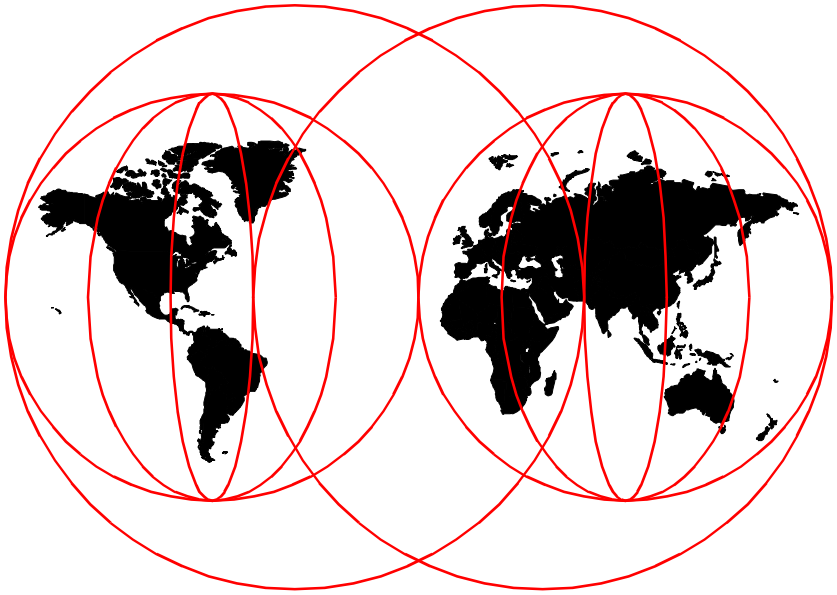
Load Balancing for eNetwork Communications Servers

Carla Sadtler, John Chambers, Ariane Schuldhaus



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5305-00

SG24-5305-00

International Technical Support Organization

Load Balancing for eNetwork Communications Servers

April 1999

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, “Special Notices” on page 243.

First Edition (April 1999)

This edition applies to Version 5 of IBM eNetwork Communications Server for AIX, 5765-D20, Version 6 of IBM eNetwork Communications Server for Windows NT, 30L9022, Version 5 of IBM eNetwork Communications Server for UnixWare 7, 5765-D48, Version 1 Release 3.0 of IBM LoadLeveler, 5765-145, and Version 1.0 of IBM WebSphere Performance Pack, 04L9231.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Preface vii
The Team That Wrote This Redbook vii
Comments Welcome viii

Chapter 1. Introduction 1
1.1 Load Balancing for TN3270 and TN5250 Clients 1
1.1.1 ISS Functions 1
1.2 Load Balancing for SNA API Clients 2
1.3 IBM Products 3
1.4 WebSphere Performance Pack Load Balancing Component 4
1.4.1 eNetwork Dispatcher (No Manager) 5
1.4.2 eNetwork Dispatcher with the Manager 5
1.4.3 ISS Component of eNetwork Dispatcher 6
1.5 Load Balancing with SLP 7
1.6 Load Balancing CS/UnixWare Win32 Clients 8
1.7 Load Balancing across Links on a Single Server 9
1.8 IBM LoadLeveler 1.3.0 9

Chapter 2. WebSphere Load Balancing Component 11
2.0.1 WebSphere Performance Pack Version 2.0 11
2.1 Load Balancing Component Overview 12
2.2 Dispatcher 13
2.2.1 How the Dispatcher Function Works 14
2.2.2 Dispatcher Components 15
2.2.3 Proportions of Importance 17
2.2.4 Information Flow 18
2.2.5 TCP Ports Used by the Dispatcher 20
2.2.6 The Flow of the IP Packets 20
2.3 Interactive Session Support 24
2.4 Summary 25
2.4.1 Using the Dispatcher Executor Alone 25
2.4.2 Using the Dispatcher Executor with Advisor Input 26
2.4.3 Using the ISS Component 27

Chapter 3. Scenario 1: Using the WebSphere Dispatcher 29
3.1 Network Environment 29
3.2 Cluster Address and Non-Forwarding Address 30
3.3 Dispatcher Configuration 31
3.3.1 Starting the Server Component 32
3.3.2 Starting the GUI 33
3.3.3 Starting the Executor 35

3.3.4 Configuring the Executor 37
3.4 TCP Server Configuration 50
3.5 TCP/IP Server Configuration Unique to Communications Server 57
3.6 Round-Robin Load Balancing for TN3270 Servers 60

Chapter 4. Scenario 2: WebSphere Dispatcher and the Advisors 67
4.0.1 Customization of the Manager and the Advisors 71
4.0.2 Saving the Configuration 72

Chapter 5. WebSphere Interactive Session Support 75
5.1 ISS Cells and Services 76
5.2 ISS Configuration Files 76
5.2.1 Cell and Its Attributes 80
5.2.2 Nodes 82
5.2.3 Services 82
5.2.4 Resources 83
5.2.5 ISS Observers 86
5.2.6 ISS Selection Methods 88
5.3 Managing ISS 89
5.3.1 The issd Command 89
5.3.2 Controlling ISS 91
5.3.3 Stopping ISS 92

Chapter 6. Scenario 3: WebSphere ISS and DNS 93
6.1 ISS Configuration 94
6.1.1 Copy the File to Each Server 97
6.2 Setting up the Domain Name Server 97
6.3 Starting the ISS Monitor and Agents 100
6.4 Load Balancing 100

Chapter 7. Scenario 4: WebSphere ISS and Dispatcher 107
7.1 ISS Configuration 108
7.2 Starting the ISS Monitor and Agents 111
7.3 Setting up the Dispatcher 112

Chapter 8. CS/AIX and LoadLeveler 115
8.1 When Would You Use the ISS Component of LoadLeveler? 115
8.2 Load Balancing Overview 115
8.3 TCP/IP Domain Name Server 118
8.4 Interactive Session Support (ISS) 119
8.5 Setting Up Load Balancing across Multiple Servers 119
8.5.1 Installation of the LoadLeveler 1.3 Base 120
8.5.2 Configuring ISS 120
8.5.3 Configuring the TCP/IP Domain Name Server 123

8.5.4 Other Important Things to Do 126
8.5.5 Starting the Configuration 127
8.6 LoadLeveler ISS Example Network 128
8.6.1 RS600033 129
8.6.2 JUPITER and CSAIX1 134
8.6.3 ISS Monitor Output 135

Chapter 9. Server Load Balancing Using SLP 141
9.1 Overview 141
9.2 Implementation 142
9.2.1 Dependent LU Sessions 143
9.2.2 Independent LU Sessions 143
9.2.3 Scopes 144
9.3 Design Tips 144
9.4 Server Configuration 147
9.4.1 Server Load Balancing Parameters 149
9.4.2 Sample Server Configuration 150
9.5 SNA API Client Configuration for LUA Sessions 151
9.5.1 3270 Session Configuration 153
9.6 Client Configuration for APPC Sessions 155
9.6.1 5250 Client Session Configuration 158
9.7 Using PCOMM 4.3 160
9.8 Monitoring Load Balancing 162

Chapter 10. Load Balancing for CS/UnixWare Win32 Clients 165
10.1 CS/UnixWare Client/Server Concepts 165
10.1.1 SLIM Concepts 166
10.1.2 The Server Role 166
10.1.3 Master/Backup Handover 167
10.1.4 Network Data File 167
10.1.5 Connecting Clients to the Network 168
10.1.6 Load Balancing and Hot Backup 169
10.2 CS/UnixWare Win32 Client 169
10.2.1 WIN32 Client Configuration 170
10.3 Load Balancing Configuration Overview 172
10.4 CS/UnixWare Load Balancing Scenario 173
10.5 Setting Up the CS/UnixWare Servers 175
10.5.1 The Domain 175
10.5.2 LU Pools 177
10.5.3 Defining Users 179
10.5.4 TN3270 Setup 182
10.6 Installing the CS/UnixWare Client 183
10.7 Configuring and Using the CS/UnixWare Client 189

10.7.1 Testing the Configuration 191

Chapter 11. Single Server Load Balancing across Links 195

Appendix A. Determining System Load with sys_load 197
A.1 Values Returned by sys_load 200
A.2 sys_load Script 201
A.3 sys_load Modified 210

Appendix B. WebSphere Load Balancing Component Installation . . 215
B.1 Installation on AIX 215
B.2 Installation on Windows NT 223

Appendix C. WebSphere Interactive Session Support Installation . . 237
C.1 Installation on AIX 237
C.2 Installation on Windows NT 239

Appendix D. Special Notices 243

Appendix E. Related Publications 247
E.1 International Technical Support Organization Publications 247
E.2 Redbooks on CD-ROMs 247
E.3 Other Publications 247

How to Get ITSO Redbooks 249
IBM Redbook Fax Order Form 250

Index 251

ITSO Redbook Evaluation 253

Preface

This redbook will help you design a load balancing technique for IBM eNetwork Communications Server environments. It covers techniques that include Interactive Session Support (ISS), the WebSphere load balancing component, and techniques unique to Communications Servers. It is primarily for environments with TCP/IP clients, such as TN3270, TN5250, and SNA API clients.

This book looks at IBM eNetwork Communications Servers on Windows NT, UnixWare, and AIX.

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.

Carla Sadtler is a Senior ITSO Specialist for Communications Server for AIX and network integration projects at the International Technical Support Organization, Raleigh Center. She has been with the ITSO for 13 years working in a number of areas including MVS, Communications Servers, Web-to-host integration, and Lotus Notes.

John Chambers is a software engineer in the United States. He has worked at IBM for four years and is currently assigned to the Host Access Server support team in the Research Triangle Park, NC. His areas of expertise include networking and communications on the Intel-based PC platform. He has written extensively on the use of digital imaging in scientific research during his previous life as a geologist.

Ariane Schuldhaus joined IBM Germany in 1984 and has been in the networking area since 1994. She is responsible for country-wide technical marketing support of Communications Servers for OS/2, Windows NT, AIX and UnixWare. She is one of the authors of the redbook *IBM Communications Server for OS/2 Warp Version 4.0 Enhancements* and is an IBM certified communications specialist for Windows NT Enterprise Communications.

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

Juan Rodriguez
International Technical Support Organization, Raleigh Center

Marco Pistoia
International Technical Support Organization, Raleigh Center

Margaret Ticknor
International Technical Support Organization, Raleigh Center

Dana Price
IBM Research Triangle Park

Paul Dowds
IBM Research Triangle Park

Paul Landay
IBM Research Triangle Park

Donna Wooten
IBM Research Triangle Park

Philip Pearl
Data Connection, Ltd.

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in “ITSO Redbook Evaluation” on page 253 to the fax number shown on the form.
- Use the online evaluation form found at `http://www.redbooks.ibm.com`
- Send your comments in an internet note to `redbook@us.ibm.com`

Chapter 1. Introduction

This book will take a look at some of the techniques available for introducing load balancing into a Communications Server environment. There are two ways to think about load balancing in this type of environment. True load balancing determines the load on a server, including CPU utilization, available memory, and available resources. This is different from session balancing where the only attempt at load balancing is by ensuring that every session does not use the same server or host link. This can be done with round-robin or random selection techniques.

We will cover instances of both types of balancing. Often the term, load balancing, will be used to represent both load and session balancing techniques. When we refer to session balancing, we will be talking only about balancing sessions through communications servers with no load information being considered in the decision.

For enterprises using IBM eNetwork Communications Servers, there are several solutions available for balancing incoming client requests among servers or host links. The appropriately balanced Communications Server environment can show increased availability, simpler client configuration and distribution of the work on your network that makes sense to you.

This book will discuss primarily, load balancing for TCP/IP workstations across Communications Servers. TCP/IP workstations include TN3270 clients, TN5250 clients, and SNA API clients. It will cover several IBM products in relation to their usefulness in this environment. All products discussed have many more features and should be investigated separately if you are considering implementing them in your environment.

1.1 Load Balancing for TN3270 and TN5250 Clients

Often, communications servers must handle large numbers of TN3270 or TN5250 clients. These are clients using TCP/IP to connect to a communications server, which provides a Systems Network Architecture (SNA) session to the host for 3270 or 5250 emulation.

1.1.1 ISS Functions

Interactive Session Support (ISS) is a primary component of two of the IBM products we will cover extensively in this book. The ISS component has been a core component of the popular LoadLeveler product up through Version 1.3

and the eNetwork Dispatcher now included in the IBM WebSphere Performance Pack.

The way that ISS spreads the workload across many providers is both efficient and simple in its design. ISS can be configured to detect conditions in the environment, enabling it to determine the appropriate servers to forward requests to. Or, ISS can be set up with a static design that allows for simple load sharing.

ISS can directly influence the load balancing by altering a TCP/IP domain name server to ensure that a common pool name used by the clients resolves to the TCP/IP address of the preferred server. ISS can also be used as feedback to other components of LoadLeveler and WebSphere Performance Pack, allowing a deeper layer of load balancing techniques.

The load sharing technique, also known as round-robin, does not take into account any variables concerning how busy a particular server is, only whether it is actually running. This provides a moderate assurance of availability for the users, in that they would request a resource from a system which in turn would assign a resource from a static list of servers it is configured to use. If the server assigned did not respond, the load sharing service would in turn attempt the next server in its list until the request had been met.

The next layer of reliability and, therefore, complexity involves the use of variables configured for the servers to reflect their capabilities and current usage statistics. The current usage (load) on the server is detected by means of a portion of code that resides on the production servers that periodically queries aspects of the servers' processor and memory utilization, connectivity options (such as speed of the links or available bandwidth) and the availability of TN3270 sessions. You can also add custom variables into the mix that reflect a unique environment. As these reports are gathered, they accumulate an integer sum that reflects their load to ISS. ISS then chooses the preferred server based on these numbers and either updates the domain name server, or passes this information back to the appropriate component (LoadLeveler or WebSphere Performance Pack eNetwork Dispatcher) which determines the most effective path for the traffic.

1.2 Load Balancing for SNA API Clients

There are other methods that can contribute to an effective distribution of the network traffic not limited to interaction between the servers but by incorporating some of the load distribution mechanics to the client systems.

By using SNA API code on the client system and an API-aware emulation program such as IBM's Personal Communications (PCOMM), the servers can communicate their resource availability via the SNA API to the clients, which in turn then choose a server for connection. IBM has been able to do this by incorporating support for the Service Location Protocol (SLP) into Communications Server for Windows NT 6.0, IntranetWare for SAA 3.0, and IBM eNetwork Personal Communications Server 4.3.

For an in-depth look at load balancing with SLP, review the Internet Draft document from the Internet Engineering Task Force (IETF), available via the Web at `http://www.ietf.org/internet-drafts/draft-ietf-tn3270e-service-loc-03.txt`. There are also numerous white papers and overviews from IBM, Novell, and the Zephyr Corporation among others, discussing both client and server implementations and configurations.

Communications Server for UnixWare ships with the SNA API client for the Win32 platform (Microsoft Windows 95, 98 and NT V4). This works in conjunction with the CS/UnixWare server to randomly distribute the load of TN3270 sessions to a group of servers configured in a domain model. It does not use SLP. The clients can be configured with a list of servers to request resources from logical unit (LU) pools or can transmit user datagram protocol (UDP) broadcasts to the servers without knowing their names or addresses. Load balancing (or more accurately, session balancing) is achieved through the randomness of the clients choice of servers.

1.3 IBM Products

IBM current offerings for enhancing performance through various methods, including load balancing, are spread across a number of components within the WebSphere Performance Pack, the separate LoadLeveler for AIX product, and some features exclusive to individual products. While the current offerings are primarily available for the server platforms of AIX and Microsoft Windows NT, there are options within the WebSphere Performance Pack that simply require network connectivity and the appropriate TCP/IP program to participate in the control of a cluster or pool of available resources (such as the TN3270 Server, HTTP daemon, etc.).

The load balancing component of WebSphere Performance Pack consists of the eNetwork Dispatcher product and offers the user the ability to spread network traffic across a group of servers offering similar resources in a way transparent to the client. The two major parts to this component are ISS and the Dispatcher. You could use either piece or both depending on your

configuration needs. The Dispatcher can provide a simple weighted load sharing configuration regardless of the participating servers in the group. It runs on AIX, Sun Solaris or Microsoft Windows NT and can even provide support across networks in a WAN configuration. ISS can be used alone with a domain name server to balance the load or can be configured to pass information back to the Dispatcher component.

The LoadLeveler for AIX 1.3 product, which includes an ISS component, will be discussed in this book even though it is an older product. It has a significant install base at the time of writing this manual and its setup might be more familiar to many readers. The current version, 2.1.0, is not covered, as it has since dropped the ISS component. The current offering for ISS is the eNetwork Dispatcher component of WebSphere Performance Pack. LoadLeveler 1.3 does include ISS and is appropriate for balancing the load of CS/AIX TN3270 and TN5250 servers. As with WebSphere, the ISS component of LoadLeveler can be used alone with a domain name server, or can pass load information back to the LoadLeveler component.

1.4 WebSphere Performance Pack Load Balancing Component

The former eNetwork Dispatcher product has been rolled into the WebSphere Performance Pack V1.0 as the load balancing component. Network Dispatcher V2.0 is included in the WebSphere Performance Pack V1.0. The next release, called SecureWay Network Dispatcher V2.1 is included in WebSphere Performance Pack V2.0.

There are several ways the eNetwork Dispatcher can be used to balance TN3270/TN5250 loads among communications servers. The load balancing server can be configured to use the Dispatcher, the ISS component, or both.

The Dispatcher includes a Manager component that can be used to provide load information to the Dispatcher. Without the Manager you have a weighted round-robin session balancing scheme. Adding the Manager to the configuration allows information regarding the load of the servers to factor into the weighting mechanism, giving you a more accurate load balancing scheme.

WebSphere Performance Pack Load Balancing servers can be:

- AIX
- Windows NT
- Sun Solaris

1.4.1 eNetwork Dispatcher (No Manager)

Using the Dispatcher without the Manager gives you a weighted round-robin session balancing scheme. The highlights of the method are:

- This method works for any communications server platform.
- Servers are defined to the Dispatcher by TCP/IP port within a cluster.
- Clients point to the IP address or host name of the Dispatcher cluster.
- Incoming traffic from clients goes through the Dispatcher machine.
- Traffic from the server to the client is direct and does not involve the Dispatcher machine.
- Weights are assigned to each server by an administrator. They can be changed by the administrator dynamically.
- No load information from the servers is given to the Dispatcher.
- The Dispatcher does not recognize if a server is no longer available, but the administrator can manually tell the Dispatcher the server is down and it will be taken out of the round-robin sequence.

For an example of this type of configuration see Chapter 3, “Scenario 1: Using the WebSphere Dispatcher” on page 29.

1.4.2 eNetwork Dispatcher with the Manager

Using the Dispatcher with the Manager gives you a weighted round-robin session balancing scheme. The Manager uses input from advisors and other external sources to dynamically change the server weights so the least loaded server will be the most likely choice.

The highlights of the method are:

- This method works for any communications server platform.
- Servers are defined to the Dispatcher by TCP/IP port within a cluster.
- Clients point to the IP address or host name of the Dispatcher cluster.
- Incoming traffic from clients goes through the Dispatcher machine.
- Traffic from the server to the client is direct and does not involve the Dispatcher machine.
- Advisors that check the response time to servers are shipped with the product for the most popular TCP/IP ports. Advisors can be modified or created for other situations. An advisor is shipped for the Telnet port (23) and can be used if the TN3270 Server is customized to use port 23. Otherwise, an advisor can be written based on a sample sent with the product.
- The Manager can take input from advisors and from other external sources such as the ISS component.
- Weights are dynamically changed by the Manager based on load information from the advisors and external sources.

An example of this type of configuration using advisor input is in Chapter 4, “Scenario 2: WebSphere Dispatcher and the Advisors” on page 67.

An example of this type of configuration using ISS input is in Chapter 7, “Scenario 4: WebSphere ISS and Dispatcher” on page 107.

1.4.3 ISS Component of eNetwork Dispatcher

ISS can be configured to determine load information on the communications servers by using internal or external metrics. The information is fed back to an observer, which can either update a TCP/IP domain name server or send the information back to the Dispatcher component.

Highlights of this method include:

- Communications Servers must run the issd daemon and therefore must be one of the platforms supported by the WebSphere Performance Pack. The supported platforms are AIX, Windows NT, and Sun Solaris.
- Internal metrics are provided to measure CPU load and free physical memory.
- External metrics can be any executable on the server that returns a numerical value indicating its load.
- CS/AIX provides an external metric that was specifically written to determine the load on a server running CS/AIX.
- Load information can be sent back to more than one observer. The observer can be the Dispatcher for server weighting by the Manager, or it can be a domain name server.

  For domain name server observers:

  - ISS modifies the IP address of the server pool to reflect the least loaded server. Clients are configured to use the server pool host name and use TCP/IP name resolution to resolve this name to the IP address of the least loaded server. This means the client must use the name resolution for each session and must be able to handle the fact that one TCP/IP host name may resolve to different IP addresses.
  - The session between the client and the server is direct and does not involve the load balancing server.
  - The domain name server can be a site server or a separate domain name server. The domain name server will be refreshed frequently, which creates a load on the domain name server machine.
  - The domain name server machine does not have to be the same machine as the load balancing machine but must be running the issd daemon.

  For Dispatcher observers:

- Clients are configured to use the IP address of the server cluster. The IP address and TCP/IP host name are always the same.
- Advisor input can be included with the load balancing information from ISS to weight the servers.
- Incoming traffic from clients goes through the Dispatcher machine.
- Traffic from the server to the client is direct and does not involve the Dispatcher machine.

For an example of this type of configuration using external metrics supplied by CS/AIX and a TCP/IP domain name server see Chapter 6, “Scenario 3: WebSphere ISS and DNS” on page 93.

For an example of using ISS to feed back load information to the Dispatcher see Chapter 7, “Scenario 4: WebSphere ISS and Dispatcher” on page 107.

1.5 Load Balancing with SLP

IBM eNetwork Communications Server for Windows NT (CS/NT) and IntranetWare for SAA Version 3 provide users a client component allowing them to use TCP/IP to reach the server components’ SNA functions. The client/server, or split-stack operation, provides a load balancing feature using the Service Location Protocol (SLP). Servers register and advertise services that contain load factors. The clients use this information to select a server that satisfies its request.

The highlights of this load balancing method are:

- Servers can be CS/NT and IntranetWare for SAA servers.
- Clients can be CS/NT or IntranetWare for SAA SNA API clients, IBM eNetwork Personal Communications (PCOMM) 4.3 clients, QEL/MU 3270 sessions (NetWare), third party 3270 emulators.
- Load balancing is provided for both LUA (LU 0-3) and LU 6.2 sessions.
- The load for dependent LUs represents a percentage of available resources from a particular server. The load percentage is calculated by dividing the number of active application connections by the total number of LUs (0 to 3) available.
- The load for LU6.2 represents a percentage of available resources from a particular server. The load percentage is calculated by dividing the total number of conversations over all local LUs on a particular server by the cumulative maximum session limit for all local LUs. The maximum session limit is the LU 6.2 session limit specified during configuration. If the maximum session limit is specified as zero (0), indicating there is no session limit, the default maximum local LU session limit of 512 per local

LU is used when the load is calculated. The default maximum local LU session limit can also be specified during configuration.

- Load factors can be defined to compensate for the differences in servers.
- Servers are organized into scopes. A server can be scoped or unscoped.
  - For LUA sessions, the load is balanced across a named LU pool. The client provides the LU pool name and defines whether to use a named scope, unscoped servers, or any server. Each new session will determine the best server, so one client may have multiple sessions using different servers.
  - APPC sessions are distributed among servers in a named scope. The initial connection determines the server that all subsequent connections will use.

An example of SLP load balancing can be seen in Chapter 9, “Server Load Balancing Using SLP” on page 141.

1.6 Load Balancing CS/UnixWare Win32 Clients

IBM eNetwork Communications Server for UnixWare (CS/UnixWare) provides client/server protocols, allowing clients to use TCP/IP to connect to server SNA functions. All servers and clients communicating with each other are referred to as a domain. LU pools can be defined across a domain, allowing clients to connect to any server with the requested LU. Session balancing is accomplished by having the clients randomly choose to which server to connect.

The highlights of this method are:

- Servers are CS/UnixWare servers.
- Clients are Windows NT or Windows 95 machines with the CS/UnixWare Win32 client installed.
- Load balancing can be accomplished for both LUA and LU 6.2 sessions.
- Clients belong to the same domain as the servers. They choose a server by specifying a list of servers to try or by using a UDP broadcast.
- If a list is specified, clients will always choose the first server in the list. If it is not available they go to the second server, and so on. Session balancing relies on the clients having servers in a random order so each client does not necessarily go to the same server as the others.
- If the broadcast method is used, the clients find out about the services in the domain from a server in its own subnet.
- There is no attempt to measure the load on each server and use the least loaded.

The CS/UnixWare Win32 Client and load leveling functions are covered in Chapter 10, “Load Balancing for CS/UnixWare Win32 Clients” on page 165.

1.7 Load Balancing across Links on a Single Server

A common feature of the Communications Servers is the ability to define an LU pool that spans multiple links on the same server. Sessions for this LU pool will be automatically rotated among the host links containing LUs in this pool. This gives a very basic session balancing mechanism among host links.

This load balancing feature is covered in Chapter 11, “Single Server Load Balancing across Links” on page 195.

1.8 IBM LoadLeveler 1.3.0

Another product we are including in this book is the IBM LoadLeveler 1.3.0, which included an earlier version of ISS than that included with eNetwork Dispatcher and the WebSphere Performance Pack. LoadLeveler 1.3.0 is still available for AIX platforms only. The latest release of LoadLeveler 2.1.0 does not include the ISS function. We include it here because there is a base of customers that have LoadLeveler installed and may want to use it for balancing sessions among communications servers.

You can find out more about using the ISS functions of LoadLeveler 1.3.0 in Chapter 8, “CS/AIX and LoadLeveler” on page 115.

Chapter 2. WebSphere Load Balancing Component

IBM WebSphere Performance Pack Version 1.0 provides a load balancing component known as IBM eNetwork Dispatcher Version 2.0. This component improves the performance of servers by directing TCP/IP session requests to different servers within a group, balancing the load transparently to end users and other applications. The load balancing component allows increased server capacity by allowing multiple TCP/IP servers to appear as a single logical server. TN3270 and TN5250 clients can benefit from this since they use TCP/IP to access the SNA servers.

The load balancing component of IBM WebSphere Performance Pack includes the following features:

- Rule-based Web traffic load balancing
- Load balancing through firewalls to remote locations
- Integration of HTTP, mail service, FTP, Telnet and news traffic load balancing
- Independent network return paths to enhance application server responses to clients
- Agents that give live multi-server monitoring feedback
- High availability fail-over schemes

IBM WebSphere Performance Pack Version 1.0 is supported on the following platforms:

- IBM AIX 4.2.1 or later
- Sun Solaris 2.5 or later
- Microsoft Windows NT 4.0

2.0.1 WebSphere Performance Pack Version 2.0

At the time of this writing, WebSphere Performance Pack V2.0 had been announced but was not yet available. There have been some changes in the load balancing function. The changes do not affect the material in this book but are nice enhancements to be considered. The Network Dispatcher code has been upgraded and is now called SecureWay Network Dispatcher.

Server Directed Affinity: Users can direct requests from specified clients to specific servers. An API provided in Network Dispatcher is used to define the client-to-server relationship desired in the Network Dispatcher affinity tables. The user defines to which server requests from the specified client are routed. Subsequent to that, for as long as the user wants it to remain in effect,

all requests from the client are routed to that server. The user has the capability of querying the relationships, adding relationships, or removing relationships from affinity tables.

Improved Logging and Statistics: The logging and statistical data kept by Network Dispatcher is maintained in binary format. In addition, the command line and graphical user interface (GUI) are used to set parameters on how the binary data will be logged. A sample Java program is included that provides access to and manipulation of the log data.

Remote Configuration: An authenticated link using the GUI as well as configuring at the Network Dispatcher server is now an option. This productivity enhancement facilitates the centralization of Network Dispatcher administration and control. The authenticated link between the client performing the remote configuration and the server uses a public key/private key pair for authentication. The private key must be distributed to the clients selected for remote configuration of the server.

Content-Based Forwarding: The administrator can route requests based on Universal Resource Locator (URL) content. The URL content of a request is matched to user-generated rules that determine to which server, or cluster of servers, it will be routed. If a user-generated rule results in a choice of multiple servers, Network Dispatcher determines the best server for the job.

Star Cluster: This catch-all cluster includes two functions:

- Configure multiple aliases on the Network Dispatcher machine and have them all use the same port/server configuration
- Make the Network Dispatcher machine the default route for some traffic and have it load balance all the Internet Protocol (IP) traffic for ports configured on the star cluster, without configuring any aliases on the Network Dispatcher machine

Star Port: You can create a load balancing configuration for traffic to any port that has not been explicitly defined in your Network Dispatcher configuration. This could be used, for example, to load balance firewalls or discard requests for ports that have not been configured.

2.1 Load Balancing Component Overview

The load balancing component consists of two functions, the Dispatcher and Interactive Session Support (ISS). These components can be used separately or combined.

The Dispatcher function balances the load on a server within a LAN or WAN using a number of weights and measurements that are set manually or dynamically. It provides load balancing at a level of specific services, such as HTTP, FTP, SSL, NNTP, POP3, SMTP, and Telnet. The Dispatcher also has a built-in high availability feature.

The ISS component is a follow-on product to the ISS component in IBM LoadLeveler V1.3.0. Initially offered as a separate product, IBM eNetwork Dispatcher, this code is now included in the WebSphere Performance Pack. You can use the ISS function by itself to balance the load on servers within a local or wide area network using a TCP/IP domain name service (DNS) round-robin approach or a more advanced user-specified approach. Load balancing is performed at the machine level. ISS can also be used to provide server load information to a Dispatcher machine.

When used for load balancing, ISS works in conjunction with the DNS server to map DNS names of ISS services to IP addresses. When used to provide server load information, a DNS is not required.

2.2 Dispatcher

Unlike ISS, the Dispatcher function does not use a TCP/IP domain name server for load balancing. It balances traffic among your servers through a unique combination of load balancing and management software.

All client requests sent to the Dispatcher machine are directed to the server selected by the Dispatcher as optimal according to certain dynamically set weights. You can use the default values for those weights or change the values during the configuration process.

The Dispatcher has two important features:

1. The server sends a response back to the client without any involvement of the Dispatcher.
2. No additional code is required on your servers to communicate with the Dispatcher.

The Dispatcher offers a built-in high availability feature. This feature involves the use of a second Dispatcher machine that monitors the main, or primary, machine and stands by to take over the task of load balancing should the primary machine fail at any time.

Design Note

The Dispatcher can reside on the same machine as the server being balanced if the node is AIX or Solaris. This is not supported on Windows NT and will cause problems.

2.2.1 How the Dispatcher Function Works

The Dispatcher creates the illusion of having just one server by grouping systems together into a cluster that behaves as a single, virtual server. The service provided is no longer tied to a specific server system, so you can add or remove systems from the cluster, or shut down systems for maintenance, while maintaining continuous service for your clients. The balanced traffic among servers seems for the end users to be a single, virtual server. The site thus appears as a single IP address to the world. All requests are sent to the IP address of the Dispatcher machine, which decides for each client request which server is the best one to accept requests, according to certain dynamically set weights. The Dispatcher routes the client's request to the selected server, and then the server responds directly to the client without any further involvement from the Dispatcher. The Dispatcher can also detect a failed server and route traffic around it.

The Dispatcher receives the packets sent to the cluster. These packets have a source and a destination address; the destination address is the IP address of the cluster. All servers in the cluster and in the Dispatcher system have their own IP address and an alias for the IP address of the cluster. The Dispatcher system checks which server is the next best server to handle the load and routes the packet to that server. Since all servers in the cluster have an alias for the cluster's IP address, the Dispatcher routes this request based on the hardware address of the network adapter (MAC address) of the chosen server. It changes the hardware address of the packet to the hardware address of the selected server and sends the packet to the server. The server receives the packet and responds directly to the client. We see a more detailed explanation of this process in "The Flow of the IP Packets" on page 20.

The fact that the server can respond directly to the client makes it possible to have a small bandwidth network for incoming traffic, such as Ethernet or token-ring, and a large bandwidth network for outgoing traffic, such as asynchronous transfer mode (ATM) or fiber distributed data interface (FDDI).

The server machines in the cluster could be a mixture of heterogeneous servers of different sizes and types, such as UNIX, Windows NT or OS/2 machines.

2.2.2 Dispatcher Components

The Dispatcher consists of three main components:

- Executor

  This component supports port-based routing of TCP and UDP connections to servers based on the type of request received (for example HTTP, FTP or SSL). This module always runs when the Dispatcher function is being used.

- Manager

  This component sets weights used by the Executor based on internal counters in the Executor itself and feedback from the advisors and ISS monitoring (if ISS is used as a monitoring tool). Each unit of information given to the Manager by the advisors, ISS and Executor has a relative proportion of importance. You can give more importance to one unit of information over the others, or totally ignore one or more units of information. Using the Manager is optional, but if the Manager is not used, load balancing is performed using weighted, round-robin scheduling based on the current server weights.

- Advisors

  Advisors send requests to the back-end servers to measure actual client response time for a particular protocol. These results are then fed to the Manager to adjust the load balancing weights. Currently, there are advisors available for HTTP, FTP, SSL, SMTP, NNTP, POP3 and Telnet. Using the advisors is optional, but recommended. You also have the option of writing your own advisors.

Figure 1 on page 16 shows an overview of the load balancing components. The example shows a TN3270 network, but this could be any type of TCP/IP network services.

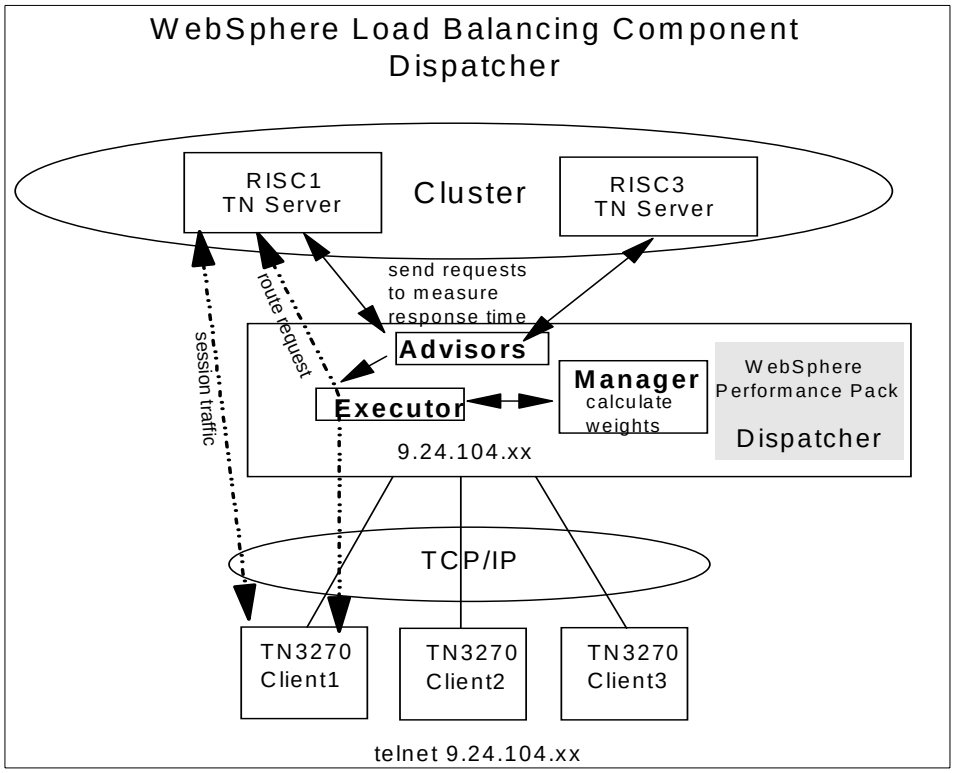


Figure 1. Dispatcher Components

There is also an SNMP subagent component that allows an SNMP-based management application to monitor the status of the Dispatcher.

2.2.2.1 Customizable Advisors

The load balancing component of IBM WebSphere Performance Pack offers you the possibility to write customizable advisors that will provide the precise information about servers that you need. Advisors, like the rest of the Dispatcher, must be compiled with Java 1.1. To ensure access to Dispatcher classes, make sure that the ibmnd.jar file (located in the lib subdirectory of the base directory) is included in the classpath system environment variable. Only advisors written to the correct level of Java will be supported.

A guideline on how to write customizable advisors, along with a sample Advisor, can be found in *eNetwork Dispatcher for Solaris, Windows NT, and AIX User's Guide Version 2,* GC31-8496. In AIX, the sample customizable advisor is located in:

```
/usr/lpp/eND/dispatcher/lib/CustomAdvisors
```

In NT the files are located in:

```
Program Files\eND\dispatcher\Lib\CustomAdvisors
```

2.2.3 Proportions of Importance

The Manager decides which is the least loaded server on a particular port in the cluster by looking at the weight of each server. The Manager will periodically update the weight of each of the server machines basing its decision on four parameters or policies:

1. The number of active connections on each TCP server
2. The number of new connections for each TCP server
3. Input from TCP server advisors
4. Information from system monitoring tools, such as ISS

Setting the server's weights in the load-balancing process is performed by using the so-called *proportions of importance*. Each of the above factors is attributed a number, from 0 to 100, that acts as a percentage. Zero means that the policy is not used, while 100 means that only that factor will be used. It is necessary that these proportions add up to 100. The default settings at the startup appears as:

```
50 50 0 0
```

2.2.3.1 Guidelines on Proportions of Importance Settings

Although generally there are no fixed rules on how to set the proportion value, it is still possible to provide some useful guidelines, described in the following.

The first two proportions are related to active and new connections respectively. Typically in an out-of-the-box configuration, this is all you will be able to use. That is why the default proportions at startup of Dispatcher appear as `50 50 0 0`.

The load on a server often depends mainly on the number of connections, both active and new. If you then start the Manager, it makes sense now to use a non-zero value for the advisor proportion. The standard advisors shipped with Dispatcher execute a trivial transaction on each TCP server. Normally you would expect a trivial response time to this transaction. Experience shows us that it is not a good idea to set the advisor proportion to a high value. We recommend `49 49 2 0` for this setup.

Things change if you start using a custom advisor. Your custom advisor can, for example, execute a Java servlet (or other application transaction) on the server side, and this servlet can gather very precise values about performance, throughput and response time via the server's API and feed it to the advisor when asked. Thus, you may find it appropriate to put the advisor proportion at a higher value. However, we recommend that you do not over-compensate, and do not set active and new connection proportions too low, otherwise you begin to restrict Dispatcher's adaptive and smoothing capabilities. Moreover, you may end up with a load balancer that follows your advisor too closely, resulting in a choppy profile that is probably less than ideal. In the final analysis, what works for you is best; you need to experiment with your custom advisor to see what results you get in the real world.

Finally, if you install the ISS daemon on your servers, it makes sense to add the ISS proportion to your mix. If you use a custom metric in ISS as well as a custom advisor, be sure to have them go after similar objectives, otherwise you may come to set one against the other with undesirable consequences.

2.2.4 Information Flow

The typical flow of information used in the Dispatcher is shown in the following diagram:

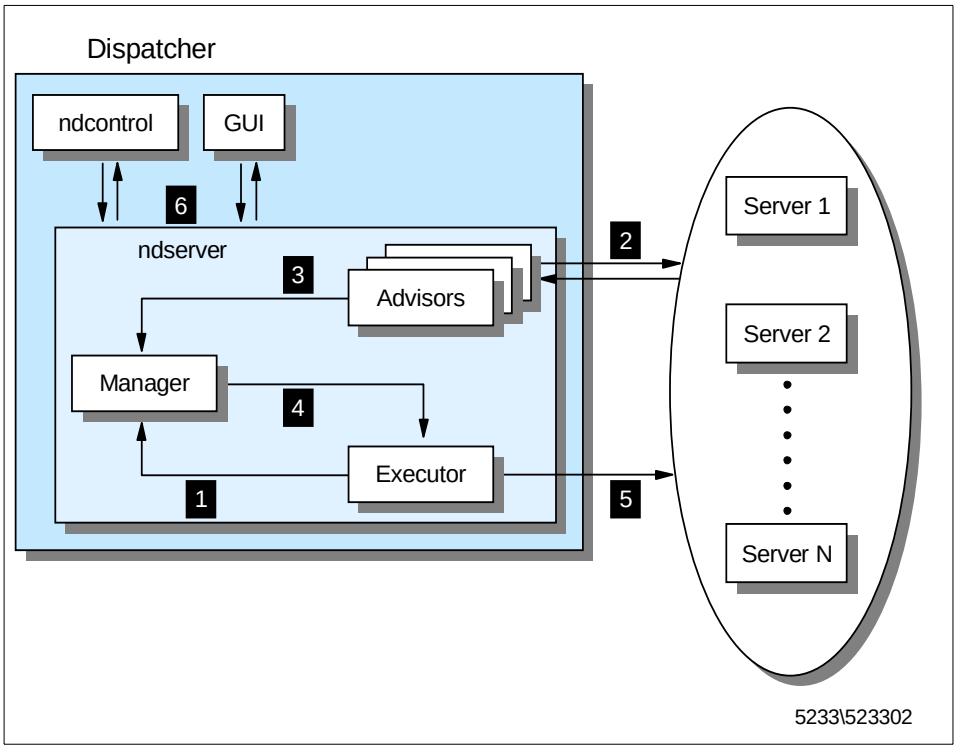


Figure 2. Information Flow Used in the Dispatcher

The logical information flow, represented in the above diagram, is described in the following list:

1. The Executor supplies information about new and active connections based on its internal counters to the Manager.
2. The advisors collect information about response time and availability from the servers for each service and each port.
3. After processing this information, they send it to the Manager.
4. The Manager uses all the information it receives (including information sent by the ISS component, if active, about the servers' load balancing according to a specific metric), calculates the new weights to be used in connection routing and sends the results to the Executor.
5. The Executor uses these new weights for TCP and UDP routing. If the Manager and the advisors are not running, the Executor does the routing based on its internal counters.

6. A command line interface, represented by the `ndcontrol` command, and a graphical user interface (GUI) are provided to configure and manage the Executor, Advisors and the Manager.

2.2.5 TCP Ports Used by the Dispatcher

The Dispatcher uses three TCP ports for its communications:

1. Port 1099: This port is used for GUI communications.
2. Port 10003: This port is used for receiving commands.
3. Port 10004: This port is used to receive metric response from ISS.

Sometimes it is necessary to use port numbers other than the defaults. If another application is already using port 1099 or port 10003 for communication, then you will need to change the `ND_PORT` value in the Dispatcher scripts to use different ports:

- On AIX, these scripts are named ndadmin, ndserver and ndcontrol and are located by default under the /usr/bin directory.
- On Windows NT, these scripts are named ndadmin.cmd, ndserver.cmd and ndcontrol.cmd and are located by default under the Bin subdirectory of the Load Balancing component eND directory.

To change the port used to receive metric reports from ISS, use the *`metric_port`* option when starting the Manager:

```
ndcontrol manager start log_file metric_port
```

Notice that if you specify a *`metric_port`* you must also specify a *`log_file`*. The above command has a correspondent version in the GUI, as we see in “Activating the Manager” on page 68.

Moreover, when you define the Dispatcher Observer in the ISS configuration file, you should indicate the port you have decided to use:

```
Dispatcher hostname metric_port
```

2.2.6 The Flow of the IP Packets

During a load balancing operation IP packets start flowing from a TCP client to a TCP server passing through the Dispatcher machine. The server’s response flows directly from the server to the client without any need to pass through the Dispatcher again. In order for this to work, each server, including the load balancing server needs to have a TCP/IP alias set up.

During the setup stage of the Dispatcher, an IP address is assigned to each cluster of servers to be balanced. This is the IP address representing all the servers that the clients use. On the load balancing server an alias for the cluster address must be defined to the network interface. On the servers, an alias for the cluster address must be defined to the loopback interface.

The reason for defining an alias to the cluster address on the network interface on the Dispatcher machine and the loopback device on all the cluster's TCP servers is based on the way the Dispatcher works and treats incoming IP packets.

The Dispatcher is designed to make several TCP servers appear as one in the TCP/IP environment. Let's assume that a request for the service managed by the cluster arrives. The incoming IP packets have, among other data, the source address, which identifies who has sent the packet, and the destination address, which is the IP address of the cluster. For example, in the environment we built, an incoming IP packet has 9.24.104.186 as its source IP address, since this was the address of our TN3270 client machine, and 9.24.104.107 as its destination IP address, since this was the cluster address (see Table 1 on page 29).

When the Dispatcher is installed, all the incoming IP packets sent by end users to the cluster address first arrive at the Dispatcher machine, not at one of the TCP servers. This is because the Dispatcher's network interface, besides having its own unique IP address, has an alias to the cluster address. The TCP servers in the cluster also have an alias to the cluster address, but this is defined on the loopback interface. One other thing to keep in mind is that the Dispatcher runs at a low level in the machine operating system so it can directly intercept all the IP packets.

Each time a new connection is initiated by a client, the Dispatcher selects which TCP server in the cluster should receive the connection packet. Now the Dispatcher should be able to send that packet to the selected TCP server.

How can the Dispatcher do that? The answer is that the Dispatcher routes the packet based on the MAC address of the network adapter on the chosen TCP server.

Media Access Control (MAC) Address

Ethernet and token-ring devices require an adapter to physically attach to the LAN. This adapter must provide both physical and logical capabilities for the device. The adapter contains a unique 48-bit address, assigned to it during the manufacturer process, called Media Access Control (MAC). All the MAC addresses are assigned by the IEEE 802 committee. The IEEE provides the vendor building adapters with a range of MAC addresses to use for assigning adapters their unique 48-bit address so that no two adapters should ever have duplicate addresses.

Ethernet and token-ring require the MAC address for both the origin and the destination adapters when communicating over a LAN. Besides the IP address, the MAC address also must be known when sending data to a LAN-attached device.

So, let's assume that the Dispatcher has decided to send the connection IP packet to the TCP server that has the following attributes (see again Table 1 on page 29):

- The unique IP address 9.24.104.16 and host name Jupiter
- The cluster address 9.24.104.107 as the alias on the loopback interface
- Adapter MAC address equal to 400052000001.

The Dispatcher must now transmit the IP packet over the token-ring network (but Ethernet would be similar), so it must know the MAC address of the selected TCP server.

Address Resolution Protocol (ARP)

If the TCP server's MAC address is unknown to the Dispatcher, the same Dispatcher is able to discover it by issuing an Address Resolution Protocol (ARP) containing a broadcast LAN MAC address that will be read by all the network interfaces attached to the same LAN. This requests also specifies the IP address 9.24.104.16 of the TCP server. Each of the network adapters attached to the LAN will determine if its relative interface has been configured with the IP address 9.24.104.16. Only the network interface adapter of Jupiter will respond with its own MAC address, 400052000001.

Once the Dispatcher knows the TCP server's MAC address, it will be stored into the ARP cache to skip successive requests.

The original destination MAC address on the IP packet was that of the network interface on the Dispatcher machine itself. When the Dispatcher knows the MAC address of the selected TCP server Jupiter, it changes the original destination MAC address on the packet to the MAC address 400052000001. This packet will be now encapsulated in the token-ring frame and transmitted to the chosen TCP server Jupiter.

The Dispatcher then sets up a connection table entry to make sure that subsequent incoming IP packets for this client continue being forwarded to the same server.

When the TCP server Jupiter receives the packet, the information related to the MAC addresses are eliminated and the source and destination IP addresses are extracted.

The destination IP address is still the cluster address 9.24.104.107, but the TCP server Jupiter has its own IP address 9.24.104.16. Jupiter can accept that packet anyway, since the cluster address is configured as an alias on the Jupiter's loopback interface.

At this point, Jupiter sends a response to the client that originated the request. Now let's see what happens to the outgoing packets.

Once all the IP packets of the originating client's request are received, the TCP server Jupiter performs the standard TCP processing that is commonly performed by all TCP servers while responding to a client: it switches the IP source and destination addresses for the outgoing packets that form the response to the client. The source address in this case becomes the cluster address 9.24.104.107, while the destination address is now the client IP address 9.24.104.186. This operation has an important consequence: the balancing function is transparent both to the client and the clustered servers.

The destination IP address is the client's IP address, not the Dispatcher's. For this reason, the TCP server Jupiter can route the IP packets through its default router directly to the client, and all the outgoing packets do not pass back through the Dispatcher. There is no need to return using the original physical path and a separate high bandwidth connection can be used.

This is important, since in many cases, the volume of the outbound server-to-client traffic is substantially greater than the inbound traffic. For example, Hypertext Markup Language (HTML) pages and imbedded images are at least 10 times the size of the client URLs that requested them.

The source IP address in the outgoing packet sent by the TCP server shows as the cluster address, and not as the TCP server's IP address. The cluster

address was also the destination IP address in the IP packets sent by the client in its request, so the client is not able to understand the TCP architecture of the target server. This security feature offered by the load balancing component ensures privacy for a site that is composed of multiple TCP servers load balanced by a Dispatcher machine.

Because the Dispatcher does not participate in bidirectional communications with the client but simply forwards the incoming packets unchanged, its presence is transparent to both client and server.

2.3 Interactive Session Support

You can use the ISS function with or without a DNS name server:

- If you are using ISS for load balancing, a DNS server is required. This can either be an actual DNS server or, if you set up a small, separate subdomain for a new name server, a replacement name server provided by ISS. Using this approach, ISS runs on a DNS machine. A client submits a request for resolutions of the DNS name for an ISS-associated service, which has been set up by an administrator. ISS then resolves the name to the IP address of a server in the cell, and forwards this IP address to the client.
- If you are using ISS to collect server load information, a DNS is not needed. The ISS monitor collects server load information from the ISS agents running on the individual servers and forwards it to the Dispatcher. The Dispatcher uses this load information, along with other sources of information, to perform load balancing.

ISS periodically monitors the level of activity on a group of servers and detects which server is the least heavily loaded. It can also detect a failed server and forward traffic around it. Once every monitoring period, ISS ensures that the information used by the DNS server or the Dispatcher accurately reflects the load on the servers. The load is a measure of how hard each server is working. The system administrator controls both the type of measurement used to measure the load and the length of the load monitoring period. You can configure ISS to suit your environment, taking into account such factors as frequency of access, the total number of users, and types of access (for example, short queries, long-running queries, or CPU-intensive loads).

All the nodes in a site work together to eliminate any single point of failure. Should the monitor machine fail, the survivors elect a new monitor to take over automatically.

ISS is described in detail in Chapter 5, “WebSphere Interactive Session Support” on page 75.

2.4 Summary

The following summary of the load balancing components will give you an idea of which configuration can help you in your networking environment.

2.4.1 Using the Dispatcher Executor Alone

Using the Dispatcher Executor component alone will give you a weighted round-robin load balancing scheme. No server code is required so it should work on a variety of platforms.

Weighted Round-Robin Algorithm

Weights are applied to all servers on a port. For any particular port, the requests will be distributed between servers based on their weights relative to each other. For example, if one server is set to a weight of 10, and the other to 5, the server set to 10 should get twice as many requests as the server set to 5.

A maximum weight can be set for each port, affecting the difference there can be between the number of requests a server can get. For example, at a maximum weight of 2, one server can get twice as many requests as another. At a maximum weight of 10, one server can get 10 times as many requests as another.

Things You Can Influence

- You can manually set the weight for the server.
- You can influence the round-robin mechanism by setting a maximum weight.
- You can tell the executor a server is down (this changes the weight to -1).

Drawbacks

The Executer is *not aware* of:

- Server status (up/down) unless the administrator sets the status to show it is down manually.
- Service status, for example TN3270 services (up/down).
- Connections to the server outside of the Dispatcher. For example, if 100 Telnet sessions are started to a server, the Dispatcher is not aware of that and still treats the server as an equal to the others. Counters are internal to Websphere.

Structure

- The Executor is active.
- A port exists for the service (for example, TCP/IP port 23 for Telnet or TN3270).
- Servers are defined to a port. The servers can be assigned a weight (the default is 10). The weights can be the same for a true round-robin balancing, or each server can be assigned a weight to determine how often it is used.
- The Executor is aware of the number of TCP connections that have been established to the port through the Dispatcher and how many are active. If the servers have different weights this is used to influence the selection of a server.

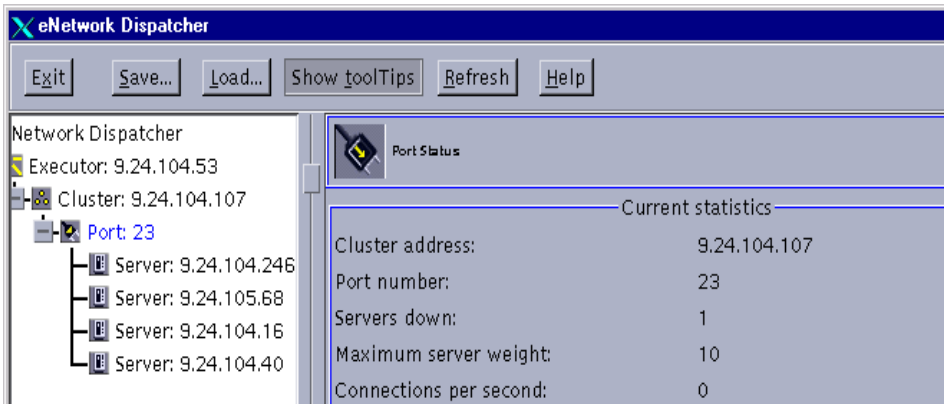


Figure 3. Weighted Round-Robin Using the Executor

2.4.2 Using the Dispatcher Executor with Advisor Input

Using the Dispatcher Executor and Advisor components will give you a weighted round-robin load balancing scheme. The difference in this and the previous situation (no advisors) is that the manager will be started and will use advisors to dynamically change the weight of the servers.

Round-Robin Algorithm

Same as “Using the Dispatcher Executor Alone” on page 25.

Things You Can Influence

- Proportions of importance: the amount of weight given to the active connections, new connections, advisor input, and ISS input.
- Smoothing index: defining how quickly the weights can change.
- You can use standard advisors or write your own customized advisor.

- WebSphere comes with a set of advisors. For advisors other than those shipped you must write or customize an advisor.

Structure

- The Executor is active.
- The Manager is active.
- A port exists for the service (for example, TCP/IP port 23 for Telnet or TN3270).
- Servers are defined to the port.
- An advisor for the service (for example, Telnet) is active.

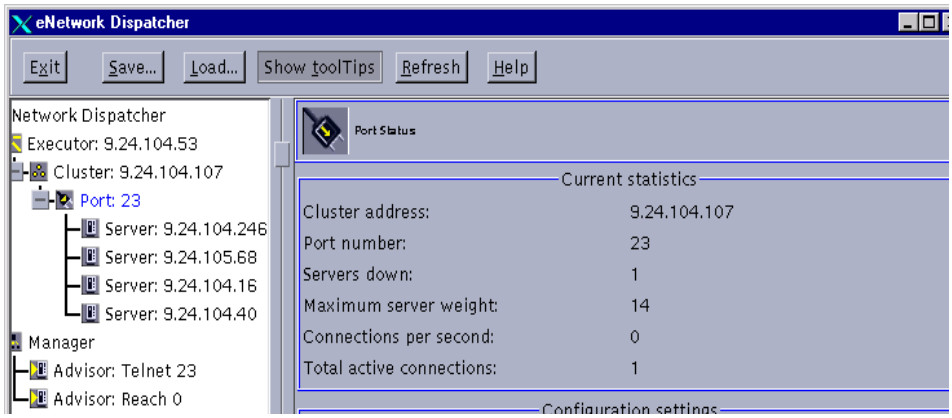


Figure 4. Weighted Round-Robin Using Advisors

2.4.3 Using the ISS Component

Using the ISS component allows you to use metrics for a more defined way of determining the preferred server. Metrics can be defined to specifically fit the environment, giving detailed information on server load and coming up with a metric to reflect that server's load. Servers that are not available are taken out of the selection list automatically.

Things You Can Influence

- You can use internal metrics or custom metrics to get a more detailed picture of the server load.
- You can choose round-robin or best method.
- You can use a domain name server to implement the load balancing.
  - With the domain name server, the traffic to and from the client will be direct, without any involvement from the load balancing machine.
  - Using a domain name server will cause some load on that machine because it has to be refreshed frequently.

- As an alternative to using a domain name server, you can send the load information collected by ISS back to the Dispatcher. The information can then be combined with advisor input by the Manager to weight the servers.

Structure

- The ISS component (issd daemon) runs on each server, both the monitoring server and the monitored. The ISS configuration file resides on each server.
- An observer is defined to get the load information. The observer can be the Dispatcher, or it can be a domain name server.
- The ISS component and Dispatcher are separate. You do not have to run the Dispatcher unless using it as an observer.

Chapter 3. Scenario 1: Using the WebSphere Dispatcher

In this section we will show you how to create a basic configuration for the Dispatcher to balance TN3270 and Telnet traffic among several servers. In this scenario, the only input to the Executor for balancing the load will be server weights set manually by the administrator. The balancing scheme is basically a round-robin approach, taking the server weights into account. This scenario is interesting because it allows you to balance the load among a variety of server platforms since there is minimal change to the server. It also allows you to use a user-defined port for TN3270 traffic.

For information on WebSphere Performance Pack installation, refer to Appendix B, “WebSphere Load Balancing Component Installation” on page 215.

3.1 Network Environment

A summary of the hardware, software and network configuration of the environment where we created this scenario is shown in the following table:

Table 1. Hardware, Software and Network Configuration

Workstation	Host Name	IP Address	Operating System	Service
IBM RS/6000	1) csaix1 2) csaix2	1) 9.24.104.53 2) 9.24.104.107	AIX 4.3.0	eNetwork Dispatcher
IBM RS/6000	rs600033	9.24.104.246	AIX 4.3.1	TN3270 Server
IBM RS/6000	Jupiter	9.24.104.16	AIX 4.3.1	TN3270 Server
IBM PC	wtr05147	9.24.104.40	Windows NT Server 4.0	TN3270 Server
IBM PC	wtr05142	9.24.104.186	UnixWare 7.0	TN3270 Server

On the Windows NT Server 4.0 we also installed Service Pack 4.

All of the above machines were provided with a token-ring interface and connected to the same LAN.

Notes:

1. The load balancing function is provided by the load balancing component (eNetwork Dispatcher) of IBM WebSphere Performance Pack Version 1.0.
2. Personal Communications V 4.2 and V 4.3 are the emulators used for the TN3270 sessions running on the clients.

3. The TN3270 Servers are provided by IBM eNetwork Communications Server for NT V 6.01, IBM eNetwork Communications Server for AIX V 5.0 and IBM eNetwork Communications Server for UnixWare 7.

The following figure shows the environment for this scenario:

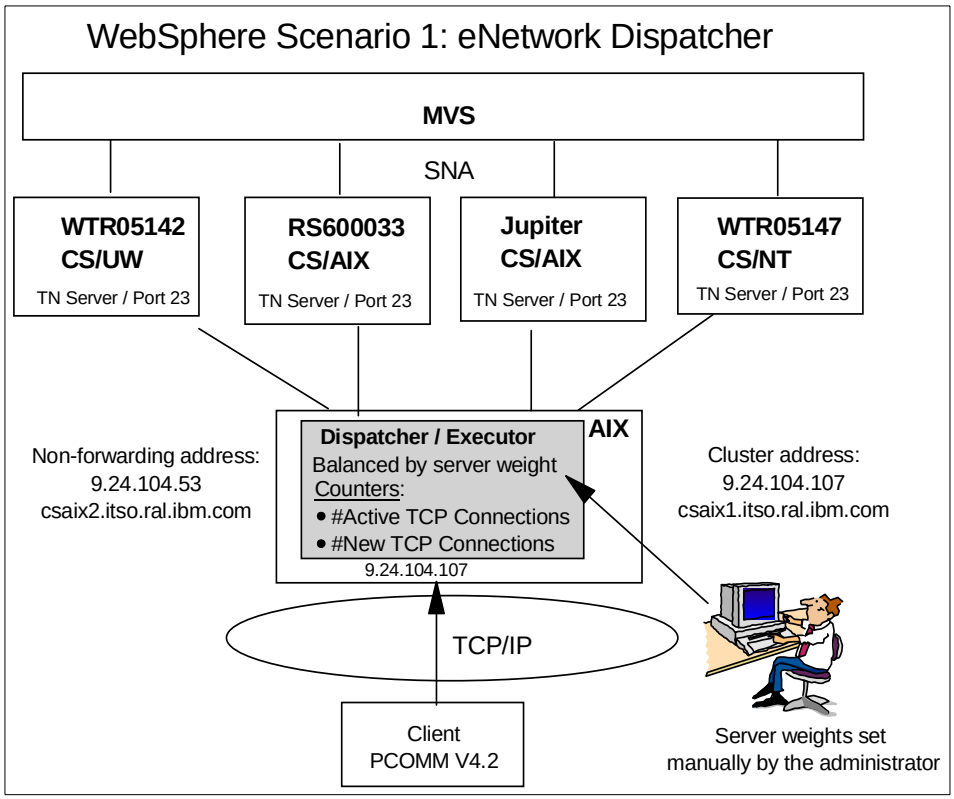


Figure 5. Graphical Representation of the Scenario Environment

The configuration process can be divided into three logical phases:

- The setup of the dispatcher machine
- The setup of the TN3270 Communication Server machines
- TCP/IP configuration modifications on each server

3.2 Cluster Address and Non-Forwarding Address

The first step in planning the environment is to assign two IP addresses to the dispatcher machine.

- The primary IP address for the Dispatcher machine is the IP address that is returned by the command:

  `host hostname`

  It is also called *non-forwarding address*.

  Through the non-forwarding address, you can connect to the machine for management purposes (using FTP or Telnet, for example). In our configuration, the non-forwarding address was 9.24.104.53. In fact, by issuing the command:

  ```
  host csaix1
  ```

  The output produced was:

  ```
  csaix1 is 9.24.104.53
  ```

- The *cluster address* is the IP address that will be used by clients to access the entire site. The dispatcher's job is to balance requests that come in to this address among the servers in the cluster. A host name can be associated to the cluster address. In our configuration, the cluster address of the Dispatcher was 9.24.104.107 and the associated host name was csaix2.itso.ral.ibm.com.

  Notice that you need to define a cluster address for each cluster you are going to define in your environment.

3.3 Dispatcher Configuration

In the following, we will refer to the configuration of the Dispatcher on the AIX platform. The configuration process on Windows NT is very similar; if something differs, we will explicitly mention it.

Three different methods can be adopted to configure the Dispatcher:

- Command line

  You can enter the configuration commands from a command prompt.

- Scripts

  You can write the commands into a configuration file script that acts as a batch file.

- Graphical user interface

  We found this very easy to use. The user on the Dispatcher machine is able to perform all the configuration steps strictly related to the dispatcher function by using the graphical user interface (GUI). In fact, by using the GUI, you can issue the `ndcontrol` command with the parameters and flags

that are used to configure the dispatcher machine, as we are going to explain. However, you cannot use the GUI for TCP/IP-related commands such as `ifconfig`, so this kind of command can be entered only from the command line.

In the following configuration steps, the GUI is usually the primary interface. Where appropriate, the command line configuration commands are also listed. To perform the Dispatcher configuration you must be the root user on AIX or, if the Dispatcher is installed on Windows NT, a system administrator.

3.3.1 Starting the Server Component

First, you need to start the server component of the Dispatcher.

3.3.1.1 Windows NT

In Windows NT the server component is a Windows NT service that starts automatically by default. The name of this service is IBM Network Dispatcher. If you want to stop or restart the IBM Network Dispatcher service, you should open the Services window of the Windows NT Control Panel, highlight **IBM Network Dispatcher** and then select **Stop** or **Start** respectively.

If you prefer that the IBM Network Dispatcher service does not start by default at each system reboot, select **IBM Network Dispatcher**, click on **Startup...**, and then set the Startup Type to **Manual**.

3.3.1.2 AIX

If on AIX, from a command line, enter:

```
ndserver start
```

If you enter the `ps` command after starting the server you can see that the server component is implemented as a Java class, running through the JVM java executable file.

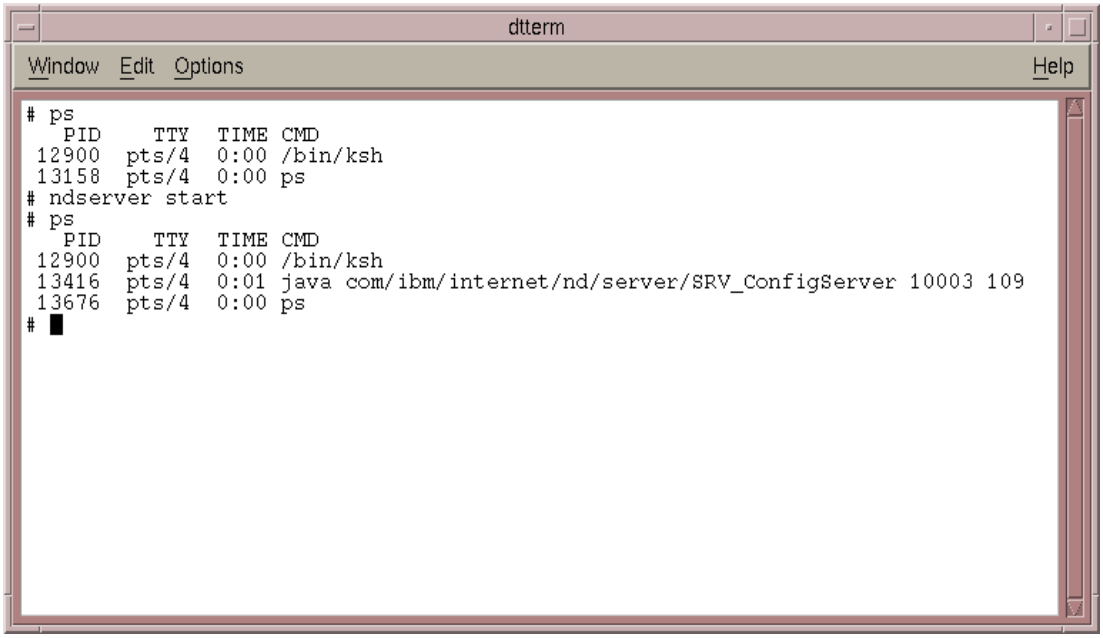


Figure 6. Server Component of the Dispatcher

3.3.2 Starting the GUI

Next you can start the GUI.

3.3.2.1 Windows NT

Click **Start**, select **Programs** and then click **Network Dispatcher**. The Network Dispatcher item in the Programs menu was created during the installation.

3.3.2.2 AIX

To start the GUI on an AIX system enter:

```
ndadmin
```

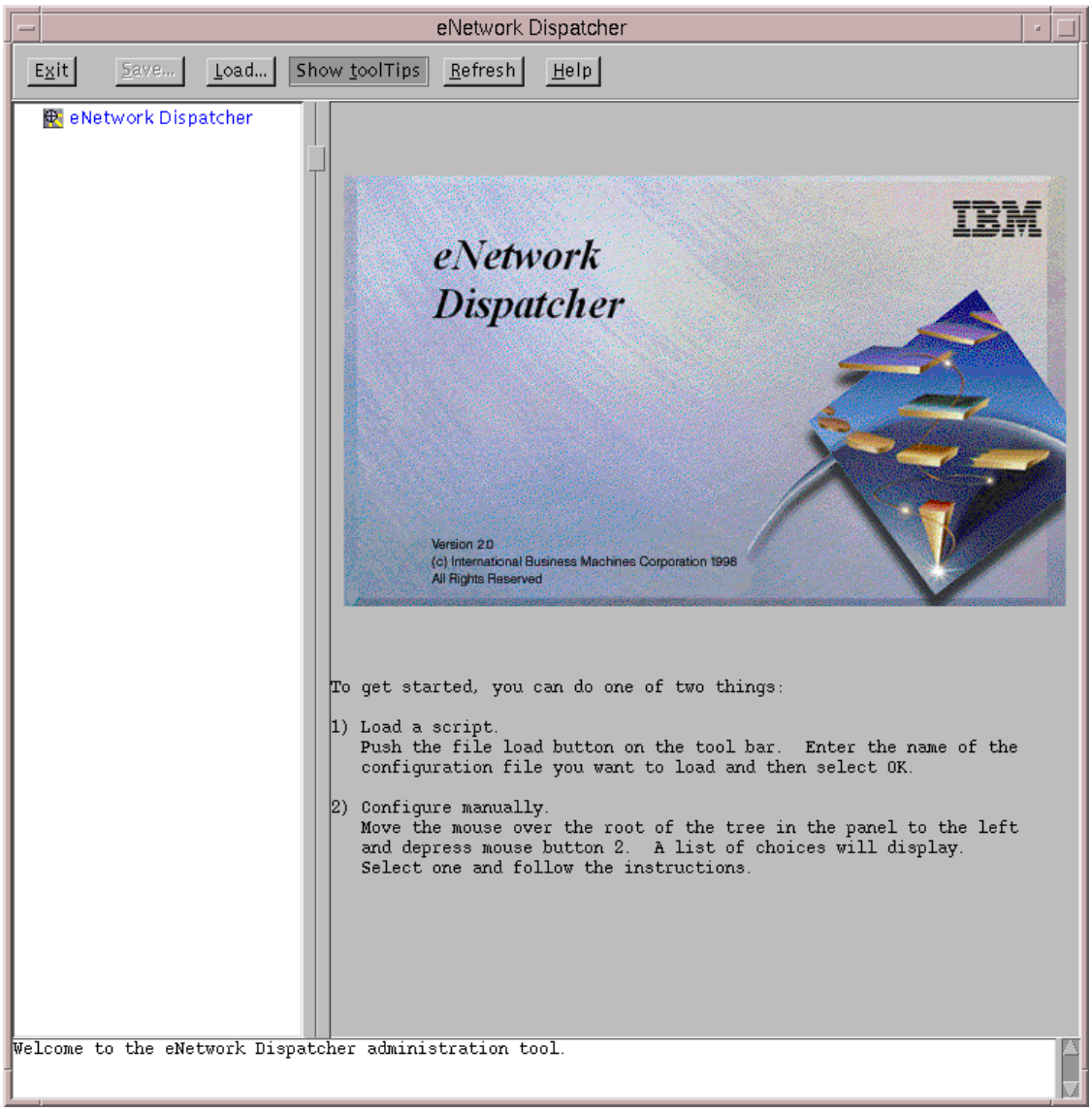


Figure 7. Graphical User Interface of the Dispatcher Configuration

The GUI works by using Java classes. On AIX, the window from which you opened the GUI will show Java messages. One of these messages may be a java.lang.NullPointerException, as shown in the following screen.

```
java.lang.NullPointerException
        at java.awt.LightweightDispatcher.retargetMouseEvent(Compiled Code)
        at java.awt.LightweightDispatcher.processMouseEvent(Compiled Code)
        at java.awt.LightweightDispatcher.dispatchEvent(Compiled Code)
        at java.awt.Container.dispatchEventImpl(CompiledCode)
        at java.awt.Window.dispatchEvemtImpl(Compiled Code)
        at java.awt.Component.dispatchEvent(Compiled Code)
        at java.awt.EventDispatchThread.run(Compiled Code)
```

This message is not a real problem and does not affect the Dispatcher configuration.

3.3.3 Starting the Executor

Once the server is started you can start the Executor component of the Dispatcher. The Executor routes requests to the TCP and UDP servers. It also monitors the number of new, active and finished connections and performs garbage collection of complete or reset connections. In this scenario, we are only using the Executor component. Later, when we modify this scenario and add the Manager component, the Executor will supply information about the new and active connections to the Manager.

To start the Executor, right click with your mouse in the left white area of the window shown in Figure 7 on page 34, and in the pop-up window that appears select **Start Executor**.

You will get a window similar to Figure 8 on page 36. This window can be used to monitor and configure the Executor.

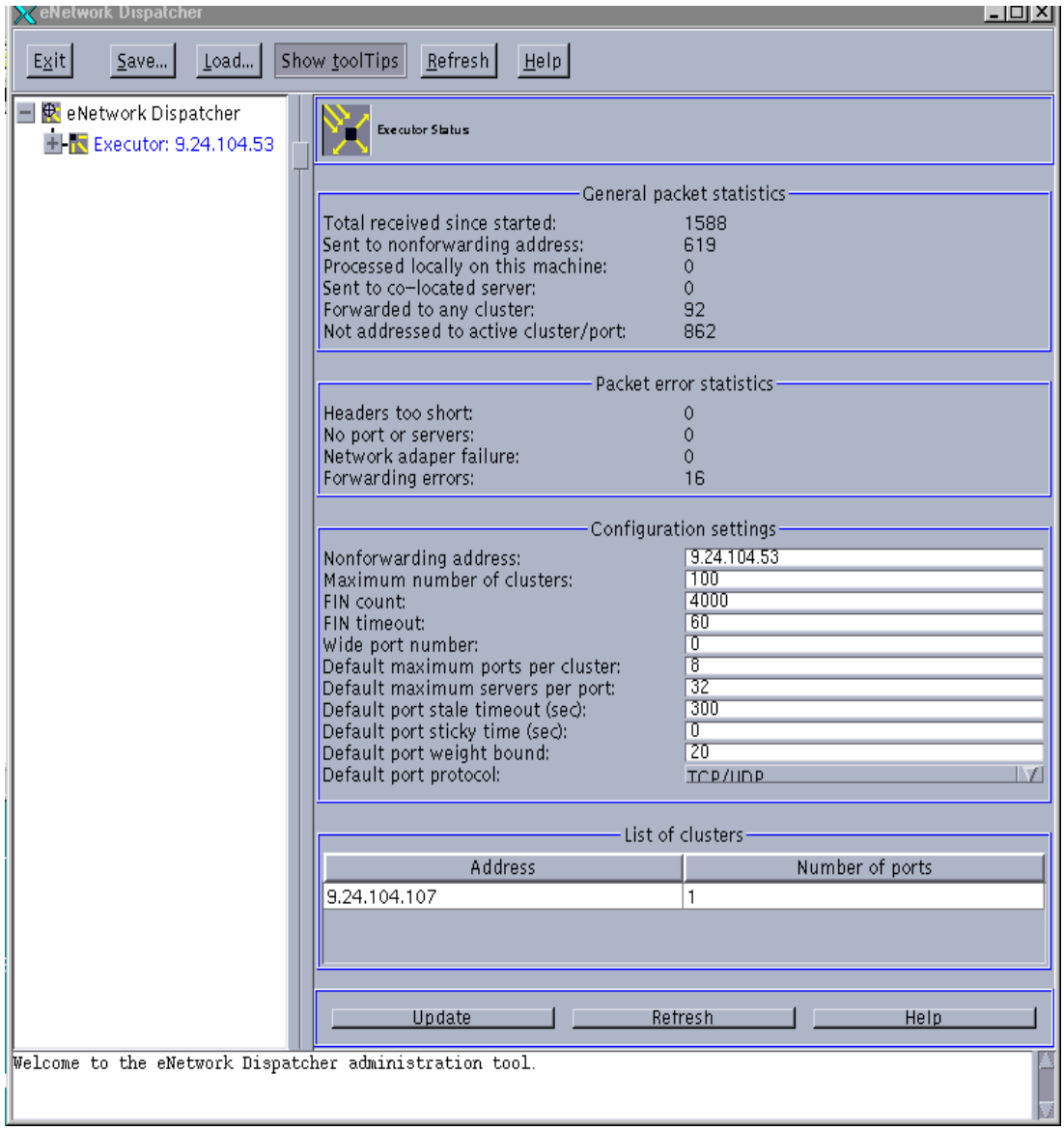


Figure 8. Executor Status Window

If you are not using the GUI, the command line to start the Executor is:

```
ndcontrol executor start
```

3.3.4 Configuring the Executor

At this point, you might want to redefine the non-forwarding address or the host name of the Dispatcher machine. It is displayed by default in the Configuration settings section of the Executor Status window (see Figure 8 on page 36). In our configuration the non-forwarding address appeared as 9.24.104.53, which was the correct address.

Using the Command Line

To set the non-forwarding address from the command line, you can enter the following command:

```
ndcontrol executor set nfa IP_address
```

and put the non-forwarding address in place of *IP_address*. For example, on our platform, the full command should have been:

```
ndcontrol executor set nfa 9.24.104.53
```

More About the Non-Forwarding Address

The non-forwarding address (NFA) has a special meaning inside the kernel code. Any packets that have the non-forwarding address as the destination address are immediately given back to the local operating system for processing. No Dispatcher logic is used to determine where the packet should go. This check is done immediately after receiving the packet.

Notice that no error checking is done to see that the non-forwarding address is an existing IP address on that machine; so you could even set the non-forwarding address to an IP address that has not been defined on that machine, and your command would be accepted, even if you could not administer the Dispatcher machine using such an address.

Another use of the NFA is for co-located servers. If the Dispatcher logic determines that the destination server has the non-forwarding address as its address, the packet is given back to the underlying operating system, not routed out the network interface. This implies that in order for co-location to work, the server on the local machine must be added with the non-forwarding address.

The Executor configuration section (see Figure 8 on page 36) lists a set of parameters you can change. Some of parameters are easy to interpret, such

as the Maximum number of clusters (default is 100), Default maximum ports per cluster (default is 8) and the Default maximum servers per port (default is 32). Others are more difficult to understand, such as the FIN count and the FIN timeout.

FIN is a control bit occupying one sequence number, which indicates that the sender will send no more data. A client sends a FIN after it has sent all its packets, so that the server will know that the transaction is finished. When the Dispatcher receives a FIN, it marks the transaction from active state to FIN state, and at this point the Dispatcher garbage collector can clear the memory reserved for those connections.

In general, you can safely take the default values for these settings. For detailed information about them, you may consult the eNetwork Dispatcher help, which can be accessed by clicking the **Help** button on the top menu bar, or see the *eNetwork Dispatcher for Solaris, Windows NT, and AIX User's Guide Version 2*, GC31-8496, shipped with the CD-ROM of IBM WebSphere Performance Pack.

3.3.4.1 Alias the Network Interface to the Cluster Address

For the Dispatcher to route packets, each cluster address must be aliased to a network interface card. This is actually a TCP/IP function. You cannot use the GUI to create the alias. The reason for this is explained in "The Flow of the IP Packets" on page 20.

AIX

If your platform is AIX, you should use the TCP/IP command `ifconfig`. The full command syntax on AIX is:

`ifconfig` *`interface_name`* `alias` *`cluster_address`* `netmask` *`netmask`*

where:

- *`interface_name`* is the name assigned to the network interface.

  On our platform, the network interface name was tr0, since our machine was equipped with a token-ring network adapter.

- *`cluster_address`* is the cluster address assigned to the Dispatcher machine.

  In our case we entered `9.24.104.107`, (see Table 1 on page 29).

- *`netmask`* is the netmask value you are using in the TCP/IP configuration of the Dispatcher machine. It can be specified either in dotted-decimal or hexadecimal form.

  In our case we could have entered either `255.255.255.0` or `0xffffff00`.

Windows NT

If your platform is Windows NT, you should use the `ndconfig` command. Actually, this command does not belong to the set of commands related to the TCP/IP protocol on the Windows NT operating system. However, during the installation of the Load Balancing component of IBM WebSphere Performance Pack, the installation process puts ndconfig.exe executable in the directory D:\Program Files\eND\Dispatcher\BIN and updates the Path system environment variable with this directory.

The syntax of the `ndconfig` command is the same as the AIX `ifconfig` command.

To determine the value of the netmask on AIX you can enter the `ifconfig` command followed by the name of the network interface. If the platform is Windows NT, `ifconfig` should be replaced by `ndconfig`.

For example, on the Dispatcher machine that we installed on AIX, we entered the command:

```
ifconfig tr0
```

The output we received is displayed in the following screen:

```
csaix1:/ > ifconfig tr0
tr0: flags=e0a0043<UP,BROADCAST,RUNNING,ALLCAST,MULTICAST,GROUPRT,64BIT>
        inet 9.24.104.53 netmask 0xffffff00 broadcast 9.24.104.255
csaix1:/ >
```

Figure 9. ifconfig Command Entered to Discover the netmask Value

So, on our platform, the full command to create the alias on the network interface tr0 to the cluster address 9.24.104.107 was:

```
ifconfig tr0 alias 9.24.104.107 netmask 255.255.255.0
```

To ensure that the alias has been created correctly, enter the command

```
netstat -ni
```

You should get two IP addresses for your network interface. The following figures show the output we got from the `netstat -ni` command before and after we created an alias on the Dispatcher network interface using the `ifconfig` command:

```
csaix1:/ > netstat -ni
Name  Mtu   Network     Address             Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 link#1                        127628     0   127628     0     0
lo0   16896 127         127.0.0.1         127628     0   127628     0     0
lo0   16896 ::1                           127628     0   127628     0     0
tr0   1492  link#2      0.4.ac.63.31.b8   5168569     0   120157     0     0
tr0   1492  9.24.104    9.24.104.53       5168569     0   120157     0     0
```

Figure 10. netstat before Creating the Alias

```
csaix1:/ > netstat -ni
Name  Mtu   Network     Address             Ipkts Ierrs    Opkts Oerrs  Coll
lo0   16896 link#1                      127628     0   127628     0     0
lo0   16896 127         127.0.0.1        127628     0   127628     0     0
lo0   16896 ::1                         127628     0   127628     0     0
tr0   1492  link#2      0.4.ac.63.31.b8  5168569     0   120157     0     0
tr0   1492  9.24.104    9.24.104.53      5168569     0   120157     0     0
tr0   1492  9.24.104    9.24.104.107     5168569     0   120157     0     0
```

There are now two network addresses for the tr0 interface

Figure 11. After the Alias Has Been Created (AIX)

On Windows you would use the `route print` command to see the results.

Cluster Addresses and Network Interfaces

Note that it is possible to configure the Dispatcher to manage more than one cluster, no matter if your Dispatcher machine is shipped with only one or more network interfaces card. Let's see some examples.

- If your Dispatcher machine has only one network interface adapter, you can configure as many aliases as the numbers of clusters you want to define on the same network interface card.
- If your Dispatcher machine is shipped with two network interface cards (for example, one Ethernet card, en0, and one token-ring card, tr0), and you want to manage two clusters, one on en0 and the second on tr0, then each cluster address must be aliased on the corresponding network interface card.
- One more example is if your Dispatcher machine is shipped with two network interface cards (one Ethernet card, en0, and one token-ring card, tr0), and you want to manage five clusters, say three on en0 and two on tr0. Then the first three cluster addresses should be aliased on tr0, and the other two cluster addresses on en0.

No limits are known to the number of cluster addresses that can be aliased on the same network interface card.

3.3.4.2 Alias Configuration: Using Scripts

Each time the Executor is started, you will have to reconfigure the aliases for the network interfaces of the dispatcher machine. To avoid having to manually configure the network interface every time, you can use one or more script files, that are automatically launched by the Dispatcher as

different conditions occur, or, using more appropriate words, as the Dispatcher changes its own *state*.

For example, when using the high availability feature and the Dispatcher begins to monitor the conditions of the currently active Dispatcher without routing any packets, it is said that the Dispatcher goes into standby state. Another example is after the Dispatcher finishes monitoring the health of the currently active machine: then it changes its state, goes into the active state and begins routing packets. Every time the Dispatcher changes its own state, a corresponding script is launched. For example, when the Dispatcher goes into the idle state, the related script that is launched is called *goIdle*. The Dispatcher goes into the idle state when it begins routing packets in a stand-alone configuration.

Sample versions of script files that accomplish these functions are located in the eND sample directory. On our systems, the eND sample directory was /usr/lpp/eND/dispatcher/bin/samples/en_US on AIX, and D:\Program Files\eND\dispatcher\BIN\samples\en_US on Windows NT.

As suggested in the publication *eNetwork Dispatcher for Solaris, Windows NT, and AIX User's Guide Version 2*, we used the goIdle.sample file that is particularly indicated in case you are not implementing the high availability feature for the Dispatcher configuration.

We made a backup copy of that file, then we modified the original one by inserting into it the correct command line to create an alias on the network interface to the cluster address as shown in the following figure:

```
#!/bin/ksh
#
#goIdle script
#
#This script is executed when a Network Dispatcher goes into
#the 'idle' state and begins routing packets.  This occurs when
#the high availability features have not been added, as in a
#standalone configuration.  It also occurs in a high availability
#configuration before the high availability features have been
#added or after they have been removed.  goIdle should not be
#used in conjunction with high availability, it's intended
#purposes were for a standalone environment.
#
# This script must be placed in the /usr/lpp/eND/dispatcher/bin
# directory in order to be executed
#
ifconfig tr0 alias 9.24.104.107 netmask 255.255.255.0
```

Figure 12. Script File goIdle.sample on AIX

In order for the above script to run, you are required to name the file as goIdle (without any extensions) and place it in the Dispatcher bin directory, which by default is /usr/lpp/eND/dispatcher/bin directory on AIX and D:\Program Files\eND\dispatcher\BIN on Windows NT.

3.3.4.3 Adding a Cluster

The next step is to define a cluster. From the window shown in Figure 8 on page 36, right click the left panel (on the **Executor** item or below it), and in the pop-up window that appears select **Add Cluster...**.

Then type the cluster address and click **OK** in the next panel.

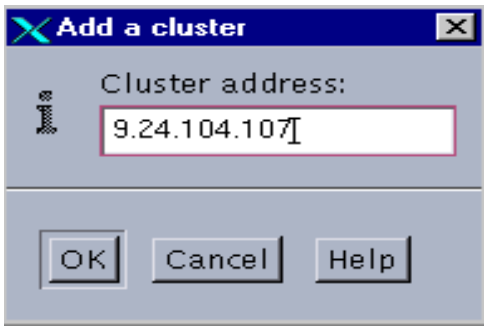


Figure 13. Enter the Cluster Address

Using the Command Line

If instead of the GUI you selected to use the command line, then you should enter:

```
ndcontrol executor set nfa cluster_address
```

In our case the command would have been:

```
ndcontrol executor set nfa 9.24.104.107
```

You will get the following Cluster status window, which confirms that a cluster has been added to the configuration:

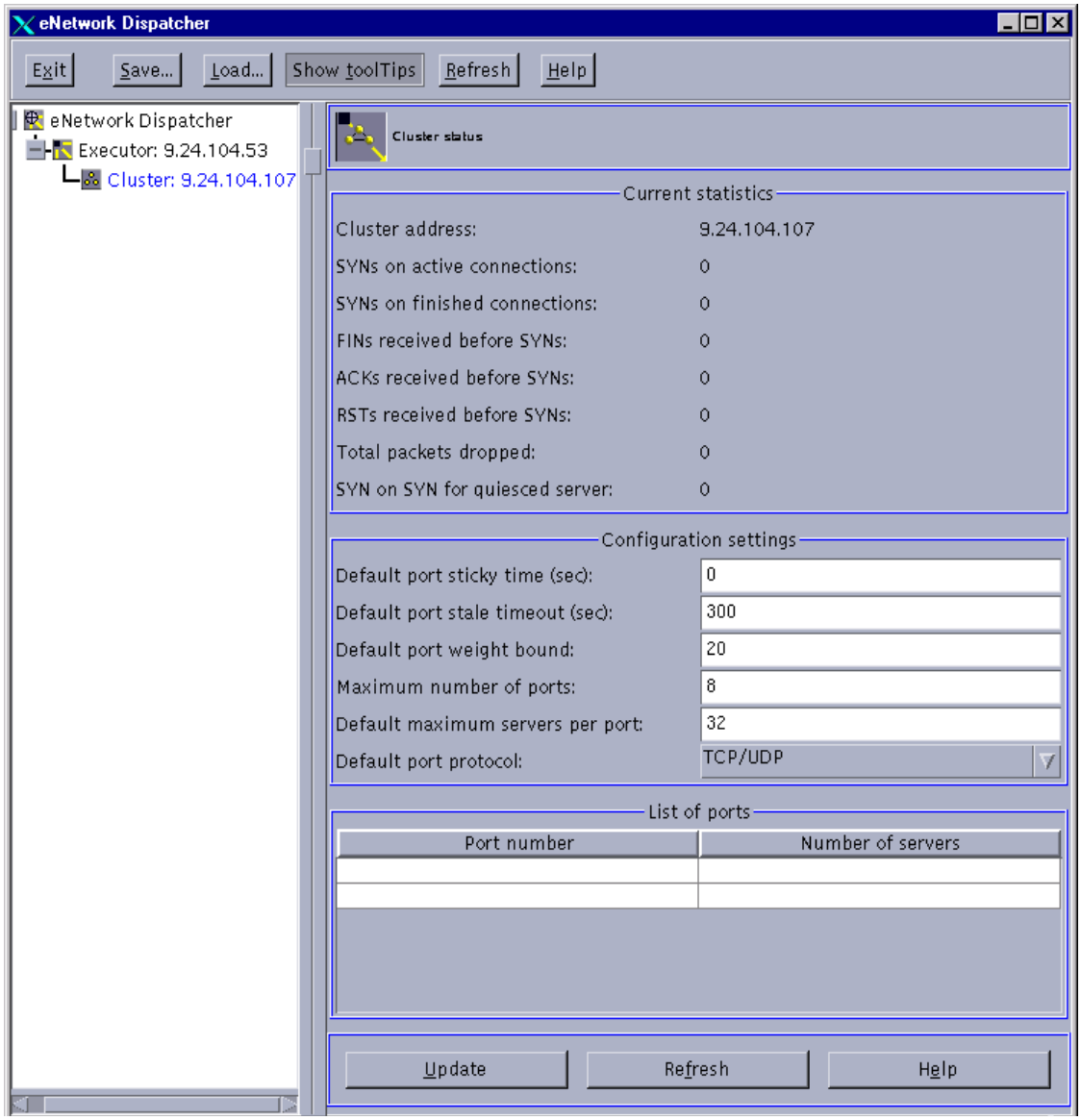


Figure 14. Cluster Added

3.3.4.4 Adding Ports

Now you need to define the ports that will be providing the services to be balanced among servers.

The port defined can be any valid TCP/IP port. If you want to define a port other than 23 for TN3270 functions, you may do so. The only drawback might be that there is no advisor provided and you will have to write your own if you want to use advisor input. For more information on advisors see “Activating the Advisors” on page 68.

To add ports from the GUI, right click the left panel (on the **Cluster** item or below it in Figure 14 on page 45), and in the pop-up window that appears select **Add Port...**.

Then type the port number you want and click **OK** in the panel that is displayed:

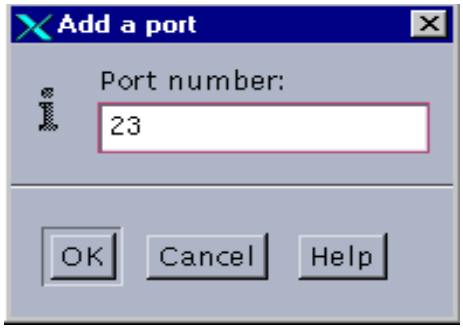


Figure 15. Enter the Port Number

We entered 23, as shown in Figure 15. If we wanted to use the command line interface the command would have been:

```
ndcontrol port add 9.24.104.107:23
```

Looking at the GUI after clicking the **Refresh** button in the top menu bar, we got the updated configuration with the port that was added.

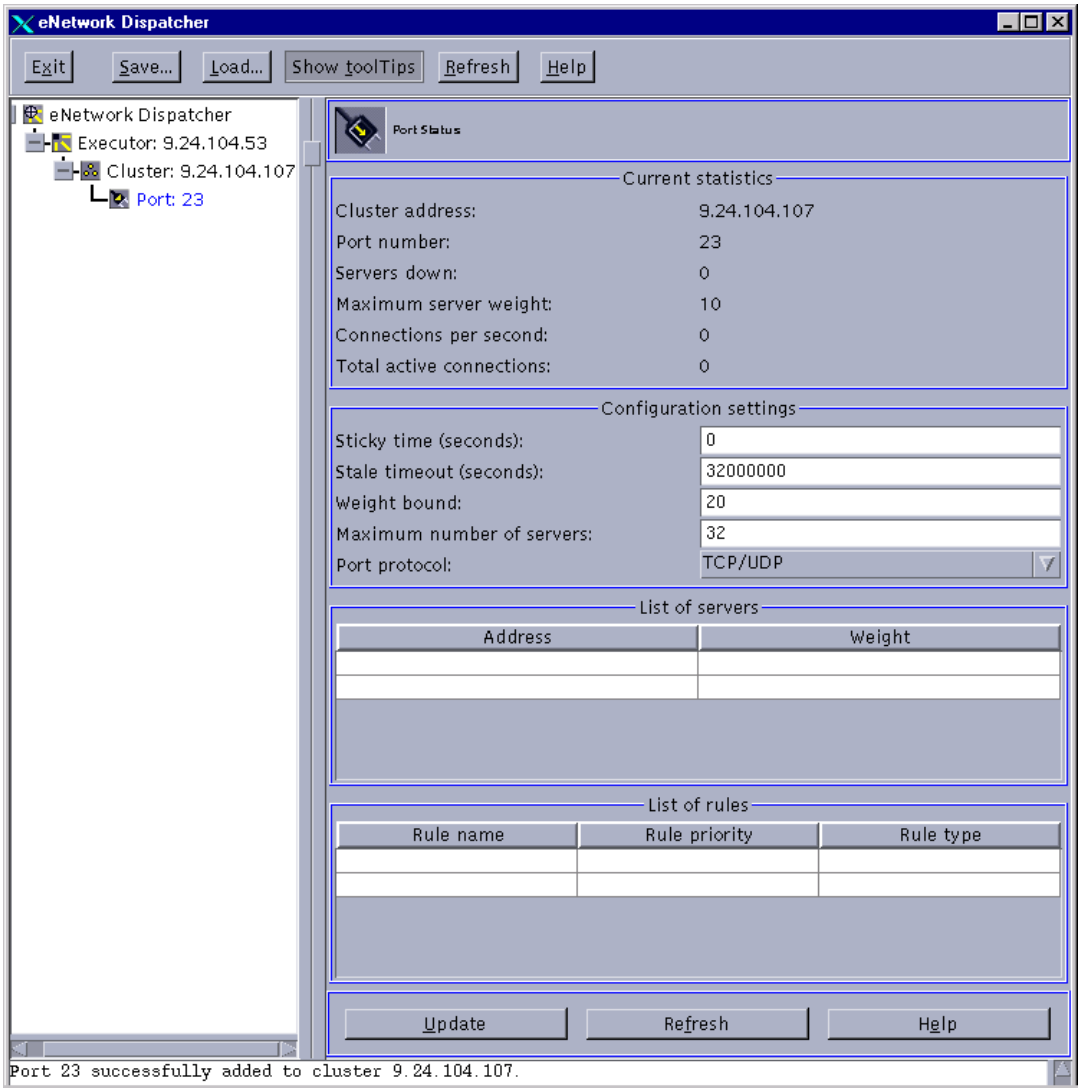


Figure 16. Adding a Port

Multiple Ports Using the Command Line

Note that by using the command line you are able to add multiple ports entering just one command. An example would look like:

```
ndcontrol port add 9.24.104.107:20+21+80+443
```

3.3.4.5 Adding Servers

At this point of the configuration process you need to define the TCP/IP server machines that you want in the cluster. To do so, we right clicked the **Port: 23** item in Figure 16 on page 47 and chose **Add Server...**.

Type the server name and click **OK** in the panel that appears:

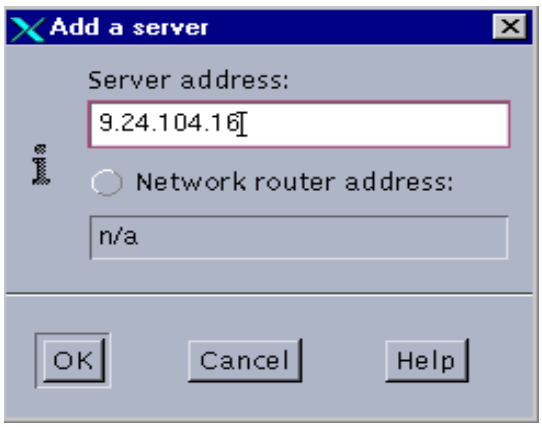


Figure 17. Enter Server Address

You can optionally click the **Network router address** radio button and fill in the network router address if you are defining a Dispatcher for a wide area network (WAN) configuration.

After clicking **OK**, you will return to the GUI. Expanding the **Port: 23** item, we ensured that the server having IP address 9.24.104.16 had been added to the port 23 of the cluster having IP address 9.24.104.107. The following window shows the port status after we added all our servers.

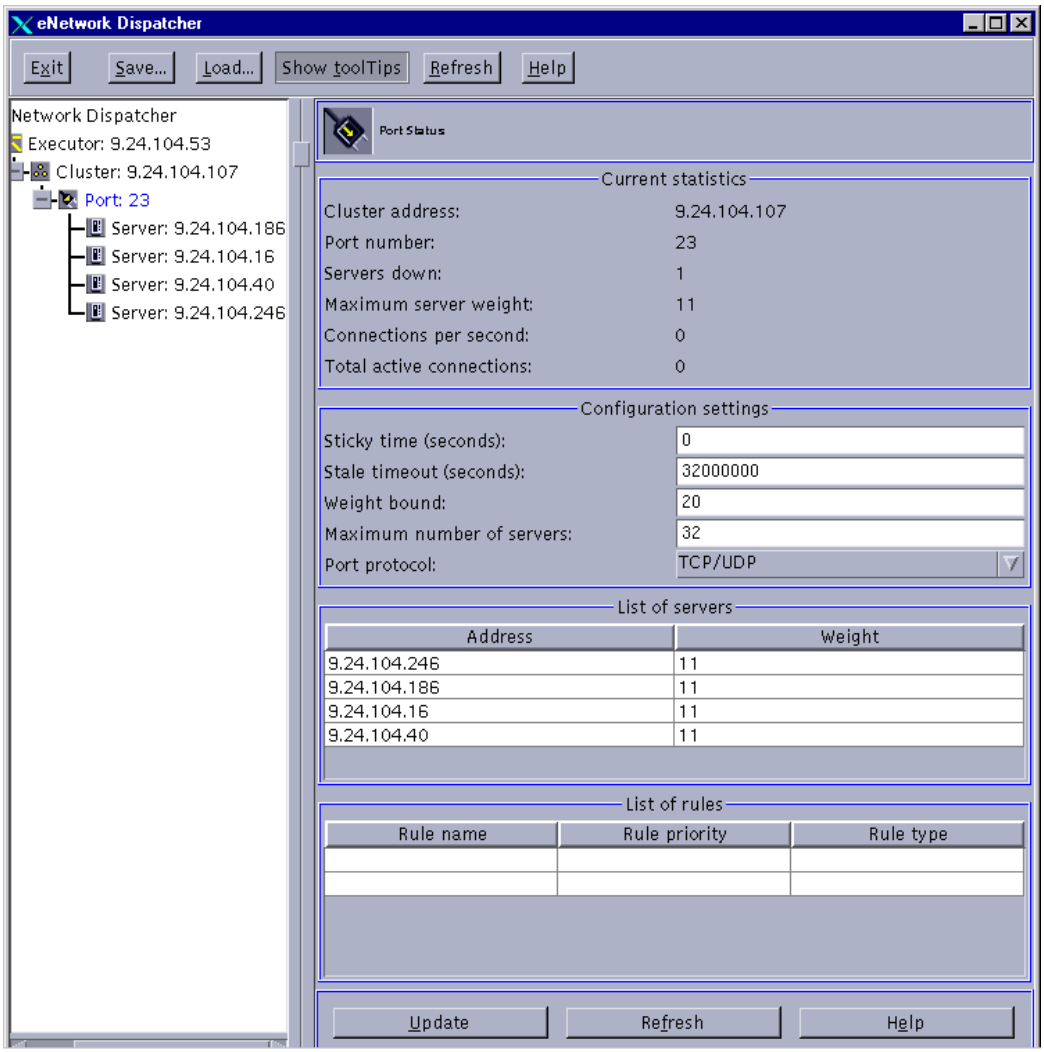


Figure 18. Servers

To add the servers through the command line, the command would have looked like this:

```
ndcontrol server add
9.24.104.107:23:9.24.104.16+9.24.104.246+9.24.104.40+9.24.104.186
```

Returning to the GUI, click the **Refresh** button in the top menu bar and the configuration will be updated.

At this point in the configuration, the Dispatcher machine is capable of performing load balancing using *weighted round-robin scheduling*, which is based on the current servers' weights.

In general, weights are set by the Manager component of the eNetwork Dispatcher machine. The Manager component decides each single weight basing its decision on internal counters in the Executor, feedback from the advisors and feedback from a system-monitoring program, such as ISS. Notice that weights are applied to all servers on a given port.

In this particular configuration, we wanted to set weights manually, and so we did not run the Manager component. We accepted the default values for weights: as you can see in Figure 18 on page 49, each server has weight 11 on the Telnet port 23. In this case, since all the servers would be balanced with the same weight, the weights would not impact the standard round-robin mechanism, and we had actually implemented a simple non-weighted round-robin.

It is important to mention that the Weight bound field, visible in Figure 18 on page 49, can be used to set how much difference there can be between the number of requests each server will get. We did not modify the default value, 20, which means that one server could get 20 times as many requests as another.

3.4 TCP Server Configuration

Before the Dispatcher starts to forward TCP/IP connection requests to the TCP servers, it is necessary to set up the TCP server machines. On each server in the cluster, you must add an alias to the cluster address on the loopback interface, often called lo0. The loopback IP address is usually 127.0.0.1 and is never forwarded as a destination on the network media. The reason for this is explained in "The Flow of the IP Packets" on page 20.

OS/2 and HP-UX Operating Systems

If you have a server running on an operating system that does not support aliases, such as HP-UX and OS/2, you must *set* the loopback device to the cluster address.

Other operating systems, such as Windows NT, AIX and Solaris, support aliases, and so you can follow the procedure we are indicating now.

The Dispatcher does not change the destination IP address in the TCP/IP packet before forwarding the packet to a TCP server machine. By setting or aliasing the loopback device to the cluster address, the TCP server machine will accept a packet that was addressed to the cluster address.

For more information about the flow of the incoming and outgoing IP packets see “The Flow of the IP Packets” on page 20.

3.4.0.1 Aliasing the Loopback Device on AIX

To alias the loopback device on an AIX machine, use the following command syntax:

```
ifconfig lo0 alias cluster_address netmask netmask
```

For example, on our AIX Web server machine (see Table 1 on page 29), we entered the command:

```
ifconfig lo0 alias 9.24.104.107 netmask 255.255.255.0
```

You can use the `netstat -ni` command to check the results.

Note to UnixWare 7 Users

The `ifconfig` command and `netstat -ni` command are also the commands you would use to alias the loopback device on UnixWare 7.

You must alias the loopback device each time your AIX machine reboots, so it is recommended that you put the previous command in a script and instruct the machine to run it at reboot time. What we did in our case was to put the previous command into the file /etc/rc.tcpip. This file contains commands that start TCP/IP daemons (`sendmail`, `inetd`, etc.) on an AIX machine, and it is launched at each reboot. We added the following two lines at the end of that file.

```
#Add alias on the loopback device: it is needed when using with eND
ifconfig lo0 alias 9.24.104.107 netmask 255.255.255.0
```

3.4.0.2 Aliasing the Loopback Device on Windows NT

On a Windows NT system you first add and then configure the loopback device.

Click **Start** then **Settings**, and select **Control Panel**. Double-click the **Network** icon, and in the Network window click the **Adapters** tab. Then click the **Add...** button and from the list of adapters select **MS Loopback Adapter**, as shown in the following window:

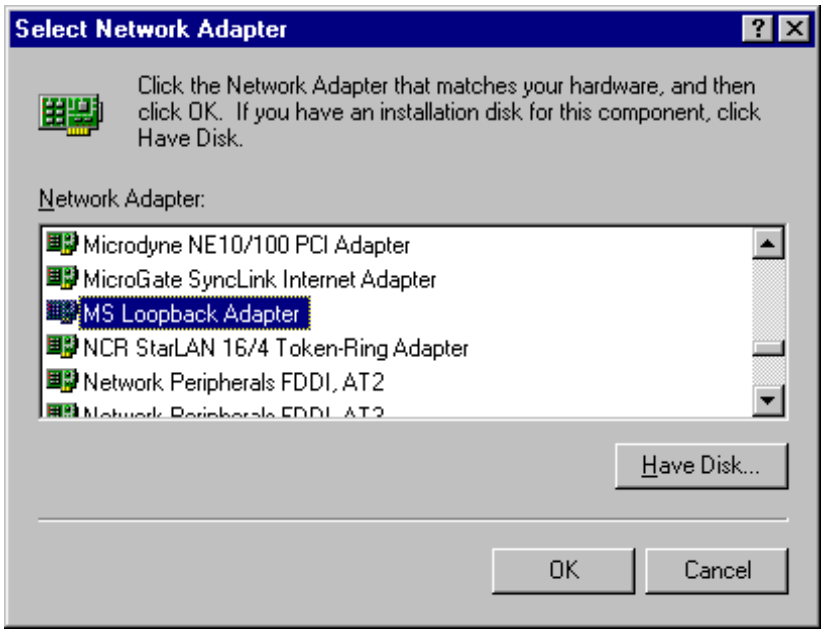


Figure 19. MS Loopback Adapter

Click **OK** and you will get the following screen:

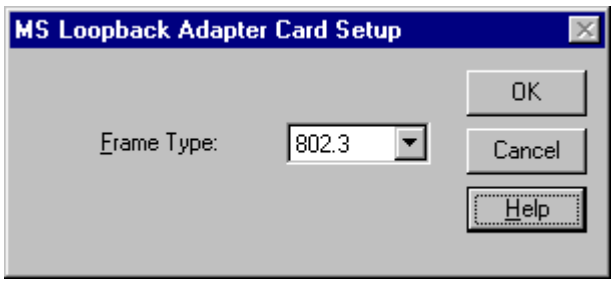


Figure 20. MS Loopback Adapter Card Setup

We accepted the default configuration, clicked **OK** and, when prompted, inserted the installation Windows NT CD-ROM.

After these simple steps, you will see another adapter added in your Windows NT machine adapter list:

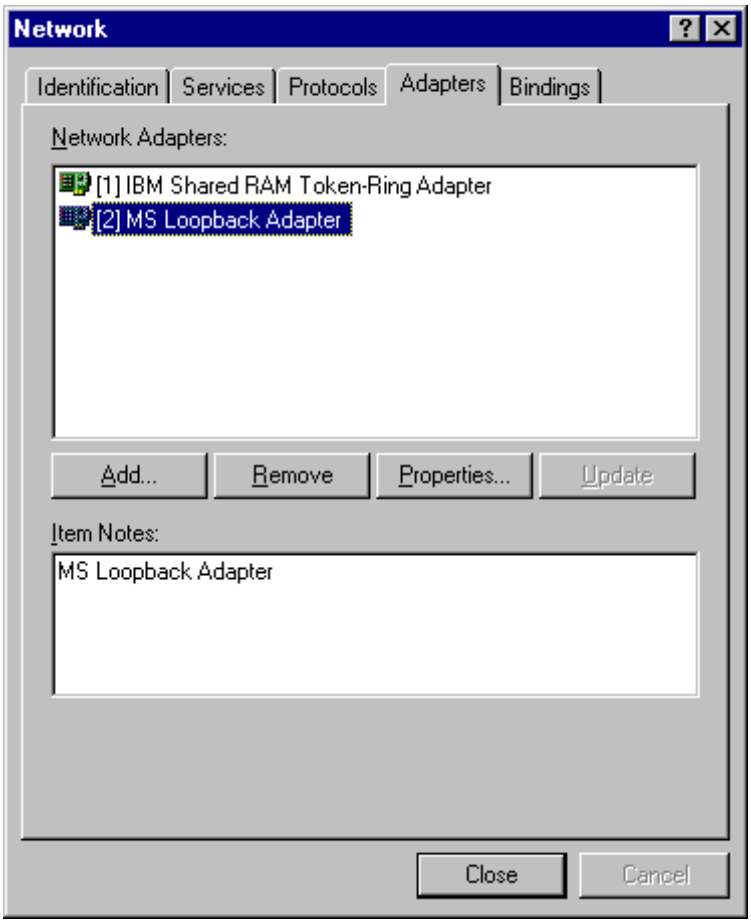


Figure 21. Windows NT Adapter List

Next, select the **Protocols** tab and double-click the **TCP/IP Protocol** item. If you open the Adapter list box in the next panel, the MS Loopback Adapter should have been added to the list.

Note

If the MS Loopback Adapter does not appear in the Adapter list box under the TCP/IP protocol configuration window, this is because the Network panel needs to be refreshed. You need to close and re-open it to see it updated with the new adapter.

Select **MS Loopback Adapter** from the Adapter list box and set the IP Address to the cluster address. We accepted the default subnet mask 255.0.0.0 and did not enter any gateway address, as shown in the following figure.

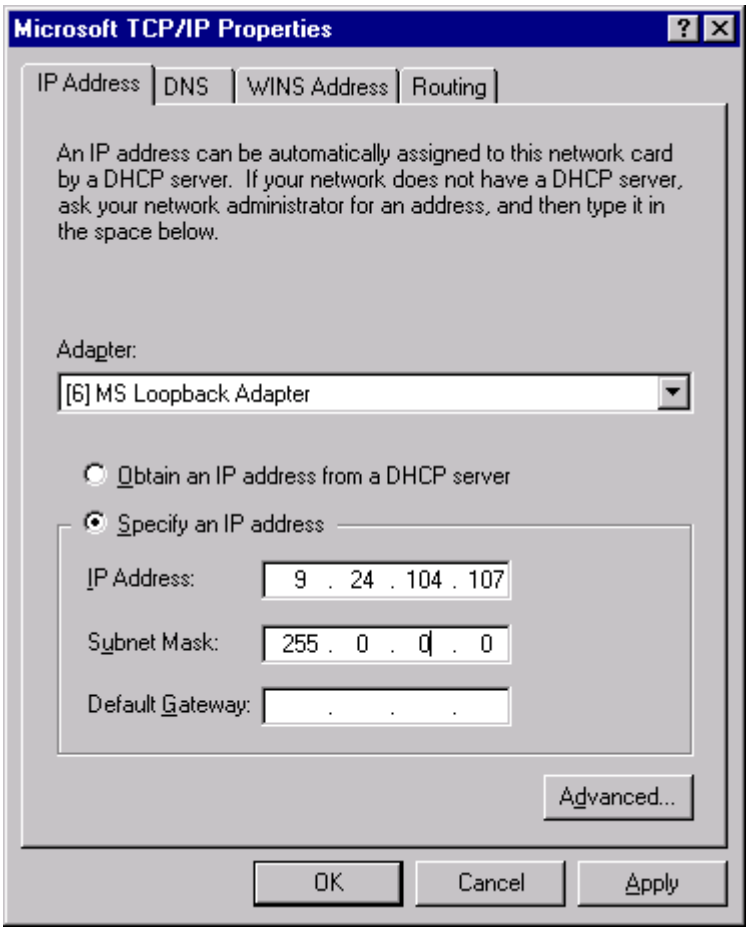


Figure 22. MS Loopback Adapter Configuration

Upon clicking the **OK** button, you will asked to reboot your system to enable the new adapter.

These definitions are permanent and will survive a reboot. However, we have found that there is an extra route created at each reboot that you need to delete.

To check for an extra route, from a command prompt enter the command

```
route print
```

A table showing all routes will be displayed.

In order to find the extra route, you should look for your cluster address under the Gateway Address column. If the cluster address appears twice, it means you have an extra route.

You need only one of those two entries. One of them is extra, and must be removed. To understand which route is extra, follow this simple rule: the extra route is the one whose address, in the Network Address column, begins with the first number of the cluster address, followed by three zeros. In our case it looked like the highlighted route in the figure below:

```
C:\WINNT\system32>route print
Active Routes:
 Network Address          Netmask  Gateway Address       Interface  Metric
         0.0.0.0          0.0.0.0      9.24.104.1     9.24.104.40       1
         9.0.0.0        255.0.0.0    9.24.104.107    9.24.104.107       1
      9.24.104.0    255.255.255.0     9.24.104.40     9.24.104.40       1
    9.24.104.107  255.255.255.255       127.0.0.1       127.0.0.1       1
    9.24.104.151  255.255.255.255       127.0.0.1       127.0.0.1       1
   9.255.255.255  255.255.255.255     9.24.104.40     9.24.104.40       1
       127.0.0.0        255.0.0.0       127.0.0.1       127.0.0.1       1
       224.0.0.0        224.0.0.0    9.24.104.107    9.24.104.107       1
       224.0.0.0        224.0.0.0     9.24.104.40     9.24.104.40       1
 255.255.255.255  255.255.255.255     9.24.104.40     9.24.104.40       1
```

Figure 23. Extra Route

Note: The extra route will appear differently if you do not take the default subnet mask shown in Figure 22 on page 54. The subnet mask is ANDed with the cluster address. We suggest you always use a subnet mask of 255.0.0.0 for this route for simplicity. However, if you choose another subnet mask, for example, 255.255.255.0, the extra route would have looked like the one highlighted in the next figure.

```
C:\WINNT\system32>route print
Active Routes:
  Network Address       Netmask     Gateway Address     Interface      Metric
     0.0.0.0         0.0.0.0        9.24.104.1     9.24.104.40       1
   9.24.104.0   255.255.255.0     9.24.104.107    9.24.104.107      1
```

Figure 24. Extra Route

To delete the route, we entered the following command:

```
route delete 9.0.0.0  9.24.104.107
```

To see the final result you would use the `route print` command. The output looks like the following figure. Note there are two lines with the gateway address of 127.0.0.1. One network destination address is the IP address of this server, 9.24.104.40. The other is the cluster address, 9.24.104.107.

```
C:\WINNT\system32>route print
Active Routes:
  Network Address              Netmask   Gateway Address       Interface   Metric
        0.0.0.0              0.0.0.0       9.24.104.1     9.24.104.40       1
      9.24.104.0        255.255.255.0      9.24.104.40     9.24.104.40       1
  9.24.104.107    255.255.255.255      127.0.0.1      127.0.0.1      1
   9.24.104.40    255.255.255.255      127.0.0.1      127.0.0.1      1
```

Figure 25. Final Routes

In order to automatically remove the extra route at each system reboot, you can copy the mentioned command in a batch text file, named for example Routedel.bat, and place it into your Windows NT Startup folder.

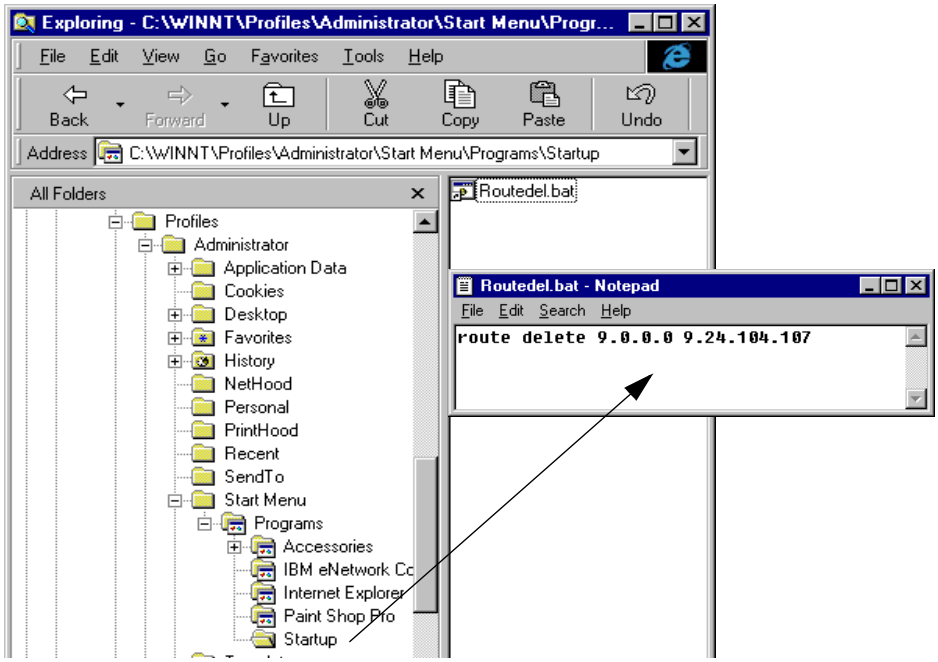


Figure 26. Automatically Deleting the Extra Route

3.5 TCP/IP Server Configuration Unique to Communications Server

In this scenario we worked with eNetwork Communications TN3270 Servers and TN3270 clients. Although you can modify the TN3270 Servers to use a user-defined port, we wanted to use the default port 23 for TN3270 so the clients would have minimal configuration changes. This has the additional advantage later of having an advisor available since we can use the Telnet advisor shipped with WebSphere. To do this we had to make sure that port 23, the well known port for Telnet, is shared with TN3270 on the AIX and UnixWare machines. On Windows NT you cannot share IP ports, so if we use TN3270 on port 23 it is not possible to do Telnet to that machine any more.

3.5.0.1 Sharing Port 23 on AIX

To share port 23 on AIX:

1. Stop CS/AIX or ensure that it is not started.
2. Edit /etc/inetd.conf and comment out the Telnet line by putting # at the beginning of the line.

```
ftp     stream  tcp6    nowait  root    /usr/sbin/ftpd          ftpd
#telnet  stream  tcp6    nowait  root    /usr/sbin/telnetd       telnetd
shell   stream  tcp6    nowait  root    /usr/sbin/rshd          rshd
kshell  stream  tcp     nowait  root    /usr/sbin/krshd         krshd
```

Figure 27. The /etc/inetd.conf Modified File

3. Create an ASCII file called /etc/snainetd.conf and put the telnetd server program and path (from the commented lines in the previous step) in it.

```
/usr/sbin/telnetd       telnetd
```

Figure 28. The /etc/snainetd.conf File

4. Get the inetd process ID and kill it.
5. Start the SNA internet daemons by running the `snainetd` command.
6. Start the inetd daemon by entering `inetd`.
7. Start CS/AIX.

Steps 4 through 7 will have to be done each time the machine is rebooted. This can be done by calling a script file in /etc/inittab to run before CS/AIX is started. The next figure shows an example script:

```
#!/bin/ksh
A=‘ps -eaf|grep snainetd|grep -v grep‘
if [ ! -n $A=0 ]
then exit
fi
kill ‘ps -eaf|grep inetd|grep -v grep|awk ’{print $2}’‘
snainetd
inetd
```

Figure 29. Script to Share TCP/IP Port 23 at Startup

Note: This script must be run before SNA is started.

3.5.0.2 Sharing Port 23 on UnixWare

We found that UnixWare requires only slight modification for CS/UnixWare to share TCP port 23 with the default Telnet server, similar in this case to AIX’s configuration. Some of the commands do differ slightly, but the intent is the same. To share port 23, take the following steps in UnixWare, with root level authority:

1. Stop CS/UnixWare or ensure that it is not running on the computer.
2. Edit the file named inetd.conf, found in the /etc subdirectory. Look for the lines beginning with "telnet". Note that there are two lines, most likely one having been commented out already and add another "#" to comment both so that it reads:

```
#
# Ftp and telnet are standard Internet services.
#
ftp     stream  tcp     nowait  root    /usr/sbin/in.tcpd       in.ftpd -a
#ftp    stream  tcp     nowait  root    /usr/sbin/in.ftpd       in.ftpd -a
#telnet stream  tcp     nowait  root    /usr/sbin/in.tcpd       in.telnetd
#telnet stream  tcp     nowait  root    /usr/sbin/in.telnetd    in.telnetd#
```

Figure 30. Excerpt from the /etc/inetd.conf File

3. Create an ASCII text file named /etc/snainetd.conf, which contains the command needed to load the Telnet daemon that was commented by step 2 above. It should read:

```
/usr/sbin/in.telnetd      in.telnetd
```

Figure 31. Sample SCO UnixWare /etc/snainetd.conf File

4. Using a utility that comes with SCO UnixWare called sacadm (service access controller administration), we can effectively unload and reload the Internet daemon (inetd) with the updated configuration files by issuing the following command:
   - `sacadm -k -p inetd`
5. Now start the CS/UnixWare Internet daemon:
   - `snainetd`
6. Restart the Internet daemon:
   - `sacadm -s -p inetd`
7. Now restart SNA and CS/UnixWare, and then the SNA node.

If the system is ever rebooted and the need to share the TCP port 23 is still present, steps 4, 5, 6, and 7 must be repeated. A shell script to provide this

may be configured to run at startup, and could look similar to the following figure:

```
#!/bin/ksh
/opt/sna/bin/sna stop
sacadm -k -p inetd
/opt/sna/bin/snainetd
sacadm -s -p inetd
/opt/sna/bin/sna start
/opt/sna/bin/X11/xsnaadmin&
exit
```

Figure 32. Sample Korn Shell Script for SCO UnixWare to Share TCP Port 23

3.6 Round-Robin Load Balancing for TN3270 Servers

Now that the TCP servers are configured, the Dispatcher can perform standard weighted round-robin load balancing.

Weights are applied to all servers on a given port, and for any particular port, the requests will be distributed among servers based on their weights relative to each other. For example, if the weight on one server is set to `10` and on the other server to `5`, the first server gets twice as many requests as the second server.

In our configuration, since the weight for each server on port 23 had been set to `10` (see Figure 18 on page 49), we expected the Dispatcher to perform the standard, not weighted, round-robin load balancing.

In a weighted configuration, the servers would be assigned more meaningful weights to determine the relative load to be distributed among the servers. The Dispatcher keeps internal statistics on the number of TCP connections to each port to determine which server is due for a connection.

The internal counters can be seen by using the ndcontrol server report command. The following command shows the counters for each server in our cluster and port 23.

```
ndcontrol server report
9.24.104.107:23:9.24.104.16+9.24.104.246+9.24.104.40+9.24.104.186
```

```
csaix1:/ > ndcontrol server report 9.24.104.107:23:9.24.104.40+9.24.104.16+9.>

--------------------------------------------------------------------------------
Cluster: 9.24.104.107 Port: 23
--------------------------------------------------------------------------------
Address          |  Total   |   TCP    |    UDP   |  Active  |  FINned  | Complete |SaWt|
--------------------------------------------------------------------------------
     9.24.104.40|         0|         0|         0|         0|         0|         0|  -1|
--------------------------------------------------------------------------------
     9.24.104.16|         0|         0|         0|         0|         0|         0|  -1|
--------------------------------------------------------------------------------
    9.24.104.186|         0|         0|         0|         0|         0|         0|  -1|
--------------------------------------------------------------------------------
    9.24.104.246|         0|         0|         0|         0|         0|         0|  -1|
--------------------------------------------------------------------------------
```

Figure 33. Server Report Output

The output from the report shown above shows that there are no active connections to our servers.

3.6.0.1 Personal Communications Client Configuration

The next step was to establish a TN3270 connection with a Personal Communications (PCOMM) client to the cluster IP address 9.24.104.107. Configuration for TN3270 in Personal Communications is very simple. When starting the configuration you choose TCP/IP as shown in the following figure.

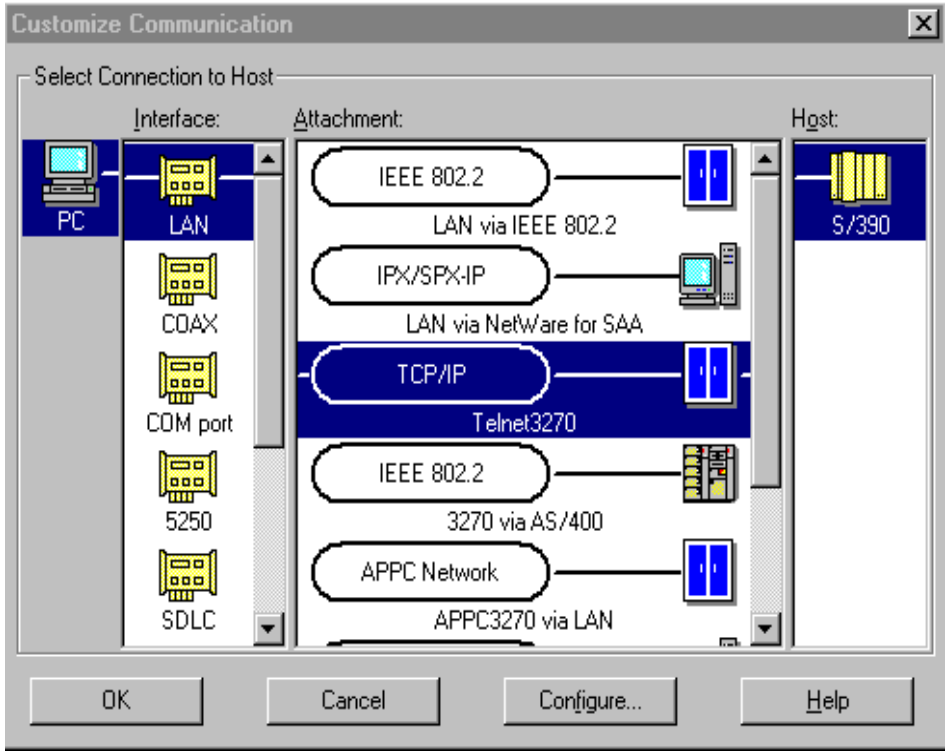


Figure 34. PCOMM Configuration Panel

After clicking **Configure** you get the following panel:

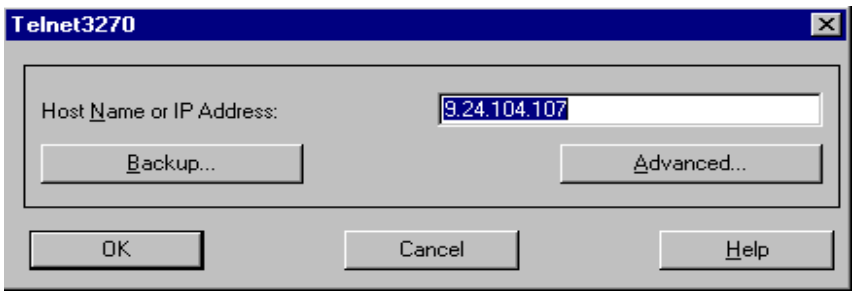


Figure 35. TN3270 Configuration

This is where we entered the IP address or TCP/IP host name of the cluster.

Note

In load balancing scenarios using ISS, load balancing is achieved by altering the TCP/IP DNS so the pool name used by clients to reach the host resolves to the preferred server address. This effectively means that a TCP/IP host name will resolve to different IP addresses depending on which server is the least loaded.

In these scenarios, some clients tended to establish all sessions subsequent to the first on a workstation to the same IP address as the first session, bypassing the name resolution. It seems that the operating system is caching IP address and host name information. This means only the first client session on a workstation benefited from load balancing until the cache expired.

Clients that have such caching get the true benefit of load balancing in this scenario since the IP address and host name are always the same.

After clicking **OK**, the configuration is finished and the connection is made. Each time you start a PCOMM session using the IP address you will be given the preferred server at the time. You could have several sessions started to the cluster address, all going through a different Communications Server.

The TCP/IP request is routed by the Dispatcher to one of the TN3270 Servers. The TN3270 Server, in turn, uses SNA to complete the session to the host.

In our scenario, we defined the host connections so the LU names from each TN3270 Server would have a unique prefix and defined the initial VTAM screen to show the LU name. This way we could easily see which server we were passing through.

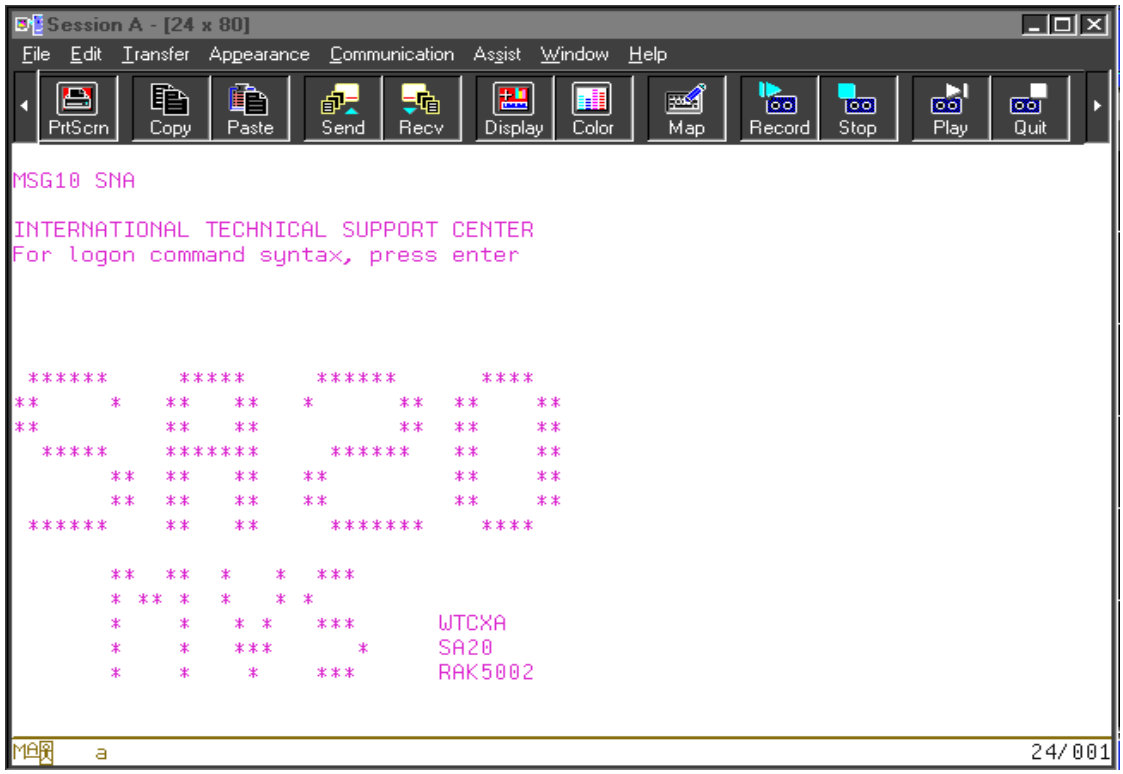


Figure 36. TN3270 Session Provided by One of the Servers

Since the screen shows the LU name RAK5002, we already knew which server provided the LU. But you can find this out as well by issuing the ndcontrol server report command:

```
ndcontrol server report
9.24.104.107:23:9.24.104.16+9.24.104.246+9.24.104.40+9.24.104.186
```

Now that we have established a session through the Dispatcher the results of the report show one active connection and which server was used.

Remember that this report only reflects the internal counters. Other sessions established to the servers outside of the Dispatcher will not be shown here.

```
csaix1:/ > ndcontrol server report 9.24.104.107:23:9.24.104.40+9.24.104.16+9.24.104.186+9.2>

------------------------------------------------------------------------------------------
Cluster: 9.24.104.107 Port: 23
------------------------------------------------------------------------------------------
Address          |  Total   |   TCP    |    UDP   |  Active  |  FINned  | Complete |SaWt|
------------------------------------------------------------------------------------------
      9.24.104.40|         1|         1|         0|         1|         0|         0|  -1|
------------------------------------------------------------------------------------------
      9.24.104.16|         0|         0|         0|         0|         0|         0|  -1|
------------------------------------------------------------------------------------------
     9.24.104.186|         0|         0|         0|         0|         0|         0|  -1|
------------------------------------------------------------------------------------------
     9.24.104.246|         0|         0|         0|         0|         0|         0|  -1|
------------------------------------------------------------------------------------------
```

Figure 37. Server Report Output Showing One Session

After we started two more client sessions, we saw that the sessions were served by 9.24.104.16 and 9.24.104.40. If one server or service (TN3270 Server) is not active, the Dispatcher is not aware of this and will try to connect to this machine when it would be the next to be chosen. When the Dispatcher cannot establish the connection, it will try the next server. The user will see a delay of a few seconds but the session will be established. This is true regardless of whether the client, in this case PCOMM, is configured to retry the session on its own.

Chapter 4. Scenario 2: WebSphere Dispatcher and the Advisors

Scenario 1 created an environment where the Executor used a weighted round-robin scheme to balance TN3270 load among several TN3270 servers. The next option we would like to show is adding advisor input to this environment. In Scenario 1 the server weights were pre-determined by the administrator and could only be changed by the administrator. Activating the manager component and defining advisors will cause the server weights to be changed dynamically based on the advisor input.

We will use the Telnet Advisor to monitor response time to the servers. We can do this because we are using the Telnet port (23) for TN3270 traffic and WebSphere ships with an advisor for Telnet. If you choose to use another port for TN3270 traffic you will need to write your own advisor from samples provided with WebSphere.

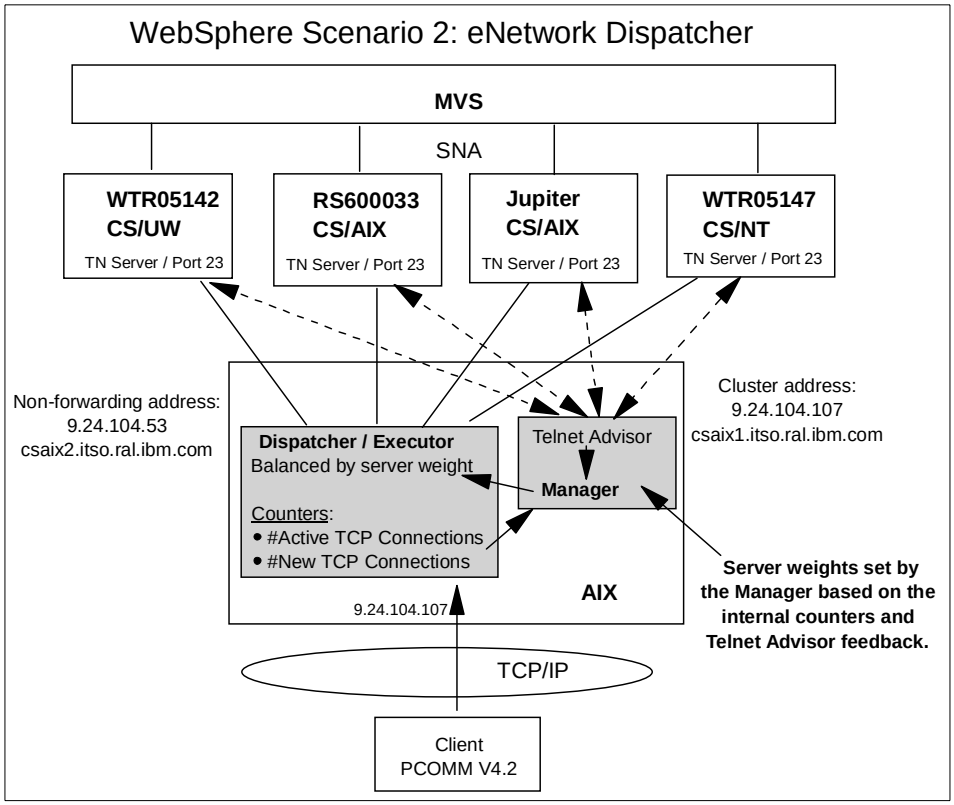


Figure 38. Dispatcher Scenario 2: Using Advisors

4.0.0.1 Activating the Manager

The main load balancing component is the *Manager*. The Manager is the component that collects the information from the advisors about the servers' condition. Based on that information, it then adjusts the weights of the single server to reconfigure the load distribution.

If you want to start the Manager component through the GUI, with your mouse right click the **eNetwork Dispatcher** item (see Figure 7 on page 34) and from the pop-up menu select **Start Manager...**. You are prompted to optionally type a log file name in which the Manager will log all its activities, and a metric port, used by the ISS function, if running, to report system loads. We set the log file name to `eNDmanager1` and we left the Metric port field blank, as shown in the following figure:

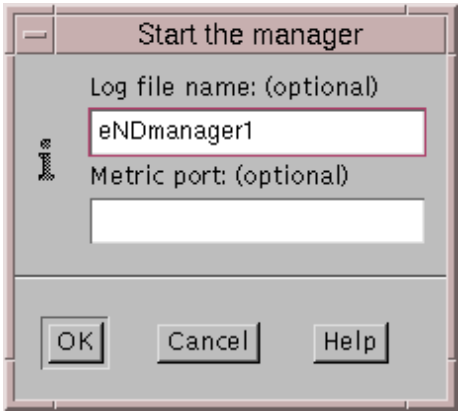


Figure 39. Starting the Manager

The command to line to start the manager is:

```
ndcontrol manager start log_file metric_port
```

Leaving blank the Metric port field forces the system to use the default value 10004.

4.0.0.2 Activating the Advisors

In order to feed the Manager with more information about the ability of the TCP servers to respond to requests, you need to start the advisors. The advisors monitor each server defined on the assigned port, and forward the information about the server's response time and availability to the Manager.

The following table lists the available advisors along with their respective protocols and ports:

Table 2. Advisors and Related Protocols and Ports

Advisor Name	Protocol	Port
ftp	FTP	21
telnet	Telnet	23
smtp	SMTP	25
http	HTTP	80
pop3	POP3	110
nntp	NNTP	119
ssl	SSL	443

Notes:

1. If you use the ftp advisor, it should be started only on the FTP control port 21, and not on the FTP data port 20.
2. Starting the Manager automatically starts the Reach advisor, used for high availability situations. For high availability, two Dispatchers with connectivity to the same clients are used. The Dispatchers use a "heartbeat" mechanism between them to detect Dispatcher failure. The Reach advisor is used to determine what can be reached from each Dispatcher. A reachability table is built and synchronized between the Dispatchers.

To start the advisor from the GUI, with your mouse right click **Manager** and select **Start Advisor...** from the pop-up menu.

Select the advisor you want to add from the list box. We selected the Telnet Advisor and the port was automatically set to 23.

Figure 40. Select the Telnet Advisor

Click **OK** and the advisor will be added to the configuration, as shown in the following panel figure.

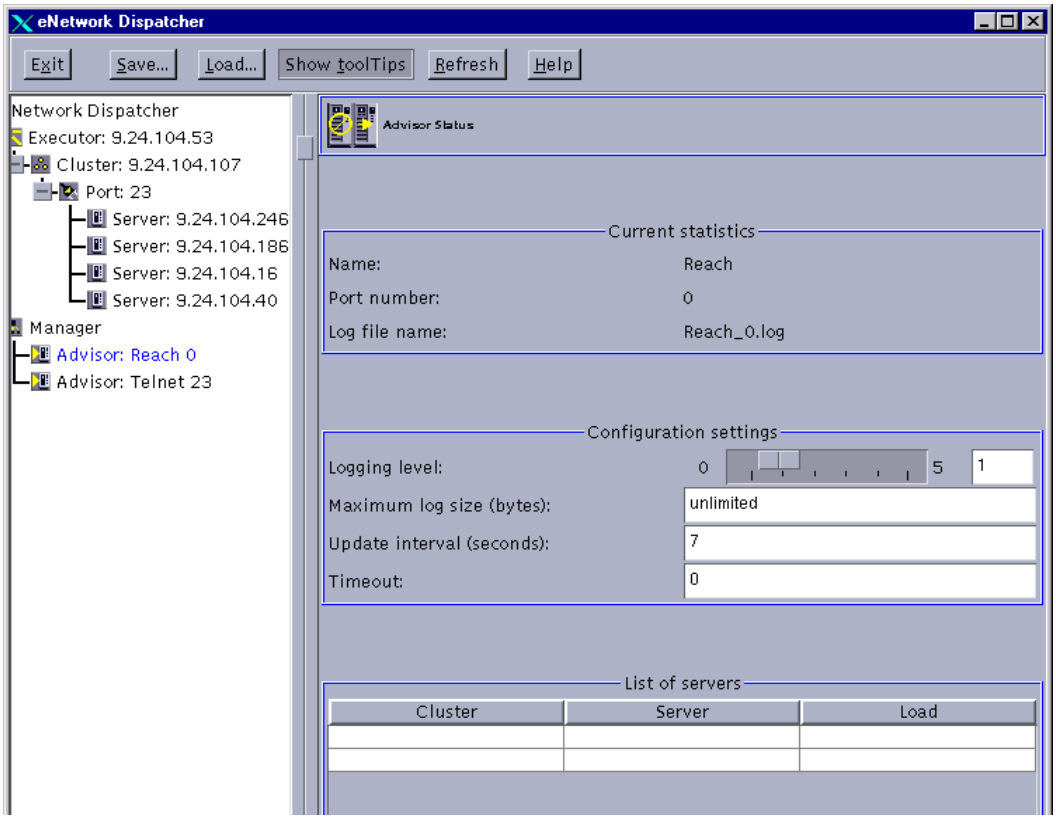


Figure 41. How to Start the Advisor

The command line to add an advisor is:

```
ndcontrol advisor start telnet 23
```

4.0.1 Customization of the Manager and the Advisors

The load balancing action performed by the Dispatcher depends on two settings:

- Proportions of importance
- Smoothing index

4.0.1.1 Proportions of Importance Settings

Next we will show you how to set the proportions of importance for each of the policies. In the configuration window shown in Figure 41 on page 70, click the **Manager** item in the tree panel and the Manager Status window will appear in the right pane.

The default values are `50 50 0 0`. We changed this by sliding the rule next to each proportion setting to try `46 46 8 0` as our settings. This means:

- Both the number of active connections and new connections will contribute 46% in the weighting process.
- The advisors will contribute 8%.
- Inputs from system monitoring tools, such as ISS, will not be considered in the weight decision. This is reasonable, because in this test we had not yet activated any monitoring tool.

4.0.1.2 Smoothing Index

Another important parameter to be considered when using the Advisors (and/or system monitoring tools such as ISS) is the *smoothing index*, which defines how smooth the change of the servers' weight will be. The Manager calculates and then updates the servers' weights dynamically: the weight values could change very rapidly, and in some circumstances this might result in an *oscillant* effect in the way the requests are load balanced. The distribution of the requests could be made in a non-optimized way, and would need to be smoothed. Based on the smoothing index value, the Manager will change the servers' weights more or less quickly. The default value for smoothing index is 1.5, which could cause the Manager to change the weights rather dynamically. Values around five are preferred for having the Manager modifying the weights slowly.

We set the smoothing index value to 6. After you modify values in the Manager Status window, click **Update** to have the configuration updated, as shown in the following window.

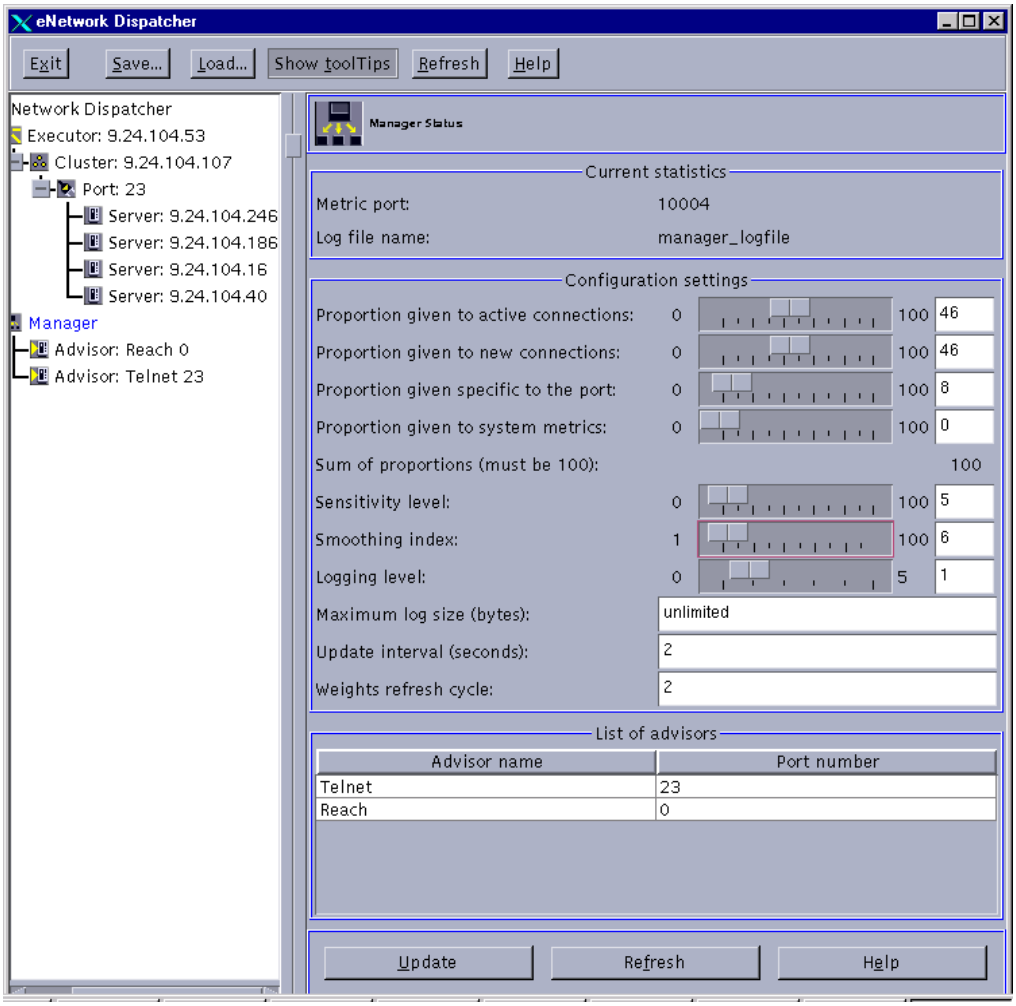


Figure 42. Proportions and Smoothing Level

4.0.2 Saving the Configuration

Once you finish the configuration steps we have described, it is recommended you save the configuration of the load balancing machine. To do so, click the **Save...** button in the top menu bar of the configuration window

and in the panel that appears, enter the configuration file name, as shown in the next figure.

Enter a file name and click **OK**. If the platform for your eNetwork Dispatcher machine is AIX, the configuration file will be saved in the location /usr/lpp/eND/dispatcher/configurations. On a Windows NT platform, the directory where the configuration files were saved on our machine was \Program Files\eND\dispatcher\Configurations.

The configuration file we saved is an ASCII file. We show it in Figure 43.

```
ndcontrol executor start
ndcontrol executor set nfa 9.24.104.53

ndcontrol cluster add 9.24.104.107

ndcontrol port add 9.24.104.107:23
ndcontrol port set 9.24.104.107:23 staletimeout 32000000

ndcontrol server add 9.24.104.107:23:9.24.104.40
ndcontrol server set 9.24.104.107:23:9.24.104.40 weight 0

ndcontrol server add 9.24.104.107:23:9.24.104.16

ndcontrol server add 9.24.104.107:23:9.24.104.246
ndcontrol server set 9.24.104.107:23:9.24.104.246 weight 0

ndcontrol server add 9.24.104.107:23:9.24.104.186

ndcontrol manager start manager_logfile
ndcontrol manager proportions 46 46 8 0
ndcontrol manager smoothing 6.0

ndcontrol advisor start Telnet 23 Telnet_23.log
```

Figure 43. Configuration File

To stop the Executor, open the eNetwork Dispatcher GUI and with your mouse right click the **Executor** item in the left area. Then select **Stop Executor** from the pop-up menu that is brought up. Otherwise, if you chose to use the command line, enter the following:

```
ndcontrol executor stop
```

To exit the GUI, click the **Exit** button.

On Closing the GUI

Remember that if you close the GUI without stopping the Executor, you leave the Executor running. In fact, if you re-start the GUI with the command `ndadmin`, you will still be shown the current running configuration.

Once you stop the Executor, if you start the GUI again, you can choose whether to load a saved configuration, if any, or create a new one. To load a configuration file, click the **Load...** button in the top menu bar and select a file name from the list box.

Chapter 5. WebSphere Interactive Session Support

IBM Communications Server for AIX is shipped with a custom metric called sys_load that, when used with Interactive Session Support (ISS), offers a robust solution for controlling and balancing the workload across TN3270 server gateways. Clients are connected to a server based on mainframe LU availability, link speed, client/mainframe connection load, overall system load, and machine processor speed. In addition to optimizing the load conditions of the servers, load balancing maintains the operation of the network during server failures by automatically re-connecting clients to an available server thereby improving system availability. The sys_load script is shown in Appendix A, “Determining System Load with sys_load” on page 197.

You can use the ISS function by itself to balance the load on servers within a local or wide area network using a DNS round-robin approach or a more advanced user-specified approach. Load balancing is performed at the machine level. ISS can also be used to provide server load information to a Dispatcher machine.

When used for load balancing, ISS works in conjunction with the DNS server to map DNS names of ISS services to IP addresses. When used to provide server load information, a DNS is not required.

Using either the wide area function of the Dispatcher or a combination of the Dispatcher and ISS allows you to balance the load on servers within both local and remote networks.

You can use the ISS function with or without a DNS name server:

- If you are using ISS for load balancing, a DNS server is required. This may be either an actual DNS server or, if you set up a small, separate subdomain for a new name server, a replacement name server provided by ISS. Using this approach, ISS runs on a DNS machine. A client submits a request for resolutions of the DNS name for an ISS-associated service, which has been set up by an administrator. ISS then resolves the name to the IP address of a server in the cell, and forwards this IP address to the client.
- If you are using ISS to collect server load information, a DNS is not needed. The ISS monitor collects server load information from the ISS agents running on the individual servers and forwards it to the Dispatcher. The Dispatcher uses this load information, along with other sources of information, to perform load balancing.

ISS periodically monitors the level of activity on a group of servers and detects which server is the least heavily loaded. It can also detect a failed server and forward traffic around it. Once every monitoring period, ISS ensures that the information used by the DNS server or the Dispatcher accurately reflects the load on the servers. The load is a measure of how hard each server is working. The system administrator controls both the type of measurement used to measure the load and the length of the load monitoring period. You can configure ISS to suit your environment, taking into account such factors as frequency of access, the total number of users, and types of access (for example, short queries, long-running queries, or CPU-intensive loads).

All the nodes in a site work together to eliminate any single point of failure. Should the monitor machine fail, the survivors elect a new monitor to take over automatically.

5.1 ISS Cells and Services

ISS logical architecture is based on the concept of a *cell*. A cell is a group of servers administered as a single, logical unit. Each server in a cell is also called a *node*.

A group of servers (or nodes) that perform the same function is a *service.* Think of a service as a group of nodes in the cell that will serve the request only for a particular protocol (for example HTTP, FTP, Telnet).

It is possible to define different services in a cell (corresponding, for example, to HTTP, FTP, Telnet). Moreover a node can belong to one or more services.

On each node of the cell the ISS component must be installed, configured and activated. The ISS will run as a daemon, called the issd daemon.

5.2 ISS Configuration Files

When configuring ISS, you need to edit a configuration file that instructs the issd daemon on how to act. Examples of configuration files are shipped with the product. In fact, the installation of the load balancing component of IBM WebSphere Performance Pack comes with three ISS sample configuration files: Iss_config_1, Iss_config_2 and Iss_config_3. These three ISS sample configuration files are very useful for understanding how to configure ISS in your environment:

1. The file Iss_config_1 shows the configuration of a local cell. It is a simple configuration file with only one local cell and one service running.

2. The file Iss_config_2 illustrates how ISS can be used with a DNS name to perform load balancing for three clusters of servers. A different load balancing method (round-robin, custom and LoadLeveler) is used for each cluster.
3. The file Iss_config_3 illustrates how ISS can be used with the Dispatcher in a two-tiered architecture. The bottom tier consists of three clusters of servers. Each cluster itself has two servers. Three Dispatchers are used to balance these clusters. The top tier consists of three Dispatcher nodes. In this configuration, ISS performs two functions:
    1. It performs DNS load balancing for the top tier.
    2. It provides the Dispatchers with additional server load information, so that they can load balance the bottom tier more accurately.

We recommend that you open these configuration files with a text editor and see how they are structured. If you selected the U.S. English language, the three ISS sample configuration files will be located in the directory:

- /usr/lpp/eND/iss/samples/en_US on AIX
- iss\Bin\samples\en_US under the Load Balancing directory on Windows NT

The configuration file must be installed on all the machines that implement ISS, either as a monitor or as agents. There are no name requirements for the file. What is important is that when you configure the ISS machines in your cell, all the ISS configuration files have the same contents.

In the following figure, we show you the ISS sample configuration file Iss_config_1.

```
# ------------------------------------------------------------------
#
# Sample ISS configuration file 1
#
#------------------------------------------------------------------
#
# Configuration of a local cell
# This is a simple configuration file,
# with only one (local) cell, and one service
# running.
# Parameters for the whole cell
Cell      Hursley               local
AuthKey     10043572 ADE4F354 7298FAE3 1928DF54 12345678
LogLevel                        info
HeartbeatInterval               5
HeartbeatsPerUpdate             3
PortNumber                      7139

# Individual node data
# Node numbers do not have to be sequential
# nemesis is prevented from taking over the role
# of monitor.
Node    delta                   001
Node    spcontrol               002
Node    spnode1                 003
Node    spnode2                 004
Node    spnode3                 005
Node    spnode4                 006
Node    spnode5                 007
Node    nemesis                 099
NotMonitor
```

Figure 44. Iss_config_1 (Part 1)

```
# The service is only configured to depend on
# one resource -- CPU availability.
# Load balancing is therefore performed based
# only on CPU utilization. However, ISS will
# not schedule work for nodes that are unreachable
# on the network.
# The specified MetricLimits indicate that a node
# will not be used if its CPU usage goes over 95%
# and will not be put back in the list until CPU usage
# goes back down to 80%.

ResourceType                          CPU
Metric Internal                       CPULoad
MetricNormalization                   0       100
MetricLimits                          80      95
Policy   Min

# The one configured service
# WWWService is an identifying string for the service, and
# is used when balancing between cells.
# The service DNS name is  www.issdev.hursley.ibm.com.
# followed by the cluster address and the port number.

Service     WWWService          www.issdev.hursley.ibm.com 129.40.161.74 21
NodeList                        spcontrol spnode1 spnode2 spnode3
ResourceList                    CPU
SelectionMethod                 Best
Overflow                        spnode5.hursley.ibm.com

# The machine 'delta.hursley.ibm.com' is configured as
# an ISS Nameserver -- when the issd program starts on
# machine delta, it will also try and run an ISS name
# server thread on port 53.
ISSNameserver                   delta    53
ServiceList  WWWService

# The machine nemesis.hursley.ibm.com has a
# Network Dispatcher configured on port 10004
Dispatcher                      nemesis 10004
ServiceList WWWService
```

Figure 45. Iss_config_1 (Part 2)

In the following sections, we see how the ISS function works to better understand the settings shown in the above configuration file. We recommend that you read the information provided in the following sections in conjunction with the above configuration file.

The following figure shows an overview of the configuration for ISS. You may want to refer back to it as you go through the following sections.

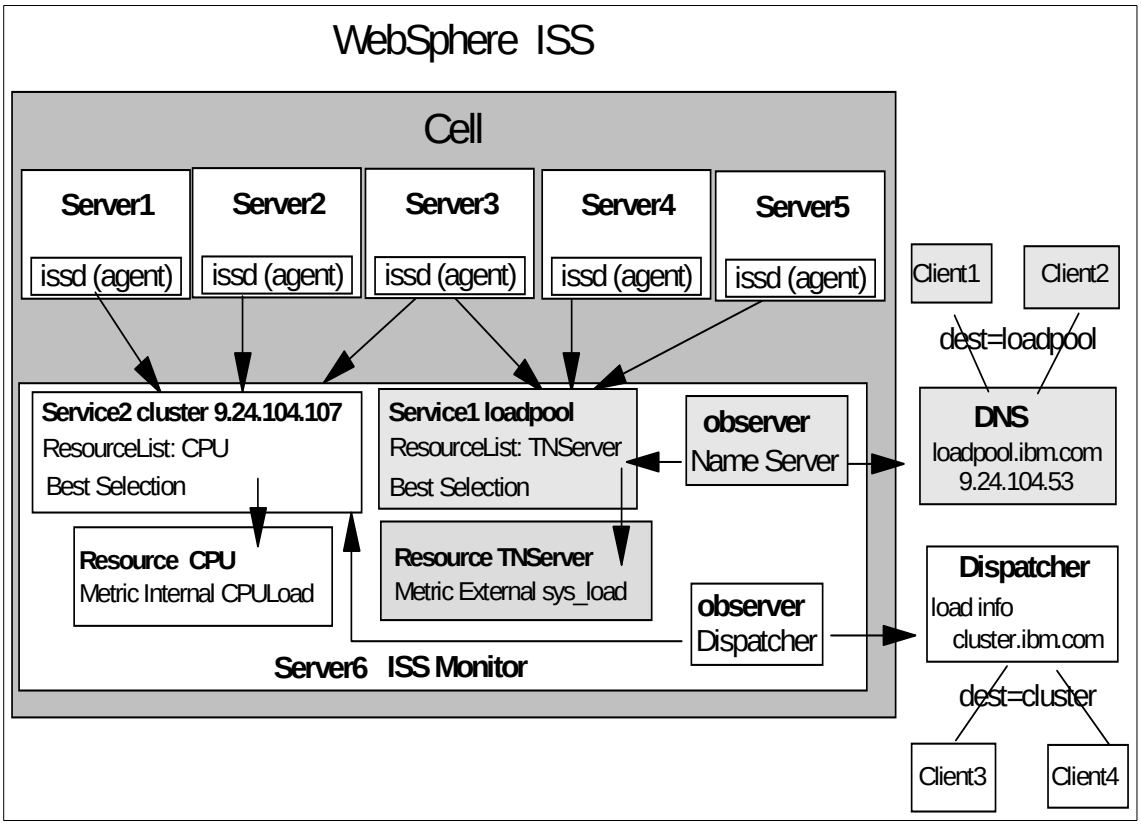


Figure 46. WebSphere ISS Overview

5.2.1 Cell and Its Attributes

The first thing that an ISS configuration file does is to define the cell and declare some cell attributes.

A cell can be local or global, and this is specified in the ISS configuration file through the keywords `local` or `global` respectively. A local cell is one that the node will be a member of. There must be one and only one local cell defined

in the ISS configuration file. A global cell is a separate, remote cell that you want your node to communicate with. You must have a global cell if you have widely separated mirror sites and want to perform *ping triangulation*, which is a technology that allows you to find out which server site is the closest to a given client.

A cell can have the following attributes specified in the ISS configuration file:

- `Authkey`

  This key is used to provide authentication during the ISS internal traffic among the nodes of the cell. If this keyword is not specified in the configuration file, a default key will be used. A node can have its own Authkey attribute, which has to be specified after the node is defined.

- `LogLevel`

  You can have five different log levels, each one providing more information than the previous one:

  `[ None | Error | Info | Trace | Debug ]`

  The above values for the `LogLevel` attribute are self-explanatory.

- `PortNumber`

  This option specifies the port number used for ISS internal traffic:

  `PortNumber [`*`number`*`]`

- `HeartbeatInterval`

  ISS will periodically contact all servers that provide a specific service to ensure that they are still alive. ISS does this using the equivalent of the `ping` command. This does not verify that the service is running on the servers, but at least ensures that the servers can be contacted at the IP level. ISS maintains a ranking that lists the servers in their current priority order according to the configured load measurement metric. If the currently top-ranked server fails to respond, the next server will be selected. This check is referred to as a heartbeat. The time between heartbeats is determined by the `HeartbeatInterval` setting. The syntax is:

  `HeartbeatInterval [`*`seconds`*`]`

- `HeartbeatsPerUpdate`

  The syntax of this keyword is:

  `HeartbeatsPerUpdate` *`count`*

  Where *`count`* is a non-negative integer.

  The `HeartbeatsPerUpdate` value is multiplied by the value of `HeartbeatInterval` to give a time called the *update interval*. The value

assigned to the update interval parameter indicates the number of heartbeats after which ISS will determine whether a domain name server update or a Dispatcher update is required.

Note that the value of `HeartbeatsPerUpdate` will be ignored in the situation where the top ranked server fails to respond to a heartbeat. The server will immediately be removed from its top ranked position.

5.2.2 Nodes

After defining the cell, you must list all the nodes that will be part of it using the `Node` keyword. The syntax is:

```
Node [ hostname | IP address ] priority
```

In your cell configuration, when using ISS, you should elect one of the nodes as the cell leader, or *monitor*, while all the other nodes act as *agents*. In other words, if ISS is used to collect server load information, there is an ISS monitor that collects this information from the ISS agents that run on the individual servers. The ISS monitor is then used to send the ISS agent information to the Dispatcher. The ISS monitor can be installed on one of the servers, a Dispatcher machine or on a different machine.

In Figure 46 on page 80, Servers 1 through 6 are listed as nodes. Server 6 is the ISS monitor.

You can configure one or more other nodes as backup to the monitor node by defining a different priority number for each one. In a backup node, the issd daemon usually runs in agent mode, and the node can be a server that provides a service. If the monitor fails, the first (in the priority scale) backup node in the list takes over, and on this node the issd monitor switches from agent mode to monitor mode. As soon as the previous node is available again, it will become the monitor because of its higher priority value. This provides ISS *high availability*.

If you don't want a particular machine to run as an issd monitor, you have to declare this using the `NotMonitor` keyword immediately after the node declaration.

In this case the priority field is only a numeric identifier for that node.

5.2.3 Services

Before explaining how to define the resources, we prefer first to focus on the section of the configuration file where the services are defined, along with their parameters:

- `Service`

  You declare a service with the `Service` keyword followed by the service name, the fully qualified service DNS name (which is the DNS name of the pool of machines that provides the same service), the cluster IP address and the port number. The correct syntax is:

  `Service` *`name DNSName`* [*`cluster address`*] [*`port number`*]

- `NodeList`

  After defining the service, you need to use the `NodeList` keyword, which allows you to specify the list of nodes and members of your cell providing the service. `NodeList` must be followed by the host names or IP addresses of those servers:

  `NodeList` *`DNSName`* [*`DNSName ...`*]

- `ResourceList`

  The `ResourceList` parameter allows you to declare which resource will be used to determine the appropriate node that will provide that service. A resource can be CPU, disk, process, and so forth.

- `Overflow`

  This keyword identifies a server on which to fall back if all the other nodes specified in `NodeList` are unable to provide the service. The syntax is:

  `Overflow` *`DNSName`*

In Figure 46 on page 80, you will see two services, Service1 and Service2. Service1 uses an internal metric called CPULoad to determine the preferred node for that service. Service2 uses an external metric to determine the preferred node for that service. Each service has a node list that includes a subset of the nodes available.

5.2.4 Resources

When you define a service in the configuration file, you specify the name of one or more resources as arguments of the `ResourceList` parameter. A single resource can be used as the selection criterion in different services. Consequently, the resources should be defined in the configuration file before the service definitions.

You define resources using the `ResourceType` parameter and its own keywords `Metric`, `MetricNormalization`, `MetricLimits`, and `Policy`.

The ResourceType keyword must be followed by the symbolic name of the resource you are defining. It specifies the criteria you have decided to use for selecting the best server in a service. The syntax is:

```
ResourceTypes Name
```

5.2.4.1 Metrics

A resource type is characterized by several parameters that define the metric that will be used to measure the load among the servers.

A metric defines how ISS will measure the load on the server for each service. This measurement should be appropriate to the particular service. For example, if the service provided is CPU-intensive, the metric could be defined as the percentage of time that the CPU is busy. If the service is disk- or I/O-intensive, the metric could be based on the amount of time that the disks are busy or the time spent waiting for I/O to complete.

The metric keywords are the following:

- Metric

  The keyword Metric in the configuration file specifies the type of measurement ISS will use to provide a specific service. The syntax is:

  Metric [*Internal* | *External*] *string*

  where the *string* field indicates a command or a program that gives a numeric output.

  The sample configuration file ISS_config_1 that we have shown uses the line:

  ```
  Metric Internal     CPULoad
  ```

  The keyword Internal identifies a given measurement method as being provided by the system. The parameter CPULoad forces ISS to measure the percentage of the CPU that is being utilized. The other system-provided measurement system is FreeMem, which returns the amount of free physical memory as a percentage.

  When you set Metric to External, you define how to measure the load on the servers. This metric is anything that can be executed on the server and returns a numeric value as a result. In this case the *string* field defines a command or program that when executed, returns a numeric value that can be used to measure the load on the server.

  For example, the following entry defines a metric based on the number of processes running on the server:

  ```
  Metric External ps -ef | wc -l
  ```

- `MetricNormalization`

  This is a range between the lower and upper limits of measurement, indicated as integer numbers. The syntax is:

  `MetricNormalization` *`LowerLimit UpperLimit`*

  The limits referred to depend upon the metric you are using. In case you want to measure the CPU utilization, as shown in the Iss_config_1 sample configuration file, the two limits are percentage points. The range is therefore 0 to 100. Any metrics coming into the monitor outside of this range would be corrected by being clipped to fit within these limits.

- `MetricLimits`

  The syntax for this entry is:

  `MetricLimits` *`RecoverLimit FailLimit`*

  You can set these limits to prevent a server from failing totally by having too many requests assigned to it. ISS measures the loads on servers periodically. If the loads on a certain server are within predefined limits, ISS continues keeping the server in the list of the available nodes. If the loads are outside the limits, ISS removes the server from the ranking for that service. If the server is handling more than one service, it may still be included in the ranking for the other service or services. There are two levels you define:

1. The *`FailLimit`* level represents the level beyond which the metric should not go. When the *`FailLimit`* level is reached, ISS removes the server from ranking. In the Iss_config_1 configuration file that we have shown, the specified value for *`FailLimit`* indicates that a node will not be used if its CPU usage goes over 95%.
2. The *`RecoverLimit`* level sets the point at which a node that had been removed from the ranking should be returned to active participation. In the Iss_config_1 ISS sample configuration file, the *`RecoverLimit`* parameter is set to 80%.

   So, defining the limits of the CPU load metric with the following entry causes the resource to be removed at 95% of utilization, and returned only when it is back down to 80%:

   ```
   MetricLimits 80 95
   ```

   Specifying a significant difference between the two above levels prevents the phenomenon wherein resources come into service, fail, then come back after minimal recovery only to quickly fail again.

- `Policy`

The parameter `Policy` can be assigned either the value `Max` or `Min`. The correct syntax is:

```
Policy { Max | Min }
```

It indicates whether a metric function is to be minimized or maximized. The default value is `Max`.

For example, a metric based on the number of active users would regard the smallest value as the best and would have the following entry:

```
Policy Min
```

A metric function based on CPU idle time on the other hand, would regard the maximum value as best and, therefore, the policy would be specified as the following:

```
Policy Max
```

5.2.5 ISS Observers

After configuring nodes and services, and after defining the resources, you should define who will use the information about the status of nodes. In other words, you should configure the observers.

An *observer* is a network process (such as a load balancer) that uses information about the status of nodes. ISS provides three observers, which perform different actions, but have the same purpose: load balancing. The three observers are named NameServer, ISSNameServer and Dispatcher. These names correspond to the keywords `NameServer`, `ISSNameServer` and `Dispatcher` that you should use in your ISS configuration file.

Depending on how you decide to configure your environment, you can configure one or more observers in your cell. In Figure 46 on page 80, two observers are defined. The Dispatcher observer uses input from the Service2 service. The NameServer observer uses input from Service1.

5.2.5.1 NameServer

When using the NameServer observer, ISS runs in conjunction with a DNS name server. It uses the DNS name server to map DNS names of ISS services to IP addresses of the most appropriate server. ISS assists the DNS server in making the balancing decision. ISS monitors the load on each server of the cell and ensures that the server currently used for a particular service is the one with the lightest load.

It is not mandatory that the machine where the issd monitor process runs is the same machine where the DNS daemon named runs. The DNS can run on

any machine with the issd daemon running. For this reason, you can configure one or more servers on your cell as backup issd monitor nodes.

The load balancing decisions are based on how you have configured ISS. In this case, ISS makes load balancing decisions by itself and instructs the DNS name server about the server to which the incoming requests are to be routed.

The NameServer observer involves some load on the DNS server machine. Every time a new server is selected as the top ranked server, the DNS configuration files are updated. A signal is then sent to the named daemon to reload the name resolution data files. If these files are large, it can take a significant amount of time and processing for the named daemon to process them. During this processing, the named is unable to respond to name resolution requests.

The syntax to define a NameServer observer is the following:

```
NameServer DNSName [ PortNumber ]
```

5.2.5.2 ISSNameServer

When using the ISSNameServer observer, ISS works as a DNS name server. In this case, ISS runs on a DNS domain name server machine, where the DNS daemon named is not running.

ISS replaces the named name server daemon and makes use of the DNS configuration file of the DNS name server. In this case ISS provides the name serving function by providing minimal name server implementation. In this way, besides taking over the DNS, ISS makes load balancing decisions by itself.

Using the ISSNameServer observer does not increase the load on the DNS machine, so it is a good option to use the ISSNameServer observer with a new subdomain for your pool of servers.

The syntax to define an ISSNameServer observer is the following:

```
ISSNameServer DNSName [ PortNumber ]
```

For example, the following is the entry used in the Iss_config_1 sample configuration file:

```
ISSNameServer delta 53
```

5.2.5.3 Dispatcher

When using a Dispatcher observer, you should have defined in your cell a Dispatcher machine, with which ISS cooperates to perform load balancing. Different from the previous two types, a DNS server is not required (neither a DNS server nor an ISSNameServer observer working as DNS).

The ISS configuration file will reflect your cell configuration, and the Dispatcher machine you selected should be specified immediately after the `Dispatcher` keyword, according to the following syntax:

`Dispatcher` *`DNSName`* [*`PortNumber`*]

In such a configuration, you can look at ISS as a monitoring tool on the TCP server machines. ISS provides the Dispatcher with load server information, but ISS does not make any load balancing decision.

The ISS monitor collects specific server information, such as CPU usage, memory usage and disk activity, from the ISS agents running on the individual servers, and forwards it to the Dispatcher. The Dispatcher uses this load information, along with other sources of information, to determine which is the least loaded server of the cluster and then performs load balancing.

5.2.6 ISS Selection Methods

For each observer you plan to use in your cell, you should specify a selection method, that defines how the ISS monitor selects the best node to perform that service. You can use two selection methods: RoundRobin and Best, identified with the keywords `RoundRobin` and `Best` respectively.

- `RoundRobin`

  The load among servers is distributed on a round-robin basis; the load on each server is ignored, although a server will not be recommended if it appears to be down.

  The main advantage of using round-robin is that round-robin avoids overhead in the server associated with other measurement types. The main disadvantage of using round-robin is that ISS does not notice whether a particular service is getting very busy, and ISS could continue to use a busy server as the server to which new users connect.

- `Best`

  For each service, the ISS monitor maintains a ranking of the best nodes to perform that service. Periodically, the monitor calculates the best server for every defined service and updates the ranking of the nodes. Before the next updating (that is, for the update period), ISS will use (or recommend to use) the highest server in the ranking.

In order to establish the ranking, ISS monitors the load on each server and selects for a particular service, the node that best performs that service. In this case, the node selection performed by ISS truly depends on the resources defined for the service.

Server Failure

For both selection methods, ISS detects if a server fails and does not schedule requests for that service.

A group of nodes performing the same service are independent of each other, and the choice of measurement type for one does not affect the choices available for the others. You can have several such groups, with each operating in a different manner. For example, you can configure a round-robin service and additionally provide a service using the `Dispatcher` observer with the `Best` selection method.

5.3 Managing ISS

This section explains how to start, control and stop the ISS function.

5.3.1 The issd Command

ISS makes use of only one daemon, named issd, which should be launched in each of the nodes belonging to a cell. The issd daemon is instructed on how to operate by the directions contained in the configuration file. For example, based on the parameter in the configuration file, the issd will operate as ISS monitor or ISS agent.

The issd daemon is launched through the `issd` command and can be used with different flags and parameters. The most important flags and parameters are `-c` *`config_file`* and `-l` *`log_file`*.

Use `-c` *`config_file`* to specify the configuration file. If not specified, `issd` should expect to find the file in a pre-defined default location, that is:

- /etc/iss.cfg on AIX
- \Program Files\eND\iss\iss.cfg on Windows NT

Notice that each node should have the same configuration file.

When you use `-l` *`log_file`*, the log information is directed, by default, to a predefined file:

- /etc/iss.log on AIX
- \Program Files\eND\iss\iss.log on Windows NT

Use this flag and parameter to use the specified *log_file* instead of the default.

You can also specify `-d` to debug the daemon startup process, `-h` to display a help message, `-i` to run the issd daemon interactively, and `-p` *port_number* to specify the number of the port to use for control traffic.

The `issd` command is in the /usr/lpp/eND/iss directory.

5.3.1.1 Starting ISS on AIX

You can run the issd daemon in the foreground or background. To start the issd daemon, log in as root, enter `issd` and use any of the parameters as indicated in 5.3.1, “The issd Command” on page 89. In our case, we put our configuration file into location /usr/lpp/eND/iss and named it Issbase.conf. We wanted to also have the log file, named Issbase.conf, in the same directory. So the full command we entered was:

```
issd -c /usr/lpp//eND/iss/Issbase.conf -l /usr/lpp/eND/iss/Issbase.log
```

5.3.1.2 Starting ISS on Windows NT

Under Windows NT, the issd daemon runs as a Windows NT service, whose name is IBM_ISS_Load_Balancing. To start this service, log on as Administrator or as a user with administrative privileges, click **Start**, select **Settings**, then open the Control Panel window and double-click on **Services**. From the Services window select **IBM_ISS_Load_Balancing**, as shown in the following figure:

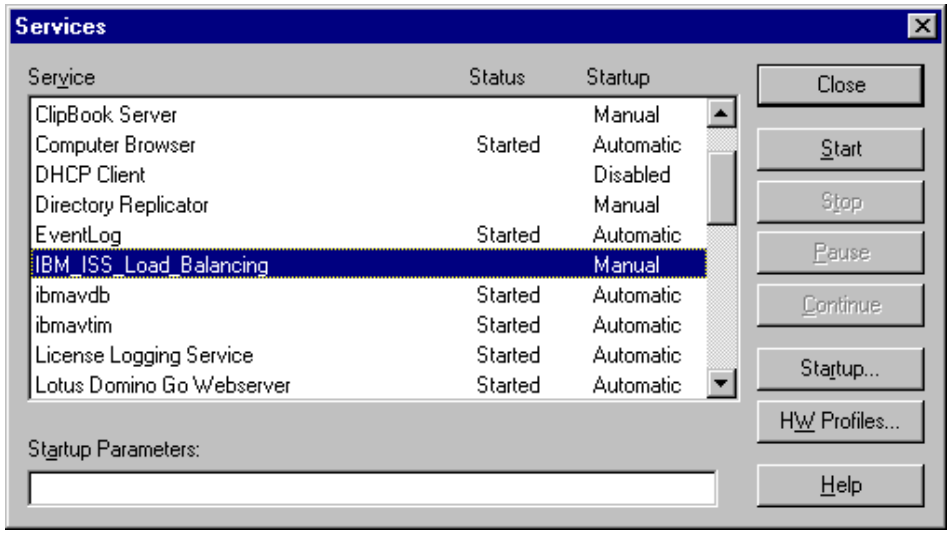


Figure 47. IBM_ISS_Load_Balancing Service on Windows NT

In the Startup Parameters text field you can specify the arguments for the `issd` command, as indicated in 5.3.1, “The issd Command” on page 89. After specifying the flags and parameters you need, click the **Start** button.

If you want to code a backslash `\`, you must specify a double backslash `\\`. For example, in one of our tests the full name of the configuration file was D:\issconf\issenzo1.cfg, and the log file we chose to use was D:\issconf\issenzo1.log, so we typed:

```
-c D:\\issconf\\issenzo1.cfg -l D:\\issconf\\issenzo1.log
```

Notice that each time you stop and restart the service, you have to re-enter the Startup Parameters. Then, if you choose to automatically start the service at the reboot by setting its Startup type to **Automatic**, you cannot decide the name and the path of your configuration and log files, but you must use the default, as indicated in 5.3.1, “The issd Command” on page 89.

5.3.2 Controlling ISS

You can control and re-configure ISS while it is running through the `isscontrol` command. Below we show an example of how you can use this command. All the details about the syntax of this command and the parameters it accepts are found in *eNetwork Dispatcher for Solaris, Windows NT, and AIX User's Guide Version 2*, GC31-8496.

For example, while ISS was running in our cell, we wanted to add the node rs600030, specifying that it would be the third in line as monitor, and that it would belong to the service WWWService. From the command line, we entered the following two commands:

```
isscontrol add node rs600030 3
isscontrol add node rs600030 service WWWService
```

Then we needed to start routing requests to this new node. To do so, we entered:

```
isscontrol start node rs600030
```

As soon as the `isscontrol` command is issued on one of the machines running the issd daemon, it propagates to the other machines running the issd daemon, so that it should not be necessary to issue it on all the machines.

The `isscontrol` command is in the /usr/lpp/eND/iss directory.

5.3.3 Stopping ISS

To stop ISS:

- On Windows NT, as administrator, from the Services window (see Figure 47 on page 91), select **IBM_ISS_Load_Balancing**, then click **Stop**. Such a button is enabled only if the service has already started.
- On AIX, log in as root and enter the command:

  `isscontrol shutdown` *`node_name`*

  where *`node_name`* is the name of the node on which you want to stop the issd daemon. For example, when we wanted to stop the issd daemon on the node rs600022 of our cell, we entered:

  ```
  isscontrol shutdown rs600022
  ```

  Note, that if you start the issd daemon interactively (using the -i parameter), you stop it with a Ctrl C.

Chapter 6. Scenario 3: WebSphere ISS and DNS

This scenario uses WebSphere ISS with a domain name service (DNS) server to balance TN3270 load among Communications Servers. In this example we are using custom metrics so we can use the sys_load script shipped with CS/AIX. The environment consists of three AIX servers, Jupiter, RS600033, and CSAIX1, each with CS/AIX running as a TN3270 Server. One of the servers, CSAIX1, has been chosen as the ISS monitor. The TN3270 load will be balanced among the three servers based on the output of the sys_load metric.

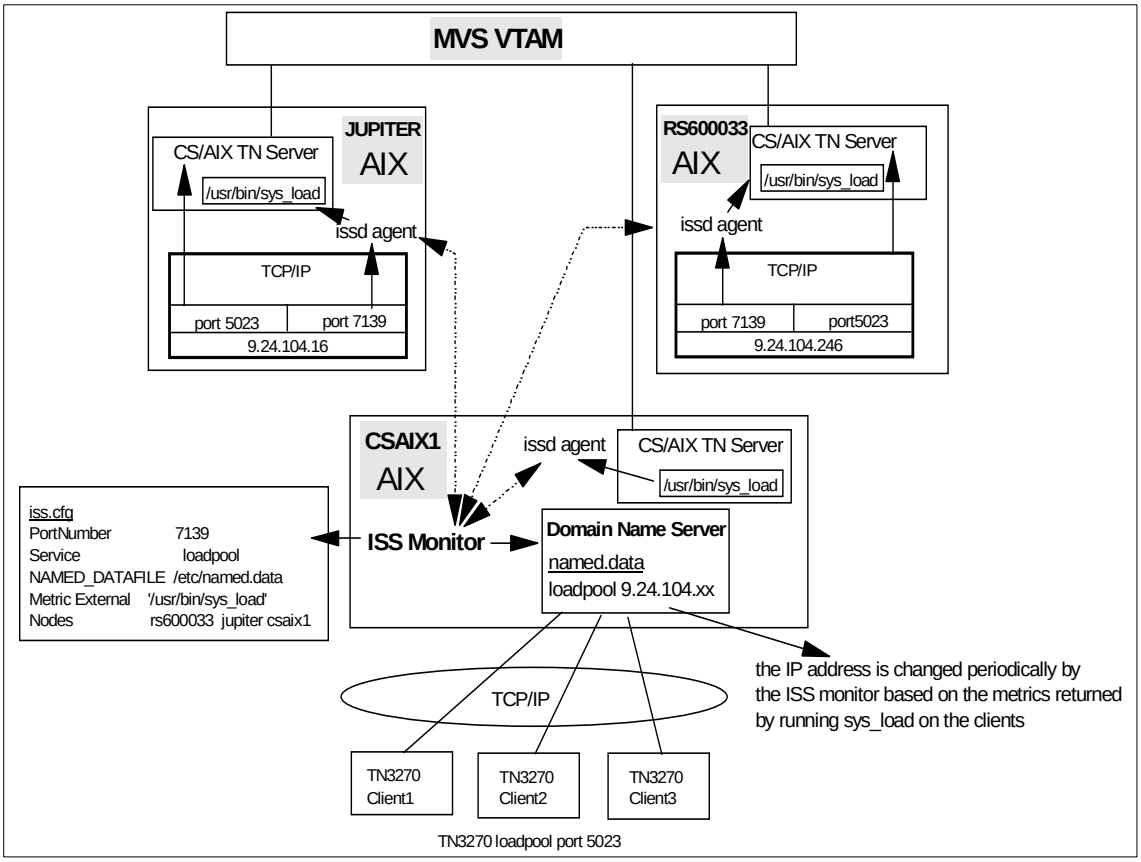


Figure 48. WebSphere ISS and DNS Scenario

The first step was to install the ISS component on each server. The installation is covered in Appendix C, “WebSphere Interactive Session Support Installation” on page 237.

6.1 ISS Configuration

Next we defined the configuration file. We used the default name and path, /etc/iss.cfg. The following describes what we defined for this scenario. There are other keywords, and there may be more to a keyword than we list. Refer to “Cell and Its Attributes” on page 80 for more information. For details on all keywords, see *eNetwork Dispatcher for Solaris, Windows NT, and AIX User’s Guide Version 2*, GC31-8496.

The following figure shows the configuration file for this scenario. Descriptions of our choices follow the figure.

```
Cell      ITSORal              local
LogLevel                       Trace
HeartbeatInterval              30
HeartbeatsPerUpdate            2
PortNumber                     7139

# Individual node data
Node csaix1                     001
Node rs600033                   002
NotMonitor
Node jupiter                    003
NotMonitor

# The service
ResourceType                    TNserver
Metric External                 /usr/bin/sys_load display
MetricLimits                    60500 90000
MetricNormalization             0 99999
Policy   Min

# followed by the cluster address and the port number.
Service                        service1 loadpool
NodeList                       rs600033 jupiter csaix1
ResourceList                   TNserver
SelectionMethod                Best

# server thread on port 53.
Nameserver                     csaix1.itso.ral.ibm.com 53
ServiceList                    service1
NamedData                      /etc/named.data
NamedRev                       /etc/named.rev
```

Figure 49. /etc/iss.cfg

6.1.0.1 Cell

The following keywords described the local cell.

Cell: Our configuration consists of a local cell called ITSORal. This is a representative name for the servers to be monitored by this machine.

LogLevel: We set this to trace. Debug seemed like too much output, but trace gave us what we needed to see what was happening.

HeartbeatInterval: This is the time, in seconds, that ISS allows to elapse before it checks the well-being of the servers (or nodes) in the cell by issuing a ping. If the server does not respond to the ping, ISS will not recommend it for the service. We chose 30 seconds.

HeartbeatsPerUpdate: This determines the number of heartbeats that will occur before the domain name server update is required. By setting this to 2, we prevented the domain name server from being updated every 30 seconds. The updates will occur every 60 seconds. This and the HeartbeatInterval are important considerations in this type of setup. Refreshing the domain name server files can create a load in itself, if this is too short a period.

PortNumber: This is the port for ISS to use to communicate between the agents and monitor. We took the value from the examples, 7139. You do not need to add this to the /etc/services file, but you should make sure no one else is using it.

Nodes: Each node is listed, with the monitor being assigned the priority 1. We specified that the others cannot be monitors by specifying Notmonitor, but if we wanted a backup, the backup should be listed with priority 2.

6.1.0.2 Resource Definition

The next keywords describe the resource to be used to monitor load on our TN3270 servers.

ResourceType: The ResourceType keyword begins the resource definition, basically defining the criteria the monitor will use to balance the servers. This name is user-defined, but must match the ResourceList keyword later. We called ours TNServer.

Metric External: This specifies that the ISS agents will run a custom script that will produce a numerical value. For this scenario, we want the ISS agents to run the /usr/bin/sys_load script on their server and return the number produced by the script.

The server LU pool is called display. Entering this as a parameter for the sys_load script makes the reporting more accurate. If you do not enter the pool name, sys_load will only look at whether LUs are available on the links and return a number indicating there are LUs available, even if the LUs are not in the TN3270 pool and cannot be used for a TN3270 session.

More information on the sys_load script can be found in Appendix A, “Determining System Load with sys_load” on page 197.

MetricLimits: The metric limit allows you to define when a server should no longer be considered for a service. The first parameter sets the point at which the resource should be returned to active participation. The second parameter sets the point at which the resource should be removed from active participation because the load has reached a critical stage.

For more information on the numerical results returned by sys_load, see Appendix A.1, “Values Returned by sys_load” on page 200.

MetricNormalization: This defines the lower and upper limits for the number coming back. Numbers out of this range will be clipped to the appropriate limit. The sys_load script returns numbers in a range from 0 to 99999.

Policy Min: In this instance, the lower the metric, the better. The server with the lowest metric will be recommended for service.

6.1.0.3 Service

The next keywords describe the service or the pool.

Service: The service defines the pool for the service. We gave it a name, service1, and defined the pool name, loadpool. The pool name must be defined in the /etc/named.data file for the DNS exactly as you have it here.

NodeList: The three nodes that provide the service are listed here.

ResourceList: This points back to the TNserver resource we defined earlier. The ISS monitor will use the resource definition we defined to determine which of the nodes listed is the best choice for incoming users.

SelectionMethod: Selection method can be RoundRobin or Best. We chose Best because we wanted the external metric values to define which server to use.

6.1.0.4 Defining the Observer

Nameserver: We will be using a TCP/IP name server for load balancing. The name given here is the name of the domain name server.

ServiceList: This is a list of all services that will send output to this observer. In our case we have only one service defined, service1. The default port number for DNS is port 53, which we verified by looking in the /etc/services table.

NamedData: This defines the exact location of the named.data file.

NamedRev: This defines the location of the named.rev file.

6.1.1 Copy the File to Each Server

The configuration file can now be copied to each server in the node list. For simplicity, we put it in the /etc directory and called it iss.cfg. This is the default. Each ISS daemon will look at the configuration and determine if it is the monitor. If not, it will accept takeover by the monitor machine.

6.2 Setting up the Domain Name Server

For this scenario we used a separate domain name server from the site server. This was for testing convenience. Each client had to point to this name server for correct name resolution. In reality, you may not want to do this. There are other choices and they are covered later in the LoadLeveler chapter. See “TCP/IP Domain Name Server” on page 118 for more information on this topic. Keep in mind that LoadLeveler uses an earlier version of ISS that required the domain name server to reside on the ISS monitor machine. This is not true for the ISS component of WebSphere.

The creation of the name server files mentioned below is also covered in a little more detail in “Configuring the TCP/IP Domain Name Server” on page 123. The domain name server set up is the same whether you are using the ISS component of LoadLeveler or WebSphere.

All the servers in our example belong to the same TCP/IP domain, itso.ral.ibm.com. This example shows what it would be like if you simply used the site domain name server. More information on configuring the name server can be found in “Configuring the TCP/IP Domain Name Server” on page 123.

The /etc/named.boot file defines the domain served by this DNS and the location of the DNS files. In our example, we are using the domain itso.ral.ibm.com for all our servers and clients.

```
; type          domain                  source file or host
;
domainitso.ral.ibm.com
primary         itso.ral.ibm.com        /etc/named.data
primary         in-addr.arpa            /etc/named.rev
```

Figure 50. /etc/named.boot on CSAIX1

The /etc/named.data file describes the servers served by this domain name server.

```
; name server data file
; NAME          TTL     CLASS   TYPE    RDATA
;
@ 9999999 IN SOA csaix1.itso.ral.ibm.com. root.csaix1.itso.ral.ibm.com. (
                                        247             ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        86400 )         ; Minimum
                9999999 IN      NS      csaix1
loopback        9999999 IN      A       127.0.0.1       ; loopback (lo0)
name/ad
dress
localhost       9999999 IN      CNAME   loopback
rs600033        9999999 IN      A       9.24.104.246
jupiter         9999999 IN      A       9.24.104.16
wtr05151        9999999 IN      A       9.24.104.45
wtr05147        9999999 IN      A       9.24.104.40
wtr05142        9999999 IN      A       9.24.104.186
csaix1          9999999 IN      A       9.24.104.53
loadpool        0       IN      A       9.24.104.53
```

Figure 51. pg named.data (Starting)

We seeded the named.data file with a line containing our service (or pool) name. Choose a valid IP address for a server in the pool.

```
; set
@  9999999 IN  SOA  csaix1.itso.ral.ibm.com. root.csaix1.itso.ral.ibm.com. (
                                        143             ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        86400 )         ; Minimum
               9999999 IN      NS      rs600033.
1.0.0.127      IN PTR loopback.itso.ral.ibm.com.
246.104.24.9   IN PTR rs600033.itso.ral.ibm.com.
16.104.24.9    IN PTR jupiter.itso.ral.ibm.com.
45.104.24.9    IN PTR wtr05151.itso.ral.ibm.com.
40.104.24.9    IN PTR wtr05147.itso.ral.ibm.com.
186.104.24.9   IN PTR wtr05142.itso.ral.ibm.com.
53.104.24.9    IN PTR csaix1.itso.ral.ibm.com.
246.104.24.9    IN PTR loadpool
```

Figure 52. /etc/named.rev

Note: We originally followed the format of the rest of this file by adding loadpool.itso.ral.ibm.com as an entry. This caused the following errors in the iss.log:

```
Nameserver: WARNING -- no loadpool entry was found in named data file
Nameserver: WARNING -- no service entry found in named data file.
```

Changing the entry to loadpool (no domain suffix) fixed the problem.

Apparently ISS is very particular about the way the name is entered in the DNS files. It must match the DNS name defined in the service statement exactly. If you enter a domain suffix in the configuration file, you must enter it that way in the DNS files. In our case, since everything was in the same domain we could leave the suffixes off.

Once the ISS and name server files are configured, we started the name server on CSAIX1:

```
startsrc -s named
```

It is very important at this point to make sure your name server is working properly. From a client that is using this name server, enter

```
nslookup poolname
```

The message coming back will indicate the domain server being used and the current IP address for the pool. This should match the IP address you defined for the pool in the /etc/named.data file.

```
F:\sg245305>nslookup loadpool.itso.ral.ibm.com
Server:  csaix1.itso.ral.ibm.com
Address:  9.24.104.53

Name:    loadpool.itso.ral.ibm.com
Address:  9.24.104.53
```

Figure 53. nslookup (Starting)

6.3 Starting the ISS Monitor and Agents

Once you are sure the domain name server is operating correctly, start the issd daemon on each system.

On each machine we used the following command to start the issd daemon.

```
/usr/lpp/eND/iss/issd -i -l /etc/iss.log
```

This started an interactive session and put the output in a log. The -i parameter gave us the trace output on the console so we could see if things were operating correctly.

The -l *log_file* parameter specifies where to send the log output.

If you did not use the default name and path for the ISS configuration file, you can use the -c *config_file* parameter to specify the correct name and path.

6.4 Load Balancing

To begin the scenario, we had the following environment:

- The issd daemon was running on all three servers.
- The named daemon (domain name server) was active on CSAIX1.
- CSAIX1 was the ISS monitor (first node in the configuration)
- CS/AIX was up and running on all three systems. The TN server on each was functional.
- CSAIX1 and JUPITER had no TN3270 sessions active.
- RS600033 had 50 TN3270 sessions active.

The following shows the log file on the ISS monitor from our scenario.

```
IBM Interactive Session Support v1.2

Licensed Materials - Property of IBM
5765-B67 (C) Copyright IBM Corporation 1993, 1997
All Rights Reserved.
US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp

ISS: Daemon started: 02/28/99 08:46:39 AM

ISS: Discovered csaix1 address 9.24.104.53
                                        Config: Validating cell ITSORal
Config: Internal addr   loopback.itso.ral.ibm.com   is   127.0.0.1
Config: Service  addr   loopback.itso.ral.ibm.com   is   127.0.0.1
Config: Internal addr   csaix1   is   9.24.104.53
Config: Service  addr   csaix1   is   9.24.104.53
Config: Internal addr   rs600033   is   9.24.104.246
Config: Service  addr   rs600033   is   9.24.104.246
Config: Internal addr   jupiter   is   9.24.104.16
Config: Service  addr   jupiter   is   9.24.104.16
Config: Added resource  TNserver  for node  csaix1
Config: Added resource  TNserver  for node  rs600033
Config: Added resource  TNserver  for node  jupiter
Config: Added service  service1  for observer  csaix1.itso.ral.ibm.com
csaix1: Beaconing election request
csaix1: Beaconing election request
UDPPort: Received  Election  packet from  rs600033
UDPPort: Received  Election  packet from  jupiter
csaix1: Beaconing election request
Agent: csaix1: Taking over as monitor
UDPPort: Received  Takeover  packet from  csaix1
Agent: csaix1: Accepting takeover by  csaix1
UDPPort: Received  Takeover  packet from  csaix1
Agent: csaix1: Accepting takeover by  csaix1
UDPPort: Received  Takeover Ack  packet from  rs600033 1
Monitor: Takeover acknowledged
UDPPort: Received  Takeover Ack  packet from  jupiter
Monitor: Takeover acknowledged
```

Figure 54. /etc/iss.log on the ISS Monitor (Part 1)

1 The ISS daemon is up and running on all the servers and they all acknowledge CSAIX1 as the ISS monitor.

```
UDPPort: Received  Takeover Ack  packet from  rs600033
Monitor: Takeover acknowledged
UDPPort: Received  Takeover Ack  packet from  jupiter
Monitor: Takeover acknowledged
csaix1: State: Running
Monitor: Sending keepalive to agent  csaix1
Monitor: Sending keepalive to agent  rs600033
Monitor: Sending keepalive to agent  jupiter
Monitor: Updating service recommendations
Monitor: All nodes for service  service1  have failed
Monitor: No overflow node available
UDPPort: Received  Heartbeat  packet from  csaix1
UDPPort: Received  ServiceInfo  packet from  csaix1
csaix1: Received recommendation of  <None>  for service
ITSORal/service1
Nameserver: Updating nameserver data file
Nameserver: Updating named data file
Agent: Measuring load on  TNserver
Agent: External metric: /usr/bin/sys_load
UDPPort: Received  ServiceInfo  packet from  csaix1
Nameserver: Nameserver: Update complete
csaix1: Received recommendation of  <None>  for service
ITSORal/service1
Nameserver: Updating nameserver data file
Nameserver: Updating named data file
Nameserver: Nameserver: Update complete
Agent: External Metric /usr/bin/sys_load: 90027.000000 2
Agent: Normalized availability: 0.099721
csaix1: Sending LoadInfo packet to csaix1
UDPPort: Received  LoadInfo  packet from  csaix1
Monitor: Load updated for csaix1/TNserver: 0.099721 3
```

Figure 55. /etc/iss.log on the ISS Monitor (Part 2)

2 The ISS agent on CSAIX1 has run the sys_load script and passes the metric back to the monitor. The value here is 90027. For more information on the values returned by sys_load see “Values Returned by sys_load” on page 200.

3 Each server is sending back the metrics and the monitor is evaluating them. The actual value can be seen on the issd console on each server.

```
UDPPort: Received  LoadInfo  packet from  rs600033
Monitor: Load updated for rs600033/TNserver: 0.099931
UDPPort: Received  LoadInfo  packet from  jupiter
Monitor: Load updated for jupiter/TNserver: 0.399374
csaix1: State: Running
csaix1: State: Running
Monitor: Sending keepalive to agent  csaix1
Monitor: Sending keepalive to agent  rs600033
Agent: Measuring load on  TNserver
Agent: External metric: /usr/bin/sys_load
UDPPort: Received  Heartbeat  packet from  csaix1
Monitor: Sending keepalive to agent  jupiter
Monitor: Updating service recommendations
Monitor: Recommending  jupiter  for service  service1 4
UDPPort: Received  ServiceInfo  packet from  csaix1
UDPPort: Received  ServiceInfo  packet from  csaix1
csaix1: Received recommendation of  jupiter  for service
ITSORal/service1
Nameserver: Updating nameserver data file 5
Nameserver: Updating named data file
Nameserver: Nameserver: Update complete
```

Figure 56. /etc/iss.log on the ISS Monitor (Part 3)

4 Based on the metrics, Jupiter is considered the best choice for new users.

5 The name server is updated to make the IP address of loadpool the IP address of Jupiter. The original named.data file has the IP address of CSAIX1 (9.24.104.53) defined as the address for loadpool. A look at the named.data file verifies that the address of Jupiter (9.24.104.16) has replaced the IP address of CSAIX1 as loadpool's IP address.

Next, we stopped the 50 TN3270 sessions on Jupiter so that no sessions existed on any of the servers.

```
csaix1: Received recommendation of  jupiter  for service
ITSORal/service1
Nameserver: Updating nameserver data file
Nameserver: Updating named data file
Agent: Measuring load on  TNserver
Agent: External metric: /usr/bin/sys_load
UDPPort: Received  ServiceInfo  packet from  csaix1
Nameserver: Nameserver: Update complete
csaix1: Received recommendation of  jupiter  for service
ITSORal/service1
Nameserver: Updating nameserver data file
Nameserver: Updating named data file
Nameserver: Nameserver: Update complete
Agent: External Metric /usr/bin/sys_load: 90027.000000
Agent: Normalized availability: 0.099721
csaix1: Sending LoadInfo packet to csaix1
UDPPort: Received  LoadInfo  packet from  csaix1
Monitor: Load updated for csaix1/TNserver: 0.099721
UDPPort: Received  LoadInfo  packet from  rs600033
Monitor: Load updated for rs600033/TNserver: 0.399944 6
UDPPort: Received  LoadInfo  packet from  jupiter
Monitor: Load updated for jupiter/TNserver: 0.399384
csaix1: State: Running
csaix1: State: Running
Monitor: Sending keepalive to agent  csaix1
Monitor: Sending keepalive to agent  rs600033
Monitor: Sending keepalive to agent  jupiter
Monitor: Updating service recommendations
Monitor: Recommending  rs600033  for service  service1 7
```

Figure 57. /etc/iss.log on the ISS Monitor (Part 4)

6 You can see that the metric is now much better for RS600033. The actual metric was 60005. That normalized to .399944. The sys_load script detected the drop in TN3270 activity and adjusted the metric accordingly.

7 RS600033 is now the best selection and is being recommended for service.

A look at the named.data file verifies that the new IP address for loadpool is the IP address of RS600033 (9.24.104.246).

Next, we started 50 TN3270 sessions to Jupiter.

```
csaix1: Received recommendation of  rs600033  for service
ITSORal/service1
Nameserver: Updating nameserver data file
Nameserver: Updating named data file
Nameserver: Nameserver: Update complete
Agent: External Metric /usr/bin/sys_load: 90026.000000
Agent: Normalized availability: 0.099731
UDPPort: Received  LoadInfo  packet from  csaix1
Monitor: Load updated for csaix1/TNserver: 0.099731
csaix1: Sending LoadInfo packet to csaix1
UDPPort: Received  LoadInfo  packet from  rs600033
Monitor: Load updated for rs600033/TNserver: 0.399934
UDPPort: Received  LoadInfo  packet from  jupiter
Monitor: Load updated for jupiter/TNserver: 0.099791 8
csaix1: State: Running
csaix1: State: Running
Monitor: Sending keepalive to agent  csaix1
Monitor: Sending keepalive to agent  rs600033
Monitor: Sending keepalive to agent  jupiter
Monitor: Updating service recommendations
Monitor: Recommending  rs600033  for service  service1
Agent: Measuring load on  TNserver
Agent: External metric: /usr/bin/sys_load
UDPPort: Received  ServiceInfo  packet from  csaix1
csaix1: Received recommendation of  rs600033  for service
ITSORal/service1
Nameserver: Updating nameserver data file
Nameserver: Updating named data file
UDPPort: Received  Heartbeat  packet from  csaix1
Nameserver: Nameserver: Update complete
UDPPort: Received  ServiceInfo  packet from  csaix1
csaix1: Received recommendation of  rs600033  for service
ITSORal/service1
Nameserver: Updating nameserver data file
Nameserver: Updating named data file
Nameserver: Nameserver: Update complete
```

8 You can see that the metric has risen for Jupiter in response to the increased load. The metric was actually 90020 (you can see this in Jupiter's iss.log), which normalized to .099791. RS600033 is still the best choice so it remains the recommendation for the service.

Note

ISS accomplishes load balancing by altering the TCP/IP DNS so the pool name used by clients to reach the host resolves to the preferred server address. This effectively means that a TCP/IP host name will resolve to different IP addresses depending on which server is the least loaded. In these scenarios, some clients tended to establish all sessions subsequent to the first on a workstation to the same IP address as the first session, bypassing the name resolution. It seems that some operating systems cache host name and IP address information. This means only the first client session on a workstation benefits from load balancing until the cache expires.

Chapter 7. Scenario 4: WebSphere ISS and Dispatcher

This scenario uses WebSphere ISS with the Dispatcher to balance TN3270 load among Communications Servers. In this example we are using custom metrics so we can use the sys_load script shipped with CS/AIX. The environment consists of three AIX servers, Jupiter, RS600033, and CSAIX1, each with CS/AIX running as a TN3270 Server. One of the servers, CSAIX1, has been chosen as the ISS monitor and the Dispatcher. The TN3270 load will be balanced among the three servers based on the output of the sys_load metric.

The difference between this scenario and scenario 3 is the observer. Scenario 3 used the domain name server as an observer, changing the IP address of the pool address to the address of the preferred server. Clients configure their TN3270 sessions to point to the pool name and use the domain name server to resolve that name to the preferred server. After that, the load balancing machine is not involved in further data transfer.

In this example, ISS will gather load information about the servers in the same way it did in scenario 3. Instead of updating the domain name server, ISS will pass this information back to the Dispatcher. Clients will configure their TN3270 sessions to point to the cluster address. The Dispatcher will receive incoming data and re-route it to the preferred server.

This scenario will illustrate the fact that you can pass ISS load information back to the Dispatcher. This information can be used alone, as in this example, or it can be used in conjunction with other factors such as advisor input and the internal counters to determine the server best suited to serve a client.

We will show you how to pass the information back. You will need to study your own environment to see if this is appropriate. You may want to consider this method if you like the ISS metric detail but do not want to build or modify a domain name server and have the load associated with the refreshes needed to maintain the domain name server. This may also solve any problems with client emulators that do not take well to having IP addresses and host names resolving differently from session to session, since the client will always be using the cluster address and host name.

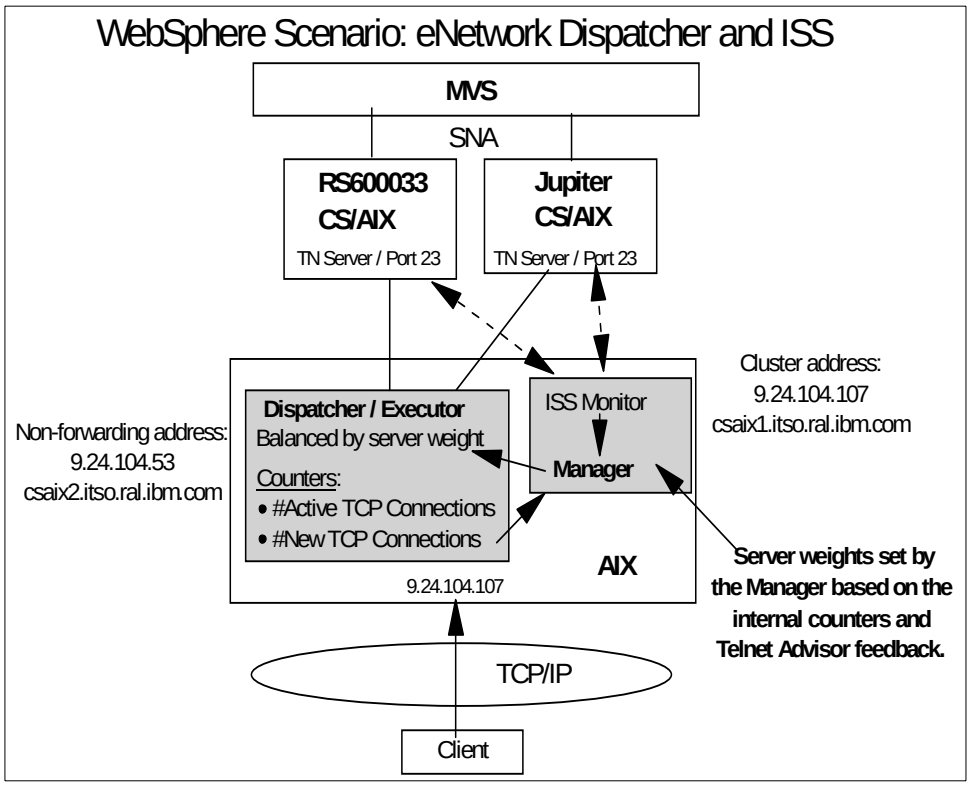


Figure 58. WebSphere ISS and Dispatcher Overview

The first step in our example was to install the ISS component on each server. The installation is covered in Appendix C, “WebSphere Interactive Session Support Installation” on page 237.

7.1 ISS Configuration

Next we defined the configuration file for ISS. We used the default name and path, /etc/iss.cfg. The following describes what we defined for this scenario. There are other keywords, and there may be more to a keyword than we list. Refer to “Cell and Its Attributes” on page 80 for more information. For details on all keywords, see the *eNetwork Dispatcher for Solaris, Windows NT, and AIX User’s Guide Version 2*, GC31-8496.

The following figure shows the configuration file for this scenario. Descriptions of our choices follow the figure.

```
#
Cell      ITSORal              local
LogLevel                       Trace
HeartbeatInterval              30
HeartbeatsPerUpdate            2
PortNumber                     7139

# Individual node data

Node csaix1                     001
Node rs600033                   002
NotMonitor
Node jupiter                    003
NotMonitor

# The resource

ResourceType                    TNserver
Metric External                 /usr/bin/sys_load display
MetricLimits                    60500 90000
MetricNormalization             0 99999
Policy   Min

# The service

Service                        service1 placeholder 9.24.104.107 5023
NodeList                       rs600033 jupiter csaix1
ResourceList                   TNserver
SelectionMethod                Best

# Observer is the Dispatcher on 9.24.104.53 port 10004
Dispatcher                     9.24.104.53  10004
ServiceList                    service1
```

Figure 59. /etc/iss.cfg

Many of the keywords are defined the same as in Chapter 6, “Scenario 3: WebSphere ISS and DNS” on page 93. Refer to “ISS Configuration” on page 108 for information on parameters not covered here.

7.1.0.1 Resource Definition

The following keywords describe the resource to be used to monitor load on our TN3270 servers.

ResourceType: The ResourceType keyword begins the resource definition, basically defining the criteria the monitor will use to balance the servers. This name is user-defined, but must match the ResourceList keyword later. We called ours TNServer.

Metric External: This specifies that the ISS agents will run a custom script that will produce a numerical value. For this scenario, we want the ISS agents to run the /usr/bin/sys_load script on their server and return the number produced by the script.

The server LU pool is called display. Entering this as a parameter for the sys_load script makes the reporting more accurate. If you do not enter the pool name, sys_load will only look at whether LUs are available on the links and return a number indicating there are LUs available, even if the LUs are not in the TN3270 pool and cannot be used for a TN3270 session.

More information on the sys_load script can be found in Appendix A, “Determining System Load with sys_load” on page 197.

MetricLimits: The metric limit allows you to define when a server should no longer be considered for a service. The first parameter sets the point at which the resource should be returned to active participation. The second parameter sets the point at which the resource should be removed from active participation because the load has reached a critical stage.

For more information on the numerical results returned by sys_load, see Appendix A.1, “Values Returned by sys_load” on page 200.

MetricNormalization: This defines the lower and upper limits for the number coming back. Numbers out of this range will be clipped to the appropriate limit. The sys_load script returns numbers in a range from 0 to 99999.

Policy Min: In this instance, the lower the metric, the better. The server with the lowest metric will be recommended for service.

7.1.0.2 Service

The next keywords describe the service or the pool.

Service: The service defines a function that relies on multiple resources, or in this case, the TNserver resource. The service name is service1. The pool name is a place holder in this case and will be ignored by ISS and the Dispatcher. The IP address and port refer to the cluster and port to which this service pertains. You can see that we have defined a cluster at 9.24.104.107 and port 5023 in Figure 60 on page 113.

NodeList: The three nodes that provide the service are listed here.

ResourceList: This points back to the TNserver resource we defined earlier. The ISS monitor will use the resource definition we defined to determine which of the nodes listed is the best choice for incoming users.

SelectionMethod: Selection method can be RoundRobin or Best. We chose Best because we want the external metric values to define which server to use.

7.1.0.3 Defining the Observer

Dispatcher: The ISS load information will be returned to the Dispatcher at 9.24.104.53 (CSAIX1) at port 10004. Port 10004 is the port used by the Dispatcher to receive metric responses from ISS. You can change this port by changing it here and by specifying the new port when you start the manager. The GUI gives you a panel to do this. The ndcontrol manager start command also allows you to change the port.

ServiceList: This is a list of all services that will send output to this observer. In our case we have only one service defined, service1.

7.1.0.4 Copy the File to Each Server

The configuration file can now be copied to each server in the node list. For simplicity, we put it in the /etc directory and called it iss.cfg. This is the default. Each ISS daemon will look at the configuration and determine if it is the monitor. If not, it will accept takeover by the monitor machine.

7.2 Starting the ISS Monitor and Agents

Once you are sure the domain name server is operating correctly, start the issd daemon on each system.

On each machine we used the following command to start the issd daemon.

```
/usr/lpp/eND/iss/issd -i -l /etc/iss.log
```

This started an interactive session and put the output in a log. The -i parameter gave us the trace output on the console so we could see if things were operating correctly.

The -l *log_file* parameter specifies where to send the log output.

If you did not use the default name and path for the ISS configuration file, you can use the -c *config_file* parameter to specify the correct name and path.

7.3 Setting up the Dispatcher

The Dispatcher setup is similar to the setup in Chapter 3, “Scenario 1: Using the WebSphere Dispatcher” on page 29 and is covered in more detail there. For the Dispatcher to use the ISS input, you must define the cluster, port, and each server in the cluster.

Start the Manager and change the proportions of importance to suit your environment. The default is that system metrics are given 0 proportion. You need to change this to a non-zero number. In our example, we are going purely by the system metrics provided by ISS in order to make the configuration clear.

Design Note

Note that when using sys_load, the Manager does not recognize that a number in the 90000 range means there are no LUs available, nor does it recognize when a server should be taken out of service. If you are using a port other than the Telnet port for your TN3270 sessions, you could create an advisor that detects no response from the port and use this as input along with ISS. If you are using the Telnet port (23) this would not really help since the Telnet stack would answer the advisor.

Once the ISS daemons have started on all the servers, including the ISS monitor, and the Dispatcher Executor and Manager have started, the Manager will begin setting the appropriate server weights by using the proportions of importance. In our case, this will be 100% ISS metric feedback. The proportions are set in the Manager Status window. You can see the proportions for this scenario in Figure 60 on page 113.

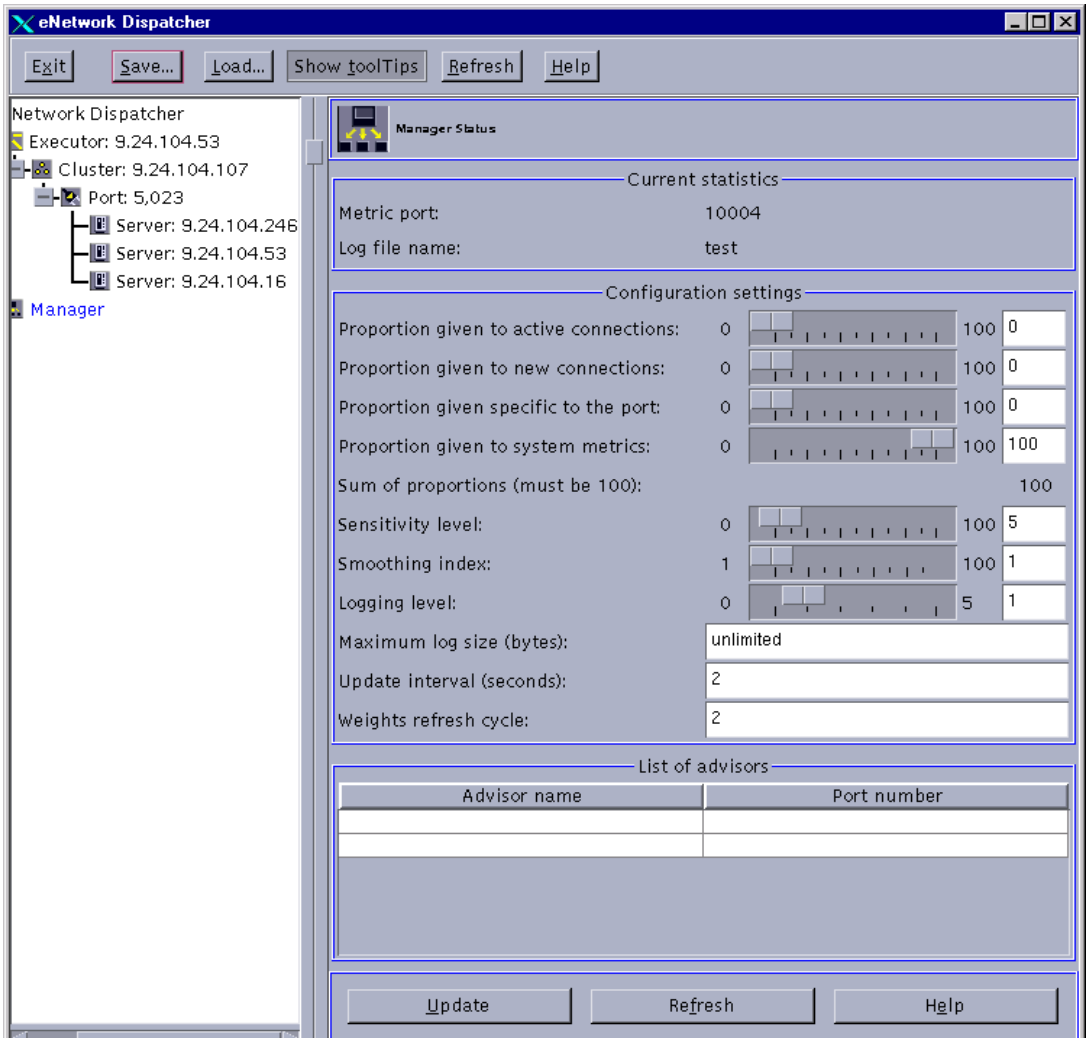


Figure 60. Dispatcher Configuration for ISS Input

To see the weights as they are being determined, click on the port entry. The weights set for this scenario are shown in Figure 61 on page 114.

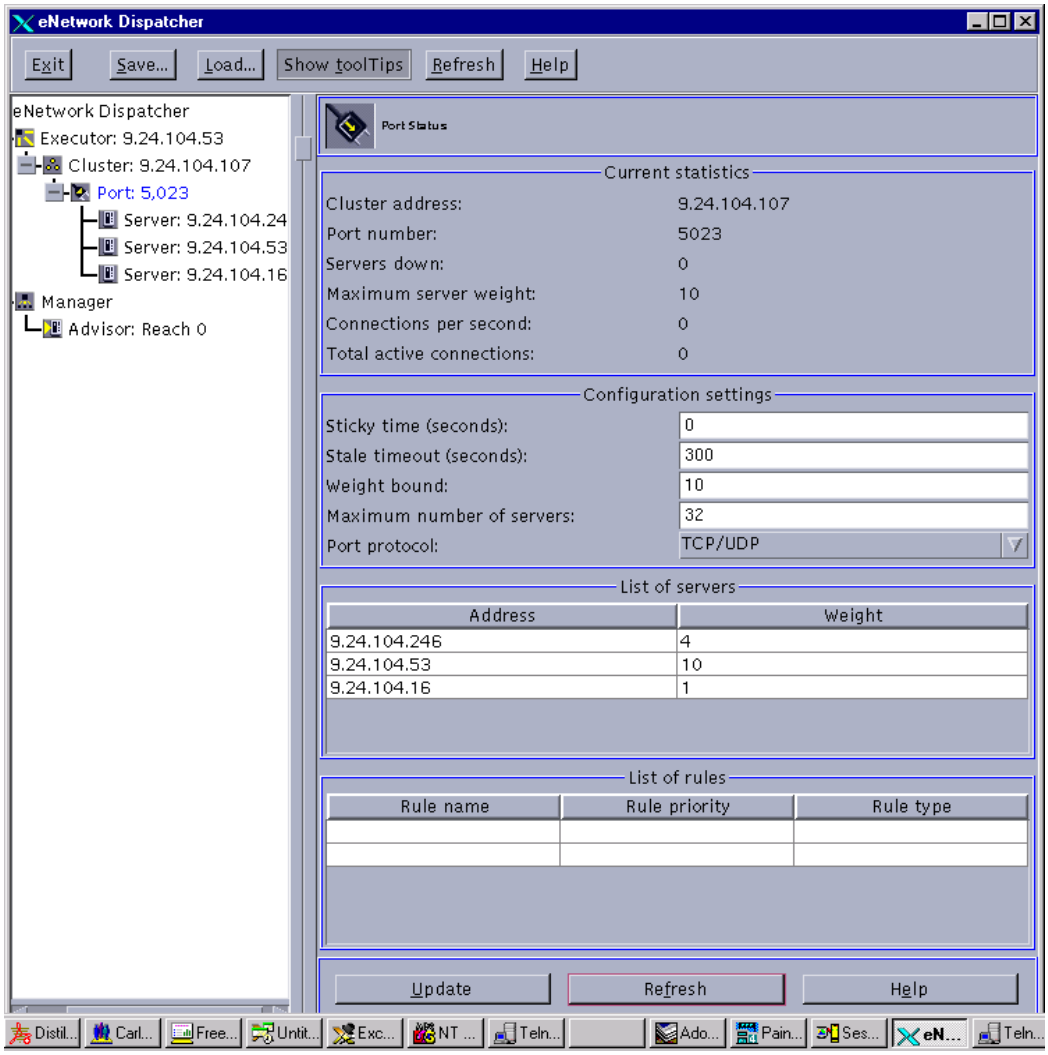


Figure 61. Server Weights

In Figure 61 the server at 9.24.104.16 has a weight of 1. In reality, the SNA node on this server has been stopped and the sys_load metric is returning a 99999, the highest (worst) metric possible in this case.

Chapter 8. CS/AIX and LoadLeveler

IBM Communications Server for AIX, when used with the Interactive Session Support (ISS) component of IBM LoadLeveler 1.3.0, offers a robust solution for controlling and balancing the workload across TN3270 Server gateways. Clients are connected to a server based on mainframe LU availability, link speed, client/mainframe connection load, overall system load, and machine processor speed. In addition to optimizing the load conditions of the servers, load balancing maintains the operation of the network during server failures by automatically re-connecting clients to an available server thereby improving system availability.

This chapter describes the system used to control and balance interactive TN3270 traffic across multiple IBM Communications Server for AIX gateways. The intent of this chapter is to thoroughly explain how load control and balancing is accomplished, and to provide enough information to configure a pool of servers and to enable load balancing. For more information on the ISS component of LoadLeveler, refer to *Using and Administering LoadLeveler,* SC23-3989.

8.1 When Would You Use the ISS Component of LoadLeveler?

LoadLeveler is only for AIX environments. IBM LoadLeveler 1.3.0 contains the ISS component, which has been removed from the later releases of LoadLeveler. The follow-on product for ISS, eNetwork Dispatcher, is now available in the WebSphere Performance Pack.

The specific implementation covered in this chapter shows the ISS component of LoadLeveler used in conjunction with a shell script shipped with Communications Server for AIX to balance TN3270 traffic across multiple Communications Server for AIX servers.

8.2 Load Balancing Overview

The IBM LoadLeveler product consists of several components. Load balancing for IBM Communications Server for AIX uses only the ISS component, which can be run independently of LoadLeveler. The function of ISS is to distribute logins and application sessions across a pool of servers in a manner that is completely transparent to end users and to applications in the network.

Without ISS, users wishing to gain access to a pool of shared computing resources from TCP/IP applications must request a connection to a specific server within the pool by supplying the machine name or Internet address that identifies that particular server. When ISS is used, a single host name is defined to represent multiple servers in a pool.

ISS uses a TCP/IP domain name server to control client connections to a server.

In the case of balancing CS/AIX TN3270 traffic, ISS works with the sys_load program shipped in CS/AIX (/usr/bin/sys_load) to monitor server load levels. As the load in each server in the pool changes, the ISS monitor changes the IP address associated with the host name representing the pool of machines in the TCP/IP name server. This results in clients using the pool name being dynamically connected to the server with the least load. According to a time interval that can be customized, the ISS monitor periodically polls each server to determine its current load. The servers communicate their load by way of a single integer ranging from 1 to 99999. Calculation of this value is covered in Appendix A, “Determining System Load with sys_load” on page 197.

A second machine can be configured to act as a standby ISS monitor, which you can start if the first monitor fails.

Figure 62 on page 117 illustrates how ISS can be used to monitor and balance the TN3270 load across multiple CS/AIX servers.

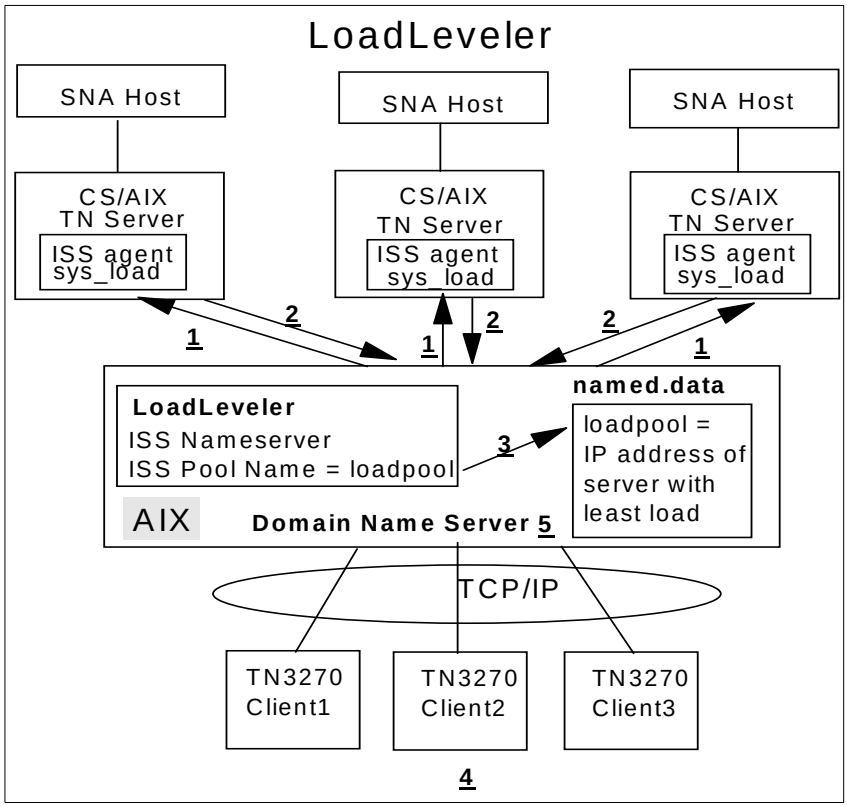


Figure 62. CS/AIX and LoadLeveler for TN3270 Sessions

1. ISS periodically polls the servers for load information. To do this, the ISS monitor connects to a user-defined TCP port. The inetd daemon on the client is listening on this same port. When it detects the ISS monitor trying to connect to this port, the inetd daemon starts the issagent process, allowing the ISS monitor and ISS agent to communicate.

2. The issagent executes the sys_load script which returns an integer that indicates the load on its server.

3. ISS updates the entry for the server pool in the domain name server named.data file with the IP address of the server with the least load.

4. The clients Telnet to the pool name (in this example, loadpool).

5. The domain name server returns the IP address in the name.data file to the clients' resolver.

8.3 TCP/IP Domain Name Server

The TCP/IP domain name servers provide name-to-IP address mapping in a TCP/IP network. ISS uses a domain name server to associate the IP address of the server with the least load with a TCP/IP host name.

A domain name server must be running on the same server with ISS. ISS determines the load on the servers to be balanced and modifies the domain name server to associate the IP address of the server with the host name of a pool.

A single host name is used to represent multiple servers in a pool. The name of the ISS pool, which the administrator defines, is the host name that the end user will Telnet to via TN3270. This name is defined as a TCP/IP host in the domain name server named.data file. During run time, ISS changes the IP address associated with the host to reflect the machine with the least overall load. The named.data file is then re-loaded by ISS and the clients connecting in will be sent to the server that will best service their connection.

When setting up a load balancing environment using ISS you can approach this in several ways.

- You can add the pool servers to the existing site name server.
- You can create a new name server if you don't have one and add the pool servers to this name server. The clients will need to be modified to point to this new name server.
- If you have a site name server but you prefer to isolate the ISS name server functions from the site name server, you can create a new name server to have authority over a new sub-domain of the main domain. The pool server machines, the name server machine, and the client machines belong to the main domain, but the host name for the pool belongs to the new sub-domain.

  In this case, clients can be configured to go to this new name server first and the new name server can be configured to pass requests for names outside the new sub-domain to an existing name server. Or, if the clients are already configured for a different name server, you can modify that name server to pass requests for the new sub-domain to the new name server.

When making your decision, keep in mind that the ISS monitor must run on the same machine as the name server and that ISS causes the name server to reload its database frequently. Also, keep in mind that the clients must be configured to point to a name server that will resolve the pool name.

8.4 Interactive Session Support (ISS)

ISS runs on the same machine that is configured as a name server. It polls all servers that are being balanced and ensures that the IP address in named.data reflects the server with the least overall load. The ISS configuration consists of the following configuration keywords:

NAME The pool name for the pool of servers.

POLL The number of seconds between polls of each server.

METRIC The metric keyword points to the name of the program shipped with CS/AIX (sys_load) that is run on each server to determine the system load. ISS contains three different metric types. Load balancing for CS/AIX uses the metric type called CUSTOM.

POLICY This is set to MIN or MAX. As previously mentioned, the servers communicate their system load to the ISS via a positive integer between 1 and 99999. A policy of MAX means that higher numbers are more favorable. Load balancing for CS/AIX uses a policy of MIN.

SERVERS A list of host names or IP addresses for each server in your pool. The ISS balances the load across each of these servers.

Upon startup of ISS, the ISS monitor connects to the ISS agent in each of the servers defined in order to run the sys_load program. Each server communicates its load to ISS, which then updates named.data with the IP address of the server with the least load. ISS then goes to sleep. After the number of seconds defined by the POLL keyword, ISS wakes up and does the following:

- Updates named.data with the IP address of the machine that returned the lowest number
- Re-polls each server to get overall system load

This process continues throughout run time.

8.5 Setting Up Load Balancing across Multiple Servers

The following is a brief overview of the steps required to set up ISS load balancing across multiple Communications Servers. They will be covered in more detail in the following sections.

- Select a machine that will act as the name server and ISS monitor.

- Install LoadLeveler 1.3 Base on the ISS monitor server
- Configure ISS on the ISS monitor server
- Configure the TCP/IP name server on the ISS monitor server
- Update the /etc/services and /inetd.conf file on all servers
- Copy the issagent executable to all machines that will be in the load balancing pool.

8.5.1 Installation of the LoadLeveler 1.3 Base

LoadLeveler will install in the /usr/lpp/LoadL directory. The installation will show that the following files were successfully applied:

- LoadL.postscript
- LoadL.nfs_so
- LoadL.nfs
- LoadL.iss
- LoadL.info

The installation level will be 1.3.0.0 and there are fixes available that you should install immediately afterward. Contact IBM Service for the latest available fixes.

8.5.2 Configuring ISS

The sample configuration file for ISS is in /usr/lpp/LoadL/iss/Iss_config. You may use this file or create a new file for the configuration. If you create a new name, you can specify the correct configuration file when you start the ISS monitor. Modify this file and code the following parameters for each pool of servers you configure:

NAME The single alias host name to which your clients will issue Telnet 3270 requests. Select the name of your choice. This will match the pool name in the name server.

POLL The number of seconds after which the ISS will wake up to tabulate the weights (1-99999) received from each server as well as to refresh the named.data file with this information. It is recommended that this be set to 30 - 40 seconds. The higher the poll is set, the less often the ISS updates the named.data file and the less current the IP address associated with the host name is. This means that clients may not get connected to the server best able to serve their connections. Setting it to low will result in extra overhead on the name server machine as well as on each of the server machines. It will also result in thrashing between the name

server (running ISS) and each of the servers; polling commands and responses will continuously flow back and forth. Set this field to what seems appropriate. It can be fine tuned later.

METRIC The type of balancing to do. CS/AIX uses the CUSTOM option. ISS will periodically (as determined by the POLL setting) cause the ISS agent to run the sys_load software on each server to determine its overall load. The directory and name must match the location of the script on the remote server. The sys_load script is covered in more detail in Appendix A, “Determining System Load with sys_load” on page 197. Code the following here:

```
CUSTOM '/usr/bin/sys_load'
```

POLICY Defines whether higher or lower numbers are favorable. Set this to MIN, which means the lower the number the more favorable (the least overall load).

SERVER A list of all host names or IP addresses in the pool of servers you are balancing across.

8.5.2.1 Sample Iss_config File:

In /usr/lpp/LoadL/iss you will find the sample configuration file called lss_config. We copied the file into a file called csaix_config that we used for our setup.

What you see in Figure 63 on page 122 is a copy of our modified configuration file csaix_config. We took out everything that was not relevant to our configuration but left the remarks and added some for your reference.

```
; Start of sample config file:
;
; The following keywords and values control the overall behaviour of ISS.
; They affect all pools. MONITOR_TRACE turns on trace level output on the
; monitor machine if present. All output on the monitor machine is produced
; via stdout. AGENT_TRACE turns on trace level output on all servers that are
; in custom pools, and thus run an agent. If AGENT_TRACE is specified then
; AGENT_LOG must be present and provide the name of a log file on the agent
; machines. There is an optional second parameter to AGENT_LOG which limits
; the size to which the log file can grow. When the log file reaches that
; size it is moved to <filename>.old, and a new log file is created. ISS_PORT
; specifies the port number on which the servers in custom pools (which will
; run agents) have issagent configured.

MONITOR_TRACE
AGENT_TRACE
AGENT_LOG /tmp/issagent.log 10000
ISS_PORT 10001


; loadpool is a custom pool whose domain name is stored in /etc/named.data and
; /etc/named.rev. It has three servers, whose domain names are rs600033, jupiter and csaix1.
; It measures the number of processes running on each of these servers by having
; the agents issue the custom program, sys_load, on the different server machines.
; sys_load is included as part of the Communications Server AIX V.5 source code.
;
; Monitor intervals: The configuration is set to monitor the issagents once
; every 2*HEARTBEAT_INTERVAL seconds, and HEARTBEAT_INTERVAL is configured
; to be 30 secs during most of the day, but only 10 secs between 8am and 10am.
; Once every HEARTBEAT_INTERVAL seconds ISS checks that the server whose
; address is currently recommended is functional at its ICMP layer.
;
; If at any time there are no servers available, the name loadpool
; will be removed from the DNS files and the pool alarm triggered, which in
; this case makes an entry into a file called /tmp/issalarm.log

NAME                    loadpool
NAMED_DATAFILE          /etc/named.data
NAMED_REVERSEFILE       /etc/named.rev
METRIC                  CUSTOM '/usr/bin/sys_load'
POLICY                  MIN
SERVERS                 rs600033 jupiter csaix1
HEARTBEAT_INTERVAL      0(30) 8(10) 10(30)
HEARTBEATS_PER_UPDATE   2
POOL_ALARM echo "loadpool ran out of servers at `date`" >> /tmp/issalarm.log

; end of sample config file
```

Figure 63. Sample Iss_config File

8.5.3 Configuring the TCP/IP Domain Name Server

The following steps are performed on the machine with LoadLeveler installed.

8.5.3.1 Creating the TCP/IP Name Server Files

If you do not already have a domain name server on the ISS monitor server you will have to create three files:

- **named.data** - This is your name server data base. This file tells the name server what dotted-decimal address to return in response to a request to convert a machine name into an internet address.
- **named.rev** - This file ensures that reverse name resolution requests (dotted-decimal to machine name) which are outside the domain of the ISS name server are passed on to the main site name server for resolving.
- **named.boot** - Read by the named daemon upon startup, it points to the named.data and named.rev files you created.

You can create skeletons of the named.data and the named.rev using the hosts.awk and addrs.awk system scripts.

1. Depending on what level of AIX you are running, your scripts may be in a different location. To find out where they are, use the following command:

   ```
   lslpp -f 'bos.net*' | grep awk
   ```

   In our system the files were located in:

   - /usr/samples/tcpip/hosts.awk
   - /usr/samples/tcpip/addrs.awk

2. Before you run these scripts, you will want to edit your /etc/hosts file to make sure it has only the hosts you want in your DNS server. Since the scripts build their output using the hosts file, it pays to clean up first because it will minimize the size of the named.dat and named.rev files.
3. Edit the sample named.boot file to set up the location of the named.data and named.rev files, as well as the name of the sub-domain. In our example we put all the DNS config files in the /etc directory. There is a sample named.boot file you can use as a template that is installed with the LoadLeveler product. You will find it in:

   /usr/lpp/LoadL/iss/named.boot

   When done editing, file it in the /etc directory as named.boot.

   The following figure is an example named.boot file:

```
; type          domain                  source file or host
;
domain          carla.itso.ral.ibm.com
primary         carla.itso.ral.ibm.com        /etc/named.data
primary         in-addr.arpa                  /etc/named.rev
```

Figure 64. /etc/named.boot

Notice that the domain name is actually the sub-domain name which we called carla.itso.ral.ibm.com. Don't worry about the in-addr.arpa. Just leave it as you find it in the sample file. You must specify the correct path for the named.data file and named.rev file. In this case the files are in the /etc directory.

4. Run the two .awk scripts to create the files. Here are the command lines we used:

   ```
   /usr/samples/tcpip/hosts.awk /etc/hosts > /etc/named.data
   /usr/samples/tcpip/addrs.awk /etc/hosts > /etc/named.rev
   ```

   The named.data and named.rev files created will still need some editing.

5. Edit named.data.

   Here are the main changes you will need to make:

   - Add the ISS pool name you configured after the NAME keyword in the lss_config file as a TCP/IP host name. The ISS pool name used in our sample configuration is loadpool.
   - Change the SOA stanza to point to your new DNS. The SOA record defines the sub-domain configured.
   - Change the NS stanza to the name of your new DNS. This is the actual host name running the sub-domain name server. In our case it was rs600033.

   The following figure shows an example named.data file:

```
; (also see /etc/named.boot)
;
; NAME          TTL     CLASS   TYPE    RDATA
;
; setting default domain to "carla.itso.ral.ibm.com"
;
@ 9999999 IN SOA rs600033.carla.itso.ral.ibm.com. root.rs600033.carla.itso.ral.ibm.com. (
                                        97              ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        86400 )         ; Minimum
                9999999 IN      NS      rs600033
loopback        9999999 IN      A       127.0.0.1
localhost       9999999 IN      CNAME   loopback
rs600033        9999999 IN      A       9.24.104.246
jupiter         9999999 IN      A       9.24.104.16
wtr05151        9999999 IN      A       9.24.104.45
wtr05147        9999999 IN      A       9.24.104.40
wtr05142        9999999 IN      A       9.24.104.186
csaix1          9999999 IN      A       9.24.104.53
loadpool        0       IN      A       9.24.104.53
```

Figure 65. named.data File

Note: Make sure the SOA line is all on one line. The most important entry is the last entry: loadpool. This is the host name alias. That is the entry that LoadLeveler will go in and change as the load of the servers change.

6. Edit named.rev

   This file is used if the DNS gets a request for the machine name of a system when the dotted-decimal address is given.

   Make sure you change the SOA and the NS stanza according to what you did in the named.data file. Your pool name does not have to be added in here but for completeness, add the pool name as it shows in the configuration file. Choose a valid IP address to use (from the list of pool servers).

   The following figure is an example named.rev file:

```
@  9999999 IN SOA rs600033.carla.itso.ral.ibm.com. root.rs600033.carla.itso.ral.ibm.com. (
                      100            ; Serial
                      3600           ; Refresh
                      300            ; Retry
                      3600000        ; Expire
                      86400 )        ; Minimum
          9999999 IN      NS      rs600033.
1.0.0.127       IN PTR loopback.carla.itso.ral.ibm.com.
246.104.24.9    IN PTR rs600033.carla.itso.ral.ibm.com.
16.104.24.9     IN PTR jupiter.carla.itso.ral.ibm.com.
45.104.24.9     IN PTR wtr05151.carla.itso.ral.ibm.com.
40.104.24.9     IN PTR wtr05147.carla.itso.ral.ibm.com.
186.104.24.9    IN PTR wtr05142.carla.itso.ral.ibm.com.
53.104.24.9     IN PTR csaix1.carla.itso.ral.ibm.com.
53.104.24.9     IN PTR loadpool
```

Figure 66. named.rev File

8.5.4 Other Important Things to Do

There are a few things to be modified before the name server can be started. Basically you set up dedicated ports on all the servers so that the monitor can open a pipe with the agent.You will be editing two files on each of the machines that will act as a server. First is the /etc/services file. Generally this is because if a socket request comes into a machine, the port number that is part of the request is used as an index into /etc/services to figure out what the port is being used for. The entry (in our case iss_port) is then used to look in inetd.conf (which is the second file you need to update) to find out the particulars of the service and how to start it if it is not already running.

So here is what you need to do:

1. Edit the /etc/services file on each of the server machines where the issagent will run. Add a port entry for iss_port. Actually the name doesn't matter, just make sure you pick a port number that isn't in use on any of the machines.

   On our machines the entries looked like this:

   ```
   iss_port        10001/tcp
   ```

2. Edit the /etc/inetd.conf file on each of the server machines where the issagent code will be run. Add a definition for iss_port or whatever you called it in the /etc/services file.

   On our machines the entries looked like this:

```
iss_port stream tcp     nowait  root    /usr/lpp/LoadL/iss/issagent issagent
```

   Make sure you refresh inetd after updating the file by typing

   ```
   refresh -s inetd
   ```

in a command line.

3. Copy the issagent executable onto the machines that will be load balanced into the /usr/lpp/LoadL/iss directory or the directory you put in inetd.conf file.

   Check the permissions for issagent on each machine to make sure it is executable. If not issue `chmod 755 issagent`.

4. Tell your DNS machine that it is a DNS machine by updating your /etc/resolv.conf file.

   This is how our resolv.conf file looked after updating it:

   ```
   nameserver      9.24.104.246
   domain  carla.itso.ral.ibm.com
   ```

5. Ensure that your TCP/IP clients will use the correct name server for host name to IP address resolution. If you have updated your main site name server to point to the newly configured name server, no changes are necessary on the client machines. If this name server is your only name server, or you have opted to make it your main name server, then you must update each client machine to point to it. On AIX you would do this in the /etc/resolv.conf file. In Windows NT you would modify the Network settings for the TCP/IP protocol from the control panel.

8.5.5 Starting the Configuration

8.5.5.1 Starting the Name Server

On a command line enter the command

```
startsrc -s named
```

If you later want to stop it, you can issue:

```
stopsrc -s named
```

If you make some changes while it's running and just want to refresh it:

```
refresh -s named
```

8.5.5.2 Starting Communications Server for AIX

Start IBM Communications Server for AIX on each server in the pool.

8.5.5.3 Starting ISS

From the root id on the load balancing machine, start the ISS monitor by typing the following command on a command line:

```
/usr/lpp/LoadL/iss/issmonitor /usr/lpp/LoadL/iss/Iss_config
```

In our example it would have been csaix_config since we are not using Iss_config. The output messages produced by the ISS monitor are sent to the the screen.

For debug purposes, or to have a hard copy of the monitor output you can send the messages to a file when you start the ISS monitor. The following command shows starting the ISS monitor and piping the output to a file called /tmp/isstrace.log.

```
/usr/lpp/LoadL/iss/issmonitor /usr/lpp/LoadL/iss/Iss_config | tee /tmp/isstrace.log
```

After ISS starts, you should immediately see each server being polled and an integer indicating its overall load being returned. If this is occurring, load balancing has been successfully enabled. As each client connects, and the load begins to vary on each server, you will see the ISS monitor changing the IP address associated with your pool host name. This is your indication that all is working well.

The metric used by the load balancing sys_load software is very granular. It is very unlikely that multiple machines will return the same number, unless Communications Server for AIX is not running in which case 99999 would be returned. In the rare event that the same number is returned by two servers and these servers happen to have the least load, the server that appears earlier in the server list as defined in the Iss_config file will be selected.

8.6 LoadLeveler ISS Example Network

This is an example network we set up in our lab. There are three AIX machines, JUPITER, RS600033, and CSAIX1, running as TN3270 servers. RS600033 will also act as an ISS monitor to balance the TN3270 load across the three servers.

In this example, there is only one TCP/IP name server, with no sub-domains. The clients point to this name server.

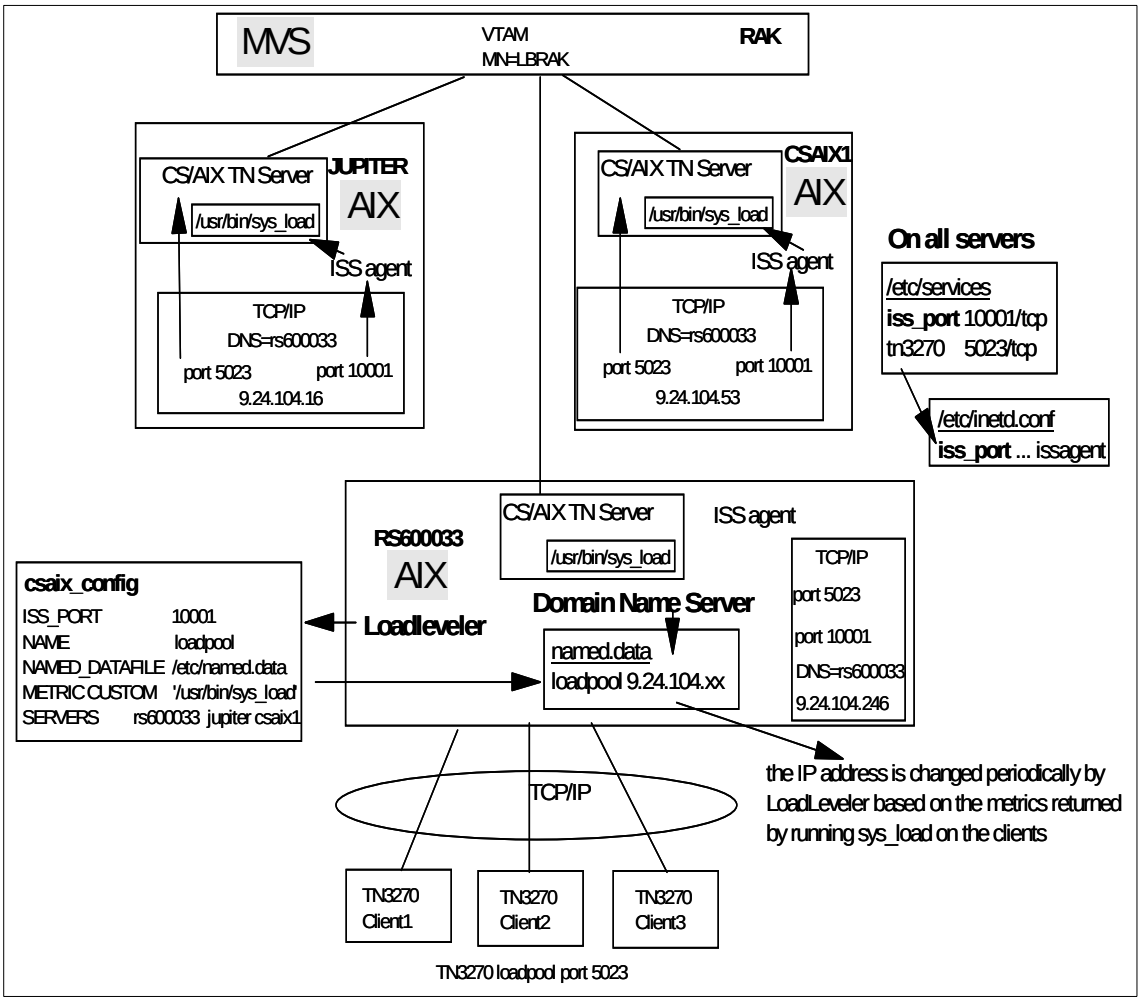


Figure 67. LoadLeveler Example

8.6.1 RS600033

In our test network, RS60003 was the ISS Monitor, the TCP/IP name server, and also a CS/AIX TN3270 node.

8.6.1.1 TCP/IP Files

The first step in setting up the test network was to decide how to configure the TCP/IP name server environment. This would determine what we put in the basic TCP/IP files.

We chose to create a new DNS for this scenario and have the clients point to it. This was the simplest way to accomplish our task. In reality, we chose a name that could be used as a sub-domain of our existing network. The only thing missing was the update to the existing site DNS to pass requests for host names in our sub-domain to our new DNS. If we had done this, the clients would not have had to point to a new or additional name server. The options for creating the DNS environment are covered in “TCP/IP Domain Name Server” on page 118.

/etc/services

The /etc/services setup is the same on all machines. The iss_port entry determines the port to be used for the ISS monitor and ISS agent to communicate. It can be any port not in use. The TN3270 port was added by the network administrator during the setup of CS/AIX TN3270.

```
iss_port        10001/tcp
tn3270          5023/tcp
```

Figure 68. /etc/services

/etc/inetd.conf

The /etc/inetd.conf file must be edited on each machine to add an entry for iss_port. This is what tells the inetd daemon which process to start when a connection is made to the iss_port defined in the /etc/services file. The connection between the two files is the name iss_port.

```
iss_port stream tcp nowait  root /usr/lpp/LoadL/iss/issagent issagent
```

Figure 69. /etc/inetd.conf

/etc/hosts

In general, you do not need to manually edit the /etc/hosts file. It will normally be filled in during TCP/IP setup and can be edited to create a connection between a machine name and an IP address. This is useful if you frequently access a machine that is not found in any TCP/IP domain name servers you access.

This file is also used when you run the hosts.awk and addrs.awk scripts to generate files for a TCP/IP name server (see the discussion in “Configuring the TCP/IP Domain Name Server” on page 123). For this reason we updated this file manually to include the servers we wanted included in the domain name server files. We found the format was easier to deal with here.

```
127.0.0.1 loopback localhost      # loopback (lo0) name/address
9.24.104.246    rs600033
9.24.104.16     jupiter
9.24.104.45     wtr05151
9.24.104.40     wtr05147
9.24.104.186    wtr05142
9.24.104.53     csaix1
```

Figure 70. /etc/hosts

/etc/resolv.conf

We updated the /etc/resolv.conf to point to this machine (RS600033) as the DNS.

```
nameserver      9.24.104.246
domain  carla.itso.ral.ibm.com
```

Figure 71. /etc/resolv.conf

8.6.1.2 TCP/IP Name Server Files

The next step was to set up the TCP/IP name server on RS600033. In our example there is one TCP/IP name server and all servers are in the carla.itso.ral.ibm.com domain. The clients will all need to point to this name server.

/etc/named.boot

We created a /etc/named.boot file to define the domain name and the location of the named.data and named.rev files. A sample named.boot file is in /usr/lpp/LoadL/iss/named.boot.

```
domain          carla.itso.ral.ibm.com
primary         carla.itso.ral.ibm.com        /etc/named.data
primary         in-addr.arpa                  /etc/named.rev
```

Figure 72. /etc/named.boot

/etc/named.data

We used hosts.awk to create the skeleton for named.data. This file contains all the servers in the domain. We edited the file as described in “Configuring the TCP/IP Domain Name Server” on page 123. The entry added for the ISS

pool is called loadpool. When adding the pool name, you choose any one of the IP addresses of the servers in the pool as the IP address. During the operation of the ISS monitor, this IP address will be updated in the file to reflect the address of the server with the least load.

```
; (also see /etc/named.boot)
;
; NAME          TTL     CLASS   TYPE    RDATA
;
; setting default domain to "carla.itso.ral.ibm.com"
;
@ 9999999 IN SOA rs600033.carla.itso.ral.ibm.com. root.rs600033.carla.itso.ral.ibm.com. (
                                        97              ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        86400 )         ; Minimum
                9999999 IN      NS      rs600033
loopback        9999999 IN      A       127.0.0.1
localhost       9999999 IN      CNAME   loopback
rs600033        9999999 IN      A       9.24.104.246
jupiter         9999999 IN      A       9.24.104.16
wtr05151        9999999 IN      A       9.24.104.45
wtr05147        9999999 IN      A       9.24.104.40
wtr05142        9999999 IN      A       9.24.104.186
csaix1          9999999 IN      A       9.24.104.53
loadpool        0       IN      A       9.24.104.53
```

Figure 73. /etc/named.data

/etc/named.rev

We used the addrs.awk script to generate the /etc/named.rev file and edited it according the directions in “Configuring the TCP/IP Domain Name Server” on page 123.

```
@  9999999 IN SOA rs600033.carla.itso.ral.ibm.com.
              root.rs600033.carla.itso.ral.ibm.com. (
                      100             ; Serial
                      3600            ; Refresh
                      300             ; Retry
                      3600000         ; Expire
                      86400 )         ; Minimum
         9999999 IN      NS      rs600033.
1.0.0.127       IN PTR loopback.carla.itso.ral.ibm.com.
246.104.24.9    IN PTR rs600033.carla.itso.ral.ibm.com.
16.104.24.9     IN PTR jupiter.carla.itso.ral.ibm.com.
45.104.24.9     IN PTR wtr05151.carla.itso.ral.ibm.com.
40.104.24.9     IN PTR wtr05147.carla.itso.ral.ibm.com.
186.104.24.9    IN PTR wtr05142.carla.itso.ral.ibm.com.
53.104.24.9     IN PTR csaix1.carla.itso.ral.ibm.com.
53.104.24.9     IN PTR loadpool
```

Figure 74. /etc/named.rev

8.6.1.3 LoadLeveler ISS Component

The ISS component can run independently of the LoadLeveler product and this is what we chose to do. The setup is quick and simple, consisting of creating a configuration file, using /usr/lpp/LoadL/iss/Iss_config as a template. For a description of this file see “Configuring ISS” on page 120.

/usr/lpp/LoadL/iss/csaix_config

```
MONITOR_TRACE
AGENT_TRACE
AGENT_LOG /tmp/issagent.log 10000
ISS_PORT 10001
NAME                    loadpool
NAMED_DATAFILE          /etc/named.data
NAMED_REVERSEFILE       /etc/named.rev
METRIC                  CUSTOM '/usr/bin/sys_load'
POLICY                  MIN
SERVERS                 rs600033 jupiter csaix1
HEARTBEAT_INTERVAL      0(30) 8(10) 10(30)
HEARTBEATS_PER_UPDATE   2
POOL_ALARM echo "loadpool ran out of servers at `date`" >>
```

Figure 75. /usr/lpp/LoadL/iss/csaix_config

issagent

The issagent code(/usr/lpp/LoadL/iss/issagent) was installed on RS600033 during the install of the LoadLeveler code.

We had to use the chmod command to make sure the permissions were set at 755.

8.6.2 JUPITER and CSAIX1

JUPITER and CSAIX1 are servers running CS/AIX V5 as a TN3270 Server. As servers to be load balanced, the changes to them were minimal.

8.6.2.1 /etc/services

The /etc/services file has been edited on each machine to add two user-defined ports. The TN3270 port was added by the network administrator during the configuration of CS/AIX TN3270. The iss_port entry was added during the configuration of the LoadLeveler ISS component. As you will see, this is done exactly the same on each server involved in the load balancing setup.

```
iss_port        10001/tcp
tn3270          5023/tcp
```

Figure 76. /etc/services

8.6.2.2 /etc/inetd.conf

The /etc/inetd.conf file must be edited on each machine to add an entry for iss_port. This is what tells the inetd daemon what process to start when a connection is made to the iss_port defined in the /etc/services file. The connection between the two files is the name iss_port.

```
iss_port stream tcp nowait  root /usr/lpp/LoadL/iss/issagent issagent
```

Figure 77. /etc/inetd.conf

8.6.2.3 Copy issagent

We used FTP to copy the issagent code from the ISS Monitor with this base code to the other servers. The inetd.conf file on each server specifies the location of the file so make sure you copy it to the directory specified in inetd.conf.

```
/usr/lpp/LoadL/iss/issagent
```

Once this was done, we had to use the chmod command to make sure the permissions were set at 755. Before this was done, the issagent was not

executable and our ISS monitor polling process did not work. This was tricky because there was no indication of what failed.

8.6.2.4 CS/AIX Files

CS/AIX naturally had to be configured to handle TN3270 traffic. The /usr/bin/sys_load file was installed during CS/AIX installation. You will need to look at the script to determine if you need any modifications. For more information on the sys_load script see Appendix A, “Determining System Load with sys_load” on page 197.

8.6.3 ISS Monitor Output

The following example monitor output shows the test network functioning correctly. You can use it as a guide to what your output should look like.

```
MONITOR_TRACE activated.
AGENT_TRACE activated.
All agents will use /tmp/issagent.log for trace/debug output
Agent log file size limited to 10000 Bytes
All CUSTOM servers should have issagent on port 10001
------------------------------START OF POOL-------------------------
Pool name:<loadpool>
DNS data file path: </etc/>
DNS data file name: <named.data>
DNS reverse file path: </etc/>
DNS reverse file name: <named.rev>
Custom metric: <'/usr/bin/sys_load'>
policy <MIN>
server:<rs600033>
server:<jupiter>
server:<csaix1>
heartbeats_per_update:<2>
pool_alarm: <echo "loadpool ran out of servers at `date`" >>
/tmp/issalarm.log>
ISSconnect(): connect...
ISSconnect(): connect rc=0
StartAgent(): connected to server
about to write...
write rc=300
```

```
StartAgent(): back from send...
ISSconnect(): connect...
ISSconnect(): connect rc=0
StartAgent(): connected to server
about to write...
write rc=300
StartAgent(): back from send...
ISSconnect(): connect...
ISSconnect(): connect rc=0
StartAgent(): connected to server
about to write...
write rc=300
StartAgent(): back from send...
--------Configuration complete - Pools being started up--------
Perform DNS update for pool loadpool, reason=2, code=1
Pool loadpool being removed from named
Pool loadpool being removed from reverse name file
WARNING:No loadpool entry was found in reverse name file
...nameserver updated with SIGHUP
Wait (1.099988 sec): Data received
------------Thu Feb 11 12:16:02 1999---------------
StartAck received
HandleStartAck(): invoked
HandleStartAck(): updating agent array
agent sockaddr: address=9.24.104.53, port=2059
Reseting configured flag for pool loadpool and server csaix1
HandleStartAck(): noting time
Wait (1.086653 sec): Data received
------------Thu Feb 11 12:16:03 1999---------------
StartAck received
HandleStartAck(): invoked
HandleStartAck(): updating agent array
agent sockaddr: address=9.24.104.246, port=38760
Reseting configured flag for pool loadpool and server rs600033
HandleStartAck(): noting time
Wait (1.072906 sec): Data received
------------Thu Feb 11 12:16:03 1999---------------
StartAck received
HandleStartAck(): invoked
HandleStartAck(): updating agent array
agent sockaddr: address=9.24.104.16, port=36446
```

```
Reseting configured flag for pool loadpool and server jupiter
HandleStartAck(): noting time
Wait (0.965800 sec): Pool requires service
------------Thu Feb 11 12:16:04 1999---------------
Service pool Named_Update_Pool
...wait time after SIGHUP complete
Wait (-0.100859 sec): Pool requires service

------------Thu Feb 11 12:16:04 1999---------------
Service pool loadpool
Issue heartbeat
empty ranking table => no servers available
Send ConfigurePool to server rs600033 (pool loadpool)
ISSconnect(): connect...
ISSconnect(): connect rc=0
Send ConfigurePool to server jupiter (pool loadpool)
ISSconnect(): connect...
ISSconnect(): connect rc=0
Send ConfigurePool to server csaix1 (pool loadpool)
ISSconnect(): connect...
ISSconnect(): connect rc=0
Wait (29.999920 sec): Data received
------------Thu Feb 11 12:16:06 1999---------------
MetricResult received
Received load figure:
Pool:Server loadpool:csaix1
LOAD=>60005
Wait (27.547785 sec): Data received
------------Thu Feb 11 12:16:07 1999---------------
MetricResult received
Received load figure:
Pool:Server loadpool:rs600033
LOAD=>90008
Wait (26.936323 sec): Data received
------------Thu Feb 11 12:16:09 1999---------------
MetricResult received
Received load figure:
Pool:Server loadpool:jupiter
LOAD=>90019
Wait (24.737620 sec): Pool requires service
```

```
------------Thu Feb 11 12:16:34 1999--------------
Service pool loadpool
Issue heartbeat
RankServers(): current server ranking:
csaix1(9.24.104.53):(60005)
rs600033(9.24.104.246):(90008)
jupiter(9.24.104.16):(90019)
Perform DNS update for pool loadpool, reason=3, code=0
Pool loadpool is coming online with server <9.24.104.53>
Address: 9.24.104.53--->9.24.104.53
Pool loadpool being activated in reverse name file (server=9.24.104.53)
WARNING:No loadpool entry was found in reverse name file
...named files updated without SIGHUP
Special Pool Named_Update_Pool added to Queue
Wait (1.999986 sec): Pool requires service
------------Thu Feb 11 12:16:34 1999--------------
Service pool Named_Update_Pool
...nameserver updated with SIGHUP
Wait (1.099994 sec): Pool requires service
------------Thu Feb 11 12:16:35 1999--------------
Service pool Named_Update_Pool
...wait time after SIGHUP complete
Wait (28.899465 sec): Pool requires service
------------Thu Feb 11 12:17:04 1999--------------
Service pool loadpool
Issue heartbeat
Wait (29.999967 sec): Data received
------------Thu Feb 11 12:17:08 1999--------------
MetricResult received
Received load figure:
Pool:Server loadpool:jupiter
LOAD=>90020
Wait (25.630497 sec): Pool requires service
```

```
------------Thu Feb 11 12:17:34 1999--------------
Service pool loadpool
Issue heartbeat
RankServers(): current server ranking:
csaix1(9.24.104.53):(60005)
rs600033(9.24.104.246):(90008)
jupiter(9.24.104.16):(90020)
Wait (29.999968 sec): Pool requires service
------------Thu Feb 11 12:18:04 1999--------------
Service pool loadpool
Issue heartbeat
Wait (29.999961 sec): Data received
------------Thu Feb 11 12:18:07 1999--------------
MetricResult received
Received load figure:
Pool:Server loadpool:jupiter
LOAD=>90019
Wait (26.403766 sec): Pool requires service
------------Thu Feb 11 12:18:34 1999--------------
Service pool loadpool
Issue heartbeat
RankServers(): current server ranking:
csaix1(9.24.104.53):(60005)
rs600033(9.24.104.246):(90008)
jupiter(9.24.104.16):(90019)
Wait (29.999965 sec): Pool requires service
------------Thu Feb 11 12:19:04 1999--------------
Service pool loadpool
Issue heartbeat
Wait (29.999966 sec): Pool requires service
------------Thu Feb 11 12:19:34 1999--------------
Service pool loadpool
Issue heartbeat
Wait (29.999960 sec): Pool requires service
------------Thu Feb 11 12:20:04 1999--------------
Service pool loadpool
Issue heartbeat
Wait (29.999962 sec): Pool requires service
------------Thu Feb 11 12:20:34 1999--------------
Service pool loadpool
Issue heartbeat
Wait (29.999964 sec): Pool requires service
```

```
------------Thu Feb 11 12:21:04 1999--------------
Service pool loadpool
Issue heartbeat
Wait (29.999965 sec): Data received
------------Thu Feb 11 12:21:04 1999--------------
MetricResult received
Received load figure:
Pool:Server loadpool:rs600033
LOAD=>90007
Wait (29.253220 sec): Pool requires service
------------Thu Feb 11 12:21:34 1999--------------
Service pool loadpool
Issue heartbeat
RankServers(): current server ranking:
csaix1(9.24.104.53):(60005)
rs600033(9.24.104.246):(90007)
jupiter(9.24.104.16):(90019)
Wait (29.999960 sec): Data received
------------Thu Feb 11 12:22:03 1999--------------
Lonely received
HandleLonely(): invoked
HandleLonely(): addr=9.24.104.53, port=2059
HandleLonely(): noting contact
Wait (0.853769 sec): Pool requires service
------------Thu Feb 11 12:22:04 1999--------------
Service pool loadpool
Issue heartbeat
Wait (29.999965 sec): Data received
------------Thu Feb 11 12:22:04 1999--------------
MetricResult received
Received load figure:
Pool:Server loadpool:rs600033
LOAD=>90008
Wait (29.502981 sec): Pool requires service
------------Thu Feb 11 12:22:34 1999--------------
Service pool loadpool
Issue heartbeat
RankServers(): current server ranking:
csaix1(9.24.104.53):(60005)
rs600033(9.24.104.246):(90008)
jupiter(9.24.104.16):(90019)
Wait (29.999961 sec): Data received
```

Chapter 9. Server Load Balancing Using SLP

IBM eNetwork Communications Server for Windows NT (CS/NT) Version 6.0 and IntranetWare for SAA 3.0 have been enhanced to provide server load balancing support for applications running on client workstations. In this chapter, we provide an overview of this load balancing function and how it can be implemented and used by client workstations. In this chapter we will discuss load balancing CS/NT servers with Service Location Protocol (SLP).

More information about CS/NT and the SNA API client can be found in *IBM eNetwork Communications Server for Windows NT Version 5.0,* SG24-2099 and *IBM eNetwork Communications Server for Windows NT Version 6.0 Enhancements,* SG24-5232.

9.1 Overview

The server load balancing function dynamically balances SNA sessions for dependent LUs (LU 0 to 3) and independent LU 6.2 (APPC and CPI-C) sessions by distributing them to communications servers with the least load. To achieve this, CS/NT implements the Service Location Protocol (SLP) designed by the Internet Engineering Task Force (IETF). SLP is described in RFC 2165.

With CS/NT, load balancing is available for the following clients:

- CS/NT SNA API clients running LU 0 to 3 (LUA) sessions. This support includes 3270 emulation sessions.
- CS/NT SNA API clients running client/server APPC and CPI-C applications.
- IBM eNetwork Personal Communications (PCOMM) 4.3
- Independent Software Vendors (ISV) QEL/MU 3270 sessions (NetWare).
- Third party 3270 emulators that connect over TCP/IP, TN3270, and TN5250 client sessions.

CS/NT registers and advertises services that contain load factors. This allows a client workstation or the server itself to gather this load information from other servers, organize it into an ordered list and make a decision to select the server that satisfies the client request.

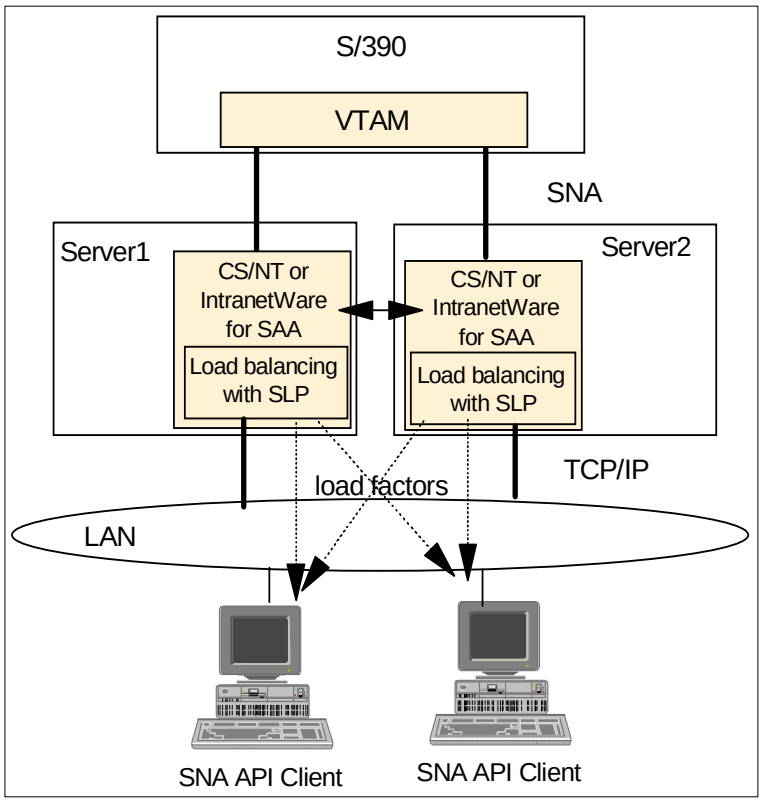


Figure 78. Load Balancing

Server session load balancing allows system administrators to have more flexibility in the way they configure and administrate the SNA networks. SNA administrators can rely on CS/NT to distribute the load of dependent and independent LU sessions. This way, client workstations request a specific service, whereas in previous releases, the administrator needed to configure each client machine to connect through a specific server.

Clients and servers communicate using TCP/IP. The server establishes an SNA session to the host on behalf of the client.

9.2 Implementation

CS/NT has implemented server load balancing for traditional LU sessions (LUA) and APPC sessions. However, load balancing for 3270 sessions differs

slightly from the implementation of load balancing for APPC sessions. In this section, we describe how load balancing works for LUA sessions which includes 3270 emulation sessions as well as for APPC sessions which includes 5250 emulation sessions.

9.2.1 Dependent LU Sessions

For traditional LUA sessions, such as 3270 sessions, the load is distributed across servers within a named LU pool and a named scope. The server with the least overall load within the scope supporting the LU pool will likely satisfy a connection request. For example, an SNA API client workstation can possibly have several 3270 sessions to multiple servers.

The load for dependent LUs represents a percentage of available resources from a particular server. The load percentage is calculated by dividing the number of active application connections by the total number of LUs (0 to 3) available.

You can influence the calculated load by specifying an LU0- to-3 load factor (host session load factor) to compensate for differences between two servers, such as available memory, processor speed, and CPU utilization.

9.2.2 Independent LU Sessions

For APPC sessions, the load is distributed across all available servers in a named scope. The initial connection that a workstation client establishes determines the server over which all subsequent APPC sessions will flow. For example, an SNA API client workstation will have all its APPC sessions use the same initially selected server.

The load for LU6.2 represents a percentage of available resources from a particular server. The load percentage is calculated by dividing the total number of conversations over all local LUs on a particular server by the cumulative maximum session limit for all local LUs. The maximum session limit is the LU 6.2 session limit specified during configuration. If the maximum session limit is specified as zero (0), indicating there is no session limit, the default maximum local LU session limit of 512 per local LU is used when the load is calculated. The default maximum local LU session limit can also be specified during configuration.

You can influence the calculated load by specifying an LU6.2 load factor (APPC session load factor) to compensate for differences between two servers, such as available memory, processor speed, and CPU utilization.

9.2.3 Scopes

Server configuration involves defining scope names. A scope in this context is the same as the SLP scope defined in RFC 2165. Scopes are used to partition access to servers and for administrative control purposes. A scope name is used by the server to advertise its services.

Client workstations will use a scope name to select one or more servers that provide the requested services. For example, you can define the same scope name in all servers providing 3270 session level encryption. By using the same scope name, client workstations can select encrypted sessions for their 3270 emulation session.

Servers can be in one scope, more than one scope (a maximum of 10), or can be unscoped. Clients must be configured for a single scope or through unscoped servers.

When designing your client/server network keep the following in mind:

- If a server is unscoped, it replies to SLP scoped and unscoped requests.
- If the SNA API client is configured to connect to unscoped servers, only unscoped servers will reply.
- Clients can reach the SNA network through servers that are configured with the same scope or are unscoped.

9.3 Design Tips

When designing a network for SLP load balancing there are several things to keep in mind. Some, such as planning for scopes, have been covered in other sections of this chapter. The following are a few more things to keep in mind.

Are You Using the Integrated Logon Feature?
If you are using the integrated logon feature in SNA API clients, it is recommended that you define a primary domain controller and backup domain servers in Windows NT. This allows you to centralize the management of user IDs in a single place.

Do You Have SLP Directory Agents in Your Network?
In addition to user agents (the SNA API client) and service agents (CS/NT server), the SLP RFC defines a directory agent that collects information from service agents to provide a single repository of service information. If directory agents are present in the network, they should be configured to handle the same scopes as are configured for the Communications Servers.

If unscoped services are to be used in a network with directory agents present, at least one unscoped directory agent should be configured.

Does Your Network Span Subnets?

If a directory agent exists, the client will unicast a request to it to resolve a request. If no directory agent exists (or the client has not found it yet), the client will multicast a request to the service specific multicast address. The service agents listen for multicasts and will respond if they can satisfy the request. This has implications in a network that spans subnets. Any routers in the network will need to be enabled for multicast.

SNA API Client/Server Options for LUA Load Balancing

In our test lab, we tried several combinations of client and server options; both to see if a client connection could be made, and whether that connection benefited from load balancing. The following table will provide you with information to consider when designing your network for LUA load balancing:

Table 3. SNA API LUA Client Settings

SNA API Client Settings	Requests Load Balanced?	Server A Settings	Will the Client Connect to Server A?
Scope X, pool name	Yes, among servers that are a part of Scope X	Part of Scope X	Yes, if it makes sense according to load balancing
Scope X, pool name	Yes, among servers that are a part of Scope X	Unscoped	No
Server A, pool name	No	Part of Scope X	Yes, absolutely
Server A, pool name	No	Unscoped	Yes, absolutely
Server *, pool name	Yes, among all scoped and unscoped servers	Part of Scope X	Yes, if it makes sense according to load balancing
Server *, pool name	Yes, among all scoped and unscoped servers	Unscoped	Yes, if it makes sense according to load balancing
Blank scope, pool name	Yes, among all unscoped servers	Part of Scope X	No

SNA API Client Settings	Requests Load Balanced?	Server A Settings	Will the Client Connect to Server A?
Blank scope, pool name	Yes, among all unscoped servers	Unscoped	Yes, if it makes sense according to load balancing

SNA API Client / Server Options for APPC Load Balancing

APPC load balancing does not involve the clients specifying a pool name. Load balancing must be enabled by checking a box in the configuration panel. You will have one APPC connection per workstation and all sessions for the client use that one.

The following table will help you determine how to design your client / server load balancing environment for APPC.

Table 4. SNA API LU6.2 Client Settings

SNA API Client Settings	Requests Load Balanced?	Server A Settings	Will the Client Connect to Server A?
Scope X	Yes, among servers that are a part of Scope X	Part of Scope X	Yes, if it makes sense according to load balancing
Scope X	Yes, among servers that are a part of Scope X	Unscoped	No
Server A	No [1]	Part of Scope X	Yes, absolutely
Server A	No [1]	Unscoped	Yes, absolutely
Server *	Yes, among all scoped and unscoped servers	Part of Scope X	Yes, if it makes sense according to load balancing
Server *	Yes, among all scoped and unscoped servers	Unscoped	Yes, if it makes sense according to load balancing
Blank scope	Yes, among all unscoped servers	Part of Scope X	No

SNA API Client Settings	Requests Load Balanced?	Server A Settings	Will the Client Connect to Server A?
Blank scope, pool name	Yes, among all unscoped servers	Unscoped	Yes, if it makes sense according to load balancing

[1] Theoretically the connection is load balanced because the Load Balancing box is checked. However, the connection will always go to server A.

9.4 Server Configuration

Figure 79 on page 147 shows an overview of the load balancing function. You may want to refer to it as you go through the examples discussed in this chapter.

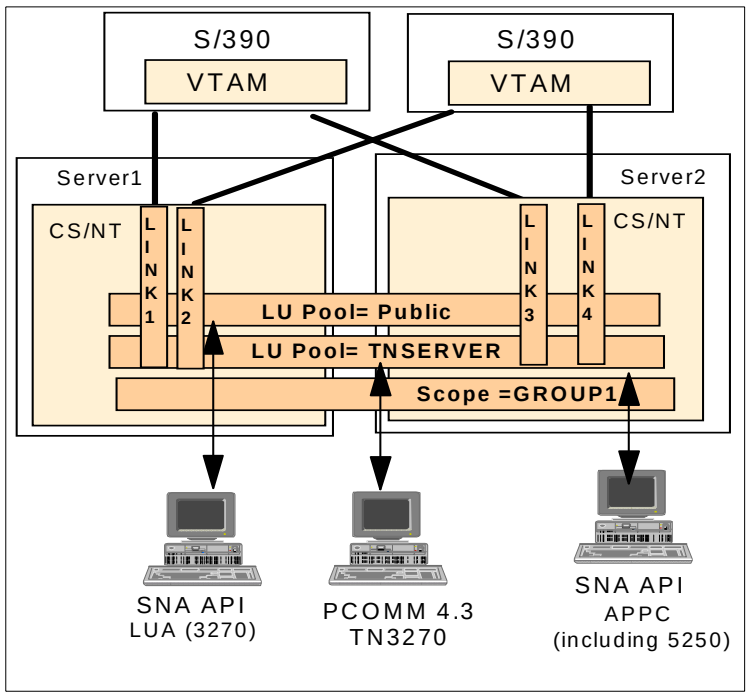


Figure 79. Load Balancing with Pools and Scopes

The first step in configuring the CS/NT server is to set up the host links, services (for example, TN3270 Server) and the LU pools. Each server to be in the scope should have one or more host links defined with LUs on each link.

An LU pool must be defined for the LUA clients that includes LUs from these host links. The LU pool name must be the same on all servers in the scope. Next you should configure the load balancing function.

The server load balancing function is common to the SNA API client services, the TN3270E Server and the TN5250 Server. To configure the server for SLP load balancing you would select the **Load Balancing** option shown in Figure 80.

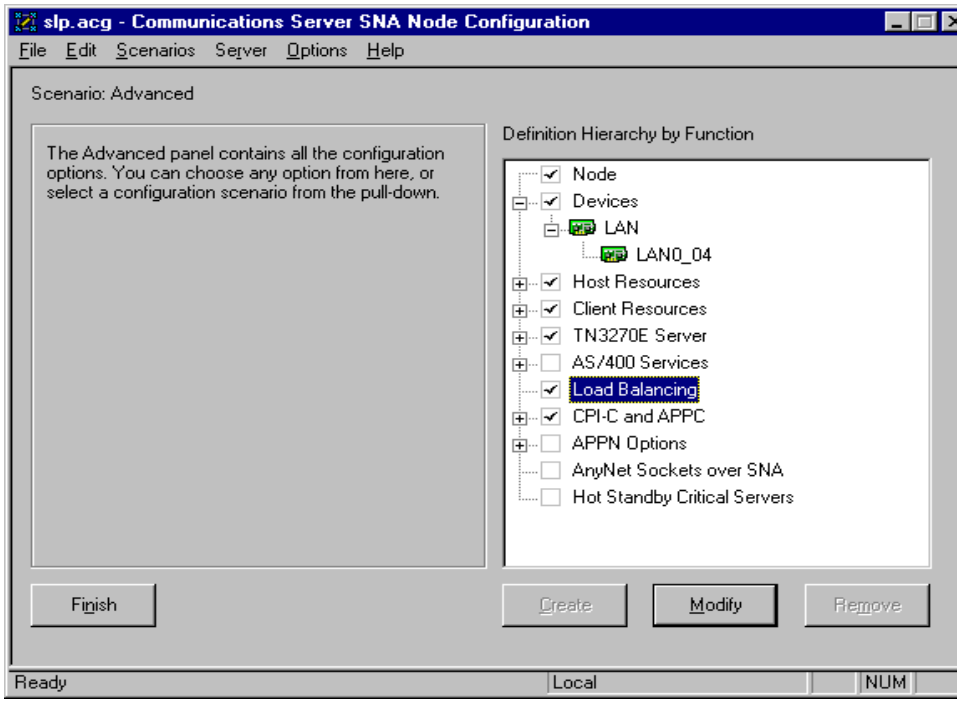


Figure 80. CS/NT Server Configuration for Load Balancing

Selecting the Load Balancing option will take you to the panel shown in Figure 81 on page 149.

The configuration screen allows you to select and configure the following:

1. Enable the load balancing check box. In the Load Balancing screen, you need to select the Enable load balancing check box in order to activate the function.
2. Add scope name or names for the server. A scope name is used by the server to advertise its services. Client workstations will use a scope name to select one or more servers that provide the requested services. For

example, you can define the same scope name in all servers providing 3270 session level encryption. By using the same scope name, client workstations can select encrypted sessions for their 3270 emulation sessions.

3. Change the default values if required.

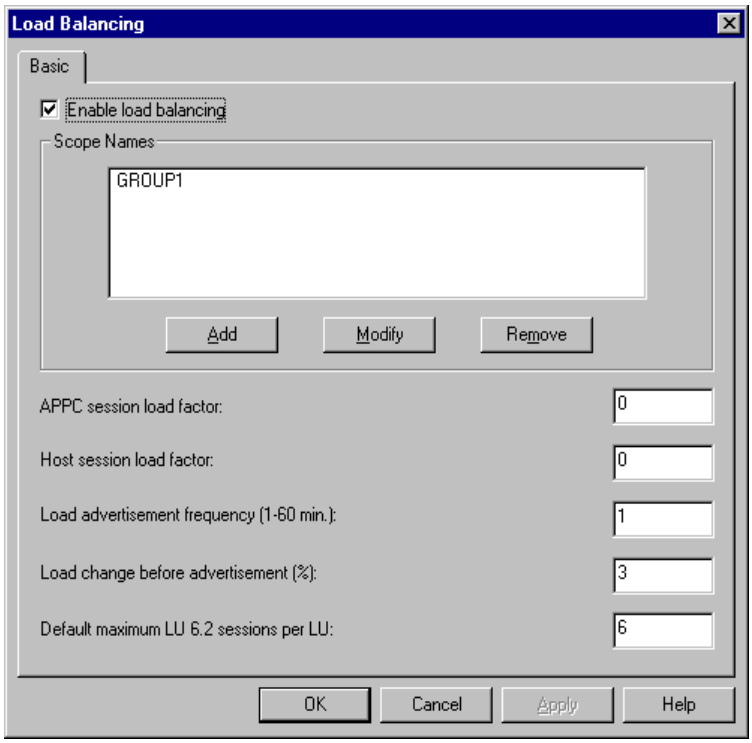


Figure 81. Load Balancing - Server Configuration Panel

9.4.1 Server Load Balancing Parameters

The following parameters allow you to configure the load balancing function in the server:

Scope Names: This field is used to indicate that this server participates in the load balancing process within that scope. A scope is the same as a named group. A server can participate in a maximum of 10 scopes, or it can be unscoped (by leaving this field blank). Clients reach the SNA network through servers that are configured with the same scope or that are unscoped. Clients can be configured to participate in load balancing through a single scope or through unscoped servers.

APPC session load factor: APPC session load factor specifies the factor used when the APPC session load for the server is calculated. Specifying a negative number decreases the calculated session load, and specifying a positive number increases the calculated session load. For example, if this server has a relatively slow CPU, you can increase the load factor to decrease the number of sessions the server manages. The range is an integer value between -100 and 100; the default is 0.

Host session load factor: Host session load factor specifies the factor used when the host session load for the server is calculated. Specifying a negative number decreases the calculated session load, and specifying a positive number increases the calculated session load. For example, if this server has a relatively slow CPU, you can increase the load factor to decrease the number of sessions the server manages. The range is an integer value between -100 and 100; the default is 0.

Load advertisement frequency: This field indicates how often the server checks the APPC and host session loads to determine if the Load change before advertisement threshold has been reached. The measure of this field is in minutes. The range is between one and 60 minutes; the default value is one minute.

Load change before advertisement: This field indicates by how much the APPC and the host session load must change, as a percentage, before the load information is updated. Only when this threshold is reached will the load information be changed. The range is between 0 and 100. The default is 3%.

Default maximum LU 6.2 sessions per LU: This field describes the default maximum number of independent LU 6.2 sessions allowed per LU. This value is used when a maximum is not specified in the LU definition itself. Default value is 512 sessions.

9.4.2 Sample Server Configuration

In our scenario we configured two servers to provide SNA API client services support for 3270 sessions using the LU2 protocol and 5250 sessions using the LU 6.2 protocol. The server was also configured to provide TN server support for TN3270 clients. Both servers have the scope name GROUP1.

In this configuration, we changed the default maximum number of APPC sessions to a value of six sessions. The objective of this was to be able to easily monitor the calculated load factor using the node operations facility.

The result of this configuration is shown in the ACG configuration file in Figure 82.

```
LOAD_BALANCING=(
     ADVERTISE_FREQUENCY=1         <==== check every minute
     APPC_LU_LOAD_FACTOR=0         <==== initial APPC load factor
     DEFAULT_MAX_LU62_SESSIONS=6   <==== default max APPC sessions
     ENABLE_LOAD_BALANCING=1       <==== enable the function
     HOST_LU_LOAD_FACTOR=0         <==== initial LUA load factor
     LOAD_VARIANCE=3               <==== update factor if 3% or more
     SCOPE_NAME=GROUP1             <==== scope name
)
```

Figure 82. Load Balancing - ACG Configuration File

9.5 SNA API Client Configuration for LUA Sessions

The clients will need to install the SNA API client from the CS/NT CD-ROM. If you are using PCOMM as the emulator, it should be installed after the SNA API client.

You will need to configure the client workstation to use the load balancing function. In this section, we show you the configuration of an SNA API client workstation to access two servers with load balancing for 3270 sessions.

To reach the panel shown in Figure 84 on page 152 on Windows NT we selected **Start->Programs->IBM Communications Server SNA Client->Local INI Configuration**.

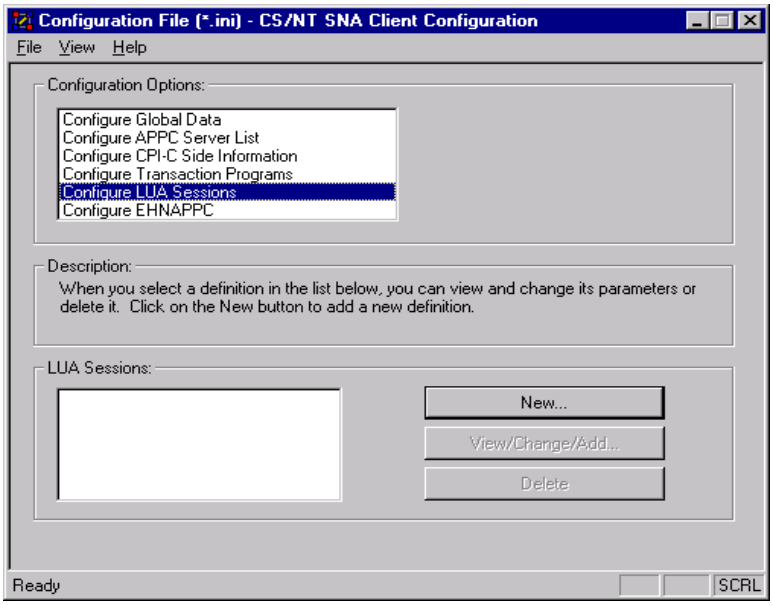


Figure 83. SNA API Client Configuration

Choose **Configure LUA Sessions** and click **New** to get to the next panel shown in the following figure.

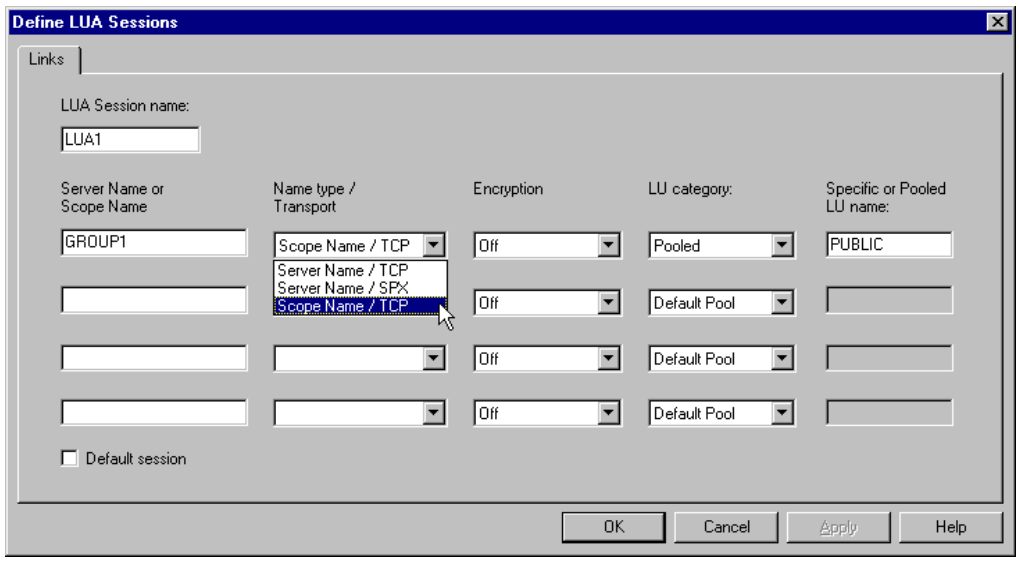


Figure 84. SNA API Client - Load Balancing for LUA Sessions

The following parameters, seen in Figure 84, allow you to configure the load balancing function for LU sessions in the SNA API client workstation:

Server Name or **Scope Name:** This field is needed to participate in load balancing. It indicates that this client's sessions are load balanced across servers within the specified scope name. If you enter an asterisk (*), the client participates in load balancing but connects through unscoped servers. If there is a server listed, the server is used and there is no load balancing.

Name type/Transport: In this field you select the protocol that will be used to transport SNA traffic between the client and the SNA communications server. In this scenario, we select the option **Scope Name/TCP**.

LU Category: If you select **Pooled** in this field, you need to specify the name of the LU pool from which an LU is to be chosen. If a pooled LU is specified with a server name of asterisk (*) or a scope, the SNA connection will be made to the least loaded compatible server and then the connection will be load balanced.

Figure 85 shows the INI configuration file for an LUA session in this client. It shows the configured scope name GROUP1.

```
[LUX_PROFILE]
PROFILE_NAME=LUA1
SESSION=*,*,0,PUBLIC,TCP,GROUP1,ENC_OFF
SESSION=,*,1,,TCP,,ENC_OFF
SESSION=,*,1,,TCP,,ENC_OFF
SESSION=,*,1,,TCP,,ENC_OFF
```

Figure 85. LUA Load Balancing Configuration File - INI File

9.5.1 3270 Session Configuration

In this section we show you the configuration of a 3270 session in PCOMM 4.2. PCOMM 4.2 does not have native SLP support so load balancing is achieved by using the CS/NT SNA API interface. PCOMM 4.3 can use either the native SLP support and TN3270, or the CS/NT SNA API support.

To use the LUA session previously defined in the SNA API client workstation we start the configuration by selecting **Start->Programs->IBM Personal Communications->Start or Configure a Session**.

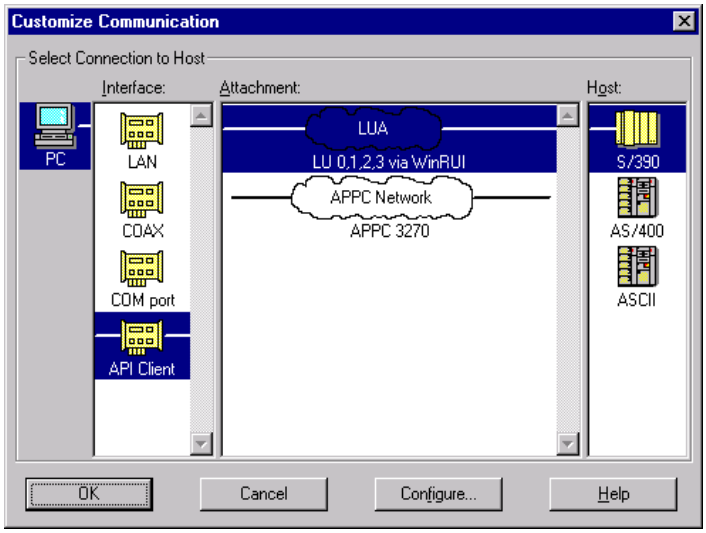


Figure 86. PCOMM 4.2 Configuration for SNA API

Choose the API client interface and the LUA attachment. Click on **Configure**.

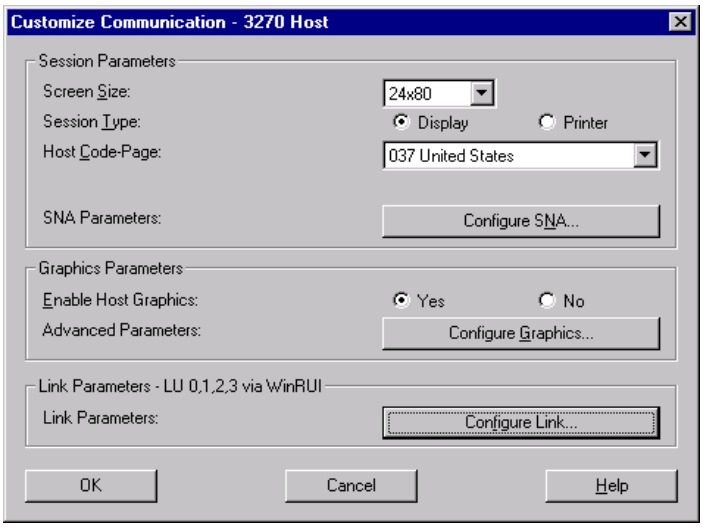


Figure 87. PCOMM 4.2 Configuration for SNA API

Change any options desired in the panel shown in Figure 87 and click **Configure Link** to see the next panel.

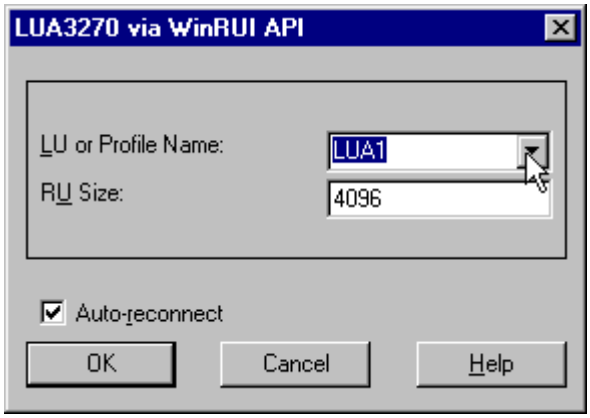


Figure 88. PCOMM 4.2 Configuration for SNA AP

The LU or Profile Name button has a pull-down showing the LU options. You should see the LUA session name (LUA1) defined earlier in Figure 84 on page 152. Choose this name and click **OK**.

Starting the PCOMM session will establish a session to the host using an LU in the public pool on the least loaded server in the GROUP1 domain.

9.6 Client Configuration for APPC Sessions

You will need to configure the client workstation to use the load balancing function. In this section, we show you the configuration of an SNA API client workstation to access two servers with load balancing for 5250 emulation sessions (APPC).

The first step starts with selecting **Start->Programs->IBM Communications Server SNA Client->Local INI Configuration.** This will give you the panel below.

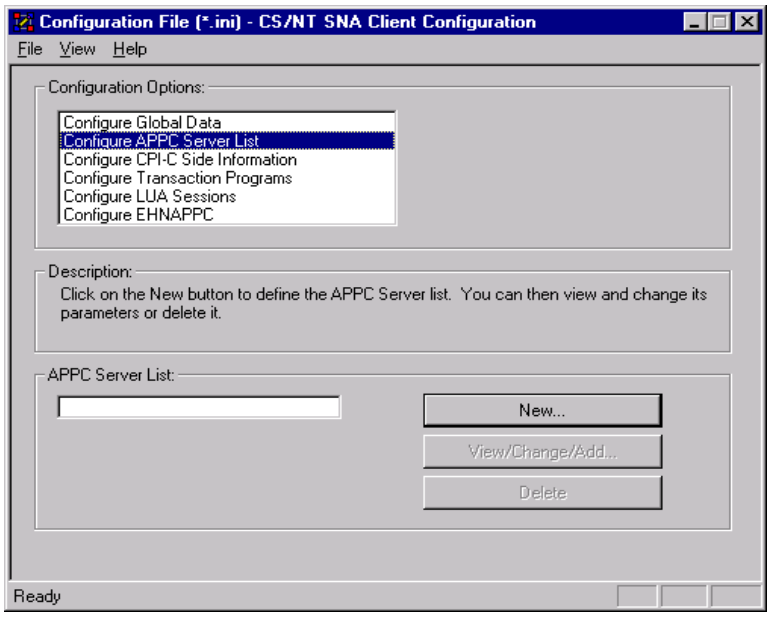


Figure 89. SNA API Client Configuration (APPC)

From here choose **Configure APPC Server List**. You will see the panel shown in the next figure.

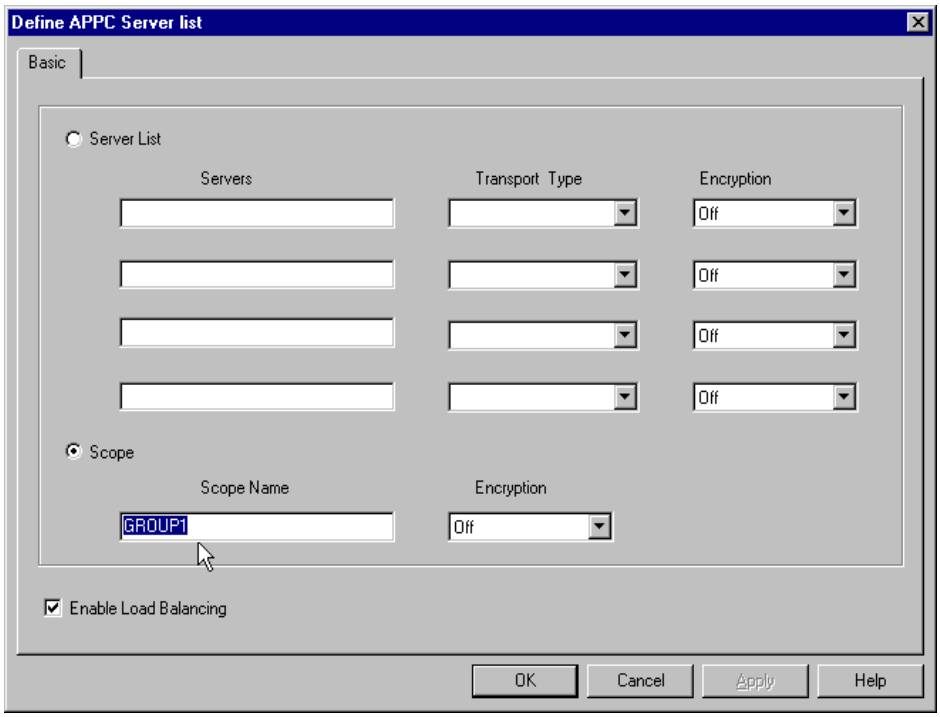


Figure 90. Load Balancing for APPC Sessions

The parameters in this panel allow you to configure the load balancing function for APPC sessions in the SNA API client workstation:

Enable Load Balancing: You need to select this check box to activate the load balancing option for APPC sessions.

Scope: You must select this radio button if you want this client to participate in the load balancing function.

Scope Name: In this field, you need to enter the scope name you configured in the servers you want to handle your APPC sessions. You can also enter an asterisk (*) to indicate that this client participates in load balancing but connects through unscoped servers.

Figure 91 shows the INI configuration file for APPC sessions in this client. It shows the configured scope name GROUP1 and that load balancing is enabled.

```
[GLOBAL]
USER_NAME=
PASSWORD=
TRANSLATE_TABLE=C:7CSNTAPI7COMTBLG.DAT
DEF_LU_ALIAS=
DEF_PLU_ALIAS=
LU62_LB=1        <==== load balancing is enabled
.*
[APPC]
SELECT=SCOPE
SERVER=
SERVER=
SERVER=
SERVER=
SCOPE=GROUP1,TCP,ENC_OFF
```

Figure 91. APPC Load Balancing - INI File

9.6.1 5250 Client Session Configuration

In this section we show you the configuration of a 5250 session in PCOMM. To start the PCOMM configuration select **Start->Programs->IBM Personal Communications->Start or Configure a Session.**

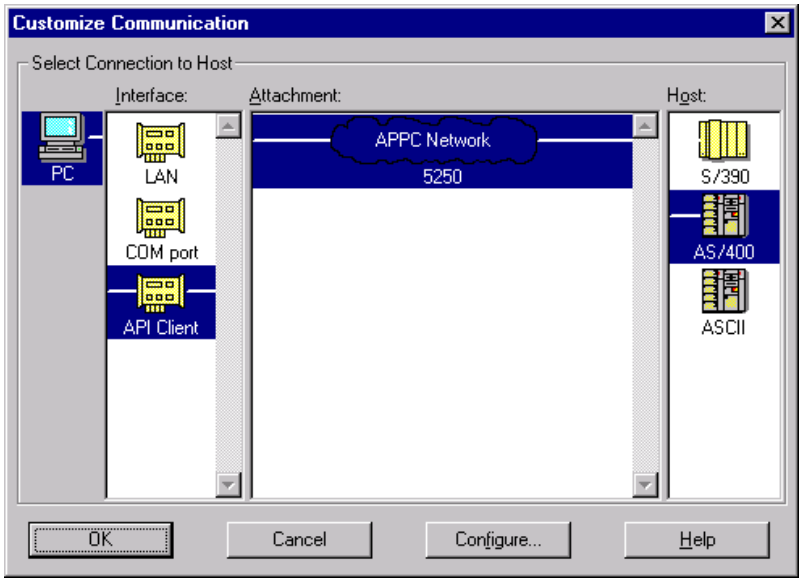


Figure 92. PCOMM Configuration for SNA API Client (TN5250)

Choose the API Client for the interface and the APPC network for the attachment. Click **Configure**.

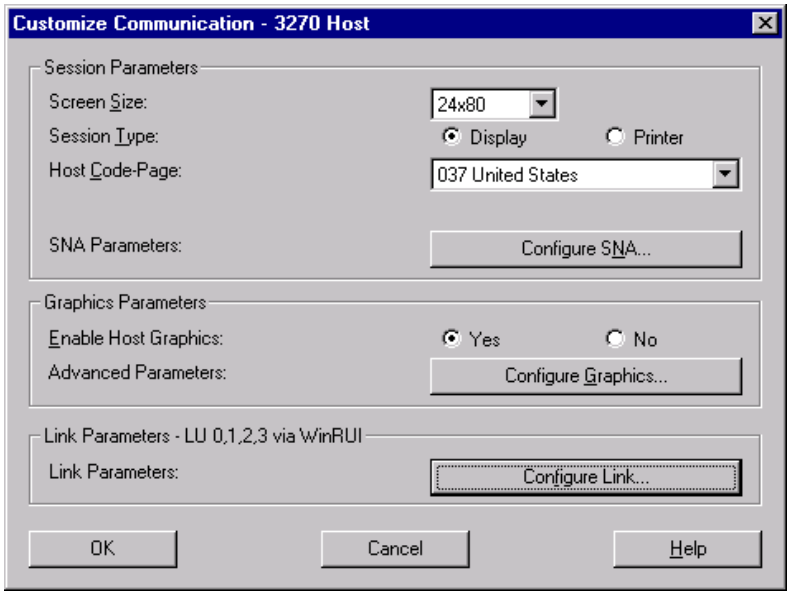


Figure 93. PCOMM Configuration for SNA API Client (TN5250)

Click **Configure Link** to see the panel in Figure 94. Since you do not know which server will handle your APPC sessions, you will leave the local LU name blank in order to use the default local LU in the server.

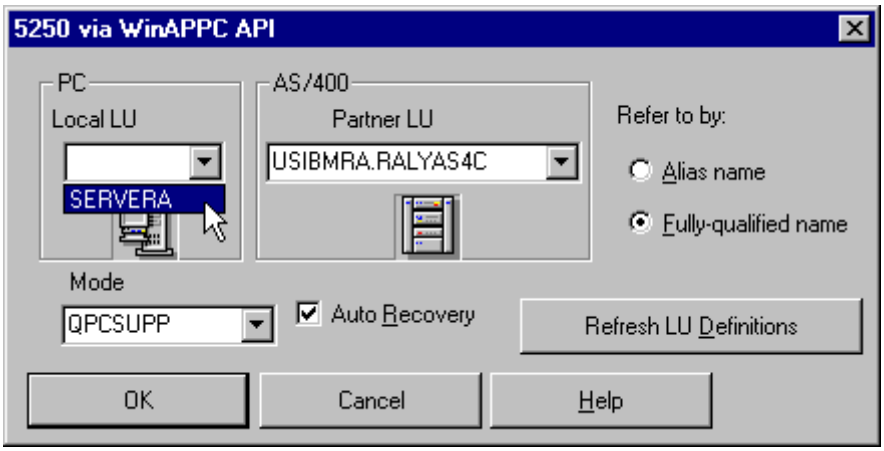


Figure 94. Configuring PCOMM

You will also be required to enter the name of your AS/400 system. In our scenario, we entered the fully qualified name USIBMRA.RALYAS4C.

Note

In SNA API clients, all APPC sessions use the same connection to the server. Therefore, all 5250 sessions and APPC sessions from the same client will go to the same server. However, the server is selected when the first APPC session connection is established with the server.

9.7 Using PCOMM 4.3

PCOMM 4.3 added SLP support. You can now configure TN3270 and TN5250 sessions using SLP without using the CS/NT SNA API client. In our network we defined a link to the host from each of the servers in scope GROUP1. We defined each CS/NT as a TN3270 Server with a default LU pool called TNSERVER. LUs from each host link are included in this pool.

The CS/NT server configuration for the load balancing function is the same as in “Server Configuration” on page 147.

The next step is for the clients to configure the PCOMM server. To start the PCOMM configuration select:

Start->Programs->IBM Personal Communications->Start or Configure a Session.

You will see the following configuration panel.

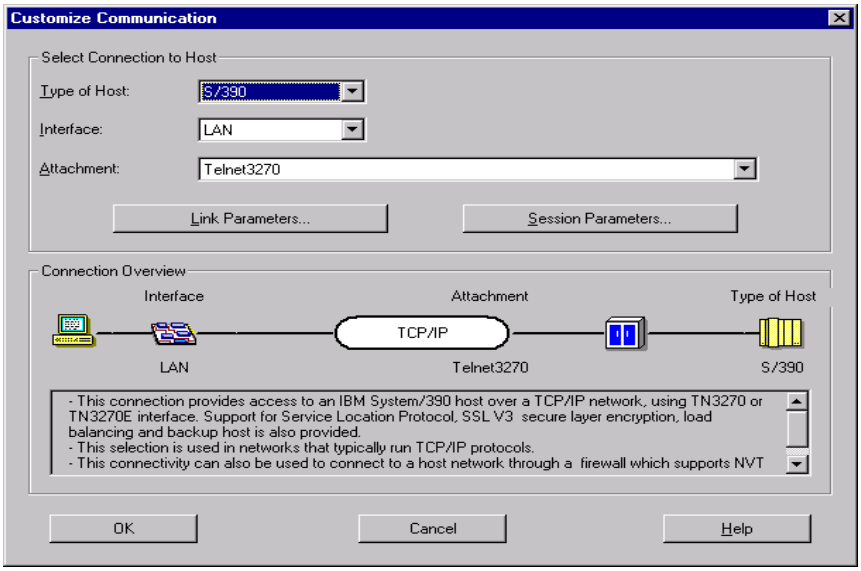


Figure 95. PCOMM 4.3 TN3270 Configuration

Choose **LAN** as the interface and **Telnet3270** as the attachment. Click **Link Parameters**. The next panel that comes up will have two tabs. Choose **Automatic Host Location**.

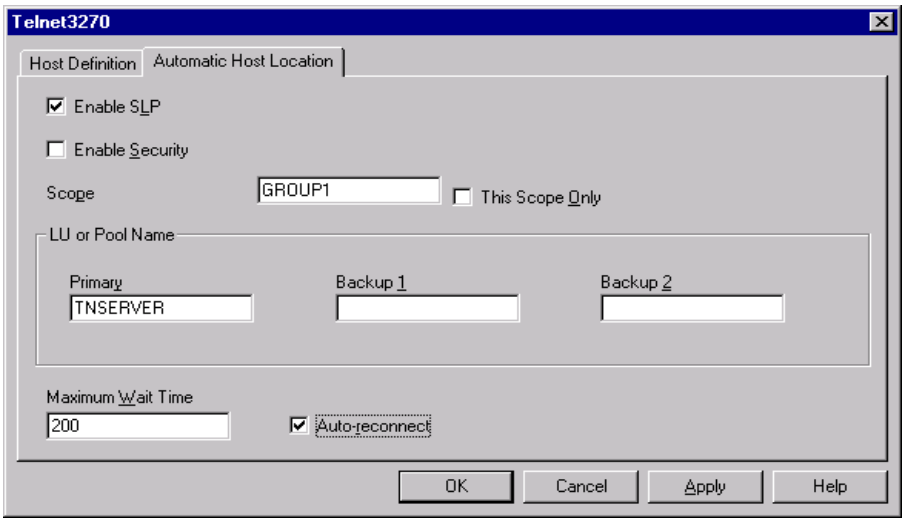


Figure 96. PCOMM 4.3 TN3270 Configuration

This is where you enable SLP. Enter the scope, GROUP1 in our case (see Figure 81 on page 149). Fill in the pool name and select **Auto-reconnect** for high availability. Click **OK**.

PCOMM will establish the host session using the least loaded server in the GROUP1 scope with an available LU in the TNSERVER LU pool.

9.8 Monitoring Load Balancing

The easiest way to monitor load balancing activity in your server is using the node operations facility. This node information has several entries that will help you monitor and understand the behavior of the server.

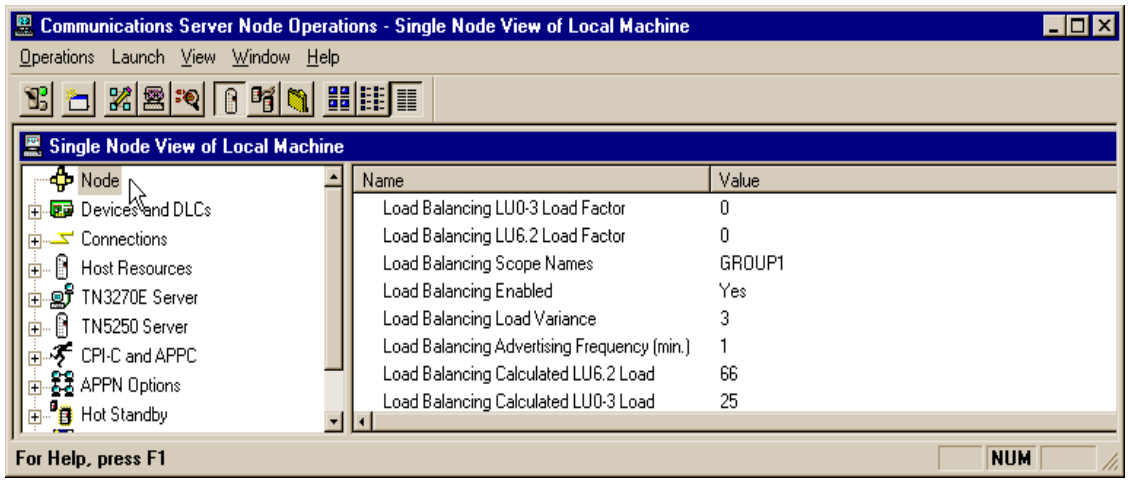


Figure 97. Monitoring Load Balancing

For example, in our scenario we display the load balancing information in our server and we can visually verify the configuration options, most importantly, the dynamically calculated load factors for the 3270 sessions and APPC sessions we are using as shown in Figure 97.

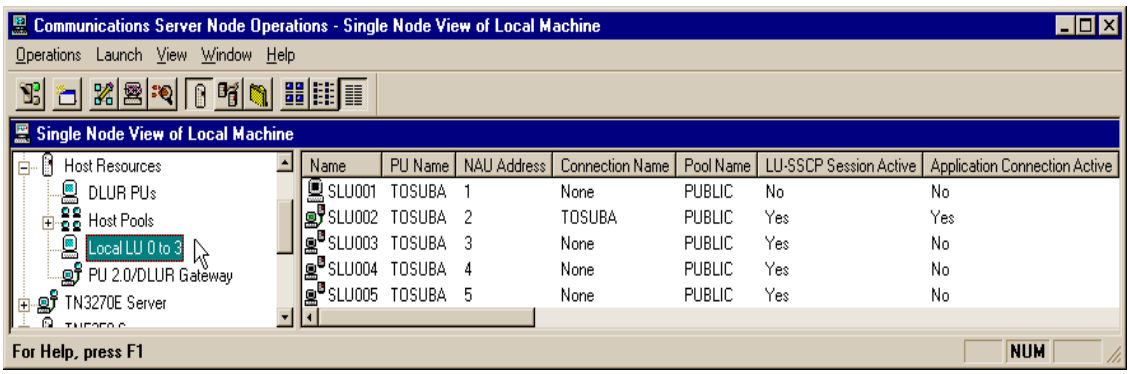


Figure 98. 3270 Emulation Sessions

The calculated load factor for the 3270 emulation sessions in Figure 97 is 25% since one session of the four possible sessions has been established. Figure 98 indicates that four LUs are communicating with VTAM and therefore have an SSCP-LU session but only one LU is in session (LU-LU session) as indicated in Application Connection Active.

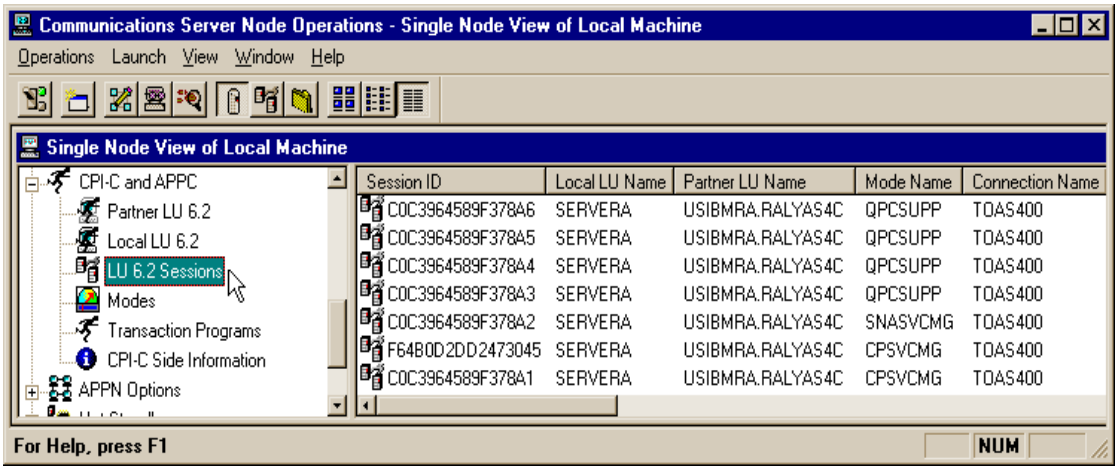


Figure 99. 5250 Emulation Sessions

The calculated load factor for the 5250 emulation sessions in Figure 98 on page 163 is 66% since four sessions of the six possible sessions have been established. Notice that in the server load balancing configuration, we set the default maximum number of APPC sessions to a value of six sessions. Figure 99 on page 164 indicates that there are four APPC sessions using mode QPCSUPP for the 5250 emulation sessions.

Chapter 10. Load Balancing for CS/UnixWare Win32 Clients

IBM eNetwork Communications Server for UnixWare (CS/UnixWare) servers can be configured to communicate using client/server protocols. CS/UnixWare performs load sharing for CS/UnixWare Win32 clients by allowing the definition of one or more common LU pools across servers.

Win32 clients use the server to establish a connection to the host using an LU pool name. By randomizing the Win32 client selection of servers, the incoming traffic can be spread across the CS/UnixWare domain to preserve availability and consistent performance.

CS/UnixWare client software includes API libraries that are fully compatible with Microsoft SNA Server and the Windows Open Services Architecture (WOSA). CS/UnixWare supports the following APIs:

- Windows APPC
- Windows CPI-C
- Windows LUA
- Windows CSV
- 3270 Emulator Interface Specification

For more information about CS/UnixWare client/server operation and SNA APIs, see the *IBM eNetwork Communications Server for UnixWare 7 Administration Guide*, SC31-8704.

10.1 CS/UnixWare Client/Server Concepts

All servers and clients communicating with each other using the client/server protocols are referred to as a domain. The domain name for servers is specified during CS/UnixWare installation, and for clients, during the installation of the CS/UnixWare Win32 code.

The servers provide CS/UnixWare SNA services. The clients have no SNA components, but use TCP/IP to communicate with the servers, which provide SNA functions to the client. The clients will receive information only about the servers in their own domain.

Servers must be CS/UnixWare 7 servers. Clients are Windows NT or Windows 95 computers with the CS/UnixWare Win32 Client installed. At the time of this writing, the Win32 client code on Windows 98 had not undergone thorough system test, but it seems to work fine.

10.1.1 SLIM Concepts

Communications between the clients and servers is accomplished by a LAN interface module referred to as SLIM. SLIM maintains a stream to the SNA API client/server router, providing the ability to connect paths between the SNA API client/server components on different machines. It extends paths across the LAN. The endpoint components don't know or care that the path goes across the LAN.

One or more domains may be defined on the same physical network. A single domain can correspond to a TCP/IP subnet, can be part of a TCP/IP subnet, or can span multiple subnets.

A single TCP/IP port is used for all domains. The traffic is filtered on domain name by SLIM. The default port number, which is 1553 is registered with IETF. This port can be changed. See the *IBM eNetwork Communications Server for UnixWare 7 Administration Guide,* SC31-8704 for details on how to change this.

10.1.2 The Server Role

Each server has its own node configuration stored locally.

The configuration for domain wide items is kept synchronized between the servers in one domain. Domain wide items are:

- 3270 users
- CPI-C side information
- Logging level
- Access to the remote command facility

The domain wide definitions are kept in /etc/opt/sna/sna_domn.cfg.

There are three different types of servers:

- Master server

  The master server holds a master copy of the domain configuration information and distributes it to all other servers.

- Backup server

  The backup server can take over as master server in the event the master server fails.

- Peer server

  The peer server provides SNA services to clients but does not provide any backup facilities.

The servers have a TCP/IP administration connection to the master that is used for:

- Client/server topology updates as machines enter and leave the network
- Modifications to the domain configuration file and the network data file
- Service information as services are enabled/disabled on machines
- Central logging, TP dynamic loading, etc.

10.1.3 Master/Backup Handover

When multiple servers exist in the domain, one server is the master server. Others may be defined as backup servers or peer servers. The master server contains the domain configuration file for the entire system. The peer servers get domain configuration information from the master as needed but cannot act as backup servers.

The backup servers contain copies of the configuration file that are updated by the master server whenever changes are made to the master configuration file. If the master server dies, backup server number one takes over as the master server. If the master server comes back up, it gets a copy of the configuration file from the current backup master server. It notifies the backup master server that it is taking over as a master server. The backup master server notifies all other servers that the master server has changed.

If a master server fails and the backup server takes over, the master server will regain control when it is restarted.

10.1.4 Network Data File

The network data file, /etc/opt/sna/sna.net, contains details of the SNA domain name and a list of master and backup servers:

- The first server listed is the master
- Other servers listed are backups
- Any server not listed is a regular server

The file is created by the installation process and is distributed from the master or backup master server when a machine starts so it is always up to date. When a new server comes in, it attempts to open an administration TCP/IP connection to each machine listed in the sna.net file. When a successful connection is made, the server downloads details of all servers in the domain. Any change of the status of the servers is sent to the other servers through the administration connection.

In order to add or remove backup servers, the NOF API and Motif Administration can be used. This will be covered later in “The Domain” on page 175.

10.1.5 Connecting Clients to the Network

The Win32 client has a TCP/IP administration connection to its sponsor server. It receives details of all servers through the administration connection, allowing it to use the resources of any server in the domain. The client then opens direct connections to other servers to use their resources.

If the server goes down, all clients using it are affected. The SNA sessions through that server are lost. If the administration connection is lost the clients will re-establish to another server automatically. The SNA sessions to other servers are unaffected.

To connect clients to the network, broadcast or non-broadcast mechanisms can be used to join the network. Win32 clients determine which mechanism to use during installation. This can be modified later.

10.1.5.1 Non-Broadcast

The client tries each server in turn until it establishes a TCP/IP connection. The server can be located on a remote subnet.

10.1.5.2 Broadcast

Using the broadcast mechanism means less configuration because there is only a check box to click. The client then broadcasts a message to its IP subnet. Only servers in the correct domain will respond. The broadcast is used to contact just one server. Once that contact is established, the server sends details about all other servers in the domain to the client. The client can then use resources on all its servers by setting up direct (non-broadcast) connections.

Design Tip

UDP broadcasts do not cross routers; therefore it follows that:

- If you are setting up a domain that spans subnets, the clients will have access to resources on all the subnets in the domain. But if they are using the UDP broadcast method, they can only learn about the resources from a server in their subnet.
- If the clients are using the UDP broadcast method (no servers specified) and all the servers in their subnet are unavailable, they will not have access to the domain resources.
- If the domain spans subnets, the clients should supplement the UDP broadcast with a list of servers in the other subnets in case the servers in the client's subnet are not available.

10.1.6 Load Balancing and Hot Backup

On the CS/UnixWare servers the 3270 LUs can be put into pools. A pool can span more than one server in the domain and can contain LUs that go to different hosts.

On an individual server with multiple links to the hosts and an LU pool that spans these links, connections will be made in a round-robin fashion, spreading the load among the host links. This accomplishes load balancing on the single server.

Load balancing among servers from an SNA API client depends on the clients having a random order of servers defined. There is no attempt to measure the load on each server and use the least loaded.

If the server fails, the client can retry and be assigned a new LU through a different server. Most emulators support automatic retry function. This is how the hot backup is achieved. But note that your LU-LU session will be lost. The new session you get will be an SSCP-LU session. A real hot backup with no loss of LU-LU sessions can only be achieved through High Performance Routing (HPR).

10.2 CS/UnixWare Win32 Client

The CS/UnixWare Win32 client uses a SLIM equivalent that runs as a service on Windows NT and as a background process on Windows 95. The application data uses a direct TCP/IP connection to the appropriate server. It does not go via the administration connection server.

There is a single TCP/IP socket used between the client and each server. Multiple paths are multiplexed on the TCP/IP socket. A path is an end-to-end connection between the API library code in the client and the API stub code in the server.

10.2.1 WIN32 Client Configuration

During installation, the client is configurable for the following values:

- Domain: The domain name must be the same as the domain name for the servers.
- Group: If you have defined an emulator group on the server and want to use this, specify the name here. You do not need this if you have defined individual emulator users or a default user.
- Servers: This is a list of servers in the domain to try. If you choose to use a UDP broadcast, you can still specify servers to try in turn if the servers cannot be reached by a UDP broadcast. An * as the first server (seen in the registry but not the configuration utility) indicates that a UDP broadcast should be tried first.
- Broadcasts: specify the maximum number of broadcasts to be made in one attempt to contact a server.
- Timers: specify the server lost timeout and LAN access timeout values.
- Tracing parameters

These parameters can be changed later by using the Windows registry or by using the configuration utility.

Start->Programs->CS-UnixWare Win32 Client-> Configuration Utility

These and other values, including trace values, can be changed in the Windows registry. The information is contained in values under subkeys of the following key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\SNAClient\SxClient\Parameters
```

You can alter these values using regedit. The registry entries are shown in the following figure.

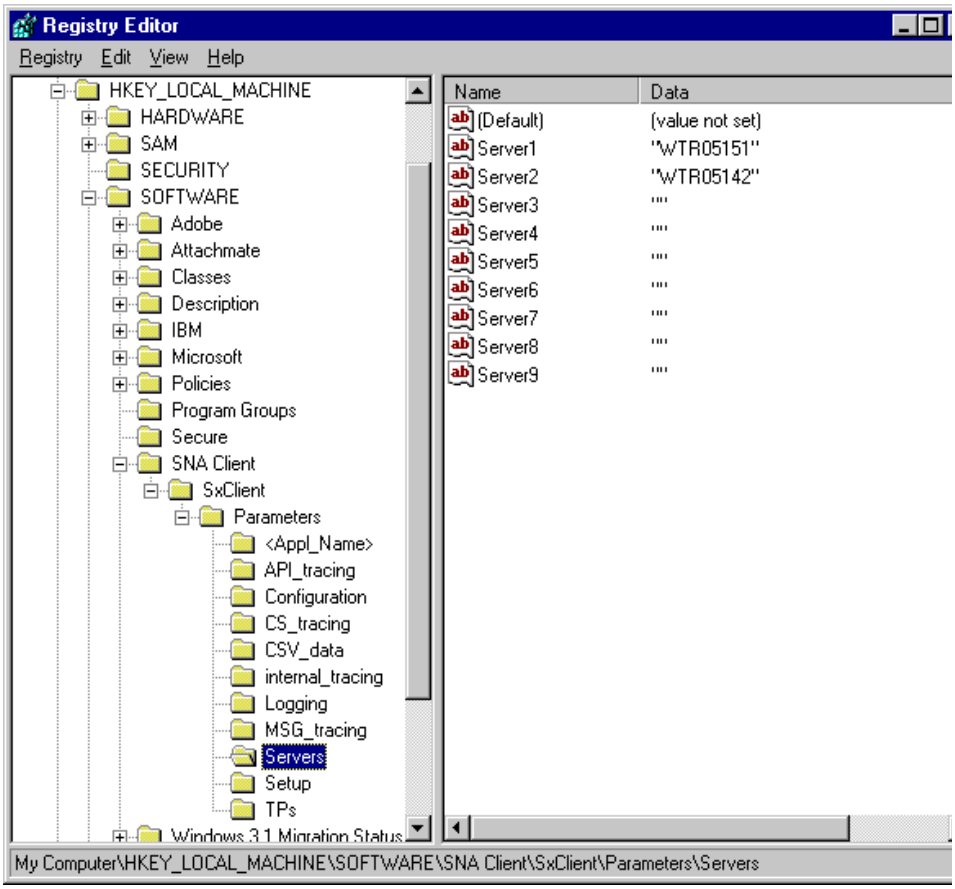


Figure 100. Registry Entries for Win32 Client

The *IBM eNetwork Communications Server for UnixWare 7 Administration Guide,* SC31-8704 has details of the parameters you can use and how to change them.

10.2.1.1 Win32 Client TP Configuration

Transaction program (TP) definitions for Win32 clients can be looked up or added by using tpinst32. On Windows NT, TPs can be services or run as executables in the security context of the sxclient service. See the *IBM eNetwork Communications Server for UnixWare 7 Administration Guide* for more details.

In general only information about servers is propagated across the network. If a client is configured to support invokable TPs, information about these TPs

is also propagated. This enables the servers to include the client when searching for the system for a target TP.

10.2.1.2 Win32 Client Security

Win32 client security applies to Windows 95 clients only. It provides a facility for validating the user name and password of any Win32 client running on Windows 95. The validation is not performed for clients running on Windows NT since the user had to enter a password and user name to open the Windows NT desktop.

The Windows 95 users must be defined on all the servers to UnixWare as a system user. The security feature is enabled on the server using the `snawinsec domain` command. The security setting is stored in the sna.net file. You can use the `snaadmin query_sna_net` command to check the setting. See Figure 102 on page 176 for an example of the output to this command.

When the client starts, a dialog asking for user details is displayed.

10.2.1.3 Win32 Client Diagnostics

Diagnostic tools are configured in the registry. Use regedit to change the registry. See the *IBM eNetwork Communications Server for UnixWare 7 Administration Guide*, SC31-8704 for details.

The following tools are available:

- API trace
- Client/Server trace
- Internal (debug) trace
- Logging

10.3 Load Balancing Configuration Overview

Load balancing can be accomplished for both LU 0 to 3 and for LU 6.2. An overview of the configuration steps are outlined below.

10.3.0.1 LU 0 to 3

To load share a 3270 display LU you first need to create an LU pool containing a number of LUs from different servers. You then need to create a 3270 user for each client and assign this LU pool to each user. The user name should match the user name used to log on at the Windows client. All this configuration is done at the server using xsnaadmin.

At the client you need to install the CS/UnixWare client software, then install a 3270 emulator. The emulator should be configured to use the SNA API and

not TN3270. Most emulators can return a list of LUs and LU pools configured for the user. In any case start a session using the LU pool name as the LU name.

The client will then go to each server it knows about in turn checking for availability of LUs in that pool. It will settle on the first available active LU it finds. If there are no available active LUs, it will choose the best LU, for example an LU on an inactive auto-startable Link Station.

Load sharing is achieved by each client randomizing its list of servers, so each client will try the servers in a different order. Thus with enough clients the load will be distributed evenly across the servers with LUs in the pool. There is no attempt to measure the load on each server and use the least loaded.

Load sharing for other LU 0 to 3 LU types is achieved similarly.

10.3.0.2 LU 6.2

For LU6.2 sessions, the choice of server occurs when the application issues TP_STARTED. To load share, the application should issue a TP_STARTED with the lu_alias field set to all zeros.

To load share dependent LU6.2 you need to define a number of dependent LU6.2 LUs and select the **LU is in pool of defaults** option. The TP_STARTED will then connect to the first unused LU with this option set. There is no attempt to pick an active LU or LU that can be activated. Load sharing is again achieved by each client randomizing its list of servers.

For independent LU6.2 the client will connect to the first active server tried. The local LU used will be the CP LU.

10.4 CS/UnixWare Load Balancing Scenario

Our scenario will show load balancing for Win32 clients using 3270 emulation.

For our example, we set up two SCO UnixWare servers running UnixWare 7.0.1 on IBM Netfinity 3500 Servers, and CS/UnixWare version 5.0.00.01 (the latest PTF level available at the time of writing). One of the servers is configured for token-ring and the other for Ethernet to illustrate the ability of the Communications Server domain to function across networks. We then set up an IBM desktop system running Microsoft Windows NT Workstation V4 on a token-ring interface with the CS/UnixWare Win32 client and IBM Personal Communications V4.3 for the 3270 emulation and SNA API client portion.

In this scenario, we will set up load balancing between two CS/UnixWare servers. The servers will be in a domain called uwcluster. One LU pool will be defined that will contain LUs from both servers.

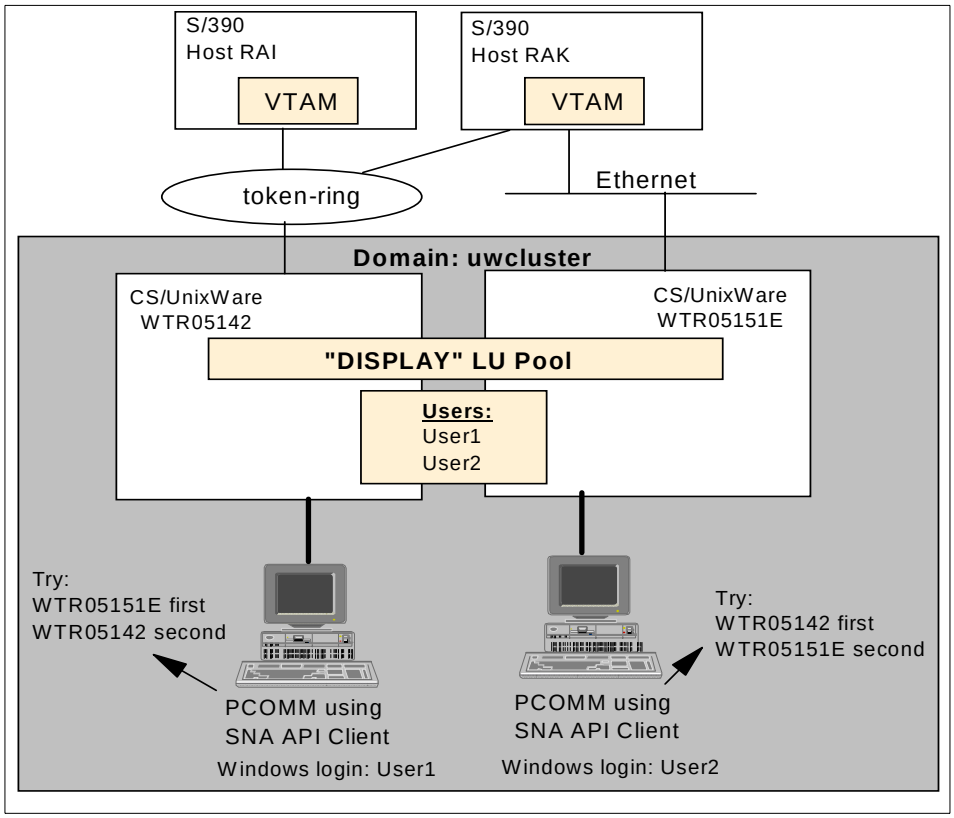


Figure 101. CS/UnixWare SNA-API Load Balancing Scenario

10.4.0.1 Setting Up the Servers

To set up the servers you perform the following steps:

1. Install CS/UnixWare on all servers, giving them the same domain name, in this case uwcluster.

2. Choose which servers will be the master and which will be the backups. The order of the servers added determines which server is the master and the order of the servers to be chosen as backups. You cannot move the servers around in the list. To change the order, you would delete all the servers from the list except the server you want to be the master and add

them again, in the preferred order. The first server added is the master server.

3. Create the host connections and LUs. Put the LUs on all the servers in one pool. In our case the LU pool is called DISPLAY.

4. Configure the CS/UnixWare servers for TN3270 using the common pool.

5. Define users for the domain. This step may or may not be necessary. It depends on the requirements of the emulators. The server does not check for the user record and PCOMM does not require it, so in our example we did not have to define the users. We included the step in case you are using something other than PCOMM.

However, some Win32 emulators only allow the user to choose an LU that is configured at the server for the user. There is an API that allows the emulator to retrieve the list of LUs configured for its user. In this case it would be necessary to define the users at the server.

10.4.0.2 Setting Up the Clients

To set up the clients you would do the following:

1. Install the CS/UnixWare Win32 client.

2. Specify the domain name of the servers during the install.

3. Configure PCOMM to use the API interface and the common LU pool name.

10.5 Setting Up the CS/UnixWare Servers

During the install of CS/UnixWare you will be asked the domain name of the server. In this scenario we called the domain uwcluster.

10.5.1 The Domain

The domain information is stored in a binary file called /etc/opt/sna/sna.net. Backup servers can be added or deleted to this file through the xsnaadmin domain window (see Figure 103 on page 176). You can also do this with snaadmin commands.

10.5.1.1 Commands to Alter the sna.net File:

- To add a backup master server:

  `snaadmin add_backup` *`backupname`*

- To delete a backup master server:

  `snaadmin delete_backup` *`backupname`*

- To query which servers can act as a backup to the master:

  `snaadmin query_sna_net,num_entries=0,list_options=FIRST_IN_LIST`

This last command is also convenient for displaying the domain name.

```
# snaadmin query_sna_net,num_entries=0,list_options=FIRST_IN_LIST
------------------------------------------------------------------------
list_options = FIRST_IN_LIST
domain_name = uwcluster
security = OFF

server_name = wtr05151e

server_name = wtr05142
```

Figure 102. query_sna_net Command

Once the CS/UnixWare servers are installed and communicating with each other, you can open the xsnaadmin window to make sure that the servers are up and communicating with each other. When starting xsnaadmin, the first dialog box is the domain view.

You should see all active servers in the domain and their status.

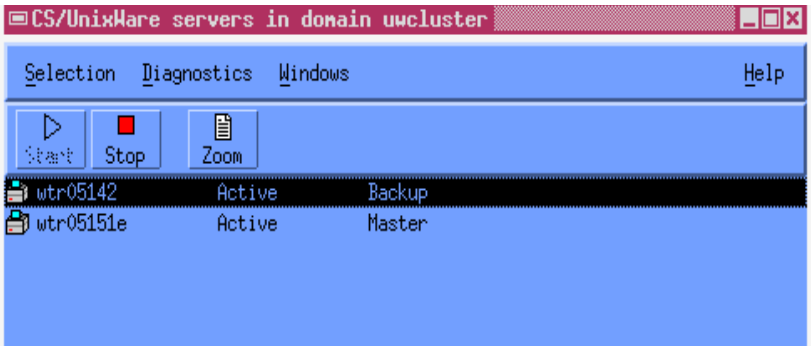


Figure 103. Domain Window

You will notice that both the master and the backup servers in our domain are active. If SNA were stopped on either system, the GUI would not be able to

run on that system and it would not show up in this domain view. From this window you can access the GUI interface for any of the active servers.

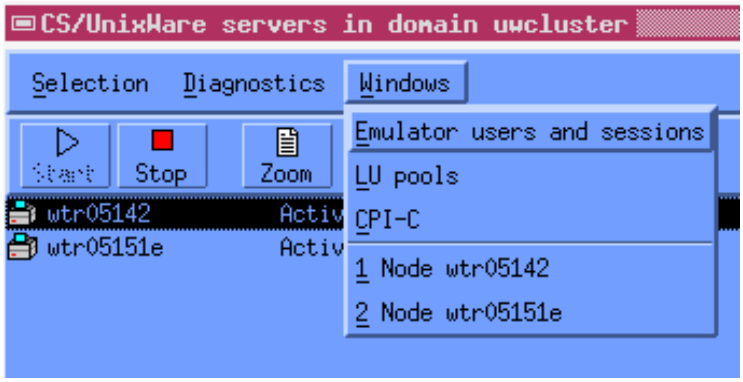


Figure 104. Accessing Servers from xsnaadmin

10.5.2 LU Pools

The next step is to set up the SNA configuration on each of the servers. To do this you need to define:

- One or more ports
- One or more link stations to the host
- Add LUs to the link stations
- Define an LU pool on each server with the same pool name and add the LUs to it

Figure 105 on page 178 shows the configuration for one of the servers in our scenario, WTR05142. There are two link stations defined to the host over the token-ring port. LUs have been added to each link station.

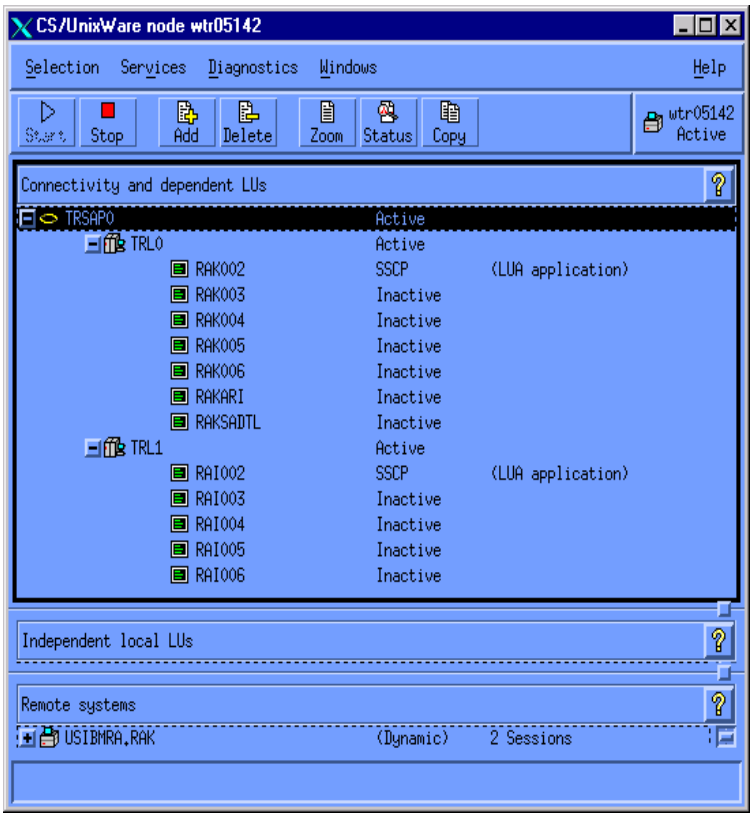


Figure 105. SNA Configuration for WTR05142

The LU pools can be modified by selecting **Services->LUA->LU Pools**.

The LUs have been put in a pool called DISPLAY.

Note: PCOMM will translate pool names to uppercase. Your pool name in CS/UnixWare must be uppercase as well.

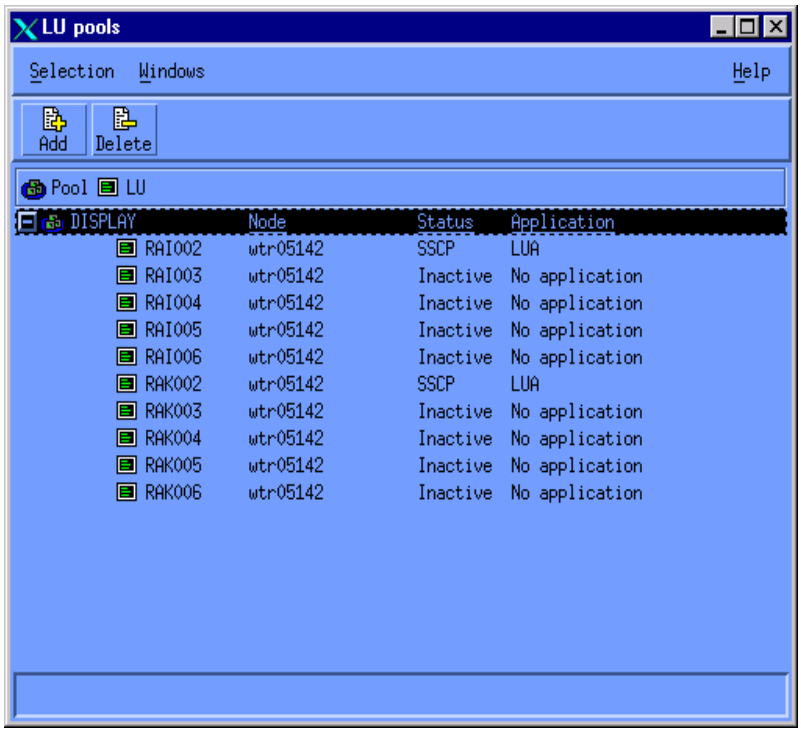


Figure 106. DISPLAY LU Pool

Set up the other servers in the same way. It is important that the LU pool be called DISPLAY (or whatever name you chose) on each server. The clients will specify this name in their configuration.

10.5.3 Defining Users

In CS/UnixWare, you can define emulator users or groups of users to enable 3270 communications. For the Win32 client, the user names are the same as the Windows NT or Windows 95 login name.

Choosing the Emulator users and sessions button in Figure 104 on page 177 will bring you to the next panel. You will see the groups and users listed (groups have two heads, users have one).

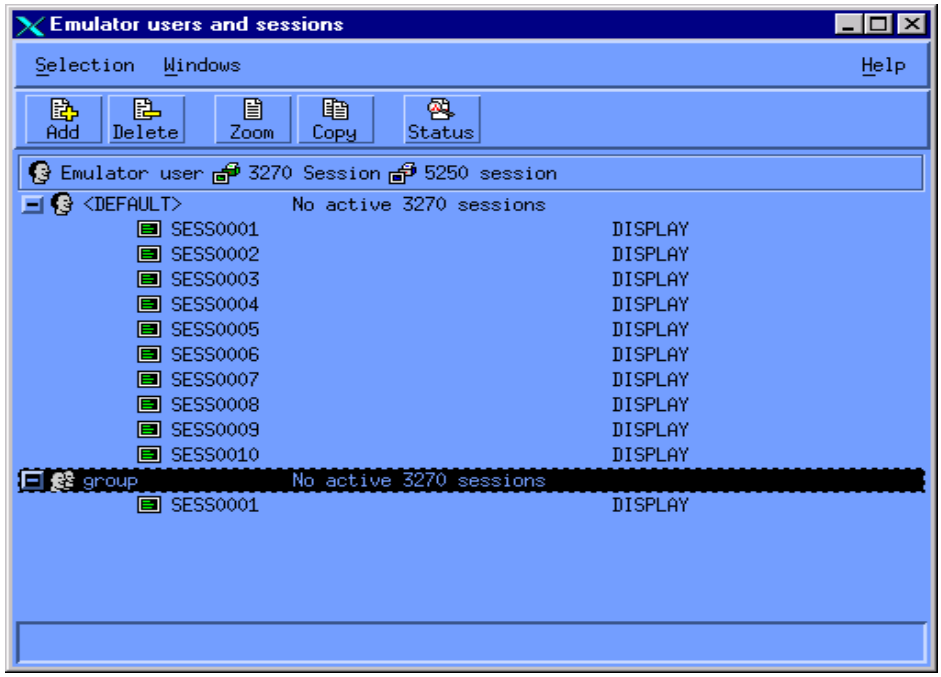


Figure 107. CS/UnixWare Emulator Users

10.5.3.1 Defining Users and Groups

You can set up a default user definition that is used by all users who do not have their own individual or group definitions. To do this, specify a user or group name of DEFAULT.

For more information on defining emulator users and groups, see the *IBM eNetwork Communications Server for UnixWare 7 Administration Guide,* SC31-8704.

For our example, we have set up a default user:

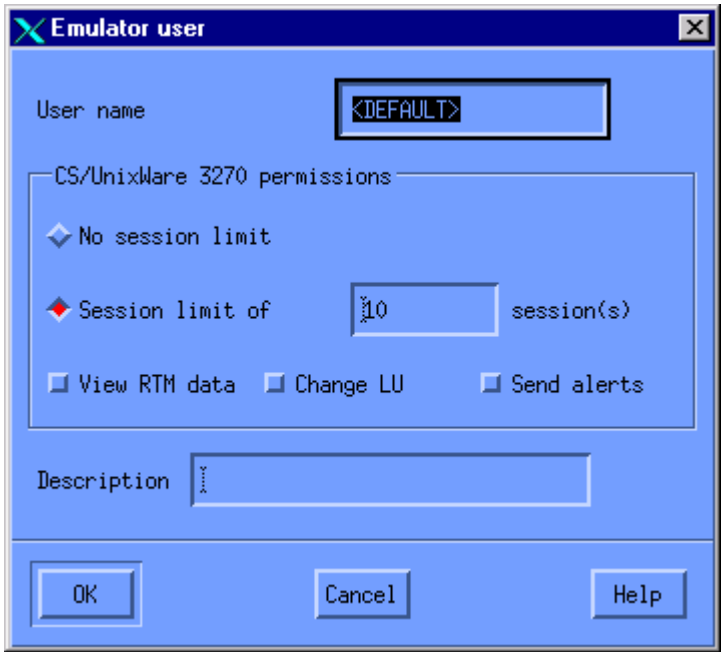


Figure 108. Adding a Default Emulator User

The next step was to add sessions for the default users. We chose to assign multiple sessions (10) using the DISPLAY pool of LUs.

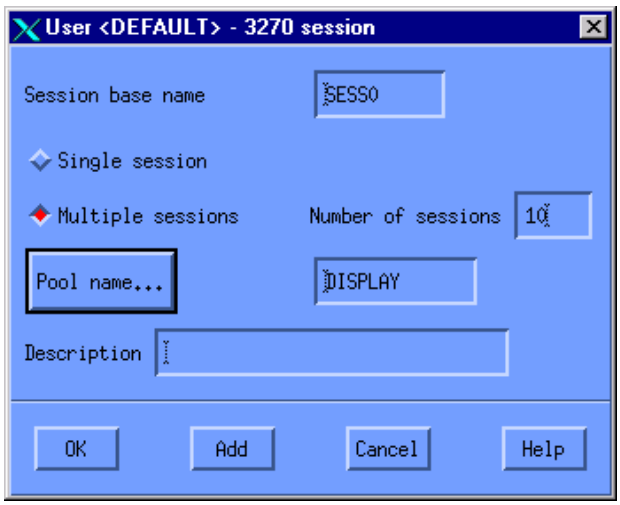


Figure 109. Adding Sessions to the User

Emulator definitions are considered global to the domain. They are stored in /opt/sna/sna_domn.cfg. This file is copied across to the servers in the domain so the definitions are used for all servers.

10.5.4 TN3270 Setup

Next you must set up the TN Server configuration on each server to allow the clients to access the host. The TN Server definitions can be modified by selecting **Services->TN server->TN server**.

The TN3270 definitions should point to the DISPLAY for the LU pool.

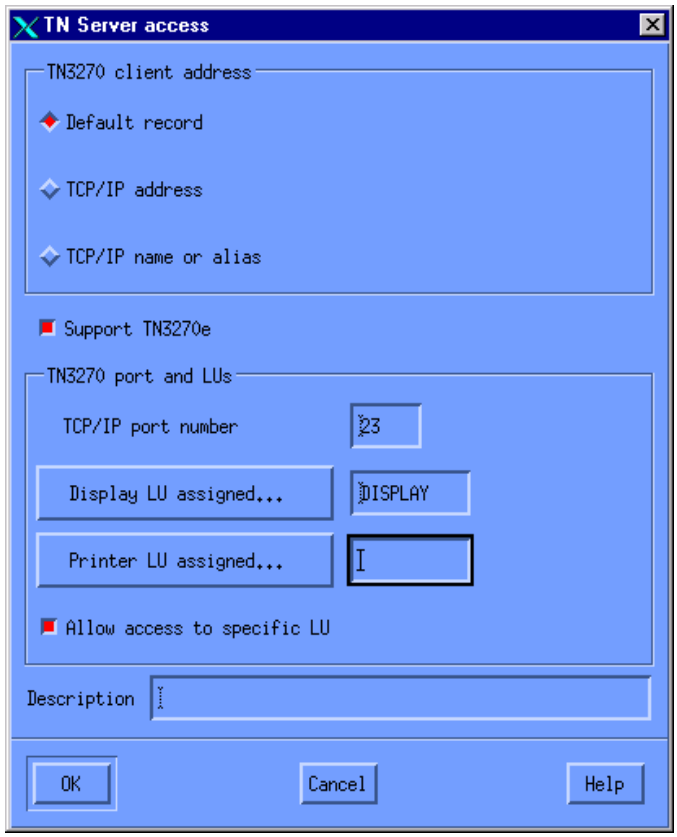


Figure 110. TN3270 Server Definitions

Now the servers are ready for the clients to access. Next, you will need to set up the clients.

10.6 Installing the CS/UnixWare Client

There are a variety of ways in which the Win32 CS/UnixWare client can be installed. We installed directly from CD ROM, but it could be installed from a shared file on the CS/UnixWare server. This might be preferred in an environment where mass installs would be taking place or when tighter restraints are in place for the distribution of media. After installing CS/UnixWare on the server, the client code itself is available in the directory /opt/sna/w32cli as filename i_w32cli.exe, a self-expanding executable. Copy this file to the client. Run it to expand the files into a temporary directory. From there you can run setup.exe.

The installation CD from IBM for CS/UnixWare is formatted using the standard CDFS, and can be read from both UnixWare and Microsoft Windows platforms with ease. The CD comes with an autorun.inf file in the root so that a Windows 95 or NT system will automatically begin the installation process of the client upon insertion of the CD if the system is enabled for such actions. If not, then simply running the setup.exe command from the <*CD-ROM-drive*>:\w32cli directory will start the process. The process is a typical *InstallShield* enabled application setup:

1. Insert the CS/UnixWare installation CD into the Microsoft Windows 95 or NT V4.0 system and run \w32cli\setup.exe.
2. When setup begins, first choose the language you wish to install (U.S. English, German or French)
3. The next screen will begin a series of dialog boxes to take you through the configuration installation.

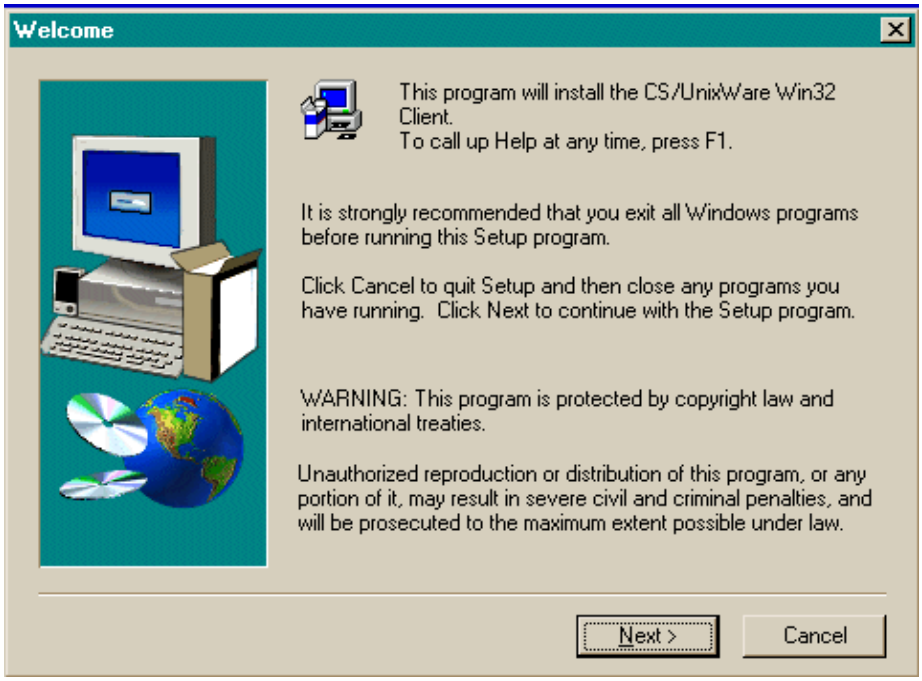
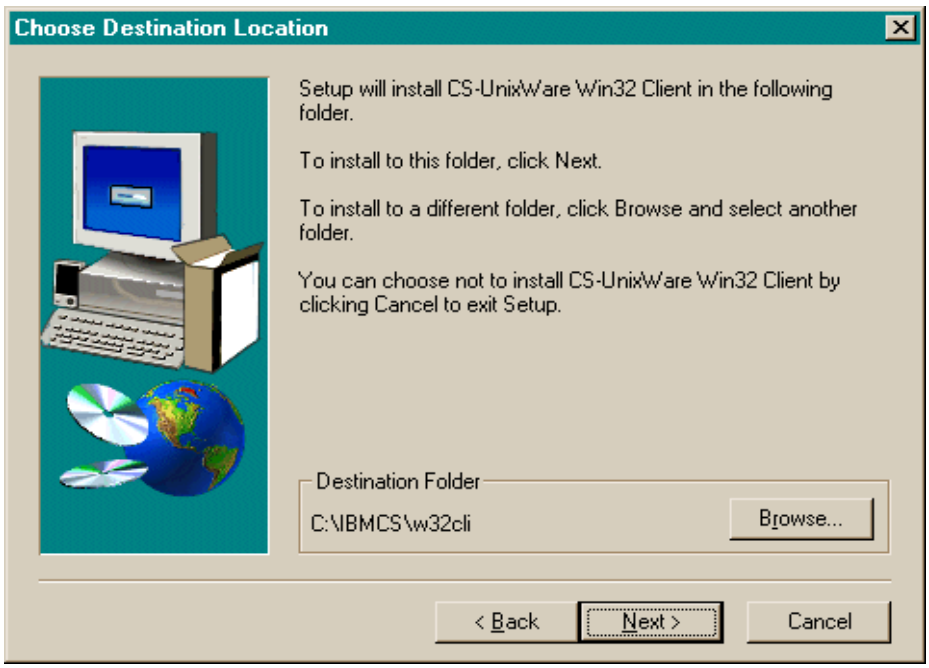
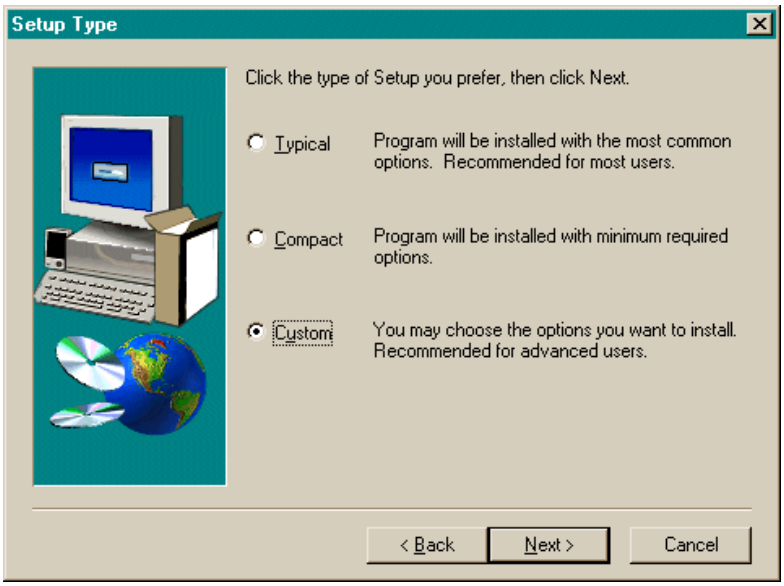


Figure 111. CS/UnixWare Win32 Client Installation

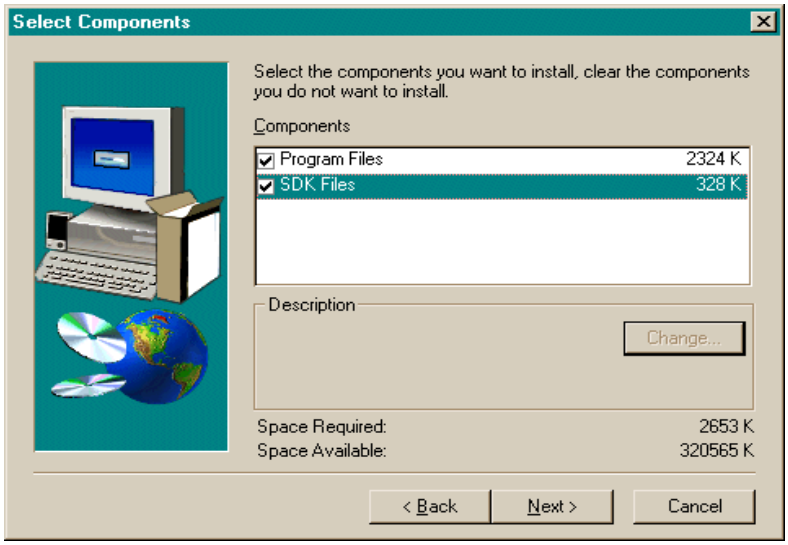
4. Continuing the installation brings us to the Software License Agreement. Read the agreement and click **Yes** to continue.
5. Choose the directory into which you will install the client. For those familiar with the Communications Server for Windows NT client installation, this is the same default directory name.



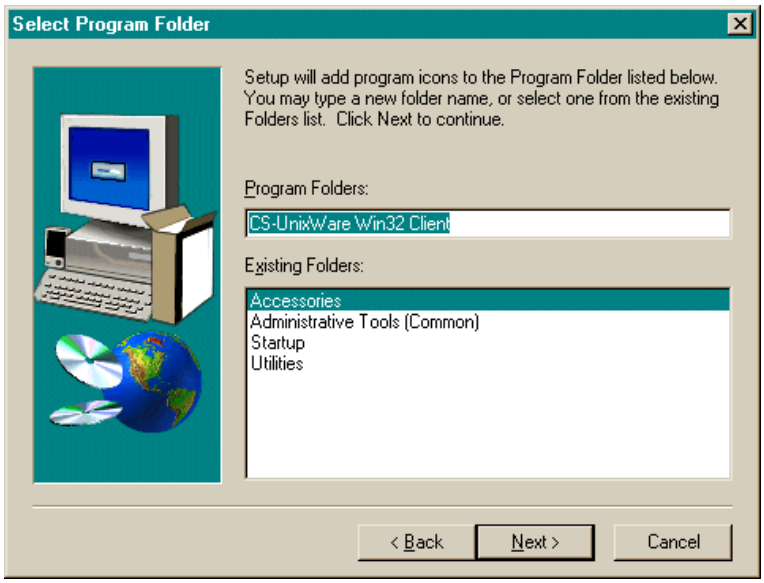
6. Note that we chose the **Custom** installation option from the choices of Typical, Compact or Custom.



7. The Custom option shows two components, both of which we installed:



8. In this example, we will stick with the default folder, so from the desktop we can choose **Start->Programs->CS-UnixWare Win32 Client** to start the monitor or configuration utility.

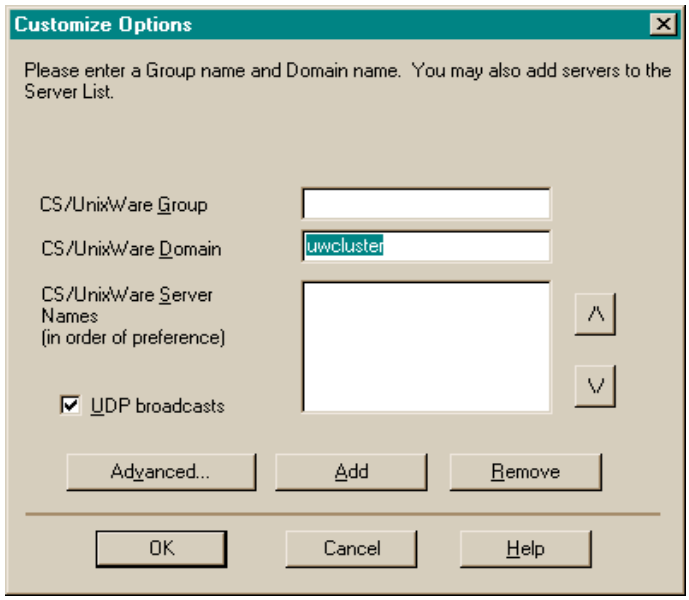


9. After the files have been installed, the initial configuration dialog box comes up for you to configure your environment. Be sure to enter the

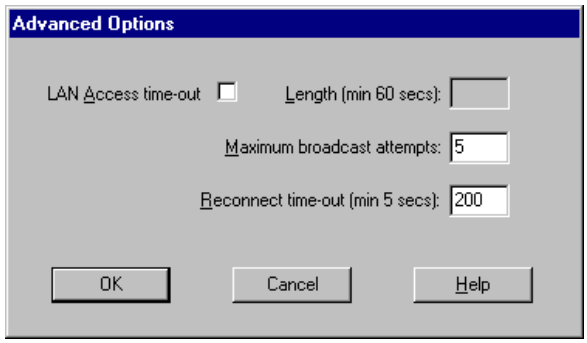
correct domain name. This would be the same name entered during the CS/UnixWare server installation. This can be seen with an snaadmin command (see Figure 102 on page 176).

The user can choose to use UDP broadcasts to select a server or enter a list of servers from which to choose. To set up a load balancing scheme, the users should put the servers in random order. The first server entered is the server the user will be connected to every time if that server is available.

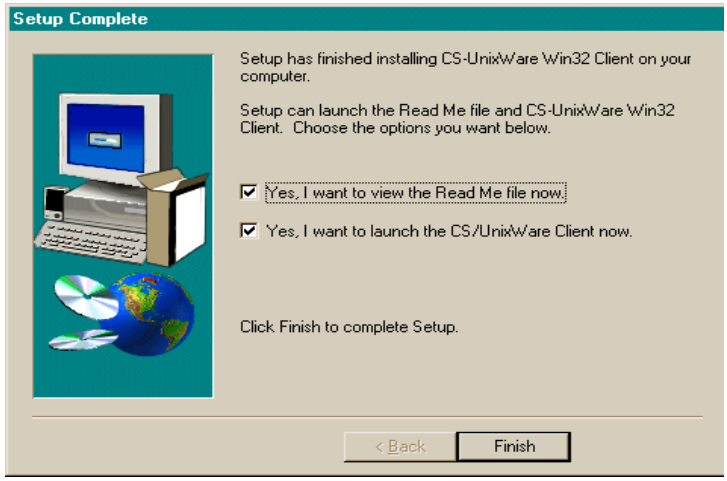
The load balancing technique relies on the randomness of this selection by each client.



10. The advanced dialog box is shown below. We did not change the defaults.



11. After clicking **OK** to get out, we see the final dialog box:



12. And the installation is complete! The next step would be to install the SNA API-aware emulator software (PCOMM, for instance), and configure it to use SNA API as its transport method (as opposed to LAN, SDLC, etc.).

Under Microsoft Windows NT the CS/UnixWare Client uses a service to operate and can be viewed by selecting **Control Panel-> Services** and finding the CS/UnixWare Client as shown in the following figure.

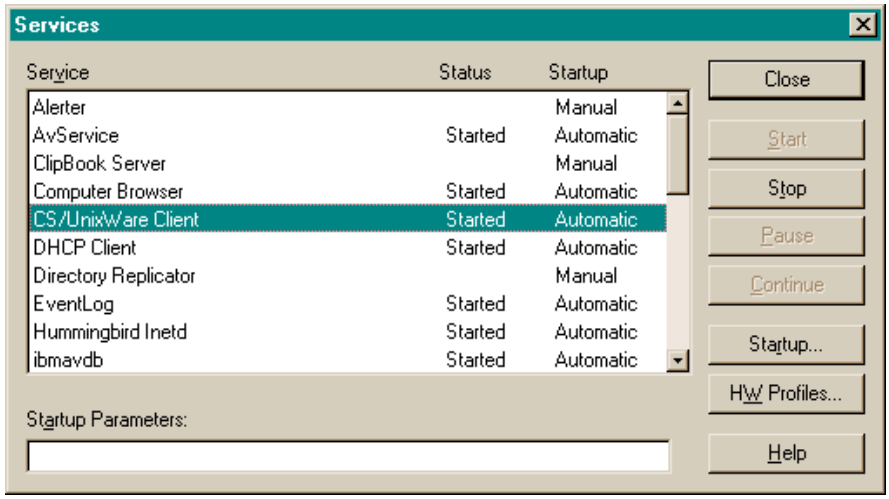


Figure 112. CS/UnixWare Client Service

Note that it automatically configures to start on bootup.

10.7 Configuring and Using the CS/UnixWare Client

Our test configuration used IBM eNetwork Personal Communications (PCOMM) clients. To begin the PCOMM configuration, we chose **Start->Programs->IBM Personal Communications->Start or Configure a Session.**

The first configuration screen is shown next. Choose the API interface as the client and LU 0,1,2,3 via WinRUI as the attachment.

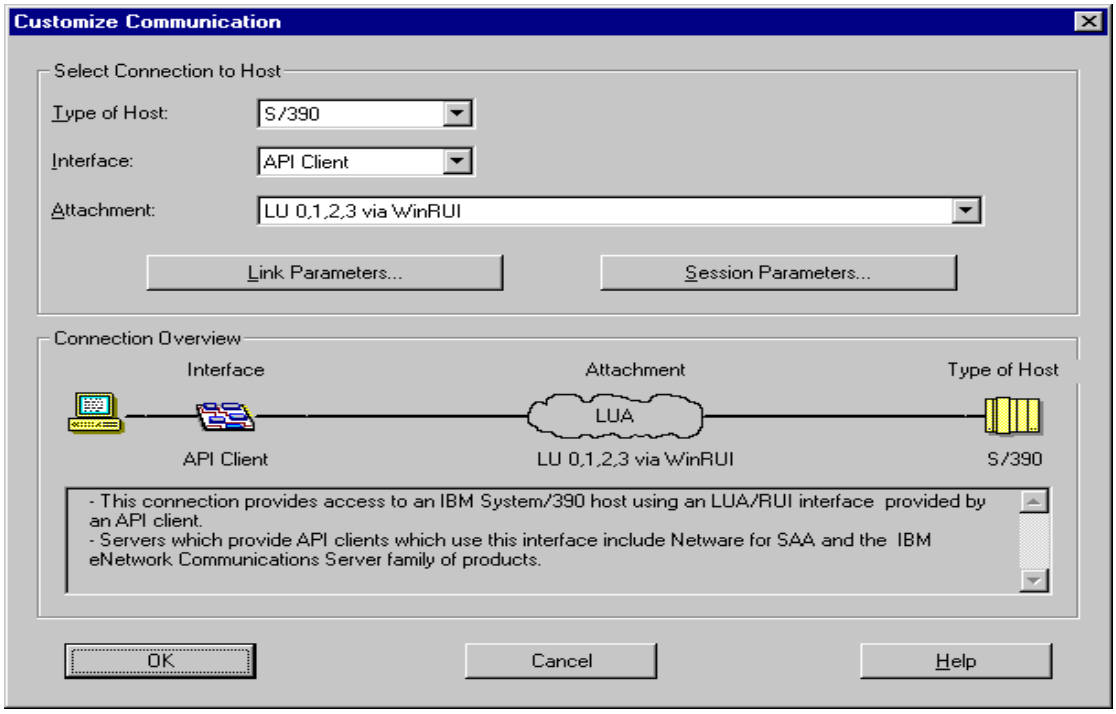


Figure 113. PCOMM Configuration

Click on **Link Parameters** to get the next screen. The clients must specify the pool name defined in the servers earlier, in this case DISPLAY.

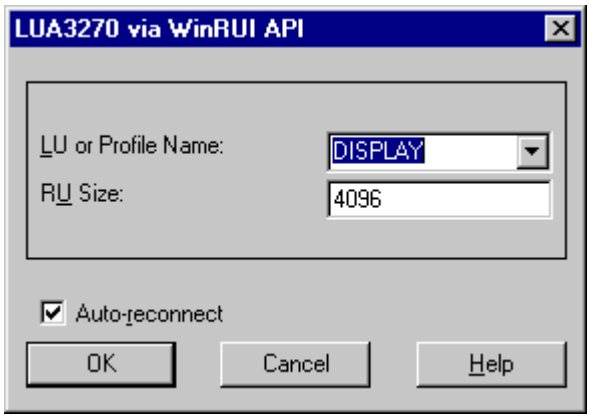


Figure 114. PCOMM Configuration

10.7.1 Testing the Configuration

The real key to the load balancing scheme is that the clients should be set up so that the order of servers chosen is random. Once the client has established a session through a server, it will continue to use that same server. Multiple sessions for that user will be spread over any links on that server that have LUs in the DISPLAY pool, but unless no more LUs are available or the server is no longer available, the user will stay on the original server chosen.

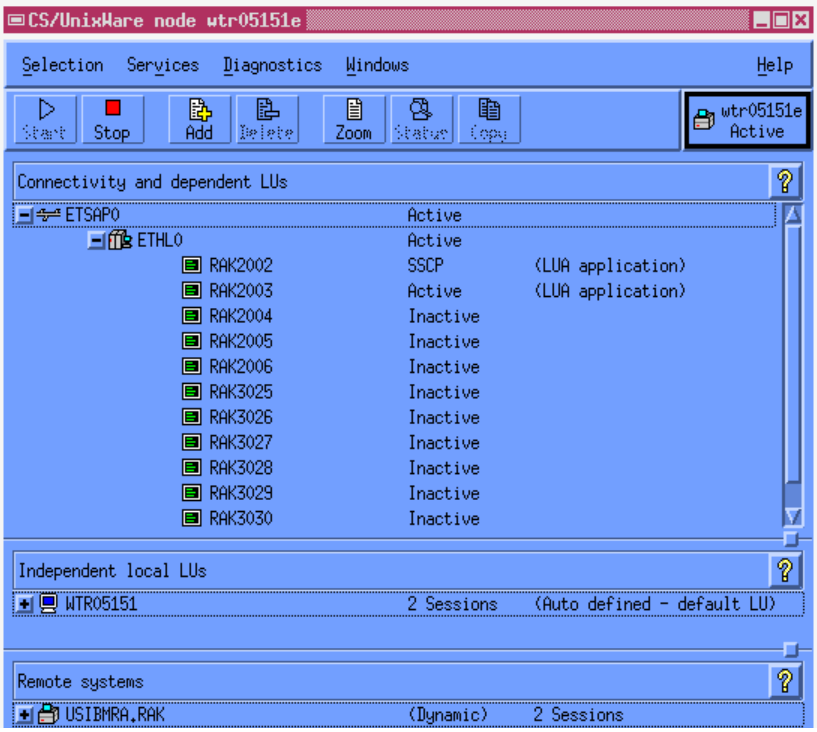


Figure 115. Server Status

Figure 115 shows the server's perception of what is going on. Note the status of LU RAK2002 shows as SSCP. This shows our client has established a session to the host using PCOMM and the SNA API client but is still showing the VTAM MSG10 screen (shown in the next figure). The session has been established to VTAM but no LU-LU session has been established yet.

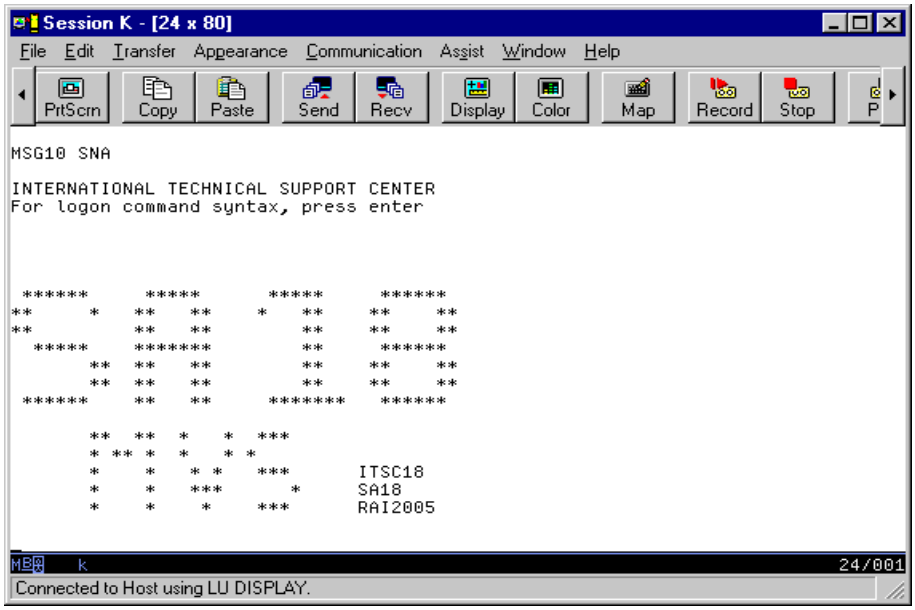


Figure 116. Host Connection

In Figure 115 on page 192, the session showing up on RAK2003 as active is where a session has been established with an application (TSO).

In the event that the host link we are using or the UnixWare server goes down and PCOMM is configured to restart, the domain model we have employed will allow us to re-establish a connection across the other server. The session is disrupted and the user will need to log back on.

If a session cannot be re-established, for example, if all of the sessions in the LU pool were in use or all servers were down, then we might see a failure message in the PCOMM screen as shown in the following figure.

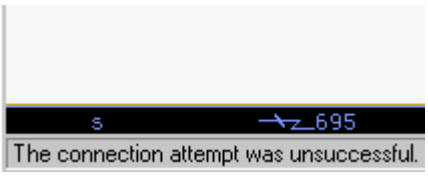


Figure 117. Connection Attempt Failure

Then we would have to wait for resources to free up or for more resources to be made available in the pools.

In this environment, both CS/UnixWare and PCOMM have vigorous and extensive tracing tools allowing for in-depth troubleshooting if needed. We only needed to troubleshoot one feature in this scenario. We realized that the reason our PCOMM client was not attaching was that even though we had configured it for the LU pool named display, PCOMM was looking for an LU pool named DISPLAY. Regardless of our efforts, PCOMM insisted that the name be in all capital letters, so once we obliged and renamed the pool on the server to DISPLAY it worked as designed.

Chapter 11. Single Server Load Balancing across Links

IBM eNetwork Communications Servers for AIX, UnixWare, and Windows NT all provide a load balancing mechanism to balance traffic destined for a specific LU pool over multiple links. For a single server serving SNA gateway and TN3270 clients, you can take advantage of this LU pool load balancing by defining one LU pool that spans multiple host links. This does not require SLP support.

By defining a single LU pool and putting LUs from two different host connections in it, you provide a built-in load balancing function to the clients. Requests for LUs in the LU pool will be spread across host connections. For example, the first request for LU pool A will have the host connection established over host link 1, the second over host link 2, the third over host link 1, and so forth.

This also provides a built-in backup scheme in case one host link goes down. Future requests will be routed over the other link.

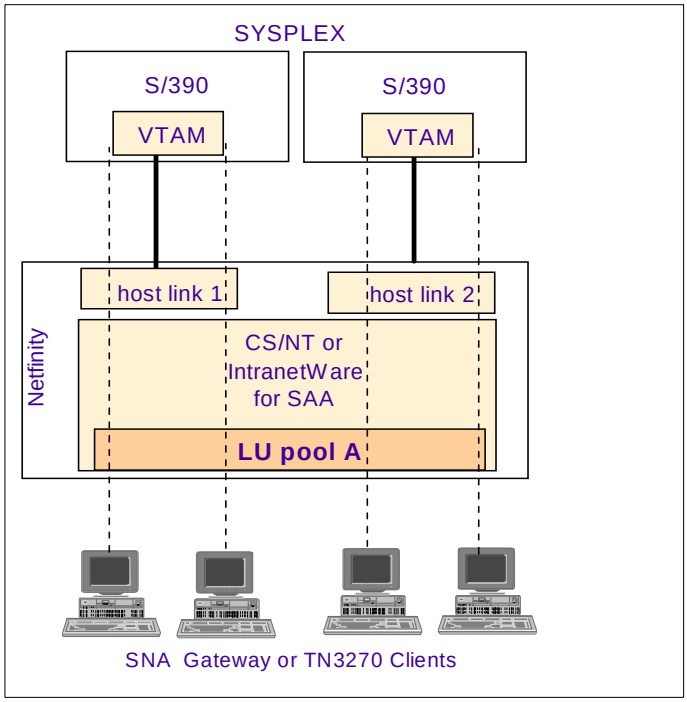


Figure 118. TN3270 and SNA Gateway LU Pool Load Balancing

Appendix A. Determining System Load with sys_load

The software shipped with CS/AIX used to determine load on each of the servers is called sys_load. It is the program that is defined in the Custom Metric parameter in the LoadLeveler ISS configuration file and the Metric External parameter in the WebSphere ISS.

The ISS agent runs sys_load on each server. The function of sys_load is to take into account factors which could impact system load, with the primary goal of providing the client with the best response time. The sys_load shell script returns this to ISS in the form of a single integer between 1 and 99999.

The program considers all of the following factors which could impact system load:

- Speed of the link
- Availability of host LUs
- Number of client-to-host connections
- Overall system load calculated after considering:
  - Paging space utilization
  - CPU utilization
  - MBUF utilization

  Note: The administrator can customize the relative importance of each.
- (Optionally) processor type and speed

When determining the load value to return to the ISS, the primary objective is to provide the client with the best response time. Since link speed is the primary factor in response time, the administrator defines what is considered to be a fast, medium or slow link. Figure 119 shows the sys_load logic.

Note

There are two known problems with sys_load. The first affects servers running AIX 4.3.2. The `netstat -m` command returns less output causing line 191 of sys_load to fail. The second affects all users. Both will be fixed in a future PTF (defect 1809). The second error has been corrected (item 1) in our modified script shown in “sys_load Modified” on page 210. You will also need to modify the script if you are using SDDLU at the host to define LUs.

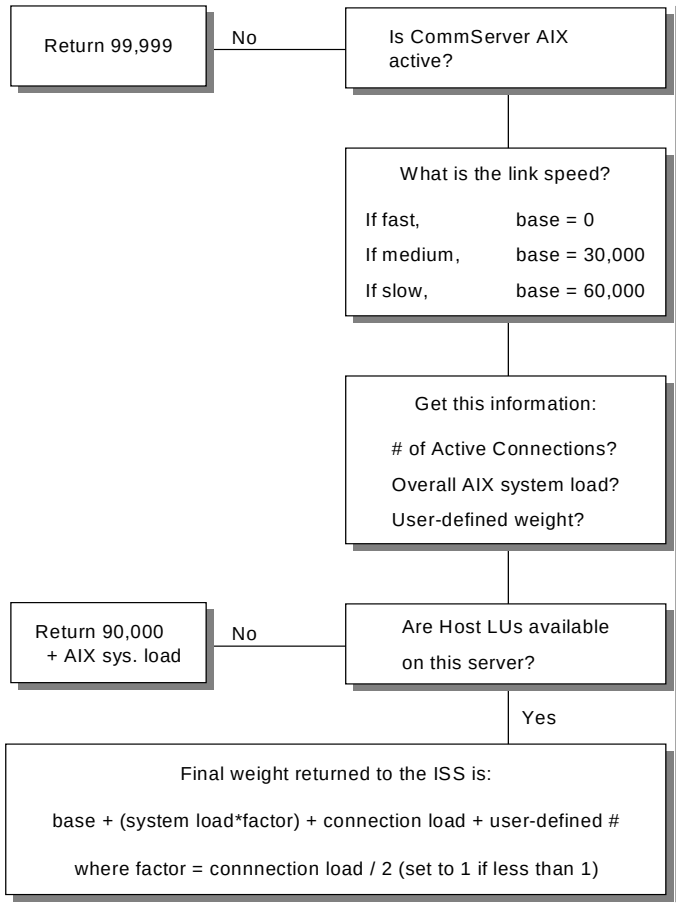


Figure 119. sys_load Processing

- Is CS/AIX active?

  A simple query is issued to see if the TN3270E Server software is active. If not, then connecting a client to this server is the least desirable option. As a result, the highest weight (99999) is returned to the ISS.

- What is the link speed?

  Link speed is measured in units of bits per second and is defined in the **effective capacity** field in the CS/AIX link station profile. You define boundaries for each of the respective link speed classifications: slow, medium and fast. There are relative classifications which will vary across installations. When defining the classifications, consider the speed of the links on all of the CS/AIX servers running in the pool you are balancing

across. Base weights of 0, 30000 and 60000 are used for fast, medium and slow links respectively. The numbers resulting from the calculation of other overall load criteria are added to this base weight. This virtually guarantees that, regardless of other criteria, a client will always be connected to a host on the fastest link.

- How many active connections are there?

  An active connection is defined as a client that is in session with a host SSCP LU or a host application LU. When using CS/AIX display commands, such sessions are denoted as *ACT/ATT* and *ACT/SESS* respectively.

- What is the overall AIX system load?

  This is defined as the average of the following:

  - Percentage of paging space utilized
  - Percentage of CPU utilized
  - Percentage of MBUFs utilized

  The relative importance of each can be tailored. For example you may want CPU utilization to be three times as important as paging space utilization.

- What is the user defined number?

  This is defined as a positive or negative number and is added to the final weight that is calculated. The most typical use of the user defined number would be to account for differences in processor speed. For example, all things being equal, if you wanted to favor a model 550 RS/6000 over a model 320, you could configure a user defined number of -100 on the model 550. Adding this to the final weight would lower it, and therefore make this server more likely to be selected for a client connection. By default the user defined number is set to zero, which assumes that all servers in your pool can deliver comparable performance given equal loads.

- What is the factor number?

  Since AIX system load is a percentage, it will never be greater than 100. The number of client connections, however, can be far greater than 100. To ensure that client connections are not weighted disproportionately to system load in the overall load calculation, the system load is multiplied by the factor number. This helps equate these important criteria.

Note

The sys_load script determines these factors by using displays of SNA resources and parsing the answers. The load on the system and the number of LUs defined determine how fast the script can run. You may need to adjust ISS parameters to account for the time it takes. You may also want to consider using sys_load as an example and modifying it to meet your own needs, slimming it down in the process.

A.1 Values Returned by sys_load

Regardless of the load factors, there are some simple guidelines to observe regarding the values returned. Assuming the script has not been customized and the scalar values are as shipped with sys_load you can use the following guidelines:

- **99999** - The SNA node is not running.
- **90000 - 99998**: If there are no available LUs you will get 90000 + system load factors. See the note to SDDLU users for more information on what makes an LU available.
- If there are LUs available the number will be based on the speed of the link. This can be seen by looking at the effective capacity (effect_cap) field of the link station.

  `snaadmin -d query_ls,ls_name=`*`link_station_name`*

  60000 - 89999: If the effective capacity is less than 4,000,000 the link is classified as slow, and the number will be 60000 + system load factors.

  30000 - 59999: If the effective capacity is between 4,000,001 and 15,000,000 bits per second, the link is classified as medium and the number will be 30000 + system load factors.

  0 - 29999: If the effective capacity is above 15,000,000, the link is considered high speed and the number returned is 0 + system load factors.

Note to SDDLU Users

The sys_load script considers an LU that has established an SSCP-LU session but not an LU-LU session to be available. This is not really accurate for SDDLU situations because it leaves out all LUs that are capable of establishing a session but have never been used.

With SDDLU, an LU does not have an SSCP-LU session established until needed the first time for an LU-LU session. In the xsnaadmin Node window the LU has a status of inactive until then. After the SSCP-LU session is established the first time, it remains. In the xsnaadmin Node window, the LUs will have SSCP as the status.

In this case it would be more accurate to look only at the LU-LU session status since the absence of an SSCP-LU session does not mean the LU is unavailable. We have altered our sys_load script to account for this. See “sys_load Modified” on page 210.

A.2 sys_load Script

```
#!/bin/ksh
#
# FUNCTIONS: sys_load.sh
#
# ORIGINS: 27
#
# (C) COPYRIGHT International Business Machines Corp. 1995, 1997
# (C) COPYRIGHT Data Connection Limited 1997
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
# USAGE: sys_load <pool_name>
#
#        where <pool_name> is an optional parameter for specifying
#        the LU pool to which LU status queries should be restricted.
#        If no pool name is specified, the status of all LUs is queried.
#
# DESCRIPTION: This ksh script queries the local nodes resources to
```

```
#              determine it's overall load. When determining the
#              load, the script gathers three types of information:
#
#              1. Over all system load, based on CPU, MBUF pool, and
#                 paging space utilization
#
#              2. Active LU 0-3 connection load
#
#              3. LUs currently available for use and what type of
#                 link (speed wise) are they availabe over
#
#              Three functions perform the three above requests. Refer
#              to the function descriptions below.
#
#              After information from all three areas are gathered,
#              it is compared to derive an overall weight for the
#              system. This weight is a numeric range of 0 - 99999 with
#              0 being most favorable. The weight is returned to
#              standard out. Refer to the header and comments of the
#              main routine below for details.
#
#              A remote, managing node will query this node's system
#              load by running this script in an rsh. The output
#              will inform the managing node of the load level on this
#              system and the managing node will decide to which node
#              to route new session requests, based on this weight.
#
# CUSTOMIZATION: Determining system load is a subjective exercise.
#                Certain assumptions have to be made of what takes
#                precedence over the other and what ranges to use
#                in qualifications.
#
#                The administrator may modify the bounds and scalars
#                used to derive system load weights. The values
#                may be customized to reflect what the administrator
#                feels is more applicable or practical for the local
#                environment. Refer to the CUSTOMIZATION topic in
#                each function header to see what can be done to
#                customize the algorithms used.#
# !!!!!!!!!!!!!! PLEASE NOTE !!!!!!!!!!!!!!
#                If this script is customized, be sure it is distributed
#                to each node participating in the load levelling
#                mechanism, or else nodes may be issued weights
#                based on different scalars and bounds.
```

```
typeset -i LUS_AVAILABLE_ON_FAST_LINK=0
typeset -i LUS_AVAILABLE_ON_MEDIUM_LINK=0
typeset -i LUS_AVAILABLE_ON_SLOW_LINK=0

typeset -i SYSTEM_LOAD=0

typeset -i CONNECTION_LOAD=0
#
# FUNCTION: end_and_return
# DESCRIPTION: Takes the return code, adds it to the
#              USER_DEFINED_SCALAR value, and exits the
#              program. If the resulting return value is
#              less than 0, then it is reset to 0.
# CUSTOMIZATION: It is possible to specify a user-defined
#                scalar which affects the weight assigned to this
#                system. This scalar can represent any pertinent
#                information about the local system which should
#                logically weigh it differently than other systems,
#                such as machine model or TCP/IP network speed, or
#                location of this node in relation to its clients
#                as compared to other server nodes. This scalar,
#                specified below as USER_DEFINED_SCALAR, can take
#                any value, positive or negative. The greater the
#                value, the less you prefer this node for being
#                chosen for session routing. The value is simply
#                added to the calculated weight to give an adjusted
#                amount. The default is 0, which indicates no
#                user-defined weight. It is suggested that the range
#     vary between -2000 and 2000.
#
     typeset -i USER_DEFINED_SCALAR=0
#
function end_and_return
{
  typeset -i RC
  RC=$1+$USER_DEFINED_SCALAR
  if [ $RC -lt 0 ]
  then
    RC=0
  fi
  echo $RC
  exit 0
}
```

```
# FUNCTION: get_active_connections_and_classify
#
# DESCRIPTION: This function classifies whether or
#              not the local system's active connection
#              load is light, medium, or high.
#
#              The sum of all active connections is made across
#              all links. If the total is low, then the
#              load on the system is lighter than if the total
#              were higher, even if only one additional LU is
#              available for connection. Once that LU is used, then
#              there will be no more LUs available and this node
#              won't even be considered.
#
function get_active_connections_and_classify
{
  cd /usr/lib/sna
  typeset -i num_act_lus=`LC_ALL=C /usr/bin/snaadmin query_lu_0_to_3 |
    awk 'BEGIN {sscp = 0; appl = 0; count = 0}
         ($1 == "") { if (sscp && appl) {count = count + 1}
                      sscp = 0; appl = 0}
         ($0 == "lu_sscp_sess_active = YES") {sscp = 1}
         ($0 == "appl_conn_active = YES") {appl = 1}
         END {print count}'`

  CONNECTION_LOAD=$num_act_lus
  return
}
#
# FUNCTION: get_overall_system_load_and_classify
#
# DESCRIPTION: This function classifies whether or
#              not the local system's resources are at
#              capacity based on the following information
#
#              o  CPU utilization (vmstat command)
#              o  Paging space utilization (lsps command)
#              o  Mbuf availability (netstat command)
#
# CUSTOMIZATION: If you wish to customize the weights
#                of each of the above factors in determining the
#                system load, redefine the values of the
#                following variable scalars. The scalars represent
#     the ratio of importance one fact has over the
```

```
#     other. For example, if CPU_SCALAR=1 and
#     MBUF_SCALAR=2, then MBUF usage has twice as much
#                impact in figuring system load than does CPU
#                utilization. If PS_SCALAR=2, then paging space
#                and MBUF utilization have equal impact.
                 CPU_SCALAR=1
                 MBUF_SCALAR=2
                 PS_SCALAR=2
#
function get_overall_system_load_and_classify
{
  typeset -i PS_AVG
  typeset -i CPU_UTIL
  typeset -i MBUFS
  typeset -i MBUF_MAX
  typeset -i MBUF_AVG

  # compute the avarage utilization of paging space
  PS_AVG=`lsps -s | awk '{if (NR==2) {l=length($2); m=substr($2,1,l-1); print m}}'`

  # get the current CPU utilization percentage
  # check for vmstat existance
  CPU_UTIL=`if [ -f /usr/bin/vmstat ]; then vmstat 1 2 | awk '{if (NR>4) print
100-$16}'; else echo 50; fi;`

  # get current MBUF pool percent usage
  MBUFS=`LC_ALL=C netstat -m | grep Kbytes | cut -d' ' -f1`
  MBUF_MAX=`LC_ALL=C no -a | grep thewall | awk '{print $3}'`
  MBUF_AVG=100*$MBUFS/$MBUF_MAX

  # compute an overall average based on the percentage
  # usage of paging space, cpu, and mbufs. The order
  # of precedence is as follows:
  #    1. paging space usage
  #    2. Mbuf usage
  #    3. CPU usage
  typeset -i TOTAL_AVG
  typeset -i SCALAR_TOTAL
  TOTAL_AVG=$PS_SCALAR*$PS_AVG+$MBUF_SCALAR*$MBUF_AVG+$CPU_SCALAR*$CPU_UTIL
  SCALAR_TOTAL=$CPU_SCALAR+$MBUF_SCALAR+$PS_SCALAR
  TOTAL_AVG=$TOTAL_AVG/$SCALAR_TOTAL
  SYSTEM_LOAD=$TOTAL_AVG
  return
}
```

```
# FUNCTION: get_lu_availability_on_links_and_classify
# DESCRIPTION: This function classifies active links
#              as FAST, MEDIUM, or SLOW and determines
#              if any LUs are available for use by
#              incoming session requests (i.e. active
#              SSCP session, but no attachment from an
#              application).
#
# CUSTOMIZATION: The determination of the speed for a link
#                is done by querying the details of the link
#                station using each PU and finding its
#                effective capacity. The weight is then
#                assigned according to the range of its effective
#                capacity. The range bounds are listed below and
#                can be customized to achieve different results.
#                Note that the capacity is in bytes per second.
#
#                DLUR PUs cannot be correlated with link stations,
#                so they are defaulted to a speed of "MEDIUM"
#
                 SLOW_LINK_BOUND=4000000
                 MEDIUM_LINK_BOUND=15000000
#
function get_lu_availability_on_links_and_classify
{
  typeset -i num_links=0
  typeset -i active=0
  typeset -i link_capacity=0
  typeset -i num_lus=0
  typeset -i index=1

  cd /usr/lib/sna
  for i in `LC_ALL=C /usr/bin/snaadmin query_pu |
            awk '($1 == "pu_name") { pu_name = $3 }
                 ($0 == "pu_sscp_sess_active = YES") {print pu_name }'`
  do
    num_links=$num_links+1
    attachment=`LC_ALL=C /usr/bin/snaadmin query_pu,pu_name=$i |
            awk 'BEGIN {direct = 0}
                 ($0 == "host_attachment = DIRECT_ATTACHED") {direct = 1}
                 END {print direct}'`
    if [ $attachment -eq 1 ]
    then
```

```
 ls_name=`LC_ALL=C /usr/bin/snaadmin query_pu,pu_name=$i |
            awk '($1 == "ls_name") {print $3 }'`
      active=`LC_ALL=C /usr/bin/snaadmin -d query_ls,ls_name=$ls_name |
                        awk 'BEGIN {active = 0}
                             ($0 = "state = ACTIVE") {active = 1}
                             END {print active}'`
      if [ $active -eq 1 ]
      then
        link_capacity=`LC_ALL=C /usr/bin/snaadmin -d query_ls,ls_name=$ls_name |
                       awk '($1 == effect_cap) {print $3}'`
        if [ $link_capacity -ge 0 -a $link_capacity -lt $SLOW_LINK_BOUND ]
        then
          link_speed[$num_links]="SLOW"
        elif [ $link_capacity -ge $SLOW_LINK_BOUND -a $link_capacity -lt
$MEDIUM_LINK_BOUND ]
        then
          link_speed[$num_links]="MEDIUM"
        else
          link_speed[$num_links]="FAST"
        fi
      else
        # Default DLUR links to MEDIUM speed
        link_speed[$num_links]="MEDIUM"
      fi

      #
      # If the pool name was specified to sys_load, then find all available LUs on a
link
      # by pool name, else just find all the available LUs on that link.
      #
      num_lus=`LC_ALL=C /usr/bin/snaadmin -d query_lu_0_to_3 |
       awk -v pool_name=$POOL_NAME -v pu_name=$i 'BEGIN{sscp = 0; appl = 0; pool = 0;
this_pu = ""; count = 0}
             ($1 == "") { if ((pu_name==this_pu) && sscp && appl && (pool ||
(pool_name==""")))
                            {count = count + 1}
                          sscp = 0; appl = 0; pool = 0}
             ($1 == "pu_name")                                    {this_pu = $3}
             (($1 == "pool_name")          && ($3 == pool_name)) {pool = 1}
             (($1 == "lu_sscp_sess_active") && ($3 == "YES"))    {sscp = 1}
             (($1 == "appl_conn_active")    && ($3 == "NO"))     {appl = 1}
             END {print count}'`
```

```
        if [ $num_lus -ge 1 ]
      then
        lus_avail[$num_links]=1
      else
        lus_avail[$num_links]=0
      fi
    fi
  done

  while [ $index -le $num_links ]
  do
    if [ "${link_speed[$index]}" = "FAST" -a "${lus_avail[$index]}" = "1" ]
    then
      LUS_AVAILABLE_ON_FAST_LINK=1
    elif [ "${link_speed[$index]}" = "MEDIUM" -a "${lus_avail[$index]}" = "1" ]
    then
      LUS_AVAILABLE_ON_MEDIUM_LINK=1
    elif [ "${link_speed[$index]}" = "SLOW" -a "${lus_avail[$index]}" = "1" ]
    then
      LUS_AVAILABLE_ON_SLOW_LINK=1
    fi
    index=$index+1
  done
  return
}

#
# Main routine: set the load weight for this node
#               based on the information gathered
#               by the above-defined function
#               calls.
#
#               The weight may range from 1-18 where
#               1 represents the lightest load.
#

#
# Get the pool name
#
POOL_NAME=$1

#
# If sna is not running, don't even bother; return 99999
```

```
typeset -i sna_active=0
sna_active=`ps -ef | grep snadaemon | grep -v grep | wc -l`

if [ $sna_active -ne 0 ]
then
  sna_active=`snaadmin -d query_node | awk '($1 == "up_time") {print $3}'`
fi

if [ $sna_active -eq 0 ]
then
  end_and_return 99999
fi

#
# Call the functions for collecting system data
#
get_active_connections_and_classify
get_overall_system_load_and_classify
get_lu_availability_on_links_and_classify

#
# Determine weight according to whether or not LUs are available
# and what the system load is and the connection load is
# at this time. The weight is prioritized first by the type of
# link (e.g. fast), then by the connection load and system load.
#

typeset -i NODE_WEIGHT=0

if [ $LUS_AVAILABLE_ON_FAST_LINK = 1 ]
then
  NODE_WEIGHT=0
elif [ $LUS_AVAILABLE_ON_MEDIUM_LINK = 1 ]
then
  NODE_WEIGHT=30000
elif [ $LUS_AVAILABLE_ON_SLOW_LINK = 1 ]
then
  NODE_WEIGHT=60000
else # There are no LUs available on any link, so
     # go by system load and connection load
  NODE_WEIGHT=90000+$SYSTEM_LOAD
  end_and_return $NODE_WEIGHT
fi
```

```
# Add number of active connections to the weight
#
NODE_WEIGHT=$NODE_WEIGHT+$CONNECTION_LOAD
# The system load percentage is factored into the over-all
# node weight by multiplying the system load average by
# by a factor to put on the same scale as the connection load, then
# adding it to the node weight
# The factor is computed by dividing the connection load by 100.
# If the connection load is less than 100, then the factor is set
# to 1.
typeset -i FACTOR=$CONNECTION_LOAD/100
typeset -i REMAINDER=$CONNECTION_LOAD%100
if [ $REMAINDER -ge 50 ]
then
  FACTOR=$FACTOR+1
fi
if [ $FACTOR -eq 0 ]; then FACTOR=1; fi
typeset -i SYS_IMPACT=$SYSTEM_LOAD*$FACTOR
NODE_WEIGHT=$NODE_WEIGHT+$SYS_IMPACT
end_and_return $NODE_WEIGHT
# We'll never be here, but I don't like exiting without saying so.
exit 0
```

A.3 sys_load Modified

We changed the sys_load function to correct two problems. Both corrections are in the get_lu_availability_on_links_and_classify Function. The new function is shown below with comments where we changed something.

```
## FUNCTION: get_lu_availability_on_links_and_classify
#
# DESCRIPTION: This function classifies active links
#              as FAST, MEDIUM, or SLOW and determines
#              if any LUs are available for use by
#              incoming session requests (i.e. active
#              SSCP session, but no attachment from an
#              application).
#
# CUSTOMIZATION: The determination of the speed for a link
#                is done by querying the details of the link
#                station using each PU and finding its
```

```
#                effective capacity. The weight is then
#                assigned according to the range of its effective
#                capacity. The range bounds are listed below and
#                can be customized to achieve different results.
#                Note that the capacity is in bytes per second.
#                DLUR PUs cannot be correlated with link stations,
#                so they are defaulted to a speed of "MEDIUM"
                 SLOW_LINK_BOUND=4000000
                 MEDIUM_LINK_BOUND=15000000
#
function get_lu_availability_on_links_and_classify
{
  typeset -i num_links=0
  typeset -i active=0
  typeset -i link_capacity=0
  typeset -i num_lus=0
  typeset -i index=1

  cd /usr/lib/sna
  for i in `LC_ALL=C /usr/bin/snaadmin query_pu |
            awk '($1 == "pu_name") { pu_name = $3 }
                 ($0 == "pu_sscp_sess_active = YES") {print pu_name }'`
  do
    num_links=$num_links+1
    attachment=`LC_ALL=C /usr/bin/snaadmin query_pu,pu_name=$i |
            awk 'BEGIN {direct = 0}
                 ($0 == "host_attachment = DIRECT_ATTACHED") {direct = 1}
                 END {print direct}'`
    if [ $attachment -eq 1 ]
    then
      ls_name=`LC_ALL=C /usr/bin/snaadmin query_pu,pu_name=$i |
            awk '($1 == "ls_name") {print $3 }'`
      active=`LC_ALL=C /usr/bin/snaadmin -d query_ls,ls_name=$ls_name |
                        awk 'BEGIN {active = 0}
                             ($0 = "state = ACTIVE") {active = 1}
                             END {print active}'`
      if [ $active -eq 1 ]
      then
        link_capacity=`LC_ALL=C /usr/bin/snaadmin -d query_ls,ls_name=$ls_name |
                       awk '($1 == "effect_cap") {print $3}'` 1
        if [ $link_capacity -ge 0 -a $link_capacity -lt $SLOW_LINK_BOUND ]
        then
          link_speed[$num_links]="SLOW"
```

1 effect_cap needs double quotes (") around it.

```
          elif [ $link_capacity -ge $SLOW_LINK_BOUND -a $link_capacity -lt
$MEDIUM_LINK_BOUND ]
        then
          link_speed[$num_links]="MEDIUM"
        else
          link_speed[$num_links]="FAST"
        fi
      else
        # Default DLUR links to MEDIUM speed
        link_speed[$num_links]="MEDIUM"
      fi

#
# If the pool name was specified to sys_load, then find all available LUs on # a
link
# by pool name, else just find all the available LUs on that link.
# 2
     num_lus=`LC_ALL=C /usr/bin/snaadmin -d query_lu_0_to_3 |
        awk -v pool_name=$POOL_NAME -v pu_name=$i 'BEGIN{ appl = 0; pool = 0;
this_pu = ""; count = 0}
             ($1 == "") { if ((pu_name==this_pu) && appl && (pool ||
(pool_name==""")))
                            {count = count + 1}
                          appl = 0; pool = 0}
             ($1 == "pu_name")                                    {this_pu = $3}
             (($1 == "pool_name")           && ($3 == pool_name)) {pool = 1}
             (($1 == "appl_conn_active")    && ($3 == "NO"))      {appl = 1}
             END {print count}'`

      if [ $num_lus -ge 1 ]
      then
        lus_avail[$num_links]=1
      else
        lus_avail[$num_links]=0
      fi
    fi
  done
```

2. We removed all references to the sscp variable. This was because we used SDDLU to define the LUs on the host and the absence of an SSCP-LU session does not mean the LU is unavailable.

```
        while [ $index -le $num_links ]
          do
            if [ "${link_speed[$index]}" = "FAST" -a "${lus_avail[$index]}" = "1" ]
            then
              LUS_AVAILABLE_ON_FAST_LINK=1
            elif [ "${link_speed[$index]}" = "MEDIUM" -a "${lus_avail[$index]}" = "1" ]
            then
              LUS_AVAILABLE_ON_MEDIUM_LINK=1
            elif [ "${link_speed[$index]}" = "SLOW" -a "${lus_avail[$index]}" = "1" ]
            then
              LUS_AVAILABLE_ON_SLOW_LINK=1
            fi
            index=$index+1
          done
          return
        }
```

Appendix B. WebSphere Load Balancing Component Installation

The Load Balancing component of IBM WebSphere Performance Pack is supported on three operating systems: IBM AIX 4.1.5 or later, Microsoft Windows NT 4.0 and Sun Solaris 2.5 or later. In this section we show you step by step how to perform the installation on AIX and Windows NT. The installation on Solaris is pretty similar to the installation on AIX. For further details you can refer to the *eNetwork Dispatcher for Solaris, Windows NT, and AIX User's Guide Version 2,* GC31-8496.

B.1 Installation on AIX

Before starting our discussion, it would be good to describe the hardware and software environment on which we performed our installation. The machine we used as our platform was a uniprocessor IBM RS/6000 43P having 192MB of RAM, 2.2GB of hard disk and one token-ring interface. We installed this machine with AIX Version 4.3.1.

The AIX installation program for each component of IBM WebSphere Performance Pack requires that the Java Virtual Machine (JVM) Version 1.1 or later is installed on the system.

We did not need to install the JVM on our machine because the Java Development Kit (JDK) 1.1.4 file set is automatically installed with AIX Version 4.3.1. Notice that IBM WebSphere Performance Pack requires AIX Version 4.2.1 or later.

By entering the following command we noticed that the default installation of AIX 4.3.1 locates the JVM Java executable file in the directory /usr/bin:

```
which java
```

The CD of IBM WebSphere Performance Pack is provided with the JDK 1.1.4 installation file for AIX, found in the directory JDK/AIX. You need to install the JVM if it is not already installed on your system, or you might want to upgrade the level if it is previous to 1.1.4. To discover the level of the JVM already installed on your machine, you can enter the command:

```
java -version
```

The installation of JDK on AIX is described in the IBM redbook *Network Computing Framework Component Guide*, SG24-2119.

To prepare for installation, follow the steps listed below:

1. Insert the IBM WebSphere Performance Pack CD-ROM in the CD-ROM drive.
2. From a command line, enter the following commands:

   ```
   mkdir /CD-ROM
   mount -rv cdrfs /dev/cd0 /CD-ROM
   ```

3. Enter `cd /CD-ROM/AIX`.
4. To start the installation program, enter `java setup`.

The first screen is the Welcome window. After clicking the **Next** button, you are required to agree to all the items of the software license. If you agree, check the box **Accept all terms of the license**, then click **Next**. You will see another window displaying the IBM WebSphere Performance Pack Version 1.0 Readme File. We suggest that you take a look at the file as it contains interesting information about the product. Then click **Next**, and a dialog will be displayed where you can select the language, but *only* for the Load Balancing Component:

Figure 120. Load Balancing Component Language Selection

Notice that this option is available even if you have not yet specified that you want to install the Load Balancing component. After you click **Next**, you are prompted to enter the IBM WebSphere Performance Pack destination

location. We accepted the default /public/WebSphere, and we had the setup program create such a directory, as shown in the next window:

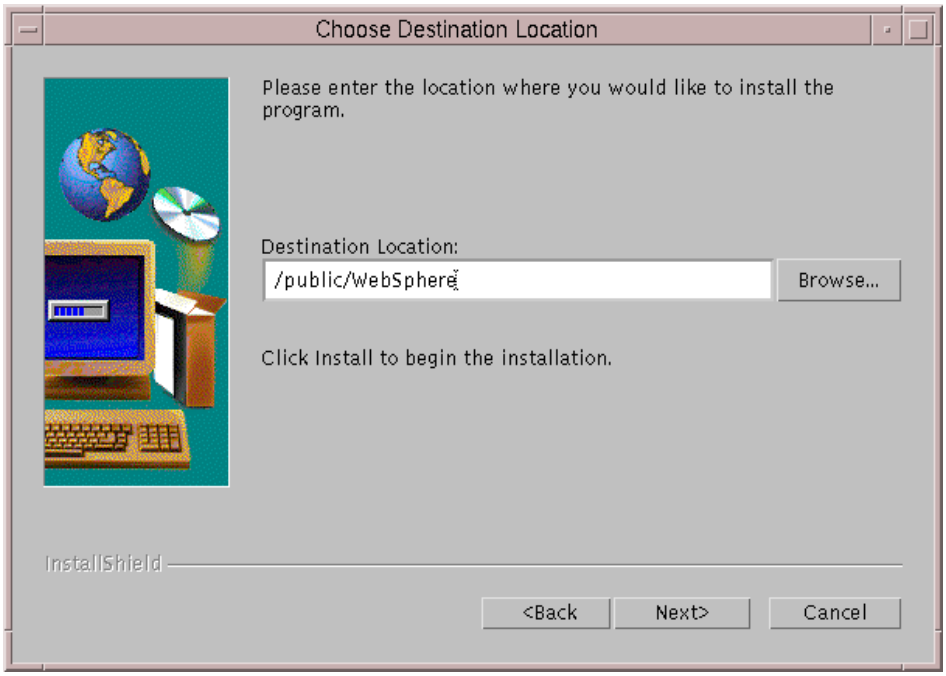


Figure 121. Choose Destination Location

In the above window, you are prompted to click **Install** to proceed. Click the button labeled **Next** to continue instead.

Select now the IBM WebSphere Performance Pack components that you want to install. Note that the installation program allows you to install on AIX these combinations of components:

- Load Balancing (eNetwork Dispatcher) component and/or Caching and Filtering component and/or File Sharing client component
- File Sharing server component and/or File Sharing client component

We selected **Load Balancing (eNetwork Dispatcher)**, and the File Sharing server check box appeared immediately disabled, as shown in the following screen:

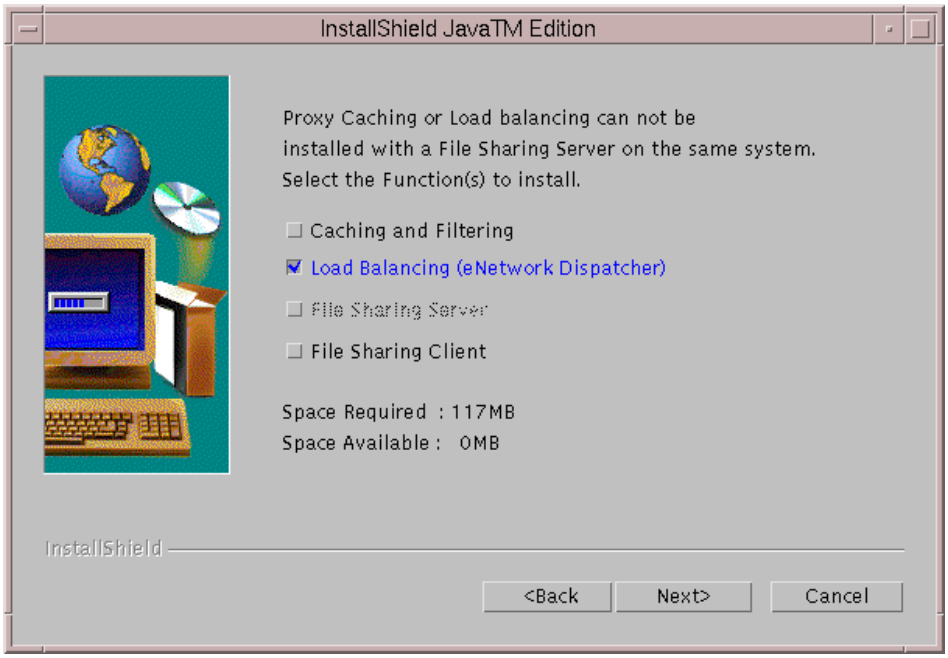


Figure 122. No Space Available for Installation

Note that we accepted the default settings to install the product in the /public/WebSphere directory. As you can see in the above window, it turned out that the available space in the / file system was 0MB. In other words, that file system was empty, and we needed to increase its space to install the Load Balancing component, which requires 117MB. For this reason, we couldn't go on with the installation. By entering the following command we received the confirmation that no free space was available in the mentioned file system:

```
df -k
```

Then we enlarged the file system size following the steps listed below:

1. From a command line, we entered `smitty jfs`.
2. We selected **Change / Show Characteristics of a Journaled File System.**
3. We chose the file system ***/.***
4. Since 117MB of free space is required (see Figure 122 on page 218), we added 250,000 512-byte blocks to the size of the file system (see Figure 123 on page 219), which became 270,336 512-byte blocks sized.

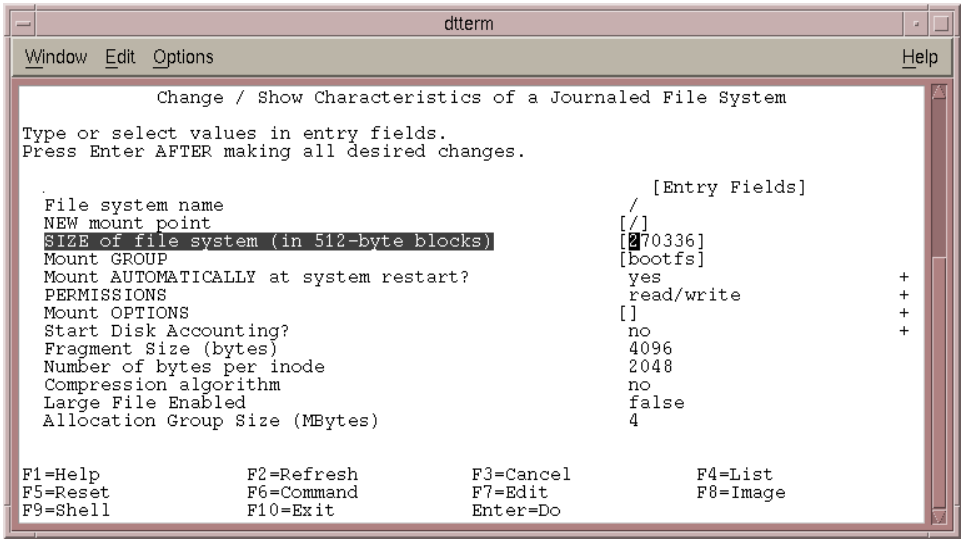


Figure 123. Enlarge the Size of the File System

After expanding the space in the file system, we had enough free space to proceed with the installation, as shown in the following window:

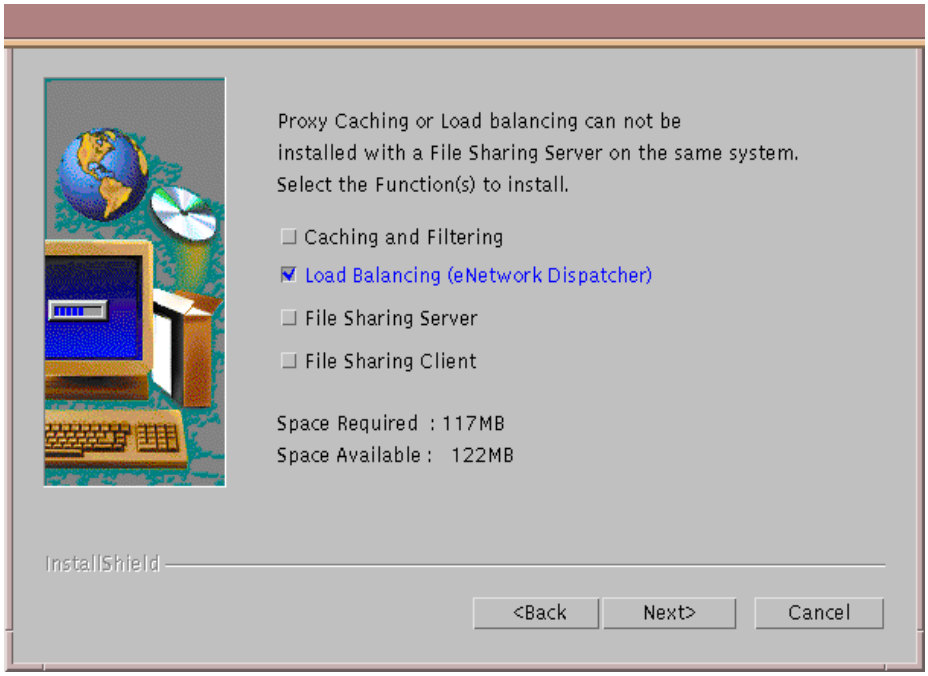


Figure 124. Enough Free Space for the Installation

After you click **Next**, you will get a window that asks if you want the installation program to replace other programs already installed on your system. We selected **No** in this case, since we had not installed any programs on our system yet.

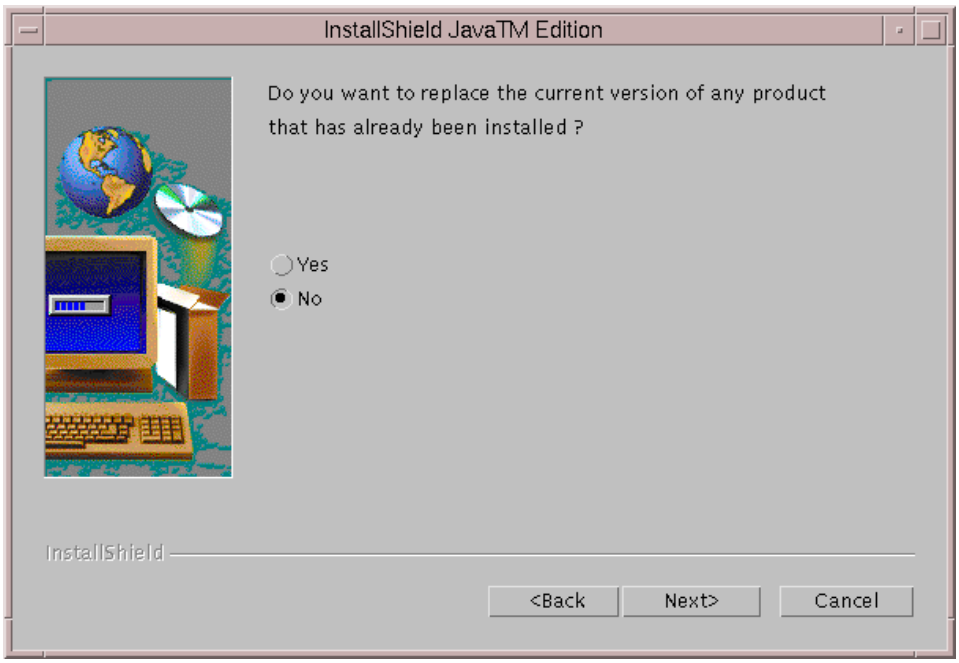


Figure 125. Choose to Replace Version

When we clicked **Next**, the following window was displayed:

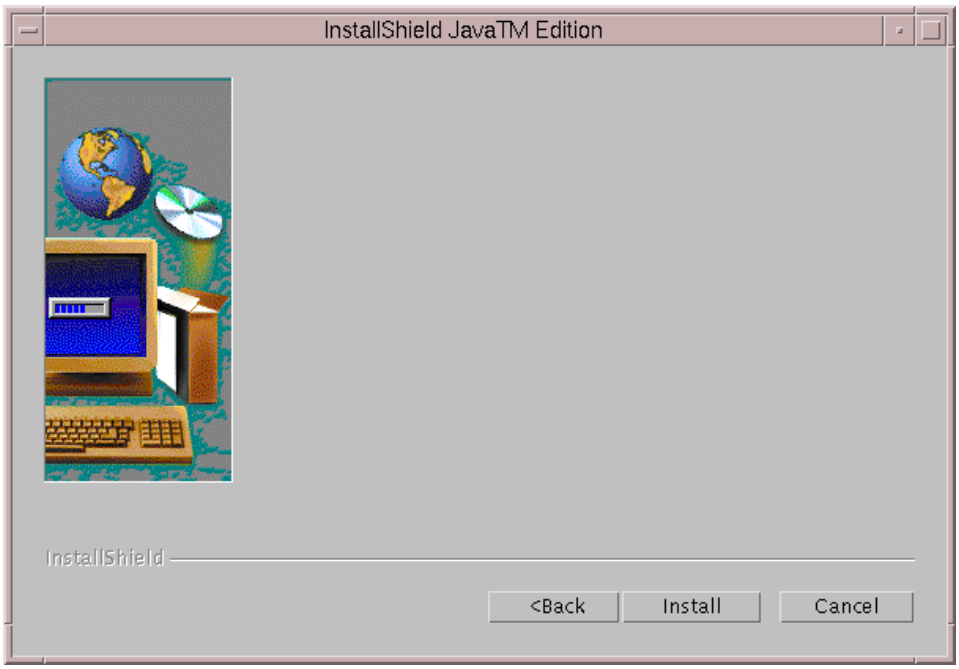


Figure 126. Start Installation Window

Click the **Install** button and you will be informed that the installation is complete.

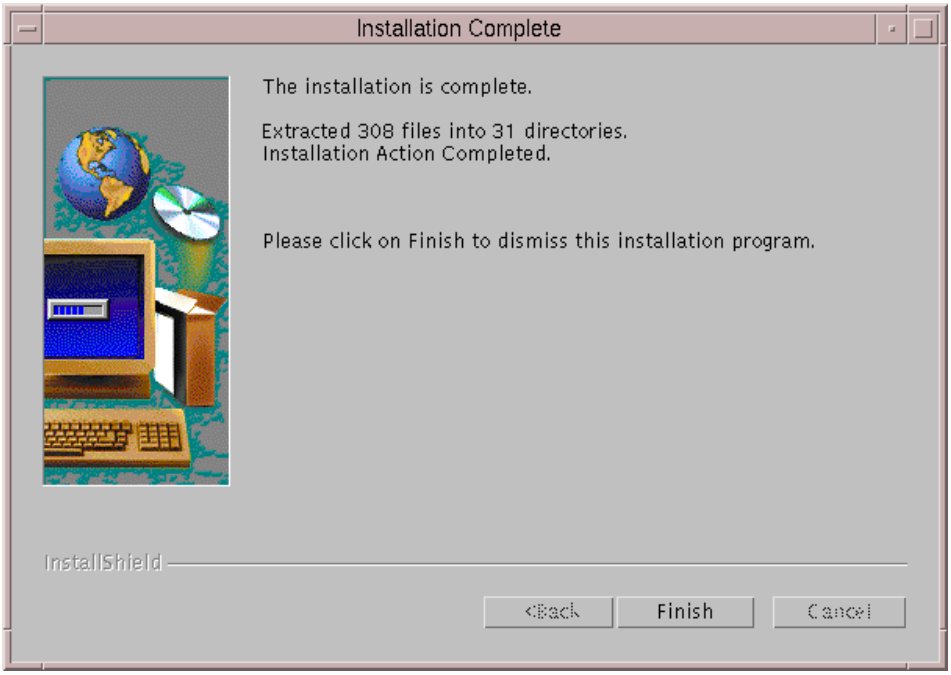


Figure 127. Installation Complete

After clicking **Finish**, we verified that the product had been installed by entering the following command:

```
lslpp -h | grep intnd
```

If the product has been correctly installed (and if you chose the U.S. English version), you will see the following:

```
intnd.iss.license
intnd.iss.rte
intnd.msg.en_US.iss
intnd.msg.en_US.nd
intnd.nd.license
intnd.nd.rte
intnd.ps.en_US
```

B.2 Installation on Windows NT

In this section we describe all the steps that were necessary to install the Load Balancing component of IBM WebSphere Performance Pack on our Windows NT platform.

Before starting with the installation, it would be good to describe the hardware and software environment on which we performed this installation. The machine was an IBM PC 750 with 166 MHz of CPU, 96MB of RAM, 1.5GB of hard disk and one token-ring adapter. This machine had been installed with Windows NT Server 4.0 and Service Pack 3.

The Windows NT installation program for each component of IBM WebSphere Performance Pack makes use of Java InstallShield's setup class.

For this reason you are required to pre-install the JVM Version 1.1 or higher, which is incorporated into the JDK Version 1.1 or higher. Actually you won't need the full JDK, but only its subset known as Java Runtime Environment (JRE), which contains just the JVM, the Java platform core classes, and supporting files. In other words, the JRE is the smallest set of executables and files that constitute the standard Java platform and it contains only the run-time part of the JDK: no compiler, no debugger, no tools. The CD of IBM WebSphere Performance Pack ships with the JDK 1.1.5 for Windows NT found in the directory JDK\NT. However, we preferred to install the JDK 1.1.6 for Windows NT since that was the latest non-beta version that was available at the time we were writing this redbook.

The latest version of the JDK can be downloaded for free from the JavaSoft Web site `http://www.javasoft.com`.

The JDK 1.1.6 installation for Windows NT is very easy and so we will skip its description. However, for further details, you can see the IBM redbook *Internet Security in the Network Computing Framework*, SG24-5220.

To install the Load Balancing component of IBM WebSphere Performance Pack, also known as eNetwork Dispatcher (eND), run the setup.exe program from the CD-ROM installation directory named NT. To do this, from the Start menu, select **Run...**, then click **Browse...** and open the setup.exe program located in the NT directory as shown in the following figure:

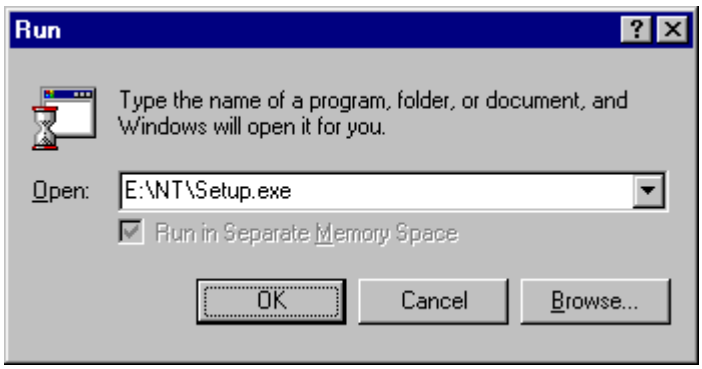


Figure 128. Installation Program's Name and Path

Notice that E in our case was the letter assigned to the CD-ROM drive. Click **OK** to run the installation routine. It will start to find all the JVMs installed on your system. Since we had already installed the JDK 1.1.6, the following window appeared:

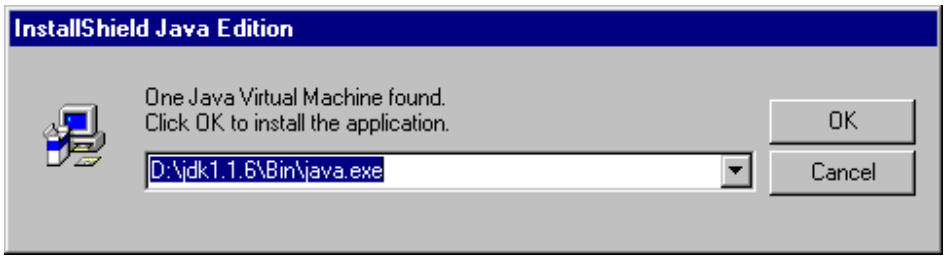


Figure 129. Selection of Java Virtual Machine Version

It's interesting to notice that if no JVMs are installed on your system, the installation routine reacts displaying the following panel:

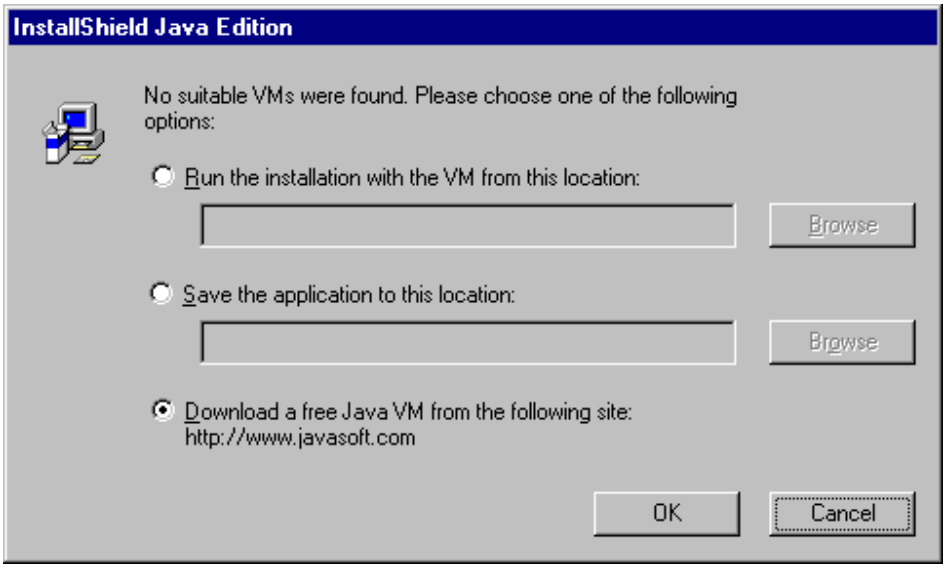


Figure 130. Choose Option If No JVM Is Installed

As you can see in the above figure, you have three options:

1. You can run the installation using a JVM located on a remote location.

   If you select this option, the installation can be performed with a JVM from a remote machine.

2. You can save the Java InstallShield's setup.class on your disk.

   If you choose this second option, the Java InstallShield's setup.class is copied onto a disk in your machine or in the network your machine belongs to, but the installation is not issued.

3. You can download the JVM from the JavaSoft Web site for free.

   By selecting this option, your default Web browser automatically starts and points to the JavaSoft Web site. Also in this case the installation is not issued.

Since we had already installed one version of JVM, we clicked **OK** in the window displayed in Figure 129 on page 225 and we saw the Welcome window. After clicking the **Next** button, you are required to agree to all items of the software license. If you agree, check the box **Accept all terms of the license** and then click **Next**. You will get the a window displaying the IBM WebSphere Performance Pack Version 1.0 readme file. We suggest you take

a look at that file as it contains interesting information about the product. Click **Next** and a dialog will be displayed where you can select the language to use, but only for the Load Balancing component, as shown in the following screen:

Figure 131. Load Balancing Component Language Selection

Notice that this screen appears even if you have not selected yet that you want to install the Load Balancing component.

After you click **Next**, you are prompted to enter the IBM WebSphere Performance Pack installation directory, which by default is named WebSphere, and we had the setup program create it on the D drive as shown in the next window:

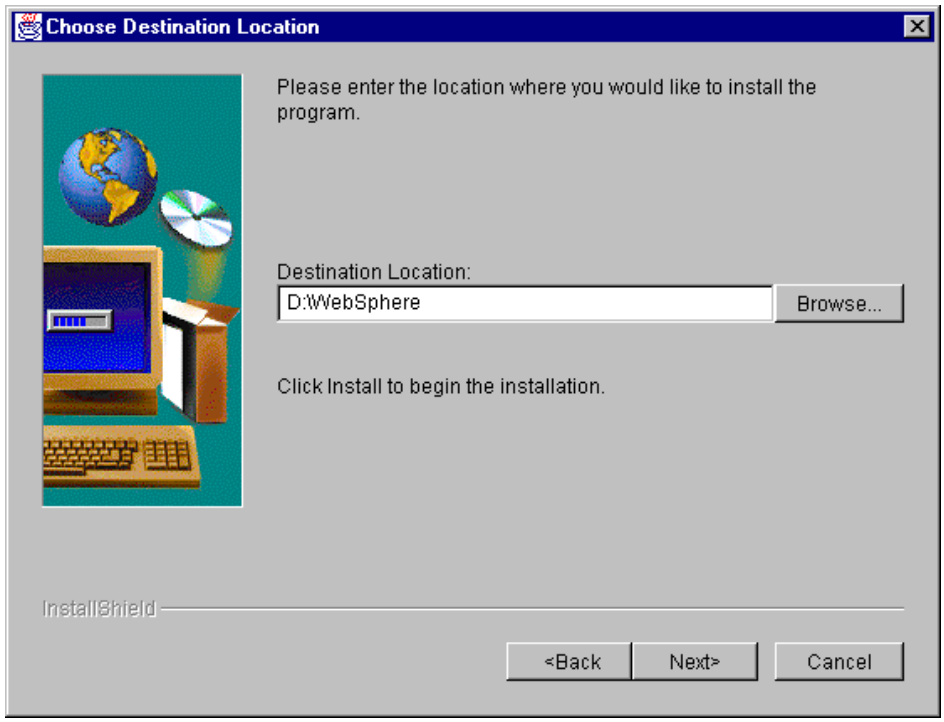


Figure 132. Choose Destination Location

Even if the above window invites you to click Install to continue with the installation, such a button does not exist. You should click **Next** instead.

After that, select the IBM WebSphere Performance Pack component that you wish to install on your Windows NT platform.

Note that the installation program allows you to install on Windows NT only the following component combinations:

- Caching and Filtering component and/or File Sharing client component
- Load Balancing (eNetwork Dispatcher) component and/or File Sharing client component

In other words, you cannot install the Caching and Filtering component and the Load Balancing component on the same Windows NT machine. We selected **Load Balancing (eNetwork Dispatcher)**, and the Caching and Filtering check box appeared immediately, disabled.

Furthermore, note that you cannot select the check box File Sharing Server, since such a component is not available on Windows NT. This is the reason why the related check box appears disabled in the following window:

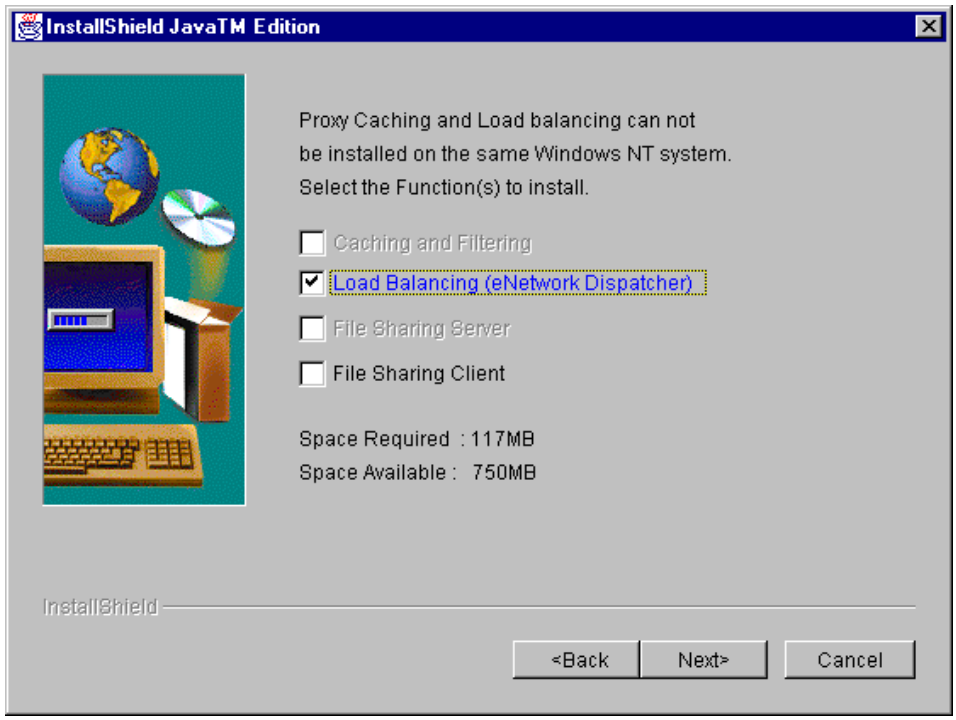


Figure 133. Select Component Window

After you click **Next**, you will get this window, which asks you if you want the installation program to replace other programs already installed on your system.

Figure 134. Choose to Replace Version

Since we had not installed any other IBM WebSphere Performance Pack components on that particular Windows NT machine, we chose **No.** (We also tried in another installation to select **Yes**, and we did not notice any differences.) Then we clicked **Next** and the installation routine started extracting the installation files onto the disk. At the end of this step the following window was shown:

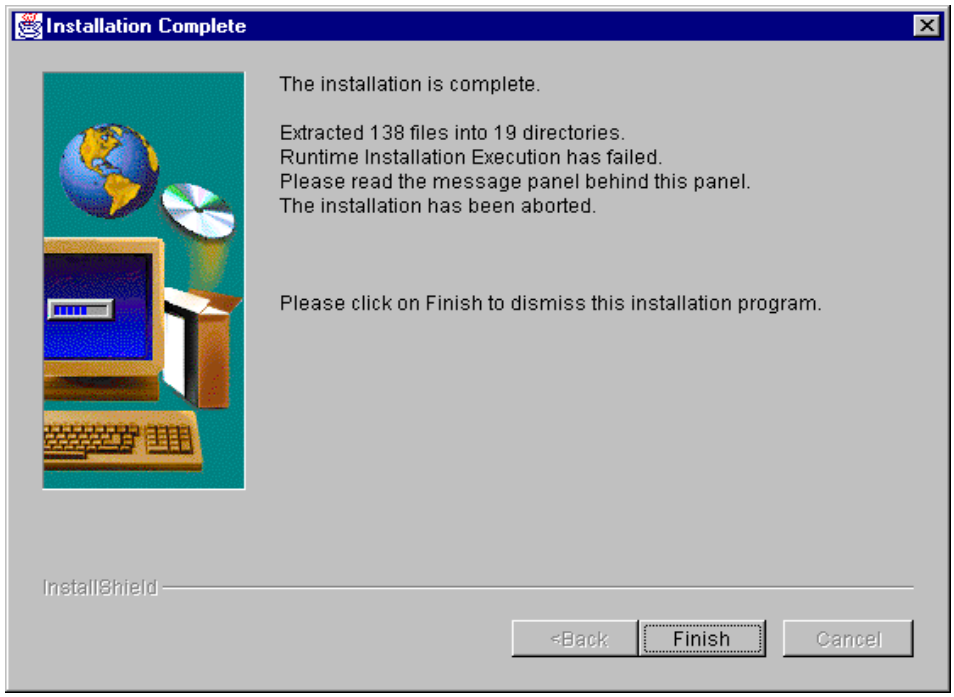


Figure 135. Installation Problem Window

Another window was brought up on our desktop:

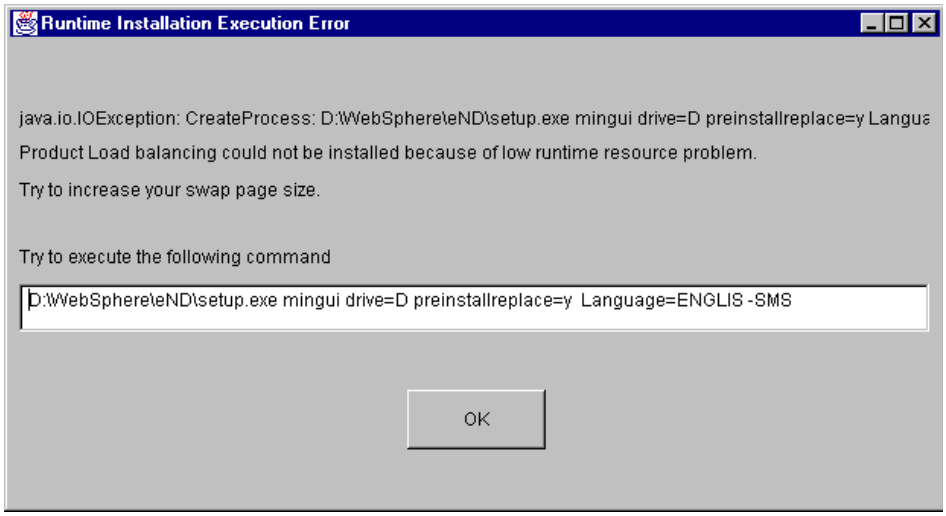


Figure 136. Run Time Installation Execution Error

This panel suggested we increase our swap page size. In our Windows NT machine, the total paging size for all disk volumes was a minimum of 200 and a maximum of 400MB. (We had set two pagefile.sys files in our system, one in the C drive and one in the D drive, each having a size variable between 100 and 200MB.) We also tried another installation where the total paging size for all disk volumes was a minimum of 250 and a maximum of 400MB. (The pagefile.sys file was 150-200MB sized in the C drive and 100-200MB sized in the D drive.) In this case the same problem also appeared.

The recommended command we copied and pasted in a MS-DOS window is shown in the following figure:

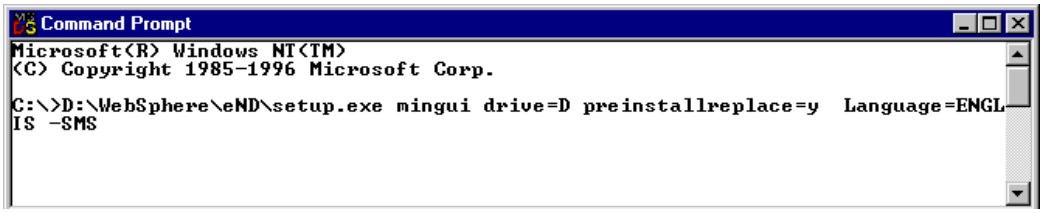


Figure 137. Installation Command

After entering the command shown in Figure 137 on page 232, the following window is displayed from which you can select the type of installation you want to perform:

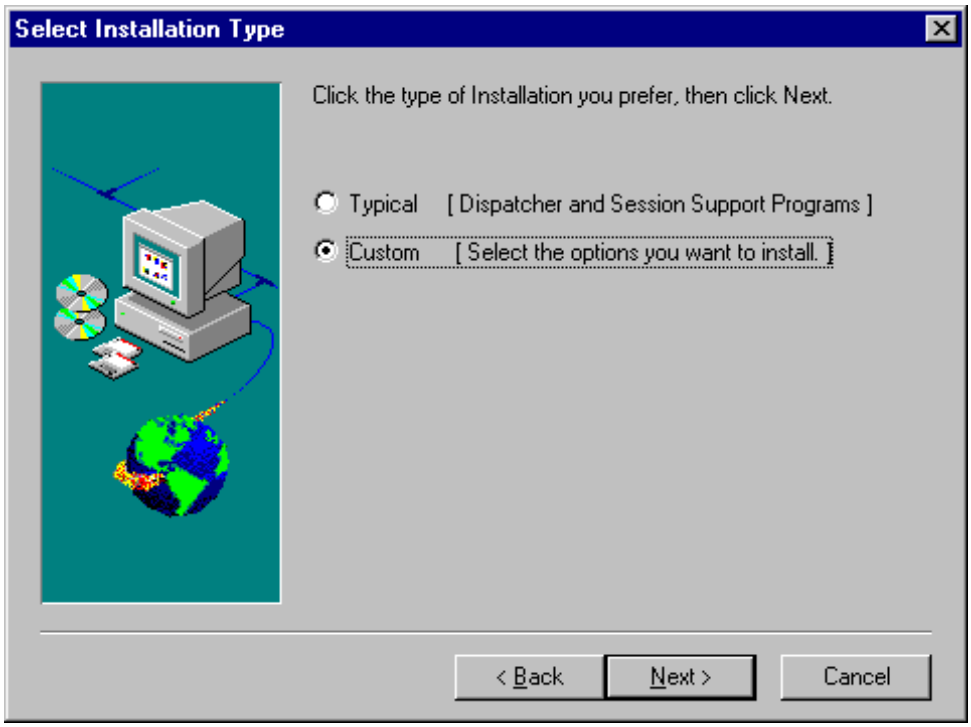


Figure 138. Selection of Installation Type

We selected **Custom** since we wanted to have the ability to personalize the installation conforming to our needs, and then we clicked **Next**. Then we chose to install the two Load Balancing subcomponents (the Dispatcher and the Interactive Session Support) plus the documentation, as you can see in the following window:

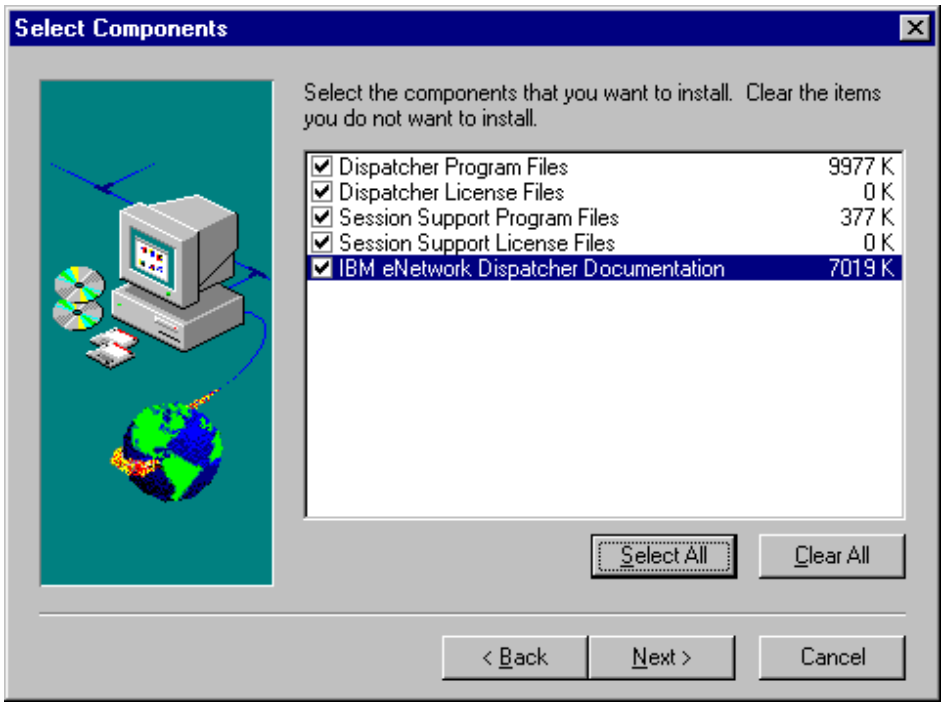


Figure 139. Selection of Components

To continue, select **Next**. The next dialog will be shown, for the selection of the Language files:

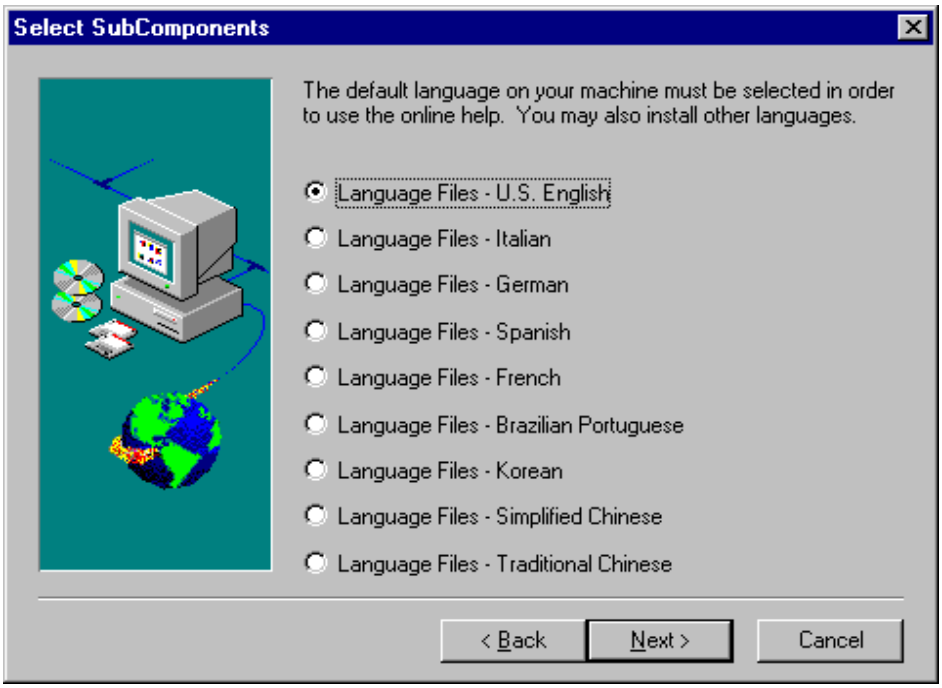


Figure 140. Select Language Files Window

As soon as you click **Next**, the above window disappears and the installation procedure goes on and completes without giving back any other signal. We did not receive any notification that the installation was completed and no panel was brought up to suggest we reboot our system, which usually happens after a complex installation.

You can see that the installation has been completed if you check the services available on your computer in the Services dialog box of the Control Panel folder. You will see that both the IBM Network Dispatcher and IBM_ISS_Load_Balancing services have been created, even if they have not started yet, as shown in the following window:

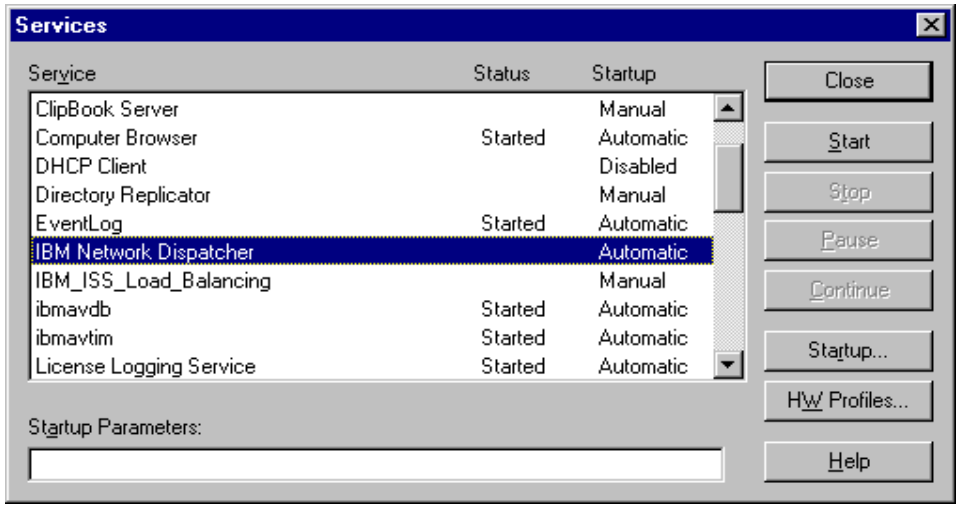


Figure 141. Available Services on Windows NT

Notice that the IBM Network Dispatcher service Startup mode is by default set to Automatic, while the IBM_ISS_Load_Balancing service has the Startup mode set to Manual. We tried to start that service by selecting the **Start** button, and the service Status changed to read `Started`.

Moreover, if you look at the Programs folder of the Start menu, you will find that the eNetwork Dispatcher icon (by which you start the GUI for the eNetwork Dispatcher configuration) is displayed. However, if you select it at this point, you get an error:

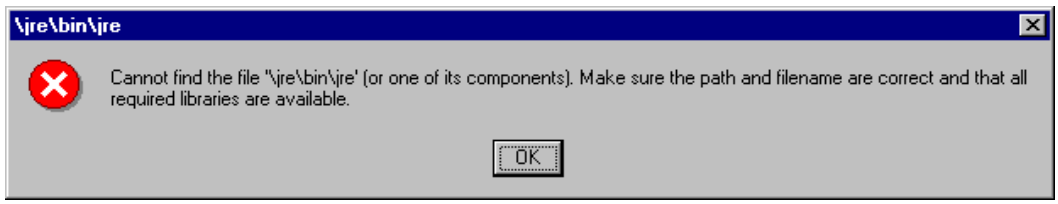


Figure 142. Execution Error

Reboot your system. After the reboot you will find two extra files in the \Temp directory of your boot logical disk. The first is a very large file, setup.class (over 26MB), and the second is getspace.exe. You can remove these files manually after rebooting the system.

Appendix C. WebSphere Interactive Session Support Installation

As we have already mentioned, the Load Balancing component of IBM WebSphere Performance Pack has two subcomponents: the Dispatcher and the Interactive Session Support (ISS).

The ISS function can be installed separately from the other component, the Dispatcher. We have explained, in fact, that ISS has the special function of system monitoring. If you want ISS to be part of your environment, you should install and configure a machine to run as an ISS monitor, while on the TCP servers in the cluster an ISS agent process should run.

On our environment we selected to have the Dispatcher machine run as the ISS monitor too. On that machine, ISS had already been installed as part of the Load Balancing component of IBM WebSphere Performance Pack, so we did not need to perform any further installation steps on it.

Then we had to install and configure ISS on the three TCP servers. On those machines, the ISS agent process would run, gathering information about the status of the systems and feeding that information back to the ISS monitor.

In this section we illustrate how you can install the ISS function on the AIX and Windows NT platforms. The installation on Sun Solaris is not very different from the installation on AIX. We recommend that for further details you refer to the *eNetwork Dispatcher for Solaris, Windows NT, and AIX User's Guide Version 2,* GC31-8496.

C.1 Installation on AIX

You can find the description of the software and hardware platform we used for this installation in B.1, "Installation on AIX" on page 215. As we mentioned there, the installation program for the eNetwork Dispatcher is performed via the Java InstallShield's setup class. If you look again at Figure 122 on page 218, during the installation you are required to select the component you want to install. You are able to select the full **Load Balancing (eNetwork Dispatcher)** component of IBM WebSphere Performance Pack, but not its ISS sub-component only, and in none of the subsequent steps do you have such a possibility.

If you plan to install ISS only, use the `smitty` or `smit` commands, and follow the steps that we are going to describe:

1. Log in as root

2. Insert the IBM WebSphere Performance Pack CD-ROM in the CD-ROM drive.
3. From a command line, enter `smitty`.
4. Select **Software Installation and Maintenance** and press the Enter key.
5. Select **Install and Update Software** and press Enter.
6. Select **Install and Update from LATEST Available Software** and press Enter.
7. Press the F4 functional key to obtain a list of the devices or directories containing installation images.
8. Select **/dev/cd0** (the CD-ROM drive) and press Enter.
9. Move to the **SOFTWARE to Install** line and press the F4 functional key again to get a list of all the software to install.
10. Select the ISS components **eND Interactive Session Support** and **eND Interactive Session Support Product License** as shown in the following figure and press Enter:

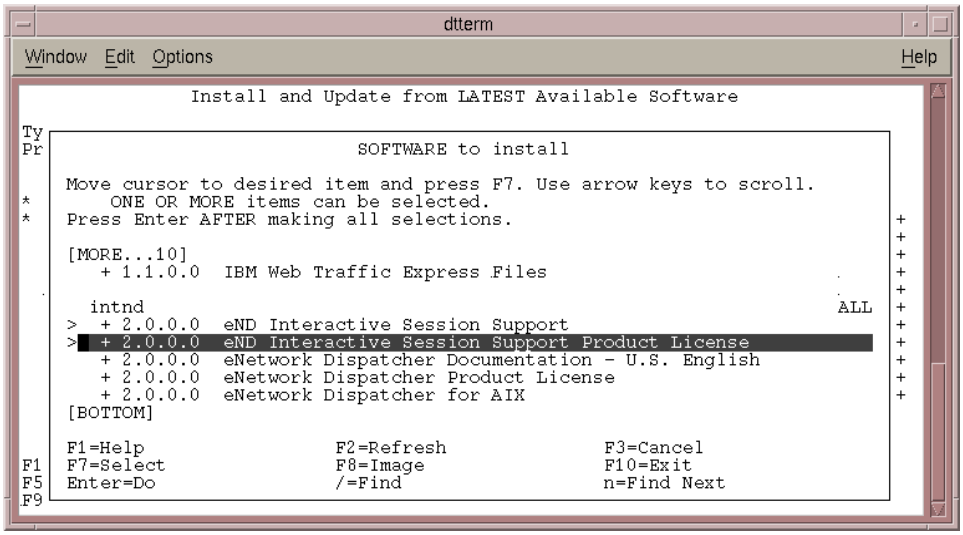


Figure 143. Installing ISS on AIX

11. Again press the Enter key to accept the installation parameters.
12. When the command completes, you can exit by pressing the F10 functional key.

To make sure that the product has been installed correctly, from a command line enter the command:

```
lslpp -h | grep intnd
```

If you have successfully installed the product (U.S. English version) the following lines will be returned:

```
intnd.iss.license
intnd.iss.rte
intnd.msg.en_US.iss
```

C.2 Installation on Windows NT

The software and hardware platform we used for this installation is described in B.2, “Installation on Windows NT” on page 223. We also performed the installation of ISS on other machines with 64MB of RAM.

In B.2, “Installation on Windows NT” on page 223, you will find how the installation program for the Load Balancing component of IBM WebSphere Performance Pack can be performed via the Java InstallShield’s setup class. We followed those directions until we were requested to choose the component to install. Since we wanted to install only ISS, we selected **Session Support Program Files** and **Session Support License Files**, as shown in the next window:

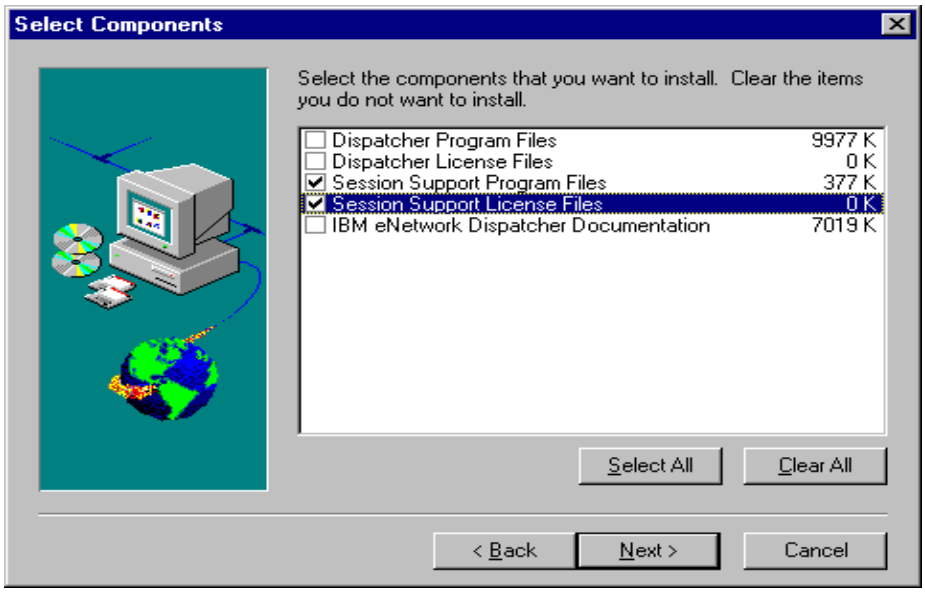


Figure 144. Selecting ISS Components

To continue, choose **Next**. The dialog for the selection of the language files is shown:

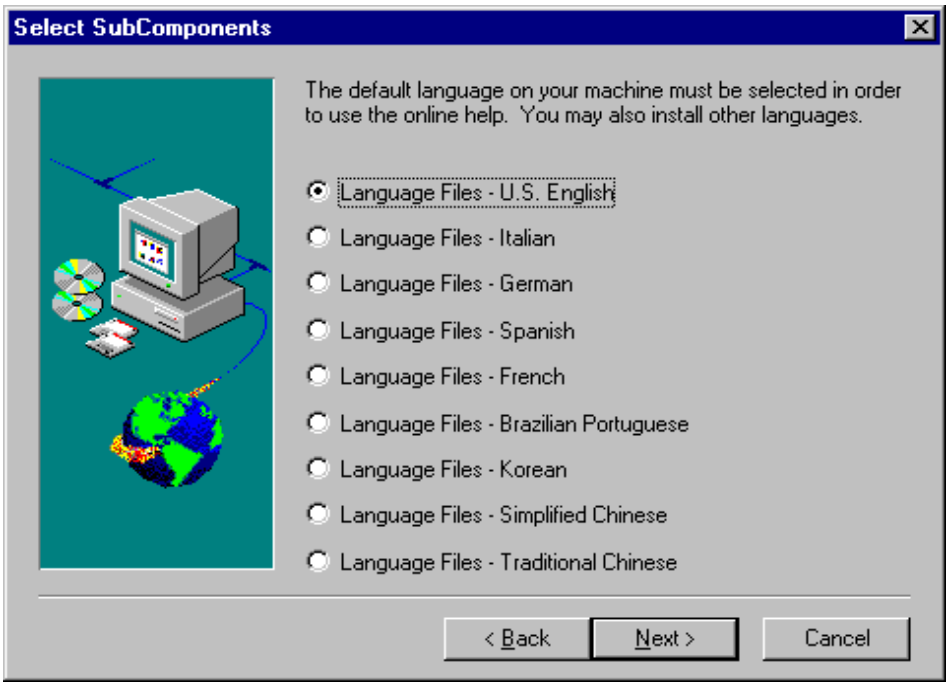


Figure 145. Select Language Files Window

We selected **Language Files - U.S. English**. As soon as you click **Next**, the window disappears and the installation procedure goes on and completes but without giving back any other signal. We did not receive any notification that the installation was completed and no panel was brought up to suggest we reboot our system, as usually happens after a complex installation.

You can see that the installation is finished if you check the services available on your computer in the Services dialog box of the Control Panel folder. You will see that a new service named IBM_ISS_Load_Balancing is in the list of all the available services of Windows NT. Notice that after the installation this service has not yet started since its Startup mode is set to Manual rather than to Automatic, as you can see in the following screen:

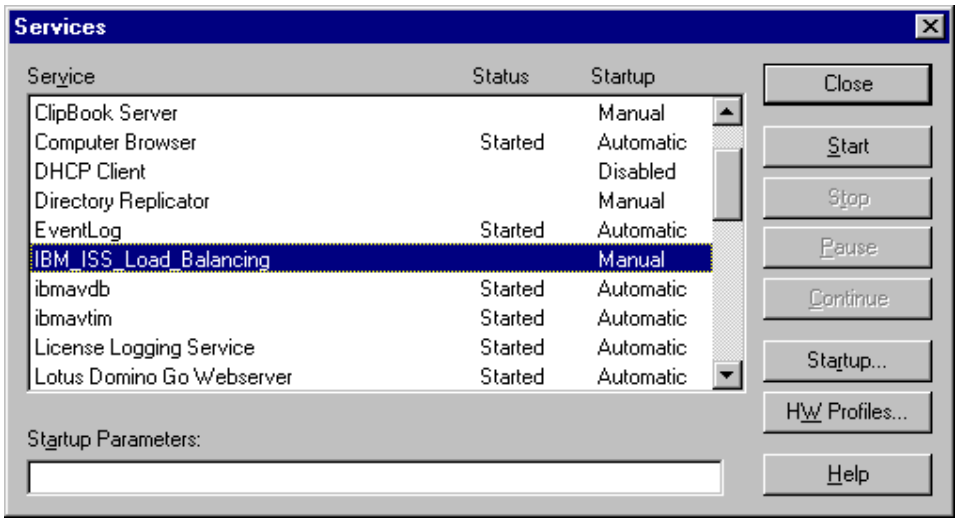


Figure 146. IBM_ISS_Load_Balancing Service Available on Windows NT

Even if the installation process did not require it explicitly, we decided to restart the machine after this installation.

Appendix D. Special Notices

This publication is intended to help technical professionals understand the variety of IBM solutions for load balancing a Communications Server environment. The information in this publication is not intended as the specification of any programming interfaces that are provided the products mentioned in this book. See the PUBLICATIONS section of the IBM Programming Announcement for each product for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have

been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain 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.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

IBM ®
AIX
AS/400
eNetwork
LoadLeveler
OS/2
Netfinity
RS/6000
SecureWay
VTAM
WebSphere

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

Appendix E. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 249.

- *IBM WebSphere Performance Pack Usage and Administration*, SG24-5233
- *IBM eNetwork Communications Server for Windows NT Version 6.0 Enhancements*, SG24-5232
- *Network Computing Framework Component Guide,* SG24-2119
- *Internet Security in the Network Computing Framework*, SG24-5220
- *IBM Communications Server for Windows NT Version 5.0,* SG24-2099

E.2 Redbooks on CD-ROMs

Redbooks are also available on the following CD-ROMs.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

E.3 Other Publications

These publications are also relevant as further information sources:

- *eNetwork Dispatcher for Solaris, Windows NT, and AIX User's Guide Version 2*, GC31-8496
- *IBM eNetwork Communications Server for UnixWare 7 Administration Guide*, SC31-8704
- *Using and Administering LoadLeveler*, SC23-3989

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** `http://www.redbooks.ibm.com/`

  Search for, view, download or order hardcopy/CD-ROM redbooks from the redbooks web site. Also read redpieces and 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.

- **E-mail Orders**

  Send orders via e-mail including information from the redbooks fax order form to:

	e-mail address
In United States	usib6fpl@ibmmail.com
Outside North America	Contact information is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the “How to Order” section at this site: `http://www.elink.ibmlink.ibm.com/pbl/pbl/`

This information was current at the time of publication, but is continually subject to change. The latest information for customer may be found at `http://www.redbooks.ibm.com/` and for IBM employees at `http://w3.itso.ibm.com/`.

IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and redbooks by accessing the IBM Intranet Web site at `http://w3.itso.ibm.com/` and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may also view redbook. residency, and workshop announcements at `http://inews.ibm.com/`.

IBM Redbook Fax Order Form

Please send me the following:

Title	Order Number	Quantity

First name	Last name	
Company		
Address		
City	Postal code	Country
Telephone number	Telefax number	VAT number

☐ Invoice to customer number _____

☐ Credit card number _____

Credit card expiration date	Card issued to	Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

Symbols
/etc/hosts 130
/etc/inetd.conf 130
/etc/resolv.conf 131
/etc/services 130, 134

Numerics
5250 155, 158, 160, 164

A
advisor 15, 17, 19, 67
  customizable 16
  starting 68, 69
AIX
  aliasing the loopback 51
alias the loopback 51
aliasing the cluster address 21, 38, 42, 50
AS/400 160

B
Best selection method 88
broadcast 168, 170

C
cluster address 21, 23, 31, 38, 41, 42, 51
CS/UnixWare 8
CS/UnixWare Win32 client 165
custom metrics 93, 107
customizable Advisors 16

D
Dispatcher 12, 13, 14, 29, 75, 88
  adding a cluster 43
  adding ports 45
  adding servers 48
  high availability feature 42
  Starting and Stopping 32
DNS 24, 75, 86, 87, 93, 97, 99
  starting 99
domain 165, 166, 170, 175, 176, 182

E
eNetwork Dispatcher 11
Executor 15, 19
  Starting 35, 36
  stopping 73

F
FIN 38

G
goIdle 42, 43
group 170, 180

H
high availability 13, 69, 82
High Performance Routing (HPR) 169

I
ifconfig 39, 51
inetd 58, 59
inetd.conf 134
Interactive Session Support (ISS) 24
IntranetWare for SAA 3.0 141
ISS 1, 12, 13, 15, 19, 20, 24, 50, 68, 71, 75, 76, 86, 93, 107, 119, 237
  configuration 76
  starting 127
  starting and stopping 89
  stopping 92
ISS cell 76, 80
ISS configuration files 76
ISS function 13
ISS observers 86
ISS resource 83
ISS selection methods 88
ISS service 76, 82
isscontrol command 91
issd
  starting 100, 111
issd command 89
ISSNameServer 87

L
lo0 50, 51
load balancing 195
LoadLeveler 77, 115
loopback 21, 50, 51, 53

LU pool 195

M
Manager 15, 19, 50, 68
  starting 20, 68
metric 84
multicast 145

N
named
  starting 99, 127
named.boot 123, 131
named.data 118, 119, 123, 124, 131
named.rev 123, 125, 132
NameServer 86
ndadmin command 33
ndconfig command 39
ndcontrol command 20, 36, 37, 44, 46, 49, 60, 64, 71, 73
ndserver command 32
netstat -ni 40, 51
node 76, 82
non-forwarding address 31, 37
nslookup 99

O
observer 86

P
PCOMM 141, 151, 153, 155, 158, 160, 162
port 23 57, 60, 67
  sharing on AIX 57
  sharing on UnixWare 58
ports used by the Dispatcher 20
proportions of importance 15, 17, 71

R
Reach advisor 69
registry 170
resolv.conf 127
round-robin 2, 88, 169
RoundRobin selection method 88
route print 55

S
sacadm 59
scope 144, 148, 149, 153, 157
server failure 89
SLP 3, 141, 160, 162, 195
smoothing index 71
SNA gateway 195
sna.net 167, 175
sna_domn.cfg 166, 182
snaadmin add_backup 175
snaadmin delete_backup 176
snaadmin query_sna_net 172, 176
snainetd command 58, 59
snawinsec domain command 172
sys_load 93, 95, 107, 110, 116, 119, 128, 197

T
TN3270 195
tpinst32 171

U
unicast 145
UnixWare 165
  aliasing the loopback 51

W
WAN 13, 48
Weight bound 50
weighted round-robin 15, 25, 50, 60, 67
Win32 client 165
Windows Open Systems Architecture 165

ITSO Redbook Evaluation

Load Balancing for eNetwork Communications Servers
SG24-5305-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Which of the following best describes you?
_ **Customer** _ **Business Partner** _ **Solution Developer** _ **IBM employee**
_ **None of the above**

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes___ No___

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____

_____

_____

_____

SG24-5305-00
Printed in the U.S.A.

Load Balancing for eNetwork Communications Servers

SG24-5305-00