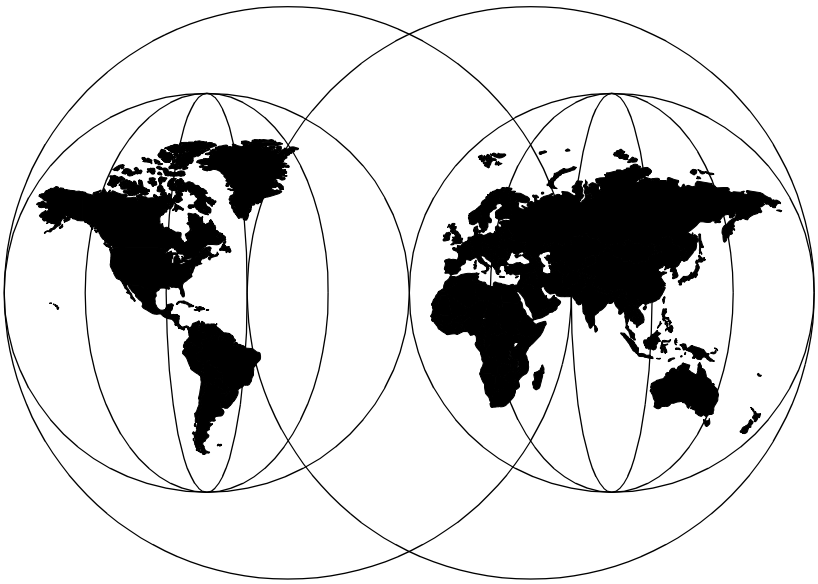
SG24-4993-00

Load-Balancing Internet Servers

December 1997



IBM

International Technical Support Organization
Austin Center

SG24-4993-00

International Technical Support Organization

Load-Balancing Internet Servers

December 1997

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 131

First Edition (December 1997)

This edition applies to IBM Interactive Network Dispatcher Version 1.2 for AIX.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

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

Contents

Figures

Tables

Preface

Written for systems administrators, Webmasters, and others, this redbook provides a comprehensive overview of IBM's Interactive Network Dispatcher product, the definitive product of its type for creating balanced Internet server environments.

A variety of configurations and environments, including World Wide Web, File Transfer Protocol and high-availability scenarios, are presented as are all the concepts involved in providing a highly efficient, cost-effective solution.

Some practical examples are presented to demonstrate how to design and configure an environment with Interactive Network Dispatcher.

The examples provided throughout this document are not supported by IBM, but are provided as a guide to allow you to gain from the experience and knowledge of the people who wrote this redbook. Without a doubt, sharing their experiences with Interactive Network Dispatcher will shorten the time it takes you to implement your own load-balancing solution.

How This Redbook is Organized

The chapters in this redbook are designed to provide a guided approach to understanding the concepts involved in, and functions provided by, Interactive Network Dispatcher; the book is organized as follows:

- Chapter 1, "The Big Picture" on page 1

  Introduces the Internet Server environment and explains the need for large-capacity, scalable servers and load balancing

- Chapter 2, "Interactive Network Dispatcher Concepts" on page 9

  Explains what the IBM Interactive Network Dispatcher is and how it works and introduces most of the concepts that will be explored in the following chapters

- Chapter 3, "Interactive Network Dispatcher in Detail" on page 17

  Explores Interactive Network Dispatcher, explaining the parameters that can be used to customize the configuration files to make them appropriate for your own needs

- Chapter 4, "Installation and Operation" on page 39

  Contains a brief explanation on how to plan, install and operate the Interactive Network Dispatcher product, presenting a few guidelines based

on the information obtained from the *Interactive Network Dispatcher User's Guide*

- Chapter 5, "WWW Server Scenario" on page 47

  Describes the specific use of Interactive Network Dispatcher to balance the load of World Wide Web (WWW) servers

- Chapter 6, "FTP Server Scenario" on page 61

  Describes the specific use of Interactive Network Dispatcher to balance the load of an File Transfer Protocol (FTP) server

- Chapter 7, "Other Server Scenarios" on page 71

  Summarizes the use of Interactive Network Dispatcher to balance the loads of some specific Internet scenarios

- Chapter 8, "High Availability Server Scenario" on page 75

  Discusses options and considerations for increasing the availability of services using Interactive Network Dispatcher

- Chapter 9, "Two-Tier Network Scenario" on page 87

  Discusses the Two-Tier configuration option that allows load-balancing across a collection of servers. This is most commonly used to serve geographically dispersed groups of users and servers

- Chapter 10, "Configuration Files - Samples" on page 93

  Presents the configuration files and the parameters used for the tests performed

- Appendix A, "Minimal Configuration" on page 111

  Shows the minimal configuration needed to perform a minimal test or to demonstrate Interactive Network Dispatcher in a restricted environment, with very few machines

- Appendix B, "Domain Name System (DNS)" on page 115

  Covers the relevant concepts for configuring a DNS server, concentrating particularly on the requirements for delegating domains to other name servers

- Appendix C, "Hints and Tips" on page 127

  Presents the hints and tips that were gained from our experiences writing this book

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, Austin Center.

Luis Ferreira is an Advisory System Engineer at the International Technical Support Organization, Austin Center. Before joining ITSO in mid-1997, Luis worked for IBM Brazil as an IBM Certified I/T Specialist in the Open System Center providing pre- and post-sales support for AIX and Internet customers. Luis has been working with the UNIX environment since 1983 as a software developer. In 1990, he joined IBM to work for the Field System Center team supporting AIX, system performance, high availability, and system administration.

Cameron Ferstat is the Webmaster and Systems Architect for the IBM Olympic Internet Team in New York. Before joining this group in mid-1997, Cameron worked at the International Technical Support Organization, Austin Center, writing redbooks and teaching IBM classes worldwide on AIX in general and on Internet servers in particular. Before joining the ITSO, Cameron worked for IBM Australia providing pre- and post-sales support to AIX customers. Cameron has been working with AIX and the RS/6000 for over eight years and running WWW servers for four years.

Laura Castia is an AIX System Engineer at IBM Italia, NC Competence Center, Cagliari. She holds an electrical engineering degree from the University of Cagliari, Italy. Laura has been working with AIX, the Internet, Intranet, and RS/6000 for two years.

Soo-Young Lee is a Network Specialist in Seoul, where she provides support and services for Internet customers on AIX and Windows NT. Soo-Young has worked at the Network Solution Service in IBM Korea for one year.

Marcus Vinicius Itala Ferreira is an AIX Product Systems Specialist at IBM Brazil, where he provides post-sales services for customers in AIX, Internet and RS/6000 SP areas. Marcus has three and a half years of experience in the AIX field and two and a half years of experience in the Internet field.

Gerardo Vega is an AIX and network computing I/T Specialist at IBM Mexico, where he provides pre- and post-sales support to customers. He has seven years of experience in AIX and three years in the Internet field. He has worked at IBM for seven years. He has written extensively on AIX, DCE and Internet Web servers.

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

Earl Mathis
Chris Gage
George McDaniel
John Balogh
Steve Roma
Steve Fonts
IBM Research Triangle Park, North Carolina

Curtis Hoskins
Crystal Rothaupt
IBM Poughkeepsie, New York, U.S.A.

Martin Murhammer
International Technical Support Organization, Raleigh Center.

Andreas Hoetzel
Heinz Johner
International Technical Support Organization, Austin Center

Marcus Beattie
IBM Hursley Park, Winchester, U.K.

Marcus Brewer, Editor
International Technical Support Organization, Austin Center

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 the ITSO Redbook Evaluation to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com/`
  For IBM Intranet users `http://w3.itso.ibm.com/`

- Send us a note at the following address:

  `redbook@US.ibm.com`

Chapter 1. The Big Picture

In the business world of the 1990s, the computer environment most often means a network environment. More and more computers are involved with integration and communication. The Internet model, structure and protocol are here to help us implement these needs.The Internet is the world's largest network, and it has become essential in organizations such as government, academia and commercial enterprises. Transactions over the Internet are becoming more common, especially in the commercial arena.

The information that organizations have on their traditional or legacy business applications may now be published and accessible to a wide audience. This access may include a person checking a bank savings account, making a hotel reservation or buying tickets for a concert. Making this information or service available for their customers is a competitive advantage for any organization. However, regardless of the innovation and potential benefits provided by a company's Internet solution, its value is greatly reduced if the information cannot be accessed in a reasonable response time.

The Internet model allows the distribution of services among different servers, called Internet Servers. It is not necessary to tie an application to one specific server. Instead, the service belongs to a group of servers; so an additional computer can be added or removed when necessary. By grouping the set of servers in a single entity, a big issue comes up—*load balancing*.

Optimum performance is achieved by balancing the load of these servers, and the IBM Interactive Network Dispatcher software is the best solution for addressing load-balancing issues.

1.1 Internet and Intranet Sizing

When a server is established on the Internet, one of the most difficult tasks is to predict the number of accesses it will receive. For some businesses, putting a site on the Internet is like opening a new branch office in every city around the world. Although the load on the site will probably start at a low level, it can grow at unpredictable rates. If a Web site is chosen as a "pick of the day" or "cool site of the day," the hits on a server can increase from hundreds to tens of thousands overnight.

In addition, Internet-style services such as the World Wide Web and the Usenet news servers are increasingly being utilized within companies on their private corporate intranets. This is due to the ease and speed with which

these services can be implemented and integrated with existing computer systems to provide very efficient intra-company communications and new internal computer systems accessed through a common (Web browser) interface. In fact, the use of such services on intranets is growing up to ten times faster than Internet implementations.

It is easier to size a server for an intranet because the potential number of users can be determined more accurately. Since the client and servers are normally connected to the same networks, the speed of the connection is usually not a problem. This means can be provided richer content (higher quality graphics, multimedia, and so on) in the services and information offered. But the richer the content, the more load it puts on the server; so there is still a strong requirement for scalability in intranet servers.

1.2 Sizing for Growth

The load on an Internet site is unlikely to remain constant. The number of accesses on a Web server or FTP server can increase for several reasons.

1. Most companies add their Web site's address to television, radio and print advertising and to product catalogues and brochures. As these "Web-aware" publications circulate and replace the previous "URL-free" versions, awareness of the Web site grows.
2. As time passes, the Web site gains better coverage in the on-line search engines such as Yahoo or Alta Vista.
3. Assuming the site is providing useful information or a useful service to customers, repeat visitors should increase.
4. Most Web sites begin simply, with fairly modest content—mostly text, with some images. As the site designers grow in confidence, more resources are allocated, and as Web users in general increase their modem speeds, most sites move towards richer content. Thus, not only do hit rates increase, but the average data transfer per hit also rises.
5. Most sites begin as presence sites—providing corporate visibility on the Internet and making information about the company available to potential customers. Most present sites use predominantly static HTML pages. Static pages are generated in advance and stored on disk. The server simply reads the page from the disk and sends it to the browser. However, many companies are now moving towards integration applications that allow users of the Web site to directly access information from the company's existing applications. This could include checking the availability of products, querying bank account balances or searching problem databases. These applications require actual processing on the

server system to dynamically generate the Web page. This dramatically increases the processing power required in the server.

1.3 Dealing with the Growth

There are several ways to deal with the growth of your Internet site:

1. Purchase an initial system that is much too large.

   This is one way to deal with Web site growth; however, most companies are not willing to invest large sums of money in a system that is much larger than they require—particularly since the benefits that they will gain from the site have yet to be proven. Most prefer to purchase a minimal initial system and to upgrade in the future as the site demonstrates its worth to the company.

2. Replace the server with a larger system.

   This is certainly a possible solution. The RS/6000 suits this solution better than many other platforms since binary compatibility is maintained across the entire range. The same operating system and applications can be run, and the same management skills will apply on the replacement system. However, this method is ultimately limited by the size of the largest system manufactured, which may not be large enough for some Web sites.

3. Purchase an upgradable SMP system.

   Purchasing a symmetric multiprocessing (SMP) system provides a low-disruption upgrade path for your Web server. However, there is a limitation to the amount of scaling that can be achieved. Most SMP systems can be upgraded to at most eight times their original performance level. This may not be sufficient alone, but may well be used in conjunction with the final method.

4. Perform load balancing between multiple servers.

   In this case, the load for the overall site is balanced between multiple servers. This allows scaling beyond the maximum performance available from a single system and allows for easy upgrading by simply installing additional servers and reconfiguring the cluster to use the additional servers. This solution can also provide the added benefit of higher server availability. The load-balancing software can automatically allow for the failure of a single server and balance the load between the remaining sites. This is the solution provided by the Interactive Network Dispatcher software.

1.4 Interactive Network Dispatcher Overview

The Interactive Network Dispatcher is a superior load-balancing software that evenly distributes load among mirrored servers by routing TCP/IP session requests to different servers within a group of servers. It increases the performance of servers, and it balances the requests among all the servers by operating transparently to users and other applications. It is useful for applications such as e-mail servers, World Wide Web servers, distributed parallel database queries, and other TCP/IP applications.

Interactive Network Dispatcher enables multiple Web servers to efficiently function as a single system to better manage high volumes of information and electronic transactions over networks. This optimizes Web site performance, maximizes existing hardware investments, simplifies the administration of Web servers and improves availability of Web site resources and end user satisfaction.

Interactive Network Dispatcher (IND) consists of two components:

- Interactive Session Support balances the load on servers within a local or wide area network using a round-robin approach or a user-specified approach, with or without a DNS name server.
- Network Dispatcher balances the load on servers within a local or wide area network using a number of weight and measurements that are dynamically set by Network Dispatcher.

These two components can be used separately or together to implement a powerful, flexible, and scalable load-balancing solution to address the Internet or intranet server needs of the largest enterprise. If visitors to your site can't get through at times of greatest demand, IND can automatically find the optimal server to handle incoming requests, thus enhancing your customers' satisfaction and your profitability.

For more information on Interactive Network Dispatcher, see Chapter 2, “Interactive Network Dispatcher Concepts” on page 9, and Chapter 3, “Interactive Network Dispatcher in Detail” on page 17.

Interactive Network Dispatcher is a member of the e–business enhancer category of the IBM Network Computing Framework. The official site of this product is at the following URL:

`http://www.ics.raleigh.ibm.com/netdispatch/`

1.5 Why Do I Need Interactive Network Dispatcher?

The number of users and networks connected to the global Internet is growing exponentially. This growth is causing problems of scale that can limit users' access to popular sites.

Currently, network administrators are using numerous methods to try to maximize access. Some of these methods allow users to choose a different server at random if an earlier choice is slow or not responding. This approach is cumbersome, annoying and inefficient. Another method is the standard round-robin approach, where the domain name server selects servers sequentially to handle requests. This approach is better, but still inefficient because it blindly routes traffic without any consideration of the server workload. In addition, even if a server fails, requests continue to be routed to it. For more information on approaches to managing server workload, including animated examples, go to the following URL:

```
http://www.ics.raleigh.ibm.com/netdispatch/examples.htm
```

The Interactive Network Dispatcher software offers numerous benefits over earlier and competing solutions:

Scalability

As the number of client requests increases, servers can be added dynamically, providing support for tens of millions of requests per day, on tens or even hundreds of servers.

Efficient use of equipment

Load-balancing ensures that each group of servers makes optimum use of its hardware by minimizing the hot-spots that frequently occur with a standard round-robin method.

Easy integration

Interactive Network Dispatcher uses standard TCP/IP protocols. It can be added to an existing network without making any physical changes to the network. It is simple to install and configure.

Low overhead

Interactive Network Dispatcher needs only to look at the inbound client-to-server flows. It does not need to see the outbound server-to-client flows. This significantly reduces its impact on the application compared with other approaches and can result in improved network performance.

Non-invasive technology

Interactive Network Dispatcher does not modify any packets, nor does it require any modifications to the operating system on which it runs.

High Availability

Interactive Network Dispatcher offers built-in high availability solutions; the primary Network Dispatcher has one or more backups that receive the status of all connections. When the primary Network Dispatcher machine fails, back-up machines take over immediately. See Chapter 8, “High Availability Server Scenario” on page 75.

1.6 Where Interactive Network Dispatcher Is Being Applied

Interactive Network Dispatcher is used on IBM's Web site with resulting throughput that is almost four-times faster than the previously used domain name server approach. Access the site at:

`http://www.ibm.com/`

During the 1996 Summer Olympics in Atlanta, Internet services were handled by Interactive Network Dispatcher, which routed requests among nearly five dozen hosts in five locations around the world. In the 17 days of the Olympics, the site received just over 192 million hits. At peak times, the Olympics site received 17 million hits per day, including real-time audio requests lasting an hour or more, as well as requests for graphic images and other very large files.

Interactive Network Dispatcher is an integral part of the architecture used to serve the official Web site of the upcoming XVIII Olympic Winter Games in Nagano, Japan. Access the site at:

`http://www.nagano.olympic.org/`

The same infrastructure is serving IBM’s Olympic Web site—a site devoted to explaining IBM’s role as a sponsor of the 1998 Winter Olympic Games. Access this site at:

`http://www.ibm.com/Olympic/`

The following high-volume sites have also used Interactive Network Dispatcher to optimize servers usage:

- American School Directory Project

  `http://www.asd.com/`

- Masters Tournament

  `http://www.masters.org/`
- PGA Championship

  `http://www.pga97.com/`
- Roland Garros, The French Open

  `http://www.frenchopen.org/`
- The Australian Open

  `http://www.ausopen.org/`
- The US Open

  `http://www.usopen.org/`
- Wimbledon Championship

  `http://www.wimbledon.org/`

Sizing for the Internet - The ACM Chess Challenge

IBM's chess-playing supercomputer, Deep Blue, challenged World Chess Champion Gary Kasparov to a battle between man and machine. For the duration of the tournament, IBM ran a Web site to provide interactive coverage, including live images and audio, and full commentary.

In the 1996 tournament, the Web site was originally sized for an estimated 200,000 hits, and a uniprocessor Web server was installed. During the first game, the site received 5 million hits! The Web server was unable to handle the load.

With 48 hours before the start of the second game, the IBM Web team installed an RS/6000 SP system using six nodes running as Web servers. The load was balanced among the nodes using a development version of the Interactive Network Dispatcher product discussed in this book.

As the tournament progressed, hit rates continued to climb, and three additional nodes were added to handle the extra load. The IBM Web site ensured that the matches could be watched by millions of people all over the planet.

1.7 Basic Terminology

This section explains the basic terminology used in discussing Interactive Network Dispatcher.

1.7.1 Interactive Network Dispatcher

Interactive Network Dispatcher is a scalable, highly-available load-balancing software solution for HTTP, FTP or other TCP-based servers. It balances the load of Internet servers on a variety of operating systems, such as AIX, Windows NT and Sun Solaris.

1.7.2 Hypertext Markup Language (HTML)

HTML is a language for creating the hypermedia documents that are the foundation of the World Wide Web.

1.7.3 Hypertext Transport Protocol (HTTP)

HTTP is a TCP/IP protocol used by World Wide Web servers and Web browsers to transfer hypermedia documents across the Internet.

1.7.4 World Wide Web (WWW) Server

HTTP is a TCP/IP application that allows Web browsers to transfer hypermedia documents across the Internet.

1.7.5 File Transfer Protocol (FTP) Server

FTP is a TCP/IP application that allows a user to send or retrieve files from a remote computer.

1.7.6 Domain Name System (DNS) Server

DNS is a TCP/IP application protocol that provides mapping between IP addresses and names. The names are arranged in hierarchical domains, where the top of hierarchy is the last element of the name (for example, www.ibm.com).

Chapter 2. Interactive Network Dispatcher Concepts

This chapter explains what IBM Interactive Network Dispatcher is and how it works.

2.1 Introduction

As previously seen in Chapter 1, “The Big Picture” on page 1, the World Wide Web’s rapid growth as an information and transaction medium is overwhelming the computing infrastructure. The number of users and networks connected to the global Internet is growing exponentially. This growth is causing scalability problems that can limit a user’s access to popular sites. Network administrators are using numerous methods to try to maximize these accesses. As explained, there are several methods to improve the service capability, but the most powerful and flexible solution is the Interactive Network Dispatcher implementation.

Interactive Network Dispatcher is a server load-balancing software that boosts the performance of servers by routing TCP/IP session requests to different servers, thereby (within a group of servers properly configured) balancing the clients’ requests among all servers. This routing is transparent to users and other applications such as e-mail servers, World Wide Web servers, distributed parallel database queries, and other TCP/IP applications.

2.2 Interactive Network Dispatcher Functions

Interactive Network Dispatcher consists of two functions that can be used separately or together to provide superior load-balancing results.

2.2.1 The Network Dispatcher Function

The Network Dispatcher function balances the load on servers within a local area network by using a number of weights and measurements that are dynamically set by Network Dispatcher. This function provides load balancing at a level of specific services, such as HTTP, FTP, SSL, NNTP, POP3, SMTP, and Telnet.

2.2.2 The Interactive Session Support (ISS) Function

The Interactive Session Support (ISS) function balances the load on servers within a local or wide area network by using an intelligent 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 Network Dispatcher machine.

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

Using them together

Using the Interactive Session Support (ISS) and the Network Dispatcher functions together, it is possible to balance the load on servers within both local and remote networks.

2.2.3 How Network Dispatcher Works

Network 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 shutdown 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 Network Dispatcher machine, which decides with each client request which server is the best one to accept requests, according to certain dynamically set weights. Network Dispatcher routes the clients' request to the selected server, and then the server responds directly to the client without any further involvement of Network Dispatcher. Network Dispatcher can also detect a failed server and route traffic around it.

Network Dispatcher for AIX contains a kernel extension that provides routing of Transport Control Protocol (TCP) and User Datagram Protocol (UDP) traffic requests. The Network 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 least busy and routes the packet to that server. Since all servers in the cluster have an alias for the cluster's IP address, Network 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.

This makes it possible to have a small bandwidth network for incoming traffic (like Ethernet or token ring) and a large bandwidth network for outgoing traffic (like ATM - Asynchronous Transfer Mode or FDDI - Fiber Distributed Data Interface).

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.

Network Dispatcher consists of three 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 is an AIX kernel extension and always runs when the Network Dispatcher function is being used.

manager The Manager 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 input given to the Manager by the Advisors, ISS and Executor factor has a relative proportion of importance; so you can give more importance to one input over the others, or totally ignore one or more inputs. 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, too.

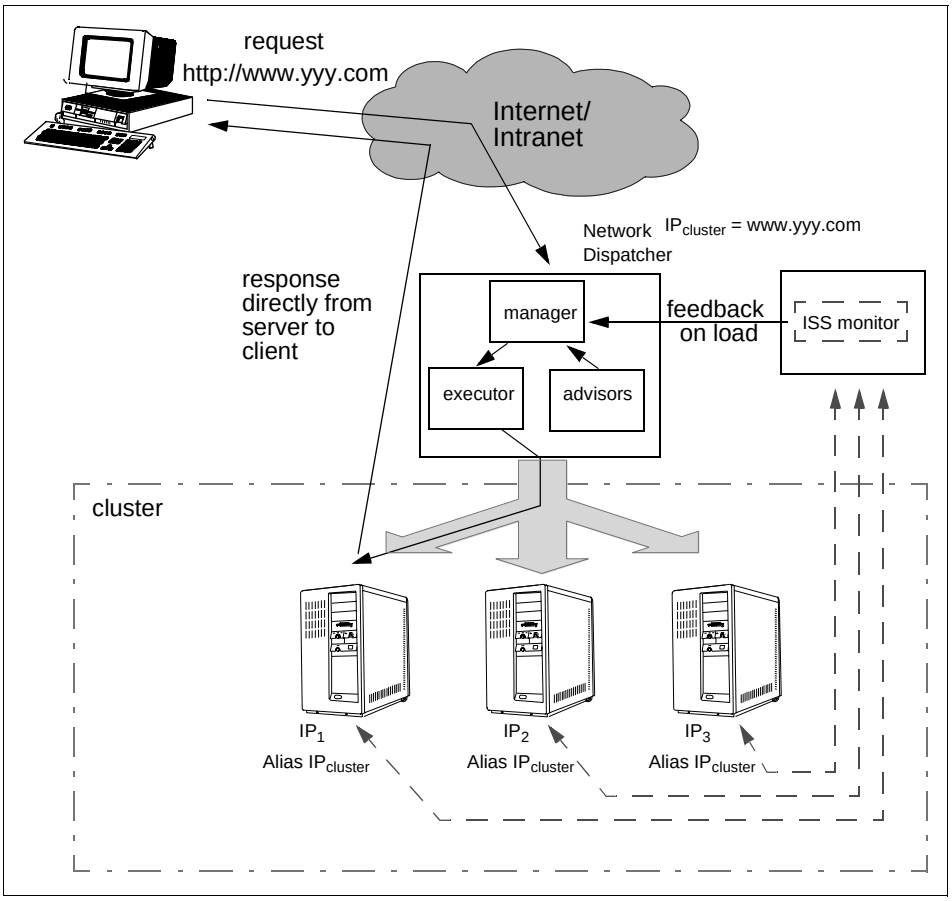


Figure 1. Network Dispatcher with ISS Providing Servers Load Information

2.2.4 How Interactive Session Support (ISS) Works

As said above, the ISS function can be used to provide server load information to a Network Dispatcher machine or to provide load-balancing by itself.

- If ISS is used to collect server load information, there is an ISS monitor that collects this information from the ISS agents that must run on the individual servers. The ISS monitor then forwards this information to the Network Dispatcher that uses this load information, along with other sources of information, to perform load balancing.
- If used for load balancing, ISS works in conjunction with a Domain Name System. This may be either an actual DNS name server or—if you set up a

small, separate subdomain for a name server—a replacement name server provided by ISS. The DNS name server daemon and the `issd` daemon running in the monitor mode can reside in different nodes. Every ISS service has its own DNS name defined in the DNS configuration files. A client submits a request referencing that DNS name. Then ISS resolves that name to the IP address of the selected server of the cell and routes this IP address to the client.

ISS periodically monitors the level of activity on a group of servers and detects which server is the least heavily loaded. The administrator must decide the criteria to be used to measure the load on the servers, depending on the environment being monitored. After every monitoring period, ISS checks the DNS configuration files to make sure that the mapping between the DNS name of the ISS service and the corresponding IP address matches the IP address of the selected server. It can also detect a failed server and route traffic around it.

The previous release of ISS introduced the concept of a *pool*, which was defined as a group of machines performing the same function. An ISS *cell* is defined as one that contains groups of servers, and each group performs one function called a *service*, which is similar to a *pool*. A server—or *node*—can run a single service (HTTP, FTP or Telnet) or many services. Moreover, *pools* of servers running the same protocol can differentiate each other for the criteria chosen to balance the load among the servers of the *service*.

For end users, a *service* is a single computational resource representing a single virtual server, and ISS balances the workload across the various systems in the service to provide optimum productivity.

The *service* is a group of servers that are equally suitable for a particular purpose. A service is identified by a hostname. A user that wants to access a service uses this hostname to do so. All servers providing one service must be indistinguishable from the user's point of view. This means they must offer the same data to the user, and if applicable, the user must be able to log in using a consistent user name and password. This can be accomplished using a distributed file system such as AFS, NFS/NIS or DFS/DCE or by using a shared disk architecture. ISS allows you to have multiple services for one cell, and a server can be a member of a number of different services, as shown in Figure 2 on page 14.

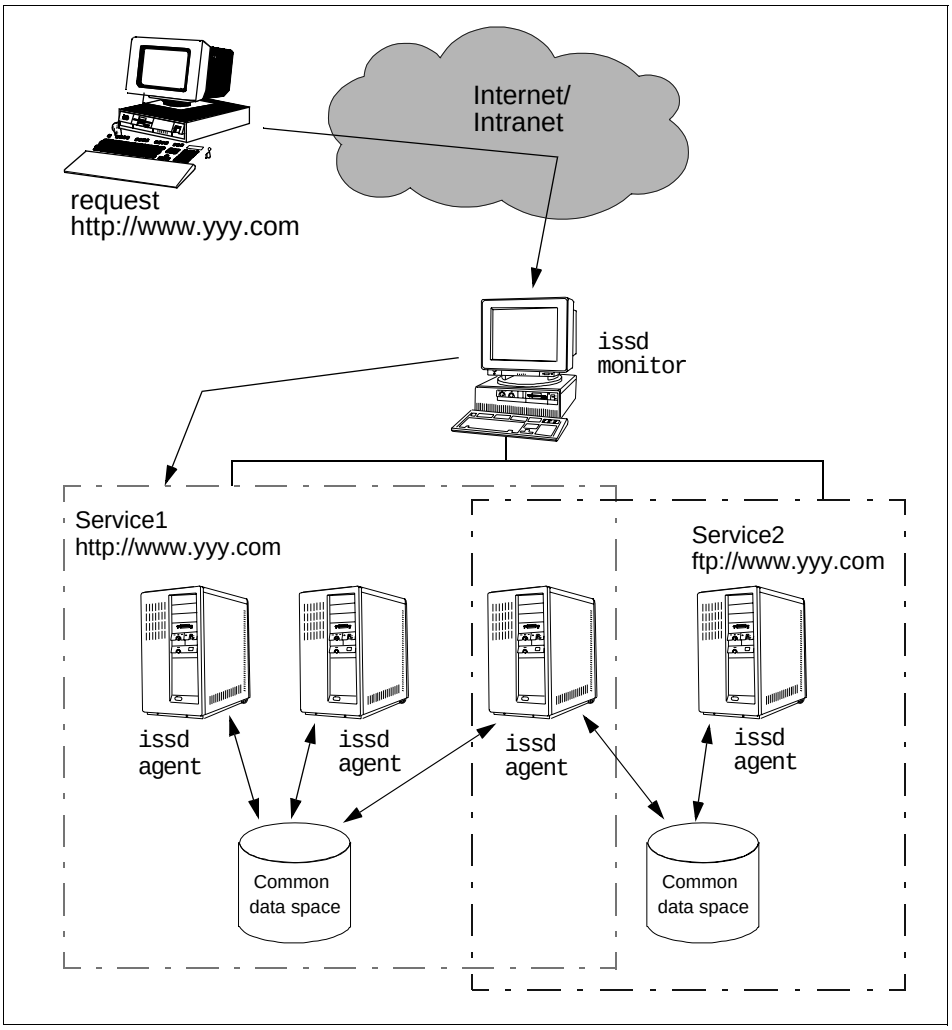


Figure 2. Multiple Services Appear as One Virtual Server

One of the main advantages of ISS is that it offers finer granularity for load balancing than many other solutions. Interactive sessions can be balanced according to the site requirements.

ISS can be used for load balancing, as the figure above shows, or to collect server load information.

In the first case, the ISS works in conjunction with a TCP/IP domain name server to manage the workload. When a user sends a request to a service in

one ISS cell, it must first resolve the hostname of the service to an IP address. This name resolution is performed either by the DNS `named` daemon or by the ISS software, and the IP address of the most appropriate server is returned.

In the second case, ISS works to provide load server information to a Network Dispatcher machine, which does not affect name resolution. This solution gives greater flexibility in the possible configurations that can be achieved, especially in environments such as large Web sites, which are characterized by a large number of clients submitting a high volume of interactive requests.

Chapter 3. Interactive Network Dispatcher in Detail

This chapter explores Interactive Network Dispatcher and explains the parameters that can be used to customize the configuration files to make them appropriate for your own needs. There are many parameters that can help you to create the most appropriate configuration to guarantee an accurate distribution of the load over your servers.

The Network Dispatcher functions are examined first; then we look at the Interactive Session Support (ISS) functions.

3.1 The Network Dispatcher Function in Detail

This section focuses on the Network Dispatcher function and its components.

3.1.1 Interaction between Network Dispatcher Components

As explained previously, Network Dispatcher consists of three components: Executor, Manager and Advisors—the last two are optional. First, let's see how they interact.

The typical flow of information used in Network Dispatcher is shown in Figure 3.

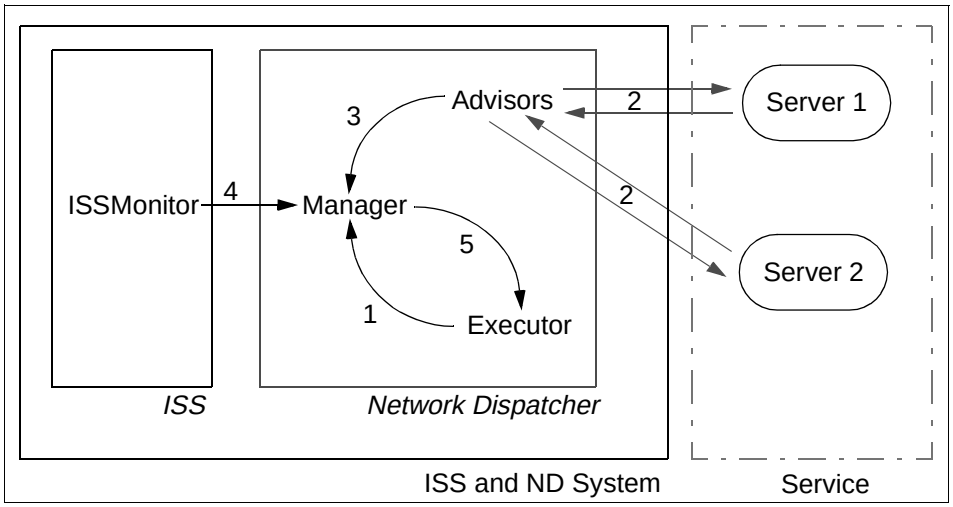


Figure 3. Typical Flow of Information in Network Dispatcher

In Figure 3, we are assuming that Network Dispatcher is using Advisors and ISS to collect servers' load information, and therefore, the Manager is running to receive this information.

As you can see, Advisors are a key component of Network Dispatcher. They are responsible for monitoring each back-end server, determining its availability and response time. After the information is gathered and processed, it is forwarded to the Manager to be used as one factor for the calculation of the least-loaded server.

Each Advisor is specific for a service. Interactive Network Dispatcher 1.2 for AIX comes with seven Advisors: HTTP, FTP, SSL, SMTP, NNTP, POP3, and Telnet.

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

A summary of the functions of each component is found in Table 1.

Table 1. Summary of Functions for Network Dispatcher Components

Component	Function	Sends Information to:
Executor	Routes TCP and UDP requests based on weights. Monitors number of active, new and finished connections. Performs garbage collection of RESET connections.	Manager
Manager	Collects information from internal counters of the Executor, ISS Dispatcher Observer and sent by Advisors. Performs load balancing decisions based on collected information. Updates weights used by the Executor.	Executor

Component	Function	Sends Information to:
Advisors	Gather information about availability and response time for each service in each port for servers.	Manager
ISS component (if present) set up as a Dispatcher Observer	Collects information about server load based on specific metrics.	Manager

3.1.2 Controlling Network Dispatcher Components

Network Dispatcher handles requests from the command line to the Executor, Managers and Advisors. The first step that must be done, after installing and configuring the Network Dispatcher, is to run it using the following command:

```
ndserver start
```

A command line interface is provided by `ndcontrol` to configure and manage the Executor, Advisors and Manager.

The Executor is the main component; it controls a set of clusters. You can set the maximum number of clusters, the maximum number of ports per cluster, the maximum number of servers per port, and so on.

For example, to set the maximum number of clusters, type:

```
ndcontrol executor set maxcluster 4096
```

to start and stop the Executor, type:

```
ndcontrol executor [ start | stop ]
```

To configure the Advisors, the `ndcontrol` command line interface must be used again.

For example, to start and stop an Advisor, use the following command:

`ndcontrol advisor [start | stop]` *service port*

where *service* indicates the name of the service to be monitored (HTTP, SSL, FTP, SMTP, POP3, NNTP, or Telnet), and *port* indicates the port number to be monitored for that service. Advisors can monitor several port numbers for the

same service name—it is necessary to issue just one `ndcontrol advisor start` command for each port to be serviced.

Observation: To start an Advisor, it is necessary for the Manager to be already running. This can be done using:

```
ndcontrol manager start
```

From time to time, each Advisor asks for a status from the servers on the monitored ports and sends the results to the Manager. This interval of time is configurable. Its default value is seven seconds. To change this value, use the following command:

`ndcontrol advisor interval` *service port interval_time*

The *`interval_time`* must not be smaller than the Manager interval time because if it is, some information delivered by the Advisor to the Manager will not be used. This value must be carefully chosen because the Advisors actually make a request for information to the servers to measure the time it takes for the request to be completed.

In order to avoid the use of out-of-date information supplied by the Advisor, the Manager will not use any information that is older than the time set in the Advisor time-out. The Advisor time-out should be greater than the Advisor pooling time to prevent the information from being discarded before being used by the Manager. To set the time-out period for an Advisor, type:

`ndcontrol advisor timeout` *service port timeout_period*

where *`timeout_period`* is either a number of seconds for the time-out period or the word `unlimited`, which means that the information supplied by the Advisor does not expire. The default value is unlimited. If you set up a specific time-out and an Advisor sends server failure information to the Manager, Network Dispatcher will use this information even if the Advisor information has timed out.

When started, Advisors create a log file to keep a log of their actions. There is a log file for each Advisor—that is, for each service and each port number. The name for the log file is: /usr/lpp/ind/nd/logs/advisor_service_port.log. For example, for an HTTP Advisor using port 8080, the file name would be /usr/lpp/ind/nd/logs/http_8080.log. If there is a file with the same name of the log file, it will be overwritten. The name of the log file is set when the Advisor is started. To use a non-default log file name, the following command must be used to start the Advisor:

`ndcontrol advisor start` *service port logfilename*

where *logfilename* is only the name of the new log file. To change the directory where the logfile is located, the /usr/bin/ndserver file must be edited.

The maximum size of the log file and a log level can also be set. The maximum size of the log file sets the greatest size the log file may grow to. If the file size reaches this size, the next entries of the log are written on the top of the file (in a round-robin fashion). The log level says how much information is saved on the log file. It is a number from 0 to 5, where 0 means only errors are logged, and 5 means everything is logged for debug purposes. To change the loglevel:

`ndcontrol advisor loglevel` *service port level*

where *level* is the log level desired. The default log level is 1.

To change the maximum log file size:

`ndcontrol advisor logsize` *service port logsize*

where *logsize* is either the maximum size of the log file in bytes or the word `unlimited`, which means that there is no maximum log file size. The default is unlimited.

When using Advisors and/or ISS, there is another important parameter that can be changed, the `smoothing` value, which decides how smooth the change of the servers' weights will be. The default is 1.5, but a higher value means that the Manager will change the weights less drastically limiting the amount that a server's weight can change. To change the value enter:

`ndcontrol manager smoothing value`.

For more information on all the possible options of the `ndcontrol` command, see "Interactive Network Dispatcher User's Guide" on page 39.

3.1.3 Proportions

As was shown in "Interaction between Network Dispatcher Components" on page 17, the factors that are used in the calculation of the least loaded server in a pool are:

- Number of active connections
- Number of new connections
- Information supplied by Advisors

- Information on load supplied by system monitoring tools, such as the ISS component

The way to establish how important each of the factors are in the load-balancing process—that is, how the Manager uses the information given to it —is by using *proportions of importance*. Each of these factors is attributed a number, from 0 to 100, that can been seen as a percentage. Zero means that the factor is not used. One hundred means that the factor is the only one used. The sum of all proportions must always be 100.

To set the proportions, use the following command:

`ndcontrol manager proportions` *active new advisor system*

where *`active`* means the proportion of weight to be given to the active connections, *`new`* to the new connections, *`advisor`* to the information provided by the Advisors, and *`system`* is the proportion of weight to be given to the information provided by system-monitoring tools such as ISS. The sum of *active+new+advisor+system* must always be 100. The default proportions are 50 50 0 0.

To set all the factors to be equally important, set the proportions to 25 25 25 25.

Since Advisors provide accurate information on how loaded a specific server is for one service and one port, it is suggested that its proportion be set to a number different than zero.

When there is no Advisor for a specific server (Telnet, for example), the proportion given to the Advisor input is ignored. So, for example, if the proportions are set to 33 33 34 0 for a Telnet port, the proportions that will be effective are 50 50 0 0.

3.1.4 Ports Used by Network Dispatcher

Network Dispatcher needs to use some ports to perform its tasks. By default, it uses certain port numbers. Sometimes, it is necessary to use port numbers other than the default (as for example when the Network Dispatcher server is an RS/6000 SP node).

If you need to change the port used by `ndserver`, which is 10003 by default, you have to modify the ND_PORT value in the usr/bin/ndserver and /usr/bin/ndcontrol files.

Moreover, the default port used by the Manager to receive load information by ISS is 10004. To change it, you have to enter the following command in the Network Dispatcher machine when you start the Manager:

`ndcontrol manager start` *logfile metric_port*

If you specify a *`metric_port`*, you must specify a *`logfile`* name.

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*

3.2 Interactive Session Support (ISS) Function in Detail

This section is about ISS and its concepts.

3.2.1 ISS Cells

An ISS *cell* is a group of nodes administered as a single, logical unit.

A *service* is a group of servers that perform the same function. Each server represents a *node* and can belong to one or more services (HTTP, FTP, Telnet). When configuring ISS and editing the configuration file, you must first declare all the nodes in your site that will be part of the ISS cell; see Chapter 10, “Configuration Files - Samples” on page 93.

When you declare a node, you can use the `Node` or the `Node NotMonitor` keyword. In this release of the Interactive Network Dispatcher, you can configure one node running the `issd` process in monitor mode and one or more others as backups of that node, defining a different priority number for each one. In a backup node, `issd` usually runs in agent mode, and the node can only be a server that provides a service. Only if the monitor node fails will the first backup node in the list take over, and the `issd` run in monitor mode. As soon as the previous node is available again, it will become the monitor because of its higher priority number.

Next, you declare the services that will be provided and their parameters. These parameters include the `NodeList` that comprises the nodes that will provide the service and the `ResourceList` that specifies which resource will be used to select the right server in the cluster.

A resource may be a CPU, a disk, a process, and so forth. A single resource may be used repeatedly; so it is defined in the file after listing the nodes and before the services definitions. The `ResourceType` parameter is used with its

own keywords: Metric, MetricNormalization, MetricLimits, and Policy. We will speak more about these parameters later in this chapter.

After configuring nodes and services, you should configure the observers. For a description of this concept, see “ISS Observers” on page 24.

3.2.2 ISS Observers

The ISS Observer uses information about the status of nodes in the cell. It provides three options for load balancing. Depending on the requirements of your particular environment, you may choose one or more of these options. Each option corresponds to one type of ISS Observer. Their names are the same as the keywords you must use to define in the ISS configuration file:

```
NameServer

ISSNameserver

Dispatcher
```

You can use one or more ISS Observers in the same cell.

3.2.2.1 NameServer

When using NameServer type, the ISS Observer works in conjunction with a DNS name server. ISS calls the name server to map DNS names of ISS services to IP addresses of the most appropriate server. The issd running in monitor mode can run in a different machine from the named daemon.

In this way, you can configure one or more servers of your cell as backup issd monitor nodes. You can define a different priority number for every node of the cell you decide to use as a backup issd node. When the first one fails, the next on the priority list takes over. Until then, the issd runs only in agent mode.

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. Of course, the criteria of selecting the best server for each service depends on how you have configured ISS and which SelectionMethod keyword you have decided to use.

There are two possible selection methods: RoundRobin and Best. The second one chooses the best server of a cluster depending on the metric you have previously defined in the ResourceType sections of the ISS configuration file.

To use this method, you must first configure a DNS server or have one already configured. You can add the servers to a domain that already exists or create a new subdomain. ISS will then update the files that DNS uses for name resolution and reverse name resolution. Figure 4 on page 25 shows ISS operating with a `NameServer` Observer. In this figure, the `issd` monitor works as a `NameServer` Observer, too. If ISS is configured to use the `Best SelectionMethod`, each `issd` running as agent provides server load information to the `issd` monitor.

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.

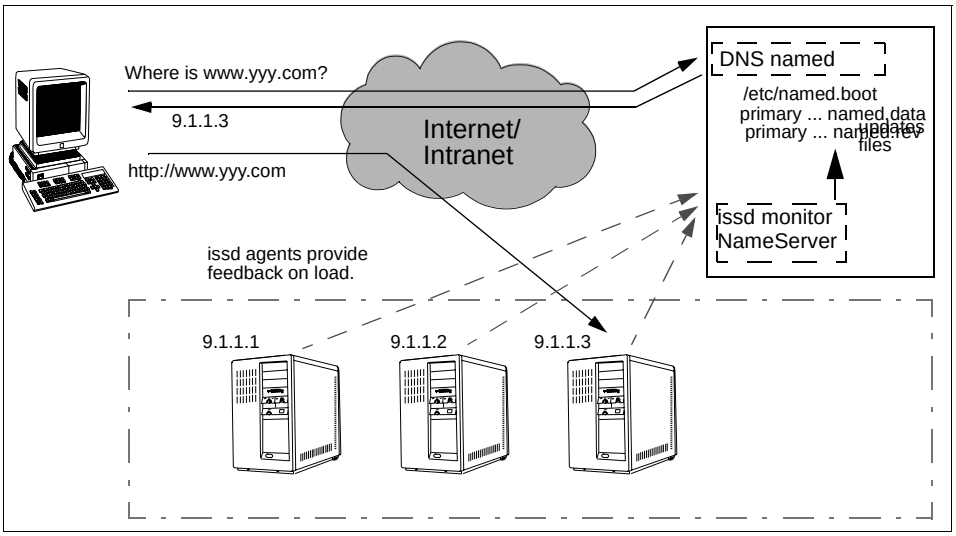


Figure 4. ISS Configured with a NameServer Observer

3.2.2.2 ISSNameserver

When using `ISSNameserver` type, the ISS Observer works as a DNS name server. In this case, ISS uses the DNS configuration files of the DNS name server, but the `named` daemon is not running. This method allows ISS to provide balancing on a per-request basis without the need to continually refresh the DNS `named`. The ISS software replaces the `named` daemon and

performs the name serving function by providing minimal name server implementation.

Since using NameServer Observer involves some load on the DNS server machine, a good option is to use ISSNameserver Observer with a new subdomain for your pool of servers.

The ISSNameserver Observer will provide forward resolution (from name to IP address), but not reverse resolution (from IP address to name). As for the NameServer Observer, you can define one or more machine to be an issd backup for the one that is currently running in monitor mode.

The selection methods available to configure load balancing are the same as with the NameServer Observer: RoundRobin and Best.

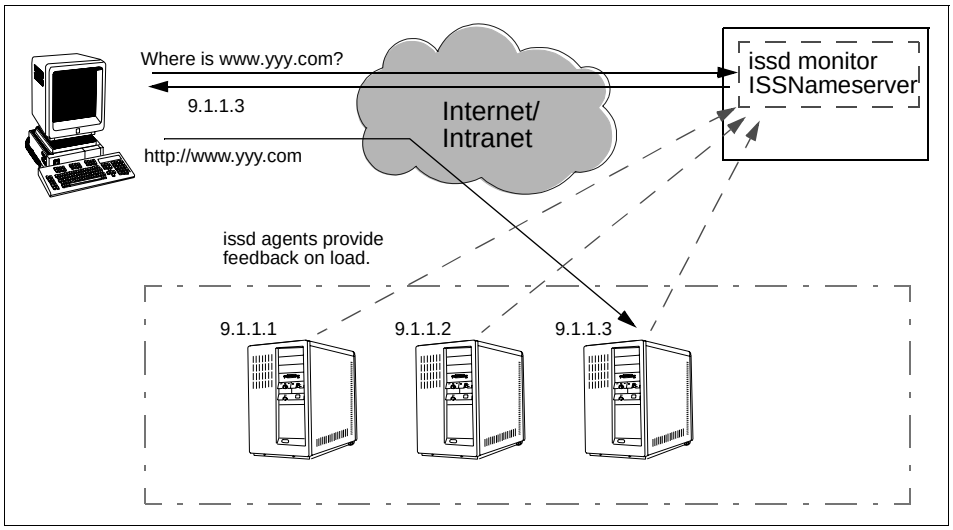


Figure 5. ISS Configured with an ISSNameserver Observer

3.2.2.3 Dispatcher

When using Dispatcher type, the ISS Observer works in conjunction with Network Dispatcher to perform the load-balancing function. Neither a DNS name server nor an ISS Observer working as DNS is required.

The previous approaches (NameServer and ISSNameserver) are the simplest to use to balance the workload, but they have their disadvantages. Most applications and browsers do caching of hostnames to efficiently handle subsequent requests. This may lead to unbalanced requests to the servers. Dispatcher Observer can be used to solve this problem giving more

granularity when routing requests to a server. `Issd` running in monitor mode is only used to provide load feedback from the `issd` agents running on each server to the Network Dispatcher. Only Network Dispatcher provides load balancing. Network Dispatcher allows you to link many individual servers into large, virtual servers. This configuration is called a *cluster*. All requests are sent to the single IP address of the Network Dispatcher, which then routes the requests to one of the servers. Naturally, Network Dispatcher uses the load information from ISS along with all its other sources of information.

You can have one or more machines defined as backup `issd` in monitor mode.

As with previous Observers, you can choose between the `RoundRobin` and `Best` selection methods.

3.2.3 ISS Selection Methods

Each service maintains a ranking of the best nodes to perform that service. You can set a particular selection method using the `SelectionMethod` keyword in the ISS configuration file. This keyword may have one of the following values:

- `RoundRobin`

  The load among servers is distributed on a round-robin basis. For example, if you have three servers providing a service (A, B and C), the first request will be sent to server A; the second will be sent to B, and the third to C. The fourth request will again be sent to A, the fifth to B, and so on. In this method, the load on each of the servers is ignored. The round-robin metric can be used for services in which all requests are similar in frequency and duration. The main advantage of this method is that the servers do not have any monitoring overhead. The main disadvantage is that this metric does not notice when the servers are unevenly loaded. But a server will not be recommended if it appears to be down. Read more about round-robin configuration in “RoundRobin” on page 32.

- `Best`

  The `issd` monitor calculates the best server for every defined service once each update interval based on the type of measurement you have decided to use to measure the load on the servers. The load metric is defined using the `ResourceType` keyword. See “ISS Configuration File” on page 28, for how to define your load metrics.

For every provided service you have, you define how to measure the load on the servers. Groups of servers providing the same service can be balanced

with different metrics. For example, A round-robin service can be configured and additionally, can provides a number of services that use a custom configuration.

3.3 ISS Configuration File

The ISS configuration file is used to specify all options for configuring the ISS component. When you start the `issd`, give the name of the configuration file as the first parameter. Use the following command as the root user to start ISS:

```
issd -c filename
```

Sample configuration files are provided in the directory /usr/lpp/ind/iss.

The ISS configuration file is an ASCII file containing a series of definitions. Each definition must be at the start of a line and must start with a keyword which is followed by possible further keywords and then a value. In this section, we discuss some of these configuration keywords. For a detailed description of the ISS configuration file, and all possible keywords, refer to the manual described on “Interactive Network Dispatcher User’s Guide” on page 39.

When you configure the ISS file, you must first define the cell and its attributes, then list the nodes that will be part of the cell, the resource types of the cell with their parameters, the services to be provided, and the Observers you will work with.

3.3.1 Cell and Its Attributes

These options control the overall behavior of the ISS monitor and should be set once per configuration file.The `cell` keyword specifies the symbolic name of the ISS cell. It must appear at the top of the file. You must specify the name of the cell. You can have *local* or *global* cells. You need to define one and only one local cell per ISS configuration file. A local cell is the one that the node will be a member of. You define a global cell as a separate, remote cell that you want your node to communicate with to implement *ping triangulation*. For more related information about ping triangulation, see the manual described in “Interactive Network Dispatcher User’s Guide” on page 39.

The syntax is:

`Cell` *Name* [`Global` | `Local`]

Then you have to specify the following cell's attributes:

- **Authkey**

  This key is used to provide an 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`, which has to be specified after the node is defined.

- **LogLevel**

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

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

- **PortNumber**

  This option specifies the port used for ISS internal traffic. Use the syntax:

  ```
  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]
  ```

  The default is 10 seconds.

- **HeartbeatsPerUpdate**

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

  - How frequently the load measurement metric will be checked to determine if the ranking has changed
  - How frequently the `named` daemon will be updated when the `NameServer` Observer is running
  - How frequently load profiles will be sent to Network Dispatcher if the `Dispatcher` Observer is running

  The syntax of this keyword is:

`HeartbeatsPerUpdate count`

Where `count` is a non-negative integer.

The update interval is calculated as:

Update interval = `HeartbeatInterval` * `HeartbeatsPerUpdate`

Note that the `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. If `NameServer` Observer has been configured, the name server will be updated immediately.

3.3.2 Node

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*

The *priority* field is a positive integer that can mean two things. In fact, in this release of ISS, you can decide to have a node running the `issd` in monitor mode, but you can set up other nodes that will take over if the monitor fails. The priority number establishes the order that will be followed in taking over the monitor. If you use only the `Node` keyword followed by the name of the server, you are declaring that machine can run as `an issd` monitor based on its priority number. If you don't want that machine to run as an `issd` monitor, you have to declare this using the `NotMonitor` keyword. The syntax is:

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

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

There is another keyword referred to as a specific node, that is the `InternalNet` keyword. If your machine has more than one network adapter and you want to define an internal network for ISS network traffic among the nodes of the cell, you can do it using this keyword. The syntax is:

`Node` *[hostname1 | IP address1] priority* `InternalNet` *[hostname2 | IP address2*]

ISS will use the `IP address2` to make the `issd` communicate with each other, but it will use the `IP address1` to provide the services to the clients.

Node definition

The hostname of an AIX machine must be set up. It can be done by using smitty. ISS needs to recognize this hostname or the corresponding IP address. If your machine has more than one network interface (each one with its own DNS name and corresponding IP address) and you do not want to use the one corresponding to your main hostname to provide the service, you should either change your hostname to the one corresponding to the network interface you want to use or maybe define an internal net.

We worked with SP nodes with their hostnames corresponding to the Ethernet interfaces in the DNS configuration. But we used the token-ring interfaces to provide ISS services. Therefore, to make ISS recognize the machine in which it was running we needed to configure an Ethernet internal network.

3.3.3 ResourceType

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. You can have many different resource types in the ISS configuration file and decide to use one or more of them for each service depending on your own needs. The syntax is:

```
ResourceTypes name1 name2
```

Every resource type is characterized by several parameters that define the metric that will be used to measure the load among the servers. The metric keywords are `Policy`, `MetricNormalization` and `MetricLimits`.

- **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. We'll speak more about metrics later in this chapter.

3.3.4 Service

The `Service` keyword must be followed by the symbolic name of the service you want to provide and the service DNS name if you are only using ISS. The cluster IP address and the port number are also used if you are using ISS in conjunction with Network Dispatcher. The syntax is:

```
Service name DNS name [ cluster address ] [ port number ]
```

After defining the service, you need to use the `NodeList` keyword, which must be followed by the hostnames or IP addresses of the servers that will provide the service. Then you declare which previously defined `ResourceTypes` ISS will use for selecting the top-ranking server by listing these `ResourceTypes` after the `ResourceList` keyword.

For each service, you can specify an `Overflow` node that will take over the service if none of the servers in the `NodeList` is available. The syntax is:

```
Overflow  DNS name.
```

3.3.5 Observers

This keyword defines which type of Observer you are using: `NameServer`, `ISSNameserver` and/or `Dispatcher`.

3.4 Metrics

ISS balances the workload over a group of servers providing one or more services. It allows the system administrator to choose one or more measurement metrics for each service. A metric defines how ISS will measure the load on the servers 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.

3.4.1 RoundRobin

When the `SelectionMethod` is set to `RoundRobin`, ISS does not measure the load on the servers that belong to a certain service. It simply cycles through the available servers in the order they appear on the keyword `NodeList`. The update interval indicates the period of time during which a server will get all incoming requests before switching to the next server on the list. This is the simplest measurement type available. Its main advantage is that it does not put any overhead on the servers since they do not have to provide any feedback on load.

This measurement algorithm can be used with any ISS Observer: `NameServer`, `ISSNameserver` or `Dispatcher`. Figure 6 shows this configuration. In this

example, each server will rotate every 20 seconds, and the ISS will check the availability of the servers every 10 seconds.

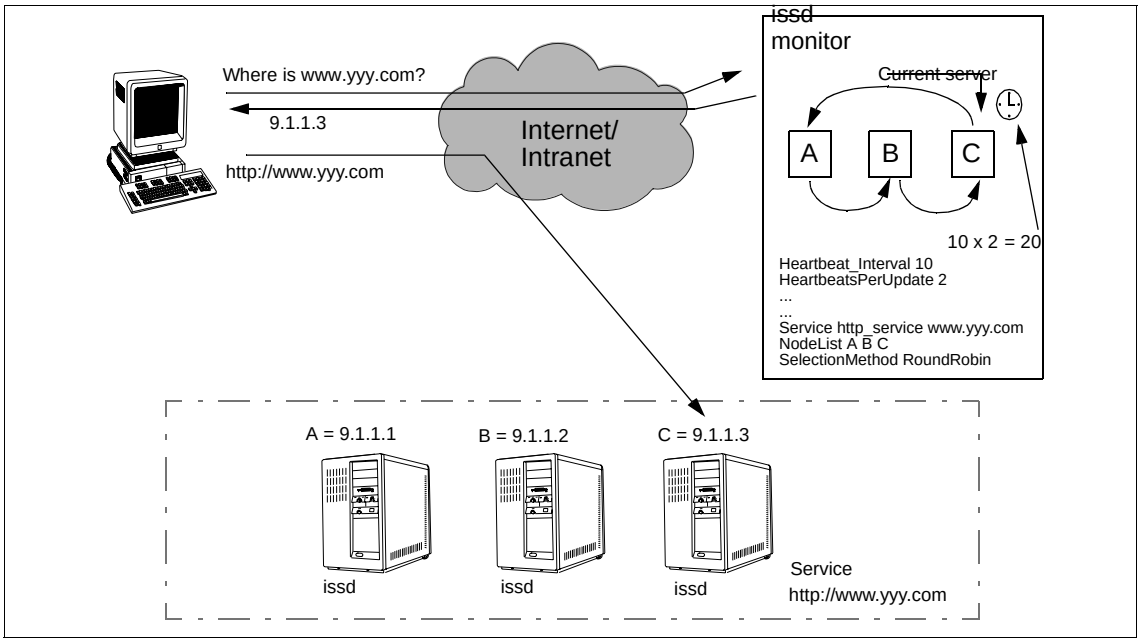


Figure 6. ISS working with a RoundRobin SelectionMethod

This metric would be useful when the access pattern of your clients to your service follows an uniform profile in terms of frequency and duration of access. In this configuration, ISS will never notice if a particular server is getting very busy and will continue sending requests to this server. This may lead to an unbalanced load on your servers.

Be aware that even if the `RoundRobin SelectionMethod` does not calculate any load on the servers, the `issd` must run on every node of the service.

Round-Robin Configuration

To make a round-robin configuration work, you should define in the ISS file at least one `ResourceType` and include it in the `Service` that will use the `RoundRobin SelectionMethod`. The `issd` monitor will not use that `ResourceType`, but it needs it to completely identify the `Service`. For example:

```
Service www_service  www.yyy.com
NodeList  server1 server2  server3
ResourceList CPULoad
SelectionMethod  RoundRobin
```

3.4.2 Internal Metric

The syntax of this option in the configuration file is:

`Metric Internal` *string*

If you specify the `Internal` metric, you can use a method provided directly by the system and the *string* field may be one of the following:

- `CpuLoad`, which measures the percentage of CPU is being utilized
- `FreeMem`, which returns the amount of free physical memory as a percentage

  `LoadLeveler`, which must work in conjunction with an existing and running LoadLeveler configuration

  LoadLeveler is a job-management system that allows users to run more jobs in less time by matching their processing needs to available resources. Loadleveler schedules and manages jobs that you submit to one or more machines under its control. It accepts the jobs and reviews the job requirements and compares them with the resources of the machines under its control to determine which is the best machine to run the job. For more related information about LoadLeveler and how to make ISS work with it, see:

  Using and Administering LoadLeveler, SC23-3989

  Also see the User's Guide described in "Interactive Network Dispatcher User's Guide" on page 39.

The `SelectionMethod` is `Best`.

3.4.3 External Metric

When you decide to use an `External` metric, you define how to measure the load on your servers. This metric is anything that can be executed on the servers and returns a numeric value as a result.

The syntax is:

```
Metric External  string
```

The `string` field defines the command or program that when executed returns a measure of the load on the server. For example, you can run a metric based on the number of processes running on the server by using the following entry:

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

Using a TCP/IP connection, the `issd` running in monitor mode communicates with the `issd` running in agent mode on each server. The port used for this connection is defined by the `PortNumber` keyword at the top of the configuration file. The `issd` agent will run the command provided by the `issd` running in monitor mode and will report back the result and any change in the load.

The `Policy` keyword indicates whether the metric function specified is to minimized or maximized—a high value indicates high load or low load. The syntax is:

```
Policy  [ Max | Min ]
```

This keyword has to be set to correspond with the function you use. In our previous example, we were counting the number of processes; so we had to favor the smallest values, and the policy had to be set to `Min`. But if you use something like:

```
Metric External vmstat 1 2 | awk ' NR == 5 { print $16 } '
```

which returns the percentage of CPU idle time on the machine, you must use a `Max` policy.

The `MetricLimits` keyword defines the limits for the server to continue to provide the service. There is a maximum value that defines the level beyond which the metric should not go, and a minimum value that defines the level down to which the metric should be taken before the server is returned to ranking. In the previous example, based on the number of processes running on the machines, you can decide that a server should provide the service

until there are not more than 250 processes running. If there are more than 250 processes, it will be excluded from the service until the number of the processes goes down to less than 200.

You can obtain this with the syntax:

```
MetricLimits 200 250.
```

Be aware that in this case we have the `Policy` keyword set to `Min`. There will be an inversion in the meaning of the values if the `Policy` has been set to `Max`.

Figure 7 shows an example of ISS running an `External` metric to provide a service. It has been defined as an overflow node, too.

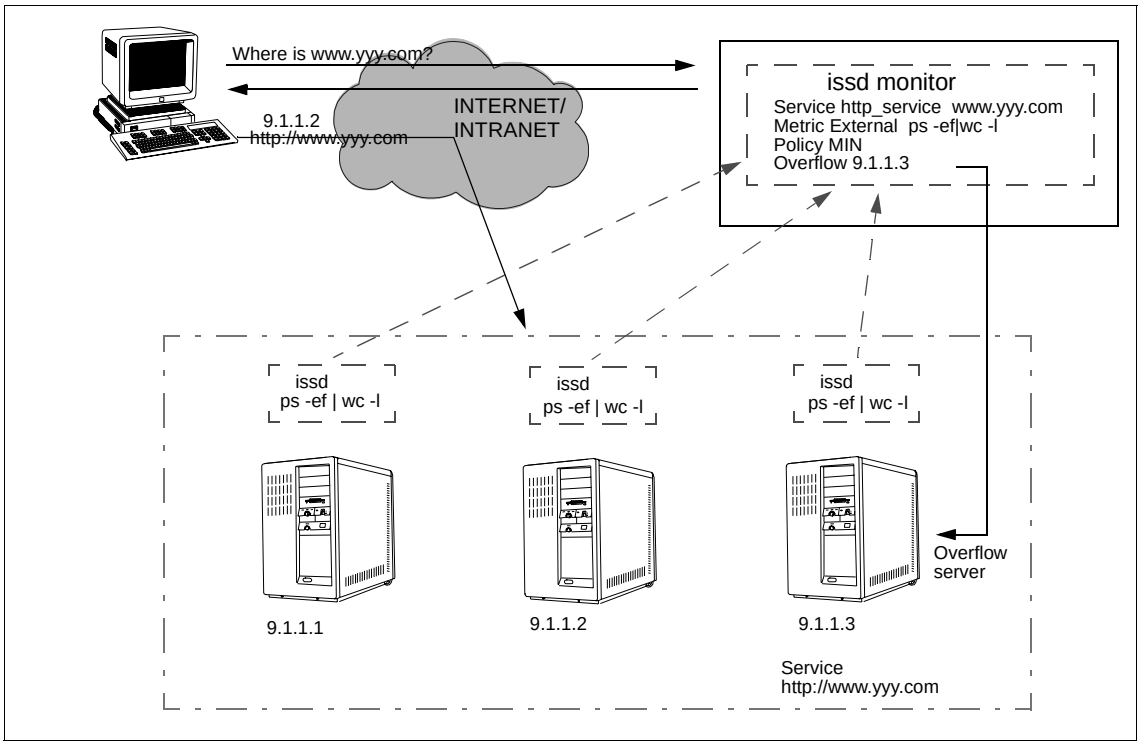


Figure 7. ISS Using an External Metric

3.4.4 Writing Your Own Metrics

The metrics you use with the `Metric External` option can be a mix of different parameters on the servers. You can make the measurement of the load on the server as complex as you need. However, if the metric function you are

planning to use is complex, it is recommended that you to encode it in a shell script or an executable program and make it available to all the servers of the service.

The resources most frequently used to determine the load on the servers are CPU, memory, paging space, I/O, and network usage. To measure CPU, you can use the output of commands like `vmstat` or `sar`. When using `vmstat`, the first line of the report contains statistics for the time since system start-up. Subsequent lines contain statistics collected during the interval since the previous report. For example, you can use the following metric to obtain the percentage of CPU usage:

```
Metric External vmstat 1 2 | awk ' NR == 5 { print $14 + $15 } '
```

This will add the 14 and 15 parameters of `vmstat` which are the CPU percentage for system and user.

Some suggestions of what can be used to obtain custom metric values, either using command line/shell scripts or using executable programs, are shown in Table 2.

Table 2. Possible Sources for External Metric Values

Item to be measured	Command Line	Program

CPU usage	`vmstat`
	`sar`
	`ps`
		special device /dev/kmem
		PTX API

Memory Usage	`vmstat`
	`sar`
	`svmon`
		PTX API

Disk activity	`iostat` `sar`	PTX API
Network traffic	`netstat`	PTX API
NFS performance	`nfsstat`	PTX API

Chapter 4. Installation and Operation

For your convenience, this chapter presents condensed guidelines based on the information found in the manual mentioned in 4.1, “Interactive Network Dispatcher User’s Guide” on page 39.

4.1 Interactive Network Dispatcher User’s Guide

The *Interactive Network Dispatcher User’s Guide*, GC31-8496, explains how to plan, install, configure, use, and troubleshoot the IBM Interactive Network Dispatcher for AIX, Microsoft Windows NT and Sun Solaris.

The most current version of this book is available in HTML and PDF formats on the Interactive Network Dispatcher Website. To access the on-line book, go to the following URL:

`http://www.ics.raleigh.ibm.com/netdispatch/downloads.htm`

4.2 Installing for AIX Installing for AIX

Before you install Interactive Network Dispatcher, you should have a design of the network as well the installation plan. The *Interactive Network Dispatcher User’s Guide* gives a broad idea on how to design your network based on your existing environment.

The hardware and software requirements are:

- A IBM RS/6000-based machine
- IBM AIX Version 4.1, modification level 5 or later, or AIX Version 4.2, modification level 1 or later
- Approximately 15 MB of available disk space for installation
- A CD-ROM drive if installing from CD-ROM

There are two options for installing this product. One is having your own CD, and the other is downloading from the Internet (if it is available). If you are downloading an evaluation copy of the product from the Web site, use the installation instructions at the Web site, whose URL is:.

`http://www.ics.raleigh.ibm.com/netdispatch/downloads.htm`

Interactive Network Dispatcher consists of two components to be installed.

- intnd.iss contains the ISS component.
- intnd.nd contains the Network Dispatcher component.

After receiving the images of Interactive Network Dispatcher, log in as *root*. Both, the command line or SMIT can be used. The following sequence presents the installation performed by SMIT.

1. Enter:

   ```
   smitty install_latest
   ```

2. From the install Software at latest Level screen:
   - If you purchased the product and have it on distribution media, press **F4** for the input device/ directory for software option and select the appropriate device.
   - If you downloaded the product, enter the name of the directory in which you extracted the packages.
3. From the Install Software Products at latest Level updated screen, press **F4** for the Software to install option.
4. When the product is successfully installed, you will see `Command: OK` in the COMMAND STATUS screen.
5. Press **F10** to exit SMIT.

Now you need to verify whether the product is installed on your machine by using the command below.

```
lslpp -h | grep intnd
```

If you have installed the entire product, this command returns the following:

```
intnd.iss.rte
intnd.msg.en_US.iss
intnd.msg.en_US.nd
intnd.nd.rte
intnd.ps.en_US
```

4.3 Configuring Network Dispatcher

This section explains briefly how to configure the Network Dispatcher function. For more conceptual information, see Chapter 2, “Interactive Network Dispatcher Concepts” on page 9, and Chapter 3, “Interactive

Network Dispatcher in Detail” on page 17. Before proceeding with the configuration, make sure the Network Dispatcher machine and all the Internet machines are properly connected to the network.

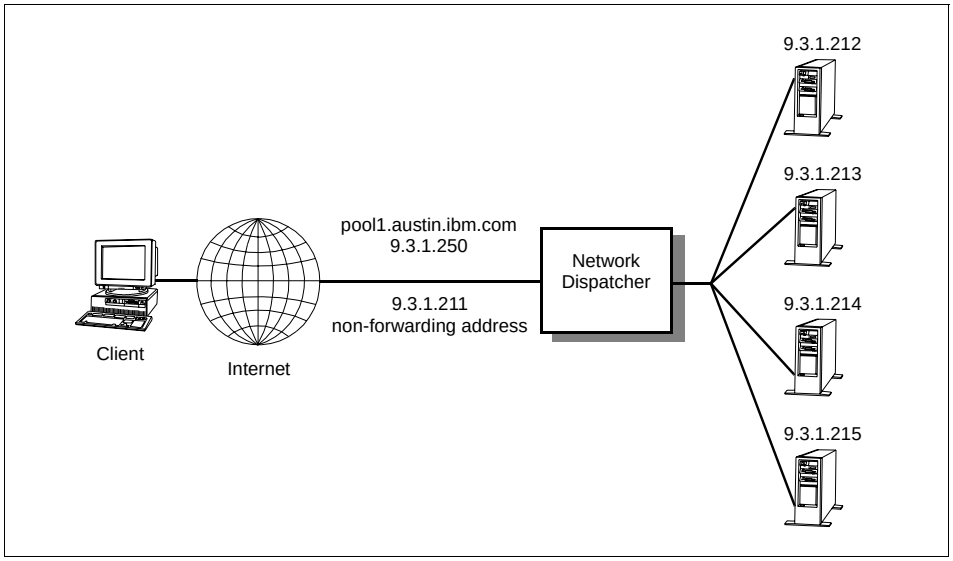


Figure 8. Sample Network with Network Dispatcher and a Group of Servers

Remember

1. You need to set up the network interface every time you start the Network Dispatcher. So before entering the `ndserver` command, check your network interface status by using this command.

   ```
   netstat -ni
   ```

   If you cannot find your network interface alias, you should set up the network interface again.

2. Once you set up your network interface and loopback device on the TCP servers, you can run the same configuration file for the Network Dispatcher machine and TCP servers machines.

The following steps present how to start Network Dispatcher and create the initial configuration.

1. You need to start the Network Dispatcher server.

   ```
   ndserver start
   ```

2. You need to start the Executor.

   `ndcontrol executor start`

3. You need to define the nonforwarding address.

   `ndcontrol executor set nfa IP_address`

4. You need to define a cluster and set cluster options.

   `ndcontrol cluster add cluster`

5. You need to define ports and set port options.

   `ndcontrol port add cluster:port`

6. You need to define TCP server machines.

   `ndcontrol server add cluster:port:server`

7. You may start the Manager component (optional)

   `ndcontrol manager start`

8. You may start the Advisor components (optional)

   `ndcontrol advisor start` *name port*

The Network Dispatcher components (Manager, Executor and Advisors) can be stopped by using `ndcontrol` at the command line.

- Stopping the Manager also stops all the Advisors. To stop it, use:

  `ndcontrol manager stop`

- To stop a specific Advisor (HTTP, FTP, and so forth), use:

  `ndcontrol advisor stop` *name port*

- To stop the Executor, use:

  `ndcontrol executor stop`

To stop the entire Network Dispatcher function, use:

`ndserver stop`

4.4 Configuring the Interactive Session Support

This section briefly explains how to configure the ISS function on Interactive Network Dispatcher with a step-by-step sample. Additional information about ISS configuring can be obtained in Chapter 2, “Interactive Network Dispatcher Concepts” on page 9.

Follow this procedure for editing your ISS configuration file:

1. Define each cell:

   Create a cell you want to manage. First, you need to put in the keyword `Cell` and include a name and a destination of `Global` or `Local`.

   Example:

   ```
   Cell              IND             Local
   ```

2. Define your cell attributes. The following attributes may be specified:

   `HeartbeatsPerUpdate 3`

   The name server updates once every three heartbeats.

   `HeartbeatInterval 10`

   Every 10 seconds, the heartbeat is initiated.

   So with these two parameters, you can determine an update interval.

   The update interval is 30 seconds, meaning the name server is updated every 30 seconds.

   `PortNumber 17800`

   You can use any number except well-known port numbers and reserved port numbers.

   `LogLevel Info`

   There are some other choices for this keyword. When you set it to `info`, it provides all error data provided by the `Error` parameter, plus information on significant events.

   `HeartbeatsToNetFail 5`

   After five heartbeats, this node is regarded as unreachable and is not be recommended.

   `HeatbeatsToNodefail 7`

   After seven heartbeats, this node is considered to have failed completely.

   `Authkey`

3. Define individual node data in the cell:

   For example:

   ```
   Node      lf5575a      001
   Node      lf5575b      002
   Node      lf5575c      003
   Node      lf5575d      099
   NotMonitor
   ```

Nodes lf5575a, lf5575b, and lf5575c act as monitors, but node lf5575d does not act as a monitor. The Node keyword needs to be specified with the IP address or a DNS name and ID number.

4. Specify the ResourceTypes for the cell:

   For example:

   ```
   ResourceType      CPU
   ```

   You can decide your ResouceType depending on your system; it can be CPU availability, CPU idle, disk usage, or memory availability. But you need to configure only one service at a time. “ISS Configuration File” on page 28, explains how to define the metric depending on the ResourceType.

   Here we configure the metric with CPU availability.

   ```
   Metric                 Internal     CPULoad
   MetricNormalization           0     100
   MetricLimits                 80     95
   Policy                              Min
   ```

5. You need to specify each of the services to be provided:

   For example:

   ```
   Service                WWWService www.lf5575.austin.ibm.com 80
   ResourceList           CPU
   Nodelist               lf5575a, f5575b, lf5575c
   SelectionMethod        Best
   Overflow               lf5575d.ind.austin.ibm.com
   ```

   The keyword Service needs to specify with a name, DNS name and optionally, a service port number. When you use the RoundRobin metric, you need to specify RoundRobin on the SelectionMethod keyword.

6. You need to specify each of the observers:

   ISS has three observers (ISSNameserver, Nameserver, Dispatcher). It depends on design of your servers on your network. The following example is for the Nameserver, which changes the DNS name server according to the ISS monitor.

   ```
   NameServer                    f01n01t 53
   ServiceList                   service
   NamedData                     /etc/named.data
   NamedRev                      /etc/named.rev.9.3.1
   ```

For starting ISS, use the following command:

```
issd -c [your configuration file]
```

Run the ISS configuration file on the every machine defined in the cell.

For stopping ISS, use the following command:

```
kill -TERM iss_pid
```

Use the `grep` command to obtain the ISS process ID:

```
ps -ef | grep issd
```

Chapter 5. WWW Server Scenario

The most well-known and widely used service today, the WWW, can be considered the most famous TCP service at present. Each day, millions of people search the World Wide Web for information and commerce. Some sites can even reach 10 million hits per day. This means these sites must have an adequate structure to provide a reasonable response time for people accessing it.

This chapter discusses some tests performed using ISS and Network Dispatcher configured to provide a WWW service.

5.1 Testing Environment Considerations

Each different TCP/IP service demands different resources from the machine, which results in different loads. So it is very difficult to characterize the load for all kinds of service using only one metric. It is necessary to individually study the loads submitted to machines for every service to define individual metrics.

Even for each different service, it is not easy to reach a consensus about what the average load for a machine is. So, a load measurement should be made for each specific situation. Since this is not always possible, people try to study load using some simulations of workload. These simulations are not always precise, but give a reasonable idea of the behavior of the machine, allowing the study of load balancing and scalability. ISS and Network Dispatcher allow you to customize your load-balancing environment.

For the tests, a 5-node RS/6000 SP was used. Four nodes–two wide (nodes 3 and 7) and two thin (nodes 5 and 6)–were the WWW application servers, and one wide node (node 1) was the ISS and Network Dispatcher server. More information about the test configuration files can be found in the Chapter 10, “Configuration Files - Samples” on page 93.

During the tests, some graphics showing network traffic and CPU load in all nodes were collected using Performance Toolbox for AIX (instruments named *f01n01/CPU*, *f01n03/CPU*, *f01n05/CPU*, *f01n06/CPU*, *f01n07/CPU* for CPU load and *f01n01*, *f01n03/TCP*, *f01n05/TCP*, *f01n06/TCP* and *f01n07/TCP* for network load). Some of these graphics are shown in this chapter.

5.2 Load Generated by WWW Servers

The load generated by WWW servers depends on which types of pages are accessed by clients. There are two types of pages:

- Static pages: These pages are ready at the server, meaning the server only gets the file from local storage (disk, network-shared file systems, and so forth) and sends it to the client. This generates only a small amount of CPU load. In this case, the network is usually the bottleneck. There are some studies that show that medium-sized servers can hold a large number of hits per day with static pages.
- Dynamic pages: These pages are generated by a program that is executed when the request for the page arrives. It can access databases, other network servers and so forth. The amount of CPU load generated by dynamic pages depends on the application that will be executed. Simple applications generate less CPU load than complex applications that access several databases and perform calculations. Usually, sites with a great number of dynamically generated pages have CPU bottlenecks instead of network bottlenecks.

When using only static pages, the main concern is network speed. When using only dynamic pages, the main concern is CPU power. Since the set of pages clients access is usually composed of both types, care should be taken in sizing CPU power and network bandwidth.

5.3 How the Tests Were Performed

The Web server used was the IBM Internet Connection Secure Server Version 4.2. All the files served were local for each machine. They were replicated for all the machines.

To generate the load, two other RS/6000 machines were used. A modified version of the WebStone Version 2.0 benchmark was used. The modification was made because in the standard benchmark, the client program resolves the server name just once at the beginning of the execution, not every time a request for a page is made. A public version of WebStone Version 2.0 benchmark can be downloaded from:

`http://www.sgi.com/Products/WebFORCE/WebStone/`

5.4 The Tests and Their Results

This session describes the tests and shows the PTX graphics that explain the behavior of three different configurations: the traditional round-robin, a balanced distribution using ISS, and a balanced distribution using Network Dispatcher.

5.4.1 Round-Robin Configuration

In the tests using only ISS, we chose the `NameServer` Observer because there was already a name server running on the node selected to run the `issd` in monitor mode. The effect of running the `ISSNameserver` Observer instead of the `NameServer` would be a slightly smaller CPU load due to the nonexistence of the `named` daemon.

Using the `RoundRobin` metric, CPU load and network traffic are ignored. If you have an homogenous and very well-known request profile and the requests are quickly over, you can use the `RoundRobin` metric. But it is not very good if you have random-sized requests (as usually is the case, especially in WWW scenarios). The main advantage of the `RoundRobin` metric is that it is very simple to configure.

In round-robin mode, the load on servers is ignored; each server gets all the requests according to the following rule:

number of the servers in the pool * heartbeat interval * heartbeats per update seconds for heartbeat interval * heartbeats per update seconds

If that interval is not long enough, the server will get all the requests again even if it has not finished serving the previous requests, especially when the load is basically composed of dynamic pages.

Figure 9 on page 50 shows an example of the CPU load obtained by configuring a `NameServer` Observer in a round-robin configuration for dynamic WWW pages. The node *f01n01* was configured to be the ISS and DNS server, and the other ones *(f01n03, f01n05, f01n06 and f01n07)* were configured as WWW servers. You can see that `issd` on node *f01n01* selects every server regardless the real load of the machines. The `HeartbeatInterval` was set to 30 seconds.

The ISS server node is not heavily loaded. You can see a slightly higher CPU load in this machine only when it updates the DNS files. The network bandwidth utilization in this case is also very low.

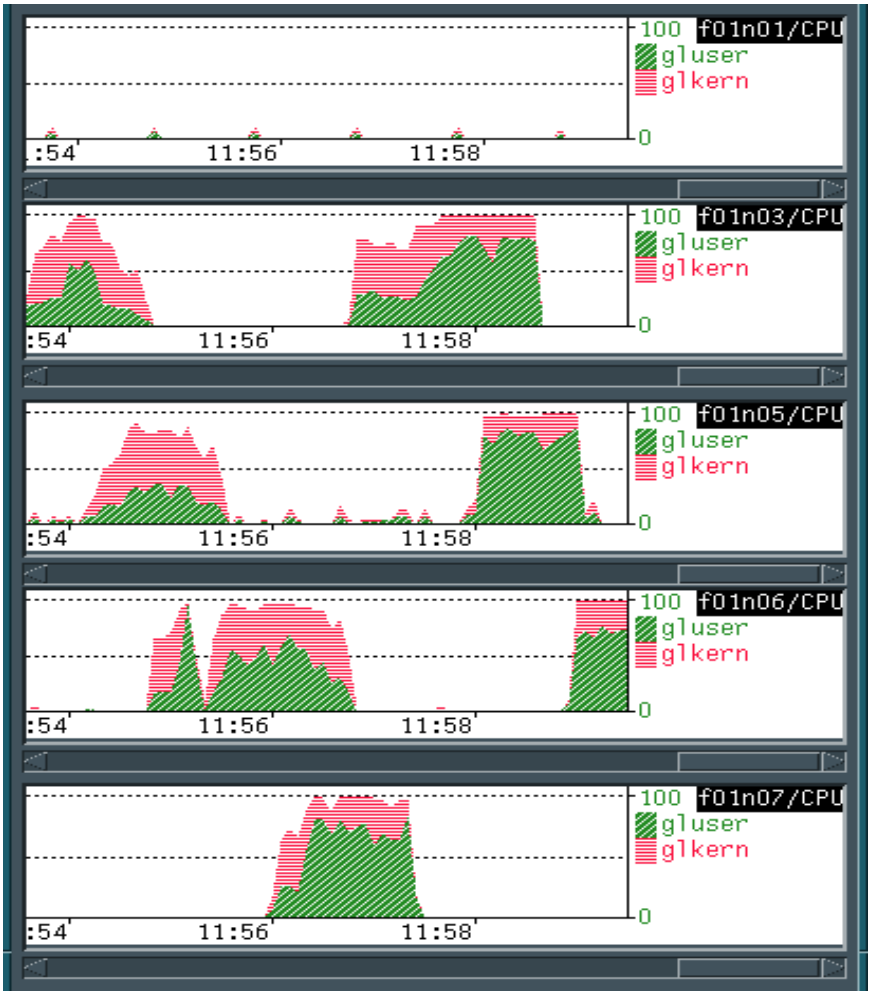


Figure 9. CPU Load - NameServer Observer: Round-Robin for Dynamic Pages

In the case of static pages, CPU utilization is very low, but network traffic is not. Network traffic using static-only pages can be seen in Figure 10 on page 51.

We chose an update interval of 60 seconds, and after every 60 seconds, the selected server changed, but it continued to send TCP packets because there were connections still open.

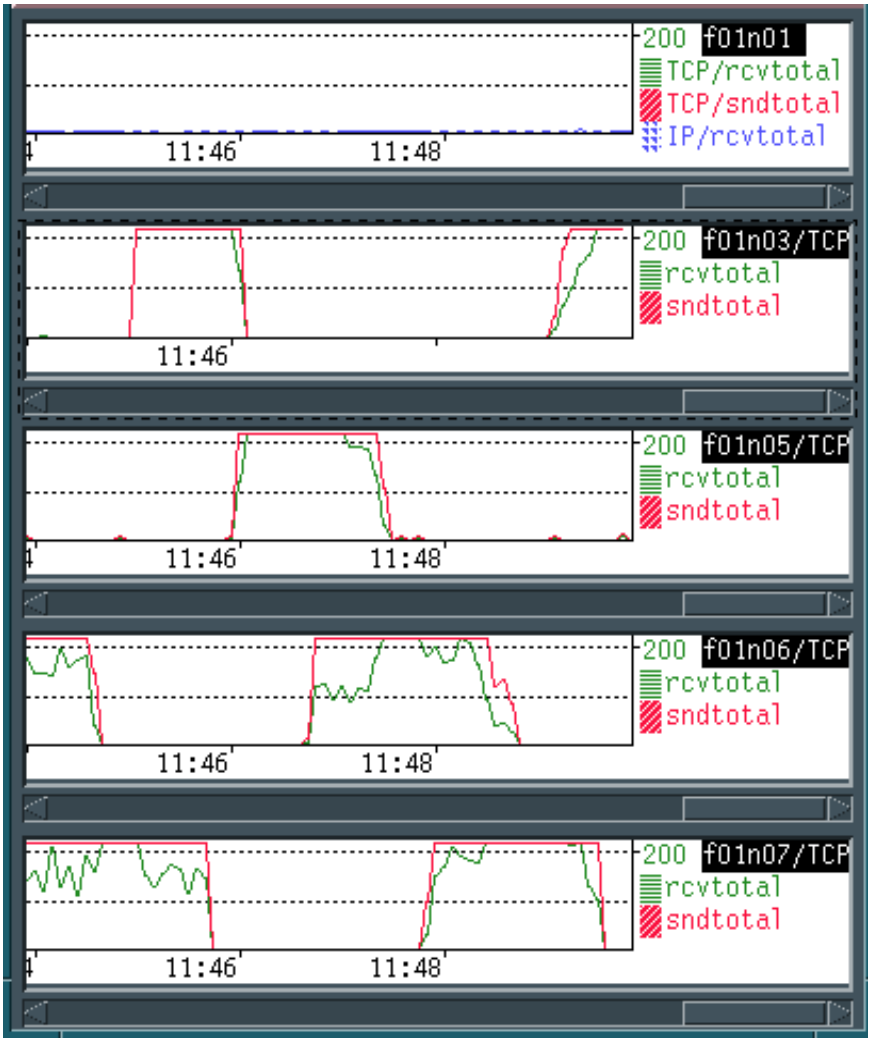


Figure 10. Network Traffic - NameServer Observer: Round-Robin for Static Pages

5.4.2 ISS with Best SelectionMethod

You can configure ISS to work only with a very simple round-robin configuration or to work in a better mode using a `Best SelectionMethod`. In this way, you decide on a specific `ResourceType` that `issd` will use to select the most appropriate server to provide a service. You can decide to balance the load based on CPU load, network traffic, disk usage, and so forth. It depends on your environment and on the service(s) you are providing.

For environments where the greatest part of the access is to dynamic pages, a CPU idle time metric can be used. For static pages, network traffic can be used. For applications that access databases, disk activity time is used.

For example, if you have a Website that provides many dynamic pages, you could use a ResourceType that monitors the CPU load:

```
ResourceType External  vmstat 1 2 | tail -1 | awk '{ print $16 } '
```

This ResourceType calculates the CPU idle time. The Policy must be set to Max to inform the issd monitor that the least loaded servers will be those with the greatest value for the metric.

You will obtain better results using this metric than a simple round-robin configuration. In fact, when you use a specific ResourceType, the issd selects the most appropriate server each time, while looking at the real load of each server. When a server is too heavily loaded, it is not chosen by ISS until either its load decreases or every other server is as loaded (or more loaded) as it is. An example of CPU load using dynamic pages and an idle CPU time ResourceType is shown in Figure 11 on page 53. Notice that the load on the ISS server (node 1) is very low but higher than in the round-robin configuration because the machine receives the load information packets from the servers and then calculates which one is the best. For the same test, the network traffic is shown in Figure 12 on page 54.

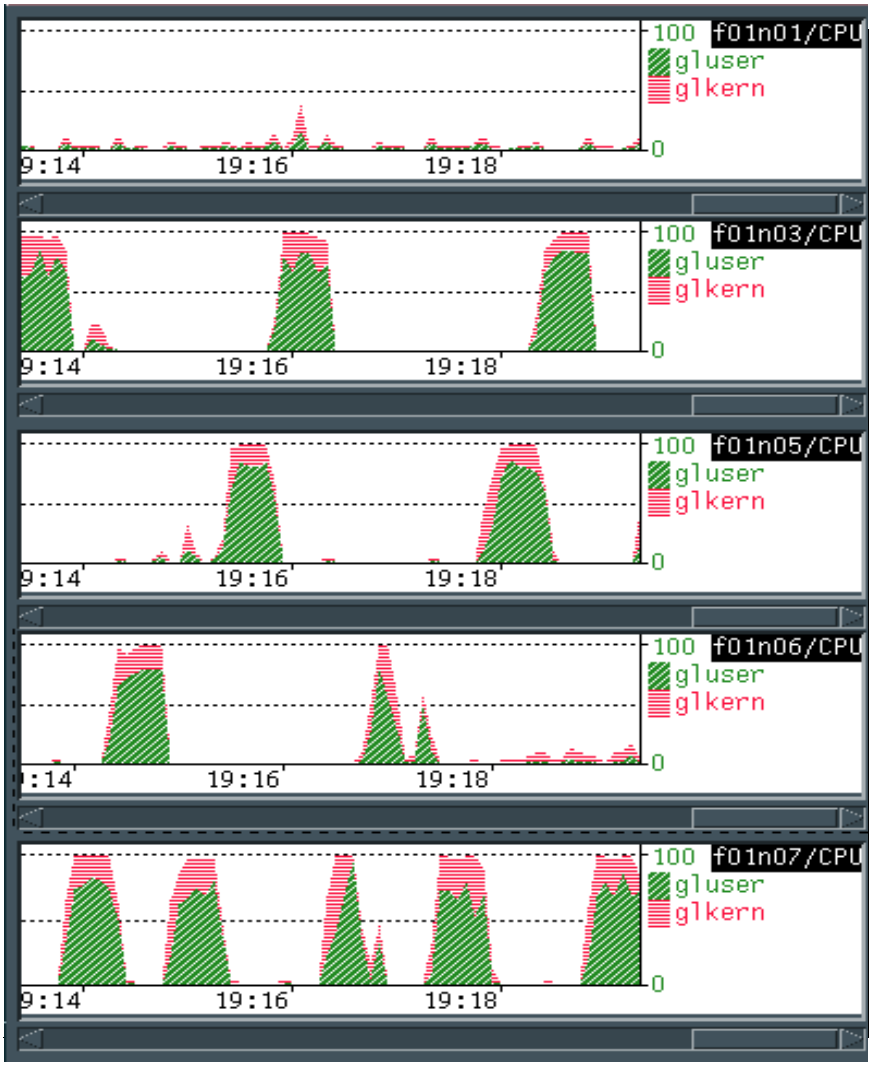


Figure 11. CPU Load - ISS with Idle CPU ResourceType for Dynamic Pages

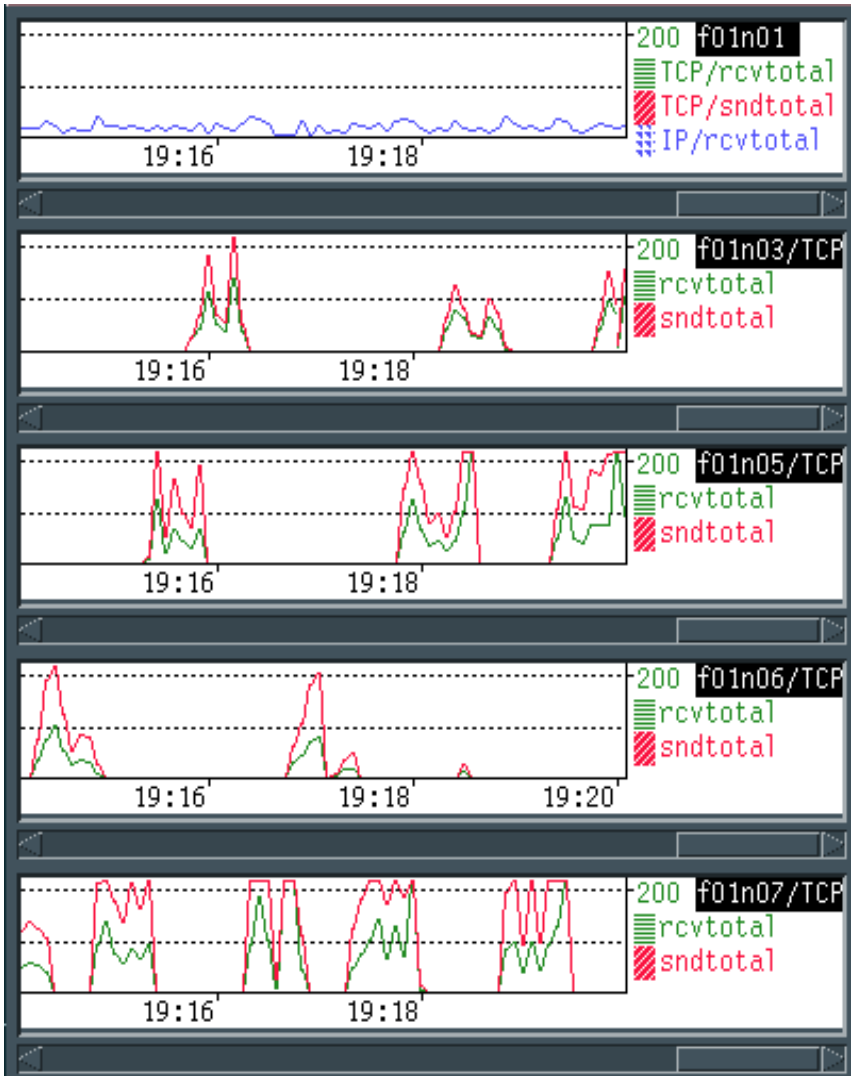


Figure 12. Network Traffic - ISS with Idle CPU ResourceType for Dynamic Pages

5.4.3 Network Dispatcher

To test Network Dispatcher, we used the same environment used for ISS and Node1 (*f01n01*) of SP2 as the Network Dispatcher machine.

Dynamic Pages

Using dynamic pages, all the CPU load graphics were similar. The distribution is smoother than using ISS alone. With Network Dispatcher, the load is better distributed among the servers because of the way Network Dispatcher routes IP connections instead of using DNS to distribute the load.

In Figure 13 on page 56, notice the difference between the configuration that uses only ISS and the one which uses Network Dispatcher. Every server is loaded all the time by simulating a heavier load. In this case, we have a better distribution of all the resources. The Network Dispatcher machine does not work so much in terms of CPU, but there is a lot of IP traffic because every client request goes to this machine before it is routed to the best server.

We set the proportions to 50 50 0 0 and the smoothing value to 1. With a higher smoothing value, there are fewer peaks, and the load distribution is smoother.

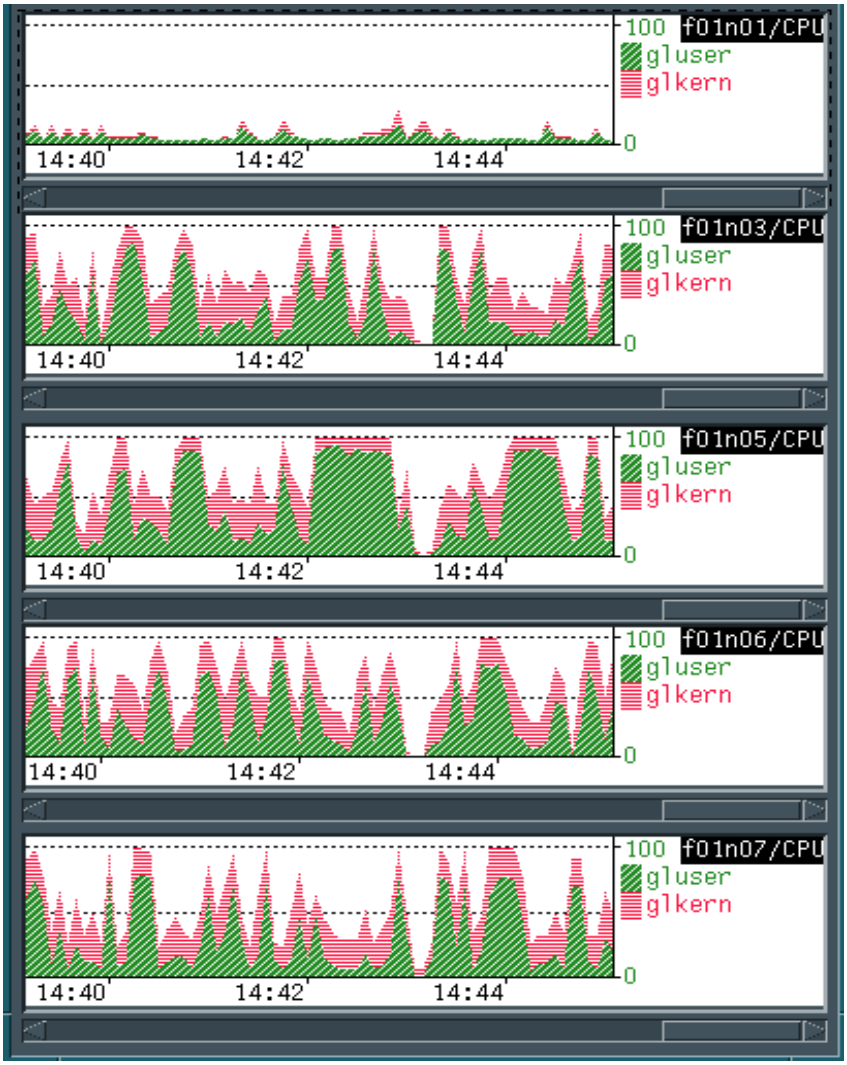


Figure 13. CPU Load Using Network Dispatcher for Dynamic WWW Pages

Figure 14 on page 57 shows the network traffic obtained with the same Network Dispatcher configuration. The number of total received IP packets iin the Network Dispatcher machine is very high because it works at IP level.

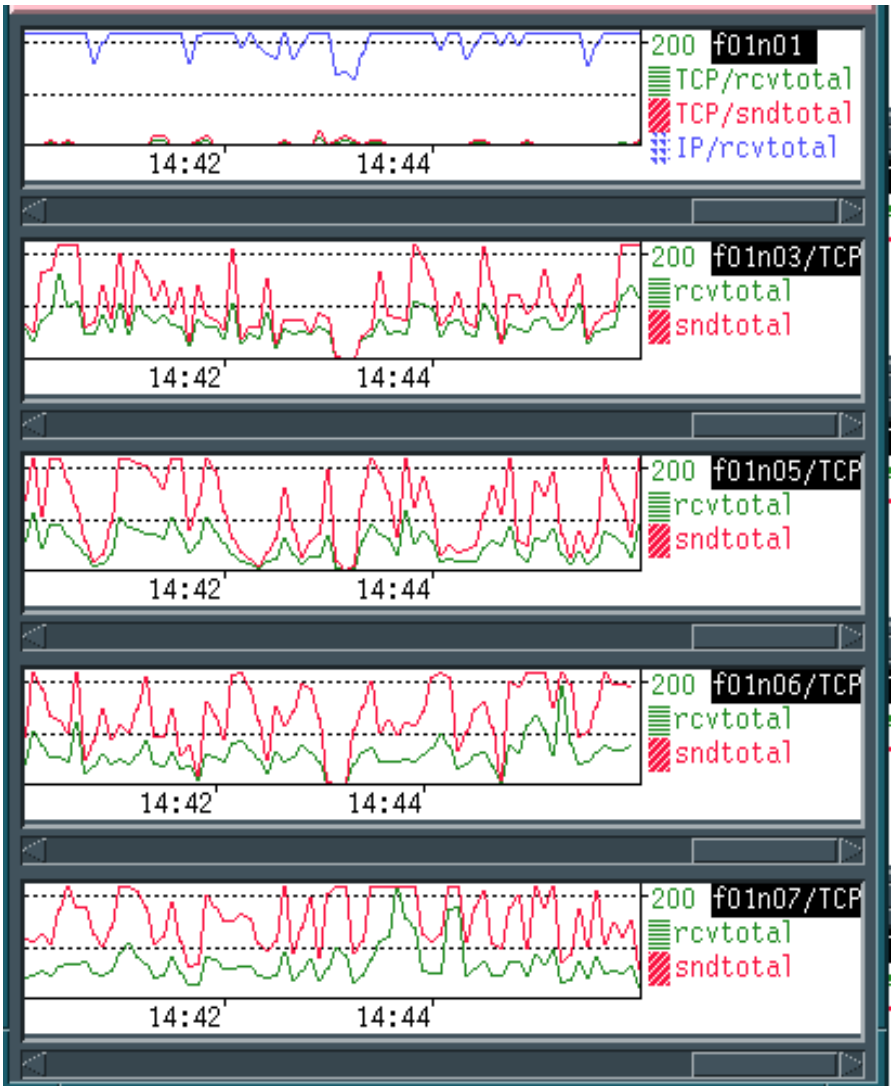


Figure 14. Network Traffic Using Network Dispatcher for Dynamic WWW Pages

Static Pages

With static pages, CPU load was very low. So, the main concern should be network traffic. If you run the Network Dispatcher advisors and/or ISS, you will have slightly higher network traffic due to the server information that the

advisors collect for the Manager and due to the communications between `issd` agents and the `issd` monitor and between the `issd` monitor and Manager, if they do not run in the same machine.

Network traffic distribution was smoother with both round-robin and best configurations than it was with ISS only. The client requests simulated were variable; that's why, in some short periods of time, the Network Dispatcher didn't receive any requests, and the server did not have any load. An example of the network traffic graphic is found in Figure 15 on page 58.

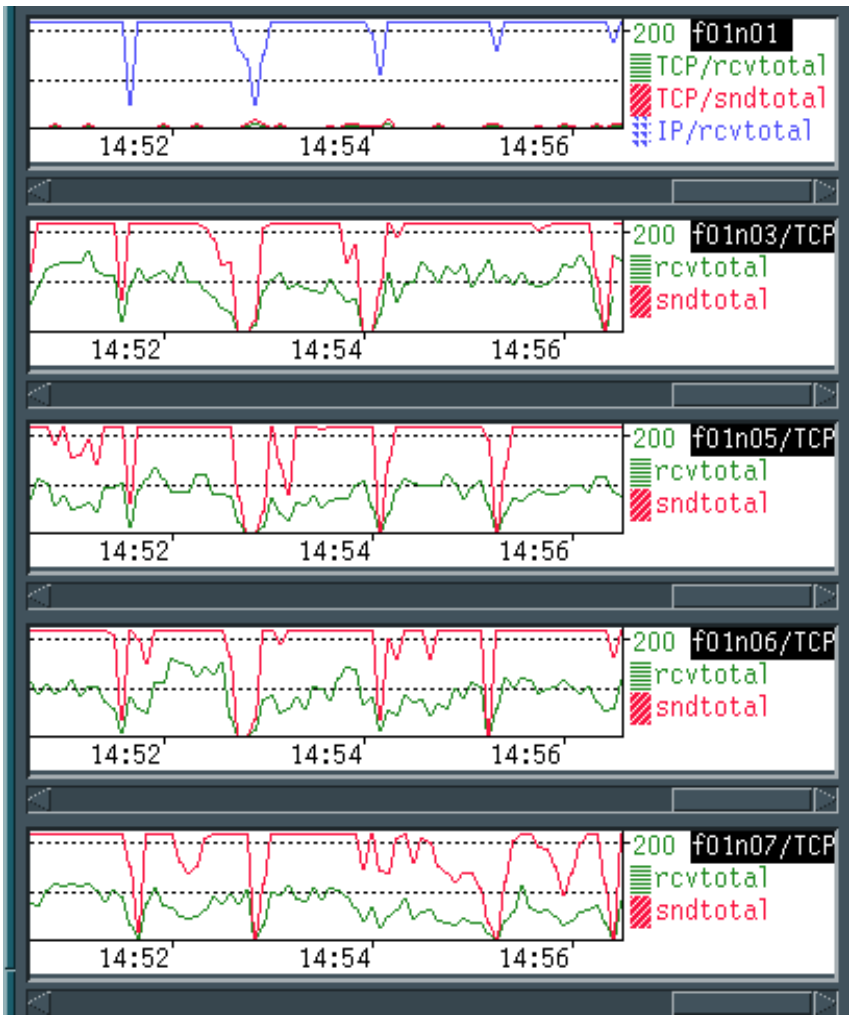


Figure 15. Network Traffic Using Network Dispatcher for Static WWW Pages

5.5 Conclusion

The RoundRobin metric is suitable only if load is well known and stable. It can load even a heavily loaded server by not considering actual load. The only advantage is that it is easy to configure.

The possibility of defining a ResourceType is suitable for every kind of application. Depending on your environment and on the service you want to provide, you can set up your own ResourceType. While CPU idle time is good for dynamic pages, network traffic is good only for static pages. Of course, measuring the CPU load you must remember that issd loads the CPU a little bit more.

Network Dispatcher uses results in a smooth distribution of CPU load for dynamic pages and of network traffic for static pages. Network traffic is higher when using both ISS and Network Dispatcher than when using ISS alone. This is the best way to distribute load among servers.

Chapter 6. FTP Server Scenario

FTP stands for File Transfer Protocol, which is defined as a protocol for file transfer between hosts on the TCP/IP network or on the Internet. The primary function of FTP is defined as transferring files efficiently and reliably among hosts, thus promoting the sharing of files. FTP is a convenient way to use remote file storage capabilities, shielding the user from variations in file storage systems among hosts. FTP is one of the most commonly used protocols on the Internet. FTP is a very useful and valuable Internet service; it is used programs, large databases, video, audio, and images. One of the most common types of servers available on the Internet are the FTP servers, which are servers with a repository of files for public access over the Internet. They are known also as *anonymous FTP* servers.

This chapter discusses some tests done using ISS and Network Dispatcher configured to provide FTP services.

6.1 Testing Environment Considerations

For the tests, a 5-node RS/6000 SP was used. Four nodes–two wide (nodes 3 and 7) and two thin (nodes 5 and 6)–were the FTP servers, and one wide node (node 1) was the ISS and Network Dispatcher server. More information about the test configuration files can be found in Chapter 10, “Configuration Files - Samples” on page 93.

During the tests, some graphics showing network traffic and CPU load in all nodes were collected using Performance Toolbox for AIX (instruments named *f01n01/CPU*, *f01n03/CPU*, *f01n05/CPU*, *f01n06/CPU*, *f01n07/CPU* for CPU load and *f01n01*, *f01n03/TCP*, *f01n05/TCP*, *f01n06/TCP* and *f01n07/TCP* for network load). Some of these graphics are shown in this chapter.

6.2 Load Generated by FTP Servers

FTP is a client/server application. On the client side, the user runs the `ftp` command, which opens a control connection with the FTP server. The FTP server on UNIX machines are implemented by a daemon called `ftpd`. The `ftp` command is a user interface; the subcommands the user types are translated into standard FTP commands and sent through the control connection using the Telnet protocol. The FTP commands specify the parameters for the data connection (data port, transfer mode, representation type, and structure) and the nature of the file system operation (store, retrieve, append, delete, and so on). To transfer data, the FTP processes use a different port, known as the

data port. On the server side, the standard port for control connection is 21, and for data, the port is 20. Figure 16 illustrates how FTP works.

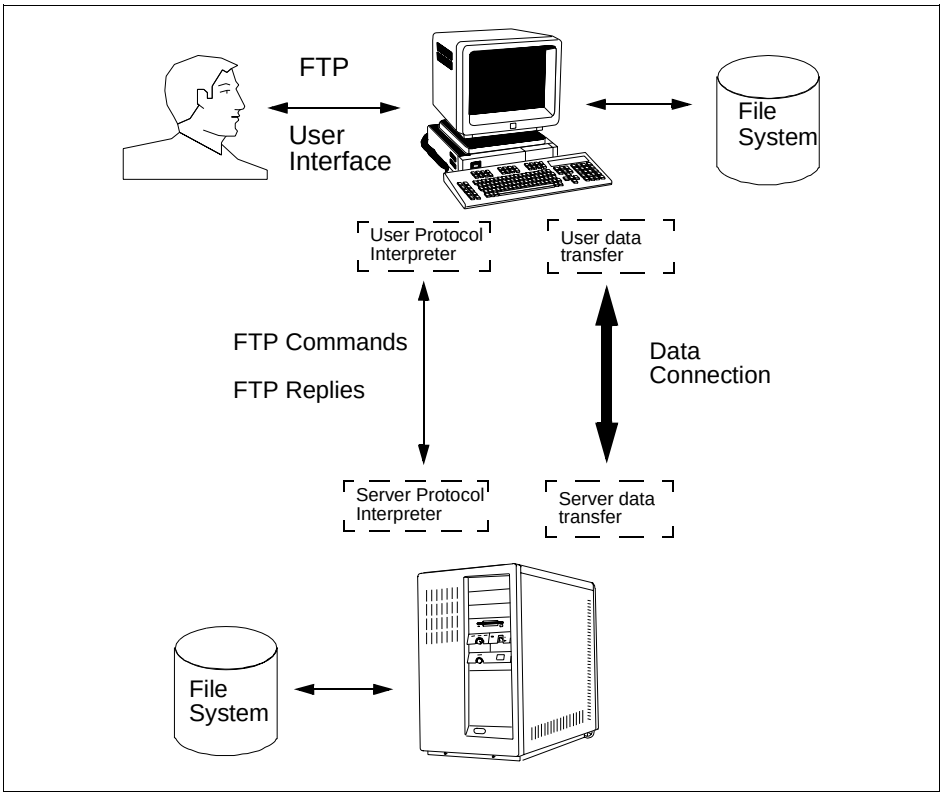


Figure 16. Interaction of the Client and Servers with File Transfer Protocol (FTP)

The FTP processes are not CPU intensive; the main load an `ftp` application puts on the server is network traffic. In order to transfer a file from the server and to the client, two TCP/IP connections are needed; so more packets are sent than with the HTTP protocol. Control commands just generate small packets. For example, the commands generated by the FTP user interface subcommands, `cd`, `pwd`, `bin`, and `prompt`, use only the control connection. But commands like `ls`, `get`, and `put` open a data connection to do a bulk transfer of data.

6.3 How the Tests Were Performed

To test this environment, we used the FTP server (`ftpd` daemon) including AIX and the `ftp` command as the user interface. All machines were connected through a token-ring network at 16 Mbps. In order to simulate various users accessing the server, we used a special utility called Expect. Expect is a program that "talks" to other interactive programs according to a script. Following the script, Expect knows what can be expected from a program and what the correct response should be. An interpreted language provides branching and high-level control structures to direct the dialogue. Expect is built on top of Tcl, which is an embedded scripting language. A public version of Expect can be downloaded from:

`http://expect.nist.gov/`

We wrote an Expect script to simulate a user accessing an FTP server using the `ftp` user interface. The script simply connects to the `ftp` server; it gives a user and a password, changes to a remote directory and selects a file to transfer (`get` command) at random. On the server, there were four different files with sizes of 10 KB, 50 KB, 500 KB, and 5 MB. The transfer time was measured. Because of the heavy load, some connections were timed out, in which case it was recorded in a log. The script executes a loop and repeats the same process for 10 minutes. It waits a random amount of time before starting the next `ftp` command (1 or 2 seconds). The script exits after 10 minutes, but waits for the current connection to finish or time out. We used four machines, each running 30 of these processes for a maximum of 120 concurrent FTP connections at all times. The load we generated, even pseudo-random, maintained a constant flow of TCP traffic through the servers, which is not always the case in a production environment, but it helps to determine the best configuration for ISS and Network Dispatcher with FTP servers.

6.4 The Tests and Their Results

This session describes the tests and shows the PTX graphics that explain the behavior of three different configurations; the traditional round-robin, a balanced distribution using ISS, and a balanced distribution using Network Dispatcher.

6.4.1 Testing with ISS

As with the WWW tests, we configured ISS in the `NameServer` mode, which sends a signal (SIGHUP) to the `named` daemon on each update, but as we said

before, the ISS machine did not have any CPU performance problems. Node 1 on the SP (*f01n01*) was configured as the name server and ISS monitor; the other four nodes (*f01n03, f01n05, f01n06 and f01n07*) were used as FTP servers.

6.4.2 Round-Robin Configuration

Round-robin could be a totally random way to distribute the load as it never knows when a server is too loaded or how many requests it is already serving. Many FTP sites use a random method to distribute the load; giving to the user a list of sites where he/she can download its files and which let him/her decide at "random". This may sound simple, but it has been proven to be a practical way to distribute the load. The ISS with round-robin will generate a slightly better distribution without bothering the user.

For the test, we set the heartbeat interval to 30 seconds and the heartbeats per update to 2 seconds. In this way a new server was chosen every 60 seconds. Since the load was constant, the monitor showed an evenly distributed load of traffic on all servers, with some peaks at certain times. These peaks were expected since ISS does not recognize when a server is heavily loaded. The results we obtained using ISS in our test is summarized in the Table 3.

Table 3. ISS Round-Robin FTP Simulation Results

File Size	Num.of Files	Total Bytes	Total Time	Avg KB/sec
10 KB	148	1515520	4.58	447.75
50 KB	125	64000000	113.84	594.79
500 KB	127	6502400	14.30	512.42
5 MB	135	42024960	72.98	625.26
Totals	535	108192880	205.70	545.05

The test ran for 13 minutes; so the total actual transfer rate for the cluster was approximately 1.43 MB/s. In this test, we set the time-out to 500 seconds, and we got 31 connections that timed out. ISS does a good job of distributing the load over the servers, but as we said before, this is not the typical load on an FTP server.

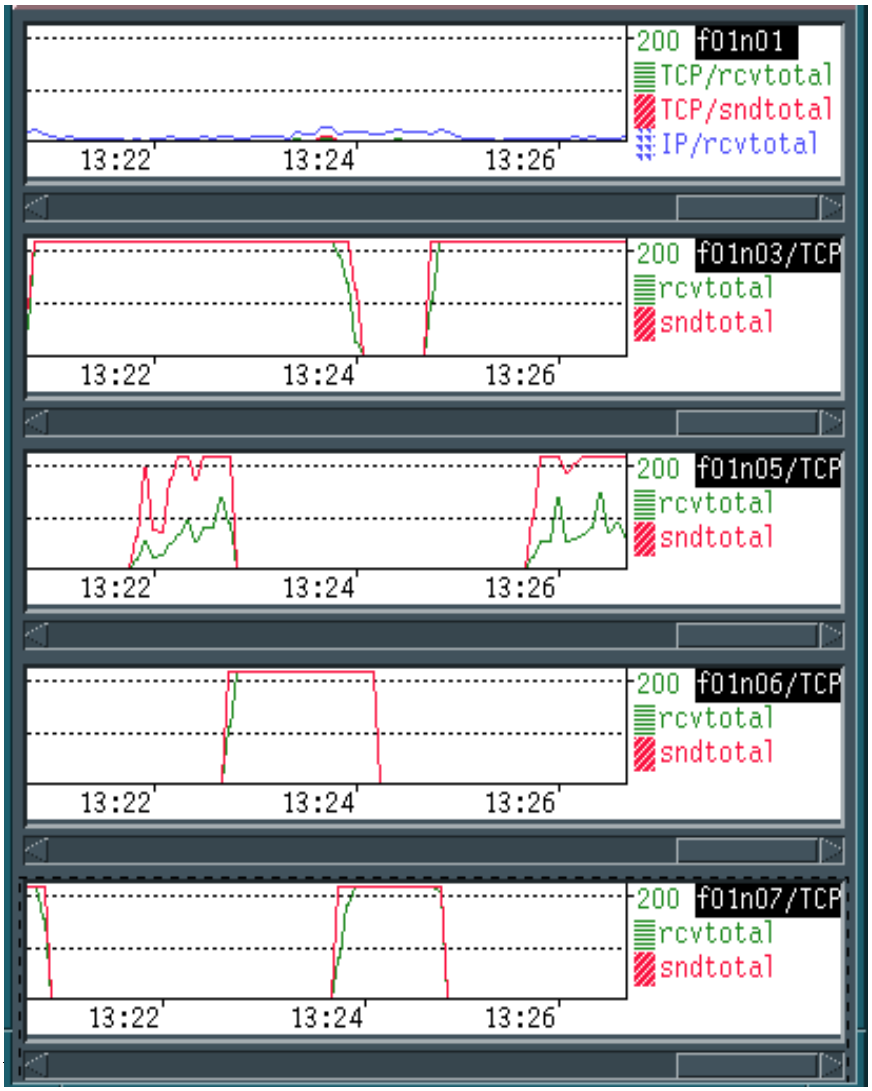


Figure 17. Network Traffic- ISS with the Round-Robin Configuration

Notice that the traffic is evenly distributed among all the servers most of the time. As you can see, there is one heavily loaded server, but others are getting loaded in turn. This is the normal behavior you would expect when using ISS in round-robin.

Other problems you may find with ISS is the response time to the users, since sometimes they will connect to a heavily loaded server and other times to a lightly loaded one. The response time they get varies greatly.

6.4.3 ISS with Best SelectionMethod

To have a better distribution of the load, you can add some intelligence to the server selection process. Because FTP load is over network traffic, we chose a metric that takes into account IP traffic over the token-ring interface. We took the output from the `netstat` command over the interface tr0, and from this we selected the input and output packets and output the sum as the metric value. On the ISS configuration file, we had the following line:

```
ResourceType External netstat -I tr0 | awk ' NR == 4 { print $1+$3; exit } '
```

We set the heartbeat interval to 3 and the heartbeats per update to 2 seconds; so every 3 seconds, the ISS monitor received input from the ISS agent, and every 6 seconds it updated the `named` daemon (if necessary). The overhead of the ISS agents on the nodes for both on traffic and CPU was minimal. The Figure 18 shows some peaks on node 1 (*f01n01*) that corresponds the `issd` daemon CPU usage.

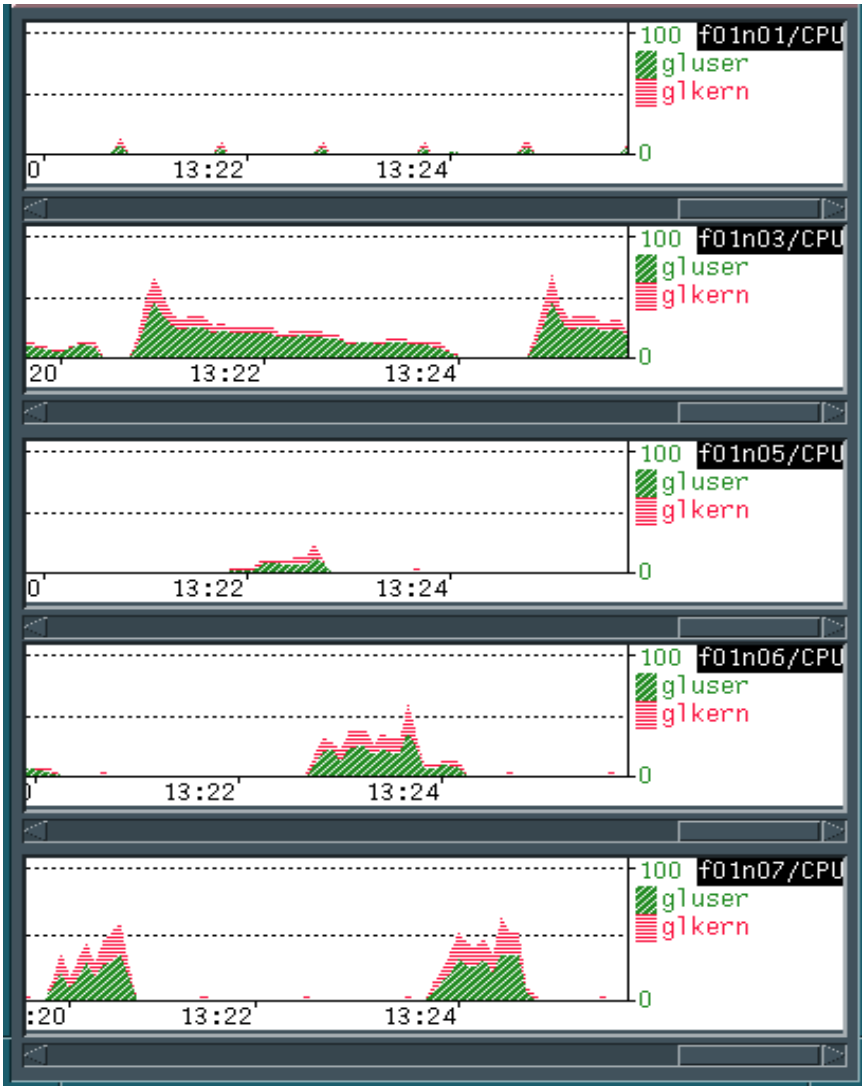


Figure 18. CPU Load - ISS with the Round-Robin Configuration

The test ran for 13 minutes. The actual transfer rate for the cluster was approximately 1.48 MB/s, which is just slightly better than the result we got with `RoundRobin`, but in this case we just got five connection time-outs, which was a significant improvement over the round-robin configuration. The distribution of work was also slightly better than with the round-robin configuration.

Another advantage that this metric gave us over the ISS in round-robin was the average response time (time to transfer a file). Using ISS with a custom metric to balance load, the clients got approximately equal response time on each request, meaning that there are not many changes from the response time of each subsequent request. We could also see that no client got a response time worse than 350 seconds as opposed to the round-robin configuration, where some clients got a response time of over 400 seconds.

6.4.4 Network Dispatcher

FTP has special consideration when used with Network Dispatcher. As we explained before, FTP uses two connections to make a transfer. The first connection is the control connection, where the user specifies the parameters for the data connection. When a transfer is going to take place, the client side opens a new process that "listens" to a specified data port for a connection from the server. The server initiates the data connection and makes the data transfer in accordance with the specified parameters.

Since the server is the one that initiates this connection, the connection is not known by Network Dispatcher. All the traffic going from the client to the server in the cluster must pass through the Network Dispatcher. The only way to ensure that this connection is going to reach the same server is though the affinity between the client and that server implemented by Network Dispatcher. In this way, all further FTP traffic (destination ports 20 and 21) from the client machine will be routed to the same server after the first connection is established. This means that if a second FTP process is initiated by any other users while the first user is still connected, it will be routed to the same server.

To define the FTP ports to the Network Dispatcher, you must enter the following commands:

```
CLUSTER=pool2.sp.itsc.austin.ibm.com
ndcontrol port add $CLUSTER:21
ndcontrol port add $CLUSTER:20
```

Some FTP clients support passive FTP, in which case the server data transfer process is placed in a "passive" state instead of the normal, active state. The server data transfer process then waits or listens for a connection rather than initiating a connection on the data port. If you are going to use FTP clients with the passive option, you need to enter the `pftp` option on the control port definition:

```
CLUSTER=pool2.sp.itsc.austin.ibm.com
ndcontrol port add $CLUSTER:21 pftp
ndcontrol port add $CLUSTER:20
```

The proportions for the network dispatcher were set to 50-50-0-0. However, since we only had four clients to test, all connections from one client went to the same server. In our case, four servers received all the connections. This can be seen in Figure 19 on page 69.

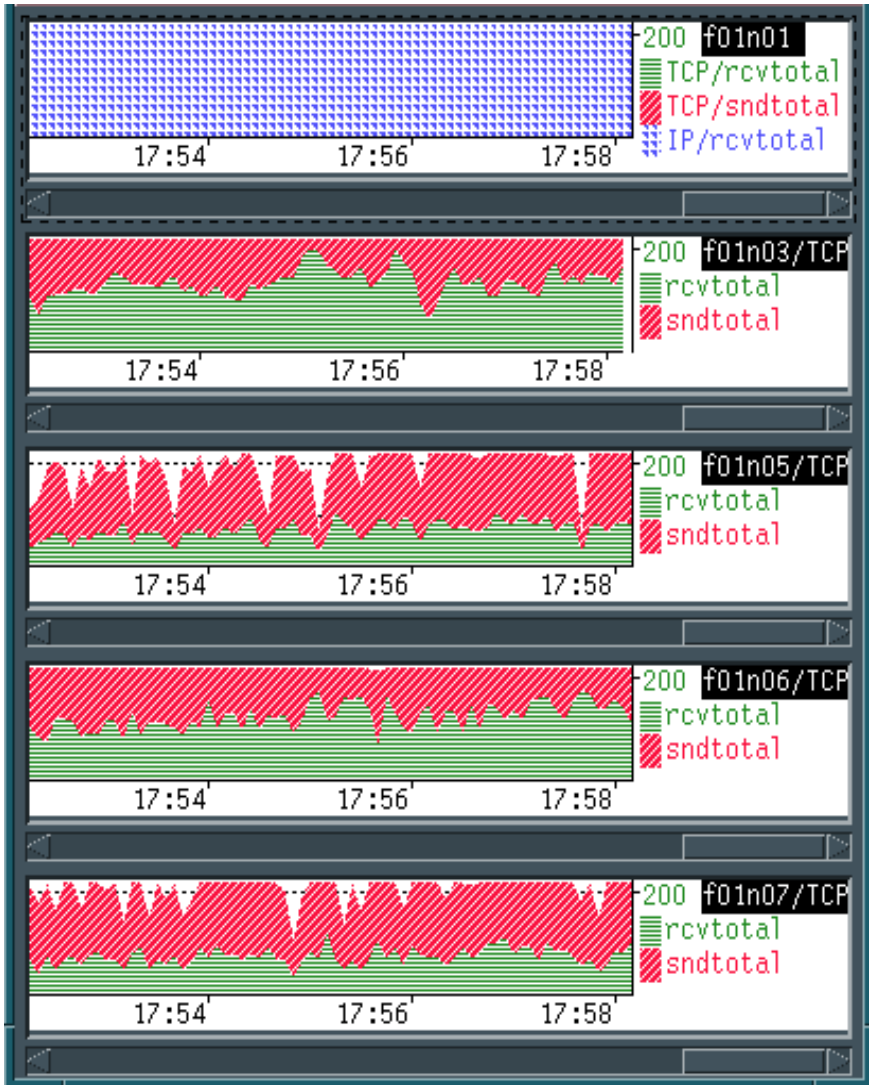


Figure 19. Network Traffic - Network Dispatcher - FTP Simulation of Load

In Figure 19, we can also see that FTP generates many packets, all of which are routed through node1 (*f01n01* in the graphic). The total number of packets generated by FTP are usually more than the ones generated with HTTP because of the two connections; so more network traffic load is generated on the network dispatcher node.

6.5 Conclusion

The `RoundRobin` metric is adequate to distribute the load over your FTP servers. Most of the time, the load will be evenly distributed, but the clients sometimes will experience bad response time.

ISS using a custom metric seems to be a good configuration option for this case; the overhead on CPU time and network traffic is insignificant. Your servers will have a well distributed load, and the clients will get a response time proportional to the load of all the cluster.

The Network Dispatcher would be a good option to use for balancing FTP traffic in most situations, except when the number of client machines accessing your FTP pool of server is small. In this case, all traffic will be routed to the same server until all active FTP connections are closed. This may not be a problem when your clients are coming from the Internet, where the odds of receiving many connections from the same client are very small, but it may be a problem on an Intranet, where there may be just a small number of server machines hosting multiple users.

Chapter 7. Other Server Scenarios

Although the most common Internet servers are WWW and FTP, there are other very important scenarios where Interactive Network Dispatcher can be useful. This chapter addresses some of these scenarios

7.1 Telnet Server Scenario

The purpose of the Telnet protocol is to provide a fairly general, bi-directional, eight-bit, byte-oriented communications facility. It is used for users to log in a host on the network. It uses a TCP connection to transmit its data. Although most of the people use it just to emulate a remote terminal, its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other. Conceptually, the protocol may also be used for terminal-terminal communication ("linking") and process-process communication (distributed computation).

In order to have a pool of servers giving access to several users, the environment each server provides to the users must be indistinguishable from the others. They must provide the same password and same data to the users. This can be accomplished by using several types of distributed file systems and user-management systems, such as NFS/NIS and DFS/DCE. Depending on your application, you can also use a distributed database management system (DBMS) like DB2, Oracle or Sybase.

The load generated on the server by the Telnet clients is application-dependent. The users connected to the server over a virtual terminal and then execute whatever command or processes they have access to. The users may start a database application, a spreadsheet, and so forth. There is not a generic load that will be generated on the server, but some loads may be more common than others. In general, user applications over a Telnet session will consume CPU, memory, and I/O most of the time, but not network bandwidth.

With ISS for AIX Release 1.0, we performed a test on Telnet. Based on the results, we reached the following conclusion.

The load on the servers from requests coming from Telnet clients will always depend on the applications and user profiles. The best way to balance this load is to use a metric that reflects the load of your application. A good option is to use ISS with a custom metric. The option that gives more granularity on directing the request is Network Dispatcher with feedback from the ISS agents.

7.2 Domino Server Scenario

Lotus Notes Workstation and Lotus Domino Server provide a powerful client/server environment for workgroups; simply stated, they are groupware. Lotus Notes allows people to work together regardless of the software or hardware platforms they are using and regardless of any organizational or geographical boundaries. It enables organizations to communicate, collaborate, and coordinate key business processes. Lotus Domino is a server technology that transforms Lotus Notes into an interactive Web application server, thus allowing any Web client to participate securely in Notes applications.

Bridging the open networking environment of Internet standards and protocols with the powerful application development facilities of Notes, Domino provides businesses and organizations with the ability to rapidly develop a broad range of business applications for the Internet and intranet. The key elements of Lotus Notes are its ability to distribute object store, its powerful messaging system, and its application development environment. In its basic form, Lotus Notes is a document database that has the ability to distribute information throughout the enterprise through replication.

A Domino server can be accessed through a Lotus Notes Workstation or through a Web browser. When configured in a cluster of multiple servers, Domino Advanced Server has its own load balancing feature for Notes Workstation clients. You can specify a server availability threshold on the notes.ini file, and when the availability index falls below this threshold, clients are redirected to another server. This option is only valid for Notes Workstations. For load balancing of requests coming from a Web browser, you can use ISS and Network Dispatcher.

Even if the documents we accessed may be considered "static" in content for a standard benchmark, these were really dynamics pages for Domino. This is because each HTTP request from a client activates a thread in Domino which has to open the database, access the particular document, and translate it to HTML format. Domino generates custom HTML pages from the object store on-the-fly based on time, user identity, user preference, client type, and data input.

Domino’s object store differs from a file store in that it separates form from content (page layout and page data); form and content are joined when a document/page is requested. At the time of the request, Domino may also integrate information from other data sources such as a relational database. So in the benchmarks, we did not make a distinction between static and dynamic pages; for Domino everything is dynamic. Also because of the way

Domino handles requests, the most constrained resource is CPU most of the time.

With ISS for AIX Release 1.0, we performed a test on Domino. Based on the results, we reached the following conclusion.

As we said before, the Domino servers will constrain the CPU of the servers most of the time; so the best load balancing solution will be one which takes this resource into consideration. ISS with a continuous custom metric is a good option to balance the load of a Domino server, but the best option for load balancing of a Domino server is Network Dispatcher with feedback from the ISS monitor using a continuous metric. Although ISS in `RoundRobin` shows good results, this may not be always the case. In the environment we tested, the load was always uniformly generated by the WebStone clients. In a real environment, this would not always be true.

Chapter 8. High Availability Server Scenario

The high availability configuration detects and recovers from network and server failures. Interactive Network Dispatcher is smart enough to determine that a server or a network is down. In case of failure, clients lose only the current connections, but they can immediately establish a new connection to the remaining servers with no problem.

This chapter covers concepts and the results of some tests performed using highly available configurations with ISS and Network Dispatcher.

8.1 ISS High Availability

The new release of Interactive Network Dispatcher now eliminates the ISS machine running in monitor mode as a single point of failure. The `issd` process running in monitor mode can be in a different machine than the one which works as a DNS name server (if you are using either the `NameServer` or the `ISSNameserver` Observer). When you declare the nodes that will be part of the ISS cell, you can define them either with the `Node` or with the `Node NotMonitor` keyword.

A `Node` machine is a node in which the `issd` process can run in monitor mode. You define for each `Node` a priority number. The `Node` machine can be a server of one or more services, can simply work as an ISS monitor, or it can wait in stand-by mode. If the first node fails, the next in the priority list takes over as `issd` monitor. As soon as the failed node is operational again, it will become the monitor because of its higher priority number.

A node `Node NotMonitor` machine is a server of one or more services of the ISS cell, and the `issd` process runs only in agent mode. For a more complete explanation of the keyword used to define a server that is part of a cell, see “ISS Configuration File” on page 28.

8.2 Failure of a DNS Server

High availability is not guaranteed if ISS is configured for working with a `NameServer` or `ISSNameserver` Observer and the DNS machine fails. If clients point only to that DNS machine, configured to be either a `NameServer` or an `ISSNameserver`, ISS can no longer provide any service when fails. A solution could be to set up a particular configuration providing the client with a substitute name server.

This option makes use of a feature of the TCP/IP name resolution system. This feature provides the ability to configure a client or a site name server to use multiple name servers. In a UNIX client system, this is implemented by simply listing multiple servers in the /etc/resolv.conf file. The name servers will be used in the order they are listed. The client will contact the first listed server to resolve a name or address. If the first server does not respond, the client will then contact the second server. This can continue to further servers if required.

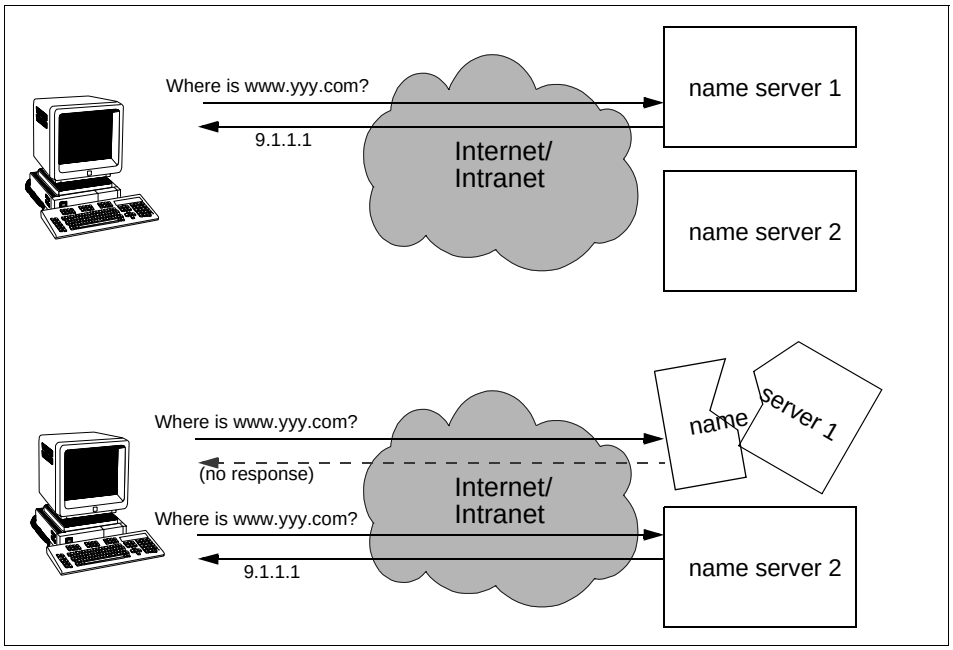


Figure 20. Configuring a Client to Use Multiple Name Servers

It is important to realize that the second name server will only be queried if the first server is not able to answer the request. However, multiple clients can be configured so that some clients list the name servers in the opposite order to spread the load. Each name server would then act as a backup for the other. This is a very common configuration in large networks.

To use this type of configuration with ISS, you will need two (or more) ISS systems. Each one should be configured as a DNS server for the ISS-served domain, either using the `named` daemon with the `NameServer` Observer or with the `ISSNameserver` Observer. Each server should contain the necessary ISS and DNS configuration files, which can be part of the same ISS cell.

An example of this configuration is shown in Figure 21.

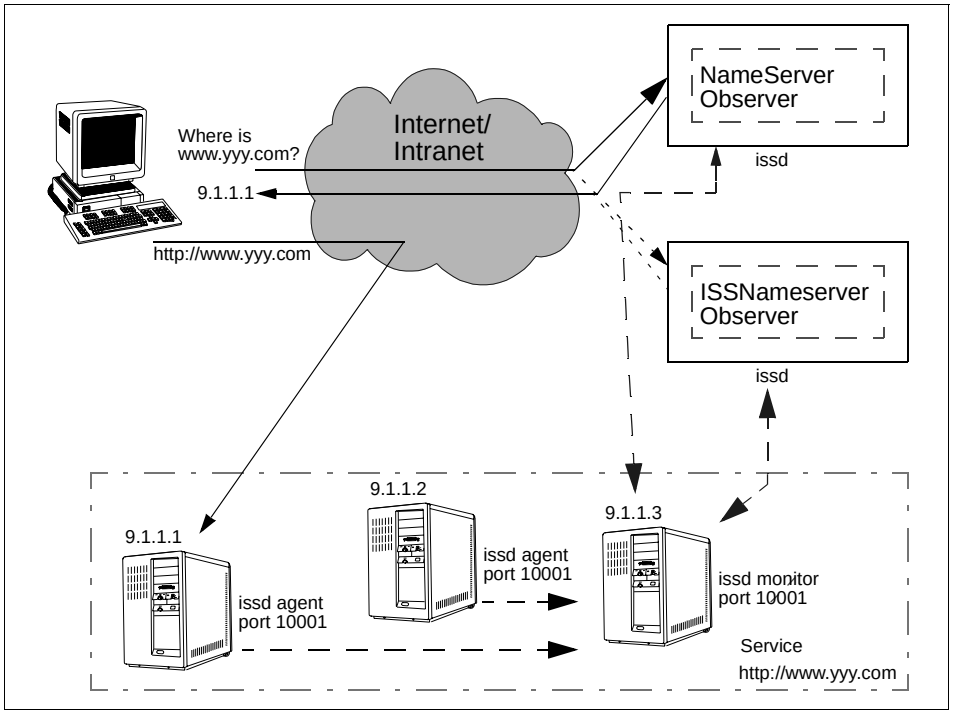


Figure 21. Example of Multiple ISS Servers in a High-Availability Configuration

This technique cannot be used with Network Dispatcher. It requires that each Network Dispatcher server system be configured with the cluster address aliased to the network adapter. Doing this on two systems in not permitted in TCP/IP because this would mean having the same IP address defined on two different systems on a single network.

8.2.1 One Possible Solution

Configure the systems for this environment as follows:

1. If you have a DNS forwarding requests to the cell name servers, you should ensure that the DNS server for the site has a name server entry listing each of the ISS systems as name servers for the cell domain.

   If you have clients sending DNS requests directly to the ISS systems, ensure that all of the ISS systems are listed in the clients' /etc/resolv.conf or equivalent files.

2. On the ISS systems, you should install ISS for AIX and store the ISS configuration files on a local disk or on some shared disk technology that will still be available in the event that the other ISS system fails.
3. If the `NameServer` Observer is being used, you should configure DNS on each system, again storing the files locally. Start the `named` daemon. If you use a `RoundRobin` metric type, no additional configuration is required.
4. If you use the `LoadLeveler` metric type, you must install the LoadLeveler product on each system, and ensure that they are configured as LoadLeveler servers capable of scheduling jobs.

8.2.2 Pros and Cons

The main advantage of this solution is that it is extremely simple to configure both clients and servers using this method. Also, no additional software or configuration and management skills are required.

The main disadvantage of this solution is that it does not work for Network Dispatcher because of the duplication of IP addresses on Network DIspatcher systems that this would require.

8.3 Network Dispatcher High Availability

This release of Interactive Network Dispatcher comes with a new high availability feature. You can now configure a backup machine that takes over the primary machine without using the HACMP/6000 product.

The environment involves a *primary* machine that works normally as a Network Dispatcher, balancing the load among the servers of its clusters. You configure a *backup* machine, too, that stands in stand-by mode unless the primary fails. The two machines are synchronized, and only the primary routes packets while the backup one is continually updated. The two machines establish communication to monitor the status of each other, referred to as a *heartbeat*, using a port that you can choose. If the primary fails, the backup detects this failure and begins to take over the routing packets. When the primary machine is operational again, you can either decide that it again automatically becomes the primary machine, or leave it in stand-by mode and act manually if you want it to become the primary again.

Figure 22 on page 79 shows an example of the Network Dispatcher high availability feature. The two machines are synchronized. There is one cluster for the HTTP service on port 80. Every server has an alias for the cluster IP address on the loopback interface. The standby machine has the same configuration file as the primary one (apart from the nonforwarding address)

and is continually updated. It does not route packets. It has an IP cluster alias to the loopback interface, and the primary Network Dispatcher has an IP cluster alias to a network interface (tr0, en0, etc.).

When a client sends a request to the HTTP service, the Network Dispatcher selects the best server that responds directly to the client.

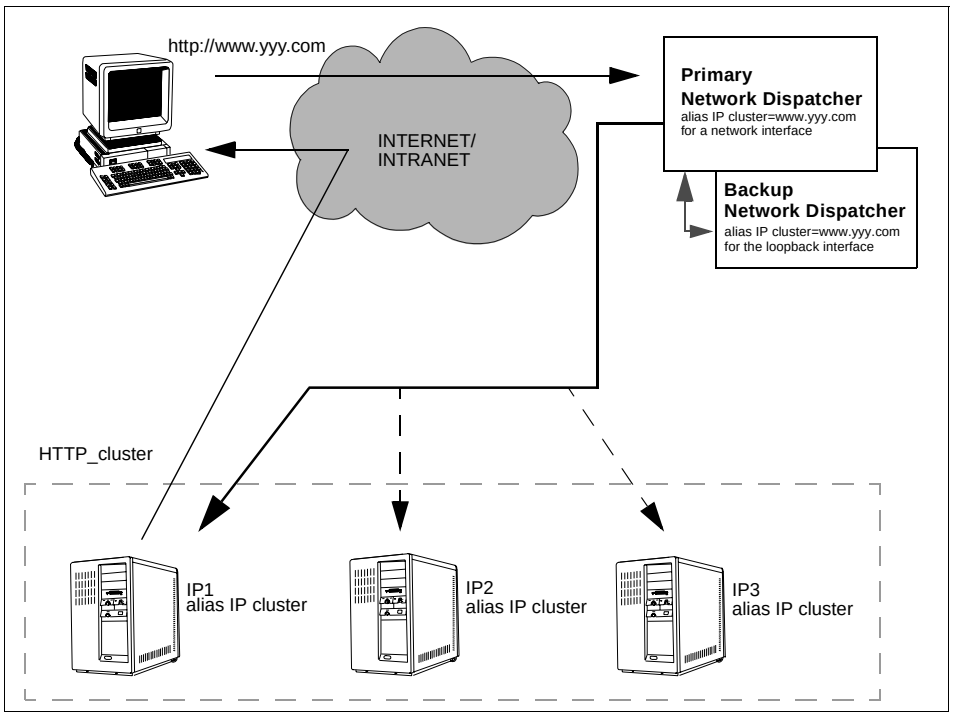


Figure 22. Network Dispatcher High-Availability Feature

8.3.1 Configuring Network Dispatcher High Availability

Both the *primary* and the *backup* machines must have their own nonforwarding address for remote administration and configuration. In the primary machine, for every address, there is a corresponding alias for a network interface (tr0, en0, and so forth), but the clusters' addresses in the backup machine must be aliased to their `loopback` interface.

All the processes (Executor, Manager and Advisors, if you are planning to use them) must run in both machines. All the necessary configuration files regarding cluster, port and server information must reside in both machines. Then you let the machines know about the availability configuration with the command:

`ndcontrol highavailability heartbeat add` *sourceaddress destinationaddress*

Of course, the *sourceaddress* and *destinationaddress* (IP addresses or DNS names of the primary and backup machine) will be reversed on the primary and backup machines.

Then you have to declare for each machine whether it will be the primary or backup by using the following command:

`ndcontrol highavailability backup add primary` [`auto` | `manual`] *portnum*

on the *primary* machine, and the command:

`ndcontrol highavailability backup add backup` [`auto` | `manual`] *portnum*

on the *backup* machine. The *portnum* field indicates the port over which the two machines will communicate. If you set the `auto` option, as soon as the primary machine becomes operational again after the takeover, it will become the active one automatically. If you set the `manual` option, the backup machine that takes over after the failure of the primary should remain active and continues to route packets even if the primary becomes operational again.

Auto Option

If, after a takeover event, the `auto` option does not work well, you should again run the commands:

`ndcontrol highavailability backup add primary auto` *`portnumber`*

`ndcontrol highavailability backup add backup auto` *`portnumber`*

on the *primary* and *backup* machines.

The `manual` option has the advantage that you can run the following command:

`ndcontrol highavailability takeover`

whenever you want to make the standby machine become the active machine.

Manual Option

During our tests, when the active machine failed, the backup took over immediately without waiting for a manual intervention. If we wanted the backup machine to go on routing the IP packets even after the primary was operational again, we had to run the takeover command.

If you want to change some configuration parameters, you must run the `ndcontrol` command first in the backup machine and then in the primary.

To delete a high availability configuration, you need to stop the Executor on one machine; then delete all the high availability definitions on the other one.

Like HACMP/6000, you need some scripts to make the high availability configuration work. These reside in the /usr/lpp/ind/nd/bin directory and are named:

- goActive script

  This script deletes every loopback alias and adds device aliases. It will be executed when a Network Dispatcher machine planned to be the primary machine in a high availability configuration goes into active state and begins routing packets.

- goStandby script

  This script deletes every device alias and adds loopback aliases. It will be executed when a Network Dispatcher machine planned to be the backup machine in a high availability configuration goes into stand-by state without routing packets.

- goInOp script

  This script deletes all device and loopback aliases. This script is executed when a Network Dispatcher Executor is stopped and before it is started for the first time.

8.4 Testing Environment

To test the high-availability feature, we used a 5-nodes RS/6000 SP. Node f01n03t was the primary Network Dispatcher; node f01n01t was the backup machine, and the other three nodes were configured as servers listening on port 80 for an HTTP cluster.

Configuration Hint

One of the default ports used by Network Dispatcher (port 10003) is also used by one of the SP daemons. So, if you use the Network Dispatcher server as one SP node, do not forget to modify this port number substituting the 10003 with another free port in the ND_PORT line of the /usr/bin/ndcontrol and /usr/bin/ndserver files.

We changed the port from 10003 to 50003.

You can find the configuration files we used on Chapter 10, “Configuration Files - Samples” on page 93.

The environment is the same as the one described in Figure 22 on page 79. We started all the daemons in both the machines, wrote the goActive, goInOp and goStandby files and set up the `auto` option. We tested the environment with WebStone, simulating a load of 200 clients asking for dynamic WWW pages. Node *f01n03t* was operational until we issued the following command:

```
ifconfig tr0 down
```

to bring the token-ring interface down. As is shown in Figure 23 on page 83, node *f01n01t* took over packet-routing immediately. The PTX window of node *f01n03t* finished working as soon as the interface went down. That is why the time of the PTX window is different from the others. From the end-user point of view, nothing has changed. The network traffic in node *f01n01t* before the takeover was also due to the communication session that the two machines established in order to be synchronized all the time.

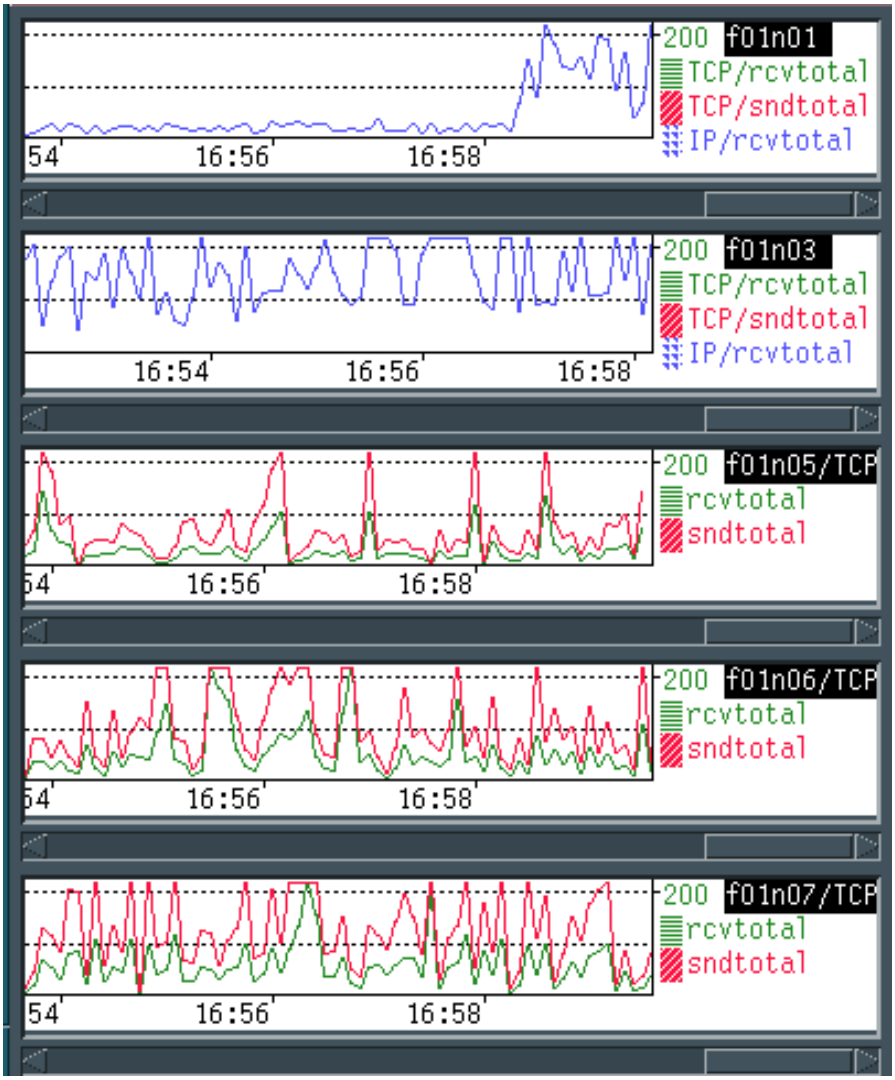


Figure 23. Network Traffic with a High-Availability Configuration

As soon as the primary machine failed, the loopback IP alias to the cluster address configured in the backup machine passed to the token-ring interface as written in the goActive file. When the primary machine became operational again, the alias returned to the loopback interface as written in the goStandby script.

Figure 24 on page 84 shows the CPU behavior for the same test. The CPU load on the servers is heavy because we used dynamic WWW pages. You can see the higher CPU load in node *f01n01t* as soon as it began actively routing packets.

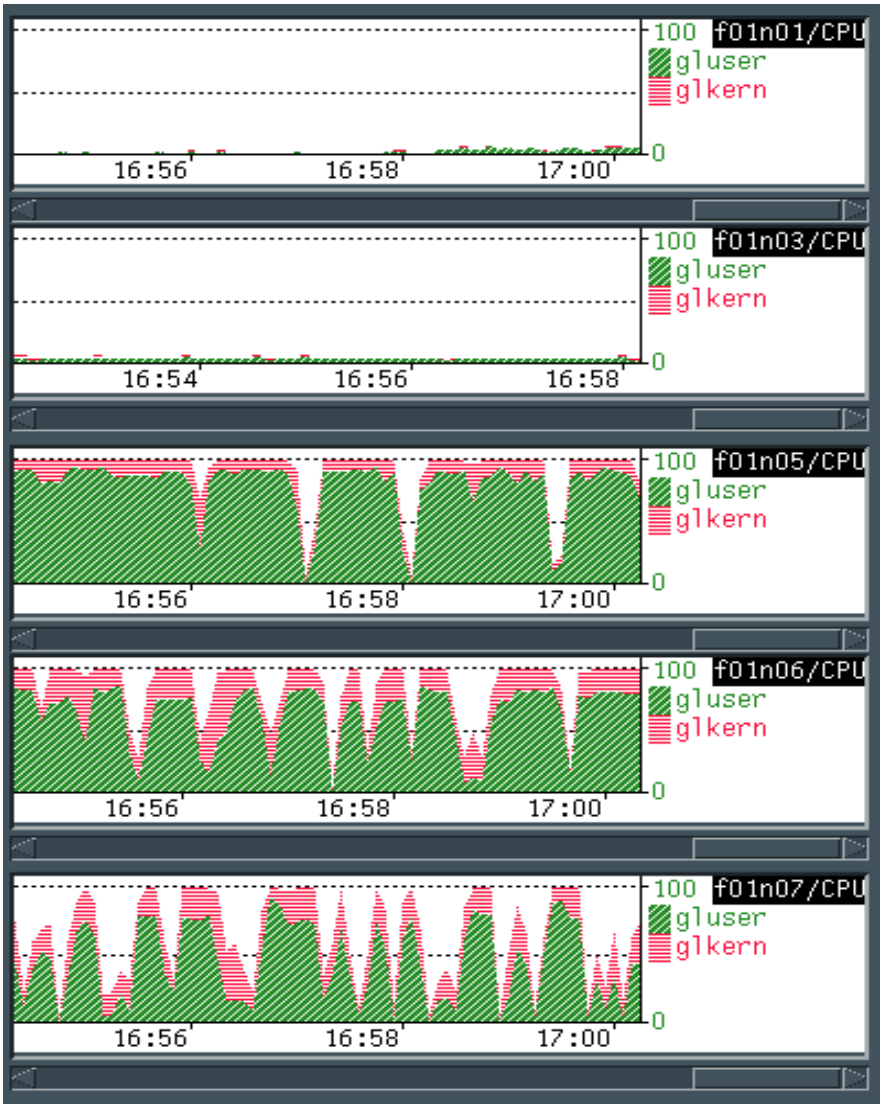


Figure 24. CPU Load with a High-Availability Configuration

8.5 Conclusion

The new high availability feature of Network Dispatcher has the great advantage that is much simpler to configure than HACMP/6000.

If you decide to use the `auto` strategy, you should make sure, after a takeover event, that when the primary machine becomes operational again, it becomes active, as well. If you decide to use the `manual` strategy and want the *backup* machine to remain active after a takeover event, and the *primary* to wait in standby mode, you should run the takeover command from the *backup* machine.

Chapter 9. Two-Tier Network Scenario

Sometimes having just one site to store and manage your information is simply not enough. The load over one site may be so high that it can not be handled. Besides the load, you may have other requirements for having more than one site, such as the need to have replicas of your information over different geographical areas for safety reasons. In case of a catastrophic event at one site, your sites in different locations can continue providing the service. This will also increase your general serviceability, allowing you to shut down a site for maintenance while the others are processing the incoming requests.

ISS in conjunction with Network Dispatcher allows you to create a two-tiered configuration for a cluster of servers located on a remote network. The first tier, or front-end tier, is composed of a machine that runs the `issd` monitor. This machine manages the front-ends of the second tier, which are Network Dispatcher machines. After receiving a request, the `issd` monitor machine will send back the IP address of the Network Dispatcher machine that will handle this particular request. The decision as to which IP address is sent back can be as simple as a round-robin selection, or it can depend on the feedback of `issd` agents running on the Network Dispatcher's machines that measure the load for each specific cluster. Network Dispatchers are the front-end of the second, or back-end, tier. They will route the request to the most appropriate server on that cluster at that moment.

The process is illustrated in Figure 25 on page 88. In this example, a client is trying to access an HTTP server with the name www.yyy.com. At step 1, the client lookup request is attended to by the `issd` monitor machine. If the `issd` daemon is not replacing the name server, the `named` daemon must be running in that machine. The `issd` daemon determines which Network Dispatcher cluster is the least-loaded at this moment, and at step 2, it sends this IP address back to the client. At step 3, the client sends the HTTP request to that IP address. Network Dispatcher receives the request and routes it (step 4) to the least-loaded server in the cluster at that moment. The server gets the request and responds directly to the client at step 5. The client has been redirected to the least-loaded server in the least-loaded cluster without ever noticing. Network Dispatcher and the cluster of machines it is managing need to be on the same network, but each of these clusters can be on any network around the world. The `issd` monitor machine can also be placed on any remote network.

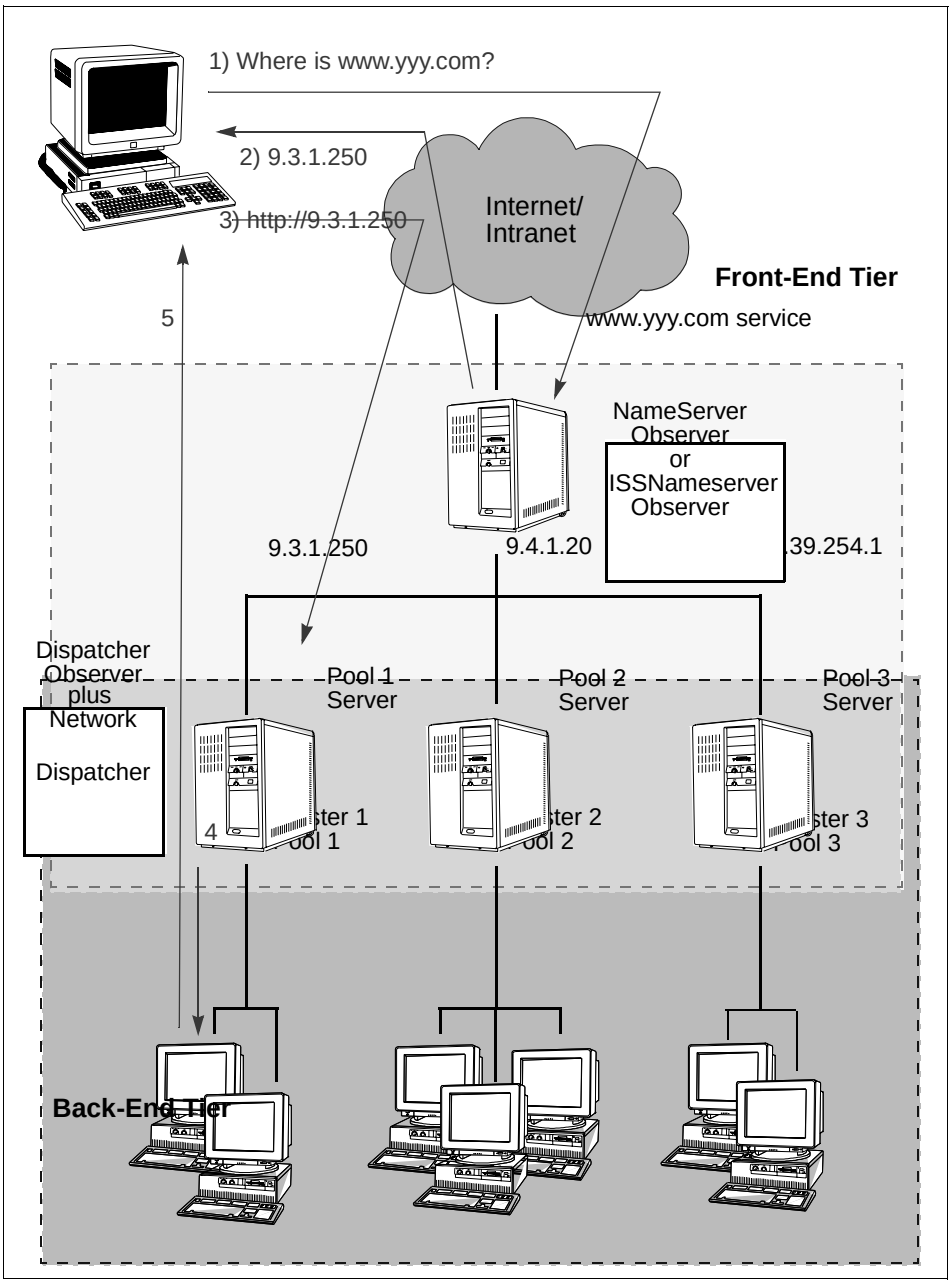


Figure 25. Redirecting Requests on a Two-Tier Configuration of Clusters

If you want a cluster to also use ISS to provide feedback to Network Dispatcher, you can define as an `issd` monitor one of the servers, the Network Dispatcher machine, or a different machine.

9.1 Front-End Tier Configuration

On a two-tier configuration, the `issd` running in monitor mode can have two functions. On the front-end tier, you must have it running. ISS is the first interface the clients get when sending a request into a two-tier network. ISS in the first tier has to be configured either to work with a `NameServer` Observer or with an `ISSNameserver` Observer. For the second tier, ISS can optionally be configured to work with a `Dispatcher` Observer to provide load feedback to Network Dispatcher.

The ISS configuration for a two-tier network is similar to the normal configuration. Assuming that you also want to use ISS to provide feedback from the servers to the Network Dispatcher, you should define several services, each one corresponding to the IP address of one cluster. Then you define the service provided by the `issd` monitor machine of the first tier. To define the set of Network Dispatcher machines to which this service is going to forward the request, use the `NodeList` keyword followed by the IP addresses of the clusters that you are planning to use in the back-end tier. There are no requirements for machines to be within the same Internet subnet or in the same DNS domain; all they need is a valid IP address accessible from the client and from the name server machine. For example, to define the service shown in Figure 25 on page 88, you can have the following lines on the ISS configuration file:

- To define, for example, one cluster HTTP service:

  ```
  Service http_dispatcher  pool1 9.3.1.250  80
  ```

  This service will be used by the `Dispatcher` Observer of this cluster to provide feedback regarding the load of the servers.

- To define the front-end tier `issd` monitor:

  ```
  Service  front_tier  www.yyy.com
  NodeList  9.3.1.250 9.4.1.20  9.39.254.1
  ```

- To define a `Dispatcher` Observer:

  ```
  Dispatcher  9.3.1.250  10004
  ServiceList http_dispatcher
  ```

  The `10004` port is the default port used by the `issd` agents and `issd` monitor to communicate. You can choose another one.

Another required keyword is `HeartbeatInterval`. It defines the interval of time in seconds in which ISS checks the cluster currently ranked at the top. To indicate the number of heart beat intervals, ISS waits before determining. If an update to the `named` daemon is required, use the `HeartbeatsPerUpdate` keyword.

If you use a `RoundRobin SelectionMethod`, the requests will be rotated among the clusters of the top-tier service. The time interval in which a cluster receives all the request is determined by:

`HeartbeatsPerUpdate * HeartbeatInterval`.

Because `RoundRobin` does not measure the load on the clusters, the distribution of work will not be equal most of the time.

The `ResourceType` keyword gives the system administrator greater flexibility in balancing the load between the clusters. In this metric, you specify a command string or a shell script that will return a value indicating how loaded a cluster is.

Figure 26 on page 91 shows the configuration of an ISS working with a `NameServer` Observer. Here the `issd` monitor receives input from the `issd` agents running on the Network Dispatcher machines. The `issd` monitor machine can be the Network Dispatcher machine, a server or another machine outside the cluster.

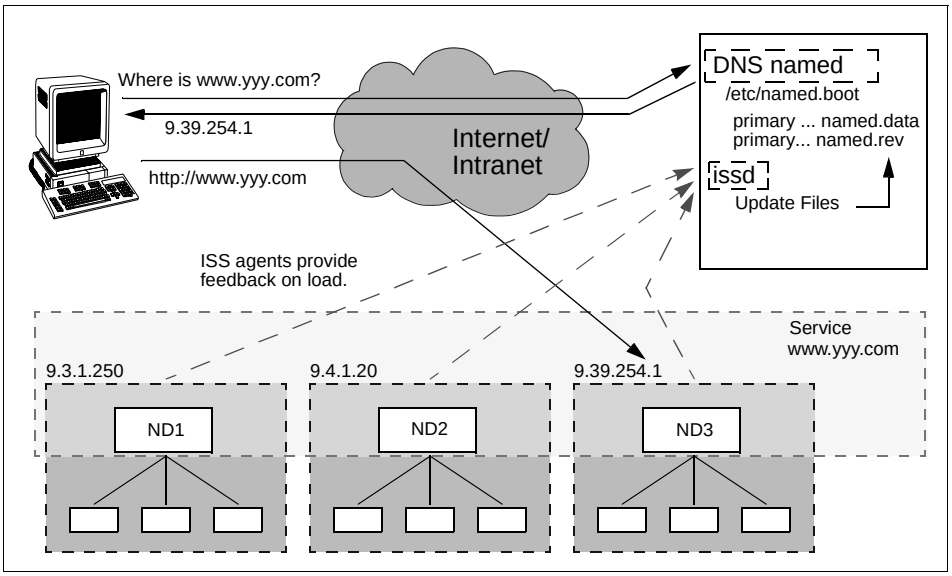


Figure 26. ISS Front-Tier Configuration

9.2 Back-End Tier Configuration

The configuration of Network Dispatcher on the back-end tier does not change in respect to a normal Network Dispatcher configuration. The only difference you have is that the `issd` agent has to run on the Network Dispatcher machine.

A Network Dispatcher machine can manage several clusters belonging to one or more services on the front-end tier. Each server can belong to one or more clusters. The cluster can be defined in terms of IP address (for example www.yyy.com and www.test.com), services offered (HTTP, FTP, and so on) or both. ISS and Network Dispatcher on a two-tier network give you the possibility of having many configuration options, and you can choose the one that best suits your needs. For example, Figure 27 on page 92 shows a two-tier configuration in which ISS manages two front-end services, each one with two Network Dispatchers. Service1 offers HTTP and FTP services for service1.www.com, and Service2 offers just FTP service for service2.www.com. ND1 belongs to Service1 and has two clusters, one for HTTP and the other for FTP. ND2 belongs to both services and has two clusters, one for HTTP and FTP in Service1 and one for FTP in Service2. ND3 has just one FTP cluster. To configure every Network Dispatcher machine, you have to follow the same steps used to configure Network Dispatcher in a

normal configuration. You must also be sure to define the same ports on the clusters that belong to the same service.

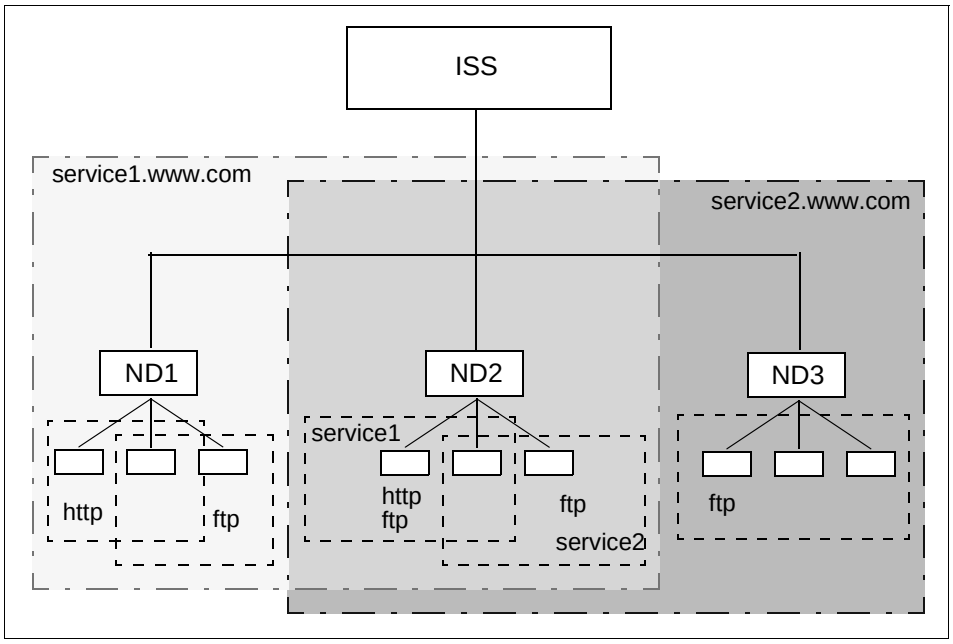


Figure 27. Different Configuration Option for Network Dispatcher and ISS

Network Dispatcher can get feedback from ISS based on a defined `ResourceType`. For example, good feedback for HTTP is CPU load; good feedback for FTP is network traffic. You have to decide which one is best for these front-end services when both services are offered.

Chapter 10. Configuration Files - Samples

This chapter presents the configuration files used for our testing, along with the parameters utilized.

10.1 The Test Environment

A 5-node RS/6000 SP was used. It had three wide nodes–nodes 1,3 and 7– and two thin nodes–nodes 5 and 6. Each node had three network interfaces:

- Token-ring: network IP address 9.3.1.0 (from 241 to 247, skipping unused node addresses) mask 255.255.255.0, connected to a private token-ring network (itsc subdomain), interface name f01n0Xt, where X is the node number (1, 3, 5, 6, and 7).
- Ethernet: SP Ethernet Network, network address 9.3.16.0 mask 255.255.255.0, interface name f01n0X.
- Switch tb2 interface (Hi-Performance Switch), network address 9.3.19.0 mask 255.255.255.0), interface name f01n0Xs.

The host name for the nodes was the name associated with the SP Ethernet adapter (f01n0X). The control workstation had two network interfaces: one token-ring (IP address 9.3.1.240 connected to a private token-ring network—itsc subdomain) and one Ethernet (SP Ethernet Network, IP address 9.3.16.240).

Figure 28 on page 94 shows the SP configuration, network topology and IP addresses used.

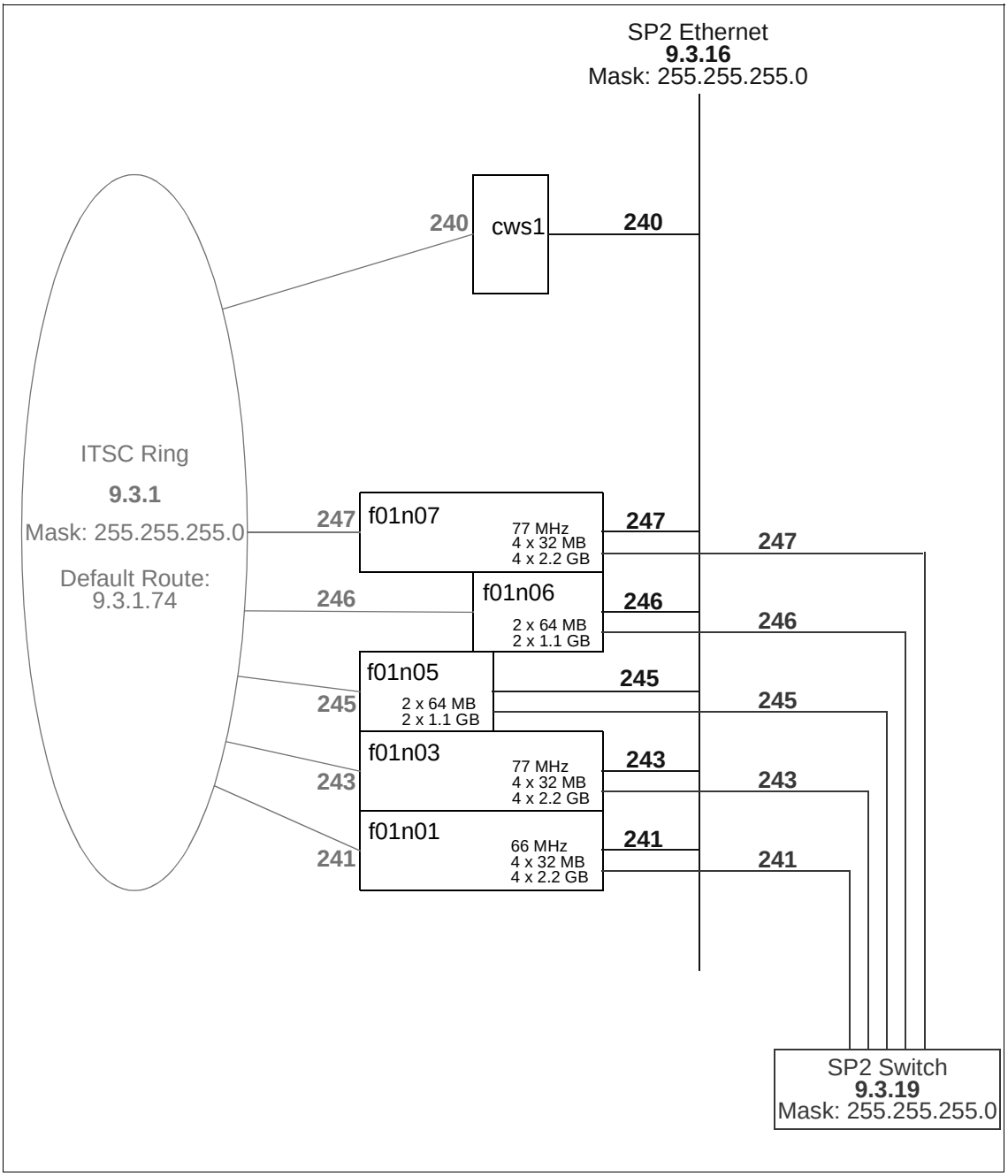


Figure 28. SP Configuration Used for Interactive Network Dispatcher Tests

A DNS domain sp.itsc.austin.ibm.com was used for the SP. Node 1 was the primary DNS server for this domain. The DNS configuration files used were:

- /etc/named.boot

```
;
; type          domain                  source file or host
;
domain          sp.itsc.austin.ibm.com
cache           /etc/named.cache

primary         sp.itsc.austin.ibm.com  /etc/named.data
primary         0.0.127.in-addr.arpa    /etc/named.rev.127.0.0
primary         16.3.9.in-addr.arpa     /etc/named.rev.9.3.16
primary         19.3.9.in-addr.arpa     /etc/named.rev.9.3.19
```

- /etc/named.cache

```
.                               9999999 IN    NS   castor.cdf.ibm.com.
castor.cdf.ibm.com.             9999999 IN    A    9.78.1.2
```

- /etc/named.data

```
; name server data file
; (also see /etc/named.boot)
;
; NAME          TTL     CLASS   TYPE    RDATA
;
; setting default domain to "sp.itsc.austin.ibm.com"
;
@               9999999 IN      SOA     f01n01.sp.itsc.austin.ibm.com.
root.f01n01.sp.itsc.austin.ibm.com. (
                                        992611          ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        300 )           ; Minimum
                9999999 IN      NS      f01n01
;
; Control Workstation
;
cws1            9999999 IN      A       9.3.16.240
cw              9999999 IN      CNAME   cws1
;
; SP Ethernet Addresses
;
f01n01          9999999 IN      A       9.3.16.241
f01n03          9999999 IN      A       9.3.16.243
```

```
f01n05          9999999 IN      A       9.3.16.245
f01n06          9999999 IN      A       9.3.16.246
f01n07          9999999 IN      A       9.3.16.247
;
; Token Ring Addresses
;
cws1t           9999999 IN      A       9.3.1.240
f01n01t         9999999 IN      A       9.3.1.241
f01n03t         9999999 IN      A       9.3.1.243
f01n05t         9999999 IN      A       9.3.1.245
f01n06t         9999999 IN      A       9.3.1.246
f01n07t         9999999 IN      A       9.3.1.247
;
; Switch Addresses
;
f01n01s         9999999 IN      A       9.3.19.241
f01n03s         9999999 IN      A       9.3.19.243
f01n05s         9999999 IN      A       9.3.19.245
f01n06s         9999999 IN      A       9.3.19.246
f01n07s         9999999 IN      A       9.3.19.247
;
; Pool Addresses
;
pool1.sp.itsc.austin.ibm.com. IN      A       9.3.1.243
pool2.sp.itsc.austin.ibm.com. IN      A       9.3.1.250
;
; Alias for the nodes
;
n01             999999 IN       CNAME   f01n01
n03             999999 IN       CNAME   f01n03
n05             999999 IN       CNAME   f01n05
n06             999999 IN       CNAME   f01n06
n07             999999 IN       CNAME   f01n07
n01s            999999 IN       CNAME   f01n01s
n03s            999999 IN       CNAME   f01n03s
n05s            999999 IN       CNAME   f01n05s
n06s            999999 IN       CNAME   f01n06s
n07s            999999 IN       CNAME   f01n07s
n01t            999999 IN       CNAME   f01n01t
n03t            999999 IN       CNAME   f01n03t
n05t            999999 IN       CNAME   f01n05t
n06t            999999 IN       CNAME   f01n06t
n07t            999999 IN       CNAME   f01n07t
```

- /etc/named.rev.127.0.0

```
; setting default domain to ... sp.itsc.austin.ibm.com
@               9999999 IN      SOA     f01n01.sp.itsc.austin.ibm.com.
```

```
root.f01n01.sp.itsc.austin.ibm.com. (
                                        970226          ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        86400 )         ; Minimum
               9999999 IN      NS      f01n01.sp.itsc.austin.ibm.com.
1       IN PTR loopback.sp.itsc.austin.ibm.com.
```

- /etc/named.rev.9.3.16

```
; setting default domain to ... sp.itsc.austin.ibm.com
@               9999999 IN      SOA     f01n01.sp.itsc.austin.ibm.com.
root.f01n01.sp.itsc.austin.ibm.com. (
                                        970226          ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        300 )           ; Minimum
               9999999 IN      NS      f01n01.sp.itsc.austin.ibm.com.
240     IN PTR cws1.sp.itsc.austin.ibm.com.
241     IN PTR f01n01.sp.itsc.austin.ibm.com.
243     IN PTR f01n03.sp.itsc.austin.ibm.com.
245     IN PTR f01n05.sp.itsc.austin.ibm.com.
246     IN PTR f01n06.sp.itsc.austin.ibm.com.
247     IN PTR f01n07.sp.itsc.austin.ibm.com.
```

- /etc/named.rev.9.3.19

```
; setting default domain to ... sp.itsc.austin.ibm.com
@               9999999 IN      SOA     f01n01.sp.itsc.austin.ibm.com.
root.f01n01.sp.itsc.austin.ibm.com. (
                                        970226          ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        300 )           ; Minimum
               9999999 IN      NS      f01n01.sp.itsc.austin.ibm.com.
241     IN PTR f01n01s.sp.itsc.austin.ibm.com.
243     IN PTR f01n03s.sp.itsc.austin.ibm.com.
245     IN PTR f01n05s.sp.itsc.austin.ibm.com.
246     IN PTR f01n06s.sp.itsc.austin.ibm.com.
247     IN PTR f01n07s.sp.itsc.austin.ibm.com.
```

10.2 Network Dispatcher Configuration

As you saw in Figure 28 on page 94, the tests were performed with a 5-node RS/6000 SP machine. Node 1 was configured as the Network Dispatcher

server. All the others were part of the Network Dispatcher pool. The IP address used for the pool was 9.3.1.250, and its name was `pool2.sp.itsc.austin.ibm.com`.

An alias for the token-ring interface of node 1 using the pool IP address was created by the command:

```
ifconfig tr0 alias 9.3.1.250 netmask 255.255.255.0
```

An alias for the loopback interface for all other nodes was created by the command:

```
ifconfig lo0 alias 9.31.250 netmask 255.255.255.0
```

****Configuration Hint****

One of the default ports used by Network Dispatcher (port 10003) is used by one of the SP daemons. So, if you use the Network Dispatcher server as one SP node, do not forget to change this port number by setting (and exporting) the environment variable `ND_MGR_COMMANDPORT` before starting Network Dispatcher execution.

- The following configuration file was used to perform the test in Chapter 5, “WWW Server Scenario” on page 47 and in Chapter 6, “FTP Server Scenario” on page 61.

```
#!/bin/ksh
#
# Make sure the root user is the one executing this script.
#

iam=’whoami’
if [ "$iam" != "root"]
  then
    echo "You must login as root to run this script"
    exit 2
fi

# First we must run the ndserver.
# NOTE: We include the ndserver start.

ndserver start

# First we must start Network Dispatcher

ndcontrol executor start
```

```
#
# The next step in configuring Network Dispatcher is to set the
# NFA or non-forwarding address and to set your CLUSTER address(es).
#
# The non-forwarding address is used to remotely access Network Dispatcher
# machine for whatever administration or configuration purposes. This is
# required because Network Dispatcher will forward requests sent to the
# CLUSTER address.
#
# The CLUSTER address is the hostname (or IP address) to which remote
# clients will connect.
# Anywhere in this file, you may use hostnames and IP addresses
# interchangeably.

NFA=9.3.1.241
CLUSTER=pool2.sp.itsc.austin.ibm.com

echo "Loading the non-forwarding address"
ndcontrol executor set nfa $NFA

# The next step in configuring Network Dispatcher is to create a
# CLUSTER. Network Dispatcher will route the requests sent to the
# CLUSTER address to the corresponding TCP server machines defined
# for that CLUSTER. You can configure and serve multiple CLUSTER
# addresses using Network Dispatcher. Use a similar configuration
# for CLUSTER2, CLUSTER3, etc.

echo "Loading first CLUSTER address "
ndcontrol cluster add $CLUSTER

echo "Creating ports for CLUSTER: $CLUSTER"

ndcontrol port add $CLUSTER:80
ndcontrol port add $CLUSTER:20
ndcontrol port add $CLUSTER:21

# The next step is to add each of the TCP server machines to the ports in
# this CLUSTER.

SERVER1=f01n03t.sp.itsc.austin.ibm.com
SERVER2=f01n05t.sp.itsc.austin.ibm.com
SERVER3=f01n06t.sp.itsc.austin.ibm.com
SERVER4=f01n07t.sp.itsc.austin.ibm.com

echo "Adding TCP server machines"

#
```

```
# server1
#

ndcontrol server add $CLUSTER:80:$SERVER1
ndcontrol server add $CLUSTER:20:$SERVER1
ndcontrol server add $CLUSTER:21:$SERVER1

#
# server2
#

ndcontrol server add $CLUSTER:80:$SERVER2
ndcontrol server add $CLUSTER:20:$SERVER2
ndcontrol server add $CLUSTER:21:$SERVER2

#
# server3
#

ndcontrol server add $CLUSTER:80:$SERVER3
ndcontrol server add $CLUSTER:20:$SERVER3
ndcontrol server add $CLUSTER:21:$SERVER3

#
# server4
#

ndcontrol server add $CLUSTER:80:$SERVER4
ndcontrol server add $CLUSTER:20:$SERVER4
ndcontrol server add $CLUSTER:21:$SERVER4

#
# Now we will start the load-balancing components of Network Dispatcher.
# The main load-balancing component is called the Manager. The Manager
# will update the weight of each of the TCP server machines based on
# three 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. Input from the system level Advisor (iss)
# Each of these policies can be given a weight or importance.

echo "Starting the manager..."
ndcontrol manager start

# give some time for the manager to load
```

```
sleep 4

# The second load-balancing component of NetworkDispatcher is the Advisors.
# The Advisors will give the Manager more insight into the TCP server's
# ability to serve TCP requests (such as HTTP).Currently,NetworkDispatcher
# comes with an HTTP Advisor, an FTP Advisor, and an SSL Advisor.

echo "Starting the http advisor on port 80..."
ndcontrol advisor start http 80
echo "Starting the http advisor on port 21..."
ndcontrol advisor start http 21

# The last step is to set the proportions for each of the policies.
#
# These proportions should be proportioned to add up to 100
# This will give the number of active connections, new
# connections, the advisor and system input the same importance in the
# weight decision.
#
# NOTE: By default the proportions are 50 50 0 0
# Set some parameters for the manager

echo "Setting the proportions..."
ndcontrol manager proportions 50 50 0 0

echo "Setting the smoothing value for the manager..."
ndcontrol manager smoothing 2.5
```

10.3 ISS Configuration

In this section, the ISS configuration files for the `RoundRobin` and `Best SelectionMethod` are shown.

10.3.1 RoundRobin

We used the same SP environment for these tests. Node 1 is for the DNS name server, and it also acts as an ISS monitor and agent.

- The following configuration file was used to perform the test in Chapter 5, "WWW Server Scenario" on page 47 and in Chapter 6, "FTP Server Scenario" on page 61.

```
# This ISS configuration file illustrates how ISS can be used with a DNS
# name server to perform load balancing for one cluster of servers using
# the RoundRobin metric.
#
# Basic parameters of AustinTx1 cell.
```

```
Cell                AustinTx1 local
Authkey             036454DC 65739CB2 23C68DF1 129C4D61
LogLevel            Debug
HeartbeatInterval   30
HeartbeatsPerUpdate 2
PortNumber          11364

# Nodes in AustinTx1 cell

# The node monitor1 has ISS monitor priority 1 and will run the issd
# process in ISS monitor mode. The node monitor2 has monitor priority 2
# and can run as a backup ISS monitor when monitor1 is down. The other
# nodes are declared NotMonitor. The issd processes on these nodes will
# run as ISS agents or perform other special ISS functions. They cannot
# assume the role of an ISS monitor.

Node        f01n01t  1
InternalNet f01n01
Node        f01n03t  2
NotMonitor
InternalNet f01n03
Node        f01n05t  3
NotMonitor
InternalNet f01n05
Node        f01n06t  11
NotMonitor
InternalNet f01n06
Node        f01n07t  12
NotMonitor
InternalNet f01n07

# Definition of "ProcessCount" ResourceType

# The metric associated with ProcessCount is assigned a range of (0-250).
# If "ps -fe | wc -l" returns a value greater than 250, the metric is
# assigned the upper limit of 250. When the value of this metric on a node
# is greater than 225, the node is removed from active participation
# in the Service that it is associated with. The node is returned to active
# participation when the metric has a value less than or equal to 200.

ResourceType        ProcessCount
Metric              External   ps -ef | wc -l
MetricNormalization 0   250
MetricLimits        200 225
Policy              Min
```

```
# Definition of Services available in AustinTx1 cell.

# The first service has the DNS name pool1.sp.itsc.austin.ibm.com
# associated with it. Also associated with service are the nodes:f01n03t,
# f01n05t,f01n06t,f01n07t. Since the observer is NameServer, ISS will work
# in conjunction with the DNS name server running on the node f01n01t to
# resolve the DNS name pool1.sp.itsc.austin.ibm.com to the IP address of
# either f01n03t,f01n05t,f01n06t,f01n07t. Since the SelectionMethod is
# RoundRobin, this mapping of the name pool1 to one of the four IP
# addresses of the name pool1 to one of the three IP addresses is done using
# a simple round-robin algorithm. A new address is selected
# once every 60 seconds (HeartbeatInterval*HeartbeatsPerUpdate) and once
# every 20 seconds (HeartbeatInterval) ISS checks that the server whose
# address is currently functional at its ICMP layer. If at
# any time there are no servers available, the name pool1 will be removed
# from the DNS files and the alarm triggered, which in this case makes and
# entry into a file called /tmp/issalarm.log.
#

Service              service  pool1.sp.itsc.austin.ibm.com
NodeList             f01n03t
                     f01n05t
                     f01n06t
                     f01n07t
ResourceList         ProcessCount
SelectionMethod      RoundRobin
Alarm  echo "pool1 ran out of servers at ‘date‘" >> /tmp/issalarm.log

# Defining a NameServer Observer

# The NameServer observer running on the node f01n01t subscribes all four
# services defined in this file. ISS will work with the DNS name server
# running on f01n01t to load balance the cluster of servers.

NameServer           f01n01t   53
ServiceList          service
NamedData            /etc/named.data
NamedRev             /etc/named.rev.9.3.1
```

10.3.2 Best - SelectionMethod

For the `Best SelectMethod` tests, we used the same SP environment we accessed before. Node 1 is for the DNS name server, and it also acts as an ISS monitor and agent.

- The following configuration files was used to perform the test in Chapter 5, “WWW Server Scenario” on page 47.

```
# This ISS configuration file illustrates how ISS can be used with a DNS
# name server to perform load balancing for three clusters of servers. A
# different load balancing method (RoundRobin, Custom, and LoadLeveler) is
# used for each cluster.

# Basic parameters of AustinTx1 cell.

Cell                 AustinTx1 local
Authkey              036454DC 65739CB2 23C68DF1 129C4D61
LogLevel             Debug
HeartbeatInterval    15
HeartbeatsPerUpdate  2
PortNumber           11364

# Nodes in AustinTx1 cell

# The node monitor1 has ISS monitor priority 1 and will run the issd
# process in ISS monitor mode. The node monitor2 has monitor priority 2
# and can run as a backup ISS monitor when monitor1 is down. The other
# nodes are declared NotMonitor. The issd processes on these nodes will
# run as ISS agents or perform other special ISS functions. They can not
# assume the role of an ISS monitor.

Node         f01n01t  1
InternalNet  f01n01
Node         f01n03t  2
NotMonitor
InternalNet  f01n03
Node         f01n05t  3
NotMonitor
InternalNet  f01n05
Node         f01n06t  11
NotMonitor
InternalNet  f01n06
Node         f01n07t  12
NotMonitor
InternalNet  f01n07

# Definition of "CPUIdle3" ResourceType

# On AIX, "vmstat 3 2 | tail -1 | awk '{print $16}'" returns the percentage
# of CPU Idle time during a three-second sampling period. This value should
# be maximized when you want to identify the least loaded server.
# (On Solaris, use: "vmstat 3 2 | tail -1 | awk '{print $22}'").
# When the value of this metric on a node is less than 20, the node
# is removed from active participation in the Service that it is associated
# with. The node is returned to active participation when the metric has a
```

```
# value greater than or equal to 60.

ResourceType          CPUIdle
Metric                External   vmstat 3 2 | tail -1 | awk '{print $16}'
MetricNormalization   0   100
MetricLimits          60  30
Policy                Max

# Definition of Services available in AustinTx1 cell.

# The first service, has the DNS name pool1.sp.itsc.austin.ibm.com
# associated with it. Also associated with service are the nodes:f01n03t,
# f01n05t,f01n06t,f01n07t. Since the observer is NameServer, ISS will work
# in conjunction with the DNS name server running on the node f01n01t to
# resolve the DNS name pool1.sp.itsc.austin.ibm.com to the IP address of
# either f01n03t,f01n05t,f01n06t,f01n07t.Since the SelectionMethod is
# RoundRobin,this mapping is done using a simple round-robin algorithm.
# A new address is selected once every 60 seconds
# (HeartbeatInterval*HeartbeatsPerUpdate) and once
# every 20 seconds (HeartbeatInterval) ISS checks that the server whose
# address is currently functional at its ICMP layer. If at
# any time there are no servers available, the name pool1 will be removed
# from the DNS files and the alarm triggered, which, in this case, makes
# an entry into a file called /tmp/issalarm.log.
#

Service               service  pool1.sp.itsc.austin.ibm.com
NodeList              f01n07t
                      f01n06t
                      f01n05t
                      f01n03t
ResourceList          CPUIdle
SelectionMethod       Best
Alarm  echo "pool1 ran out of servers at ‘date‘" >> /tmp/issalarm.log

# Defining a NameServer Observer
#
# The NameServer observer running on the node f01n01t subscribes all four
# services defined in this file. ISS will work with the DNS name server
# running on f01n01t to load balance the cluster of servers.

NameServer            f01n01t   53
ServiceList           service
NamedData             /etc/named.data
NamedRev              /etc/named.rev.9.3.1
```

10.4 High Availability Configuration

This section shows the configuration script of the high availability function. The result of this test is in Chapter 8, “High Availability Server Scenario” on page 75.

```
#!/bin/ksh
#
# This is the Sample configuration file for Interactive Network Dispatcher
# for High Availability.

#
# Make sure the root user is the one executing this script.
#

iam=‘whoami‘

if [ "$iam" != "root" ]
  then
   echo "You must login as root to run this script"
   exit 2
fi

# First we must run the Ndserver

ndserver start

# We must start Interactive Network Dispatcher

ndcontrol executor start

# The next step in configuring Interactive Network Dispatcher is to set the
# NFA (non-forwarding address) and to set the cluster address(es).
#
# The non-forwarding address is used to remotely access the Interactive
# Network Dispatcher machine for administration or configuration purposes.
# This address is required since Interactive Network Dispatcher will
# forward packets to the cluster address.
#
# The CLUSTER address is the hostname (or IP address) to which remote
# clients will connect.
#
# Anywhere in this file, you may use hostnames and IP addresses
# interchangeably.
# NFA=[non-forwarding address]
# CLUSTER=[your clustername]
```

```
NFA=9.3.1.241
CLUSTER=9.3.1.250

echo "Loading the non-forwarding address"
ndcontrol executor set nfa $NFA

# The next step in configuring Interactive Network Dispatcher is to create
# a CLUSTER. Interactive Network Dispatcher will route requests sent to the
# CLUSTER address to the corresponding server machines defined to that
# CLUSTER. You may configure and serve multiple CLUSTER addresses using
# Interactive Network Dispatcher.
# Use a similar configuration for CLUSTER2, CLUSTER3, etc.

echo "Loading first CLUSTER address "
ndcontrol cluster add $CLUSTER

# Now we must define the ports this CLUSTER will use.
# The most common ports used are 80 for HTTP and 20 and 21 for FTP.
# Any request received by Interactive Network Dispatcher on one of these
# ports will be forwarded to the corresponding port of one of the server
# machines.

echo "Creating ports for CLUSTER: $CLUSTER"

ndcontrol port add $CLUSTER:20
ndcontrol port add $CLUSTER:21
ndcontrol port add $CLUSTER:80

# The last step is to add each of the server machines to the ports in
# this CLUSTER. Again, you can use either the hostname or the IP address of
# the server machines.

SERVER1=f01n05t
SERVER2=f01n06t
SERVER3=f01n07t

echo "Adding server machines"

#
# f01n05t
#

ndcontrol server add $CLUSTER:20:$SERVER1
ndcontrol server add $CLUSTER:21:$SERVER1
ndcontrol server add $CLUSTER:80:$SERVER1

#
```

```
# f01n06t
#

ndcontrol server add $CLUSTER:20:$SERVER2
ndcontrol server add $CLUSTER:21:$SERVER2
ndcontrol server add $CLUSTER:80:$SERVER2

#
# f01n07t
#

ndcontrol server add $CLUSTER:20:$SERVER3
ndcontrol server add $CLUSTER:21:$SERVER3
ndcontrol server add $CLUSTER:80:$SERVER3

# We will now start the load balancing components of Interactive Network
# Dispatcher. The main load balancing component is called the manager. The
# The manager will update the weight of each of the server machines based
# on four 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. Input from the system level advisor (ISS).
# Each of these policies can be given a weight or importance.

echo "Starting the manager..."
ndcontrol manager start

# The second load balancing component of Interactive Network Dispatcher is
# the advisors.The advisors give the manager more insight into the server's
# ability to serve TCP requests (such as HTTP). Interactive Network
# Dispatcher has advisors for FTP, Telnet, SMTP, HTTP, POP3, NNTP and SSL.

echo "Starting the FTP advisor on port 21 ..."
ndcontrol advisor start ftp 21
echo "Starting the HTTP advisor on port 80 ..."
ndcontrol advisor start http 80

# The last step is to set the proportions for each of the policies.
# For example:
#   ndcontrol manager proportions 15 15 35 35
#
# These proportions must be proportioned to add up to 100.
# This will give both the number of active connections and new
# connections 15% input in the weighting process, while the advisor
# and system inputs will contribute 35% each to the weight decision.
#
```

```
# NOTE: By default the proportions are 50 50 0 0

echo "Setting the proportions..."
ndcontrol manager proportions 50 50 0 0
echo "Setting the smoothing value for the manager..."
ndcontrol manager smoothing 1.0

# The following commands are set to the default values.
# Use these commands to guide to change from the defaults.

#  ndcontrol manager loglevel    1
#  ndcontrol manager logsize     unlimited
#  ndcontrol manager sensitivity 5.000000
#  ndcontrol manager interval    2
#  ndcontrol manager refresh     2

#  ndcontrol advisor interval ftp  21  5
#  ndcontrol advisor loglevel ftp  21  1
#  ndcontrol advisor logsize  ftp  21  unlimited
#  ndcontrol advisor timeout  ftp  21  unlimited
#  ndcontrol advisor interval telnet 23 5
#  ndcontrol advisor loglevel telnet 23 1
#
# These proportions must be proportioned to add up to 100.
# This will give both the number of active connections and new
# connections 15% input in the weighting process, while the advisor
# and system inputs will contribute 35% each to the weight decision.
#
# NOTE: By default the proportions are 50 50 0 0

echo "Setting the proportions..."
ndcontrol manager proportions 50 50 0 0
echo "Setting the smoothing value for the manager..."
ndcontrol manager smoothing 1.0
```

Appendix A. Minimal Configuration

Sometimes it may be necessary to test or to demonstrate ISS and Network Dispatcher in a restricted environment, with very few machines. This appendix shows the minimal configuration needed to do that.

If you need further information, you may refer to Chapter 10, “Configuration Files - Samples” on page 93, and to Chapter 4, “Installation and Operation” on page 39.

A.1 ISS

For ISS, the minimal configuration depends on how you want ISS to balance the load on the servers. This session focuses on the use of ISS using the `RoundRobin` metric and custom metric configurations.

A.1.1 RoundRobin Metrics

When using ISS with the `RoundRobin` metric, it is possible to use just one machine with one network interface to implement an ISS server. Using this implementation, the ISS server machine will also be the pool. To do this, it is necessary to create several aliases for the existing network interfaces. These aliases will be the pool ISS will use. An example of this configuration is shown on Figure 29.

It is preferable that all of the alias addresses are on the same network as the main network interface address. If this is not the case, it is necessary to create routes pointing to the main network interface as the route for addresses on the other networks.

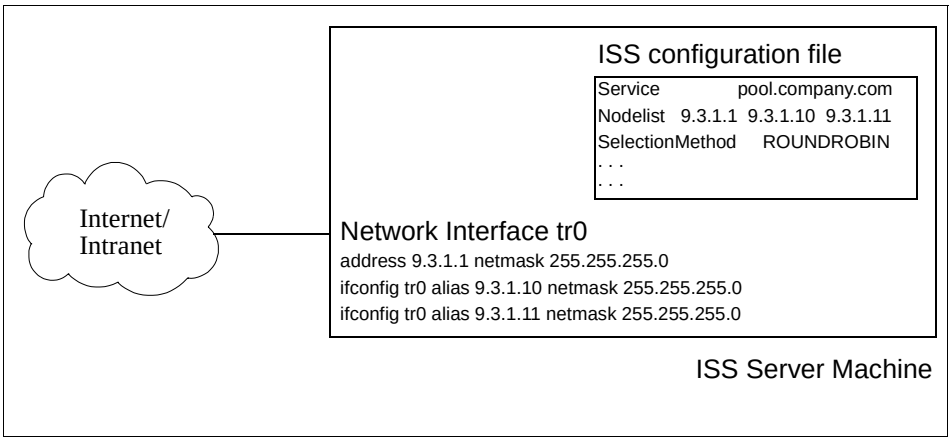


Figure 29. Example of Minimum Configuration for ISS in Round-Robin

A.1.2 Custom Metrics

For custom metrics, this one-machine configuration is not possible because ISS tries to start an instance of ISS for each machine in the pool. Since all the addresses of the pool would be in just one machine, the ISS (monitor) would try to start several instances of ISS on the same server, which is not possible.

So, the minimum configuration that can be used for ISS with a custom metric is a two-machine configuration, each with one network interface. One machine would be the ISS server and a part of the pool of two machines. This configuration was used for the basic tests with ISS. See 4.1, “Interactive Network Dispatcher User’s Guide” on page 39.

A.2 Network Dispatcher

For Network Dispatcher, the minimum configuration is also two servers, each of which has one network interface.

When we tried to test using just one machine with several network interface aliases, the execution of the configuration script was fine. But when we tried to access the pool address and the selected server was the alias address, the machine hung, and a reboot was necessary.

The configuration of these two machines with one network interface each was used for the basic tests with Network Dispatcher, and the document containing the configuration is described in 4.1, “Interactive Network Dispatcher User’s Guide” on page 39.

Appendix B. Domain Name System (DNS)

Every machine in a network needs a name and a address. Although 32-bit Internet addresses provide machines an efficient means of identifying the source and destination of datagrams sent across a network, users prefer meaningful, easily remembered names. TCP/IP provides a naming system that supports both flat and hierarchical network organizations.

This appendix explains some key elements to understand how DNS–the Domain Name System–works.

B.1 Naming

Naming in flat networks is very simple: Host names consist of a single set of characters and generally are administered locally. In flat TCP/IP networks, each machine on the network has a file (/etc/hosts) containing the name-to-Internet-address mapping information for every host on the network. As a TCP/IP network grows, the administrative duty of keeping each machine's naming file up-to-date grows. When TCP/IP networks become very large, as in the Internet, naming is divided hierarchically. Typically, the divisions follow the network's organization. In TCP/IP, hierarchical naming is known as the domain name system (DNS), and it uses the Domain protocol. The Domain protocol is implemented by the `named` daemon in TCP/IP.

As in naming for flat networks, the domain name hierarchy provides for the assignment of symbolic names to networks and hosts that are meaningful and easy for users to remember. However, instead of each machine on the network keeping a file containing the name-to-address mapping for all other hosts on the network, one or more hosts are selected to function as name servers. Name servers translate (resolve) symbolic names assigned to networks and hosts into the efficient Internet addresses used by machines. A name server has complete information about some part of the domain, referred to as a zone, and it has authority for its zone.

B.2 Naming Authority

In a flat network, all hosts in the network are administered by a central authority. This form of network requires that all hosts in the network have unique host names. In a large network, this requirement creates a large administrative burden on the central authority.

In a domain network, groups of hosts are administered separately within a tree-structured hierarchy of domains and subdomains. In this case, host names need to be unique only within the local domain, and only the root domain is administered by a central authority. This structure allows subdomains to be administered locally and reduces the burden on the central authority. For example, the root domain of the Internet consists of such domains as COM (commercial organizations), EDU (educational organizations), GOV (governmental organizations), and MIL (military groups). New top-level domains can only be added by the central authority. Naming at the second level is delegated to designated agents within the respective domains. An example of a domain structure is shown in Figure 30.

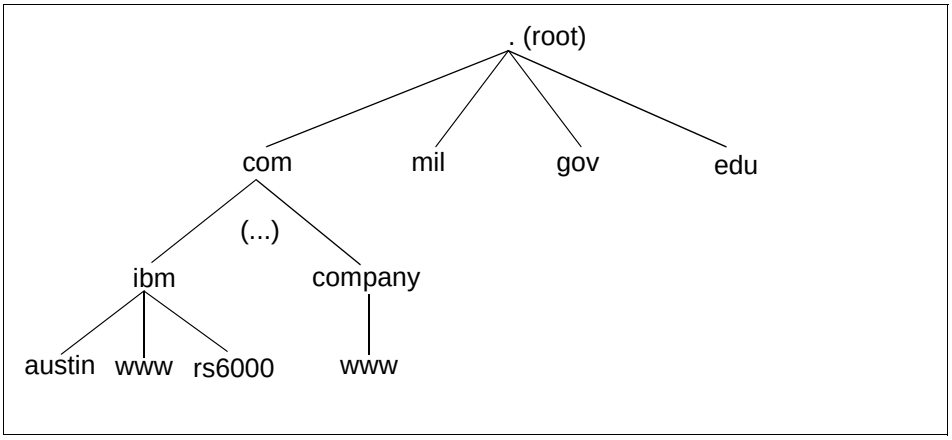


Figure 30. Example of Domain Structure

B.3 Naming Conventions

In the hierarchical domain name system, names consist of a sequence of case-insensitive subnames separated by periods, with no embedded blanks. The Domain protocol specifies that a local domain name must be less than 64 characters and that a host name must be less than 32 characters in length. The host name is given first, followed by a . (period), a series of local domain names separated by periods, and finally the root domain. A fully specified domain name for a host, including periods, must be less than 255 characters in length and in the following form:

host.subdomain1.[subdomain2 . . . subdomainN].rootdomain

Since host names must be unique within a domain, you can use an abbreviated name when sending messages to a host within the same domain.

For example, instead of sending a message to `smith.eng.lsu.edu`, a host in the `eng.lsu.edu` domain could send a message to `smith`. Additionally, each host can have several aliases that other hosts can use when sending messages.

B.4 Name Servers

In a flat name space, all names must be kept in the /etc/hosts file on each host on the network. If the network is very large, this can become a burden on the resources of each machine.

In a hierarchical network, certain hosts designated as name servers resolve names into Internet addresses for other hosts. This has two advantages over the flat name space: It keeps the resources of each host on the network from being tied up in resolving names, and it keeps the person who manages the system from having to maintain name resolution files on each machine on the network. The set of names managed by a single name server is known as its *zone of authority*.

Note: Although the host machine that performs the name-resolution function for a zone of authority is commonly referred to as a *name server host*, the process controlling the function, the `named` daemon, is the actual name server process.

To further reduce unnecessary network activity, all name servers cache (store for a period of time) name-to-address mappings. When a client asks a server to resolve a name, the server checks its cache first to see if the name has been resolved recently. Since domain and host names do change, each item remains in the cache for a limited length of time specified by the record's time to live (TTL). In this way, authorities can specify how long they expect the name resolution to be accurate.

Within any autonomous system, there can be multiple name servers. Typically, name servers are organized hierarchically and correspond to the network's organization. Each domain might have a name server responsible for all subdomains within the domain. Each subdomain name server communicates with the name server of the domain above it (called the *parent name server*), as well as with the name servers of other subdomains.

There are several types of name servers:

- Primary Name Server: loads its data from a file or disk and may delegate authority to other servers in its domain.
- Secondary Name Server: receives its information at system startup time for the given zone of authority from a primary name server, and then periodically asks the primary server to update its information. On expiration of the *refresh* value in the start of authority (SOA) resource record on a secondary name server, the secondary will reload the database from the primary if the serial number of the database on the primary is greater than the serial number in the current database on the secondary. To force a new zone transfer from the primary, simply remove the existing secondary databases and refresh the `named` daemon on the secondary name server.
- Caching-Only Server: indicates that the caching-only server responds to queries by asking other servers that have the authority to provide the information needed if a caching-only server does not have a name-to-address mapping in its cache.
- Forwarder or Client Server: forwards queries it cannot satisfy locally to a fixed list of forwarding servers. Slave forwarders (a forwarder which obtains information and passes it on to other clients, but which is not actually a server) will not interact with the primary name servers for the root domain and other domains. The queries to the forwarding servers are recursive. There may be one or more forwarding servers, which are tried sequentially until the list is exhausted. A client and forwarder configuration is typically used when you do not wish all the servers at a given site to interact with the rest of the Internet servers.
- Remote Server: runs all the network programs that use the name server without the name server process running on the local host. All queries are serviced by a name server that is running on another machine on the network.

One name server host can perform in different capacities for different zones of authority. For example, a single name-server host can be a primary name server for one zone and a secondary name server for another zone.

B.5 Planning for Domain Name Resolution

If you are part of a large network, you need to coordinate setting up your domain and name servers with their central authority.

Some hints in planning your own domain name resolution system:

- Plan ahead.
- Changing a name is much more difficult than setting up the initial one. Obtain consensus from your organization on network, gateway, name server, and host names before you set up your files.
- Set up redundant name servers.
- If you cannot set up redundant name servers, be sure to set up secondary and cache name servers so you have some type of backup.

B.6 Configuring Name Servers

In a hierarchical network, certain hosts are designated as name servers. These hosts resolve names into Internet addresses for other hosts. The `named` daemon controls the name server function and, therefore, must be run on a name server host.

Before you configure a name server, decide which type or types best fit the network it will serve. There are several types of name servers.

A primary name server actually stores the database containing name-to-address mapping information. It loads its data from a file or disk and may delegate authority to other servers in its domain. A secondary name server receives its information at system startup time for a given zone of authority from a primary name server, and then periodically asks the primary server to update its information. A caching-only server responds to requests to resolve names by querying other servers that have the authority to provide the information needed.

Keep in mind that a name server may function in different capacities for different *zones of authority*. For example, one name server host can be a primary name server for one zone and a secondary name server for another zone. If your system has the Network Information Service (NIS) application installed, NIS can also provide name server service.

There are several files involved in configuring name servers.

- boot: This file is read when the named daemon starts. The records in the boot file tell the `named` daemon which type of server it is, which domains it has authority over (its zones of authority), and where to get the data for initially setting up its database. The default name of this file is /etc/named.boot. However, you can change the name of this file by specifying the name and path of the file on the command line when the `named` daemon is started. When using the /etc/named.boot as the boot file, if it does not exist, a message is generated in syslog and `named` terminates.

However, if an alternative boot file is specified, and the alternative file does not exist, no error message will be generated and named will continue.

- cache: contains information about the local cache. The local cache file contains the names and addresses of the highest authority name servers in the network. The cache file uses the standard resource record format. The name of the cache file is set in the boot file.
- domain data: there are three domain data files, also referred to as the named *data* files. The named *local* file contains the address resolution information for local loopback. The named *data* file contains the address resolution data for all machines in the name server's zone of authority. The named *reverse data* file contains the reverse address-resolution information for all machines in the name server's zone of authority. The domain data files use the standard resource record format. Their names are user-definable and are set in the boot file. By convention, the names of these files generally include the name of the daemon (named), and in the extension, the type of file and name of the domain. For example, the name server for the domain abc might have the following files: named.abcdata, named.abcrev and named.abclocal. When modifying the named data files the serial number in the SOA resource record should be increased.
- resolv.conf: The presence of this file indicates to a host that it should go to a name server to resolve a name first. If the resolv.conf file does not exist, the host looks in the /etc/hosts file for name resolution. On a name server, the resolv.conf file must exist and can contain the local host's address, the loopback address (127.0.0.1), or be empty.

Time-to-live (TTL) is specified in resource records. If TTL is not specified in a record, the length of this time period defaults to the minimum field as defined in the start of authority (SOA) record for that zone. TTL is used when data is stored outside a zone (in a cache) to ensure that the data does not stay around indefinitely.

B.6.1 Configuring a Primary Name Server

The following steps describe how to configure a primary name server by editing a series of files and then using the System Management Interface Tool (SMIT) or the command line to start the named daemon.

1. Edit the /etc/named.boot file. If there is no named.boot file in the /etc directory, copy the /usr/samples/tcpip/named.boot sample file into the /etc directory and edit it. This file is read each time the named daemon starts. It tells the server which type of server it is, the zone for which it is responsible, and where to get its initial information.

- Specify the directory in which the named data files can be located (optional). Use this line if you want the named data files to use paths relative to this directory. For example:

  ```
  directory /usr/local/domain
  ```

- Define the name of the default domain for the name server. For example:

  ```
  domain abc.aus.century.com
  ```

- Optionally, specify the name of the cache file. For example:

  ```
  cache . /etc/named.ca
  ```

- Specify a name server type of primary, and define the names of the named hosts data file and named reverse hosts data file. For example:

  ```
  primary abc.aus.century.com    /etc/named.abcdata
  primary 201.9.192.in-addr.arpa /etc/named.abcrev
  ```

  Note: Include lines for each zone for which the name server is primary.

- Define the name of the named local file. For example:

  ```
  primary 0.0.127.in-addr.arpa /etc/named.local
  ```

2. Edit the /etc/named.ca file. This file contains the addresses of the servers that are authoritative or that are root name servers for the domain. For example:

   ```
   ; root name servers.
   .                          IN NS relay.century.com.
   relay.century.com. 3600000 IN A 129.114.1.2
   ```

   Note: All lines in this file must be in standard resource record format.

3. Edit the /etc/named.local file.

   - Specify the start of authority of the zone and the default time-to-live information. For example:

     ```
     @ IN SOA venus.abc.aus.century.com. gail.zeus.abc.aus.century.com.
             ( 1.1 ;serial
              3600 ;refresh
               600 ;retry
           3600000 ;expire
             86400);minimum
     ```

   - Specify the name server (NS) record. For example:

     ```
     IN NS venus.abc.aus.century.com.
     ```

   - Specify the pointer (PTR) record.

     ```
     1 IN PTR localhost.
     ```

Note: All lines in this file must be in standard resource record format.

4. Edit the /etc/named.data file. The /usr/samples/tcpip/hosts.awk file contains directions for creating the /etc/named.data file. Use the /usr/samples/tcpip/named.data sample file as an example when creating the /etc/named.data file.

   - Specify the start of authority of the zone and the default time-to-live information for the zone. This record designates the start of a zone. There should only be one start of authority record per zone. For example:

     ```
     @ IN SOA venus bob.robert.abc.aus.century.com.
             ( 1.1 ;serial
              3600 ;refresh
               600 ;retry
           3600000 ;expire
            86400) ;minimum
     ```

   - Include name-to-address resolution information on all hosts in the name server's zone of authority. For example:

     ```
     venus  IN A 192.9.201.1
     kronos IN A 192.9.201.2
     mars   IN A 192.9.201.3
     ```

   - Include name server records for all primary name servers in the zone. For example:

     ```
     IN NS venus.abc.century.com
     IN NS kronos.xyz.century.com
     ```

   - Include other types of entries, such as canonical name records and mail exchanger records as needed.

     Note: All lines in this file must be in standard resource record format.

5. Edit the /etc/named.rev file. The /usr/samples/tcpip/addrs.awk file contains directions for creating the /etc/named.rev file.

   - Specify the start of authority (SOA) of the zone and the default time-to-live information. This record designates the start of a zone. There should only be one start of authority (SOA) record per zone. For example:

     ```
     @ IN SOA venus.abc.aus.century.com. bob.robert.abc.aus.century.com.
             ( 1.1 ;serial
              3600 ;refresh
               600 ;retry
           3600000 ;expire
            86400) ;minimum
     ```

- Include address to name resolution information on all hosts to be in the name server's zone of authority.

  ```
  ;ABC.AUS.CENTURY.COM Hosts
  1 IN PTR venus.abc.aus.century.com.
  2 IN PTR kronos.abc.aus.century.com.
  3 IN PTR mars.abc.aus.century.com.
  ```

- Include other types of entries, such as name server records and canonical name records (optional).
- Note: All lines in this file must be in standard resource record format.

6. Create an /etc/resolv.conf file by issuing the following command:

   ```
   touch /etc/resolv.conf
   ```

   The presence of this file indicates that the host should use a name server, not the /etc/hosts file, for name resolution. This file must exist on a name server host and may contain either the local host's address, the loopback address (127.0.0.1) or be empty.

   Alternatively, the /etc/resolv.conf file may contain the following entry:

   ```
   nameserver 127.0.0.1
   ```

   The 127.0.0.1 address is the loopback address, which causes the host to access itself as the name server. The /etc/resolv.conf file may also contain an entry like the following:

   ```
   domain domainname
   ```

   In the previous example, `domainname` would be austin.century.com.

7. Perform one of the following steps:

   - Enable the `named` daemon using the following SMIT fastpath:

     ```
     smit stnamed
     ```

     This initializes the daemon with each system startup. Indicate whether you want to start the `named` daemon now, at the next system restart, or both.

   - Edit the /etc/rc.tcpip file. Un-comment the line for the `named` daemon by removing the comment (`#`) symbol from the following line:

     ```
     #start /etc/named "$src_running"
     ```

     This initializes the daemon with each system startup.

8. If you chose not to initialize the `named` daemon through SMIT, start the daemon for this session by issuing the following command:

   ```
   startsrc -s named
   ```

B.7 Delegating Domains to Other Name Servers

To delegate a subdomain to other name servers for normal DNS queries (hostname to IP address conversion), do the following:

1. Create all the data files for the subdomain in the primary name server for the new domain as explained in item B.6.1, “Configuring a Primary Name Server” on page 120. If it is desired for the clients to be able to resolve names outside their domain, don’t forget to include a cache and/or forwarders line in the named boot file.
2. Start the name server daemon on the primary name server for the delegated domain, and test it to see if everything is working correctly.
3. If necessary, set up a secondary name server.
4. Include in the named data file of the main name server for the domain a NS record indicating that queries for the subdomain should be handled by other name servers.

   For example, if your domain is company.com and you wish to create a new subdomain, dept.company.com, and the name server for this domain should be machine.dept.company.com, the following lines must be added to the named data file (/etc/named.data) on the main DNS server for company.com:

   ```
   dept.company.com.          9999999 IN  NS machine.dept.company.com.
   machine.dept.company.com. 9999999 IN  A   10.0.1.100
   ```

5. Refresh the named daemon to the changes in the data files become effective.

In the case of reverse DNS (IP address to hostname conversion), there are two cases:

- If the subdomain will be the owner of a whole range of IP addresses (for example, 9 or 9.179 or 9.179.84), the delegation should be done the same way as explained above. It is necessary just to set up all the files in the subdomain name server and to put an entry in the domain name server reverse data file pointing to the subdomain. In this case, the domain being delegated will be *.in-addr.arpa, where * are the first bytes of the IP address for the range in reverse order.

  Continuing the previous example of domain delegation, to delegate a reverse domain whose network IP address is 10.0.1.0 to a server named machine.dept.company.com, the entry in the reverse data file (/etc/named.rev) of the main domain server for company.com (which is the owner of all the 10 network) would be:

  ```
  1.0.10.in-addr.arpa.     9999999 IN NS machine.dept.company.com.
  ```

- If the subdomain will only have a subset of a range of IP addresses (for example, 9.179.184.100 to 9.179.184.200), the name server of the owner of the range must have the IP address-to-hostname entries for all the subdomains in its reverse name data files because you cannot delegate a part of a domain (or reverse domain) to other servers.

Appendix C. Hints and Tips

This appendix contains some hints and tips for ISS and Network Dispatcher configuration that were learned from experiences in writing this book.

C.1 ISS

- The `issd` must be able to resolve all the names contained in the ISS configuration file. To avoid having problems in case of external DNS failure, put all addresses (qualified and non-qualified) used in the ISS configuration file in the local /etc/hosts file.
- Do not forget to put the full, qualified pool name followed by a period (.) in the DNS named.data file when using `NameServer` Observer. Otherwise `issd` will not find the service name in that file.
- ISS always puts the full reverse DNS address when you use the NamedRev field in the ISS configuration file instead of putting only the part that was not mentioned in the named.boot file.
- When using `Dispatcher` Observer, specify the same server name or IP address in the InternalNet of the Network Dispatcher Node definition of the ISS configuration file as the one specified for the Network Dispatcher cluster.

  For example, if you have the following lines in the ISS configuration file:

  ```
  Node  9.1.1.1
  InternalNet  10.1.1.1
  ```

  the command to use to add this server to a Network Dispatcher cluster should be:

  ```
  ndcontrol server add <cluster_address>:<port>:10.1.1.1
  ```

  If you do not do this, Network Dispatcher will not use any information ISS sends.
- If your round-robin configuration does not work, you should define in the ISS file at least one `ResourceType`, and include it in the `Service` that will use the `RoundRobin SelectionMethod`. The `issd` monitor will not use that `ResourceType`, but it needs it to completely identify the Service. For example:

  ```
  Service www_service www.yyy.com
  NodeList  server1 server2 server3
  ResourceList CPULoad
  SelectionMethod  RoundRobin
  ```

- The hostname of an AIX machine must be set up through SMIT. ISS needs to recognize this hostname or the corresponding IP address. If your machine has more than one network interface (each one with its own DNS name and corresponding IP address) and you do not want to use the one corresponding to your main hostname to provide the service, you should either change your hostname to the one corresponding to the network interface you want to use or maybe define an internal net.
- When testing ISS, the SIGINT signal used to stop the `issd` did not work. We used the SIGTERM signal instead.
- For clean administration of your machines, do not forget to stop the `issd` monitor and agents first if you have to shut down your machines. Moreover, it is a good rule always to have a backup copy of your DNS configuration files.

C.2 Network Dispatcher

- The Network Dispatcher server must be able to resolve all the names contained in the configuration script. To avoid having problems in case of external DNS failure, put all addresses (qualified and non-qualified) used in the Network Dispatcher configuration script in the local */*etc/hosts file.
- It is **always** necessary to create an alias for the network interface that will receive the client requests for the Network Dispatcher server.
- It is **always** necessary to create an alias for the loopback interface for all the servers that will receive packets forwarded by the Network Dispatcher server.
- If you are using address mapping files, when you add a server to a cluster (using `ndcontrol server add` command), you must use the address of the back-end network interface and not the address of the public network interface.
- Network Dispatcher uses two default ports: 10003, used by `ndserver`, and 10004 to receive load feedback by ISS.

  To change the 10003 port, you have to change the ND_PORT value in the /usr/bin/ndserver and /usr/bin/ndcontrol files. To change the 10004 value, you have to enter the following command in the Network Dispatcher machine when you start the Manager:

  `ndcontrol manager start` *`logfile metric_port`*.

  If you specify a `metric_port`, you must specify a `logfile` name.

Furthermore, 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`*

- One of the default ports used by Network Dispatcher (port 10003) is used by one of the SP daemons. So, if you use the Network Dispatcher server as one SP node, do not forget to change this port number by changing the ND_PORT value in the /usr/bin/ndserver and /usr/bin/ndcontrol files.

Appendix D. Special Notices

This publication is intended to help system administrators to implement the Interactive Network Dispatcher to balance the load of Internet servers. The information in this publication is not intended as the specification of any programming interfaces that are provided by Interactive Network Dispatcher product. See the PUBLICATIONS section of the IBM Programming Announcement for Interactive Network Dispatcher 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 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.

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

AIX ®	Deep Blue
HACMP/6000	IBM®
LoadLeveler®	OS/2®
RS/6000	Scalable POWERparallel Systems®

The following terms are trademarks of other companies:

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.

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 135.

- *Olympic-Caliber Computing*, SG24-4279
- *High Availability on the RISC System/6000 Family,* SG24-4551
- *An HACMP Cookbook*, SG24-4553
- *HACMP/6000 Customization Examples,* SG24-4498
- *RS/6000 SP System Management: Easy, Lean and Mean*, GG24-2563
- *High Availability on RISC/6000 SP*, SG24-4742
- *SP Problem Determination Guide*, SG24-4778

E.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

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
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
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
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

E.3 Other Publications

These publications are also relevant as further information sources:

- *Interactive Network Dispatcher User's Guide - Version 1.2 for AIX, Windows NT, and Sun Solaris,* GC31-8496
- *Using and Administering Interactive Session Support for AIX, Release 1,* SC28-1548
- *Using and Administering LoadLeveler Release 3.0*, SC23-3989
- *HACMP/6000 Concepts and Facilities*, SC23-2699
- *HACMP Planning Guide*, SC23-2700
- *HACMP/6000 Installation Guide*, SC23-2701
- *Performance Toolbox for AIX Guide and Reference, Version 1.2 and 2,* SC23-2625
- *Performance Toolbox Parallel Extensions for AIX Guide and Reference*, SC23-3997
- *DNS and BIND,* Paul Albitz & Cricket Liu, O'Reilly & Associates, Inc., SR28-4970, ISBN 1-56592-236-0

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at `http://www.redbooks.ibm.com`.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** – to order hardcopies in United States
- **GOPHER link to the Internet** – type `GOPHER WTSCPOK.ITSO.IBM.COM`
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks`

- **IBM Direct Publications Catalog on the World Wide Web**

  `http://www.elink.ibmlink.ibm.com/pbl/pbl`

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **On-line** – send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **On-line Orders** (Do not send credit card information over the Internet) – send orders to:

	IBMMAIL	**Internet**
In United States	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** – send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark

- **Fax** – send orders to:

United States (toll free)	1-800-445-9269
Canada	1-800-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 408 256 5422 (Outside USA)** – ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** – send note to `softwareshop@vnet.ibm.com`

- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank).

IBM Redbook 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.

Abbreviations

ANSI	American National Standards Institute
ARP	Address Resolution Protocol
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AFS	Andrew File System
AIX	Advanced Interactive eXecutive
APAR	Authorized Program Analysis Report
ATM	Asynchronous Transfer Mode
BOS	Base Operating System
CGI	Common Gateway Interface
CNAME	Canonical Name (DNS Configuration)
CPU	Central Processing Unit
DBMS	Database Management System
DCE	Distributed Computing Environment
DFS	Distributed File System
DNS	Domain Name Server
IN	Internet (DNS configuration)
FDDI	Fiber Distributed Data Interface
FTP	File Transfer Protocol
HA	High Availability
HACMP	High Availability Cluster MultiProcessing
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IBM	International Business Machines Corporation
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IND	Interactive Network Dispatcher
I/O	Input/Output
IP	Internet Protocol
ISP	Internet Service Provider
ISS	Interactive Session Support
ITSO	International Technical Support Organization
JFS	Journaled File System
LAN	Local Area Network
MAC	Medium Access Control
ND	Network Dispatcher
NFA	Non-Forwarding Address
NIS	Network Information System
NFS	Network File System
NS	Name Server (DNS Configuration)
NNTP	Network News Transfer Protocol
OLTP	On-Line Transaction Processing
OS/2	Operating System/2

POP	Post Office Protocol
PTF	Program Temporary Fix
PTPE	Performance Toolbox Parallel Extension
PTR	Pointer (DNS Configuration)
PTX	Performance Toolbox
RDBMS	Relational Database Management System
RS/6000	IBM RISC System/6000
RS/6000 SP	IBM RS/6000 Scalable POWERparallel Systems
SHTTP	Secure HyperText Transfer Protocol
SMIT	System Management Interface Tool
SMP	Symmetric MultiProcessor
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOA	Start of Authority (DNS Configuration)
SP	IBM RS/6000 Scalable POWERparallel Systems
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TTL	Time-To-Live
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WAN	Wide Area Network
WWW	World Wide Web

Index

L

M

N

O

P

R

S

T

W

ITSO Redbook Evaluation

Load-Balancing Internet Servers
SG24-4993-00

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 on-line evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

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!)

_____

_____

_____

_____

_____

Printed in U.S.A.