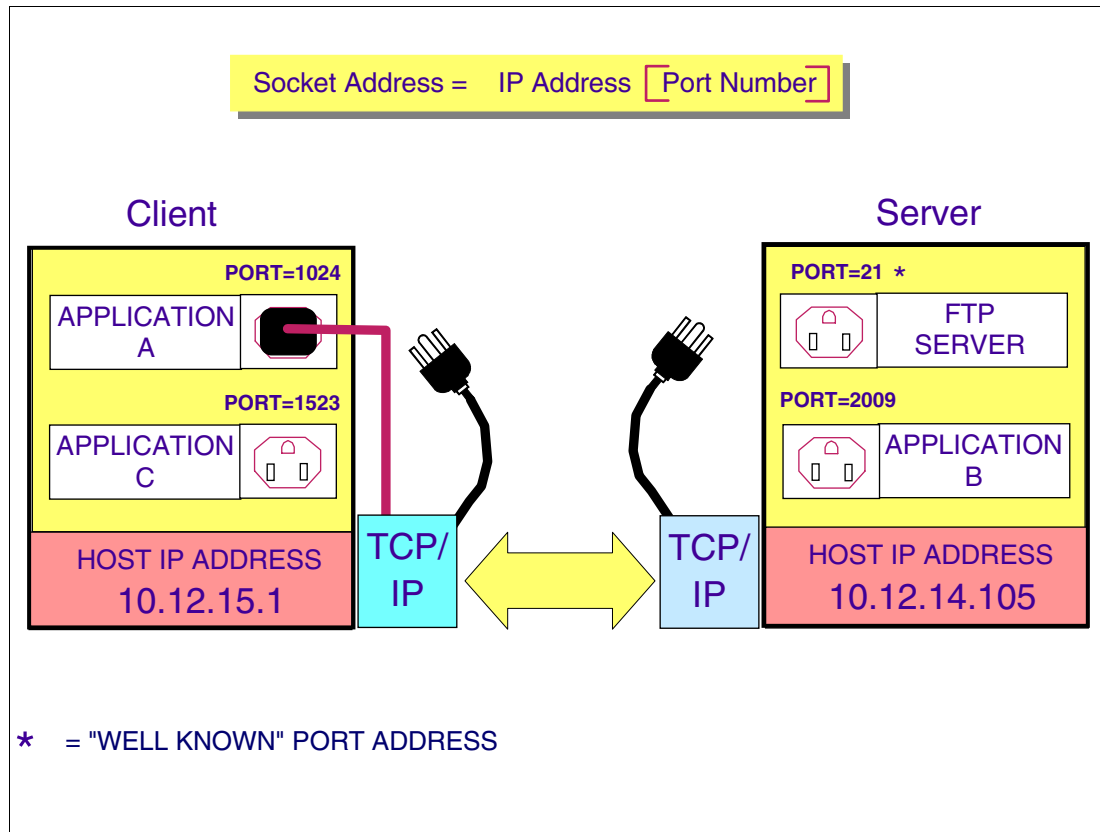


Figure 3-26 UNIX sockets

z/OS UNIX sockets

This should be a refresher topic. It is intended to stress that sockets (at least in the classic UNIX interpretation) are not to be treated any differently than ordinary data files insofar as program logic is concerned. Also, it should be reiterated that sockets are used for local (intrahost) as well as remote (interhost or network) communication between UNIX processes.

Simply consider the concept of an API as being the set of C library service calls used to work with a particular resource. Although sockets and HFS files are both streams, the complexity of socket handling requires more powerful syntax and options for handling socket data. This is relevant in a later discussion where we see that z/OS has perhaps eight different socket APIs!

A powerful design principle was used in UNIX data processing: any type of data manipulated by a process can be represented by a standard file/stream structure. Also, a single set of program calls is used to open streams, read and update data, and close streams, insulating the program from dealing with the physical origin of stream data.

UNIX technologies

Of course, as with many things in UNIX, this is an idealized picture that fits various commercial UNIX "flavors" to a greater or lesser extent. The currently available UNIX flavors are derived from two main UNIX technologies:

- **Berkeley Software Division (BSD)** - This is UNIX produced by the University of California. It is the strongest influence in UNIX systems produced by DEC Ultrix, SunOS,

HP-UX from Hewlett Packard, and, in the early stages, IBM OpenEdition, which is now named z/OS UNIX.

- ▶ **AT&T** - This is UNIX produced by AT&T, and after that, USL UNIX system laboratories. The current UNIX version is UNIX System VR4. System VR4 is a major influence in IBM AIX and Apple AUX.

In addition to many other differences, the two flavors of UNIX differ in the way that they see sockets, as follows:

- ▶ Berkeley was the original developer of socket protocols, which it connected with TCP/IP network protocols to provide remote process-to-process communication over the network. As a new add-in to the UNIX kernel, and because functionality was considerably more complex in BSD sockets compared to ordinary file handling, BSD sockets use a set of new program calls to manipulate data over sockets. For example, a program would open a file, read or write data to it, and close it. To initialize socket processing, on the other hand, the program issues a socket call, uses **send** and **recv** commands to transfer data, and then will close the socket.
- ▶ The AT&T System V approach is more classical. Both files and sockets are opened, data can be manipulated in both cases using read and write commands, and both are closed to terminate stream usage. Also, AT&T pioneered the idea of the *UNIX-domain* or *local socket*, used to communicate between two UNIX processes in the same system – what MVS would call cross-memory communication. A UNIX domain socket is considerably simpler than a network socket, and can be easily handled with the classic socket approach.

Other UNIX platforms

In modern UNIX systems, there has been considerable cross-pollination between the two UNIX flavors, such that, for example, AIX now supports many BSD features, including BSD sockets. Also, BSD sockets provide both local domain and Internet-type socket support. However, when porting the C source code for a new application to a UNIX system, allowances have to be made for the fact that, for example, program calls made to sockets could be written using either the BSD or the System V approach. To prevent code rewrite, system compiler and run-time support must be able to handle both flavors of calls. This principle must apply to z/OS UNIX as well. The original (4.3) approach in z/OS UNIX was oriented to BSD socket support rather than to System V.

In the classical (System V) approach, all open streams, whether files, terminals, sockets, and so on, are pointed to by descriptors (file handles) in the same file descriptor table. The contents of a socket descriptor will be different from that of a file descriptor in the FDT, but the index of a socket descriptor is an integer number identical to the type of index pointing to a file descriptor for a file. So, for example, a program might have an open data file with file descriptor FD=4, and an initialized socket at socket descriptor SD=5, in consecutive slots of the table. Obviously, the index numbers cannot be allowed to overlap.

Well-known ports

Well-known ports belong to standard servers; for example, Telnet uses port 23. Well-known port numbers range between 1 and 1023 (prior to 1992, the range between 256 and 1023 was used for UNIX-specific servers). Well-known port numbers are typically odd, because early systems using the port concept required an odd/even pair of ports for duplex operations. Most servers require only a single port. The well-known ports are controlled and assigned by the Internet central authority (IANA) and on most systems can only be used by system processes or by programs executed by privileged users. The reason for well-known ports is to allow clients to be able to find servers without configuration information.

3.24 Using TCP/IP with IMS and CICS

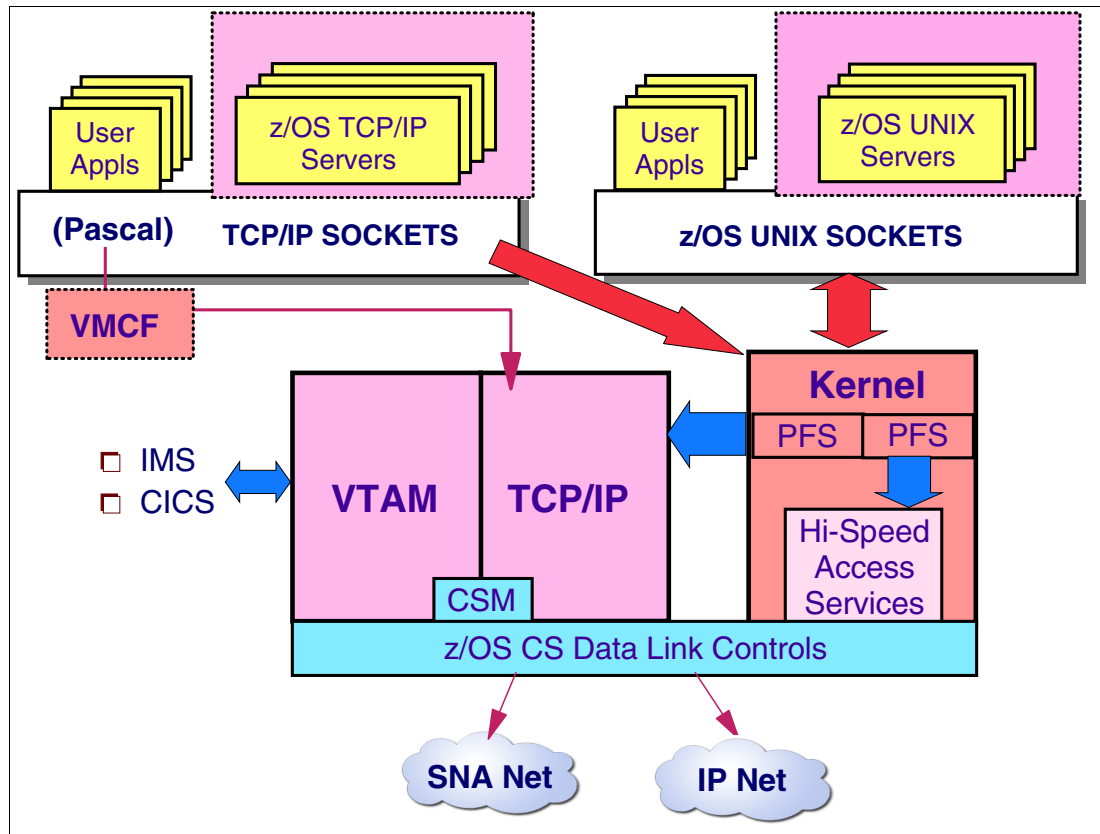


Figure 3-27 IMS and CICS communication with TCP/IP

z/OS Communications Server

Since OS/390 V2R5 was shipped, the z/OS Communications Server has been included as a base element. The TCP/IP z/OS component, called z/OS Communications Server, is a reconstructed stack that is able to support both UNIX and non-UNIX socket APIs. It is often called the converged IP stack.

All the TCP/IP socket APIs that supported HPNS are now transparently redirected by run-time support to call the UNIX kernel LFS. The REXX socket API has also been directed to call the kernel. HPNS support is no longer required.

Pascal API

The Pascal API, however, still requires the VMCF/IUCV address space to be started, which links the API to the new stack. If VMCF and TNF do not respond to commands, this is probably because one or both of the on-restartable versions of VMCF or TNF are still active. To get them to respond to commands, stop all VMCF/TNF users, FORCE ARM VMCF and TNF, then use the EZAZSSI procedure to restart.

The Pascal application programming interface enables you to develop TCP/IP applications in Pascal language. Supported environments are normal MVS address spaces. The Pascal programming interface is based on Pascal procedures and functions that implement conceptually the same functions as the C socket interface. The Pascal routines, however, have different names than the C socket calls. Unlike the other APIs, the Pascal API does not

interface directly with the LFS. It uses an internal interface to communicate with the TCP/IP protocol stack.

Communications Server stacks

The SNA and IP networking stacks have been integrated to a considerable extent. Both stacks use common Data Link Control (DLC) routines to access network hardware, and both types of protocols can flow over the same hardware link. Also, common service routines such as Communications Storage Manager (CSM) exploit use of buffers in common storage for both IP and SNA performance.

3.25 Using CICS sockets API

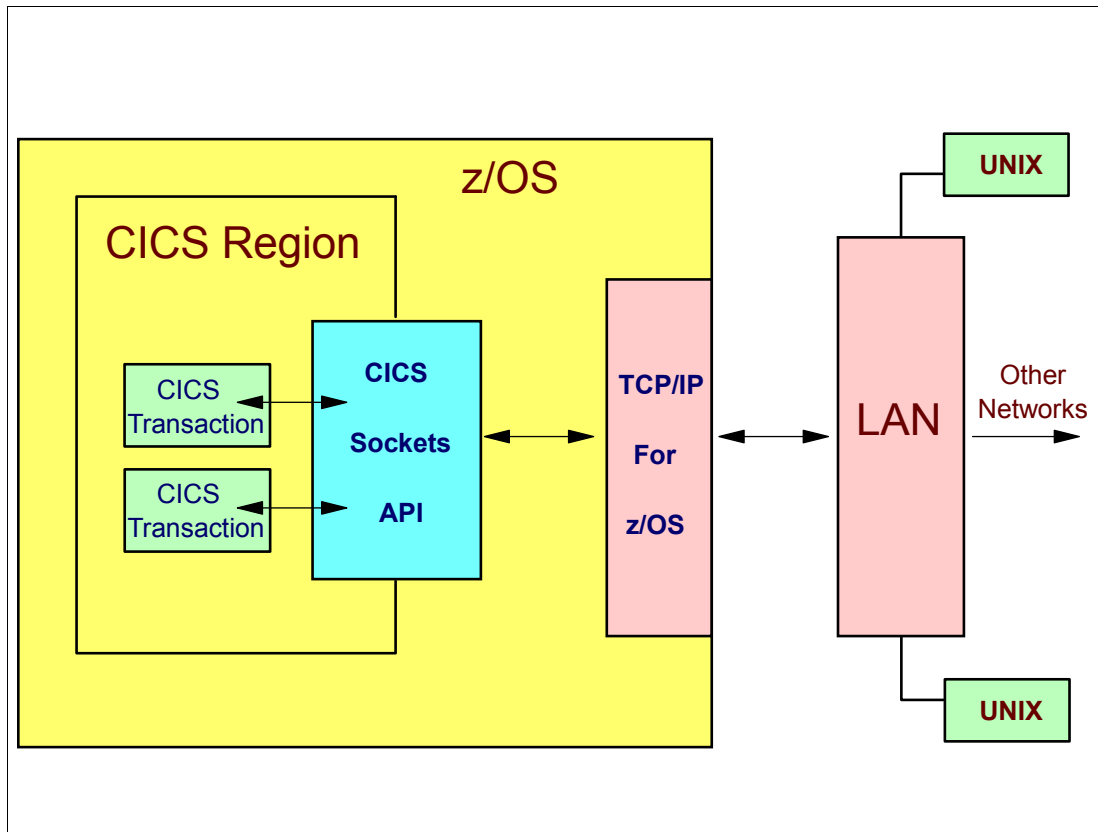


Figure 3-28 Using CICS sockets

CICS Transaction Server

CICS Transaction Server (CICS TS) is an online transaction processing system. Application programs using CICS can handle large numbers of data transactions from large networks of computers and terminals. Communication throughout these networks has often been based on the Systems Network Architecture (SNA) family of protocols. CICS TCP/IP offers CICS users an alternative to SNA, the TCP/IP family of protocols for those users whose native communications protocol is TCP/IP. CICS TCP/IP allows remote users to access CICS client/server applications over TCP/IP internets. Figure 3-28 shows how these two products give remote users peer-to-peer communication with CICS applications.

Applications that use the IP CICS sockets API can be written in the following languages:

- ▶ C language
- ▶ COBOL language call format
- ▶ Assembler language call format
- ▶ PL/I language call format

CICS transactions

In a CICS system based on SNA terminals, the CICS terminal management modules perform the functions of a concurrent server. Because the TCP/IP interface does not use CICS terminal management, CICS TCP/IP provides these functions in the form of a CICS application transaction, the listener. The CICS transaction ID of the IBM distributed listener is

CSKL. This transaction is defined at installation to execute the EZACIC02 program and is to be further referenced as the listener. This transaction ID can be configured to a transaction ID suitable for the user's requirements through the use of the EZACICD macro or the EZAC CICS transaction and the accompanying RDO transaction definition.

CICS sockets API

The client might be running TCP/IP under one of the various UNIX operating systems such as AIX. TCP/IP CICS socket support provides a CICS resource manager that invokes z/OS UNIX socket services. It also provides a CICS Listener transaction that accepts passive connections from a listening socket. Each listening socket has a configured transaction that is launched to process a single established connection.

IBM CICS Transaction Gateway has been proven over many years to provide high-performing, security-rich, and scalable access to CICS Transaction Server, requiring minimal changes to CICS systems and usually no changes to existing CICS applications. CICS Transaction Gateway is a Java™ 2 Platform, Enterprise Edition (J2EE) connector to CICS applications. It supports the standard J2EE Connector Architecture 1.5 specification (JCA 1.5), which enables better applications to be developed faster and more easily.

With IBM CICS Transaction Gateway for z/OS, you can use your CICS communications area (COMMAREA)-based applications in comprehensive and sophisticated Java and Web services solutions hosted on powerful application servers such as IBM WebSphere® Application Server.

IBM CICS Transaction Gateway for z/OS, Version 7.2, provides the highest quality of service. In this z/OS environment, CICS Transaction Gateway can support thousands of transactions per second by using multiple gateway regions, and by reusing memory-based External CICS Interface (EXCI) pipes.

Another advantage of deploying CICS Transaction Gateway on the z/OS platform is the provision of global transactional integrity. With global two-phase commit, a composite transaction can be physically distributed across heterogeneous servers and operating environments – without compromising data integrity.

3.26 Using IMS with SNA or TCP/IP

- ❑ IMS as an online processing system
 - Uses SNA and TCP/IP as protocols
- ❑ Mainframe interactive processing with IMS
 - IMS MFI applications use:
 - IMS TCP/IP using Client/server processing
 - Telnet 3270-emulation
- ❑ Protocols used by IMS
 - Transmission Control Protocol (TCP)
 - User Datagram Protocol (UDP)
 - Internet Protocol (IP)

Figure 3-29 IMS and communication protocols

IMS transaction protocols

IMS is an online transaction processing system. This means that application programs using IMS can handle large numbers of data transactions from large networks of computers and terminals. Communication throughout these networks has often been based on the Systems Network Architecture (SNA) family of protocols. IMS TCP/IP offers IMS users an alternative to SNA, the TCP/IP family of protocols for those users whose native communications protocol is TCP/IP.

Mainframe interactive processing

TCP/IP services supports traditional 3270 mainframe interactive (MFI) applications with an emulator function called Telnet (TN3270). For these applications, all program logic runs in the mainframe, and the remote host uses only that amount of logic necessary to provide basic communications services. Thus, if your requirement is simply to provide access from a remote TCP/IP host to existing IMS MFI applications, you should consider Telnet rather than IMS TCP/IP as the communications vehicle. Telnet 3270-emulation functions allow your TCP/IP host to communicate with traditional applications without modification.

Client/server processing

TCP/IP also supports client/server processing, where processes are either:

- ▶ Servers that provide a particular service and respond to requests for that service
- ▶ Clients that initiate the requests to the servers

With IMS TCP/IP, remote client systems can initiate communications with IMS and cause an IMS transaction to start. It is anticipated that this will be the most common mode of operation. (Alternatively, the remote system can act as a server with IMS initiating the conversation.)

Protocols used by IMS

The protocols implemented by TCP/IP services and used by IMS TCP/IP, are:

- ▶ **Transmission Control Protocol (TCP)**

In terms of the OSI model, TCP is a transport-layer protocol. It provides a reliable virtual-circuit connection between applications; that is, a connection is established before data transmission begins. Data is sent without errors or duplication and is received in the same order as it is sent. No boundaries are imposed on the data; TCP treats the data as a stream of bytes.

- ▶ **User Datagram Protocol (UDP)**

UDP is also a transport-layer protocol and is an alternative to TCP. It provides an unreliable datagram connection between applications (that is, data is transmitted link by link; there is no end-to-end connection). The service provides no guarantees; data can be lost or duplicated, and datagrams can arrive out of order.

- ▶ **Internet Protocol (IP)**

In terms of the OSI model, IP is a network-layer protocol. It provides a datagram service between applications, supporting both TCP and UDP.



Network connectivity

There are several ways to make network connections. The mainframe originally relied upon the channel subsystem to offload I/O processing to channel programs. DASD is still accessed using ESCON channels, but for networking connectivity, OSA-Express cards offer better performance and availability.

The OSA-Express and OSA-Express2 cards provide redundancy capability, as well as throughput improvements when running in QDIO mode. QDIO mode allows direct access to central memory. QDIO mode can be emulated within a CPC by allowing memory-to-memory data transfer among LPARs running z/VM, Linux, or z/OS.

Connections for the mainframe

The design intention of the mainframe, and most of its evolution, is for the mainframe to be a highly available transaction processing server. Obviously, central processing capabilities are evolving to handle more and more transactions. However, to be an effective transaction processing server, there must be a proportional capability of moving data in and out of the central processor complex rapidly (CPC, the physical collection of hardware that consists of main storage, one or more central processors, timers, and channels). The result is that the I/O options, capabilities, and configuration choices of an IBM mainframe are varied, complex, and performance-oriented.

Mainframe computers are probably unique in that they require a Hardware Management Console, or HMC. The HMC is a separate interface to the central processor complex that is used for hardware configuration operations. It also provides an interface to the z/OS system console.

Channel subsystem (CSS)

The heart of moving data into and out of a mainframe host is the channel subsystem, or CSS. The CSS is, from a central processor standpoint, independent of the processors of the mainframe host itself. This means that input/output within a mainframe host can be done asynchronously.

The mainframe channel subsystem and network links

A CHPID no longer directly corresponds to a hardware channel, and CHPID numbers can be arbitrarily assigned. A hardware channel is now identified by a physical channel identifier, or PCHID.

Hardware channels

There are effectively three ways that network traffic can travel between an external network and a z/OS host: through a channel-command word channel, a coupling channel, or a QDIO channel.

HiperSockets

Mainframe HiperSockets is a technology that provides high-speed TCP/IP connectivity within a central processor complex. It eliminates the need for any physical cabling or external networking connection between servers running in different LPARs. The communication is through the system memory of the processor, so servers are connected to form an “internal LAN.” The HiperSockets implementation is based on the OSA-Express Queued Direct I/O (QDIO) protocol, hence HiperSockets is also called internal QDIO, or IQDIO. The microcode emulates the link control layer of an OSA-Express QDIO interface.

The I/O cage

The connections to the central processor complex are made in a physical area of the processor frame called an I/O cage. Within the cage, OSA cards and memory modules (and other devices) are physically attached to the central processor complex. Parallel, FICON, and ESCON connections are all made within the cage as well, using an adapter card.

4.1 System z network connectivity

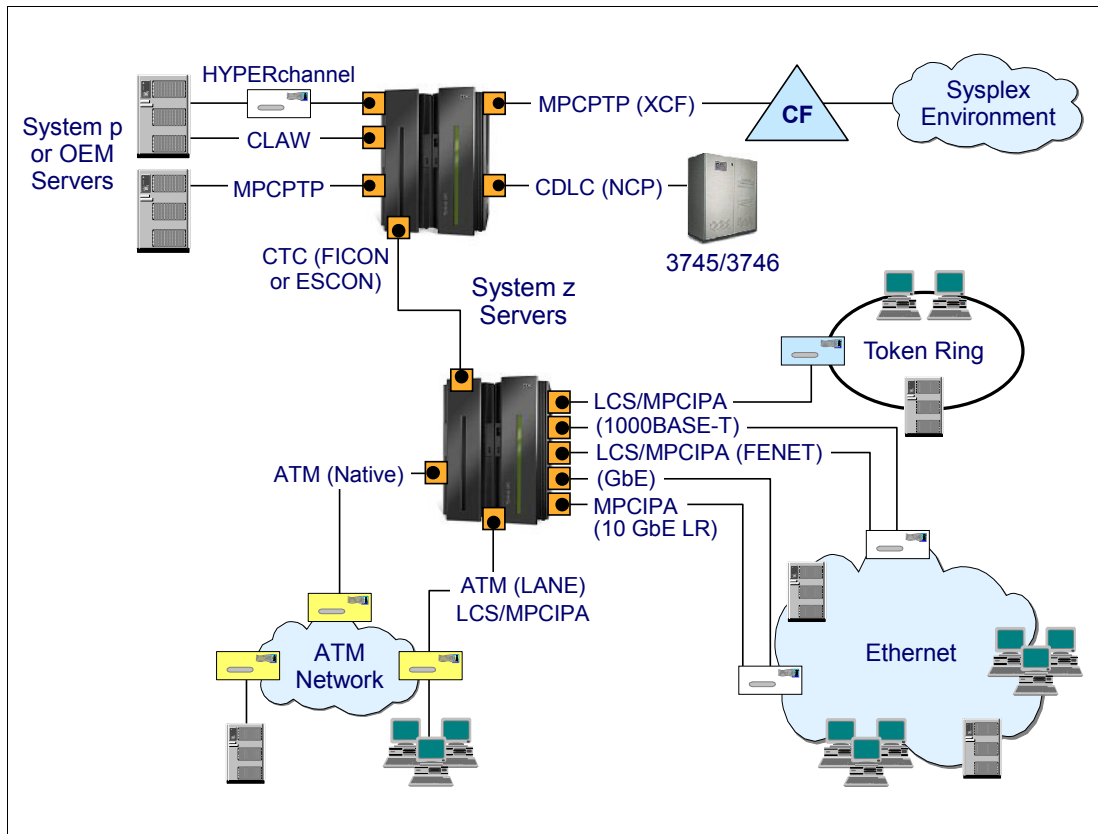


Figure 4-1 System z network connectivity

Physical computer networks

A computer network is a group of computer nodes electronically connected by some communication medium. Each node has the hardware and the programs necessary to communicate with other computer nodes across this communication medium. The node can be a PC, workstation, departmental computer, or large computer system. The size of the computer is not important. The ability to communicate with other nodes is important. The network consists of the hardware and software that enables computers to share files and resources and exchange data. Networks play a significant role in much of the world's transaction processing. A large corporation conducts daily operations over one or more networks that connect the business—locally or remotely—to partners, suppliers, and customers around the world.

A physical network consists of electrical wiring and components, such as modems, bridges, controllers, access units, telephone lines, fiber optic cables, and coaxial cables. These are used to connect the computer nodes. The physical network can connect two nodes in a single room or thousands of nodes communicating across large geographic areas. The most common networks in use today are Local Area Networks (LANs) and Wide Area Networks (WANs). LANs cover a limited distance, generally one or two floors or buildings, whereas WANs, using telecommunication facilities, are used for longer distances.

Note: This support is limited with the newer IBM servers. The supported IPv4 interfaces are listed in Table 4-1 on page 106.

Table 4-1 System z network interfaces

Interface type	Attachment type	Protocol type	Description
Asynchronous transfer mode (ATM)	ATM Native mode through OSA-Express	ATM network	Enables TCP/IP to send data to an ATM network using an OSA-Express ATM adapter.
Channel data link control (CDLC)	Network Control Program through 3745/3746	Point-to-point	ESCON attachments can be used to provide native IP transport between the 3746 IP and host systems running the z/OS Communications Server.
Common link access to workstation (CLAW)	IBM System p® Channel attached routers	Point-to-point Point-to-Multipoint	Provides access from IBM System p server directly to a TCP/IP stack over a channel. Can also be used to provide connectivity to other vendor platforms.
Channel-to-channel (CTC)	FICON/ESCON channel	Point-to-point	Provides access to TCP/IP hosts by way of a CTC connection established over a FICON or ESCON channel.
HYPERchannel	Series A devices	Point-to-Multipoint	Provides access to TCP/IP hosts by way of a series A devices and series DX devices that function as series A devices.
LAN Channel Station (LCS)	OSA-Express: <ul style="list-style-type: none"> ▶ 1000BASE-T ▶ Fast Ethernet ▶ Token Ring ▶ ATM Native and LAN Emulation 	LAN: <ul style="list-style-type: none"> ▶ IEEE802.3 ▶ IEEE802.3 ▶ IEEE802.5 ▶ ATM network 	A variety of channel adapters support a protocol called the LCS. The most common are OSA-Express features.
MultiPath Channel IP Assist (MPCIPA)	HiperSockets OSA-Express: <ul style="list-style-type: none"> ▶ 10 Gigabit Ethernet ▶ Gigabit Ethernet ▶ 1000BASE-T ▶ Fast Ethernet ▶ Token Ring ▶ ATM LAN Emulation 	Internal LAN LAN: <ul style="list-style-type: none"> ▶ IEEE802.3 ▶ IEEE802.3 ▶ IEEE802.3 ▶ IEEE802.3 ▶ IEEE802.5 ▶ ATM network 	Provides access to TCP/IP hosts, using OSA-Express in Queued Direct I/O (QDIO) mode and HiperSockets using the internal Queued Direct I/O (iQDIO).
MultiPath Channel Point-to-Point (MPCPTP)	IUTSAMEH (XCF link)	Point-to-point	Provides access to directly connect z/OS hosts or z/OS LPARs, or by configuring it to utilize Coupling Facility links (if it is part of a sysplex).
SAMEHOST (Data Link Control)	SNALINK LU0 SNALINK LU6.2 X25NPSI	Point-to-point Point-to-point X.25 network	Enables communication between z/OS Communications Server IP and other servers running on the same MVS image.

4.2 TCP/IP and network connectivity

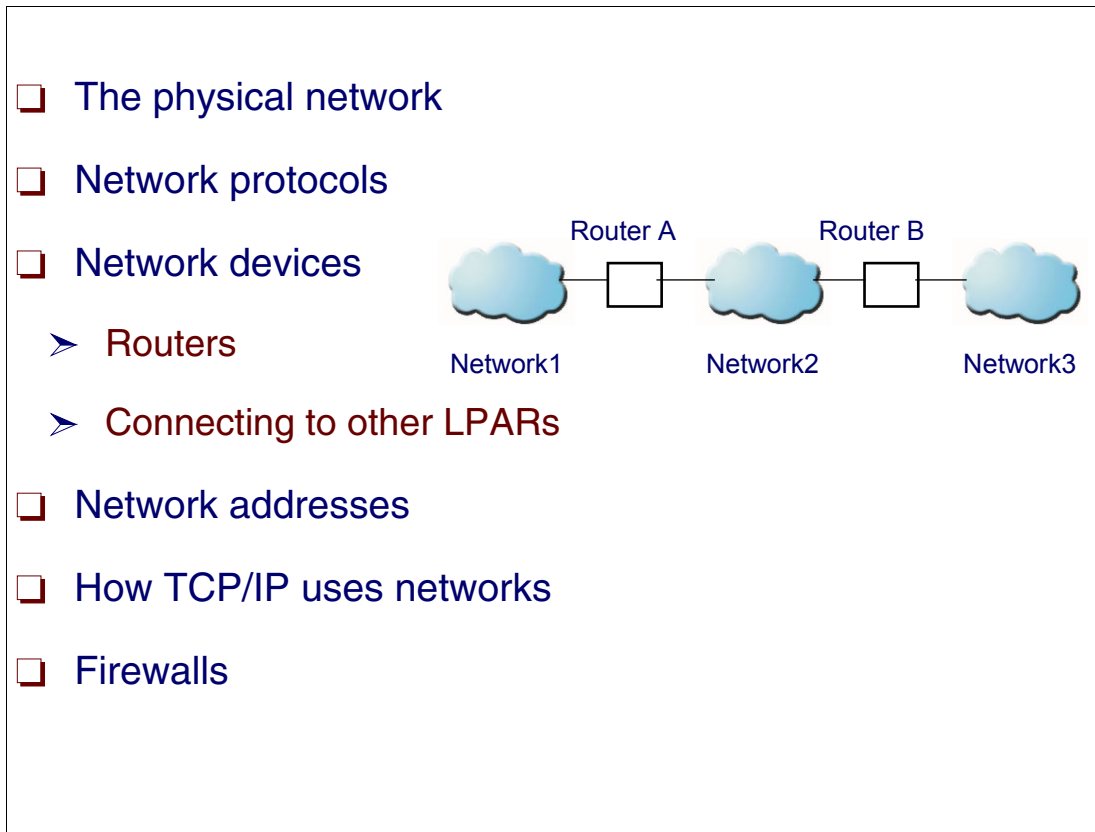


Figure 4-2 TCP/IP and network connectivity

The physical network

Computer networks enable you to share the data and computing resources of many computers. Applications, such as departmental file servers, rely on networking as a way to share data and programs. Many forms of communication media are available today. Each is designed to take advantage of the environment in which it operates. Communication media consist of a combination of the physical network used to connect the computer nodes and the language, or protocol, they use to communicate with each other.

Network protocols

TCP/IP on z/OS supports all of the well-known server and client applications. The TCP/IP started task is the engine that drives all IP-based activity on z/OS. The TCP/IP profile data set controls the configuration of the TCP/IP environment. Network protocols are the rules that define how information is delivered between nodes. They describe the sequence and contents of the data exchanged between nodes in the network. Network protocols determine how a computer node functions during communication with another node, how data is encoded to reach its destination safely, and what path it should follow. Protocols coordinate the flow of messages and can specify which node a message is destined for in the network. A variety of protocols are used to take advantage of the characteristics of each of the physical network types. The most common protocols are Ethernet, 802.3, token ring, X.25, and System Network Architecture (SNA).

Network devices

Routers are used to have z/OS talk through such a channel-attached router. This is a protocol above the channel protocol which must be agreed upon. The protocol used for this is called CLAW, as shown in Figure 4-1 on page 105. CLAW stands for Common Link Access to Workstation. CLAW can be used to talk to either a CISCO CIP (Channel Interface Processor) host or an AIX pSeries host. CLAW-connected hosts are steadily becoming less common in z/OS networks.

Connecting to other LPARs is done with a parallel channel, which is used to connect two LPARs directly, or even two separate central processor complexes. The other LPAR could be running z/OS, or it could be running z/VM with multiple Linux images within a single LPAR.

Network addresses

A network address is a component of the communication network and is associated with both hardware and software. The address is the means by which the sending node selects the receiving node for data transfer. It is also used by the receiving node to recognize what data is destined for it. An address is a unique code assigned to every node on a network. But an address is formed differently for different protocols. The length, position, and method used to specify an address are unique for each protocol. A communication node using one protocol cannot recognize the address of another protocol.

How TCP/IP uses networks

TCP/IP consists of a layered structure of protocols ranging from hardware-dependent programs to high-level applications. Each TCP/IP layer provides services to the layer above it and uses the services of the layer below it. The lowest layer, which is next to the physical layer, is not part of TCP/IP. This layer consists of existing protocols, such as Ethernet and token ring. TCP/IP uses the services of this layer to transport data across dissimilar networks, much like a gateway.

Firewalls

Firewalls are so common that a definition is hardly needed; however, in a large organization the term should be formally defined. A firewall is an implementation (or extension) of an organization's security policies. Any large organization has (or should have) a formal document explaining the classification of company data, as well as the classification of company networks. A firewall controls and limits access between networks of different security classifications, and sometimes even within a network that is already protected by a firewall. Firewalls can filter based upon port numbers and IP addresses (or networks).

Firewalls also often function as endpoints for secure communications across a non-secure network. Data travelling from the secure network outward will be secured as it crosses the non-secure network (a requirement of the organization's security policy, no doubt). Data travelling into this firewall would likewise be secured. A firewall that acts as a VPN endpoint and allows data to continue on through the secure network to destination hosts is often called a security router. The term router is traditionally used to describe a host that connects networks using different protocols.

4.3 OSA Express connectivity

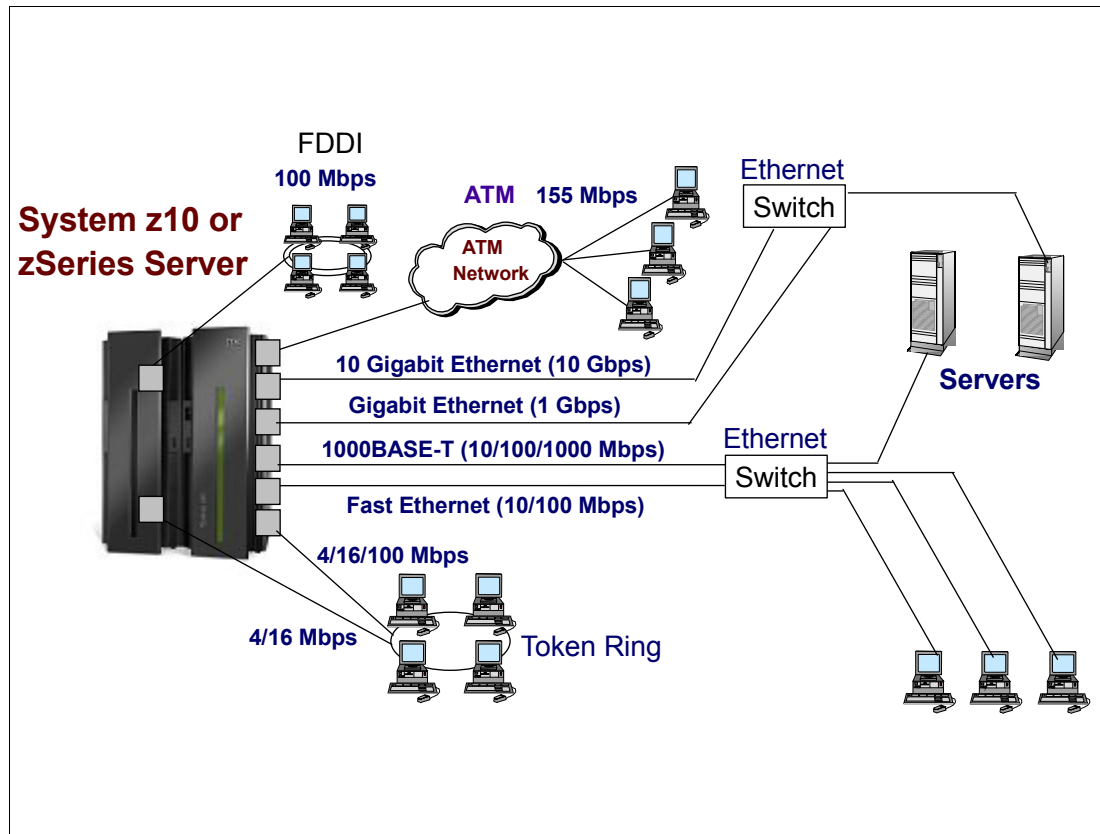


Figure 4-3 OSA Express connectivity

OSA Express connectivity

The Open Systems Adapter is actually a network controller that you can install in a mainframe I/O cage. The adapter integrates several hardware features and supports many networking transport protocols. The OSA card is the strategic communications device for the mainframe architecture. It has several key features that distinguish it from CCW¹-based communications.

Effectively, the OSA integrates the control unit and device into the same hardware. It does so by placing it on a single card that directly connects to the central processor complex I/O bus.

There are now two main versions of the Open Systems Adapter:

- ▶ The OSA-Express2
- ▶ The OSA-Express3

The Open Systems Adapter-Express (OSA-Express) features bring the strengths of System z, such as security, availability, enterprise-wide access to data, and systems management to the client/server environment. OSA-Express2 and OSA-Express3 features are designed to provide direct, industry standard, local area network (LAN) connectivity in a multi vendor networking infrastructure (Figure 4-3).

¹ channel control word

OSA-Express2

OSA-Express2 provides connectivity support to the following LAN types:

- ▶ 1000BASE-T Ethernet (10/100/1000 Mbps)
- ▶ 1 Gbps Ethernet
- ▶ 10 Gbps Ethernet

The OSA-Express2 features (10 Gigabit Ethernet, Gigabit Ethernet (GbE), and 1000BASE-T) are available on System z10 and System z9 servers; the OSA-Express3 features (10 Gigabit Ethernet, Gigabit Ethernet (GbE), and 1000BASE-T) are the current offerings on System z10.

Note: The OSA-Express2 10 GbE and GbE features have been withdrawn from marketing for the System z10 servers and can no longer be ordered.

4.4 OSA-Express3 or OSA-Express2 Ethernet connectivity

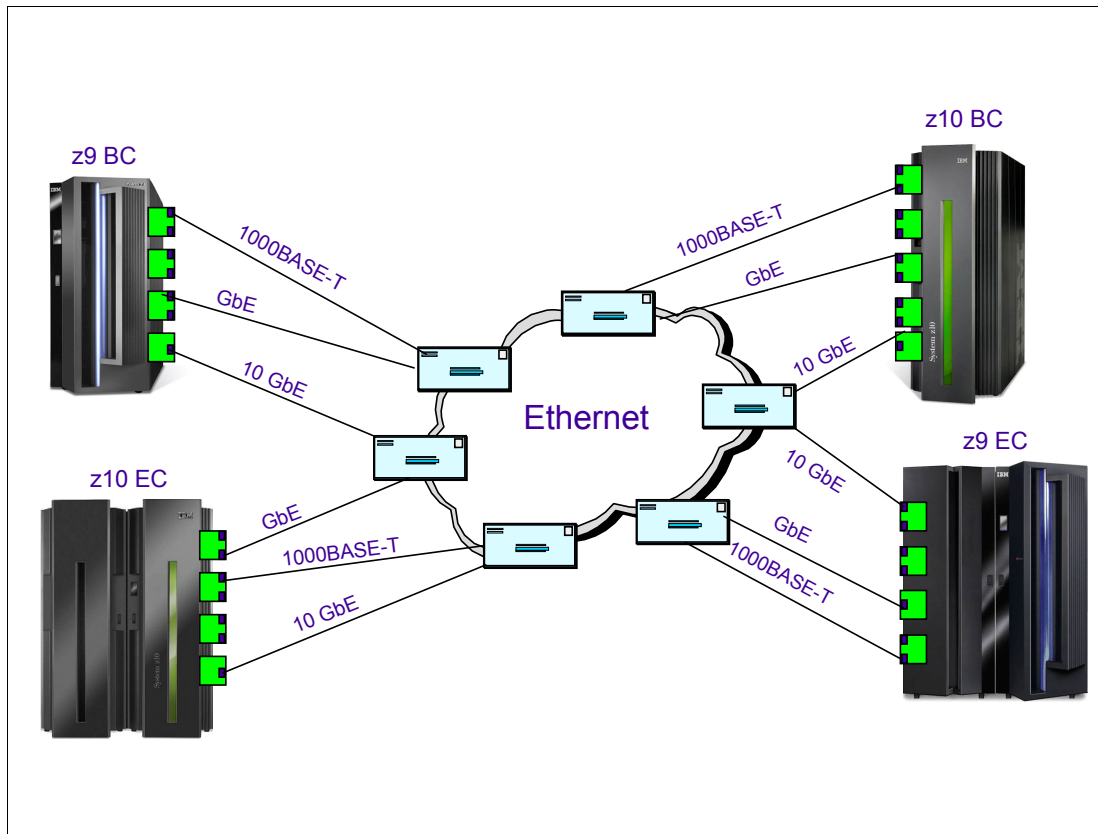


Figure 4-4 OSA-Express connectivity

OSA-Express connectivity

The OSA-Express3 and OSA-Express2 features comprise a number of integrated hardware features that can be installed in a System z I/O cage, becoming integral components of the server's I/O subsystems. They provide high function, connectivity, bandwidth, data throughput, network availability, reliability, and recovery. All OSA-Express3 and OSA-Express2 features are *hot-pluggable*.

Figure 4-4 shows the OSA-Express3 and OSA-Express2 Ethernet features available on the System z10 and System z9 servers.

Operating modes

The integration of a channel path with network ports makes the OSA a unique channel or CHPID type, recognized by the hardware I/O configuration on a port-by-port basis as one of the following:

- ▶ OSD (Queued Direct Input/Output)
- ▶ OSE (non Queued Direct Input/Output)
- ▶ OSC (OSA Integrated Console Controller)
- ▶ OSN (Open System Adapter for NCP)

Note: Not all features support all CHPID types.

4.5 Ethernet protocol

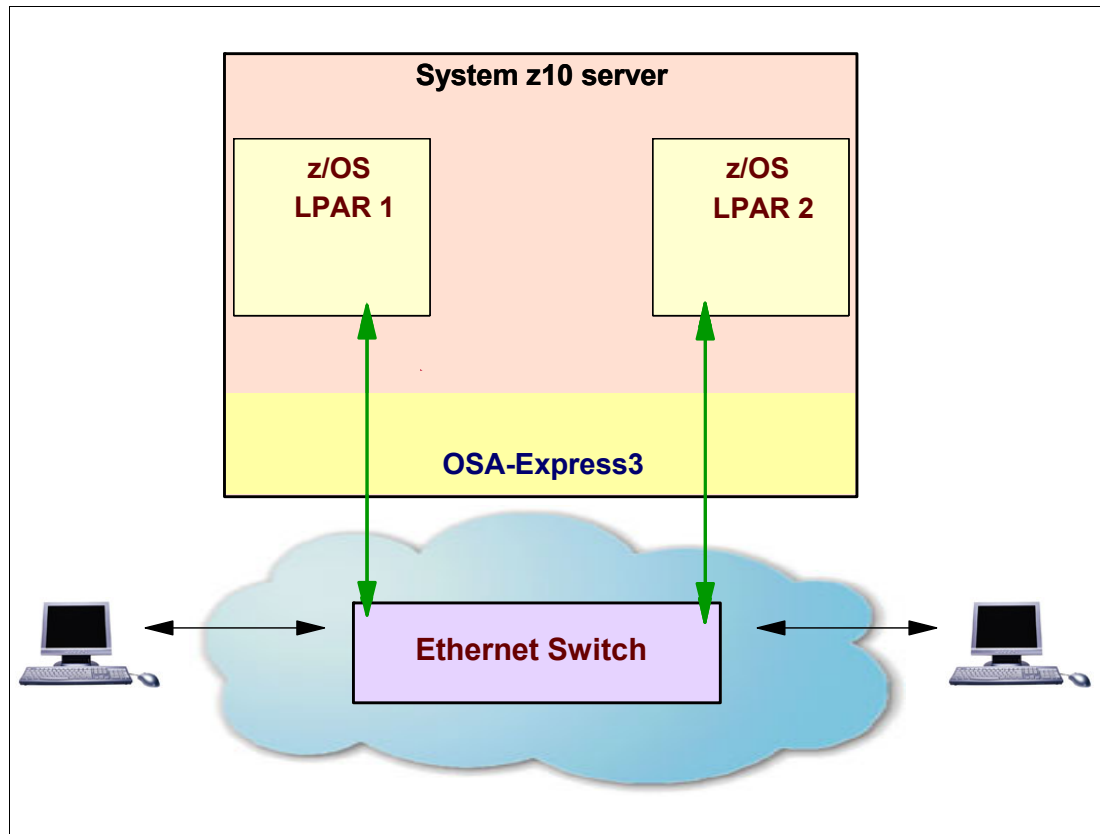


Figure 4-5 Ethernet protocol

Ethernet protocol

Like a check being placed into an envelope, the Ethernet protocol encapsulates data passed to it from higher layers. It also does the reverse: it unencapsulates data that is presented to it from the physical layer. Thus, it stuffs envelopes when data is moving down through the layers, and it opens envelopes and passes the contents upward at the receiving end. The Ethernet envelope is called a *frame*.

Ethernet technology is everywhere. It is believed that more than 90% of network installations use Ethernet. The remaining network connections are a combination of Token Ring, Fiber Distributed Data Interface (FDDI), Asynchronous Transfer Mode (ATM), and other protocols. Ethernet gained acceptance because it is simple to install and manage. The Ethernet standard was defined in 1985 by the Institute of Electrical and Electronic Engineers (IEEE) in a specification known as IEEE 802.3. The standard specifies the physical medium, carrier sense multiple access with collision detection (CSMA/CD) access method, and frame format.

In the CSMA/CD access method, each station contends for access to the shared medium. If two stations try sending packets at the same time, a collision will result. The CSMA/CD access method is designed to restore the network to normal activity after a collision occurs, and collisions are normal in an Ethernet shared network. The original 10 Mbps shared Ethernet network was based on coaxial cable physical medium, and later the standard was extended to shielded and unshielded twisted pair, and fiber optic cable media. The most common physical media is unshielded twisted pair (UTP), because it is inexpensive, easy to install, and allows star topology.

Ethernet data packet

The DLCETHER Frame Encapsulation shows the Ethernet data packet. The line contains the following: preamble (8 bytes), destination address (6 bytes), source address (6 bytes), type field (2 bytes), data (m bytes), and CRC (4 bytes), as shown in Figure 4-6.

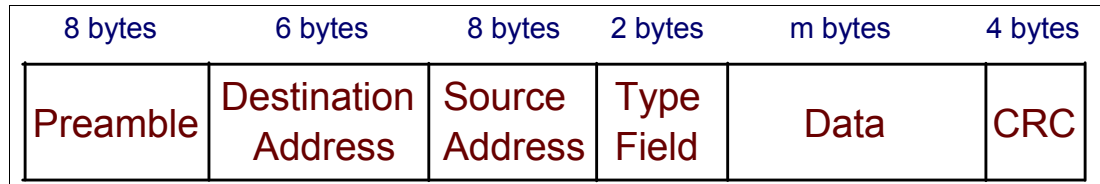


Figure 4-6 Ethernet data packet

QDIO interface isolation

Some environments require strict controls for routing data traffic between servers or nodes. In certain cases, the LPAR-to-LPAR capability of a shared OSA port can prevent such controls from being enforced. With interface isolation, internal routing can be controlled on an LPAR basis. When interface isolation is enabled, the OSA will discard any packets destined for a z/OS LPAR that is registered in the OAT as isolated.

For example, in Figure 4-5 on page 112, LPAR 1 has interface isolation enabled. Therefore, data traffic from LPAR 2 destined for LPAR 1 will be dropped by the OSA.

Important: QDIO interface isolation is supported by the Communications Server for z/OS V1R11 and all OSA-Express3 and OSA-Express2 features on System z10, and by all OSA-Express2 features on System z9, with an MCL update. Refer to the appropriate Preventive Service Planning bucket for details regarding your System z server.

z/OS V1R11 Communications Server enables a stack that is using an OSA-Express feature to prevent packets from flowing directly between two stacks that are sharing the OSA device. This is called connection isolation; when it is in effect, the OSA-Express feature discards packets whose next-hop address was registered by a sharing stack. The OSA-Express feature requires that both stacks that share the port be non-isolated for direct routing to occur.

Restrictions: OSA-Express connection isolation is supported only for OSA-Express features in QDIO mode.

OSA-Express connection isolation is not supported when the OSA-Express feature is defined using a DEVICE and LINK statement.

4.6 Non-QDIO versus QDIO data paths

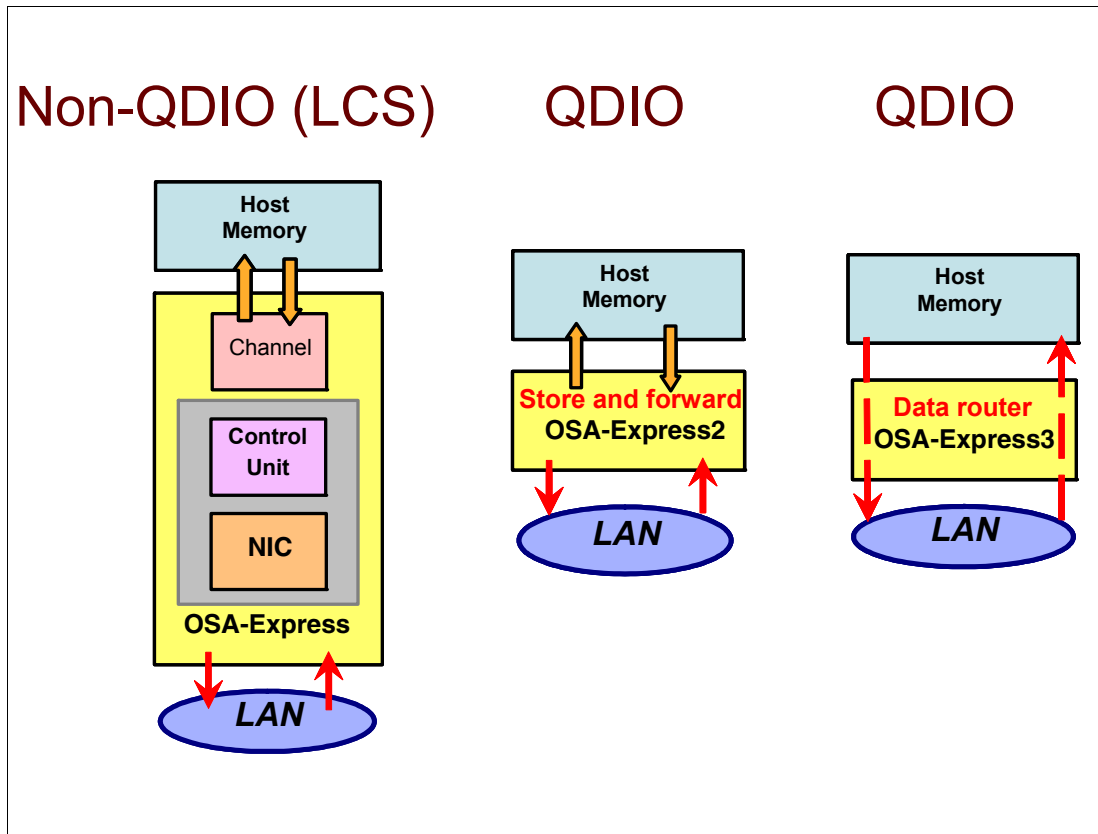


Figure 4-7 Non-QDIO versus QDIO

Non-QDIO versus QDIO

Figure 4-7 illustrates the much shorter I/O process when in QDIO mode compared with non-QDIO mode. I/O interrupts and I/O path-lengths are minimized, resulting in improved performance versus non-QDIO mode, reduction of System Assist Processor (SAP) utilization, improved response time, and server cycle reduction.

QDIO mode

QDIO is a highly efficient data transfer mechanism that is designed to dramatically reduce system overhead and improve throughput by using system memory queues and a signaling protocol to directly exchange data between the OSA microprocessor and network software. QDIO is the interface between the operating system and the OSA hardware.

The components that make up QDIO are DMA, data router (OSA-Express3 only), Priority Queuing (z/OS only), dynamic OSA Address Table building, LPAR-to-LPAR communication, and Internet Protocol (IP) Assist functions.

QDIO supports IP and non-IP traffic with the OSA-Express3 and OSA-Express2 features. These features support two transport modes: Layer 2 (Link Layer) for IP and non-IP traffic, and Layer 3 (Network Layer) for IP traffic only.

Non-QDIO mode

Like any other channel-attached control unit and device, an OSA port can execute channel programs (CCW chains) and present I/O interrupts to the issuing applications. For non-QDIO mode, the OSA ports are defined as channel type OSE. The non-QDIO mode requires the use of the OSA/SF for setup and customization of the OSA features.

The 1000BASE-T features support non-QDIO mode. This mode supports SNA/APPN/HPR and TCP/IP traffic simultaneously through the OSA port. The non-QDIO mode types are as follows:

- ▶ TCP/IP passthru

In TCP/IP passthru mode, an OSA feature transfers data between a TCP/IP stack to which it is defined and clients on an Ethernet 10/100/1000 Mbps LAN that is attached to the port on a 1000BASE-T feature, and supports one of the following frame protocols:

- Ethernet II using the DEC Ethernet V 2.0 envelope
- Ethernet 802.3 using the 802.2 envelope with SNAP

For TCP/IP passthru mode, the default OAT can be used. In that case, no configuration or setup is required.

- ▶ SNA/APPN/HPR support

In this mode, an OSA feature acts as a SNA passthru agent to clients that use the SNA protocol on the LAN that is directly attached to the OSA-Express. If an OSA feature is running in the SNA mode, it is viewed by VTAM as an external communications adapter (XCA) that can have either switched or non-switched lines of communication.

Important: In non-QDIO mode, an OSA-Express card can support SNA and APPN traffic (using 802.2 frames). In addition, in OSE mode, an OSA-Express card can run IP and SNA/APPN traffic concurrently. Some manual configuration is required, using a program called the OSA Support Facility, or OSA/SF.

When the CHPID type is set to OSE, the OSA-Express card is functioning in non-QDIO mode. An OSE channel type does not support many of the features of an OSA-Express running QDIO mode. For example, direct memory access and enhanced IP availability are only available with a channel type of OSD.

4.7 Internal Queued Direct Input/Output (iQDIO)

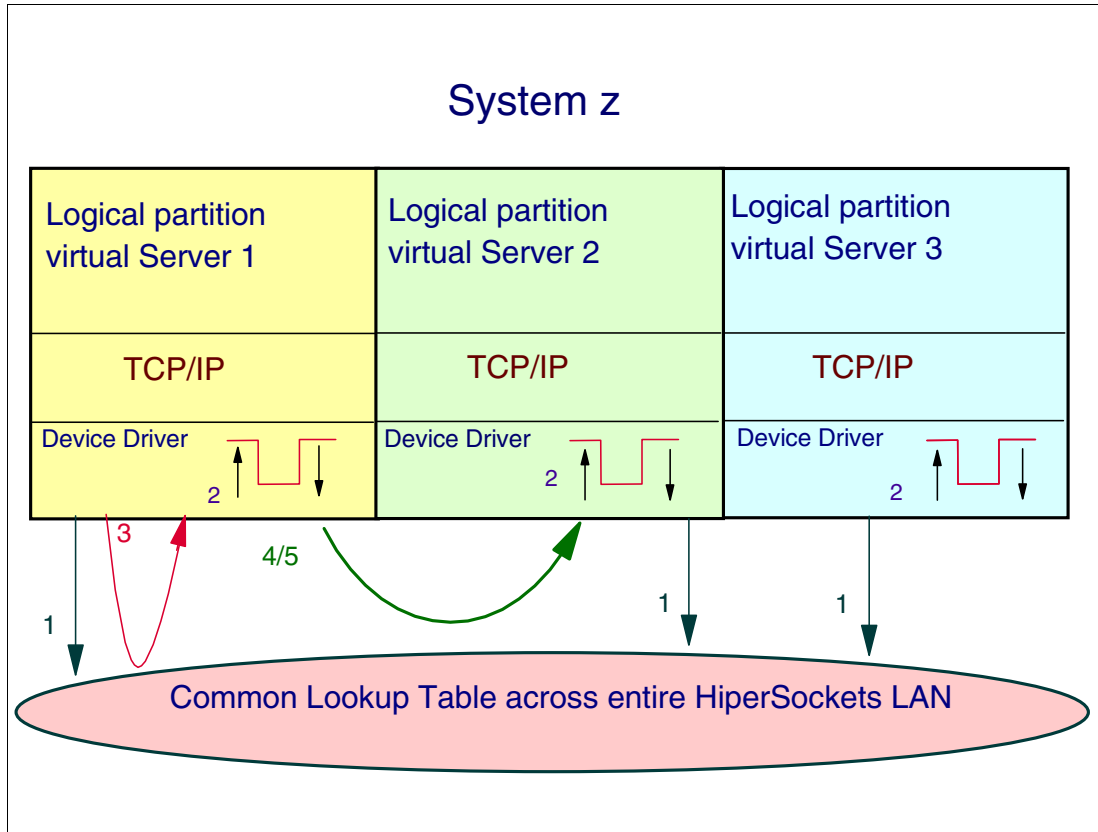


Figure 4-8 HiperSockets mode of operation

HiperSockets mode of operation

HiperSockets implementation is based on the OSA-Express Queued Direct Input/Output (QDIO) protocol, hence HiperSockets is called internal QDIO (iQDIO). The LIC emulates the link control layer of an OSA-Express QDIO interface. Typically, before you can transport a packet on an external LAN, you have to build a LAN frame, and insert the MAC address of the destination host or router on that LAN into the frame. HiperSockets do not use LAN frames, destination hosts, or routers. TCP/IP stacks are addressed by inbound data queue addresses instead of MAC addresses.

The System z LIC maintains a lookup table of IP addresses for each HiperSocket. This table represents an *internal LAN*. When a TCP/IP stack starts a HiperSockets device, the device is registered in the IP address lookup table with its IP address, and its input and output data queue pointers. If a TCP/IP device is stopped, the entry for this device is deleted from the IP address lookup table.

HiperSockets copy data synchronously from the output queue of the sending TCP/IP device to the input queue of the receiving TCP/IP device by using the memory bus to copy the data through an I/O instruction.

The controlling operating system that performs I/O processing is identical to OSA-Express in QDIO mode. The data transfer time is similar to a cross-address space memory move, with latency close to zero. To get a data move total elapsed time, you have to add the operating system I/O processing time to the LIC data move time.

HiperSockets operations are executed on the CP where the I/O request is initiated. HiperSockets starts read or write operations. The completion of a data move is indicated by the sending side to the receiving side with a Signal Adapter (SIGA) instruction. Optionally, the receiving side can use dispatcher polling instead of handling SIGA interrupts. The I/O processing is performed reducing demand on the System Assist Processor (SAP). This new implementation is also called *thin interrupt*.

HiperSockets operational flow

Figure 4-8 on page 116 shows the HiperSockets operational flow, which consists of five steps:

1. Each TCP/IP stack (image) registers its IP addresses into the HiperSockets server-wide Common Address Lookup table. There is one lookup table for each HiperSockets internal LAN. The scope of each LAN is the logical partitions that are defined to share the HiperSockets IQD CHPID.
2. The address of the TCP/IP stack's receive buffers are appended to the HiperSockets queues.
3. When data is being transferred, the send operation of HiperSockets performs a table lookup for the addresses of the sending and receiving TCP/IP stacks and their associated send and receive buffers.
4. The sending processor copies the data from its send buffers into the target CP processor's receive buffers (System z server memory).
5. The sending processor optionally delivers an interrupt to the target TCP/IP stack. This optional interrupt uses the thin interrupt support function of the System z server, which means the receiving host is going to "look ahead," detecting and processing inbound data. This technique reduces the frequency of real I/O or external interrupts.

Note: You must define the source and destination interfaces to the same HiperSockets.

Configuring HiperSocket TCP/IP devices

Configuration of HiperSockets TCP/IP devices is similar to that of OSA-Express QDIO devices. Each HiperSockets requires the definition of a channel path identifier (CHPID) similar to any other I/O interface. HiperSockets is not allocated a CHPID until it is defined. It also does not take an I/O cage slot. Customers who have used all the available CHPIDs on the server cannot enable HiperSockets; therefore, you must include HiperSockets in the customer's overall channel I/O planning. The CHPID type for HiperSockets is IQD, and the CHPID number must be in the range from hex 00 to hex FF. No other I/O interface can use a CHPID number defined for a HiperSockets, even though HiperSockets does not occupy any physical I/O connection position.

4.8 HiperSockets

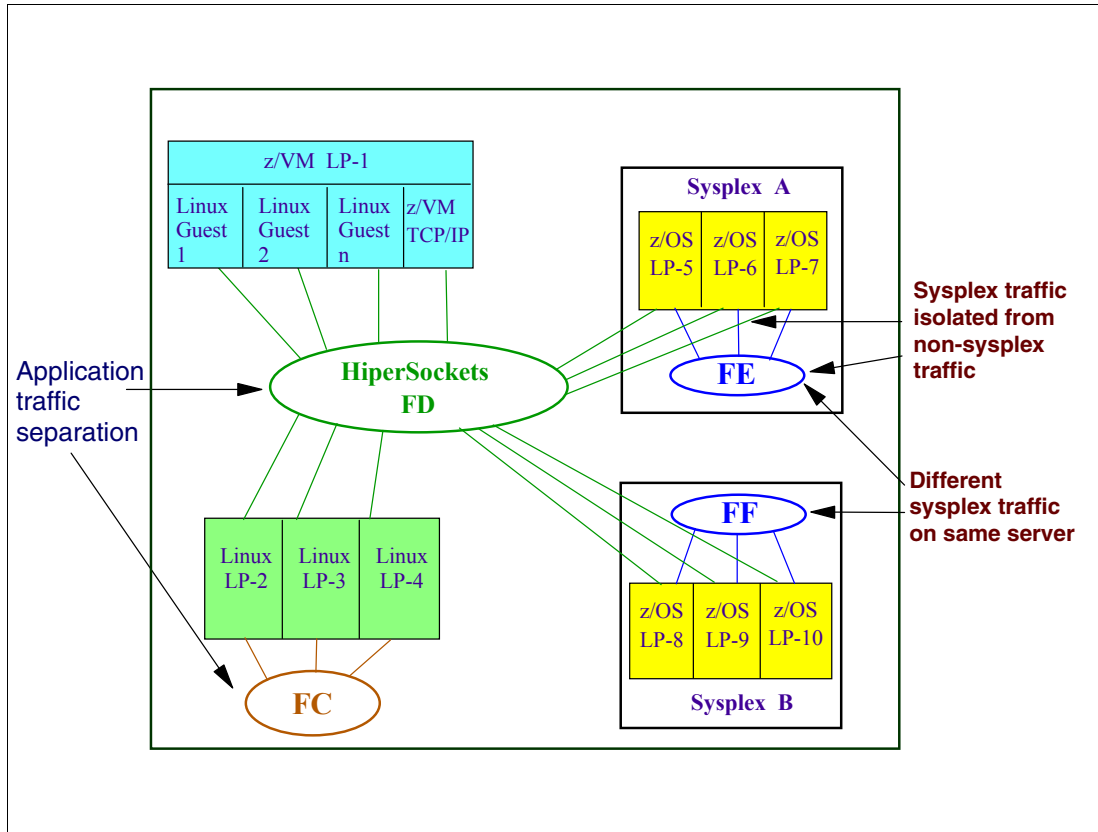


Figure 4-9 HiperSockets

HiperSockets

HiperSockets provide a seamless network to consolidate servers into an advanced infrastructure intraserver network. HiperSockets creates multiple independent, integrated, virtual LANs within System z. Up to 16 HiperSockets can be configured on System z servers.

HiperSockets are accessible among combinations of logical partitions or virtual servers within the System z. This *network within the box* concept minimizes network latency and maximizes bandwidth capabilities between z/VM, Linux on System z, z/VSE™, and z/OS images, or combinations of these. It is also possible to have HiperSockets under z/VM, which permits internal networks between guest operating systems, such as many Linux servers, for example. Each HiperSockets LAN uses a Channel Path ID (CHPID) with a channel type IQD. Figure 4-9 shows an example of HiperSockets connectivity with multiple LPs and virtual servers.

HiperSockets on System z is a technology that provides high-speed connectivity between virtual servers within a System z server. It eliminates the need for any physical cabling or external networking connection between these virtual servers.

Note: HiperSockets are implemented in the Licensed Internal Code (LIC) of a System z, with the communication path being within system memory. The connections between the virtual servers form a virtual LAN. HiperSockets uses internal Queued Direct Input/Output (iQDIO) at memory speed to transfer information between the virtual servers.

4.9 Open Systems Adapter Support Facility (OSA/SF)

- OSA/SF is used primarily to:
 - Manage all OSA ports
 - Configure all OSA non-QDIO ports
 - Configure local MAC
 - Display registered IPv4 addresses
 - It is supported on System z servers for QDIO ports
 - Display registered IPv4 or IPv6 Virtual MAC and VLAN ID associated with all OSA Ethernet features configured as QDIO Layer 2
 - Provide status information about an OSA port

Figure 4-10 Open Systems Adapter Support Facility (OSA/SF)

Open Systems Adapter Support Facility (OSA/SF)

The TCP/IP subagent can retrieve SNMP management data from the Open Systems Adapter Support Facility (OSA/SF) for several OSA adapters. The OSA product also provides an SNMP subagent, the OSA-Express Direct subagent, which supports management data for OSA-Express adapters. The OSA-Express Direct subagent can be used with Communications Server SNMP support to retrieve the management data. You should use the OSA-Express Direct subagent for OSA management data, rather than the TCP/IP subagent, because the OSA-Express Direct subagent communicates directly with the OSA-Express adapters and does not require the OSA/SF and IOASMP applications.

OSA/SF includes a Java-based graphical user interface (GUI) in support of the client application. The Java GUI is independent of any operating system or server (transparent to the operating system), and is expected to operate wherever the current Java runtimes are available.

Use of the GUI is optional. A REXX command interface is also included with OSA/SF. OSA/SF is not required to set up the OSA features in QDIO mode (CHPID type OSD), but it can be used for monitoring and controlling ports. OSA/SF has been, and continues to be, integrated in z/OS, z/VM, and z/VSE, and runs as a host application. For OSA/SF, Java GUI communication is supported via TCP/IP only. In the past, communication was supported via EHLLAPI (3270), APPC, and TCP/IP.

This integrated version of OSA/SF is a complete replacement for the currently integrated versions in z/OS, z/VM, and z/VSE. This version of OSA/SF is not being offered as a separately orderable program product.

The OSA/SF is used primarily to:

- ▶ Manage all OSA ports.
- ▶ Configure all OSA non-QDIO ports.
- ▶ Configure local MAC.
- ▶ Display registered IPv4 addresses (in use and not in use). It is supported on System z servers for QDIO ports.
- ▶ Display registered IPv4 or IPv6 Virtual MAC and VLAN ID associated with all OSA Ethernet features configured as QDIO Layer 2.
- ▶ Provide status information about an OSA port (shared or exclusive use state).

This support is applicable to all OSA features on System z servers. With z/OS, a second interface using a set of REXX EXECs through the Time Sharing Option Extensions (TSO/E) can be used to control the OSA features defined to System z servers on which the TSO/E is running.

Note: OSA/SF is not always required to customize an OSA feature, but is highly recommended to gather operational information and to assist in problem determination. The OSA/SF Query function provides performance information about the OSA CHPIDs.

OSA/SF is not required to configure the OSA features in operating modes OSD, OSC, and OSN.

VLAN support IPv4 and IPv6

The IBM Open Systems Adapter provides support for IEEE standards 802.1p/q, which describes priority tagging and VLAN identifier tagging. Deploying VLAN IDs allows a physical LAN to be partitioned or subdivided into discrete virtual LANs. This support is provided by the z/OS TCP/IP stack and the OSA-Express feature in QDIO mode.

When you use VLAN IDs, the z/OS TCP/IP stack can have multiple connections to the same OSA-Express feature. One connection is allowed for each unique combination of VLAN ID and IP version (IPv4 or IPv6). Each connection is defined by an INTERFACE statement and uses one channel unit address for communication, which is assigned by VTAM from the DATAPATH parameter of the TRLE definition.

4.10 Virtual LAN (VLAN) overview

- A VLAN configuration provides several benefits:
 - VLANs can improve network performance by reducing traffic on a physical LAN
 - VLANs can enhance security by isolating traffic
 - VLANs provide more flexibility in configuring networks
 - VLANs can be used to increase link optimization by allowing networks to be organized for optimum traffic flow through implementation of network segregation and a quality of service policy
 - VLANs can be used to increase bandwidth and reduce overhead
 - Types of connections

Figure 4-11 VLAN overview

VLAN overview

A virtual LAN (VLAN) is the grouping of workstations, independent of physical location, that have a common set of requirements. VLANs have the same attributes as physical LANs, although they might not be located physically on the same LAN segment.

VLAN benefits

A VLAN configuration provides several benefits, including:

- ▶ VLANs can improve network performance by reducing traffic on a physical LAN.
- ▶ VLANs can enhance security by isolating traffic.
- ▶ VLANs provide more flexibility in configuring networks. A multiple VLAN function allows a customer to consolidate multiple OSAs (for example, three 1 Gb OSA ports) into a single OSA (for example, one 10 Gb OSA port) serving multiple VLANs. In addition, multiple VLAN function allows consolidation of multiple application servers across multiple stacks into a single z/OS image where the traffic related to these servers is on unique VLANs.
- ▶ VLANs can be used to increase link optimization by allowing networks to be organized for optimum traffic flow through implementation of network segregation and a quality of service policy. Beginning with z/OS V1R9, a policy-based routing function allows a z/OS stack to make routing decisions for IPv4 traffic that take into account additional criteria such as job name, source port, destination port, protocol type (TCP or UDP), source IP address, NetAccess security zone, and multilevel secure environment security label. It

enables routing traffic meeting a certain criteria to one VLAN and traffic meeting a different criteria to another VLAN.

- ▶ VLANs can be used to increase bandwidth and reduce overhead.

Types of connections

VLANs operate by defining switch ports as members of virtual LANs. Devices on a VLAN can use three types of connections, based on whether the connected devices are VLAN-aware or VLAN-unaware. VLAN-aware devices understand VLAN memberships (which users belong to a particular VLAN) and VLAN formats.

Ports used to attach VLAN-unaware equipment are called *access ports*, whereas ports used to connect to other switches or VLAN-aware servers are known as *trunk ports*. Network frames generated by VLAN-aware equipment are marked with a *tag*, which identifies the frame to the VLAN.

4.11 OSA VLAN support

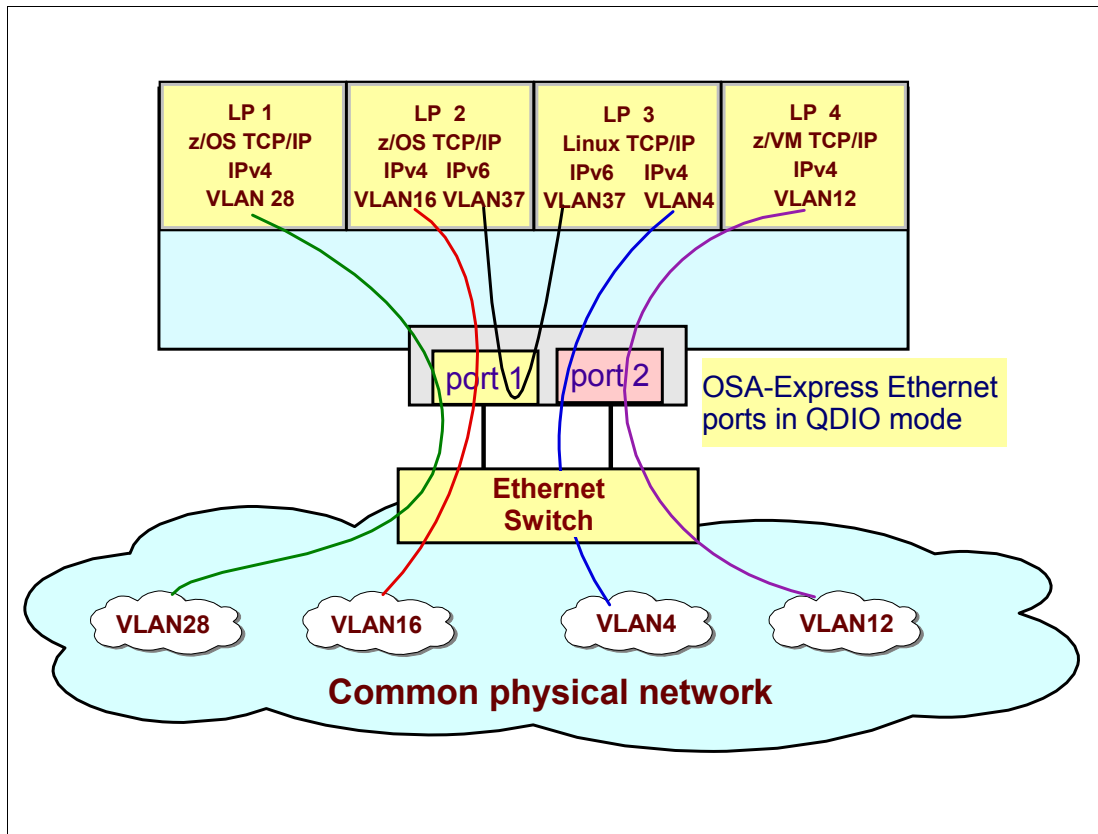


Figure 4-12 OSA VLAN support

Virtual LAN (VLAN)

A local area network (LAN) is a broadcast domain. Nodes on a LAN can communicate with each other without a router, and nodes on different LANs need a router to communicate. A virtual LAN (VLAN) is a configured logical grouping of nodes using switches. Nodes on a VLAN can communicate with each other as though they were on the same LAN, whereas nodes on different VLANs need a router to communicate.

Although a LAN segment represents a physically contiguous network with ARP broadcast capabilities, it might also be desirable to divide such a LAN into one or more logical LANs. Such a LAN is called a virtual LAN (VLAN). A VLAN is implemented as an extension to the 802.3 protocol and is defined as 802.1Q.

When using 802.1Q protocol, frames leaving a host are tagged with a VLAN ID. The VLAN ID causes the packet to be recognized only by other hosts that have adapters activated to recognize that same VLAN ID. The result is that multiple VLANs can exist completely independent of each other on a single physical segment.

Note: The advantage to this is that congestion on a LAN segment can be reduced and security can be improved by isolation of the traffic.

In addition, the VLAN ID can span multiple switches in a corporation. Thus, a VLAN ID can differentiate traffic across a network.

VLAN support in z/OS

Full VLAN support is offered for all OSA Ethernet features available on System z servers. z/OS Communications Server supports VLAN IDs. Support is offered for up to eight global VLAN IDs per OSA port, based on the IP version:

- ▶ Eight Global VLAN IDs for IPv4
- ▶ Eight Global VLAN IDs for IPv6

VLAN support in z/VM

z/VM exploits the VLAN technology and conforms to the IEEE 802.1q standard. Support is offered for one global VLAN ID per OSA port, based on the IP version:

- ▶ One Global VLAN ID for IPv4
- ▶ One Global VLAN ID for IPv6

The z/VM TCP/IP stack supports one VLAN ID per OSA port. Each port can be configured with a different VLAN ID.

VLAN support in Linux on System z

VLAN support in a Linux on System z environment is available for the OSA Ethernet features operating in QDIO mode.

Note: When you use VLAN IDs, the z/OS TCP/IP stack can have multiple connections to the same OSA-Express feature. One connection is allowed for each unique combination of VLAN ID and IP version (IPv4 or IPv6). Each connection is defined by an `INTERFACE` statement and uses one channel unit address for communication, which is assigned by VTAM from the `DATAPATH` parameter of the `TRLE` definition.

4.12 Dynamic XCF connectivity

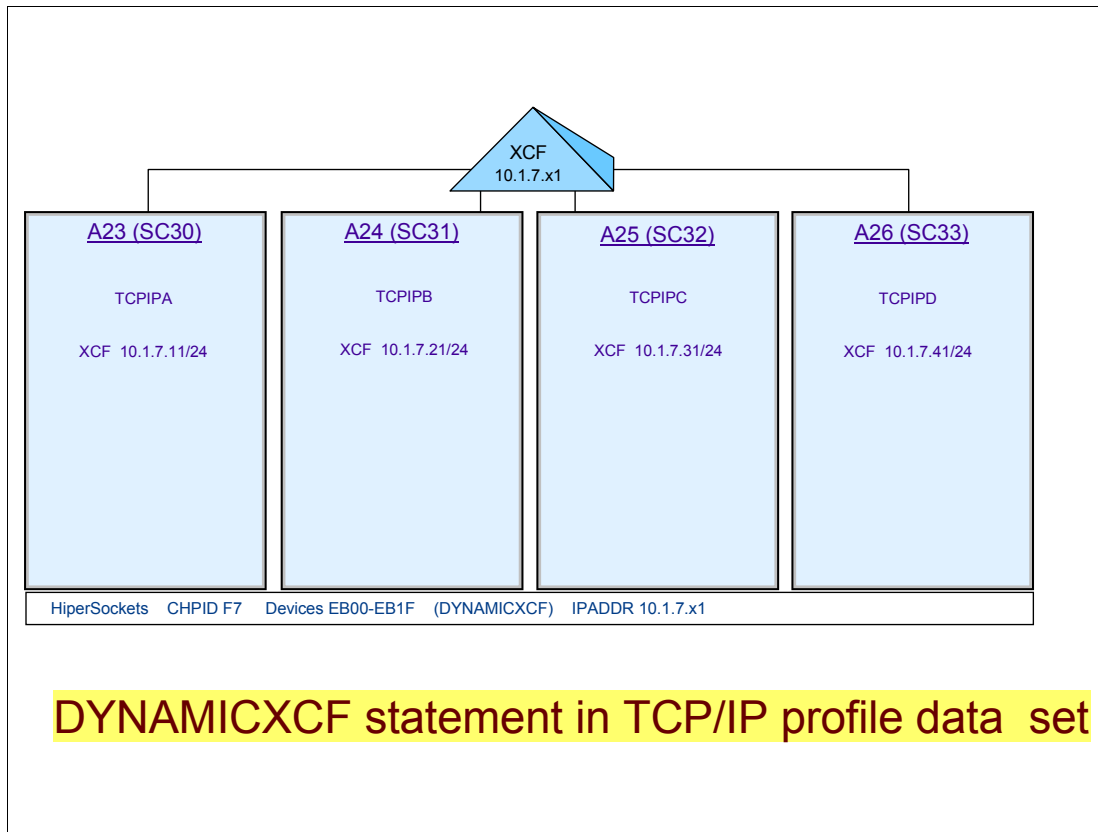


Figure 4-13 Dynamic XCF connectivity

Dynamic XCF connectivity

The connectivity scenario shown in Figure 4-13 connects all images within the same sysplex environment through a dynamic XCF connection that is created by the DYNAMICXCF definition in the TCP/IP profile data set. For most point-to-point links, a unique pair of IP addresses is needed for each stack within a sysplex. This requirement tends to use more IP addresses; however, IP addresses can be saved by using dynamic XCF. Dynamic XCF creates a single IP address by which all other stacks in a sysplex can reach a stack. Dynamic XCF creates trusted, internal links to other stacks within a sysplex, generating dynamic definitions for TCP/IP stacks that reside on another z/OS host in a sysplex or for TCP/IP stacks that reside on the same z/OS host.

When dynamic XCF is functional within a sysplex, a point-to-multipoint network is established among all participating LPARs. Each host in the sysplex has a direct connection to any other host in the same sysplex.

TCP/IP profile data set

Within the TCP/IP profile data set, there is only one configuration option required, DYNAMICXCF. This option falls within the IPCONFIG statement group, as shown in the DYNAMICXCF option of the IPCONFIG statement, as follows:

```
IPCONFIG
    DYNAMICXCF 192.168.80.1 255.255.255.0 2
```

This definition would cause the TCP/IP stack to create a link using the IP address of 192.168.80.1 within the sysplex subnetwork. This IP address would be directly reachable on the sysplex point-to-multipoint network by any other TCP/IP stack in the sysplex that also has DYNAMICXCF coded. Each TCP/IP stack would code a unique IP address within the same subnetwork.

Configuring DYNAMICXCF

To implement the XCF connections, you can use three different types of devices:

- ▶ DynamicXCF HiperSockets device (IUTIQDIO) for connections between z/OS LPARs within the same server
- ▶ DynamicXCF SAMEHOST device (IUTSAMEH) for stacks within the same LPAR
- ▶ VTAM dynamically created ISTLSXCF to connect z/OS LPARs in other servers within the same sysplex

When a TCP/IP stack becomes active in the sysplex and this stack has DYNAMICXCF coded, the following sequence of events occurs internally within the TCP/IP stack:

1. A DEVICE statement for this stack's XCF device is automatically generated.
2. A corresponding LINK statement is automatically generated.
3. A HOME statement entry using the DYNAMICXCF IP address is added to the active HOME list for the stack.
4. The device is started.

Note: If a TCP/IP stack does not have DYNAMICXCF coded, it does not participate in the dynamic XCF communications. In other words, both end points must code DYNAMICXCF in order for a link to be established.

4.13 Enterprise Extender (EE)

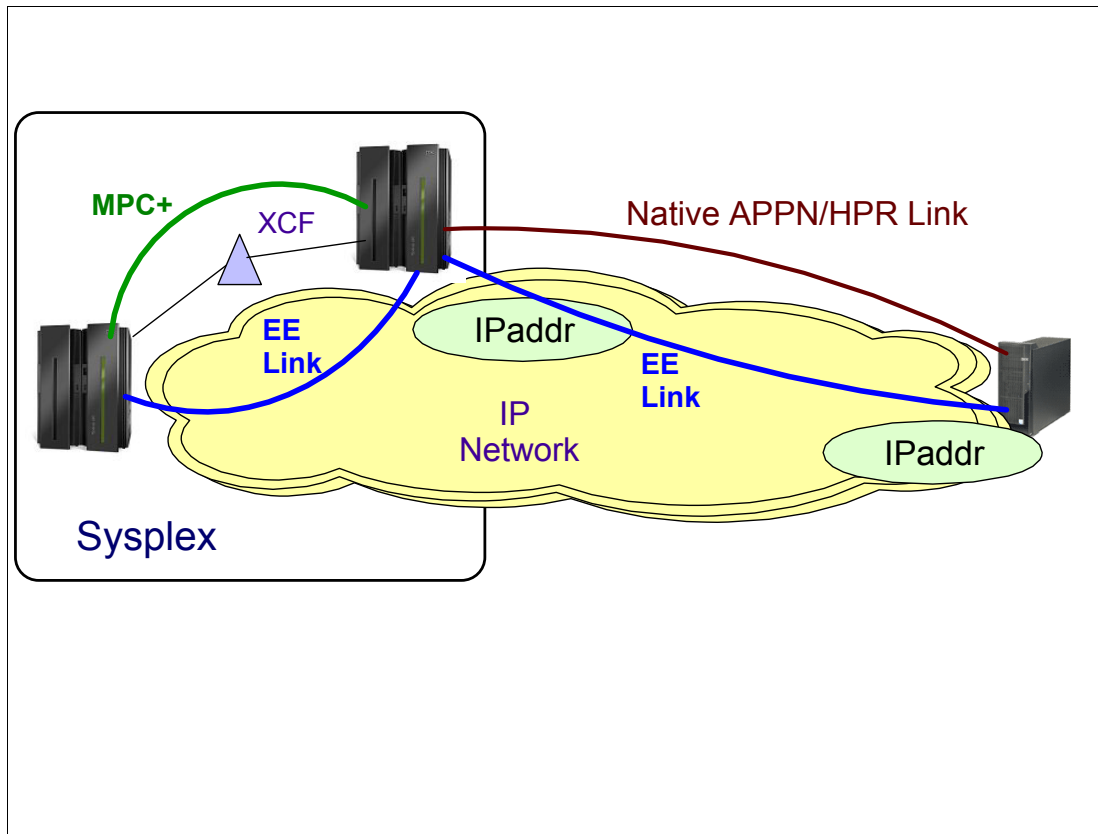


Figure 4-14 Enterprise Extender

Enterprise Extender (EE)

The Enterprise Extender (EE) technology was implemented on most IBM networking platforms during 1998. Its main objective is to provide SNA-over-IP integration that is significantly superior to its predecessors, such as data link switching (DLSw) and AnyNet. AnyNet is no longer supported in Communications Server for z/OS V1R8 and later.

Enterprise Extender has provided a useful solution to the dilemma of running SNA applications over IP networks. “Extending the enterprise” is an appropriate description. Enterprise Extender is a standard created by the Internet Engineering Task Force (IETF) and APPN Implementers' Workshop (AIW). It is documented in RFC 2353.

Because of its design, the EE architecture is extremely flexible. It can be used in all networks from the smallest to the largest, and provides the network architect with a wide choice of locations within the network to place the SNA/IP boundary. EE exploits all the latest routing performance enhancements provided by APPN/HPR.

APPN HPR technology

EE uses APPN's High-Performance Routing (HPR) technology to provide encapsulation of SNA application traffic within UDP frames by HPR-capable devices at the *edges* of an IP network. These edge devices are called *endpoints*. This is illustrated in Figure 4-14. To the IP network, the SNA traffic is UDP datagrams that get routed without hardware or software changes to the IP network. To the SNA application, the session is normal SNA with predictable performance and high availability. By wrapping the SNA application in this way, EE

enables SNA data to be carried over an IP network without changing either the SNA applications or the IP hardware.

EE architecture

The Enterprise Extender architecture carries the SNA High-Performance Routing (HPR) traffic of any logical unit type over an IP infrastructure without requiring changes to that infrastructure. It treats the IP network as a particular type of SNA logical connection. In this manner, the SNA protocols act as transport protocols on top of IP, as does any other transport protocol such as Transmission Control Protocol. An important aspect of Enterprise Extender is the ability to view the IP network as an APPN connection network. In this case, the benefit comes from the ability to establish dynamically a single one-hop HPR link to any host to which IP connectivity is enabled, provided that the host implements Enterprise Extender. In general, this allows the routing function to be handled entirely within IP. IP routers serve as the only routing nodes (hosts) in the network.

APPN and High-Performance Routing protocols

EE is valid only for APPN configurations and is an extension of APPN and High-Performance Routing (HPR) protocols. Your subarea network must first be migrated to APPN for the subareas where EE is to be deployed.

Implementing EE requires little change to your APPN-enabled network. There are four categories of definitions needed for VTAM:

- ▶ Start options
- ▶ EE XCA major node
- ▶ Model PU major node definition
- ▶ Switched major node definitions

There are three definitions needed for the TCP/IP stack profile:

- ▶ A static VIPA for EE partner communication
- ▶ The IUTSAMEH device for VTAM-to-TCP/IP communication
- ▶ A port reservation for the EE ports

Note: Communications Server provides internal links between TCP/IP stacks that are running within the same MVS image. This support is referred to as a Same Host (IUTSAMEH) link. If DYNAMICXCF is defined, TCP/IP always creates and activates a same host (IUTSAMEH) device and link (unless a static IUTSAMEH device is already defined) even if this is the only stack on the MVS image. When TCP/IP activates the IUTSAMEH device, VTAM dynamically builds the IUTSAMEH TRLE. The generated device name is IUTSAMEH, the generated link name is EZASAMEMVS (IPv4), and the generated interface name is EZ6SAMEMVS (IPv6). As other stacks are brought up within the same MVS image, a host route is created to each of these stacks across the same host link. It is recommended that users do not configure a static device for IUTSAMEH (allow TCP/IP to dynamically create the device and link). Communications Server also uses the IUTSAMEH link for Enterprise Extender support.

The IUTSAMEH DEVICE and LINK (IPv4) or INTERFACE (IPv6) statements will not become active until either another TCP/IP stack or Enterprise Extender is activated on this MVS image.

4.14 APPN/HPR functions

- ❑ HPR is an extension to the base APPN architecture
 - It can be implemented on an APPN network node or an APPN end node
 - HPR provides enhancements to APPN
- ❑ HPR is a set of enhancements for base APPN whose main objectives are:
 - To improve APPN data routing
 - To improve APPN reliability and performance
 - To provide compatibility with base APPN
 - To enable easy migration to higher speed technologies (gigabit and beyond)

Figure 4-15 APPN/HPR architecture

APPN/HPR architecture features

One of the main reasons for networks to migrate to APPN is the implementation of a sysplex. A sysplex environment provides some unique advantages to the VTAM installation, namely generic resources (GR) and multinode persistent sessions (MNPS). With the latest emphasis on IP application load balancing within the sysplex through the use of Sysplex Distributor, and SNA application load balancing within the sysplex through the use of generic resources, migrating to an HPR/IP (EE) infrastructure has become a natural step toward supporting SNA applications through native IP connectivity to the mainframe. HPR/IP accommodates the presence of SNA data within the IP packet, and exploits the dynamic routing capabilities of both HPR and IP.

APPN/HPR improves the availability of sysplex resources. Generic resources enables multiple application copies within a sysplex to present a single image to the user, resulting in better performance through load balancing and improved availability. The multinode persistent sessions function enables sessions to survive the failure of a VTAM, or a z/OS system, or even a processor, without disruption.

HPR features

The two important features of HPR are:

- ▶ Automatic Network Routing (ANR) on intermediate nodes
 - ANR is designed to function on the *intermediate nodes* along the transport path, routing data quickly and efficiently from one link to another, without regard to session awareness.

► Rapid Transport Protocol (RTP) on endpoints

RTP is implemented in the *endpoints* of a connection, providing error recovery from packet loss and support of control flows between endpoints. It does have session awareness.

Note: RTP is a connection-oriented, full-duplex protocol designed to support data in high-speed networks. RTP connections are established within an HPR subnet and are used to carry session traffic. These connections can be thought of as transport pipes over which sessions are carried.

You can use an IP network for SNA sessions with Enterprise Extender, which provides enablement of IP applications and convergence on a single network transport while preserving SNA application and endpoint investment. An EE connection is a logical connection that represents IP connectivity from a host to a specified IP address or host name. Conceptually, an IP network looks like an APPN/HPR transmission group (TG) in a session route.

Restriction: Because EE relies on the APPN/HPR protocols, only SNA sessions are supported. SNA sessions involving both independent LUs and dependent LUs are supported across EE connections, with DLUR providing the boundary services for the dependent LUs.

4.15 High-Performance Routing (HPR)

- ❑ High-Performance Routing (HPR) is a set of enhancements for APPN whose main objectives are:
 - Improved APPN data routing by performing error recovery at the endpoints only
 - Improved APPN reliability by dynamic path switching around failed paths
 - Compatibility with base APPN functions and use of the common topology
 - Migration path to gigabit networks
- ❑ HPR error recovery implemented at an endpoint
 - Major part of endpoint function is called rapid transport protocol (RTP)
 - RTP provides a reliable end-to-end connection for sessions using HPR

Figure 4-16 High-Performance Routing (HPR)

High-Performance Routing (HPR)

High-Performance Routing (HPR) is an addition to APPN that improves reliability, increases network performance, and was designed to exploit higher link speed technologies. HPR allows you to migrate network control program (NCP) connections to APPN connections without incurring the associated increase in storage and cycles. HPR utilizes a rapid transport protocol (RTP) connection to transport session traffic between session endpoints. HPR routes can also traverse an existing subarea network because HPR support provides for the mapping of HPR routes over VR-based TGs.

Note: Migrating NCP connections to APPN connections results in the creation of additional NCP control blocks, which can require additional storage space and cycles in the NCP. In addition, when a node or link along a route fails, sessions using that route fail.

In addition, if a node or link on an HPR route fails and an alternate HPR route exists between HPR session endpoints, HPR automatically switches routes and sessions are not disrupted. A method is also available to allow the operator to force an HPR path switch.

Note: To use HPR over an NCP you must have, at minimum, NCP Version 7 Release 3 (and 37xx controllers supported by NCP Version 7 Release 3). To use HPR over a subnetwork boundary where one or both of the border nodes are composite network nodes, you must have, at minimum, NCP Version 7 Release 5.

HPR enhancements for APPN

High-Performance Routing is a set of enhancements for APPN that meets the following requirements:

- ▶ **Improved APPN data routing:** HPR transports data at high speeds by using low level intermediate routing and by minimizing the number of flows over the links for error recovery and flow control protocols. The flows are minimized by performing these functions at the endpoints rather than at each hop (link) along the path.
- ▶ **Improved APPN reliability:** HPR switches paths within the HPR portion of the network to bypass link and node failures if an acceptable alternate path is available. This occurs transparently to the sessions; in other words, the session is not disrupted.
- ▶ **Functional equivalence:** HPR maintains functional equivalence with APPN. To do so, HPR continues to support priority routing, connection networks and multiple network connectivity. Priority routing enables higher priority traffic to pass lower priority traffic in intermediate nodes within the HPR portions of the network. HPR also routes across connection networks or subnetwork boundaries in much the same way as APPN.

HPR routes are not given preferential treatment by the APPN routing algorithm. Existing non-HPR APPN routes will also be used if they meet the requirements of the APPN Class of Service.

- ▶ **Seamless migration:** HPR is designed for drop-in migration. A given APPN node can be upgraded to the HPR level of function without taking down the network, without configuring new parameters at its adjacent nodes, and without any logistical complications or coordination.

IP routing

One of the major functions of a network protocol such as TCP/IP is to connect together a number of disparate networks efficiently. These networks can include LANs and WANs, fast and slow, reliable and unreliable, inexpensive and expensive connections. The simplest way to connect them together is to bridge them. However, this results in every part of the network receiving all traffic and leads to slower links being overloaded and perhaps to network failure altogether. What is needed, particularly when all these networks are joined together in a worldwide network such as the Internet, is some form of intelligence at the boundaries of all these networks, which can look at the packets flowing and make rational decisions as to where and how they should be forwarded. This function is known as *IP routing*. Routing allows you to create networks that can be managed separately but are still linked and can communicate with one another.

4.16 Enterprise Extender and HPR links

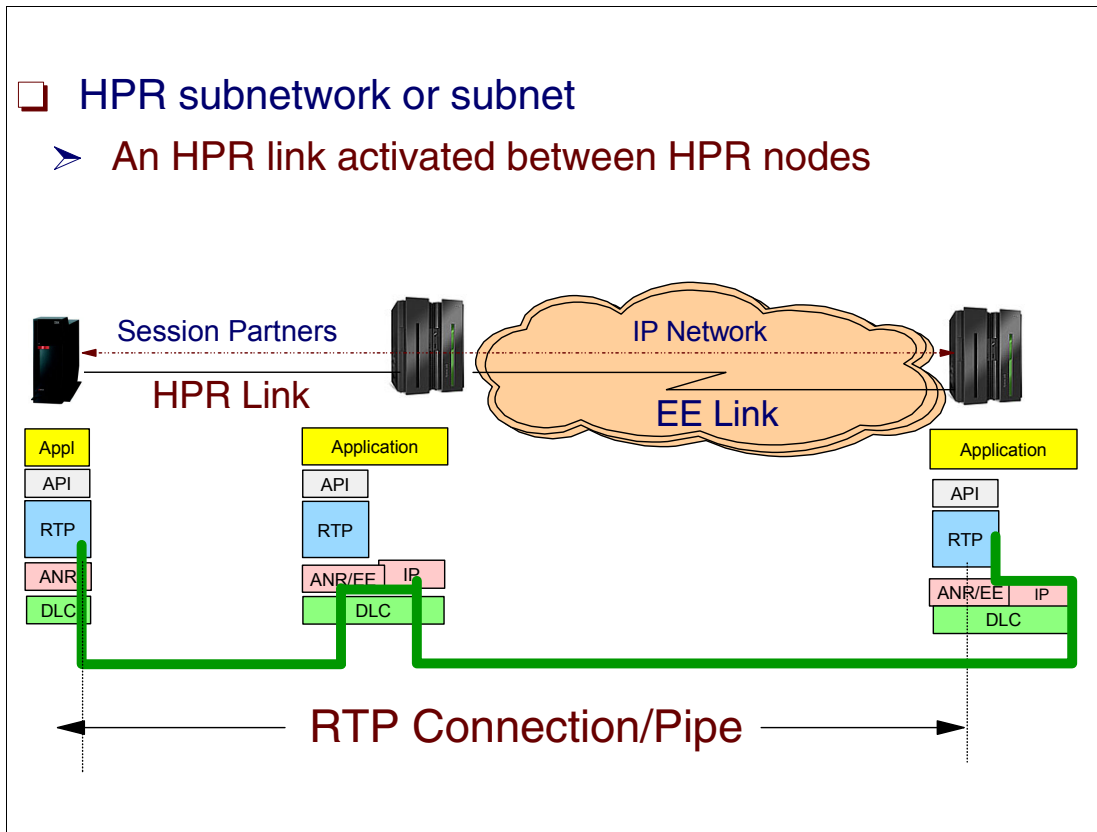


Figure 4-17 Enterprise Extender link as part of a HPR link

APPN with HPR over IP

In a mixed APPN and HPR topology network, a group of interconnected HPR nodes is sometimes referred to as an *HPR subnetwork* or an *HPR subnet*. When an HPR link is activated between a pair of adjacent HPR nodes, an HPR subnet is formed. The terms *base APPN subnetwork* and *base APPN subnet* might be used when referring to a part of the network that is not an HPR subnet.

HPR-only connections

Enterprise Extender requires *HPR-only connections*. An HPR-only connection is one in which the ANR and RTP functions are required. Therefore, an EE connection does not support nor permit just base APPN flows, because base APPN does not support the ANR and RTP functions. Figure 4-17 illustrates how Enterprise Extender utilizes HPR links.

With HPR, sessions are grouped together on logical connections called rapid transport protocol (RTP) connections. RTP is a connection-oriented, full-duplex protocol designed to transport data in high-speed networks. Sessions traversing the same route and using the same class of service can, and usually do, share an RTP connection. Data flowing on RTP connections flows as network layer packets (NLPs).

Rapid Transport Protocol (RTP)

RTP connections are established within an HPR subnet and are used to carry session traffic. These connections can be thought of as transport pipes over which sessions are carried, as shown in Figure 4-18 on page 134.

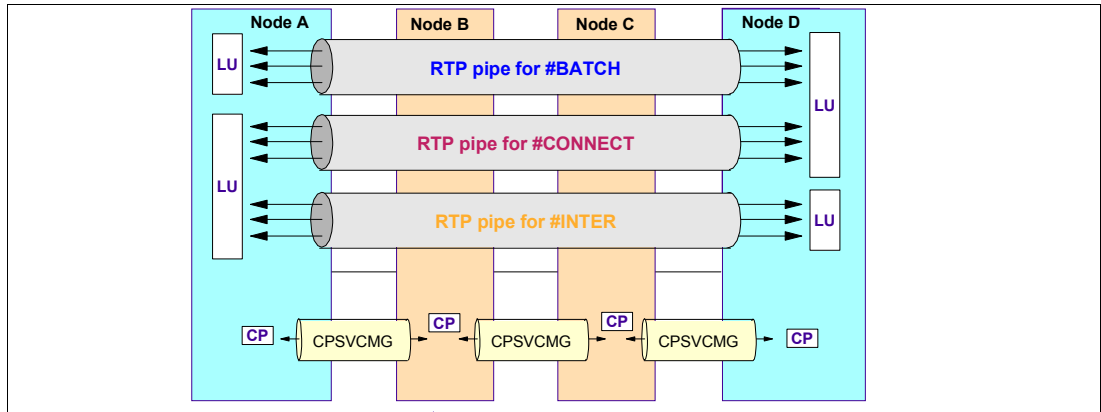


Figure 4-18 Multiple RTP pipes supporting respective classes of service

RTP protocol support

Rapid transport protocol (RTP) is a transport layer protocol that provides the following:

- ▶ Flow control
- ▶ End-to-end error recovery
- ▶ Selective retransmission of lost packets
- ▶ Nondisruptive rerouting of sessions
- ▶ Segmenting and reassembly of packets
- ▶ Translation of APPN to HPR protocols (boundary function) for sessions that have both HPR and non-HPR portions
- ▶ Translation of subarea to HPR protocols for sessions that have subarea and HPR portions

Session partners

Sessions between LU 6.2 session partners are used by transaction programs that operate in each partner LU to perform data exchange. Rather than sessions, the basic communication link between logical units using LU 6.2 protocols is a conversation. A conversation is the exclusive use of a session between two partner LUs for some unit of work. Sessions between LU 6.2 session partners are serially reusable. When one conversation ends, another conversation can use that same session between the partner LUs.



IP routing

This chapter explains how to use your TCP/IP stack for dynamic, static, or policy-based routing. The contents of this chapter are based on the assumption that you understand your entire network configuration.

When describing networks, one of the key issues is how to transport data across the network. The act of moving data traffic across a network from a source to a destination can be accomplished either by bridging or routing this data between the endpoints.

The topics in this chapter help you to:

- ▶ Understand dynamic, static, and policy-based routing
- ▶ Use the **ping** command to ping a remote host by IP address
- ▶ Use the **traceroute** command to determine the path that will be taken to reach a particular destination using static or dynamic routing
- ▶ Use the **Netstat** command to display a TCP/IP stack's routing information
- ▶ Use **DISPLAY** commands to display dynamic routing information

5.1 Network routing terminology

- IP routing terms
 - Routing
 - Static routing
 - Replaceable static routes
 - Routing daemon
 - Autonomous system
 - Router
 - Gateway
 - Interior gateway protocols (IGP)
 - Exterior gateway protocols (EGP)

Figure 5-1 Network routing terminology

Routing terminology

To route packets in the network, each network interface must have a unique IP address assigned. Whenever a packet is sent, the destination and source IP addresses are included in the packet's header information. The network layer (Layer 3) of the TCP/IP stack examines the destination IP address to determine how the packet should be forwarded. The packet is either sent to its destination on the same network (direct routing) or, based on a routing table entry, to another network using a router (indirect routing).

IP routes

The IP route is the direction sign of internets, and hence of the Internet itself. An IP route consists of simply a mapping of a destination IP address or network to a next hop and interface. The routes are collected into a routing table. Each time an IP packet needs to be sent from a host, the routing table is consulted for information about where next to send the packet.

IP routing terms

To help understand the concepts described in this section, Table 5-1 on page 137 lists some of the common IP-routing-related terms. Most of the functions or protocols listed are supported by the z/OS Communications Server.

Table 5-1 IP routing terms

Term	Definition
Routing	The process used in an IP network to deliver a datagram to the correct destination.
Static routing	Routing that is manually configured and does not change automatically in response to network topology changes.
Replaceable static routes	Static routes that can be replaced by OMPROUTE.
Dynamic routing	IP layer routing table entries that are dynamically managed and can automatically change in response to network topology changes. For IPv4, these routes are managed by a routing daemon. For IPv6, these routes can be managed by a routing daemon, and they can also be learned by listening to router advertisement messages received from routers as part of the router discovery protocol.
Routing daemon	A server process that manages the IP routing table.
Autonomous system (AS)	A group of routers exchanging routing information through a common routing protocol. A single AS can represent a large number of IP networks.
Router	A device or host that interprets protocols at the Internet Protocol (IP) layer and forwards datagrams on a path toward their correct destination.
Gateway	A router that is placed between networks or subnetworks. The term is used to represent routers between autonomous systems.
Interior gateway protocols (IGP)	Dynamic route update protocol used between dynamic routers running on TCP/IP hosts within a single autonomous system.
Exterior gateway protocols (EGP)	Dynamic route update protocols used between routers that are placed between two or more autonomous systems.

Additional routing terminology

► Main route table

An IPv4 or IPv6 route table that is populated using static routes and dynamic routes. These route tables have the name EZBMAIN. A TCP/IP stack has one main IPv4 route table and one main IPv6 route table. In the absence of policy-based routing, the IPv4 main route table contains all of the routes used by a TCP/IP stack when making IPv4 routing decisions. When policy-based routing is in use, policies can be configured to use the IPv4 main route table in IPv4 routing decisions when no route is found in a policy-based route table.

► Policy-based route table

An IPv4 route table that is configured for use by policy-based routing. A TCP/IP stack can have zero or more policy-based route tables. A policy-based route table can be defined using a flat file that is parsed by the Policy Agent, or using the IBM Configuration Assistant for z/OS Communications Server. A policy-based route table definition can contain static routes, dynamic routing parameters for controlling the scope of routes added to the table by OMPROUTE, or both. Policy rules and actions can then be defined to indicate that, for given types of traffic, the TCP/IP stack should use specific sets of route tables when making routing decisions.

5.2 MAC addresses

- ❑ Ethernet designates the frame format and the speed of the data travelling over the physical network
 - For controlling how individual hosts (workstations) attached to the physical network locate each other
 - Media access control (MAC) addresses are used
 - Every host connected to a network has a MAC address

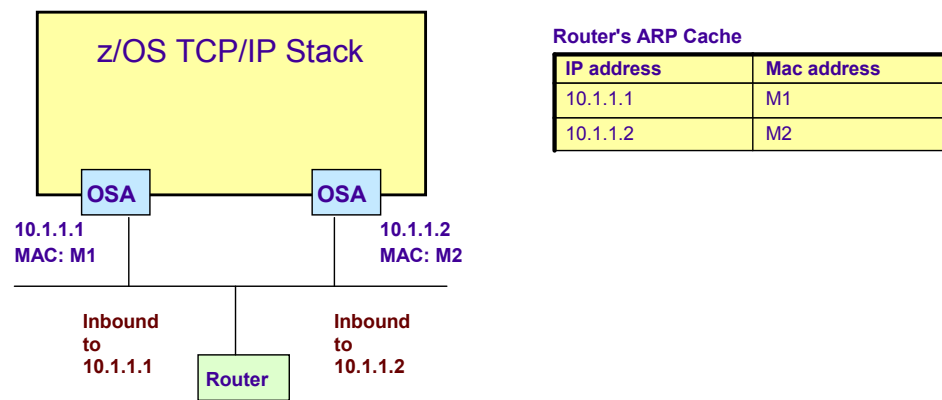


Figure 5-2 MAC addresses

MAC addresses

Ethernet designates the frame format and the speed of the data travelling over the physical network. However, there is still a need for controlling how individual hosts (workstations) attached to the physical network locate each other. The answer is the media access control (MAC) address. Every host connected to the network has a unique MAC address associated with its NIC. This MAC address, via the NIC, uniquely identifies the host.

Note: An OSA feature is a network interface card (NIC). The OSA card supports Ethernet in all of its current implementations. This means that it physically can connect to either a fiber optic cable or a copper (twisted pair) media. MAC addresses are generally built into the NIC itself, but TCP/IP on z/OS does allow MAC addresses of OSA cards to be manually altered.

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.

Address resolution protocol (ARP)

The address resolution protocol (ARP) is a layer 2 protocol used to map MAC addresses to IP addresses. All hosts on a network are located by their IP address, but NICs do not have IP addresses, they have MAC addresses. ARP is the protocol used to associate the IP address to a MAC address.

When a host wants to send a packet to another host, say IP address 10.1.1.1, on its local area network, it first broadcasts (sends out) an ARP packet. The ARP packet contains a simple question, as follows:

What is the MAC address corresponding to IP address 10.1.1.1?

The host that has been configured to use that IP address responds with an ARP packet containing its MAC address.

5.3 Static and dynamic VIPA

- ❑ Virtual IP Address (VIPA)
 - A generic term that refers to an internet address on a z/OS host that is not associated with a physical adapter
- ❑ There are two types of VIPAs:
 - A static VIPA
 - A dynamic VIPA (DVIPA)
- ❑ Distributed DVIPA
 - A special type of DVIPA, can distribute connections within a sysplex
- ❑ Dynamic routing VIPAs
 - Designed to interoperate with a dynamic routing daemon

Figure 5-3 Static and dynamic VIPA

Static and dynamic VIPA

A VIPA is a generic term that refers to an internet address on a z/OS host that is not associated with a physical adapter. There are two types of VIPAs:

- ▶ A *static* VIPA cannot be changed except through a **VARY TCPIP, ,OBEYFILE** operator command.
- ▶ A *dynamic* VIPA (DVIPA) can move to other TCP/IP stack members in a sysplex or it can be activated by an application program or by a supplied utility. Dynamic VIPAs are used to implement sysplex distributor.

Note: The virtual IP address (VIPA) removes the adapter as a single point of failure by providing an IP address that is associated with a stack without associating it with a specific physical network attachment. Because the virtual device exists only in software, it is always active and never experiences a physical failure. A VIPA has no single physical network attachment associated with it. Also, the TCP/IP stack does not maintain interface counters for VIPA interfaces (virtual links).

Distributed and dynamic routing VIPAs

A distributed DVIPA, which is a special type of DVIPA, can distribute connections within a sysplex.

Dynamic routing VIPAs are designed to interoperate with a dynamic routing daemon. Therefore, it is highly recommended that a routing daemon be used on a z/OS host that uses VIPAs.

VIPA takeover

VIPA takeover improved with the introduction of dynamic virtual IP address (DVIPA) and distributed dynamic virtual IP address (distributed DVIPA). The DVIPA function improves VIPA takeover by allowing a system programmer to plan for system outages and provide for backup systems to take over without operator intervention or external automation. The distributed DVIPA function allows the connections for a single DVIPA to be serviced by applications on several stacks listed in the configuration statement (the distribution list). This adds the benefit of limiting the scope of an application or stack failure, and also providing enhanced workload balancing.

To the routing network, a VIPA appears to be a host address indirectly attached to the z/OS. When a packet with a VIPA destination reaches the stack, the IP layer recognizes the address and passes it to the protocol layer in the stack.

The failure of the physical interface can result in failure of the TCP/IP address space, the entire z/OS, or for planned outages. A VIPA just has to move to a backup stack, and the routes to the VIPA must to be updated. Then clients can transparently connect to the backup stack.

5.4 Virtual IP Address (VIPA)

- ❑ A **static VIPA** cannot be changed except through a:
 - **VARY TCPIP,,OBEYFILE** operator command
- ❑ A **dynamic VIPA (DVIPA)**
 - Can move to other TCP/IP stack members in a sysplex or it can be activated by an application program or by a supplied utility
- ❑ Dynamic VIPAs used to implement sysplex distributor
- ❑ Distributed VIPAs
- ❑ Defining VIPAs

Figure 5-4 Virtual IP Address (VIPA)

Static VIPAs

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 **SOURCEVIP** 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. For IPv6 static VIPAs to be used as source addresses, the **SOURCEVIP** configuration option must be enabled and the VIPA interface must appear on the **SOURCEVIP** keyword on some other **INTERFACE** statement. This provides tolerance of device and adapter failures for connection requests originating at a z/OS TCP/IP stack.
- ▶ They can be specified as the source IP address for outbound TCP connection requests for all applications using this stack with **TCPSTACKSOURCEVIP**. They can be specified as the source IP address for outbound TCP connection requests for specific jobs or specific destinations 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 failed, by using **VARY TCPIP,,OBEYFILE** command processing to configure the VIPA on the backup stack and updating the routers.

Dynamic VIPA

Dynamic VIPAs 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.
- ▶ They can be moved manually by deactivating or reactivating them with the **VARY TCP/IP, ,SYSPLEX** operator command.
- ▶ They can be dynamically activated by an application program.
- ▶ They can distribute connections within a sysplex.
- ▶ They can be specified on a TCPSTACKSOURCEVIPAs statement. This allows a user to specify one VIPA to be used as the source IP address for outbound datagrams for TCP-only requests.
- ▶ They can be specified as the source IP address for outbound TCP connection requests for specific jobs or specific destinations by using the SRCIP profile statement block.
- ▶ Unlike static VIPAs, dynamic VIPAs:
 - Are limited to 1024 per stack
 - Cannot be specified as the VIPA used by Enterprise Extender for connectivity purposes

Distributed DVIPA

A distributed DVIPA, which is a special type of DVIPA, can distribute connections within a sysplex and they cannot be dynamically activated by an application program. A distributed DVIPA exists on several stacks, but is advertised outside the sysplex by only one stack. This stack receives all incoming connection requests and routes them to all the stacks in the distribution list for processing. This provides the benefit of distributing the workload of incoming requests and providing additional fail-safe precautions in the event of a server failure.

Defining VIPAs

You can distribute connections destined for a DVIPA by adding a VIPADISTRIBUTE configuration statement for a previously defined dynamic VIPA. The order of the statements is important. The VIPA is first defined with the VIPADEFINE statement and then included on a VIPADISTRIBUTE statement. Another TCP/IP address space can act as a backup for the distributed DVIPA by properly coding a VIPABACKUP statement; the backup will perform the routing function in the event of a failure. The options specified on a VIPADISTRIBUTE statement are inherited by a backup stack unless the second stack has its own VIPADISTRIBUTE statement for that DVIPA, in which case it will use its own VIPADISTRIBUTE statement for distributing. You can also code a VIPADISTRIBUTE statement with just the VIPABACKUP statement and not for the VIPADEFINE statement. This would allow workload distribution only during a primary outage.

You can change the distribution of a DVIPA after a backup stack has activated it. However, if the backup stack did not have its own distribution defined by a VIPADISTRIBUTE statement before it activated the DVIPA, any distribution changes made while the DVIPA is active on the backup stack are temporary. Those changes will be in effect while the DVIPA remains active on the backup stack, but will not be remembered if this stack takes over the DVIPA again in the future.

5.5 Virtual media access control (VMAC)

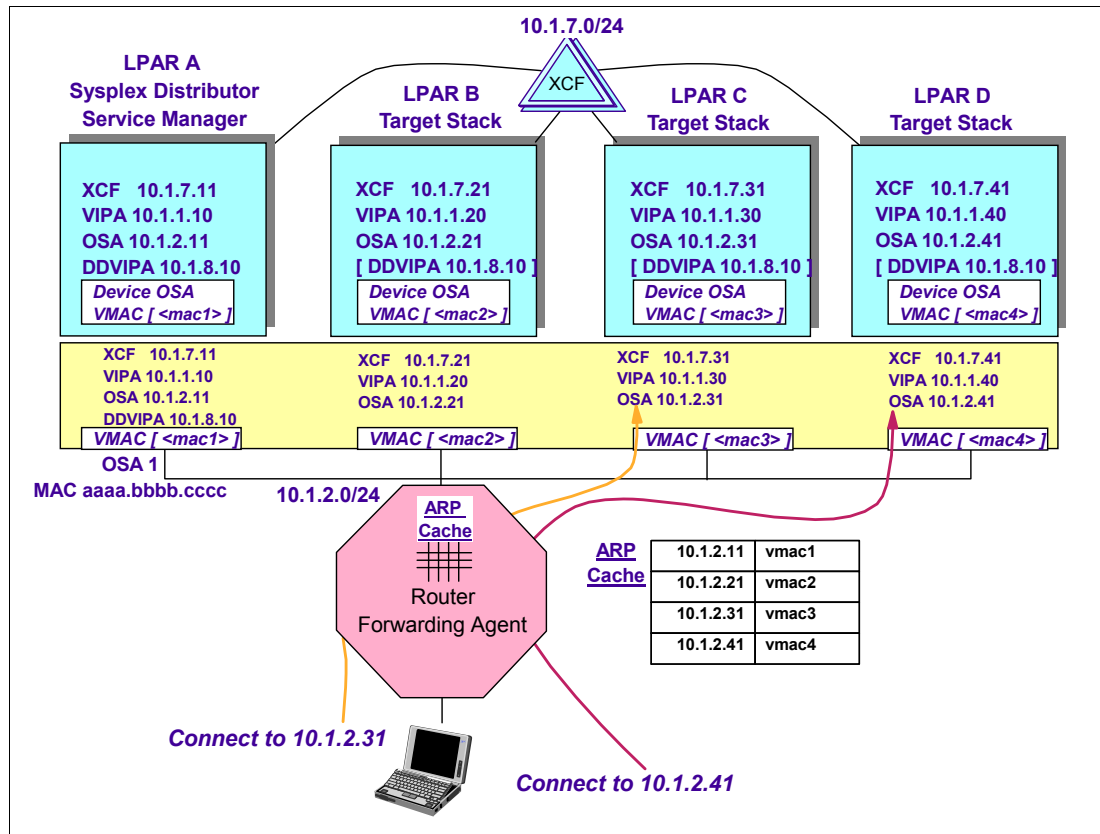


Figure 5-5 Virtual MAC (VMAC)

VMAC support

Prior to the introduction of the *virtual* MAC function, an OSA interface only had one MAC address. This restriction caused problems when using load balancing technologies in conjunction with TCP/IP stacks that share OSA interfaces. The single MAC address of the OSA also causes a problem when using TCP/IP stacks as a forwarding router for packets destined to unregistered IP addresses.

VMAC support enables an OSA interface to have not only a physical MAC address, but also many distinct virtual MAC addresses for each device or interface in a stack. That is, each stack can define up to eight VMACs per protocol (IPv4 or IPv6) for each OSA interface.

With the use of VMACs, forwarding decisions in the OSA can be made without having to involve OSI layer 3 level (network layer/IP layer). From a LAN perspective, the OSA interface with a VMAC appears as a dedicated device or interface to a TCP/IP stack. Packets destined for a TCP/IP stack are identified by an assigned VMAC address and packets sent to the LAN from the stack use the VMAC address as the source MAC address. This means that all IP addresses associated with a TCP/IP stack are accessible using their own VMAC address, instead of sharing a single physical MAC address of an OSA interface.

OSA-Express virtual MAC routing

If multiple TCP/IP instances are sharing an OSA-Express feature, the preferred method of routing is to define or generate a virtual MAC (VMAC) for each stack and for each protocol being used (IPv4 or IPv6). For IPv4, this results in the OSA-Express feature using the VMAC

address rather than the physical burned-in MAC for all ARPs sent for that TCP/IP stack's registered IP addresses, and using the VMAC as the source MAC address for all packets sent from that stack. In this way, all routers on the same LAN as the OSA-Express feature use only the VMAC address as the destination for all packets destined for that specific TCP/IP stack. From a network routing perspective, the OSA-Express feature with this VMAC appears as a dedicated device to that TCP/IP stack.

Shared OSA configuration

This simplifies a shared OSA configuration significantly. The routers on the LAN always send any packets destined for a particular TCP/IP stack to the VMAC defined for that stack. The OSA-Express feature knows by VMAC address exactly which stack should receive a given packet. Even if the IP address is not registered with the OSA-Express feature, if the packet is destined for that VMAC, the router has determined which stack should be the intermediate router, and the OSA can forward the packet directly to that stack. If the stack is not an intermediate router, the capability is provided for a stack to indicate to the OSA that it wants to receive that stack.

Sysplex distributor

Optimized connection load balancing using sysplex distributor in a network with Cisco routers is available in IPv4 only. The IBM sysplex distributor function provides a workload balancing function within a parallel sysplex. The sysplex distributor consists of a primary distributor stack (denoted by a dynamic VIPA) and a set of target stacks. An inbound packet destined for that DVIPA flows through the primary distributor stack, which then forwards the packet over an internal link (XCF, IUTSAMEH, or HiperSockets) to the selected target stack.

The Cisco Multi-Node Load Balancer (MNLB) provides a workload balancing function which distributes traffic through Cisco routers across multiple destination TCP/IP stacks. The MNLB consists of a service manager (the Cisco local director, denoted by a cluster IP address) and a set of forwarding agents (Cisco routers). For a TCP connection to the cluster IP address, the forwarding agent sends the SYN packet 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.

5.6 Forwarding packets to VMAC targets

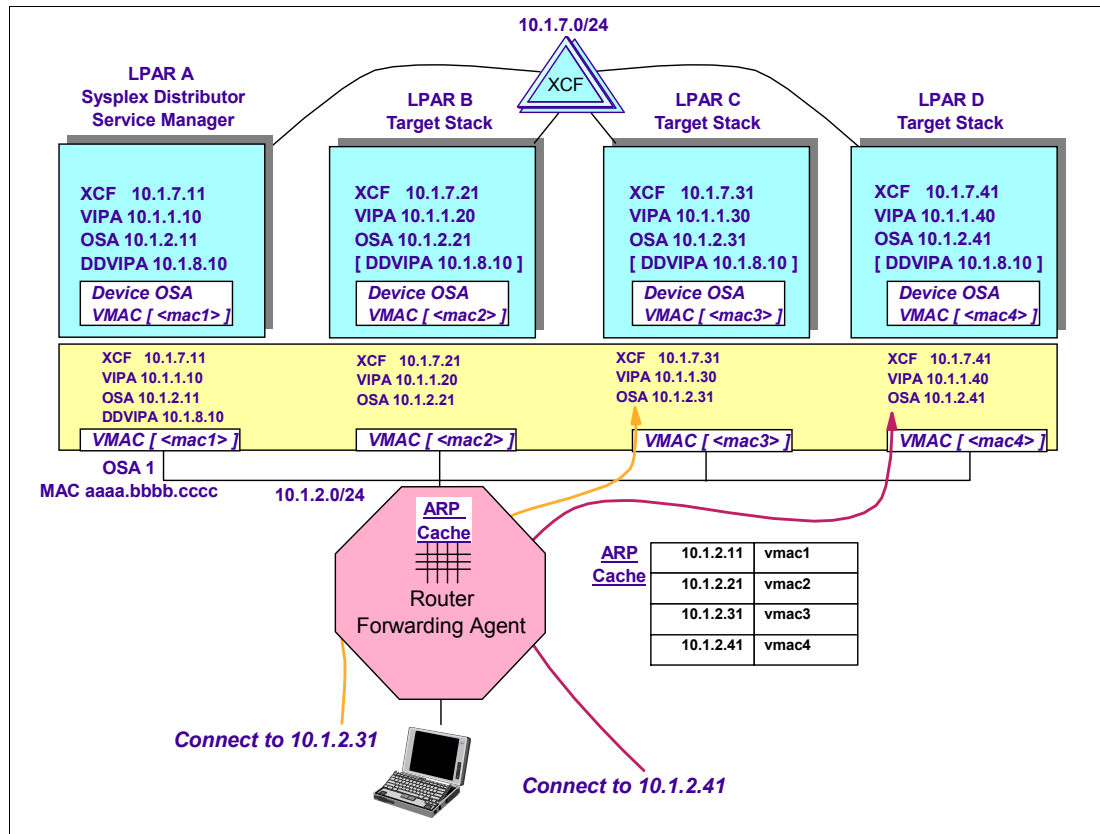


Figure 5-6 Forwarding packets to VMAC targets

Forwarding packets to VMAC targets

Figure 5-6 depicts how the definition of VMACs in the TCP/IP stacks gives the appearance of having a dedicated OSA interface on each stack. When packets arrive at the shared OSA interface, the individual VMAC assignments allow the packets to be forwarded directly to the correct stack. In the example shown, no individual stack needs to be defined as a primary or secondary router, thus offloading this function from a TCP/IP stack.

This simplifies a shared OSA configuration significantly. Defining VMACs has very little administrative overhead. It is also an alternative to GRE or NAT when load balancing technologies are used. In Figure 5-6, the Dynamic VIPA targets are found without the use of GRE and without routing through the sysplex distributor. One of the options for defining VMACs permits the OSA to bypass IP address lookup. As a result, when the packet arrives at the correct VMAC, it is routed to the stack even though the ODVIPA is not registered in the OAT.

For IPV6, TCP/IP uses the VMAC address for all neighbor discovery address resolution flows for that stack's IP addresses, and likewise uses the VMAC as the source MAC address for all IPv6 packets sent from that stack. Again, from a LAN perspective, the OSA interface with a VMAC appears as a dedicated device to that stack.

Note: VMAC definitions on a device in a TCP/IP stack override any NONRouter, PRIRouter, or SECRouter parameters on devices in a TCP/IP stack. If necessary, selected stacks on a shared OSA can define the device with VMAC and others can define the device with PRIRouter and SECRouter capability.

Virtual MAC address assignment

The VMAC address can be defined in the stack or it can be generated by the OSA. If generated by the OSA, it is guaranteed to be unique from all other physical MAC addresses and from all other VMAC addresses generated by any OSA-Express feature.

Note: We recommend letting the OSA generate the VMACs instead of assigning an address in the TCP/IP profile. If VMACs are defined in the LINK statement, they must be defined as locally administered MAC addresses, and should be unique to the LAN on which they reside.

The same VMAC can be defined for both IPv4 and IPv6 usage, or a stack can use one VMAC for IPv4 and one for IPv6. Also, a VLAN ID can be associated with an OSA-Express device or interface defined with a VMAC.

Target stack

It is important to note that the z/OS Communications Server environment consists of the TCP/IP address space, z/OS Communications Server applications, and the TCP/IP MVS applications. The TCP/IP address space functions are also referred to as the *stack*, as shown in Figure 5-6. The z/OS Communications Server applications refer to those applications using the z/OS UNIX socket API. The TCP/IP MVS applications refer to those applications written to the MVS APIs (for example, C, Sockets-Extended, CICS, IMS, and REXX). The TCP/IP stack and both sets of applications have some common (or global) configuration files, but they also use configuration files that are different.

5.7 Routing table (static or dynamic)

- ❑ Static routing
 - Manually configure routing tables
 - During customization of TCP/IP
 - Use when network environment is small
- ❑ Dynamic routing
 - Routing table is built dynamically
 - Automatically exchanges route information
 - Enables routers to always use the best path
- ❑ Routing information protocol (RIP)
- ❑ Open Shortest Path First (OSPF)

Figure 5-7 Static and dynamic routing

Static routing

Static routing requires you to *manually* configure the routing tables yourself. This task is part of the configuration steps you follow when customizing TCP/IP. It implies that you know the address of every network you want to communicate with and how to get there. That is, you must know the address of the first router on the way.

The task of statically defining all necessary routes can be simple for a small network. It offers the advantage of avoiding the network traffic overhead of a dynamic route update protocol. It also allows you to enforce rigid control over the allocation of addresses and resource access. However, it will require manual reconfiguration if you move or add a resource.

The drawback of static routing is that, even if a network failure occurs in the intermediate path to the destination, the routing table remains unchanged and keeps sending the packet according to the statically defined next hop routers. Sometimes it might cause the network to be unreachable. Also, if you fail to define the right next hop router in the route entry, the routers keep forwarding the packet using that entry. Even if there is a better route, the router does not change its next hop router until the changes are made to the static route entry.

If your network environment is small and manageable, with few to no network changes anticipated, then using static routes is an option (keeping in mind that your z/OS system is basically an application server environment). A good practice is to define only the default gateways to the exterior networks, and let the routers do the exterior routing. You can implement static routing between the z/OS system and external router, and still let the external routers use the dynamic routing protocol to exchange route information.

Dynamic routing

Dynamic routing removes the need for static definition of the routing table. The network routing table is built dynamically, automatically exchanging route information among the routers in the network. This sharing of the routing information enables the routers to always calculate the best path through the network to any destination. When a network outage occurs in the intermediate route to the destination, the routers exchange information about the outage and the best path is recalculated. If your routing tables are complex due to network growth, or if the system must act as a gateway, it is far easier to let the system do the work for you by using dynamic routing. The drawback of dynamic routing is the burden of route information exchange.

Dynamic routing protocols

Dynamic routing protocols can be divided into two types: interior gateway protocols, and exterior gateway protocols.

- ▶ Interior gateway protocols (IGPs) are dynamic route update protocols used between dynamic routers running on TCP/IP hosts within a single autonomous system. These protocols are used by the routers to exchange information about which IP routes the IP hosts have knowledge of. By exchanging IP routing information with each other, the routers are able to maintain a complete picture of all available routes inside an autonomous system.
- ▶ Exterior gateway protocols (EGPs) are dynamic route update protocols that are used between routers that are placed between two or more autonomous systems.

OSPF and RIP

RIP protocol comes in two versions, RIPv1 and RIPv2. Both protocols require a server process (called a *daemon*) to be running on the host. This daemon communicates with other hosts running an RIP daemon in the network. Information about the routing tables of each daemon host is exchanged periodically. Routing tables are built based upon information about the network supplied from other routers. The advantage here is that if a network changes, for any reason, the exchange of information among routers allows this change to be communicated. The drawback of RIP is that the routing tables become large very quickly. A large network can require huge routing tables. And, RIP can be slow in recognizing changes in the network. The recognition of changes in a network by a dynamic routing protocol, which produces the best routes over time, is referred to as *convergence*. Most z/OS networks are moving away from using RIPv2 and are instead utilizing OSPF as the dynamic routing protocol. RIPv2 is defined in RFC 2453.

Open Shortest Path First (OSPF)

OSPF effectively accomplishes the same thing as RIP does: it populates the routing table of a host with routes. It essentially has all the capabilities of RIPv2. However, OSPF is more scalable and configurable than RIP. In addition, OSPF supports the organization of networks into areas. These areas can be used to limit the amount of information that must be moved around an entire internet, yet there is no compromise of connectivity. From a network route management perspective, OSPF differs significantly from RIP. OSPF exchanges information on the state of links (interfaces) instead of routing information. Link state changes are immediately reported (using a Link State Advertisement). Consequently, network convergence is fast and consistent. In addition, hosts participating in OSPF routing are assigned specific roles (for example, a designated router or an area border router). The protocol itself is a state-oriented protocol. Interfaces and neighboring routers are always classified as being in a particular state. The routing daemon on z/OS (called OMPROUTE) is capable of handling both OSPF and RIP interfaces concurrently. OSPF is one of the most widely implemented routing protocols. It is defined in RFC 2328.

5.8 Routing IP datagrams

- Three types of routing entries
 - Direct routes - Indirect routes - Default route

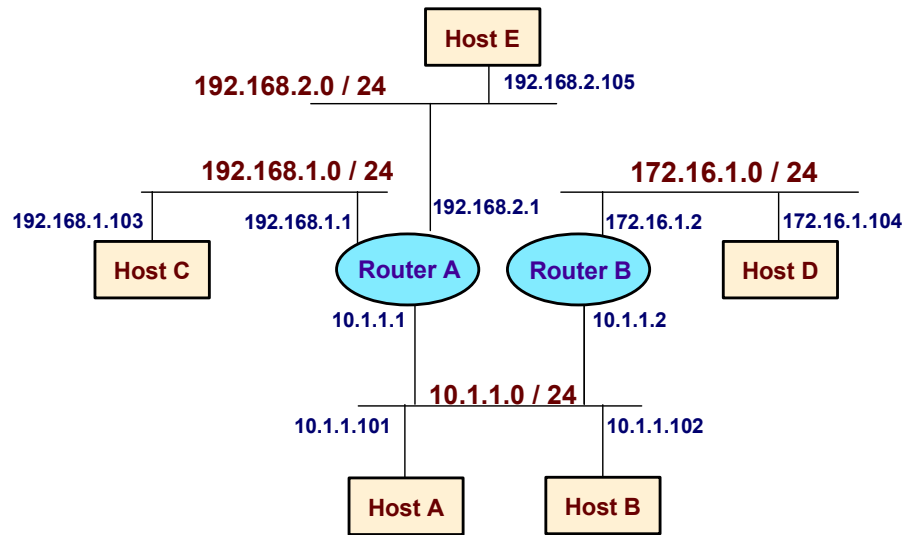


Figure 5-8 Routing IP datagrams (Sample network with multiple subnetworks)

Direct routes, indirect routes, and default route

Every IP host is capable of routing IP datagrams and maintaining an IP routing table. There are three types of entries in an IP routing table:

► Direct routes

The networks to which the host is directly attached are called direct routes. If the destination host is attached to the same physical network as the source host, IP datagrams can be directly exchanged. This is done by encapsulating the IP datagram in the physical network frame.

► Indirect routes

The networks to which the host is not directly attached and reachable through one or more IP routers are called indirect routes. When the destination host is not connected to a network directly attached to the source host, the only way to reach the destination is through one or more IP routers. The routing entry with destination IP address and the IP address of the first router (the next hop) is called an indirect route in the IP routing algorithm.

The IP address of the first router is the only information needed by the source host to send a packet to the destination host. If the source and destination hosts are on the same physical network, but defined in different subnetworks, indirect routing is used to communicate between the endpoints. A router is needed to forward packets between subnetworks.

► **The default route**

The default route entry contains the IP address of the first router (the next hop) to be used when the destination IP address or network is not found in any of the direct or indirect routes.

Sample network example

Figure 5-8 on page 150 shows hosts and routers located in multiple networks. To achieve connectivity between these hosts, the routers are connected to multiple networks, creating a path between them. In this scenario, if host A wants to connect to host D, both resources must create and maintain a routing table to define which path has to be used to reach the destination. Host A in this example might contain the (symbolic) entries shown in Table 5-2.

Table 5-2 IP routing table for host A

Destination	IP address of Next Hop Router
10.1.1.0/24	Directly connected
192.168.1.0/24	10.1.1.1 (Router A)
172.16.1.0/24	10.1.1.2 (Router B)
Default	10.1.1.1 (Router A)
127.0.0.1	Loopback
The /24 notation represents the length of subnet mask (a 24-bit mask, in this case).	

The routing table contains routes to different routers in this network. When host A has an IP datagram to forward, it determines which IP address to forward it to using the IP routing algorithm and the routing table.

Because Host A is directly attached to network 10.1.1.0/24, it maintains a direct route for this network. To reach other networks, such as 192.168.1.0/24 and 172.16.1.0/24, it must have an indirect route through router A and router B, respectively, because these networks are not directly attached to it. Another option is to define a default route. If the indirect route to the network is not defined explicitly, the default route is used.

In this example, Host A reaches to Host B using the direct route. To reach Host C (192.168.1.103), it uses the indirect route to 192.168.1.0/24 and forwards the packet to Router A (10.1.1.1).

Likewise, to reach Host D, it uses the indirect route to 172.16.1.0/24 and forwards the packet to Router B (10.1.1.2). The indirect route to Host E (192.168.2.105) is not explicitly defined in Host A. So, the default route is used and Host A forwards the packet to Router A (10.1.1.1).

To reach any given IP network address, each host or router in the network needs to know only the next hop's IP address and not the full network topology.

5.9 Dynamic routing using OMPROUTE

- ❑ OSPF protocol is implemented by OMPROUTE
 - The OMPROUTE server is a z/OS UNIX daemon
 - OMPROUTE is to be started by a procedure
 - OMPROUTE is compatible with adjacent routers that are capable of handling OSPF communications
- ❑ OMPROUTE is a fully functional OSPF router
 - The OMPROUTE subagent implements the Open Shortest Path First (OSPF) MIB variable containing OSPF protocol and state information
 - The OMPROUTE subagent supports selected MIB objects defined in RFC 1850
 - OMPROUTE supports Virtual IP Addressing (VIPA)

Figure 5-9 Dynamic routing using OMPROUTE

Dynamic routing using OMPROUTE

For IPv4, OMPROUTE supports RIPv1, RIPv2, and OSPF routing protocols. You can send RIPv1 or RIPv2, but not both at the same time on a single interface. However, you can configure a RIP interface to receive both versions. It provides an alternative to the static TCP/IP gateway definitions. The MVS host running with OMPROUTE becomes an active OSPF or RIP router in a TCP/IP network. Either or both of these routing protocols can be used to dynamically maintain the host routing table. OMPROUTE must build a routing table for TCP/IP for network communication to occur. TCP/IP can be configured to ensure that it does not join (or rejoin) the sysplex until OMPROUTE itself has been started. This reduces the possibility of a connection being routed to a host that does not yet have connectivity to the network.

OMPROUTE server is a z/OS UNIX daemon

If OMPROUTE is to be started by a procedure, update the cataloged procedure OMPROUTE by copying the sample in SEZAINST(OMPROUTE) to your system or recognized PROCLIB. Specify OMPROUTE parameters and change the data set names to suit your local configuration. You can start OMPROUTE from the MVS operators console by starting the OMPROUTE start procedure.

Note: For z/OS Communications Server IP, there is a multiprotocol routing daemon available. *Daemon* is a UNIX term for a background server process, and daemons are used for dynamic routing.

OMPROUTE is a fully functional OSPF router

OMPROUTE installs the routes that are learned dynamically through other routers with routing protocol (OSPF or RIP) to the TCP/IP stack's routing table. When routing a packet to its destination, the TCP/IP stack makes decisions for route selection based on the TCP/IP stack's routing table, not the OMPROUTE routing table. A one-to-one relationship exists between an OMPROUTE and a TCP/IP stack. OSPF/RIP support for multiple TCP/IP stacks requires multiple instances of OMPROUTE. The affinity to the TCP/IP stack is made by specifying the TCPIPJobname statement with the TCP/IP stack name in the TCPIP.DATA file that OMPROUTE uses.

OMPROUTE supports VIPA

OMPROUTE supports VIPA to handle network interface failures by switching to alternative paths. VIPA routes are included in the OSPF and RIP advertisements to adjacent routers, which can then use them to reach destinations in the z/OS.

5.10 Sysplex distributor

- ❑ Sysplex distributor is a combination of:
 - High availability features of DVIPA
 - Workload optimization capabilities of WLM
- ❑ Sysplex distributor roles
 - LPARs in the sysplex are assigned roles
 - A TCP/IP stack in one LPAR is given the role of being a front-end host
 - It receives inbound connection requests and redirects them to a specific target back-end TCP/IP stack
 - Other LPARs run target TCP/IP stacks that ultimately function as the real endpoint of communication
- ❑ Requires dynamic XCF

Figure 5-10 Sysplex distributor

Sysplex distributor

Sysplex distributor extends the notion of dynamic VIPA and automatic VIPA takeover to allow for load distribution among target servers within the sysplex. It extends the capabilities of dynamic VIPAs to enable distribution of incoming TCP connections to ensure high availability of a particular service within the sysplex.

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 through 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 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 can result from some sort of failure). This takeover is nondisruptive to all existing connections.

High availability features of DVIPA

The functionality of sysplex distributor is that one IP entity advertises ownership of an IP address by which a particular service is known. In this fashion, the single system image of sysplex distributor is that of a special IP address. This IP address is called a *distributed DVIPA*. Further, in sysplex distributor, the IP entity advertising the distributed DVIPA and dispatching connections destined for it is itself a system image within the sysplex, referred to as the *distributing stack*.

Workload optimization with WLM

Sysplex distributor makes use of Workload Manager (WLM) and its ability to gauge server load and provide a WLM recommendation. In this paradigm, WLM provides the distributing stack with a WLM recommendation for each target system (a WLM system weight), or the target stacks provide the distributing stack with a WLM recommendation for each target server (a WLM server-specific weight). The distributing stack uses this information to optimally distribute incoming connection requests among a set of available servers. Additionally, sysplex distributor has the ability to specify certain policies within the Policy Agent so that it can use QoS information from target stacks to further modify the WLM recommendation. Further, these policies can specify which target stacks are candidates for clients in particular subnetworks.

Sysplex distributor roles

With sysplex distributor, the LPARs in the sysplex are assigned roles. A TCP/IP stack in one LPAR is given the role of being a front-end host. It receives inbound connection requests and redirects them to a specific target back-end TCP/IP stack. Other LPARs run target TCP/IP stacks that ultimately function as the real endpoint of communication.

Sysplex distributor requires that a choice be made as to whether a host is to function as a distributor, a target, a backup distributor, or some combination, as follows:

Distributing host	The designated contact (point of entry) for the sysplex. It is the normal owner of the IP address that clients out in the network use to connect to the sysplex.
Target host	A host within the sysplex to which a distributing host can redirect a connection request. The target host must be running an instance of the target application. For example, if a client wants to connect to FTP, then there must be an FTP server running on the target host to which the session is distributed.
Backup host	A host that is designated as a backup in the event that the distributing host should malfunction. The backup host takes over the IP address of the distributing host when required. There can be more than one backup host.
Combinations	A distributing host can also be a target host, meaning some sessions are distributed to itself. A target host can also be a backup distributing host. The idea here is to not waste an LPAR just because it is assigned a certain role.

Dynamic XCF

When dynamic XCF is functional within a sysplex, a point-to-multipoint network is established among all participating LPARs. Each host in the sysplex has a direct connection to any other host in the same sysplex.

Within the TCP/IP profile data set, there is only one configuration option required, DYNAMICXCF. This option falls within the IPCONFIG statement group.

5.11 Sysplex distributor workload balancing

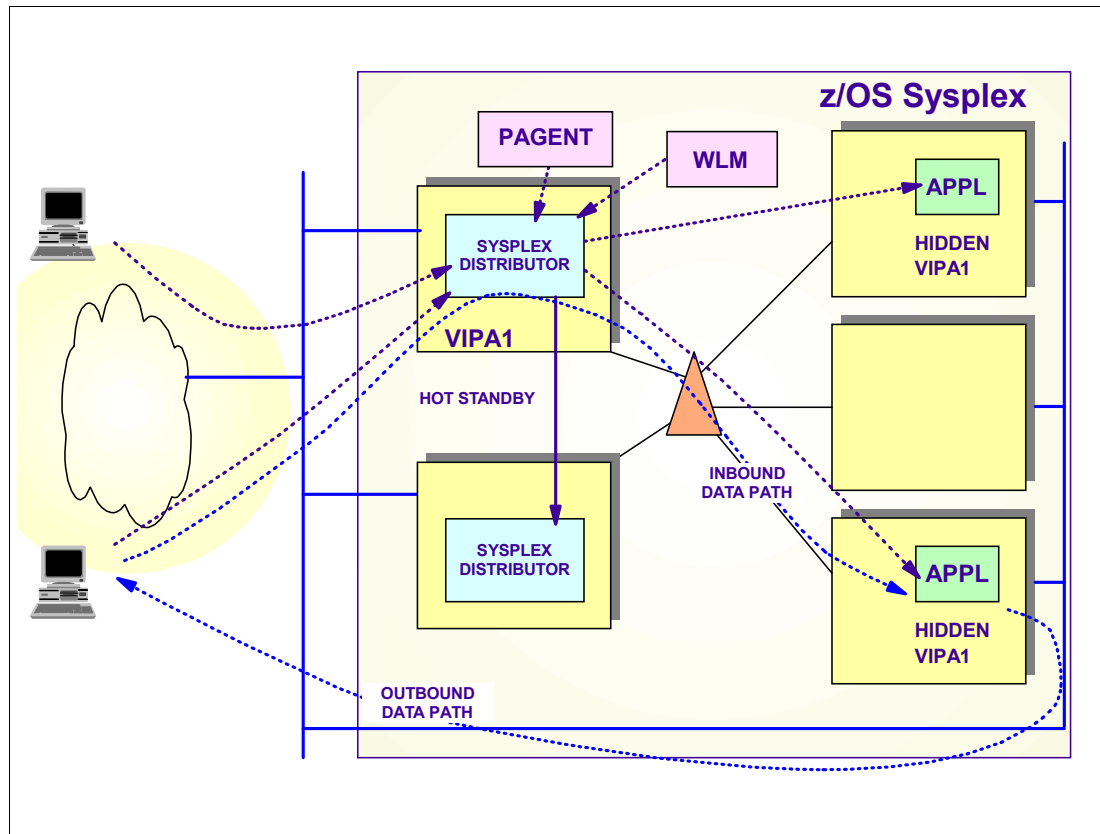


Figure 5-11 Sysplex distributor for z/OS-integrated intra-sysplex workload balancing

Policy Agent

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 these distinct functions to assist the sysplex distributor:

- ▶ Policies can be defined to control which stacks the sysplex distributor can select. The definition of the *outbound interface* on the PolicyAction statement can limit the potential target stacks to a subset of those defined on the VIPADISTRIBUTE statement in the TCPIP.PROFILE. The stacks to which work is distributed can vary (for example, based on time periods in the policy). Another possibility is to limit the number of the sysplex distributor target stacks for inbound traffic from a given subnet. In this way, the total set of target stacks can be partitioned among different groups of users or applications requesting connections to distributed applications.
- ▶ The PolicyPerfMonitorForSDR statement in the pagent.conf file activates the Policy Agent QoS performance monitor function. When activated, the Policy Agent uses data about packet loss and timeouts that exceed defined thresholds and derives a QoS weight fraction for that target stack. This weight fraction is then used to reduce the WLM weight assigned to the target stacks, so that the sysplex distributor stack can use this information to better direct workload to where network traffic is best being handled. In this way, processor performance, server performance, and application network performance are taken into account when a decision is made for work distribution. This policy is activated on the sysplex distributor and the target stacks.

- ▶ The Policy Agent at each target now collects information with an additional level of granularity, and the QoS performance data is collected for each service level that a target's DVIPA port or application supports.
- ▶ The Policy Agent on the distributing stack drives the collection of this information by pulling it from the Policy Agents on the target stacks.

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



z/OS UNIX, FTP, and security

The Communications Server protects data in the network by supporting a variety of cryptographic-based network security protocols, such as IPSec, SSL, and SNA Session Level Encryption. These security protocols ensure that data received originated from the claimed sender (data origin authentication), that contents were unchanged in transit (message integrity), and that sensitive data is concealed using encryption (data privacy).

The following security features are incorporated into various Communications Server components:

- ▶ Application security
- ▶ TCP/IP resource protection
- ▶ Network security principles
- ▶ Network security protocols
- ▶ Security event reporting: Integrated Intrusion Detection Services
- ▶ Defensive filtering
- ▶ Network security services for the IPSec discipline
- ▶ Network security services for the XMLAppliance discipline

This chapter describes some security considerations that have a product-wide effect. Descriptions of security considerations that affect specific servers or components are included with the information for each server and component.

For z/OS UNIX, the security considerations are:

- ▶ Requirement for an OMVS segment
- ▶ Authorization of TCP/IP started task user ID
- ▶ Other user IDs requiring z/OS UNIX superuser authority
- ▶ BPX.DAEMON FACILITY class profile
- ▶ Program control

6.1 z/OS UNIX OMVS segment

- ❑ z/OS UNIX users require an OMVS segment
 - Identify all the users in your environment that use TCP/IP services
 - Then define OMVS RACF segments for the associated user IDs

User profile

Userid	Default Group	Connect Groups		TSO	DFP	OMVS		
						UID	Home	Program
SMITH	PROG1	PROG1	PROG2	15	/u/smith	/bin/sh

Figure 6-1 z/OS UNIX OMVS segment

Application security

The Communications Server protects data and other system resources accessed by applications included in the Communications Server element. This protection requires verification of the identity of the end user requesting access. This process is called identification and authentication. In addition, access to resources must be limited to those users with permission. This process is called access control. Communications Server applications use Resource Access Control Facility (RACF) for identification and authentication, and access control decisions. Authenticated users are granted access to only those RACF resources for which they have permission.

RACF initially identifies and authenticates users from a user ID and password when they log on to the system. When a user tries to access a resource, RACF checks its database. Then, based upon the information in the database, RACF either allows or denies the access request. It displays an ICH408I message if the access is denied.

z/OS UNIX OMVS segment

Many TCP/IP Services components in z/OS Communications Server now exploit z/OS UNIX services in both the native MVS environment and in the z/OS UNIX environment. For example, all TCP/IP socket APIs and TCP/IP applications (whether they are provided by z/OS Communications Server, z/OS, other IBM and non-IBM products, or written by users) now make use of z/OS UNIX services.

Use of z/OS UNIX services requires a z/OS UNIX security context, referred to as an OMVS segment, for the user ID associated with any unit of work requesting these services. In other words, most user IDs requiring access to TCP/IP functions now require an OMVS segment to be defined in RACF.

Note: The TCP/IP address space operates as a transport provider for the INET physical file system. For this to occur, the TCP/IP system address space must connect to z/OS UNIX and become a z/OS UNIX process. Therefore, the started task UID that is assigned to the TCP/IP system address space must have a valid OMVS segment.

To satisfy the requirement for an OMVS segment in RACF, do one of the following:

- ▶ Identify all the users in your environment that use TCP/IP services and then define OMVS RACF segments for the associated user IDs.
- ▶ Use the default OMVS segment support provided by RACF and z/OS UNIX for users and groups.

Note: If you are using a security product from another vendor, read the documentation for that product for instructions on task performance.

6.2 TCP/IP started task user ID

- ❑ The TCP/IP system address space must connect to z/OS UNIX and become a z/OS UNIX process
 - The TCP/IP system address space must have a valid OMVS segment
 - As a transport provider, the TCP/IP address space requires superuser privileges in z/OS UNIX
 - Define the TCP/IP system address space started task UID as UID=0 or
 - Define the TCP/IP system address space as a trusted environment in the RACF started class profile

```
ALU tcpip_userid OMVS(UID(0) HOME(/) PGM(/bin/sh))
```

Figure 6-2 TCP/IP started task user ID

TCP/IP started task user ID

The TCP/IP address space operates as a transport provider for the INET physical file system. For this to occur, the TCP/IP system address space must connect to z/OS UNIX and become a z/OS UNIX process. Therefore, the started task UID that is assigned to the TCP/IP system address space must have a valid OMVS segment.

As a transport provider, the TCP/IP address space requires superuser privileges in z/OS UNIX. Define the TCP/IP system address space started task UID as UID=0, or define the TCP/IP system address space as a trusted environment in the RACF started class profile for the TCP/IP system address space. Use the following command to assign an OMVS segment to the TCP/IP started task user ID specified as UID=0.

Defining superuser authority

You can assign superuser authority in three ways:

- ▶ Using resource profiles in the UNIXPRIV class (preferred method).
- ▶ Using the BPX.SUPERUSER resources in the FACILITY class.
- ▶ Assigning a UID of 0 (least desirable method).

You might choose to assign a UID of 0 to multiple RACF user IDs. However, you should minimize the number of users you assign the UID of 0 because a user with a UID of 0 can perform any z/OS UNIX function and passes all z/OS UNIX security checks.

6.3 User IDs requiring superuser authority

- ❑ When a started procedure is used to start the following servers
 - Daemons and agents - the user must be a superuser [UID(0)] or permitted to a BPX.SUPERUSER profile
- ❑ Using the BPX.SUPERUSER resource in the FACILITY class
 - Allows users to get the authority to do most of the tasks that require superuser authority

```
ALTUSER SYSPROG OMVS(UID(7) HOME('/u/sysprog') PROGRAM('/bin/sh'))  
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)  
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(SYSPROG) ACCESS(READ)
```

Figure 6-3 User IDs requiring superuser authority

User IDs requiring superuser authority

Using the BPX.SUPERUSER resource in the FACILITY class is another way for users to get the authority to do most of the tasks that require superuser authority.

Perform the following steps to set up BPX.SUPERUSER:

- ▶ Alter an existing user (sysprog) to have an OMVS segment.

```
ALTUSER SYSPROG OMVS(UID(7) HOME('/u/sysprog') PROGRAM('/bin/sh'))
```
- ▶ Define the BPX.SUPERUSER resource in the FACILITY class.

```
RDEFINE FACILITY BPX.SUPERUSER UACC(NONE)
```
- ▶ Permit all users who need superuser authority to this RDEFINE profile. Use the RACF command shown in the following example, which gives the user ID SYSPROG permission to use the **su** command to obtain superuser authority. It is assumed that the default group for SYSPROG is set up with a GID.

```
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(SYSPROG) ACCESS(READ)
```

Note: You must use the name BPX.SUPERUSER. Substitutions for the name are not allowed.

6.4 BPX.DAEMON in the FACILITY class

- ❑ If the BPX.DAEMON resource in the FACILITY class is not defined, your system has UNIX-level security
 - In this case, the system is less secure
- ❑ If the BPX.DAEMON resource profile in the FACILITY class is defined, your system has z/OS UNIX security
- ❑ Security requirements
- ❑ Communications Server security requirements
- ❑ BPX.DAEMON and setuid()

Figure 6-4 BPX.DAEMON in the FACILITY class

BPX.DAEMON in the FACILITY class

If the BPX.DAEMON resource in the FACILITY class is not defined, your system has UNIX-level security. In this case, the system is less secure. If the BPX.DAEMON resource profile in the FACILITY class is defined, your system has z/OS UNIX security. Your system can exercise more control over your superusers.

Security requirements

This level of security is for customers with stricter security requirements who need to have some superusers maintaining the file system, but who want to have greater control over the z/OS resources that these users can access. Although BPX.DAEMON provides some additional control over the capabilities of a superuser, a superuser should still be regarded as a privileged user because of the full range of privileges the superuser is granted.

Communications Server security requirements

Certain z/OS Communications Server TCP/IP Services servers need to change the security environment of the process in which they currently execute. For example, the FTPD daemon creates a new z/OS UNIX process for every FTP client connecting to it. After the new process is created, the daemon changes the security environment of the process so that it is associated with the security context of the logged-in user. The RACF FACILITY class resource BPX.DAEMON is used for this purpose.

BPX.DAEMON and setuid()

The additional control that BPX.DAEMON provides involves the use of kernel services such as **setuid()**, which changes a caller's z/OS user identity. Any user can issue a **setuid()** which follows a successful **__passwd()** call to the same target user ID. However, a user with daemon authority can issue **setuid()** without knowing the target user's password or password phrase. With BPX.DAEMON defined, a superuser process can run these types of change services and identity whether the following statements are true:

- ▶ The caller's user identity was permitted to BPX.DAEMON.
- ▶ All programs running in the address space have been loaded from a library that is controlled by a security product.

6.5 RACF program control

- ❑ Any program loaded into an A/S with daemon authority
 - **Must be a controlled program**
- ❑ Define programs to Program Control
- ❑ z/OS daemons reside in the HFS and are controlled
- ❑ User defined daemons
 - **Specific daemon program names or ***

```
RDEFINE PROGRAM * UACC(READ) ADDMEM +
('SYS1.LINKLIB'/'*****'/NOPADCHK +
'CEE.SCEERUN'/RLTPAK/NOPADCHK +
'SYS1.SEZALOAD'//NOPADCHK )
SETROPTS WHEN(PROGRAM) REFRESH
```

Figure 6-5 Defining RACF program control

RACF program control for daemons

The purpose of protecting load modules is to provide installations with the ability to control who can execute what programs and to treat those programs as assets.

You protect individual load modules (programs) by creating a profile for the program in the PROGRAM general resource class. A program protected profile in the PROGRAM class is called a controlled program.

The name of the profile can be complete, in which case the profile protects only one program, or the name of the profile can end with an asterisk (*), in which case the profile can protect more than one program. For example, a profile named ABC* protects all programs that begin with ABC, unless a more specific profile exists.

The profile for a controlled program must also include the name of the program library that contains the program and the volume serial number of the volume containing the program library. The profile can also contain a standard access list of users and groups and their associated access authorities.

Program access to data sets

Program access to data sets (PADS) allows an authorized user or group of users to access specified data sets with the user's authority to execute a certain program. That is, some users can access specified data sets at a specified access level only while executing a certain program (and the program access is restricted to controlled programs).

To set up program access to data sets, create a conditional access list for the data set profile protecting the data sets. To do this, specify `WHEN(PROGRAM(program-name))` with the `ID` and `ACCESS` operands on the `PERMIT` command. Specifying the `WHEN(PROGRAM)` operand requires that the user or group specified must be running the specified program to receive the specified access.

PADCHK and NOPADCHK operands

Choosing between the `PADCHK` and `NOPADCHK` operands: With the `ADDMEM` operand of the `RDEFINE` and `RALTER` commands, you can also specify `PADCHK` or `NOPADCHK` as follows:

NOPADCHK `NOPADCHK` means that RACF does not perform the program-accessed data checks for the program. The program is loaded and has access to any currently opened program-accessed data sets, even though the user ID/program combination is not in the conditional access list. `NOPADCHK` allows an installation to define entire libraries of modules (such as the PL/I transient routines or ISPF) as controlled programs without having to give each of these modules explicit access to many program-accessed data sets. Use `NOPADCHK` if you trust the programs to access only data they should.

PADCHK `PADCHK` (the default) means that RACF checks for program-accessed data sets that are already open before executing the program. If there are any open program-accessed data sets, RACF ensures, before it allows this program to be loaded, that this user ID/program combination is in the conditional access list of each data set.

6.6 z/OS UNIX-level security for daemons

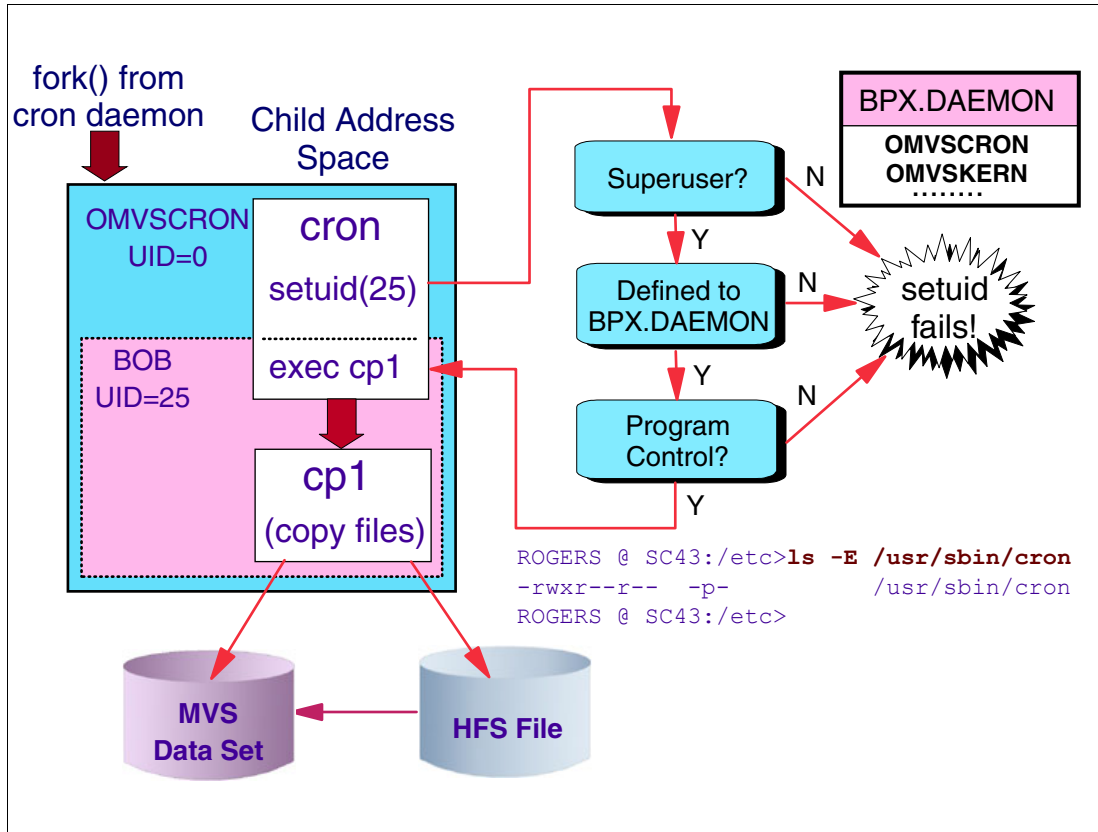


Figure 6-6 z/OS UNIX security with daemons

Example using the cron daemon

The clock daemon *cron* runs commands at specified dates and times.

Figure 6-6 shows the cron daemon running a shell script for the user ID BOB (UID=25). The script will copy HFS files to an MVS data set. Before *cron* can run the script, it will fork a new process and set the identity of this process to UID=25 and the MVS identity to BOB. This will ensure that the script can be run successfully with BOB's shell environment and BOB's access to his MVS data sets. When the job is done, the *cron* child process will end, and *cron* will not have any access to BOB's MVS data sets.

To obtain a higher level of security in a z/OS system with daemons, the RACF FACILITY class called BPX.DAEMON can be used to control the use of the **setuid/seteuid** functions. Only the superusers permitted to the BPX.DAEMON class will be allowed to use the **setuid/seteuid** functions. In addition, there is a program control requirement.

When **setuid()** SYSCALL is issued, the caller (daemon) program, and all other programs currently loaded in the address space, must have been loaded from a z/OS data set with the RACF Program Control activated, meaning they must be controlled programs. Because it is the cloned child daemon program that issues the request, it will inherit the contents of its address space from the parent daemon via fork.

This solution enables an installation to have some superusers with authority to perform system maintenance (for example, to manage the hierarchical file system), whereas other special superusers (daemon user IDs) are allowed to change the identity of a process.

6.7 z/OS UNIX daemons

- ❑ Daemons are computer program that run in the background
 - Typically started when the operating system is initialized
 - Daemon processes can have one or more threads
- ❑ Daemons have names that end with the letter "d"
 - ftpd, used to transfer files between TCP/IP hosts
 - syslogd, the daemon that handles the system log
 - otelnetd, provides access to z/OS UNIX shell applications on the host

Figure 6-7 z/OS UNIX daemons

z/OS UNIX daemons

In UNIX and other computer multitasking operating systems, a daemon is a computer program that runs in the background, rather than under the direct control of a user. Daemons are usually initiated as background processes. Typically daemons have names that end with the letter "d."

Daemons are typically started when the operating system is initialized and remain active to perform standard services. Some programs that initialize processes for users are considered daemons, even though these daemons are not long-running processes.

A user or daemon process can have one or more threads. A thread is a single flow of control within a process. Application programmers create multiple threads to structure an application in independent sections that can run in parallel for more efficient use of system resources.

syslogd

syslogd is the daemon that handles the system log. It is a server process that is typically started as one of the first processes in a z/OS UNIX environment. Servers and stack components use *syslogd* for logging purposes and can also send trace information to *syslogd*. The named daemon logs messages to the *syslog* daemon. For BIND 9, specify the *syslog* option in the channel phrase of the logging statement in the *named.conf* file in order to use this function. Also for BIND 9, you can direct a category to the *default_syslog* channel.

ftpd

ftpd is the daemon used to transfer files between TCP/IP hosts.

otelnstd

The *otelnstd* daemon provides access to z/OS UNIX shell applications on the host using the Telnet protocol. The z/OS UNIX Telnet server lets hosts in an IP network log on to the z/OS shell environment directly, without going through TSO.

Daemon processes

In a UNIX environment, the parent process of a daemon is often (but not always) the `init` process (PID=1). Processes usually become daemons by forking a child process and then having their parent process immediately exit, thus causing `init` to adopt the child process. See “FTPD daemon processing” on page 175.

6.8 TCP/IP and z/OS

- ❑ TCP/IP on z/OS supports all of the well-known server and client applications
- ❑ The TCP/IP started task is the engine that drives all IP-based activity on z/OS
 - The TCP/IP profile data set controls the configuration of the TCP/IP environment
- ❑ The FTP server implements the FTP standard and can communicate with any FTP clients on the network
- ❑ The telnet server implements a standard line mode telnet daemon
- ❑ Even though z/OS is an EBCDIC host, communication with ASCII-based IP applications is seamless

Figure 6-8 TCP/IP and z/OS

TCP/IP and z/OS

The z/OS operating system has existed in one form or another for decades, and has been known by many other names since it was introduced in 1964 as “OS/360.” The most common of the older names is Multiple Virtual Storage or MVS. Even today, you will often hear z/OS system programmers use the term “MVS” to mean “z/OS.”

Because UNIX is an operating system on other platforms, you might wonder which of the two, MVS or z/OS UNIX System Services, is the real operating system. The answer is both, actually. MVS services and z/OS UNIX services are two sets of services available in z/OS and there are many others (such as TSO/E services and JES services). Most system functions of z/OS fall under the MVS heading, but the z/OS UNIX environment forms a significant subset.

TCP/IP daemon

The single entity that handles, and is required for, all IP-based communications in a z/OS environment is the TCP/IP daemon itself. The TCP/IP daemon implements the IP protocol stack and runs a huge number of IP applications to the same specifications as any other operating system would.

► TCP/IP profile

The TCP/IP profile is read by TCP/IP when it is started. If a change needs to be made to the TCP/IP configuration after it has been started, TCP/IP can be made to re-read the profile dynamically (or read a new profile altogether).

FTP server

FTP, like some other IP applications, is actually a z/OS UNIX System Services application. It can be started within an MVS environment, but it does not remain there very long. It immediately forks itself into the z/OS UNIX environment and tells the parent task to issue itself a `ki11` command.

Telnet daemon

There are two Telnet servers available in the z/OS operating environment. One is the TN3270 server, which supports line mode Telnet, but it is seldom used for just that. Instead, it is primarily used to support the TN3270 Enhanced protocol. The other Telnet server is a line mode server only, referred to as the z/OS UNIX Telnet server, or `otelnetd`.

EBCDIC and ASCII

z/OS data sets are encoded in the Extended Binary Coded Decimal Interchange (EBCDIC) character set. This is a character set that was developed before ASCII (American Standard Code for Information Interchange) became commonly used. Most systems that are UNIX-based use ASCII. In addition, z/OS UNIX files are encoded in ASCII.

You need to be aware of the differences in encoding schemes when moving data from ASCII-based systems to EBCDIC-encoded systems. Generally the conversion is handled internally, for example, when text is sent from a 3270 emulator running on a PC to a TSO session. However, when transferring programs, normally these must not be translated and a binary transfer must be specified. ASCII and EBCDIC are both 8-bit character sets.

6.9 FTP as a network protocol

- ❑ FTP allows users to copy files from one machine to another
 - Data transfer between the client (the end user) and the server in either direction
 - A client can issue FTP commands to the server
- ❑ Network protocol
 - FTP user
 - FTP client
 - FTPD task
 - Started task JCL

Figure 6-9 FTP as a network protocol

File Transfer Protocol (FTP)

The File Transfer Protocol (FTP) allows a user to copy files from one machine to another. The protocol allows for data transfer between the client (the end user) and the server in either direction. In addition to copying files, the client can issue FTP commands to the server to manipulate the underlying file system of the server (for example, to create or delete directories, delete files, rename existing files, and so on). FTP is the most frequently used TCP/IP application for moving files between computers.

Network protocol

FTP is built on a client-server architecture and utilizes separate control and data connections between the client and server.

From an FTP user's point of view, the link is connection-oriented. FTP uses TCP as a transport protocol to provide reliable end-to-end connections. Both hosts must run TCP/IP to establish file transfer.

FTP users

FTP users can authenticate themselves using a clear-text sign-in protocol but can connect anonymously if the server is configured to allow it.

FTP client

The first FTP client applications were interactive command-line tools, implementing standard commands and syntax. Graphical user interface clients have since been developed for many of the popular desktop operating systems in use today. To access remote files, the user must identify himself or herself to the server. At this point the server is responsible for authenticating the client before it allows the file transfer.

FTPD task

The FTPD task could very well be executed using `/usr/sbin/ftpd` and a few organizations probably do just that. However, the FTP server can be autologged by the TCP/IP started task if JCL is used. So, the FTP daemon is best started using JCL, as shown in Example 6-1.

Example 6-1 FTPD started task JCL

```
//FTPD  PROC  MODULE='FTPD',PARMS=''
//FTPD  EXEC  PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//      PARM=' /&PARMS'
//SYSFTPD DD  DISP=SHR,DSN=SYS1.TCPPARMS(FTPSDATA)
```

6.10 FTPD daemon processing

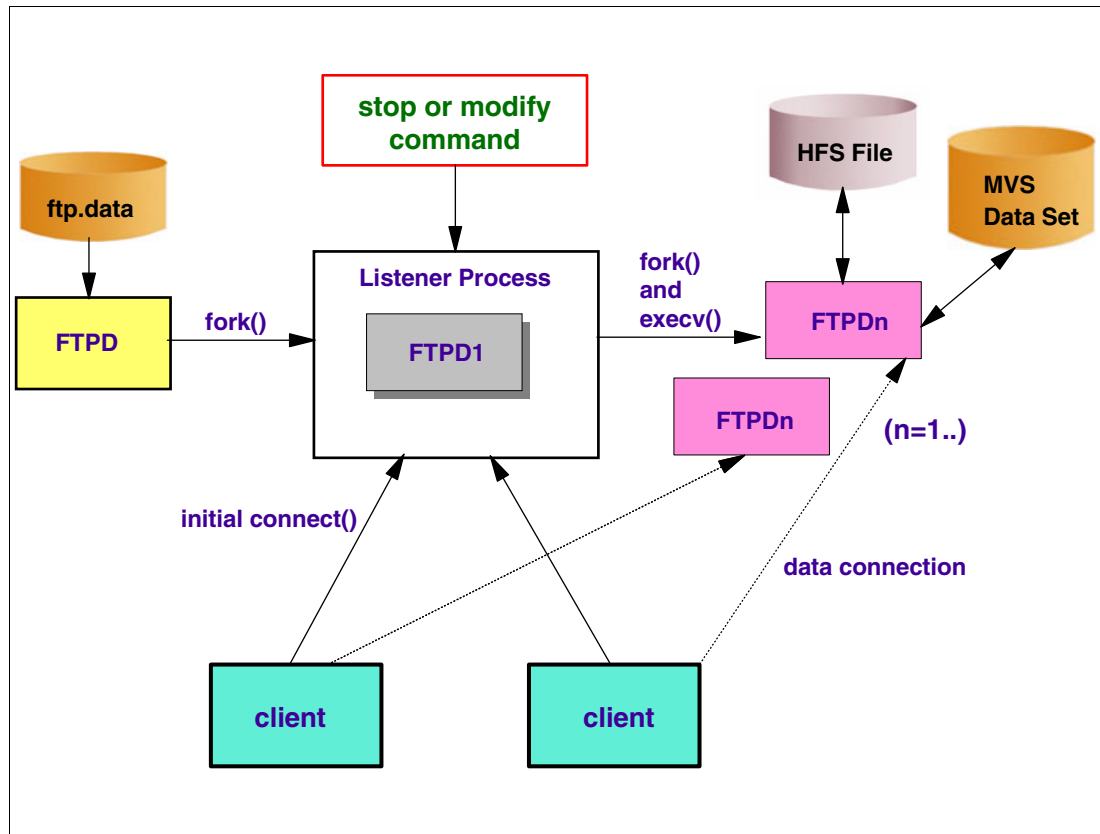


Figure 6-10 FTPD daemon processing

FTPD daemon processing

FTP is used to transfer files between TCP/IP hosts. The FTP client is the TCP/IP host that initiates the FTP session; the FTP server is the TCP/IP host to which the client connects.

The z/OS model for the FTP server includes a daemon process and a server process. The daemon process starts when you start your cataloged procedure (for example, START FTPD) and it listens for connection requests on a specific port.

When the daemon accepts an incoming connection, it creates a new process (server's address space) for the FTP server, which handles the connection for the rest of the FTP login session. Each login session has its own server process.

The FTP server uses two different ports and manages two TCP connections as follows:

- ▶ The port is the well-known port 21 unless otherwise specified. Port 21 is used to control the connection (user ID and password).
- ▶ Port 20 is used for actual data transfer based on the FTP client's requests.

Listener process

The FTP server in z/OS IP consists of the daemon (the listener) or *ftpd1* and server address space (or processes). The daemon performs initialization, listens for new connections, and starts a separate server address space for each connection (*ftpdn*).

FTP server process

When a new client FTP connects to the FTPD1 daemon process, ftpd forks an FTP server process; thus, a new jobname is generated by z/OS UNIX System Services.

The client and server use a different connection for transferring data; this connection is called the data connection. By default, the data port is one less than the control connection port. For example, if the control connection port is 21, the data port is 20. An FTP client can override the default data port by directing the server to run in passive mode. In passive mode, the server uses an ephemeral port for the data port. Passive mode is requested by firewall-friendly clients and by clients initiating three-way data transfers.

6.11 syslogd daemon

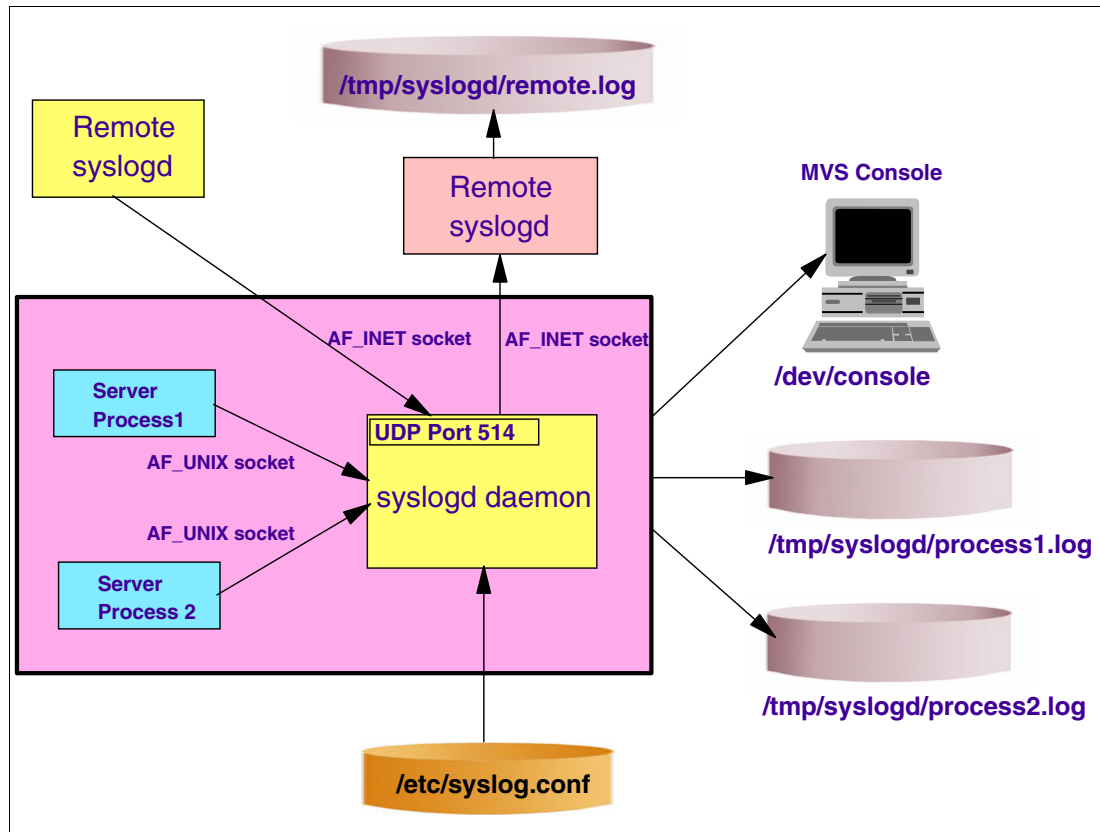


Figure 6-11 syslogd daemon

syslogd daemon

The syslog daemon (syslogd) is a server process that must be started as one of the first processes in your z/OS UNIX environment. Communications Server for z/OS server applications and components use syslogd for logging purposes and can also send trace information to syslogd. Servers on the local system use AF_UNIX sockets to communicate with syslogd; remote servers use the AF_INET socket.

The syslogd daemon reads and logs system messages to the MVS console, log files, other machines, or users, as specified by the configuration file `/etc/syslog.conf`. A sample is provided in `/usr/lpp/tcpip/samples/syslog.conf`.

If the syslog daemon is not started, the application log may appear on the MVS console. The syslog daemon must have a user ID; for example, SYSLOGD defined in RACF with UID=0. The syslogd daemon uses the following files:

<code>/dev/console</code>	Operator console
<code>/etc/syslog.pid</code>	Location of the process ID
<code>/etc/syslog.conf</code>	Default configuration file
<code>/dev/log</code>	Default log path for z/OS UNIX datagram socket
<code>/usr/sbin/syslog</code>	Syslog server

syslogd can only be started by a superuser. It can be terminated using the SIGTERM signal. If you want syslogd to receive log data from or send log data to remote syslogd servers, reserve UDP port 514 for the syslogd job in your PROFILE.TCP/IP data set and enter the syslog service for UDP port 514 in the services file or data set (for example, /etc/services).

Starting syslogd

syslogd can be started only by a task or user with superuser authority (UID 0). You can start syslogd from a procedure or from the UNIX shell. If you start syslogd from a procedure that does not use BPXBATCH, the resulting job name is the same as the procedure name. If you start syslogd from the UNIX shell or from a procedure that uses BPXBATCH, the resulting job name is the user ID or the value of the `_BPX_JOBNAME` environment variable.

If you start syslogd from the UNIX shell or from BPXBATCH, start it as a background shell command by specifying an ampersand (&) as the last character on the command. If you do not specify the ampersand, control does not return to the shell until syslogd ends. Specifying an ampersand is especially important if syslogd is started from a shell script, such as /etc/rc.

If you start syslogd from a cataloged procedure that uses BPXBATCH, you need to include a sleep command in your script after the start for syslogd to provide syslogd the time to initialize before the shell script ends.

6.12 Monitoring the TCP/IP network

- ❑ TSO NETSTAT and z/OS UNIX netstat/onetstat
 - This command is probably the most essential IP command used by network administrators
- ❑ TSO PING and z/OS UNIX ping/oping
 - No IP implementation is complete without the ability to perform rudimentary connectivity tests
- ❑ TSO TRACERTE and z/OS UNIX traceroute/otracer
 - This command performs the UDP expired datagram method of testing the reachability of every hop in a network path
- ❑ TSO RPCINFO and z/OS UNIX orpcinfo
 - This command displays the servers that are registered and operational with any portmapper on your network

Figure 6-12 Monitoring the TCP/IP network

TSO NETSTAT and z/OS UNIX netstat/onetstat commands

These commands provide the following:

- ▶ Information about the status of the local host, including information about TCP/IP connections, network clients, gateways, and devices
- ▶ DNS cache information from the system-wide resolver
- ▶ The ability to drop connections for users who have the MVS.VARY.TCPIP.DROP statement defined in their RACF profile

When new functions are added to TCP/IP in the z/OS Communications Server, new information is also needed from the **netstat** command in terms of new command options, new Netstat reports, or changes to existing Netstat reports. Any program that post processes output lines from the **netstat** command and depends on the content of these output lines from the **netstat** command should be reviewed and possibly modified when maintenance or a new release of z/OS is being installed.

TSO and z/OS UNIX ping commands

A TSO **PING** or z/OS UNIX **ping** command sends an echo request to a foreign node (remote host) to determine whether the node is accessible. When a response to a **ping** command is received, the elapsed time is displayed. The time does not include the time spent communicating between the user and the TCP/IP address space.

TSO TRACERTE and z/OS UNIX traceroute/otracert commands

The TSO **TRACERTE** and z/OS UNIX **traceroute/otracert** commands help you debug network problems.

The TSO **TRACERTE** command is useful for debugging various network problems. It sends UDP requests with varying TTL (time-to-live) or hop count values and then waits for the routers between the local and remote hosts to send TTL-exceeded messages.

The z/OS UNIX **traceroute** command is useful for debugging various network problems. This command sends UDP requests with varying TTL (time-to-live) or hop limit values and then waits for the routers between the local and remote hosts to send time-exceeded messages.

TSO RPCINFO and z/OS UNIX orpcinfo commands

The TSO **RPCINFO** and z/OS UNIX **orpcinfo** commands display the servers that are registered and operational with any portmapper or rcpbind servers on your network that use RPC binding protocol Version 2.

Use the **RPCINFO** command to display the servers that are registered and operational with any portmapper or rcpbind servers on your network.

Use the **orpcinfo** command to display the servers that are registered and operational with any portmapper on your network. The **orpcinfo** command makes a remote procedure call (RPC) to an RPC server and displays the results.

6.13 z/OS Communications Server security

- ❑ z/OS Communications Server provides numerous enterprise-strength security services to protect mission-critical data
- ❑ Application security
- ❑ TCP/IP resource protection
- ❑ Network security principles
 - Cryptography: The foundation of good security
 - End to end security
 - Workload-based security deployment
- ❑ Network security protocols

Figure 6-13 z/OS Communications Server security

z/OS Communications Server security

The z/OS Communications Server, along with other elements of z/OS, provides numerous enterprise-strength security services to protect your mission-critical data. This topic provides an overview of these technologies and how they can be used for a safe and secure z/OS TCP/IP deployment.

The Communications Server provides security event reporting to record potential security violations. These services can help you identify potential sources of subsequent attacks, respond more quickly to network attacks, and manage system resources during periods of high network traffic for key applications.

Application security

The Communications Server protects data and other system resources accessed by applications included in the Communications Server element. This protection requires verification of the identity of the end user requesting access. This process is called identification and authentication. In addition, access to resources must be limited to those users with permission. This process is called access control. Communications Server applications use RACF for identification and authentication, and for access control decisions. Authenticated users are granted access to only those RACF resources for which they have permission.

TCP/IP resource protection

The Communications Server uses the System Authorization Facility (SAF) to protect TCP/IP resources from unauthorized access. These resources are represented by resource profiles defined in the SERVAUTH class. When describing resource profile names, the following conventions are used:

- ▶ *sysname* is the MVS system name. Substitute your system name.
- ▶ *tcpname* is the TCP/IP job name. Substitute your job name.
- ▶ *ftpdaemonname* is the job name of the FTP daemon. Substitute your FTPD job name. If your FTPD job name contains fewer than 8 characters, substitute the job name of the process that is started by FTPD, which is usually the original job name with the number 1 appended.
- ▶ *rpcbindname* is the job name of rpcbind. Substitute your rpcbind job name. If your rpcbind job name contains fewer than 8 characters, substitute the job name of the process that is started by rpcbind, which is usually the original job name with the number 1 appended.

Network security principles

The foundation of good security methods begins with cryptography. Cryptography keeps your data and communications secure using techniques such as encryption, authentication, and data integrity.

Cryptographic security solutions can be applied to a portion of the data path or end-to-end, whichever is appropriate for your security policy. Generally, the greatest degree of security is provided when cryptographic methods are used end-to-end. However, if only portions of the data path are considered untrusted by an enterprise (such as the Internet) it may be adequate to protect only the untrusted portion with cryptography. z/OS offers security protocols that can be configured to protect portions of the data path or the entire data path.

In making a security protocol selection, an important consideration is the application workload to be protected. To appreciate this concept, it is helpful to understand where various protocols are implemented from a protocol layering perspective.

Network security protocols

Network security protocols are used to protect data in your network. The security protocols used by Communications Server are:

- ▶ IPsec and VPNs
- ▶ SSL and TLS
- ▶ Application transparent transport layer security
- ▶ Kerberos
- ▶ OSPF authentication
- ▶ Secure DNS
- ▶ SNMPv3

6.14 Cryptography security

- ❑ Cryptography keeps your data and communications secure
 - Using encryption, authentication, and data integrity
 - Encryption services protect sensitive data from being read by other than the intended receiver
- ❑ Cryptographic authentication and data integrity
 - Allows hosts to detect if data is altered in transit
- ❑ Public key cryptography
 - Identify and authenticate hosts or users
 - Secure creation of symmetric session keys for both security endpoints only if both hosts

Figure 6-14 Cryptography security

Cryptography security

Cryptography includes a set of techniques for scrambling or disguising data. The scrambled data is available only to someone who can restore the data to its original form. The purpose is to make data unintelligible to unauthorized persons, but readily decipherable to authorized persons.

The growth of distributed systems and the increasing use of the Internet have resulted in the need for increased data security. Cryptography provides a strong, economical basis for keeping data confidential and for verifying data integrity. Cryptography is already playing a critical and expanding role in electronic commerce and electronic mail services. Emerging markets that require secure data transmission and the authentication of the sender are already relying on cryptography.

The cryptography facility protects the confidentiality of data transmitted between network resources by enciphering and deciphering session data. Cryptography is available for both LU 6.2 and non-LU 6.2 sessions. Support is available for both switched and nonswitched LUs. However, support is not available for binary synchronous communication (BSC) or local non-SNA devices.

The facility establishes cryptographic sessions for application programs and peripheral node LUs that require cryptographic services. For an LU to have a cryptographic session, the host processor must support cryptography.

Encryption services

Encryption services protect sensitive data from being read by other than the intended receiver. The encryption facility uses services provided by the z/OS Integrated Cryptographic Service Facility (ICSF) and S/390® or zSeries Cryptographic Co-Processor. ICSF is a licensed program that runs under MVS and provides access to the hardware cryptographic feature for programming applications. The combination of the hardware cryptographic feature and ICSF provides secure high speed cryptographic services.

Cryptographic authentication

Cryptographic authentication and data integrity services allow communicating hosts to detect whether data is altered in transit. Public key cryptography can identify and authenticate hosts or users.

Public key cryptography

Public key cryptography can also be used in the secure creation of symmetric session keys for both security endpoints. After a secure session is created, successful data authentication and decryption occur only if both hosts have the correct session keys.

Requirement: Triple-DES 24-byte encryption requires the use of Common Cryptographic Architecture (CCA). CCA defines a set of cryptographic functions, external interfaces, and a set of key management rules that provide a consistent, end-to-end cryptographic architecture across various IBM platforms.

Services provided by the cryptographic facilities include handling requests that VTAM receives to generate a cryptographic key. The cryptographic key is used to encipher and decipher session data.

6.15 IPsec in the IP layer

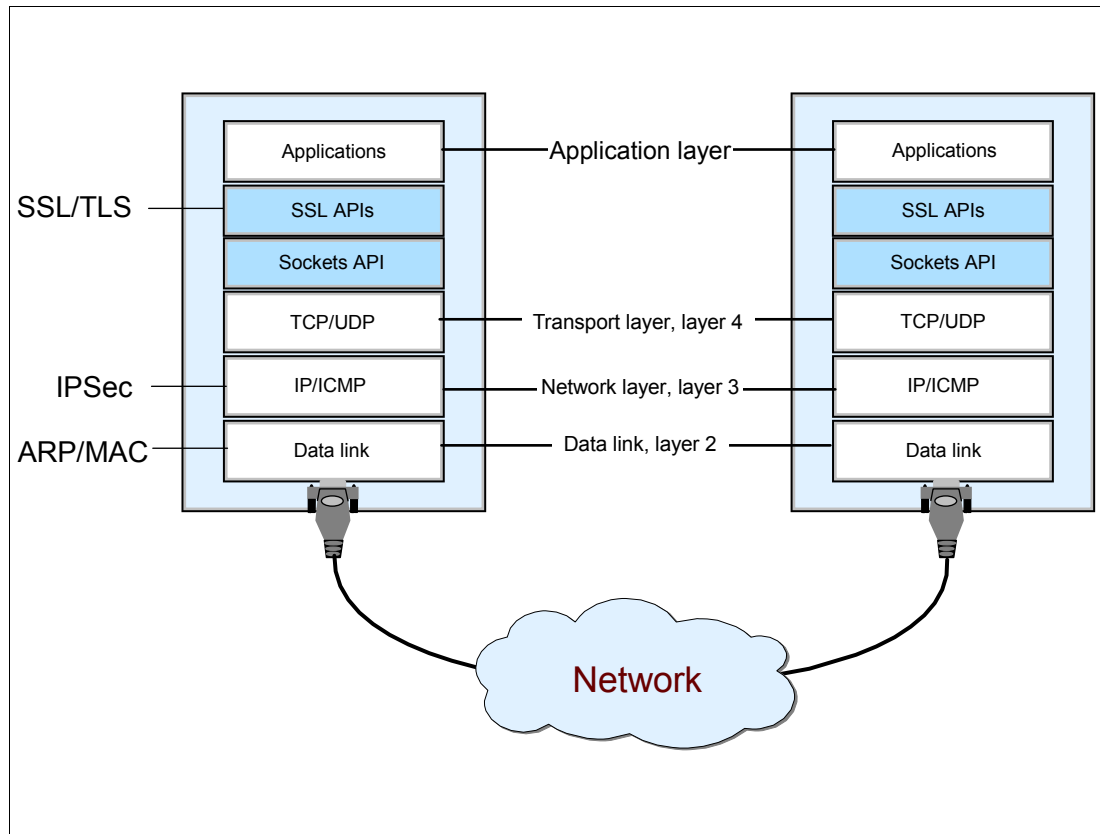


Figure 6-15 IPsec in the IP Layer

IPsec and VPNs

A Virtual Private Network (VPN) is a general term used to describe a secure tunnel (data stream) between two endpoints. The term does not describe a protocol. The industry standard protocol for a VPN is an architecture called IPsec. The IPsec architecture is outlined in RFC 2401, and its implementation encompasses RFCs 2402, 2406, and 2407 (there are various others, but those three are the most significant).

There are some similarities between IPsec and Transport Layer Security (TLS). They both provide encryption of data in the network between two endpoints. They both can provide authentication of those endpoints. An important difference is that IPsec is implemented at the network layer, as shown in Figure 6-15.

IPsec is defined by the IPsec Working Group of the IETF. It provides authentication, integrity, and data privacy between any two IP entities. Management of cryptographic keys and security associations can be done manually or dynamically using an IETF-defined key management protocol called Internet Key Exchange (IKE).

IPsec as a network layer

Because IPsec is at the network layer, the endpoints of a VPN occur at the TCP/IP stack. The endpoints of TLS occur only at an application (like the FTP server). This implies that the endpoint of a VPN can exist on the same host the application is running on, or the endpoint could be at an adjacent firewall in the network. It all depends on the organization's needs.

The other implication of being at the network layer is that all IP traffic can be directed through a VPN. “All traffic” implies not just traffic from different applications, but also traffic from applications using other protocols like UDP or ICMP. With TLS, only the traffic between the two implementing applications is protected.

VPNs

There are two types of VPNs: manual VPN and dynamic VPN. Although z/OS supports manual VPNs, they are not commonly used. Consequently, this section only discusses dynamic VPNs.

A VPN enables an enterprise to extend its private network across a public network, such as the Internet, through a secure tunnel called a *security association*. IPSec VPNs enable the secure transfer of data over the public Internet for same-business and business-to-business communications, and protect sensitive data within the enterprise's internal network.

A dynamic VPN requires a separate server to support the exchange of the keys that are used to encrypt data at each endpoint. In z/OS, the key exchange is supported by the IKE daemon. IKE stands for Internet Key Exchange, which is the standard (RFC 2409) protocol used to exchange keys for a VPN.

How is this all accomplished on z/OS? The characteristics of the dynamic VPN are controlled by the TCP/IP stack using information from the *policy agent*, which is a daemon that runs with the purpose of reading policy definitions from a Lightweight Directory Access Protocol (LDAP) server. The policy definitions are in turn read by the TCP/IP stack.

6.16 SSL and TLS

- ❑ SSL protocol provides data encryption, data origin authentication, and message integrity
 - As SSL gained in popularity, the IETF formally standardized SSL, made a few improvements and changed the name to Transport Layer Security (TLS)
 - The SSL protocol provides data encryption, data origin authentication, and message integrity
- ❑ A TLS connection begins with a handshake and an exchange of information includes the following:
 - Authentication of the server
 - A decision about how the data is to be encrypted
 - Optionally, the authentication of the client

Figure 6-16 SSL and TLS

SSL and TLS

Secure Sockets Layer (SSL) is a protocol standard developed by the Netscape Communications Corporation that uses encryption to provide confidentiality and authentication between two TCP/IP applications. As SSL gained in popularity, the IETF formally standardized SSL, made a few improvements, and changed the name to Transport Layer Security (TLS). TLS is defined in RFC 2246.

With z/OS, the FTP server, the FTP client, and the TN3270 server all support TLS. Configuration of TLS is similar in all three environments, and the principles involved are common throughout as well. RFC 2246 describes TLS in general – but FTP has the added complexity of a control and data connection.

RFC 4217 was published to clarify how FTP handles TLS. An easy way to introduce the principles is by beginning with statements in the FTP.DATA file required to support a TLS connection.

SSL protocol

The SSL protocol provides data encryption, data origin authentication, and message integrity. It also provides server and client authentication using X.509 certificates. SSL begins with a handshake during which the server is authenticated to the client using X.509 certificates. Also, the client can optionally be authenticated to the server. During the handshake, security session parameters, such as cryptographic algorithms, are negotiated and session keys are

created. After the handshake, the data is protected during transmission with data origin authentication and optional encryption using the session keys.

Transport Layer Security (TLS)

The term “industry standard” is used to describe security features of z/OS that are widely implemented and either formally or informally standardized. One item, Transport Layer Security (TLS), is treated in some detail here because it is probably the most widely used security protocol on the Internet today. And in order to implement it on z/OS, it is a concept that should be fully understood.

Security entails the reduction of the likelihood of damage, whether intentional or inadvertent. Within a context of z/OS, features such as TLS/SSL, IPsec, and SSH can be used to improve the security of data in the network. Other features, such as AT-TLS and SAF-based security can also be used. Intrusion detection services are already active to some degree within the TCP/IP stack. The policy agent is the repository for many security-related configuration values. Due to the complexity of the SNA architecture and its present limited use in the telecommunications portion of the network, security in this context is not as much of a concern as it is in an IP context.

The TN3270 environment is unique and complex enough to warrant some special attention. As mentioned, the TN3270 server supports TLS. In addition, the TN3270 server makes full use of SAF-based authentication. And, if desired, TLS and SAF can be used together to force a TN3270 client to send a certificate that is associated with an SAF-controlled user ID, allowing a product like RACF further control.

6.17 Application transparent transport layer security

- ❑ AT-TLS transparently performs Transport Layer Security (TLS) for applications
 - By invoking the z/OS System Secure Socket Layer (SSL) in the TCP transport layer
- ❑ Communications Server support for AT-TLS
 - Provides for invocation of System SSL in the TCP transport layer of the stack
 - Application Transparent Transport Layer Security (AT-TLS) support is controlled by:
 - The TTLS or NOTTLS parameter on the TCPCONFIG statement in the TCP/IP profile
 - AT-TLS policy

Figure 6-17 Application transparent transport layer security

Application transparent transport layer security

AT-TLS transparently performs Transport Layer Security (TLS) on behalf of the application by invoking the z/OS System Secure Socket Layer (SSL) in the TCP transport layer. System SSL provides support for the TLSv1, SSLv3 and SSLv2 protocols. AT-TLS uses a policy-based configuration, and the Policy Agent application is required to define rules and actions to the TCP/IP stack for TCP connections using AT-TLS. Displays for AT-TLS policy are provided by pasearch and Netstat.

AT-TLS and TLS

Application Transparent TLS (AT-TLS) is a unique form of TLS on the z/OS end of the session. In principle, it is quite simple: instead of having the application itself be TLS-capable and TLS-aware, the establishment of the TLS connection is pushed down the stack into the TCP layer.

AT-TLS will appear identical to normal TLS to any application connecting to the z/OS host. The AT-TLS environment is activated by a simple option within the TCPCONFIG statement block in the TCP/IP profile data set: TTLS. When coded, the TCP/IP stack will use the policy agent (in the same fashion as it does for IPsec) to determine how to handle each application's communication.

Many applications on z/OS can run without even being aware that the connection is using TLS. Remote clients cannot distinguish between "normal" TLS (where the application is doing

the socket calls necessary for TLS) and AT-TLS (where the TCP layer handles the connection).

Communications Server support for AT-TLS

Communications Server provides for invocation of System SSL in the TCP transport layer of the stack. AT-TLS support is controlled by the TTLS or NOTTLS parameter on the TCPCONFIG statement in the TCP/IP profile. When AT-TLS is enabled, AT-TLS statements in Policy Agent define the security attributes for connections that match AT-TLS rules. This policy-driven support can be deployed transparently underneath many existing sockets, leaving the application unaware of the encryption and decryption being done on its behalf. Support is also provided for applications that need to negotiate TLS or need to participate in client authentication. These applications must be aware of AT-TLS support and use ioctl support provided by AT-TLS. AT-TLS supports the TLS, SSLv3, and SSLv2 protocols.

AT-TLS policy

An AT-TLS policy is provided to the stack by the Policy Agent. The Policy Agent main configuration file contains a TcplImage statement for each stack that is to receive policy, and can optionally contain a CommonTTLSConfig statement that identifies a local shared AT-TLS policy file.

The TcplImage statement identifies the z/OS UNIX file or MVS data set that contains policy for that stack. This policy file can contain a TTLSConfig statement to identify the z/OS UNIX file or MVS data set that contains the local AT-TLS policy. The TTLSConfig statement is required for each stack that is to receive AT-TLS policy. If both a TTLSConfig statement and a CommonTTLSConfig statement are defined, the specified CommonTTLSConfig file is processed before the TTLSConfig policy file specified for that stack.

6.18 Kerberos

- ❑ Kerberos is a network authentication protocol
 - Kerberos provides strong authentication for:
 - Client/server applications using secret-key cryptography
 - Allows for data encryption and prevents passwords from having to be retyped to access networked services
 - Prevents their transmission in plain text over the network
- ❑ Kerberos builds on symmetric key cryptography
- ❑ otelnetd support
- ❑ Kerberos authentication service
- ❑ Obtaining credentials with tickets

Figure 6-18 Kerberos

Kerberos

Kerberos is a network authentication protocol that is designed to provide strong authentication for client/server applications using secret key cryptography. The Kerberos network authentication protocol assumes that services and workstations communicate over an insecure network. It allows clients and servers to do either one-way or two-way (mutual) authentication. It allows for data encryption and prevents passwords from having to be retyped to access networked services and also prevents their transmission in plain text over the network. This feature can help reduce the need to manage multiple passwords.

Key cryptography

Kerberos builds on symmetric key cryptography and requires a trusted third party, and optionally can use public key cryptography by utilizing asymmetric key cryptography during certain phases of authentication.

otelnetd support

otelnetd supports Kerberos Version 5 for authentication on IPv4 connections. Authentication is not supported on IPv6 connections (that is, if tcp6 is specified for otelnetd in inetd.conf). On z/OS, Kerberos is implemented by Security Server.

Kerberos authentication service

Kerberos performs authentication as a trusted third-party authentication service by using conventional shared secret key cryptography. Kerberos provides a means of verifying the

identities of principals, without relying on authentication by the host operating system, without basing trust on host addresses, without requiring physical security of all the hosts in the network, and under the assumption that packets traveling along the network can be read, modified, and inserted at will.

Obtaining credentials with tickets

The two methods for obtaining credentials, the initial-ticket exchange and the ticket-granting-ticket exchange, use slightly different protocols and require different API routines.

The basic difference an application programmer sees is that the initial-ticket exchange does not require a ticket-granting-ticket (TGT) but does require the client's secret key. Usually, the initial-ticket exchange is for a TGT, and TGT exchanges are used from then on. In a TGT exchange, the TGT is sent as part of the request for a ticket and the reply is encrypted in the session key obtained from the TGT. Thus, after a user's password is used to obtain the initial TGT, it is not required for subsequent TGT exchanges to obtain additional tickets.

6.19 OSPF authentication

- ❑ OSPF (Open Shortest Path First) dynamic routing protocol supports message authentication and message integrity of OSPF
 - Routing messages through the use of the OSPF MD5 Authentication security protocol
 - OSPF MD5 Authentication ensures that an unauthorized IP resource cannot inject OSPF routing messages
- ❑ OMPROUTE computes a secure MAC for the routing message using the MD5 algorithm
 - This MAC is sent with the routing message so that the message can be authenticated by the receiver
- ❑ IPv4 and IPv6 considerations

Figure 6-19 OSPF authentication

OSPF authentication

Communications Server OSPF (Open Shortest Path First) dynamic routing protocol supports message authentication and message integrity of OSPF routing messages through the use of the OSPF MD5 Authentication security protocol as defined by RFC 2328. OSPF MD5 Authentication ensures that an unauthorized IP resource cannot inject OSPF routing messages into the network without detection, thus ensuring the integrity of the routing tables in the OSPF routing network.

OMPROUTE processing

OMPROUTE computes a secure MAC for the routing message using the MD5 algorithm. This MAC is sent with the routing message so that the message can be authenticated by the receiver.

IPv4 OSPF

IPv4 OSPF includes authentication as part of the OSPF protocol. OMPROUTE supports both password authentication and MD5 cryptographic authentication for IPv4 OSPF. For IPv6 OSPF, authentication has been removed from OSPF itself. Instead, IPv6 OSPF relies on IPsec to ensure integrity and authentication of routing exchanges. As a result, OMPROUTE does not include any explicit authentication support, but instead relies on the underlying support provided by the z/OS TCP/IP stack.

IPv6 OSPF

To use IPsec to authenticate IPv6 OSPF routing exchanges on a link over which OMPROUTE establishes adjacencies, you must create a single manual security association (SA) for all traffic on that link, with corresponding filter definitions to permit the OSPF traffic. Use the interface SECCLASS to define unique security associations for each link.

IPv4 and IPv6 usage considerations

IPv4 OSPF authentication is implemented within the IPv4 OSPF protocol. However, IPv6 OSPF security (both authentication and encryption) is implemented using IPsec. Because OSPF uses both multicast messages and unicast messages, it is not possible to use dynamic tunnels for OSPF traffic. Instead, manual tunnels must be used. The IBM Configuration Assistant for z/OS Communications Server automates the process of creating IPv6 OSPF tunnels.

It is expected that the same manual tunnel is to be used for all link-local unicast and multicast traffic. Additional tunnels might be used for IPv6 OSPF virtual links.

Because multicast traffic is one-to-many, the manual tunnel must use the same Security Parameter Index (SPI) and keys for inbound and outbound traffic. Whatever SPI values and keys are used must be coordinated with all IPv6 OSPF peers on the LAN segment. Also, because this manual tunnel is to be used to protect traffic with various source and destination addresses, you must specify any6 for the local and remote security endpoint locations.

6.20 Secure DNS

- ❑ Communications Server supports DNS at the Version 9.1 of BIND
 - This level of DNS has built-in security features, DNSSEC and TSIG
- ❑ DNSSEC support
 - Ensures a trusted DNS
- ❑ TSIG protocol
 - Allows for transaction level authentication using shared secrets and one way hashing

Figure 6-20 Secure DNS

Secure DNS

The Communications Server supports DNS at Version 9.1 of BIND. This level of DNS has the built-in security features DNSSEC and TSIG.

DNSSEC support

DNSSEC ensures that DNS query results are not spoofed and in fact originate from a trusted DNS. DNSSEC defines extensions to DNS that provide data integrity and authentication to security-aware resolvers and applications through the use of cryptographic digital signatures. DNSSEC is defined by the IETF in RFC 2535.

TSIG protocol

TSIG is a protocol for Secret Key Transaction Signatures for DNS. This protocol allows for transaction-level authentication using shared secrets and one way hashing. It authenticates dynamic updates as coming from an approved client, or responses as coming from an approved recursive name server.

6.21 SNMPv3

- ❑ **SNMPv3 is the solution for SNMP security**
 - **USM provides different levels of security based on the user accessing the managed information**
 - **USM used for authentication and privacy**
 - **VACM provides the use of authentication and data encryption for privacy**
- ❑ **An architecture for SNMP management frameworks**
- ❑ **Three major subsystems are defined:**
 - **Message processing subsystem**
 - **Security subsystem**
 - **Access control subsystem**

Figure 6-21 SNMPv3

SNMPv3

z/OS Communications Server SNMP supports SNMPv3. The established community-based protocols SNMPv1 and SNMPv2 are also supported. SNMPv3, defined in RFCs 3410 through 3415, is the standards-based solution for SNMP security.

USM

SNMPv3 is categorized as a user-based security model (USM), which provides different levels of security based on the user accessing the managed information. To support this security level, the SNMPv3 framework defines several security functions, such as USM for authentication and privacy, and view-based access control model (VACM), which provides the ability to limit access to different MIB objects on a per-user basis, and the use of authentication and data encryption for privacy.

SNMP management framework

However, SNMP is not just enhanced security. It defines an architecture for SNMP management frameworks, with the intent that pieces of the architecture can advance over time without requiring the entire structure to be rewritten.

For that reason, three major subsystems are defined:

- ▶ Message processing subsystem
- ▶ Security subsystem
- ▶ Access control subsystem

The framework is structured so that multiple models can be supported concurrently and replaced over time. For example, although there is a new message format for SNMPv3, messages created with the SNMPv1 and SNMPv2 formats can still be supported. Similarly, the user-based security model can be supported concurrently with the community-based security models previously used.

In addition, SNMPv3 added other key updates to the protocol, such as the following:

- ▶ Improved notification support

A new notification type, INFORM, is like a TRAP that requires an acknowledgement. If the acknowledgment is not received, the INFORM is re-sent.

- ▶ Trap filtering

With SNMPv3, a TRAP can also be filtered at the sender.

- ▶ Dynamic configuration

The SNMP agent can be dynamically configured using MIB modules defined in RFC 3584 and RFCs 3411 through 3415.

6.22 Protection network resources

- ❑ Protecting network resources by SERVAUTH resource class profiles
- ❑ Communications Server uses SAF to protect TCP/IP resources from unauthorized access
 - These resources are represented by resource profiles defined in the SERVAUTH class
- ❑ Allow users to access IP addresses when executing certain programs
 - Protect the names of network security zones (containing IP addresses) using SERVAUTH class
 - Set up program control for a SERVAUTH resource profile

Figure 6-22 Protecting network resources

Protecting network resources

When you specify the user IDs that are allowed to connect into a specific Telnet port, DCAS server, or FTP port, you associate the user IDs to each server's RACF SERVAUTH profile. The user ID associated with the client certificate can then be checked against the SERVAUTH class profile entry. The use of this RACF class is optional. If the SERVAUTH RACF class is active and a RACF profile for the port is defined, this level of RACF authorization will be verified prior to connection negotiation. If the SERVAUTH class is not active or there is no RACF profile, this indicates that this level of check is not required and the client is allowed to connect to the server as long as the client certificate was validated.

The TCP/IP address space uses the server access authorization (SERVAUTH) class of the System Authorization Facility (SAF) to protect TCP/IP resources from unauthorized access. The use of SERVAUTH can be optional and is available in degrees so that installations can pick and choose the access needed.

Access to IP addresses

You can allow users to access IP addresses only when executing certain programs when you protect the names of network security zones (containing IP addresses) using SERVAUTH class resources. For example, when you control access to network security zones, you can permit network administrators to access certain zones only when using the **ping** and **traceroute** commands.

Program control

To set up program control for a SERVAUTH resource (representing a network security zone), create a profile in the SERVAUTH class specifying UACC(NONE), or specify ID(*) ACCESS(NONE) to ensure no access by general users.

Then, permit certain users using WHEN(PROGRAM(program-name)) with the ID and ACCESS operands on the PERMIT command, as follows:

```
RDEFINE SERVAUTH resource-name UACC(NONE)
PERMIT resource-name CLASS(SERVAUTH) ID(user or group or *) ACCESS(READ)
        WHEN(PROGRAM(program-name))
```

These profiles permit the specified users or groups to access network security zones protected by SERVAUTH resources only when executing the specified program or command.

Note: Program access to SERVAUTH resources in ENHANCED program security mode operates much the same as it does in BASIC program security mode, with one exception. RACF allows program access to SERVAUTH resources to operate in ENHANCED program security mode.

6.23 Network access control

- ❑ Provide administrators the ability to assign permission for z/OS users to access certain networks and hosts
 - Allows users to send or receive data between z/OS and certain networks controlled through z/OS
 - Provides an additional layer of security to any authentication and authorization
- ❑ Essential elements of this function are as follows:
 - IP network is considered the resource to be protected
 - Intranode management network is protected by OSM access control
 - IP addresses are classified into security zones

Figure 6-23 Network access control

Network access control

Network access control gives system administrators the ability to assign permission for z/OS users to access certain networks and hosts. With this function, the ability of users to send or receive data between z/OS and certain networks can be controlled through z/OS. Network access control provides an additional layer of security to any authentication and authorization security that is used in the network or at the peer system by disallowing the unauthorized user to communicate with the peer network resource.

Network access control enables system administrators to represent access to an IP network, subnetwork, or host as a RACF resource. The ability to send IP packets to those networks, subnetworks, or hosts can then be permitted or denied at a RACF user or group level.

This feature provides an additional layer of security to any authentication or authorization that is used at the target system. It might be used, for example, to prevent access to the Internet by anyone except the SMTP server, or it could be used to stop general users attempting to Telnet to a server that contained payroll information.

Defining network access control

In the TCP/IP profile, there is a parameter block, NETACCESS/ENDNETACCESS. This is where you specify the mapping of an IP network, subnetwork, or host to an SAF profile.

Use the NETACCESS statement to configure network access control. Specifically, it allows for the one-to-one mapping between a network, subnetwork, or host, and a Security Access

Facility (SAF) resource name. The network specifications are used to build an internal data structure that maps networks, subnetworks, and hosts to SAF resource names. The mapping is used to construct a complete resource name that is passed to the security product to determine the user's permission to access the network resource. The most specific mapping is used to determine the resource name for the SAF authorization check.

Figure 6-24 shows a sample NETACCESS block.

```
NETACCESS
 10.1.100.0/24      MYSUBNET      ;my workstation subnet
 10.1.100.223/32   MYPC          ;my workstation
 DEFAULT 0         WORLD          ;everything else
ENDNETACCESS
```

Figure 6-24 Sample NETACCESS block

Essential elements of this function

The IP network is considered the resource to be protected.

Use of the IBM zEnterprise System (zEnterprise) intranode management network is protected by OSM access control and is exempt from network access control.

IP addresses are classified into security zones, in which each zone has a certain level of security sensitivity. A default security zone exists for interfaces that are not explicitly associated with a specific security zone.

Note: The intranode management network can be accessed only through OSM interfaces. To send or receive data over OSM interfaces on this network, an application must have READ authorization to the EZB.OSM.sysname.tcpname resource.

SAF is used to check whether users or groups of users have permission to access the security zone. The installation defines a network access resource for each security zone and permits users or groups of users access to the resource. The security zone is represented by an SAF SERVAUTH profile name of:

EZB.NETACCESS.sysname.tcpname.zonename

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 203. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-7800
- ▶ *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-7798
- ▶ *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-7799
- ▶ *IBM z/OS V1R11 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-7801

Other publications

These publications are also relevant as further information sources:

- ▶ *z/OS UNIX System Services Planning*, GA22-7800
- ▶ *z/OS Communications Server: SNA Resource Definition Samples*, SC31-8736
- ▶ *z/OS Communications Server: SNA Resource Definition Reference*, SC31-8778
- ▶ *z/OS Communications Server: IP User's Guide and Commands*, SC31-8780
- ▶ *z/OS Communications Server: IP Configuration Guide*, SC31-8775

Online resources

These Web sites are also relevant as further information sources:

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



ABCS of z/OS System Programming Volume 4

(0.2" spine)
0.17" x 0.473"
90 x 249 pages



ABCs of z/OS System Programming

Volume 4



**SNA, TCP/IP,
Hardware interfaces,
Enterprise Extender**

**Routing, Parallel
Sysplex, Operations**

**TCP/IP applications,
Security**

The ABCs of z/OS System Programming is a 13-volume collection that provides an introduction to the z/OS operating system and the hardware architecture. Whether you are a beginner or an experienced system programmer, the ABCs collection provides the information that you need to start your research into z/OS and related subjects. If you want to become more familiar with z/OS in your current environment, or if you are evaluating platforms to consolidate your e-business applications, the ABCs collection will serve as a powerful technical tool.

The contents of the volumes are as follows:

- ▶ Volume 1: Introduction to z/OS and storage concepts, TSO/E, ISPF, JCL, SDSF, and z/OS delivery and installation
- ▶ Volume 2: z/OS implementation and daily maintenance, defining subsystems, JES2 and JES3, LPA, LNKLST, authorized libraries, SMP/E, Language Environment
- ▶ Volume 3: Introduction to DFSMS, data set basics storage management hardware and software, catalogs, and DFSMSStvs
- ▶ Volume 4: Communication Server, TCP/IP, and VTAM
- ▶ Volume 5: Base and Parallel Sysplex, System Logger, Resource Recovery Services (RRS), global resource serialization (GRS), z/OS system operations, automatic restart management (ARM), Geographically Dispersed Parallel Sysplex (GDPS)
- ▶ Volume 6: Introduction to security, RACF, Digital certificates and PKI, Kerberos, cryptography and z990 integrated cryptography, zSeries firewall technologies, LDAP, and Enterprise identity mapping (EIM)
- ▶ Volume 7: Printing in a z/OS environment, Infoprint Server and Infoprint Central
- ▶ Volume 8: An introduction to z/OS problem diagnosis
- ▶ Volume 9: z/OS UNIX System Services
- ▶ Volume 10: Introduction to z/Architecture, zSeries processor design, zSeries connectivity, LPAR concepts, HCD, and HMC
- ▶ Volume 11: Capacity planning, performance management, RMF, and SMF
- ▶ Volume 12: WLM
- ▶ Volume 13: JES3

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

**BUILDING TECHNICAL
INFORMATION BASED ON
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:
ibm.com/redbooks**

SG24-6984-00

ISBN 073843499X