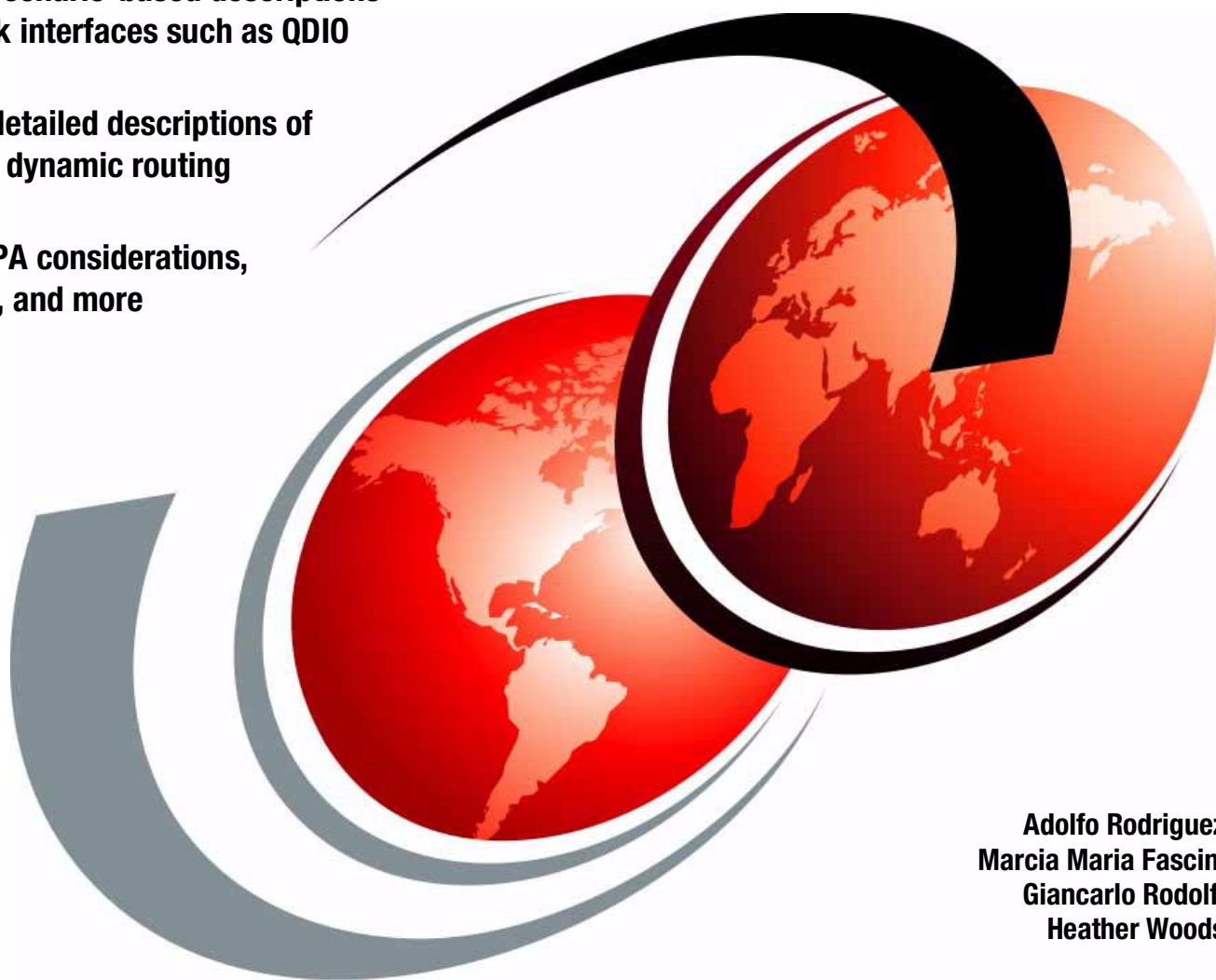


# Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing

Includes scenario-based descriptions  
of network interfaces such as QDIO

Provides detailed descriptions of  
static and dynamic routing

Covers VIPA considerations,  
OSPF, RIP, and more



Adolfo Rodriguez  
Marcia Maria Fascini  
Giancarlo Rodolfi  
Heather Woods





International Technical Support Organization

**Communications Server for z/OS V1R2 TCP/IP  
Implementation Guide Volume 4: Connectivity and  
Routing**

October 2002

**Take Note!** Before using this information and the product it supports, be sure to read the general information in “Notices” on page vii.

**First Edition (October 2002)**

This edition applies to Version 1, Release 2 of Communications Server for z/OS.

Comments may be addressed to:  
IBM Corporation, International Technical Support Organization  
Dept. HZ8 Building 662  
P.O. Box 12195  
Research Triangle Park, NC 27709-2195

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2002. All rights reserved.

Note to U.S Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Contents</b> .....	iii
<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
The team that wrote this redbook .....	ix
Special notice .....	x
Comments welcome .....	xi
<b>Chapter 1. Introduction</b> .....	1
1.1 Network interfaces .....	2
1.1.1 CS for z/OS IP supported devices .....	3
1.1.2 Connectivity restrictions .....	5
1.2 Routing with CS for z/OS IP .....	5
1.2.1 What is routing? .....	5
1.2.2 Routing in a z/OS environment .....	6
1.2.3 z/OS-supported routing daemons .....	7
1.3 Communications Server for z/OS V1R2 IP enhancements .....	7
1.3.1 Hipersockets support .....	8
1.3.2 64-bit real memory exploitation .....	8
1.3.3 IP routing enhancements .....	8
<b>Chapter 2. z/OS interface environment</b> .....	11
2.1 CSM .....	12
2.2 Displaying TCP/IP device resources in VTAM .....	13
<b>Chapter 3. Network interface types</b> .....	17
3.1 ATM .....	19
3.1.1 Native ATM or classic IP .....	19
3.1.2 ATM configuration example .....	20
3.1.3 Modifying the ATM setup .....	23
3.2 CDLC .....	25
3.2.1 MVS IOCP definitions .....	25
3.2.2 TCPIP profile definitions .....	26
3.2.3 CDLC NCP channel definitions .....	26
3.3 CLAW .....	29
3.3.1 Cisco channel host IOCP definitions .....	29
3.3.2 Router definitions .....	29
3.3.3 Host TCP/IP profile statements .....	30
3.4 CTC .....	31
3.5 Hipersockets .....	32
3.5.1 Hipersockets Accelerator .....	36
3.6 Hyperchannel .....	37
3.7 LCS .....	37
3.7.1 PROFILE definitions .....	39
3.8 MPC .....	40
3.8.1 MPCPTP .....	41
3.8.2 MPCOSA .....	43

3.8.3 MPCPTP samehost . . . . .	44
3.9 MPCIPA (QDIO) . . . . .	45
3.9.1 Queued Direct I/O overview . . . . .	45
3.9.2 Configuring for the OSA-Express adapter. . . . .	46
3.9.3 Sample OSA-Express GbE connection between two systems. . . . .	49
3.10 SNALINK LU0 . . . . .	51
3.10.1 SNALINK LU0 definition . . . . .	53
3.10.2 Operating SNALINK . . . . .	55
3.11 SNALINK LU6.2 . . . . .	55
3.11.1 SNALINK LU6.2 definition. . . . .	56
3.12 Virtual devices (VIPA) . . . . .	58
3.13 X.25 NPSI connection . . . . .	58
3.13.1 Supported NPSI configurations. . . . .	59
3.13.2 NPSI GATE. . . . .	59
3.13.3 NPSI GATE Fast Connect . . . . .	59
3.13.4 NPSI DATE . . . . .	60
3.13.5 X.25 NPSI definition . . . . .	60
3.14 XCF . . . . .	64
3.14.1 Dynamic XCF . . . . .	65
3.14.2 Migrating from static XCF definition . . . . .	68
<b>Chapter 4. Routing overview . . . . .</b>	<b>69</b>
4.1 IP addressing . . . . .	70
4.1.1 IP address classes . . . . .	70
4.1.2 Reserved IP addresses. . . . .	71
4.1.3 IP subnets. . . . .	71
4.2 Classless Inter-Domain Routing . . . . .	73
4.3 Routing terminology . . . . .	74
4.4 TCP/IP routing. . . . .	74
4.4.1 Types of IP routing . . . . .	75
4.4.2 IP routing table . . . . .	75
4.4.3 IP routing algorithm. . . . .	77
4.4.4 Displaying the IP routing table . . . . .	78
4.4.5 Static and dynamic routing . . . . .	79
4.4.6 Maintaining routing tables. . . . .	79
4.5 Interior gateway protocols (IGP) . . . . .	80
4.6 Routing daemons . . . . .	81
4.7 Open Shortest Path First (OSPF) . . . . .	82
4.7.1 OSPF terminology. . . . .	82
4.7.2 Link state routing. . . . .	85
4.7.3 Link State Advertisements (LSAs) . . . . .	86
4.7.4 Link state database. . . . .	86
4.7.5 Physical network types . . . . .	88
4.7.6 A basic network with OSPF. . . . .	88
4.8 Routing Information Protocol (RIP) . . . . .	89
4.8.1 Distance vector algorithms . . . . .	89
4.8.2 RIP Version 1 . . . . .	90
4.8.3 RIP V1 limitations . . . . .	91
4.8.4 RIP Version 2 . . . . .	92
4.8.5 RIP Version 2 sample configuration . . . . .	93
4.8.6 RIP database . . . . .	96
<b>Chapter 5. Working with static routes . . . . .</b>	<b>97</b>

5.1 Replaceable static routes . . . . .	98
5.2 Defining static routes . . . . .	98
5.3 Displaying static routes . . . . .	100
5.4 Updating static routes . . . . .	101
5.5 Static routing and OMPROUTE . . . . .	101
<b>Chapter 6. Dynamic routing with OMPROUTE . . . . .</b>	<b>103</b>
6.1 OMPROUTE overview . . . . .	104
6.2 Configuring OMPROUTE . . . . .	105
6.2.1 Create the OMPROUTE cataloged procedure . . . . .	105
6.2.2 Define OMPROUTE environment variables . . . . .	106
6.2.3 Update PROFILE.TCPIP . . . . .	108
6.2.4 Create the OMPROUTE configuration file . . . . .	109
6.2.5 Configuring interfaces to OMPROUTE . . . . .	112
6.3 Configuring OSPF-only options . . . . .	117
6.3.1 OSPF authentication . . . . .	117
6.4 Configuring RIP-only options . . . . .	118
6.4.1 Selective RIP filters in OMPROUTE . . . . .	118
6.5 OMPROUTE commands . . . . .	120
6.5.1 OMPROUTE OSPF commands . . . . .	121
6.5.2 OMPROUTE RIP commands . . . . .	130
6.6 Interface cost considerations . . . . .	131
6.7 Multipath considerations . . . . .	133
6.7.1 Multipath example . . . . .	133
<b>Chapter 7. Routing with VIPA . . . . .</b>	<b>141</b>
7.1 Shared OSA pre-routing . . . . .	143
7.2 VIPA considerations . . . . .	144
7.2.1 VIPA address assignment . . . . .	144
7.2.2 Fault tolerance with VIPA . . . . .	145
7.2.3 Beware of ICMP redirection . . . . .	145
7.2.4 Using SOURCEVIPA . . . . .	146
7.3 Configuring OMPROUTE . . . . .	146
7.3.1 OMPROUTE with OSPF . . . . .	148
7.3.2 OMPROUTE with RIP . . . . .	150
7.4 A VIPA takeover example . . . . .	152
<b>Chapter 8. Interfacing with Cisco EIGRP . . . . .</b>	<b>155</b>
8.1 EIGRP overview . . . . .	156
8.1.1 Features of EIGRP . . . . .	156
8.1.2 Terminology . . . . .	156
8.1.3 Neighbor discovery and recovery . . . . .	158
8.1.4 The DUAL algorithm . . . . .	158
8.2 Sample topology . . . . .	161
8.2.1 Routing topology . . . . .	162
8.3 OSPF configuration in the sysplex . . . . .	163
8.3.1 OMPROUTE configuration . . . . .	164
8.3.2 Verify routing from the host . . . . .	169
8.4 ASBR configuration and redistribution . . . . .	171
8.4.1 Redistribution . . . . .	172
8.4.2 Verify routing from the router . . . . .	176
<b>Appendix A. Working with ORouteD . . . . .</b>	<b>181</b>
Configuring ORouteD . . . . .	182

Configuring ORouteD and VIPA . . . . .	192
Migrating from ORouteD to OMPROUTE . . . . .	194
<b>Appendix B. NCPROUTE</b> . . . . .	195
Configuring NCPROUTE . . . . .	197
NCP definitions . . . . .	201
Token-ring adapter . . . . .	201
Ethernet adapter . . . . .	202
Frame relay connection . . . . .	202
NCST logical unit . . . . .	204
NCPROUTE server host . . . . .	204
IP home addresses and optional IP routing table entries . . . . .	205
Configuration summary . . . . .	205
Controlling the use of SNALINK . . . . .	206
CDLC for NCPROUTE . . . . .	207
NCPROUTE and SNALINK for remote NCPs . . . . .	209
<b>Related publications</b> . . . . .	211
IBM Redbooks . . . . .	211
Other resources . . . . .	211
Referenced Web sites . . . . .	212
How to get IBM Redbooks . . . . .	212
IBM Redbooks collections . . . . .	212
<b>Index</b> . . . . .	213



# 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 illustrates 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. 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 IBM's application programming interfaces.

## Trademarks

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

IBM®	OS/2®	Tivoli®
Redbooks™	OS/390®	TME®
Redbooks(logo)™ 	PAL®	VM/ESA®
AIX®	Parallel Sysplex®	VTAM®
APPN®	pSeries™	WebSphere®
CICS®	RACF®	xSeries™
CUA®	RS/6000®	z/Architecture™
DPI®	S/390®	z/OS™
ESCON®	S/390 Parallel Enterprise Server™	z/VM™
FICON™	SecureWay®	zSeries™
iSeries™	SP™	
MVS™	S/390 Parallel Enterprise Server™	

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

Lotus®	Notes®
Word Pro®	Domino™

The following terms are trademarks of other companies:

ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

# Preface

The Internet and enterprise-based networks have led to a rapidly increasing reliance upon TCP/IP implementations. The zSeries platform provides an environment in which critical business applications flourish. The demands placed on these systems is ever-increasing and such demands require a solid, scalable, highly available, and highly performing Operating System and TCP/IP component. z/OS and Communications Server for z/OS provide for such a requirement with a TCP/IP stack that is robust and rich in functionality. The *Communications Server for z/OS TCP/IP Implementation Guide* series provides a comprehensive, in-depth survey of CS for z/OS.

*Volume 4* covers in great detail the connectivity options available to CS for z/OS. In particular, we provide scenario-based descriptions of interface types such as Hipersockets, QDIO, MPC, and LCS. The main thrust of this redbook, however, is in the interconnection of networks as pertains to CS for z/OS. That is, we focus on the mechanism by which z/OS can interface with other networks via routing. We discuss static and dynamic routing, including RIP and OSPF, as well as VIPA routing considerations. We also cover interoperation with EIGRP.

Because of the varied scope of CS for z/OS, this volume is not intended to cover all aspects of this topic. The main goal of this volume is to provide an overview of the network interfaces available to CS for z/OS as well as the routing foundation necessary to interoperate in the complex internetworks of today. For more information, including applications available with CS for z/OS IP, please reference the other volumes in the series. These are:

- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- ▶ *OS/390 eNetwork Communications Server for V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840

## The team that wrote this redbook

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

**Adolfo Rodriguez** is a Senior I/T Specialist at the International Technical Support Organization, Raleigh Center. He writes extensively and teaches IBM classes worldwide on all areas of TCP/IP. Before joining the ITSO, Adolfo worked in the design and development of CS for z/OS, in RTP, NC. He holds a B.A. degree in Mathematics and B.S. and M.S. degrees in Computer Science from Duke University. He is currently pursuing a Ph.D. degree in Computer Science at Duke University, with a concentration on Networking Systems.

**Marcia Maria Fascini** is a Systems Specialist in Brazil. She has four years of experience in CICS and six years of experience in the networking field. She holds a degree in Mathematics from Fundação Santo André. Her areas of expertise include VTAM and TCP/IP networks.

**Giancarlo Rodolfi** is a zSeries FTSS for Latin America. He has 16 years of experience in the zSeries field. His areas of expertise include TCP/IP, z/VM, z/OS, UNIX System Services, CS for z/OS, Security, Linux, WebSphere Application Server, Firewall, TN3270 Server, and Domino. He has written extensively about CS for z/OS TCP/IP services and security.

**Heather Woods** is a Network Systems Programmer with IBM Strategic Outsourcing in the UK. She has eight years of experience in S/390 systems, and has spent the past four years working mainly with TCP/IP, CS for OS/390(z/OS), NCP and SNA.

Thanks to the following people for their contributions to this project:

Bob Haimowitz, Jeanne Tucker, Margaret Ticknor, Tamikia Barrow, Gail Christensen, Linda Robinson  
International Technical Support Organization, Raleigh Center

Barry Mosakowski, Jeff Haggar, Bebe Isrel, Van Zimmerman, Jerry Stevens, Tom Moore, Mike Fox  
Communications Server for z/OS Development, Raleigh, NC

Garth Madella  
IBM South Africa

Peter Focas  
IBM New Zealand

Octavio Ferreira  
IBM Brazil

Steve Zammit  
IBM Canada

Gwen Dente  
z/OS Server Software Technical Sales Support, Gaithersburg, MD

Rick Williams  
Cisco Systems, Inc.

## Special notice

This publication is intended to help customers implement and configure Communications Server for z/OS IP. The information in this publication is not intended as the specification of any programming interfaces that are provided by Communications Server for z/OS. See the PUBLICATIONS section of the IBM Programming Announcement for Communications Server for z/OS for more information about what publications are considered to be product documentation.

## Comments welcome

Your comments are important to us!

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

- ▶ Use the online **Contact us** review redbook form found at:  
[ibm.com/redbooks](http://ibm.com/redbooks)
- ▶ Send your comments in an Internet note to:  
[redbook@us.ibm.com](mailto:redbook@us.ibm.com)
- ▶ Mail your comments to the address on page ii.





# Introduction

Communications Server for z/OS IP is the second phase of the z/OS TCP/IP evolution towards using z/OS UNIX System Services. Today's CS for z/OS IP enjoys improved performance due to a redesigned stack, one that takes advantage of Communications Storage Management (CSM) and of VTAM's Multi-Path Channel (MPC) and Queued Direct I/O (QDIO) capabilities. This tight coupling with VTAM provides enhanced performance and serviceability. In CS for z/OS IP, two worlds converge, providing access to z/OS UNIX System Services and the traditional MVS environment via network attachments, both old and new.

Because CS for z/OS IP provides an implementation of the TCP/IP family of protocols for the z/OS platform, it implements and interfaces with a large number of network connectivity environments such as ATM and Gigabit Ethernet. To complement this, CS for z/OS IP supports industry standard routing protocols and configuration mechanisms.

In this chapter, we outline the functional capabilities of CS for z/OS IP in this space. Additionally, we cover basic routing concepts that are necessary to understand and implement the advanced routing features included with the product.

# 1.1 Network interfaces

Figure 1-1 illustrates the complex environment of Communications Server for z/OS IP. Because CS for z/OS IP is the marriage between a traditional MVS environment and the industry standard UNIX environment, the product delivers a number of Application Programming Interfaces (APIs). As a result, many of the products provided for the z/OS environment use different APIs, thereby complicating the picture. In addition, requirements for z/OS connectivity to different networks is largely varied, leading to a large number of configuration options in this regard.

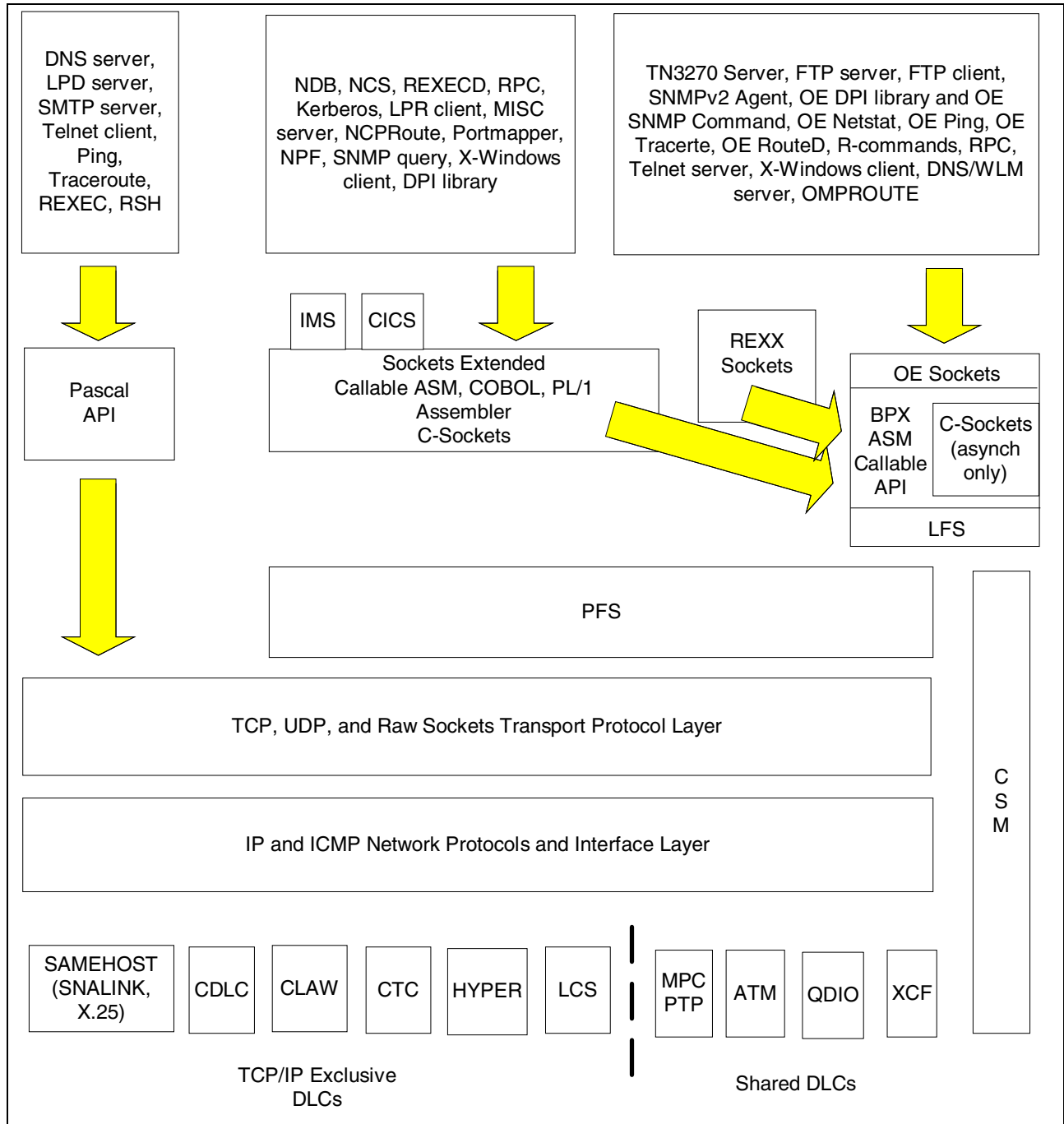


Figure 1-1 Communications Server for z/OS IP stack overview



## 1.1.1 CS for z/OS IP supported devices

To interface with the varied scope of network types, CS for z/OS IP implements devices (or device drivers) that are generally referred to as *Data Link Controls* (DLCs). In CS for z/OS IP, these can be classified into two categories: TCP exclusive DLCs and shared DLCs.

### TCP exclusive DLCs

TCP exclusive DLCs are only available for the CS for z/OS IP stack and cannot be shared between multiple instances of CS for z/OS IP. The TCP exclusive DLCs supported by CS for z/OS IP include the following channel protocols:

- ▶ Channel Data Link Control (CDLC)

This protocol supports a native IP connection between CS for z/OS IP and an IP router coded within a 374x running Network Control Program (NCP) or within a 3746 9x0 channel-attached router.
- ▶ Common Link Access to Workstation (CLAW)

This protocol is used to connect the CS for z/OS IP to a 3172 running ICCP, an RS/6000, and Cisco routers supporting this interface.
- ▶ Channel-to-Channel (CTC)

This protocol is supported between two CS for z/OS IP systems and uses one read/write channel pair. Both parallel and ESCON channels are supported.
- ▶ Hyperchannel  

This protocol is used to connect via the NSC A220 Hyperchannel Adapter and its descendants.
- ▶ LAN Channel Station (LCS)  

This protocol is used by OSA, the 3172 running ICP, the 2216, and the 3746-9x0 MAE.

In addition, one more TCP/IP exclusive DLC protocol exists, although it does not make use of zSeries channels. Not to be confused with PTP Samehost, the SAMEHOST DLC enables communication between CS for z/OS IP and other servers running on the same MVS image. In the past, this communication was provided by IUCV. Currently, three such servers exploit the SAMEHOST DLC:

- ▶ SNALINK LU0  

This server provides connectivity through SNA networks using LU0 traffic. It acts as an application to z/OS VTAM.
- ▶ SNALINK LU6.2  

This server provides connectivity through SNA networks using LU6.2 traffic. It also acts as an SNA application to z/OS VTAM.
- ▶ X.25  

This server provides connectivity to X.25 networks by using the NCP packet switching interface (NPSI).

## Shared DLCs

Shared DLCs are those that can be simultaneously used by multiple instances of multiple protocol stacks. For example, a shared DLC may be used by one or more instances of CS for z/OS IP and one or more instances of z/OS VTAM. These shared DLCs include:

### ► Multipath Channel+ (MPC+)

MPC+ is an enhanced version of the Multipath Channel (MPC) protocol. It allows for the efficient use of multiple read and write channels. High Performance Data Transfer (HPDT) uses MPC+ together with Communication Storage Manager (CSM) to decrease the number of data copies required to transmit data. This type of connection can be used in two ways. The first of these is called MPCPTP, in which CS for z/OS IP is connected to a peer IP stack in a point-to-point fashion. In this way, CS for z/OS IP can be connected to each of the following:

- Another CS for z/OS IP stack
- 2216
- RS/6000
- 3746-9x0 MAE
- Cisco routers via the Cisco Channel Interface Processor (CIP) or the Cisco Channel Port Adapter (CPA)

The second way to use MPC+ is to connect to an Open Systems Adapter (OSA). In this configuration, OSA acts as an extension of the CS for z/OS IP stack and not as a peer IP stack as in MPCPTP. The following are supported in this manner:

- OSA-2 native ATM (RFC1577)
- OSA-2 Fast Ethernet and FDDI (MPCOSA)
- OSA-Express QDIO uses MPC+ for the exchange of control signals between CS for z/OS IP and the OSA-Express

### ► MPCIPA (QDIO)

The OSA-Express provides a mechanism for communication called Queued Direct I/O (QDIO). Although it uses the MPC+ protocol for its control signals, the QDIO interface is quite different from channel protocols. It uses Direct Memory Access (DMA) to avoid the overhead associated with channel programs. The OSA-Express and CS for z/OS IP support Gigabit Ethernet, Fast Ethernet, Fast Token Ring, and ATM LAN emulation.

A partnership between CS for z/OS IP and the OSA-Express Adapter provides offload of compute-intensive functions from the zSeries to the adapter. This interface is called IP Assist (IPA). Offloading reduces zSeries cycles required for network interfaces and provides an overall improvement in the OSA-Express environment compared to existing OSA-2 interfaces.

### ► XCF

The XCF DLC allows communication between multiple CS for z/OS IP stacks within a Parallel Sysplex via the Cross-System Coupling Facility (XCF). The XCF DLC can be defined, as with traditional DLCs, but it also supports XCF Dynamics, in which the XCF links are brought up automatically.

### ► PTP Samehost

Sometimes referred to as IUTSAMEH, this connection type is used to connect two or more CS for z/OS IP stacks running on the same MVS image. In addition, it can be used to connect these CS for z/OS IP stacks to z/OS VTAM for the use of Enterprise Extender.

- ▶ Hipersockets

Hipersockets provides very fast TCP/IP communication between servers running in different Logical Partitions (LPARs) on a z800 or z900 CEC. The communication is through processor system memory via Direct Memory Access (DMA). The virtual servers that are connected in this manner form a virtual LAN. Hipersockets uses internal QDIO at memory speeds to pass traffic between virtual servers.

## 1.1.2 Connectivity restrictions

Certain DLCs are no longer considered strategic for the networking environment of today and are therefore no longer supported in the OS/390 V2R5 IP (and later) stack:

- ▶ Offload for 3172

There are no plans to support 3172 offload in any future releases. The improved performance and the reduced CPU cycle provided by the new TCP/IP stack should substantially reduce or eliminate offload for most environments.

- ▶ High Performance Parallel Interface (HiPPI)

- ▶ Continuously Executing Transfer Interface (CETI)

## 1.2 Routing with CS for z/OS IP

In order to provide for the *interconnection* of networks, Communications Server for z/OS IP provides routing capabilities. This section introduces the concept of routing and describes the z/OS-supported routing functionality.

### 1.2.1 What is routing?

One of the basic functions provided by the IP protocol is the ability to form connections between different physical networks. A system that performs this function is termed an *IP router*. This type of device attaches to two or more physical networks and forwards datagrams between the networks.

When sending data to a remote destination, a host passes datagrams to a local router. The router forwards the datagrams toward the final destination. They travel from one router to another until they reach a router connected to the destination's LAN segment. Each router along the end-to-end path selects the *next hop* device used to reach the destination. The next hop represents the next device along the path to reach the destination. It is located on a physical network connected to this intermediate system. Since this physical network differs from the one on which the system originally received the datagram, the intermediate host has *forwarded* (that is, routed) the IP datagram from one physical network to another.

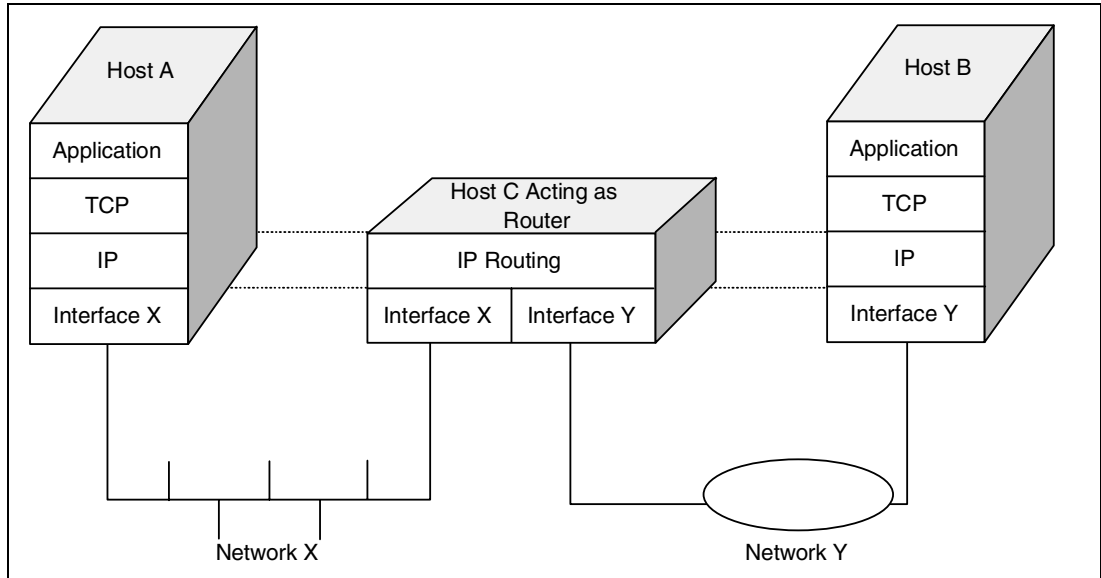


Figure 1-2 IP routing operations

Figure 1-2 shows an environment where Host C is positioned to forward packets between Network X and Network Y.

### The routing table

A table known as the *routing table* in each device is used to forward packets between network segments. The basic table contains information about a router's locally connected networks. The configuration of the device can be extended to contain information detailing remote networks. This information provides a more complete view of the overall environment.

A robust routing protocol provides the ability to dynamically build and manage the information in the IP routing table. As network topology changes occur, routing tables are updated with minimal or no manual intervention. There are a number of routing protocols implementing algorithms designed to maintain consistent and current routing tables within intra- and Internetworks, such as Routing Information Protocol (RIP) and Open Shortest Path First (OSPF). These dynamic routing protocols overcome the manual intervention limitations of traditional static routing.

For additional information, please refer to Chapter 4, "Routing overview" on page 69.

## 1.2.2 Routing in a z/OS environment

The z/OS environment is rich in application heterogeneity and critical in today's networked enterprises. Because of this, a number of functions available in the z/OS system must interface with routing infrastructures. For example, Distributed VIPAs provide the ability to move a Virtual IP Address (VIPA) from one stack in a sysplex to another. Such a move must be reflected in current routing tables through the network fabric. What results is an interesting interaction between routing protocols and takeover mechanisms that provide such attractive functionality. For more information on routing considerations for Dynamic VIPA, please consult Chapter 7, "Routing with VIPA" on page 141.

CS for z/OS IP once provided an interface for the coding of static routes that was cumbersome and non-standard. Though this GATEWAY statement of the TCP/IP profile is still supported, a much more efficient static route specification is included. The BEGINROUTES statement provides an interface that adheres to de facto industry standards. For more on this topic, please refer to Chapter 5, “Working with static routes” on page 97.

In addition, CS for z/OS IP provides a number of routing daemons, applications which use the stack to communicate with their partner applications on other stacks to ensure routing table consistency and currency. We summarize these here, but provide much more detailed information in Chapter 6, “Dynamic routing with OMPROUTE” on page 103 and Appendix A, “Working with ORouted” on page 181.

### 1.2.3 z/OS-supported routing daemons

CS for z/OS IP ships two routing applications, ORouted and OMPROUTE. OMPROUTE and ORouted cannot run on the same TCP/IP stack concurrently. These applications add, delete, and change routing entries in the routing table and can be used as an alternative to static routes created via GATEWAY or BEGINROUTES definitions in the CS for z/OS IP profile.

#### **ORouted**

ORouted is the CS for z/OS IP implementation of the Routing Information Protocol (RIP) Version 1 (RFC 1058) and RIP Version 2 (RFC 1723). A much older application than OMPROUTE, ORouted has limitations. ORouted does not support zero subnets. In addition, ORouted does not support equal-cost multipath routes to a destination network or host. As a result, OMPROUTE is the recommended routing application (also called routing daemon).

#### **OMPROUTE**

In OS/390 V2R6 IP and later, OMPROUTE implements the Open Shortest Path First (OSPF) protocol described in RFC 1583 (OSPF Version 2) as well as RIP V1 and RIP V2. When configured properly, the OS/390 host running with OMPROUTE becomes an active OSPF and/or RIP router in a TCP/IP network. Either (or both) of these two routing protocols can be used to dynamically maintain the host routing table. Additionally, OS/390 V2R7 IP provided an OMPROUTE subagent that implements the OSPF MIB variable containing OSPF protocol and state information for SNMP. This MIB variable is defined in RFC 1850.

## 1.3 Communications Server for z/OS V1R2 IP enhancements

In this section, we outline the enhancements included with Communications Server for z/OS V1R2 IP relevant to this volume. For enhancements in all other areas, please consult the other V1R2 versions of each volume in the series:

- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840

### 1.3.1 Hipersockets support

High speed, low latency, inter-LPAR IP interface using extensions to the Queued Direct I/O (QDIO) interface provides communications between instances of z/OS and Linux/390 called internal QDIO (iQDIO). The iQDIO interface is tightly integrated into the system to provide users with a high speed “logical” LAN with minimal system and network overhead. It eliminates the need to utilize I/O subsystem operations and the use of an OSA network adapter.

### 1.3.2 64-bit real memory exploitation

The Communication Storage Manager (CSM) has been enhanced to exploit real storage above the 2 gigabyte range when running on Freeway processors configured with more than 2 gigabytes of real storage. CSM Data Space storage is used by both SNA and IP stacks for inbound and outbound application data traffic. In CS for z/OS V1R2, enhancements are made to allow CSM Data Space virtual storage to be backed by real storage that resides above 2 Gigabytes and is 64-bit addressable. Real storage below the 2 Gigabyte bar, previously used by CSM for SNA and IP communications, is now available for other application use. In order to derive benefits without introducing any additional data moves that would degrade throughput, device drivers that currently perform network I/O directly from CSM data space buffers have been enhanced to allow them to perform I/O from 64-bit addressable real storage. These include the QDIO (Queued Direct I/O), iQDIO (internal Queued Direct I/O), MPC (MultiPath Channel) and ATM (Asynchronous Transfer Mode) device drivers. These enhancements are automatically enabled when running on Freeway processors, requiring no explicit actions to realize these benefits.

### 1.3.3 IP routing enhancements

CS for z/OS V1R2 provides IP routing enhancements for installations using dynamic routing protocols and the IBM recommended routing daemon (OMPROUTE) as well as automated migration assistance from the ORouted routing daemon to OMPROUTE.

Several enhancements are available to installations using dynamic routing protocols with OMPROUTE. These include the following:

- ▶ The ability to configure static routes while also using dynamic routing protocols has been enhanced. Previously, when static routes were configured, they always took precedence over any dynamic routes discovered even when the dynamic routes were more optimal than the static routes. In this release, a new type of static route is introduced, referred to as a *replaceable* static route. When configured in a dynamic routing protocol environment, replaceable static routes act as backup routes; they are only used by TCP/IP when no other route is discovered by the dynamic routing protocols. Replaceable static routes can be used to provide better fault tolerance in cases where there is a failure in discovering routes using dynamic routing protocols. Replaceable static routes are only available using the OMPROUTE routing daemon.
- ▶ OMPROUTE can now coexist on networks with RIP routers running both the RIP V1 and RIP V2 routing protocols. This enhancement allows OMPROUTE to receive RIP V1 and RIP V2 route updates sent by these routers running each of these routing protocols. This enhancement enables migration from RIP V1 to RIP V2 in a gradual manner instead of using an “all or nothing” approach.
- ▶ There is enhanced security when using Open Shortest Path First (OSPF) dynamic routing protocol. In an OSPF environment, users can now specify Message Digest (MD5) authentication of OSPF updates. When this option is configured, the OMPROUTE routing daemon will verify that routing updates are only received from authenticated routers within

the network. This allows z/OS to be part of the same OSPF area as other routers that also support MD5 authentication (as specified in RFC 2328).

- ▶ A conversion tool is now available that provides assistance for users of the ORouteD routing daemon in migrating to the OMPROUTE. OMPROUTE offers significant advantages over the ORouteD daemon, such as being able to support not only the same routing protocols as ORouteD (RIP V1 and RIP V2) but also newer and more efficient routing protocols such as OSPF. Users of the ORouteD routing daemon are encouraged to migrate to the OMPROUTE daemon, as future routing enhancements on z/OS will not be provided for ORouteD. The conversion tool provided can greatly simplify this task and reduce the migration effort. It gives you the ability to convert existing ORouteD configuration files into configuration files that can be used by the OMPROUTE daemon.







## **z/OS interface environment**

Communications Server for z/OS provides an interface environment that leverages functional similarities among device drivers to provide for protocol implementations that are efficient and robust. Specifically, the VTAM component of CS for z/OS achieves high performance through the use of services such as the Communication Storage Manager (CSM). This chapter presents CSM and provides insight into how VTAM represents network interfaces.

## 2.1 CSM

The Communications Storage Manager (CSM) is a component of VTAM that allows authorized host applications to share data with VTAM and other CSM users without having to physically copy the 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, CPU resources (CPU cycles, memory bus, and cache) are conserved.

CSM includes an application programming interface (API) that allows users to obtain and return CSM buffers, change ownership of buffers, copy buffers and perform other functions related to CSM buffer management. Users set up and access data that resides in CSM buffers. These buffers are obtained from buffer pools that are identified by their buffer size and storage type according to Table 2-1.

Table 2-1 Buffer pools in CSM

Storage Types	Buffer Sizes				
31-bit backed data space	4K	16K	32K	60K	180K
64-bit backed data space	4K	16K	32K	60K	180K
ECSA	4K	16K	32K	60K	180K

CSM is an integral component of z/OS and is shipped as one of the base z/OS components. However, many CSM functions are independent of VTAM. CSM storage limits and tuning parameters are defined in the CSM parmlib member, IVTPRM00. CSM is initialized by the first request to create a pool of buffers and remains active for the life of the system. Upon initialization, CSM reads the CSM parmlib member to determine storage limits and buffer pool related values. Once CSM is initialized, it persists for the life of the system.

System operators can monitor CSM storage by using the **DISPLAY CSM** command.

In z/OS V1R2, TCP/IP exploits real storage in excess of 2 gigabytes by allowing z/OS to back most fixed CSM data space pages on or above the 2 gigabyte real storage bar. This enhances performance when z/OS V1R2 Communications Server is executing in z/Architecture mode.

CSM data space will be backed by 64-bit real storage when the machine is in z/Architecture mode. This storage may be passed to applications. If an application accepts CSM data space and determines that the data should be saved on external media, the data is usually copied to primary storage then passed to an access method.

The 64-Bit Real Addressing Support function is automatically enabled at initialization time for z/OS V1R2 Communications Server when the system is executing in z/Architecture mode. Although this function cannot be disabled, you can control whether or not your application programs are passed 64-bit backed storage by using the new API64R VTAM start option.

CSM data space storage is used by both SNA and IP stacks for inbound and outbound application data traffic. Device drivers that currently perform network I/O directly from CSM data space buffers are also being enhanced to be able to perform I/O from 64-bit addressable real storage. These include the QDIO (Queued Direct I/O), IQDIO (Internal Queued Direct I/O), HPDT MPC (High Performance Data Transport MultiPath Channel) and ATM device drivers.

## 2.2 Displaying TCP/IP device resources in VTAM

The device drivers for z/OS Communications Server IP are provided by VTAM. The LCS, CLAW, CDLC, and CTC device support is for the exclusive use of z/OS Communications Server IP. The MPC device support is shared between CS for z/OS IP and VTAM. When CS for z/OS IP devices are activated, there must be an equivalent Transport Resource List Element (TRLE) defined to VTAM. The devices that are exclusively used by z/OS Communications Server IP have TRLEs that are automatically generated for them. The MPC device must have a TRL major node defined with the TRLE representing the MPC device.

Since the device driver resources are provided by VTAM, you have the ability to display the resources using VTAM display commands. See Figure 2-1 on page 13 to view the command to display all the TRLEs known to VTAM.

For dynamically generated TRLEs, the device type and address can be decoded from the generated TRLE name. The format is IUTtaaaa:

- ▶ IUT - Fixed for all dynamically generated TRLEs.
- ▶ t - Shows the device type.
  - C - Indicates this is a CDLC device.
  - H - Indicates this is a HYPERCHANNEL device.
  - I - Indicates this a QDIO device.
  - L - Indicates this is a LCS device.
  - S - Indicates this is a SAMEHOST device.
  - W - Indicates this is a CLAW device.
  - X - Indicates this is a CTC device.
- ▶ aaaa - The read device number. For SAMEHOST connections, this is just a sequence number.

For XCF links, the format of the TRLE name is ISTTxxyy. ISTT is fixed, xx is the SYSCLONE value of the originating VTAM and yy is the SYSCLONE value of the destination VTAM.

```
D NET,TRL
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = TRL 141
IST1314I TRLE = IUTSAMEH STATUS = ACTIV CONTROL = MPC
IST1314I TRLE = IUTL2062 STATUS = ACTIV CONTROL = TCP
IST1314I TRLE = IUTL2020 STATUS = ACTIV CONTROL = TCP
IST1314I TRLE = IUTL2026 STATUS = ACTIV CONTROL = TCP
IST1314I TRLE = IUTL2060 STATUS = ACTIV CONTROL = TCP
IST1314I TRLE = M392216B STATUS = NEVAC CONTROL = MPC
IST1314I TRLE = M032216B STATUS = ACTIV CONTROL = MPC
IST1314I TRLE = MPC1000 STATUS = INACT CONTROL = MPC
IST1314I TRLE = ISTT0339 STATUS = ACTIV CONTROL = XCF
IST1314I TRLE = ISTT0328 STATUS = ACTIV CONTROL = XCF
IST1314I TRLE = RA03TR20 STATUS = ACTIV CONTROL = MPC
IST1454I 11 TRLE(S) DISPLAYED
IST314I END
```

Figure 2-1 VTAM display of TRLEs for TCP/IP devices

If you have multiple CS for z/OS IP stacks running, you may display the TRLEs associated to a specific stack. See Figure 2-2 for the command to display the TRLEs for a specific TCP/IP stack. The Upper Layer Protocol ID (ULPID) references the TCP/IP stack name.

```
D NET,TRL,CONTROL=TCP,ULPID=TCPIPA
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = TRL 156
IST1314I TRLE = IUTL2026 STATUS = ACTIV CONTROL = TCP
IST1314I TRLE = IUTL2060 STATUS = ACTIV CONTROL = TCP
IST1454I 2 TRLE(S) DISPLAYED
IST314I END
```

Figure 2-2 VTAM display of TRLEs for a specific TCP/IP stack

You may display the status of the specific TRLE (see Figure 2-3 for this command).

```
D NET,TRL,TRLE=IUTL2026
IST097I DISPLAY ACCEPTED
IST075I NAME = IUTL2026, TYPE = TRLE 223
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
IST087I TYPE = LEASED , CONTROL = TCP , HPDT = *NA*
IST1717I ULPID = TCPIPA
IST1221I READ DEV = 2026 STATUS = ACTIVE STATE = N/A
IST1221I WRITE DEV = 2027 STATUS = ACTIVE STATE = N/A
IST314I END
```

Figure 2-3 VTAM display of a specific TRLE

If you have any HPDT devices, you may display the TRLEs that are related to HPDT. See Figure 2-4 for the command to display TRLEs related to HPDT.

```
D NET,TRL,CONTROL=MPC
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = TRL 226
IST1314I TRLE = IUTSAMEH STATUS = ACTIV CONTROL = MPC
IST1314I TRLE = M392216B STATUS = NEVAC CONTROL = MPC
IST1314I TRLE = M032216B STATUS = ACTIV CONTROL = MPC
IST1314I TRLE = MPC1000 STATUS = INACT CONTROL = MPC
IST1314I TRLE = RA03TR20 STATUS = ACTIV CONTROL = MPC
IST1454I 5 TRLE(S) DISPLAYED
IST314I END
```

Figure 2-4 VTAM display of TCP/IP MPC device status

You may also display the status of the specific TRLE associated with HPDT. See Figure 2-5 for the display of a specific HPDT device.

```

D NET,TRL,TRLE=M032216B
IST097I DISPLAY ACCEPTED
IST075I NAME = M032216B, TYPE = TRLE 229
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
IST087I TYPE = LEASED           , CONTROL = MPC , HPDT = YES
IST1715I MPCLEVEL = HPDT       MPCUSAGE = SHARE
IST1717I ULPID = TCPIPC
IST1577I HEADER SIZE = 4096 DATA SIZE = 32 STORAGE = ***NA***
IST1221I WRITE DEV = 0380 STATUS = ACTIVE     STATE = ONLINE
IST1577I HEADER SIZE = 4092 DATA SIZE = 32 STORAGE = DATASPACE
IST1221I READ  DEV = 0381 STATUS = ACTIVE     STATE = ONLINE

```

*Figure 2-5 VTAM display of TRLE for MPC device*

If you have any XCF devices, you may display the XCF TRLEs. See Figure 2-6 for the command to display all XCF TRLEs and Figure 2-7 for a specific XCF TRLE.

```

D NET,TRL,CONTROL=XCF
IST097I DISPLAY ACCEPTED
IST350I DISPLAY TYPE = TRL 232
IST1314I TRLE = ISTT0339 STATUS = ACTIV      CONTROL = XCF
IST1314I TRLE = ISTT0328 STATUS = ACTIV      CONTROL = XCF
IST1454I 2 TRLE(S) DISPLAYED
IST314I END

```

*Figure 2-6 VTAM display of XCF devices*

```

D NET,TRL,TRLE=ISTT0339
IST097I DISPLAY ACCEPTED
IST075I NAME = ISTT0339, TYPE = TRLE 235
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
IST087I TYPE = LEASED           , CONTROL = XCF , HPDT = *NA*
IST1715I MPCLEVEL = HPDT       MPCUSAGE = SHARE
IST1717I ULPID = TCPIPC
IST1717I ULPID = ISTP0339
IST1503I XCF TOKEN = 0300002100160002     STATUS = ACTIVE
IST1502I ADJACENT CP = USIBMRA.RA39M
IST314I END

```

*Figure 2-7 VTAM display of TRLE for XCF device*



## Network interface types

In this chapter, we look at some of the devices that can be defined and used in z/OS Communications Server IP to connect the host to your IP network. Most of the sample configurations use dynamic routing definitions. This does not imply that they can only be used with dynamic routing; static routing is equally supported. You may find more information about static and dynamic routing and the parameters from the device definition that could affect your routing program in Chapter 5, “Working with static routes” on page 97 and Chapter 6, “Dynamic routing with OMPROUTE” on page 103.

Most of the sample configurations use 3-digit device addressing, although 4-digit device numbers for network attachment devices are fully supported. Dynamic Unit Control Blocks (UCBs) and UCBs that reside above the 16 MB address are also supported by z/OS CS IP. These UCB characteristics are defined via HCD with DYNAMIC=YES and LOCANY=YES.

CS for z/OS IP provides a wide range of options for connecting your z/OS TCP/IP system, as shown in Table 3-1.

*Table 3-1 Device attachments*

Device	Adapter or peer
ATM	ATM network via OSA-2 or OSA-Express in ATM Native mode
CDLC	3745/3746 network via NCP, 3746/9x0
CLAW	RISC 6000, IBM pSeries servers, CISCO CIP or CPA
CTC	z/OS via channel-to-channel adapter
Hipersockets	Another TCP/IP within the same CEC
Hyperchannel	Hyperchannel hardware (AT&T NSCA220)
LCS	IBM 8232 LAN Channel Station (LCS) device, IBM 3172 Interconnect Controller, IBM 2216 Multiaccess Connector Model 400, IBM FDDI, OSA, OSA-2 (Ethernet, Token Ring, ATM LAN emulation mode)
MPCIPA	OSA-Express (Gigabit Ethernet, Fast Ethernet, High-speed Token Ring, ATM155)
MPCOSA	OSA-2 Fast Ethernet, OSA-2 FDDI

Device	Adapter or peer
MPCPTP	zSeries, iSeries, xSeries, Cisco CIP or CPA
XCF	Another TCP/IP within the same z/OS sysplex
IUTSAMEH	Another TCP/IP on the same z/OS, VTAM for Enterprise Extender
SNA LU 0	SNA network via SNALINK LU 0 appl on same z/OS
SNA LU 6.2	SNA network via SNALINK LU 6.2 appl on same z/OS
X25 NPSI	X.25 network via X.25 appl on same z/OS

**Note:** With CS for z/OS IP, you do not need to specify a missing interrupt handler (MIH) value for any READ subchannels on devices defined to TCP/IP, as the MIH is automatically configured OFF for READ subchannels. An MIH value should be set for the WRITE subchannel for some device types, including ATM, MPCPTP, MPCOSA, MPCIPA, LCS, CLAW, CTC and CDLC. See *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 for more information.

Some of the devices no longer supported are:

- ▶ DDN1822
- ▶ CETI
- ▶ CLAW Offload
- ▶ HIPPI
- ▶ IUCV/PVMIUCV (IUCV DLC support is replaced by SAMEHOST DLC support)
- ▶ X25ICA



## 3.1 ATM

Asynchronous Transfer Mode (ATM) technology is based on powerful, yet flexible concepts:

- ▶ When information needs to be communicated, the sender transmits a "requested path" with the network for a connection to the destination. When setting up this connection, the sender specifies the type, speed and other attributes of the call, which determine the end-to-end quality of service. An analogy for this request of qualities would be determining a method of delivery using the US mail. One can choose to send first class, overnight, two-day delivery, etc. and ask for certified mail.
- ▶ Another key concept is that ATM is a switched-based technology. By providing connectivity through a switch (instead of shared media such as a LAN), several benefits are provided:
  - Dedicated bandwidth per connection
  - Higher aggregate bandwidth
  - Well-defined connection procedures
  - Flexible access speeds
  - Increased network efficiency due to switched versus router protocols

An ATM network is comprised of two types of nodes. The ATM endpoint is a piece of end-user equipment that interfaces to an ATM network in a native way. An endpoint sends and receives ATM cells on link connections defined by ATM standards. The ATM switch routes connection requests and data from endpoints to ATM switches or endpoints. ATM switches are usually classified as either "private ATM switches" or "public ATM switches". Private ATM networks use a "private" ATM network address format. A "local" switch in a private network takes a remote destination address and processes according to the destination.

- ▶ If the destination is for this network, it sends the call to the destination.
- ▶ If the destination is not for this network, it resolves the public gateway through which to go and routes the call to the public network.

Using ATM, information to be sent is segmented into fixed-length cells, transported to and re-assembled at the destination. The ATM cell has a fixed length of 53 bytes; the fixed length allows the information to be transported in a predictable manner. This predictability accommodates different traffic types on the same network.

The cell is broken into two main sections: the header and the payload. The payload (48 bytes) is the portion that carries the actual information: either voice, data or video. The header (5 bytes) is the addressing mechanism.

### 3.1.1 Native ATM or classic IP

Native ATM or Classic IP support was introduced with OS/390 eNetwork Communications Server V2R5. LAN Emulation was already supported previously and has to be defined as an LCS device.

z/OS CS IP supports data transmission to members of an ATM network over an ATM Virtual Circuit (VC). z/OS CS IP implementation supports only best effort virtual circuits. HPR supports both best effort and reserved bandwidth connections. Support extends beyond LAN Emulation; native definitions of ATM addresses do not require definitions of LAN addresses. z/OS Communications Server IP uses the IBM Open Systems Adapter-2 (OSA-2) to enable this support.

There are two types of VCs, both of which can be used for either HPR or TCP/IP:

► Permanent Virtual Circuit (PVC)

A permanent (predefined) connection between two ATM endpoints. These are conceptually "leased" connections. For a PVC, two nodes "subscribe" to the network. A channel number is reserved for communication between the two nodes. The channel is always assigned to the nodes whether in use or not. Both nodes agree upon information relating to the PVC such as ATM addresses, quality of service parameters, and the carrier. For CS for z/OS IP, the subscribed PVC connection is activated when TCP/IP starts the ATM OSA2 device and will remain active until the device is stopped.

► Switched Virtual Circuit (SVC)

A dynamic connection between two ATM endpoints. These are activated when needed and deactivated when not needed. The calls across SVCs are charged based on the connect time and the amount of data traffic across the connection. The initiator (caller) of the connection supplies the remote ATM address, quality of service parameters such as user cell rates, traffic type, QOS class, and transmit network selector.

For IP, SVCs use an ATMARP server. You can also use the TRANSLATE statement to manually define the remote ATM address, but most ATM switches include the ATMARP server function which will perform the address translation.

An Open Systems Adapter-2 (OSA-2) and an ATM switch are required for ATM connectivity. TCP and HPR can both use the same OSA-2. However, they cannot share the same virtual circuit. The same TRLE used to define an OSA-2 adapter for HPR is shared and used for TCP/IP.

### 3.1.2 ATM configuration example

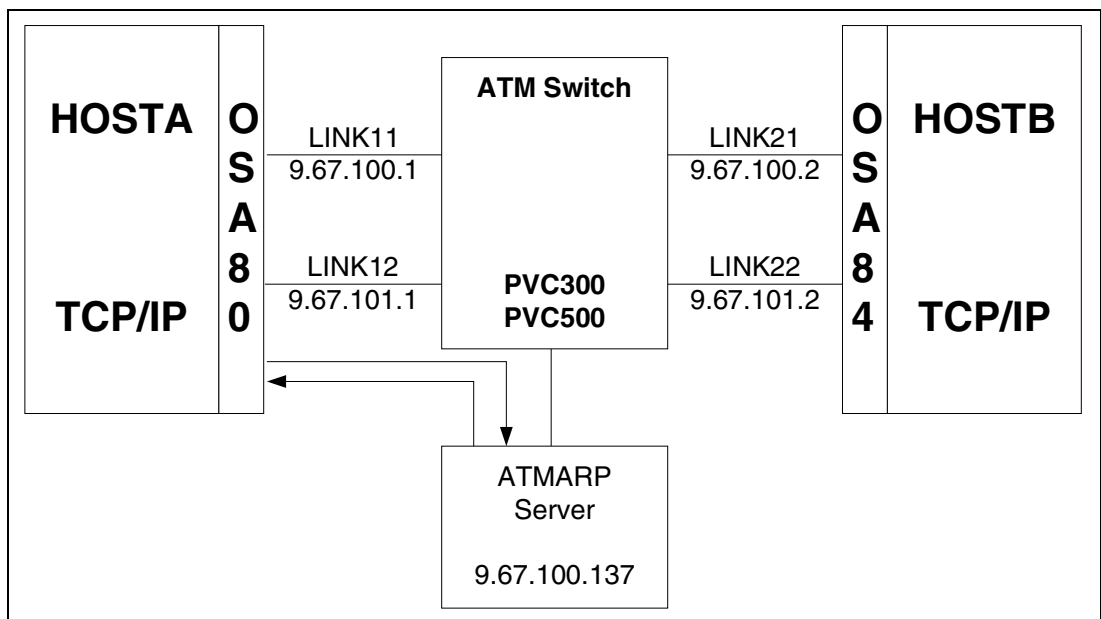


Figure 3-1 Sample ATM environment

Figure 3-2 shows the TRLE definition for the ATM device that we defined in VTAM. The TRLE name **1** must be the same name used in configuring the ATM OSA device in OSA/SF.







and we received the following messages:

*Example 3-2 Resulting messages*

```
V TCP/IP,T03ATCP,OBEY,TCP.TCPPARMS(OBEY03)
EZZ0060I PROCESSING COMMAND: VARY TCP/IP,T03ATCP,OBEY,
        TCP.TCPPARMS(OBEY03)
EZZ0300I OPENED OBEYFILE FILE 'TCP.TCPPARMS(OBEY03)'
EZZ0309I PROFILE PROCESSING BEGINNING FOR 'TCP.TCPPARMS(OBEY03)'
EZZ0327I ATMARPSV ARPSV1 ON LINE 2 IS ALREADY DEFINED
EZZ0316I PROFILE PROCESSING COMPLETE FOR FILE 'TCP.TCPPARMS(OBEY03)'
EZZ0053I COMMAND VARY OBEY COMPLETED SUCCESSFULLY
```

Based on **1**, we assumed that the ATMARP server should have some definitions in it, but the device status command shows otherwise, as you can see from Figure 3-5.

```
D TCP/IP,T03ATCP,N,DEVL
EZZ2500I NETSTAT CS V2R7 T03ATCP 533
DEVNAME: LOOPBACK          DEVTYPE: LOOPBACK  DEVNUM: 0000
LNKNAME: LOOPBACK          LNKTYPE: LOOPBACK  STATUS: READY
:
DEVNAME: OSARA03           DEVTYPE: ATM       DEVNUM: 0000
LNKNAME: LINK11           LNKTYPE: ATM       STATUS: NOT ACTIVE
NETNUM: 0  QUESIZE: 0  BYTEIN: 0000032524  BYTEOUT: 0000051776
BSD ROUTING PARAMETERS:
MTU SIZE: 00576           METRIC: 01
DESTADDR: 0.0.0.0         SUBNETMASK: 255.255.255.0
ATM SPECIFIC:
ATM PORTNAME: OSA80
ATM LIS NAME: LIS1
SUBNETVALUE: 9.67.100.0   SUBNETMASK: 255.255.255.0
DEFAULTMTU: 0000009180   INACTVTIMEOUT: 0000000000
MINHOLDTIME: 0000000000  MAXCALLS: 0000001000
CACHENTRYAGE: 0000000300  ATMARPRETRY: 0000000006
ATMARPTIMEOUT: 0000000004  PEAKCELLRATE: 0000353207
NUMOFSVCS: 0000000000
MULTICAST SPECIFIC:
MULTICAST CAPABILITY: NO
```

*Figure 3-5 ARTMAP server definitions missing*

Notice the absence of the ATMARP Server configuration in **2**.

It was only when we ran another **VARY OBEYFILE** command using the same config file that we got the definition for the ATMARP server correctly, as shown in **3** of Figure 3-6.

```

D TCPIP, T03ATCP, N, DEVL
EZZ2500I NETSTAT CS V2R7 T03ATCP 533
DEVNAME: LOOPBACK          DEVTYPE: LOOPBACK  DEVNUM: 0000
LNKNAME: LOOPBACK          LNKTTYPE: LOOPBACK  STATUS: READY
:
DEVNAME: OSARA03           DEVTYPE: ATM        DEVNUM: 0000
LNKNAME: LINK11           LNKTTYPE: ATM       STATUS: NOT ACTIVE
NETNUM: 0  QUESIZE: 0     BYTEIN: 0000032524  BYTEOUT: 0000051776
BSD ROUTING PARAMETERS:
MTU SIZE: 00576           METRIC: 01
DESTADDR: 0.0.0.0         SUBNETMASK: 255.255.255.0
ATM SPECIFIC:
ATM PORTNAME:  OSA80
ATM LIS NAME:   LIS1
SUBNETVALUE:   9.67.100.0  SUBNETMASK:  255.255.255.0
DEFAULTMTU:    0000009180  INACTVTIMEOUT: 0000000000
MINHOLDTIME:   0000000000  MAXCALLS:    0000001000
CACHENTRYAGE:  0000000300  ATMARPRETRY: 0000000006
ATMARPTIMEOUT: 0000000004  PEAKCELLRATE: 0000353207
NUMOFSVCS:    0000000000
ATMARPSV NAME: ARPSV1     3
VCTYPE:       SVC         ATMADDRTYPE:  NSAP
ATMADDR:      399999999999990000999901014000821000010A
IPADDR:       9.67.100.137
MULTICAST SPECIFIC:
MULTICAST CAPABILITY: NO

```

Figure 3-6 Display of the ATM device with correct definitions

### Redefining the ATM logical IP subnet

The ATMLIS redefinition can also be performed through the **VARY OBEYFILE** command. You must ensure that the ATM devices are restarted after changing TCP/IP's ATMLIS definition. We discovered that, although the definition has been changed, as confirmed by the **D TCPIP, , N, DEVL** command, the old definition is still being used. Only after stopping and starting the device does the new logical IP subnet definition work.

## 3.2 CDLC

Channel DLC (CDLC) devices provide connectivity to a NCP through IBM 3745/3746 Communications Controllers.

### 3.2.1 MVS IOCP definitions

Example 3-3 shows a sample MVS IOCP definition for a 37xx CDLC connection. As you can see from the example, the CDLC connection for an IP connection looks the same in the IOCP as for any type of 37xx channel connection.

*Example 3-3 MVS CDLC IOCP definition*

```

*-----
* IOCP 37xx Directly Attached (NO ESCD)
*-----
          CNTLUNIT CUNUMBR=900,UNIT=3745,PATH=(1A),          +
          CUADD=0,LINK=(**),UNITADD=((01,16))
DEV900   IODEVICE UNIT=3745,ADDRESS=(900,16),CUNUMBR=(900),UNITADD=01

```

### 3.2.2 TCPIP profile definitions

Sample DEVICE and LINK statements necessary in the TCPIP profile are shown in Figure 3-7.

```

; *****
; *          NCP IP over Channel attachment (parallel)          *
; *          To RA8NCPZ                                         *
; *****
DEVICE DEVNCP C  CDLC E22 15 15 4096 4096
LINK  LINKNCP C  CDLC 0   DEVNCP
;
; *****
; *          NCP IP over Channel attachment (ESCON)           *
; *          To RA8NCPZ                                         *
; *****
DEVICE DEVNCPCE CDLC 907 15 15 4096 4096
LINK  LINKNCPCE CDLC 0   DEVNCPCE

```

*Figure 3-7 CDLC PROFILE.TCPIP definitions*

NCP can handle a maximum of 64 KB of data in a transfer at a time. We have allocated 15 read and 15 write buffers to the CDLC where the size of the R/W buffers is 4096 bytes. If the product of (number of buffers x buffer size) exceeds 65,535 bytes, TCP/IP will reduce the maximum number of buffers allocated to make the product meet the maximum allowable number of bytes that can be allocated for every read or write operation. Possible combinations of buffer size and buffer number that would avoid exceeding a product of 65,535 are shown in Table 3-2.

*Table 3-2 CDLC buffer size summary*

Buffer Size	Number of Buffers
4096	15
2048	31
1024	63

### 3.2.3 CDLC NCP channel definitions

#### ICDLC definitions: parallel channel

In Figure 3-8, you see the definitions for the parallel CDLC connection. Note that the CDLC definition is treated as a PU Type 1 and can be manipulated with VTAM **V NET,ACT|INACT** commands.



```

*****
* PARALLEL CHANNEL ADAPTER LINK FOR NCP IP TRAFFIC TO MVS2 (MVS18) *
*****
RA8GC05  GROUP LNCTL=CA,                                X
          CAPACITY=1M
*
RA8LC06  LINE ADDRESS=01,                                LINE NUMBER ADDRESS, MODE X
          CA=TYPE6,                                     CADS OR BCCA CHANNEL TYPE X
          CASDL=120,                                    BEFORE CHANNEL SLOWDOWN X
          DELAY=0.0,                                    CHAN ATTNDelay X
          DYNADMP=NONE,                                 X
          NCPCA=ACTIVE,                                 NATIVE SUBCHANNEL ACTIVE X
          TIMEOUT=120                                  INTERVAL BEFORE DISCONTACT
*
RA8PC06  PU  PUTYPE=1,                                    IP NATIVE RAI 18, VIA P06 X
          ANS=STOP,                                     NCP AUTO NETWORK SHUTDOWN X
          DELAY=1,                                      NCP ATTENTION SIGNAL TIME X
          INTFACE=RA8PC06                              INTERFACE NAME OF ADDRESS

```

Figure 3-8 CDLC parallel channel definitions in the NCP

The DELAY= value for a parallel channel attachment is an installation tuning decision. Normally, a non-zero value is coded. TRANSFR=18 together with an NCP buffer size of 240 (BFRS=240 on the NCP BUILD statement, which is not depicted here) gives a value of 4320 bytes. This product of (TRANSFR x BFRS) should be greater than the TCP/IP MTU size, which is 4096 bytes in our configuration.

**CDLC definitions: ESCON channel**

In Figure 3-9, you see the definitions for an ESCON CDLC connection. ESCON channels are defined with a physical **A** and a logical definition **B**. You should use the ESCON Generation Assistant to create the z/OS IOCP, 3745/6 MOSS-E, and NCP definitions.

```

*-----*
*  ESCON PHYSICAL CHANNEL DEFINITIONS  A  *
*-----*
G08CAPG1 GROUP LNCTL=CA, X
                NPACOLL=(YES,EXTENDED), X
                SPEED=14400000, X
                ANS=CONTINUE, X
                SRT=(32768,32768), X
                XID=YES, X
                XMONLNK=YES, SSCP MONITOR MODE ACT. LINK STATIONPX
                ISTATUS=ACTIVE
L082240 LINE ADDRESS=2240 ESCON COUPLER
P082240 PU PUTYPE=1, X
                TIMEOUT=120, X
                TRANSFR=18, X
                XID=YES, X
                ISTATUS=ACTIVE
:
*****
*  ESCON LOGICAL CHANNEL DEFINITIONS FOR ESCP 2240  B  *
*****
G08CALG1 GROUP LNCTL=CA, X
                PHYSRSC=P082240, X
                MAXPU=16, X
                NPACOLL=(YES,EXTENDED), X
                SPEED=14400000, X
                ANS=CONTINUE, X
                CASDL=120, X
                DELAY=0, 1 X
                MAXBFRU=32, X
                PUDR=NO, X
                SRT=(32768,32768), X
                TIMEOUT=120, X
                TRANSFR=18, 2 X
                XID=YES, X
                ISTATUS=ACTIVE
:
:
L0822401 LINE HOSTLINK=1, LOGICAL LINE IDENTIFIER FOR MOSS-E X
                MONLINK=CONT
P0822408 PU ADDR=08, DEVICE NUMBER 907 *ABC*X
                PUTYPE=1, USED FOR IP TRAFFIC TO MVS18 *ABC*X
                ANS=STOP, AUTO NETWORK SHUTDOWN *ABC*X
                PUDR=NO, NO DYNAM RECONF *ABC*X
                INTERFACE=P0822408 INTERFACE NAME *ABC*

```

Figure 3-9 CDLC ESCON definitions in the NCP

A DELAY=0 1 value is recommended in the NCP when ESCON channels are in use. This is to eliminate a potential internal 200-millisecond delay between communication in the 900-frame and the NCP. TRANSFR=18 2 together with an NCP buffer size of 240 (BFRS=240 on the NCP BUILD statement) gives a value of 4320 bytes. This product of (TRANSFR x BFRS) should be greater than the sum of (TCP/IP MTU size + 29 bytes). In our configuration, MTU size is 4096; with the additional 29 bytes (26 for SNA Transmission Header and 3 for SNA Request/Response Header), we meet this requirement.

## 3.3 CLAW

Cisco 7000 and 7500 series routers with CIP card and 7200 series routers with xCPA card supply channel connectivity to the IBM S/390 via parallel and ESCON channel types. Common Link Access to Workstation (CLAW) can also be used to define RS/6000 devices. The DEVICE statement has a parameter that must match the HOST name as specified on the RS/6000 along with buffer number and size specifications.

In CS for OS/390 V2R10, CLAW performance was improved for small MTU traffic when CS for z/OS was communicating with Cisco 7200-series and 7500-series routers. z/OS V1R2 Communications Server enhances the CLAW support to fully exploit the datagram packing feature of the Cisco 7200 and 7500 Series Routers. While CS for OS/390 V2R10 limited buffer size to 4K, z/OS V1R2 Communications Server allows for buffer sizes of up to 60 KB.

The z/OS V1R2 Communications Server Enhanced CLAW Packing function provides a significant performance improvement over prior releases. Specifically, throughput is increased and CPU consumption is reduced because the ESCON overhead involved in CLAW communications is reduced.

To enable this feature, you must update the CLAW device statement in the PROFILE.TCPIP (using the PACKED parameter). You must also update the CLAW statement within the Cisco router.

### 3.3.1 Cisco channel host IOCP definitions

Cisco routers can be channel attached as CLAW devices within the TCP/IP environment. Each CLAW connection requires two devices out of a maximum of 256. Although this allows for a maximum of 128 CLAW connections per interface, a maximum of 32 CLAW connections per interface is recommended by Cisco.

XCPA channel IOCP is defined as SCTC type. A sample host IOCP statement is shown in Figure 3-10. It includes 16 unit addresses (00-0F).

```
CHPID  PATH=12,TYPE=CNC
CNTLUNIT  CUNUMBR=1000,PATH=(12),UNIT=SCTC,          X
          UNITADD=(00,16)
IODEVICE ADDRESS=(800,8),CUNUMBR=800,UNIT=RS6K
```

Figure 3-10 Host IOCP definition of Cisco router channel

### 3.3.2 Router definitions

The Cisco router channel port is configured with an IP address for the connection to the z/OS TCP/IP stack. A sample definition is shown in Figure 3-11.

```
claw path device-address ip-address host-name
      device-name host-app device-app [broadcast]

interface Channel6/0
ip address 192.168.128.1 255.255.255.0
no keepalive
claw 0110 02 192.168.128.3 RA03 ITSOCISC TCPIPA TCPIP broadcast
```

Figure 3-11 Cisco router definition

- ▶ The path value's first two digits are between 01-FF. For parallel or directly connected ESCON channels, these first two digits are 01. For a channel through an ESCON director, this value shows the ESCON director path. The third digit shows the LPAR number. The last digit is generally not used. But if the IOCP has a definition regarding uni taddr, this value must match.
- ▶ The device-address should match the control unit addresses used for the device. In our example, this value is 02. That means unit addresses 02 and 03 are used for this CLAW device.
- ▶ The ip-address is the host IP address of the router channel interface.
- ▶ The host-name is the name of the host and can be any name. It must match the host name specified in the DEVICE statement in PROFILE.TCPIP.
- ▶ The device-name is the name of the router and can be any name. It must match the router name specified in the DEVICE statement in PROFILE.TCPIP.
- ▶ The host-app and device-app should be TCPIP. If the same interface is used by more than one TCP/IP stack, the host-app has to be the TCP/IP stack name.
- ▶ The broadcast parameter is required for passing broadcasts between the router and the host. For example, if OSPF is configured as a routing protocol and the router channel is an OSPF interface, the broadcast parameter definition is required.

### 3.3.3 Host TCP/IP profile statements

Sample interface TCP/IP profile definition statements are shown in Figure 3-12.

```

DEVICE dev-name CLAW cuaddr host-name router-name NONE
      read-buf write-buf read-size write-size AUTORESTART
LINK link-name IP 0 dev-name

DEVICE CLAW1002 CLAW RA03 ITSOCSC NONE 20 20 4096 4096 AUTORESTART
LINK LNKCLAW1 IP 0 CLAW1002

```

Figure 3-12 Host CIP interface TCP/IP profile definitions

- ▶ The dev-name parameter defines the interface device name.
- ▶ The cuaddr parameter has to match the channel IOCP cuaddr value.
- ▶ The host-name is the host name supplied in the router configuration above.
- ▶ The router-name should be the same as the one used in the router configuration above.
- ▶ The read-buf parameter specifies the number of buffers for the read channel program. It is between 1 and 512. The default is 15.
- ▶ The write-buf parameter specifies the number of buffers for the write channel program. It is between 1 and 512. The default is 15.
- ▶ The read-size value is the size of each read buffer. The default value is 4096 for CLAW devices that do not operate in packed mode, and 32k for devices that operate in packed mode.
- ▶ The write-size value is the size of each write buffer. The default value is 4096 for CLAW devices that do not operate in packed mode, and 32k for devices that operate in packed mode.
- ▶ AUTORESTART is the parameter for restarting the device when stopped.

## 3.4 CTC

Channel-to-channel (CTC) provides point-to-point connectivity from a z/OS system to another. Utilizing only one read and one write channel, CTC is suboptimal to the newer MPCPTP device and subsequent non-channel interfaces such as Hipersockets. This section describes our CTC configuration, shown in Figure 3-13.

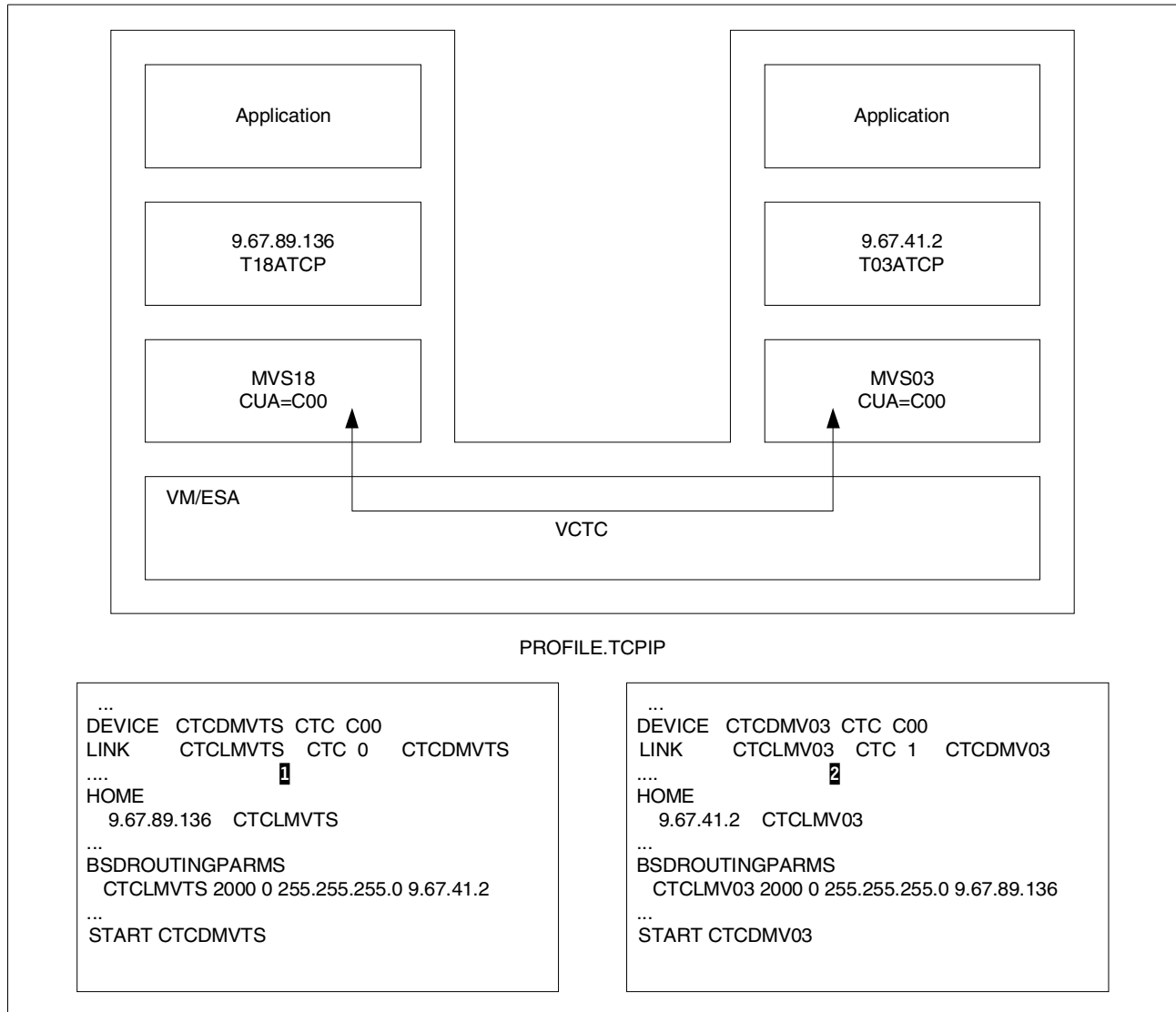


Figure 3-13 Virtual CTC connection between MVS18 and MVS03

The MVS system at the ITSO ran as a guest under VM/ESA. We defined two virtual channel-to-channel devices to interconnect two of our MVS systems, MVS18 and MVS03.

The CTC device addresses defined on one side must be contiguous and must be coupled with the device addresses defined on the other side. This means that the read device number on one z/OS must be coupled with the write device number on the other one. This is done through the `adapter_addr` parameter of the CTC LINK statement. In Figure 3-13, 1 shows a value of zero for `adapter_addr` in MVS18, which means that device address C00 is to be used as the read device. For MVS03, `adapter_addr` has a value of one, as shown by 2, making device address C00 in MVS03 the write device.

Each CTC attachment must be defined via the input/output control program (IOCP) in the host to which the CTC is attached, as shown in Figure 3-14. The devices must be defined with a device type of 3088 or SCTC.

```

P1      IOCONFIG ID=03
      ID      MSG1='V2.4ESA I/O CONFIGURATION FOR 9121/320      ',+
           MSG2='Sep 08-1996',                                +
           SYSTEM=(9121,1)

*IOCP
      CHPID PATH=((0A)),TYPE=BL
*IOCP*****
*IOCP      P A T H      0A      *
*IOCP*****
*IOCP      3088-G161      *
*IOCP*****
           CNTLUNIT CUNUMBR=0C0,UNIT=3088,PATH=(0A),SHARED=N,      +
           UNITADD=(00,64),PROTOCL=S4
DEVOCO  IODEVICE UNIT=3088,ADDRESS=(C00,64),CUNUMBR=(0C0)

```

Figure 3-14 CTC IOCP definitions

CTC devices support parallel channels, ESCON and FICON.

### 3.5 Hipersockets

Hipersockets (Internal Queued Direct I/O - iQDIO) is a new S/390 zSeries hardware feature that provides high-speed communicating LPAR-to LPAR on the same processor (via memory). It also provides secure data flows between LPARs and high availability, in that there is no network attachment dependency or exposure to adapter failures.

Using the Hipersockets function requires the IBM eServer zSeries 900 Driver level 3C. It is not necessary to have parallel sysplex, however. Hipersockets may also be used when some of the LPARs are running Linux or z/VM.

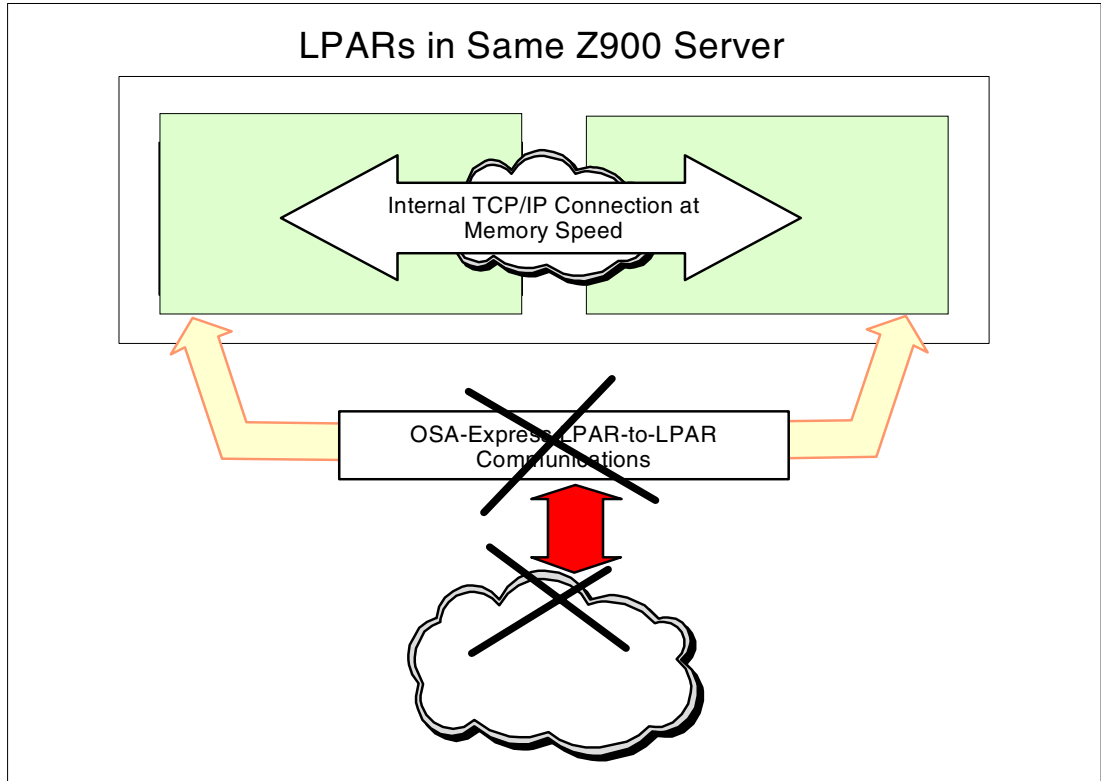


Figure 3-15 Hipersockets network

iQD chpid can be viewed as a “logical LAN” within the CEC. The iQDIO hardware allows up to four separate IQD chpids to be defined per CEC, creating the capability of having four separate “logical LANs” within the same CEC. Figure 3-16 shows an example consisting of multiple logical LANs.

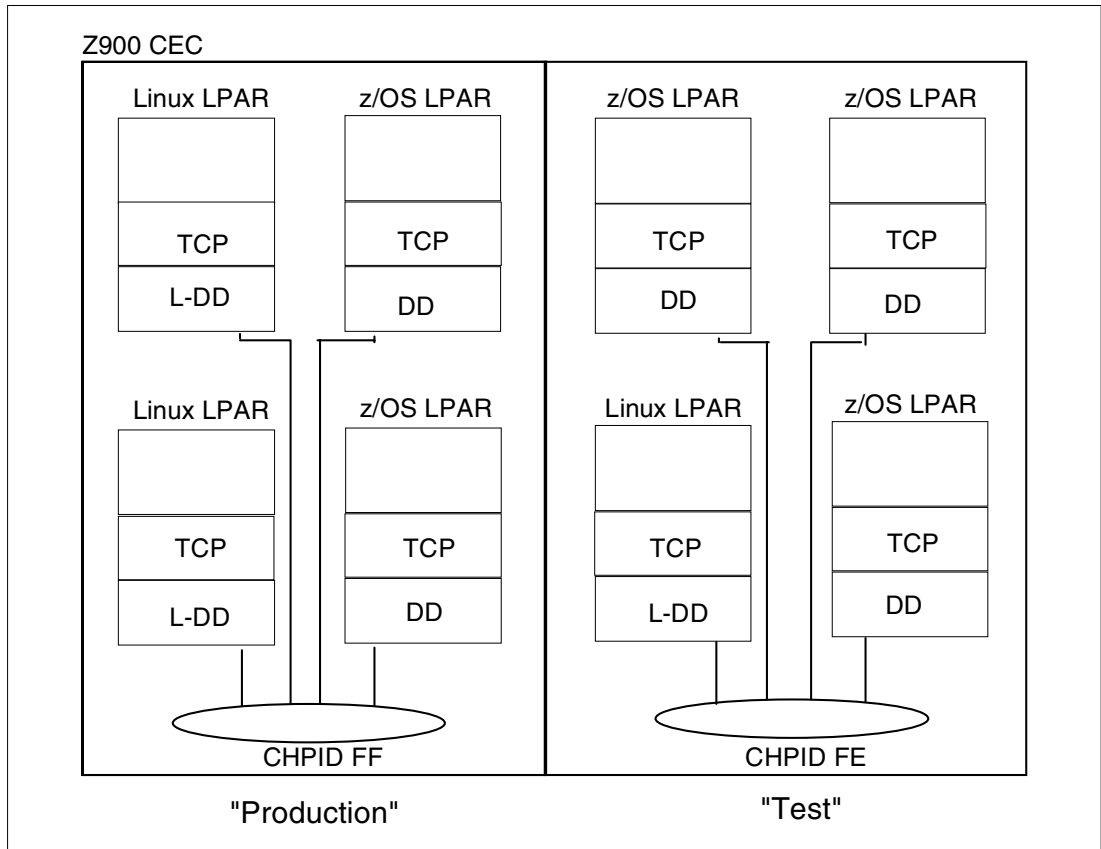


Figure 3-16 Hipersockets multiple LANs

LPARs can have from zero to four Hipersockets chpids which can be shared by all defined LPARs. Each chpid has configurable IQD frame sizes; they are related to TCP/IP MTU sizes, as shown in Table 3-3.

Table 3-3 iQDIO frame sizes and TCP/IP MTU sizes.

OS=value	iQDIO frame size	TCP/IP MTU size
00	16 K	8 K
40	24 K	16 K
80	40 K	32 K
C0	64 K	56 K

The default iQDIO MFS is 16 K. In most work load environments, the default size will result in better storage and CPU utilization.



```

*****
*OS values are '00'=16K, '40'=24K, '80'=40K and 'C0'=64K.*
**
*Need at least 3 addresses per z/OS,maximum of 10:*
*-2 addresses for control *
*-1 address for data for each TCP stack (between 1 and 8)*
*****
ID SYSTEM=(2064,1)
*
CHPID PATH=FC,TYPE=IQD,SHARED,OS=00
CHPID PATH=FD,TYPE=IQD,SHARED,OS=40
CHPID PATH=FE,TYPE=IQD,SHARED,OS=80
CHPID PATH=FF,TYPE=IQD,SHARED,OS=C0
*
CNTLUNIT CUNUMBR=FC00,PATH=FC,UNIT=IQD
IODEVICE ADDRESS=(2C00,32),CUNUMBR=FC00,UNIT=IQD
*
CNTLUNIT CUNUMBR=FD00,PATH=FD,UNIT=IQD
IODEVICE ADDRESS=(2C20,32),CUNUMBR=FD00,UNIT=IQD
*
CNTLUNIT CUNUMBR=FE00,PATH=FE,UNIT=IQD
IODEVICE ADDRESS=(2C40,32),CUNUMBR=FE00,UNIT=IQD
*
CNTLUNIT CUNUMBR=FF00,PATH=FF,UNIT=IQD
IODEVICE ADDRESS=(2C60,32),CUNUMBR=FF00,UNIT=IQD

```

Figure 3-17 Sample IOCP definition

Hipersockets appears in z/OS TCP/IP as Dynamic XCF (DXCF) or Non Dynamic XCF:

► Dynamic XCF

The IPCONFIG DYNAMICXCF parameter has to be defined.

There is no need to code a DEVICE/LINK statement. It is dynamically built and the DEVICE is started during TCP/IP DYNAMICXCF stack initialization. The generated DEVICE name is IUTIQDIO and the generated LINKNAME is IQDIOLNKxxxxxxx, where xxxxxxxx is the character representation of the hexadecimal version of the DYNAMICXCF IP address.

The TRLE created by VTAM will have the name IUTIQDIO and the PORTNAME will have the name IUTIQDxx, where xx is the defined chpid.

► Non Dynamic XCF

In this case, the Hipersocket function must be enabled by defining IQD chpids for access to Hipersockets via HCD. The manually configured IQD chpids cannot conflict with the Dynamic XCF IQD chpid.

New DEVICE and LINK statements are needed. They are similar to MPCIPA devices and can be started and stopped.

Figure 3-18 shows a sample definition for DEVICE and LINK statements.

```

DEvIce device_name MPCIPA NOAUTORestart/AUTORestart
LINK link_name IPAQIDIO device_name

```

Figure 3-18 DEVICE and LINK statements for Non Dynamic XCF

For more information about defining the DEVICE/LINK statements, please refer to *z/OS V1R2.0 Communications Server IP Configuration Reference*, SC31-8776.

In both cases, when the device is started, VTAM dynamically builds the corresponding TRLE. VTAM will use two subchannel devices for the read and write control devices, and one to eight devices for “data devices”. Each TCP/IP stack will be assigned a single data device.

In order to build the MPC group, there must be a minimum of three subchannel devices defined (within HCD) associated with the same IQD chpid. The maximum number of subchannel devices that VTAM will use is ten (supporting eight data devices or eight TCP/IP stacks) per LPAR or z/OS image. The subchannel devices must be configured for the LPAR and online prior to TCP/IP stack initialization.

Hipersockets is enabled when TCP/IP is started. It is not possible to use the **VARY OBEY** command to activate it.

If more than one iQDIO chpid is defined for LPAR, the VTAM start option parameter IQDCHPID must be coded. The value of IQDCHPID parameter can be:

▶ **IQDCHPID=ANY**

This is the default and indicates that any iQDIO chpid can be used. If ANY is specified or defaulted, then:

- When the first iQDIO device is activated, VTAM will use the first acceptable iQDIO chpid detected (having at least three subchannel devices associated with the chpid) to dynamically build the iQDIO TRLE.
- Within HCD (or IOCDS), only one iQDIO chpid must be defined for this logical partition. If VTAM finds more than one iQDIO chpid, the iQDIO activation will not be allowed (ambiguous definition). Users who require (and therefore define) a single iQDIO chpid for a given LPAR can use the default of ANY.

▶ **IQDCHPID=NONE**

This will prevent VTAM from activating an iQDIO TRLE. It will not disrupt the current usage (an already active iQDIO device). All subsequent attempts to activate iQDIO from any TCP/IP stack within this LPAR will be rejected.

▶ **IQDCHPID=*chpid***

*chpid* is the (hexadecimal) value for a specific iQDIO chpid. If the *chpid* is specified, it must match one of the iQDIO chpids that was defined using HCD. This LPAR must be eligible to connect to this specific chpid and must have a minimum of three subchannels devices associated with this chpid. The value must be specified as a single byte hexadecimal number (for example IQDCHPID=F2). This option is intended for users who must define multiple iQDIO chpids within HCD.

VTAM commands can be used to display (**D NET,VTAMOPTS**) and dynamically modify (**MODIFY VTAMOPTS**) the IQCHPID start option. If the processor does not support Hipersockets, then the value for the start options will display as N/A.

### 3.5.1 Hipersockets Accelerator

Hipersockets Accelerator is a function that optimizes the IP packet forwarding processing that occurs across iQDIO and QDIO links. It allows a user to choose one specific TCP/IP stack having direct physical connectivity to the OSA LAN(s) to be an iQDIO router. This stack can connect to all remaining TCP/IP stacks in other LPARs within the same CEC that require connectivity to the same OSA LANs, using Hipersockets connectivity.

The packets that are forwarded between the iQDIO link and the QDIO link are never processed by the TCP/IP stack; they are processed by the DLC (Data Link Control) layer. So, TCP/IP resources such as storage and machine cycles are not expended for routing and forwarding packets.

Hipersockets Accelerator can be implemented by coding the IQDIORouting option in the IPCONFIG statement. The iQDIO routing entries are created dynamically and can be displayed with the **NETSTAT** command. There is also an IQDIOPriority option on IQDIORouting that allow users to specify which of the four priority queues can be used.

From a performance perspective, Hipersockets Accelerator attempts to make the router TCP/IP stack appear as if it did not exist in the path. Each LPAR will appear as if each were directly attached to the physical network.

For additional information on Hipersockets and Hipersockets Accelerator, please reference *zSeries HiperSockets*, SG24-6816.

## 3.6 Hyperchannel

Hyperchannel devices provide access to TCP/IP hosts via Hyperchannel series A devices and series DX devices that function as series A devices. Use the HCH DEVICE and LINK statements to define connectivity via the HYPERchannel A220 adapter.

Figure 3-19 shows a sample definition for Hyperchannel devices:

```
DEVICE HCH1 HCH E00 AUTORESTART
LINK HCHE00 HCH 1 HCH1
```

*Figure 3-19 Hyperchannel sample definition*

## 3.7 LCS

LAN Channel Station (LCS) devices can be used to define an IBM 8232 LAN channel station device, an IBM 3172 Interconnect Controller, an IBM 2216 Multiaccess Connector Model 400, an IBM FDDI, Ethernet, Token Ring OSA or an IBM ATM OSA-2 in LAN emulation mode.

Figure 3-20 shows a sample IBM 3172 hardware configuration. It consists of two IBM 3172s with ICP, equipped with one ESCON channel adapter and four LAN adapters (two Ethernet and two token ring). The purpose here is to show how to configure CS for z/OS IP to use the Ethernet and token ring adapters.

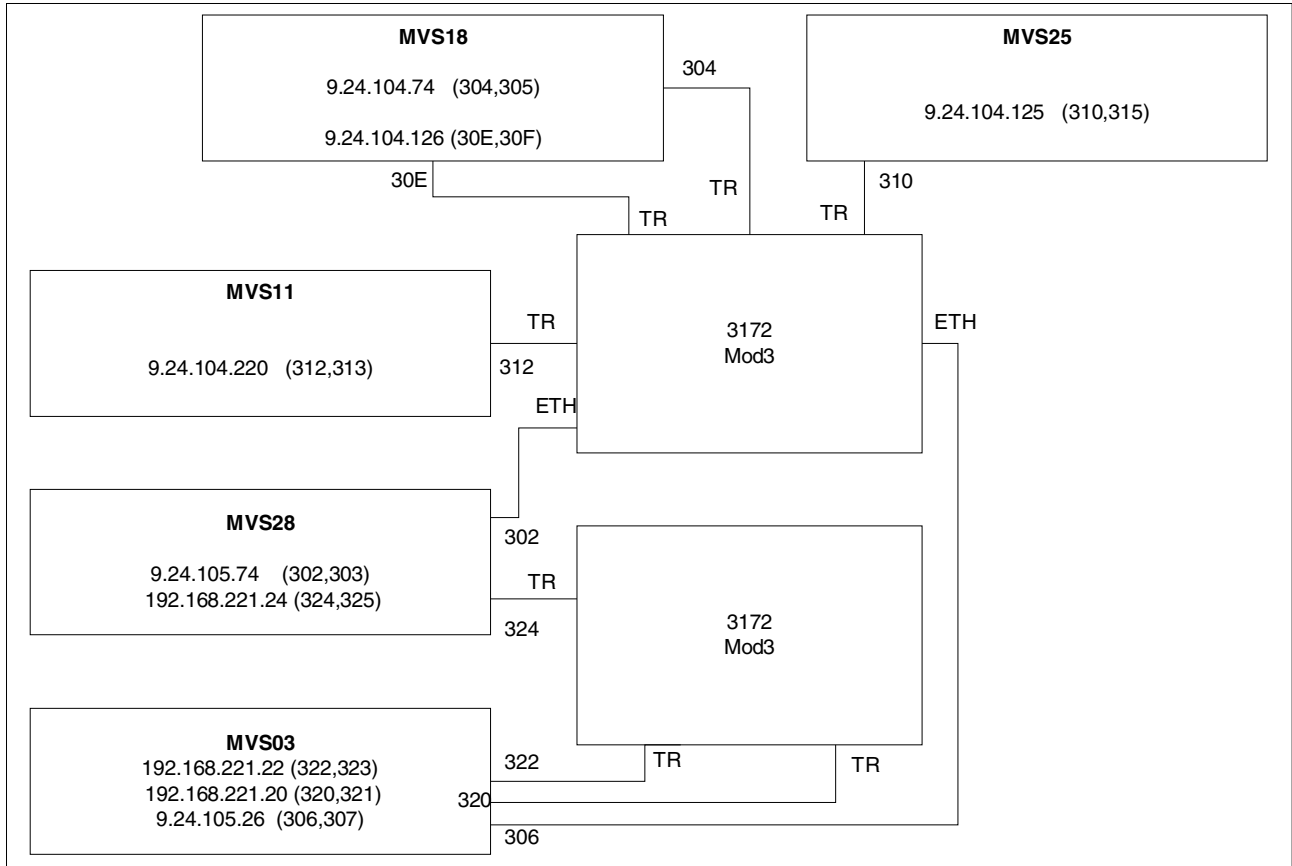


Figure 3-20 IBM 3172 Model 3 sample configuration

### 3.7.1 PROFILE definitions

```
DEVICE icp03 LCS 320 autorestart 1
LINK LICP03 IBMTR 0 icp03 2
;
DEVICE DEVEN1 LCS 306 NETMAN
LINK EN1 ETHEROR802.3 1 DEVEN1
;
HOME
9.24.105.126 EN1 ; 9.24.105.0
192.168.221.20 LICP03
;
GATEWAY
9 = EN1 1500 0.255.255.0 9.24.105
192.168.221 = LICP03 4000 0
DEFAULTNET 9.24.105.1 EN1 1500 0
;
START ICP03
START DEVEN1
```

Figure 3-21 3172 PROFILE.TCPIP definitions

Figure 3-21 shows the TCP/IP profile statements corresponding to this example.

The IBM 3172 system configuration requires TCP/IP LAN Channel Station (LCS) definitions. The required TCP/IP LAN channel station definition statements are:

- ▶ Device
- ▶ Link

**1** The TCP/IP LCS DEVICE statement specifies the address assigned to the device. TCP/IP uses one device address pair to simulate a full-duplex I/O protocol. The even device address is used for the READ I/O-channel while the odd device address is used for the WRITE I/O-channel. The even device address is the one used in the LCS device statement. In this case, TCP/IP uses the token ring adapter device address 320 for reading and address 321 for writing.

**2** The LINK statement parameter link\_number must match and correspond exactly to the IBM 3172 configuration parameter relative adapter number. The relative adapter number is the order in which the Ethernet/802.3 adapter has been defined in the 3172 profile.

Each IBM 3172 channel attachment must be defined to both the channel subsystem and to the z/OS operating system. The IBM 3172 is defined as a 3088 channel-to-channel (CTC) unit with 32 or 64 device numbers. The IBM 3172 should always be defined with 32 device numbers though only two device addresses are used (base device address and base device address+1) for send and receive.

Figure 3-22 shows a sample IOCP definition for the LCS device.

```

*IOCP*****
*IOCP  3172 MOD3 THROUGH SWITCH E1 *
*IOCP*****
*
      CNTLUNIT CUNUMBR=300,UNIT=3172,PATH=(18), +
          CUADD=1,LINK=(E9),UNITADD=((00,32))
DEV300  IODEVICE UNIT=SCTC,ADDRESS=(300,32), +
          CUNUMBR=(300)
*IOCP*****
*IOCP  3172 MOD3 THROUGH SWITCH E1 *
*IOCP*****
*
      CNTLUNIT CUNUMBR=320,UNIT=3172,PATH=(18), +
          CUADD=1,LINK=(E4),UNITADD=((20,32))
DEV320  IODEVICE UNIT=SCTC,ADDRESS=(320,32), +
          CUNUMBR=(320)

```

Figure 3-22 IOCP 3172 Model 3 definition

## 3.8 MPC

Multipath Channel (MPC) or High Performance Data Transfer (HPDT) connections provide multiple read and write subchannels that expand the bandwidth and availability of the channels by increasing the number of logical paths available. As a result, HPDT increases throughput of the channels while reducing CPU costs. Furthermore, IP and SNA traffic can share the HPDT connection, reducing the amount of overhead required to manage separate connections for SNA and IP.

For MPC connections, define a VTAM transport resource list entry (TRLE) for the HPDT connection and activate it using the **VARY ACT, ID=trle\_name** command.

Define the TRLE as MPCLEVEL=HPDT. The maximum transmission unit (MTU) size of the receiving side of an HPDT multipath channel is affected by the MAXBFRU setting in the VTAM on the sending side of the channel. Calculate the MTU size as follows:

$(\text{MAXBFRU}-1) \times 4\text{K}$

A MAXBFRU value of 9 will give an MTU size of 32768. The maximum supported MTU size is 60 KB for MPC and 56 KB for XCF but should not exceed 32 KB for MPC.

Example 3-4 shows the VTAM definitions for the TRL definition.

*Example 3-4 VTAM MPC TRL definition*

---

```

RA3AHC VBUILD TYPE=TRL
* APPN MPC LINK FROM RA3 TO RAK
RA3MPCA TRLE LNCTL=MPC,READ=(C15),WRITE=(C14)
* MPC LINK FROM SA03 TO SA25
MPCT025 TRLE LNCTL=MPC,READ=(C09),WRITE=(C07),MAXBFRU=9

```

---

Configure the TCP/IP address space for the HPDT connection. HPDT is defined by a DEVICE and LINK statement in the PROFILE.TCPIP data set for the TCP/IP address space as a multipath channel point-to-point (MPCPTP) device.

### 3.8.1 MPCPTP

MPC point-to-point (MPCPTP) support is similar to CTC support, but allows multiple read/write Channel Unit Addresses (CUAs) to be viewed as one device:

- ▶ VTAM multiplexes the IP datagrams over available CUAs, thus increasing throughput above CTC.
- ▶ If one CUA fails, the device remains active as long as one read and one write CUA remain active.
- ▶ The MPC connections can be to either another z/OS, a 2216, a Cisco router, or an RS/6000.
- ▶ MPC connections require that the MPC definition in VTAM (the TRLE) be activated before TCP can use it. The same TRLE coded for APPN's use of MPC is shared by TCP. That TRLE name is passed by TCP using the IUTIL interface when TCP desires to use the MPC link.
- ▶ If the MPC TRLE is only to be used by TCP, the VTAM node can be a subarea node and still have the TRLE be HPDT capable.
- ▶ Multiple TCP/IP instances in one host can share the same MPC connection to the same destination host. APPN and IP traffic can share the same MPC link.

In Example 3-5, you can see the TCP/IP definitions used in our test environment for MPC.

*Example 3-5 MPC definition in our PROFILE.TCPIP*

---

```
*****
MPC Definition
VTAM requirement for TRLE Definition in VTAM VBUILD=TRL
The DEVICE name must match the TRLE name in VTAM
The TRLE entry must have MPCLEVEL=HPDT (default)
*****
DEVICE MPCT025 MPCPTP AUTORESTART
LINK MPCT025 MPCPTP MPCT025
```

---

#### Cisco CMPC+ support

The IBM MPC+ protocol has been implemented by Cisco in Cisco routers as Cisco CMPC+ support with Cisco IOS Release 12.0(3) and later router software. CMPC+ requires the use of two data channels, one for read and the other for write. These channels can be through the same or different physical channels. MPC+ definition requires IOCP channels to be defined as CTC. In addition, a VTAM TRLE definition for MPC and TCP/IP profile interface definitions on the host side are required. Router configuration is required on the router side. Figure 3-23 shows the connection of the Cisco router to the z/OS.

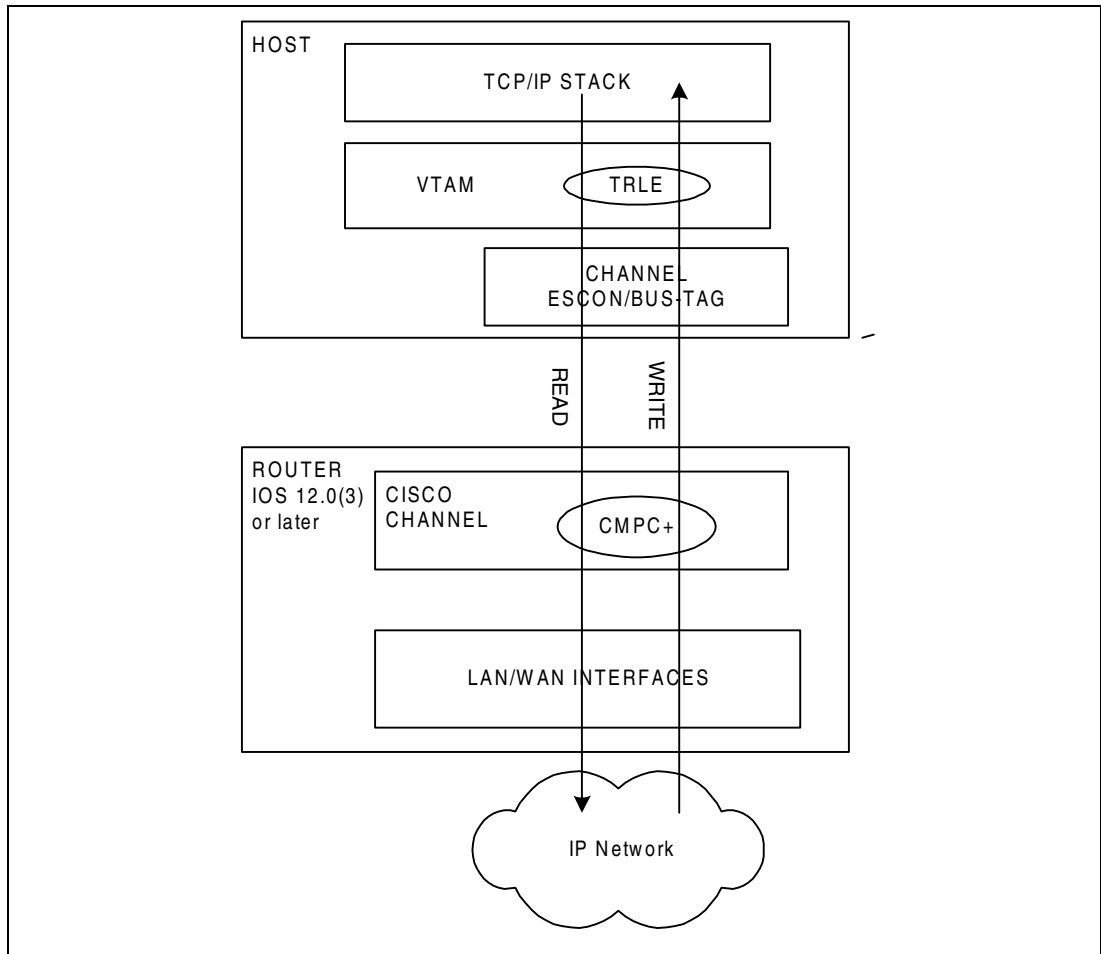


Figure 3-23 Cisco MPC+ channel connection to IBM S/390

### Cisco MPC+ host definitions

The IOCP definition is shown in Example 3-6. In our example, the first two unit addresses (00 and 01) are used for CMPC+.

#### Example 3-6 CMPC IOCP definition sample

```

CHPID  PATH=12,TYPE=CNC
        CNTLUNIT CUNUMBR=1000,PATH=(12),UNIT=SCTC,          X
        UNITADD=(00,16)
        IODEVICE ADDRESS=(800,8),CUNUMBR=800,UNIT=RS6K

```

The VTAM TRLE definition is included in Example 3-7. The TRLE name must match the MPC device name given in the TCP/IP profile, as shown in Example 3-8. For our example, this name is MPC1000.

#### Example 3-7 MPC+ VTAM TRLE definition

```

TRLOC0A VBUILD TYPE=TRL
MPC1000 TRLE LNCTL=MPC,MAXBFRTU=16,X
READ=(1000),X
WRITE=(1001)

```



*Example 3-8 MPC+ TCP/IP profile interface definitions*

```
DEVICE MPC1000 MPCPTP
LINK LNKMPC1 MPCPTP MPC1000
```

**Cisco MPC+ router definitions**

Cisco routers support CMPC+ with IOS Release 12.0(3) and later on 7000, 7200 and 7500 series platforms. In our network, we have a channel interface in slot 6 and we will configure the interface as channel6/0. It has one read and one write data channel, as shown in Example 3-9. The IP address 192.168.128.3 is the IP address of this interface.

*Example 3-9 Router CMPC+ definitions*

```
interface Channel6/0
ip address 192.168.128.1 255.255.255.0
no keepalive
no ip directed-broadcast
no ip mroute-cache
no keepalive
cmpc 0110 00 TG03 READ
cmpc 0110 01 TG03 WRITE
tg TG03 ip 192.168.128.3 192.168.128.1 broadcast
```

## 3.8.2 MPCOSA

MPCOSA implements MPC for the Fast Ethernet and FDDI. MPCOSA is supported from OS/390 V2R8 on. Figure 3-24 shows OSA-2 Fast-Ethernet and FDDI TCP/IP profile MPCOSA definitions.

```
;OSA-2 FDDI MPC Definition
DEVICE DMPCFDDI MPCOSA AUTORESTART
LINK LMPCFDDI OSAFDDI 0 DMPCFDDI
;OSA-2 ENET MPC Definition
DEVICE DMPCENET MPCOSA AUTORESTART
LINK LMPCENET OSAENET 0 DMPCENET
```

*Figure 3-24 MPCOSA device and link definitions for OSA-2 Fast Ethernet and FDDI interfaces*

Also, required VTAM TRLE definitions such as other MPC devices and the TRLE name must match the TCP/IP profile device name. You can see TRLE definitions for OSA-2 Fast Ethernet and FDDI interfaces in Figure 3-25.

```
VBUILD TYPE=TRL
* FDDI TRLE Definition
DMPCFDDI TRLE LNCTL=MPC,READ=(23A0),WRITE=(23A1),MAXBFPU=9
* Fast Ethernet TRLE Definition
DMPCENET TRLE LNCTL=MPC,READ=(2380),WRITE=(2381),MAXBFPU=9
```

*Figure 3-25 OSA-2 MPCOSATRL definitions*

MPCOSA does not support IP multicast and broadcast. The **NETSTAT ARP** command does not show any ARP cache entry for MPCOSA.

### 3.8.3 MPCPTP samehost

MPCPTP samehost (also referred to as IUTSAMEH) can be used to define a connection between two TCP/IP stacks on the same system depicted in Figure 3-26. Additionally, it can provide connectivity between TCP/IP and SNA (VTAM) when using Enterprise Extender.

Use the LINK statement to define a network interface link associated with the IUTSAMEH interface. The reserved TRLE name IUTSAMEH can be used to bring up an MPCPTP connection between two TCP/IP stacks on the same system without the need for a physical device connection between the two stacks. The reserved TRLE name IUTSAMEH can also be used to define an Enterprise Extender connection to the VTAM instance running on this host.

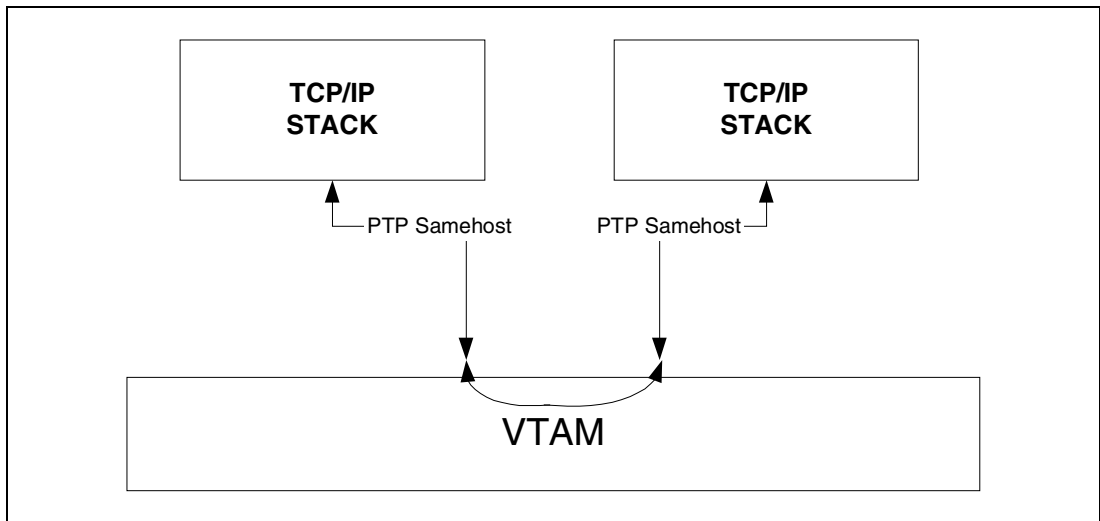


Figure 3-26 MPCPTP samehost connection

As with legacy DLCs, a dynamic TRLE will be created to represent the MPCPTP samehost DLC. The reserved TRLE name IUTSAMEH is used. The DLC code, rather than sending data over an MPC device, will simply route the data between the two TCP/IP instances.

VTAM automatically activates the IUTSAMEH TRLE when an MPCPTP device is started by TCP/IP with a TRLE name of IUTSAMEH.

Figure 3-27 shows a sample TCP/IP definition.

```

*****
SAMEHOST Definition
*****
DEVICE IUTSAMEH    MPCPTP
LINK IUTSAMEH MPCPTP      IUTSAMEH

```

Figure 3-27 MPCPTP definition in our PROFILE.TCPIP

## 3.9 MPCIPA (QDIO)

Since OS/390 eNetwork Communications Server V2R7 IP, you can use Open Systems Adapter Express. The original OSA-Express implementation within CS for z/OS included support only for Gigabit Ethernet (GbE). CS for z/OS IP includes support for both Fast Ethernet and high speed token ring, allowing users of OSA-Express access to either 16M or 100M token ring networks.

To use OSA-Express Token Ring support, you must be running on IBM eserver zSeries 900 Driver level 3C.

z/OS V1R2 Communications Server IP also allows defining the OSA-Express as an LCS device. This option allows you to migrate from your existing OSA token ring attachments to OSA-Express without changing your profile definitions, but without the performance advantages of Queued Direct I/O (QDIO).

OSA-Express is only supported in the S/390 Parallel Enterprise Servers - Generation 5 (G5) and higher family of processors. Gigabit Ethernet employs the same Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol, the same frame format and the same frame size as 10 and 100 Mbps Ethernet. The primary difference is the transmission rate of 1000 Mbps. The OSA-Express provides interoperability with all sorts of networking media through a switch or router.

### 3.9.1 Queued Direct I/O overview

The OSA-Express can use the I/O architecture called Queued Direct I/O. This architecture provides a highly optimized data transfer interface that eliminates the need for channel command words (CCWs) and interrupts during data transmission, resulting in accelerated TCP/IP packet transmission. This is done by providing a data queue between TCP/IP and the OSA-Express. OSA-Express utilizes a direct memory access (DMA) protocol to transfer the data to and from TCP/IP. This design eliminates ESCON bus performance limitations.

The OSA-Express also provides the offloading of IP processing from the host, which is called IP assist (IPA). With IP assist, the OSA-Express offloads the following processing from the host:

- ▶ All MAC handling is done in the card. TCP/IP no longer has to fully format the datagrams for LAN-specific media.
- ▶ ARP processing for identifying the physical address.
- ▶ Packet filtering, screening and discarding of LAN packets.

The data transfer model of the OSA-Express and the OSA-2 interfaces is shown in Figure 3-28.

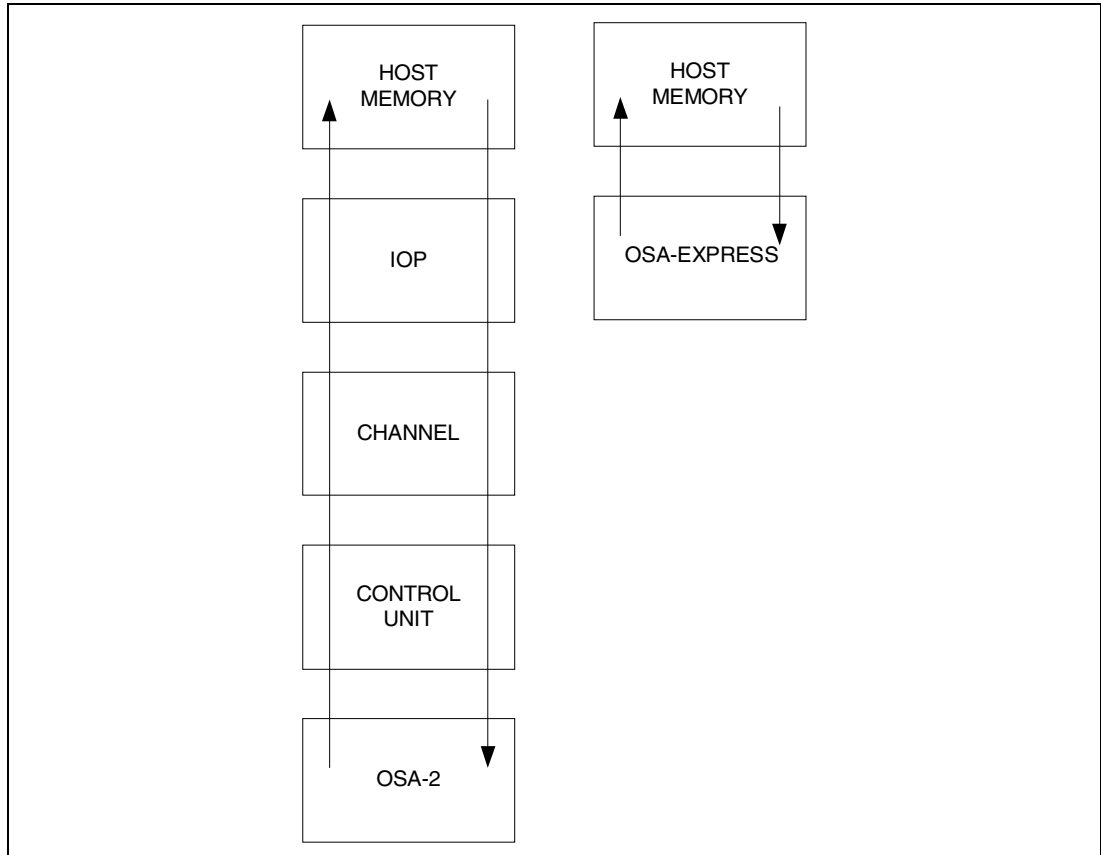


Figure 3-28 OSA-Express Gigabit Ethernet and OSA-2 host memory access

### 3.9.2 Configuring for the OSA-Express adapter

The OSA-Express can be shared by two or more LPARs in the same system. There are several parameters that you must use when defining the OSA-Express feature. In general, we must make IOCP definitions as well as VTAM and TCP/IP PROFILE.TCPIP configuration changes.

#### IOCP

An OSA-Express device is defined in IOCP as of the type OSD channel. At least three devices must be defined to each partition that will use the OSA-Express adapter. One of the devices will be used to transfer data while the other two are for read and write control data. Figure 3-29 shows the IOCP definition for OSA-Express.

```
CHPID PATH=(F8),SHARED,PARTITION=((A10,A8,A9),(A10,A8,A9)), *
TYPE=OSD
CNTLUNIT CUNUMBR=23A0,PATH=(F8),UNIT=OSA
IODEVICE ADDRESS=(23A0,015),UNITADD=00,CUNUMBR=(23A0),UNIT=OSA
```

Figure 3-29 IOCP definition for OSA-Express Gigabit Ethernet

Unlike an OSA ATM, you do not need to use OSA/SF to define or configure the OSA-Express Gigabit Ethernet. You can use OSA/SF for monitoring purposes. You need to define an OSAD channel for this utility, but it is not required. If you want to use OSA/SF, you can use the sample OSAD channel IOCP definition shown in Figure 3-30.

```
IODEVICE ADDRESS=23AF,UNITADD=FE,CUNUMBR=(23A0),UNIT=OSAD
```

Figure 3-30 IOCP definition for OSA/SF

**VTAM and TCP/IP definition**

Currently, only CS for z/OS IP uses the OSA-Express function (SNA can only use it indirectly through Enterprise Extender). You need to define a TRLE statement in VTAM. Figure 3-31 shows the TRL major node for the OSA-Express device.

```
SC650SF8 VBUILD TYPE=TRL
SC6523A0 TRLE LNCTL=MPC, *
          READ=23A0, 1 *
          WRITE=23A1, 2 *
          DATAPATH=(23A2), 3 *
          MPCLEVEL=QDIO, 4 *
          PORTNAME=OS6523A0, 5
```

Figure 3-31 TRL definition for OSA-Express

- 1 defines the device address that is used for reading control data.
- 2 defines the device address used for writing control data.
- 3 specifies the subchannel addresses used to read and write data through an OSA-Express connection. Each TCP/IP instance that issues a START DEVICE statement for an OSA-Express feature will be assigned one of the DATAPATH channels by VTAM. Sufficient DATAPATH subchannel addresses must be coded for the number of concurrent instances that will be using an OSA-Express port.
- 4 indicates that VTAM is to use the Queued Direct I/O interface for communicating with the device. OSA-Express must use QDIO in its MPCLEVEL definition.
- 5 specifies the name that will be used in TCPIP.PROFILE to define the device. In comparison, the other TCP/IP devices use the TRLE name in the device definition. Figure 3-32 shows the TCPIP.PROFILE for OSA-Express device definition.

```
DEVICE OS6523A0 MPCIPA PRIRouter 2
LINK OS6523AOLINK IPAQENET OS6523A0 1
```

Figure 3-32 TCP/IP device definition for OSA-Express Gigabit Ethernet

- 1 should match the PORTNAME specified in the TRLE definition of this OSA-Express Gigabit Ethernet device.
- 2 defines the OSA-Express device to TCP/IP. The PRIROUTER parameter specifies that if a datagram is received at this device for an unknown IP address, the device will route the datagram to this TCP/IP stack. Alternatively, this TCP/IP stack could also function as a backup to the primary router by coding SECROUTER as a parameter. However, if NONROUTER was specified or no TCP/IP stack was designated as PRIRouter or SECRouter, the device will simply discard the datagram.

A display of the device in TCP/IP shows the following:

```
DEVNAME: OS6523A0          DEVTYPE: MPCIPA    DEVNUM: 0000
DEVSTATUS: READY          CFGROUTER: NON    ACTROUTER: NON
LNKNAME: OS6523AOLINK     LNKTYPE: IPAQENET 1 LNKSTATUS: READY
  NETNUM: 0    QUESIZE: 0    SPEED: 0000001000
  BYTESIN: 0000000948     BYTESOUT: 0000000948
  ARPOFFLOAD: YES  ARPOFFLOADINFO: YES
BSD ROUTING PARAMETERS:
  MTU SIZE: 00000          METRIC: 00
  DESTADDR: 0.0.0.0       SUBNETMASK: 255.255.255.0
MULTICAST SPECIFIC:
  MULTICAST CAPABILITY: YES
  GROUP              REFCNT
  -----
  224.0.0.1         0000000001
```

Figure 3-33 D TCPIP, N, Dev display for MPCIPA device

The link type **1** shows IPAQENET, which indicates that the link uses the IP assist based interface, belongs to the QDIO family of interfaces, and uses the Gigabit Ethernet or Fast Ethernet protocol.

If you have multiple TCP/IP stacks sharing the OSA-Express device, you must ensure that there is only one designated primary and secondary router for the device. Trying to assign another TCP/IP stack as either primary or secondary will result in a warning message for the START processing of the device for that TCP/IP stack.

When the OSA-Express GbE is started, TCP/IP registers the entire set of local (home) IP addresses for this TCP/IP instance to OSA-Express. This allows the device to route datagrams destined for those IP addresses to this TCP/IP instance. If the device receives a datagram destined for an unregistered IP address, then OSA-Express sends the datagram to the TCP/IP instance that is defined as the primary router or secondary router for this device. If you change any of the home IP addresses in this TCP/IP stack, you must stop and restart the MPCIPA device for the OSA-Express routing to take effect for the new home IP addresses. Once registered, IP addresses remain on the OSA-Express until the card is reset.

The VTAM **TRLE** display command is shown in Figure 3-34. Each read, write and control channel is shown with its status by the display command. You can also see new priority queue information with the **TRLE** display command. This shows the last ten seconds of congestion if congestion is detected.

```

D NET,TRL,TRLE=SC6523A0
IST097I DISPLAY ACCEPTED
IST075I NAME = SC6523A0, TYPE = TRLE 962
IST486I STATUS= ACTIV, DESIRED STATE= ACTIV
IST087I TYPE = LEASED           , CONTROL = MPC , HPDT = YES
IST1715I MPCLEVEL = QDIO           MPCUSAGE = SHARE
IST1716I PORTNAME = OS6523A0 LINKNUM = 0 OSA CODE LEVEL = 0406
IST1577I HEADER SIZE = 4096 DATA SIZE = 0 STORAGE = ***NA***
IST1221I WRITE DEV = 23A1 STATUS = ACTIVE STATE = ONLINE
IST1577I HEADER SIZE = 4092 DATA SIZE = 0 STORAGE = ***NA***
IST1221I READ DEV = 23A0 STATUS = ACTIVE STATE = ONLINE
IST1221I DATA DEV = 23A2 STATUS = ACTIVE STATE = N/A
IST1724I I/O TRACE = OFF TRACE LENGTH = *NA*
IST1717I ULPID = TCPIPA
IST1757I PRIORITY1: UNCONGESTED PRIORITY2: UNCONGESTED
IST1757I PRIORITY3: UNCONGESTED PRIORITY4: UNCONGESTED
IST314I END

```

Figure 3-34 VTAM display TRLE command

### 3.9.3 Sample OSA-Express GbE connection between two systems

We connect two LPARs with OSA-Express GbE as shown in Figure 3-35. The system names are SC64 and SC65. The SC64 system GbE interface CHPID is F4 and uses three subchannels: 2380 to 2382. One of these subchannels serves as the data channel, one for read, and the last for write. The SC65 system is configured to have the exact opposite of these configurations. The CHPID for the SC65 system is F8 and the subchannels are from 23A0 to 23A2. Figure 3-35 shows the connection between the two systems. After the IOCP definitions are in place, the channel addresses are made online with system channel online commands.

First we execute the CHPID online commands:

- ▶ **CF CHP (F4), ONLINE** on the SC64 system
- ▶ **CF CHP (F8), ONLINE** on the SC65 system

Then, the CUADDR online commands:

- ▶ **V 2380-2382, ONLINE** on the SC64 system
- ▶ **V 23A0-23A2, ONLINE** on the SC65 system

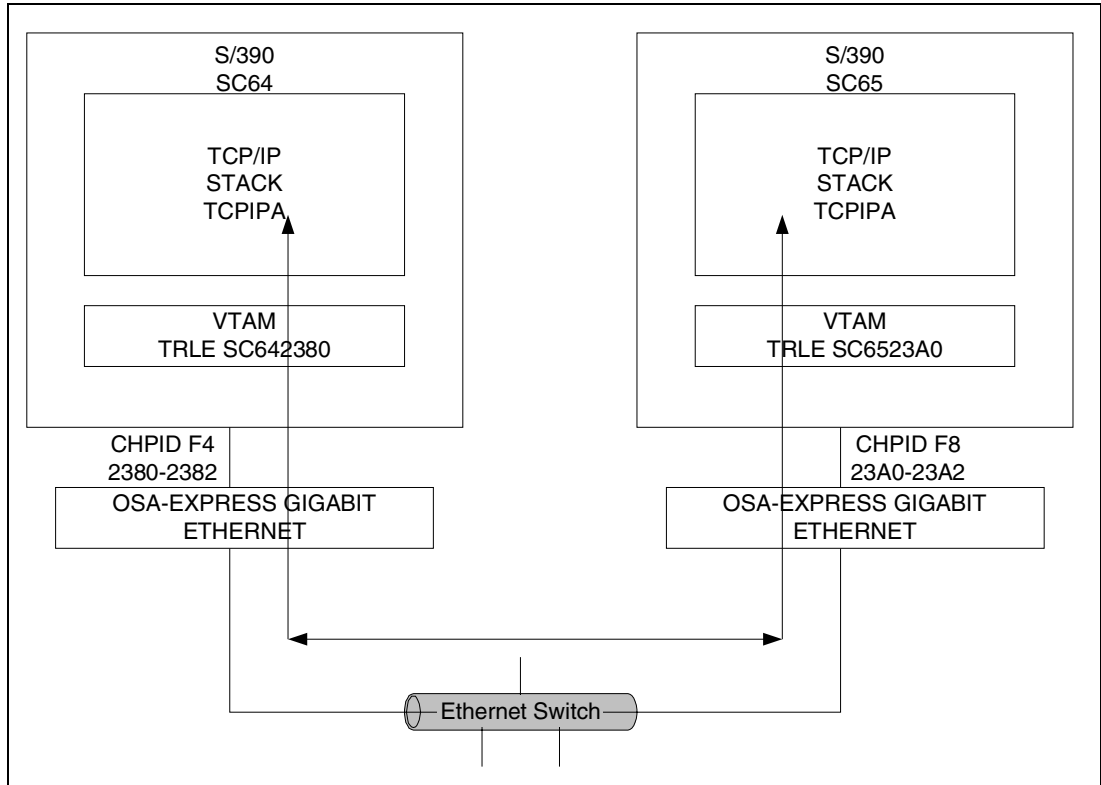


Figure 3-35 OSA-Express GbE connection between two separate systems

### VTAM TRL major node definitions on each system

Figures 3-36 and 3-37 show the VTAM TRL definitions required for each of the two systems.

```

SC640SF8 VBUILD TYPE=TRL
SC642380 TRLE LNCTL=MPC, *
          READ=2380, *
          WRITE=2381, *
          DATAPATH=(2382), *
          PORTNAME=OS642380, *
          MPCLEVEL=QDIO

```

Figure 3-36 SC64 system's VTAM TRL definition

```

SC650SF8 VBUILD TYPE=TRL
SC6523A0 TRLE LNCTL=MPC, *
          READ=23A0, *
          WRITE=23A1, *
          DATAPATH=(23A2), *
          MPCLEVEL=QDIO, *
          PORTNAME=OS6523A0

```

Figure 3-37 SC65 system's VTAM TRL definition

VTAM TRL entries have to be activated before TCP/IP profile updates activate the device and link. To activate the TRL entry, issue the VTAM TRL **ACTivate** command:

**V NET,ACT,ID=tr1name**



### TCP/IP profile **DEVICE** and **LINK** definitions

The TCP/IP device names must match the name given on the TRLE (TRL entry). After creating the appropriate **DEVICE** and **LINK** definitions, you must update your profile either with the **V TCPIP, ,OBEY** command or by restarting your TCP/IP stack.

```
DEVICE OS642380 MPCIPA
LINK OS642380LINK IPAQENET OS642380
```

Figure 3-38 SC64 system's TCP/IP profile GbE device definition

```
DEVICE OS6523A0 MPCIPA
LINK OS6523A0LINK IPAQENET OS6523A0
```

Figure 3-39 SC65 system's TCP/IP profile GbE device definition

At this point, the device has been completely configured and can be started either with the **V TCPIP, ,OBEY** command with a start directive in the TCP/IP profile or by issuing the **start** command for this device to the TCP/IP stack.

## 3.10 SNALINK LU0

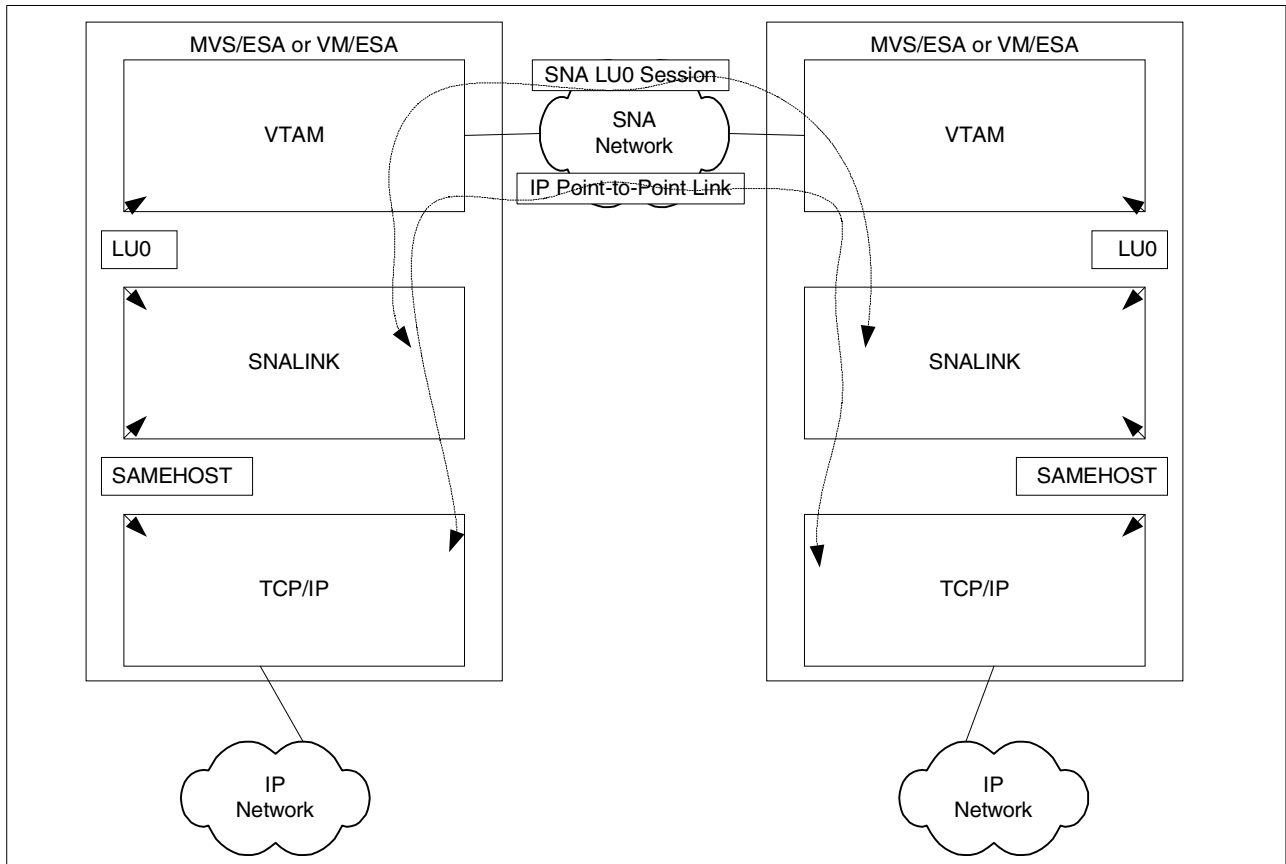


Figure 3-40 SNALINK LU0 architecture

SNALINK is a very convenient way to connect two remote TCP/IP networks using an existing SNA backbone network. The SNALINK address space runs a VTAM application program, which communicates with another VTAM SNALINK LU0 application on the remote side. IP datagrams are packed into SNA request units and transmitted using an LU0 SNA session.

Two SNALINK LU0 applications can be configured to connect using a single, bi-directional session, or with two separate sessions (one dedicated to send data in each direction). Each SNALINK server can communicate with up to 9999 other SNALINKs simultaneously.

To improve SNALINK performance, you can use the largest request unit size that the underlying SNA network will support. The maximum RU size is 32 KB.

An SNALINK connection is a point-to-point type of connection. The routing is based on the network part of the IP address.

With the introduction of dynamic NCP routing in NCP V7R1, SNALINK was also used to transport the dynamic routing messages between the NCP client and the NCPROUTE server at the z/OS TCP/IP host. Although NCP V7R3 introduced IP over Channel (CDLC), SNALINK was still required for the NCPROUTE communication. It was not until NCP V7R4 was made available that the use of SNALINK to support NCPROUTE communications became optional. This subject is addressed in more detail in Appendix B, "NCPROUTE" on page 195.

### 3.10.1 SNALINK LU0 definition

<pre> USER.PROCLIB(T03AT25A) ----- MVS03  //T03AT25A PROC MODULE=SNALINK, //      TCP='T03ATCP' //      APPLID='RAKT03A' //      MAXSESS='6' //      MAXWAIT='0005' //      STYPE='SINGLE' //      MAXRU='C7' //SNALINK EXEC PGM=&amp;MODULE,REGION=2048K,TIME=1440, // PARM='&amp;TCP &amp;APPLID &amp;MAXRU &amp;MAXSESS &amp;MAXWAIT &amp;STYPE' //STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR </pre>	<pre> USER.PROCLIB(T25AT03A ----- MVS25 ) //T25AT03A PROC MODULE=SNALINK, //      TCP='T25ATCP' //      APPLID='RAKT25A' //      MAXSESS='6' //      MAXWAIT='0030' //      STYPE='SINGLE' //      MAXRU='C7' //SNALINK EXEC PGM=&amp;MODULE,REGION=2048K,TIME=1440, // PARM='&amp;TCP &amp;APPLID &amp;MAXRU &amp;MAXSESS &amp;MAXWAIT &amp;STYPE' //STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR </pre>
<pre> TCP.TCPPARMS(PROF03A) ----- MVS03  HOME      192.168.128.239.3   LINKT25A ; AUTOLOG T03ATA25A ENDAUTOLOG ; DEVICE DEVT25A SNAIUCV SNALINK RAPT25A T03AT25A LINK LINKT25A SAMEHOST 0 DEVT25A ; GATEWAY 192.168.239.27 LINKT25A 2000 HOST ; START DEVT25A </pre>	<pre> SYS1.TCPPARMS(PROFILE) ----- MVS25  HOME      192.168.128.239.27   LINKT03A ; AUTOLOG T25AT03A ENDAUTOLOG ; DEVICE DEVT03A SNAIUCV SNALINK RAPT03A T25AT03A LINK LINKT03A SAMEHOST 0 DEVT03A ; GATEWAY 192.168.239.3 LINKT03A 2000 HOST ; START DEVT03A </pre>
<pre> SYSPLEX.VTAMLST(RA3SNALK) -----  RA3SNALK VBUILD TYPE=APPL RAKT03A APPL ACBNAME=RAKT03A,         AUTH=(ACQ,VPACE), PARSESS=YES, VPACING=0,         EAS=12, SONCIP=YES, SRBEXIT=YES </pre>	<pre> RISC.VTAMLST(RAPBTCP) -----  RABATCP VBUILD TYPE=APPL RAPT25A APPL ACBNAME=RAPT25A,         AUTH=(ACQ,VPACE), PARSESS=YES, VPACING=0,         EAS=12, SONCIP=YES, SRBEXIT=YES </pre>

Figure 3-41 SNALINK LU0 definition

The following definitions are required to define an SNALINK. The reference numbers refer to the numbers in Figure 3-41:

- ▶ Device and link definitions:

The PROFILE.TCPIP configuration data set defines and names these resources at startup time. The *remote LU name* is referenced in the DEVICE statement and has to match the VTAM definitions on the remote side of the SNALINK. Each host to which an SNALINK is attached requires a pair of SNA LU0 DEVICE and LINK statements.

SNALINK uses SAMEHOST to connect to the stack. Specifying IUCV on the LINK statement is still supported for migration purpose.

- ▶ Home Address:

Each defined SNALINK needs a *local* IP address (HOME statement).

► Routing table entry:

SNALINK needs a description of how to reach the destination network (routing table entry in the BSDROUTINGPARMS statement). You just need to define the network part of the IP address. In the sample, the net is 192.168.239. If you want to define the complete IP address, you have to define the remote IP address. SNALINK LU0 does not require the -s option in the RouteD started task, because it is the default if a point-to-point link is in use.

► VTAM definitions:

The VTAM application SNALINK requires the following definitions:

- VTAM APPL definitions.
- Local LU name defined as a parameter of the started procedure name. **2**
- The local LU name must be defined as a minor cross-domain or cross-network resource name to the remote SNALINK application.
- The VTAM application definition statement for SNALINK LU0 must have the SRBEXIT=YES parameter coded to ensure that VTAM passes control to SNALINK LU0 in SRB mode. ABEND x'0C2' or x'0F8' will occur if you do not use the SRBEXIT parameter.

**Important:** You must ensure that the EAS parameter in VTAM has a large enough value. In dual mode, two sessions are required per SNALINK. In single mode, one session is required per SNALINK. The EAS parameter in a VTAM major node must be twice the number of sessions defined.

The VTAM definitions for the Telnet virtual terminals, X.25 and the SNALINK may be combined in the same VTAM major node.

► SNALINK Server PROCLIB Updates:

The parameter field PARM of the EXEC statement *must* contain:

- The started procedure name (jobname) of the TCP/IP server. **1**
- The local VTAM LU name. **2**
- Retry: defines retry delay for VTAM sense codes:

```
087D0001 SSCP Rerouting failed (ADJSSCP table exhausted)
080A0001 Session establishment rejected by SSCP
08570003 Target LU is not active
```

- DUAL or SINGLE session type:
  - When operating in DUAL session type, SNALINK opens two SNA sessions for each remote LU with which it communicates, one for sending and one for receiving. This type is compatible with the previous version of SNALINK.
  - It requires the use of PARSESS=YES (Parallel session support) in the VTAM application major node.
  - When operating in SINGLE session type, SNALINK only opens a single session and multiplexes send and receive operations onto this single session. When SNALINK communicates with the ACF/NCP router, it must use the SINGLE session type.
- Maximum request unit size:

This improves the performance of SNALINK on z/OS using the largest RU size that VTAM and NCP support. The value of RU size is in the form X'mn' which means m multiplied by 2 to the power of n. In the sample we used X'88', resulting in 2048 bytes.

- For NCP, check MAXDATA, MAXBFRU, TRANSFR and BFRS.
- For VTAM, check MAXBFRU.

### 3.10.2 Operating SNALINK

You use the following commands to operate a SNALINK:

- ▶ Stopping and starting:
  - Use the **STOP** command on the operator's console. For example:  
`STOP T18ASNO`  
or use the **MODIFY** command with the **HALT** parameter.
  - Use the **START** command on the operator's console. For example:  
`START T18ASNO`
- ▶ **NETSTAT DEVLINKS** determines the connection status.
- ▶ The **MODIFY** command with **PKTTRACE** enables or disables tracing of IP packets over the SNALINK.

## 3.11 SNALINK LU6.2

SNALINK LU6.2 network interface supports the passing of TCP/IP datagrams among multiple hosts and/or workstations through SNA LU type 6.2 conversations.

This is a point-to-point connection, and each SNALINK LU6.2 connection will require an additional IP address on a different network or subnet. You also can have an SNALINK LU6.2 connection between TCP/IP for z/OS and TCP/IP for OS/2.

### 3.11.1 SNALINK LU6.2 definition

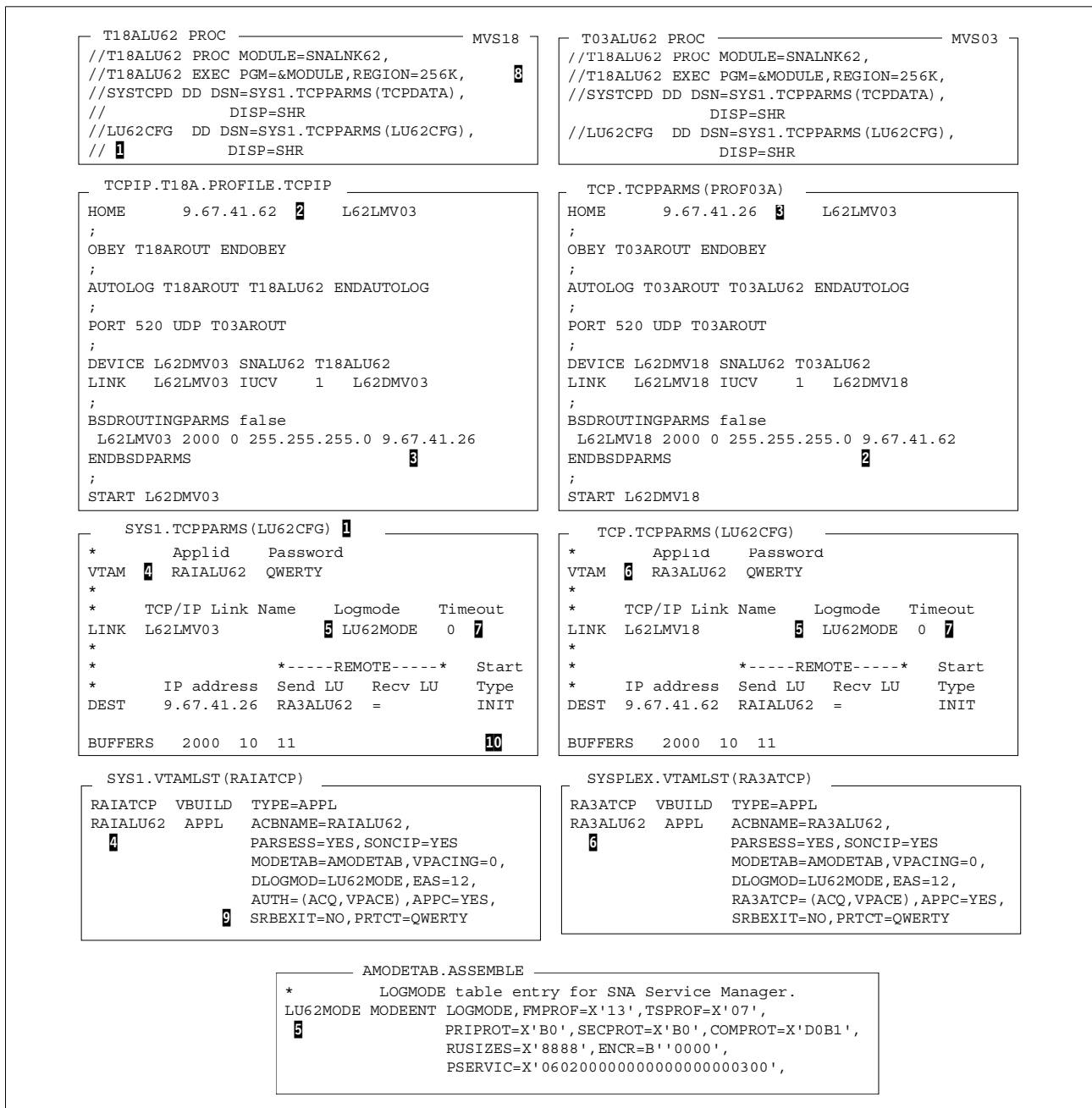


Figure 3-42 SNALINK LU6.2 definition using dynamic routing

The following definitions are required to define an SNALINK LU6.2. The reference numbers refer to the numbers in Figure 3-42.

**Using the INIT option:** The delay in establishing a connection could be long enough to cause a Telnet or FTP command to time out during initialization before the connection can be established. The INIT option in the SNALINK LU6.2 configuration data set can be used to establish the connection to a specific destination during the initialization of the SNALINK LU6.2 address space rather than when the first packet is sent or received.

The following steps are required to define an SNALINK LU6.2 connection:

► **DEVICE and LINK definitions:**

- You have to update the PROFILE.TCPIP data set to define the LINK and DEVICE statements for each connection to be established over an LU 6.2 link.
- SNALINK uses SAMEHOST to connect to the stack. Specifying IUCV on the LINK statement is still supported for migration purposes.
- The LU names are not defined in PROFILE.TCPIP, as they were for the SNALINK LU0.

► **Home address:**

You have to define a local address for this link in the HOME statement. [2](#) [3](#)

► **Routing Table Entry:**

SNALINK LU6.2 needs a description of how to reach the destination network (routing table entry in the BSDROUTINGPARMS statement). You need to define the complete remote IP address. SNALINK LU6.2 does not require the `-s` option in the Routed started task, because it is the default if a point-to-point link is in use.

► **VTAM definitions:**

The following VTAM definitions are needed by SNALINK LU6.2:

- VTAM definition for the local application (an APPL statement) [4](#) [6](#)
- VTAM definition for the remote application (a CDRSC statement)
- VTAM logmode table entry [5](#)
- You must code `SRBEXIT=N0` [9](#)

SNALINK LU6.2 is not designed to run with VTAM in SRB mode. The z/OS S0F8 abend code indicates that an SVC was issued in SRB mode.

► **SNALINK LU6.2 proclib update:**

The TCPIP.DATA configuration data set. This data set is used by the SNALINK LU6.2 address space to find the TCP/IP jobname (the name of the TCP/IP started task).

There are two different ways to code the TCPIPjobname statement in the TCPIP.DATA:

- TCPIPjobname T18ATCP
- system\_name: TCPIPjobname T18ATCP

The second way is used if you share the TCPIP.DATA data set between z/OS systems. You may have more than one TCPIPjobname statement in the data set.

**Note:** [3](#) SNALINK LU6.2 does not use the standard search order for TCPIP.DATA, and does not look into SYS1.TCPPARMS. The //SYSTCPD DD statement is used to explicitly allocate TCPIP.DATA.

► **SNALINK LU6.2 configuration data set: [1](#)**

All the statements and related parameters are explained in *z/OS V1R2.0 Communications Server IP Configuration Reference*, SC31-8776.

- [7](#) The LINK statement contains an `idle_disconnect` parameter. This parameter indicates the time after which an idle session will be terminated if the value is not set to zero. The session will be re-established when needed.

We set this parameter to zero to disable the inactivity checking on the line.

- `BUFFERS 10` statement. The maximum IP packet size should match the `max_packet` size in the GATEWAY or BSDROUTINGPARMS statement. It must be less than the

maximum physical information unit (PIU) in the VTAM definition for your LU 6.2 connection.

## 3.12 Virtual devices (VIPA)

Virtual devices are used to define Virtual IP Addresses (VIPAs) to TCP/IP. VIPA is a term that refers to an Internet address on a z/OS host that is not associated with a physical address.

There are two types of VIPA: Static and Dynamic.

Following is a sample definition for virtual devices:

```
DEVICE VDEV1 VIRTUAL 0
LINK VLINK1 VIRTUAL 0 VDEV1
DEVICE VDEV2 VIRTUAL 1
LINK VLINK2 VIRTUAL 0 VDEV2
```

Figure 3-43 VIPA definition

More information about Virtual devices and VIPA can be found in *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517.

## 3.13 X.25 NPSI connection

Packet switching data networks (PSDN) carry messages that have been divided into segments, called packets, over circuits that are shared by many users simultaneously.

Some of the advantages of using a PSDN are:

- ▶ Low cost of shared links
- ▶ Short access time
- ▶ Connection flexibility

A packet switching data network (PSDN) generally consists of data switching equipment (DSE) and high-speed links between them. The DSEs are often computers dedicated to handling data packets. The links are sometimes relayed via satellites or consist of fiber cables. This equipment makes up the PSDN and is often depicted as a cloud. The programs and protocols used inside the cloud are the exclusive property of the network owner.

The user of the PSDN services has access to the network through data circuit-terminating equipment (DCE). From the PSDN point of view, the user's equipment is called data terminal equipment (DTE). The protocol for information interchange between the DTE and DCE is called X.25.

A PSDN can be either public or private. The public networks are usually owned by telephone companies or the telegraph and telephone administration in a country.

The private packet networks are often used inside enterprises. They make up a part or the entire backbone network in a company. For example, an SNA network could be configured as a private packet network by implementing the IBM X.25 SNA INTERCONNECT (XI) in the communication controller nodes, which must provide the X.25 protocol.



z/OS CS IP can be connected to a packet switching data network (PSDN) through the X.25 NCP packet switching interface (NPSI).

**Note:** Please note that the X.25 support that was made available on the 3746-9x0 platform in 1997 would be a logical migration path from X.25 NPSI.

Call redirection is a PSDN subscription option that allows incoming calls destined for a DTE that is out of order to be redirected. When a call is redirected, a search for a specified group of DTEs (HUNT GROUP) is initiated. z/OS CS IP supports HUNT GROUP. You can use the ALTLINK statement in the TCP/IP X.25 configuration data set to specify members of a hunt group for incoming calls. See *z/OS V1R2.0 Communications Server IP Configuration Reference*, SC31-8776 for more details.

### 3.13.1 Supported NPSI configurations

CS for z/OS IP supports the following NPSI configurations:

- ▶ GATE
- ▶ GATE Fast Connect
- ▶ DATE

### 3.13.2 NPSI GATE

General Access to X.25 Transport Extension (GATE) allows a host application called the *communication and transmission control program* (CTCP) to control the virtual circuits. GATE uses logical link control type 4 (LLC4).

The CTCP controls the setup and the termination of the virtual circuit by exchanging commands with the X.25 NPSI GATE. In addition, all application data passes through the CTCP.

The CTCP communicates with NPSI GATE by using one of two types of LU-LU sessions:

- ▶ The first type of session is between the CTCP and one of the logical units defined for the multichannel link (MCH).
- ▶ The second type is between the CTCP logical unit and the logical units for the virtual circuits (VC).

GATE is recommended because it allows NPSI to handle more details of error recovery and allows an X.25 physical link to be shared with other functions.

### 3.13.3 NPSI GATE Fast Connect

Fast Connect is an integrated option of the NPSI GATE function. Fast Connect will give switched virtual circuits a leased-line appearance, enabling session establishment between the CTCP and the virtual circuit logical unit before the actual virtual call is established. Also, when the virtual call is cleared, the CTCP-VCLU session does not end but will continue carrying data of the next virtual circuit to be established.

When Fast Connect is selected on an MCH; the entire MCH is dedicated to the fast connect operation.

### 3.13.4 NPSI DATE

Dedicated Access to X.25 Transport Extension (DATE) provides extended support for LLC types 0, 2, 3 and 5. DATE support requires the use of a CTCP. The CTCP controls the setup and the termination of the virtual circuits, and processes all X.25 control and qualified packets, except when using LLC type 5. DATE can be used for security control, resource allocation, accounting and other extended requirements.

The application communicates directly with the terminal for data exchange. Consequently, the CTCP is not involved. For control and qualified packets, the CTCP communicates with the X.25 NPSI DATE function over the CTCP to MCH LU session.

It is not possible to use DATE and GATE functions on the same MCH.

### 3.13.5 X.25 NPSI definition

The purpose here is to connect TCP/IP z/OS to TCP/IP AIX via X.25 NPSI as shown in Figure 3-44. The following steps to create this connection must be observed:

1. Create X.25 configuration statements in PROFILE.TCPIP. Basically, we have to create one X25NPSI DEVICE and LINK statement pair pointing to an X.25 configuration data set. Also, we have to create routing entries for TCP/IP hosts connected via an X.25 network.
2. Create X.25 configuration statements in the X25CONF member. Basically, we have to specify the MCH LU name in order to establish a session between the X.25 TCP/IP application (RAIAX25) and the NPSI MCH (L08000A5). This MCH has GATE support, 18 logical channels, a packet size of 128, and a packet window size of 7. Also, we have to code up to 18 destination IP addresses and their DTE addresses as informed by PSDN.
3. Create 18 PU/LU pairs in the switched major node to represent each virtual circuit in VTAM.
4. Create an RAIAX25 application to establish a session with the MCH LU.
5. Create an MCH in the X.25 NPSI with 18 logical channels.

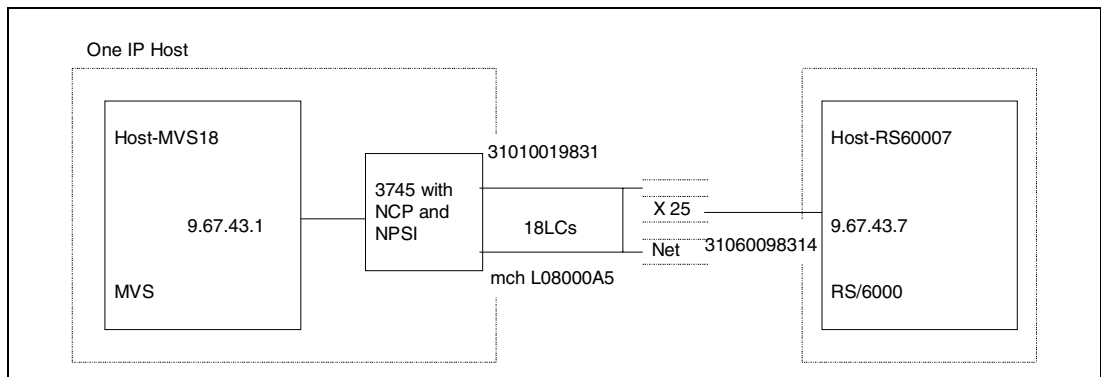


Figure 3-44 X.25 connectivity overview

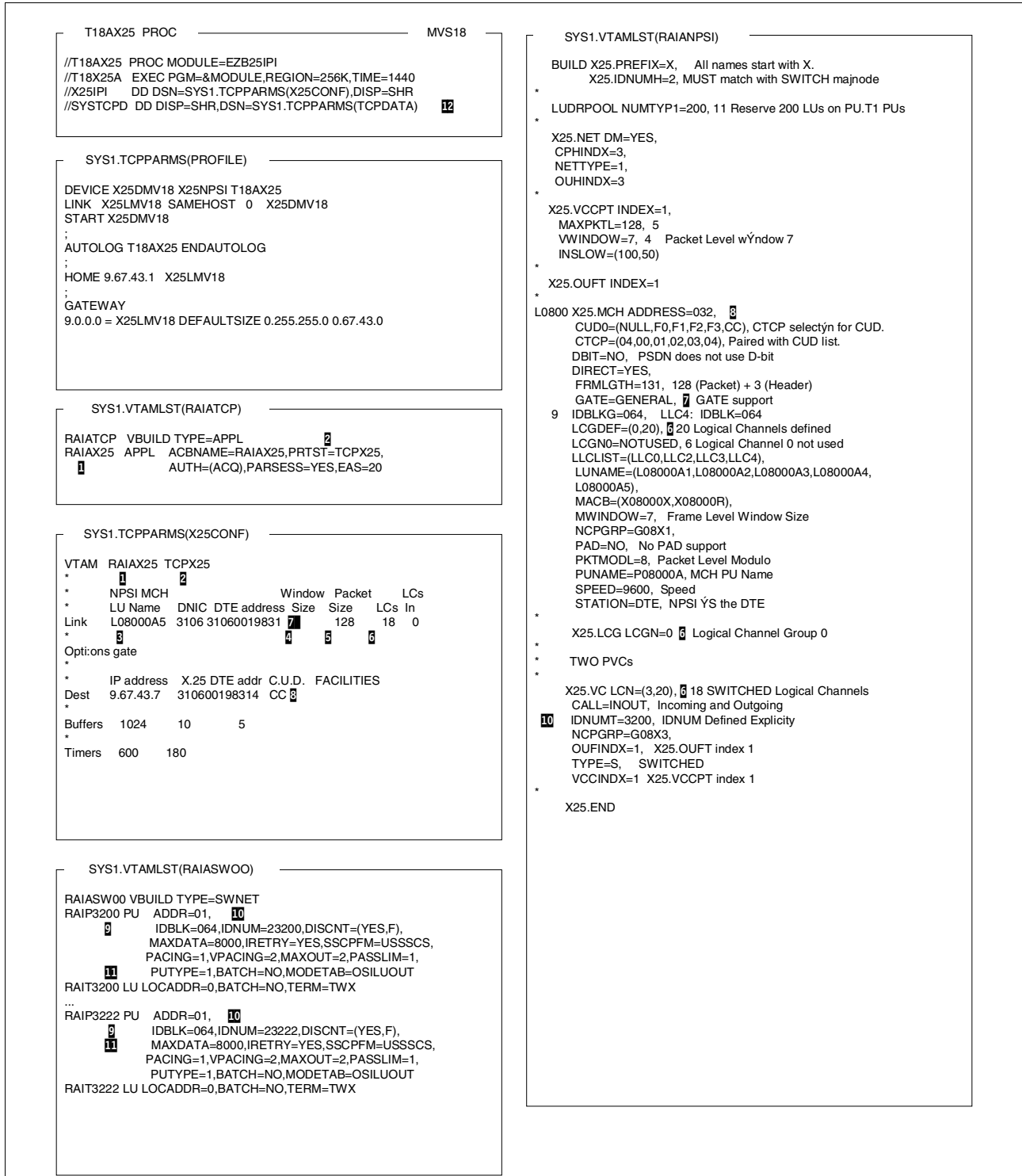


Figure 3-45 X.25 NPSI definition using static routing

The following explanations are related to Figure 3-45. Some of the markers are not explained in detail, but are included in the figure to show the relationship between definitions in different configuration data sets.

**4** VWINDOW keyword must match the WINDOW SIZE parameter in the LINK statement. It specifies the value of the transmit or receive window used by the packet protocol for the VCs that use this entry. This value is coded according to your PSDN.

**5** PACKET SIZE parameter in the LINK statement must match the MAXPKTL keyword in the X25.VCCPT statement. It specifies the length of packets to be sent or received over a virtual circuit. This value is coded according to your PSDN.

**6** Logical channels:

- ▶ The LCGDEF keyword specifies the highest logical channel number in decimal format for each logical channel group. We used LCGDEF=(0,20) to specify the Logical Channel Group (LCG) 0 with 20 logical channels.
- ▶ The LCGN keyword in the X25.LCG statement specifies the logical channel group number. In this case, we defined LCG 0.
- ▶ The LCN keyword in the X25.VC statement specifies the range of logical channels in decimal. We defined LCN=(3,20) to use from LCN 3 to LCN 20, which means 18 logical channels.
- ▶ The LCGN0 keyword specifies that LCN 0 is not used. It depends on the network characteristics.
- ▶ We created the logical channel group 0 with 20 logical channels: LCGDEF=(0,20). The logical channel 0 was reserved LCGN0=NOTUSED, and the logical channel 1 and 2 was not included in the logical channel group 0 LCN=(3,20), because they were used for other purposes. Thus we used 18 logical channels from 3 to 20.

**33** LUNAME, CUD0, and CTCP keywords are used to create a correspondence table to select the proper CTCP (TCP/IP) based on the call user data (CUD). In this case, the CTCP keyword says that TCP/IP is the fifth CTCP, and the CUD0 keyword says that every incoming call with CUD=CC or CUD=null will be directed to CTCP 04, which is represented by the L08000A5 LU name.

**9** The IDBLKG keyword in the X25.MCH statement specifies a IDBLK value for LLC4 (GATE) connections. We used IDBLK=064 in the switched major node since we specified IDBLKG=064 in the X25.MCH statement, instead of using the default value x'003' for all LLC types (LLCLIST keyword). The IDBLKG keyword is valid only in the X.25 NPSI V3R3 later. For X.25 NPSI V3R3 and previous releases, you have to specify IDBLK=003 in the switched major node.

**10** IDNUM

- ▶ For X.25 NPSI V3R2 and previous releases, an IDNUM is constructed with:
  - XX 2-byte prefix from X25.IDNUMH in the NCP BUILD statement
  - YYY 3-byte suffix generated by X.25 NPSI
- ▶ For X.25 NPSI V3R3 and later releases, an IDNUM is constructed with:
  - X 1-byte prefix from X25.IDNUMH in the NCP BUILD statement
  - YYYY 4-byte suffix generated by X.25 NPSI
- ▶ The IDNUM suffix corresponds to the resource order in the VTAMLST NCP source statement generated by NDF, which is the reverse order of X.25 NPSI input definitions for SVCs. It starts at x'0002' by increments of x'0002' to x'FFFF'. X.25 NPSI V3R3 allows you to define the IDNUMT keyword, which specifies the last four digits of the IDNUM value. We used IDNUMT=3200 to define the first IDNUM value. Thus, IDNUM starts at x'3200' by increments of x'0002' to x'3222' since we have only 18 logical channels defined in the LCN=(3,20) keyword.

**11** The LUDRPOOL statement must be used to enable dynamic allocation of LU control blocks for switched virtual circuits. The NUMTYP1 keyword is required when the LU is associated with a PU type 1.

**12** We discovered that the X.25 address space does not find TCPIP.DATA dynamically in the SYS1.TCPPARMS(TCPDATA). Thus, we had to use the //SYSTCPD DD statement to explicitly allocate TCPIP.DATA.

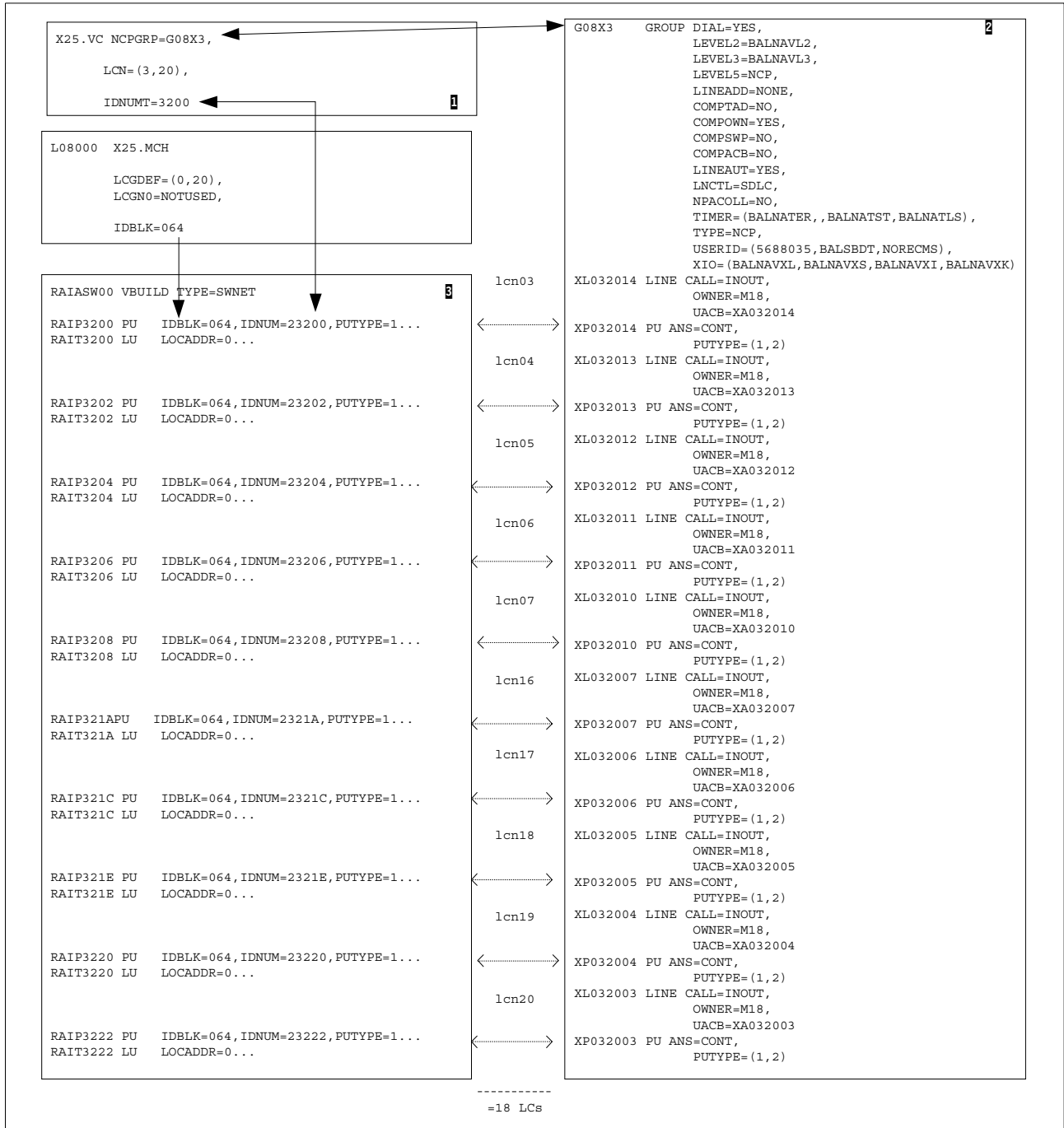


Figure 3-46 X25.VC generation by NDF

NDF formats the NPSI input **1** and produces a VTAMLST NCP major node **2** which contains LINE and PU definitions for every switched virtual circuit defined. You have to create one PU/LU pair in the switched major node **3** to match the PU dummy generated by NDF.

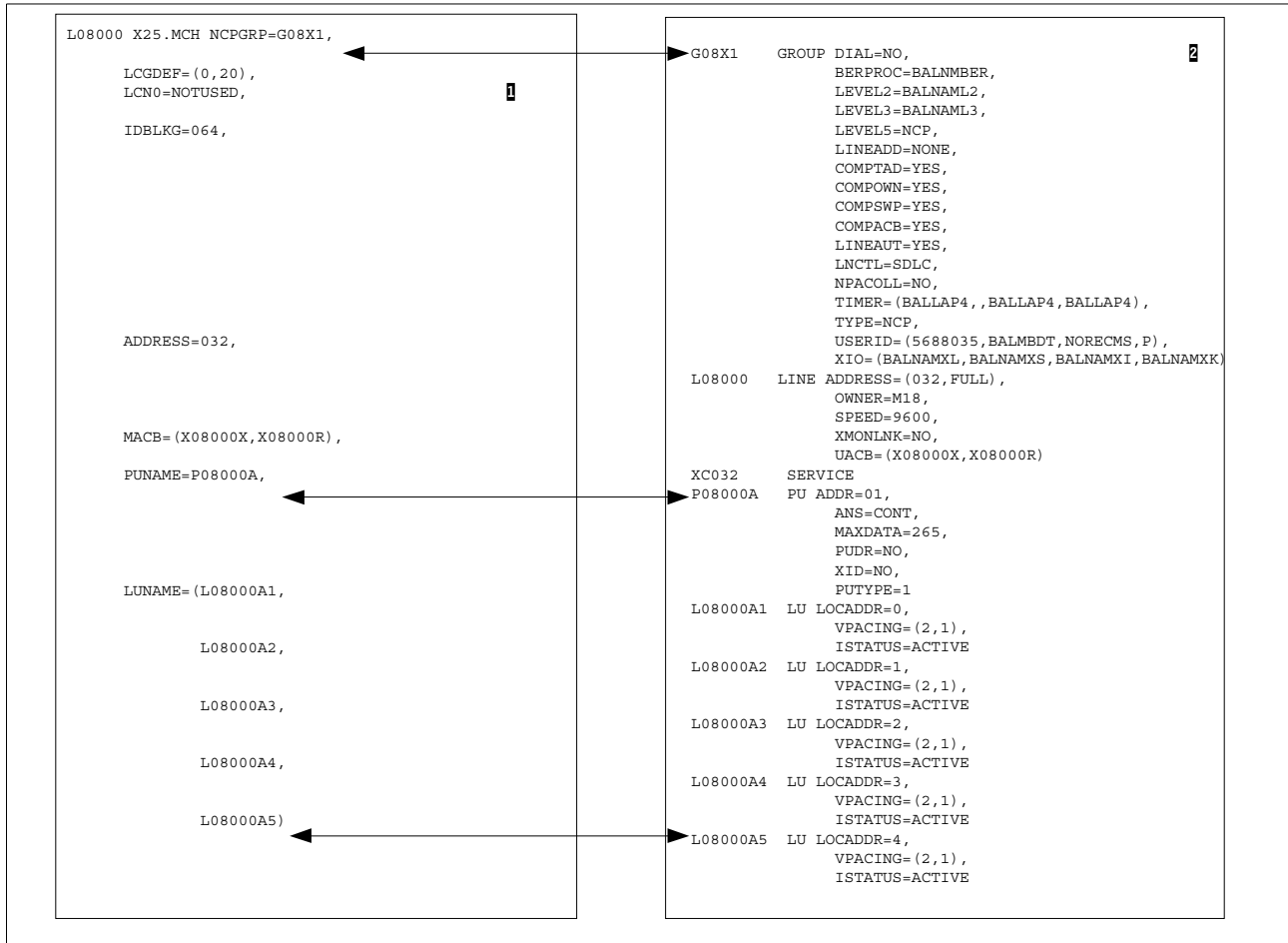


Figure 3-47 X25.MCH generation by NDF

NDF formats the NPSI input **1** and produces a VTAMLST NCP major node **2** which contains LINE definition for every MCH defined. In this case, the MCH has five LU names, and will be able to establish session up to five CTCPs (TCP/IP, CSFI, or X400).

### 3.14 XCF

Cross-System Coupling Facility (XCF) support for CS for z/OS IP allows IP traffic across the XCF connection being used by systems that are operating in the same sysplex group. The XCF environment should have been previously set up by the MVS systems programmer. VTAM will use XCF depending on whether the start option XCFINIT has been set. By default, VTAM will use XCF if it is available. A dynamically created local SNA major node called ISTLSXCF is then activated to represent the other VTAMs in the sysplex. VTAM will also add dynamic TRLE entries to the TRL major node, ISTTRL, to describe the connectivity characteristics used for XCF connections.

Since CS for z/OS IP uses the I/O facilities of VTAM, the same dynamic TRLE used for APPN's use of XCF must be shared by TCP/IP, and the deactivation of the XCF major node (ISTLSXCF) will also take down any TCP/IP connection that is using the XCF connection.

There are some special naming conventions that need to be followed in order for the devices to be used for XCF connection. If TCP/IP needs to communicate with another TCP/IP stack residing on a different host, TCP/IP must use the CPNAME of the remote VTAM host on the DEVICE statement representing the XCF connection. For a connection between two TCP/IP stacks on the same system, the device name should be called IUTSAMEH. The same naming convention needs to be followed if you are defining an Enterprise Extender connection.

Multiple TCP/IP instances in one host can share the same XCF connection to the same destination. APPN and IP traffic can share the same XCF link.

In Figure 3-48, you can see the TCP/IP definitions used in our test environment for XCF.

```
*****
XCF Definition
The DEVICE name must match the host name of VTAM on other side of XCF
The TRLE will be dynamically created if XCFINIT=YES in VTAM
start options
*****
DEVICE RAS MPCPTP autorestart
LINK RAS MPCPTP RAS
```

Figure 3-48 XCF definition in our PROFILE.TCPIP

### 3.14.1 Dynamic XCF

Starting with OS/390 eNetwork Communications Server V2R7 IP, you have a choice of defining the XCF connectivity to other TCP/IP stacks individually or using the dynamic XCF definition facility. This significantly reduces the number of definitions that you need to create whenever a new system joins the sysplex or when you need to start up a new TCP/IP stack. These changes become more numerous as the number of stacks and systems in the sysplex grows. This could lead to configuration mismatch and errors. With XCF Dynamics, you do not need to change the definitions of the existing stacks in order to accommodate the new stack.

XCF Dynamics uses the Sysplex Sockets support that was introduced in OS/390 V2R7 IP. Sysplex Sockets allows the stacks to communicate with each other and exchange information such as VTAM CPNAMEs, MVS SYSCONE value and IP addresses. A dynamic XCF definition is activated by coding the IPCONFIG DYNAMICXCF parameter in TCPIP.PROFILE.

XCF Dynamics creates dynamic definitions for DEVICE, LINK, HOME, BSDROUTINGPARMS statements, and the START statement. Dynamic XCF devices and links, when activated, appear to the stack as though they had been defined in the TCP/IP profile. They can be displayed using standard commands, and they can be stopped and started.

During TCP/IP initialization, the stack joins the XCF group, ISTXCF, through VTAM. When other stacks in the group discover the new stack, the definitions are created automatically, the links are activated, and the remote IP address for each link is added to the routing table. After the remote IP address has been added, IP traffic proceeds as usual.

In our system at the ITSO, we configured all of our TCP/IP stacks to use Dynamic XCF definition. Figure 3-49 shows the sysplex environment that we used for testing XCF connections among the different stacks.

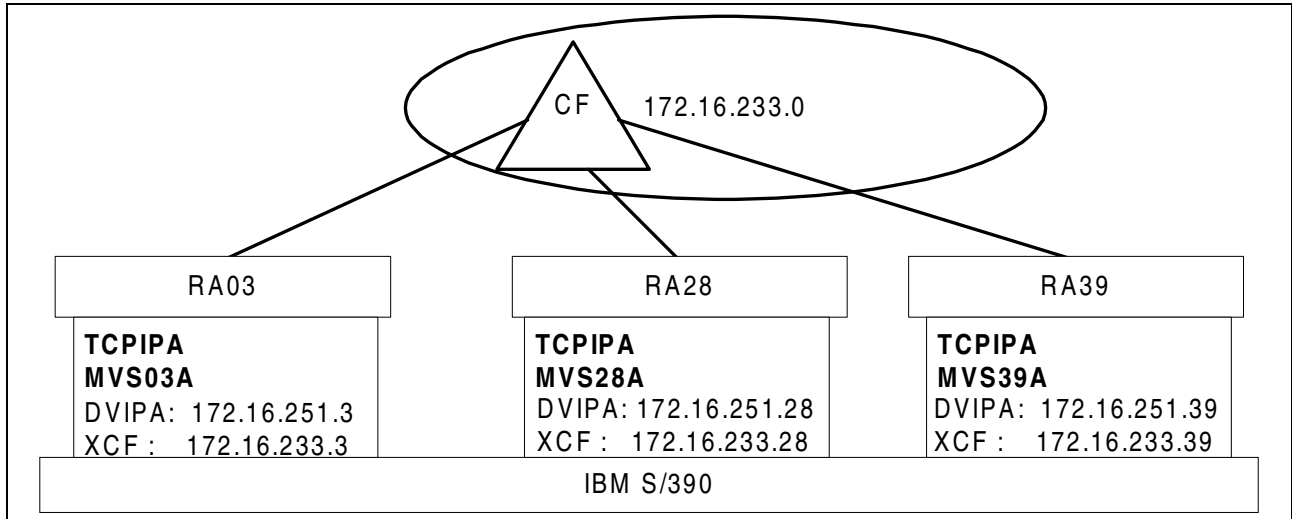


Figure 3-49 ITSO sysplex working environment for TCP/IP testing

Each stack is then enabled for Dynamic XCF definition by coding the following in the TCPIP.PROFILE:

```
IPCONFIG
DYNAMICXCF IPAddress subnet_mask cost_metric
```

The device and link names that will be used to represent the connection will be as follows:

- ▶ Connection to TCP/IP stack on the same system

TCP/IP will use the definitions shown in Figure 3-50 for the DEVICE and LINK statements if there are no devices and links existing with the same name.

```
DEVICE IUTSAMEH MPCPTP AUTORESTART
LINK EZASAMEMVS MPCPTP IUTSAMEH
HOME IPAddress EZASAMEMVS
BSDROUTINGPARMS EZASAMEMVS DEFAULTSIZE CostMetric SubnetMask DestIPAddress
START IUTSAMEH
```

Figure 3-50 XCF default device definition

**Note:** The DestIPAddress in the dynamically generated BSDROUTINGPARMS is always zero.

- ▶ Connection to TCP/IP stack on a different system

TCP/IP will use the CP name and the value of the system variable &SYSCLONE of the other systems in the sysplex to create the DEVICE and LINK statements to be used to represent the connection. The definitions shown in Figure 3-51 will then be created.

```
DEVICE <CPName> MPCPTP AUTORESTART
LINK EZAXCFnn MPCPTP <CPName>
HOME IPAddress EZAXCFnn
START <CPName>
BSDROUTINGPARMS EZAXCFnn DEFAULTSIZE CostMetric SubnetMask DestIPAddress
```

Figure 3-51 Dynamic XCF device statements are created dynamically



CPNAME represents the CP name of the VTAM hosting the TCP/IP stack in the other system and will contain the value of &SYSCONE (whatever that may be) in that other system. Dynamic XCF definitions will not be generated if the following conditions are present:

- There is an existing device name that has the same value as the CP name of the remote VTAM.
- There is a link name which will have the same value as EZAXCFnn and which will be dynamically generated from the SYSCONE of the remote system.

However, if TCP/IP does not find a device with the same CP name as the remote VTAM, but an existing link has the same value as the dynamically generated one, TCP/IP will proceed with the dynamic definition. This scenario is allowed, since the remote VTAM's CP name might have been changed, but it is not recommended.

You can see the XCF devices and links dynamically created for system RA03 in Figure 3-52.

```

DEVNAME: RA39M          DEVTYPE: MPC          DEVNUM: 0000
DEVSTATUS: READY
LNKNAME: EZAXCF39      LNKTYP: MPC          LNKSTATUS: READY
  NETNUM: 0  QUESIZE: 0
  BYTESIN: 0000550992  BYTESOUT: 0000689100
BSD ROUTING PARAMETERS:
  MTU SIZE: 32768      METRIC: 08
  DESTADDR: 0.0.0.0    SUBNETMASK: 255.255.255.0
MULTICAST SPECIFIC:
  MULTICAST CAPABILITY: YES
  GROUP              REFCNT
  -----
  224.0.0.5          0000000001
  224.0.0.1          0000000001
DEVNAME: RA28M          DEVTYPE: MPC          DEVNUM: 0000
DEVSTATUS: READY
LNKNAME: EZAXCF28      LNKTYP: MPC          LNKSTATUS: READY
  NETNUM: 0  QUESIZE: 0
  BYTESIN: 0000553520  BYTESOUT: 0000552752
BSD ROUTING PARAMETERS:
  MTU SIZE: 32768      METRIC: 08
  DESTADDR: 0.0.0.0    SUBNETMASK: 255.255.255.0
MULTICAST SPECIFIC:
  MULTICAST CAPABILITY: YES
  GROUP              REFCNT
  -----
  224.0.0.5          0000000001
  224.0.0.1          0000000001

```

Figure 3-52 Dynamic XCF devices in system RA03

By displaying a stack's configuration, you can also find out what will be the IP address that the other stacks will use for connection through XCF, plus the subnet mask and its assigned metric. You can see this in [Figure 3-53](#).

```

D TCPIP,TCPIPC,N,CONFIG
EZZ2500I NETSTAT CS V2R10 TCPIPC 016
TCP CONFIGURATION TABLE:
DEFAULTRCVBUFSIZE: 00016384 DEFAULTSNDBUFSIZE: 00016384
DEFLIMAXRCVBUFSIZE: 00262144
MAXRETRANSMITTIME: 120.000 MINRETRANSMITTIME: 0.500
ROUNDRIPGAIN: 0.125 VARIANCEGAIN: 0.250
VARIANCEMULTIPLIER: 2.000 MAXSEGLIFETIME: 60.000
DEFAULTKEEPALIVE: 0.120 LOGPROTOERR: 00
TCPFLAGS: 90
UDP CONFIGURATION TABLE:
DEFAULTRCVBUFSIZE: 00016384 DEFAULTSNDBUFSIZE: 00016384
CHECKSUM: 00000001 LOGPROTOERR: 01
UDPFLAGS: 3F
IP CONFIGURATION TABLE:
FORWARDING: YES TIMETOLIVE: 00060 RSMTIMEOUT: 00060
FIREWALL: 00000 ARP TIMEOUT: 01200 MAXRSM SIZE: 65535
IGREDIRECT: 00001 SYSPLXROUT: 00001 DOUBLENOP: 00000
STOPCLAWER: 00001 SOURCEVIP: 00001 VARSUBNET: 00001
MULTIPATH: NO PATHMTUDSC: 00000 DEVTRYDUR: 0000000090
DYNAMICXCF: 00001
IPADDR: 172.16.233.3 SUBNET: 255.255.255.0 METRIC: 01
SMF PARAMETERS:
INITTYPE: 00 TERMTYPE: 00 CLIENTTYPE: 00 TCPIPSTATS: 00
GLOBAL CONFIGURATION INFORMATION:
TCPIPSTATS: 00

```

Figure 3-53 Display of DYNAMICXCF configuration

### 3.14.2 Migrating from static XCF definition

If you have a statically configured XCF link definition, you may convert to a dynamically generated one by doing the following:

1. Stop the configured XCF device on all the systems.
2. Remove the configured XCF link from the HOME statement using the **VARY OBEYFILE** command.
3. Delete the configured XCF link and device using the **VARY OBEYFILE** command.
4. Activate DYNAMICXCF by using the **VARY OBEYFILE** command.



## Routing overview

This chapter provides a brief overview of z/OS Communications Server IP routing. For more details on this subject, please refer to:

- ▶ *TCP/IP Tutorial and Technical Overview*, GG24-3376
- ▶ *The Basics of IP Network Design*, SG24-2580
- ▶ *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775

In this chapter, we introduce general routing concepts and dynamic routing protocols.

## 4.1 IP addressing

IP addresses are represented by a 32-bit unsigned binary value, usually expressed in the form of four decimal numbers, one decimal number for each octet, for example 9.158.3.2. The IP address consists of a pair of numbers, the network number and the host number:

IP address = <network number> <host number>

Each IP address belongs to a class, and it is the class type which defines how the IP address is separated into its network and host parts.

### 4.1.1 IP address classes

There are five classes of IP addresses, as shown in Figure 4-1.

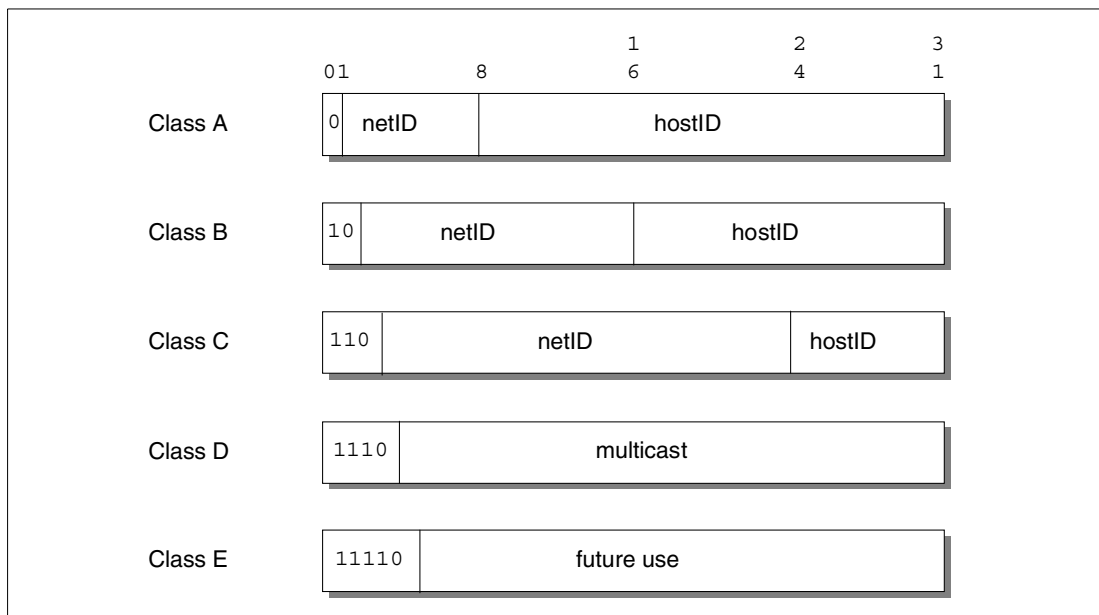


Figure 4-1 IP classes

- ▶ Class A addresses use 7 bits for the network portion and 24 bits for the host portion of the IP address. This allows for  $2^{24}-2$  (16777214) hosts; a total of over 2 billion addresses.
- ▶ Class B addresses use 14 bits for the network portion and 16 bits for the host portion of the IP address. This allows for  $2^{16}-2$  (65534) hosts; a total of over 1 billion addresses.
- ▶ Class C addresses use 21 bits for the network portion and 8 bits for the host portion of the IP address. This allows for  $2^8-2$  (254) hosts; a total of over half a billion addresses.
- ▶ Class D addresses are reserved for multicasting (a sort of broadcasting, but in a limited area, and only to hosts using the same class D address).
- ▶ Class E addresses are reserved for future use.

## 4.1.2 Reserved IP addresses

You will see IP addresses where part of the IP address has a value of *all bits 0* or *all bits 1*. These have a special meaning:

- ▶ All bits 0: an address with all bits zero in the host number portion is interpreted as *this* host. *All bits zero* in the network number portion is interpreted as *this* network.
- ▶ All bits 1: an address with all bits one in the host number portion is interpreted as *all* hosts. *All bits one* in the network number portion means *all* networks.

In this chapter, there are examples that calculate the number of available networks/hosts, for example  $2^{14}-2 = 16382$  networks/hosts. By subtracting two, the all zeros and all ones networks/hosts are not included in the number of available networks/hosts.

**Note:** Some routers do support the use of all zeros and all ones in the subnet field, but this is not consistent with the defined standards.

There is another IP address that is reserved: the class A network address 127.0.0.0, which is defined as the *loopback* network. Addresses on this network are assigned to interfaces that process data within the local system. The loopback interfaces do not access a physical network.

## 4.1.3 IP subnets

The use of IP address classes means that there is a finite number of networks available. By splitting the host portion of the IP address into two parts, a second network number and a host number, it is possible to divide the main network into a series of smaller networks. These smaller networks are called subnetworks or subnets.

IP address = <network number> <subnet number> <host number>

The assignment of subnets is performed by local network administrators; the entire network still appears as one IP network to the outside world.

The combination of the subnet number and the host number is often called the *local address* or the *local portion* of the IP address.

There are two types of subnetting: static and variable length. Variable length subnetting is more flexible than static. Native IP routing and the RIP Version 1 routing protocol support only static subnetting. RIP Version 2 and OSPF routing protocols support both types of subnetting.

*Static subnetting* implies that all subnets on one network use the same subnet mask. Static subnetting is simple to implement and easy to maintain, but can waste address space in small networks. For example, a network of four hosts using a subnet of 255.255.255.0 wastes 250 IP addresses. All hosts and routers are required to support static subnetting.

Consider the class A address 9.67.38.1. The network address is 9, and 67.38.1 is the host address. The 32-bit representation of this address is:

00001001 01000011 00100110 00000001 32-bit address

The network administrator may choose bits 8 to 25 to indicate the subnet address, in which case bits 26 to 31 would indicate the actual host addresses. A bit mask, known as the subnet mask, is used to identify which bits of the original host address indicate the subnet number. In this case the subnet mask would be:

11111111 11111111 11111111 11000000 or 255.255.255.192

By convention, the network mask is included in the subnet mask. This subnet mask allows for  $2^{18-2}$  (262142) valid subnets, each with  $2^{6-2}$  (62) hosts.

Applying the 255.255.255.192 subnet mask to the sample class A address 9.67.38.1 provides the subnet address:

```
00001001 01000011 00100110 00000001 = 9.67.38.1 (class A address)
11111111 11111111 11111111 11----- = 255.255.255.192 (subnet mask)
===== logical AND
00001001 01000011 00100110 00----- = 9.67.38.0 (subnet address)
```

This leaves a host address of:

```
----- ----- ----- --000001 = 1 (host address)
```

IP will recognize all host addresses for which this logical AND operation produces the same result as being on the same local network. This is important for routing IP datagrams in subnet environments.

*Variable length subnetting* allows subnets within the same network to use different subnet masks. Variable length subnetting divides the network so that each subnet contains sufficient addresses to support the required number of hosts. A small subnet with only a few hosts can use a mask that accommodates this need; a subnet with many hosts requires a different subnet. The ability to assign subnet masks according to the needs of the individual subnets helps conserve network addresses.

Consider class C network 165.214.32.0.

```
10100101 11010110 00100000 00000000 = 165.214.32.0
```

The network administrator wants to split this address range into three separate networks; one network will support 50 hosts, and the other two will each support 25 hosts. This cannot be achieved using static subnetting. Using static subnetting would allow the network to be divided into, for example, 2 subnets with 62 hosts each, or 6 subnets with 30 hosts each.

To divide the network into the three subnets required, multiple, variable-length, subnet masks are required. Firstly, using a mask of 255.255.255.192, the network is divided into two subnets, each with  $2^{6-2}$  (62) hosts.

```
11111111 11111111 11111111 11----- = 255.255.255.192 subnet mask
```

The two subnet addresses are:

```
10100101 11010110 00100000 01----- = 165.214.32.64
10100101 11010110 00100000 10----- = 165.214.32.128
```

Secondly, using a mask of 255.255.255.224 allows one of these subnets to be further divided into two subnets, each with  $2^{5-2}$  (30) hosts.

```
11111111 11111111 11111111 111----- = 255.255.255.224 subnet mask
```

If the first subnet, 165.214.32.64, is divided, the two subnet addresses would be:

```
10100101 11010110 00100000 010----- = 165.214.32.64
10100101 11010110 00100000 011----- = 165.214.32.96
```

and if the second subnet, 165.214.32.128, is divided, the two subnet addresses would be:

```
10100101 11010110 00100000 100----- = 165.214.32.128
10100101 11010110 00100000 101----- = 165.214.32.160
```

## 4.2 Classless Inter-Domain Routing

Standard IP addressing understands only class A, B and C network addresses. Within each of these networks, subnetting can be used to provide better granularity, but there is no way to specify that multiple class C networks are related. The result of this is termed the *routing table explosion* problem: a class B network of 3000 hosts requires one routing table entry at each backbone router, but the same environment addressed as a range of class C addresses requires 16 entries.

The solution to this problem is called Classless Inter-Domain Routing (CIDR). CIDR does not route according to the network class. It is based solely on the high order bits of the IP address. These bits are termed the IP prefix.

Each CIDR routing table entry contains a 32-bit IP address and a 32-bit network mask, which together give the length and value of the IP prefix. This is represented by <IP\_address network\_mask>. For example, to address a block of eight class C addresses with a single routing table entry, the following representation suffices: <192.32.136.0 255.255.248.0>. This would, from a backbone point of view, refer to the class C network range from 192.32.136.0 to 192.32.143.0 as one single network.

```
11000000 00100000 10001000 00000000 = 192.32.136.0 (class C address)
11111111 11111111 11111--- ----- = 255.255.248.0 (network mask)
===== logical AND
11000000 00100000 10001--- ----- = 192.32.136 (IP prefix)
```

```
11000000 00100000 10001111 00000000 = 192.32.143.0 (class C address)
11111111 11111111 11111--- ----- = 255.255.248.0 (network mask)
===== logical AND
11000000 00100000 10001--- ----- = 192.32.136 (same IP prefix)
```

This process of combining multiple networks into a single entry is referred to as *supernetting*. Routing is based on network masks that may be shorter than the natural network mask of an IP address. This is in contrast to subnetting, where the subnet masks are longer than the natural network mask.

Both dynamic routing protocols, OSPF and RIP, support supernetting.

## 4.3 Routing terminology

This section describes some of the more common IP routing-related terms and concepts. Most of the listed functions or protocols are supported by Communications Server for z/OS IP.

Table 4-1 Terms and concepts

Term	Definition
<b>Routing</b>	The process used in an IP network to deliver a datagram to the correct destination.
<b>Static routing</b>	Routing that is manually configured and does not change automatically in response to network topology changes.
<b>Replaceable static routes</b>	Static routes that can be replaced by OMPROUTE.
<b>Dynamic routing</b>	Routing that is dynamically managed by a routing daemon and automatically changes in response to network topology changes
<b>Routing daemon</b>	A server process that manages the IP routing table.
<b>Autonomous system (AS)</b>	A group of routers exchanging routing information through a common routing protocol. A single AS may represent a large number of IP networks.
<b>Router</b>	A device or host that interprets protocols at the Internet Protocol (IP) layer and forwards datagrams on a path towards their correct destination.
<b>Gateway</b>	A router that is placed between networks or subnetworks. The term is used to represent routers between autonomous systems.
<b>Interior gateway protocols (IGP)</b>	Dynamic route update protocols that are used between routers and hosts inside your own network and between your network and a service provider.
<b>Exterior gateway protocols (EGP)</b>	Dynamic route update protocols that are used between routers that are placed between two or more Autonomous Systems.

## 4.4 TCP/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 may include LANs and WANs, fast and slow, reliable and unreliable, inexpensive and expensive connections. The simplest way to connect them all 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 the worldwide 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 which are still linked and can communicate with one another.

IP is the de facto standard for most large routed networks. It is also the protocol used on the global Internet. In order to route packets, each network interface on a device (PC, UNIX workstation, router, mainframe, etc.) on the network has a unique IP address. Whenever a packet is sent, the destination and source addresses are included in the packet's header information. Routers examine the destination address to determine what to do with the packet.

### 4.4.1 Types of IP routing

There are two types of IP routing, direct and indirect.

*Direct routing:* 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. This is called direct delivery and is referred to as direct routing.

*Indirect routing* occurs 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 gateways. Note that in TCP/IP terminology, the terms *gateway* and *router* are used interchangeably. This describes a system that performs the duties of a router. The address of the first gateway (the first hop) is called an indirect route in the IP routing algorithm. The address of the first gateway 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 subnets, indirect routing is used to communicate between the two hosts. A router is needed to forward packets between subnets.

### 4.4.2 IP routing table

Every IP host is capable of routing IP datagrams and maintains an IP routing table. There are three types of entry in the IP routing table:

- ▶ Direct routes describing locally attached networks
- ▶ Indirect routes describing networks reachable through one or more gateways
- ▶ The default route, which contains the (direct or indirect) route to be used when the destination IP (sub)network is not found in any of the direct or indirect routes

Each entry contains a destination IP (sub)network address and the IP address of the next gateway (next hop) that the packet should be sent to.

As an example, let us look at Figure 4-2 which represents a simple network.

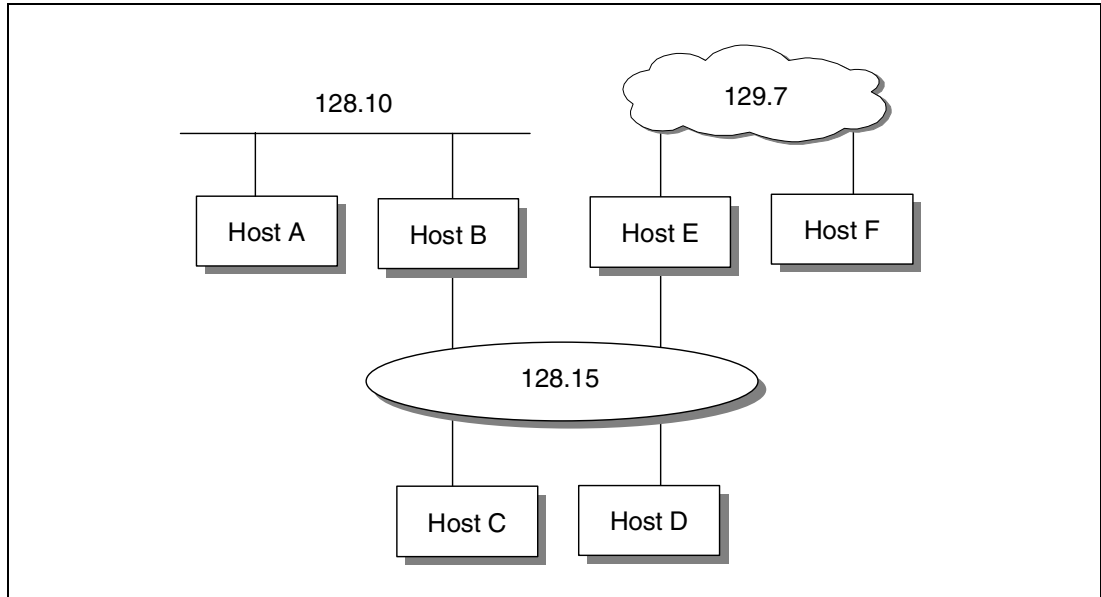


Figure 4-2 Sample point-to-point network

The routing table of host D might contain the following (symbolic) entries.

Table 4-2 IP routing table for Host D

Destination	Router
129.7.0.0	E
128.15.0.0	D
128.10.0.0	B
default	B
127.0.0.1	loopback

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

Since D is directly attached to network 128.15.0.0, it maintains a direct route for this network. To reach networks 129.7.0.0 and 128.10.0.0, however, it must have an indirect route through E and B, respectively, since these networks are not directly attached to it.

If an IP datagram is to be sent to a destination IP (sub)network, for which there is no explicit entry in the IP layer routing table of the sending IP host, such an IP datagram will be sent to the default router.

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

### 4.4.3 IP routing algorithm

IP uses a unique algorithm to route an IP datagram, called the *IP routing algorithm*. In a network without subnets, each host in the path from source host to destination host will:

1. Inspect the destination address of the packet
2. Divide the destination address into network and host addresses
3. Determine whether the destination host is on the same network
  - If so, send the IP datagram directly to the destination
  - If not, send the IP datagram to the next gateway as defined by the routing tables

In a subnetted network, each host in the path from source host to destination host will:

1. Inspect the destination address of the packet
2. Divide the destination address into subnetwork and host addresses
3. Determine whether the subnetwork is directly attached to it
  - If so, forward the packet directly to the destination
  - If not, forward the packet to the next router as defined in the routing tables

Another example of IP routing appears in Figure 4-3.

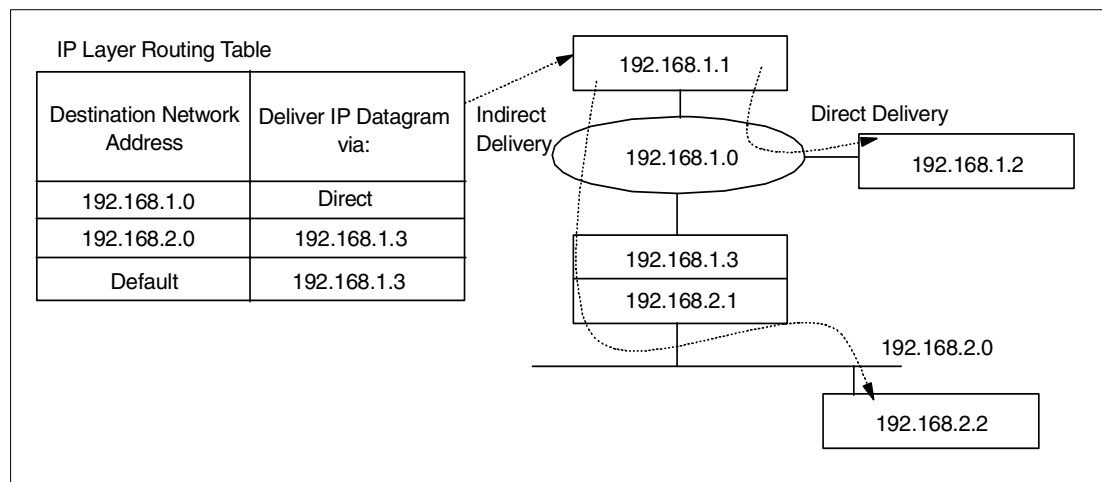


Figure 4-3 Basic routing mechanisms

If the destination IP (sub)network is directly connected, find the associated hardware address and deliver the datagram directly over the associated interface. This may require ARP processing.

- ▶ In Figure 4-3, one IP datagram is destined for host 192.168.1.2. The sending IP host is directly connected to the destination network, the Class C network 192.168.1.0, via the interface with address 192.168.1.1. The IP datagram will be sent across this interface and delivered directly to the destination IP host.

If the destination IP (sub)network is not directly connected, send the IP datagram to a directly connected IP router that has been selected as a first-hop towards the destination (sub)network, or send the IP datagram to a default router.

- ▶ In Figure 4-3, another IP datagram is destined for host 192.168.2.2. The destination network, the Class C network 192.168.2.0, is reachable via the router at address

192.168.1.3, which is on a directly connected (sub)network. The IP datagram will be sent to this router.

- ▶ If an IP datagram is to be sent to a destination (sub)network, for which there is no explicit entry in the IP layer routing table of the sending IP host, such an IP datagram will be sent to the default router at 192.168.1.3.

If the destination (sub)network cannot be found, report an error of Network Unreachable and discard the IP datagram.

#### 4.4.4 Displaying the IP routing table

The contents of the CS for z/OS IP layer routing table can be displayed using the TSO **NETSTAT ROUTE** or the **D TCPIP,tcpproc,NETSTAT,ROUTE** command.

The **NETSTAT ROUTE** command lists destination IP addresses and information about routes to those destinations. For each destination address, the following information is given: the gateway used when forwarding IP packets to that destination, the state of the route, the current number of active users for the route, and the link name for the route.

```
D TCPIP,TCPIPA,NETSTAT,ROUTE
EZZ2500I NETSTAT CS V1R2 TCPIPA 147 147
DESTINATION      GATEWAY      FLAGS      REFCNT  INTERFACE
DEFAULT          9.12.6.75    UGS        000000  OSA22EOLINK
9.12.6.0         0.0.0.0      UZ         000000  OSA22EOLINK
9.12.6.60        0.0.0.0      UH         000000  OSA22EOLINK
127.0.0.1        0.0.0.0      UH         000003  LOOPBACK
4 OF 4 RECORDS DISPLAYED
```

The **DEFAULT** value under Destination indicates that for routes not configured explicitly, this is the route that will be used.

The TSO **NETSTAT GATE** or **D TCPIP,tcpproc,NETSTAT,GATE** command gives details about each of the gateways, including packet size and subnet mask used.

EZZ2350I MVS TCP/IP NETSTAT CS V2R7		TCPIP NAME: T39ATCP			
EZZ2635I Known gateways:					
EZZ2636I	NetAddress	FirstHop	Link	Pkt Sz	Subnet Mask
	Value				
EZZ2637I	-----	-----	----	-----	-----
EZZ2638I	Default	9.24.104.3	TR1	17914	<none>
EZZ2638I	9.0.0.0	9.24.104.3	TR1	17914	<none>
EZZ2638I	9.0.0.0	9.24.104.3	TR1	17914	0.255.255.0 <b>2</b>
EZZ2638I	9.0.0.0	9.24.104.3	TR1	17914	0.255.255.240 <b>3</b>
EZZ2638I	192.168.41.12	9.24.104.3	TR1	17914	HOST
EZZ2638I	192.168.100.0	9.24.104.113	TR1	17914	<none>
EZZ2638I	192.168.233.0	<direct>	EZAXCF03	576	<none>
EZZ2638I	192.168.233.3	<direct>	EZAXCF03	55296	HOST <b>4</b>
EZZ2638I	192.168.233.4	<direct>	EZAXCF03	55296	HOST
EZZ2638I	192.168.233.28	<direct>	EZAXCF28	55296	HOST
EZZ2638I	192.168.233.29	<direct>	EZAXCF28	55296	HOST

Figure 4-4 The TSO **NETSTAT GATE** or **D TCPIP,tcpproc,NETSTAT, GATE** commands

**2** CS for z/OS IP often uses a subnet mask where the network part of the subnet mask is represented by zeros. In the above example, where 0.255.255.0 is displayed, most IP network protocols implemented would display it as 255.255.255.0.

3 First hop as an IP address means the connection is connected to that IP address.

4 In the FirstHop column, the <direct> indicates that direct connection. The subnet mask is host and this indicates that the link is a point-to-point.

For more information on these commands, refer to *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781.

#### 4.4.5 Static and dynamic routing

There are two ways to set up the necessary routing table for your CS for z/OS IP system, using static routing or dynamic routing.

*Static routing* requires you to manually configure the routing tables yourself. This is part of the configuration steps when you customize 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 the necessary routes may be simple for a small network and has the advantage of avoiding the network traffic overhead of a dynamic route update protocol. It also allows you to enforce rigid control on the allocation of addresses and resource access. However, it will require manual reconfiguration if you move or add a resource. In many configurations, static routing is sufficient to provide efficient routing. If your configuration is such that you do not have any backup route to remote networks, there is no need for dynamic routing.

*Dynamic routing* removes the need for static definition of the routing table. The router addresses and information detailing available paths to a destination will be built dynamically. These dynamically built routing tables are automatically exchanged between all the routers in your network. This sharing of the routing information enables the routers to calculate the best path through the network to any destination.

Static definitions are recommended if you have a very simple network or only one physical interface to the network. In this case, the routing table is very simple and there is no reason to choose dynamic routing. However, if your routing tables become more and more complex due to network growth and if the z/OS system must act as a gateway, then it is far easier to let the system do the work for you by using dynamic routing. Another reason for using dynamic routing is to take advantage of redundant network interfaces and to enable fault-tolerant network attachments.

#### 4.4.6 Maintaining routing tables

The most complex part of configuring TCP/IP network routing is establishing the routing tables. It is easy if you use a dynamic route update protocol, such as RIP or OSPF, on all your hosts, especially on the routers. But some hosts do not support RIP or OSPF, so you have to set up the static routing tables on your own.

In general, we categorize IP hosts into two major groups:

- ▶ Those IP hosts whose primary purpose it is to execute server or client application programs.

Most workstations that run OS/2, DOS/Windows or Windows 95 belong in this category. z/OS systems most often also belong in this category, since the z/OS system executes server programs that are used by client workstations in your IP network. Depending on your network topology, you may be able to define a single static default route on such

hosts. This is often desirable, especially for the bulk of IP hosts that are represented by workstations where we want to maintain as few and simple configuration parameters as possible. If more optimal routes than the default route exist for selected destinations, we can rely on normal ICMP redirects to optimize the IP layer routing tables in these IP hosts.

- ▶ IP hosts whose primary purpose is to interconnect different IP (sub)nets in your IP network and to route IP datagrams between these different sections of your network.

Such routers need to be able to react fast to changes in the IP network topology in order to compute new alternate routes in case sections of the network become unavailable. To do so, they need to participate in dynamic route update protocols, typically in the form of RIP Version 1, RIP Version 2, OSPF, or even for some routers, a mix of all three dynamic route update protocols. In certain configurations, a CS for z/OS IP system may belong in this category. This is especially true if you want to implement fault-tolerant network attachments for your CS for z/OS IP system. In such a configuration, your CS for z/OS IP system relies on dynamic route updates with neighboring routers to handle the situation where it becomes necessary to switch from one network interface to another in order to recover from a network malfunction.

IP route changes may be transparent if you use a dynamic route update protocol. This is because changes in network topology are noticed by the involved IP hosts, and information pertaining to the change is exchanged with all routing daemons that participate in dynamic route updates. Keep in mind, the network will need some time to converge. This is due to the propagation of route updates together with the time needed, in every router on the way to compute a new routing table. In the meantime, parts of the network may be unreachable.

## 4.5 Interior gateway protocols (IGP)

An interior gateway protocol (IGP) is a dynamic route update protocol used between dynamic routers running on TCP/IP hosts within a single autonomous system. The protocol is 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.

The interior gateway protocols supported by Communications Server for z/OS IP are:

- ▶ Open Shortest Path First (OSPF)

OSPF utilizes a Link State or Shortest Path First algorithm. The biggest advantage of OSPF over RIP is the time taken to converge after a network change. It will stabilize the propagated routing tables much faster than the distance vector protocols. However, in doing so, it will require more CPU time and may generate extra network traffic. OSPF is more complicated to configure than RIP.

- ▶ Routing Information Protocol (RIP)

RIP is the term commonly used to describe the Routing Information Protocol Version 1. It uses a distance vector algorithm to calculate the best path to a destination based on the number of hops in the path.

RIP Version 1 is easy to implement, but has several limitations, some of which are resolved in RIP Version 2.

► RIP Version 2

RIP Version 2 was created in order to address some of the limitations of RIP Version 1. It includes support for variable length subnetting and multicasting.

Table 4-3 Interior gateway protocol characteristics

Feature	RIP V1	RIP V2	OSPF
Algorithm	Distance Vector	Distance Vector	Link State
Network load <sup>1</sup>	High	High	Low
CPU processing requirement <sup>1</sup>	Low	Low	High
IP network design restrictions	Many	Some	Virtually none
Convergence time	Up to 180 secs.	Up to 180 secs.	Low
Multicast supported <sup>2</sup>	No	Yes	Yes
Multiple equal-cost routes	No <sup>3</sup>	No <sup>3</sup>	Yes

**Notes:**

<sup>1</sup> Depends on network size and (in)stability.

<sup>2</sup> By using multicast, we save CPU cycles on hosts that are not interested in certain periodic updates, such as OSPF link state advertisements or RIP V2 routing table updates. Multicast frames are filtered out either in the device driver or directly on the interface card if this host has not joined the specific multicast group.

<sup>3</sup> RIP in OMROUTE allows multiple equal-cost routes only for directly connected destinations over redundant interfaces.

## 4.6 Routing daemons

A daemon is a UNIX term for a background server process. Daemons are used for dynamic routing. The routing daemon adds, deletes or changes route entries within a host's routing table. The routing daemon also communicates with routing daemons on neighboring hosts to exchange topological routing information. This information exchange leads to the update of routing table entries.

CS for z/OS IP implements both RIP and OSPF dynamic routing protocols. There are two routing daemons, OMROUTE and ORouted. Both OMROUTE and ORouted are UNIX System Services applications and require the HFS.

### OMROUTE

OMROUTE is an IP routing daemon that supports RIP Version 1, RIP Version 2 and OSPF protocols. OMROUTE was introduced in OS/390 eNetwork Communications Server V2R6 and is the recommended routing daemon application for z/OS CS IP. It provides the technology to determine the most efficient route for use in the AS. OSPF is the preferred routing protocol in OMROUTE.

## **ORouted**

ORouted is an IP routing daemon that implements RIP Version 1 and RIP Version 2.

**Note:** OMPROUTE and ORouted will not run on the same TCP/IP stack.

## **4.7 Open Shortest Path First (OSPF)**

This section provides a brief overview of Open Shortest Path First (OSPF) routing protocol.

The OSPF protocol is based on link-state or shortest path first technology. In other words, OSPF routing tables contain details of the connections between routers, their status (active or inactive), their cost (desirability for routing) and so on. Updates are broadcast whenever a link changes status, and consists merely of a description of the changed status. OSPF can divide its network into topology subsections, known as areas, within which broadcasts are confined. It has been designed for the TCP/IP Internet environment.

OSPF features include the following:

- ▶ OSPF supports variable length subnetting.
- ▶ OSPF can be configured such that all its protocol exchanges are authenticated.
- ▶ Only trusted routers can participate in an AS that has been configured with authentication.
- ▶ In CS for z/OS IP, OSPF is configured using the UNIX daemon OMPROUTE. For more information about configuring OMPROUTE, see Chapter 6, “Dynamic routing with OMPROUTE” on page 103. OMPROUTE implements the OSPF protocol described in RCF 1583 (OSPF Version 2).
- ▶ Least-cost routing allows you to configure path costs based on any combination of network parameters. Bandwidth, delay and metric cost are some examples.
- ▶ No limitations to the routing metric. While RIP restricts the routing metric to 16 hops, OSPF has no restrictions.
- ▶ Allows multipath routing. OSPF supports multiple paths of equal cost that connect the same points. These paths are then used for network load distribution resulting in more use of the network bandwidth.
- ▶ OSPF’s area routing capability provides an additional level of routing protection and a reduction in routing protocol traffic

### **4.7.1 OSPF terminology**

This section describes some of the more common IP routing-related terms and concepts used in OSPF.

#### ***Router ID***

A 32-bit number allocated to each router in the OSPF network protocol. This number is unique in the autonomous system.

#### ***Areas***

OSPF networks may be divided into areas. An area consists of networks and routers that are logically grouped together. All routers within an area maintain the same topology database. Details of an area are not available to other areas in the same AS. All OSPF networks consist of at least one area, the backbone area. Areas are identified by an area number, which is an unsigned 32-bit word similar to an IP address that is by convention represented with the same dotted decimal notation as an IP address.



The division of an AS into areas improves performance and network resources are released for other functions. The size of link state advertisements, the link state database and routing tables are reduced. The link state database contains only information about an area within an AS, so convergence, flooding and synchronization of the database times are improved by the use of areas.

### ***Backbone area***

All OSPF networks must have a backbone area. The area identifier of the backbone area is always 0.0.0.0. The backbone area is special in that it distributes routing information to all areas connected to it. All routing information from one area to another should go through the backbone area. If an area is not physically connected to the backbone area, OSPF uses virtual links.

### ***Area border routers (ABR)***

These are routers that connect two or more areas. The area border routers maintain a topology database of each area to which it is attached. Area border routers also transmit link state advertisement during flooding across the areas to help synchronize the database across areas. All area border routers must have at least one interface in the backbone area.

### ***AS boundary routers***

These are the routers that connect the OSPF internetwork and exchange reachability information with other routers in other AS. They may use the exterior gateway protocols. The AS boundary routers are used to import static routes, RIP routes into the OSPF network (and vice-versa). AS boundary routing parameters are coded in the OSPF configuration file.

### ***Virtual link***

The virtual link is a logical link that connects an area that does not have a physical route to a backbone area. The link is treated as a point-to-point link.

### ***Neighboring routers***

Routers that have interfaces to the same network are called neighboring routers. To become neighbors, routers must belong to the same OSPF area, use the same security scheme, and have the same Hello and Dead intervals.

### ***Adjacency***

Neighboring routers are considered adjacent after they have exchanged link state information and synchronized their topology database.

### ***Link State Advertisement***

Link State Advertisement (LSA) is the unit of data describing the topology of the network and its adjacent routers. LSAs are flooded to other routers once the Hello protocol has established connection.

### ***Link state database***

Also called the topology database, the link state database contains the link state advertisements that describe the OSPF area. Each router within the OSPF area maintains an identical copy of the link state database.

### ***Flooding***

Flooding is the OSPF function that distributes link state advertisements and synchronizes the link state database between routers once the network topology has changed.

### ***OSPF Hello protocol***

The OSPF Hello protocol is used to detect and establish contact with neighboring routers. It dynamically maintains the relationship by periodically sending a Hello packet to all adjacent routers.

### ***Stub area***

A stub area is one that does not learn advertisements about destinations outside the AS. Instead, its area border routers advertise default routes to represent all those destinations. Stub areas can have more than one route but no virtual links can be defined through the stub area.

In Figure 4-5, area-1 within autonomous system-1 is configured as a stub area. You can see that area-1 has only one exit route to area-2 and cannot be used to configure virtual links. However, area-5 can have more than one route and can be configured as a stub area. Why? The answer is that area-5 cannot route directly to an AS using ASBRs but can only route within its own AS. The use of stub areas reduces the requirements for router storage and processing within an AS.

### ***Totally stubby area***

A totally stubby area is a stub area that does not learn about destinations external to the area (as opposed to the AS, which is what defines a stub area), and relies on default routes advertised by the ABRs to reach all destinations which are inter-area or external to the AS.

### ***Designated router***

A designated router is a router on a shared multi-access medium such as a LAN or ATM network that performs most of the routing functions for that network. It must be adjacent to all other routers on the medium. This saves all the routers on the medium from having to have mesh connected adjacencies; they all become adjacent with the DR who coordinates routing functions for all routers on the medium. The backup DR is also adjacent to all other routers on the medium, listens to the DR conversations and takes over if the DR fails.

OSPF's Hello protocol identifies a router that is adjacent to most routers in the network. This router is elected a designated router (DR). Normally, the first OSPF router to be identified in an IP network becomes a DR. The second router becomes a Backup DR. There is no need to configure a DR or Backup DR in an OSPF IP subnet network. The DR is available to all broadcast and nonbroadcast multi-access networks. It generates special functions in the maintenance of the network. It has the capability of generating LSAs and flooding them to all other adjacent routers. The DR synchronizes the link state database. Once the DR has been elected, this becomes the endpoint for most routers in the network.

Once the DR fails, the Backup DR becomes the DR and a new Backup DR is elected according to the router priority value. The router priority value is between 0 and 127. If you do not want a router to be elected a DR, you should configure it with a router priority of zero.

### ***Transit Area***

An area through which the virtual link ends. Remember that virtual links behave like point-to-point links. In Figure 4-5, area-2 and area-4 are transit areas.

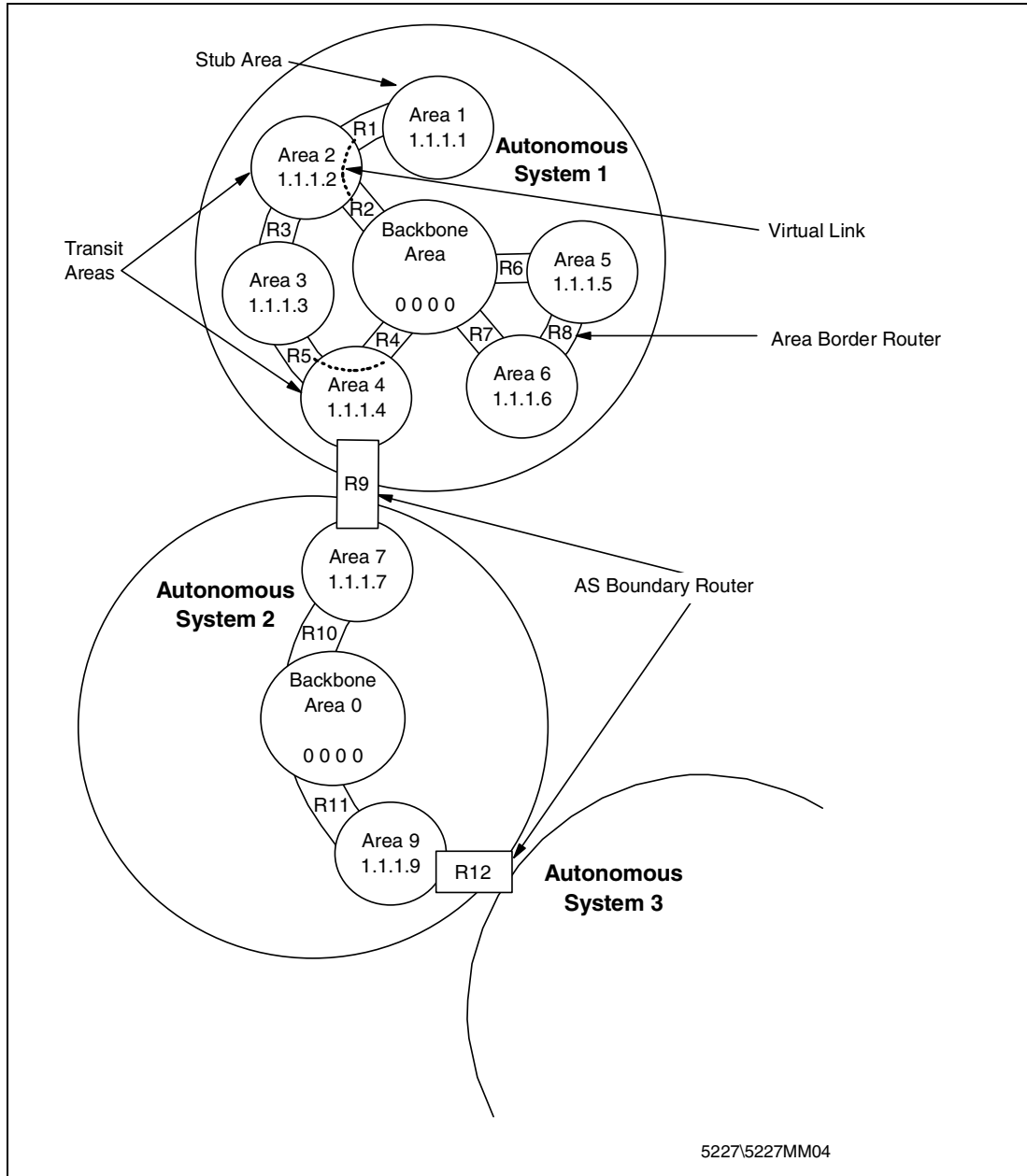


Figure 4-5 A Sample network diagram

## 4.7.2 Link state routing

The concept of link state routing is implemented by OSPF:

- ▶ Routers identify other routing devices on directly connected networks and exchange identification information with them.
- ▶ Routers advertise the details of directly connected network links and the cost of those links by exchanging link state advertisements (LSAs) with other routers in the network.
- ▶ Each router creates a link state database based on the link state advertisements: it describes the network topology for the OSPF area.
- ▶ All routers in an area maintain an identical link state database.

- ▶ A routing table is constructed from the link state database.
- ▶ OSPF is a dynamic routing protocol with all routers in an area being able to detect a topological change in the AS.

Link state advertisements are normally sent under the following circumstances:

- ▶ When a router discovers a new neighbor has been added to the area network.
- ▶ When a connection to a neighbor is unavailable.
- ▶ When the cost of a link changes.
- ▶ When basic LSA refreshes are transmitted every 30 minutes.

Each area has its own topology and has a gateway that connects it to the rest of the network. It dynamically detects and establishes contacts with its neighboring routers by periodically sending the Hello packets.

### 4.7.3 Link State Advertisements (LSAs)

As mentioned previously, OSPF routers exchange one or more link state advertisements (LSAs) with adjacent routers. LSAs describe the state and cost of an individual router's interfaces that are within a specific area, the status of an individual network component, and/or the link state database. There are five types of LSA:

- ▶ Router LSAs (Type-1) describe the state and cost of the routers interfaces within the area. They are generated by every OSPF router and are flooded throughout the area.
- ▶ Network LSAs (Type-2) describe all routers attached to the network. They are generated by the designated router, and are flooded through the area.
- ▶ Summary LSAs (Type-3) describe routes to destinations in other areas in the OSPF network. They are generated by an area border router.
- ▶ Summary LSAs (Type-4) are also generated by an area border router and describe routes to an AS boundary router.
- ▶ AS External LSAs (Type-5) describe routes to destinations outside the OSPF network. They are generated by an AS boundary router.

### 4.7.4 Link state database

The link state database is a collection of OSPF Link State Advertisements. OSPF, being a dynamic IP routing protocol, does not need to have routes defined to it. It dynamically discovers all the routes and the attached routers through its Hello part of the protocol. The OSPF Hello part of the protocol transmits Hello packets to all its router neighbors to establish connection. Once the neighbors have been discovered, the connection is made.

Before the link state databases are exchanged, however, the OSPF routers transmit only their LSA headers. After receiving the LSA headers, they are examined for any corruptions. If everything is fine, the request for the most recent LSAs is made. This process is bidirectional between routers. Once the Hello protocol has concluded that all the connections have been established, the link state databases are synchronized. This exchange is performed starting with the most recently updated LSAs. The link state databases are synchronized until all router LSAs in the network (within an area) have the same information. The link state protocol maintains a loop-free routing because of the synchronization of the link state databases.

Link state databases are also synchronized when two or more neighboring routers start communicating for the first time. They first synchronize their link state databases before they exchange information and start datagram forwarding. Thereafter, the OSPF Hello part of the protocol identifies when new routers are introduced and others become unavailable. As soon as there is a change in the link state databases, the process of synchronization begins. This synchronization process is called *flooding*.

Flooding starts when a router's status changes or a new router is introduced into the area network. The router sends an updated LSA to its neighboring router. The process is transferred from one router to another until all the routers in the area have been updated with the new changes.

The process counter checks that no corrupt or unwanted LSAs should be transmitted and updates the link state database. The original router periodically transmits the updated LSA until an acknowledgment from the receiving router is received. On the other hand, the receiving router checks the updated incoming LSA for any corruptions, duplication, and other unwanted information before accepting it. If the LSA is corrupt, it is rejected; otherwise, it is accepted and an acknowledgment transmitted back.

When displayed, the link state database gives a complete picture of the status of the network at that time. Figure 4-6 shows the link state database for a backbone area.

```

F T39BOMPR, OSPF, DATABASE, AREAID=0.0.0.0
      EZZ7853I AREA LINK STATE DATABASE 140
TYPE LS DESTINATION      LS ORIGINATOR      SEQNO      AGE      XSUM
  1 @9.3.1.1              9.3.1.1            0X80000801 520 0X8111
  1 @9.24.104.42          9.24.104.42        0X80000036 774 0X7BE1
  1 @9.24.104.43          9.24.104.43        0X8000002F 769 0X433D
  1 @9.24.104.113         9.24.104.113       0X80000036 1040 0X703C 1
  1 @9.24.104.149         9.24.104.149       0X80000034 9 0X2090
  1 @192.168.10.1         192.168.10.1       0X8000245C 568 0X9EF4
  1 @192.168.21.12        192.168.21.12     0X800013B5 1600 0X631D
  1 @192.168.31.12        192.168.31.12     0X80000854 553 0X78D7
  1 @192.168.41.12        192.168.41.12     0X800006E5 448 0XFB43
  1 @192.168.234.40       192.168.234.40    0X8000002F 517 0X31A0
  1 @192.168.251.4        192.168.251.4     0X80000033 486 0X387D
  2 @9.3.1.75             192.168.31.12     0X800000BC 553 0X22CC 2
  2 @9.3.240.1            9.3.1.1            0X800003A2 1115 0XB630
  2 @9.24.104.3           192.168.41.12     0X80000047 968 0XC4E8

      # ADVERTISEMENTS:      14
      CHECKSUM TOTAL:        0X64F27
  
```

Figure 4-6 Output from a link state database display command

The display shows the contents of an OSPF link state database from T39BOMPR, which is an OMPROUTE address space. The three fields which make up the LSA header are the LS Type, Link State ID (shown as LS Destination), and LS Advertising Router (shown as LS Originator). Only two of the five LSA types available are displayed here. The other three LSA types, 3, 4 and 5, are not displayed because they are used to implement hierarchical routing within OSPF: we did not implement hierarchical routing at ITSO-Raleigh.

**1** Shows LS Type 1 (router links advertisement) with Link State ID and LS Advertising Router as the same 9.24.104.113. This LSA is from 9.24.104.113.

**2** Shows LS Type 2 (network-LSA) with Link State ID 9.3.1.75 and LS Advertising Router as 192.168.31.12.

## 4.7.5 Physical network types

OSPF supports a combination of different physical networks. In this section, we give a brief description of each physical network and how OSPF supports them.

### ***Point-to-point***

A network that connects two routers together. A 56 Kbps serial line that connects two routers is an example of a point-to-point network.

### ***Point-to-multipoint***

Networks that support more than two attached routers with no broadcast capabilities. These networks are treated as a collection of point-to-point links. OSPF does not use designated routers on point-to-multipoint networks. The Hello protocol is used to detect the status of the neighbors.

### ***Broadcast multiaccess***

Networks that support more than two attached routers and are capable of addressing a single message to all the attached routers. OSPF's Hello Protocol discovers the adjacent routers by periodically sending/receiving Hello packets. This is a typical example of how OSPF makes use of a broadcast network to its advantage. OSPF utilizes multicast in a broadcast network if implemented. Token-ring and Ethernet are some of the examples of a broadcast network.

This is one of the two types of networks that uses a designated router.

### ***Nonbroadcast multiaccess (NBMA)***

Networks that support more than two attached routers but have no broadcast capabilities. Since NBMA does not support multicasting, the OSPF Hello packets must be specifically addressed to each router. Since OSPF cannot discover its neighbors through broadcasting, more configuration is required: all routers attached to the NBMA network must be configured. These routers must be configured whether or not they are eligible to become designated routers. All routers send periodic hellos to all other routers that are eligible to become designated routers. ATM and X.25 public data networks are examples of non-broadcast networks.

This is one of the two types of networks that uses a designated router.

## 4.7.6 A basic network with OSPF

A simple network containing two subnets is shown in Figure 4-7. There are four routers R1, R2, R3 and R4, and the cost of the routes between them is illustrated. Information about route costs is kept in the link state database, and each router in an OSPF area maintains a copy of the link state database. The routers will calculate the shortest paths, as shown in Table 4-4.

Table 4-4 Shortest paths between routers

Router	R1	R2	R3	R4
R1	n/a	cost=2	cost=1	cost=1
R2	cost=2	n/a	cost=1	cost=2
R3	cost=1	cost=1	n/a	cost=2
R4	cost=1	cost=2	cost=2	n/a

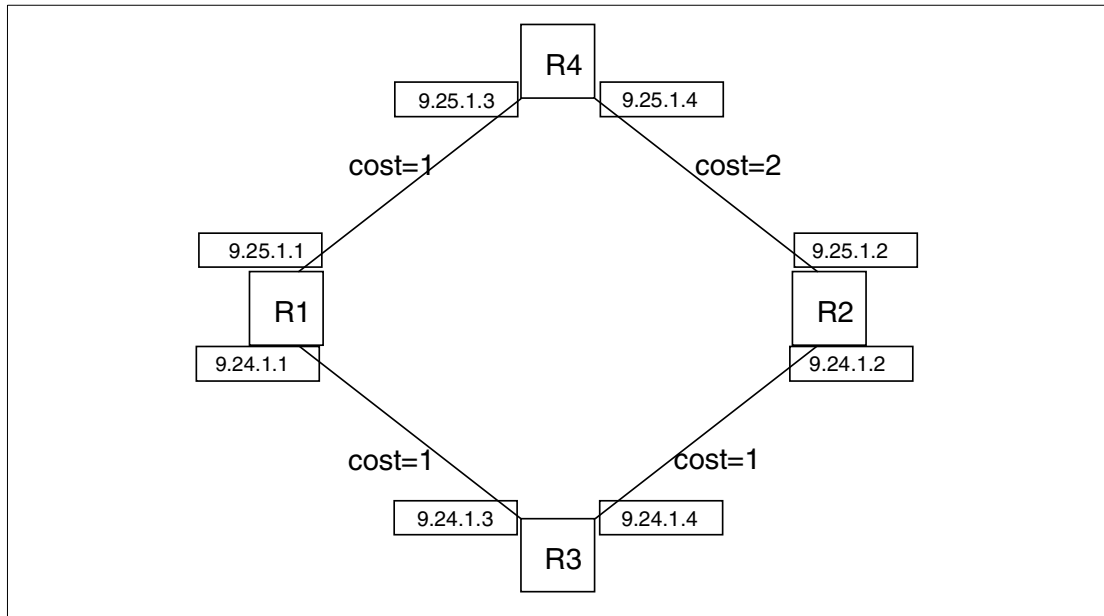


Figure 4-7 Simple network

In an OSPF environment, OSPF Hello packets are sent periodically (usually every ten seconds, based on Hello\_Interval) between neighboring routers. Routers send link state update messages describing their directly attached links when the network changes, or every 30 minutes.

Assume the network is stable for some time. Then the route between routers R1 and R4 fails. Router R1 or router R4 will detect that there is a failure. The failure will be detected within a maximum of 40 seconds. As soon as the failure is detected, the affected routers will initiate a link state advertisement (LSA). The new LSA will inform adjacent routers about the change in the topology. Routers R2 and R3 continue with the flooding process by sending to their adjacent routers. The shortest paths are recalculated and the link state database is updated with new link state routes.

Before the route failure, all packets from router R3 to R4 were being routed via R1, since this was the shortest path, with a route cost of 2. Following the network failure, these packets will be routed via R2, with a route cost of 3. Table 4-4 will change to show the new costs.

This is the basic algorithm used by OSPF in dynamic routing.

## 4.8 Routing Information Protocol (RIP)

This section provides an overview of the RIP protocol. RIP is designed to manage relatively small networks.

### 4.8.1 Distance vector algorithms

RIP uses a hop count (distance vector) to determine the best possible route to a network or host. The hop count is also known as the routing *metric*, or the *cost* of the route. A router is defined as being zero hops away from its directly connected networks, one hop away from networks that can be reached through one gateway, and so on. The fewer hops, the better. The route which has the fewest hops will be the preferred path to a destination. A hop count of 16 means infinity, or that the destination cannot be reached. Thus, very large networks with

more than 15 hops between potential partners cannot make use of RIP. The information is kept in a distance vector table, which is periodically advertised to each neighboring router. The router also receives updates from neighboring gateways and uses these to update its routing tables. If an update is not received for three minutes, a gateway is assumed to be down, and all routes through that gateway are set to a metric of 16 (infinity).

### ***Basic distance vector algorithm***

The following procedure is carried out by every entity that participates in the RIP routing protocol. This must include all of the gateways in the system. Hosts that are not gateways may participate as well.

- ▶ Keep a table with an entry for every possible destination in the system. The entry contains the distance  $D$  to the destination, and the first gateway  $G$  on the route to the network.
- ▶ Periodically, send a routing update to every neighbor. The update is a set of messages that contains all the information from the routing table. It contains an entry for each destination, with the distance shown to that destination.
- ▶ When a routing update arrives from the neighbor  $G'$ , add the metric associated with the network that is shared with  $G'$ . Call the resulting distance  $D'$ . Compare the resulting distance with the current routing table entries. If the new distance  $D'$  for  $N$  is smaller than the existing value  $D$ , adopt the new route. That is, change the table entry for  $N$  to have metric  $D'$  and gateway  $G'$ . If  $G'$  is the gateway from which the existing route came,  $G' = G$ , then use the new metric, even if it is larger than the old one.

## **4.8.2 RIP Version 1**

RIP is a protocol that manages IP routing table entries dynamically. The gateways using RIP exchange their routing information in order to allow the neighbors to learn of topology changes. The RIP server updates the local routing tables dynamically, resulting in current and accurate routing tables. See RFC 1058 for a complete definition of the RIP Version 1 protocol.

The protocol is based on the exchange of protocol data units (PDUs) between RIP servers (traditionally ORoutedD). There are various types of PDUs, but the two most important PDUs are:

- ▶ REQUEST PDU: sent from an ORoutedD server as a request to other ORoutedD servers to transmit their routing tables immediately.
- ▶ RESPONSE PDU: sent from an ORoutedD server to other ORoutedD servers either as a response to a REQUEST PDU or as a result of expiration of the broadcast timer (every 30 seconds).

The RIP V1 message format is extended as shown in Figure 4-8.



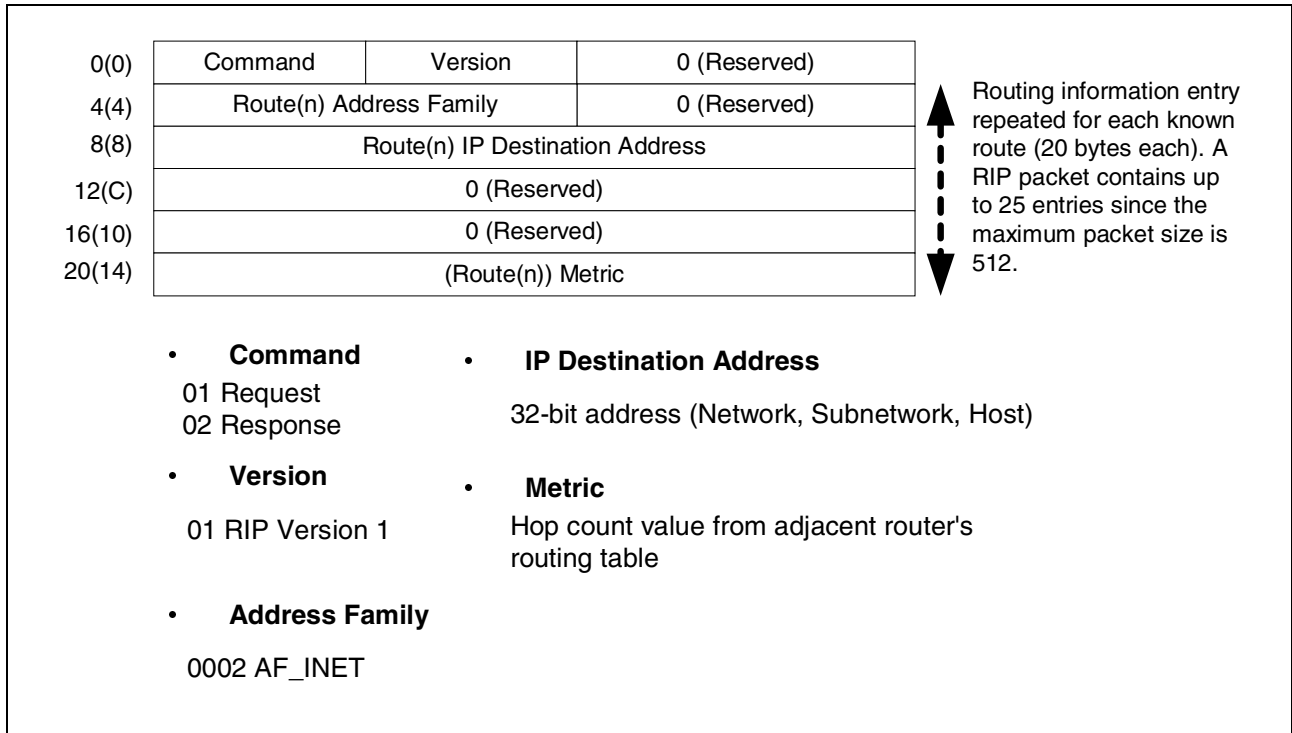


Figure 4-8 RIP V1 message

**Note:** Up to 25 routes may be included in a single RIP V1 message. With 25 routes, the message is 504 bytes long (25x20+4), which is the maximum size message that can be transmitted in a 512 bytes UDP datagram.

### 4.8.3 RIP V1 limitations

Since RIP is designed for a specific network environment, it has some limitations. These should be considered before implementing RIP in your network.

- ▶ RIP V1 declares a route invalid if it passes through 16 or more gateways. Therefore, RIP V1 places a limitation of 15 hops on the size of a large network.
- ▶ RIP V1 uses fixed metrics to compare alternative routes versus actual parameters, such as measured delay, reliability, and load. This means that the number of hops is the only parameter that differentiates a preferred route from non-preferred routes.
- ▶ The routing tables can take a relatively long time to converge or stabilize.
- ▶ RIP V1 does not support *variable subnet masks* or *variable subnetting* because it does not pass the subnet mask in its routing advertisements. Variable subnet masking refers to the capability of assigning different subnet masks to interfaces that belong to the same Class A, B or C network.
- ▶ RIP V1 does not support *discontiguous subnets*. Discontiguous subnets are built when interfaces belonging to the same Class A, B, or C network but to different subnets are not adjacent to each other. Rather, they are separated from each other by interfaces that belong to a different network. With RIP Version 1, *discontiguous subnets* represent unreachable networks. If you find it necessary to build discontiguous subnets, you must use one of the following techniques:

- An OSPF implementation
- A TCP/IP dynamic routing implementation that uses the RIP Version 2 protocol
- Static routing
- Routed with filtering and some static definitions to reach any discontinuous subnets

#### 4.8.4 RIP Version 2

RIP V2 extends RIP V1 functions and is specified in RFC 1721, 1722, 1723 and 1724. RIP V2 protocol extensions provide features such as:

##### ***Route tags to provide EGP-RIP and BGP-RIP implementation***

The route tags are used to separate "internal" RIP route (routes for networks within the RIP routing domain) from "external" RIP routes, which may have been imported from an EGP (external gateway protocol) or another IGP. OMROUTE does not generate route tags, but preserves them in received routes and readvertises them when necessary.

##### ***Variable subnetting support***

Variable length subnet masks are included in routing information so that dynamically added routes to destinations outside subnetworks or networks can be reached.

##### ***Immediate next hop for shorter paths***

Next hop IP addresses, whenever applicable, are included in the routing information. Its purpose is to eliminate packets being routed through extra hops in the network. OMROUTE will not generate immediate next hops, but will preserve them if they are included in RIP packets.

##### ***Multicasting to reduce load on hosts***

An IP multicast address 224.0.0.9, reserved for RIP Version 2 packets, is used to reduce unnecessary load on hosts that are not listening to RIP Version 2 messages. RIP Version 2 multicasting is dependent on interfaces that are multicast-capable.

##### ***Authentication for routing update security***

Authentication keys can be included in outgoing RIP Version 2 packets for authentication by adjacent routers as a routing update security protection. Likewise, incoming RIP Version 2 packets are checked against local authentication keys. The authentication keys are configurable on a router-wide or per-interface basis.

##### ***Configuration switches for RIP V1 and RIP V2 packets***

Configuration switches are provided to selectively control which versions of RIP packets are to be sent and received over network interfaces. You can configure them router-wide or per-interface.

##### ***Supernetting support***

The supernetting feature is part of the Classless InterDomain Routing (CIDR) function. Supernetting provides a way to combine multiple network routes into fewer *supernet* routes. Therefore, the number of network routes in the routing tables becomes smaller for advertisements. Supernet routes are received and sent in RIP V2 messages.

RIP V2 packets are backward compatible with existing RIP V1 implementations. A RIP V1 system will process RIP V2 packets but without the RIP V2 extensions and broadcast them as RIP V1 packets to other routers. Note that routing problems may occur when variable subnet masks are used in mixed RIP V1 and RIP V2 systems. RIP V2 is based on a distance vector algorithm, just as RIP V1 is.

The RIP V2 message format is extended as shown in Figure 4-9.

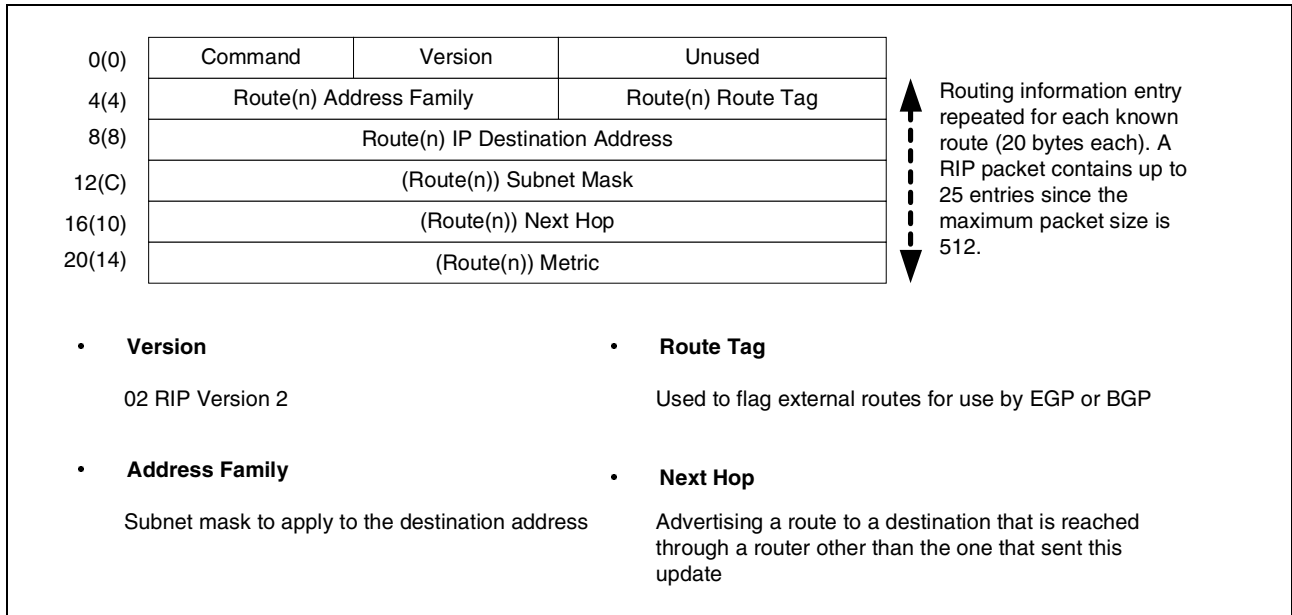


Figure 4-9 RIP V2 message

**Note:** The first entry in the message may be an authentication entry, or it may be a route as in a RIP V1 message. If the first entry is an authentication entry, only 24 routes may be included in a message; otherwise the maximum is 25 as in RIP V1.

### 4.8.5 RIP Version 2 sample configuration

RIP V2 extends RIP V1 functions and relieves most RIP V1 limitations. Using the RIP V2 protocol, you will be able to implement networks without many of those limitations.

Here are some examples of a RIP V2 network.

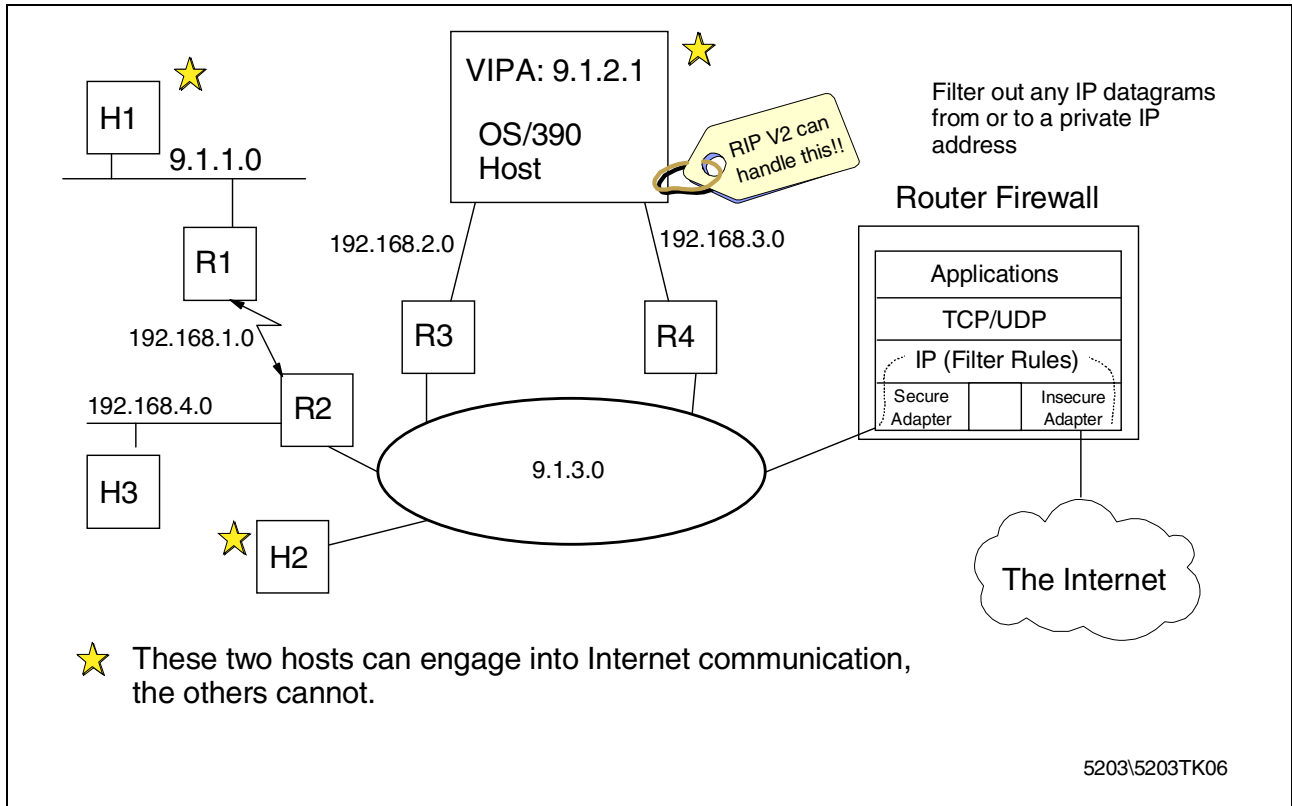


Figure 4-10 Private addresses on intermediate links

With RIP V2, you can make much better use of private addresses on intermediate links. Using private address spaces, you can use your limited IP address space more efficiently.

Figure 4-10 is an example of private addresses on intermediate links. All intermediate links in an intranet can be considered private, so they are not assumed to have the requirement for Internet communication.

The routers must be dedicated routers that never establish IP communication as an endpoint IP host with the outside world. The point-to-point link address never gets outside the local IP network. Only the officially assigned network address (in this case, the network address 9.0.0.0) is known outside. Hosts with the officially assigned address can communicate with the outside world through the external router.

As you can see in Figure 4-10, using a private address on intermediate links causes discontinuous subnets. While RIP V1 cannot support this network as we mentioned before, RIP V2 can be used in such a network.

A client inside this network can access an MVS host using a VIPA address configured on MVS, but not private addresses assigned to particular network interfaces. VIPA is discussed in more detail in Chapter 7, "Routing with VIPA" on page 141.

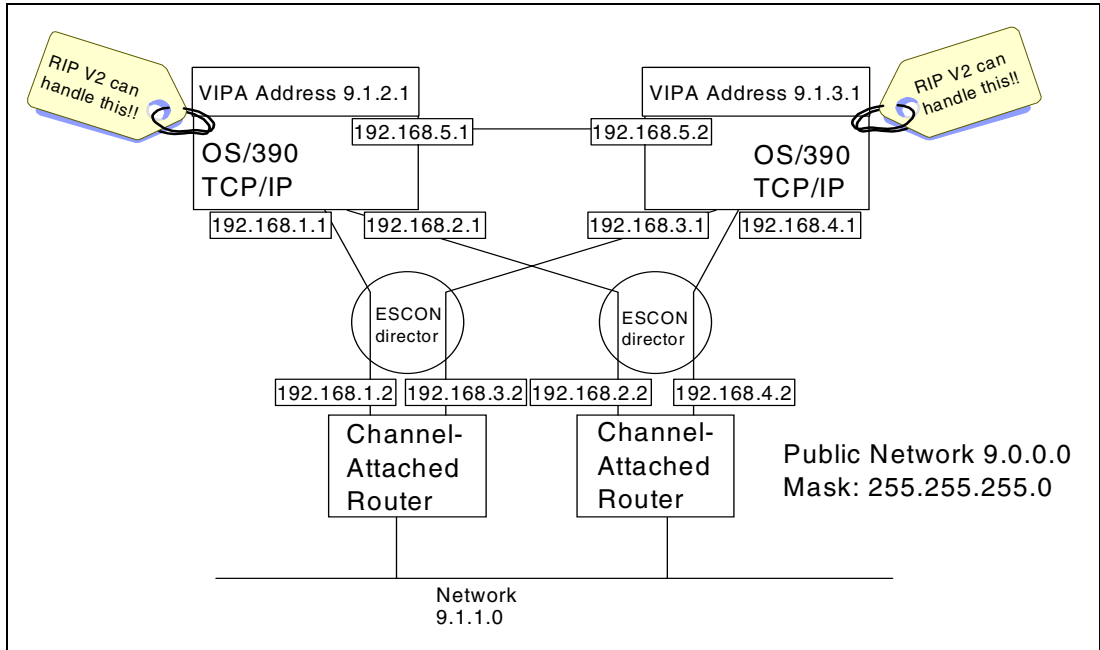


Figure 4-11 RIP V2 and channel attached routers

Figure 4-11 is another example of private addresses being used on links between an z/OS host and channel-attached routers. In this sample network, only the VIPA addresses and the backbone network use subnets of the public address spaces. Note that the VIPA addresses and the backbone network are examples of discontinuous subnets of the class A 9.0.0.0.

If you want to use the private address space for point-to-point links, such as a CTC or MPCPTP link, the only way is to use subnets of the public network, in this example the 9.0.0.0 network.

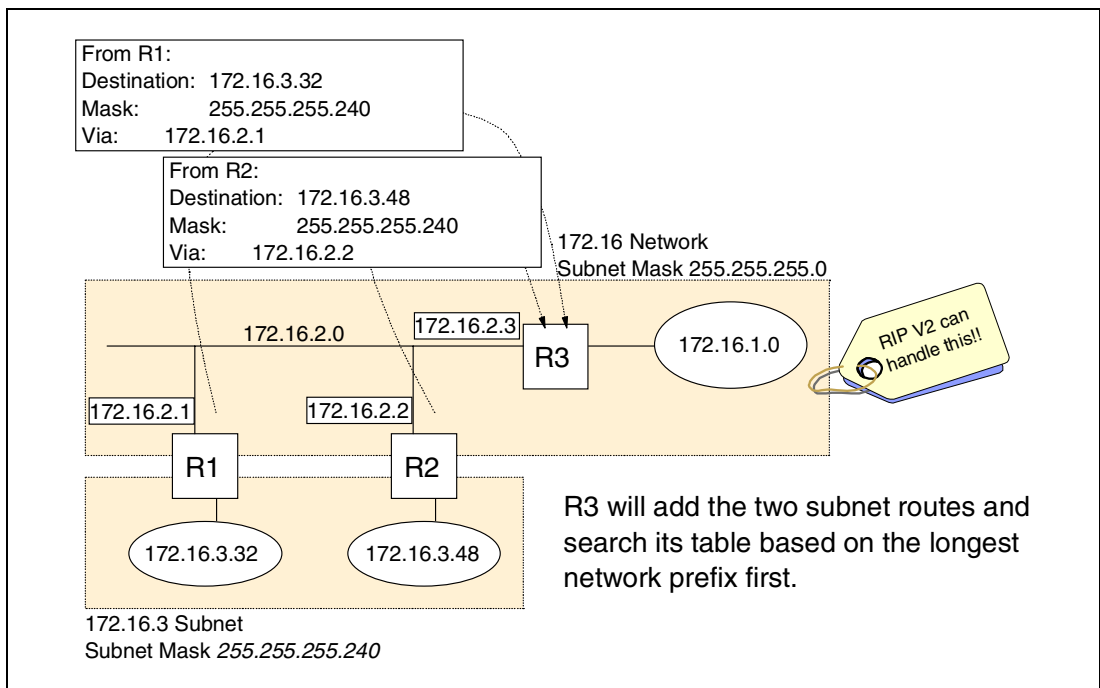


Figure 4-12 Variable length subnet mask

Figure 4-12 is an example of a variable-length subnet masks network. If every router in this figure supports the RIP V2 protocol, R3 will add the two routers to its routing tables when it receives a RIP V2 packet from R1 or R2. When R3 searches for a router in order to forward an IP packet, it will look in its IP routing tables based on the longest network prefix first.

Suppose R3 has an IP packet for 172.16.3.49 and its tables look like this:

- ▶ 172.16.3.32/255.255.255.255.240 via R1
- ▶ 172.16.3.48/255.255.255.255.240 via R2
- ▶ 172.16.2.0/255.255.255.255.0 directly connected
- ▶ 172.16.1.0/255.255.255.255.0 directly connected

Then R3 will select the second routing table entry, and transmit the IP packet to R2.

If you use RIP within the z/OS and OSPF in the IP backbone, it is required that intermediate routers export routing information between RIP and OSPF. On routers that support route exports, you configure per network interface which routing protocols are supported, and the router will then extract information from one protocol and export it into the other. Because Communications Server for z/OS IP now supports OSPF fully (with the use of OMPROUTE), we highly recommend *against* doing this.

## 4.8.6 RIP database

We said before that each entity keeps a routing database with one entry for every possible destination in the system. An actual implementation is likely to need to keep the following information about each destination:

<b>Address</b>	The IP address of the host or network.
<b>Subnet Mask Value</b>	The subnet mask value for the destination (sub)network. The subnet mask value is available only if RIP V2 is used in a network.
<b>Gateway</b>	The first gateway along the route to the destination.
<b>Interface</b>	The physical network that must be used to reach the first gateway.
<b>Metric</b>	A number indicating the distance to the destination.
<b>Timer</b>	The amount of time since the entry was last updated.



## Working with static routes

This chapter provides an overview of using static routing with Communications Server for z/OS IP. We introduce static routing concepts and illustrate how to *define* static routes. For more details on static routing, please refer to:

- ▶ *TCP/IP Tutorial and Technical Overview*, GG24-3376
- ▶ *The Basics of IP Network Design*, SG24-2580
- ▶ *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775
- ▶ *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776

Static routes are configured manually for each router by the system administrator. Available network interfaces and routes through the network must be determined before the routes are configured. Except for potential ICMP router redirections, routers do not communicate with each other about the topology of the network. If a destination (sub)network becomes unreachable, the static routes for that (sub)network remain in the routing table, and packets will still be forwarded to the destination. The only way to remove static routes from the routing table is for the network administrator to update the routing table.

Static routing is normally used in simple network topologies. There are other circumstances when static routing is useful:

- ▶ To define a default route that is used to forward traffic when the routing table does not contain a more specific route to a destination
- ▶ To define a route that is not automatically advertised within a network
- ▶ When line utilization or tariffs make it undesirable to send routing advertisements through lower capacity connections
- ▶ To ensure traffic for a specific destination takes a specific route through the network
- ▶ To provide a secure network by defining routes to authorized (sub)networks only
- ▶ Used in conjunction with the OMPROUTE routing daemon, to provide backup routes when OMPROUTE cannot find a dynamic route to a destination (replaceable static routes)

## 5.1 Replaceable static routes

With CS for z/OS V1R2, static routes can be defined as replaceable. In previous releases, static routes could not be replaced. Replaceable static routes are defined using the BEGINROUTES statement. TCP/IP treats replaceable static routes as last resort routes. If a dynamic route to the destination (sub)network exists, the dynamic route will replace the replaceable static route in the routing table. If that (sub)network subsequently cannot be reached via the dynamic route, the replaceable static route will be reinstalled. OMPROUTE is required to enable the use of replaceable static routes.

### Notes:

- ▶ The GATEWAY statement and routing daemon ORouted do not support replaceable static routes.
- ▶ You cannot define replaceable and non-replaceable static routes to the same destination.
- ▶ If a static route is defined as non-replaceable, it will behave as in previous releases, and override all other routes to that destination.

## 5.2 Defining static routes

There are two ways to define static routes, using the BEGINROUTES and GATEWAY statements in PROFILE.TCPIP. We recommend using the BEGINROUTES statement as it is easier to use, and it is compatible with UNIX standards. It uses BSD-style syntax to define IP addresses and address masks. Using BEGINROUTES, you can define static routes as replaceable. Also, in the future, any enhancements made to static routing will only be available via the BEGINROUTES statement.

Notes on the use of the BEGINROUTES statement:

1. The first BEGINROUTES statement in PROFILE.TCPIP or **VARY OBEY** command replaces the existing static routes within the TCP/IP routing table. All static routes are replaced, but dynamic routes created by routing daemons OMPROUTE or ORouted are not affected, except that nonreplaceable static routes will replace any dynamic routes to the same destination. Subsequent BEGINROUTES statements in the same data set add entries to the routing table.
2. You cannot define a BEGINROUTES-ENDROUTES block and a GATEWAY statement in the same configuration data set. If you do, the first statement found will be used and the other statement(s) will be discarded with warning messages being issued to the console. You may, however, use a BEGINROUTES-ENDROUTES block in the initial profile and a GATEWAY statement in a later OBEYFILE data set, and vice versa.
3. Non-replaceable static routes defined by the BEGINROUTES-ENDROUTES block cannot be deleted by the routing daemons OMPROUTE and ORouted. If you want OMPROUTE or ORouted to manage all routes, an empty BEGINROUTES-ENDROUTES block can be used to eliminate the static routes.
4. When an incorrect route entry statement is encountered, the route entry will be rejected with an error message, and the rest of the route entries in that BEGINROUTES-ENDROUTES block will still be processed. Subsequent BEGINROUTES-ENDROUTES blocks in the same profile or obeyfile are also processed.
5. A specific host route takes precedence over a subnetwork route, followed by a network route, followed by a supernet route, and finally, a default route.
6. You cannot define VIPA links on a ROUTE entry statement.



7. BEGINROUTES statements can only be coded for LINK names that exist in the home list when the statement is processed. Interfaces, such as dynamic VIPA and Dynamic XCF interfaces, might require the BEGINROUTES statements to be processed again after the interface is defined.

For information on coding the BEGINROUTES and GATEWAY statements, see *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776. The GATEWAY statement is class-based (that is, how you code the GATEWAY statement depends on which class, A, B or C, the IP address belongs to). We contrast the BEGINROUTES and GATEWAY statement with the following examples:

1. This example contains nonreplaceable static routes. Note that specifying a subnet mask of 255.255.255.0 has the same effect as specifying an address mask of /24. The use of the equal sign (=) for the next hop address indicates that the next hop address is the IP destination address of the packet.

```
BEGINROUTES
  ROUTE 9.24.104.0      255.255.255.0  =   TR2B   MTU 1492
  ROUTE 192.168.100.0  255.255.255.0  =   TR1B   MTU 1492
  ROUTE 9.179.98.0/24  9.24.104.200   TR2B   MTU 1492
  ROUTE DEFAULT        9.24.104.1     TR2B   MTU 1492
ENDROUTES
```

2. This example contains a route has been defined as replaceable. The route has a subnet mask of 255.255.255.0.

```
BEGINROUTES
  ROUTE 9.12.6.0 255.255.255.0 = OSA22EOLINK MTU 1492 REPL
  ROUTE DEFAULT 9.12.6.75      OSA22EOLINK MTU 1492
ENDROUTES
```

This example can also be coded using an address mask instead of the subnet mask. In this case, the /24 specifies the number of leftmost bits to be used as the address mask. 24 bits are to be used, 11111111 11111111 11111111 00000000 (255.255.255.0).

```
BEGINROUTES
  ROUTE 9.12.6.0/24 = OSA22EOLINK MTU 1492 REPL
  ROUTE DEFAULT 9.12.6.75 OSA22EOLINK MTU 1492
ENDROUTES
```

If the same routes were defined using the GATEWAY statement, this is how it would look:

```
GATEWAY
  9 = OSA22EOLINK 1492 255.255.255.0 0.12.6.0
  DEFAULTNET 9.12.6.75 OSA22EOLINK 1492
```

3. This example contains a supernet route. Note that the address mask specified in the ROUTE statement (/14) is shorter than the natural network mask of a Class B network address (16 bits or 255.255.0.0).

```
BEGINROUTES
  ROUTE 136.152.0.0/14 9.12.7.55 CIP1 MTU 4096
ENDROUTES
```

To define these routes using the GATEWAY statement, we would code:

```
GATEWAY
  136.152 9.12.7.55 CIP1 4096 0.3.0.0 0
```

To calculate the subnet mask in the GATEWAY statement, subtract the desired network mask (in this case 255.252.0.0) from the natural network mask (in this case 255.255.0.0). The result, 0.3.0.0, is specified as the subnet mask for the supernet route. As you can see, when defining a supernet route, the BEGINROUTES statement is much easier to understand.

This supernet route addresses the four Class B networks 136.152.0.0 through 136.155.0.0.

## 5.3 Displaying static routes

To display the routing table enter either of the following commands:

- ▶ D TCPIP,tcpproc,NETSTAT,ROUTE
- ▶ TSO NETSTAT ROUTE
- ▶ onetstat -r

The routing table display looks like this:

```
D TCPIP,TCPIPA,NETSTAT,ROUTE
EZZ2500I NETSTAT CS V1R2 TCPIPA 147 147
DESTINATION      GATEWAY      FLAGS      REFCNT  INTERFACE
DEFAULT          9.12.6.75    UGS        000000  OSA22EOLINK
9.12.6.0         0.0.0.0      UZ         000000  OSA22EOLINK
9.12.6.60        0.0.0.0      UH         000000  OSA22EOLINK
127.0.0.1        0.0.0.0      UH         000003  LOOPBACK
4 OF 4 RECORDS DISPLAYED
```

The flags have the following meaning:

- ▶ Flag U: the route is up
- ▶ Flag G: the route is a gateway
- ▶ Flag H: this is a host route
- ▶ Flag S: this is a static route that is not replaceable by a routing daemon
- ▶ Flag Z: this is a static route that can be replaced by dynamic routes learned by OMPROUTE

You may also see the following flags, although they are not illustrated in the above example:

- ▶ Flag C: this route was created by a connection (not using a definition or a routing protocol).
- ▶ Flag D: this route was created dynamically by a redirect
- ▶ Flag O: this route was created by OSPF
- ▶ Flag R: this route was created by RIP

To display static routes that have been defined as replaceable, use TSO command **NETSTAT ROUTE RSTAT**.

```
NETSTAT ROUTE RSTAT
MVS TCP/IP NETSTAT CS V1R2          TCPIP NAME: TCPIPA
Destination      Gateway      Interface
=====
9.12.6.0         0.0.0.0      OSA22EOLINK
```

For more information on the routing table display, see *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781.

## 5.4 Updating static routes

Static routes can be updated by:

- ▶ Replacing the routing table using the **VARY TCPIP, tcpproc, OBEYFILE** command
- ▶ Incoming ICMP redirect packets (if `IPCONFIG IGNOREREDIRECT` is not enabled)
- ▶ Incoming ICMP must fragment packets (if `IPCONFIG PATHMTUDISCOVERY` is enabled)

Also:

- ▶ **OMPROUTE**: if a static route is defined as being replaceable, **OMPROUTE** can replace it with a dynamic route, if a dynamic route is found to the same destination.

For more information on the **OBEYFILE** command, see *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781 and *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776. For details on `IPCONFIG IGNOREREDIRECT` and `PATHMTUDISCOVERY` options, see *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

## 5.5 Static routing and OMPROUTE

Static routes are defined using the **BEGINROUTES** or **GATEWAY** statement. When **OMPROUTE** initializes, it learns of static routes by reading the internal routing table created by TCP/IP. If static routes are changed during execution by a **VARY OBEY** command, TCP/IP will notify **OMPROUTE** of the changes.

**OMPROUTE** will advertise static routes to other routers if `IMPORT_STATIC_ROUTES=YES` is coded on the `AS_BOUNDARY OSPF` configuration statement, or `SEND_STATIC_ROUTES=YES` is coded on the `RIP_Interface` statement. **OMPROUTE** will only advertise static routes if these routes are available.

Using nonreplaceable static routes with **OMPROUTE** is typically not recommended because **OMPROUTE** cannot replace or modify nonreplaceable static routes, even if a lower cost dynamic route exists, or the static route is not available.

**OMPROUTE** will replace a replaceable static route if it finds a dynamic route to the destination. It will always use a dynamic route in preference to a replaceable static route. Replaceable static routes are treated as last resort routes, and will only be used if no dynamic routes exist to the destination. If a static route is replaced with a dynamic route, TCP/IP retains knowledge of the static route and can re-install it, should the destination become unreachable, using dynamic routes.

There are some situations where static routes might be required; for example, if you needed to define routes over an interface over which no routing protocol is used. Another situation is when multiple, equal-cost routes to a destination are needed and the RIP routing protocol is being used. This is because, with the exception of directly attached resources, the RIP protocol will not create multiple, equal-cost routes to a destination. If multiple adjacent routers are advertising via RIP that they can reach the same destination, RIP will add a route to the TCP/IP routing table via only one of those adjacent routers. If it is required that more than one route exist, they would have to be statically configured using the **BEGINROUTES** or **GATEWAY** statement.





## Dynamic routing with OMPROUTE

This chapter provides a brief overview of how to implement dynamic routing using OMPROUTE. For more details on this subject, please refer to:

- ▶ *z/OS V1R2.0 CS: IP Configuration Guide, SC31-8775*
- ▶ *z/OS V1R2.0 CS: IP Configuration Reference, SC31-8776*

Though not recommended, the UNIX routing daemon ORouteD can also be used to implement dynamic routing in CS for z/OS IP. For more information, see Appendix A, “Working with ORouteD” on page 181.

OMPROUTE is a UNIX routing application daemon. OMPROUTE implements the OSPF protocol, described in RFC 1583 (OSPF Version 2) and RFC 1850 (OSPF subagent protocol), as well as the RIP protocol, described in RFC 1058 (RIP Version 1) and RFC 1723 (RIP Version 2). The Communications Server for z/OS IP host running OMPROUTE becomes an active OSPF or RIP router in a TCP/IP network, or both. The OSPF and/or RIP protocol can be used to dynamically maintain the host routing tables.

## 6.1 OMPROUTE overview

OMPROUTE features include the following:

- ▶ OMPROUTE manages a TCP/IP stack's routing table. It is not involved in any decisions made by the stack when routing a packet to its destination.
- ▶ During initialization OMPROUTE deletes all dynamic routes from the TCP/IP stack's routing table. OMPROUTE then repopulates the stack routing table using information learned via the routing protocols.
- ▶ OMPROUTE does not make use of the BSDROUTINGPARMS statement. Instead, its parameters are defined in the OMPROUTE configuration file. See Figure 6-5 for sample OMPROUTE configuration statements. The OMPROUTE configuration file is used to define both OSPF and RIP environments.
- ▶ The OSPF and RIP protocols are communicated over interfaces defined with the OSPF\_INTERFACE and RIP\_INTERFACE configuration statements. Interfaces that are not involved in the communication of the RIP or OSPF protocol are configured to OMPROUTE via the INTERFACE configuration statement (unless it is a non-point-to-point interface and all default values specified on the INTERFACE statement are acceptable).
- ▶ 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.
- ▶ In an OMPROUTE environment, OSPF takes precedence over RIP and other routing protocols. If both OSPF and RIP protocols are used, OSPF routes will be preferred over RIP routes to the same destination.
- ▶ OMPROUTE allows generation of multiple, equal-cost routes to a destination, thus providing load-balancing support.
- ▶ OMPROUTE supports Virtual IP Addressing (VIPA) to handle network interface failures by switching to alternate paths. VIPA routes are included in the OSPF and RIP advertisements to adjacent routers. Adjacent routers learn about VIPA routes from advertisements and can use them to reach destinations at the z/OS.
- ▶ OMPROUTE uses the MVS operator console, syslogd, CTRACE, and STDOUT for its logging and tracing. The MVS operator console and syslogd are used for major events such as initialization, termination, and error conditions. CTRACE is used for tracing the receipt and transmission of OSPF/RIP packets as well as communications between OMPROUTE and the TCP/IP stack. STDOUT is used for detailed tracing and debugging.
- ▶ OMPROUTE can be started as an MVS procedure, from the z/OS shell, or from AUTOLOG.
- ▶ OMPROUTE must be started by a RACF-authorized user ID.
- ▶ OMPROUTE and ORouteD cannot run concurrently on the same stack.

Special care is needed when coding static routes in an OMPROUTE environment. There are two types of static routes: nonreplaceable (the default) and replaceable. OMPROUTE will replace a replaceable static route if it detects a dynamic route to the same destination. OMPROUTE cannot replace a nonreplaceable static route, even if it has detected a better, dynamic route to the same destination. For more information on using static routes with OMPROUTE, see Chapter 5, "Working with static routes" on page 97.

## 6.2 Configuring OMPROUTE

Some of the configuration of OMPROUTE is the same regardless of whether you are implementing RIP, OSPF, or both in your network. This section describes common configuration steps:

1. Create the OMPROUTE catalogued procedure (if starting OMPROUTE as an MVS procedure)
2. Define the OMPROUTE environment variables
3. Update the TCPIP.DATA or RESOLVER\_CONFIG file for use by the OMPROUTE application server address space
4. Update PROFILE.TCPIP
5. Create the OMPROUTE configuration file

Configuration steps specific to OSPF are detailed in “Configuring OSPF-only options” on page 117, and configuration steps specific to RIP are detailed in “Configuring RIP-only options” on page 118.

### 6.2.1 Create the OMPROUTE catalogued procedure

The OMPROUTE server is a UNIX System Services application and requires the HFS. If you plan to run OMPROUTE as an MVS started task, copy the sample OMPROUTE member from hlq.SEZAINST(OMPROUTE) to your system procedure library. Update the OMPROUTE in your system PROCLIB to suit your local configuration. You can start an OMPROUTE server from an MVS procedure library, from the UNIX System Services shell or from AUTOLOG. Since the OMPROUTE server uses raw sockets, it has to have sufficient authority. In most implementations, OMPROUTE would be started with superuser authority.

In our test system at ITSO-Raleigh, we started the OMPROUTE server as an MVS started task which had been associated with a superuser named *TCPIP3*.

All users who needed to start OMPROUTE had their user IDs defined to RACF with control access to MVS.ROUTEMGR.OMPROUTE. The following commands should be issued by a user ID with Special Operations to authorize users to start OMPROUTE:

```
RDEFINE OPERCMDS (MVS.ROUTEMGR.OMPROUTE) UACC(NONE)
PERMIT MVS.ROUTEMGR.OMPROUTE ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Figure 6-1 shows the procedure for the OMPROUTE server in our system.

```

/**
/** TCP/IP for MVS
/** SMP/E Distribution Name: EZBORPRC
/**
//OMPROUTE EXEC PGM=BPXBATCH,REGION=4096K,TIME=NOLIMIT,           1
//      PARM='PGM /usr/lpp/tcpip/sbin/OMPROUTE '
/**
//STDOUT  DD PATH='/tmp/OMPROUTE.39B.stdout',                       2
//      PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//      PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
/**
//STDERR  DD PATH='/tmp/OMPROUTE.39B.stderr',                       2
//      PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//      PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
/**
//STDENV  DD DSN=TCP.TCPPARMS(OM39BENV),DISP=SHR                   3
/**
//CEEDUMP DD DUMMY
/**CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)

```

Figure 6-1 Sample OMPROUTE procedure - T39BOMPR

1 The executable module is installed in the HFS with the sticky bit set.

2 STDOUT and STDERR data definition must be HFS files. MVS data sets are not supported by OMPROUTE servers on these DD statements.

3 You can use the STDENV statement to specify environment variables for use by the OMPROUTE server. You can use either an HFS file or an MVS data set for the STDENV statement. However, you have to use the PATH keyword to allocate an HFS file and the DSN keyword for an MVS data set, as shown in our sample procedure.

**Note:** To start OMPROUTE from the z/OS shell, use the following command:

```
omproute -tn -dn -sn
```

where -tn specifies the level of external tracing, -dn specifies the level of debugging and -sn specifies the level of subagent debugging.

## 6.2.2 Define OMPROUTE environment variables

OMPROUTE uses the following environment variables:

- ▶ RESOLVER\_CONFIG points to the TCPIP.DATA file
- ▶ OMPROUTE\_FILE points to the OMPROUTE configuration file
- ▶ OMPROUTE\_DEBUG\_FILE overrides the debug output destination
- ▶ OMPROUTE\_DEBUG\_FILE\_CONTROL specifies the size and number of debug output files

In the sample OMPROUTE procedure, in Figure 6-1, the STDENV statement is used to specify the environment variables:

```
//STDENV DD DSN=TCP.TCPPARMS(OM39BENV),DISP=SHR
```



Member OM39BENV points to two data sets, as shown in Figure 6-2.

```
RESOLVER_CONFIG=// 'TCP.TCPPARMS(TDATA39B) ' 1  
OMPROUTE_FILE=// 'TCP.TCPPARMS(OM39BCF) ' 2
```

Figure 6-2 Sample environment configuration file

While we used only MVS data sets in our test system at ITSO-Raleigh, you may use either HFS files or MVS data sets to store this configuration information.

## Resolver configuration file

The resolver configuration file contains the following two keywords which are used by OMPROUTE:

► TCPIPJOBNAME

OMPROUTE uses the TCPIPJOBNAME statement to determine with which TCP/IP stack to establish a connection.

► DATASETPREFIX

The value assigned to DATASETPREFIX will determine the high level qualifier (HLQ) used when searching for configuration files.

The resolver uses the following search order to allocate the actual resolver configuration file or data set to use:

1. GLOBALTCPIPDATA statement in System Resolver configuration file
2. The environment variable RESOLVER\_CONFIG
3. /etc/resolv.conf
4. The dataset specified on the SYSTCPD DD card
5. *userid*.TCPIP.DATA
6. SYS1.TCPPARMS(TCPDATA)
7. The hlq.TCPIP.DATA or DEFAULTTCPIPDATA statement in System Resolver configuration file

We have used the RESOLVER\_CONFIG environment variable to specify the actual resolver configuration file for our test environment at ITSO-Raleigh.

Partial contents of this file in our test system at ITSO-Raleigh are shown in Figure 6-3.

```
TCPIPJOBNAME T39BTCP 1  
DATASETPREFIX TCP 2
```

Figure 6-3 Sample resolver configuration file - TCP.TCPPARMS(TDATA39B)

**1** TCPIPJOBNAME is used by OMPROUTE to establish connection with TCP/IP stack T39BTCP.

**2** The value assigned to DATASETPREFIX in the TCPDATA will determine the high-level qualifier used when searching for the OMPROUTE configuration file. If the DATASETPREFIX is not defined, then TCPIP is used as a default.

## 6.2.3 Update PROFILE.TCPIP

In order to start an OMPROUTE server in a Communications Server for z/OS system, you have to update your PROFILE.TCPIP configuration data set. The following is part of a sample PROFILE data set we used at ITSO-Raleigh.

```
; Member TCP.TCPPARMS (PROF39B)
; *****
TCPCONFIG
  SENDGARBAGE FALSE
UDPCONFIG
  UNRESTRICTLOWPORTS
  UDPCHKSUM
IPCONFIG
  VARSUBNETTING      1
  IGNOREREDIRECT    2
  DATAGRAMFWD
  SOURCEVIPA
  ARPTO 1200
  TTL 60
  MULTIPATH          3
  SYSPLEXROUTING
;
  DYNAMICXCF 192.168.233.38 255.255.255.0 1
;If OMPROUTE is being used for dynamic routing the parameters
;should also be specified in the OMPROUTE configuration file.
;The statements to use are the OSPF_interface or RIP_interface
;depending on whether OSPF or RIP is used.

AUTOLOG 5
  T39BOMPR          4
ENDAUTOLOG

PORT
  520 UDP T39BOMPR  5

DEVICE TR1  LCS 2062 autorestart
LINK TR1  IBMTR 0 TR1
DEVICE VIPA39B VIRTUAL 0
LINK VIPA39B VIRTUAL 0 VIPA39B
HOME
  192.168.234.40 VIPA39B ; VIPA
  9.24.104.38 TR1 ; osa TO public network
START TR1
```

Figure 6-4 Sample TCP.TCPPARMS(PROF39B)

**1** OMPROUTE requires that VARSUBNETTING be configured in order to allow OSPF to add subnet routes. If NOVARSUBNETTING is coded, the message

```
EZZ0352I VARIABLE SUBNETTING SUPPORT IS DISABLED
```

will be displayed when the TCP/IP configured to run OMPROUTE is started. However, when OMPROUTE is started with VARSUBNETTING enabled, we receive the following messages (hence, even if it is not code, variable subnetting will be enabled):

```
EZZ7476I VARSUBNETTING IS ENABLED DUE TO ROUTING APPLICATION
BEING ACTIVE
BPXF024I (TCPIP3) EZZ7898I OMPROUTE INITIALIZATION COMPLETE
```

2 Using the OMPROUTE server implies that ICMP redirect messages will be ignored since OMPROUTE will control the routing table. OMPROUTE forces the IGNOREREDIRECT if it is not coded in the profile data set.

```
EZZ0335I ICMP WILL IGNORE REDIRECTS
```

3 Multipath is supported by OMPROUTE. This will enable data traffic to be distributed among all the possible paths.

4 If OMPROUTE is to be started automatically when TCP/IP starts up, specify the cataloged procedure name for your OMPROUTE server on the AUTOLOG statement.

5 If OMPROUTE is to be started from the UNIX System Services shell, the following PORT statement should be used to reserve the RIP port:

```
PORT
  520 UDP OMVS
```

#### Notes:

► If OMPROUTE is included in the AUTOLOG list, port 520 is reserved for OMPROUTE, and there is no application listening connection on port 520, TCP/IP will periodically cancel and restart OMPROUTE. If you are using the RIP protocol, there will be a listening connection on port 520. If you are using only the OSPF protocol, there will not be a listening connection on that port, and you should do one of the following to prevent TCP/IP from cancelling and restarting OMPROUTE:

- Do not reserve port 520
- Add the NOAUTOLOG parameter to the PORT statement, for example:

```
PORT
  520 UDP T39BOMPR NOAUTOLOG
```

Also, the monitoring and auto-restart features of AUTOLOG require a listening connection. If you are using OSPF only, there will be no listening connection, so if OMPROUTE stops for any reason it will not be restarted by AUTOLOG.

► If OMPROUTE will be communicating over token ring using OSPF or RIPv2 protocol with routers attached to the token ring that are not listening (at the DLC layer) for the token-ring multicast address 0xC000.0004.0000, the following TRANSLATE statement should be coded in TCPIP.PROFILE:

```
TRANSLATE 224.0.0.0 IBMTR FFFFFFFFFF linkname
```

Without this statement, OSPF and RIPv2 multicast packets will be discarded at the DLC layer by those routers which are not listening for the token ring multicast address.

For more information on PROFILE.TCPIP statements, refer to *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

## 6.2.4 Create the OMPROUTE configuration file

OMPROUTE uses a single configuration file to define the systems routing capabilities and network interfaces. The following search order is used to locate the OMPROUTE configuration file:

1. If the environment variable OMPROUTE\_FILE has been defined, OMPROUTE uses the value as the name of an MVS data set or HFS file to access the configuration data.

- The syntax for an MVS data set name is `/'MVS.DATASET.NAME'`.
- The syntax for an HFS file name is `/dir/subdir/file.name`.

2. /etc/omproute.conf
3. hlq.ETC.OMPROUTE.CONF

Figure 6-5 shows a typical OMPROUTE configuration file.

```

Area      Area_Number=0.0.0.0           1
          Stub_Area=NO
          Authentication_type=none;
OSPF_Interface IP_Address=9.24.104.38  2
          Name=TR1
          Cost0=6
          Subnet_mask=255.255.255.0
          MTU=4000;
OSPF_Interface IP_Address=9.24.105.73  3
          Name=EN2
          Subnet_mask=255.255.255.0
          Cost0=10
          MTU=1500;
OSPF_Interface IP_Address=192.168.*.*  4
          Subnet_mask=255.255.255.0
          MTU=32764;
RIP_Interface IP_Address=192.168.235.3  5
          Name=MPC25
          RipV2=YES
          Subnet_mask=255.255.255.0
          MTU=32764;
AS_Boundary_routing  6
          Import_RIP_Routes=YES
          Import_Direct_Routes=YES
          Import_Static_Routes=YES;
ROUTESA_CONFIG ENABLED=YES          7
          COMMUNITY="publicv2c";

```

Figure 6-5 A sample OMPROUTE configuration file

**1** This parameter sets up the OSPF area ID for any area in the network. In our example, the area is defined as Area\_Number=0.0.0.0 which is the backbone area, and must never be configured as Stub\_Area=YES. The Stub\_Area=NO indicates that this is not a stub area. If you specify Stub\_Area=YES, the area will not receive any AS external link statements. Stub areas are like dead ends and must never be configured for *virtual links*. Stub\_Area=NO indicates that this is not a stub and will be able to receive AS external link advertisements.

**2** This is the token ring OSPF interface definition. The IP address and name are the same as those defined in the HOME statement of the PROFILE.TCPIP configuration file.

**3** This is the definition for an Ethernet network IP address for OSPF\_Interface statement.

**4** This shows an OSPF\_interface with (\*) as a wild card. The IP address can be coded as a valid IP address that is configured on the network system, or a wild card could be specified. All configured interface IP addresses will be matched against possible wild cards in the order below, with x.y.z.\* being the best match, x.y.\*.\* being second, and so forth.

- ▶ x.y.z.\*
- ▶ x.y.\*.\*
- ▶ x.\*.\*.\*
- ▶ \*.\*.\*.\* is the same as coding ALL

**Note:** Be careful with MTU sizes. Adjacent routers must be able to receive packets of the defined MTU size. Otherwise, they will not exchange LSAs with each other.

5 This is the RIP\_Interface statement in the OMPROUTE configuration file. The interface is an MPCPTP connection with RIPv2 enabled.

6 AS\_Boundary\_routing enables OMPROUTE to import routes from other algorithms into the OSPF domain.

7 This parameter is used for the SNMP subagent. The OMPROUTE subagent community name must match the name defined in a data set configured to the SNMP agent or on the -c start option when the SNMP agent is started.

For more information, refer to *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

## 6.2.5 Configuring interfaces to OMPROUTE

There are three different interface statements to define interfaces in the OMPROUTE configuration file. These configuration statements are the OSPF\_Interface statement, the RIP\_Interface statement, and the Interface statement. Use the following guidelines when deciding which statement to use:

- ▶ Use the OSPF\_Interface statement if communication with other routers over the interface is going to be via the OSPF protocol.
- ▶ Use the RIP\_Interface statement if communication with other routers over the interface is going to be via the RIP protocol.
- ▶ Use the Interface statement for all other interfaces.

**Note:** The only exception to the above point is when the interface to be configured is a VIPA interface. For more information on configuring VIPA interfaces, see “Configuring VIPA interfaces (static and dynamic)” on page 116.

In the next sections, we discuss the process of configuring various types of interfaces:

- ▶ Configuring point-to-point interfaces
- ▶ Configuring point-to-multipoint interfaces
- ▶ Configuring non-broadcast network interfaces
- ▶ Configuring broadcast network interfaces
- ▶ Configuring VIPA interfaces

A sample configuration is included with each configuration statement.

**Note:** Not all parameters are coded on the sample configuration statements. Most parameters are allowed to take their default values unless there is a need to emphasize some parameters on some interfaces.

### Configuring point-to-point interfaces

Point-to-point interfaces include channel-to-channel (CTC) and CLAW interfaces. The DESTINATION\_ADDR parameter should be coded for these interfaces. This is to allow for the creation of a host route directly to the remote end of the interface. Figure 6-6 through Figure 6-8 show sample interface configurations for point-to-point interfaces.

```
OSPF_Interface
  IP_Address=9.24.104.113
  NAME=CTC800
  Subnet_Mask=255.255.255.254
  Destination_Addr=9.24.104.125
  Cost0=5;
```

Figure 6-6 Sample OSPF\_Interface statement for point-to-point interfaces

```
RIP_Interface
  IP_Address=9.24.104.33
  NAME=CTC600
  Subnet_Mask=255.255.255.254
  Destination_Addr=9.24.104.42
  RIPv2=YES;
```

Figure 6-7 Sample RIP\_Interface statement for point-to-point interfaces

```
Interface
  IP_Address=9.24.104.43
  NAME=CTC900
  Subnet_Mask=255.255.255.254
  Destination_Addr=9.24.104.65;
```

Figure 6-8 Sample Interface statement for point-to-point interfaces

**Note:** If another router is directly attached via a CLAW device and the OSPF protocol is communicating with that router, it is a requirement that the other router also be configured to view the CLAW device as a point-to-point interface. Failure to comply with this requirement will result in a failure to add any routes via that router.

## Configuring point-to-multipoint interfaces

Point-to-multipoint interfaces include such interfaces as MPC, XCF, and MPCPTP samehost. If OSPF or RIP will be used for communication with other routers over an interface of this type, it is required that OMPROUTE know the IP addresses of the neighbor routers to which it needs to transmit the OSPF or RIP packets. However, due to underlying signaling that takes place when a host connects to these network types, the TCP/IP stack is able to learn their IP addresses. OMPROUTE then learns the IP addresses from the stack when it connects. Therefore, there is no need to configure the destination addresses of these routers on the OSPF\_Interface and RIP\_Interface statements.

Figure 6-9 through Figure 6-11 show sample interface configurations for point-to-multipoint interfaces.

```
OSPF_Interface
  IP_Address=9.24.104.149
  NAME=XCF800
  Subnet_Mask=255.255.255.0
  Cost0=7;
```

*Figure 6-9 Sample OSPF\_Interface statement for point-to-multipoint interfaces*

```
OSPF_Interface
  IP_Address=9.24.104.149
  NAME=XCF800
  Subnet_Mask=255.255.255.0
  Cost0=7;
```

*Figure 6-10 Sample RIP\_Interface statement for point-to-multipoint interfaces*

```
Interface
  IP_Address=192.168.221.38
  NAME=XCFTR1
  Subnet_Mask=255.255.255.0;
```

*Figure 6-11 Sample Interface statement for point-to-multipoint interfaces*

## **Configuring non-broadcast network interfaces**

Non-broadcast network interfaces are interfaces such as HyperChannel and ATM. As a result, there is no underlying signaling with these networks to allow the TCP/IP stack to learn neighbor IP addresses. If OSPF or RIP is to be used for communication with routers over a non-broadcast network interface, it is a requirement that IP addresses of the neighbor routers be configured so that OSPF or RIP packets can be sent. OMPROUTE requires the DR\_Neighbor and/or NO\_DR\_Neighbor parameters on the OSPF\_Interface statement. DR\_Neighbor defines routers which can become the designated router, and NO\_DR\_Neighbor defines routers that cannot become the designated router. In addition, the OSPF\_Interface statement must include the Non\_Broadcast=Yes parameter to indicate that it is a non-broadcast multi-access interface and Router\_Priority parameter to assist in electing a designated router for the network. OMPROUTE requires the Neighbor parameter on the RIP\_Interface statement.

Figures 6-12 through 6-14 illustrate sample interface configurations for non-broadcast interfaces.



```
OSPF_Interface
  IP_Address=9.24.104.38
  NAME=HCH000
  Subnet_Mask=255.255.255.0
  Attaches_To_Area=0.0.0.0
  Cost0=3
  Router_Priority=2
  Non_Broadcast=yes
  DR_Neighbor=9.24.104.67
  No_DR_Neighbor=9.24.104.88;
```

Figure 6-12 Sample OSPF\_Interface statement for non-broadcast interfaces

```
RIP_Interface
  IP_Address=192.168.233.3
  NAME=ATMA02
  Subnet_Mask=255.255.255.0
  RIPv2=YES
  Neighbor=192.168.233.233;
```

Figure 6-13 Sample RIP\_Interface statement for non-broadcast interfaces

```
Interface
  IP_Address=192.168.233.4
  NAME=ATMB00
  Subnet_Mask=255.255.255.0;
```

Figure 6-14 Sample Interface statement for non-broadcast Interfaces

## Configuring broadcast network interfaces

Broadcast network interfaces are interfaces such as token ring, Ethernet, and FDDI. These network media allow for the broadcasting and multicasting of packets. Therefore there is no need to configure the IP addresses of the other routers on the network for OSPF or RIP packets to be transmitted. OMPROUTE uses broadcast for RIPv1 and multicasting for OSPF/RIPv2 address. The IP addresses of the other routers are learned during the exchange of OSPF/RIP packets between routers. The OSPF\_Interface statement for a broadcast network interface must include the Router\_Priority parameter to assist in selecting a designated router for the network.

Figures 6-15 through 6-17 show sample interface statements for broadcast interfaces.

```
OSPF_Interface
  IP_Address=9.24.104.113
  NAME=TR1
  Subnet_Mask=255.255.255.0
  Attaches_To_Area=0.0.0.0
  Cost0=2
  Router_Priority=1;
```

Figure 6-15 Sample OSPF\_Interface statement for broadcast interfaces

```
RIP_Interface
  IP_Address=9.24.104.34
  NAME=TR2
  Subnet_Mask=255.255.255.0
  RIPv2=YES;
```

Figure 6-16 Sample RIP\_Interface statement for broadcast interfaces

```
Interface
  IP_Address=9.24.104.38
  NAME=ETHE10
  Subnet_Mask=255.255.255.0;
```

Figure 6-17 Sample interface statement for broadcast interfaces

**Note:** Multicasting capability is required for any adapter that is being used to connect to the broadcast network if either the OSPF or the RIP Version 2 protocol will be used for communication. This is necessary to allow the transmission and receipt of the protocol packets that use multicast addressing on the broadcast network. Should multicast not be available, then the OSPF\_Interface and RIP\_Interface statements will need to be configured as for the non-broadcast network interface.

### Configuring VIPA interfaces (static and dynamic)

The purpose of VIPA is to free other hosts from dependence on a specific physical attachment to a z/OS TCP/IP stack. VIPA provides an IP address associated with a z/OS stack without being specific to any physical attachment. Inbound and outbound packets with VIPA destinations can be routed via any available physical network attachment.

If OMPROUTE uses only the RIP protocol, VIPA interfaces should be defined using the Interface statement as in the following example.

*Example 6-1 Interface statement to define a VIPA interface*

---

```
INTERFACE IP_Address=192.168.232.39
          Subnet_mask=255.255.255.0
          Name=VIPA39A
          MTU=32764;
AS_Boundary_Routing
          .....
          Import_Direct_Routes=Yes;
          .....
```

---

If OMPROUTE uses only the OSPF protocol, or both OSPF and RIP, VIPA interfaces should be defined using the OSPF\_Interface statement as in the following example.

*Example 6-2 OSPF\_Interface statement to define a PVIPA interface*

---

```
OSPF_Interface IP_Address=192.168.232.39
              Subnet_mask=255.255.255.0
              Name=VIPA39A
              MTU=32764;
```

---

For Dynamic VIPA (DVIPA), link names are assigned by the TCP/IP stack when the DVIPA is created; any name specified on the Interface or OSPF\_Interface statement will be ignored. Because a TCP/IP stack can have a large number of DVIPAs, ranges of DVIPA interfaces can be defined on one Interface or OPSF\_Interface statement. For example, defining the following two parameters on either the Interface or OSPF\_Interface statement:

```
IP_Address=9.24.104.160
Subnet_mask=255.255.255.240
```

would define all DVIPA interfaces in the range 9.24.104.160 through 9.24.104.175.

Define all VIPA and DVIPA interfaces that are usually active on the TCP/IP stack, and any that may be moved to the TCP/IP stack. VIPAs can be moved manually, or automatically, if using DVIPAs. For the VIPAs to be correctly processed and advertised by the routing protocols, they must be configured to OMPROUTE at the time they become active on the TCP/IP stack.

## 6.3 Configuring OSPF-only options

OMPROUTE has OSPF-specific options, most notably regarding authentication.

### 6.3.1 OSPF authentication

OSPF authentication is a means of verifying that OSPF packets are genuine. Packet contents are not protected, and packets are not encrypted. There are two ways to use OSPF authentication: password and MD5 cryptographic. All hosts on a network must be configured to use the same security scheme.

## Password

An 8-byte password is appended to all OSPF packets and sent in the clear with the rest of the packet. If the sent password matches the password defined by the packet receiver, the packet is accepted. To use password authentication, specify the following parameters on the OSPF\_INTERFACE configuration statement:

```
Authentication_Type=Password
Authentication_Key=your_password
```

## MD5 cryptographic

The OSPF packet and an MD5 key are summarized as a 16-byte message digest which is appended to the packet. The receiver creates a message digest using the OSPF packet and its MD5 key; if it matches the message digest sent, the packet is accepted. The MD5 key is a 16-byte hexadecimal string which is defined in the OSPF\_INTERFACE configuration statement:

```
Authentication_Type=MD5
Authentication_Key=MD5_key
```

The MD5 key can be obtained using the **pwtokey** command. For information on how to use the **pwtokey** command, see *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781.

## 6.4 Configuring RIP-only options

OMPROUTE also has RIP-specific options which are summarized here.

### 6.4.1 Selective RIP filters in OMPROUTE

OMPROUTE can use selective RIP filters against inbound and outbound traffic. Filters are implemented using the FILTER statement, the FILTER parameter on the RIP\_Interface statement (note that the FILTER parameter on the RIP\_Interface statement has more options), the Send\_Only statement, the Send\_Only parameter on the RIP\_Interface statement, and the IGNORE\_RIP\_NEIGHBOR statement. The use of RIP filters allows a specific destination address to be selected for filtering. Filtering can be used to limit the RIP broadcasting to a limited set of route types for sending or receiving. Conditions can be set on the RIP interface to filter only when certain terms are satisfied.

The syntax for coding filter statements is:

```
filter = (filter_type,dest_route,filter_mask)
```

The `dest_route` is the network destination, subnet or host IP address.

The `filter_mask` indicates the portion of the destination in the RIP broadcast to be compared with the destination on the filter statement. This parameter is optional and the default is 255.255.255.255.

The `filter_type` may be one of the following:

- ▶ `nosend`  
Do not broadcast routes matching the `dest_route` and `filter_mask`.
- ▶ `noreceive`  
Ignore broadcast routes that match the `dest_route` and `filter_mask`.

- ▶ send  
Routes matching the `dest_route` and `filter_mask` are to be broadcast over this interface only.
- ▶ send\_cond  
Routes matching the `dest_route` and `filter_mask` are to be broadcast over this interface only, when this interface is active. If the specified interface is inactive, the routes can be broadcast over other routes.
- ▶ receive  
Routes matching the `dest_route` and `filter_mask` are to be received over this interface only.
- ▶ receive\_cond  
Routes matching the `dest_route` and `filter_mask` are to be received over this interface only, when this interface is active. If the specified interface is inactive, the routes can be received over all other active interfaces.

Notes:

1. Multiple filter statements may be coded on a `RIP_Interface` statement.
2. Only filter types *nosend* and *noreceive* are valid on a globally coded filter statement.
3. An asterisk (\*) may be coded as the destination on the filter statement in conjunction with filter types *nosend* and *noreceive*. The asterisk matches all destinations and serves as a blackhole filter. A blackhole *nosend* filter, for example, can be used with a *send* filter to cause only a specific destination to be advertised over a given interface.
4. The filter mask should not be confused with the subnet mask. The operation of the filter mask is: does filter destination EQUAL RIP packet destination ANDed with Filter Mask?
5. The `Send_Only` statement is used to define global `Send_Only` settings. `Send_Only` parameters coded on individual `RIP_Interface` statements override the global `Send_Only` settings. Only one `Send_Only` parameter is allowed per `RIP_Interface` statement.

The syntax for coding `Send_Only` statements is as follows:

```
Send_Only=(<option>,<option>,...) | ALL
```

The valid <options> are:

VIRTUAL: Broadcast virtual IP addresses only

DEFAULT: Broadcast the default route only

DIRECT: Broadcast direct routes only

TRIGGERED: Broadcast triggered updates only

If ALL (the default) is coded, no `Send_Only` restrictions will be enforced. This is equivalent to not coding the `Send_Only` statement.

**Note:** When VIRTUAL, DEFAULT, or DIRECT are coded together, they have no effect on each other. However, if TRIGGERED is coded with one of the others, it applies to the other settings. So if `Send_Only(DIRECT, TRIGGERED)` is coded, direct routes will only be broadcast once they are triggered by a state change.

6. New in CS for z/OS V1R2 is the `IGNORE_RIP_NEIGHBOR` statement. This statement allows you to ignore routing table updates from a particular gateway. Specify one `IGNORE_RIP_NEIGHBOR` statement for each gateway whose routing table updates are to be ignored.

Below is an example of how to code RIP interfaces using filters:

```
RIP_Interface Name=TR1
              IP-address=9.24.104.3
              Subnet_mask=255.255.255.0
              filter=(nosend,9.24.100.0,255.255.255.0);

RIP_Interface Name=TR1
              IP-address=9.24.104.83
              Subnet_mask=255.255.255.0
              filter=(noreceive,9.25.0.0,255.0.0.0.0);
```

## 6.5 OMPROUTE commands

OMPROUTE can be started as an MVS started task, from the z/OS shell or from AUTOLOG. Once running, OMPROUTE is controlled from the MVS operator's console using MVS system commands:

1. Using **MODIFY (F)** commands.
  - Stop OMPROUTE by issuing `P <procname>` or `F <procname>,KILL`.
  - To reread the configuration file, issue `F <procname>,RECONFIG`. This command will reread the configuration file defined during startup. This command will only read new OSPF\_Interface, RIP\_Interface and Interface statements from the configuration file. It will ignore all duplicated statements.
  - The OMPROUTE **MODIFY** command can be used to disable and enable the subagent. We issue `F<procname>,ROUTESA=DISABLE|ENABLE`.
2. There are two ways to display OMPROUTE information, the **MODIFY** command and the **DISPLAY TCPIP,tcpproc,OMPROUTE** command.

For more information on these commands, refer to *z/OS V1R2.0 CS: IP System Administrator's Commands*, .

### Display the OMPROUTE routing table

To display the OMPROUTE routing table, enter one of the following commands:

- ▶ `F procname,RTTABLE`
- ▶ `D TCPIP,tcpproc,OMPROUTE,RTTABLE`

F T39BOMPR,RTTABLE						
EZZ7847I ROUTING TABLE 985						
	TYPE	DEST NET 6	MASK 7	COST 8	AGE 9	NEXT HOP(S) 10
1	SPE2	0.0.0.0		0 1	2200	9.24.104.3
	SPF	2 9.3.1.0	FFFFFFF0	1808	4820	9.24.104.3
1	SPE2	9.12.13.0	FFFFFFF0	0	2200	9.24.104.3
	SPF	2 192.168.40.0	FFFFFFF0	8	77328	9.24.104.3
1	SPE2	192.168.41.0	FFFFFFF0	1	2200	9.24.104.3
	DIR	3 192.168.233.0	FFFFFFF0	1	77313	192.168.233.38
2	SPF	192.168.233.38	FFFFFFF0	0	77338	EZASAMEMVS
	STAT	4 192.168.233.39	FFFFFFF0	0	77343	192.168.233.38
5	SBNT	192.168.235.0	FFFFFFF0	1	2199	NONE

Figure 6-18 OMPROUTE routing table

Figure 6-18 shows the routing table display output from OMPROUTE. The column type indicates different types with meanings indicating how the route was derived.

- 1 The SPE2 display indicates that this was derived from OSPF external route type 2.
- 2 The SPF shows that this is an OSPF intra-area route.
- 3 DIR is the result of a directly connected network or subnet.
- 4 STAT shows that the route is a static configured route.
- 5 SBNT shows a subnetted network.
- 6 Dest Net column means the IP destination address.
- 7 This is the IP destination subnet mask.
- 8 This is the cost of the route.
- 9 Age is the time elapsed since the last refresh of the routing table.
- 10 Next hop indicates the IP address of the next router on the path towards the final destination of the IP packet.

## 6.5.1 OMPROUTE OSPF commands

### Display OSPF configuration

- ▶ To display the OSPF configuration information, issue either of the following:  
D TCP/IP,T39BTCP,OMPROUTE,OSPF,LIST,ALL  
F T39BOMPR,OSPF,LIST,ALL

Both commands display the same output, where T39BOMPR is the OMPROUTE job name and T39BTCP is the TCP/IP job name. The sample output that shows what has been defined is displayed in Figure 6-19.

**F T39BOMPR,OSPF,LIST,ALL**

```
EZZ7831I GLOBAL CONFIGURATION 163
TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
STACK AFFINITY:          T39BTCP
OSPF PROTOCOL:          ENABLED
EXTERNAL COMPARISON:    TYPE 2
AS BOUNDARY CAPABILITY: ENABLED
IMPORT EXTERNAL ROUTES: RIP STA SUB
ORIG. DEFAULT ROUTE:    NO
DEFAULT ROUTE COST:     (1, TYPE 2)
DEFAULT FORWARD. ADDR.: 0.0.0.0
DEMAND CIRCUITS:        ENABLED
```

9  
8  
7  
6  
5  
4  
3  
2  
1

EZZ7832I AREA CONFIGURATION

AREA ID	AUTYPE	STUB?	DEFAULT-COST	IMPORT-SUMMARIES?
0.0.0.0	0=NONE	NO	N/A	N/A

EZZ7833I INTERFACE CONFIGURATION

IP ADDRESS	AREA	COST	RTRNS	TRNSDLY	PRI	HELLO	DEAD
192.168.234.40	0.0.0.0 <b>10</b>	5	1	1	10	40	
9.24.104.38	0.0.0.0 <b>10</b>	5	1	1	10	40	
192.168.233.38	0.0.0.0 <b>11</b>	5	1	1	10	40	
192.168.233.38	0.0.0.0	1	5	1	1	10	40
192.168.233.38	0.0.0.0	1	5	1	1	10	40

Figure 6-19 Display command to list all

- 1** The STACK AFFINITY displays the stack on which OMPROUTE is running.
- 2** Displays OSPF enabled.
- 3** OSPF supports two types of external metrics. Type 1 external metrics are equivalent to the link state metric. Type 2 external metrics are greater than the cost of any path internal to the AS. Type 2 external metrics assumes that routing between AS is the major cost of routing a packet and eliminates the need for conversion of external costs to internal link state metrics.
- 4** Indicates whether or not the router will import external routes into the OSPF domain.
- 5** Indicates the type of external routes that will be imported into the OSPF domain. This is displayed only when AS Boundary Capability is enabled.
- 6** Indicates whether the router will originate a default route into the OSPF domain.
- 7** Displays the default route cost.
- 8** Displays the forwarding address. The forwarding address is displayed only when AS Boundary is enabled. In our example the default forwarding address is the backbone address.
- 9** Indicates the demand circuit support availability.
- 10** Displays the interface address. OSPF dynamically obtains this address from the HOME statement in the TCP/IP profile data set.
- 11** This is the DYNAMICXCF IP address that OSPF obtains dynamically from the TCP/IP profile data set.



## Display neighbors

Figure 6-20 displays the OSPF neighbors from OMPROUTE T39BOMPR.

```

F T39BOMPR,OSPF,NBR
EZZ7851I NEIGHBOR SUMMARY 621
NEIGHBOR ADDR  NEIGHBOR ID  STATE  LSRXL  DBSUM  LSREQ  HSUP  IFC
9.24.104.113   9.24.104.113   128    0      0      0     OFF TR1
9.24.104.149   9.24.104.149   128    0      0      0     OFF TR1
9.24.104.42    9.24.104.42    4      0      0      0     OFF TR1
9.24.104.43    9.24.104.43    4      0      0      0     OFF TR1
9.24.104.3     192.168.41.12  128    0      0      0     OFF TR1
192.168.233.3  9.24.104.113   128    0      0      0     OFF EZAXCF03
192.168.233.39 0.0.0.0        1      0      0      0     OFF EZASAMEMV
192.168.233.28 9.24.104.42    128    0      0      0     OFF EZAXCF28
192.168.233.29 9.24.104.43    128    0      0      0     OFF EZAXCF28

```

Figure 6-20 OSPF neighbor (NBR) statistics and parameters

Each printed line gives a summary of each neighbor. This display is a snapshot of the current OSPF neighbors and their status at that time.

**Neighbor ADDR** This column displays all the neighbors identified by their IP addresses.

**Neighbor ID** Displays the OSPF neighbor router ID.

**State** The neighbor routers are discovered by the OSPF's Hello Protocol. The discovery is a process from being a non-neighbor to being a full neighbor. The functional process together with different states are described hereafter:

- 1 (Down)** Initially, the routers are down and have no contact with each other. This is the down state indicated by 1. No recent exchange of information has taken place with neighbor routers with state 1. The exceptions are the NBMA networks, where the Hello packets are still being sent even when the routers are marked down.
- 2 (Attempt)** The state is only valid for neighbor routers in an NMBA network. No contact has been made but Hello packets will continue to be sent at intervals specified by the HelloInterval.
- 4 (Initialize)** The Hello packet has been identified/received but no exchange of information has taken place between neighboring routers. There is, however, a contact made by the neighboring routers.
- 8 (2-way)** The Hello packets have been received and acknowledged by both neighboring routers. The designated router (DR) is selected and thereafter follows the selection of the Backup DR.
- 16 (ExStart)** Neighbor routers form adjacencies between themselves. Neighbor routers' communication is more advanced in this state. The transmission of link state database starts.
- 32 (Exchange)** The neighbors send their link state database to their adjacent routers. The link state database describes their characteristics. The link state request packets requesting the neighbors' recent LSA status may be sent to the neighboring routers. In this state, the neighbors are capable of receiving and sending all types of OSPF routing protocol packets.

- 64 (Loading)**      The link state request packets are being transmitted and received from neighboring routers requesting the most recent LSAs.
- 128 (Full)**      The neighboring routers displayed with this state mean they are fully adjacent. The adjacent routers exchange LSAs and appear in their neighbors' router-LSAs.

**Note:** The routers in an OSPF environment can be in any of the above listed states at any given time. Not all routers contacted by the hello packets of the OSPF's Hello protocol will undergo the full cycle until they are adjacent neighbors.

- LSRXL**      Displays the size of the current link state retransmission list for each neighbor.
- DBSUM**      Displays the size of the database summary list waiting to be sent to the neighbor.
- LSREQ**      Displays the number of more recent advertisements that are being requested from the neighbor. If no acknowledgment has been received from the receiving neighbor router during the LSA transmission, the LSA will be retransmitted until an acknowledgment is received.
- HSUP**      Displays whether Hello Suppression is active for that neighbor. In this example, all the suppressions are off and the OSPF Hello packets will be used.

### Display OSPF areas

```
F T39B0MPR,OSPF,LIST,AREAS
EZZ7832I AREA CONFIGURATION 625
AREA ID          AUTYPE      STUB?  DEFAULT-COST  IMPORT-SUMMARIES?
0.0.0.0          0=NONE    NO      N/A            N/A
```

Figure 6-21 Display of configured areas

Figure 6-21 shows the display of the configured areas. It confirms that there is only one area in our configuration: the backbone area.

**Note:** The backbone area is the minimum default for the OMPROUTE configuration file.

### Display area statistics

Figure 6-22 displays all the statistics for the areas in an AS. Our example had only the backbone area.

```
F T39B0MPR,OSPF,AREASUM
EZZ7848I AREA SUMMARY 747
AREA ID          AUTHENTICATION  #IFCS  #NETS  $RTRS  #BRDRS  DEMAND
0.0.0.0          NONE              5       3      10     0       ON
```

Figure 6-22 Display of area statistics

The display shows:

- Authentication** Indicates the type of authentication in use, if any. Two types of authentication are available, password and MD5 key.
- #IFCS** Indicates the total number of attached router interfaces to an area. However, the router interfaces might not be operational.
- #NETS** Indicates the number of transit networks that have been discovered while calculating the short path tree for this area.
- #RTRS** The number of routers discovered by the calculation of the short path tree for this area.
- #BRDRS** Shows the number of area border routers that have been found during the calculation of the short path tree for this area.
- Demand** Indicates whether or not demand circuit is supported.

### Display external advertisements

Figure 6-23 is a summary of the display of all the OSPF external advertisements.

```
F T39BOMPR,OSPF,EXTERNAL
EZZ7853I AREA LINK STATE DATABASE 749
TYPE LS DESTINATION      LS ORIGINATOR      SEQNO      AGE      XSUM
 5 @0.0.0.0              9.24.104.42       0X80000076 1560    0X3652
 5 @9.32.41.40           192.168.10.1      0X80001828 223     0XBCC9
 5 @192.168.31.0         192.168.10.1      0X800001C1 270     0X26CC
 5 @192.168.233.0        9.24.104.42       0X80000077 1560    0X75BE
 5 @192.168.233.3        9.24.104.149      0X80000073 1296    0XD0F9
 5 @192.168.233.4        192.168.234.40    0X80000073 1616    0XCBA0
 5 @192.168.233.28       9.24.104.149      0X80000073 1296    0XD5DB
 5 @192.168.233.29       9.24.104.113      0X80000078 176     0X9A35
```

Figure 6-23 OSPF external advertisements

The column Type will always be 5 for external advertisements. The rest of the display shows:

- LS Destination** This display indicates an IP address from another AS network.
- LS Originator** Indicates the router that initiated the advertisement.
- SEQNO AGE XSUM** These fields are used to check the validity of the LSA. If the LS age (AGE) has reached its highest value (MaxAge, 3600 seconds), it will not be used for route calculations.

### Display interfaces

The OSPF display output is from the configuration file as configured in Figure 6-5. The result is shown in Figure 6-24.

```
F T39BOMPR,OSPF,INTERFACE
EZZ7849I INTERFACES 905
IFC ADDRESS      PHYS      ASSOC. AREA  TYPE      STATE  #NBRS  #ADJS
192.168.234.40   VIPA39B   0.0.0.0      MULTI     128    0       0 1
9.24.104.38      TR1       0.0.0.0      BRDCST    64     3       3 2
192.168.233.38   EZAXCF03  0.0.0.0      P-2-MP    1      0       0
192.168.233.38   EZASAMEMVS 0.0.0.0      P-2-MP    16     1
192.168.233.38   EZAXCF28  0.0.0.0      P-2-MP    16     0       0 3
```

Figure 6-24 OSPF interface statistics and parameters

This command displays the interface addresses (IFC), physical names, associated areas, network types, current router state, number of neighbors (#NBRS) and number of adjacent routers (#ADJS). The #ADJS is the number of neighbors that the router has synchronized or is in the process of synchronizing.

**1** IFC indicates the address of the VIPA name VIPA39B, which is shown as a nonbroadcast multiaccess interface.

**2** Token-ring, broadcast network and Backup DR router.

**3** Point-to-multipoint and currently in a point-to-point state.

The router's status depends on the network and what the functions are like at that moment. The router's status is indicated by the following: 1 (down), 2 (backup), 4 (looped back), 8 (waiting), 16 (point-to-point), 32 (DR other), 64 (backup DR) or 128 (designated router).

Figure 6-24 displays the statistics related to the OSPF interfaces. Each line summarizes one interface. The state of the interfaces should not be confused with the state in which the routers are at a certain stage after being contacted by the OSPF's Hello protocol. The state display here indicates whether the router is DR or Backup, point-to-point connected, and so forth. Figure 6-20 indicates the progress status during the identification of the adjacent neighbor routers, while Figure 6-24 indicates the interface routers.

Figure 6-25 and Figure 6-26 show interface displays for TR1 and VIPA39B.

```

F T39BOMPR,OSPF,INTERFACE,NAME=TR1                                00001000
EZZ7850I INTERFACE DETAILS 909                                    00001100
    INTERFACE ADDRESS:      9.24.104.38                          1 00001200
    ATTACHED AREA:         0.0.0.0                               00001300
    PHYSICAL INTERFACE:    TR1                                   2 00001400
    INTERFACE MASK:       255.255.255.0                         00001500
    INTERFACE TYPE:       BRDCST                                3 00001600
    STATE:                64                                   4 00001700
    DESIGNATED ROUTER:    9.24.104.3                            5 00001800
    BACKUP DR:           9.24.104.38                            6 00001900
    7 00002000
DR PRIORITY:             1 HELLO INTERVAL: 10 RXMT INTERVAL:    5 00002100
DEAD INTERVAL:          40 TX DELAY:       1 POLL INTERVAL:    0 00002200
DEMAND CIRCUIT: OFF HELLO SUPPRESS: OFF SUPPRESS REQ: OFF 00002300
MAX PKT SIZE: 32764 TOS 0 COST: 1 00002400
8 00002500 9 10
# NEIGHBORS:            3 # ADJACENCIES:    3 # FULL ADJS.:    3 00002600
# MCAST FLOODS:        624 # MCAST ACKS:   1545 DL UNICAST:    OFF 00002700
MC FORWARDING: OFF 00002800
00002900
NETWORK CAPABILITIES:11 00003000
BROADCAST 00003100
DEMAND-CIRCUITS 00003200

```

Figure 6-25 Display of interface TR1

In Figure 6-25, we notice that the **Interface display** command for the token-ring address 9.24.104.38 provides the following statistics:

**1** Interface address.

**2** Physical interface name. This name is coded in the HOME statement of the TCP/IP stack that is connected to OMPROUTE.

**3** This interface is a broadcast type with a status state of 64 indicated by **4**.

The 64 means that it is a backup designated router with its IP address indicated by **6**. The designated router's IP address is 9.24.104.3 as indicated by **5**.

**7** This displays the current values of:

- ▶ DR Priority
- ▶ Hello Interval
- ▶ RXMT (Retransmission) Interval
- ▶ Dead Interval
- ▶ TX (Transmission) Delay
- ▶ Poll Interval
- ▶ Demand Interval
- ▶ Hello Suppression
- ▶ Suppression Request
- ▶ OSPF Maximum Packet Size

**8** indicates one of the following:

- ▶ #NEIGHBORS - this is the total number of routers whose Hellos packets have been received plus those that have been configured.
- ▶ #MCAT FLOODS - the number of link state updates that have been flooded out of this interface, excluding the retransmissions.

**9** indicates one of the following:

- ▶ Adjacencies - the number of neighbors in a state of exchange.
- ▶ MCAST ACKS - the number of acknowledgments out of the interface, excluding the retransmissions.

**10** FULL ADJS - this is the total number of neighbors who have been involved in database synchronization.

**11** This displays the network capabilities. This TR1 interface has a broadcast capability.

```

F T39BOMPR,OSPF,INTERFACE,NAME=TR1                                00001000
EZZ7850I INTERFACE DETAILS 909                                    00001100
      INTERFACE ADDRESS:      9.24.104.38      1 00001200
      ATTACHED AREA:         0.0.0.0          00001300
      PHYSICAL INTERFACE:    TR1              2 00001400
      INTERFACE MASK:        255.255.255.0    00001500
      INTERFACE TYPE:        BRDCST          3 00001600
      STATE:                 64              4 00001700
      DESIGNATED ROUTER:    9.24.104.3      5 00001800
      BACKUP DR:            9.24.104.38      6 00001900
      7 00002000
DR PRIORITY:      1 HELLO INTERVAL:  10 RXMT INTERVAL:  5 00002100
DEAD INTERVAL:   40 TX DELAY:         1 POLL INTERVAL:  0 00002200
DEMAND CIRCUIT: OFF HELLO SUPPRESS: OFF SUPPRESS REQ:  OFF 00002300
MAX PKT SIZE:   32764 TOS 0 COST:     1 00002400
      8 9 10 00002500
# NEIGHBORS:     3 # ADJACENCIES:     3 # FULL ADJS.:   3 00002600
# MCAST FLOODS: 624 # MCAST ACKS:    1545 DL UNICAST:  OFF 00002700
MC FORWARDING:  OFF 00002800
00002900
NETWORK CAPABILITIES:11 00003000
BROADCAST 00003100
DEMAND-CIRCUITS 00003200

```

Figure 6-26 Display of interface VIPA39B

Comparing Figure 6-25 with Figure 6-26, we notice the following differences:

- ▶ The designated interface router is the interface address with a current network state of 128, as indicated by 1.
- ▶ The backup designated router is the backbone area router, as indicated by 2.
- ▶ Currently, there are no discovered neighbors, as indicated by 3; no flooding has taken place (4) and no acknowledgments (5) have taken place either.

### Display routers

```

F T39BOMPR,OSPF,ROUTERS
EZZ7855I OSPF ROUTERS 130
 1 2 3 4 5 6
DTYPE RTYPE DESTINATION AREA COST NEXT HOP(S)
ASBR SPF 9.24.104.113 0.0.0.0 1 9.24.104.113
ASBR SPF 192.168.10.1 0.0.0.0 2 9.24.104.3
ASBR SPF 192.168.251.4 0.0.0.0 1 9.24.104.33
FADD SPF 9.12.14.9 0.0.0.0 8 9.24.104.3
FADD SPF 9.12.14.8 0.0.0.0 8 9.24.104.3

```

Figure 6-27 OSPF routes to other routers

Figure 6-27 shows the display of all the routes to OSPF configured routers on the network.

1 The DTYPE indicates the destination type of the IP router. The ASBR means that it is an Autonomous System boundary router and FADD indicates the forwarding address router. This is used for external routes.

**2** RTYPE indicates how the route type was derived. The SPF (shortest path first) is an intra-area route from OSPF.

**3** The destination column indicates the destination router's OSPF identification.

**4** Indicates the OSPF area to which the router belongs. In our display, all the routers belong to the backbone area.

**5** Cost indicates the cost calculated to reach the router.

**6** The next hop is the IP address of the next router towards the destination.

## Display OSPF statistics

The display command in Figure 6-28 shows the statistics of the OSPF routing protocol.

```
F T39BOMPR,OSPF,STATISTICS
EZZ7856I OSPF STATISTICS 222
      OSPF ROUTER ID:      192.168.234.40
      EXTERNAL COMPARISON: TYPE 2
      AS BOUNDARY CAPABILITY: YES
      IMPORT EXTERNAL ROUTES: RIP STA SUB
      ORIG. DEFAULT ROUTE: NO
      DEFAULT ROUTE COST:  (1, TYPE 2)
      DEFAULT FORWARD. ADDR.: 0.0.0.0

ATTACHED AREAS:          1  OSPF PACKETS RCVD: 2          1231
OSPF PACKETS RCVD W/ERRS:1 1090 TRANSIT NODES ALLOCATED: 3    56
TRANSIT NODES FREED: 4    45  LS ADV. ALLOCATED: 5        52
LS ADV. FREED: 6         37  QUEUE HEADERS ALLOC:          32
QUEUE HEADERS AVAIL:    32  MAXIMUM LSA SIZE:          2048
# DIJKSTRA RUNS:        2  INCREMENTAL SUMM. UPDATES:    0
INCREMENTAL VL UPDATES: 0  MULTICAST PKTS SENT: 7 310
UNICAST PKTS SENT: 8     37  LS ADV. AGED OUT:            0
LS ADV. FLUSHED:        0  PTRS TO INVALID LS ADV:      0
INCREMENTAL EXT. UPDATES: 10
```

Figure 6-28 OSPF routing protocol statistics

The display will help you track how your OSPF configuration is performing. You can compare the output against your network's objective of OSPF implementation. From the above output, you can adjust the parameters until you reach your goal.

**1** Indicates the OSPF packets received with errors. From this number, you can see if there are more OSPF packets received with errors or not. Compare it with **2**, the number of OSPF packets received, and you can see if more packets have errors or not.

**3** Transit allocated against transit nodes freed (**4**).

**5** LS advertisements allocated compared to **6** LS advertisements freed.

**7** How many packets have been sent through multicast.

**8** How many packets have been sent through unicast.

The OSPF Routing Protocol Statistics command is a useful tool for performance evaluation and monitoring.

## 6.5.2 OMPROUTE RIP commands

### List RIP Interfaces

Figure 6-29 shows the display of RIP interfaces.

```
F T03AROU,RIP,LIST,INTERFACES
EZZ7843I RIP CONFIGURATION 923
TRACE: 0, DEBUG: 0, SADEBUG LEVEL: 0
STACK AFFINITY: T03ATCP
RIP: ENABLED 1
RIP DEFAULT ORIGINATION: DISABLED
PER-INTERFACE ADDRESS FLAGS:
MPC25          192.168.235.3  RIP-2 MULTICAST. 2
                SEND NET AND SUBNET ROUTES 3
                RECEIVE NO DYNAMIC HOST ROUTES 4
                RIP INTERFACE INPUT METRIC: 1 5
                RIP INTERFACE OUTPUT METRIC: 0
```

Figure 6-29 RIP configured interface display

- 1 RIP: ENABLED indicates that the RIP protocol has been enabled.
- 2 The RIP Protocol communication is RIP Version 2 with multicast support over this interface.
- 3 This indicates which types of send will be communicated over this interface.
- 4 Specifies the type of receive that will be accepted. In this display, no dynamic host routes will be received.
- 5 The RIP interface metric is 1. This is the value to be added to the metric for RIP routes received over this interface.

### Display a specific RIP interface

Figure 6-30 shows the display of a specific RIP interface.



```

F T03COMPR,RIP,IF,NAME=TR1
EZZ7860I RIP INTERFACE DETAILS
INTERFACE ADDRESS:    9.24.104.33
INTERFACE NAME:      TR1
SUBNET MASK:         255.255.255.0
MTU                  4000
DESTINATION ADDRESS: N/A

RIP VERSION:         2      SEND POIS. REV. ROUTES: YES
IN METRIC:           1      OUT METRIC:           0
RECEIVE NET ROUTES: YES    RECEIVE SUBNET ROUTES: YES
RECEIVE HOST ROUTES: NO    SEND DEFAULT ROUTES:  NO
SEND NET ROUTES:     YES    SEND SUBNET ROUTES:   YES
SEND STATIC ROUTES:  NO     SEND HOST ROUTES:    NO

SEND ONLY: ALL

FILTERS: NOSEND      9.24.106.1      255.255.255.0

```

Figure 6-30 Specific RIP interface display

### Display RIP filters

Figure 6-31 shows the display of RIP filters.

```

F T03COMPR,RIP,FILTERS
EZZ8016I GLOBAL RIP FILTERS
SEND ONLY: VIRTUAL

FILTERS: NOSEND      9.24.106.1      255.255.255.0

```

Figure 6-31 RIP filter display

## 6.6 Interface cost considerations

All interfaces have a cost value associated with them in both the OSPF and RIP routing protocols. The value of a route to reach a destination is calculated by the sum of the costs of each link that will be traversed on the way to the destination.

The method for defining cost values for each interface differs between the OSPF and RIP protocols. In the OSPF\_Interface statement, we use the Cost0 parameter, which can range from 0 to 65535. In the RIP\_Interface statement, we use the IN\_Metric parameter to define a value from 1 to 15 to be added to every route received over the interface, and the Out\_Metric parameter to define a value from 0 to 15 to be added to every route advertised over the interface. Please see *z/OS V1R2.0 CS: IP Configuration Guide, SC31-8775* and *z/OS V1R2.0 CS: IP Configuration Reference, SC31-8776* for more detailed information.

As you can see, it is quite complicated to set route precedence for both protocols. If you use only one of the routing protocols, route precedence is not a concern, but if you use multiple protocols or AS Boundary Routers in your network, you should be careful with this topic.

The COMPARISON statement is set to let OMPROUTE know if the cost value received from another AS or from another routing protocol can be comparable and useful. You can configure the COMPARISON value as Type1 or Type2.

When COMPARISON=Type1 is set, the cost value received from other routing protocols or exchanged between AS Boundary Routers is to be used because they are comparable. Note that if you use the OSPF and RIP protocols and set COMPARISON=Type1, your cost values in the OSPF interfaces must be low, because the OSPF AS must be reached from the RIP AS with a cost value less than 16.

A COMPARISON value configured as Type2 indicates that cost values are non-comparable. In this case, there are some rules to set cost values in a RIP AS and/or OSPF AS.

Table 6-1 shows the order of precedence used by routers that must learn routing information in a network with multiple routing protocols or when there is an OSPF Boundary Router.

Table 6-1 Route precedence

Source Comparison	Route A - Type	Route B -Type	Route chosen
Type 1	OSPF Internal	RIP	OSPF Internal
Type 1	OSPF Internal	OSPF Type 1 External	OSPF Internal
Type 1	OSPF Internal	OSPF Type 2 External	OSPF Internal
Type 1	RIP	OSPF Type 1 External	Lowest Cost Route
Type 1	RIP	OSPF Type 2 External	RIP Route
Type 1	OSPF Type 1 External	OSPF Type 2 External	OSPF Type 1 External
Type 2	OSPF Internal	RIP	OSPF Internal
Type 2	OSPF Internal	OSPF Type 1 External	OSPF Internal
Type 2	OSPF Internal	OSPF Type 2 External	OSPF Internal
Type 2	RIP	OSPF Type 1 External	OSPF Type 1 External
Type 2	RIP	OSPF Type 2 External	Lowest Cost Route
Type 2	OSPF Type1 External	OSPF Type 2 External	OSPF Type 1 External

Source Comparison is the value to which COMPARISON is set. Route A and Route B are two possible routes from one source to the same destinations that can be chosen.

The following terms are used in the table:

- ▶ *RIP Route* - a route learned from the RIP protocol.
- ▶ *OSPF Internal Route* - a route learned from the OSPF protocol where the entire path lies within the same OSPF AS.
- ▶ *OSPF External Route* - a route learned from the OSPF protocol whose path traverses another AS. There are two categories of OSPF external routes: *OSPF external route type 1*, when COMPARISON Type 1 is set and OSPF External routes are imported from another AS, and *OSPF External routes type 2*, when COMPARISON Type 2 is set and OSPF External routes are imported from another AS.

## 6.7 Multipath considerations

The OMPROUTE daemon enables multipath equal-cost routes for each source and destination pair. To enable the multipath function, code the MULTIPATH parameter in the IPCONFIG statement in the TCP/IP profile. Using the multipath function enables outbound connections to be spread over existing routes. Up to four equal-cost routes can be used for multipath in OSPF routes. For RIP, multiple equal-cost routes will be added only to directly connected destinations over redundant interfaces. The traffic spread over these routes depends on the parameter coded; it can be done on a connection basis or a packet basis.

By specifying the PERCONNECTION parameter, TCP/IP will select a route on a round-robin basis from a multipath routing list to the destination and will use this route for all IP packets that use this connection. This is independent of whether the data is connection- or connectionless-oriented. All other associations will be spread over the multiple equal-cost routes.

If the PERPACKET parameter is specified and there are IP packets to be delivered over a multipath route, a route will be selected on a round-robin basis to send this IP packet to its destination, and connection- or connectionless-oriented IP packets using the same source and destination pair will not always use the same route.

You should pay attention if you choose the PERPACKET option, because it may consume additional CPU cycles on the final destination to reassemble packets if they arrive out of order. Fragmented IP datagrams are restricted from using the PERPACKET option.

The subparameter FWMULTIPATH PERPACKET in the DATAGRAMFWD parameter is used for transferring data between networks if there are multipath equal-cost routes. It is coded in the IPCONFIG statement and indicates that the connection- or connectionless-oriented IP packets using the same destination address do not always use the same route.

### 6.7.1 Multipath example

Figure 6-32 shows sample MULTIPATH connections between three systems. All systems connect to each other in three different ways: XCF, token-ring, and Ethernet. All paths are defined with equal cost in the OMPROUTE configuration profile.

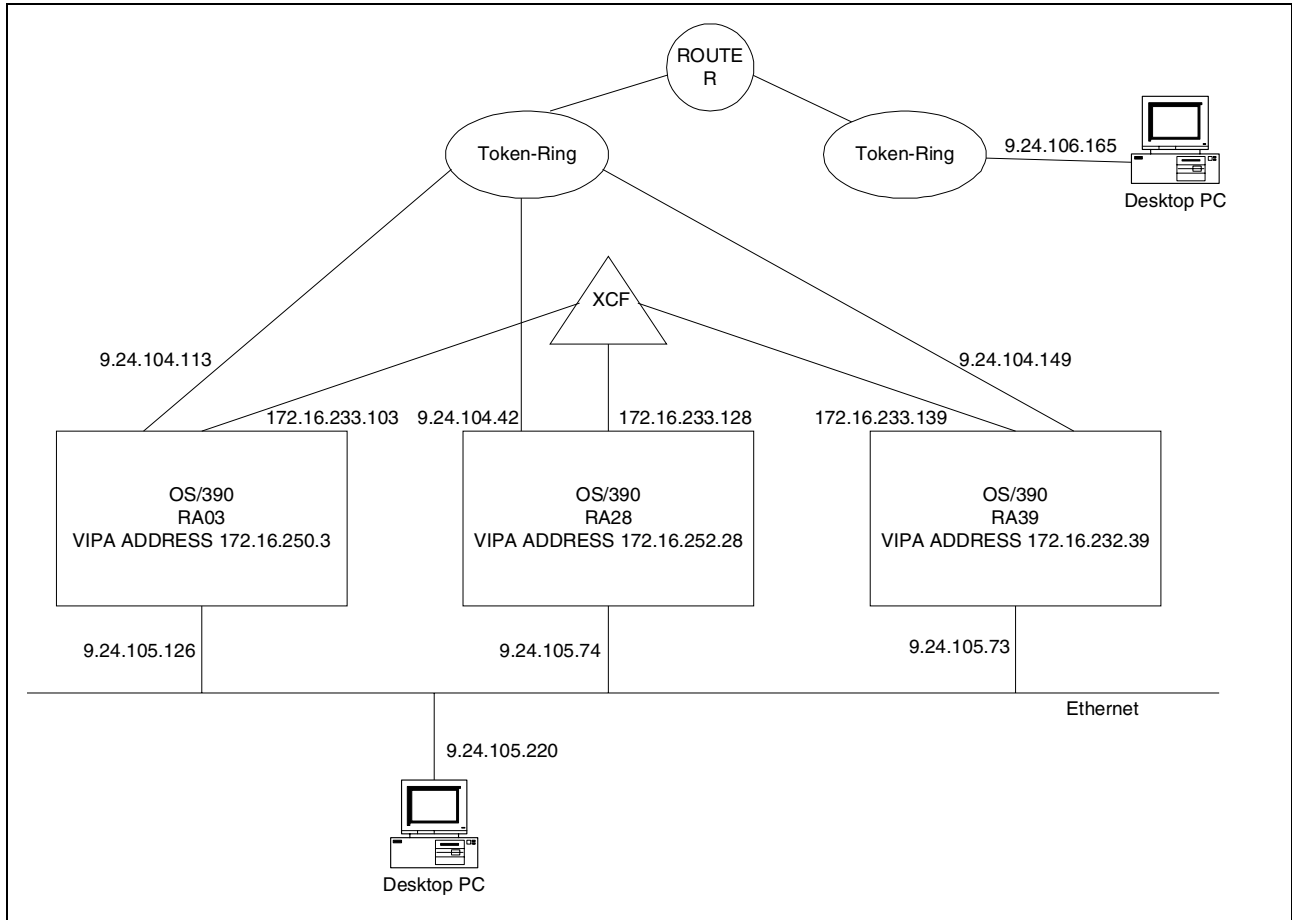


Figure 6-32 Sample multipath connected system configuration

Figure 6-33 shows the RA03 system's TCP/IP profile configuration. The MULTIPATH PERCONNECTION statement in IPCONFIG statement provides the function. The PERCONNECTION parameter is the default. You can use the PERPACKET parameter as an alternative. You can define the function in the initial profile or you can add this function with the **OBEY** command. When you enable this function, you will see the message MULTIPATH PERCONNECTION SUPPORT IS ENABLED in the TCP/IP started task log. If you use the PERPACKET function, you will see the MULTIPATH PERPACKET SUPPORT IS ENABLED message in the log.

```

IPCONFig
ARPTO 1200
NOSOURCEVIPA
VARSUBNETTING
SYSPLEXRouting
DYNAMICXCF 172.16.233.103 255.255.255.0 1
IGNORERedirect
REASSEMBLytimeout 15
STOPONclawerror
TTL 60
DATAGRAFWD FWMULTIPATH PERPACKET
MULTIPATH PERCONNECTION
; MULTIPATH PERPACKET
.
.
AUTOLOG 1
  OMPROUTA          ; OMPROUTE
ENDAUTOLOG
.
.
; *****
; VIPA Definition
; *****
  DEVICE VIPA3A  VIRTUAL  0
  LINK  VIPA3A  VIRTUAL  0  VIPA3A
; *****
; LCS Definition
; osa ch 78          Device # 2060-2061
; *****
  DEVICE TR1    LCS  2060 autorestart
  LINK  TR1    IBMTR  0 TR1

  DEVICE EN1  LCS          2026
  LINK  EN1  ETHEROR802.3 1  EN1

; HOME Internet (IP) addresses of each link in the host.
HOME
  172.16.250.3  VIPA3A  ; 1st VIPA Link
  9.24.104.113  TR1     ; 9.24.104.0

```

Figure 6-33 System RA03 TCPIP profile

Figure 6-34 shows the OMPROUTE configuration profile for system RA03. The Interfaces, VIPA address, and dynamic XCF address were added to this table's OSPF interfaces.

```

Area      Area_Number=0.0.0.0
          Stub_Area=NO
          Authentication_type=None;
RouterID=172.16.250.3;
ROUTESA_CONFIG ENABLED=YES
          COMMUNITY=(MVSsubagent);

OSPF_Interface IP_Address=172.16.233.*
              Subnet_mask=255.255.255.0
              Cost0=60 ;it was 8
              Non_Broadcast=Yes
              MTU=32768;

OSPF_Interface IP_Address=172.16.250.3
              Name=VIPA3A
              Subnet_mask=255.255.255.0
              Non_Broadcast=Yes
              Destination_Addr=172.16.250.3
              Cost0=1
              MTU=32768;

OSPF_Interface IP_Address=9.24.104.113
              Name=TR1
              Cost0=60
              Subnet_mask=255.255.255.0
              MTU=4082;

OSPF_Interface IP_Address=9.24.105.126
              Name=EN1
              Cost0=66
              Subnet_mask=255.255.255.0
              MTU=1492;

```

*Figure 6-34 System RA03 OMPROUTE configuration profile*

You can see all the routing information for RA03 system in Figure 6-35.

```

D TCPIP,TCPIPA,N,ROUTE
EZZ2500I NETSTAT CS V2R10 TCPIPA 301
DESTINATION GATEWAY FLAGS REFCNT INTERFACE
DEFAULT 9.24.104.3 UG 000000 TR1
9.0.0.0 9.24.104.3 UG 000000 TR1
9.1.150.0 9.24.104.3 UG 000000 TR1
9.3.1.0 9.24.104.3 UG 000000 TR1
9.3.240.0 9.24.104.3 UG 000000 TR1
9.12.0.0 9.24.104.3 UG 000000 TR1
9.12.2.0 9.24.104.3 UG 000000 TR1
9.12.3.0 9.24.104.3 UG 000000 TR1
9.12.3.16 9.24.104.3 UG 000000 TR1
9.12.3.32 9.24.104.3 UG 000000 TR1
9.12.3.48 9.24.104.3 UG 000000 TR1
9.12.6.0 9.24.104.3 UG 000000 TR1
9.12.9.0 9.24.104.3 UG 000000 TR1
9.12.13.0 9.24.104.3 UG 000000 TR1
9.12.14.0 9.24.104.3 UG 000000 TR1
9.12.15.0 9.24.104.3 UG 000000 TR1
9.24.104.0 0.0.0.0 U 000000 TR1
9.24.104.1 9.24.105.1 UGH 000000 EN1
9.24.104.1 0.0.0.0 UH 000000 TR1
9.24.104.18 0.0.0.0 UH 000000 TR1
9.24.104.113 0.0.0.0 UH 000000 TR1
9.24.105.0 0.0.0.0 U 000000 EN1
9.24.105.126 0.0.0.0 UH 000000 EN1
9.24.106.0 9.24.105.1 UG 000000 EN1
9.24.106.0 9.24.104.18 UG 000000 TR1
9.24.106.0 9.24.104.3 UG 000000 TR1
9.24.106.0 9.24.104.1 UG 000000 TR1
9.32.41.40 9.24.104.3 UG 000000 TR1
127.0.0.1 0.0.0.0 UH 000004 LOOPBACK
172.16.100.254 9.24.104.18 UGH 000000 TR1
172.16.101.254 9.24.104.18 UGH 000000 TR1
172.16.102.254 9.24.104.18 UGH 000000 TR1
172.16.232.0 172.16.233.139 UG 000000 EZAXCF39
172.16.232.0 9.24.104.149 UG 000000 TR1
172.16.232.0 9.24.105.73 UG 000000 EN1
172.16.232.39 9.24.104.149 UGH 000000 TR1
172.16.232.39 172.16.233.139 UGH 000000 EZAXCF39
172.16.232.39 9.24.105.73 UGH 000000 EN1
172.16.233.0 0.0.0.0 U 000000 EZAXCF28
172.16.233.0 0.0.0.0 U 000000 EZAXCF39
172.16.233.103 0.0.0.0 UH 000001 EZAXCF28
172.16.233.103 0.0.0.0 H 000000 EZASAMEMVS
172.16.233.103 0.0.0.0 UH 000000 EZAXCF39
172.16.233.128 0.0.0.0 UH 000000 EZAXCF28
172.16.233.139 0.0.0.0 UH 000000 EZAXCF39
172.16.240.0 9.24.105.73 UG 000000 EN1
172.16.240.0 172.16.233.139 UG 000000 EZAXCF39
172.16.240.0 9.24.104.149 UG 000000 TR1
172.16.240.3 0.0.0.0 UH 000000 VIPLAC10F003
172.16.240.28 172.16.233.128 UGH 000000 EZAXCF28
172.16.240.28 9.24.104.42 UGH 000000 TR1
172.16.240.39 172.16.233.139 UGH 000000 EZAXCF39
172.16.240.39 9.24.104.149 UGH 000000 TR1
172.16.240.39 9.24.105.73 UGH 000000 EN1
172.16.250.3 0.0.0.0 UH 000000 VIPA3A
172.16.252.0 9.24.104.42 UG 000000 TR1
172.16.252.0 172.16.233.128 UG 000000 EZAXCF28
172.16.252.28 9.24.104.42 UGH 000000 TR1
172.16.252.28 172.16.233.128 UGH 000000 EZAXCF28
192.166.236.1 0.0.0.0 H 000000 M3A2216A
192.168.10.1 9.24.104.3 UGH 000000 TR1
192.168.21.0 9.24.104.3 UG 000000 TR1
192.168.21.1 9.24.104.3 UGH 000000 TR1
192.168.21.2 9.24.104.3 UGH 000000 TR1
192.168.21.12 9.24.104.3 UGH 000000 TR1
192.168.31.0 9.24.104.3 UG 000000 TR1
192.168.31.1 9.24.104.3 UGH 000000 TR1
192.168.31.2 9.24.104.3 UGH 000000 TR1
192.168.31.12 9.24.104.3 UGH 000000 TR1
192.168.41.0 9.24.104.3 UG 000000 TR1
192.168.41.1 9.24.104.3 UGH 000000 TR1
192.168.41.12 9.24.104.3 UGH 000000 TR1

```

Figure 6-35 System RA03 system routing display response

Figure 6-36 shows the RA03 system TCPIPA stack HOME information, which is displayed as each interface's IP address.

```
D TCPIP,TCPIPA,N,HOME
EZZ2500I NETSTAT CS V2R10 TCPIPA 491
HOME ADDRESS LIST:
ADDRESS          LINK          FLG
172.16.250.3    VIPA3A       P
9.24.104.113    TR1
9.24.105.126    EN1
192.168.64.2    EZASAMEMVS
172.16.233.103  EZAXCF28
172.16.233.103  EZAXCF39
172.16.240.3    VIPLAC10F003
127.0.0.1       LOOPBACK
```

Figure 6-36 RA03 system TCPIPA stack NETSTAT HOME response

Figure 6-37 shows the OSPF routing information with line costs. If you check the IP addresses, TR1, EN1 and XCF connection have the same cost, which is 60.

```
D TCPIP,TCPIPA,OMPR,OSPF,LIST,ALL
EZZ7831I GLOBAL CONFIGURATION 139
TRACE: 0, DEBUG: 0, SADEBUB LEVEL: 0
STACK AFFINITY:          TCPIPA
OSPF PROTOCOL:          ENABLED
EXTERNAL COMPARISON:    TYPE 2
AS BOUNDARY CAPABILITY: DISABLED
DEMAND CIRCUITS:        ENABLED

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

EZZ7833I INTERFACE CONFIGURATION
IP ADDRESS      AREA      COST  RTRNS  TRNSDLY  PRI  HELLO  DEAD
172.16.233.103  0.0.0.0   60    5      1        1    10     40
172.16.240.3    0.0.0.0   1     5      1        1    10     40
172.16.233.103  0.0.0.0   60    5      1        1    10     40
172.16.233.103  0.0.0.0   60    5      1        1    10     40
9.24.105.126    0.0.0.0   60    5      1        1    10     40
9.24.104.113    0.0.0.0   60    5      1        1    10     40
172.16.250.3    0.0.0.0   1     5      1        1    10     40

EZZ7835I NBMA CONFIGURATION
INTERFACE ADDR  POLL INTERVAL
172.16.250.3   120
```

Figure 6-37 System RA03 OMPROUTE OSPF display command

Figure 6-38 shows equal cost paths between the RA03 and RA28 systems. If the RA03 system sends IP packets to the RA28 system, the packets go over three paths randomly.



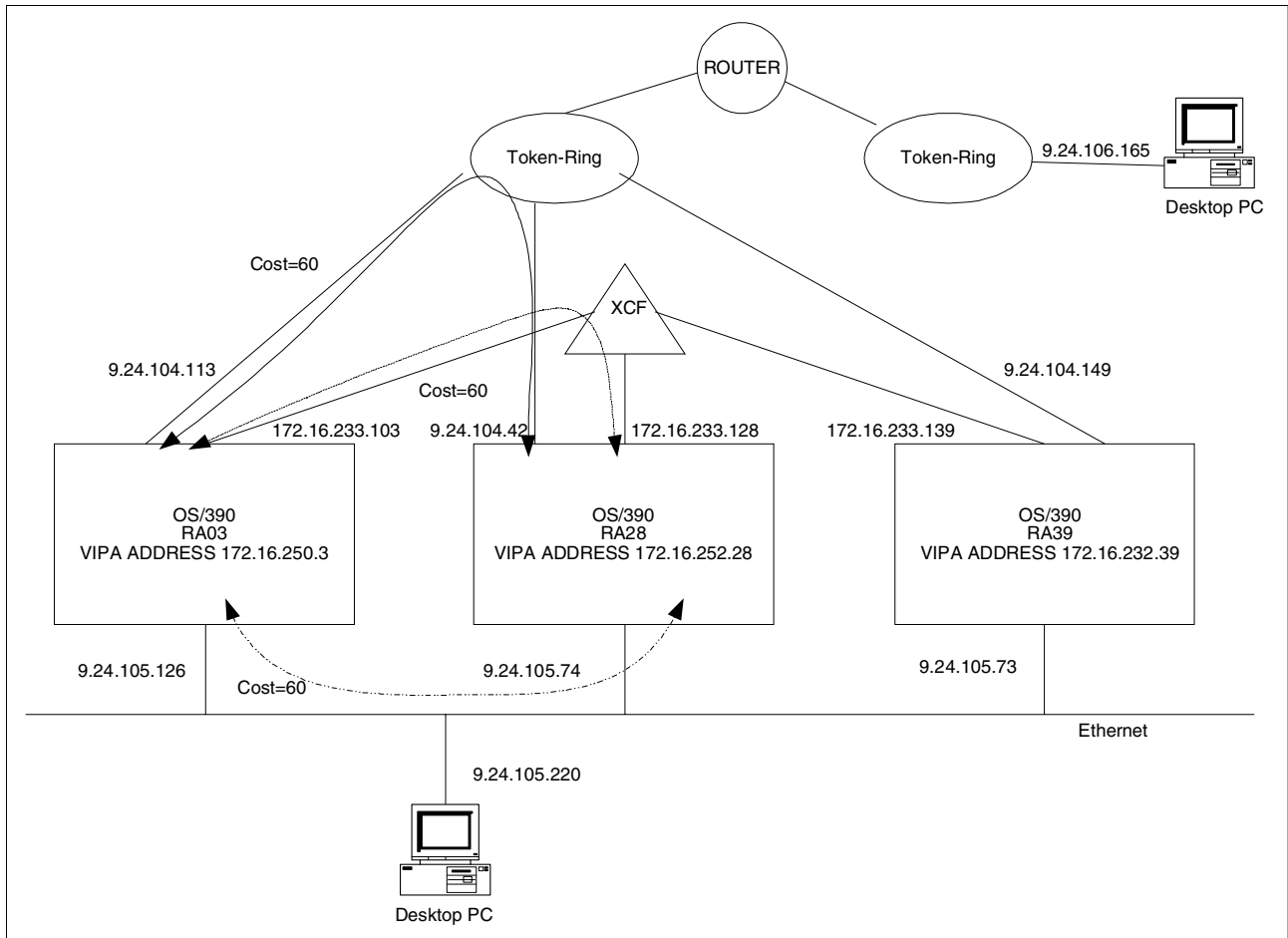


Figure 6-38 Equal cost multipath connection between RA03 and RA28 systems





## Routing with VIPA

This chapter provides an overview of routing with VIPA.

For more information on VIPA, please refer to the following manuals:

- ▶ *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775
- ▶ *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517

The concept of virtual IP address (VIPA) was introduced by IBM to remove the dependencies of other hosts on particular network attachments to TCP/IP running on an MVS image. Prior to VIPA, other hosts were bound to one of the home IP addresses and, therefore, to a particular network address. If the physical network interface failed, the home IP address became unreachable and all the connections already established with this IP address failed also. VIPA provides a *virtual* network interface with a *virtual* IP address that other TCP/IP hosts can use to select a CS for z/OS IP stack without choosing a specific network interface. If a specific physical network interface fails, the VIPA address remains reachable by other physical network interfaces. Hosts that connect to z/OS IP applications can send data to a VIPA address using whatever path is selected by the dynamic routing protocol (OSPF or RIP).

A static VIPA is configured in the same way as a normal IP address for a physical adapter, except that it is not associated with any particular interface. VIPA uses a virtual device and a virtual IP address. The virtual IP address is added to the home address list. The virtual device defined for the VIPA using `DEVICE`, `LINK` and `HOME` statements is always active and it never fails. The CS for z/OS IP stack advertises routes to the VIPA as if it were one hop away and it can be reached via the stack.

To an attached router, the CS for z/OS IP stack simply looks like another router. When the IP stack receives a packet destined for its VIPA, the inbound IP function of the stack notes that the IP address of the packet is in the stack's home list and forwards the packet up the stack. Assuming that the IP stack has more than one network interface, if a particular network interface fails, the downstream router will simply route VIPA-targeted packets to the stack via an alternate route. In other words, the destination IP stack on z/OS is still reachable and looks like another intermediate node. The VIPA may thus be thought of as an address to the stack, and not to any particular adapter.

While VIPA certainly removes the dependency on a particular network interface as a single point of failure, the connectivity to a server can still be lost if a single stack or an entire z/OS image fails. We can manually move a VIPA to another stack using the **OB**EY command (or semi-automatically using any system management automation of the manual process). Even if the VIPA is moved, however, we have to restart the IP sessions because they were lost due to the IP stack or z/OS image failure.

SecureWay Communications Server for OS/390 V2R8 and later releases provide improvements by introducing Dynamic VIPA (DVIPA). The Dynamic VIPA function builds on the VIPA concept and automates the movement of the VIPA to an appropriate surviving stack.

Automatic *VIPA takeover* allows a VIPA address to move automatically to a stack where an existing suitable application instance already resides, allowing that instance to serve clients formerly connected to the failed stack. Automatic *VIPA takeback* allows a VIPA address to move back automatically to the failed stack once it is restored. With IP V2R8, VIPA takeback was either disruptive or occurred only when all connections on the stack that originally took over the VIPA terminated. Since CS for OS/390 V2R10 IP, VIPA takeback can be immediate and nondisruptive. The VIPA can be taken back by its rightful owner immediately without disrupting existing connections to the current owner.

*Application-specific Dynamic VIPAs* allow VIPAs to be defined and activated by individual applications, so that the VIPA moves when the application is moved.

*Distributed DVIPAs* were introduced with CS for OS/390 V2R10 IP and are used for implementing the Sysplex Distributor function. This function allows connections to be distributed among TCP/IP stacks in a sysplex environment.

For the remainder of this chapter, we will refer to the sample topology shown in Figure 7-1. This topology includes two LPARs in a sysplex, each containing one stack. Additionally, each LPAR has two links to the 2216 router in our network, one MPC+ and one via a shared OSA adapter (LCS). In some scenarios in this chapter, however, both links are not used. Within the sysplex, XCF links connect the stacks.

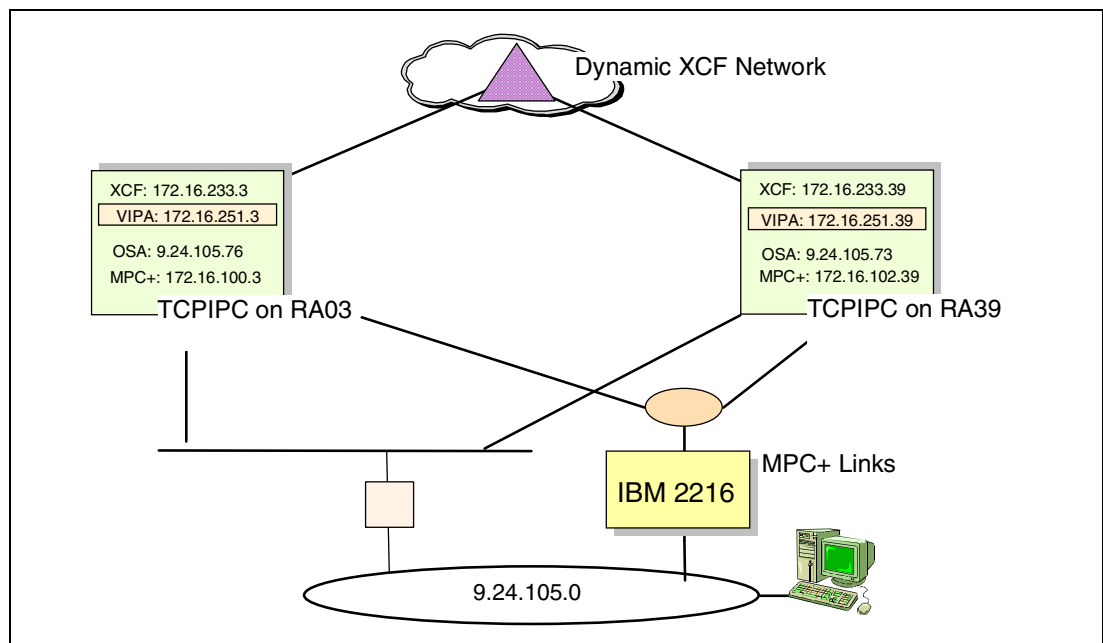


Figure 7-1 Sample network topology

## 7.1 Shared OSA pre-routing

Because multiple sysplex LPARs (and stacks) can simultaneously make use of an OSA adapter, the card itself must make certain pre-routing decisions regarding incoming packets. That is, when a packet arrives at the OSA adapter, the only indication as to which LPAR (and stack) it should be forwarded to is the IP information contained within it. In particular, the ultimate destination of the IP packet is used for this purpose. Each of the LPARs (stacks) owns a set of IP addresses and will receive all IP packets destined for them. Additionally, exactly one LPAR can be identified as the default LPAR, thereby receiving packets with IP destination addresses not owned by any LPAR.

When making use of VIPA in a shared OSA environment, the OSA must be made aware of the VIPA in the owning stack. The OSA simply treats the VIPA as another IP address owned by the particular LPAR (stack). If the VIPA is active on multiple LPARs (stacks) within the sysplex, however, the OSA might be confused as to which LPAR (stack) it should direct packets. This restriction is corrected with the expected introduction of Generic Routing Encapsulation (GRE) within CS for z/OS IP V1R2.

The method in which the OSA adapter is informed of a particular LPAR's (stack's) IP addresses is different depending on what type of adapter it is. With OSA-Express, this is done dynamically via communication between CS for z/OS IP and the OSA adapter. In OSA-2 (which we are using in our topology), this is done via manual configuration. The configuration is in the form of the OSA Address Table (OAT) and we can simply include the VIPA address in this configuration. We issued the **IOACMD get OAT** command to show you the RA03 entry in Figure 7-2. If the address is not in an OAT entry, the packet is forwarded to the default image (stack). If there is no default, the OSA adapter discards the packet.

```

*****
*** OSA/SF Get OAT output created 09:17:39 on 09/21/2000      ***
*** IOACMD APAR level - OW39984      ** sandra.sample(OATTABD8) ***
*****
***          Start of OSA address table for CHPID D8          ***
*****
* UA(Dev) Mode      Port      Entry specific information      Entry Valid
*****
                                LP 1 (A1      )
00(2060) passthru   00  PRI  009.024.104.113                SIU  ALL
                                172.016.250.003
01(2061) passthru   00  PRI  009.024.104.113                SIU  ALL
                                172.016.250.003
02(2062) passthru   00  no   009.024.104.033                S    ALL
                                172.016.251.004
03(2063) passthru   00  no   009.024.104.033                S    ALL
                                172.016.251.004
04(2064) passthru   01  no   009.024.105.076                S    ALL 1
                                172.016.251.003 2
05(2065) passthru   01  PRI  009.024.105.076                S    ALL
                                172.016.250.003
06(2066) N/A
07(2067) N/A
08(2068) N/A
09(2069) N/A
0A(206A) N/A
0B(206B) SNA      00
0C(206C) N/A
0D(206D) N/A
0E(206E) N/A

```

Figure 7-2 OSA address table

**1,2** We are using device 2064 for the RA03 host and have defined both IP addresses in the OAT: 9.24.105.76 (OSA card IP address) and 172.16.251.3 (VIPA address).

## 7.2 VIPA considerations

Because VIPAs are different from normal physical interface IP addresses, there are some special considerations that must be taken into account when using them. When a host owns a VIPA, it advertises reachability to it as if it were one hop away from it. Dynamic VIPAs can come and go; the dynamic routing protocol must therefore distribute this information to neighboring routers.

### 7.2.1 VIPA address assignment

It is strongly recommended that you define a subnet for VIPA separate from the physical interfaces. We defined the same subnet for all VIPA IP addresses in our sysplex environment. The routing daemon must therefore advertise the host and subnet routes of each VIPA to downstream routers. OMPROUTE and ORouteD have this capability.

In our network, shown in Figure 7-1, our 2216's route table will include a host route for each of the VIPAs (172.16.251.3 and 172.16.251.39) through the local token-ring and MPC+ interfaces. It will also have a subnet route for the 172.16.251.0 subnet through one of the interfaces. This subnet route is not a concern because host routes take precedence over subnet routes; hence the subnet route will not be used.

**Note:** You should make sure that your routers accept host routes. Otherwise, each VIPA address must be in a separate subnetwork.

## 7.2.2 Fault tolerance with VIPA

To improve fault tolerance beyond the use of VIPA, you can make use of redundant hardware. In our environment, we have an OSA card and an MPC-attached 2216 router connected to both hosts RA03 and RA39. The RA03 host is using the MPC+ link for outbound packets. If the interface goes down:

- ▶ The routing daemon in the RA03 host will switch the outbound traffic from the MPC+ interface to the OSA interface as soon as it detects the interface failure
- ▶ The RA03 adjacent routers will update their route table in light of the new information being provided by RA03

In our sysplex environment, we also have the XCF link that can be used as a backup interface in case of failure. We can use the RA39 host as a path from RA03 to network 9.24.105.0 after an MPC+ interface failure instead of using the OSA interface.

## 7.2.3 Beware of ICMP redirection

In general, we found that the alternate routing available when VIPA was implemented allowed us to maintain sessions whenever one path failed and alternates were available. However, there are a couple of issues which should be considered:

1. Some workstations can dynamically add routes even when dynamic routing is disabled on the workstation. With a Windows NT server on the same network as the OSA card and the 2216 router, we found a route created dynamically on the Windows NT server. We had set the 2216 router as the default router for the Windows NT server. The 2216 router was aware of the existence of the MP redirect message so that it would send data directly to a host owning the VIPA via the OSA card.

We set up a TN3270 session from Windows NT to the VIPA and then killed the OSA card link. Our session died. When we looked at the routes on the Windows NT server, we found a host route for the VIPA via the OSA card IP address. It took some time for this route to disappear.

This scenario would be uncommon for a user's workstation, but many installations may well have Windows NT servers on the same network as their z/OS servers.

2. Routing protocols require time to converge whenever the status of a link changes. If the session (TN3270, for example) times out before the routers can converge, then the session will be lost. In the case of host links into the z/OS stack, then probably only one or two levels of routers need to change routing tables to react to a network change. That will normally occur quickly, but you should keep this in mind.

If the routing protocol is OSPF, the convergence time can be improved because the OSPF protocol is based on the link state and not a distance vector.

## 7.2.4 Using SOURCEVIPA

Coding SOURCEVIPA in the profile will make the TCP/IP stack insert a static VIPA IP address as a source address on certain outbound packets. If SOURCEVIPA is coded, the VIPA address will be used for UDP (and RAW) data originating from this host and for new outbound connections in which the application does not specify which local IP address should be used.

The VIPA address used is always the closest static VIPA configured above the address of the outgoing interface in the home list. That is, once a packet's outgoing interface is determined, the local IP address corresponding to that interface is located in the home list. The stack then chooses the lowest VIPA in the home list above the local IP address found. Since this selection depends on VIPAs higher in the home list, dynamic VIPAs are never candidates for selection. Only static VIPAs are used with SOURCEVIPA.

We usually code the VIPA IP address as the first in the home list and the physical interface next will be used to transport outbound datagrams. If you do not want to use the VIPA address as a source in packets through a particular link, you can specify the IP address of the link before the VIPA IP address in the home list.

## 7.3 Configuring OMPROUTE

The following sections describe the OMPROUTE configurations that we used. The configuration common in OMPROUTE, whether using the OSPF or RIP routing protocol, is listed first, followed by OSPF and RIP specific configurations.

For further information on OMPROUTE, refer to Chapter 6, "Dynamic routing with OMPROUTE" on page 103. For information on configuring ORouted in a VIPA environment, see Appendix A, "Working with ORouted" on page 181.

The contents of TCPIP.DATA are shown in Figure 7-3.

```
TCPIPJOBNAME TCPIPC 1
HOSTNAME MVS03C
DOMAINORIGIN itso.ral.ibm.com
NSINTERADDR 9.24.106.15
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 10
RESOLVERUDPRETRIES 2
DATASETPREFIX TCPIP 2
MESSAGECASE MIXED
```

Figure 7-3 TCPIP.DATA

OMPROUTE uses the TCPIPJOBNAME parameter **1** to determine which TCP/IP stack to connect to, and DATASETPREFIX **2** is the data set hlq used when searching for the OMPROUTE configuration file.

Figure 7-4 shows part of the TCPIPC profile we used on the RA03 host.



```

;***** TOP OF DATA *****
;*TCPIP.TCPPARMS.R2617(PROF03C) SYSPLEX DISTRIBUTOR RA03 - SANDRAEF
;*****
IPCONFIG
  DATAGRAMFWD           1
  SYSPLEXROUTING       2
  ARPTO 1200
  IGNOREREDIRECT       3
  DYNAMICXCF 172.16.233.3 255.255.255.0 1 4
  STOPONCLAWERROR
  TTL 60
  VARSUBNETTING        5

PORT
  520 UDP OMPROUTE     6
;520 UDP OMVS

AUTOLOG 5
  OMPROUTE              7
  ENDAUTOLOG

  DEVICE M032216B MPCPTP AUTORESTART
  LINK M032216B MPCPTP M032216B

  DEVICE EN103 LCS 2064
  LINK EN103 ETHEROR802.3 1 EN103

VIPADYNAMIC 8
  VIPADEFINE MOVE IMMED 255.255.255.0 172.16.251.3
  VIPABACKUP 100 172.16.251.39
ENDVIPADYNAMIC

HOME
  172.16.100.3 M032216B
  9.24.105.76 EN103

START M032216B
START EN103

```

Figure 7-4 TCPIP profile

- 1 The DATAGRAMFWD parameter enables the forwarding of datagrams through this stack.
- 2 The SYSPLEXROUTING parameter specifies that this host is a part of a sysplex environment and will communicate the interface changes to WLM. The following message will confirm that it is enabled: SysplexRouting support is enabled.
- 3 The IGNOREREDIRECT parameter makes the stack ignore ICMP redirect packets. The OMPROUTE server does not support ICMP redirect packets, and will force the use of IGNOREREDIRECT if it is not coded.
- 4 Dynamic XCF support is enabled by this parameter. 172.16.233.3 is the XCF IP address used for the home list. The subnet mask and metric will be defined in the OMPROUTE configuration file.
- 5 VARSUBNETTING ensures this stack will support variable subnets.

6 The PORT statement is used to reserve port 520 for the OMPROUTE daemon. This is only needed if RIP is being implemented. If OMPROUTE is started from the UNIX System Services shell, the PORT statement should be used to reserve the RIP as shown here:

```
PORT
    520 UDP OMVS
```

7 The AUTOLOG statement starts the OMPROUTE server when the TCPIP stack starts.

8 The VIPADYNAMIC block defines the Dynamic VIPAs used by this stack.

### 7.3.1 OMPROUTE with OSPF

We carried out tests using the OSPF routing protocol. We use the network configuration described in Figure 7-1 on page 142, which includes our two systems in separate LPARs, RA03 and RA39.

Dynamic routing was enabled by starting the OMPROUTE daemon on each stack. We configured OMPROUTE to use the OSPF routing protocol. The two external routers were also configured to run OSPF. Since we were connected to a production network (the 9.0.0.0 IBM network), we also received some OSPF information from other production routers.

The contents of the OMPROUTE configuration file are found in Figure 7-5. The OMPROUTE routing application reads the file to determine which routing protocol to implement: OSPF and/or RIP protocols. If you want to implement your CS for z/OS IP as an OSPF router, you must configure the interfaces with the OSPF\_Interface statement. If there is an interface that implements the RIP protocol, you define this interface with the RIP\_Interface statement. OMPROUTE can handle both routing protocols at the same time.

Parameters such as maximum transmission unit (MTU), subnet mask, and interface name are configured in the OSPF\_interface, RIP\_interface, and Interface statements in the OMPROUTE configuration file.

```

Area      Area_Number=0.0.0.0          1
Stub_Area=NO
      Authentication_type=None;
RouterID=172.16.100.3;                2
ROUTESA_CONFIG ENABLED=YES           3
      COMMUNITY="MVSsubagent";
OSPF_Interface IP_Address=172.16.100.3  4
      Name=M032216B
      Cost0=3
      Subnet_mask=255.255.255.0
      MTU=32768;
OSPF_Interface IP_Address=9.24.105.76  5
      Name=EN103
      Cost0=6
      Subnet_mask=255.255.255.0
      MTU=32768;
OSPF_Interface IP_Address=172.16.251.*  6
      Subnet_mask=255.255.255.0
      Cost0=8
      Non_Broadcast=Yes
      MTU=32768;
OSPF_Interface IP_Address=172.16.233.*  7
      Subnet_mask=255.255.255.0
      Cost0=8
      Non_Broadcast=Yes
      MTU=32768;
OSPF_Interface IP_Address=172.16.252.*
      Subnet_mask=255.255.255.0
      Cost0=8
      Non_Broadcast=Yes
      MTU=32768;
AS_Boundary_routing                    8
      Import_Direct_Routes=YES;

```

Figure 7-5 OMPROUTE configuration file OM03CCFG

**1** These parameters are set for an OSPF area. In our network, the only area is defined as the backbone area (Area\_Number=0.0.0.0) and as a non-stub area (Stub\_Area=NO). If you specify Stub\_Area=YES, the area will not receive any inter-autonomous system (AS) routes. You cannot configure virtual links through a stub area or a router within the stub area as an AS boundary router.

**2** ROUTERID statement should be configured to assign every router a unique router ID. It is highly recommended that the router ID be an IP address of a physical interface or a static VIPA, not a Dynamic VIPA.

**3** This statement is coded to configure the OMPROUTE SNMP subagent to communicate with the CS for z/OS IP agent. The COMMUNITY parameter must match the one defined in SNMP.

**4** This statement sets the OSPF parameters for the 2216 ESCON MPC+ interface. The IP address and the name of the interface should match the ones in the HOME statement in the TCP/IP profile.

**5** This statement sets the OSPF parameters for the OSA Token-Ring interface. We configured a higher cost for this interface than the 2216 router interface.

**6** We configured DVIPA using the OSPF\_Interface statement. Because the stacks have DVIPA and DVIPA ranges defined, this statement is coded with a wild card value. By using a wild card, we ensure that all configured interfaces whose IP address match will be configured as interfaces. A point-to-point interface that is neither an OSPF nor a RIP interface should be configured to OMPROUTE via the Interface statement.

**7** This statement sets the OSPF parameters for the XCF links between the stacks in the sysplex. This time, we specified the IP address with a wild card (\*): 172.16.233.\* instead of 172.16.233.3.

**8** This statement enables AS boundary routing capability. It allows OMPROUTE to import routes learned from other methods such as the RIP protocol, static routes, and direct routes from other ASs into this OSPF domain. Because we have defined VIPA on the OSPF\_Interface statement, Import\_Direct\_Routes=YES is necessary. This definition will import the direct routes to VIPA into the OSPF routing domain and the routes will be advertised to the adjacent routers.

**4, 5, 6, 7** The MTU size defined on each OSPF interface must not exceed the MTU size defined on the IP interface in the partner OSPF router. The OSPF MTU size is exchanged between routers, and some products (such as z/OS) check that the largest possible OSPF packet can be received on the appropriate interface. If not, OSPF is disabled on that interface.

**Note:** According to the Information APAR II11555, the OSPF\_Interface statement is recommended for use when configuring VIPA interfaces to OMPROUTE in an OSPF environment. There is no need to define Import\_Direct\_Routes=YES in the AS\_Boundary\_routing statement for the VIPA interface. If the OSPF protocol is *not* being used on *any* interfaces, then the Interface statement is used to configure the VIPA to OMPROUTE. Please refer to the text of APAR II11555 for details. We did not test this alternative form of definition.

### 7.3.2 OMPROUTE with RIP

The OMPROUTE daemon can be used to implement RIP. Since it can also implement OSPF, OMPROUTE can make it easier to migrate your network's routing protocol from RIP to OSPF.

The contents of the OMPROUTE configuration file are found in Figure 7-6. The OMPROUTE routing application reads the file to determine which routing protocol it is going to use: RIP and/or OSPF protocols. If you want to implement your CS for z/OS IP as a RIP router, you must configure your interfaces with the RIP\_Interface statement. To implement OSPF protocol over an interface, define this interface with the OSPF\_Interface statement. OMPROUTE can handle both routing protocols at the same time.

Parameters such as maximum transmission unit (MTU), subnet mask, and interface name are configured via the OSPF\_interface, RIP\_interface, and Interface statements in the OMPROUTE configuration file.

```

ROUTESA_CONFIG ENABLED=YES 1
                COMMUNITY="MVSsubagent";
RIP_Interface IP_Address=172.16.100.3 2
              Name=M032216B
              Subnet_mask=255.255.255.0
              RIPV2=Yes
              In_Metric=0
              Out_Metric=0
              MTU=32768;
RIP_Interface IP_Address=9.24.105.76 3
              Name=EN103
              Subnet_mask=255.255.255.0
              RIPV2=Yes
              In_Metric=1
              Out_Metric=1
              MTU=32768;
Interface IP_Address=172.16.251.* 4
          Subnet_mask=255.255.255.0
          In_Metric=2
          Out_Metric=2
          MTU=32768;
RIP_Interface IP_Address=172.16.233.* 5
              Subnet_mask=255.255.255.0
              RIPV2=Yes
              In_Metric=2
              Out_Metric=2
              MTU=32768;

```

Figure 7-6 OMPROUTE configuration file

- 1** This statement is coded to configure the OMPROUTE SNMP subagent to communicate with the CS for z/OS IP agent. The COMMUNITY parameter must match the one defined in SNMP.
- 2** This statement sets the RIP parameters for the 2216 ESCON MPC+ interface. The IP address and the name of the interface should match the ones on the HOME statement in the TCP/IP profile. We configured a lower metric for the 2216 ESCON MPC+ than the other interfaces to ensure that this interface will take precedence over others.
- 3** This statement sets the RIP parameters for the OSA card interface.
- 4** We configured DVIPA using the Interface statement. Because the stacks have DVIPA and DVIPA ranges defined, this statement is coded with a wild card value. By using a wild card, we ensure that all configured interfaces whose IP address matches the wild card will be configured as an interface. A point-to-point interface that is neither an OSPF nor a RIP interface should be configured to OMPROUTE via the Interface statement.
- 5** This statement sets the OSPF parameters for the XCF links between the stacks in the sysplex. This time, we specified the IP address with a wild card (\*): 172.16.233.\* instead of 172.16.233.3.

## 7.4 A VIPA takeover example

One of the functions supported by Dynamic VIPA is automatic VIPA takeover. It provides VIPA backup capability in another stack in the sysplex environment, should the stack fail.

We implemented automatic VIPA takeover in our environment, as shown in Figure 7-7.

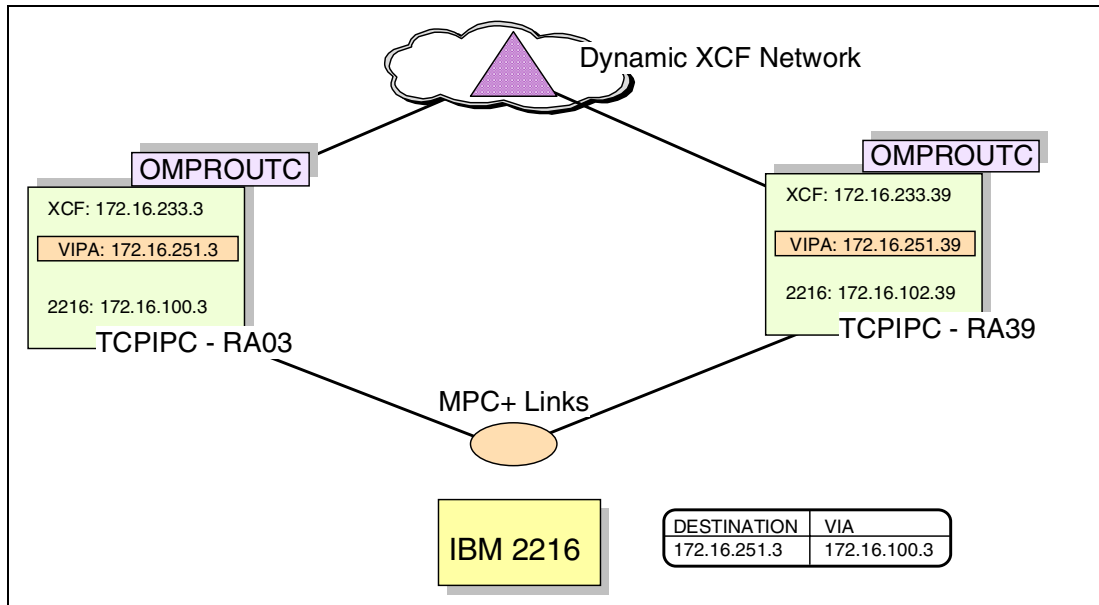


Figure 7-7 Dynamic VIPA example

The DVIPA IP address in the RA03 host is 172.16.251.3, and the DVIPA IP address owned by RA39 is 172.16.251.39. RA03 and RA39 provide VIPA backup for each other. Figure 7-8 shows a portion of the profile data set on RA03.

```
VIPADYNAMIC
VIPADefine MOVE IMMED 255.255.255.0 172.16.251.3
VIPABACKUP 100 172.16.251.39
VIPARANGE DEFINE MOVEABLE NONDISRUPT 255.255.255.0 172.16.240.193
ENDVIPADYNAMIC
```

Figure 7-8 VIPADYNAMIC block on RA03 host

In case of failure of the TCPIPC stack in RA03, the VIPA IP address 172.16.251.3 will continue to be reachable and will move to the RA39 host.

The result of the command **D TCPIP,TCPIPC,NETSTAT,ROUTE** issued to the TCPIPC stack in the RA39 host is illustrated in Figure 7-9.

```
D TCPIP,TCPIPC,N,ROUTE
EZZ2500I NETSTAT CS V2R10 TCPIPC 189
DESTINATION      GATEWAY          FLAGS  REFCNT  INTERFACE
172.16.251.0     172.16.102.254  UG     000000  M392216B
172.16.251.3     172.16.233.3   UGH    000000  EZAXCF03
172.16.251.39   0.0.0.0         UH     000000  VIPLAC10FB1C
```

Figure 7-9 Display route command on RA39 host

The display shows the host route to the DVIPA IP address of RA03 (172.16.251.3) in the 2216 during normal operation. If the TCPIPC stack on the RA03 host fails, its DVIPA IP address moves to the TCPIPC stack on the RA39 host, as shown in Figure 7-10.

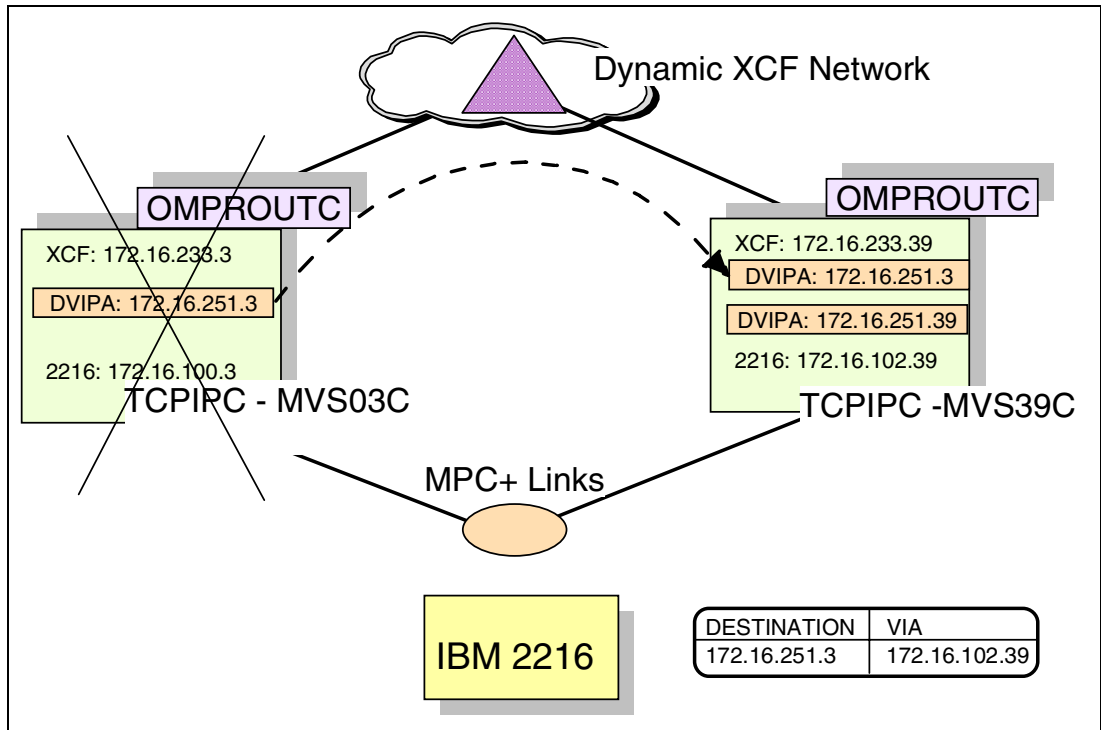


Figure 7-10 Backup dynamic VIPA

In Figure 7-10, we can see the host route to DVIPA 172.16.251.3 in the 2216 router after a TCPIPC stack failure on RA03. Figure 7-11 shows the result of the `D TCPIP,TCPIPC,NETSTAT,ROUTE` command issued to the TCPIPC stack in the RA39 host. It shows the host route to DVIPA 172.16.251.3 after the TCPIPC stack on RA03 failure.

```

D TCPIP,TCPIPC,N,ROUTE
EZZ2500I NETSTAT CS V2R10 TCPIPC 168
DESTINATION      GATEWAY          FLAGS  REFCNT  INTERFACE
DEFAULT          172.16.102.254  UG     000000  M392216B
172.16.251.0     172.16.102.254  UG     000000  M392216B
172.16.251.3     0.0.0.0         UH     000000  VIPLAC10FB03
172.16.251.39   0.0.0.0         UH     000000  VIPLAC10FB1C
    
```

Figure 7-11 Display route in RA39 host after RA03 host failure







## Interfacing with Cisco EIGRP

This chapter describes the configuration and interaction of dynamic routing within the sysplex using OSPF and dynamic routing within Cisco EIGRP networks. While dynamic routing is normally thought of purely as a networking function, in the sysplex environment with multiple system images, TCP/IP stacks, and application spaces, dynamic routing on the host becomes attractive. This is even more apparent when you want to make use of dynamic VIPAs, Sysplex Distributor, and redundant shared interfaces.

In this chapter, we cover the configuration used in our lab environment. We'll examine the specific configuration steps to implement OSPF in the sysplex and how to connect to a Cisco network that uses EIGRP.

## 8.1 EIGRP overview

The Enhanced Interior Gateway Routing Protocol (EIGRP) is categorized as a hybrid routing protocol. Similar to a distance vector algorithm such as RIP, EIGRP uses metrics to determine network paths. However, like a link state protocol such as OSPF, topology updates in an EIGRP environment are event driven.

EIGRP, as the name implies, is an interior gateway protocol designed for use within an autonomous system. In properly designed networks, EIGRP has the potential for improved scalability and faster convergence over standard distance vector algorithms. EIGRP is also better positioned to support complex, highly redundant networks.

EIGRP is a proprietary protocol developed by Cisco Systems, Inc. At the time of writing this redbook, it is not an IETF standard protocol.

### 8.1.1 Features of EIGRP

EIGRP provides several benefits. Some of these benefits are also available in distance vector or link state algorithms.

- ▶ **Faster convergence:** EIGRP maintains a list of alternate routes that can be used if a preferred path fails. When the path fails, the new route is immediately installed in the IP routing table. No route recomputation is performed.
- ▶ **Partial routing updates:** when EIGRP discovers a neighboring router, each device exchanges its entire routing table. After the initial information exchange, only routing table changes are propagated. There is no periodic rebroadcasting of the entire routing table.
- ▶ **Low bandwidth utilization:** during normal network operations, only hello packets are transmitted through a stable network.
- ▶ **CIDR and VLSM:** EIGRP supports supernetting and variable-length subnet masks. This allows the network administrator to efficiently allocate IP address resources.
- ▶ **Route summarization:** EIGRP supports the ability to summarize routing announcements. This limits the advertisement of unnecessary subnet information.
- ▶ **Multiple protocols:** EIGRP can provide network layer routing for AppleTalk, IPX and IP networks.
- ▶ **Unequal cost load balancing:** EIGRP supports the simultaneous use of multiple unequal cost paths to a destination. Each route is installed in the IP routing table. EIGRP also intelligently balances traffic load over the multiple paths.

### 8.1.2 Terminology

EIGRP uses specific terminology to describe the operation of the protocol:

- ▶ **Successor:** for a specific destination, the successor is the neighbor router currently used for packet forwarding. This device has the least-cost path to the destination and is guaranteed not to be participating in a routing loop. To reach the target network shown in Figure 8-1, router B is the current successor for router A.
- ▶ **Feasible successor:** a feasible successor assumes forwarding responsibility when the current successor router fails. The set of feasible successors represents the devices that can become a successor without requiring a route recomputation or introducing routing loops.
- ▶ **The set of feasible successors to a destination is determined by reviewing the complete list of minimum cost paths advertised by neighboring routers.** From this list, neighbors that

have an advertised metric less than the current routing table metric are considered feasible successors.

Figure 8-1 provides an example of a feasible successor relationship.

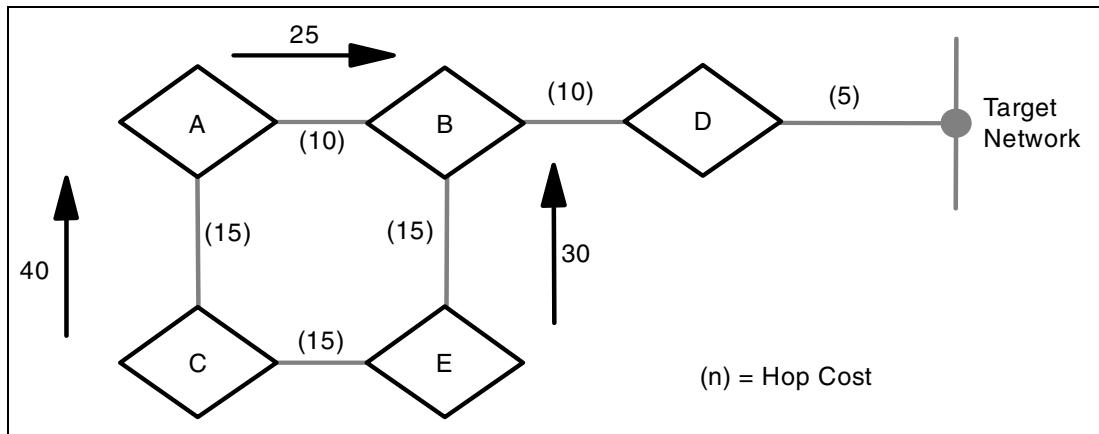


Figure 8-1 EIGRP feasible successors

In this diagram, the costs to reach the target network are shown. For example, the cost from router C to the target network is 40 (15 + 10 + 10 + 5). The cost from router E to the target network is 30 (15 + 10 + 5).

Router E is advertising a cost (30) that is lower than the current routing table metric on router C (40). Therefore, router C recognizes router E as a feasible successor to reach the target network. Note that the reverse is not true. The cost advertised by router C (40) is higher than the current route on router E (30). Therefore, router E does not recognize router C as a feasible successor to the destination network.

- ▶ Neighbor table: EIGRP maintains a table to track the state of each adjacent neighbor. The table contains the address and interface used to reach the neighbor. It also contains the last sequence number contained in a packet from the neighbor. This allows the reliable transport mechanism of EIGRP to detect out-of-order packets.
- ▶ Topology table: EIGRP uses a topology table to install routes into the IP routing table. The topology table lists all destination networks currently advertised by neighboring routers. The table contains all the information needed to build a set of distances and vectors to each destination. This information includes:
  - The smallest bandwidth available on a segment used to reach this destination.
  - The total delay, reliability, and loading of the path.
  - The minimum MTU used on the path.
  - The feasible distance of the path. This represents the best metric along the path to the destination network. It includes the metric used to reach the neighbor advertising the path.
  - The reported distance of the path. This represents the total metric along the path to a destination network, as advertised by an upstream neighbor.
  - The source of the route. EIGRP marks external routes. This provides the ability to implement policy controls that customize routing patterns.

An entry in the topology table can have one of two states:

- Passive state: the router is not performing a route recomputation for the entry.
  - Active state: the router is performing a route recomputation for the entry. If a feasible successor exists for a route, the entry never enters this state. This avoids processor-intensive route recomputation.
- ▶ Reliable transport protocol: EIGRP can guarantee the ordered delivery of packets to a neighbor. However, not all types of packets must be reliably transmitted. For example, in a network that supports multicasting, there is no need to send individual, acknowledged hello packets to each neighbor. To provide efficient operation, reliability is provided only when needed. This improves convergence time in networks containing varying speed connections.

### 8.1.3 Neighbor discovery and recovery

EIGRP can dynamically learn about other routers on directly attached networks. This is similar to the Hello protocol used for neighbor discovery in an OSPF environment.

Devices in an EIGRP network exchange hello packets to verify that each neighbor is operational. Like OSPF, the frequency used to exchange packets is based on the network type. Packets are exchanged at five-second intervals on high bandwidth links (for example, LAN segments). Otherwise, hello packets on lower bandwidth connections are exchanged every 60 seconds.

Like OSPF, EIGRP uses a hold timer to remove inactive neighbors. This timer indicates the amount of time that a device will continue to consider a neighbor active without receiving a hello packet from the neighbor.

### 8.1.4 The DUAL algorithm

A typical distance vector protocol uses periodic updates to compute the best path to a destination. It uses distance, next hop, and local interface costs to determine the path. Once this information is processed, it is discarded. EIGRP does not rely on periodic updates to converge on the topology. Instead, it builds a topology table containing each of its neighbors' advertisements. Unlike a distance vector protocol, this data is not discarded.

EIGRP processes the information in the topology table to determine the best paths to each destination network. EIGRP implements an algorithm known as DUAL (Diffusing Update Algorithm). This algorithm provides several benefits:

- ▶ The DUAL algorithm guarantees loop-free operations throughout the route computation and convergence period.
- ▶ The DUAL algorithm allows all routers to synchronize at the same time. This is unlike a RIP environment, in which the propagation of routing updates causes devices to converge at different rates.
- ▶ The DUAL algorithm allows routers not involved with a topology change to avoid route recomputation.

The DUAL algorithm is used to find the set of feasible successors for a destination. When an adverse condition occurs in the network, the alternate route is immediately added to the IP routing table. This avoids unnecessary computation to determine an alternate path. If no feasible successor is known, a route recomputation occurs. This behavior is shown in Figure 8-2 on page 159 and Figure 8-3 on page 160.

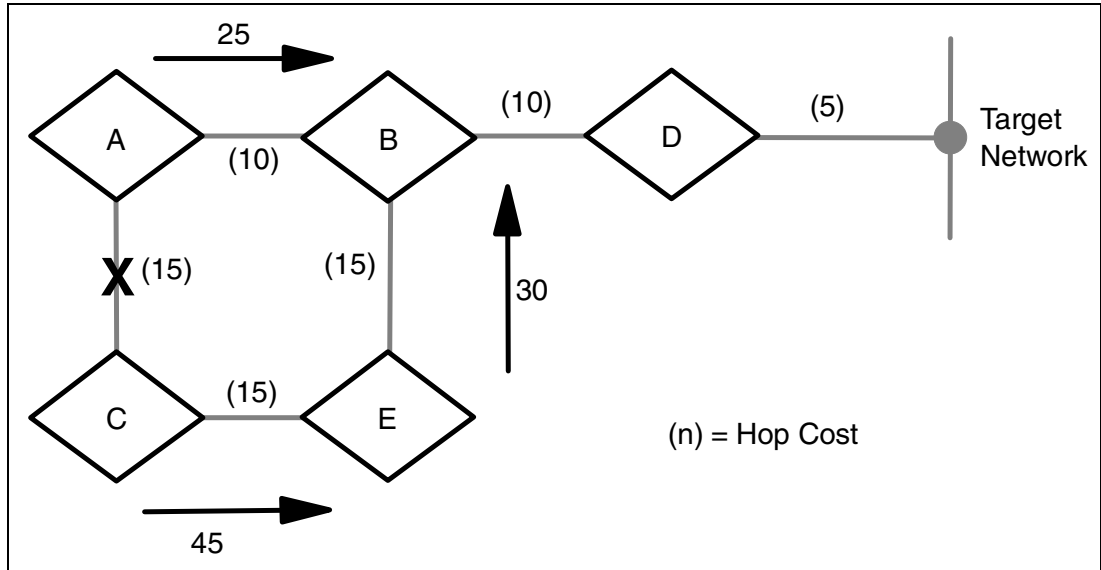


Figure 8-2 Using a feasible successor

In this example, router C uses router E as a feasible successor to reach the target network. If the connection between router A and router C fails, router C will immediately reroute traffic through router E. The new route is updated in the IP routing table.

### Route recomputation

A route recomputation occurs when there is no known feasible successor to the destination. The process starts with a router sending a multicast query packet to determine whether any neighbor is aware of a feasible successor to the destination. A neighbor replies if it has a feasible successor. If the neighbor does not have a feasible successor, the neighbor may return a query indicating it also is performing a route recomputation.

Figure 8-3 shows an example of querying to determine a feasible successor. In this example, router E does not have a feasible successor to the target network. When the link connecting router E and router B fails, router E must determine a new path. Router E sends a multicast query to each of its neighbors. Router C has a feasible successor and responds to router E. Router E updates its IP routing table with the new path at a cost of 55.

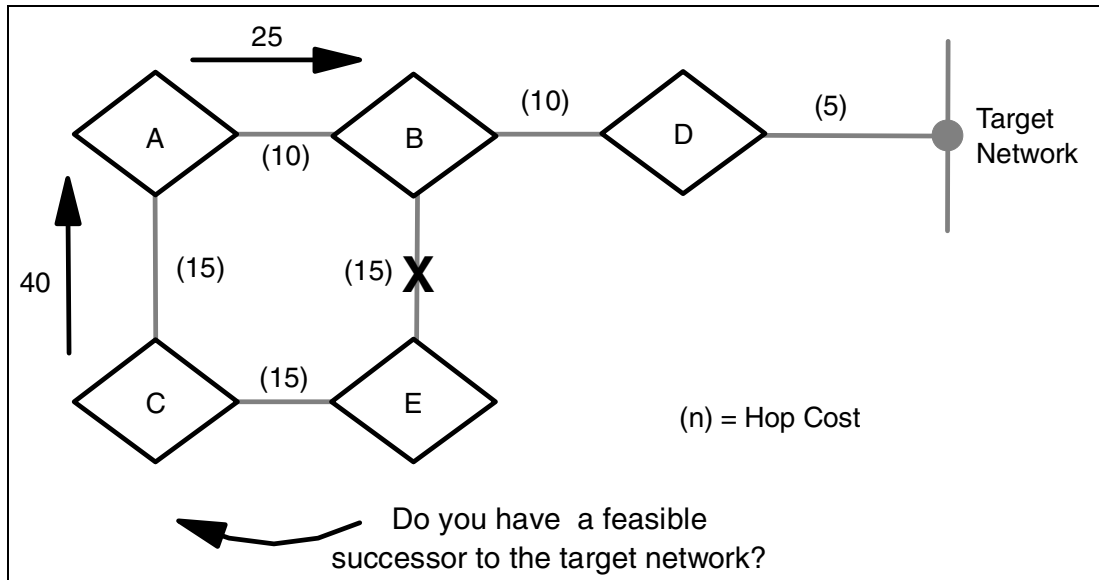


Figure 8-3 Query for a feasible successor

When the link to a neighbor fails, all routes that used that neighbor as the only feasible successor require a route recomputation.

### EIGRP metrics

EIGRP uses a mathematical formula to determine the metric associated with a path. By default, the formula references the minimum bandwidth of a segment used to reach the destination. It also sums the delays on the path. The default formula to determine the metric is:

$$\left[ \left( \frac{10^7}{\text{minbandwidth}} \right) + \text{sumofdelays} \right] \times 256$$

EIGRP supports the inclusion of other measurements in the metric calculation.

## 8.2 Sample topology

Before we can discuss configuration details, we must have a good understanding of the environment and the connectivity we want to achieve. Our goal in creating the test scenario was to show the configuration of dynamic routing using OSPF within the sysplex and the connectivity to a Cisco network using EIGRP routing, along with the redistribution necessary to achieve a stable environment. Figure 8-4 illustrates the network used in our lab environment.

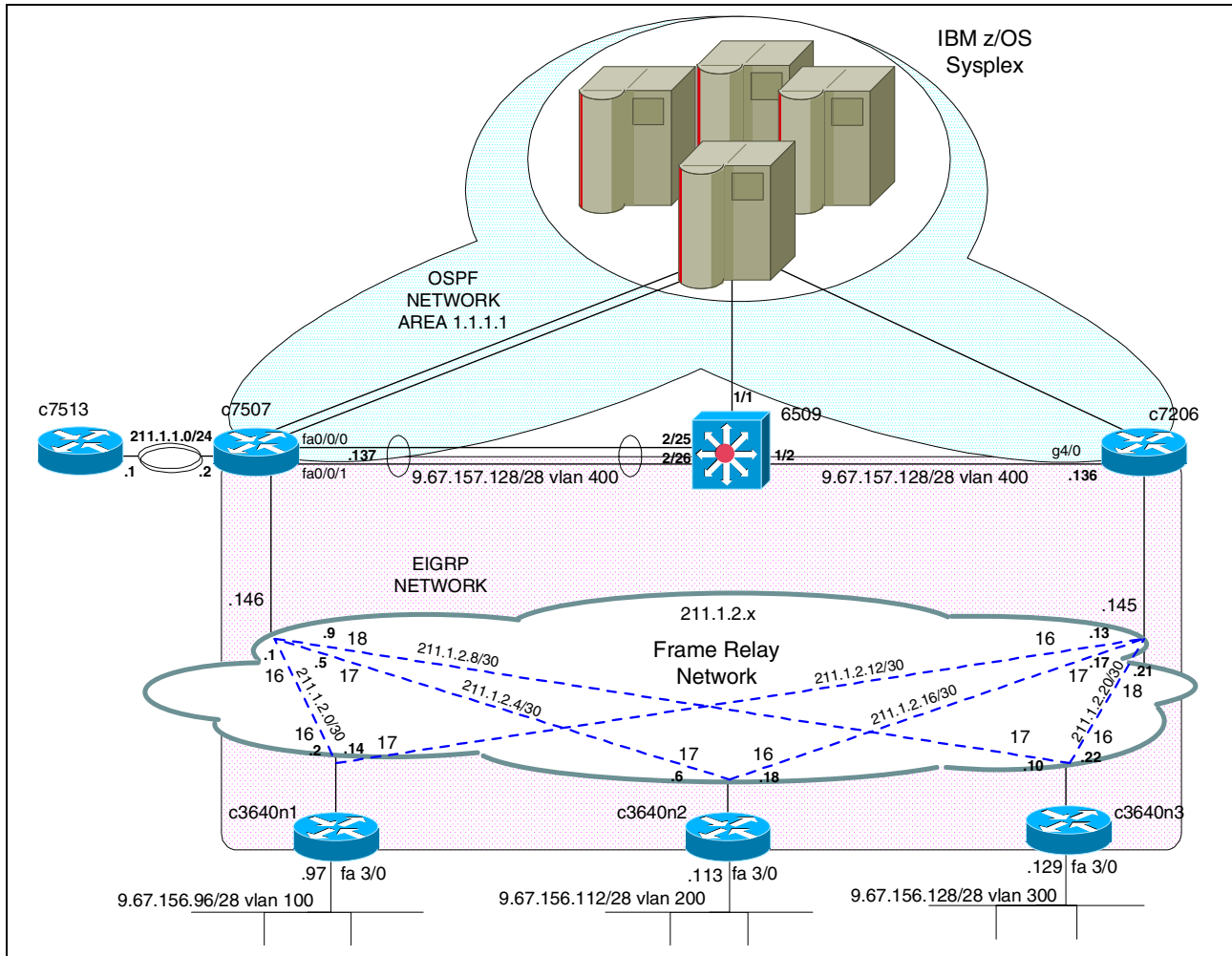


Figure 8-4 High-level network diagram

As shown in Figure 8-4, our sysplex is connected to a Cisco Catalyst 6509 switch and two Cisco routers: a 7507 and a 7206. There is also connectivity from the sysplex to a Cisco 7513 that is not detailed in the above diagram. Three remote Cisco 3640 routers are connected via a frame relay network. The frame relay network included a Cisco IGX 8400 WAN switch.

**Important:** The sysplex environment used in our lab tests was shared among other groups and connected to a production network. Because of this, some special design considerations were taken into account in the creation of our test network.

This meant, for example, that we did not code our sysplex as a stub area, as is typically recommended. Also, we coded our Dynamic VIPAs as Interface statements and not as OSPF\_Interface statements.

In our case, many of the subnets used in the network were previously assigned. Some of the subnets were added specifically for our purposes and are not really part of any engineered addressing scheme. When you deploy OSPF routing and initially design your own addressing scheme for your sysplex environment, it is important that you choose your subnets and addresses carefully, following an enterprise addressing scheme. Plan ahead and reserve enough address space for all of the potential systems and TCP/IP stacks that you will implement. You will find that the extra time spent analyzing your requirements and planning will pay off.

## 8.2.1 Routing topology

We used EIGRP for the network routing protocol so we could describe the configuration required to redistribute routing information between EIGRP and OSPF and vice versa. As mentioned before, the sysplex environment was already connected to a production network and shared by other groups. Therefore, we decided to separate the production network from our test environment in such a way as to be able to accomplish our objectives with the least disruption possible to other testers.

Our network, therefore, was segmented into a separate OSPF area that contained only the sysplex. We designated this area to be area 1.1.1.1. This strategy is recommended for most enterprises. It is wise to create an area specifically for the sysplex. It is also recommended that the area be defined as a stub area to minimize routing updates and OSPF topology calculations on the host systems.

The Cisco router network will be configured for EIGRP with the two 7000 series routers acting as autonomous system border routers (ASBRs). It will be the responsibility of the ASBRs to perform the redistribution of routing information between EIGRP and OSPF.



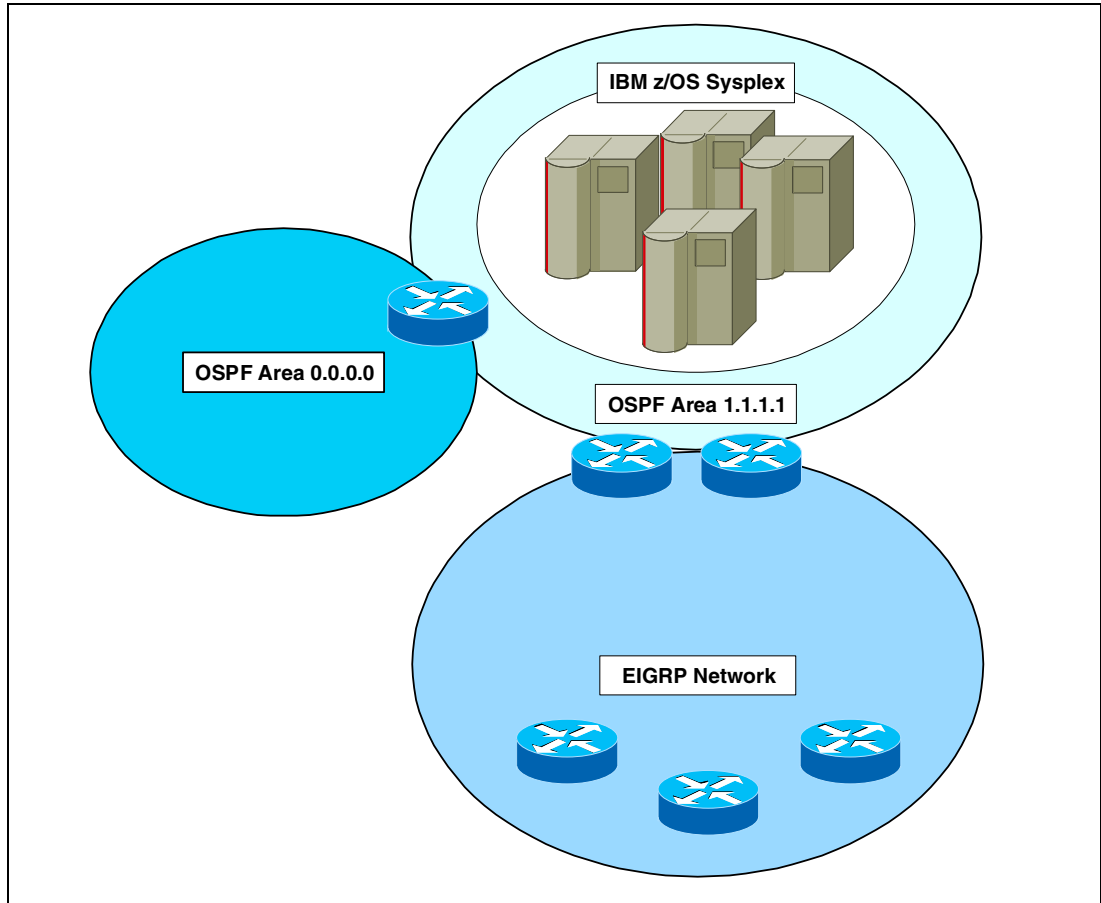


Figure 8-5 Routing topology

As depicted in Figure 8-5, the sysplex environment we used in our tests was configured as OSPF area 1.1.1.1. There was a connection to a production network through a 7513 router (see path to area 0.0.0.0). Our small test network consisted of two 7000 series routers and three 3640 routers. The two 7000 series routers redistribute routing information between OSPF and EIGRP.

### 8.3 OSPF configuration in the sysplex

The topology for the sysplex includes four LPARs, MVS001, MVS062, MVS069, and MVS154. Each LPAR contains one TCP/IP stack. Each of the LPARs is connected to the two 7000 series routers over an ESCON channel as well as over a shared Gigabit Ethernet OSA-Express adapter. The ESCON interface to the 7507 is configured for CMPC+ and the ESCON interface to the 7206 is configured for CLAW. Refer to Figure 8-6 on page 164 for details on IP interface and subnet addresses.

The routers that connect via the shared OSA-Express adapter are also configured with Generic Routing Encapsulation (GRE) tunnels. This is necessary to ensure that the shared OSA-Express adapter can distinguish packets destined for each stack when distributed VIPAs are active on multiple stacks, as in the case of multi-node load balancing.

Within the sysplex, XCF links also connect each of the stacks on each LPAR.

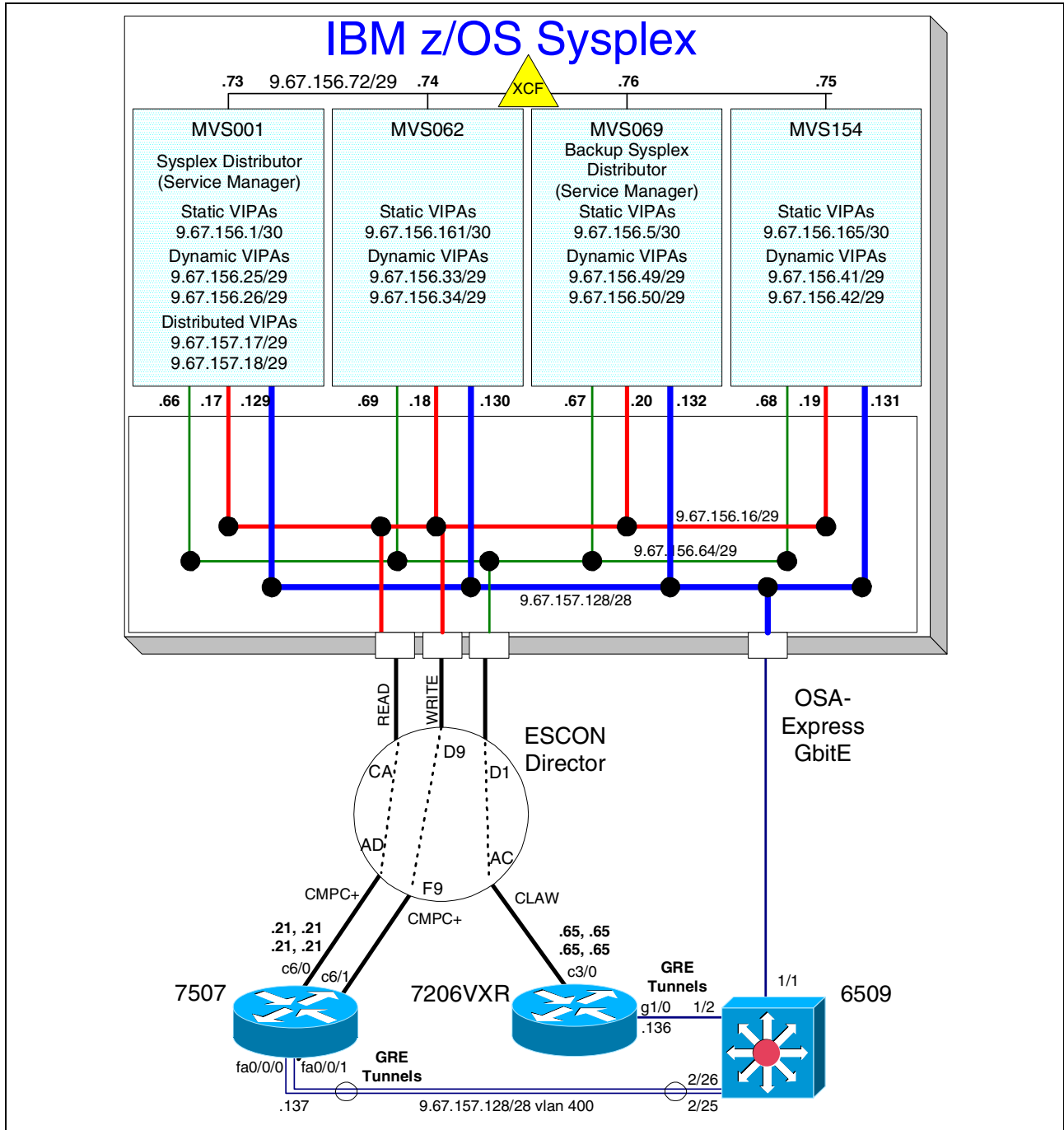


Figure 8-6 z/OS sysplex topology

### 8.3.1 OMPROUTE configuration

OMPROUTE was defined to be a started task. The AUTOLOG statement in the TCPIP.PROFILE starts OMPROUTE automatically when the stack is activated. The OMPROUTE procedure (Example 8-1) references data set DPTA39.TCPIP.PROFILES.OMPENV to set its environment.

### Example 8-1 OMPROUTE PROC

---

```
//OMPROUTE PROC RE=OM,HOST=&SYSNAME
//OMPROUTE EXEC PGM=BXPBATCH,REGION=&RE,TIME=NOLIMIT,
//      PARM='PGM /usr/sbin/omproute'
//*      PARM='PGM /usr/sbin/omproute -s1 -t2 -d4'
//STDENV DD DSN=DPTA39.TCPIP.PROFILES.OMPENV(&HOST),DISP=SHR
//STDOUT DD PATH='/tmp/omproute.stdout',
//      PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//      PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDERR DD PATH='/tmp/omproute.stderr',
//      PATHOPTS=(OWRONLY,OCREAT,OAPPEND),
//      PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
```

---

The member, MVS001, from DPTA39.TCPIP.PROFILES.OMPENV is included in Example 8-2.

### Example 8-2 OMPROUTE ENV

---

```
RESOLVER_CONFIG=/'DPTA39.TCPIP.DATA(MVS001) '
OMPROUTE_FILE=/'DPTA39.TCPIP.PROFILES.OMPCONF(MVS001) '
OMPROUTE_DEBUG_FILE=/tmp/omproute.log
```

---

The environment variable OMPROUTE\_FILE points to the OMPROUTE configuration data set. This data set contains the statements and parameters that dictate OMPROUTE's operation. The OMPROUTE configuration file for MVS001 is listed in Example 8-3.

### Example 8-3 OMPROUTE configuration file for MVS001

---

```
RouterID=9.67.156.1;
Area
  Area_Number=1.1.1.1;
AS_Boundary_Routing
  Import_Direct_Routes=YES;
Comparison=Type1;
OSPF_Interface
  Name = GIGELINK
  IP_Address = 9.67.157.129
  Attaches_To_Area = 1.1.1.1
  MTU = 1500
  Retransmission_Interval = 5
  Transmission_Delay = 40
  Hello_Interval = 10
  Dead_Router_Interval = 40
  Cost0 = 1
  Subnet = Yes
  Subnet_Mask = 255.255.255.240;
OSPF_Interface
  Name = CISCO1
  IP_Address = 9.67.156.66
  Destination_Addr = 9.67.156.65
  Attaches_To_Area = 1.1.1.1
  MTU = 4096
  Retransmission_Interval = 5
  Transmission_Delay = 120
  Hello_Interval = 30
  Dead_Router_Interval = 120
  Cost0 = 25
  Router_Priority = 2
```

```

Subnet = Yes
Subnet_Mask = 255.255.255.248;
OSPF_Interface
Name = CISCO2
IP_Address = 9.67.156.17
Destination_Addr = 9.67.156.21
Attaches_To_Area = 1.1.1.1
MTU = 4472
Retransmission_Interval = 5
Transmission_Delay = 120
Hello_Interval = 30
Dead_Router_Interval = 120
Cost0 = 25
Router_Priority = 2
Subnet = Yes
Subnet_Mask = 255.255.255.248;
;SPF_Interface
; Name = CISCO3
; IP_Address = 9.67.157.242
; Destination_Addr = 9.67.157.241
; Attaches_To_Area = 1.1.1.1
; MTU = 4096
; Retransmission_Interval = 5
; Transmission_Delay = 120
; Hello_Interval = 30
; Dead_Router_Interval = 120
; Cost0 = 25
; Router_Priority = 2
; Subnet = Yes
; Subnet_Mask = 255.255.255.248;
OSPF_Interface
Name = EZAXCFN5
IP_Address = 9.67.156.73
Subnet_Mask = 255.255.255.248
MTU = 4472
Hello_Interval = 20
Retransmission_Interval = 40
Dead_Router_Interval = 120
Cost0 = 200
Attaches_To_Area = 1.1.1.1
Router_Priority = 0
Subnet = YES;
OSPF_Interface
Name = EZAXCFN6
IP_Address = 9.67.156.73
Subnet_Mask = 255.255.255.248
MTU = 4472
Hello_Interval = 20
Retransmission_Interval = 40
Dead_Router_Interval = 120
Cost0 = 200
Attaches_To_Area = 1.1.1.1
Router_Priority = 0
Subnet = YES;
OSPF_Interface
Name = EZAXCFN7
IP_Address = 9.67.156.73
Subnet_Mask = 255.255.255.248
MTU = 4472
Hello_Interval = 20

```

```

Retransmission_Interval = 40
Dead_Router_Interval = 120
Cost0 = 200
Attaches_To_Area = 1.1.1.1
Router_Priority = 0
Subnet = YES;
Interface
  Name = DynamicVipa1
  IP_Address = 9.67.156.25
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DynamicVipa2
  IP_Address = 9.67.156.26
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DynamicVipa3
  IP_Address = 9.67.156.33
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DynamicVipa4
  IP_Address = 9.67.156.34
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DynamicVipa5
  IP_Address = 9.67.156.41
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DynamicVipa6
  IP_Address = 9.67.156.42
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DynamicVipa7
  IP_Address = 9.67.156.49
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DynamicVipa8
  IP_Address = 9.67.156.50
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DistVipa1
  IP_Address = 9.67.157.17
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = DistVipa2
  IP_Address = 9.67.157.18
  MTU = 4472
  Subnet_Mask = 255.255.255.248;
Interface
  Name = TOVTAM
  IP_Address = 9.67.156.2
  MTU = 4472

```

```
Subnet_Mask = 255.255.255.252;  
Interface  
Name = VLINK0  
IP_Address = 9.67.156.1  
MTU = 4472  
Subnet_Mask = 255.255.255.252;
```

---

Note the `Area_Number = 1.1.1.1` and the `Attaches_To_Area = 1.1.1.1` statements in the file above. These statements set the area number for the stack and the area to which the OSPF interface is attached.

While we recommend the use of `OSPF_Interfaces` for Dynamic VIPAs, we used `Interface` statements because our sysplex was connected to a backbone production network. Using `Import_Direct_Routes=YES` afforded us the ability to still distribute our DVIPAs to our EIGRP while having them not participate in OSPF directly.

Note also that we did not configure the area as a stub area. While this would be recommended for most enterprise customers, we elected not to configure a stub area because we did not want to inject a default route into the sysplex area. This was because, as stated earlier, our test environment was connected via another path to a production network. The other reason we did not configure the sysplex as a stub area is so we could show how to perform redistribution in autonomous system border routers.

**Important:** It is recommended that you configure your sysplex OSPF area as a stub area. In general, we recommend that you put your sysplex in a stand-alone stub if you have any other OSPF network machines besides the sysplex. If your sysplex is the only OSPF portion in your network, you should not make it a stub.

To configure the area as a stub area, on the `Area` statement use the parameter:

```
Stub_Area = Yes
```

If you specify `Stub_area = YES`, the area does not receive any AS external link advertisements, reducing the size of your database and decreasing memory usage for routers in the stub area. Be aware that you cannot configure virtual links through a stub area. Nor can you configure a Cisco router within the stub area as an AS boundary router (some routers allow an ASBR to be the ABR between the backbone and the stub area).

You also cannot configure the backbone as a stub area. External routing in stub areas is based on a default route. Each border area router attaching to a stub area originates a default route for this purpose. The cost of this default route is also configurable with the AREA statement.

Note the following in the configuration shown in Example 8-3 on page 165:

```
AS_Boundary_Routing
  Import_Direct_Routes=YES;
```

This is used to advertise the non-OSPF\_Interfaces to OSPF neighbors. By configuring local interfaces (XCF, VIPA, etc.) as Interface rather than OSPF\_Interface, extraneous routing information is not exchanged between stacks in the sysplex.

### 8.3.2 Verify routing from the host

There are several useful commands you can use to verify your routing configuration from the host.

The output from the **netstat route** command is shown in Example 8-4.

Example 8-4 netstat route command

netstat route:

```

MVS TCP/IP NETSTAT CS V1R2          TCPIP NAME: TCP          14:28:09,
Destination      Gateway          Flags      Refcnt      Interface
-----
1.1.1.1          9.67.157.137   UGHO      000000     GIGELINK
1.1.1.2          9.67.157.136   UGHO      000000     GIGELINK
9.67.156.1       0.0.0.0        UH         000000     VLINKO
9.67.156.2       0.0.0.0        UH         000000     TOVTAM
9.67.156.4       9.67.157.132   UGO       000000     GIGELINK
9.67.156.5       9.67.157.132   UGHO      000000     GIGELINK
9.67.156.16      0.0.0.0        UC         000000     CISCO2
9.67.156.17      0.0.0.0        UH         000000     CISCO2
9.67.156.18      9.67.157.130   UGHO      000000     GIGELINK
9.67.156.19      9.67.157.131   UGHO      000000     GIGELINK
9.67.156.20      9.67.157.132   UGHO      000000     GIGELINK
9.67.156.21      0.0.0.0        UHC        000000     CISCO2
9.67.156.21      9.67.157.137   UGHO      000000     GIGELINK
9.67.156.25      0.0.0.0        UH         000000     VIPL09439C19
9.67.156.26      0.0.0.0        UH         000000     VIPL09439C1A
9.67.156.32      9.67.157.130   UGO       000000     GIGELINK
9.67.156.33      9.67.157.130   UGHO      000000     GIGELINK
9.67.156.34      9.67.157.130   UGHO      000000     GIGELINK
9.67.156.40      9.67.157.131   UGO       000000     GIGELINK
9.67.156.41      9.67.157.131   UGHO      000000     GIGELINK
9.67.156.42      9.67.157.131   UGHO      000000     GIGELINK
9.67.156.48      9.67.157.132   UGO       000000     GIGELINK
9.67.156.49      9.67.157.132   UGHO      000000     GIGELINK
9.67.156.50      9.67.157.132   UGHO      000000     GIGELINK
9.67.156.65      9.67.157.136   UGHO      000000     GIGELINK
9.67.156.66      0.0.0.0        UH         000000     CISCO1
9.67.156.67      9.67.157.132   UGHO      000000     GIGELINK
9.67.156.68      9.67.157.131   UGHO      000000     GIGELINK
9.67.156.69      9.67.157.130   UGHO      000000     GIGELINK
9.67.156.72      0.0.0.0        UC         000000     EZAXCFN6
9.67.156.72      0.0.0.0        UC         000000     EZAXCFN7
9.67.156.72      0.0.0.0        UC         000000     EZAXCFN5
9.67.156.73      0.0.0.0        UH         000001     EZAXCFN6
9.67.156.73      0.0.0.0        UH         000000     EZAXCFN7
9.67.156.73      0.0.0.0        UH         000000     EZAXCFN5
9.67.156.74      0.0.0.0        UHS        000000     EZAXCFN5
9.67.156.75      0.0.0.0        UHS        000000     EZAXCFN6
9.67.156.76      0.0.0.0        UHS        000000     EZAXCFN7
9.67.156.96      9.67.157.136   UGO       000000     GIGELINK
9.67.156.112     9.67.157.136   UGO       000037     GIGELINK
9.67.156.128     9.67.157.136   UGO       000000     GIGELINK
9.67.156.160     9.67.157.130   UGO       000000     GIGELINK
9.67.156.161     9.67.157.130   UGHO      000000     GIGELINK
9.67.156.164     9.67.157.131   UGO       000000     GIGELINK
9.67.156.165     9.67.157.131   UGHO      000000     GIGELINK
9.67.157.0       9.67.157.136   UGO       000000     GIGELINK
9.67.157.4       9.67.157.136   UGO       000002     GIGELINK
9.67.157.8       9.67.157.136   UGO       000000     GIGELINK
9.67.157.17      0.0.0.0        UH         000000     VIPL09439D11
9.67.157.18      0.0.0.0        UH         000000     VIPL09439D12
9.67.157.128    0.0.0.0        UO         000000     GIGELINK
9.67.157.129    0.0.0.0        UH         000001     GIGELINK

```



127.0.0.1	0.0.0.0	UH	000004	LOOPBACK
211.1.2.12	9.67.157.136	UGO	000000	GIGELINK
211.1.2.16	9.67.157.136	UGO	000000	GIGELINK
211.1.2.20	9.67.157.136	UGO	000000	GIGELINK

The output from the **DISPLAY TCPIP, ,OMPROUTE,OSPF,NEIGHBOR** command issued from MVS001 is shown in Example 8-5.

*Example 8-5 D TCPIP, ,OMPR,OSPF,NBR*

**D TCPIP, ,OMPR,OSPF,NBR**

```
EZZ7851I NEIGHBOR SUMMARY 405
NEIGHBOR ADDR  NEIGHBOR ID  STATE  LSRXL  DBSUM  LSREQ  HSUP  IFC
9.67.156.76     9.67.156.5      128    0      0      0     OFF  EZAXCFN7
9.67.156.65     9.67.156.89     128    0      0      0     OFF  CISCO1
9.67.157.131    9.67.156.75     128    0      0      0     OFF  GIGELINK
9.67.157.130    9.67.156.74     8      0      0      0     OFF  GIGELINK
9.67.157.132    9.67.156.5      128    0      0      0     OFF  GIGELINK
9.67.157.137    9.67.156.82     8      0      0      0     OFF  GIGELINK
9.67.157.136    9.67.156.89     8      0      0      0     OFF  GIGELINK
9.67.156.74     9.67.156.74     128    0      0      0     OFF  EZAXCFN5
9.67.156.75     9.67.156.75     128    0      0      0     OFF  EZAXCFN6
```

## 8.4 ASBR configuration and redistribution

An autonomous system border router (ASBR) is a router that uses both OSPF and any other routing protocol (in our case, EIGRP).

Let us examine the OSPF configuration for the 7206 ASBR. The configuration for the 7507 is not detailed because it is, essentially, the same as what is shown here.

*Example 8-6 Router OSPF configuration*

```
router ospf 1
router-id 9.67.156.89
log-adjacency-changes
auto-cost reference-bandwidth 1000
redistribute eigrp 1 subnets route-map sendem
network 1.1.1.2 0.0.0.0 area 1.1.1.1
network 9.67.156.0 0.0.0.255 area 1.1.1.1
network 9.67.157.0 0.0.0.255 area 1.1.1.1
maximum-paths 3
```

The following items pertain to Example 8-6:

1. The **router ospf 1** command configures the OSPF routing process.
2. The network statements define the interfaces where OSPF runs and determine the OSPF area for the interfaces.
3. The **auto-cost reference-bandwidth 1000** command controls how OSPF calculates default metrics for the interfaces. The OSPF metric is calculated as the reference-bandwidth-value divided by the bandwidth, with reference-bandwidth-value equal to 108 by default, and bandwidth determined by the **bandwidth** command. The calculation gives Fast Ethernet a metric of 1. Setting reference-bandwidth to 1000

changes the calculation, so that Fast Ethernet has a cost of 10 and Gigabit Ethernet has a cost of 1.

4. The **redistribute** command causes EIGRP routing information to be imported to the OSPF process.

If you want to force the ASBR to generate a default route into an OSPF routing domain, you can do this. Whenever you specifically configure redistribution of routes into an OSPF routing domain, the router automatically becomes an ASBR. However, an ASBR does not, by default, generate a default route into the OSPF routing domain. To force a default route to be injected, use the command:

```
#default-information originate [always] [metric metric-value] [metric-type type-value]
[route-map map-name]
```

## 8.4.1 Redistribution

In our network, the 7000 series routers ran two routing protocols simultaneously: OSPF and EIGRP. The Cisco IOS software can redistribute information from one routing protocol to another. In our case, we redistributed information from EIGRP into OSPF and from OSPF into EIGRP. This allows OSPF routes to be known in the remote 3640 routers within the EIGRP network and allows the LPARs within the sysplex to be aware of routes from EIGRP.

You must be careful not to cause routing loops when using redistribution. We used a method utilizing *route maps* to conditionally control the redistribution of routes between routing domains.

Refer to Example 8-7 as we describe the configuration.

*Example 8-7 Redistribution using route maps*

---

```
router eigrp 1
 redistribute ospf 1 route-map stopem

router ospf 1
 redistribute eigrp 1 subnets route-map sendem

route-map stopem deny 10
 match tag 1
!
route-map stopem permit 11
 match tag 0
!
route-map sendem permit 10
 match ip address 10
 set tag 1

access-list 10 permit any
```

---

Notice the **redistribute** command under router eigrp 1 and router ospf 1. When considering the action of the **redistribute** command, remember that the associated **router** command indicates the *destination* routing process and the **redistribute** command specifies the *source* of the routing information.

The **redistribute** command is defined as follows:

```
redistribute protocol [process-id] {level-1 | level-1-2 | level-2} [as-number] [metric
metric-value] [metric-type type-value] [match {internal | external 1 | external 2}] [tag
tag-value] [route-map map-tag] [weight number-value] [subnets]
```

In our case, the configuration commands **router ospf 1** and **redistribute eigrp 1 subnets route-map sendem** mean:

- ▶ That we want to redistribute from EIGRP (source) into OSPF (destination).
- ▶ That the scope of redistribution includes subnets.
- ▶ That the route map named sendem should be interrogated to filter the importation of routes from EIGRP into OSPF. Our route map tags these routes with a value of 1 so they do not come back into EIGRP.

The other **redistribute** command accomplishes the reverse, that is, OSPF into EIGRP.

In Example 8-7, two route maps are defined, called sendem and stopem.

The route-map named sendem is applied to the **redistribute** command for OSPF and the route-map named stopem is applied to the **redistribute** command for EIGRP.

The route-map command is defined as follows:

```
route-map map-tag [permit | deny] [sequence-number]
```

Where:

- ▶ The *map-tag* is the name that is referenced by the **redistribute** command.
- ▶ The **permit** or **deny** keyword has the following effect:
  - If the match criteria are met for this route map, and the **permit** keyword is specified, the route is redistributed as controlled by the set actions. In the case of policy routing, the packet is policy routed.  
  
If the match criteria are not met, and the **permit** keyword is specified, the next route map with the same map tag is tested. If a route passes none of the match criteria for the set of route maps sharing the same name, it is not redistributed by that set.
  - If the match criteria are met for the route map, and the **deny** keyword is specified, the route is not redistributed or in the case of policy routing, the packet is not policy routed, and no further route maps sharing the same map tag name will be examined. If the packet is not policy routed, the normal forwarding algorithm is used.
- ▶ The *sequence-number* is a number that indicates the position a new route map is to have in the list of route maps already configured with the same name. If given with the no form of this command, the position of the route map should be deleted.

Note also the match and set commands. One or more **match** commands and one or more **set** commands typically follow a **route-map** command. If there are no **match** commands, then everything matches. If there are no **set** commands, nothing is done (other than the match). Therefore, you need at least one **match** or **set** command.

In our case, these configuration commands apply when EIGRP routes are redistributed into OSPF:

```
route-map sendem permit 10
match ip address 10
set tag 1
```

- ▶ The **route-map** command defines a route map named **sendem**.
- ▶ Because of the **permit** keyword, all routes that match are redistributed from EIGRP into OSPF.
- ▶ The match condition, **match ip address 10**, refers to access control list 10 (permits any) and matches all routes so that all routes redistributed from EIGRP into OSPF are tagged with a value of 1 using **set tag 1**.

These commands apply when OSPF routes are redistributed into EIGRP:

```
route-map stopem deny 10
  match tag 1
route-map stopem permit 11
  match tag 0
```

- ▶ Remember, all native OSPF routes will have a tag value of 0.
- ▶ The **route-map** commands define a route map named **stopem** with match conditions evaluated in the order designated by the sequence numbers, 10 and 11.
- ▶ The first **match tag 1** matches all routes tagged with 1 (these are tagged by the route map **sendem** below). Because of the **deny** keyword, these routes will not be redistributed into EIGRP from OSPF.
- ▶ The second route map is evaluated and the match condition allows all EIGRP routes to be redistributed.

**Important:** Our route-maps, then, have the effect of blocking circular routes. Routes that are learned by OSPF are redistributed into EIGRP and routes that are learned by EIGRP are redistributed into OSPF. But, routes learned by EIGRP that are redistributed into OSPF are not passed back into EIGRP again.

The effect of the **route-map** commands and redistribution can be seen using the **show ip ospf database** command, as illustrated in Example 8-8.

*Example 8-8 show ip ospf database command*

```
C7200-Z55#sh ip ospf dat

          OSPF Router with ID (9.67.156.89) (Process ID 1)

          Router Link States (Area 1.1.1.1)

Link ID        ADV Router    Age         Seq#          Checksum Link count
9.67.156.1    9.67.156.1   1295       0x800000CC   0x931E   9
9.67.156.5    9.67.156.5   1291       0x800000C8   0x65C8   11
9.67.156.74   9.67.156.74  1263       0x800000C0   0xE915   11
9.67.156.75   9.67.156.75  1247       0x800000B7   0xC837   11
9.67.156.82   9.67.156.82  1739       0x80000076   0xF5B3   6
9.67.156.89   9.67.156.89  1552       0x80000275   0xF5C7   7

          Net Link States (Area 1.1.1.1)

Link ID        ADV Router    Age         Seq#          Checksum
9.67.157.131  9.67.156.75  1258       0x8000003F   0x4F4D

          Type-5 AS External Link States

Link ID        ADV Router    Age         Seq#          Checksum Tag
9.67.156.0    9.67.156.1   156        0x800000A0   0x259D   0
```

9.67.156.1	9.67.156.1	156	0x800000A0	0x2D91	0
9.67.156.4	9.67.156.5	715	0x800000A0	0xE4D5	0
9.67.156.5	9.67.156.5	719	0x800000A0	0xECC9	0
9.67.156.16	9.67.156.5	430	0x80000066	0xC824	0
9.67.156.16	9.67.156.74	367	0x80000066	0x297E	0
9.67.156.16	9.67.156.75	139	0x80000066	0x2383	0
9.67.156.24	9.67.156.1	160	0x800000A0	0x1C92	0
9.67.156.25	9.67.156.1	160	0x800000A0	0x3C6A	0
9.67.156.26	9.67.156.1	160	0x800000A0	0x3273	0
9.67.156.32	9.67.156.74	920	0x8000009F	0x1648	0
9.67.156.33	9.67.156.74	920	0x8000009F	0x3620	0
9.67.156.34	9.67.156.74	920	0x8000009F	0x2C29	0
9.67.156.40	9.67.156.75	455	0x800000A0	0xBD96	0
9.67.156.41	9.67.156.75	455	0x800000A0	0xDD6E	0
9.67.156.42	9.67.156.75	455	0x800000A0	0xD377	0
9.67.156.48	9.67.156.5	719	0x800000A0	0x137F	0
9.67.156.49	9.67.156.5	720	0x800000A0	0x3357	0
9.67.156.50	9.67.156.5	720	0x800000A0	0x2960	0
9.67.156.64	9.67.156.82	994	0x8000005E	0x994B	1
9.67.156.72	9.67.156.1	161	0x800000A0	0x3A44	0
9.67.156.72	9.67.156.5	720	0x800000A0	0x2258	0
9.67.156.72	9.67.156.74	921	0x8000009F	0x84B1	0
9.67.156.72	9.67.156.75	455	0x800000A0	0x7CB7	0
9.67.156.96	9.67.156.82	491	0x80000061	0x22A7	1
9.67.156.96	9.67.156.89	1044	0x8000015B	0x1C6	1
9.67.156.112	9.67.156.82	492	0x80000061	0x8138	1
9.67.156.112	9.67.156.89	1044	0x8000015B	0x6057	1
9.67.156.128	9.67.156.82	492	0x80000061	0xE0C8	1
9.67.156.128	9.67.156.89	1044	0x8000015B	0xBFE7	1
9.67.156.160	9.67.156.74	922	0x8000009F	0x29B0	0
9.67.156.161	9.67.156.74	922	0x8000009F	0x31A4	0
9.67.156.164	9.67.156.75	456	0x800000A0	0xF8DA	0
9.67.156.165	9.67.156.75	456	0x800000A0	0x1CE	0
9.67.157.0	9.67.156.82	492	0x80000061	0x23F9	1
9.67.157.0	9.67.156.89	1044	0x8000015B	0x219	1
9.67.157.4	9.67.156.82	492	0x80000061	0xFA1E	1
9.67.157.4	9.67.156.89	1044	0x8000015B	0xD93D	1
9.67.157.8	9.67.156.82	492	0x80000061	0xD242	1
9.67.157.8	9.67.156.89	1044	0x8000015B	0xB161	1
9.67.157.16	9.67.156.1	163	0x800000A0	0x6154	0
9.67.157.17	9.67.156.1	163	0x800000A0	0x812C	0
9.67.157.18	9.67.156.1	163	0x800000A0	0x7735	0
211.1.2.0	9.67.156.82	493	0x80000061	0xA18E	1
211.1.2.0	9.67.156.89	551	0x80000061	0x77B1	1
211.1.2.4	9.67.156.82	493	0x80000061	0x79B2	1
211.1.2.4	9.67.156.89	551	0x80000061	0x4FD5	1
211.1.2.8	9.67.156.82	493	0x80000061	0x51D6	1
211.1.2.8	9.67.156.89	552	0x80000061	0x27F9	1
211.1.2.12	9.67.156.82	493	0x80000061	0x29FA	1
211.1.2.12	9.67.156.89	552	0x8000015C	0x61B	1
211.1.2.16	9.67.156.82	493	0x80000061	0x11F	1
211.1.2.16	9.67.156.89	552	0x8000015C	0xDD3F	1
211.1.2.20	9.67.156.82	493	0x80000061	0xD843	1
211.1.2.20	9.67.156.89	552	0x8000015C	0xB563	1
C7200-Z55#					

---

Notice that the route tag values in the right-hand column show a value of 0 for native OSPF routes and 1 for routes learned from EIGRP.

**Tip:** Route maps are an effective way to safely redistribute routing information between autonomous systems. When routes from other protocols are redistributed into OSPF, each route is advertised individually in an external LSA. In many cases, you can limit the number of individual routes by configuring the Cisco IOS software to advertise a single route for all the redistributed routes that are covered by a specified network address and mask. Doing this will decrease the size of the OSPF link-state database and minimize OSPF route calculations on the host. Of course, to take advantage of this type of configuration, your addressing scheme must be such that summarization at the boundary is possible.

To advertise one summary route for all redistributed routes covered by a network address and mask, use the following command:

```
#summary-address ip-address mask prefix mask [not-advertise][tag tag]
```

## 8.4.2 Verify routing from the router

Here are some useful commands to be sure your routing configuration is operating as expected.

Example 8-9 shows the output from the **show ip ospf neighbor** command. In our network, each 7000 series router is connected to all four LPARs over an Ethernet (Ethernet port channel for the 7507 and Gigabit Ethernet for the 7206) and a channel interface. Notice that each LPAR shows a state of FULL.

*Example 8-9 show ip ospf neighbor*

---

```
C7200-Z55#sh ip ospf nei
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
9.67.156.1	1	FULL/DROTHER	00:00:36	9.67.157.129	GigabitEthernet 4/0
9.67.156.75	1	FULL/DROTHER	00:00:33	9.67.157.131	GigabitEthernet 4/0
9.67.156.5	1	FULL/DROTHER	00:00:34	9.67.157.132	GigabitEthernet 4/0
9.67.157.243	1	FULL/DROTHER	00:00:36	9.67.157.130	GigabitEthernet 4/0
9.67.156.82	1	FULL/BDR	00:00:32	9.67.157.137	GigabitEthernet 4/0
9.67.156.1	2	FULL/ -	00:01:39	9.67.156.66	Channel3/0
9.67.156.75	2	FULL/ -	00:01:48	9.67.156.68	Channel3/0
9.67.156.5	2	FULL/ -	00:01:50	9.67.156.67	Channel3/0
9.67.157.243	2	FULL/ -	00:01:56	9.67.156.69	Channel3/0
N/A	0	DOWN/ -	-	9.67.156.82	ATM2/0.1
N/A	0	DOWN/ -	-	9.67.156.81	ATM2/0.1

```
C7200-Z55#
```

```
NIVT7507#sh ip ospf nei
```

Neighbor ID	Pri	State	Dead Time	Address	Interface
9.67.156.1	1	FULL/DROTHER	00:00:36	9.67.157.129	Port-channel1
9.67.156.75	1	FULL/DROTHER	00:00:34	9.67.157.131	Port-channel1
9.67.156.5	1	FULL/DROTHER	00:00:35	9.67.157.132	Port-channel1
9.67.157.243	1	FULL/DROTHER	00:00:37	9.67.157.130	Port-channel1
9.67.156.89	1	FULL/DR	00:00:33	9.67.157.136	Port-channel1
9.67.156.1	2	FULL/ -	00:01:33	9.67.156.17	Channel6/2
9.67.156.75	2	FULL/ -	00:01:38	9.67.156.19	Channel6/2

```

9.67.156.5      2  FULL/ -      00:01:40    9.67.156.20  Channel6/2
9.67.157.243   2  FULL/ -      00:01:51    9.67.156.18  Channel6/2
NIVT7507#

```

Example 8-10 shows the output from the **show ip ospf interface** command for both the 7206 and the 7507. This command shows all interfaces configured for OSPF using the **network** command. Notice the OSPF costs for each of the interfaces in the output.

*Example 8-10 show ip ospf interface*

**C7200-Z55#sh ip ospf int**

```

CASA1 is up, line protocol is up
  Internet Address 1.1.1.2/32, Area 1.1.1.1
  Process ID 1, Router ID 9.67.156.89, Network Type LOOPBACK, Cost: 1
  Loopback interface is treated as a stub Host
GigabitEthernet4/0 is up, line protocol is up
  Internet Address 9.67.157.136/28, Area 1.1.1.1
  Process ID 1, Router ID 9.67.156.89, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DROTHER, Priority 0
  Designated Router (ID) 9.67.156.75, Interface address 9.67.157.131
  Backup Designated router (ID) 9.67.156.5, Interface address 9.67.157.132
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  Hello due in 00:00:02
  Index 8/8, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 6, maximum is 27
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 5, Adjacent neighbor count is 2
    Adjacent with neighbor 9.67.156.75 (Designated Router)
    Adjacent with neighbor 9.67.156.5 (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
Channel3/0 is up, line protocol is up
  Internet Address 9.67.156.65/29, Area 1.1.1.1
  Process ID 1, Router ID 9.67.156.89, Network Type POINT_TO_MULTIPOINT, Cost: 10
  Transmit Delay is 1 sec, State POINT_TO_MULTIPOINT,
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
  Hello due in 00:00:15
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 6, maximum is 27
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 4, Adjacent neighbor count is 4
    Adjacent with neighbor 9.67.156.74
    Adjacent with neighbor 9.67.156.75
    Adjacent with neighbor 9.67.156.1
    Adjacent with neighbor 9.67.156.5
  Suppress hello for 0 neighbor(s)
ATM2/0.1 is administratively down, line protocol is down
  Internet Address 9.67.156.89/28, Area 1.1.1.1
  Process ID 1, Router ID 9.67.156.89, Network Type POINT_TO_MULTIPOINT, Cost: 1
  Transmit Delay is 1 sec, State DOWN,
  Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
C7200-Z55#

```

**NIVT7507#sh ip ospf int**

```

Port-channel1 is up, line protocol is up
  Internet Address 9.67.157.137/28, Area 1.1.1.1

```

```

Process ID 7514, Router ID 9.67.156.82, Network Type BROADCAST, Cost: 1
Transmit Delay is 1 sec, State DROTHER, Priority 0
Designated Router (ID) 9.67.156.75, Interface address 9.67.157.131
Backup Designated router (ID) 9.67.156.5, Interface address 9.67.157.132
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
  Hello due in 00:00:09
Index 4/4, flood queue length 0
Next 0x0(0)/0x0(0)
Last flood scan length is 1, maximum is 25
Last flood scan time is 0 msec, maximum is 0 msec
Neighbor Count is 5, Adjacent neighbor count is 2
  Adjacent with neighbor 9.67.156.75 (Designated Router)
  Adjacent with neighbor 9.67.156.5 (Backup Designated Router)
Suppress hello for 0 neighbor(s)
Channel6/2 is up, line protocol is up
Internet Address 9.67.156.21/29, Area 1.1.1.1
Process ID 7514, Router ID 9.67.156.82, Network Type POINT_TO_MULTIPOINT, Cost
: 10
Transmit Delay is 1 sec, State POINT_TO_MULTIPOINT,
Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
  Hello due in 00:00:29
Index 3/3, flood queue length 0
Next 0x0(0)/0x0(0)
Last flood scan length is 1, maximum is 12
Last flood scan time is 0 msec, maximum is 0 msec
Neighbor Count is 3, Adjacent neighbor count is 3
  Adjacent with neighbor 9.67.156.74
  Adjacent with neighbor 9.67.156.75
  Adjacent with neighbor 9.67.156.5
Suppress hello for 0 neighbor(s)
ATM1/0.1 is administratively down, line protocol is down
Internet Address 9.67.156.82/28, Area 1.1.1.1
Process ID 7514, Router ID 9.67.156.82, Network Type POINT_TO_MULTIPOINT, Cost
: 1
Transmit Delay is 1 sec, State DOWN,
Timer intervals configured, Hello 30, Dead 120, Wait 120, Retransmit 5
CASA1 is up, line protocol is up
Internet Address 1.1.1.1/32, Area 1.1.1.1
Process ID 7514, Router ID 9.67.156.82, Network Type LOOPBACK, Cost: 1
Loopback interface is treated as a stub Host
NIVT7507#

```

Example 8-11 shows the output from the **show ip eigrp interface** command for both the 7206 and the 7507. This command shows all interfaces configured for EIGRP using the **network** command. Here, you can see the interfaces that connect to the three remote 3640 routers.

*Example 8-11 show ip eigrp interface*

```

C7200-Z55# sh ip eigrp int
IP-EIGRP interfaces for process 1

```

Interface	Peers	Xmit Queue Un/Reliable	Mean SRTT	Pacing Time Un/Reliable	Multicast Flow Timer	Pending Routes
Se5/0.1	1	0/0	40	0/23	151	0
Se5/0.2	1	0/0	41	0/23	143	0
Se5/0.3	1	0/0	38	2/95	227	0

```

C7200-Z55#

```



```
NIVT7507#sh ip eigrp int
IP-EIGRP interfaces for process 1
```

Interface	Peers	Xmit Queue Un/Reliable	Mean SRTT	Pacing Time Un/Reliable	Multicast Flow Timer	Pending Routes
Fd5/0	0	0/0	0	0/10	0	0
Se4/0/0.1	1	0/0	64	2/95	287	0
Se4/0/0.2	1	0/0	65	2/95	307	0
Se4/0/0.3	1	0/0	65	2/95	319	0

Example 8-12 shows the output from the **show ip route** command issued from the 7507. Notice the codes at the start of each line of output. The codes designate the type of route (C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area, E1 - OSPF external type 1, etc.). You can also see the administrative distance and the metric (or OSPF cost) in the brackets ([ ]) for each entry.

*Example 8-12 show ip route*

```
NIVT7507#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route
```

Gateway of last resort is not set

```

1.0.0.0/32 is subnetted, 2 subnets
C    1.1.1.1 is directly connected, CASA1
O    1.1.1.2 [110/2] via 9.67.157.136, 07:38:15, Port-channel1
7.0.0.0/30 is subnetted, 1 subnets
C    7.7.7.0 is directly connected, Tunnel1
8.0.0.0/30 is subnetted, 1 subnets
C    8.8.8.0 is directly connected, Tunnel62
9.0.0.0/8 is variably subnetted, 45 subnets, 4 masks
D    9.67.156.128/28 [90/10514432] via 211.1.2.10, 2d05h, Serial4/0/0.3
C    9.67.157.128/28 is directly connected, Port-channel1
O E1 9.67.156.164/30 [110/2] via 9.67.157.131, 07:38:15, Port-channel1
O E1 9.67.156.165/32 [110/2] via 9.67.157.131, 07:38:15, Port-channel1
O E1 9.67.156.160/30 [110/2] via 9.67.157.130, 07:38:15, Port-channel1
O E1 9.67.156.161/32 [110/2] via 9.67.157.130, 07:38:17, Port-channel1
O    9.67.156.68/32 [110/1] via 9.67.157.131, 07:38:17, Port-channel1
O    9.67.156.69/32 [110/1] via 9.67.157.130, 07:38:17, Port-channel1
O    9.67.156.66/32 [110/1] via 9.67.157.129, 07:38:17, Port-channel1
O    9.67.156.67/32 [110/1] via 9.67.157.132, 07:38:17, Port-channel1
D EX 9.67.156.64/29 [170/11024128] via 211.1.2.2, 2d05h, Serial4/0/0.1
           [170/11024128] via 211.1.2.10, 2d05h, Serial4/0/0.3
           [170/11024128] via 211.1.2.6, 2d05h, Serial4/0/0.2
O    9.67.156.65/32 [110/1] via 9.67.157.136, 07:38:17, Port-channel1
S    9.67.156.76/32 is directly connected, Tunnel69
S    9.67.156.74/32 is directly connected, Tunnel62
S    9.67.156.75/32 is directly connected, Tunnel154
O E1 9.67.156.72/29 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
           [110/2] via 9.67.157.132, 07:38:17, Port-channel1
           [110/2] via 9.67.157.130, 07:38:17, Port-channel1
S    9.67.156.73/32 is directly connected, Tunnel1
```

```

D      9.67.156.112/28 [90/10514432] via 211.1.2.6, 2d05h, Serial4/0/0.2
D      9.67.156.96/28 [90/10514432] via 211.1.2.2, 2d05h, Serial4/0/0.1
S      9.67.156.20/32 [1/0] via 9.67.156.20, Channel6/2
S      9.67.156.18/32 [1/0] via 9.67.156.18, Channel6/2
S      9.67.156.19/32 [1/0] via 9.67.156.19, Channel6/2
O E1   9.67.157.18/32 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
O E1   9.67.157.17/32 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
C      9.67.156.16/29 is directly connected, Channel6/2
O E1   9.67.157.16/29 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
O E1   9.67.156.26/32 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
O E1   9.67.156.24/29 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
O E1   9.67.156.25/32 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
O E1   9.67.156.4/30 [110/2] via 9.67.157.132, 07:38:17, Port-channel1
O E1   9.67.156.5/32 [110/2] via 9.67.157.132, 07:38:17, Port-channel1
D      9.67.157.4/30 [90/10639872] via 211.1.2.6, 2d05h, Serial4/0/0.2
O E1   9.67.156.0/30 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
O E1   9.67.156.1/32 [110/2] via 9.67.157.129, 07:38:17, Port-channel1
D      9.67.157.0/30 [90/10639872] via 211.1.2.2, 2d05h, Serial4/0/0.1
D      9.67.157.8/30 [90/10639872] via 211.1.2.10, 2d05h, Serial4/0/0.3
O E1   9.67.156.50/32 [110/2] via 9.67.157.132, 07:38:17, Port-channel1
O E1   9.67.156.48/29 [110/2] via 9.67.157.132, 07:38:17, Port-channel1
O E1   9.67.156.49/32 [110/2] via 9.67.157.132, 07:38:17, Port-channel1
O E1   9.67.156.34/32 [110/2] via 9.67.157.130, 07:38:17, Port-channel1
O E1   9.67.156.32/29 [110/2] via 9.67.157.130, 07:38:17, Port-channel1
O E1   9.67.156.33/32 [110/2] via 9.67.157.130, 07:38:17, Port-channel1
O E1   9.67.156.42/32 [110/2] via 9.67.157.131, 07:38:17, Port-channel1
O E1   9.67.156.40/29 [110/2] via 9.67.157.131, 07:38:17, Port-channel1
O E1   9.67.156.41/32 [110/2] via 9.67.157.131, 07:38:17, Port-channel1
11.0.0.0/30 is subnetted, 1 subnets
C      11.11.11.0 is directly connected, Tunnel69
211.1.2.0/30 is subnetted, 6 subnets
D      211.1.2.16 [90/11023872] via 211.1.2.6, 2d05h, Serial4/0/0.2
D      211.1.2.20 [90/11023872] via 211.1.2.10, 2d05h, Serial4/0/0.3
C      211.1.2.0 is directly connected, Serial4/0/0.1
C      211.1.2.4 is directly connected, Serial4/0/0.2
C      211.1.2.8 is directly connected, Serial4/0/0.3
D      211.1.2.12 [90/11023872] via 211.1.2.2, 2d05h, Serial4/0/0.1
12.0.0.0/30 is subnetted, 1 subnets
C      12.12.12.0 is directly connected, Tunnel154
NIVT7507#

```

---



## Working with ORouteD

ORouteD is a routing daemon that implements the RIP protocols described in RFC 1058 (RIP Version 1) and RFC 1753 (RIP Version 2). This appendix describes how to configure the ORouteD server. It provides practical examples of configuring ORouteD, based on configurations used in our test environment at ITSO-Raleigh.

We recommend OMPROUTE as the dynamic routing daemon of choice, rather than ORouteD. This is because ORouteD supports only RIP Version 1 and RIP Version 2 routing protocols, while OMPROUTE supports RIP Version 1, RIP Version 2 and OSPF protocols. OSPF is now the recommended routing protocol. See Chapter 6, “Dynamic routing with OMPROUTE” on page 103 for details on OMPROUTE and the various routing protocols.

For more information on ORouteD refer to *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775 and *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

# Configuring ORouted

This section describes the steps needed to configure an ORouted server.

## Resolver configuration file (TCPIP.DATA)

The resolver configuration file contains the following two important keywords:

► TCPIPjobname

The value assigned to this keyword will be used as the name of the TCP/IP system address space with which ORouted attempts to establish a connection.

► DATASETPREFIX

The value assigned to DATASETPREFIX will determine the high-level qualifier (hlq) used in the search order for other configuration files.

The resolver uses the following search order to allocate the actual resolver configuration file or data set to use:

1. GLOBALTCPIPDATA in the System Resolver configuration file
2. The environment variable RESOLVER\_CONFIG
3. /etc/resolv.conf
4. The data set specified on the SYSTCPD DD card
5. userid.TCPIP.DATA
6. SYS1.TCPPARMS(TCPDATA)
7. hlq.TCPIP.DATA or DEFAULTTCPIPDATA in the System Resolver configuration file

We have used the RESOLVER\_CONFIG environment variable to allocate the actual resolver configuration file for our test environment at ITSO-Raleigh.

The contents of this file in our test system at ITSO-Raleigh are shown below:

```
TCPIPJOBNAME T03ATCP
DATASETPREFIX TCP
```

## ROUTED server configuration file

The ORouted server on Communications Server for z/OS IP supports sending and receiving both RIP V1 and RIP V2 packets. The ORouted server configuration file is used to determine the mode of operation.

The following search order is used to locate the ORouted server configuration data set or file:

1. The environment variable ROUTED\_PROFILE
2. /etc/routed.profile
3. hlq.ROUTED.PROFILE

We have used the ROUTED\_PROFILE environment variable to allocate the actual ORouted server configuration file for our test environment at ITSO-Raleigh.

The contents of this file in our test system at ITSO-Raleigh are shown below:

```
; RIP_SUPPLY_CONTROL: RIP1
RIP_SUPPLY_CONTROL: RIP2M
RIP_RECEIVE_CONTROL: ANY
; RIP2_AUTHENTICATION_KEY: PASSWORD
```

In the ORouteD configuration, the keywords can be specified in mixed case. Comments and blanks are supported in ORouteD profile. As shown in the sample configuration, comments are identified by a semicolon (;) in any column. There should be no sequence numbers in the data set or file.

**Note:** You should be careful with RIP\_SUPPLY\_CONTROL. Adjacent routers that support only RIP V1 can misinterpret RIP V2 packets or even crash. For a detailed description of these options, see *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776. These options can be specified differently for each interface in the GATEWAYS file.

## GATEWAYS file

The GATEWAYS file or data set is used to further configure the routing table in addition to the information from received RIP packets.

The ORouteD server uses the following search order to locate the GATEWAYS configuration file or data set.

1. The environment variable GATEWAYS\_FILE
2. /etc/gateways
3. hlq.ETC.GATEWAYS

We have used the GATEWAYS\_FILE environment variable to allocate the actual ORouteD server configuration file for our test environment at ITSO-Raleigh.

## PROFILE.TCPIP configuration

In order to start an ORouteD server in an Communications Server for z/OS system, you have to update your TCPIP.PROFILE configuration data set. Figure A-1 shows part of a sample PROFILE data set we used at ITSO-Raleigh.

```

IPCONFig
VARSUBNETTING 1
IGNORERedirect 2
DATAGRamfwd 3
NOSOURCEVIPA 4
;SOURCEVIPA 4
SYSPLEXROUTING 9
DYNAMICXCF 192.168.233.38 255.255.255.0 1 10

AUTOLOG 5
  T03AROU 5
ENDAUTOLOG

PORT
  520 UDP T03AROU 6

; Link      Maxmtu  Metric  Subnet Mask  Dest Addr
BSDROUTINGPARMS TRUE
  EN1       1500    0       255.255.255.0  0           7
  TR1       1500    0       255.255.255.0  0           7
  LTR2      4000    0       255.255.255.0  0
  T03CTCP   4096    0       0               192.168.252.2  8
  RAS       32768   0       0               192.168.236.2  8
ENDBSDROUTINGPARMS

```

Figure A-1 Sample TCP/IP profile

1 Since our ORouteD server listens for both RIP V1 and RIP V2 packets, the variable subnet mask support of TCP/IP must be enabled.

2 The ORouteD server cannot recognize ICMP Redirect messages. So to allow the ORouteD server to have full control over the IP routing table contents, you should specify the IGNOREREDIRECT keyword in the IPCONFIG statement.

3 The IP forwarding function must be enabled when you are configuring an ORouteD server. However, if you use an z/OS system as an end-node system, such as an application server, the IP forwarding function is not mandatory. In this case, you can configure the NODATAGRAMFWD keyword in the IPCONFIG statement, and prevent an z/OS system from forwarding any IP packets. Note that if you want your z/OS system to take a role not only as an end node but also as an intermediate node, you have to enable the IP forwarding function on your system.

4 The SOURCEVIPA function is discussed in more detail in Chapter 7, “Routing with VIPA” on page 141.

5 If an ORouteD server will be started automatically when TCP/IP starts up, specify the cataloged procedure name for your ORouteD server on the AUTOLOG statement.

6 To ensure that UDP port 520 is reserved for your ORouteD server, add the name of the member containing the ORouteD cataloged procedure to the PORT statement.

If you want to start an ORouteD server from the UNIX System Services shell, use the special name OMVS as follows:

```
PORT 520 UDP OMVS
```

Note that in this case, you must be a superuser.

7 Configure the BSDROUTINGPARMS for each network interface with the MTU value and subnet mask for use by ORouteD on your z/OS system.

8 These links are point-to-point links (in this case MPC SAMEHOST link and MPC XCF link). If you want to transmit RIP V1 packets over a point-to-point link, you have to specify the -h option on the ORouteD server startup option. The -h option will force ORouteD to broadcast host routes and subnetwork routes for point-to-point links.

9 The SYSPLEXROUTING parameter is specified if TCP/IP is part of the MVS sysplex domain and should communicate interface changes to the workload manager (WLM). The following message confirms that its enabled: SysplexRouting support is enabled

10 The DYNAMICXCF parameter indicates that XCF dynamic support is enabled. The first address 192.168.233.38 is the IP address to be used for HOME statements for all dynamically generated XCF or Samehost links. The second parameter is the subnet\_mask and third is the cost\_metric. The subnet\_mask and cost\_metric are used by BSDROUTINGPARMS entry for ORouteD.

**Note:** The VTAM major node ISTLSXCF must be activated for dynamic XCF to work, except for stacks running on the same MVS image. The message DYNAMIC XCF DEFINITIONS ARE ENABLED will be displayed when the stack is started.

For more information on all the above parameters, refer to *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776.

## Creating a catalogued procedure

The ORoutedD server is a UNIX System Services application and requires the HFS. You can start an ORoutedD server from an MVS procedure or from the UNIX System Services shell command line. Since the ORoutedD server uses raw sockets, it has to have sufficient authority. In most implementations, ORoutedD would be started under the superuser authority.

In our test system at ITSO-Raleigh, we started an ORoutedD server as an MVS started task which had been associated with a superuser named *TCPIP3*.

All users who needed to start ORoutedD had their user IDs defined to RACF with control: access to MVS.ROUDEMGR.ORoutedD. The following commands should be issued by a user ID with Special Operations to authorize users to start ORoutedD:

```
RDEFINE OPERCMDS (MVS.ROUDEMGR.ORoutedD) UACC(NONE)
PERMIT MVS.ROUDEMGR.ROUTED ACCESS(CONTROL) CLASS(OPERCMDS) ID(userid)
SETROPTS RACLIST(OPERCMDS) REFRESH
```

Figure A-2 shows part of our catalogued procedure for the ORoutedD server in our system.

```
//T03AROU PROC MODULE='BPXBATCH'
/**
//ORoutedD EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
// PARM='PGM /usr/lpp/tcpip/sbin/ORoutedD' 1
/** PARM='PGM /usr/lpp/tcpip/sbin/ORoutedD -ep -hv'
/** PARM='PGM /usr/lpp/tcpip/sbin/ORoutedD -ep -t -t -t -t'
/**
//STDOUT DD PATH='/tmp/ORoutedD.03a.stdout', 2
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
/**
//STDERR DD PATH='/tmp/ORoutedD.03a.stderr', 2
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
/**
//STDENV DD PATH='/etc/ORoutedD.03a.env' 3
/**STDENV DD DSN=TCP.TCPPARMS(RD03AENV),DISP=SHR
/**
//CEEDUMP DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
/**
//SYSERR DD PATH='/tmp/ORoutedD.03a.syserr',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
```

Figure A-2 Our ORoutedD catalogued procedure

1 The executable module is installed in the HFS with the sticky bit set.

2 All output from the -d and -dp parameters is sent to STDOUT. The output from MODIFY TABLES is also written in STDOUT. Note, however, that if you want to get this information, -ep must be specified in the startup parameter list except when you specify the -d parameter, which enables the -ep parameter internally. All messages and trace information other than that from -d and -dp parameters is sent to the syslogd daemon also.

You can use only HFS files for STDOUT and STDERR. MVS data sets are not supported by ORoutedD servers.

**3** You can use the STDENV statement to define environment variables for use by the ORoutedD server. You can use either an HFS file or MVS data set for the STDENV statement. However, you have to use the PATH keyword to allocate an HFS file and the DSN keyword for an MVS data set, as shown in our sample procedure.

The contents of the STDENV file in our test system at ITSO-Raleigh are shown below:

```
RESOLVER_CONFIG=// 'TCP.TCPPARMS(TD03AROU) '  
ROUTED_PROFILE=// 'TCP.TCPPARMS(RD03APR) '  
GATEWAYS_FILE=// 'TCP.TCPPARMS(RD03AGW) '
```

You may define the following three environment variables using an environment configuration file allocated by the STDENV statement.

- ▶ Resolver configuration file - RESOLVER\_CONFIG
- ▶ RoutedD server configuration file - ROUTED\_PROFILE
- ▶ GATEWAYS file - GATEWAYS\_FILE

While we used only MVS data sets in our test system at ITSO-Raleigh, you may use either HFS files or MVS data sets to store this configuration information.

## **RIP Version 2 communication over token ring**

If ORoutedD will be communicating over the token-ring using RIP V2 multicast with routers attached to the token-ring that are not listening (at the DLC layer) for the token-ring multicast address 0xC000.0004.0000, the following TRANSLATE statement should be coded in the TCPIP profile: TRANSLATE 224.0.0.0 IBMTR FFFFFFFFFF linkname. Without this statement, RIP V2 multicast packets will be discarded at the DLC layer by those routers that are not listening for the token-ring multicast address.

Instead of the TRANSLATE statement, you can use RIP\_SUPPLY\_CONTROL RIP2B in the GATEWAYS file.

## **Configuring ORoutedD servers on multiple TCP/IP stacks**

You can use ORoutedD servers in multiple stack environments. In this case, a copy of ORoutedD server has to be started for each stack. ORoutedD server determines which TCP/IP stack it should connect to based on the TCPIPJOBNAME keyword. To associate with a particular TCP/IP stack, use the environment variable RESOLVER\_CONFIG to point to the data set or HFS file that defines a unique TCPIPJOBNAME keyword.

A unique GATEWAYS file can be associated with each copy of ORoutedD by using the GATEWAYS\_FILE environment variable as shown in Figure A-3.

In our test system at ITSO-Raleigh, we created a unique environment configuration file that is allocated by the STDENV DD statement in each cataloged procedure. In each STDENV data set or HFS file, you can allocate a unique resolver configuration file and GATEWAYS file.



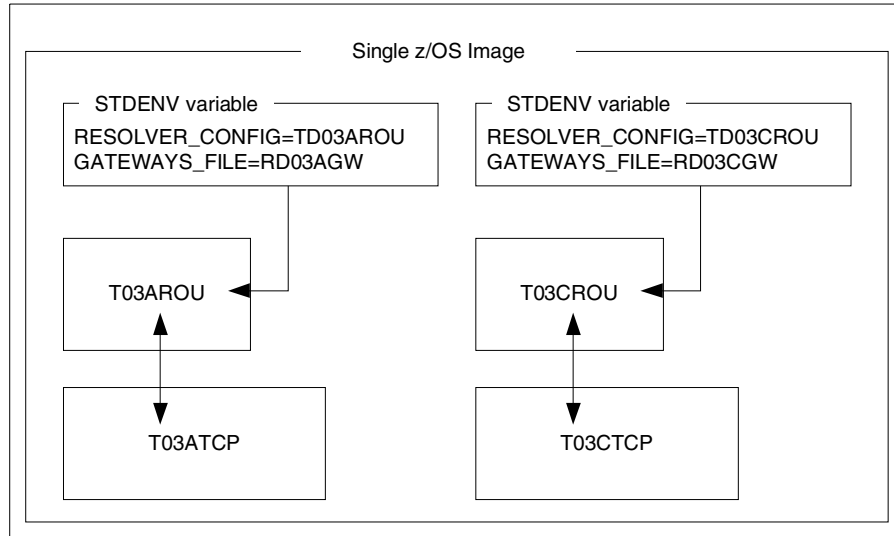


Figure A-3 Multiple stack configuration

Figure A-4 shows the catalogued procedure for our ORouted server running on the second TCP/IP stack. We named it T03CROU.

```

//T03CROU EXEC PGM=BPXBATCH,REGION=4096K,TIME=NOLIMIT,
//*      PARM='PGM /usr/lpp/tcpip/sbin/ORouted'
//      PARM='PGM /usr/lpp/tcpip/sbin/ORouted -ep -hv'
//*
//STDOUT DD PATH='/tmp/ORouted.03c.stdout',
//      PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//      PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//STDERR DD PATH='/tmp/ORouted.03c.stderr',
//      PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//      PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//*
//STDENV DD DSN=TCP.TCPPARMS(RD03CENV),DISP=SHR

```

Figure A-4 Second stack's cataloged procedure

**1** To start one or more ORouted servers in a multiple stack environment, you have to create a unique environment configuration file for each ORouted server, then specify a unique resolver configuration file. Moreover, in each environment configuration file, you might need to create a unique gateways file and ORouted profile for each ORouted instance. Or, you might prefer to assign STDOUT and STDERR to different files.

From the UNIX System Services shell, you can also start one or more copies of an ORouted server. In this case, you would use the export command in order to set each environment variable as follows:

```

# export RESOLVER_CONFIG=/etc/resolv.conf.3a      (set T03ATCP resolver file)
# export ROUTED_PROFILE=/'TCP.TCPPARMS(PRD03APR)' (set T03ATCP gateways file)
# export GATEWAYS_FILE=/'TCP.TCPPARMS(PRD03AGW)' (set T03ATCP routed profile)
# _BPX_JOBNAME='T03AROU' ORouted &              (start ORouted for T03ATCP)

```

```
# export RESOLVER_CONFIG=/etc/resolv.conf.3c      (set T03CTCP resolver file)
# export ROUTED_PROFILE=/'TCP.TCPPARMS(PRD03CPR)'  
(set T03CTCP gateways file)
# export GATEWAYS_FILE=/'TCP.TCPPARMS(PRD03CGW)'  
(set T03CTCP routed profile)
# _BPX_JOBNAME='T03CROU' ORoutedD &           (start ORoutedD for T03CTCP)
```

You can verify to which TCP/IP instance a particular ORoutedD server has connected by using the **NETSTAT** command and/or the **DISPLAY MVS** command. Figure A-5 illustrates the output from the **onetstat** command in our test system.

```
# onetstat -p t03atcp -c
MVS TCP/IP onetstat CS/390 V2R5      TCPIP Name: T03ATCP      11:54:01
User Id  Conn  Local Socket      Foreign Socket      State
-----  ---  -
T03ATCP 00012 0.0.0.0..23      0.0.0.0..0         Listen
T03ATCP 0000F 127.0.0.1..1026  127.0.0.1..1027   Establish
T03ATCP 0000E 127.0.0.1..1027  127.0.0.1..1026   Establish
T03ATCP 0000A 0.0.0.0..1026    0.0.0.0..0         Listen
T03ATCP 000AC 9.24.105.126..23  9.24.104.213..1528 Establish
T03AROU 00191 0.0.0.0..520      *..*                UDP
T03ATCP 00193 0.0.0.0..1155    *..*                UDP
T03SYSL1 0014A 0.0.0.0..514     *..*                UDP

# onetstat -p t03ctcp -c
MVS TCP/IP onetstat CS/390 V2R5      TCPIP Name: T03CTCP      11:54:08
User Id  Conn  Local Socket      Foreign Socket      State
-----  ---  -
T03CTCP 0000F 127.0.0.1..1026  127.0.0.1..1027   Establish
T03CTCP 0000E 127.0.0.1..1027  127.0.0.1..1026   Establish
T03CTCP 0000A 0.0.0.0..1026    0.0.0.0..0         Listen
T03CROU 000DB 0.0.0.0..520      *..*                UDP
T03SYSL1 000B9 0.0.0.0..514     *..*                UDP
```

Figure A-5 onetstat command output

## Controlling the ORoutedD server

The **MODIFY** command allows you to change ORoutedD options or re-read the GATEWAYS file without recycling ORoutedD. If you need to add, for example, a passive route, you can add it to your GATEWAYS file and then request ORoutedD to re-read the data set with the new information.

Using the **TABLES** option, you can display the ORoutedD routing tables. An example of the RIP table is shown in Figure A-6.

```

F T03CROU, TABLES

EZZ4867I Displaying internal routing table:
-----
Destination      Router      Metric Subnetmask  Interface  Timer Flags      States
-----
192.168.236.1    192.168.236.3  1 255.255.255.255  RAS        0  UP|HOST      PASSIVE|INTERFACE
192.168.251.1    192.168.252.1  2 255.255.255.255  T03ATCP    15 UP|GATEWAY|HOST
192.168.252.1    192.168.252.2  1 255.255.255.255  T03ATCP    15 UP|HOST      INTERFACE
192.168.236.2    192.168.236.3  1 255.255.255.255  RAS        30 UP|HOST      INTERFACE
192.168.251.2    192.168.251.2  1 255.255.255.255  LNKVIPA1   0  UP|HOST      INTERFACE|INTERNAL|VIRTUAL
192.168.100.0    192.168.100.98 1 255.255.255.0    TR1        0  UP          INTERFACE
192.168.230.0    192.168.252.1  2 255.255.255.0    T03ATCP    15 UP|GATEWAY
9.24.105.0       9.24.105.76   1 255.255.255.0    EN1        0  UP          INTERFACE|SUBNET
9.0.0.0          9.24.105.76   1 255.0.0.0        EN1        0  UP          INTERFACE|INTERNAL
9.24.104.0       9.24.105.126  2 *255.255.255.0  EN2        15 UP|GATEWAY  SUBNET
192.168.202.0    192.168.252.1  3 255.255.255.0    T03ATCP    15 UP|GATEWAY
192.168.236.0    192.168.236.3  1 255.255.255.0    RAS        0  UP          INTERFACE|INTERNAL
192.168.237.0    192.168.252.1  3 255.255.255.0    T03ATCP    15 UP|GATEWAY
192.168.251.0    192.168.251.2  1 255.255.255.0    LNKVIPA1   0  UP          INTERFACE|INTERNAL|VIRTUAL
192.168.252.0    192.168.252.2  1 255.255.255.0    T03ATCP    0  UP          INTERFACE|INTERNAL
192.168.253.0    192.168.252.1  3 255.255.255.0    T03ATCP    15 UP|GATEWAY
192.168.254.0    192.168.252.1  3 255.255.255.0    T03ATCP    15 UP|GATEWAY
-----

EZZ4867I Displaying internal interface table:
-----
Interface      Address      Metric Subnetmask  Destination  Flags      States
-----
LNKVIPA1       192.168.251.2  0 255.255.255.0    0.0.0.0     UP|VIRTUAL  INTERFACE
EN1            9.24.105.76   1 255.255.255.0    9.24.105.255 UP|BROADCAST SUBNET|INTERFACE|PRIMARY
EN2            9.24.105.73   0 255.255.255.0    9.24.105.255 UP|BROADCAST SUBNET|INTERFACE
T03ATCP        192.168.252.2  2 255.255.255.0    192.168.252.1 UP|POINTOPOINT INTERFACE|PRIMARY
RAS            192.168.236.3  0 255.255.255.0    192.168.236.2 UP|POINTOPOINT INTERFACE|PRIMARY
TR1            192.168.100.98 3 255.255.255.0    192.168.100.255 UP|BROADCAST INTERFACE|PRIMARY
EN1V           1.1.1.1       0 255.0.0.0        0.0.0.0     UP|BROADCAST INTERFACE|PRIMARY
-----

Interface      Options
-----
LNKVIPA1       OUTRIP1|INRIP1|INRIP2
EN1            OUTRIP1|INRIP1|INRIP2
EN2            OUTRIP1|INRIP1|INRIP2
T03ATCP        OUTRIP1|INRIP1|INRIP2
RAS            OUTRIP1|INRIP1|INRIP2
TR1            OUTRIP1|INRIP1|INRIP2
EN1V           OUTRIP1|INRIP1|INRIP2
-----

```

Figure A-6 Display of the internal routing table

Since the output from this **MODIFY** command goes to the ORouteD STDOUT file, you have to specify the **-ep** ORouteD startup option to get this information.

**Note:** By default, ORoutedD closes STDIN, STDOUT and STDERR. You cannot open the STDOUT file using the **MODIFY** command.

It contains the ORoutedD routing table, not the IP layer routing table. You may want to execute a **NETSTAT GATE** or **DISPLAY TCPIP** command at the same time, which will give you the IP layer routing table contents. You can then correlate the output between the two to aid in debugging or understanding what is happening.

## Summary of ORoutedD gateways options

The following table summarizes options and filters that can be used to control traffic flow in an ORoutedD environment.

Table 8-1 Options and filters

Option or Filter
options gateway ip_addr block
options gateway ip_addr noreceive
options gateway ip_addr none
options interface.poll.intervall time-value
options interface.scan.intervall time-value
options interface name ip_addr block dest_ipaddr <b>1</b>
options interface name ip_addr forward dest_ipaddr fmask n.n.n.n <b>2</b>
options interface name ip_addr forward.cond dest_ipaddr fmask n.n.n.n
options interface name ip_addr noforward dest_ipaddr fmask n.n.n.n
options interface name ip_addr none <b>3</b>
options interface name ip_addr noreceive dest_ipaddr fmask n.n.n.n
options interface name ip_addr passive
options interface name ip_addr ripon
options interface name ip_addr ripoff
options interface name ip_addr receive dest_ipaddr fmask n.n.n.n <b>4</b>
options interface name ip_addr receive.conf dest_ipaddr fmask n.n.n.n
options interface name ip_addr supply off
options interface name ip_addr supply on
options interface name ip_addr key authentication_key
options interface name ip_addr nokey
options interface name ip_addr supply.control keyword <b>5</b>
options interface name ip_addr receive.control keyword <b>6</b>

**1** The NORECEIVE filter is a replacement for the BLOCK filter.

**2** The FORWARD filters are used to control which routes are advertised over which interfaces.

The FMASK parameter can be used to apply the filter to multiple routes in just a single options statement. The FMASK is *not* a subnet mask; it is merely used to filter out multiple routes in a single options statement, such as:

```
options interface tr1 9.1.1.1 forward 9.2.0.0. fmask 255.255.0.0
```

Any route that matches 9.2.x.x will be forwarded over the tr1 interface and filtered out on all other interfaces.

**3** NONE is used to clear all filters for a given interface.

**4** The RECEIVE filters are used to control which routes are accepted over which interfaces.

5 Choose one of the following keywords:

- ▶ RIP1
- ▶ RIP2B
- ▶ RIP2M
- ▶ RIP2
- ▶ NONE

6 Choose one of the following keywords:

- ▶ RIP1
- ▶ RIP2
- ▶ ANY
- ▶ NONE

Please refer to *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775 and *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776 for instructions on how to specify the various routing filters.

### Maintaining dynamic route entries

When you use an ORouteD server, several TCP/IP routes might be added and maintained by the ORouteD server. These kinds of TCP/IP routes are referred to as RIP routes. In CS for z/OS IP, ORouteD is the only application that can add or delete RIP routes. Therefore, if ORouteD is terminated, RIP routes are not deleted, but kept in the IP routing table forever.

In order to help maintain the integrity of the routing table, at initialization, ORouteD attempts to delete all RIP routes from the stack routing table, and then adds RIP routes based on BSDROUTINGPARMS and the optional GATEWAYS file.

If you want to delete RIP routes in an IP routing table, you can use the **MODIFY** command with appropriate options. Using these options, you can control the entries in the IP routing table without recycling the ORouteD server.

The following options are supported by the **MODIFY** command to maintain IP routing entries:

**-f** flush all indirect routes known by ORouteD from IP routing table.

**-fh** flush all indirect hosts routes known by ORouteD from IP routing table.

When stopping ORouteD, you can delete all dynamic routes from the IP routing table. The **-kdr** option has been added for this purpose. Figure A-7 shows a sample usage of the **-kdr** command.

```

F T28AROU,PARMS=-KDR 1
+EZZ4993I OE ROUTED: TERMINATED
-T28AROU          *OMVSEX      16      323      60      .04      .00      35.7
80148  0      0      0      0      0
-T28AROU ENDED. NAME-          TOTAL TCB CPU TIME=      .0
TOTAL ELAPSED TIME= 35.7
$HASP395 T28AROU ENDED
IEA989I SLIP TRAP ID=X33E MATCHED.  JOBNAME=*UNAVAIL, ASID=0034.

D TCPIP, ,N,ROUTE 2
EZZ2500I NETSTAT CS V2R5 T28ATCP 656
DESTINATION      GATEWAY          FLAGS  REFCNT  INTERFACE
0 OF 0 RECORDS DISPLAYED

```

Figure A-7 Sample of -kdr option

1 Issuing the **MODIFY** command with the **-kdr** option, ORouted will delete all dynamic routes and then end. ORouted server will post a message to the console and to STDERR.

2 In this case, all route entries in the IP routing table were added using dynamic routing, so no entry was left in the routing table after shutting down the ORouted server.

## Configuring ORouted and VIPA

The following configuration examples relate to the sample network defined in Chapter 7, “Routing with VIPA” on page 141.

We added the procedure illustrated in Figure A-8 and parameter files to enable RIP for our testing. The JCL that runs the ORouted server under UNIX System Services follows in Figure A-8.

```

//OROUTEDC  PROC MODULE='BPXBATCH'
//OROUTED  EXEC PGM=&MODULE,REGION=4096K,TIME=NOLIMIT,
//          PARM='PGM /usr/lpp/tcpip/sbin/orouted -h'
//STDOUT   DD PATH='/tmp/orouted.&SYSCLONE.c.stdout',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//STDERR   DD PATH='/tmp/orouted.&SYSCLONE.c.stderr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)
//STDENV   DD DSN=TCPIP.TCPPARMS(RD&SYSCLONE.CENV),DISP=SHR
//CEEDUMP  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=132,BLKSIZE=132)
//SYSERR   DD PATH='/tmp/orouted.&SYSCLONE.c.syserr',
//          PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//          PATHMODE=(SIRUSR,SIWUSR,SIRGRP,SIWGRP)

```

Figure A-8 Another procedure

We can use this start procedure for both TCPIP stacks, RA03 and RA39, since we are using system variables (&SYSCLONE.). The **-h** parameter allows ORouted to handle host routes, which is necessary for point-to-point connections to be broadcast. The **-hv** parameter enables host virtual routing, which enables adjacent routers in the network to receive VIPA host routes. We highly recommend that you assign a separate subnetwork for the VIPA. The same subnetwork can be used for all VIPA addresses in the sysplex environment.

The TCP/IP stack profile statements relevant to our ORouteD discussion are shown in Figure A-9.

```
IPCONFIG
VARSUBNETTING
IGNOREREDIRECT
DATAGRAMFWD
SOURCEVIPA
SYSPLEXROUTING
DYNAMICXCF 172.16.233.3 255.255.255.0 1
;
PORT
520 UDP OROUTEDC
;
AUTOLOG
OROUTEDC
ENDAUTOLOG

VIPADYNAMIC
VIPADefine MOVE IMMED 255.255.255.0 172.16.251.3
VIPABackup 100 172.16.251.39
ENDVIPADYNAMIC

;
BSDROUTINGPARMS FALSE
VIPAO3C DEFAULTSIZE 0 255.255.255.0 0
EN103 1492 1 255.255.255.0 0
M032216B DEFAULTSIZE 0 255.255.255.0 0
ENDBSDROUTINGPARMS
```

Figure A-9 TCPIP profile statements

- 1** VARSUBNETTING makes this stack support variable subnet since ORouteD is going to work with RIP-2.
- 2** The IGNOREREDIRECT parameter makes the stack ignore ICMP redirect packets.
- 3** The DATAGRAMFWD parameter enables the forwarding of datagrams through this stack.
- 4** The SOURCEVIPA parameter forces the TCP/IP stack to insert the VIPA address as the source field in some outbound packets. In this example, coding SOURCEVIPA has no effect since we have no static VIPAs defined.
- 5** The SYSPLEXROUTING parameter specifies that this host is a part of a sysplex environment and will communicate the interface changes to WLM. The following message will confirm that it is enabled: SysplexRouting support is enabled.
- 6** Dynamic XCF support is enabled by this parameter. 172.16.233.3 is the XCF IP address used for home list. The subnet mask and the metric will be used for BSDROUTINGPARMS.
- 7** The PORT statement is to reserve port 520 for ORouteD.
- 8** The AUTOLOG statement starts the ORouteD server when the TCPIP stack starts.
- 9** The VIPADYNAMIC block defines the Dynamic VIPAs used by this stack.

**10** BSDROUTINGPARMS statements are used to define the interface information, MTU value, and subnet mask. BEGINROUTES or GATEWAYS statements are not used with ORouted. If you need to define any static routes, you must code an ORouted gateway file.

The contents of our standard environment file are listed below:

```
RESOLVER_CONFIG=// 'TCPIP.TCPPARMS(TCPD03C) '  
ROUTED_PROFILE=// 'TCP.TCPPARMS(RD03CCFG) '
```

This file is pointed to by the STDENV DD statement in the start procedure. This DD JCL card is part of the BPXBATCH function, and is a mechanism for setting USS environment variables. For details on how to define an environment, see the *z/OS V1R2.0 UNIX System Services User's Guide, SA22-7801*.

The first variable controls where to find the TCPIP.DATA file, or the equivalent of /etc/resolv.conf. You can also set TCPIP.DATA in the ENVAR parameter in your EXEC JCL card.

The profile used by the Routed daemon is shown below:

```
RIP_SUPPLY_CONTROL: RIP2  
RIP_RECEIVE_CONTROL: ANY
```

We enabled RIP Version 2 to supply control and RIP Version 1 and Version 2 receiving packets for our interfaces.

We did not define a static network and subnet route or static host route. If this is needed, you can define it in the ORouted GATEWAY file. In the GATEWAY file, you can also alter the default values for the RIP timers and define filters to the interfaces.

Our procedures and data files were identical for the TCPIPC stacks on RA03 and RA39 hosts. The only changes were the necessary (obvious) stack, file, and address differences.

## Migrating from ORouted to OMPROUTE

The ORouted routing daemon supports the RIP Version 1 protocol and the RIP Version 2 protocol. The OMPROUTE routing daemon supports both RIP versions, and the OSPF protocol. If you currently use ORouted, we recommend you migrate to OMPROUTE, as support for ORouted will be withdrawn in a future release.

To make migration from ORouted to OMPROUTE easier, a new function has been added to ORouted with CS for z/OS V1R2. If ORouted is started with the -c parameter, it takes existing routes, interfaces and filters and creates a file that can be used as an OMPROUTE configuration file.

For more information on migrating from ORouted to OMPROUTE, please refer to *z/OS V1R2.0 CS: IP Migration, GC31-8773* and *z/OS V1R2.0 CS: IP Configuration Reference, SC31-8776*.





## NCPROUTE

With ACF/NCP V7, the following IP support is possible in an IBM 3745:

- ▶ Route IP traffic over Ethernet LAN adapters (Both Ethernet V2 and IEEE 802.3).
- ▶ Route IP traffic over token-ring LAN adapters (TIC Type 1 and 2 only).
- ▶ Route IP traffic over frame relay connections.
- ▶ Dynamic Reconfiguration (DR) capability for IP over frame relay.
- ▶ Route IP traffic natively over a channel connection (CDLC) between NCP and a system running TCP/IP for z/OS.
- ▶ Route IP traffic over SNA sessions between NCP and other NCPs or z/OS systems running TCP/IP for z/OS.
- ▶ Support for both static IP routing and dynamic IP routing (RIP protocol).

When you consider an IBM 3745 as an IP router, you have to look at it as an independent IP host with its own IP address(es) and, although limited in function, its own IP protocol stack. This implies that you must be able to route IP traffic between the IBM 3745 and other IP hosts, to which it may be connected by means of channels, WAN lines or LANs.

ACF/NCP V6 supported only static IP routing. You defined in the NCP deck which routes the IP router could use. If the routes were to be changed, you had to update the NCP deck and regenerate your NCP load module.

In ACF/NCP V7, you are able to use the dynamic routing facility of IP, as it is implemented in the Routing Information Protocol Version 1 (RIP V1). ACF/NCP V7 does not implement a standard routing daemon, but relies on a server program in TCP/IP for MVS, called the NCPROUTE server, for maintaining its routing tables. Figure B-1 shows the relationship between NCPROUTE and the NCP IP function. Note the following:

- ▶ ESCON channel attachment requires the 3746-900 module on the 3745.
- ▶ IP over token-ring is limited to TIC-1 and TIC-2.

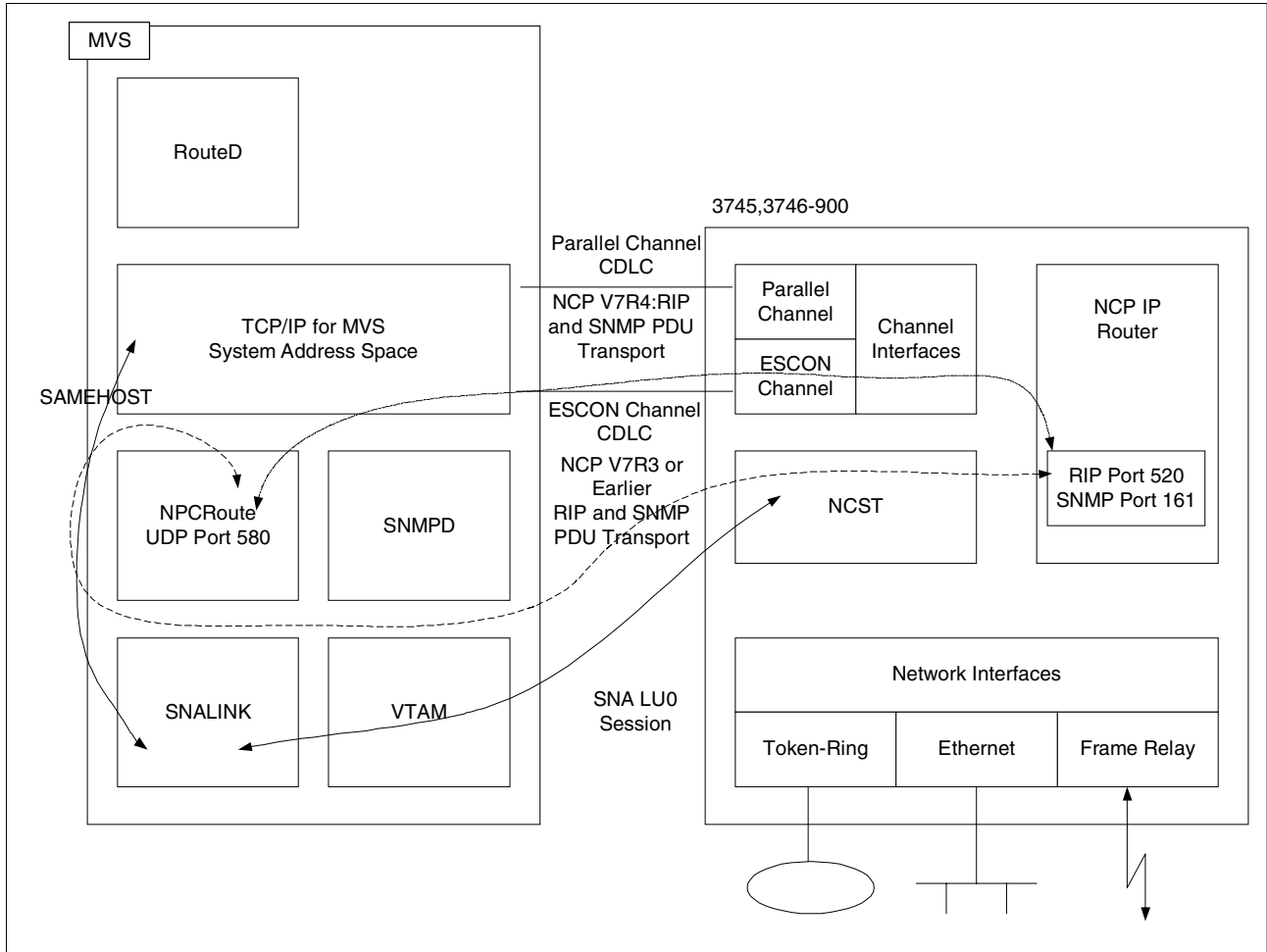


Figure B-1 NCP IP router overview

In the NCP, all RIP packets received for the well-known RouteD port (port 520) are forwarded to port 580 on the z/OS host, which is specified as the IPOWNER of this NCP. On this z/OS system, the NCPROUTE server executes; it connects to port 580. It is the job of the NCPROUTE server to maintain the routing tables on behalf of the NCP and to respond to SNMP routing queries that are sent to the NCP IP router interfaces. NCPROUTE communicates with SNMPD to respond to the queries. The tables are periodically distributed to the NCP, so it can perform its IP routing correctly. The communication between port 520 on the NCP and port 580 at z/OS TCP/IP occurs over a SNALINK connection prior to NCP V7R4. With NCP V7R4 and higher, native IP communication using the CDLC protocol may be used for NCPROUTE traffic.

**Important:** Even if you do not use RIP V1 in your network, you can still use NCPROUTE to maintain the static routes of the NCP IP router. Start NCPROUTE in quiet mode, and add passive routes to the NCPROUTE GATEWAYS data set. If you need to change the NCP IP router static routes, just update the NCPROUTE GATEWAYS data set and request NCPROUTE to re-read it. The alternative would be to regenerate the NCP to change the static route definitions using IPRROUTE statements.

## Configuring NCPROUTE

If you want the NCP IP router to take part in dynamic IP routing, then you configure and start NCPROUTE in z/OS. The following instructions apply also to NCPROUTE connectivity through CDLC channel connections.

To set up NCPROUTE, the following definitions are needed:

1. Define NCPROUTE in PROFILE.TCPIP with a port statement and an AUTOLOG statement:

*Example: B-1 Defining NCPROUTE*

---

```
AUTOLOG
  T18ANCPR                ; NCP Route server
  . . .

PORT
  580 UDP T18ANCPR        ; NCPROUTE for RIP support on NCPs
  . . .
```

---

2. Ensure that ETC.SERVICES has an entry for NCPROUTE:

*Example: B-2 ETC.SERVICES NCPROUTE entry*

---

```
#
#      ITS0 added services
#
ncproute      580/udp          # NCP Route server
```

---

3. Create a JCL procedure for the NCPROUTE started task, which in our sample setup is called T18NNCPR:

*Example: B-3 Creating a JCL procedure*

---

```
//T18NNCPR PROC MODULE=NCPROUTE,PARMS='/'
//*
//* NCPROUTE for T18N stack on z/OS18
//*
//NCPROUT EXEC PGM=&MODULE,
//          PARM='&PARMS',
//          REGION=4096K,TIME=1440
//STEPLIB DD DSN=TCPIP.SEZATCP,DISP=SHR
//          DD DSN=TCPIP.SEZALINK,DISP=SHR
//          DD DSN=ITSC.NCPLOAD,DISP=SHR
//          DD DSN=RISC.NCPLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSERR DD SYSOUT=*
//NCPRPROF DD DSN=TCPIP.T18N.TCPPARMS(NCPROUTE),DISP=SHR
//SYSTCPD DD DSN=TCPIP.T18N.TCPPARMS(TCPDATA),DISP=SHR
//MSNCPROU DD DSN=TCPIP.SEZAINST(EZBNRMSG),DISP=SHR,FREE=CLOSE
```

---

Remember to place your NCP load library into the concatenation sequence of STEPLIB. When you generate an NCP with IP router support, an extra load module will be generated into your NCP load library. It is called the routing information table (RIT). It contains the NCP IP definitions that NCPROUTE needs to know and its name is your NCP name suffixed with a P. In our setup, the NCP name is RA7NCPX. The load module that NCPROUTE loads is RA7NCPXP.

You do not need to define the NCP name to NCPROUTE. When an NCP IP router client connects to NCPROUTE, it sends a message with information about the NCP name, which NCPROUTE uses to determine the name of a corresponding RIT table.

You can connect multiple NCPs running IP routing to one CS for z/OS IP running NCPROUTE. All NCPs must have a point-to-point link to CS for z/OS IP, either a SNALINK or a CDLC link.

You can run multiple copies of NCPROUTE in one CS for z/OS IP installation (for example, for test purposes). Each NCPROUTE server should then listen on a separate port (for example, 580 and 581). You should reserve a port in PROFILE.TCPIP and update ETC.SERVICES accordingly, such as:

*Example: B-4 Updating ETC.SERVICES*

---

```
#
#      ITS0 added services
#
T18ANCPR      580/udp   ncproute     # NCP Route server production
T18ANCPT      581/udp   ncproute     # NCP Route server test
```

---

Note that when you run NCPROUTE on another port, you should specify this on the IPOWNER macro of the NCP deck (UDPPORT=581).

4. Create an NCPROUTE profile data set that defines with which SNMP agent NCPROUTE should open a DPI connection (see *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228 for more details about the SNMP agent distributed program interface).
5. The SNMP DPI connection provides support for network management route table queries to the NCPROUTE dynamic router.

```
SNMP_AGENT: 9.24.104.74
SNMP_COMMUNITY: public
GATEWAY_PDS: TCPIP.T18N.NCPROUTE.GATEWAYS
```

*Figure B-2 NCPROUTE profile*

The data set must be allocated to a DD statement: NCPRPROF. We placed the configuration statements in TCPIP.T18N.TCPPARMS(NCPROUTE).

Do not forget the colons. If you enter SNMP\_AGENT only, NCPROUTE will complain about an invalid keyword.

NCPROUTE uses the contents of the gateways data set for the same purpose as Routed uses the ETC.GATEWAYS data set. Interface metric customization is specified in a metric options statement because there is no BSDROUTINGPARMS section for the NCP IP router.

If you want to add additional static routes or remove static routes without stopping and restarting the NCP(s), you create a member in the gateways data set for each NCP that uses this NCPRROUTE server. The member names equal the name of the corresponding RIT members. If we wanted to add a static route for RA7NCPX, we would place the definitions for that static route in member RA7NCPXP in TCPIP.T18N.NCPRROUTE.GATEWAYS. The definitions are read at NCPRROUTE initialization or if you issue the z/OS command:

```
MODIFY NCPRROUTE,GATEWAYS,C=RA7NCPXP
```

We defined a member for RA7NCPXP just to set some default options and enable tracing of packets exchanged with the NCP.

```
# NCPRROUTE gateways and options for RA7NCPX
#
options default.router no supply on trace.level 1
#
# net 0.0.0.0 gateway 10.0.30.18 metric 1 passive
```

Figure B-3 NCPRROUTE gateways data set member

The following is an example of the messages we receive from NCPRROUTE when it is started and a connection with the NCP is established:

Example: B-5 Message from NCPRROUTE

```
Port 580 assigned to ncprout
Input parameter(s): -t
Global tracing actions enabled
*****
* Opening NCPRROUTE profile dataset (DD:NCPRPROF)
*****
** Attempting to (re)start SNMP connection
Connecting to agent 9.24.104.74 on DPI port 1024
SNMP DPI connection established
1.3.6.1.4.1.2.6.17. registered with SNMP agent
NCPRROUTE Server started
Waiting for incoming packets
*****
* Hello from new client 10.0.28.7 1
* RIT dataset name: RA7NCPXP
* RIT ID: 10/17/96 07:55:02
*****
Acknowledge to 10.0.28.7: Hello Received
Establishing session with client 10.0.28.7
Acknowledge to 10.0.28.7: RIT Loaded OK
Session with client 10.0.28.7 started
Waiting for incoming packets
*****
* Recv: Inactive Interface List from 10.0.28.7 3
* 6 interface(s) found:
* 10.0.2.7 - L071070
* 10.0.25.7 - P7280100
* 10.0.20.7 - P7280110
* 10.0.23.7 - P7A36100
* 10.0.30.7 - P07C01
* 10.0.31.7 - P0710767
*****
```

```

*****
* Processing interface L071070
*****
This interface is not point-to-point
Interface L071070 not up
*****
* Processing interface L071093 2
*****
This interface is not point-to-point
Adding network route for interface
Tue Dec 10 11:19:38:
ADD destination 10.0.0.0, router 10.0.32.7, metric 1
flags UP state INTERFACE|CHANGED|INTERNAL|SUBNET timer 0
Adding subnetwork route for interface
ADD destination 10.0.32.0, router 10.0.32.7, metric 1
flags UP state INTERFACE|CHANGED|SUBNET|PERM timer 0
*****
* Processing interface P7280100
*****
Point-to-point interface, using dstaddr
Interface P7280100 not up
*****
* Processing interface P7280110
*****
Point-to-point interface, using dstaddr
Interface P7280110 not up
*****
* Processing interface P7A36100
*****
Point-to-point interface, using dstaddr
Interface P7A36100 not up
*****
* Processing interface RA7TNCS1 4
*****
Point-to-point interface, using dstaddr
Adding subnetwork route for interface
ADD destination 10.0.28.0, router 10.0.28.7, metric 1
flags UP state INTERFACE|CHANGED|SUBNET|PERM timer 0
Adding host route for interface
ADD destination 10.0.28.18, router 10.0.28.7, metric 1
flags UP|HOST state INTERFACE|CHANGED|SUBNET|PERM timer 0
*****
* Processing interface P07C01
*****
Point-to-point interface, using dstaddr
Interface P07C01 not up
*****
* Processing interface P0710767
*****
Point-to-point interface, using dstaddr
Interface P0710767 not up
destination is 10.0.21.0, gateway is 10.0.20.9
Route 0: 10.0.21.0 via 10.0.20.9 metric 2
ADD destination 10.0.21.0, router 10.0.20.9, metric 2
flags UP|GATEWAY state CHANGED|SUBNET|PERM timer 0
*****
* Opening GATEWAYS dataset for client 10.0.28.7
* 'TCPIP.T18N.NCPRUTE.GATEWAYS(RA7NCPXP)' 5
*****
Start of GATEWAYS processing:

```

```

Option(s): default.router no supply on trace.level 1
(no etc.gateway definitions)
End of GATEWAYS processing
Waiting for incoming packets

```

---

1 NCP client connects to NCROUTE.

Token-ring (2) and NCST interface (4) definitions are processed.

3 Inactive interfaces are reported to NCROUTE.

5 The gateways data set definitions are read and interpreted.

## NCP definitions

If you want to use the IP router functions of an NCP, you may need to add the following definitions to your NCP source deck. For SNALINK and CDLC definitions, please consult Chapter 3, "Network interface types" on page 17.

## Token-ring adapter

The token-ring adapter can be used for both SNA and IP traffic. In this context, we focus on the IP side of the story.

```

*-----
* NTRI PHYSICAL DEFINITIONS
*-----
RA7GT93P GROUP ECLTYPE=(PHYSICAL,ANY),      TOKEN RING PHYSICAL GROUP *
                CAPACITY=1M,                *
                ADAPTER=TIC2                  TICS ADAPTER CONNECT TYPE
*
L071093 LINE   STATOPT='NTRI TIC2'
                ADDRESS=(1093,FULL),          LINE NUMBER ADDRESS, MODE *
                PORTADD=1,                    PHYSICAL LINE PORT NUMBER *
                LOCADD=400002070000,          LOCAL ADMINED TIC ADDRESS *
                INTFACE=L071093, 3           INTERFACE NAME OF ADDRESS *
                MAXFRAME=2106,                EQUAL TO MAXTSL *
                DATABLK=2048,                 TRANS-UNIT FOR COMRATE *
                RCVBUFC=8192,                 MAX # OF DATA RECV IN ONE XFER *
                MAXTSL=4060,                  MAX VALUE FOR TC12/4MB *
                XMONLNK=YES,                  SMMF CONTROL *
                TRSPEED=4,                    SPEED *
                ANS=CONTINUE
*
P071093S PU    NETWORK=SNA, 2                SNA, IP OR FRELAY SUPPORT *
                COMRATE=(,64)
*
P071093I PU    NETWORK=IP, 1                 SNA, IP OR FRELAY SUPPORT *
                COMRATE=(,8)

```

Figure B-4 ACF/NCP token-ring definitions

The following explanations refer to Figure B-4.

For each TIC you define a LINE and a PU with the NETWORK=IP parameter **1**. If you want to use the same TIC for SNA traffic, you would define another PU on the same line; but with NETWORK=SNA **2** instead of IP.

INTERFACE=L071093 **3** defines the unique interface name required for each token-ring line and is used to associate IP interface definitions with this line.

## Ethernet adapter

```

*-----
*          ETHERNET ADAPTER DEFINITION          *
*-----
RA7GE70  GROUP  ETHERNET=PHYSICAL,              ETHERNET PHYSICAL DEFINED  *
              LANTYPE=DYNAMIC 2
*
L071070  LINE   ADDRESS=(1070,FULL),           LINE NUMBER ADDRESS, MODE  *
              INTERFACE=(L071070,1492), 3    INTERFACE NAME / MTU SIZE  *
              FRAMECNT=(100000,5000)        ALERT STATISTIC THRESHOLD
*
P071070  PU     PUTYPE=1, 1                   I.P. OVER ENET ON 1070 PU  *
              LANTYPE=802.3,                 FRAME FORMAT SUPPORT TYPE  *
              ARPTAB=(50,20),                ARP ENTRY, TIMER & FORMAT  *
              ANS=CONTINUE                    NCP AUTO NETWORK SHUTDOWN

```

Figure B-5 ACF/NCP Ethernet definitions

The following explanations refer to Figure B-5.

Each Ethernet port will be defined in the NCP as an SNA line and a type 1 PU **1**. LANTYPE=DYNAMIC **2** specifies that this Ethernet interface supports both the Ethernet V2 (DIX) and the IEEE 802.3 DLCs. INTERFACE=L071070 **3** defines the unique interface name required for each Ethernet line and is used to associate IP interface definitions with this line. In this example, we also define the MTU size for this interface as 1492 bytes.

## Frame relay connection

If you are familiar with frame relay definitions in the NCP, you know that you can define Frame Relay Terminating Equipment (FRTE) connections for the equipment serviced by the frame relay network and Frame Relay Frame Handler (FRFH) definitions plus FRSESETs if your NCP is to be a frame relay switch itself. You also define Local Management Interface (LMI) definitions if you are keeping statistics on your frame relay connections.

The addresses for the circuit endpoints in a frame relay network are called Data Link Control Identifiers (DLCIs) and these can be specified in the NCP or in a VTAM Switched Net Major Node (SMN) for SNA connections. Normally you have physical line groups in which you define the FRFH DLCIs and the LMI DLCIs; then you have logical line groups in which you define the FRTE DLCIs for SNA subarea connections or in which you leave placeholders for DLCIs you may have coded in the VTAM SMN.



```

*****
*           FRAME RELAY DEFINITIONS           *
*****
RA7G280P GROUP FRELAY=(PHYSICAL,SUB)          FRAME RELAY PHYSICAL CODE
*
L07280  LINE  ADDRESS=(280,FULL), ②          LINE ADDRESS NUMBER, MODE *
          PORTADD=80,                    PHYSICAL LINE PORT NUMBER *
          MAXFRAME=2052,                  *
          ARPTAB=(4,,NOTCANON),          ARP ENTRY, TIMER & FORMAT *
          XMONLNK=YES,                    *
          SPEED=56000                     *
*
P07280  PU    LMI=ANSI                    LMI PROTOCOL/ ECHO DETECT
*
P728090 PU    DLCI=90,                    FRFH PORT *
          COMRATE=(,32)
*
P7280100 PU   DLCI=100, ①                I.P TESTING FRMR TO SA 06 *
          INTFACE=P7280100,              INTERFACE NAME OF ADDRESS *
          LADDR=10.0.25.7,                *
          SNETMASK=255.255.255.0,        NETWORK CLASS / HOST MAPS *
          METRIC=1,                       INTERFACE METRIC = 1 *
          PROTOCOL=RIP,                   MANAGED BY NCPROUTE *
          P2PDEST=10.0.25.6,             IP ADDRESS OF REMOTE END *
          COMRATE=(,16)

```

Figure B-6 ACF/NCP frame relay definitions

With IP over frame relay, you find yourself dealing only with DLCIs coded on ports or IP interfaces within the physical line group. You do *not* code logical lines for IP connections. Notice in Figure B-6 how P7280100 (①), an IP interface with IP address 10.0.25.7, has been assigned a DLCI number of 100 in the physical line group called L07280 (②). The IP interface, like the LMI interface, is treated as a PUTYPE of 1 and can be activated or inactivated by VTAM.

As is the case with other types of IP definitions in the NCP, you may code the IP interface in one of two fashions:

1. Specify the IP keywords directly on the PU statement
2. Specify them in an IPLOCAL statement

**Note:** For a complete treatment of coding IP over frame relay, refer to the appropriate NCP documentation.

## NCST logical unit

```
*****
*      NCST LU DEFINITIONS                               *
*****
RA7GNCS1 GROUP NCST=IP                                NCST IP SESSION FOR SA 25
*
RA7LNCS1 LINE                                         DUMMY RESOURCE IS DEFINED
*
RA7PNCS1 PU                                           DUMMY RESOURCE IS DEFINED
*
RA7TNCS1 LU      INTFACE=(RA7TNCS1,2048), 1          INTERFACE NAME / MTU SIZE *
                  REMLU=(RAIASN18) 2              SNALINK/NCST PARTNER NAME
```

Figure B-7 ACF/NCP NCST LU definitions

The NCST GROUP defines the connectionless transport LUs and their sessions' characteristics. INTFACE=RA7TNCS1 1 defines the name used to associate IP interface definitions with the session that will transport the IP traffic through the SNA network. REMLU=RAIASN18 2 defines the SNALINK partner LU name in MVS18 to be used for the NCST-SNALINK session.

## NCPROUTE server host

```
RA7IHOST IPOWNER HOSTADDR=10.0.28.18, 1          IP HOST, NCPROUTE & FR DR *
          NUMDRIF=10,                             FR INTERFACES FOR DR ADDS *
          INTFACE=RA7TNCS1, 2                     INTERFACE NAME OF ADDRESS *
          MAXHELLO=6,                             NCPROUTE CONTACT ATTEMPTS *
          NUMROUTE=(25,25,25),                   ADDITIONAL NCPROUTE ENTRY *
          UDPPORT=580                             USER DATAGRAM PORT NUMBER
```

Figure 8-7 ACF/NCP IPOWNER definitions

In the IPOWNER statement, you define the IP address 1 and the IP interface 2 to use for connection to the z/OS that runs the NCPROUTE server.

**Note:** Prior to NCP V7R5, the IPOWNER statement is required even if you are not using NCPROUTE function. It usually points to the SNALINK IP address. However, if you are using a CDLC with no NCST-SNALINK session, the IPOWNER points to the z/OS TCP/IP address of the CDLC connection.

Observe the following rules when you are coding IPOWNER:

- ▶ IPOWNER must include an INTFACE keyword and a HOSTADDR IP address.
- ▶ The HOSTADDR that is specified in IPOWNER must be the endpoint IP address of the interface that is specified in the INTFACE keyword.
- ▶ The HOSTADDR IP address may not be a VIPA address.

## IP home addresses and optional IP routing table entries

```

*****
*      IP_ADDRESS      INTERFACE      DESCRIPTION      *
*      -----      -
*      10.0.28.7      RA7TNCS1      SNALINK TO MVS18      *
*      10.0.32.7      L071093      TOKENRING INTERFACE      *
*      10.0.2.7      L071070      ETHERNET INTERFACE      *
*      10.0.25.7      P7280100      ROUTE SA#6      *
*****
RA7INCST IPLOCAL INTFACE=RA7TNCS1,      NCST CONNECTION INTERFACE      *
          LADDR=10.0.28.7,      THIS INTERFACE IP ADDRESS      *
          SNETMASK=255.255.255.0,      NETWORK CLASS / HOST MAPS      *
          P2PDEST=10.0.28.18,      DESTINATION INTERNET HOST      *
          METRIC=1,      NCPROUTE EVALUATE VIA RIP      *
          PROTOCOL=RIP      NCPROUTE AND RIP TRANSMIT      *
*
RA7ILE70 IPLOCAL INTFACE=L071070,      I.P. OVER ENET ON 1070 PU      *
          LADDR=10.0.2.7,      THIS INTERFACE IP ADDRESS      *
          SNETMASK=255.255.255.0,      NETWORK CLASS / HOST MAPS      *
          METRIC=1,      NCPROUTE EVALUATE VIA RIP      *
          PROTOCOL=RIP      NCPROUTE AND RIP TRANSMIT      *
*
RA7IT93P IPLOCAL INTFACE=L071093,      I.P. OVER T.R. ON 1088 PU      *
          LADDR=10.0.32.7,      THIS INTERFACE IP ADDRESS      *
          SNETMASK=255.255.255.0,      NETWORK CLASS / HOST MAPS      *
          METRIC=1,      NCPROUTE EVALUATE VIA RIP      *
          PROTOCOL=RIP      NCPROUTE AND RIP TRANSMIT      *
*
RA7I1109 IPRROUTE DESTADDR=10.0.21.0,      ROUTE DESTINATION ADDRESS      *
          DISP=PERM,      PERMANENT OR NCPROUTE RIP      *
          HOSTRT=NO,      IP HOST / NETWORK ADDRESS      *
          INTFACE=P7280110,      CONNECTION INTERFACE NAME      *
          METRIC=2,      NCPROUTE EVALUATE VIA RIP      *
          NEXTADDR=10.0.20.9      NEXT INTERNET DESTINATION

```

Figure B-8 ACF/NCP IP addresses

IP home addresses are defined in the IPLOCAL statements and the optional static routing table entries in IPRROUTE statements. If you use dynamic IP routing, you do not need to specify any IPRROUTE statements.

The meaning of the keywords on the IPLOCAL and IPRROUTE statements may easily be related to the HOME and GATEWAY statements respectively of a PROFILE.TCPIP data set.

If you use static routing, you have to add IPRROUTE statements with static routing definitions. NEXTADDR=0 is used for direct routing. For indirect routing, the NEXTADDR parameter specifies the IP address of the next router on the path towards the destination network.

## Configuration summary

Figure B-9 summarizes the configuration process and illustrates which names must match in these configurations.

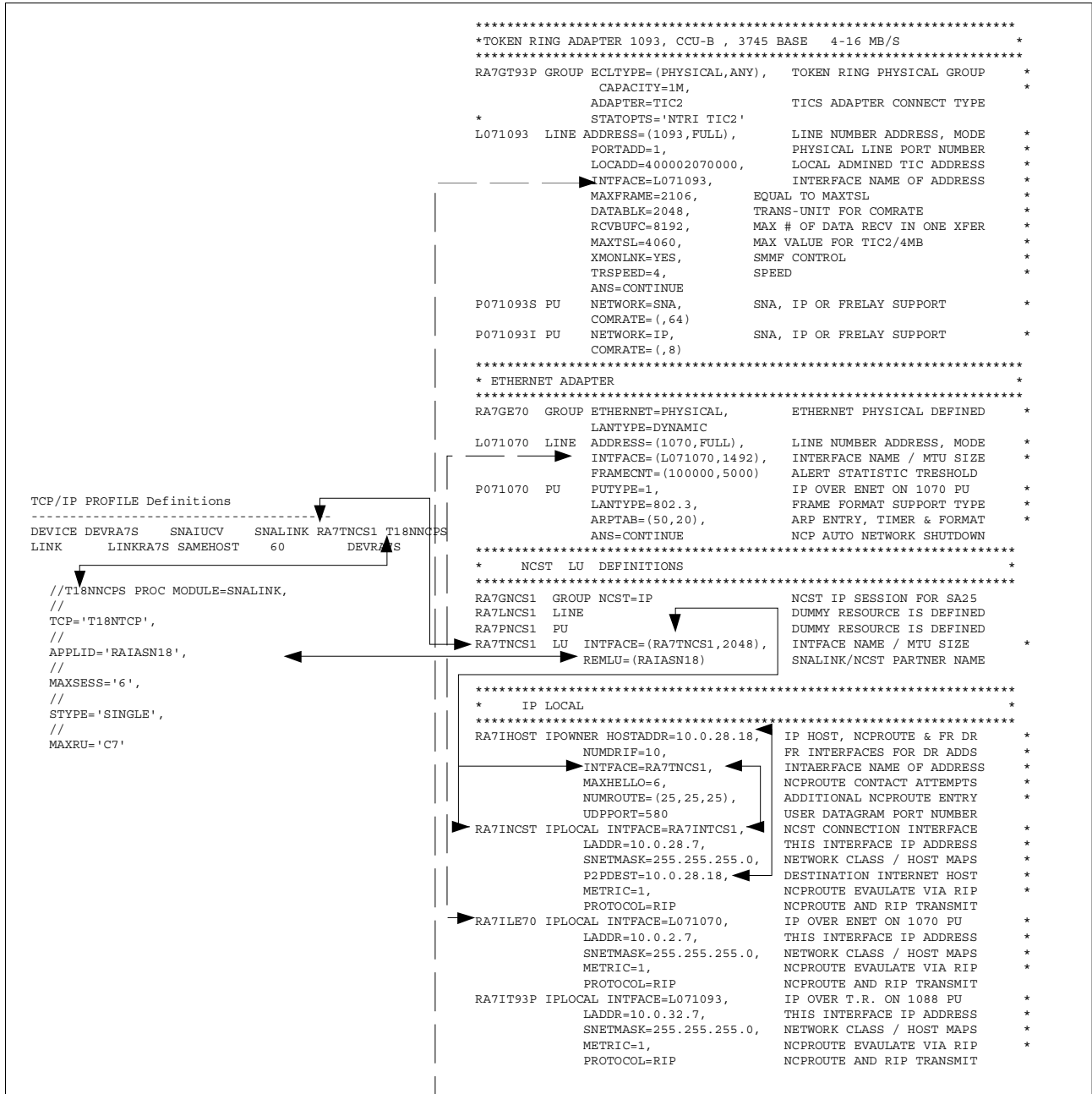


Figure B-9 ACF/NCP IP router definitions overview

## Controlling the use of SNALINK

If you have implemented an NCPROUTE server with an NCP V7R1, V7R2, or V7R3 client, the routing PDUs must flow over an SNALINK connection. With NCP V7R1 and NCP V7R2, only one IP connection exists between TCP/IP for MVS and NCP: the SNALINK-NCST session. However, with NCP V7R3 you may also have implemented a second IP connection between CS for z/OS IP and the NCP: a CDLC connection. In such a configuration you need to control the use of the two parallel connections, SNALINK and CDLC.

SNALINK should be used only for RIP and SNMP PDU transport and not for normal IP routing. To accomplish this separation of function between the CDLC and the SNALINK connections, you must use interface metric customization, making the CDLC link the preferred link, but ensuring that SNALINK is still available as a backup link should the CDLC link fail. Figure B-10 depicts such a scenario.

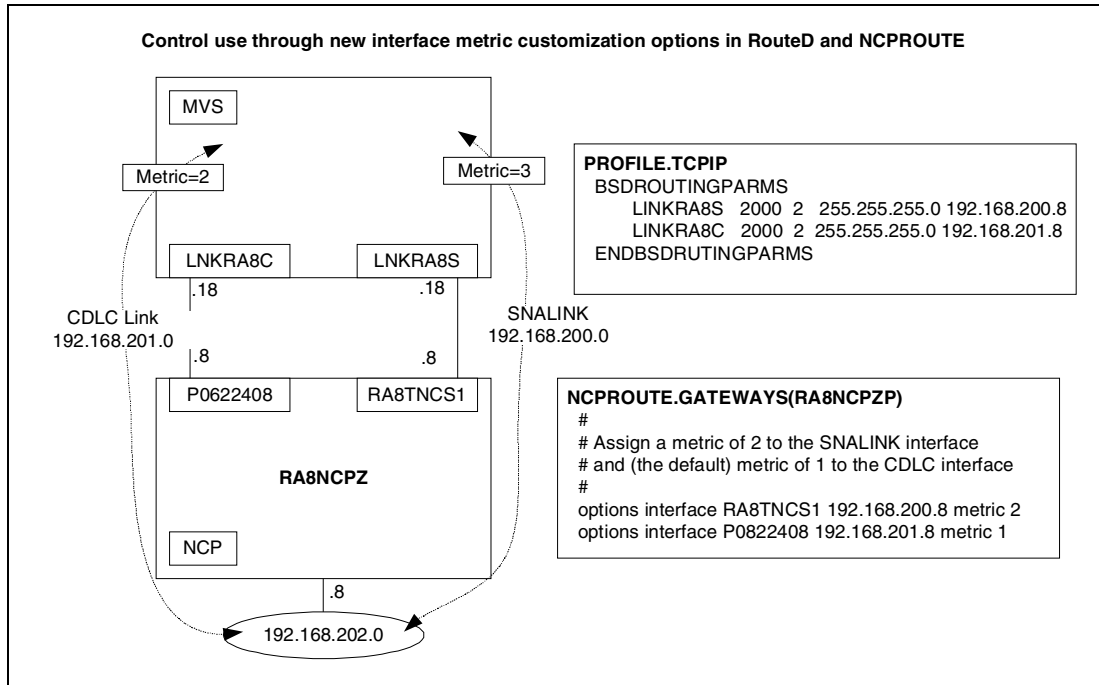


Figure B-10 Controlling the use of SNALINK

You see that we have used interface metric customization in both RouteD on z/OS (the metrics in the BSDROUTINGPARMS section of PROFILE.TCPIP) and in NCPROUTE (the options interface statements in the NCP GATEWAYS member) for the NCP IP router. We used the default metric of 1 for the CDLC link and a higher metric of 2 for the SNALINK connection.

By using metrics in both CS for z/OS IP and in the NCP IP router, both z/OS and the NCP IP router will choose the CDLC link. This will result in the CDLC link being preferred for both inbound traffic to CS for z/OS IP and outbound traffic from CS for z/OS IP.

## CDLC for NCPROUTE

This section shows you the coding necessary to implement the IP configuration at MVS2 and the NCP IP router in Figure B-11. We intend to use the CDLC connection for the transport of PDUs for the NCPROUTE function. Recall that NCP V7R4 eliminated the SNALINK requirement for NCPROUTE.

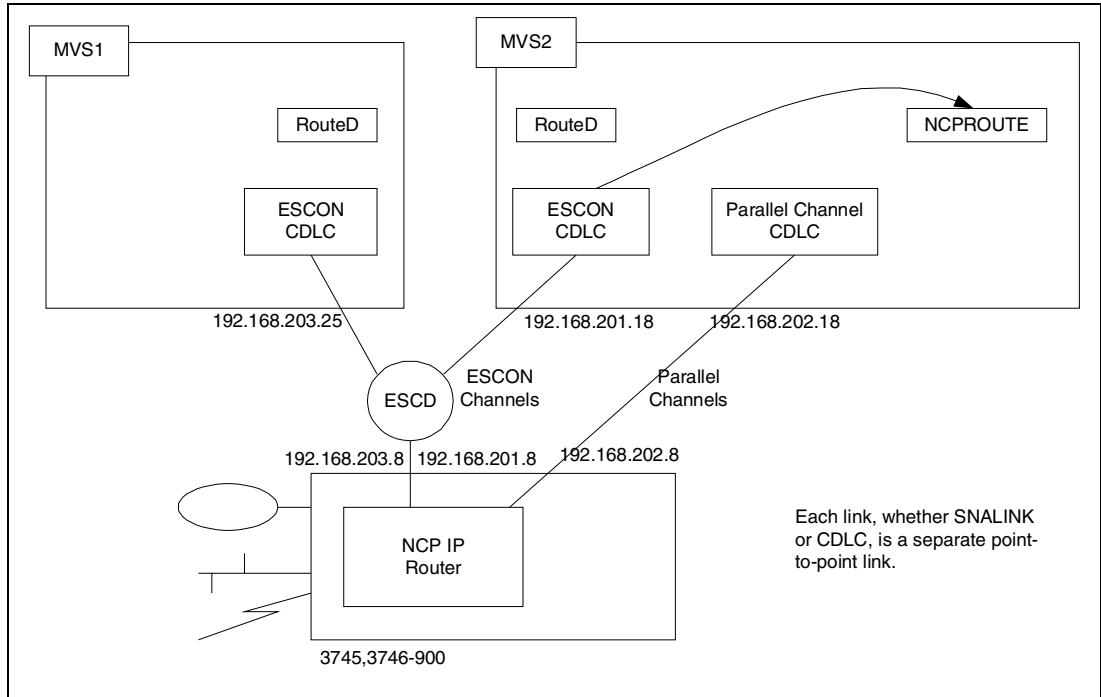


Figure B-11 NCP IP router sample: CDLC for NCPROUTE

Remember that there are two methods available for defining parallel channel and ESCON IP interfaces:

1. Code an IPLOCAL definition statement and its associated keywords.

**Note:** This method is also supported for NCST, frame relay, Ethernet, and token-ring. It is also supported for CDLC communication with NCPROUTE.

2. Define the necessary interface definition keywords on the PU statement (LADDR, P2PDEST, SNETMASK). If you use this method, you may not use the IPLOCAL statement for the interface.

**Note:** This method is not supported for CDLC communication with NCPROUTE.

Our choice for coding CDLC connections is clear: since we want to use dynamic routing in the NCP, we must use the IPLOCAL interface definitions that we introduced in “ICDLC definitions: parallel channel” on page 26 and “CDLC definitions: ESCON channel” on page 27.

```

RA8IHOST IPOWNER HOSTADDR=192.168.201.18, ① SNALINK END-POINT MVS18
      NUMDRIF=2,
      INTFACE=P0822408, ②          ESCON INTERFACE
      MAXHELLO=6,                MAX 6 CONTACT ATTEMPTS
      NUMROUTE=(25,25,25),       ADD. NCPROUTE ENTRIES
      UDPPORT=580                NCPROUTE ON PORT 580
*
*RA8INCST IPLOCAL INTFACE=RA8TNC1,    NCST INTERFACE
* ③          LADDR=192.168.200.8,      LOCAL IP ADDRESS
*          P2PDEST=192.168.200.18,    PTP ENDPOINT IP ADDRESS
*          METRIC=1,                  ROUTE METRIC
*          PROTOCOL=RIP              USE NCPROUTE
*
RA8IESC9 IPLOCAL INTFACE=P0822408,    ESCON INTERFACE
      LADDR=192.168.201.8,            LOCAL IP ADDRESS
      P2PDEST=192.168.201.18,        PTP ENDPOINT IP ADDRESS
      METRIC=1,                      ROUTE METRIC
      PROTOCOL=RIP                  USE NCPROUTE
*
RA8IPC06 IPLOCAL INTFACE=RA8PC06,     PARALLEL INTERFACE
      LADDR=192.168.202.8,           LOCAL IP ADDRESS
      P2PDEST=192.168.202.18,       PTP ENDPOINT IP ADDRESS
      METRIC=1,                     ROUTE METRIC
      PROTOCOL=RIP                  USE NCPROUTE

```

Figure B-12 IPOWNER and IPLOCAL in the NCP (CDLC connection)

Note in Figure B-12 how the IPOWNER definitions in our new NCP coding refers to the CDLC connection (①). Also note that we have identified the ESCON CDLC INTFACE of P0822408 (②). Finally notice how we have commented out the NCST definitions here and in other parts of the NCP source (③).

## NCPROUTE and SNALINK for remote NCPs

If you are using NCP dynamic routing with NCPROUTE, you may not be able to eliminate SNALINK entirely from your network. For example, if you plan to use NCPROUTE communication to a remote NCP that has only a link attachment to a channel-attached NCP, you must include an NCST-SNALINK connection in the remote NCP to carry the NCPROUTE communication. The simple diagram in Figure B-13 shows how you could implement dynamic routing in both a channel-attached NCP and a remote NCP.

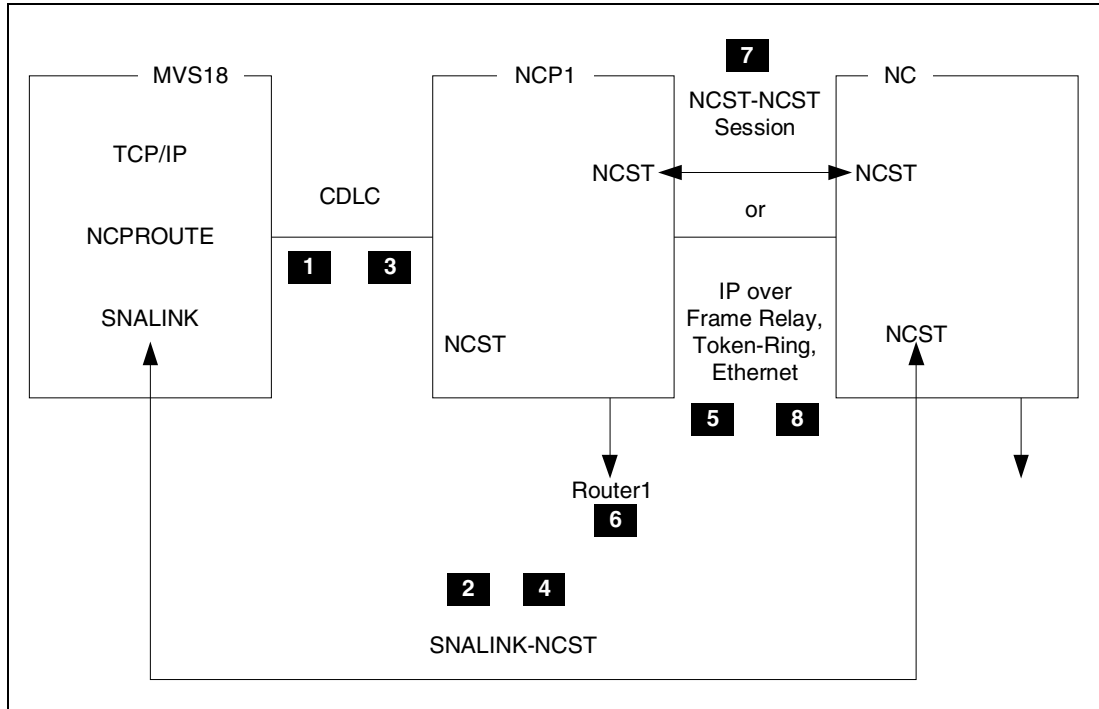


Figure B-13 NCPROUTE and SNALINK for remote NCPs

The following explanations refer to Figure B-13.

- ▶ **1** The communication between the NCPROUTE server at MVS18 and the NCP1 client can flow either over a CDLC IP connection or over an SNALINK-NCST session.
- ▶ **2** The communication between the NCPROUTE server at MVS18 and the NCP2 client can flow *only* over an SNALINK IP connection. Note that we have not implemented a CDLC connection between NCP2 and MVS18; otherwise the communication between NCPROUTE and NCP2 could use a CDLC link.
- ▶ **3** NCP1 is informed of NCP2 routes dynamically via its communication with the NCPROUTE server at MVS18.
- ▶ **4** NCP2 is informed of NCP1 routes dynamically via its communication with the NCPROUTE server at MVS18.
- ▶ **5** Even if broadcast datagrams are received by NCP1 from NCP2 across a NTRI token-ring, Ethernet, or frame relay interface, these datagrams do not directly update NCP1's routing table; the routing table is updated only via the communication with NCPROUTE. The same can be said for broadcast datagrams received by NCP2 from NCP1 across the named interfaces.
- ▶ **6** The only benefit to an IP route between NCP1 and NCP2 is to provide communication between networks attached to NCP1 (Router1) and networks attached to NCP2 (Router2).
  - If the IP connection is to be provided by an NCST-NCST session **7**, a subarea connection must exist between NCP1 and NCP2. (In fact, we need a subarea connection in any case so that VTAM can activate NCP2.)
  - Alternatively, the IP connection may be provided by an Ethernet, token-ring, or frame relay network between NCP1 and NCP2 **8**.



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 212.

- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 1: Base and TN3270 Configuration*, SG24-5227
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 2: UNIX Applications*, SG24-5228
- ▶ *OS/390 eNetwork Communications Server for V2R7 TCP/IP Implementation Guide Volume 3: MVS Applications*, SG24-5229
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 5: Availability, Scalability, and Performance*, SG24-6517
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 6: Policy and Network Management*, SG24-6839
- ▶ *Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 7: Security*, SG24-6840
- ▶ *TCP/IP in a Sysplex*, SG24-5235
- ▶ *Managing OS/390 TCP/IP with SNMP*, SG24-5866
- ▶ *Secure e-business in TCP/IP Networks on OS/390 and z/OS*, SG24-5383
- ▶ *TCP/IP Tutorial and Technical Overview*, GG24-3376
- ▶ *Migrating Subarea Networks to an IP Infrastructure Using Enterprise Extender*, SG24-5957
- ▶ *Networking with z/OS and Cisco Routers: An Interoperability Guide*, SG24-6297
- ▶ *zSeries HiperSockets*, SG24-6816

## Other resources

These publications are also relevant as further information sources:

- ▶ *z/OS V1R2.0 UNIX System Services Planning*, GA22-7800
- ▶ *z/OS V1R2.0 UNIX System Services User's Guide*, GA22-7801
- ▶ *z/OS V1R1.0-V1R2.0 MVS Initialization and Tuning Guide*, SA22-7591
- ▶ *z/OS V1R2.0 C/C++ Programming Guide*, SC09-4765
- ▶ *z/OS V1R2.0 C/C++ Run-Time Library Reference*, SA22-7821
- ▶ *z/OS V1R2.0 CS: IP Migration*, GC31-8773
- ▶ *z/OS V1R2.0 CS: IP Configuration Guide*, SC31-8775
- ▶ *z/OS V1R2.0 CS: IP Configuration Reference*, SC31-8776
- ▶ *z/OS V1R2.0 CS: IP User's Guide and Commands*, SC31-8780

- ▶ *z/OS V1R2.0 CS: IP System Administrator's Commands*, SC31-8781
- ▶ *z/OS V1R2.0 CS: IP Diagnosis*, GC31-8782
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 1 (EZA)*, SC31-8783
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 2 (EZB)*, SC31-8784
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 3 (EZY)*, SC31-8785
- ▶ *z/OS V1R2.0 CS: IP Messages Volume 4 (EZZ-SNM)*, SC31-8786
- ▶ *z/OS V1R2.0 CS: IP Application Programming Interface Guide*, SC31-8788

## Referenced Web sites

These Web sites are also relevant as further information sources:

- ▶ The z/OS Web pages

<http://www-1.ibm.com/servers/eserver/zseries/zos/installation/installz12.html>

## How to get IBM Redbooks

Search for additional Redbooks or Redpieces, view, download, or order hardcopy from the Redbooks Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become Redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

## IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

# Index

## Numerics

3172

Interconnect Controller 37

3745

and SNALINK 55

Ethernet connection 202

frame relay connection 202

NCPROUTE 195

token ring connection 201

## A

ACF/NCP 195

address 96

ALTLINK (X.25) 58

APPN 64

MPC use 41

ARP 77

AS Boundary Routers 83

AS\_Boundary\_Routing statement 169

ASBR 162

ATM 19

and OMPROUTE 114

configuration 20

ATM switch 20

ATM Virtual Circuit (VC) 19

ATMARP 22

server 20

definition 23

ATMLIS 22

definition 25

autonomous system

border router (ASBR) 162

configuration and redistribution 171

## B

backbone area 83

BEGINROUTES statement 98

broadcast 88

## C

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) 45

CDLC

ESCON channel 27

parallel channel 26

Channel Data Link Control (CDLC)

description 3

NCP ESCON channel 27

NCP parallel channel 26

NCPROUTE PDUs via CDLC 207

SNALINK and CDLC coexistence 206

Channel DLC (CDLC) 25

Channel Protocols 3

Channel Unit Address (CUA) 41

channel-to-channel (CTC) 3

overview 31

Cisco

Catalyst 6509 switch 161

IGX 8400 WAN switch 161

MPC router definitions 43

Cisco 3640 161

Cisco 7206 161

Cisco 7507 161

Cisco CMPC+ 41

CLAW 163

and OMPROUTE 112

datagram packing 29

CMPC+ 163

Common Link Access to Workstation (CLAW) 3, 29

communication and transmission control program (CTCP)

59

Communication Storage Manager (CSM) 11

64-bit memory 8

and HPDT 4

cost

equal cost 82

CPNAME 64

Cross-System Coupling Facility (XCF) 4, 64

CS for OS/390 IP

ORouteD start procedure 192

CSM 12

API 12

CTC

and OMPROUTE 112

CTCP 59

## D

daemon 81

data switching equipment (DSE) 58

DATAGRamfwd 183

DATE 60

DCE 58

Dedicated Access to X.25 Transport Extension (DATE)

60

designated router 84

direct IP datagram delivery 77

Direct Memory Access (DMA) 4, 5

discontiguous subnets 91

DISPLAY CSM command 12

distance vector algorithm 90

DSE 58

DUAL algorithm 158

DUAL session type 55

dynamic routing 79

dynamic UCBs 17

DYNAMIC XCF

- device name 66
- IPCONFIG definition 66
- link name 66
- overview 65
- static definition migration 68

DYNAMIC=NO 17  
DYNAMIC=YES 17

## E

EIGRP

- feasible successor 156
- features 156
- metrics 160
- neighbor discovery 158
- neighbor table 157
- topology table 157

Enhanced Interior Gateway Routing Protocol (EIGRP)  
156, 162

Enterprise Extender 44

Ethernet

- and OMPROUTE 115

Ethernet 3745 connection 202

exterior gateway protocols 74

## F

Fast Ethernet 43

FDDI 43

- and OMPROUTE 115

figuration 112

flooding 83

frame relay 3745 connection 202

## G

GATE 59

gateway 96

GATEWAY statement 98

GATEWAYS file 183

GATEWAYS\_FILE environment variable 186

Generic Routing Encapsulation (GRE) 163

Gigabit Ethernet (GbE)

- and QDIO 4
- overview 45
- support 45

GRE 143

## H

HCD 17

High Performance Data Transfer (HPDT) 4, 40

Hipersockets 5

Hipersockets (Internal Queued Direct I/O - iQDIO) 32

Hipersockets Accelerator 36

HPDT 40

- CPNAME 64
- displaying TCP/IP device resources in VTAM 13
- MPC 41
- MPCPTP (Multipath channel point-to-point) 40
- XCF 64

HPR 20

HyperChannel

- and OMPROUTE 114

Hyperchannel 37

## I

IBM 3745/3746 Communications Controllers 25

ICMP redirection 145

ICP 37

IDBLK 62

IDNUM 62

IGNORERedirect 183

Import\_Direct\_Routes 169

indirect IP datagram delivery 77

Interconnect Controller Program (ICP) 37

interface 96

interior gateway protocols 74, 80

IOACMD command 143

IOCP

- CDLC 25
- CLAW 29
- CTC 32
- Hipersockets 35
- LCS 39
- MPC 42
- MPCIPA 46

IP Assist 4

IP offload 45

IPA 4

IPAQENET 48

IPCONFIG DYNAMICXCF 65

IPLOCAL in ACF/NCP 205

IPOWNER in ACF/NCP 204

IPROUTE in ACF/NCP 205

iQD chpid 33

iQDIO 32

ISTLSXCF 64

IUTSAMEH 4, 44

- and XCF 65

## L

LAN Channel Station (LCS) 3

- overview 37

link state advertisement 83

Linux 32

LIS 22

LLC4 59

Logical IP subnet (LIS) 22

logical link control type 4 (LLC4) 59

Logical Partitions (LPARs) 5

LPAR 143

LU 6.2 55

## M

maximum transmission unit (MTU) 150

MD5 118

metric 91, 96

MPC 40

- and CTC 41

- and OMPROUTE 113
- displaying resources 13
- TRL definition 40
- MPC+ 4
- MPCLEVEL=NOHPDT 40
- MPCOSA 43
  - TRL definition 43
- MPCPTP 44
  - and CTC 31
- MPCPTP (Multipath Channel point-to-point) 40
- MPCPTP samehost
  - and OMPROUTE 113
- multi stacks. 186
- Multipath 82
- Multipath Channel (MPC) 40

## N

- native ATM 19
- NBMA 88
- NCP 26
  - definitions 201
- NCP IP router 195
- NCP packet switching interface (NPSI) 59
- NCPROUTE 195
  - and AUTOLOG 197
  - and SNALINK 52
  - and SNALINK for remote NCPs 209
  - configuring 197
- NCPROUTE use of SNMP DPI 198
- NCST Logical Unit 204
- nonbroadcast multiaccess 88
- NPSI 59

## O

- offload 5
- OMPROUTE 7, 81
  - cataloged procedure 105
  - commands 120
  - configuration 105, 164
  - configuration file 109
  - configuring interfaces 112
  - displaying routing table 120
  - environment variables 106
  - migrating from ORouteD 194
  - OSPF-specific options 117
  - overview 104
  - RIP-specific options 118
- OMPROUTE started procedure 105
- Open Shortest Path First (OSPF) 6
  - and OMPROUTE 7
- ORouteD 82, 181
  - configuring VIPA 192
  - migrating to OMPROUTE 194
- OS/390 IP
  - introduction 1
- OSA Address Table (OAT) 143
- OSA/SF
  - ATM 20
- OSA-2 143

- ATM 19
- OSA-Express 4, 143, 163
  - configuration 46
- OSPF 7, 80, 162
  - authentication 117
  - configuration 163
  - configuration in the sysplex 163
  - hello protocol 84
  - MTU sizes 150
  - OMPROUTE configuration 148, 150
  - stub area 84, 149
    - interfacing with EIGRP 168

## P

- packet switching data networks (PSDN) 58
- parallel sessions 54
- PARSESS 54
- Permanent Virtual Circuit (PVC) 20
- point-to-multipoint 88
- point-to-point 88
  - SNALINK 52
- PRIROUTER 47
- private address 96
  - and RIP 94
- private ATM switches 19
- protocol data units
  - request 90
  - response 90
- PSDN 58
- PTP Samehost 4
- public ATM switches 19
- PVC 20

## Q

- QDIO 45
- QoS
  - and ATM 20
- Quality of Service (QoS) 20
- Queued Direct I/O (QDIO) 4
  - overview 45

## R

- Redbooks Web site 212
  - Contact us xi
- redistribution 172
- reliable transport protocol 158
- replaceable static routes 98
- resolver configuration file
  - and OMPROUTE 107
  - and ORouteD 182
- RESOLVER\_CONFIG
  - OMPROUTE 107
  - ORouteD 194
- RESOLVER\_CONFIG environment variable 186
- RFC 1058 7
- RFC 1583 7
- RFC 1723 7
- RIP 7, 89

- filters 118
- ORouteD configuration 192
- RIP-2 81
  - configuration 93
  - overview 92
- RIP-2 message format 92
- RIT 197
- route maps 172
- RouteD
  - dumping RIP tables 189
  - MODIFY command support 188
- RouteD server configuration file 182
- RouteD started procedure 185
- ROUTED\_PROFILE environment variable 186
- router 74
  - AS Boundary routers 83
  - designated router 84
  - neighbor router 83
- routing
  - area 82
  - area border routers 83
  - autonomous system 82
  - d parameter 185
  - daemons 81
  - DATAGRamfwd 183
  - definition 89
  - distance vector algorithm 90
  - dynamic 79
  - ep parameter 185
  - GATEWAYS file 183
  - GATEWAYS\_FILE environment variable 186
  - IGNORERedirect 183
  - maximum transmission unit 104
  - mtu 104
  - OMPROUTE 104
  - OMPROUTE started procedure 105
  - OMPROUTE virtual link 104
  - open shortest path first 82
  - OpenMvs Multi Protocol Router 104
  - overview 75
  - resolver configuration file 107, 182
  - RESOLVER\_CONFIG 107
  - RESOLVER\_CONFIG environment variable 186
  - routed server configuration file 182
  - RouteD started procedure 185
  - ROUTED\_PROFILE environment variable 186
  - router ID 82
  - STDENV keyword 106, 185
  - STDERR keyword 106, 185
  - STDOUT keyword 106, 185
  - tables 79
  - TCPIP.PROFILE 183
  - terminology 74
  - VARSUBNETTING 183
  - vipa 104
- Routing Information Protocol 89
  - database 96
  - definition 89
  - discontiguous subnets 91
  - limitations 90, 91

- RIP-2 92, 93
- Routing Information Protocol (RIP) 6
- routing information table (RIT) 197

## S

- SECROUTER 47
- session type 55
- SINGLE session type 55
- SNA Link 55
- SNALINK 52
  - controlling use of 206
- SNMP DPI and NCPROUTE 198
- SOURCEVIPA 146, 193
- SRBEXIT 54, 57
- static routes
  - and OMPROUTE 101
  - displaying 100
  - updating 101
- static routing 79
- STDENV keyword 106, 185
- STDERR keyword 106, 185
- STDOUT keyword 106, 185
- sticky bit 106, 185
- stub area 84
- successor 156
- SVC 20
- Switched Virtual Circuit (PVC) 20
- SYSCONE system variable
  - dynamic xcf usage 65
- sysplex 64

## T

- TCPIP.PROFILE 183
- timer 96
- token ring
  - and OMPROUTE 115
- token-ring 3745 connection 201
- totally stubby area 84
- transit area 84
- TRANSLATE statement 20
- Transport Resource List Element (TRLE) 13
- TRL 13
- TRLE 13, 20
  - ATM 21
  - displaying 13

## U

- UCB
  - 4-digit addressing 17
  - dynamic 17
- Unit Control Blocks (UCBs) 17
- Upper Layer Protocol ID (ULPID) 14

## V

- VARSUBNETTING 183
- VC 19
- VIPA
  - and OMPROUTE 116

- for fault tolerance 145
- VIPA takeover 152
- Virtual IP Address (VIPA) 58
- Virtual IP Addressing
  - gateways options 190, 191
  - interfaces 117
  - multi stacks 186
- virtual link 83
- VTAM 13

## **X**

- X.25 58
  - call redirection 58
  - fast connect 59
  - hunt group 58
  - SNA INTERCONNECT (XI) 58
- X.25 NPSI
  - definition 60
- X.25 NPSI connection
  - ALTLINK 58
  - DATE 60
  - GATE 59
  - IDBLK 62
  - IDNUM 62
- X.25 SNA INTERCONNECT 58
- XCF
  - and MPC 40
  - and OMPROUTE 113, 150
  - migrating from static to dynamic 68
  - OMPROUTE configuration 151
  - sysplex 64
  - XCFINIT=YES 64
- XCFINIT=YES 40, 64
- XI 58

## **Z**

- z/VM 32







**Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing**

(0.2"spine)

0.17"->0.473"

90-<->249 pages







# Communications Server for z/OS V1R2 TCP/IP Implementation Guide Volume 4: Connectivity and Routing



**Includes scenario-based descriptions of network interfaces such as QDIO**

**Provides detailed descriptions of static and dynamic routing**

**Covers VIPA considerations, OSPF, RIP, and more**

The Internet and enterprise-based networks have led to the rapidly increasing reliance upon TCP/IP implementations. The zSeries platform provides an environment in which critical business applications flourish. The demands placed on these systems is ever-increasing and such demands require a solid, scalable, highly available, and highly performing Operating System and TCP/IP component. z/OS and Communications Server for z/OS provide for such a requirement with a TCP/IP stack that is robust and rich in functionality. The *Communications Server for z/OS TCP/IP Implementation Guide* series provides a comprehensive, in-depth survey of CS for z/OS.

*Volume 4* covers in great detail the connectivity options available to CS for z/OS. In particular, we provide scenario-based descriptions of interface types such as hipersockets, QDIO, MPC, and LCS. The main thrust of this redbook, however, is in the interconnection of networks as pertains to CS for z/OS. That is, we focus on the mechanism by which z/OS can interface with other networks via routing. We discuss static and dynamic routing including RIP and OSPF as well as VIPA routing considerations. We also cover interoperability with EIGRP.

Because of the varied scope of CS for z/OS, this volume is not intended to cover all aspects of this topic. The main goal of this volume is to provide an overview of the network interfaces available to CS for z/OS as well as the routing foundation necessary to interoperate in the complex internetworks of today. For more information, including details about applications available with CS for z/OS IP, please reference the other volumes in the series.

## **INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

## **BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)

SG24-6516-00

ISBN 0738424145