

Workload Characterization of NetWare Clients and Servers Using a Protocol Analyzer

Ron Lee
Senior Consultant
Systems Research Department

This AppNote introduces two LANalyzer applications you can use to characterize workloads associated with NetWare clients, servers, and segments that support NetWare traffic exclusively. Although the instructions are specifically for Novell's LANalyzer, you can adapt them for other protocol analyzers.

Copyright © 1991 by Novell, Inc., Provo, Utah. All rights reserved.

As a means of promoting NetWare AppNotes, Novell grants you without charge the right to reproduce, distribute, and use copies of the AppNotes, provided you do not receive any payment, commercial benefit, or other consideration for the reproduction or distribution, or change any copyright notices appearing on or in the document.

Disclaimer

Novell, Inc. makes no representations or warranties with respect to the contents or use of these Application Notes (AppNotes) or of any of the third-party products discussed in the AppNotes. Novell reserves the right to revise these AppNotes and to make changes in their content at any time, without obligation to notify any person or entity of such revisions or changes. These AppNotes do not constitute an endorsement of the third-party product or products that were tested. Configuration(s) tested or described may or may not be the only available solution. Any test is not a determination of product quality or correctness, nor does it ensure compliance with any federal, state, or local requirements. Novell does not warranty products except as stated in applicable Novell product warranties or license agreements.

Contents

Introduction	33
Characterizing Workload with a Protocol Analyzer	33
On the Wire Versus Inside the Box	33
Levels of Characterization	34
The Physical Level	34
Filters for Requests and Responses	34
The Logical Level	35
Six Prominent NCPs	35
Housekeeping Traffic	36
The Functional Level	36
Client and Server Workload Applications	36
Understanding the True Workload of NetWare Clients	38
Comparing Server Workloads	39
Creating and Running WC_NCPC	41
Specifications	42
Receive Channels	42
Filter Pattern Template	42
Filter Patterns	43
Operating Instructions	45
Trigger Setup	45
Data Collection Setup	45
Target Addressing	46
Running WC_NCPC	46
Creating and Running WC_NCPS	46
Specifications	46
Receive Channels	46
Filter Pattern Template	48
Filter Patterns	49
Operating Instructions	50
Trigger Setup	50
Data Collection Setup	50
Target Addressing	51
Running WC_NCPS	52
Characterizing NetWare Segments Using WC_NCPS	52
Operating Instructions	52
Converting the Data	52

Introduction

The transparency of network traffic, or workload, has always created challenges for anyone tackling network design and optimization projects. As discussed in "An Introduction to Workload Characterization" (May 1991 *NetWare Application Notes*), workload characterization is the one toolkit that has the potential to eliminate much of the guesswork you've been accustomed to. Once you characterize the workloads related to specific servers, clients, and network segments, decisions regarding network design and optimization strategies can become amazingly simple.

This AppNote introduces two LANalyzer applications tailored to characterizing workloads associated with NetWare clients, servers, and segments that support NetWare traffic exclusively.

- The WC_NCPC application characterizes the traffic associated with an individual NetWare client. A client can represent a person using a network workstation, or an application (such as a report generator) that runs unabated without user intervention.
- The WC_NCPS application characterizes the traffic associated with a NetWare server. It can also be used to characterize workloads created by multiple servers and their associated clients on a single network segment.

I'll also discuss briefly how you can use the workload characterization results provided by these tools.

Characterizing Workload with a Protocol Analyzer

Ideally, there are many points at which you need to measure workload in order to paint a complete picture of what's happening on the network. The ability to measure every point, however, may be limited by the tools available to you. This AppNote focuses on only one of those tools—the protocol analyzer, specifically Novell's LANalyzer.

On the Wire Versus Inside the Box

To form a complete picture of workload, you need to understand how each component within a LAN or workgroup is being used. In a small workgroup network with one server and three workstations, for example, you'd have to take measurements in five general locations:

- inside the shared server
- inside each of the three client workstations
- inside the shared network cable plant

Protocol analyzers are designed to capture and analyze network traffic as it is propagated through a network cable plant. However, they also allow you to gather a surprising number of client and server workload statistics without ever looking inside either the file server or the workstations. By focusing your analysis on the network cable plant, you can characterize not only the shared network media traffic but also the request and response workload produced by the network's servers and clients.

The only missing piece is the effect of workload on shared resources other than the wire—how the resources of the servers and clients are being utilized or affected by the request and response workload on the wire. For example, using a protocol analyzer, there is no way of measuring server disk channel utilization or client CPU utilization. Tools for measuring workload and resource utilization in those devices will be the subject of future AppNotes.

Levels of Characterization

"An Introduction to Workload Characterization" introduced three levels of characterization: physical, logical, and functional. These levels provide a means of classifying and simplifying the mountain of data produced by a protocol analyzer.

The Physical Level

Within the physical level, workload is made up of a continuous stream of bytes and packets. By analyzing activity at this level, you can determine whether a particular network component is under- or over-utilized, what level of fairness (or access) is available to a specific client, and what type of error conditions exist.

Using the filtering tools provided by protocol analyzers, you can focus your view on one or more clients, on specific servers, or even on an entire network segment. For example, by filtering network traffic on a specific node address and isolating both request and response traffic, you can get an accurate picture of that node's workload volume, intensity, and patterns. Both the client and server characterization applications introduced in this AppNote use filtering to focus on specific devices. Removing the address filters from the WC_NCPS application allows you to characterize an entire segment.

Filters for Requests and Responses. It's easy to determine which packets are requests and which are responses by creating one filter for packets transmitted by a client (service requests) and another filter for packets received by a client (service responses). Of course, when analyzing a server, it's the other way around—transmissions are service responses, and receptions are service requests. So, by including two separate filters, you can capture statistics including:

- total bytes received
- total bytes transmitted
- total packets received
- total packets transmitted
- client-server utilization of the media
- peak bytes per second
- peak packets per second

After tracking these figures over a long period of time, you can make more informed decisions when the need arises. For example, you might find that, on the average, one server is processing 8.28MB of raw data per hour while another is processing 277.22MB per hour. With data like this, it's not hard to determine which server should be the focus of your attention.

The Logical Level

Within the logical level, workload is made up of request and response packets coded in Novell's NetWare Core Protocol (NCP). These, together with a variety of housekeeping packets, make up the

entire workload viewed within the physical level.

Six Prominent NCPs. Tracking NCPs is a bit trickier than just watching request and response packets, though. Novell now has over 300 NCPs defined, any of which may occur on the wire. In a joint research project involving Engineering and Systems Research staff, we isolated six NCPs that make up the majority of all traffic in a production environment. They are:

- Read From A File
- Write To A File
- Open A File
- Close A File
- File Search Initialize
- File Search Continue

These six NCPs can represent up to 98 percent of a client's or server's workload, or as little as 76 percent, depending upon the application set. (The means for calculating these percentages for your network components is included in the applications introduced below.) With few exceptions, none of the remaining NCPs individually make up more than 1 percent of the total client/server workload being measured, whether the measured object is a client, a server, or a segment.

The percentage of workload represented by these NCPs, both individually and in combination, is an important part of the workload puzzle. Some of the statistics you can derive from the NCP counts gathered by your protocol analyzer include:

- the total number of Read From A File requests;
- the percentage of Read From A File requests over total requests;
- the ratio of Read From A File requests to Write To A File requests.

These workload percentages and ratios are particularly useful in network optimization strategies. For example, suppose that after some analysis you find that the ratio of reads to writes in a large database server is 12:1. Your optimization recommendation would be much different than if the ratio was just the opposite or even a different number altogether.

Housekeeping Traffic. Once you account for the percentage of workload created by the six file-service NCPs above, the remaining workload is largely housekeeping functions. Depending on your customized set of applications and network operations, some of those functions may include:

- Get Bindery Access Level
- Create Service Connection
- End of Job
- Get File Server Date and Time
- Close File and Start A Queue Job
- Lock Physical Record Set

The percentage of housekeeping traffic can vary from 2 percent to as high as 20 percent. Both extremes tell you a great deal about how the network is being used by the clients and applications.

The Functional Level

The functional level of data analysis focuses on transactions made up of multiple NCPs, the ordering of the NCPs within transactions, and the order and response times of transactions. However, until protocol analyzers mature somewhat and allow you to build additional logic into your filters, this level of analysis can quickly become a laborious process of searching through trace files by hand. The most commonly used technique for analysis at this level is usually performed by the programmer who codes performance meters directly into the application program.

Our discussion of the two enclosed applications will focus on the physical and logical levels of workload characterization, leaving the functional level analysis using protocol analyzers to a future AppNote.

Client and Server Workload Applications

This AppNote explains how to create and use two LANalyzer applications to gather workload characterization data for network clients and servers. The name for the first application, WC_NCP, stands for "workload characterization of NetWare Core Protocol traffic at a client." The second application, WC_NCPS, is similar but for the server. Instructions for setting up and running these applications are included at the end of the AppNote. Although the instructions are specifically for Novell's LANalyzer, you can adapt them for other protocol analyzers.

Once you gather workload characterization data and convert it to spreadsheet-compatible files, your analysis options are endless. To give you an idea of the types of results you can obtain, Figures 1 and 2 show sample graph and spreadsheet printouts of WC_NCP data that I produced with Borland's Quattro Pro. Figures 3 and 4 are sample graph and spreadsheet printouts of NC_NCPS data that I produced with Quattro Pro.

Figure 1: Graph of client workload characterization data obtained from WC_NCPC.

Figure 2: Spreadsheet report of client workload characterization data from WC_NCPC.

Figure 3: Graph of server workload characterization data gathered with WC_NCPS.

Figure 4: Spreadsheet report of server workload characterization data from WC_NCPS.

I've found the three most useful numbers in the spreadsheet reports to be:

- Average Utilization of the Thrput channel
- Peak Utilization of the Thrput channel
- Reads / Writes Ratio (RPCs)

The following sections suggest just a few ways you can use these workload statistics in real-life network design, optimization, and management.

Understanding the True Workload of NetWare Clients

Networks are very good at hiding the complexity of protocols, routers, bridges, tokens, and so forth. As a result, technical and non-technical people alike sometimes have incorrect perceptions about how much workload one individual or group of individuals place on a network. These misconceptions can create confusion, especially when response times go into the red zone, or when additional design or optimization decisions come up.

In "An Introduction to Workload Characterization," we looked at the sample case of Laura Maxwell, a legal secretary running WordPerfect 5.1 from within Microsoft Windows. The graph in Figure 1 represents the overall network utilization by Laura's workstation as sampled on March 5, 1991. With her legal responsibilities, Laura uses her word processor more heavily than most other types of users would. Yet her workstation's use of the network is very light. Why? Due to the design of WordPerfect, most of the processing takes place in the workstation. Only file import, export, and search activity uses the server's resources.

Looking at the corresponding numbers in Figure 2, however, you get quite a different picture of the workload. During the sampling period, Laura's workstation sent 30,705 requests to the server and received 30,705 responses, for a total of 61,410 packets. That represents just

over 16MB of raw data in four hours!

As you can see, WC_NCPC results give you some hard data regarding average workload, peak periods, and trends that affect your design and optimization decisions. The resulting mental images of workload on the network are often quite different from previous misconceived notions.

Comparing Server Workloads

Server workloads are misunderstood and miscalculated just as often as those of individual users. As an example, consider three separate servers serving three different environments—a workgroup, a department, and a division.

Figure 5 represents the workgroup server used by Laura Maxwell and the legal staff that works with her—an average of eight users throughout the day.

Figure 5: Workload graph for a small workgroup server.

Figure 6 represents a departmental server used by an accounting department in a \$500M company, servicing an average of 50 users.

Figure 6: Workload graph for a departmental server.

Figure 7 represents a large server with 4GB of disk space and a mission critical database with over 200 users.

Figure 7: Workload graph for a division server.

With just one quick glance at these three figures, it's easy to see why early versions of NetWare running on XTs were sufficient for most user workloads created by fewer than 50 users. With much of the workload distributed to the workstation, the volume and intensity of service requests aimed at the server are fairly low given that number of users. When we move to several hundred users, as in Figure 7, workloads begin to inch up towards a more efficient use of the media and network components.

Hopefully, the workload characterization data provided by WC_NCPC and WC_NCPS will provide both technical and non-technical decision makers with a correct perception of how much their network resources are being utilized.

Creating and Running WC_NCPC

As explained earlier, the application name "WC_NCPC" stands for "workload characterization of NetWare Core Protocol traffic at a client." WC_NCPC is a LANalyzer application that uses the LANalyzer's enhanced filter mode, start and stop triggers, all eight receive channels, and 16 patterns. If you own a LANalyzer, you can easily create WC_NCPC using the following specifications. (See your *Novell LANalyzer Reference Manual* for instructions on creating, editing, and saving applications.)

Specifications

Receive Channels. The following specifications describe the parameters for eight receive channels, which are:

- Reads
- Writes
- FsearchC
- FsearchI
- Opens
- Closes
- Xmit

- Thrput

The first six channels compute statistics regarding the six most prevelant NCPs mentioned earlier. Since transmitted packets represent client requests, these six channels are designed to track only packets being transmitted by the target client.

The Xmit channel gathers statistics regarding the total packet stream transmitted by the client. The Thrput channel gathers statistics regarding both packets transmitted and received. With this design, it's easy to compute the Rcv traffic by subtracting the Xmit totals from the Thrput totals. Refer to Figure 8 as you set up these channels.

Since the majority of NetWare clients do not normally participate in RIP or SAP communications, there is no need to track that traffic with this application. However, you will want to track RIP and SAP traffic with the server application.

Filter Pattern Template. The NetWare template used with all of the filter patterns is supplied with the LANalyzer software.

Template Search Path: C:\XLN\LANZ\TEMPLATE

Pattern Template		Overriding Offset
rcv	NetWare	None
xmit	NetWare	None
open	NetWare	None
opennew	NetWare	None
close	NetWare	None
read	NetWare	None
write	NetWare	None
fsearchc	NetWare	None
fsearchi	NetWare	None

Figure 8: Channel settings for the WC_NCPC application.

```
Channel 1
Name: Reads
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (xmit) AND (read)

Channel 2
Name: Writes
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (xmit) AND (write)

Channel 3
Name: FsearchC
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (xmit) AND (fsearchc)

Channel 4
Name: FsearchI
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (xmit) AND (fsearchi)
```

Channel 5
Name: Opens
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (xmit) AND (open or opennew)

Channel 6
Name: Closes
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (xmit) AND (close)

Channel 7
Name: Xmit
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (xmit)

Channel 8
Name: Thrput
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (xmit or rcv)

Filter Patterns. In the pattern descriptions shown in Figure 9, I've used a workstation with the Ethernet address NOVELL30EB7F as an example target workstation. Before running WC_NCPC on your network, you'll have to fill in a new target address.

The word "None" means the same thing as "Don't care" in the LANalyzer Reference Manual. "Req Type" and "Function Code" refer to fields within the filter pattern template.

Figure 9: Filter patterns for the WC_NCPD application.

Pattern Name:	rcv
Station:	XX XX XX XX XX XX -> NOVELL30EB7F
Type:	None
Data:	None
Pattern Name:	xmit
Station:	NOVELL30EB7F -> XX XX XX XX XX XX
Type:	None
Data:	None
Pattern Name:	read
Station:	None
Type:	None
Data:	Req Type: 2222 Function Code: 72
Pattern Name:	write
Station:	None
Type:	None
Data:	Req Type: 2222 Function Code: 73
Pattern Name:	fsearchc
Station:	None
Type:	None
Data:	Req Type: 2222 Function Code: 63
Pattern Name:	fsearchi
Station:	None
Type:	None
Data:	Req Type: 2222 Function Code: 62
Pattern Name:	open
Station:	None
Type:	None
Data:	Req Type: 2222 Function Code: 65
Pattern Name:	opennew
Station:	None
Type:	None
Data:	Req Type: 2222 Function Code: 76
Pattern Name:	close
Station:	None
Type:	None
Data:	Req Type: 2222 Function Code: 66

Operating Instructions

Trigger Setup. For comparative purposes based on time of day and length of test, you should always run WC_NCPD with triggers. The most useful test periods for my analysis purposes have been one hour, four hours, ten hours, and twelve hours. My recent favorite is

the ten hour test, from 7:30am to 5:30pm. Because the test has to be reset each night, the 5:30pm stop trigger allows me to reset the test for the next day and still leave early enough to get home at a reasonable hour.

LANalyzer users will recognize the following lines from the LANalyzer's trigger parameters screen:

- Start collecting at 07:30:00 hrs. or no other trigger.
- Fire stop trigger at 17:30:00 hrs. or no other trigger.
- Once stop trigger has fired, collect an additional 0 packets.
- When the trace buffer is full, overwrite old packets.

Data Collection Setup. Due to the extended length of most workload characterization tests, the resulting mountain of information presents some interesting storage and analysis problems.

At the LANalyzer's default compilation rate of once per second, a ten hour test would create 32,000 records. Since I use Borland's Quattro Pro as my analysis tool, I can only process up to 7,200 records at one time. I assume most AppNote readers will use a similar tool.

The solution to this problem is to decrease the rate at which the LANalyzer samples the board and compiles statistical data records. This effectively reduces the statistical data to a more useable form while maintaining a sufficient level of accuracy. The spreadsheet I've designed to process the WC_NCPC data is built around a 7,200 record data model. This fits easily into any spreadsheet and is obtainable at any test period length that is divisible by two. For example:

Test Period	Sampling Rate	Record count
2 hr.	every 1 sec.	7,200
4 hr.	every 2 sec.	7,200
10 hr.	every 5 sec.	7,200
12 hr.	every 6 sec.	7,200
24 hr.	every 12 sec.	7,200

As an example, I used the following LANalyzer data collection settings for a ten hour test period of Laura Maxwell's workstation on June 6th. No trace file is necessary, since the resulting data would quickly fill the disk capacity of the LANalyzer.

- Collection Performance: Normal
- Trace Slice: Offset: 0 Length: Max
- Trace File:
- Statistics File: C:\STAT\LAUR0610
- Save statistics to file every 5 second(s).
- Print screen every 0 minute(s).
- Collect transmit statistics: No

With a 7,200 record count, each test created 12 statistics files with the extension .ST1 through .STD (using the hexadecimal numbering system). Processing the LANalyzer data with the LANZSTAT utility resulted in an additional 12 files, for a total of approximately 6MB of disk storage.

Target Addressing. To filter on a single client workstation, determine the workstation's node address and place it in the "rcv" and "xmit"

filter patterns described above. For example:

Pattern Name: rcv
Station: XX XX XX XX XX XX -> NOVELL30EB7F

Pattern Name: xmit
Station: NOVELL30EB7F -> XX XX XX XX XX XX

Running WC_NCPC. Once you have finished setting triggers, data collection parameters, and target addresses, you're ready to start the application. Because of the way the LANalyzer's triggers operate, you can set up the LANalyzer and start the application up to 24 hours before the start trigger is set to fire. This should allow plenty of time for setup.

Creating and Running WC_NCPS

The application name WC_NCPS stands for "workload characterization of NetWare Core Protocol traffic at a NetWare server." However, WC_NCPS can also be used to characterize entire network segments, a technique covered in the next section of this AppNote.

WC_NCPS is a LANalyzer application that uses the LANalyzer's enhanced filter mode, start and stop triggers, all eight receive channels, and 16 patterns. If you own a LANalyzer, WC_NCPS can easily be created using the following specifications. (See your *Novell LANalyzer Reference Manual* for instructions on creating, editing, and saving applications.)

Specifications

Receive Channels. The following specifications describe the parameters for eight receive channels, including:

- Reads
- Writes
- Fsearch
- OpnCls
- Delays
- SapRip
- Recv
- Thrput

The first four channels compute statistics regarding the six most prevalent NCPs mentioned earlier. (This is different than in WC_NCPC because we need to use some of the channels for additional workload types not encountered by workstations.) Since received packets represent service requests, these four channels are designed to track only packets being received by the target server.

The Delays channel records statistics regarding an NCP response packet from server to client called Request Being Processed. The server returns this response to the client when the server receives a duplicate request. This condition usually occurs when the server encounters peak utilization of some sort and must therefore delay

processing the client's request. If the delay is long enough, the client's timer can expire and cause the client to send a duplicate request. This resend process is built into the IPX protocol to safeguard against a rare condition in which the initial packet is lost. Most of the time the packet is not lost; it is still being processed by the server—hence the Request Being Processed reply. Tracking these replies along with the other workload statistics is helpful in understanding the frequency and causes of such delays. (For more information, see "Resolving Performance Problems on a NetWare Network," March 1991 *NetWare Application Notes*.)

The SapRip channel captures statistics regarding all Service Advertising Protocol (SAP) traffic and Router Information Protocol (RIP) traffic both received and transmitted by the server. (For more information, see "NetWare Communications Processes," September 1990 *NetWare Application Notes*.)

The Recv channel gathers statistics regarding the total packet stream received by the server. The Thrput channel gathers statistics regarding both packets transmitted and received. With these two channels, it's easy to compute the Xmit traffic by subtracting the Xmit totals from the Thrput totals. I use this design because the service request workload is the primary focus on the server. Of course, if I had more channels to work with, I'd probably have a Xmit channel as well as several others. Refer to Figure 10 to set up the channels.

Figure 10: Channel settings for the WC_NCPS application.

```
Channel 1
Name: Reads
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (rcv) AND (read)

Channel 2
Name: Writes
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (rcv) AND (write)

Channel 3
Name: Fsearch
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (rcv) AND (fsearchc or fsearchi)
```

Channel 4
Name: OpnCIs
Active: Yes
Size Range From/To: Min/Max
Packets Allowed: Good Packets
Collect Stats.: Yes
Start Count: Off
Stop Count: Off
Filter patterns: (rcv) AND (open or opennew or close)

Channel 5
 Name: Delays
 Active: Yes
 Size Range From/To: Min/Max
 Packets Allowed: Good Packets
 Collect Stats.: Yes
 Start Count: Off
 Stop Count: Off
 Filter patterns: (xmit) AND (delays)

Channel 6
 Name: SapRip
 Active: Yes
 Size Range From/To: Min/Max
 Packets Allowed: Good Packets
 Collect Stats.: Yes
 Start Count: Off
 Stop Count: Off
 Filter patterns: (rcv or xmit) AND (Skt452 or Skt453)

Channel 7
 Name: Recv
 Active: Yes
 Size Range From/To: Min/Max
 Packets Allowed: Good Packets
 Collect Stats.: Yes
 Start Count: Off
 Stop Count: Off
 Filter patterns: (rcv)

Channel 8
 Name: Thrput
 Active: Yes
 Size Range From/To: Min/Max
 Packets Allowed: Good Packets
 Collect Stats.: Yes
 Start Count: Off
 Stop Count: Off
 Filter patterns: (xmit or rcv)

Filter Pattern Template. The NetWare template used with all of the filter patterns is supplied with the LANalyzer software.

Template Search Path: C:\XLN\LANZ\TEMPLATE

Pattern	Template	Overriding Offset
rcv	NetWare	None
xmit	NetWare	None
open	NetWare	None
opennew	NetWare	None
close	NetWare	None
read	NetWare	None
write	NetWare	None
fsearchc	NetWare	None
fsearchi	NetWare	None
delays	NetWare	None
Skt452	NetWare	None
Skt453	NetWare	None

Filter Patterns. In the pattern descriptions shown in Figure 11, I've used the Ethernet address NOVELL30EB7F as an example target address. Before running WC_NCPS on your network, you'll have to fill in a new target address.

The word "None" means the same thing as "Don't care" in the *LANalyzer Reference Manual*. "Req Type" and "Function Code" refer to fields within the filter pattern template.

Figure 11: Filter patterns for the WC_NCPS application.

Pattern Name: rcv
Station: XX XX XX XX XX XX -> NOVELL30EB7F
Type: None
Data: None

Pattern Name: xmit
Station: NOVELL30EB7F -> XX XX XX XX XX XX
Type: None
Data: None

Pattern Name: read
Station: None
Type: None
Data: Req Type: 2222
Function Code: 72

Pattern Name: write
Station: None
Type: None
Data: Req Type: 2222
Function Code: 73

Pattern Name: fsearchc
Station: None
Type: None
Data: Req Type: 2222
Function Code: 63

Pattern Name: fsearchi
Station: None
Type: None
Data: Req Type: 2222
Function Code: 62

Figure 11: Filter patterns for WC_NCPS (continued).

Pattern Name: open
Station: None
Type: None
Data: Req Type: 2222
Function Code: 65

Pattern Name: opennew
Station: None
Type: None
Data: Req Type: 2222

Function Code: 76

Pattern Name: close
Station: None
Type: None
Data: Req Type: 2222
Function Code: 66

Pattern Name: delays
Station: None
Type: None
Data: Req Type: 9999

Pattern Name: Skt452
Station: None
Type: None
Data: Dest Socket: 0452

Pattern Name: Skt453
Station: None
Type: None
Data: Dest Socket: 0453

Operating Instructions

Trigger Setup. For comparative purposes based on time of day and length of test, you should always run WC_NCPS with triggers. LANalyzer users will recognize the following lines from the LANalyzer's trigger parameters screen. These settings use for an example a test period of 12 hours, from 6am to 6pm.

- Start collecting at 06:00:00 hrs. or no other trigger.
- Fire stop trigger at 18:00:00 hrs. or no other trigger.
- Once stop trigger has fired, collect an additional 0 packets.
- When the trace buffer is full, overwrite old packets.

Data Collection Setup. Due to the extended length of most workload characterization tests, the resulting mountain of information presents some interesting storage and analysis problems.

The LANalyzer's default rate of statistical compilation is once per second. At this rate, a ten hour test would create 32,000 records. Since I use Borland's Quattro Pro as my analysis tool, I can only process up to 7,200 records at one time. I assume that most AppNote readers will use a similar tool.

The solution to this problem is to decrease the rate at which the LANalyzer samples the board and compiles statistical data records. This effectively reduces the statistical data to a more useable form while keeping a sufficient level of accuracy. The spreadsheet I've designed to process the WC_NCPS data is built around a 7,200 record data model. This fits easily into any spreadsheet and is obtainable at any test period length that is divisible by two. For example:

Test Period	Sampling Rate	Record count
2 hr.	every 1 sec.	7,200
4 hr.	every 2 sec.	7,200
10 hr.	every 5 sec.	7,200

12 hr.	every 6 sec.	7,200
24 hr.	every 12 sec.	7,200

The following LANalyzer data collection settings are for a twelve hour test period of the LEGAL server on June 6th. No trace file is necessary, since the resulting data would quickly fill the disk capacity of the LANalyzer.

- Collection Performance: Normal
- Trace Slice: Offset: 0 Length: Max
- Trace File:
- Statistics File: C:\STAT\LGL0610
- Save statistics to file every 6 second(s).
- Print screen every 0 minute(s).
- Collect transmit statistics: No

With a 7,200 record count, each test will create 12 statistics files with the extension .ST1 through .STD (using the hexadecimal numbering system). Processing the LANalyzer data with the LANZSTAT utility results in an additional 12 files, making a total of approximately 6MB of disk storage.

Target Addressing. To filter on a single server, determine the server's node address and place it in the "rcv" and "xmit" filter patterns described above.

Pattern Name: rcv
Station: XX XX XX XX XX XX -> NOVELL30EB7F

Pattern Name: xmit
Station: NOVELL30EB7F -> XX XX XX XX XX XX

Running WC_NCPS. Once you have finished setting triggers, data collection parameters, and target addresses, you're ready to start the application. Because of the way the LANalyzer's triggers operate, you can set up the LANalyzer and start the application up to 24 hours before the start trigger is set to fire. This should allow plenty of time for setup.

Characterizing NetWare Segments Using WC_NCPS

Using WC_NCPS to characterize entire network segments is easy to do. The benefits of segment characterization include:

- the ability to track segment workload averages, peaks, and growth trends for capacity planning purposes; and
- the ability to locate underutilized segments that can be consolidated with other segments, thereby simplifying network management processes.

Operating Instructions

Set up WC_NCPS using the directions given above, except clear the target server addresses to eliminate all address-related filtering. Then run WC_NCPS using the instructions above. Your data will reflect the total workload on the attached segment rather than that of an isolated server.

Converting the Data

Before you can effectively analyze your LANalyzer statistical data, you must convert it into a useable form. LANZSTAT.EXE is a utility program provided with the LANalyzer software that converts LANalyzer statistics files into spreadsheet-compatible files.

LANZSTAT creates the following files:

OUTPUT.TSU	Test summary statistics
OUTPUT.GSU	Global counter summary
OUTPUT.CSU	Channel counter summary
OUTPUT.SMP	Sample numbers and timestamps
OUTPUT.CH1	Channel 1 counters
OUTPUT.CH2	Channel 2 counters
OUTPUT.CH3	Channel 3 counters
OUTPUT.CH4	Channel 4 counters
OUTPUT.CH5	Channel 5 counters
OUTPUT.CH6	Channel 6 counters
OUTPUT.CH7	Channel 7 counters
OUTPUT.CH8	Channel 8 counters
OUTPUT.TRX	Transmit counters
STATNAME.TXT	Primary name of the output files