podbooks.com™

Advanced Network Analysis Techniques

Laura A. Chappell, Sr. Protocol Analyst
Protocol Analysis Institute, LLC
www.packet-level.com

- **Diagnose network statistics including short and long term trends and patterns.**
- **Build and efficiently use capture and display filters based on addresses, protocols and data patterns.**
- **Analyze and document a network application using the Application Analysis Form.**
- **Practice decoding and troubleshooting various communication patterns.**
- **Learn to manually decode communications for new or undecoded protocols.**
- **Set up an automated capture trigger system and catch data on an unattended analyzer.**
- **Learn to analyze a switched network using port redirection.**

The **ultimate reference guide** to using a protocol analyzer to troubleshoot and test network communications and applications.

This text is the follow-up study guide to “Introduction to Network Analysis” book and video-based course also available from *podbooks.com*.

Advanced Network Analysis Techniques

Proven techniques for analyzing and troubleshooting network communications and protocol behavior.

Laura A. Chappell
Sr. Protocol Analyst
Protocol Analysis Institute, LLC
lchappell@packet-level.com

Published by

podbooks.com™

Advanced Network Analysis Techniques
By Laura A. Chappell
Cover Model: Chadwick Chappell

Published by
podbooks.com
5339 Prospect Road, Suite 343
San Jose, CA 95129
USA

Library of Congress Catalog Card No.: Pending
ISBN: 1-893939-28-6
Printed in the United States of America (Saratoga, California)
10 9 8 7 6 5 4 3 2 1
Distributed worldwide through podbooks.com.

For general information on podbooks.com, including information on corporate licenses, book authoring procedures, and future titles, contact podbooks.com at 408/378-7841 or info@podbooks.com.

For authorization to photocopy items for corporate, personal, or educational use, contact the copyright division at copyright@podbooks.com.

Limit of Liability/Disclaimer of Warranty. Authors and publishers have used their best efforts in preparing this book. Podbooks.com and authors make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. There are no warranties that extend beyond the descriptions contained in this paragraph. No warranty may be created or extended by sales representatives or written sales materials. The accuracy and completeness of the information provided herein and the opinions stated herein are not guaranteed or warranted to produce any particular result, and the advice and strategies contained herein may not be suitable for every individual, especially individuals named 'Fred'. Neither podbooks.com nor author shall be liable for any loss of profit or any other commercial damages, including without limitation special, incidental, consequential, or other damages.

Copy Protection. In all cases, reselling or duplication of these materials without authorization is expressly forbidden.

podbooks.com™

podbooks.com
info@podbooks.com
www.podbooks.com

Welcome to podbooks.com

The first podbook was released in May 1999 and introduced a new type of book to the technical industry -- *pods* are typically smaller and less formal than industry books. Pods address the immediate need for information on a variety of subjects ranging from basic analysis techniques to Cisco router configuration. Pods often come with supplemental materials such as trace files). Additional pods are available in electronic and hardcopy form at *www.podbooks.com*

In May 2000, we videotaped my favorite seminars and bundled them with comprehensive workbooks for the “Laura Chappell presents... “ seminars. The videos are uncut, unedited, unpolished - and often irreverant -- the only way to learn!

We are also most interested in your feedback on this and all other pods. Send your thoughts and comments to me at **lchappell@packet-level.com**. If you would like to receive release information about future projects, please join the mailing list at **www.packet-level.com**.

Laura A. Chappell
Founder, podbooks.com

About the Author

Laura Chappell, Sr. Protocol Analyst, is a highly-energetic speaker and author who has trained thousands of LAN/WAN administrators, engineers, technicians and developers. She admittedly 'lives, eats and breathes' packets and packet-level communications. She analyzes and documents network performance by providing onsite and offsite network analysis, troubleshooting and optimization services. Her international client base ranges from governments to healthcare to financial/banking institutions.

Ms. Chappell has authored and edited numerous titles including:

- "Introduction to Network Analysis"
- “TCP/IP Analysis and Troubleshooting”
- “Hands-On Cisco: TCP/IP LAN Configurations”
- "Novell's Guide to LAN/WAN Analysis: IPX/SPX"
- "Novell's Guide to Internet Access Solutions"
- "Novell's Guide to Multiprotocol Internetworking"
- "Introduction to Cisco Router Configuration"
- "Advanced Cisco Router Configuration"
- "Cisco Internetwork Troubleshooting"

Ms. Chappell can be reached via email at **lchappell@packet-level.com**.

In memory of Drake,

my dear friend.

We all miss you so much.

About This Pod

This pod is the follow-up to “Introduction to Network Analysis,” released by podbooks.com last year. This book focuses on the advanced techniques that I have used over the years to document protocols, troubleshoot applications, identify network bottlenecks, isolate network hackers, and baseline network performance.

Over the years, I have used numerous analyzer products -- from the early DOS-based analyzers that required manual decoding of the up-to-date protocols to the most recent graphical analyzers that illustrate the network traffic in methods too numerous to mention.

I’ve used good analyzers.

I’ve used bad analyzers.

I’ve used really lousy analyzers.

I hope you select an analyzer that can do all of the tasks shown in this book. The analyzers that I use are Network Associates’ Sniffer Pro and WildPackets’ EtherPeek. Both analyzers offer a wide range of features and are relatively easy to operate. I highly recommend these two analyzers to all my students and readers. In fact, I will use these two analyzers throughout this book.

Who Should Read This Book

This book is designed to support anyone who wants to work in networking with a focus on troubleshooting and optimization.

Chapter Information

Chapter 1, “Statistics, Trends, Patterns and Timestamping,” covers statistics, trends, patterns and timestamping. OK, I was just checking to see if anyone really reads this front matter stuff. <grin> In all seriousness, folks, this chapter focuses on the meaning and usage of those lovely statistics dials and charts. I’ve included some information about false positives and false negatives and some hints on how to use your alarms as triggers for automated analyzer startup and packet capture.

Chapter 2, “Capture and Display Filtering” covers packet filtering techniques based on source/destination MAC-layer or network-layer address, source/destination port numbers and advanced pattern/offset matching. In this chapter, you will build a filter designed to capture packets to specific multicast addresses as a basic task. In a more advanced lab task, you will build a capture filter to identify packets used in a Distributed Denial of Service (DDoS) attack. The Application Analysis Form is included in Appendix D.

Chapter 3, “Application Analysis,” shows you how to use the Application Analysis Form (AAF) to diagnose and document an application on your network. You’l learn the 10-step process used in a complete application analysis session. Finally, I included an analysis of an FTP file transfer and an HTTP web browsing process.

Chapter 4, “Manual Decoding,” gets down to the nitty gritty of the packets. In this chapter, you’ll learn how to handle life when your protocol analyzer doesn’t decode a packet completely. Don’t cry - decode!

Chapter 5, “The Master Analyst’s Toolkit,” focuses on the various tools that you should have at hand when you are analyzing your network. From a hex decoder to a packet sanitizer, these are the tools ya just gotta have!

Appendix A, “Answers to Quizzes,” provides the answers to the quizzes contained at the end of each chapter (except Chapter 5 which I didn’t write quiz question for).

Appendix B, “Swiched LAN Analysis,” provides details on configuring and using single spans, multiple spans, and VLAN spans for analyzing switched networks. In this chapter, I focus on the Cisco CAT 4000, 5000 and 6000 family as well as the 3Com 9100 switch family.

Appendix C, “Resources for Analysts,” provides details on various articles, websites and books that should be reviewed by any network analyst.

Appendix D, “Application Analysis Form,” provides several pages of the AAF form that is referred to in Chapter 3.

Ok... here we go! Happy reading!

Special thanks to Jill Poulsen and Brenda Czech who put up with me daily. Sorry about the deep fried chicken head wallpaper on your desktop, Jill... <grin>

Thanks to Robert Kusters at Network Associates and Janice Spampinato at WildPackets -- I appreciate the assistance and generosity you both have provided to me.

Table of Contents

List of Figures

CHAPTER 1

Statistics, Trends, Patterns and Timestamping

In this chapter, we'll look at the various types of statistics that you can gather on your network and how to use those statistics for optimization and troubleshooting. We'll also look at the short- and long-term trends that can help you understand and work with your network's unique traffic patterns. The other patterns of interest will be the request/reply pattern structures -- using pattern analysis, you can identify good, bad and ugly applications. Finally, we'll focus on timestamping -- this includes relative, absolute and delta timestamping methods.

Alarms are definitely related to these statistics and trends, so we'll discuss alarms and alerts as well.

If your analyzer isn't shown in the screen shots of this book, don't despair -- learn everything your analyzer has to offer. If you spend the time to fully understand what your analyzer can do and you are disillusioned or frustrated, consider getting another analyzer.

LINK

Use the information contained at ***www.packet-level.com*** as a reference while you read this chapter.

Statistics

There are numerous statistics that you can get with a good analyzer. Among these statistics, you will find the 'personality' of your network and the applications that reside on it.

NOTE *Believe me, all networks and applications* **do** *have a personality. AppleTalk networks are typically chatty little buggers; Token Ring networks are filled with paranoid little rat finks; SPX-based applications are just plain stupid and NetBIOS networks are worth your pity. -- Laura*

These statistics, if presented in a graphical and easy-to-understand format, can be presented to non-technical management "mucky mucks" to obtain funding for more advanced devices, more personnel to manage and troubleshoot your network or additional equipment to route or switch communications effectively. In the most delightful situation, these statistics can be used to justify the removal of misbehaving applications or protocols from your network.

FIGURE 1-1. **Even non-technical managers can understand the graphical views shown on many analyzers.**

Gathering statistics is part of the baselining process. Baselining is the process of documenting the 'normal' behavior of the network. Baselines provide a measurement of the health and activity of a network over time (ideally several months). Unusual network activity can be compared to the baseline to determine whether the activity is due to typical network growth or the activity is indicative of a fault on the network. Now Let's look at the variety of statistics that characterize your network personality.

HANDS-ON

Fire up your analyzer! The best way to relate this information to your network is to check out each of these statistics on your analyzer as you read through the definitions and examples shown in this section. --Laura

Packets Per Second

Knowing your network's packet-per-second rate is critical. Just consider how many devices need to make decisions on those packets. Packets sent to a router, for example, must be examined at the network layer before they can be forwarded. If you buy a router that can't support your packet-per-second rate, packets are dropped and you're in trouble.

The packet-per-second rate does *NOT* tell you the network utilization percentage -- a common confusion -- because we do not know the size of each packet. You may find one network that has 5,000 packets per second with less than 2% utilization (64 byte packets) and another network that has 5,000 packets per second experiences 60% utilization (with 1518 byte packet sizes).

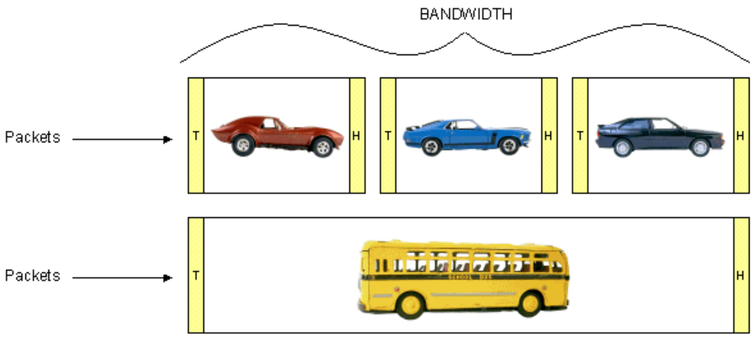


FIGURE 1-2. Smaller packets take up less bandwidth, but require more processing time for the same amount of data. They are a waste of bandwidth because they contain so much overhead (with a series of headers and a trailer required for each packet).

Another issue that we will focus on later in this *podbook* will be the packet size distribution factor -- hey -- Size DOES matter! <grin>

Figure 1-3 shows the dashboard from the Sniffer Network Analyzer (version 3.5), hereinafter referred to simply as the 'Sniffer.' As you will notice, there are two numbers under the "Packets/s" dial. The first number (148 in the graphic) is the current packets per second rate. The second number (372 in the graphic) is the peak packets per second rate.

This dial only shows the packet-per-second rate peak since the time you launched the analyzer window.

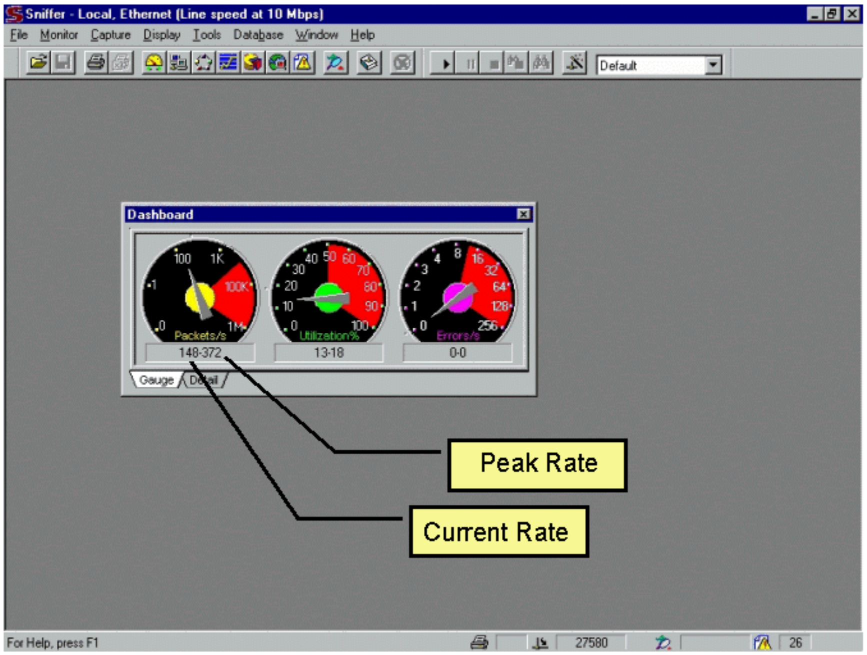


FIGURE 1-3. **The Sniffer *dashboard* illustrates the current packet rate of 148 packets per second and the peak packet rate at 372 packets per second.**

HANDS-ON

Analyze it! What is the packet-per-second rate on your network? Run your analyzer for at least 24-hours and see how high the packet-per-second rate gets. Later, we'll look at how the packet-per-second history can be interpreted to set alarms and identify traffic trends.

How many packets can typically flow across a network? It can be anything from 100 packets per second to millions of packets per second. Knowing the packets-per-second rate seen on your network enables you to know which type of device to use when interconnecting that network.

NOTE

As our interconnecting devices get smarter and smarter, we see them look way into the packets -- up higher and higher into the layers -- to make their forwarding decisions. Features such as Access Lists, policy-based routing, QoS (Quality of Service) and other queuing mechanisms can affect the packet-per-second rate that a router can handle. Cisco Access Lists, for example, can be defined based on the process identified in a packet. These Access Lists are invoked after a routing decision has been made by the Cisco router, but before the packet is sent out on an interface. This type of overhead can eventually cause a router to drop packets on a very busy network. Watch it! -- Laura

Utilization (Percentage)

This is the measurement of how much of the ‘pipe’ is in use and a key indicator of network congestion. Utilization is measured by the number of bits (grouped into kilobytes by most analyzers) that cross the cabling system, which makes this statistic roughly related to the kybtes/second statistic that follows. How many do bits does your network support? Well....

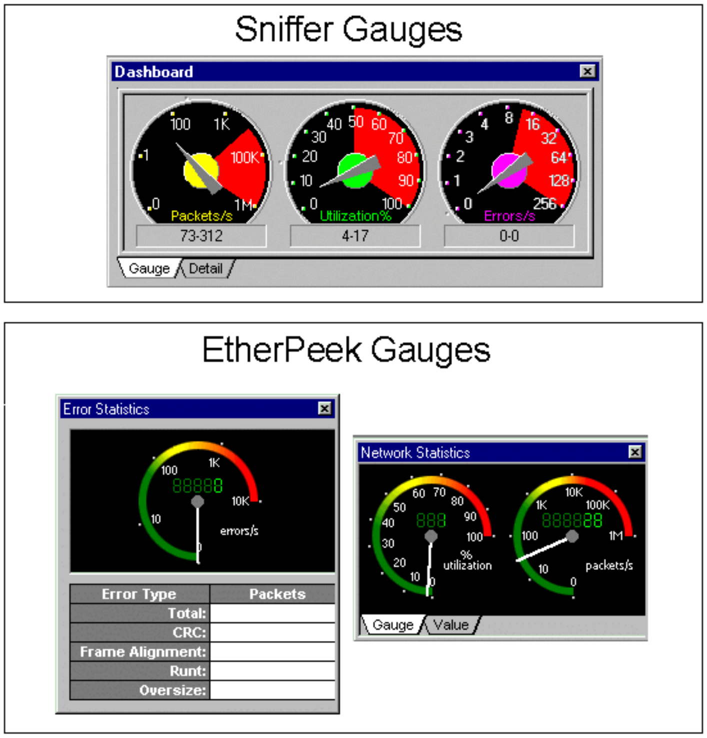
- A 10 Mbps network supports 10,000,000 bits per second.
- A 100 Mbps network supports 100,000,000 bits per second.

The thought of using the entire 10,000,000 or 100,000,000 bits per second for transferring data is truly a ‘pipe dream.’ It just ain’t gonna happen folks. Why not? Well, it deals primarily with the layer 2 (data link layer) overhead. On an Ethernet network there will be collisions (especially on a busy network). The aftermath of these collisions is a jam signal that propagates through the network. There is a backoff and recovery phase (retransmission) required due to these collisions. There may be subsequent collisions that occur... which causes the jam signal to be propagated again... That will certainly stop you from getting an entire 10,000,000 bits per second of data across the wire. Don’t even get me started on the overhead of Token Ring! Jeeesh!

Figure 1-4 depicts the statistics dials from the Sniffer and EtherPeek products . The Alarm threshold is set at about 50% utilization for each products by default.

HANDS-ON

Analyze it! What is the utilization on your network? A gradual increase in utilization is typically caused by adding workstations over time, or increasing the users' workload. If you are baselining your network properly, you can do some trending and analysis to forecast your network growth with a reasonable degree of confidence. A sudden increase in utilization typically comes from adding applications or making a change in the network (potential misconfiguration). If you haven't made any changes or added any applications, check to be certain you are not a victim of bandwidth-robbing denial of service (DoS) or reconnaissance probe operations. Also note that users love nothing more than to add their own shared applications (such as games) to the network just to see if you'll notice. Later in this chapter, we'll check out the utilization trends to help set the alarm threshold based on your network traffic.



Error Type	Packets
Total:	
CRC:	
Frame Alignment:	
Runt:	
Oversize:	

FIGURE 1-4. **EtherPeek and Sniffer have similar gauges to track Packets/s, the Utilization % and Errors/s.**

NOTE

I know it's confusing to go back and forth between the two analyzers/views in a single book. I will use Sniffer as my primary analyzers but remind you -- I also used EtherPeek during the writing of this book. Both analyzers are top quality products! If you'd like more information on how to use EtherPeek, check out www.packet-level.com and look for the article on "10 Cool Things You Can Do Today with the EtherPeek Demo." -- Laura

So how high is too high on network utilization? There are two ways to look at this -- the theoretical way or the practical way. The theoretical way is to use a static number for all networks -- "oh... 70% is way too high. End of story." In the real world, however, if your network supports 70% utilization with no packet loss, then the 70% utilization rate is not excessive. If you start to see high collision rates and dropped packets at this rate and users are beginning to squirm (which can be fun to watch)... well, then you'd better keep the traffic under control and under 70%.

Check the statistics of your routers, switches and servers to identify dropped packets. Look for collisions with your analyzer -- they show up in the Errors/s column. We'll cover errors-per-second next.

NOTE

Remember that you'll need a promiscuous mode analyzer and driver to see errors on the network. -- Laura

Errors-per-second

This one should be obvious -- this is the statistic you hate to see rise. errors-per-second stats typically include a range of MAC-layer (Media Access Control layer) errors. This could include:

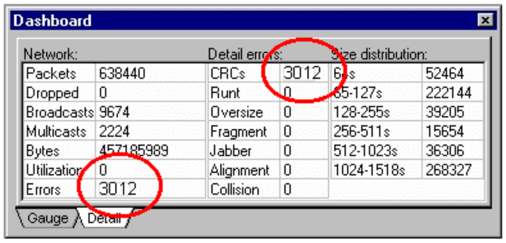
- CRC errors (incorrect CRC value at end of packet)
- fragments (packets <64 bytes with a bad CRC)
- undersized packets (packets <64 bytes with a good CRC)
- oversized packets (packets >1518 bytes with a good CRC)

Those examples are Ethernet MAC errors only -- naturally, there are other errors that indicate problems on token ring, FDDI or other network types.

LINK

Refer back to “Part 3: Identifying Typical Problems” in the Introduction to Network Analysis book by podbooks.com. In that chapter, I detailed collisions, CRC errors, and more. You can also get the specifications on packet sizes and error types online at **www.ieee.org**. This is one of the premier web sites you should have bookmarked. The individual specifications, such as the 802.3 specifications, are available in PDF format for around $200 US.

Figure 1-5 shows how you can click on the Detail tab to see the breakdown of each data link error traced by the Sniffer.



Network:		Detail errors:		Size distribution:	
Packets	638440	CRCs	3012	64s	52464
Dropped	0	Runt	0	65-127s	222144
Broadcasts	9674	Oversize	0	128-255s	39205
Multicasts	2224	Fragment	0	256-511s	15654
Bytes	457185989	Jabber	0	512-1023s	36306
Utilization	0	Alignment	0	1024-1518s	268327
Errors	3012	Collision	0		

FIGURE 1-5. **The Errors/second statistic details can be seen by clicking on the Detail tab on Sniffer. Most analyzers have a way to breaking down the errors based on their types.**

HANDS-ON

Analyze it! What is the error per second rate on your network? This is one of the most important charts to watch on a network. For example, when someone is complaining about network performance, take a look at the errors that are currently seen on the network. Identifying the type of errors can help pinpoint faulty network drivers, devices and cables.

Now... sometimes people look at this as the ‘ultimate’ indicator of network faults. It is not. These errors are simply the data link errors -- you need to look beyond that -- inside the packets.

If you aren’t familiar with the layers of the OSI model -- you are not reading the right book. Run (don’t walk) and get the “Introduction to Network Analysis” podbook -- pay special attention to the appendix that focuses on the flow of data on the network.

Broadcasts

Broadcast packets are sent to all devices on the network. At the MAC layer, they are addressed to 0xFF-FF-FF-FF-FF-FF (the broadcast address). Broadcasts have to be processed by all devices on the network, regardless of the network layer protocol they support.

For example, consider a NetWare broadcast on an Ethernet network that is received by an IP-based device. The IP-based device looks at the MAC header and sees the broadcast address -- “Wow! That packet is for me!” The IP-based device will perform all the necessary MAC-layer error checking (CRC, length, etc.) before looking at the protocol ID field to determine the desired network layer protocol. “Yuck -- I don’t support IPX! Phooey!” *<make spitting sound now>*

These broadcasts are an ugly thing... depending on the design of your network. Hubs forward all broadcasts. Switches forward all broadcasts. Stupid, ugly, lousy, pathetic, misconfigured routers forward all broadcasts.

NOTE

Do you get the idea that it’s a really, really bad idea to set up your routers to forward broadcasts? -- Laura

What protocols use broadcast? ARP, DHCP (discovery), IP RIP, SAP, IPX RIP, and more. Yuck.

Analyzers typically have multiple places that broadcast information is shown -- on the Sniffer, you can see the broadcast information in the Detail window (see Figure 1-6), but this only gives you an idea of how many broadcasts have occurred since the Dashboard window has been opened. You really need to look further into the type of broadcasts and the trends of broadcasts.

Often, broadcasts just sort of creep up on you. They just add up over time as you add devices, protocols, and applications on the network. Figure 1-7, however, shows a sudden increase in broadcasts. This just ain't normal, folks (yes, boss, red is bad). This indicates a fault in either an application, device or protocol.

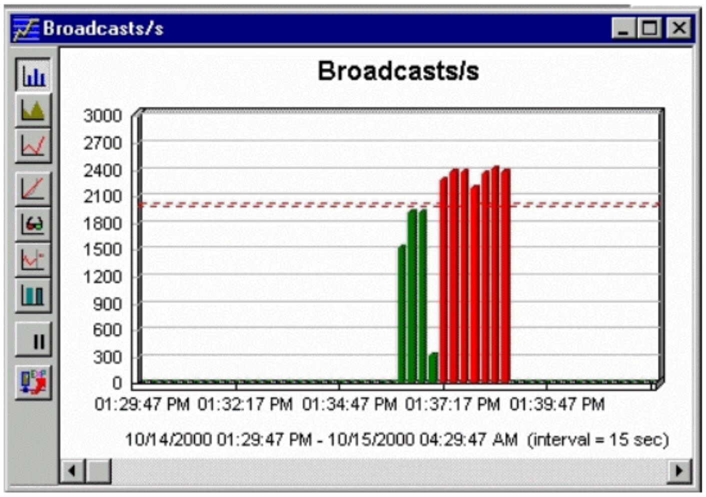


FIGURE 1-6. **Youch! What's wrong with this picture? Sudden spikes in broadcasts indicate a definite problem!**

NOTE

Sometimes, when companies complain of slowness, I have found lots of broadcasts on the network -- and most of the broadcasts are queries. Devices are looking for some service that is not located where it used to be. By checking out who is broadcasting and the protocol in use, we can figure out what the problem might be. For example, there used to be a print driver that had an ugly fault. When users booted their workstations, the driver would start sending an IPX RIP broadcast storm onto the network to find one of the known print servers on the network. Unfortunately, the driver used the wrong frame type -- duh. Can you say 'lousy programming?' -- Laura

Check out Figure 1-7. We've decided to see what type of traffic is hogging up this IPX network -- stinky! Wow. This network sucks. By looking further into the packets further, we can pinpoint who is broadcasting all these ugly little things.

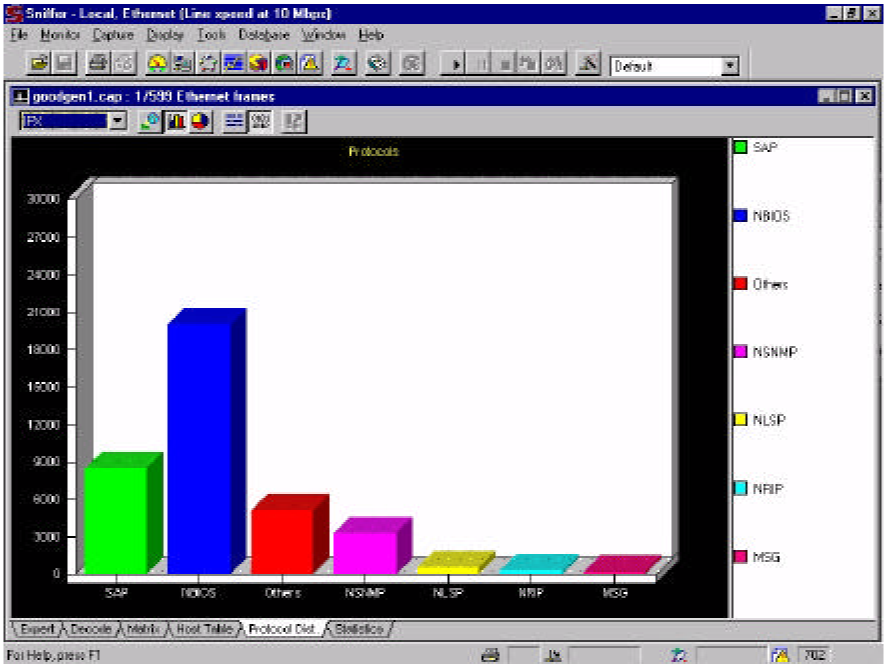


FIGURE 1-7. **An excessive number of IPX SAPs indicate a real problem on the network that is experiencing a broadcast storm. Looking into these SAP packets will indicate which service is missing or misconfigured.**

Analyze it! What is the broadcast rate on your network? What type of broadcasts do you see? Can you use some analyzer function to find out who the broadcasts are coming from?

How many broadcasts is 'too many'? This is one of those questions like, "how many collisions are too many"? When you start seeing systems dropping packets or sending some type of 'delay packets' to another device and you notice the broadcast rate is high.... consider lowering the broadcast rate.

“How?” you ask. Well... look for any unnecessary broadcasts. NetBIOS, for example. If you’re not using NetBIOS for anything on your network (like WINS), why don’t you dump it? In other cases, you can set up a router to reply to broadcast queries, such as NetWare SAP (Service Advertising Protocol) queries. You can also look for folks broadcasting with protocols that you don’t support on your network (such as HP JetDirects that send AppleTalk broadcasts onto the network unnecessarily).

NOTE

I’m not fond of AppleTalk, but I have nothing against Macintoshes... they’re so cute. -- Laura

Multicasts

Multicasts are packets sent to a group of devices, such as all OSPF routers. These packets are almost as ugly as typical broadcast packets. I say ‘almost’ because many routers do not forward multicasts by default, thereby reducing the overall effect of widespread multicasting.

LINK

You can get the list of assigned multicast addresses at **www.iana.org**. Select the “Protocol Numbers and Assignment Services” and then click on the letter “M” and scroll down to the Multicast Address section.

HANDS-ON

Analyze it! How much multicast traffic do you have on your network? Who is sending those multicasts? You might want to check out Chapter 2, “Capture and Display Filtering” to learn how to identify the source of multicasts. Write up a short list of your multicast destination addresses and look up their assignments at **www.iana.org**. If your network is flooded with multicasts -- find the source... do you have some blabbering application on your network (like a music application that plays in the background on the network all day)? Take a look at Internet Group Management Protocol (IGMP), RFC 2236 by W. Fenner dated November 1997, as well.

Packet Size Distribution - Size Does Matter!

This is an interesting one... first of all, you need to know your minimum and maximum packet sizes. For Ethernet, the minimum packet size is 64 bytes -- the maximum is 1518 bytes. Knowing that -- if you see most of your packets are 64 bytes in length, it’s not a pretty picture.

Take a look at Figure 1-8. Notice that most of this network's packet sizes are between 256 and 511 bytes. Pretty shrimpy little packets. If I saw this at a company, I'd look at the type of traffic cruising the wire to see what's going on. If most of the traffic is for command sequences, then I would not be surprised to see smaller packet sizes. If, however, most of the traffic is file transfer traffic, then I'd figure we have pretty inefficient network.

What can you do about small packet sizes? Well... you can look at the type of application that you're using for transferring packets. You can also look at the protocol being used to transfer data. For example, SPX is a protocol that uses minimum sized packets by default. Packets can also be fragmented if there's any sort of small MTU (maximum transmission unit) network along their path.

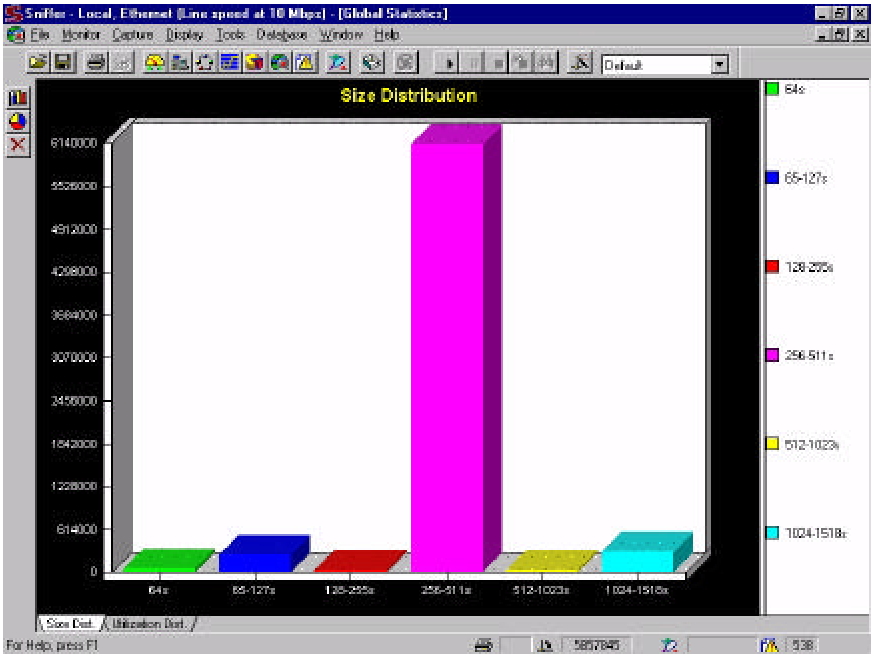


FIGURE 1-8. Aren't they cute? No.

HANDS-ON

Analyze it! How does your packet size distribution look? What could be the cause of those smaller packets? In Chapter 3 you learn to perform application analysis. Remember to look at the typical packet size used to download data within an application -- this statistic says much for the general ‘drag’ of an application on the network. You will probably find that database applications use lots of smaller packets (especially when they read out-of-sequence data in a file) whereas file transfer applications read data with larger packets (hopefully).

Hosts

Who’s talking? Who’s talking the most? There are many ways to view this information. Consider Figure 1-9 that shows the Sniffer matrix view -- cool! We can, with a simple glance, see who is talking to whom. We can get a feel for which devices are critical (look at the devices that are being pointed to in the matrix).

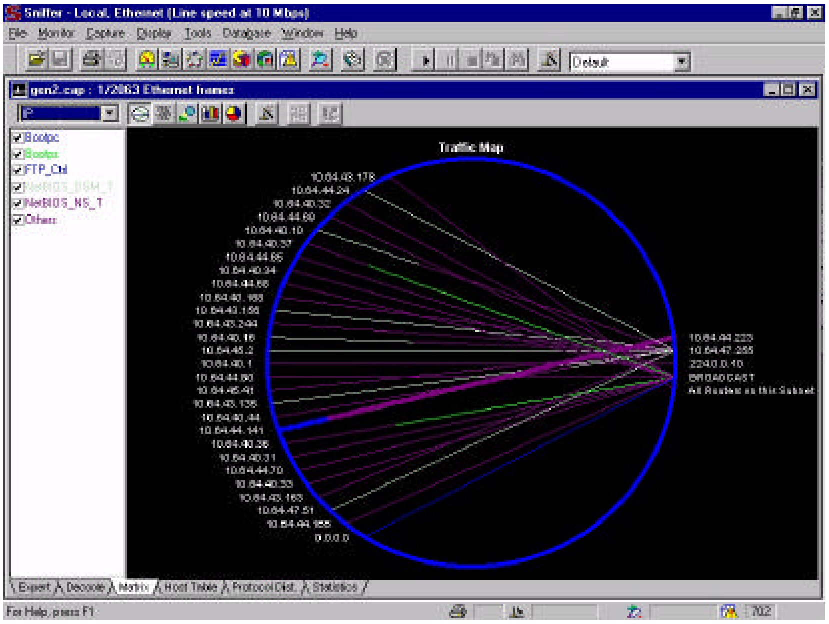


FIGURE 1-9. **The matrix shows a definite arrow point.**

The matrix is just absolutely cool. It displays the conversation pairs on the network in a wonderfully graphical way. You can also see the conversations color-coded to indicate the protocols in use -- if you purchased the PDF version of this book, you can see the colors in the figure above for example.

There are many different types of communications that show up on the matrix. The most common type of communication on the network is from client to server or router. You may see the ‘point’ style, as shown on the top of Figure 1-10. There are often, however, side conversations that are clearly visible on the matrix. In these cases, you need to look at why there are other conversations going on. Could it be that someone has loaded an FTP server process on their system? Is there a rogue DHCP server out there (don’t ya just hate that)? Is there some lousy Win95 box out there with “I want to share my files/printer” checked? Did someone startup an NT workstation with server services enabled? Who ARE these people?

Finally, you might find that your network is just plain scattered. In other words, you have server processes loaded on numerous client stations. That can be a bit ugly on the matrix, but... hey, if that's what you want on your network -- enjoy. <*make that spitting sound again now*>.

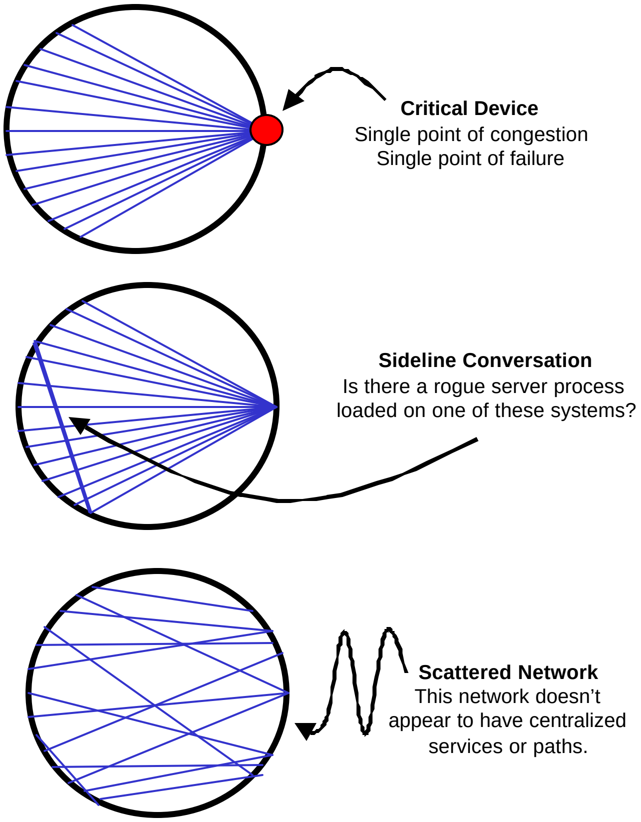

Critical Device
Single point of congestion
Single point of failure
Sideline Conversation
Is there a rogue server process
loaded on one of these systems?
Scattered Network
This network doesn't
appear to have centralized
services or paths.

FIGURE 1-10. **The matrix view of a network illustrates the many traffic patterns seen and helps you identify critical devices.**

HANDS-ON

Analyze it! Who are your top talkers? Do routers and servers show up as the most 'popular' devices on the network? Did broadcast show up as a popular destination? Be careful of analyzing a switched network -- see Appendix B, "Analyzing Switched Networks," for more details.

Protocols

This is really a fun statistic to gather on a network. Often, I find network administrators or technicians are shocked to see which protocols are using up their precious bandwidth. One recent use for this statistic is the migration to NetWare 5's pure IP architecture -- NCP (NetWare Core Protocol) over IP with no encapsulation. If you've recently performed a migration, check your protocol distribution -- got any old IPX/SPX around? What about SAP or IPX RIP?

Figure 1-11 shows the breakdown of the various protocols crossing this network. This is especially important when you want to know

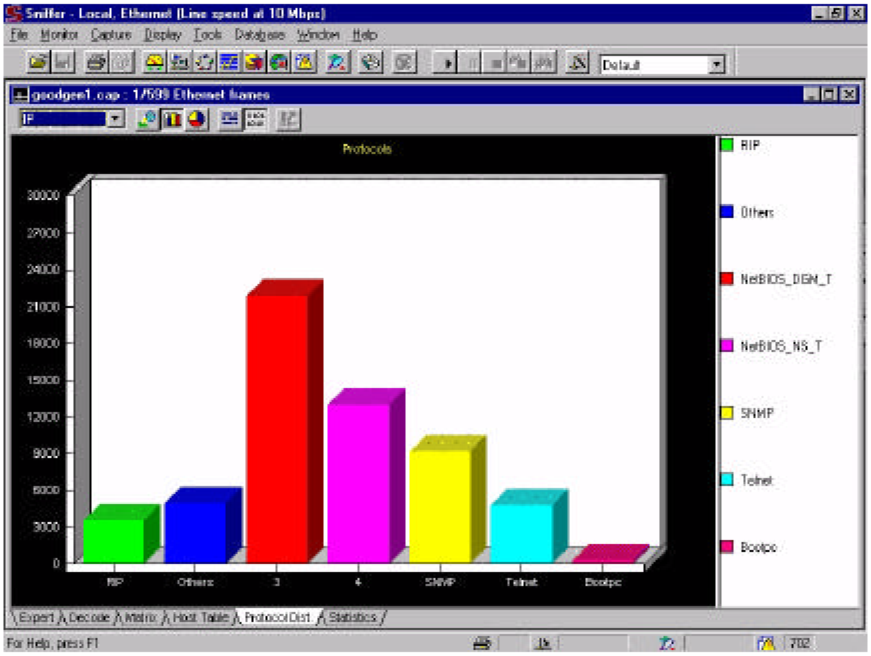


FIGURE 1-11. Yuck.

HANDS-ON

Analyze it! What protocols are running on your network? Ideally, why don't you check out this information over a week or so. Do you see any communications listed as 'other'? These are typically communications that can't be categorized as TCP/IP, IPX/SPX or AppleTalk. For example, Cisco's CDP (Cisco Discovery Protocol) and the BPDU (Bridge Protocol Data Units) used by Spanning Tree may show up under this heading.

Ok... so we've looked at the basic statistics on the network including packets per second, utilization, errors, packet sizes and conversations. Now let's look at alarms and alerts.

Alarms and Alerts

The following is a sample list of the types of alarms you will find on most analyzers:

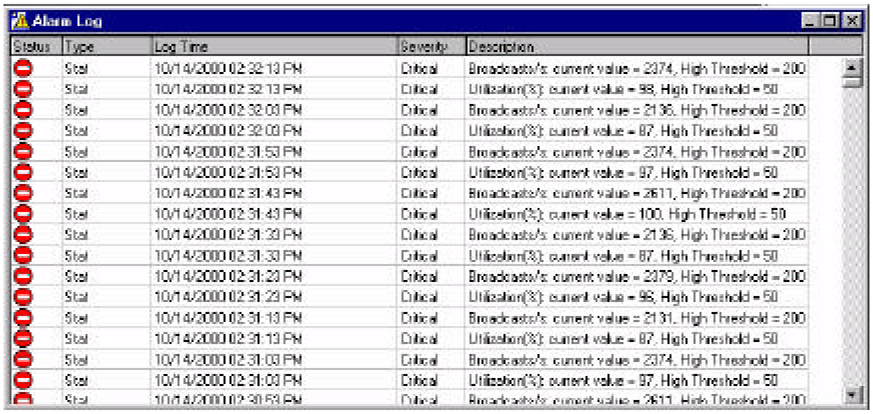
- Packets/Second exceeded threshold
- Broadcasts/Second exceeded threshold
- Errors/Second exceeded threshold
- Device xxx Not Found

The following is a list of alarms/alerts that I wish I had (note that some analyzers have included similar alarms in their latest products, however, they may have used different language):

- Lousy application present
- Redundant file read
- Reconnaissance probe underway
- Clueless user on the network
- User environment configured improperly

I'm sure you could add to this list -- and maybe you should. Then submit those ideas to the analyzer manufacturers!

Figure 1-12 shows the alarm report log from my system.



Status	Type	Log Time	Severity	Description
⛔	Stat	10/14/2000 02:32:13 PM	Critical	Broadcasts/s: current value = 2374, High Threshold = 200
⛔	Stat	10/14/2000 02:32:13 PM	Critical	Utilization(%): current value = 98, High Threshold = 50
⛔	Stat	10/14/2000 02:32:03 PM	Critical	Broadcasts/s: current value = 2136, High Threshold = 200
⛔	Stat	10/14/2000 02:32:03 PM	Critical	Utilization(%): current value = 87, High Threshold = 50
⛔	Stat	10/14/2000 02:31:53 PM	Critical	Broadcasts/s: current value = 2374, High Threshold = 200
⛔	Stat	10/14/2000 02:31:53 PM	Critical	Utilization(%): current value = 97, High Threshold = 50
⛔	Stat	10/14/2000 02:31:43 PM	Critical	Broadcasts/s: current value = 2611, High Threshold = 200
⛔	Stat	10/14/2000 02:31:43 PM	Critical	Utilization(%): current value = 100, High Threshold = 50
⛔	Stat	10/14/2000 02:31:33 PM	Critical	Broadcasts/s: current value = 2136, High Threshold = 200
⛔	Stat	10/14/2000 02:31:33 PM	Critical	Utilization(%): current value = 87, High Threshold = 50
⛔	Stat	10/14/2000 02:31:23 PM	Critical	Broadcasts/s: current value = 2379, High Threshold = 200
⛔	Stat	10/14/2000 02:31:23 PM	Critical	Utilization(%): current value = 96, High Threshold = 50
⛔	Stat	10/14/2000 02:31:13 PM	Critical	Broadcasts/s: current value = 2131, High Threshold = 200
⛔	Stat	10/14/2000 02:31:13 PM	Critical	Utilization(%): current value = 87, High Threshold = 50
⛔	Stat	10/14/2000 02:31:03 PM	Critical	Broadcasts/s: current value = 2374, High Threshold = 200
⛔	Stat	10/14/2000 02:31:03 PM	Critical	Utilization(%): current value = 97, High Threshold = 50
⛔	Stat	10/14/2000 02:30:53 PM	Critical	Broadcasts/s: current value = 2611, High Threshold = 200

FIGURE 1-12. **Alarm report log.**

As you can see, the network I'm on right now isn't too healthy. Yipes -- 98% utilization? Tons of broadcasts!? This is a really bad day on the network. Of course this is a simple scenario -- it's easy to see what the problem is. The next step is to find the source of the broadcasts, the type of broadcasts and reason for the broadcasts.

Watch the Default Alarm Settings

How are the default alarm settings determined? Well... there are a number of ways. An analyzer manufacturer may look at the average network size and figure out the appropriate alarm settings based on that. Or... the product manager may go to an engineer and ask him/her/it to define default alarm settings before the next product status meeting (thereby causing the engineer to guess at the appropriate settings). Then again, the product team may just sit down to a nice cold beer over lunch and write their ideas on napkins -- then comparing them to see which settings are most popular.

NOTE

I've seen analyzer settings that could have been defined in any of the three methods mentioned earlier. Don't trust those settings -- they are some else's guidelines for how your network should function -- with absolutely no concern for the design/function of your specific network. -- Laura

I know this can seem quite daunting, but you must **learn** your network first -- then determine the settings.

Setting Your Own Alarm Thresholds

Ok... here's how to do this -- start following your network's statistics and trends over at least one month. You will use this information to define your alarm settings. For example, let's discuss how to set your broadcasts/second alarm threshold.

Look at the typical number of broadcasts you experience and get a feel for who is broadcasting and why. If those broadcasts are necessary and acceptable on your network, set a broadcasts/second alarm threshold around the top of the trend peak.   You should be notified when your broadcast rate begins rising toward 'concern' levels.

Use the same method to set your kbytes/second, multicasts/second, and packets/second thresholds. This will give you a more appropriate alarm configuration for your network.

Now... when we get into the filtering stuff (in the next chapter), keep in mind that many analyzers can trigger an alarm when a packet meets a filter criteria. For example, you could make a filter for all SNMP query traffic that comes from devices that are NOT your management stations. Kinda suspect traffic, eh? With some of the really cool analyzers (such as EtherPeek and Sniffer), you can set up an alarm based on those filter criteria -- 'Put up the ol' red flag when SNMP queries come from other devices!' Ahhh.... this just gives me chills!

False Positives

A false positive is an alarm that has been triggered by a normal communication process. False positives can lead you down a fruitless and time consuming troubleshooting path if you don't spot them right away.

NOTE

The most common false positive I encounter is the 'duplicate read' alarm false positive that some of the analyzers display. Analyzer may look at burst mode file reads (especially in the area of NetWare's IPX burst mode operations) and report that the same file is being read over and over and over and over again. In reality, however, the client is reading the file at a different offset. -- Laura

Another false positive I've encountered indicates 'slow reads' on the network when in reality, the server is taking time to build a reply based on data that is located in several files.

When an alarm is logged, look at the traffic that triggered the alarm to verify the problem. This is especially important when an alarm (or set of alarms) may indicate a possible cyber attack on the network.

False Negatives

False negatives occur when a problematic network condition exists, but no alarm is triggered. In this case, either the alarm thresholds are set incorrectly, or the analyzer can't see or identify the problem. False negatives give you a sense of security and happiness -- doom lurks around the corner.

Sometimes, you can build advanced filters (as depicted in Chapter 2, “Capture and Display Filtering) to help identify these problem situations. In other situations, you may need to perform periodic checks of your network traffic to identify any unusual patterns or behaviors.

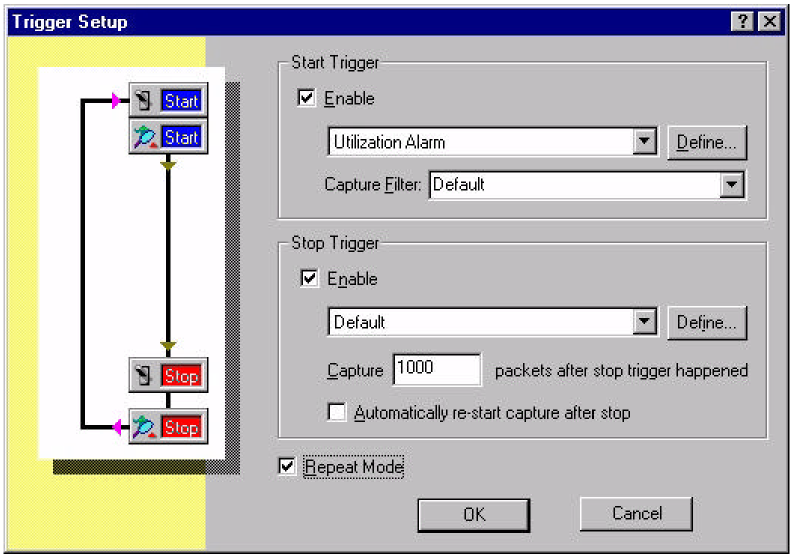
NOTE

If you do find an area that should be tracked by the analyzers, let the manufacturers know. Too often, the manufacturers have to ***drag*** *this needed information from their clients. -- Laura*

Using Alarms as Triggers

What if you need to capture specific traffic in the middle of the night? You can set up triggers on your network that are based on the alarm settings you configured. For example, perhaps you want your analyzer to start capturing data once the utilization threshold of 80% is exceeded.

Figure 1-13 shows a trigger setup for just such a task. The Sniffer will start capturing packets once the Utilization threshold (80%) is reached. The analyzer will then capture the 1000 packets that follow the alarm trigger and then restart the trigger again.



Trigger Setup

Start Trigger

☑ Enable

Utilization Alarm

Capture Filter: Default

Stop Trigger

☑ Enable

Default

Capture 1000 packets after stop trigger happened

□ Automatically re-start capture after stop

☑ Repeat Mode

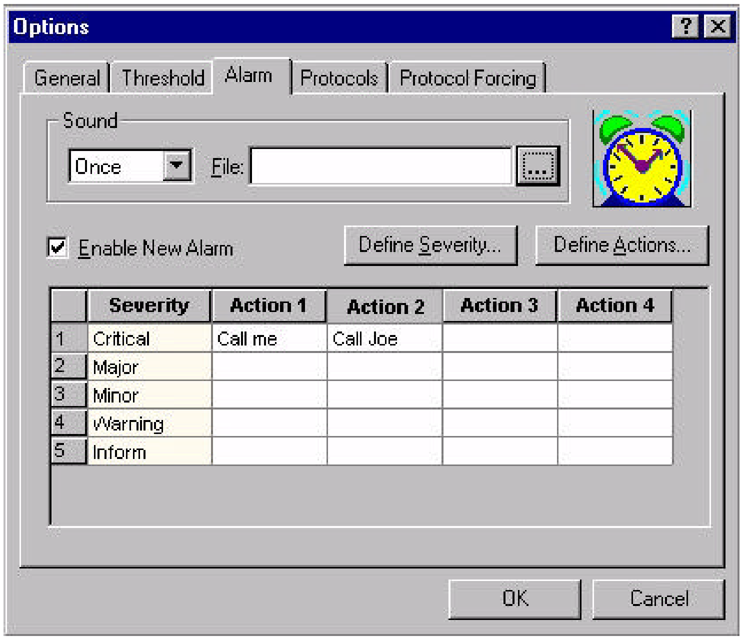
FIGURE 1-13. **We can easily configure this analyzer (the Sniffer) to start capturing packets automatically when an alarm (Utilization) has been triggered.**

Notification Options

Ideally, your analyzer should be able to notify you when a critical alarm is triggered.

In the example shown in Figure 1-14, I configured the analyzer to send a message to me on my beeper when any critical alarm is triggered. I've also configured my analyzer to call my backup, Joe, as well.

The difference between the two actions in this case is that I configured each action for a specific time. If the alarm occurs during the hours of 10:00 and 11:00 a.m., my beeper will be contacted. Joe, however, will be notified whenever the alarm is triggered -- whether it is 11:00 a.m. or 2:00 a.m. Poor Joe.



Sound: Once File:

☑ Enable New Alarm

	Severity	Action 1	Action 2	Action 3	Action 4
1	Critical	Call me	Call Joe		
2	Major				
3	Minor				
4	Warning				
5	Inform				

FIGURE 1-14. **He he he... Joe will be called day or night whenever a critical alarm is triggered.**

Trends

Trends are used to identify overall patterns for traffic flow. Any decent analyzer should provide you with nice graphical trend charts. There are two types of trends:

- Short-term trends
- Long-term trends

NOTE *Trend graphs and charts are extremely effective when presenting network information to non-technical management. I have seen the light bulb go off over the head of many net-challenged individuals when they view a nice trend graph of their increasing bandwidth traffic or their already deadly error rates. -- Laura*

Short-Term Trends

A short-term trend depicts the current traffic. The trend information might go back as far as one or two hours, but the sampling time (the times that the network traffic is reviewed and averaged over) is usually small -- perhaps in 15 second intervals as shown in Figure 1-15.

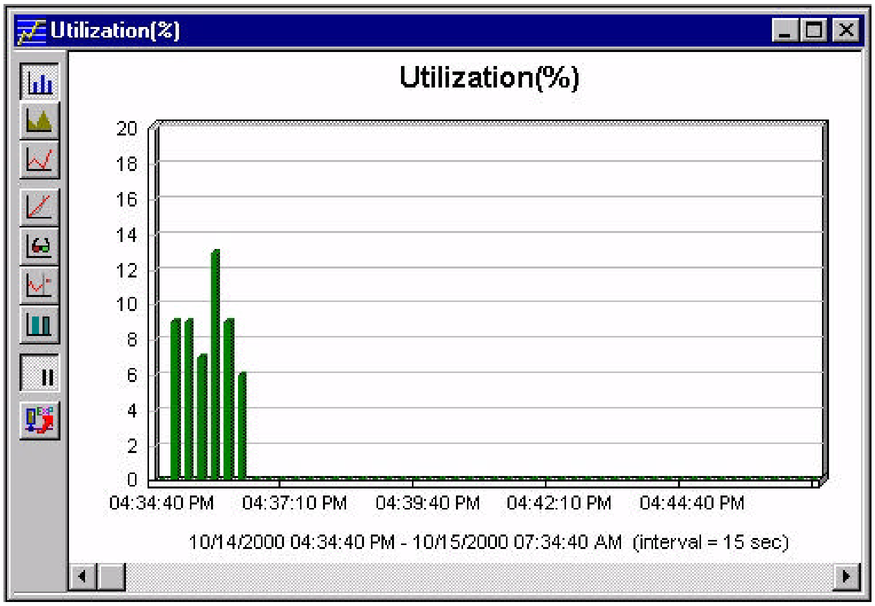


FIGURE 1-15. **Short-term trends can indicate the current network condition.**

Short-term trends are great to use when you need to know what is happening right now. For example, if someone calls you and states that the network is slow, you can grab a quick trend sample to see if you have a high bandwidth usage rate or broadcast rate right now. If your network traffic rate appears relatively low, you can focus in on problems with specific communication pairs.

Remember that you are only looking at a snapshot of network communications. Short-term trends do not provide a good measurement for typical day in, day out communication patterns. In other words, short-term trends give you the current 'mood' of the network -- they do not define the personality. For that, you need to examine your long-term trends.

NOTE

In some instances, analyzers don't differentiate between short-term and long-term trends. Sniffer is one of those analyzers. As you'll notice in the next section, long-term trends are gathered by simply keeping the history windows open over days and weeks. You might want to increase the sampling rate or consider buying a bigger hard drive and a few reams of paper for the charts!

Long-Term Trends

Long-term trends identify the network communication pattern over days, weeks, and months. By examining your long-term trends, you can really see the personality of your network. Does your network typically ‘wake up’ at 8:00 a.m. (log in time) and go to ‘sleep’ at 5:00 p.m.? Does your network have afternoon naps (logouts during lunch hour)? Do you see busy activity at the end of the month when reports are due?

These long-term trends can help you determine what’s normal for your network. Figure 1-16 shows the long-term broadcast trend information.

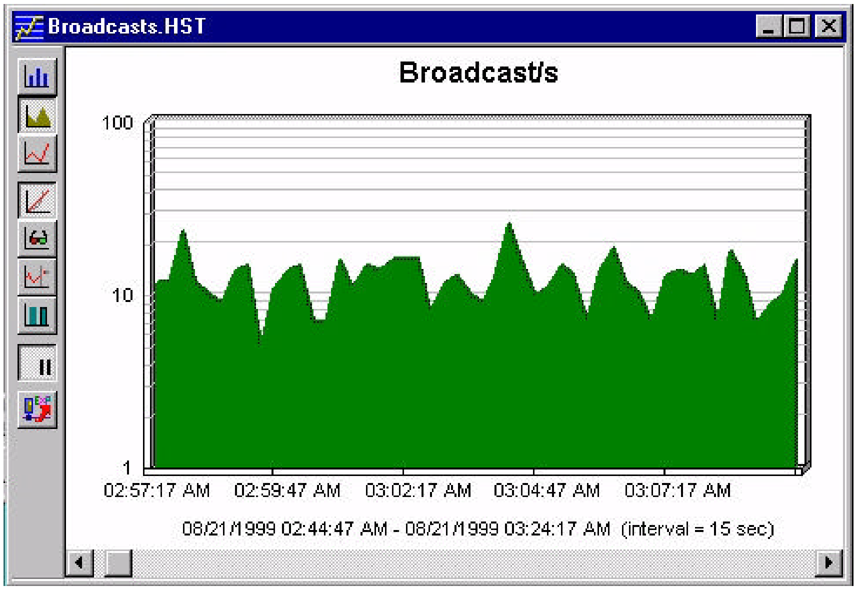


FIGURE 1-16. **Watching your trends over time can help you identify peaks in traffic trends.**

NOTE *In the last several years, we have seen the idea of a ‘stable network design’ go out the window. Today’s networks are changing and evolving at such a rate that we may not be able to get a very long long-term trend -- there’s never a point where the network stays the same long enough (architecturally or functionally). -- Laura*

Exporting Graphics Into a Report

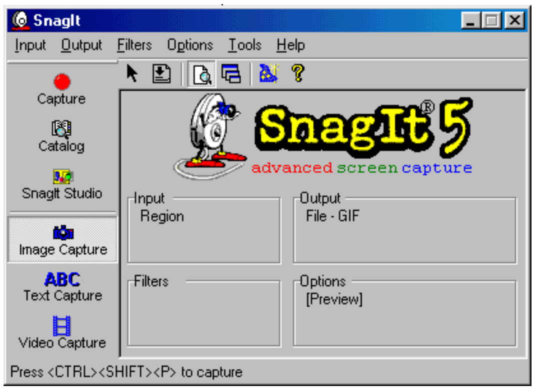
Some analyzers can export their graphics into a nice format, but they are often not depicting the part of the screen or graph that I want to include in my report. This is why I always have a resident screen shot program loaded on my analyzer system.

Although I've used many screen capture programs over the years, I have standardized on a screen shot program called SnagIt/32 by TechSmith. Easy to use and cooperative with other applications, SnagIt/32 is a blessing.

LINK

You can find out more information about SnagIt/32 at **www.techsmith.com**. There's a sample report with graphics online as well. Check out **www.packet-level.com**.

Figure 1-17 shows the SnagIt/32 control window.



Input
Region

Output
File - GIF

Filters

Options
[Preview]

Press <CTRL><SHIFT><P> to capture

FIGURE 1-17. **SnagIt is configured to capture the active window in GIF file format (one of the many formats available).**

One of the nicest features of SnagIt is the ability to autoscroll a screen as it is capturing it. For example, sometimes I need to capture a screenshot of the entire decode of a packet. This information runs off a single screen, however. By enabling the autoscroll feature, SnagIt captures the portion of the packet that I can see and scrolls down to capture the rest of the packet as well. Nice!

Patterns

Pattern analysis is not taught anywhere, as far as I can tell. That’s too bad. I have caught numerous problems by simply identifying unusual traffic patterns. I believe that all analysts should have basic pattern analysis skills and learn to trust their instincts a bit.

In pattern analysis, you open the summary screen only -- minimize the decode and hex dump screens on the analyzer.

These are the traffic patterns typically seen:

- Request - Reply, Request - Reply (Commands)
- Request - Reply, Request - Reply (Slow File Transfer)
- Request, Request, Request (Service Lookup)
- Request - Reply - Reply - Reply (Windowed File Transfer)
- Reply - Reply - Reply (Information Distribution)
- Request - Request - Reply (Weird Problem)

Let’s look closely at each of these patterns -- I’ll try to give you examples of each one along the way.

Request - Reply, Request - Reply (Commands)

This is a good pattern and quite normal for command sequences. For example, in Figure 1-18, you can see the initial FTP connection process. During this connection process, the communication follows a ‘ping pong’ request-reply pattern.

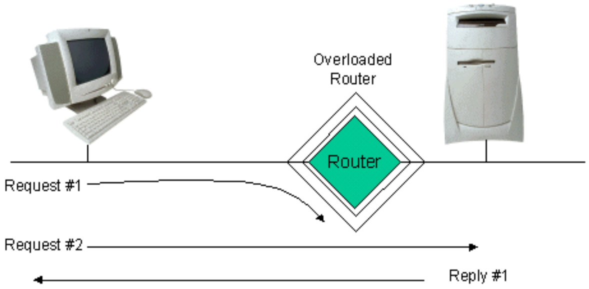


FIGURE 1-18. **A sample summary screen in which we'll identify some patterns.**

Request - Reply, Request - Reply (Slow File Transfer)

This is the same process as shown above, but it is now being used to transfer data -- this is the slowest method for getting data from one device to another. You are limited by the maximum payload (or data area) size of the packet. On an Ethernet network, the maximum payload size is 1500 bytes.

NOTE

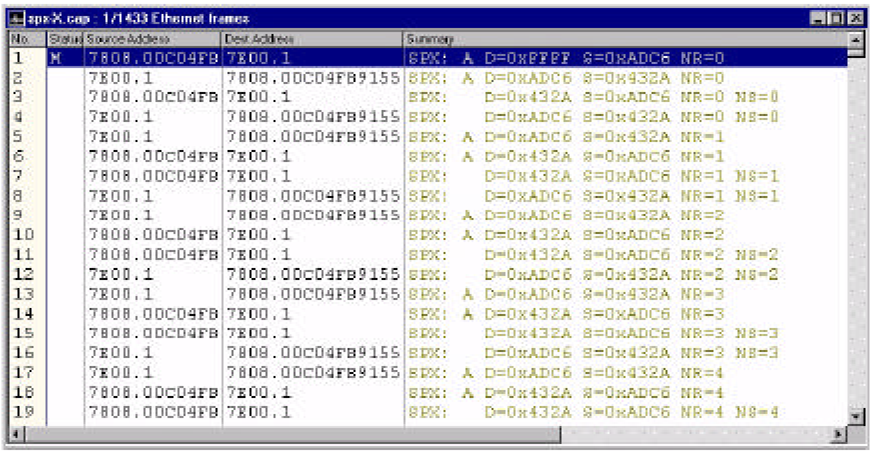
Although the maximum packet size is 1518 for Ethernet, the data portion can only be 1500 bytes long. -- Laura

When will you see this type of operation for file exchange? You may see this type of ping-pong operation in the following circumstances:

- non-sequential offset reads of a database file
- file read process that only supports single packet window
- application limit on maximum payload size
- ineffective network communications causing minimum window size

Figure 1-19 shows a trace of a non-windowing protocol, SPX (Sequenced Packet Exchange). Although there are two versions of this protocol, the widely implemented version 1 only supports a window size of 1.

In Figure 1-19 we know some data is being exchanged because the length field column indicates larger packets (now I didn't say 'large' -- I said 'larger' -- remember that standard SPX has a maximum packet size of 576 bytes).



spx-X.cap : 1/1433 Ethernet frames

No.	Status	Source Address	Dest Address	Summary
1	M	7808.00C04FB	7E00.1	SPX: A D=0xFFFF S=0xADC6 NR=0
2		7E00.1	7808.00C04FB9155	SPX: A D=0xADC6 S=0x432A NR=0
3		7808.00C04FB	7E00.1	SPX: D=0x432A S=0xADC6 NR=0 NS=0
4		7E00.1	7808.00C04FB9155	SPX: D=0xADC6 S=0x432A NR=0 NS=0
5		7E00.1	7808.00C04FB9155	SPX: A D=0xADC6 S=0x432A NR=1
6		7808.00C04FB	7E00.1	SPX: A D=0x432A S=0xADC6 NR=1
7		7808.00C04FB	7E00.1	SPX: D=0x432A S=0xADC6 NR=1 NS=1
8		7E00.1	7808.00C04FB9155	SPX: D=0xADC6 S=0x432A NR=1 NS=1
9		7E00.1	7808.00C04FB9155	SPX: A D=0xADC6 S=0x432A NR=2
10		7808.00C04FB	7E00.1	SPX: A D=0x432A S=0xADC6 NR=2
11		7808.00C04FB	7E00.1	SPX: D=0x432A S=0xADC6 NR=2 NS=2
12		7E00.1	7808.00C04FB9155	SPX: D=0xADC6 S=0x432A NR=2 NS=2
13		7E00.1	7808.00C04FB9155	SPX: A D=0xADC6 S=0x432A NR=3
14		7808.00C04FB	7E00.1	SPX: A D=0x432A S=0xADC6 NR=3
15		7808.00C04FB	7E00.1	SPX: D=0x432A S=0xADC6 NR=3 NS=3
16		7E00.1	7808.00C04FB9155	SPX: D=0xADC6 S=0x432A NR=3 NS=3
17		7E00.1	7808.00C04FB9155	SPX: A D=0xADC6 S=0x432A NR=4
18		7808.00C04FB	7E00.1	SPX: A D=0x432A S=0xADC6 NR=4
19		7808.00C04FB	7E00.1	SPX: D=0x432A S=0xADC6 NR=4 NS=4

FIGURE 1-19. **SPX requires an acknowledgment after every data packet sent.**

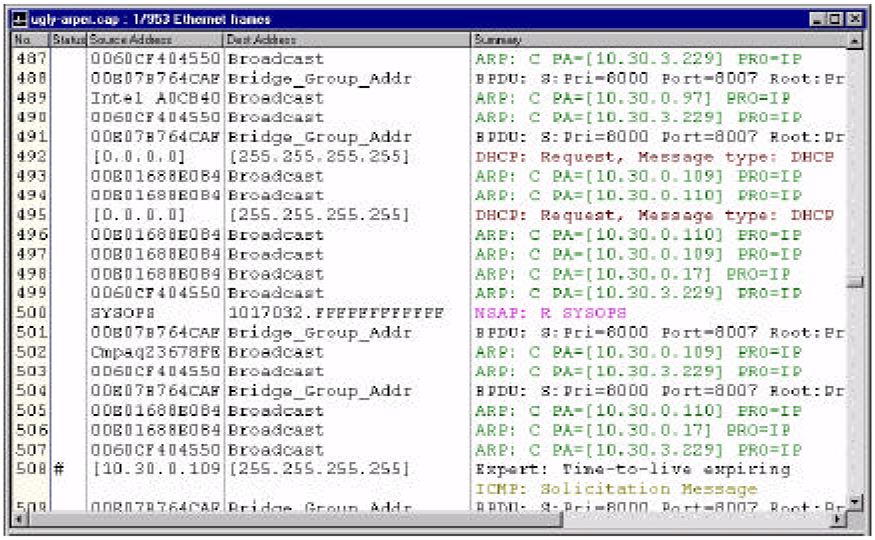
Request, Request, Request (Service Lookup)

This type of pattern is a waste of bandwidth -- some device is doing endless lookups without receiving a reply. Typically, we like to see some sort of timeout mechanism kicking in to stop the repetitive unanswered queries. Consider the case of a process that suddenly locks up -- for instance, an HTTP server process. As the client is making HTTP queries to the server, the replies suddenly stop. The client will continue making queries until a timeout occurs.

If no timeout occurs, the request-request-request process continues. Some examples of this pattern are:

- unanswered IPX RIP queries
- unanswered ARP broadcasts
- unanswered DHCP discover broadcasts

Figure 1-20 shows a lousy request-request-request pattern (see all the ARPs?). As you can see, this is not a good situation -- it doesn't look good; it doesn't feel good.



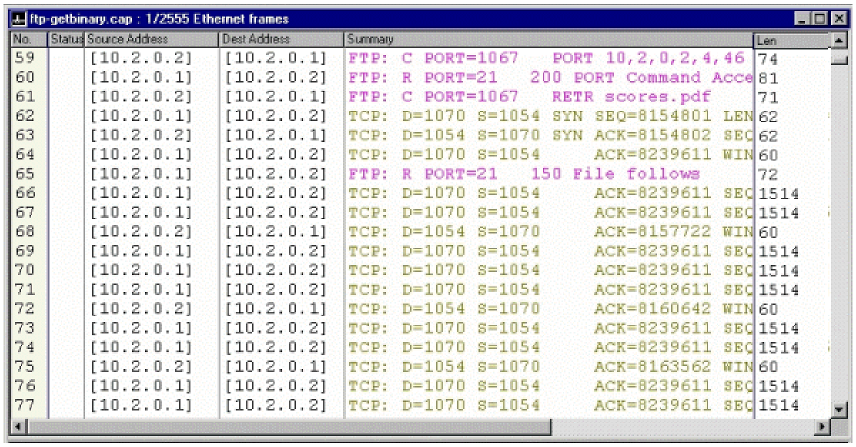
```
ugly-arper.cap : 1/953 Ethernet frames
No.  Status  Source Address  Dest Address          Summary
487          0060CF404550    Broadcast             ARP: C PA=[10.30.3.229] PRO=IP
488          00E07B764CAF    Bridge_Group_Addr     BPDU: S:Pri=8000 Port=8007 Root:Pr
489          Intel A0CB40    Broadcast             ARP: C PA=[10.30.0.97] PRO=IP
490          0060CF404550    Broadcast             ARP: C PA=[10.30.3.229] PRO=IP
491          00E07B764CAF    Bridge_Group_Addr     BPDU: S:Pri=8000 Port=8007 Root:Pr
492          [0.0.0.0]       [255.255.255.255]     DHCP: Request, Message type: DHCP
493          00E01688E084    Broadcast             ARP: C PA=[10.30.0.109] PRO=IP
494          00E01688E084    Broadcast             ARP: C PA=[10.30.0.110] PRO=IP
495          [0.0.0.0]       [255.255.255.255]     DHCP: Request, Message type: DHCP
496          00E01688E084    Broadcast             ARP: C PA=[10.30.0.110] PRO=IP
497          00E01688E084    Broadcast             ARP: C PA=[10.30.0.109] PRO=IP
498          00E01688E084    Broadcast             ARP: C PA=[10.30.0.17] PRO=IP
499          0060CF404550    Broadcast             ARP: C PA=[10.30.3.229] PRO=IP
500          SYSOPS          1017032.FFFFFFFFFFFF  NSAP: R SYSOPS
501          00E07B764CAF    Bridge_Group_Addr     BPDU: S:Pri=8000 Port=8007 Root:Pr
502          Cmpaq23678FE    Broadcast             ARP: C PA=[10.30.0.109] PRO=IP
503          0060CF404550    Broadcast             ARP: C PA=[10.30.3.229] PRO=IP
504          00E07B764CAF    Bridge_Group_Addr     BPDU: S:Pri=8000 Port=8007 Root:Pr
505          00E01688E084    Broadcast             ARP: C PA=[10.30.0.110] PRO=IP
506          00E01688E084    Broadcast             ARP: C PA=[10.30.0.17] PRO=IP
507          0060CF404550    Broadcast             ARP: C PA=[10.30.3.229] PRO=IP
508  #       [10.30.0.109    [255.255.255.255]     Expert: Time-to-live expiring
                                                   ICMP: Solicitation Message
509          00E07B764CAF    Bridge_Group_Addr     BPDU: S:Pri=8000 Port=8007 Root:Pr
```

FIGURE 1-20. ARP! ARP! ARP!

Request - Reply - Reply - Reply (Windowed File Transfer)

This is a good pattern that is seen on file transfers and protocols that use a window. For example, a host sends a single request for data at a specific offset in a file. The request is for more than one-packet's worth of data.

The reply consists of multiple packets, as shown in Figure 1-21. Two examples of this technology are TCP-based file transfer applications (such as FTP) and Novell's burst mode IPX-based file transfer process.



ftp-getbinary.cap : 1/2555 Ethernet frames

No.	Status	Source Address	Dest Address	Summary	Len
59		[10.2.0.2]	[10.2.0.1]	FTP: C PORT=1067 PORT 10,2,0,2,4,46	74
60		[10.2.0.1]	[10.2.0.2]	FTP: R PORT=21 200 PORT Command Acce	81
61		[10.2.0.2]	[10.2.0.1]	FTP: C PORT=1067 RETR scores.pdf	71
62		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 SYN SEQ=8154801 LEN	62
63		[10.2.0.2]	[10.2.0.1]	TCP: D=1054 S=1070 SYN ACK=8154802 SEQ	62
64		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 WIN	60
65		[10.2.0.1]	[10.2.0.2]	FTP: R PORT=21 150 File follows	72
66		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514
67		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514
68		[10.2.0.2]	[10.2.0.1]	TCP: D=1054 S=1070 ACK=8157722 WIN	60
69		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514
70		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514
71		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514
72		[10.2.0.2]	[10.2.0.1]	TCP: D=1054 S=1070 ACK=8160642 WIN	60
73		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514
74		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514
75		[10.2.0.2]	[10.2.0.1]	TCP: D=1054 S=1070 ACK=8163562 WIN	60
76		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514
77		[10.2.0.1]	[10.2.0.2]	TCP: D=1070 S=1054 ACK=8239611 SEQ	1514

FIGURE 1-21. **A close look at the packet sizes and the RETR (retrieve) command indicates that a windowed file transfer is underway.**

Reply - Reply - Reply (Information Distribution)

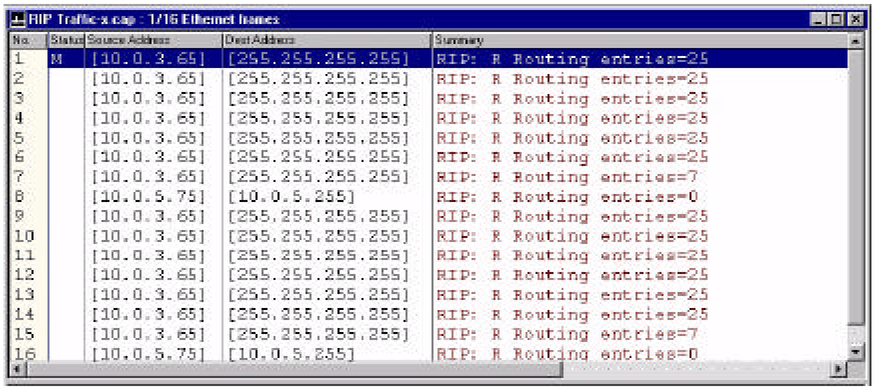
This type of communication is not very pretty on the network. It is typically a chatty informational protocol, such as IPX RIP and IP RIP. Each of these routing protocols sends routing information out onto the network periodically. IPX RIP routers send routing broadcasts out every 60 seconds. IP RIP version 1 sends routing broadcasts out every 30 seconds. (IP RIP version 2 sends multicast packets instead of broadcast.)

NOTE

Consider turning RIP off on all interfaces that do not support a network with another router. There is no reason to RIP in that direction -- no one's listening.

These are relatively stupid processes and they have been replaced by much less chatty, more intelligent link state routing protocols.

Figure 1-22 shows the traffic from an IP RIP router. You can see the destination address is 255.255.255.255 (broadcast) indicating that all IP devices must look at the packet contents.



RIP Traffic-x.cap : 1/16 Ethernet frames

No.	Status	Source Address	Dest Address	Summary
1	M	[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
2		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
3		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
4		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
5		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
6		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
7		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=7
8		[10.0.5.75]	[10.0.5.255]	RIP: R Routing entries=0
9		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
10		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
11		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
12		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
13		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
14		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=25
15		[10.0.3.65]	[255.255.255.255]	RIP: R Routing entries=7
16		[10.0.5.75]	[10.0.5.255]	RIP: R Routing entries=0

FIGURE 1-22. **IP RIP broadcasts are ok.... but OSPF multicasts are better.**

Request - Request - Reply (Weird Problem)

In this instance, a client’s requests are either being destroyed, routing to a dead end or timing out. This causes the client to retransmit the request as shown in Figure 1-23. The single reply indicates that the server side only received a single request.

If this type of pattern shows up consistently on the network, you may want to trace the requests to see what happens to them.

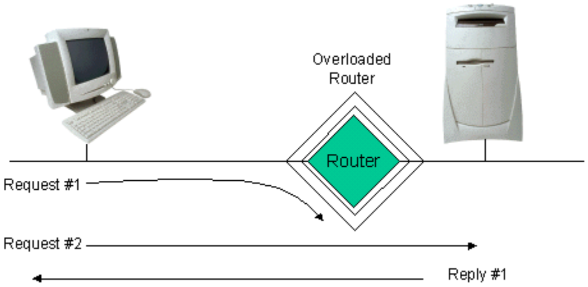


FIGURE 1-23. **Hello.....? Where’s the packet going? No device should have to make multiple requests to get an answer.**

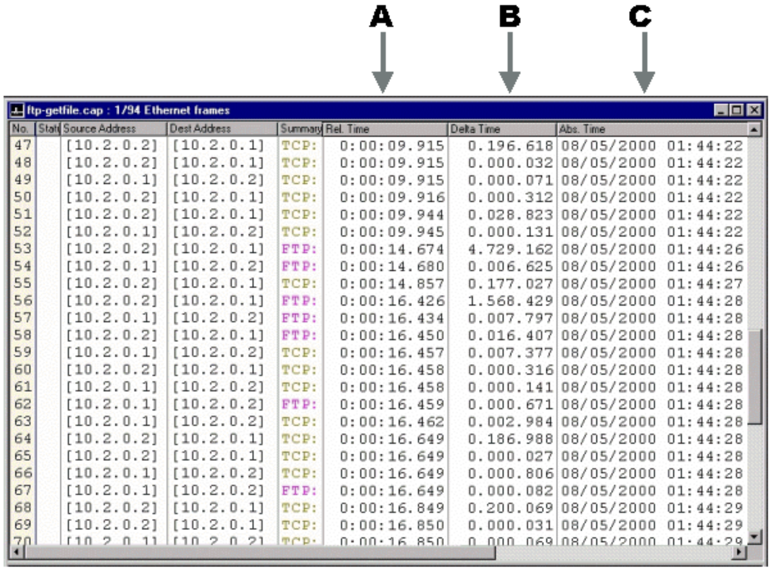
LINK

*Start to look at the typical patterns seen on your network. For additional traces to review, check out **www.packet-level.com**.*

Now let’s look a bit more closely at those summary screens and the packet timestamping mechanisms.

Timestamping

Analyzers timestamp each packet captured. They typically include the timestamp information on the decode screen, as shown in Figure 1-24.



No.	Stat	Source Address	Dest Address	Summary	Rel. Time	Delta Time	Abs. Time
47		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:09.915	0.196.618	08/05/2000 01:44:22
48		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:09.915	0.000.032	08/05/2000 01:44:22
49		[10.2.0.1]	[10.2.0.2]	TCP:	0:00:09.915	0.000.071	08/05/2000 01:44:22
50		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:09.916	0.000.312	08/05/2000 01:44:22
51		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:09.944	0.028.823	08/05/2000 01:44:22
52		[10.2.0.1]	[10.2.0.2]	TCP:	0:00:09.945	0.000.131	08/05/2000 01:44:22
53		[10.2.0.2]	[10.2.0.1]	FTP:	0:00:14.674	4.729.162	08/05/2000 01:44:26
54		[10.2.0.1]	[10.2.0.2]	FTP:	0:00:14.680	0.006.625	08/05/2000 01:44:26
55		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:14.857	0.177.027	08/05/2000 01:44:27
56		[10.2.0.2]	[10.2.0.1]	FTP:	0:00:16.426	1.568.429	08/05/2000 01:44:28
57		[10.2.0.1]	[10.2.0.2]	FTP:	0:00:16.434	0.007.797	08/05/2000 01:44:28
58		[10.2.0.2]	[10.2.0.1]	FTP:	0:00:16.450	0.016.407	08/05/2000 01:44:28
59		[10.2.0.1]	[10.2.0.2]	TCP:	0:00:16.457	0.007.377	08/05/2000 01:44:28
60		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:16.458	0.000.316	08/05/2000 01:44:28
61		[10.2.0.1]	[10.2.0.2]	TCP:	0:00:16.458	0.000.141	08/05/2000 01:44:28
62		[10.2.0.1]	[10.2.0.2]	FTP:	0:00:16.459	0.000.671	08/05/2000 01:44:28
63		[10.2.0.1]	[10.2.0.2]	TCP:	0:00:16.462	0.002.984	08/05/2000 01:44:28
64		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:16.649	0.186.988	08/05/2000 01:44:28
65		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:16.649	0.000.027	08/05/2000 01:44:28
66		[10.2.0.1]	[10.2.0.2]	TCP:	0:00:16.649	0.000.806	08/05/2000 01:44:28
67		[10.2.0.1]	[10.2.0.2]	FTP:	0:00:16.649	0.000.082	08/05/2000 01:44:28
68		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:16.849	0.200.069	08/05/2000 01:44:29
69		[10.2.0.2]	[10.2.0.1]	TCP:	0:00:16.850	0.000.031	08/05/2000 01:44:29
70		[10.2.0.1]	[10.2.0.2]	TCP:	0:00:16.850	0.000.069	08/05/2000 01:44:29

FIGURE 1-24. **Most analyzers support three basic timestamps.**

There are three basic timestamps used in analysis:

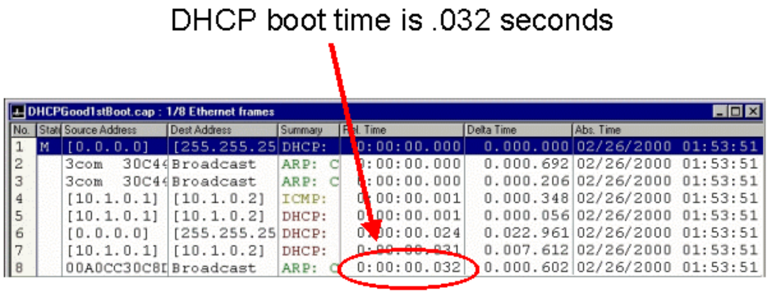
- Relative (“A” in Figure 1-26)
- Delta (aka “Interpacket”) (“B” in Figure 1-26)
- Absolute (“C” in Figure 1-26)

All three are useful for a variety of purposes. In the next several pages, I will explain each timestamp and give you examples on how to use each one. Finally, I’ll give you an analysis task that focuses on timestamping something on your network.

Relative Timestamps

The relative timestamp indicates the amount of time between the first packet received in the trace buffer and the current packet. This type of timestamp is great when you are timing an entire process.

For example, consider Figure 1-25. We booted up a PC and watched it get it's IP address using DHCP. By using the relative timestamps in the trace buffer, we can easily determine the length of time required for this process.



DHCP boot time is .032 seconds

DHCPGood1stBoot.cap : 1/8 Ethernet frames

No.	Stat	Source Address	Dest Address	Summary	Rel. Time	Delta Time	Abs. Time
1	M	[0.0.0.0]	[255.255.25	DHCP:	0:00:00.000	0.000.000	02/26/2000 01:53:51
2		3com 30C44	Broadcast	ARP: C	0:00:00.000	0.000.692	02/26/2000 01:53:51
3		3com 30C44	Broadcast	ARP: C	0:00:00.000	0.000.206	02/26/2000 01:53:51
4		[10.1.0.1]	[10.1.0.2]	ICMP:	0:00:00.001	0.000.348	02/26/2000 01:53:51
5		[10.1.0.1]	[10.1.0.2]	DHCP:	0:00:00.001	0.000.056	02/26/2000 01:53:51
6		[0.0.0.0]	[255.255.25	DHCP:	0:00:00.024	0.022.961	02/26/2000 01:53:51
7		[10.1.0.1]	[10.1.0.2]	DHCP:	0:00:00.031	0.007.612	02/26/2000 01:53:51
8		00A0CC30C8D	Broadcast	ARP: C	0:00:00.032	0.000.602	02/26/2000 01:53:51

FIGURE 1-25. **By capturing and documenting the boot process, we can compare later boot performance to identify errors.**

Delta (Interpacket) Timestamps

Interpacket timestamps indicate the time between packets that have arrived in the trace buffer. This type of timestamp is particularly useful for determining the latency between requests and replies. For example, Figure 1-26 shows a latency test used to compare an application's performance on the local network compared to the performance across the WAN.

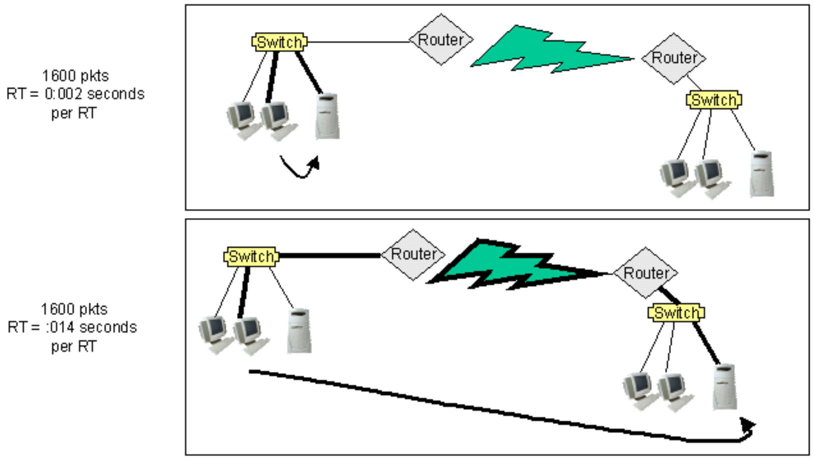


FIGURE 1-26. **Look at the delta time between requests and replies and multiply that by the number of request/reply sets to figure out how long it may take to load an application or perform a task.**

Absolute Timestamps

Absolute timestamps indicate the time the packet arrived based on the clock of the analyzer system. This type of timestamp is useful when you know the approximate time that an interesting event occurred -- just scroll down through the trace to see that time of day. The following lists some predictable network events:

- IPX RIP’s frequency should be about every minute; IP RIP about every 30 seconds.
- NLSP does a consistency resynchronize every 2 hours while OSPF resynchronizes every 30 minutes.
- The Ring Poll process occurs every seven seconds on a Token Ring network.

HANDS-ON

Analyze It! Ok. now it’s time to check your own timestamping. Take a look at your broadcast traffic -- find a typical route broadcast (IPX RIP, IP RIP, OSPF, NLSP, whatever). Now -- use the timestamp information to determine the frequency of those broadcasts. Be certain you are looking at the packets the protocol uses to inform other routers about known routes or route entries. What is the frequency? You will use timestamping extensively in Chapter 3, “Application Analysis.”

Chapter Quiz

Spend a few moments reviewing this quiz. Review the chapter as needed. The answers are contained in Appendix A, “Answers to Chapter Quizzes.”

Question 1-1: Why would an analyzer drop packets?

Question 1-2: Your broadcast alarm has been triggered again and again over the night. Since it’s sending messages to Joe, you really aren’t that put out, however, Joe is certainly a cranky little brat the following morning. What can you do to reduce the broadcast alarms?

Question 1-3: What devices need to process packets addressed to 0xFF-FF-FF-FF-FF-FF? What devices need to process packets sent to a multicast address?

Question 1-4: What is the difference between the delta (interpacket) timestamp and the relative timestamp?

Question 1-5: When is the communication pattern request-reply-request-reply acceptable? Give an example of a communication that might use this pattern.

Question 1-6: What type of communication pattern does IPX burst mode use?

Question 1-7: What type of communication pattern does IP RIP use?

Question 1-8: What type of communication pattern is often seen from database queries?

Question 1-9: What is a false positive? How can you reduce false positives?

Question 1-10: What is a false negative? How can you reduce false negatives?

CHAPTER 2

Capture and Display Filtering

In this chapter, we'll examine the most exciting part of capturing packets -- the filtering process. Some solid understanding of packet structures, protocol specifications and communication procedures will come in handy in this chapter.

NOTE

Network analysis references and resources are listed in Appendix C.

I really feel like a kid in a candy store when I start working with filters. There are so many possibilities! Why.... we can catch all sorts of traffic from the good guys (management) and the bad guys (hackers... or maybe management)... <grin>. Imagine being able to set up a filter to catch anyone who might be running an FTP service regardless of the port that they established the service on. Wheeee!

I have no idea who this guy is, but he sure looks happy, doesn't he?

Filtering Overview

Filters are used to reduce the number of packets to a manageable number that focus on the area you are most interested in.

Figure 2-1 shows the basic flow of filtered data.

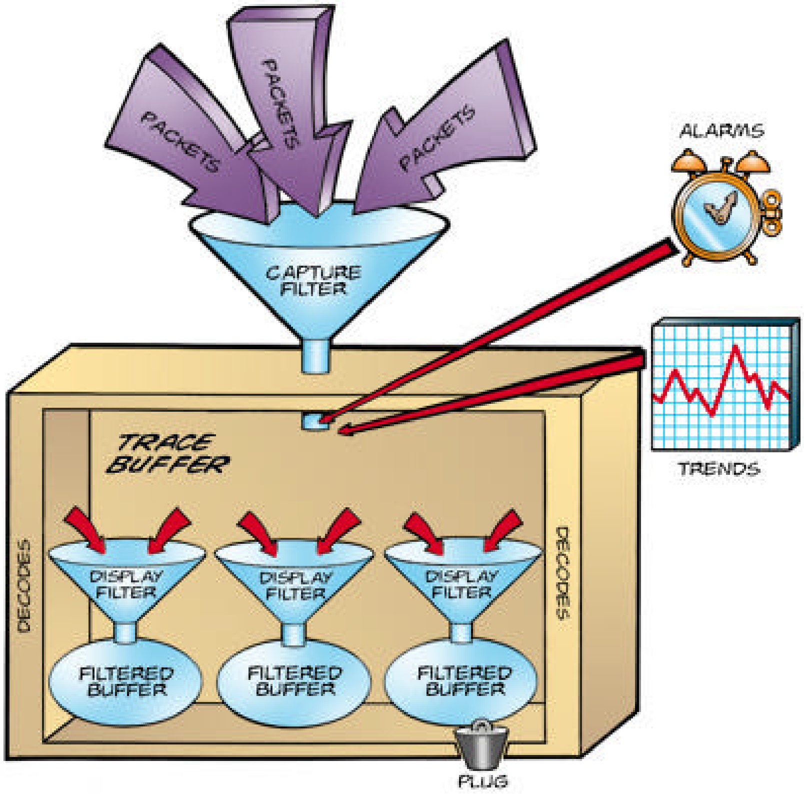


FIGURE 2-1. **Protocol analyzer architecture.**

There are two basic types of filters used in analysis:

- Capture filters (aka ‘pre-filters’)
- Display filters (aka ‘post-filters’)

Capture Filters

Capture Filters (also referred to as pre-filters) are used to reduce the number of packets that are saved into the trace buffer. For example, you could set a capture filter to gather all broadcast traffic.

To capture your broadcast traffic, you would build a filter that looks for all packets addressed to MAC address 0xFF-FF-FF-FF-FF-FF (the broadcast MAC address). If you are interested in IP broadcasts, you could also set up a filter for packets addressed to 255.255.255.255 (or 0xFF.FF.FF.FF), the IP broadcast address. You may also be interested in looking for subnet broadcasts, such as 130.57.255.255, as well.

NOTE

Since capture filters are applied at the time that packets are arriving at the analyzer, they do require some CPU cycles. If the analyzer is already dropping packets because of a high network load, it may drop even more packets when you apply a capture filter.

Display Filters

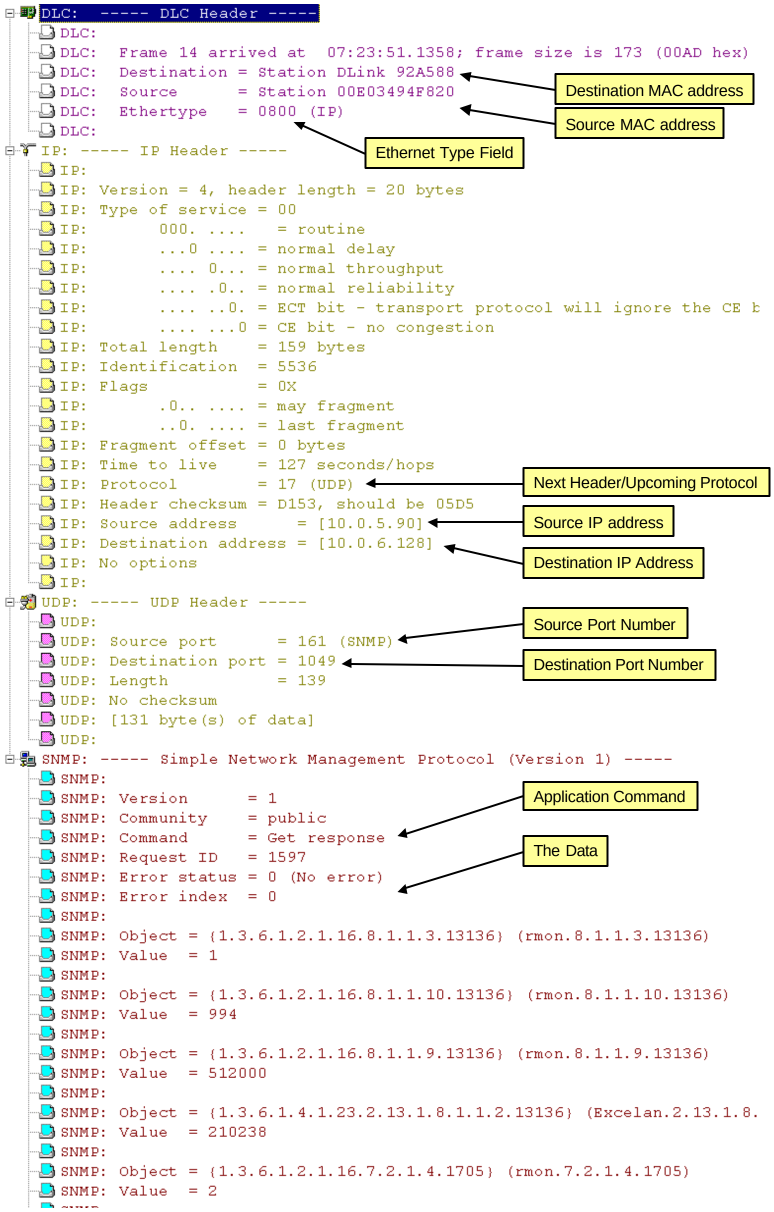
Display Filters (also referred to as post-filters) are applied to a trace buffer after packets are already captured. For example, if you set up a capture filter for all broadcast traffic you can further define that to look for only DHCP broadcast traffic.

These capture and display filters are used together to pinpoint the communications you are most interested in.

There are hundreds of possible ways to filter on packets -- we will summarize these methods into the four basic filter types:

- address filters
- protocol filters
- data pattern matching filters
- complex boolean filter sets

On the next page, you'll see a graphic of a packet showing some of the locations that you may want to apply a filter to.



```
DLC:  ----- DLC Header -----
  DLC:
  DLC:  Frame 14 arrived at  07:23:51.1358; frame size is 173 (00AD hex)
  DLC:  Destination = Station DLink 92A588
  DLC:  Source      = Station 00E03494F820
  DLC:  Ethertype   = 0800 (IP)
  DLC:
IP: ----- IP Header -----
  IP:
  IP: Version = 4, header length = 20 bytes
  IP: Type of service = 00
  IP:       000. ....   = routine
  IP:       ...0 .... = normal delay
  IP:       .... 0... = normal throughput
  IP:       .... .0.. = normal reliability
  IP:       .... ..0. = ECT bit - transport protocol will ignore the CE b
  IP:       .... ...0 = CE bit - no congestion
  IP: Total length    = 159 bytes
  IP: Identification  = 5536
  IP: Flags           = 0X
  IP:       .0.. .... = may fragment
  IP:       ..0. .... = last fragment
  IP: Fragment offset = 0 bytes
  IP: Time to live    = 127 seconds/hops
  IP: Protocol        = 17 (UDP)
  IP: Header checksum = D153, should be 05D5
  IP: Source address      = [10.0.5.90]
  IP: Destination address = [10.0.6.128]
  IP: No options
  IP:
UDP: ----- UDP Header -----
  UDP:
  UDP: Source port      = 161 (SNMP)
  UDP: Destination port = 1049
  UDP: Length           = 139
  UDP: No checksum
  UDP: [131 byte(s) of data]
  UDP:
SNMP: ----- Simple Network Management Protocol (Version 1) -----
  SNMP:
  SNMP: Version      = 1
  SNMP: Community    = public
  SNMP: Command      = Get response
  SNMP: Request ID   = 1597
  SNMP: Error status = 0 (No error)
  SNMP: Error index  = 0
  SNMP:
  SNMP: Object = {1.3.6.1.2.1.16.8.1.1.3.13136} (rmon.8.1.1.3.13136)
  SNMP: Value  = 1
  SNMP:
  SNMP: Object = {1.3.6.1.2.1.16.8.1.1.10.13136} (rmon.8.1.1.10.13136)
  SNMP: Value  = 994
  SNMP:
  SNMP: Object = {1.3.6.1.2.1.16.8.1.1.9.13136} (rmon.8.1.1.9.13136)
  SNMP: Value  = 512000
  SNMP:
  SNMP: Object = {1.3.6.1.4.1.23.2.13.1.8.1.1.2.13136} (Excelan.2.13.1.8.
  SNMP: Value  = 210238
  SNMP:
  SNMP: Object = {1.3.6.1.2.1.16.7.2.1.4.1705} (rmon.7.2.1.4.1705)
  SNMP: Value  = 2
```

FIGURE 2-2. **Just some of the fields that you may want to apply a filter to.**

Address Filters

This is the basic filter type that you should be able to do quickly and efficiently. Address filters can typically be placed on the MAC-layer address or the network layer address. For example, you could define a filter for all packets addressed to the broadcast MAC address (0xFF-FF-FF-FF-FF-FF) or the network subnet broadcast address (10.255.255.255, for example).

You can use address filters to capture traffic to and from any device on the network. Be careful of the MAC address filtering process -- that process is only used to capture local traffic pairs (i.e., to and from the router and local systems). If you want to capture traffic pairs to and from devices on separate networks, build your address filters based on their layer 3 (network layer) addresses.

Sample Address Filtering Process

Figure 2-3 shows a sample network to illustrate this point. As you can see, there are two networks, 10.1.0.0 and 10.2.0.0. If our analyzer is on network 10.1.0.0, we can build and use the following address filters.

- `[NIC A] <--> [NIC B]` captures all traffic to and from the router and the client regardless of the protocol in use. This filter won't capture the broadcast traffic, however, so it might look a bit strange if you only see a reply from the router without a corresponding request from the client. For example, consider what you'd capture during a DHCP bootup process where the router is the relay agent or if the client is a NetWare client that SAPs and RIPs during startup.
- `[10.1.0.99] <--> [10.2.4.99]` captures all IP traffic to and from the server and the client. Of course, again, we wouldn't see any broadcast traffic. We wouldn't capture any IPX traffic either.
- `[10.1.0.22] <--> [any]` captures all IP traffic to and from the router. This would capture any broadcasts the router sends as well.

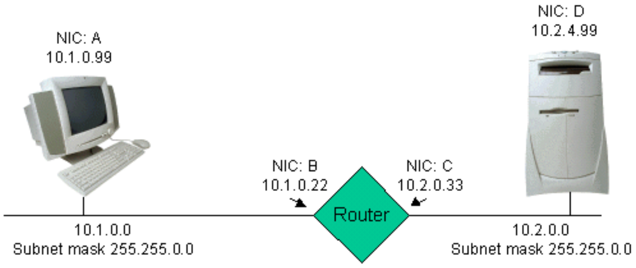
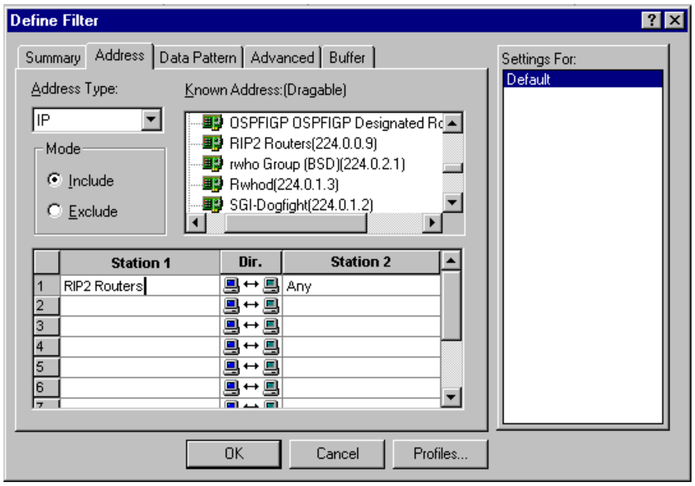


FIGURE 2-3. **Our little sample network -- true... it's an almost ideal network -- one client, one server and not a user in sight! <grin>.**

What type of address filter would you build if you wanted to capture all traffic on network 10.1.0.0 regardless of protocol? 10... 9... 8... 7... 6... 5... 4... 3... 2... 1...Time's up. <any> <--> <any> would do it. This is the default capture filter, by the way.

Let's see what it looks like to set up one of these address filters on the Sniffer. Figure 2-4 shows the address filter window on the Sniffer. In this example, we have set up a filter for all traffic sent to the RIP2 multicast address.



Address Type: IP

Known Address:(Dragable)

- OSPFIGP OSPFIGP Designated Rc
- RIP2 Routers(224.0.0.9)
- rwho Group (BSD)(224.0.2.1)
- Rwhod(224.0.1.3)
- SGI-Dogfight(224.0.1.2)

Mode

● Include

○ Exclude

	Station 1	Dir.	Station 2
1	RIP2 Routers	↔	Any
2		↔	
3		↔	
4		↔	
5		↔	
6		↔	

Settings For:

Default

FIGURE 2-4. **Sniffer address filter window.**

In this case, the process was very simple. I clicked and dragged the address from the Sniffer's address book.

In fact, on the Sniffer, you can choose your address filter type:

- MAC (data link address)
- IP
- IPX (how thoughtful to include a 'dead and dying protocol')

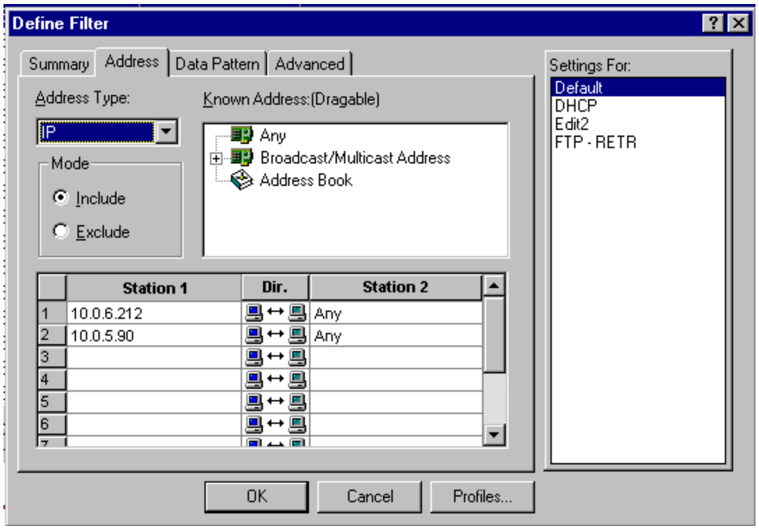
HANDS-ON

Analyze it! Set up a filter for all the broadcast traffic on your network. Let the analyzer run overnight and see what type of packets you find.

Complex Address Filter Techniques

You can build complex address filter groups to look at a set of devices communicating on the network. For example, perhaps you want to look at all traffic to and from the primary and secondary HTTP servers that use IP addresses (10.0.6.212 and 10.0.5.90).

You need to put in more than one address and you need to leave one end of the communications open `[any]`, as shown in Figure 2-5.



Address Type: IP

Mode

● Include

○ Exclude

Known Address:(Dragable)

- Any
- Broadcast/Multicast Address
- Address Book

	Station 1	Dir.	Station 2
1	10.0.6.212	↔	Any
2	10.0.5.90	↔	Any
3		↔	
4		↔	
5		↔	
6		↔	
7		↔	

Settings For:

- Default
- DHCP
- Edit2
- FTP - RETR

FIGURE 2-5. **The Sniffer will logically OR any addresses in the filter definition enabling you to capture traffic from a number of devices with a single filter.**

You can become very creative with the address filters!

NOTE

I highly recommend that you build a set of address filters based on your critical devices and personnel. When the boss states that he/she cannot connect to the server, you don't want to spend time looking up his/her address. -- Laura

Subnet Address Filters

Unfortunately, you can't use this address filtering technique if you want to build a filter for traffic to/from anyone on a specific network (such as to/from any device on network 10.2.0.0). This filter only works for specific complete addresses. Later in this chapter, we'll examine how to build a 'subnet address' filter using data pattern filtering.

Protocol Filters

Protocol filters are based on some unique characteristic or identifier within the packet. For example, the IP protocol filter is based on the value 0x0800 in the Ethernet type field of an Ethernet II or Ethernet SNAP packet. A DNS filter would be based on the source/destination port values of 53 in the UDP or TCP header.

Although I've pointed out some special fields here, you can actually build filters on the value of any field in a packet.

The following pages provide details on the unique identifiers used in each protocol filter.

LINK

For additional filters not shown in this list, refer to the assigned number information at **www.iana.org**.

Decent analyzers should have most of these filters pre-defined for you -- if you are missing one or two, make them yourself using the following list and the steps for building data pattern filters later in this chapter.

TCP/IP Protocol Filters

The following is the basic list of filters you should have available for analyzing TCP/IP networks.

IP
TCP
UDP
ARP
ICMP
DNS
DHCP/BOOTP - Client
DHCP/BOOTP - Server
SLP
NTP
POP
IMAP
FTP
HTTP

HTTPS
Telnet
SNMP General
SNMP Traps
IP RIP (version 1 or 2)
OSPF

IPX Protocol Filters and Definitions

The following is the basic list of filters you should have available for analyzing IPX networks.

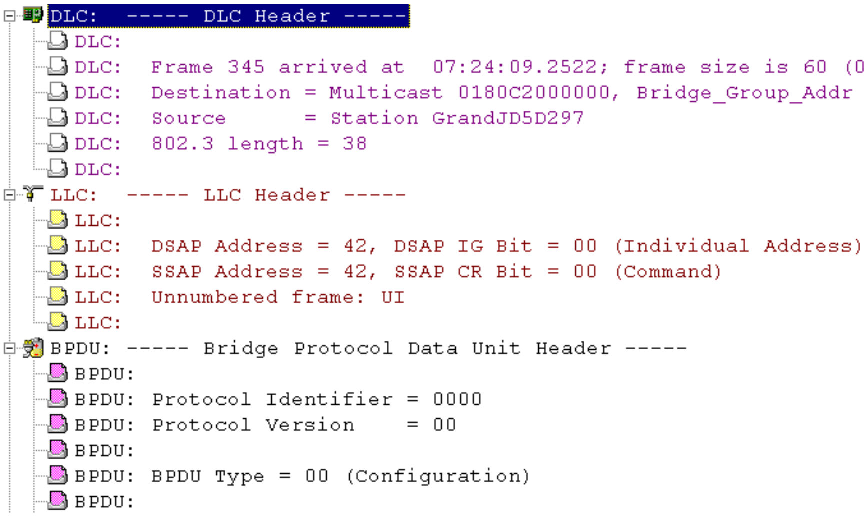
IPX
SPX (version 1)
SPX (version 2)
NCP requests
NCP replies
Request Being Processed
Connection Request
Connection Destroyed
IPX RIP
SAP
NLSP

Miscellaneous Protocol Filters and Definitions

These two protocols can be considered ‘miscellaneous.’ You should have filters available for analyzing this traffic as well.

CDP (Cisco Discovery Protocol)
BPDU (Bridge Protocol Data Unit)

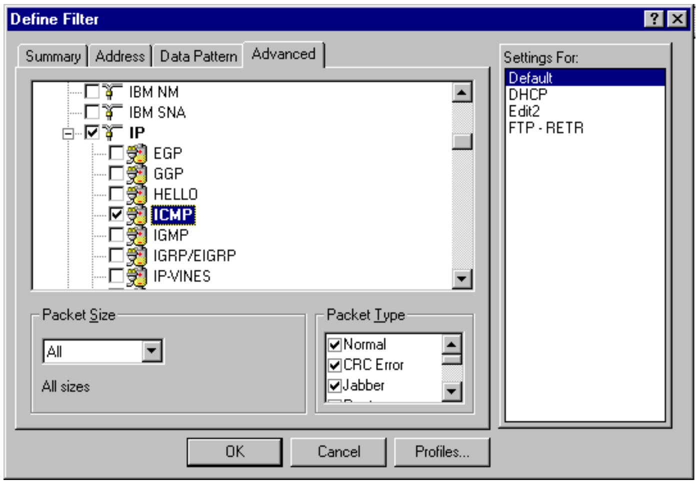
Figure 2-6 shows a BPDU packet. You can see why this is considered an unusual packet on the network.



```
DLC:  ----- DLC Header -----
   DLC:
   DLC:  Frame 345 arrived at  07:24:09.2522; frame size is 60 (0
   DLC:  Destination = Multicast 0180C2000000, Bridge_Group_Addr
   DLC:  Source      = Station GrandJD5D297
   DLC:  802.3 length = 38
   DLC:
LLC:  ----- LLC Header -----
   LLC:
   LLC:  DSAP Address = 42, DSAP IG Bit = 00 (Individual Address)
   LLC:  SSAP Address = 42, SSAP CR Bit = 00 (Command)
   LLC:  Unnumbered frame: UI
   LLC:
BPDU: ----- Bridge Protocol Data Unit Header -----
   BPDU:
   BPDU: Protocol Identifier = 0000
   BPDU: Protocol Version    = 00
   BPDU:
   BPDU: BPDU Type = 00 (Configuration)
   BPDU:
```

FIGURE 2-6. **Here is part of a BPDU packet -- do you see why it wouldn't fit the IP or IPX protocol filter? The BPDU header/data resides directly after the Ethernet header.**

Most analyzers have a very simple way of selecting the protocol you want to filter on. In the case of the Sniffer, it's as simple as selecting the protocol off a list, as shown in Figure 2-7.



☐ IBM NM
☐ IBM SNA
☑ IP
☐ EGP
☐ GGP
☐ HELLO
☑ ICMP
☐ IGMP
☐ IGRP/EIGRP
☐ IP-VINES

Packet Size: All

All sizes

Packet Type:
☑ Normal
☑ CRC Error
☑ Jabber

Settings For:
Default
DHCP
Edit2
FTP - RETR

FIGURE 2-7. **Analyzers typically have a nice list of protocol filters prebuilt -- just select the protocol you are interested in.**

The filter shown in Figure 2-7 is a ‘must have’ for any network analyst... you should **always** be able to filter out all your ICMP traffic at a moment’s notice.

NOTE

If you read the “TCP/IP Analysis and Troubleshooting” book (also from podbooks.com), then you should already know that EVERY troubleshooter needs to understand ICMP inside and out. Get RFC 792 immediately -- read it, eat it, breath it... it can save your butt. --Laura

Ok... so the protocol filters are pretty easy, eh? Time to do some serious data pattern filtering!

Data Pattern Filters (Advanced Filters)

Here’s where we separate the wheat from the chaff. Now you need a solid understanding of the packet structures, possible field values, and hexadecimal/decimal numbering scheme.

Data pattern filters are used when you want to match a specific value at a specific location -- and there isn’t a predefined filter to be found anywhere! For example, perhaps you want to capture all IP packets that have the ‘don’t fragment’ bit set to 1, or you want to gather all packets that have the ASCII character set ‘PONG’ in the data portion (used in the Trinoo Distributed Denial of Service attack).

LINK

If you want to see some packet-level breakdowns of various DDoS attacks, check out Dave Dittrich’s web site at **www.washington.edu/People/dad/.**

Consider the following packet structures for UDP/IP, TCP/IP and IPX communications (shown with the field offsets). We will use these packets to build a series of data pattern filters.

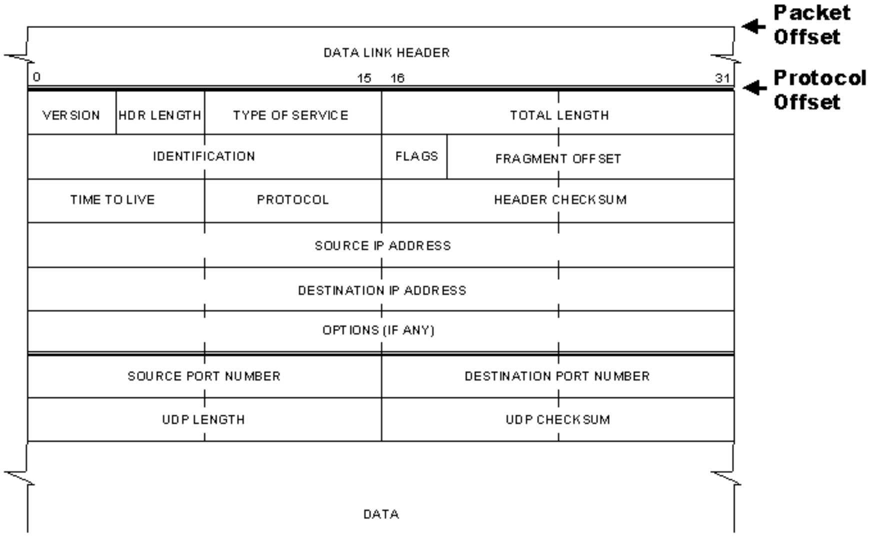


FIGURE 2-8. **UDP/IP packet structure and offsets.**

Be careful here. The offsets shown in Figure 2-8 through 2-10 are in decimal format. Some analyzers define the offsets in hexadecimal (0x) format. For example, if you want to filter packets based on the Protocol field in the IP header, you'll need to convert the IP header offset value to hexadecimal.

There are also two offset types typically seen.

- Packet-level offsets: From the start of the data link header.
- Protocol-level offsets: From the start of the protocol that follows the data link header.

The preferable offset to use is protocol-level offset. This offset allows the analyzer to calculate the actual offset if the data link header length varies. In other words, if your network supports Ethernet II and SNAP header structures, the location of the Protocol field will be different in each of these packets.

Using packet-level offsets on an Ethernet II packet, the Protocol field is located at offset 17 hex, or 23 decimal. On this same packet, the protocol-level offset for the same field is 9 hex (also 9 in decimal). Look back at Figure 2-8 and count the bytes (each byte is 8 bits long -- one row across is 4 bytes).

NOTE

You can use the Windows calculator in scientific mode to do hex-to-decimal conversions - better yet, get Hex Workshop (see Chapter 5). -- Laura

Ok... back to the packet fields for IP and IPX.

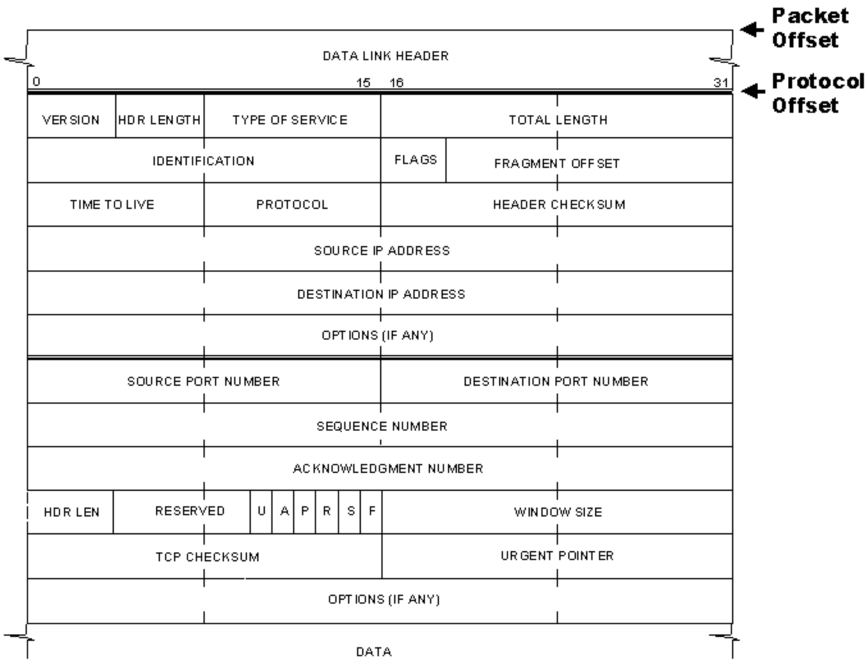


FIGURE 2-9. **TCP/IP packet structure and offsets.**

So... let’s do a quick test. What is the hex offset you’d use to identify a specific destination port number in a packet (assuming a protocol-level filter and no IP header options)?

The answer is 16 hex or 22 decimal. It’s really nice to have the analyzer figure out these offsets for you too. Later, you’ll see how you can simply set data based on an existing packet structure.

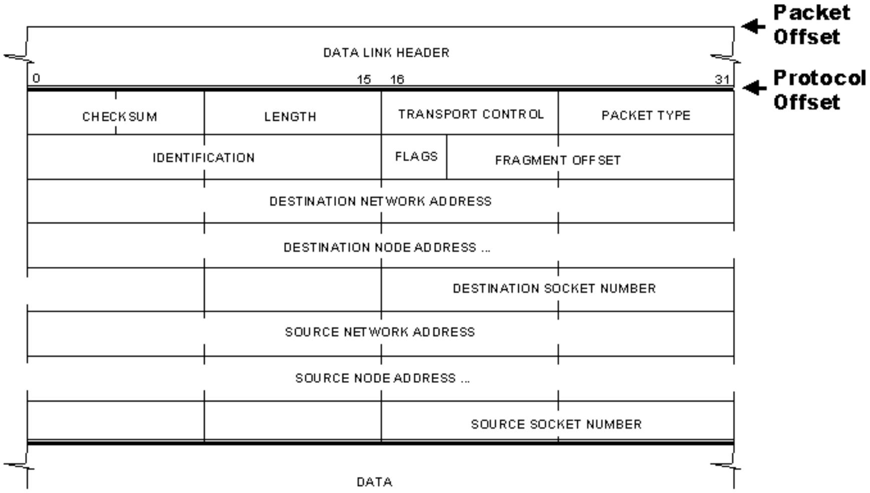
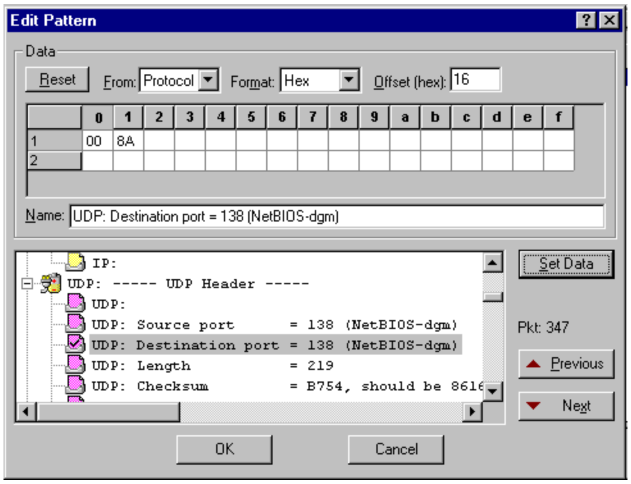


FIGURE 2-10. **IPX packet structure and offsets.**

The process of building data pattern filters will be different on each analyzer you are using, so pay close attention to how they represent the offset value -- hex or decimal. That’s usually where people get tripped up.

Figure 2-11 shows what the data pattern filter window looks like on the Sniffer. In this example, I've built a filter for stinky NetBIOS traffic (destination port number 138.



From: Protocol
Format: Hex
Offset (hex): 16

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	00	8A														
2																

Name: UDP: Destination port = 138 (NetBIOS-dgm)

```
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port      = 138 (NetBIOS-dgm)
UDP: Destination port = 138 (NetBIOS-dgm)
UDP: Length           = 219
UDP: Checksum         = B754, should be 8616
```

Pkt: 347

FIGURE 2-11. **In Sniffer, you don't have to know the offsets because you can 'set data' from a previously captured packet.**

In this next section we'll use a 5-step data pattern filtering process to catch the following traffic:

- any FTP traffic using the RETR command (to get files) to find out if there's a rogue FTP server using a non-standard port number
- any traffic that contains the value PONG to catch a specific Denial of Service attack packet if it is on your network
- any traffic to/from a specific subnet address (not possible with a simple address filter)
- any fragmented data on the network to determine if fragmentation is necessary or if there may be a security breach

The 5-Step Data Pattern Filtering Process

The process uses 5 basic steps:

Step 1: Determine what you are interested in.
Step 2: Find out the field value.
Step 3: Find the offset value.
Step 4: Find a similar packet structure/copy up the field of interest.
Step 5: Input the value you want to filter on.

We'll use the steps listed above to set up a data pattern filter for all FTP RETR commands crossing the wire. This will alert us to anyone attempting to transfer files even if they are not using the standard FTP command port number (21).

Step 1: Determine what you are interested in.

Ok... We'll look at the RFC for FTP communications to find out what command is used to get files. Go to www.ietf.org to get RFC 959.

As you'll see in section 4.1.3. (FTP Service Commands) on page 30, the command RETR is used to get files. Although port 21 is typically used to issue these commands, we'll have to assume would-be thieves will use non-standard port numbers in an attempt to thwart firewall filters.

Step 2: Find out the field value.

We know the value we are interested in is RETR in ASCII.

NOTE

I really hope your analyzer can accept ASCII input for filters -- there are so many commands that cross the line in this format and translating from ASCII to hex is a really pain). If you need to figure out ASCII-to-hex translations, use Hex Workshop, one of the tools listed in Chapter 5, "Master Analyst's Toolbox." -- Laura

Step 3: Find the offset value.

The offset we are interested in is 28 (hex) or 40 (decimal) from the protocol layer (after the MAC header). How did we know this? Easy... just check out an FTP communication and count the bytes from the start of the IP header to the field you are interested in. In this case, the RETR command follows the TCP header (20 bytes long) and the IP header (20 bytes long) making the offset 40 bytes.

If you don’t want to figure out the offset yourself, go on to step 4.

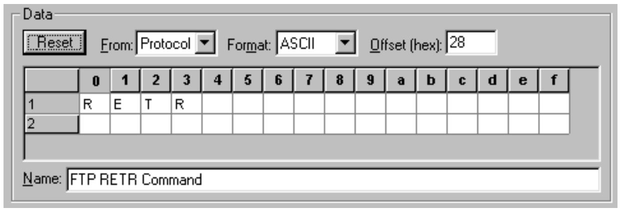
Step 4: Find a similar packet structure/copy up the field of interest.

If you don’t want to figure out the offset, you can set the data from an existing packet that contains the same field values. This will be shown in the step-by-step examples later in this chapter.

Step 5: Input the value you want to filter on.

Place the value you are interested in (in the proper format) into the filter definition. If you just used the ‘cut and paste’ process, don’t worry about this step -- you should have pasted the desired value in the filter.

Voila! You’re set! Run the filter on your network traffic and see what you get! Figure 2-12 shows the filter setup screen for an RETR filter.



From: Protocol
Format: ASCII
Offset (hex): 28

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	R	E	T	R												
2																

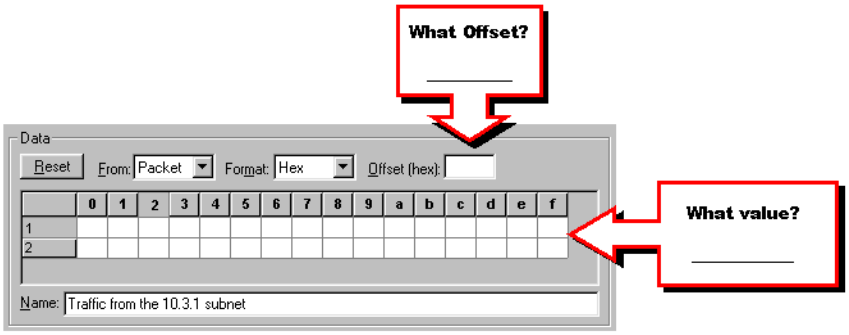
Name: FTP RETR Command

FIGURE 2-12. **A filter to catch all FTP RETR commands issued.**

Now you try one... this is based on an actual email sent to me:

> "I am having a problem in getting the following question answered about Sniffer Pro v3.50.02. I thought maybe you could help. Is there a way to set up a capture filter that would capture only traffic coming off a specific segment? For example, I would like to capture all traffic coming from IP address 10.3.1.0 (subnet mask 255.255.255.0). Is there are wildcard that can be used for the fourth octet (i.e., 10.3.1.*)? There is only space for a full IP address in the address filter. -- Bob"

Seriously, take a moment and write in your filter setting on this one...write directly in the book.... c'mon now!



What Offset? ________

What value? ________

From: Packet

Format: Hex

Offset (hex):

Name: Traffic from the 10.3.1 subnet

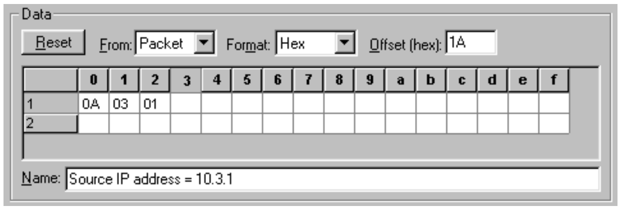
FIGURE 2-13. **Fill out a filter for all traffic from the 10.3.1 segment.**

Here are some hints:

- Hint #1: Check the IP header structure shown in Figure 2-9.
- Hint #2: Translate the desired IP address to hex.
- Hint #3: You're not filtering on four bytes, are you?

Take a moment and fill out your answer ***before you turn the page***.

Here's the answer. Note that the first filter is based on the *packet offset*. The second filter is based on the *protocol offset*.

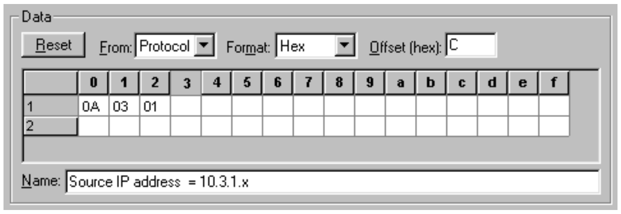


From: Packet Format: Hex Offset (hex): 1A

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	0A	03	01													
2																

Name: Source IP address = 10.3.1

or



From: Protocol Format: Hex Offset (hex): C

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	0A	03	01													
2																

Name: Source IP address = 10.3.1.x

FIGURE 2-14. **Either of these filters would catch all packets that contain the source IP address of 10.3.1.**

Not too tough, eh? Consider, however, that this filter would only capture packets ***from*** network 10.3.1 -- what about traffic ***to*** network 10.3.1? What if you want to set up a filter that captures only FTP traffic from network 10.3.1? Aaaaahhhhh... this is where it gets interesting! In the next section we'll cover the boolean pattern filtering used to combine various packet characteristics into a single filter.

Let's make one more filter -- this time we'll look for anyone trying to establish a TCP connection on the network. This next filter is interesting because we'll be looking at the value of a single bit within a packet.

Filtering on a Single Bit Value

Connection-oriented services, such as FTP and HTTP, require an initial TCP handshake to establish the connection and exchange a starting sequence number. This sequence number increases in accordance with the amount of data received, thereby offering reliable, guaranteed service for TCP data.

In the first packet of the TCP handshake, the SYN (SYNchronize sequence number) bit in the TCP header is set to 1.

Why would you care about this packet? Well… let's say you've configured a very secure network. You decided that no one should be able to connect to your inside computers from the Internet. That means no SYN packets should make it through the firewall, right? You might want to check that out by building a filter on all SYN traffic crossing the wire on the inside of the firewall.

The bit sequence in TCP headers is structured as shown in Figure 2-15.

r	r	U	A	P	R	S	F	=
0	0	0	0	0	0	1	0	

FIGURE 2-15. **The TCP flag set.**

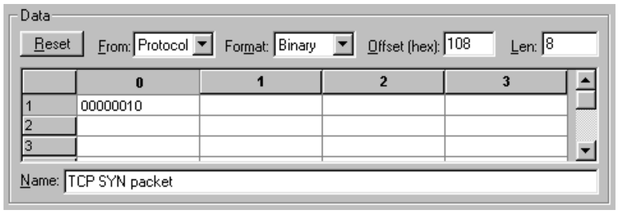
where,

r = reserved bits - set to 0
U = Urgent bit (value 0x20)
A = Acknowledgment bit (value 0x10)
P = Push bit (value 0x08)
R = Reset bit (value 0x04)
S = Synchronize bit (value 0x02)
F = Finish bit (value 0x01)

NOTE

If someone is really trying to be sneaky, they may try to go through your firewall using a packet with both the SYN flag and another flag set. Ugh. In this case, you might have to build a more advanced filter - a boolean filter that ORs together various flag setting patterns. - Laura

Figure 2-16 shows a filter that would capture all packets that have the SYN bit set.



From: Protocol
Format: Binary
Offset (hex): 108
Len: 8

	0	1	2	3
1	00000010			
2				
3				

Name: TCP SYN packet

FIGURE 2-16. **Building a filter for TCP SYN packets.**

Yes. You really do need to know the protocols and how they operate! Consider getting “TCP/IP Illustrated” by W. Richard Stevens. The reason I like that book more than the others is (a) it is accurate and detailed, and (b) it shows packets throughout the book.

HANDS-ON

Do It! Want to try one out yourself? Ok.... can you make a filter that would catch all traffic from a network that uses the private network address 172.16.34.0 with the network mask 255.255.248.0? The answer is online at www.packet-level.com/pdfs/vlsm-filter.pdf.

Now, let’s go on to the boolean filtering -- we’ll see how to mix and match these filters to narrow down the focus of our analysis.

Complex Boolean Data Pattern Filter Techniques

Not a math major? Not familiar with the term ‘boolean’?

> ***Bool·e·an - adjective***
> Using combinatory system of symbolic logic: using or characterized by a system of symbolic logic that uses combinations of such logical operators as "AND," "OR," and "NOT" to determine relationships between entities. Boolean operations are extensively used in writing computer programs and in computer searches using keywords.
>
> [Mid-19th century. Named for the English mathematician George Boole (1815-1864), who devised the system.]

You can create some very complex filter sets by using these boolean operands.

- ***ICMP Port Unreachable Filter (AND):*** You could build a filter for all traffic that contains the ICMP type field value of 3 (Destination Unreachable) ***and*** packets with the ICMP Code value 3 (Port Unreachable). Go run and get the RFC on ICMP, RFC 792.
- ***Key FTP Operations Filter (OR):*** You could build a filter for all traffic that contains the FTP RETR or STOR or NLIST command.
- ***Subnet Bi-Directional Filter (OR):*** You could set up a filter for all traffic that comes from network 10.3.1 or traffic that goes to network 10.3.1.
- ***Fragmented Packets Filter (AND NOT):*** You could build a filter for all fragmented packets by filtering on packets with the 'more fragments' bit set to 1 and not packets with IP fragment offset of 0.

As you can see, a solid understanding of the protocols is required to really take full advantage of the advanced boolean filtering. Once you put in the time and effort, however, you will find that your analyzer has become ten times as useful as before.

Let's go through each of the boolean examples above to show you how these boolean filters are defined on the Sniffer.

AND (Catching Port Unreachables)

In this example, you want to capture a specific type of ICMP Destination Unreachable packet - you want to look for the Port Unreachable type packets that indicate someone is looking for a service that just doesn't exist at that location.

ICMP Destination Unreachable code numbers are listed below.

Codes

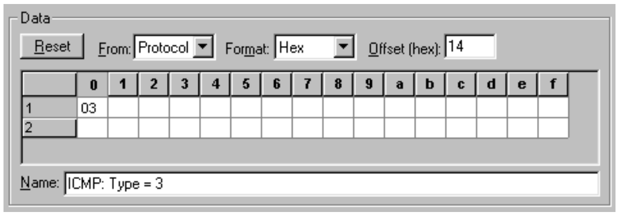
0 Net Unreachable
1 Host Unreachable
2 Protocol Unreachable
3 Port Unreachable
4 Fragmentation Needed and Don't Fragment was Set
5 Source Route Failed
6 Destination Network Unknown
7 Destination Host Unknown
8 Source Host Isolated
9 Communication with Destination Net is Administratively Prohibited
10 Communication with Destination Host is Administratively Prohibited
11 Destination Network Unreachable for Type of Service
12 Destination Host Unreachable for Type of Service
13 Communication Administratively Prohibited [see RFC1812]
14 Host Precedence Violation [see RFC1812]
15 Precedence cutoff in effect [see RFC1812]

NOTE

The ICMP code numbers list is maintained by IANA (Internet Assigned Numbers Authority). As a protocol analyst, you should consult them at www.iana.org to stay up-to-date on the latest assigned numbers for various protocols. - Laura

In order to find out what the offset value should be, we reference RFC 792 and look at page 4 -- seriously, folks -- go get the RFC and look at page 4. Go to www.ietf.org > RFC pages. Enter 792 and click **go**. You can see on that page that the type and code fields follow immediately after the IP header (typically 20 bytes long). They are both single byte fields. Ok.... so this means that their offsets are at 20 bytes (type field) and 21 bytes (code field). Translate those offsets to hex and here we go!

Figures 2-17 and 2-18 show the two patterns we are interested in.



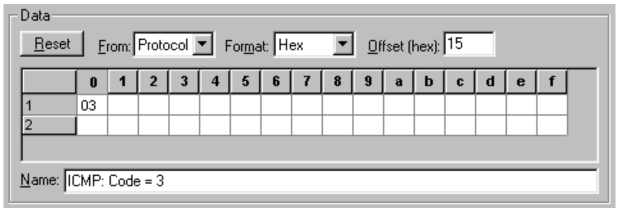
Data

Reset From: Protocol Format: Hex Offset (hex): 14

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	03															
2																

Name: ICMP: Type = 3

FIGURE 2-17. ***Pattern 1*: Packets with the ICMP type field value of 3 (Destination Unreachable)**



Data

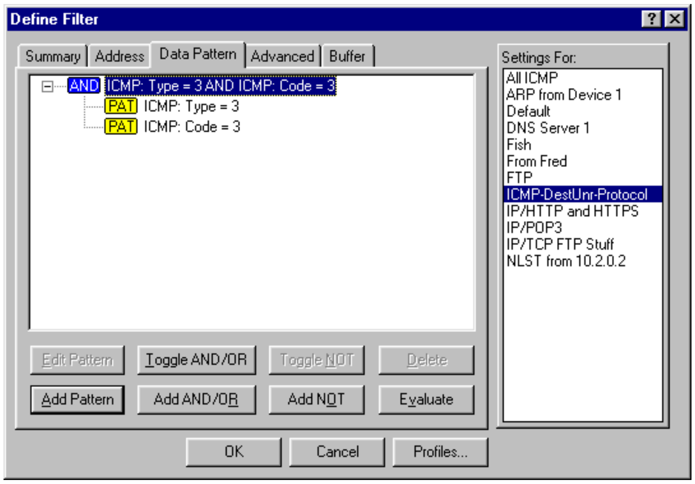
Reset From: Protocol Format: Hex Offset (hex): 15

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	03															
2																

Name: ICMP: Code = 3

FIGURE 2-18. ***Pattern 2*: Packets with the ICMP Code value of 3 (Port Unreachable)**

We want to "AND" these two operations because we are looking for packets that have both the type 3 value and the code 3 value at the correct offset. The final data pattern filter should look like the one shown in Figure 2-19.



AND ICMP: Type = 3 AND ICMP: Code = 3
PAT ICMP: Type = 3
PAT ICMP: Code = 3

All ICMP
ARP from Device 1
Default
DNS Server 1
Fish
From Fred
FTP
ICMP-DestUnr-Protocol
IP/HTTP and HTTPS
IP/POP3
IP/TCP FTP Stuff
NLST from 10.2.0.2

FIGURE 2-19. **This AND filter will catch all ICMP packet indicating misconfigured or unavailable services.**

OR (Catching Non-Standard FTP Operations)

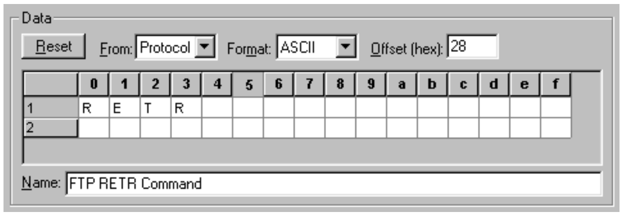
Earlier in this chapter, we used a data pattern filter to capture all FTP RETR traffic. In this example, we want to look for other FTP commands as well.

FTP uses a series of commands directly after the TCP header. The commands are:

USER: Login with this username
PASS: Use this password for login
NLST: List files on remote system
CWD: Change working directory (on remote system)
PORT: Use the following ephemeral port number
RETR: Retrieve a file
STOR: Put a file on the remote system
QUIT: Logout

What if you are interested all the traffic that is used to put files onto local systems, retrieve files from local systems or view file lists? In this case we want to build a filter that identifies RETR, STOR, and NLST packets.

Figures 2-20 through 2-22 detail the three patterns we are interested in.

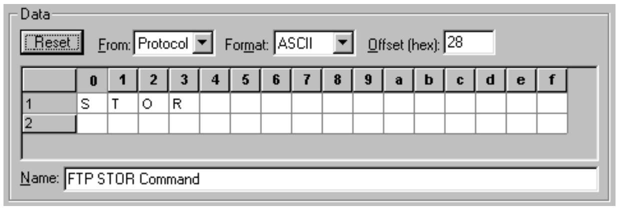


From: Protocol Format: ASCII Offset (hex): 28

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	R	E	T	R												
2																

Name: FTP RETR Command

FIGURE 2-20. **Pattern 1: Packets with the RETR pattern (get files using FTP)**

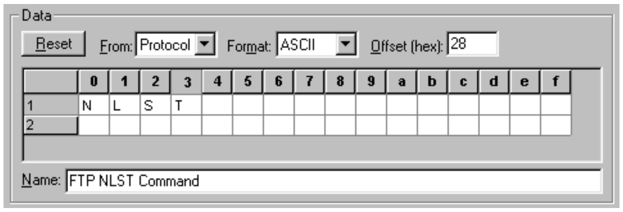


From: Protocol Format: ASCII Offset (hex): 28

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	S	T	O	R												
2																

Name: FTP STOR Command

FIGURE 2-21. **Pattern 2: Packets with the STOR pattern (put files using FTP).**



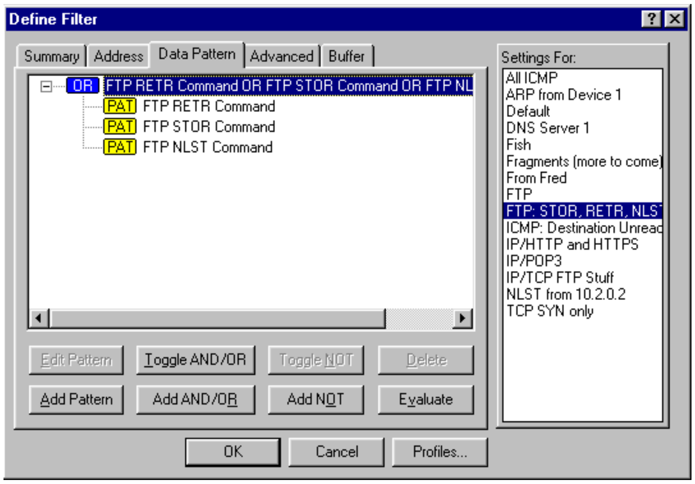
From: Protocol Format: ASCII Offset (hex): 28

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	N	L	S	T												
2																

Name: FTP NLST Command

FIGURE 2-22. **Pattern 3: Packets with the NLST pattern (list files using FTP).**

We want to "OR" these three operations because you are looking for packets that have either the value RETR, STOR and NLST commands. The final data pattern filter should look like the one shown in Figure 2-23.



OR FTP RETR Command OR FTP STOR Command OR FTP NL
PAT FTP RETR Command
PAT FTP STOR Command
PAT FTP NLST Command

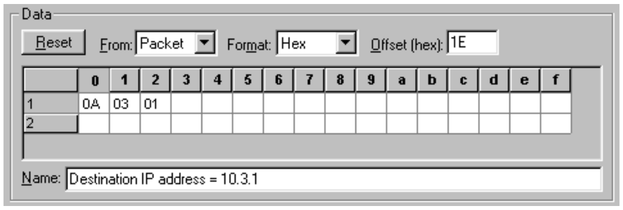
Settings For:
All ICMP
ARP from Device 1
Default
DNS Server 1
Fish
Fragments (more to come)
From Fred
FTP
FTP: STOR, RETR, NLST
ICMP: Destination Unreac
IP/HTTP and HTTPS
IP/POP3
IP/TCP FTP Stuff
NLST from 10.2.0.2
TCP SYN only

FIGURE 2-23. **The "OR" operand widens the number of possible data matches.**

OR (Catching Subnet Traffic - Bidirectionally)

Earlier in this chapter, we built a subnet filter. That filter was unidirectional -- only capturing packets coming from network 10.3.1. Now let's build a bidirectional filter so we can see all traffic to *and* from network 10.3.1.

Figures 2-24 and 2-25 detail the two patterns we are interested in.



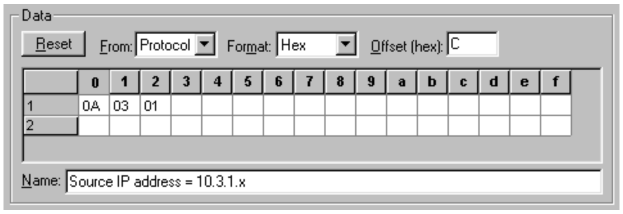
Data

Reset From: Packet Format: Hex Offset (hex): 1E

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	0A	03	01													
2																

Name: Destination IP address = 10.3.1

FIGURE 2-24. **A filter for all packets to the destination subnet 10.3.1.**



Data

Reset From: Protocol Format: Hex Offset (hex): C

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	0A	03	01													
2																

Name: Source IP address = 10.3.1.x

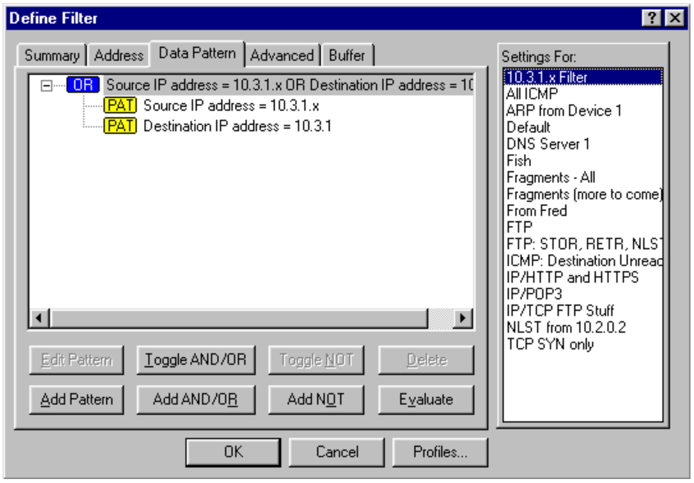
FIGURE 2-25. **A filter for all packets from the source subnet 10.3.1.**

We want to “OR” these two patterns to catch packets to or from this subnet.

NOTE

Now, let me ask you a question -- Assuming we are on a different subnet (such as the 10.3.5 subnet), would you ever see a packet that contained the subnet value 10.3.1 in the source AND destination fields? Hmmmm..... not if we’re on a different network. If we did see this traffic pattern on our network, we’d have to assume there is something very wrong with the routing on this network. --Laura

The final filter definition is shown in Figure 2-26.



OR Source IP address = 10.3.1.x OR Destination IP address = 1(
PAT Source IP address = 10.3.1.x
PAT Destination IP address = 10.3.1

10.3.1.x Filter
All ICMP
ARP from Device 1
Default
DNS Server 1
Fish
Fragments - All
Fragments (more to come)
From Fred
FTP
FTP: STOR, RETR, NLST
ICMP: Destination Unreac
IP/HTTP and HTTPS
IP/POP3
IP/TCP FTP Stuff
NLST from 10.2.0.2
TCP SYN only

FIGURE 2-26. **This OR filter will capture traffic going to or coming from subnet 10.3.1.**

AND NOT (Catching All Fragmented Packets)

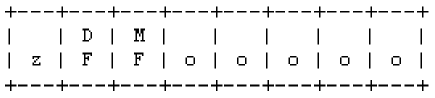
IP has the ability to fragment packets when packets must cross a network that supports a smaller MTU (maximum transmission unit), or payload. For example, consider a 4 KB packet originating on a Token Ring network that needs to cross a router onto an Ethernet network that only supports a maximum packet size of 1518 bytes. This packet would need to be fragmented into three separate packets to make it onto the Ethernet network.

LINK

I don't like fragmentation. There are too many possibilities of lost packets that could cause a full retransmission and many network hacks are based on fragmented packets. If you would like more information on IP fragmentation, refer to the NetWare Connections article I wrote - http://www.nwconnection.com/2000_03/fragment30/index.html. --Laura

Check back to figures 2-8 or 2-9 to review the format of TCP/IP and UDP/IP communications.

The breakdown of the Flags and Fragment Offset fields shown below in binary format:



```
+---+---+---+---+---+---+---+---+
|   | D | M |   |   |   |   |   |
| z | F | F | o | o | o | o | o |
+---+---+---+---+---+---+---+---+
```

FIGURE 2-27. **The IP flags field breakdown.**

Where 'z' is always set to zero, and

'DF' identifies the 'Don't Fragment bit' which means
'Don't Fragment' when set to 1
'OK to Fragment' when set to 0
and

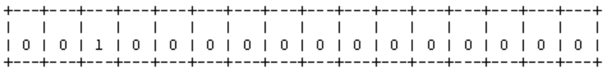
'MF' identifies the 'More Fragments' bit which means
'More Fragments to Come' when set to 1
'Last Fragment' when set to 0
and

"o" identifies the offset field which defines the location of this packet in the entire data stream.

A simple filter based on "1 in the more to come bit" captures all fragments except the last fragment in the set because the last fragment in the set contains the value 0 in the 'more to come' bit field. This requires some extra thinking…

Here are the characteristics of the various fragmented and unfragmented packets seen on the network:

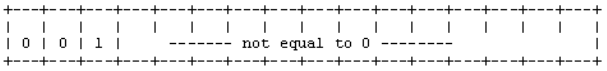
First Fragment: More to come bit = 1; Fragment Offset = 0



```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

FIGURE 2-28. **Bit values for first fragment of a set.**

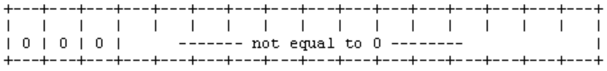
Middle Fragments: More to come bit = 1; Fragment Offset NOT 0



```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 1 |     ------- not equal to 0 --------               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

FIGURE 2-29. **Bit values for middle fragment of a set.**

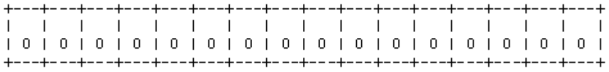
Last Fragment: More to come bit = 0; Fragment Offset NOT 0



```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 |     ------- not equal to 0 --------               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

FIGURE 2-30. **Bit values for last fragment of a set.**

Unfragmented Packets: More to come bit = 0; Fragment Offset = 0

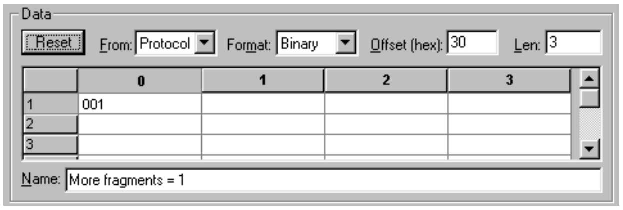


```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

FIGURE 2-31. **Bit values for unfragmented packet.**

Can you find the defining patterns that would enable you to capture all fragments?

Figures 2-32 and 2-33 show the two simple filters that can capture all fragmented traffic on the network.

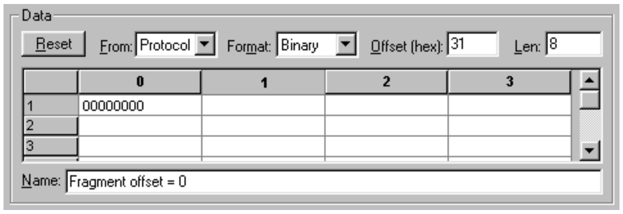


From: Protocol Format: Binary Offset (hex): 30 Len: 3

	0	1	2	3
1	001			
2				
3				

Name: More fragments = 1

FIGURE 2-32. **Pattern 1: Packets with more to come bit set to 1.**



From: Protocol Format: Binary Offset (hex): 31 Len: 8

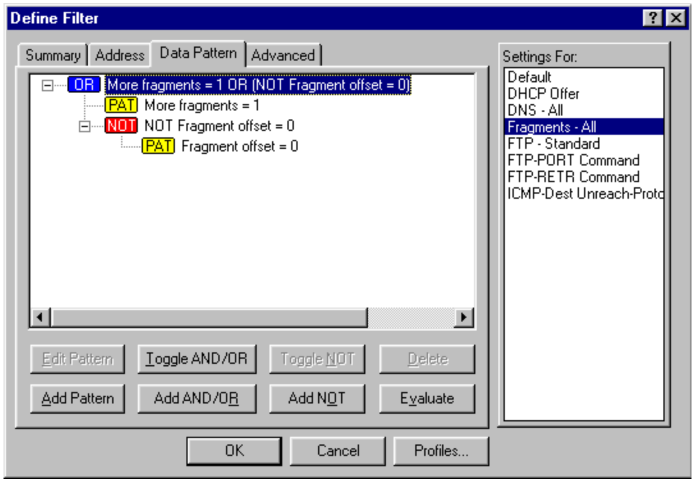
	0	1	2	3
1	00000000			
2				
3				

Name: Fragment offset = 0

FIGURE 2-33. **Pattern 2: Packets that do not contain offset value 0.**

Yes! That's it! All packets in a fragment set have either '1' in the 'More Fragments' bit location or they have some value other than 0 in the offset field!

And we can even use Sniffer's binary format to make it a bit more clear instead of jumping to hex translations and all that garbage! Check out Figure 2-34 that contains the summary of the final filter that uses the AND NOT operand.



OR More fragments = 1 OR (NOT Fragment offset = 0)
  PAT More fragments = 1
  NOT NOT Fragment offset = 0
    PAT Fragment offset = 0

Default
DHCP Offer
DNS - All
Fragments - All
FTP - Standard
FTP-PORT Command
FTP-RETR Command
ICMP-Dest Unreach-Protc

FIGURE 2-34. **Wow! What a slick little filter!**

If you've ever heard me lecture on filtering, then you know that I consider it a true art form. You need to know your protocols - know where to get the offset and field value information - know how to figure out what the possible variations are - and test your filters.

NOTE

Once I was onsite doing a lecture about the glories of protocol analysis when the client reported that a hacker had gotten through their firewall the night before. Well... there went that class schedule! We worked as a group to build a firewall filter test on my analyzer, then we packed up the analyzers and moved to the server room. We placed the analyzer on the firewall network and started it up. Then we went to lunch. When we returned, the analyzer buffer was full. We had a lovely trace of a sequenced break in -- we caught the reconnaissance portion of the attack. Since the probing device used it's actual IP address, we could track the source and nail them. Aaaaaahhhh.... analysis can be such fun. --Laura

Chapter Quiz

Spend a few moments reviewing this quiz. Review the chapter as needed. The answers are contained in Appendix A, “Answers to Chapter Quizzes.”

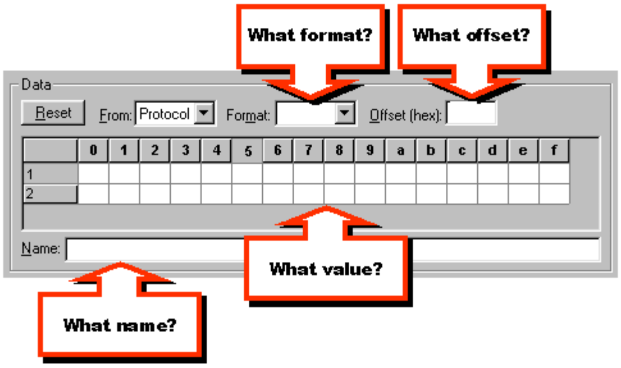
Question 2-1: What is the purpose of a boolean filter?

____

Question 2-2: What portion of the packet could you build a filter on to capture all traffic that passes through a specific router onto the local network?

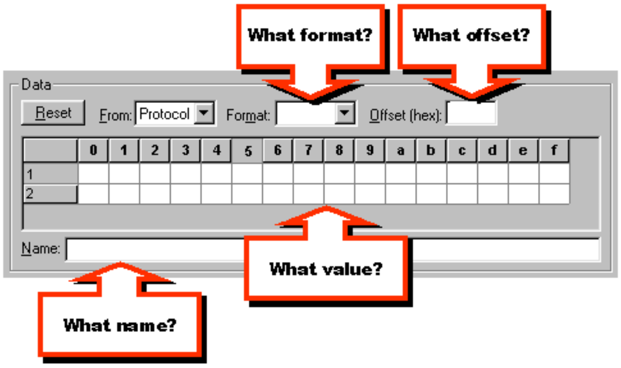
____

Question 2-3: Fill out the following filter window to capture all traffic to network 130.56.x.x.


From: Protocol

Question 2-4: Where do you obtain the latest UDP/TCP port number assignments?

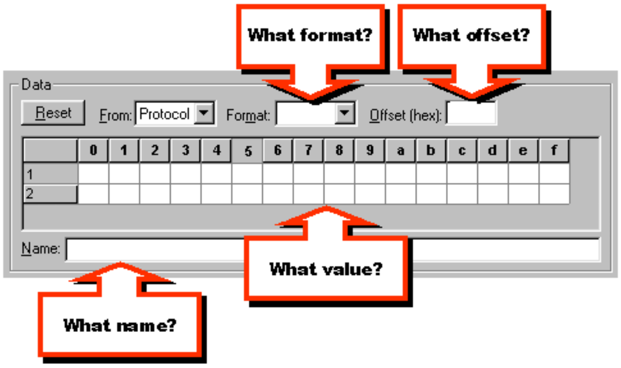
Question 2-5: Refer to the HTTP RFC to set up a filter to catch all HTTP GET traffic.



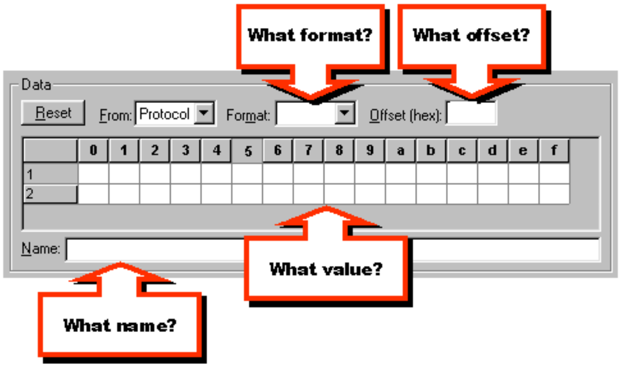
From: Protocol

Question 2-6: How can you use filters to identify hacker attacks?

Question 2-7: Fill out the following window to create a filter for all ARP communications (requests and replies).

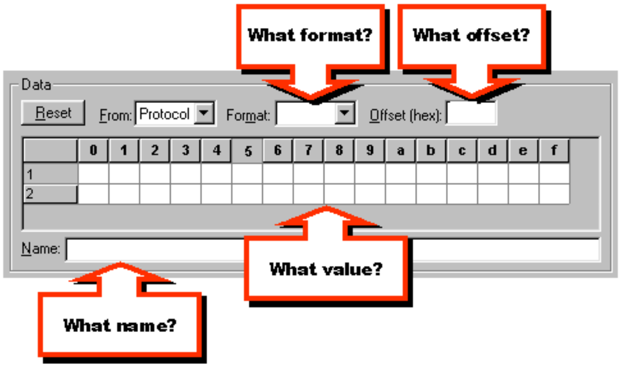


Question 2-8: Fill out the following filter window to create a filter for all DHCP Discovery packets, but no other DHCP packets.



Question 2-9: What type of filter can be used to create a filter for all DNS communications over UDP and TCP?

Question 2-10: Fill out the following window to create a filter to capture all ICMP 'fragmentation required, but don't fragment bit set' messages.



From: Protocol

CHAPTER 3

Application Analysis

This chapter provides step-by-step instructions on how to analyze an application using an Application Analysis Form (see Appendix D). Softcopy of the form can be found online at **www.packet-level.com**. You will learn what to look for in applications (the signatures and unique characteristics of applications), the methods for running the tests, and application timestamp tracking.

Why Analyze an Application

There are many reasons to analyze an application:

- prepare for the necessary bandwidth required to deploy the application throughout the network
- baseline an application to ensure that when users complain of network problems, you can compare the application's current communications with the baseline you created
- identify supplemental files that the application relies on and determine the optimal location for those files
- time the application launch and key tasks to create a baseline of application performance timing
- determine the level of security offered by the application
- determine if the application relies on any special protocols that require network reconfiguration (multicasting, for example)
- identify any 'idle time' processes that might keep a dial-on-demand link up indefinitely
- ... and more.

Big Money Applications

When management cuts your Christmas bonus because they spent too much money on some ‘commode diode’ application, you gotta make sure that baby runs. God forbid the application would fall apart in the middle of the night (which is the only time those applications crash, eh?). One great reason to perform an application analysis session is to document how that application works when it *does* work.

When the application barfs all over the server room whenever anyone wants to run it (technically called a purge process), you can analyze the communication to find out what’s different... what’s new. This process can also help resolve some ‘finger pointing’ that occurs when these big money apps die the big death.

Applications From Hell

I have been on numerous networks that were crippled by a single application that the company was totally dependent on. In many cases, the application was customized (spelled cu$tomized) to address the company’s specific needs and the company was so dependent on the application that the very thought of removing the lousy stinkin’ application made people sick to their stomachs.

In one case, I analyzed an application just days before deployment -- as it turned out, the application (a simple news feed program) would have taken up all the bandwidth on the WAN links, leaving absolutely no room for data.

Management From Hell

You have the dubious responsibility of convincing your management that application traffic should be analyzed BEFORE they buy it. There’s nothing more frustrating (ok... that might be a stretch) than arriving at a network to ‘solve’ their problems, when their management is forcing lousy applications down the collective MIS throat. Ugh.

In a recent analysis session, I encountered a ‘standardized desktop application’ that had been required by the owner of the company. Instead of scheduling a test of the application before deployment, the company owner insisted on immediate rollout. The application is a dog -- no, that’s not fair to dogs. The application is a piece of dog poop. Requiring up to 80,000 packets for the login process, users wait up to 3 minutes to log in each morning. Because the process is so lengthy, users don’t log out at night (even from remote sites). As you can imagine, this application that “standardized and simplified the network” turned the network into a tangled web of traffic.

When to Perform a Complete Application Analysis

In an ideal world, every application would undergo a thorough analysis BEFORE being rolled out onto the network. Unfortunately, we often have non-technical folks in charge of evaluating and purchasing software for the network.

So how do you convince management that application analysis is a requirement? Easy -- analyze one of the most popular applications currently running on your network -- then build a nice application analysis report and let the management see how much information you can gather about the effect that application has on the network.

Application Analysis Procedures

In this section, we'll look at the basic steps required to analyze an application. These steps include:

1 Outline the application functions you want to analyze.

2 Prepare the Application Analysis Form (see Appendix D).

3 Launch your analyzer with a filter on the test station.

4 Record starting packet count (0).

5 Launch the application.

6 Record packet count (when it stops incrementing).

7 Execute command #1.

8 Record packet count (when it stops incrementing).

9 Execute command #2 and other commands in the test.

10 View the trace file to obtain timestamps and characteristics.

Let's walk through this process one step at a time. After we go through the basic steps required to analyze your application, we'll take a look at several applications to see how they came through their application analysis tests.

Step 1: Outline the application functions you want to analyze

You will need some working knowledge of the application to set up your test properly. You should know how users will execute the file, how to execute the most common tasks with the application, and how to close the application.

NOTE

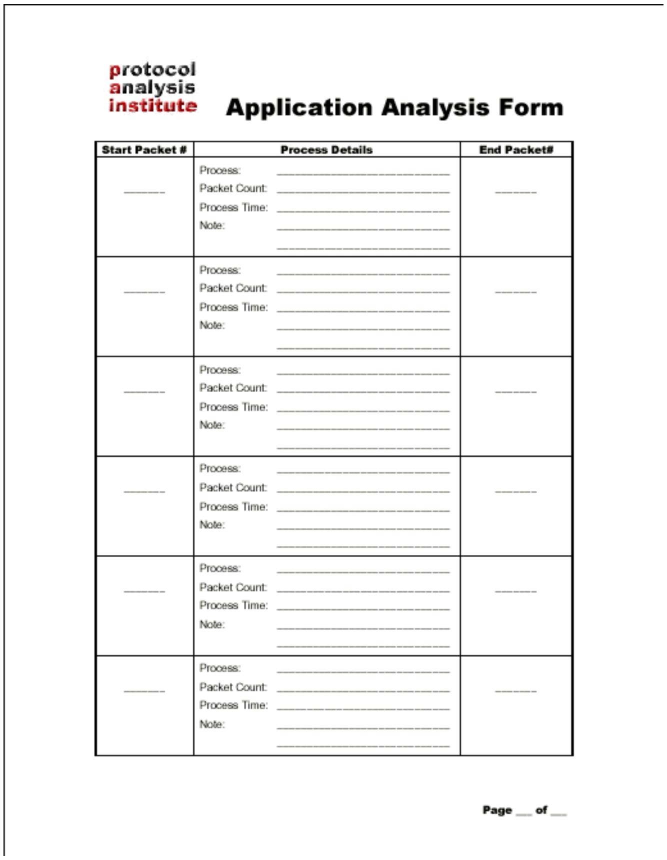
Try to get an assistant who knows the application to help you -- but be forewarned. It can be very frustrating to have the assistant tapping away at the keyboard when you are trying to clock 'idle time between commands' as defined later in this section. --Laura

Step 2: Prepare the Application Analysis Form

Figure 3-1 shows the Application Analysis Form.

NOTE

You can copy, distribute and use the forms in any way you wish. Although the form is copyrighted, we give blanket permissions to use the form for analysis purposes. --Laura



protocol
analysis
institute

Application Analysis Form

Start Packet #	Process Details	End Packet#
_____	Process: ____________________ Packet Count: ____________________ Process Time: ____________________ Note: ____________________ ____________________	_____
_____	Process: ____________________ Packet Count: ____________________ Process Time: ____________________ Note: ____________________ ____________________	_____
_____	Process: ____________________ Packet Count: ____________________ Process Time: ____________________ Note: ____________________ ____________________	_____
_____	Process: ____________________ Packet Count: ____________________ Process Time: ____________________ Note: ____________________ ____________________	_____
_____	Process: ____________________ Packet Count: ____________________ Process Time: ____________________ Note: ____________________ ____________________	_____
_____	Process: ____________________ Packet Count: ____________________ Process Time: ____________________ Note: ____________________ ____________________	_____

Page ___ of ___

FIGURE 3-1. **The Application Analysis Form.**

AAF contains the following fields:

- Application Title/Version Information
- Date
- Start Time
- Stop Time
- Start Packet Number
- Stop Packet Number
- Process Time
- Process Definition

To start filling out an Application Analysis Form, enter the application title/ version information. Enter the date and start packet number (0) and list the processes that you plan on analyzing.

NOTE

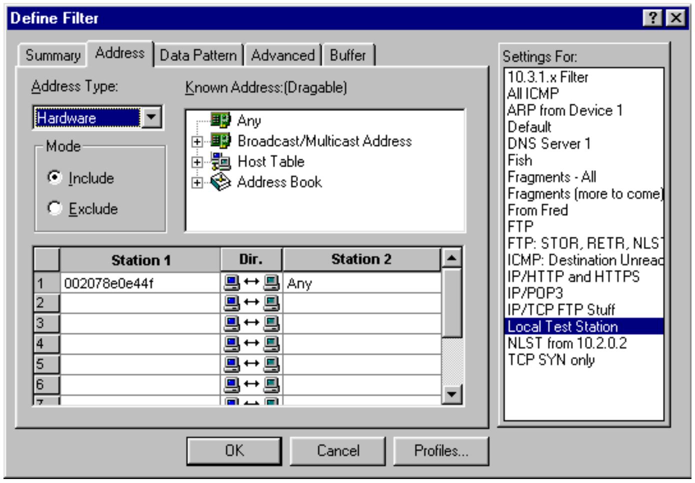
I recommend that you find out about the application launch and shutdown process if possible. You will need to pay special attention to Step 4 -- be certain you have configured your analyzer for a large enough buffer to hold all the packets. -- Laura

Step 3: Launch your analyzer with a filter on the test station.

Now you are ready to start up the analyzer.   This will require three basic tasks:

1. Build a test station filter.

Make sure you build and apply a filter for all traffic to and from the test station. If you use a dynamic addressing system (such as DHCP), set up the filter based on the test station’s hardware address, as shown in Figure 3-2.



Address Type: Hardware

Mode

● Include

○ Exclude

Known Address:(Dragable)

- Any
- ⊞ Broadcast/Multicast Address
- ⊞ Host Table
- ⊞ Address Book

	Station 1	Dir.	Station 2
1	002078e0e44f	↔	Any
2		↔	
3		↔	
4		↔	
5		↔	
6		↔	

Settings For:

10.3.1.x Filter
All ICMP
ARP from Device 1
Default
DNS Server 1
Fish
Fragments - All
Fragments (more to come)
From Fred
FTP
FTP: STOR, RETR, NLST
ICMP: Destination Unreac
IP/HTTP and HTTPS
IP/POP3
IP/TCP FTP Stuff
Local Test Station
NLST from 10.2.0.2
TCP SYN only

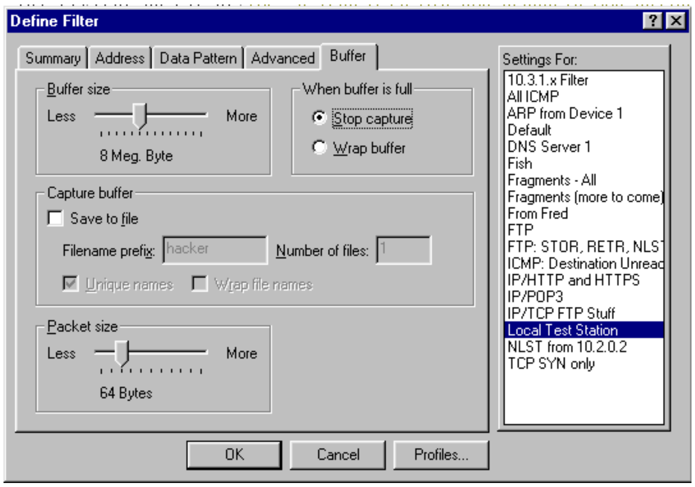
FIGURE 3-2. **Use a MAC filter when testing the bootup sequence of devices that use DHCP for addressing.**

By now, you should be aware of the analyzer placement (see “Introduction to Network Analysis” available at www.podbooks.com). If you are working on a switched network, see Appendix B, “Analyzing Switched Networks.”

2. Set up the appropriate buffer size.

You may need to run through the entire exercise without filling out the Application Analysis Form to find out how much data will cross the wire during the entire test. If you just want to try a size, 16 Mb is a good starting buffer size for most basic tests.

If your buffer fills up during the test and you have to split the operation over multiple trace files, it really can ‘dirty up’ the timestamping mechanism. I highly recommend that you build a ‘single pass’ test that can hold all the test packets in the buffer. This may require you to reduce the packet size to hold more packets in the buffer. Figure 3-3 shows the Sniffer filter configured for minimal Ethernet packets - packet slices.



Define Filter

Buffer size
Less — More
8 Meg. Byte

When buffer is full
● Stop capture
○ Wrap buffer

Capture buffer
□ Save to file
Filename prefix: hacker
Number of files: 1
☑ Unique names
□ Wrap file names

Packet size
Less — More
64 Bytes

Settings For:
10.3.1.x Filter
All ICMP
ARP from Device 1
Default
DNS Server 1
Fish
Fragments - All
Fragments (more to come)
From Fred
FTP
FTP: STOR, RETR, NLST
ICMP: Destination Unreac
IP/HTTP and HTTPS
IP/POP3
IP/TCP FTP Stuff
Local Test Station
NLST from 10.2.0.2
TCP SYN only

FIGURE 3-3. **Packet slicing enables you to get more packets into the buffer.**

3. Test your filter.

You can test your filter by simply starting up the test station and performing a ping or some other identifiable task. If you startup your analyzer when the test station is off, you should not see any packets in the buffer.

Step 4: Record starting packet count

When you start your test, the initial packet count should be 0 if the test station was idle. If the station was not idle, ensure you place the current number of packets in the trace buffer in the first Start Packet box of the Application Analysis Form.

Step 5: Launch the application

Now you are finally ready to launch the application. Just launch the application and then take your hands (or your assistant's hands) off the computer. Do not enter a password. Do not provide any manual interaction at all.

NOTE

This is the most difficult part of application analysis. For some reason, people cannot keep their hands off the keyboard during the test -- I guess we are all trained to interact the moment we are prompted. <sigh> In some instances, I have had to place a fake keyboard in front of a system when testing the startup sequence (just the startup - no login or anything - a great test to do). -- Laura

Step 6: Record packet count (when it stops incrementing)

When the communication stops between the devices (as witnessed by the capture dial typically, shown in Figure 3-4), record the packet count.

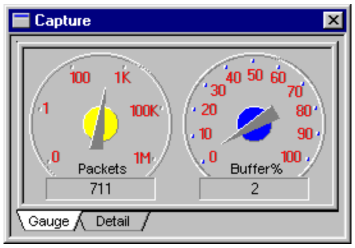
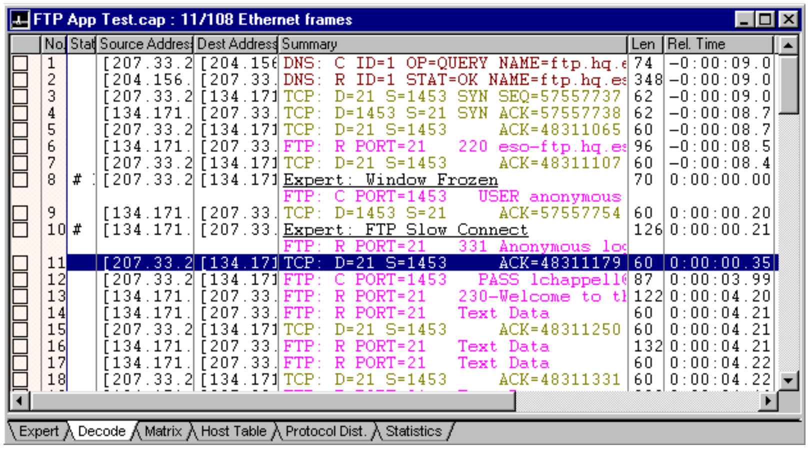


FIGURE 3-4. **When the capture dial stops incrementing, you can assume the process has completed.**

Later, you will use the timestamping feature of the analyzer to determine the total time required for the application launch (subtract the time of the first packet arrival from the time of the last packet arrival or just mark the first packet of the process and use the relative timestamp information).

In Figure 3-5, I have highlighted packet 11, which is the end of the login process. I have marked packet number 8 (which sets the timestamp to

0:00:00:00 to find the total time required to perform an anonymous login (just submitting the user name) on my test.



FTP App Test.cap : 11/108 Ethernet frames

	No	Stat	Source Address	Dest Address	Summary	Len	Rel. Time
□	1		[207.33.2	[204.156	DNS: C ID=1 OP=QUERY NAME=ftp.hq.e	74	-0:00:09.0
□	2		[204.156.	[207.33.	DNS: R ID=1 STAT=OK NAME=ftp.hq.es	348	-0:00:09.0
□	3		[207.33.2	[134.171	TCP: D=21 S=1453 SYN SEQ=57557737	62	-0:00:09.0
□	4		[134.171.	[207.33.	TCP: D=1453 S=21 SYN ACK=57557738	62	-0:00:08.7
□	5		[207.33.2	[134.171	TCP: D=21 S=1453 ACK=48311065	60	-0:00:08.7
□	6		[134.171.	[207.33.	FTP: R PORT=21 220 eso-ftp.hq.es	96	-0:00:08.5
□	7		[207.33.2	[134.171	TCP: D=21 S=1453 ACK=48311107	60	-0:00:08.4
□	8	# :	[207.33.2	[134.171	Expert: Window Frozen FTP: C PORT=1453 USER anonymous	70	0:00:00.00
□	9		[134.171.	[207.33.	TCP: D=1453 S=21 ACK=57557754	60	0:00:00.20
□	10	#	[134.171.	[207.33.	Expert: FTP Slow Connect FTP: R PORT=21 331 Anonymous log	126	0:00:00.21
□	11		[207.33.2	[134.171	TCP: D=21 S=1453 ACK=48311179	60	0:00:00.35
□	12		[207.33.2	[134.171	FTP: C PORT=1453 PASS lchappell@	87	0:00:03.99
□	13		[134.171.	[207.33.	FTP: R PORT=21 230-Welcome to th	122	0:00:04.20
□	14		[134.171.	[207.33.	FTP: R PORT=21 Text Data	60	0:00:04.21
□	15		[207.33.2	[134.171	TCP: D=21 S=1453 ACK=48311250	60	0:00:04.21
□	16		[134.171.	[207.33.	FTP: R PORT=21 Text Data	132	0:00:04.21
□	17		[134.171.	[207.33.	FTP: R PORT=21 Text Data	60	0:00:04.22
□	18		[207.33.2	[134.171	TCP: D=21 S=1453 ACK=48311331	60	0:00:04.22

FIGURE 3-5. **Packet 8 is marked so it's timestamp is now 0:00:00:00.**

The time to complete a name submission for login (from packet 8 through packet 11) is .35 seconds.

Step 7: Execute command #1

Now you are ready to perform another test -- execute the first command. Again, try to remember not to interact with the computer other than to perform the basic command.

Step 8: Record packet count (when it stops incrementing)

Again, record the ending packet count when the traffic from the test device appears to idle (as shown in the capture dial).

Step 9: Execute command #2 and other commands in the test

Now you are ready to execute the second command and continue the testing process by following the Step 7 through Step 8 sequence until you have completed your entire test.

Step 10: View the trace file to obtain timestamps and characteristics

Now you are ready to check out the timestamps of each process that you tested. On the Sniffer, simply go to one of the start packet numbers and mark the packet -- the relative timestamp value changes to 0:00.00:00. Scroll to the packet that was the Stop packet in the test. What is the relative timestamp value? This provides the execution time for the task.

In the next pages, we will perform some application analysis tasks and show you the results of the tests.

Sample Application Analysis: FTP File Transfer

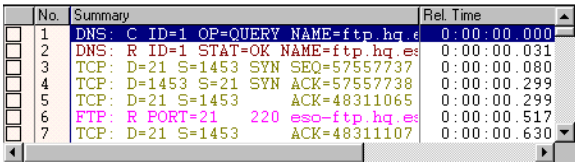
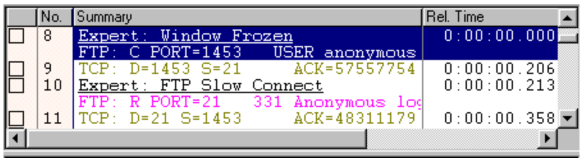
To show how an application test works I've decided to analyze an FTP communication to get a file from an FTP site on the Internet. Specifically, I am looking for the logo for Skycat Tool.

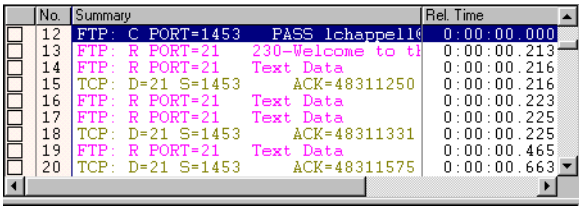
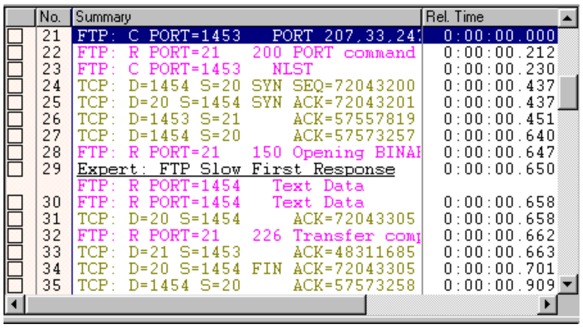
LINK

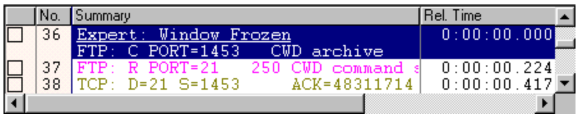
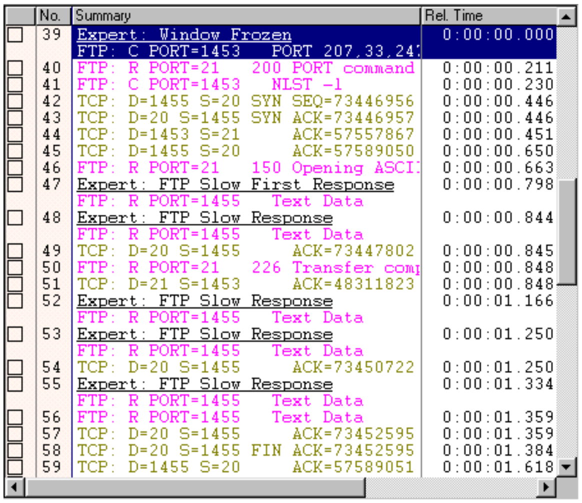
SkyCat is a tool that combines visualization of images and access to catalogs and archive data for astronomy. It is available from the European Southern Observatory (ESO) -- a must-know website for astronomy lovers. --Laura

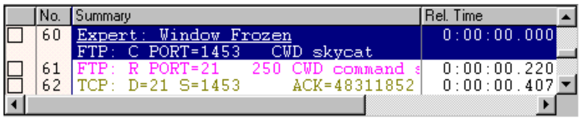
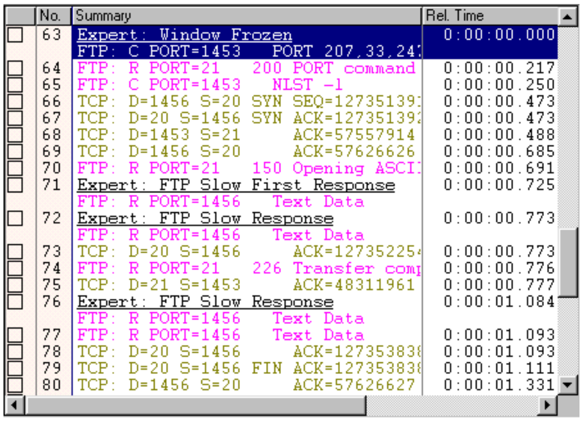
The following pages shows my Application Analysis Form. You can get the actual trace file at www.packet-level.com/traces/ftp-skycat.zip. I've included the Sniffer screenshots that include the timestamp settings.

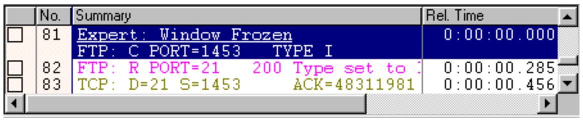
Test Name: FTP File Transfer Test

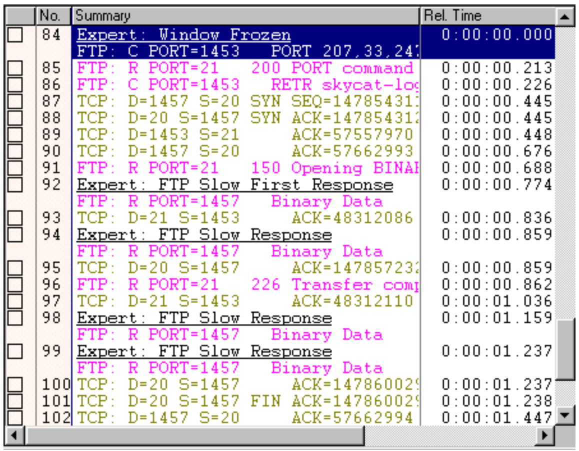
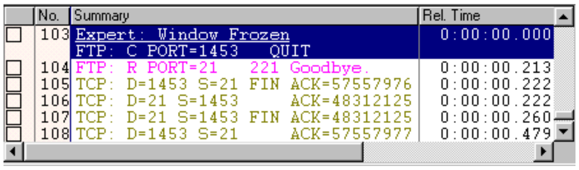
Start Packet	Process Description	End Packet
0	Process: Make FTP Connection Packet Count: 7 Process Time: .630 seconds Notes: Includes DNS lookup, TCP Handshake to 134.171.15.219, and the initial FTP server response.  No. Summary Rel. Time □ 1 DNS: C ID=1 OP=QUERY NAME=ftp.hq.e 0:00:00.000 □ 2 DNS: R ID=1 STAT=OK NAME=ftp.hq.es 0:00:00.031 □ 3 TCP: D=21 S=1453 SYN SEQ=57557737 0:00:00.080 □ 4 TCP: D=1453 S=21 SYN ACK=57557738 0:00:00.299 □ 5 TCP: D=21 S=1453 ACK=48311065 0:00:00.299 □ 6 FTP: R PORT=21 220 eso-ftp.hq.es 0:00:00.517 □ 7 TCP: D=21 S=1453 ACK=48311107 0:00:00.630	7
8	Process: Input “anonymous” Username to Login to FTP Server Packet Count: 4 Process Time: .358 seconds Notes: Includes submission of name and server’s request for password.  No. Summary Rel. Time □ 8 Expert: Window Frozen 0:00:00.000 FTP: C PORT=1453 USER anonymous □ 9 TCP: D=1453 S=21 ACK=57557754 0:00:00.206 □ 10 Expert: FTP Slow Connect 0:00:00.213 FTP: R PORT=21 331 Anonymous log □ 11 TCP: D=21 S=1453 ACK=48311179 0:00:00.358	11

Start Packet	Process Description	End Packet
12	Process: Input email address as password. Packet Count: 9 Process Time: .663 seconds Notes: Entering password and receiving back the server's welcome screen.  □ No. Summary Rel. Time □ 12 FTP: C PORT=1453 PASS lchappell 0:00:00.000 □ 13 FTP: R PORT=21 230-Welcome to t 0:00:00.213 □ 14 FTP: R PORT=21 Text Data 0:00:00.216 □ 15 TCP: D=21 S=1453 ACK=48311250 0:00:00.216 □ 16 FTP: R PORT=21 Text Data 0:00:00.223 □ 17 FTP: R PORT=21 Text Data 0:00:00.225 □ 18 TCP: D=21 S=1453 ACK=48311331 0:00:00.225 □ 19 FTP: R PORT=21 Text Data 0:00:00.465 □ 20 TCP: D=21 S=1453 ACK=48311575 0:00:00.663	20
21	Process: List Files in Directory (LS Command) Packet Count: 15 Process Time: .909 seconds Notes: This includes the PORT command execution, an automatic binary connection for transfer the directory.  □ No. Summary Rel. Time □ 21 FTP: C PORT=1453 PORT 207,33,24 0:00:00.000 □ 22 FTP: R PORT=21 200 PORT command 0:00:00.212 □ 23 FTP: C PORT=1453 NLST 0:00:00.230 □ 24 TCP: D=1454 S=20 SYN SEQ=72043200 0:00:00.437 □ 25 TCP: D=20 S=1454 SYN ACK=72043201 0:00:00.437 □ 26 TCP: D=1453 S=21 ACK=57557819 0:00:00.451 □ 27 TCP: D=1454 S=20 ACK=57573257 0:00:00.640 □ 28 FTP: R PORT=21 150 Opening BINA 0:00:00.647 □ 29 Expert: FTP Slow First Response 0:00:00.650 FTP: R PORT=1454 Text Data □ 30 FTP: R PORT=1454 Text Data 0:00:00.658 □ 31 TCP: D=20 S=1454 ACK=72043305 0:00:00.658 □ 32 FTP: R PORT=21 226 Transfer com 0:00:00.662 □ 33 TCP: D=21 S=1453 ACK=48311685 0:00:00.663 □ 34 TCP: D=20 S=1454 FIN ACK=72043305 0:00:00.701 □ 35 TCP: D=1454 S=20 ACK=57573258 0:00:00.909	35

Start Packet	Process Description	End Packet
36	Process: Change into the *archive* Directory Packet Count: 3 Process Time: .417 seconds Notes: Simple CWD (Change Working Directory) command and ACK.  No. Summary Rel. Time 36 Expert: Window Frozen 0:00:00.000 FTP: C PORT=1453 CWD archive 37 FTP: R PORT=21 250 CWD command s 0:00:00.224 38 TCP: D=21 S=1453 ACK=48311714 0:00:00.417	38
39	Process: Full Directory Listing (LS -L) in *archive* Directory. Packet Count: 21 Process Time: 1.618 seconds Notes: Includes another NLST command (this time with the -L parameter) as well as another PORT command. Interestingly, the -L parameter triggers the directory list transfer to be made in ASCII format.  No. Summary Rel. Time 39 Expert: Window Frozen 0:00:00.000 FTP: C PORT=1453 PORT 207,33,24 40 FTP: R PORT=21 200 PORT command 0:00:00.211 41 FTP: C PORT=1453 NLST -l 0:00:00.230 42 TCP: D=1455 S=20 SYN SEQ=73446956 0:00:00.446 43 TCP: D=20 S=1455 SYN ACK=73446957 0:00:00.446 44 TCP: D=1453 S=21 ACK=57557867 0:00:00.451 45 TCP: D=1455 S=20 ACK=57589050 0:00:00.650 46 FTP: R PORT=21 150 Opening ASCI 0:00:00.663 47 Expert: FTP Slow First Response 0:00:00.798 FTP: R PORT=1455 Text Data 48 Expert: FTP Slow Response 0:00:00.844 FTP: R PORT=1455 Text Data 49 TCP: D=20 S=1455 ACK=73447802 0:00:00.845 50 FTP: R PORT=21 226 Transfer com 0:00:00.848 51 TCP: D=21 S=1453 ACK=48311823 0:00:00.848 52 Expert: FTP Slow Response 0:00:01.166 FTP: R PORT=1455 Text Data 53 Expert: FTP Slow Response 0:00:01.250 FTP: R PORT=1455 Text Data 54 TCP: D=20 S=1455 ACK=73450722 0:00:01.250 55 Expert: FTP Slow Response 0:00:01.334 FTP: R PORT=1455 Text Data 56 FTP: R PORT=1455 Text Data 0:00:01.359 57 TCP: D=20 S=1455 ACK=73452595 0:00:01.359 58 TCP: D=20 S=1455 FIN ACK=73452595 0:00:01.384 59 TCP: D=1455 S=20 ACK=57589051 0:00:01.618	59

Start Packet	Process Description	End Packet
60	Process: Change to *skycat* Directory. Packet Count: 3 Process Time: .407 seconds Notes: Simple CWD command with an ACK.  No. Summary Rel. Time ☐ 60 Expert: Window Frozen 0:00:00.000 FTP: C PORT=1453 CWD skycat ☐ 61 FTP: R PORT=21 250 CWD command s 0:00:00.220 ☐ 62 TCP: D=21 S=1453 ACK=48311852 0:00:00.407	62
63	Process: View skycat Directory in Long Form (LS -L) Packet Count: 18 Process Time: 1.331 seconds Notes: Again, we find that the -L command causes the FTP communications to use ASCII mode.  No. Summary Rel. Time ☐ 63 Expert: Window Frozen 0:00:00.000 FTP: C PORT=1453 PORT 207,33,24 ☐ 64 FTP: R PORT=21 200 PORT command 0:00:00.217 ☐ 65 FTP: C PORT=1453 NLST -l 0:00:00.250 ☐ 66 TCP: D=1456 S=20 SYN SEQ=12735139 0:00:00.473 ☐ 67 TCP: D=20 S=1456 SYN ACK=12735139 0:00:00.473 ☐ 68 TCP: D=1453 S=21 ACK=57557914 0:00:00.488 ☐ 69 TCP: D=1456 S=20 ACK=57626626 0:00:00.685 ☐ 70 FTP: R PORT=21 150 Opening ASCI 0:00:00.691 ☐ 71 Expert: FTP Slow First Response 0:00:00.725 FTP: R PORT=1456 Text Data ☐ 72 Expert: FTP Slow Response 0:00:00.773 FTP: R PORT=1456 Text Data ☐ 73 TCP: D=20 S=1456 ACK=12735225 0:00:00.773 ☐ 74 FTP: R PORT=21 226 Transfer comp 0:00:00.776 ☐ 75 TCP: D=21 S=1453 ACK=48311961 0:00:00.777 ☐ 76 Expert: FTP Slow Response 0:00:01.084 FTP: R PORT=1456 Text Data ☐ 77 FTP: R PORT=1456 Text Data 0:00:01.093 ☐ 78 TCP: D=20 S=1456 ACK=12735383 0:00:01.093 ☐ 79 TCP: D=20 S=1456 FIN ACK=12735383 0:00:01.111 ☐ 80 TCP: D=1456 S=20 ACK=57626627 0:00:01.331	80

Start Packet	Process Description	End Packet
81	Process: Manually Switch to BINARY Mode. Packet Count: 3 Process Time: .456 seconds Notes: Command to set communications to type “I” (which stands for Image mode).  No. Summary Rel. Time □ 81 Expert: Window Frozen 0:00:00.000 FTP: C PORT=1453 TYPE I □ 82 FTP: R PORT=21 200 Type set to I 0:00:00.285 □ 83 TCP: D=21 S=1453 ACK=48311981 0:00:00.456	83

Start Packet	Process Description	End Packet
84	Process: Get *skycat* Logo File - *skycat-logo.gif.* Packet Count: 23 Process Time: 1.447 seconds Notes: Here we see the PORT command, the RETR command, another TCP handshake to open the new port, the binary command (indicating that we did not need to issue the command in the previous step) and finally, the graphic file is transferred.  No. Summary Rel. Time 84 Expert: Window Frozen 0:00:00.000 FTP: C PORT=1453 PORT 207.33.24 85 FTP: R PORT=21 200 PORT command 0:00:00.213 86 FTP: C PORT=1453 RETR skycat-lo 0:00:00.226 87 TCP: D=1457 S=20 SYN SEQ=14785431 0:00:00.445 88 TCP: D=20 S=1457 SYN ACK=14785431 0:00:00.445 89 TCP: D=1453 S=21 ACK=57557970 0:00:00.448 90 TCP: D=1457 S=20 ACK=57662993 0:00:00.676 91 FTP: R PORT=21 150 Opening BINA 0:00:00.688 92 Expert: FTP Slow First Response 0:00:00.774 FTP: R PORT=1457 Binary Data 93 TCP: D=21 S=1453 ACK=48312086 0:00:00.836 94 Expert: FTP Slow Response 0:00:00.859 FTP: R PORT=1457 Binary Data 95 TCP: D=20 S=1457 ACK=14785723 0:00:00.859 96 FTP: R PORT=21 226 Transfer comp 0:00:00.862 97 TCP: D=21 S=1453 ACK=48312110 0:00:01.036 98 Expert: FTP Slow Response 0:00:01.159 FTP: R PORT=1457 Binary Data 99 Expert: FTP Slow Response 0:00:01.237 FTP: R PORT=1457 Binary Data 100 TCP: D=20 S=1457 ACK=14786002 0:00:01.237 101 TCP: D=20 S=1457 FIN ACK=14786002 0:00:01.238 102 TCP: D=1457 S=20 ACK=57662994 0:00:01.447	102
103	Process: Quit Packet Count: 6 Process Time: .479 seconds Notes: Includes the QUIT command, the server's 'Goodbye' message and the FIN flags setting to indicate the end of the transaction.  No. Summary Rel. Time 103 Expert: Window Frozen 0:00:00.000 FTP: C PORT=1453 QUIT 104 FTP: R PORT=21 221 Goodbye. 0:00:00.213 105 TCP: D=1453 S=21 FIN ACK=57557976 0:00:00.222 106 TCP: D=21 S=1453 ACK=48312125 0:00:00.222 107 TCP: D=21 S=1453 FIN ACK=48312125 0:00:00.260 108 TCP: D=1453 S=21 ACK=57557977 0:00:00.479	108

Wow! It’s amazing how much information you can gather by watching the application communication process step-by-step.

Can you just perform all steps at one without stopping and logging the various stages of the application? Certainly. Unfortunately, however, you would end up with one large file -- in some applications, it may not be so easy to find out where one step of the process ends and another one starts.

Now let’s analyze an HTTP session using NetScape Communicator v4.7.

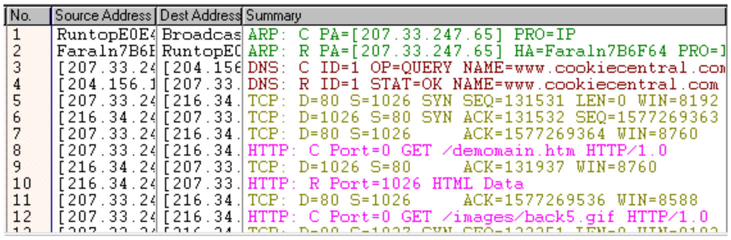
Sample Application Analysis: HTTP Web Browsing Test

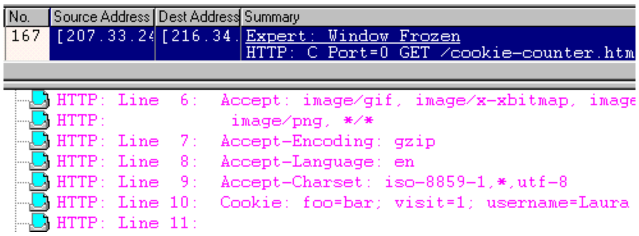
In this next test, we'll filter on the test station again, but this time, we are interested in watching the process of accessing a website (www.packet-level.com) and examining a cookie being sent and reset on the local drive.

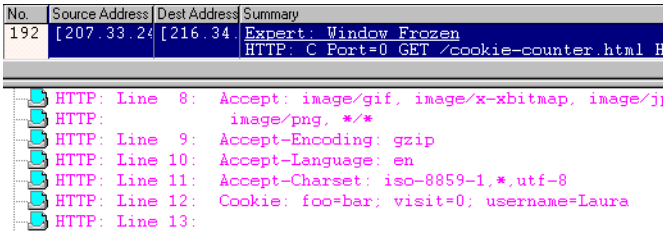
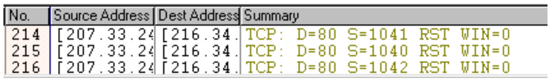
LINK

This trace file is online at www.packet-level.com/traces/http-cookie.zip. -Laura

In this example, we'll be accessing www.cookiecentral.com -- "dedicated to provide full information upon Internet cookies."

Start Packet	Process Description	End Packet
0	Process: Launch Communicator and Access the www.cookiecentral.com/demomain.htm page. Packet Count: 363 Process Time: 18.980 seconds Notes: In this section, you will see the initial ARP and DNS query for the cookiecentral.com website. Next, you see the 3-way handshake and the initial site data transfer.  No. Source Address Dest Address Summary 1 RuntopE0E4 Broadcas ARP: C PA=[207.33.247.65] PRO=IP 2 Faraln7B6F RuntopE0 ARP: R PA=[207.33.247.65] HA=Faraln7B6F64 PRO=I 3 [207.33.24 [204.156 DNS: C ID=1 OP=QUERY NAME=www.cookiecentral.com 4 [204.156.1 [207.33. DNS: R ID=1 STAT=OK NAME=www.cookiecentral.com 5 [207.33.24 [216.34. TCP: D=80 S=1026 SYN SEQ=131531 LEN=0 WIN=8192 6 [216.34.24 [207.33. TCP: D=1026 S=80 SYN ACK=131532 SEQ=1577269363 7 [207.33.24 [216.34. TCP: D=80 S=1026 ACK=1577269364 WIN=8760 8 [207.33.24 [216.34. HTTP: C Port=0 GET /demomain.htm HTTP/1.0 9 [216.34.24 [207.33. TCP: D=1026 S=80 ACK=131937 WIN=8760 10 [216.34.24 [207.33. HTTP: R Port=1026 HTML Data 11 [207.33.24 [216.34. TCP: D=80 S=1026 ACK=1577269536 WIN=8588 12 [207.33.24 [216.34. HTTP: C Port=0 GET /images/back5.gif HTTP/1.0	166

Start Packet	Process Description	End Packet
167	Process: Launch the cookie test. Packet Count: 25 Process Time: less than 1 second (note that this process required manual input when a window popped up -- you can see the delay in the trace file at packet number 188). Notes: Now we can look inside packet #167 to see the cookie being sent to my workstation. As you can see, this cookie is tracking my number of visits to the website.  No. Source Address Dest Address Summary 167 [207.33.24 [216.34. Expert: Window Frozen HTTP: C Port=0 GET /cookie-counter.htm HTTP: Line 6: Accept: image/gif, image/x-xbitmap, image HTTP: image/png, */* HTTP: Line 7: Accept-Encoding: gzip HTTP: Line 8: Accept-Language: en HTTP: Line 9: Accept-Charset: iso-8859-1,*,utf-8 HTTP: Line 10: Cookie: foo=bar; visit=1; username=Laura HTTP: Line 11:	191

Start Packet	Process Description	End Packet
192	Process: Clear the cookie counter. Packet Count: 22 Process Time: .643 seconds -- you'll note however, that there are some FIN flagged packets at the end of this process. I have not counted the delay time to the FIN process. Notes: This website is used to test cookies and see how they function. I simply clicked a 'reset' button to change the cookie value, as you can see in packet 192.  No. | Source Address | Dest Address | Summary 192 | [207.33.24 | [216.34. | Expert: Window Frozen HTTP: C Port=0 GET /cookie-counter.html H HTTP: Line 8: Accept: image/gif, image/x-xbitmap, image/j HTTP: image/png, */* HTTP: Line 9: Accept-Encoding: gzip HTTP: Line 10: Accept-Language: en HTTP: Line 11: Accept-Charset: iso-8859-1,*,utf-8 HTTP: Line 12: Cookie: foo=bar; visit=0; username=Laura HTTP: Line 13:	213
214	Process: Close the browser. Packet Count: 3 Process Time: 0 seconds Notes: You'll see different packet sequences depending on which shutdown method you use.  No. | Source Address | Dest Address | Summary 214 | [207.33.24 | [216.34. | TCP: D=80 S=1041 RST WIN=0 215 | [207.33.24 | [216.34. | TCP: D=80 S=1040 RST WIN=0 216 | [207.33.24 | [216.34. | TCP: D=80 S=1042 RST WIN=0	216

This entire process takes 17 connections starting at 1026 and ending at 1042. Not all these connections are simply to www.cookiecentral.com.

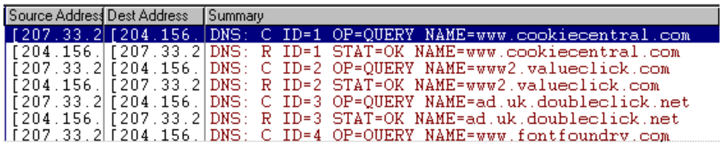
- In packet 24, the client makes a connection to www2.valueclick.com.
- In packet 56, the client makes a connection to ad.uk.doubleclick.net.
- In packet number 78, the client makes a connection to www.fontfoundry.com.

Sometimes these ‘external links’ can really bog down the web browsing process -- especially if the external link is slow in responding.

You won’t always see the same secondary connections on this site -- the connections are based on the banner ads that are painted on the screen. For example, yesterday, I accessed the site and connected to www.anonymiser.com. Not today, though.

Building a simple DNS filter will give you an idea of all the interdependencies that are woven into a web site. The sites that include lots of advertising will load slowly as you make connections to all the banner ad sites. For example, see Figure 3-5, where I have applied a DNS filter. By examining the results, I determined that my client has made connections to the following devices:

216.34.245.27.........	www.cookiecentral.com
209.85.28.135.........	www2.valueclick.com
213.86.246.40.........	ad.uk.doubleclick.net
209.239.56.221.......	www.fontfoundry.com



Source Address	Dest Address	Summary
[207.33.2	[204.156.	DNS: C ID=1 OP=QUERY NAME=www.cookiecentral.com
[204.156.	[207.33.2	DNS: R ID=1 STAT=OK NAME=www.cookiecentral.com
[207.33.2	[204.156.	DNS: C ID=2 OP=QUERY NAME=www2.valueclick.com
[204.156.	[207.33.2	DNS: R ID=2 STAT=OK NAME=www2.valueclick.com
[207.33.2	[204.156.	DNS: C ID=3 OP=QUERY NAME=ad.uk.doubleclick.net
[204.156.	[207.33.2	DNS: R ID=3 STAT=OK NAME=ad.uk.doubleclick.net
[207.33.2	[204.156.	DNS: C ID=4 OP=QUERY NAME=www.fontfoundry.com

FIGURE 3-6. **Applying a DNS filter to your trace will show you other websites that you contacted during the process.**

NOTE

If your clients looked up IP addresses in a local host table, you won’t see a DNS query occur. You might consider filtering on the SYN flag bit.

Why is this dependency important to know? If the connections to these secondary servers cannot be established or are simply slow, the site loading process will be slow and cumbersome. Too many of these dependencies is certainly not a good thing.

pl;\8What other things should you look for when you perform an application test?

- dependencies to other files
- transport process (connection-oriented/connectionless?)
- duplicate requests/replies
- consistent faults
- typical process time (on LAN or WAN)

Application analysis is an important task in determining the various dependencies, security level, packet count, latency and behavior of applications.

For example, I have discovered the following characteristics about various applications I have analyzed:

- The application was not built to run across a network. The application does not cache information and loads and reloads the same file numerous times. The round-trip latency time between requests and replies is quite long. If you multiply that number by the number of requests, you can see why the application loading process is taking so long.
- The application is configured improperly. The application believes that all of its supplemental files are on the server when they are actually on a local drive.
- The application fails to load one of the supplemental files. Why? Perhaps the user does not have sufficient rights, as shown in the NCP reply to the client's Get Effective

Rights request. Perhaps the supplemental files are no longer where they're supposed to be.

- The application is too large to load across the WAN. You should move the application to a local drive or a local server.

- The application has a programming fault (a.k.a., a "bug" or "feature"). An error reply is causing an excessive delay at one point in the application load process.

- The application doesn't read files in sequential order, making windowed reads useless.

- The application uses plaintext passwords and usernames. Whoops!

- The application is a bandwidth hog: Only two users can simultaneously load the application before they consume available bandwidth. (This was a management application, by the way -- Dang, I hate management applications!)

Try performing an application analysis session on your network. I highly recommend that you analyze your email application, login sequence, and any other major application running on your network. Make a binder -- keep the trace files and your analysis of those applications handy -- you'll need 'em someday!

Chapter Three Quiz

Spend a few moments reviewing this quiz. Review the chapter as needed. The answers are contained in Appendix A, “Answers to Chapter Quizzes.”

Question 3-1: When is the best time to analyze an application?

____

Question 3-2: What is the most common time that applications are analyzed?

____

Question 3-3: List at least three characteristics you should document when analyzing an application.

____

____

____

Question 3-4: Should you use a MAC address or network address filter when applying a filter on a test device?

____

Question 3-5: You are analyzing an application that sends 20 MB of data on the wire during the key operations. How do you ensure that you capture all application traffic during the test?

Question 3-6: How do you set up your analyzer if you know you can't fit all the packets into your buffer?

Question 3-7: Why should you track the start packet and stop packet count during an application analysis session?

Question 3-8: How do you identify dependent web sites that are located during an HTTP session?

Question 3-9: Which window should you examine to find out if unencrypted data is crossing the network?

Question 3-10: You've captured all FTP communications to and from your test station. How can you compare what you see in the trace to what is considered 'by the spec' on the application's behavior?

CHAPTER 4

Manual Decoding

This chapter is really fun -- ok, so you don't think manually decoding packets is fun -- C'mon! You don't know what you're missing. And... if you don't know how to decode -- you won't know what you're missing on the wire.

For example, when Novell came out with NetWare 5, suddenly the protocol analyzer companies had to get up to speed with a new set of protocols --- NCP over UDP, NCP over TCP and SLP (Service Location Protocol).

As a protocol analyst, you can't always wait for the manufacturers to get their decodes up to speed -- you need to know how to do manual decoding.

Remember -- every layer of a packet has some identifier that tells you what's coming up next. Here's an example of what I mean:

- The Ethernet header has a type field that indicates the network layer coming up next -- the value 0x0800 means an IP header is coming up next. So.... you go look up RFC 791 to find out what the structure of an IP header is.
- Using RFC 791, you decode the IP header you find there is a protocol field that identifies the next layer. Let's say you find that the value is 17 (decimal). Hey... you find out that 17 is assigned to UDP (you checked www.iana.org > Protocol Num-

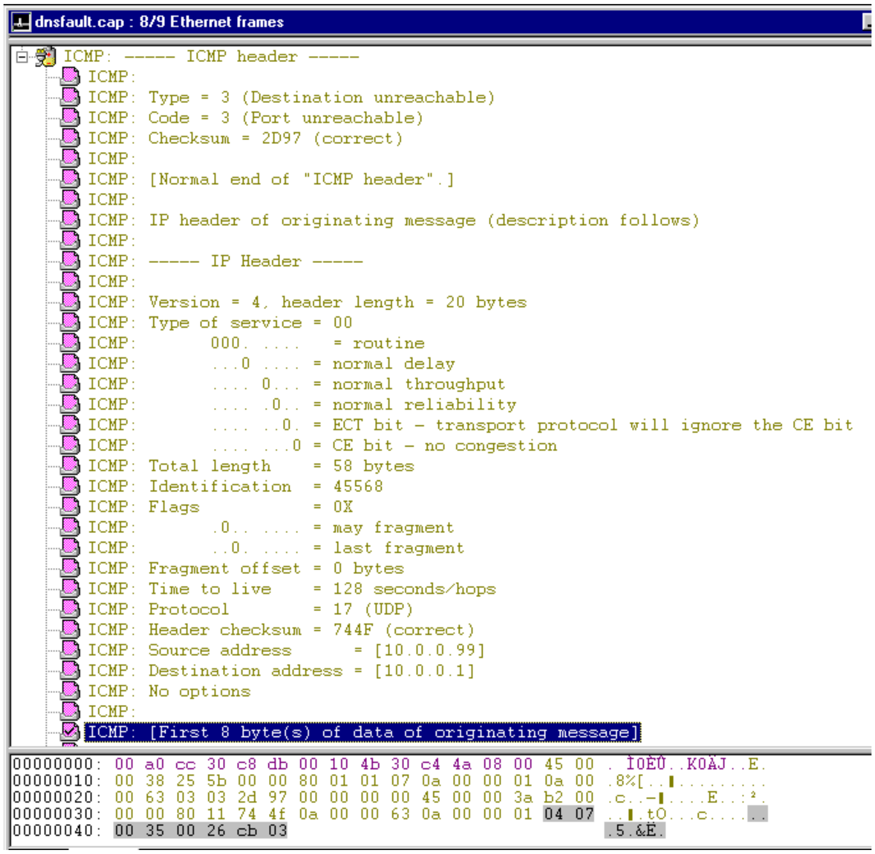
bers and Assignment Services > Protocol Numbers). Now you know to check RFC 768 to find out how to decode UDP.

- So... you use RFC 768 to decode the UDP header and you find that UDP has a port number field that indicates what application is using the UDP transport.
- And so on...

Make sense? Ok... let's go through some decoding...

When the Decodes End

In some instances, you may find that a packet is only partially decoded. For example, Figure 4-45 shows a portion of a ICMP Destination Unreachable message from Sniffer v3.5. The figure shows a portion of the decode window of the ICMP packet as well as the hex window.



```
ICMP: ----- ICMP header -----
ICMP:
ICMP: Type = 3 (Destination unreachable)
ICMP: Code = 3 (Port unreachable)
ICMP: Checksum = 2D97 (correct)
ICMP:
ICMP: [Normal end of "ICMP header".]
ICMP:
ICMP: IP header of originating message (description follows)
ICMP:
ICMP: ----- IP Header -----
ICMP:
ICMP: Version = 4, header length = 20 bytes
ICMP: Type of service = 00
ICMP:       000. ....   = routine
ICMP:       ...0 .... = normal delay
ICMP:       .... 0... = normal throughput
ICMP:       .... .0.. = normal reliability
ICMP:       .... ..0. = ECT bit - transport protocol will ignore the CE bit
ICMP:       .... ...0 = CE bit - no congestion
ICMP: Total length    = 58 bytes
ICMP: Identification  = 45568
ICMP: Flags           = 0X
ICMP:       .0.. .... = may fragment
ICMP:       ..0. .... = last fragment
ICMP: Fragment offset = 0 bytes
ICMP: Time to live    = 128 seconds/hops
ICMP: Protocol        = 17 (UDP)
ICMP: Header checksum = 744F (correct)
ICMP: Source address      = [10.0.0.99]
ICMP: Destination address = [10.0.0.1]
ICMP: No options
ICMP:
ICMP: [First 8 byte(s) of data of originating message]
```

```
00000000: 00 a0 cc 30 c8 db 00 10 4b 30 c4 4a 08 00 45 00  . ÌOÈÛ..K0ÄJ..E.
00000010: 00 38 25 5b 00 00 80 01 01 07 0a 00 00 01 0a 00  .8%[..▮.........
00000020: 00 63 03 03 2d 97 00 00 00 00 45 00 00 3a b2 00  .c..-▮....E..:².
00000030: 00 00 80 11 74 4f 0a 00 00 63 0a 00 00 01 04 07  ..▮.tO...c......
00000040: 00 35 00 26 cb 03                                .5.&Ë.
```

FIGURE 4-1. **A partially decoded packet -- what a bummer!**

The problem with this version of Sniffer's decode is that the packet isn't fully decoded. The highlighted section is not decoded - you only get the irritating message, “First 8 bytes of data of originating message” -- Aaaaargh!

There is NO REASON that the analyzer should leave this undecoded. This is an easy decode to figure out... let's look at the ICMP Destination Unreachable message.

NOTE

RFC 792 provides the packet format for ICMP Destination Unreachable messages. As I've said before, you MUST KNOW this RFC! -- Laura

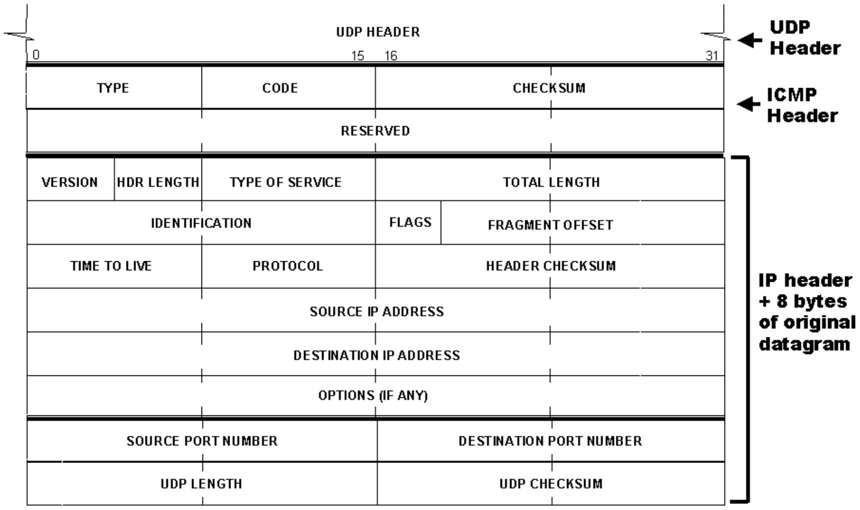
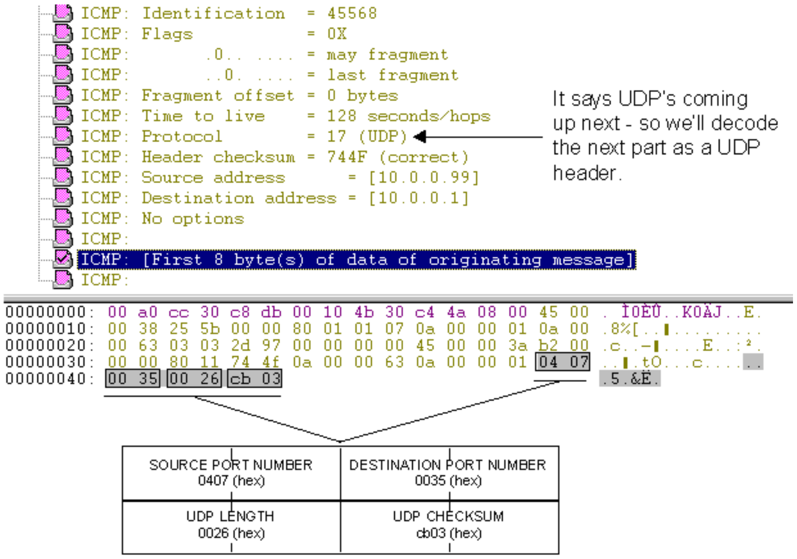


FIGURE 4-2. **The ICMP Destination Unreachable packet format.**

In order to complete the decode, we will need to apply the packet format shown in Figure 4-2 to the undecoded hex data shown in Figure 4-1.

Here's how we do it... check out Figure 4-3.



```
ICMP: Identification  = 45568
ICMP: Flags           = 0X
ICMP:        .0.. .... = may fragment
ICMP:        ..0. .... = last fragment
ICMP: Fragment offset = 0 bytes
ICMP: Time to live    = 128 seconds/hops
ICMP: Protocol        = 17 (UDP)
ICMP: Header checksum = 744F (correct)
ICMP: Source address       = [10.0.0.99]
ICMP: Destination address = [10.0.0.1]
ICMP: No options
ICMP:
ICMP: [First 8 byte(s) of data of originating message]
ICMP:

00000000: 00 a0 cc 30 c8 db 00 10 4b 30 c4 4a 08 00 45 00 . ÌOÈÛ..K0ÄJ..E.
00000010: 00 38 25 5b 00 00 80 01 01 07 0a 00 00 01 0a 00 .8%[..▮.........
00000020: 00 63 03 03 2d 97 00 00 00 00 45 00 00 3a b2 00 .c..-▮....E..:².
00000030: 00 00 80 11 74 4f 0a 00 00 63 0a 00 00 01 04 07 ..▮.tO...c......
00000040: 00 35 00 26 cb 03                               .5.&Ë.
```

It says UDP's coming up next - so we'll decode the next part as a UDP header.

SOURCE PORT NUMBER 0407 (hex)	DESTINATION PORT NUMBER 0035 (hex)
UDP LENGTH 0026 (hex)	UDP CHECKSUM cb03 (hex)

FIGURE 4-3. **Our manual decode of the UDP header indicates the destination port number of the original packet was 0x0035.**

Here's a breakdown of the undecoded hex section based on the format shown in Figure 4-3.

Field	**Hex**	**Decimal**
UDP Source Port	0407	1031
UDP Destination Port	0035	0053

Sooooo? What do you think triggered that ICMP reply?

That's right! A DNS query!

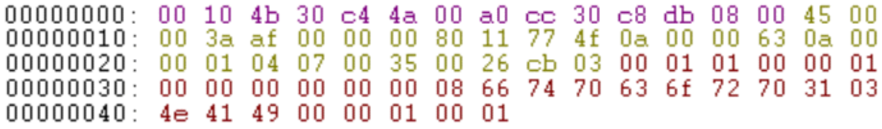
NOTE

Don't know DNS? Don't know your port numbers? Don't know ICMP!? Hey -- get the "TCP/IP Analysis and Troubleshooting" podbook! -- Laura

That illustrates how to use the specifications to complete a partially decoded packet. Now let's look at a raw packet to focus on the total packet format and decoding methods.

Understanding Raw Packet Formats

As you know, packets start off with a MAC header. If you know which network media access control method is used, then you are on your way. Consider the raw packet shown in Figure 4-4.



```
00000000: 00 10 4b 30 c4 4a 00 a0 cc 30 c8 db 08 00 45 00
00000010: 00 3a af 00 00 00 80 11 77 4f 0a 00 00 63 0a 00
00000020: 00 01 04 07 00 35 00 26 cb 03 00 01 01 00 00 01
00000030: 00 00 00 00 00 00 08 66 74 70 63 6f 72 70 31 03
00000040: 4e 41 49 00 00 01 00 01
```

FIGURE 4-4. **Ahhhhhh... Pure hex!**

Now, let’s perform a full packet decode from hex. The goal of this exercise is to identify the source and destination devices, identify the application in use, and (if applicable) find out what data is crossing the wire. We took this packet from an Ethernet network. We do not know, however, what frame type was in use on the network.

This is a great exercise, so stick with it -- if you are new to hex decoding, be patient. You will need the following resources to perform this decode:

- Knowledge of the Ethernet header structure (lots of Ethernet reference books out there... see the www.ieee.org specifications for 802.3 or “Ethernet: The Definitive Guide” by Charles E. Spurgeon - Published by O’Reilly. ISBN 1-56592-660-9.

- Knowledge of the IP header structure (RFC 792 - go to www.ietf.org).

- Knowledge of the UDP header (RFC 768 - go to www.ietf.org).

- The Port list (http://www.iana.org/numbers.htm#P).

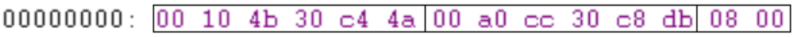
- Knowledge of DNS (RFC 1035 - go to www.ietf.org).

Decoding the MAC Header

In Figure 4-5, we can see the breakdown of the Ethernet header. We know the destination MAC address is first and the source MAC address follows (all Ethernet frames are constructed this way).

NOTE

The Preamble and Start Frame Delimiter are discarded at the card level, so the analyzer doesn’t see them. We always see Ethernet packets starting with the destination MAC address.



```
00000000: 00 10 4b 30 c4 4a 00 a0 cc 30 c8 db 08 00
```

FIGURE 4-5. **The MAC header hex dump.**

The header contains the following fields:

Destination Address 0x00-10-4b-30-c4-4a	6 bytes
Source Address 0x00-a0-cc-c0-c8-db	6 bytes
Type Field 0x08-00 (IP)	2 bytes

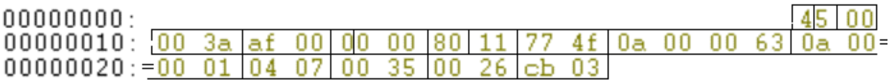
The value 0x0800 after the destination and source MAC addresses indicates (a) that this packet is an Ethernet_II address and (b) an IP header follows this MAC header.

NOTE

If the packet used the Ethernet 802.2 or SNAP format, we would have seen a length field value immediately following the source address field. The length field indicates the amount of data between the Ethernet header and the Ethernet trailer (which holds the CRC). Given that the valid Ethernet packet can contain hold be are between 46 and 1500 bytes of data, the valid length fields are between 0x002E and 0x05DC. The value we see here, 0x0800 indicates that this is not a length field -- therefore it must be a type field. --Laura

Decoding the IP and UDP Headers

The IP header is outlined as 20 bytes long. The UDP header is 8 bytes long.



```
00000000:                                                 45 00
00000010: 00 3a af 00 00 00 80 11 77 4f 0a 00 00 63 0a 00=
00000020:=00 01 04 07 00 35 00 26 cb 03
```

FIGURE 4-6. **The IP/UDP header hex dump.**

This header contains the following fields:

Version 4	4 bits
Header Length 20 (multiply by 4 bytes)	4 bits
Type of Service 00 (default)	1 byte
Total Length 0x003a = 58 decimal	2 bytes
Identification 0xaf 00	2 bytes
Flags 0	3 bits
Fragment Offset 0	13 bits
Time to Live 0x80 = 128	1 byte
Protocol Field 0x11 = 17 decimal	1 byte
Header Checksum 0x774f	2 bytes
Source IP Address 0x0a000063 (10.0.0.99)	4 bytes
Destination IP Address 0x0a000001 (10.0.0.1)	4 bytes

The protocol type field value 17 indicates that UDP is coming up next.

Source Port Number 0x0407 (1026 decimal)	2 byte
Destination Port Number 0x0035 (53 decimal)	2 bytes
UDP Length 0x0026 (38 decimal)	2 bytes
UDP Checksum 0xcb03	2 bytes

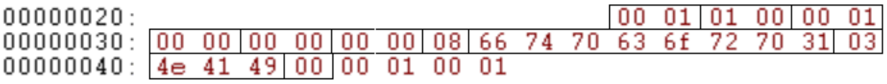
NOTE

Again, if you don't feel comfortable with the fields of the IP or UDP header, get "TCP/IP Analysis and Troubleshooting." -Laura

The destination port number value contains the value 0x0035 (which is 53 decimal), indicating that this is a DNS packet. We don't know if it is a query or reply however. We need to decode the packet further using the specifications for DNS (RFC 1035).

Decoding the Application Information

Using RFC 1035, we can decode the DNS section shown in Figure 4-7.



```
00000020:                                  00 01 01 00 00 01
00000030: 00 00 00 00 00 00 08 66 74 70 63 6f 72 70 31 03
00000040: 4e 41 49 00 00 01 00 01
```

FIGURE 4-7. **The DNS structure in hex format.**

Figure 4-8 shows the DNS packet structure.

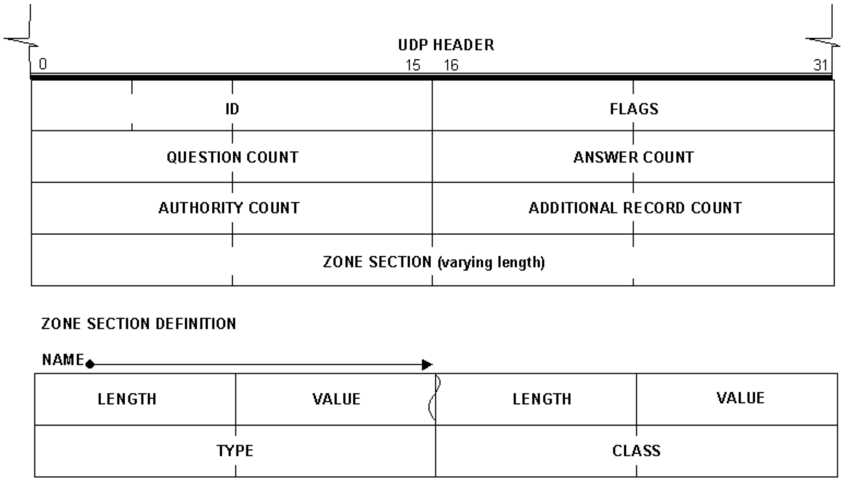


FIGURE 4-8. **The DNS packet structure per RFC 1035.**

This header contains the following fields:

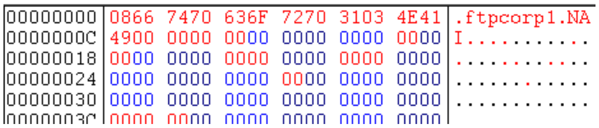
ID Number 0x0001	2 bytes
Flags 0x0100 (Command; Recursion Desired)	2 bytes
[Refer to page 26 of RFC 1035 for details on this field.]	
Question Count - 1 Question	2 bytes
Answer Count 0	2 bytes
Authority Count 0	2 bytes
Additional Information Count 0	2 bytes
Name Fields	
Length/Value 0x08/0x66-74-70-63-6f-72-70-31	
Length/Value 0x03/0x4e-41-49	
Length 0x00 (signifies end)	
Type 0x0001 - Host Address	
Class 0x0001 - Internet	

Now we know the purpose of the packet -- it is a DNS query -- the name that is being looked up is indicated in the value fields under the Name section.

Length/Value 0x08/0x66-74-70-63-6f-72-70-31 =

Length/Value 0x03/0x4e-41-49

I typically use Hex Workshop to do translations between ASCII and hex, as shown in Figure 4-9.



```
00000000 0866 7470 636F 7270 3103 4E41 .ftpcorp1.NA
0000000C 4900 0000 0000 0000 0000 0000 I...........
00000018 0000 0000 0000 0000 0000 0000 ............
00000024 0000 0000 0000 0000 0000 0000 ............
00000030 0000 0000 0000 0000 0000 0000 ............
0000003C 0000 0000 0000 0000 0000 0000
```

FIGURE 4-9. **Hex Workshop provides an ASCII to Hex translation.**

So... this packet is an attempt to get the IP address for a device called ftpcorpfs1.NAI.

Easy, eh? The key to decoding manually is to ensure you have the right references. Appendix C lists resources and links you can use to decode packets manually.

Decoding at the Itty-Bitty Level

In that last example, the DNS flags field really offered a challenge because the field did not consist of a byte or set of bytes -- it consists of numerous bit-length fields. Sometimes, you may need to decode these fields by hand. In this section, we'll focus on decoding at the bit level -- making sure you understand how to perform the translations necessary to determine how bit values can be interpreted in a packet.

Bit-Level Decode of the DNS Flags Field

Once again, we refer to RFC 1035 as the key information source for the DNS packet fields.

The entire DNS flags field is a bit-level field where individual bits or groups of bits (not bytes) delimit a field, as shown in Figure 4-10.

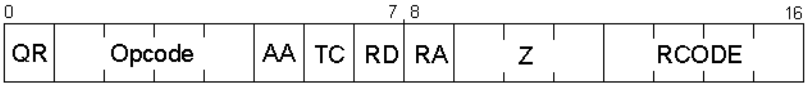


FIGURE 4-10. **The DNS header's 'Flags' field actually contains 8 separate fields.**

The DNS flags are:

QR Query/Response field. A ‘0’ indicates a query, a ‘1’ indicates a response.

OPCODE Operation Code field. Four-bit field that defines the type of query, where

0	a standard query
1	an inverse query
2	a server status request
3-15	reserved for future use

AA Authoritative Answer field (valid in responses only). A ‘1’ indicates that the answering server is the authoritative server for the domain name requested.

TC Truncation field. A ‘1’ in this field indicates that this message was truncated.

RD Recursion Desired field. Set in the query and copied into the result, if this bit is set to ‘1,’ it tells the name server to pursue the query recursively, as opposed to iteratively. Recursive querying enables the server to ask another server for information. Iterative queries tell the client to ask other servers.

Z Reserved field. Set to all ‘0s’ in queries and responses.

RCODE Response code field. This field indicates the status of the response, where

0 indicates no error condition

1 indicates a format error where the server could not interpret the query has occurred.

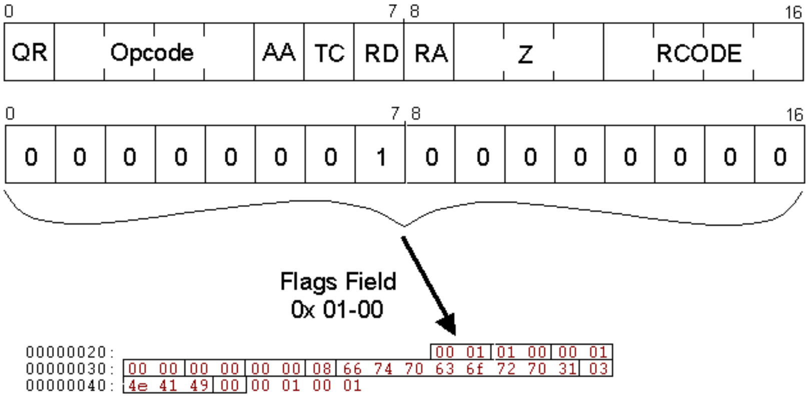
2 indicates a name server failure has occurred.

3 indicates a name error has occurred. This name error signifies that the authoritative server does not know about the domain name referenced in the query.

4 indicates that the name server does not support the query type.

5 indicates that the name server refuses to perform an operation, such as a zone transfer.

So... now you know what the values mean... let’s look at the details of the DNS header shown again in Figure 4-11.



QR	Opcode				AA	TC	RD	RA	Z			RCODE			
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

```
00000020:                                         00 01 01 00 00 01
00000030: 00 00 00 00 00 00 08 66 74 70 63 6f 72 70 31 03
00000040: 4e 41 49 00 00 01 00 01
```

FIGURE 4-11. **Now we can break down the Flags field.**

The fields of interest in this packet are, of course, the Opcode field (value ‘0’) that indicates this packet is a query, and the RD field that indicates recursion is desired.

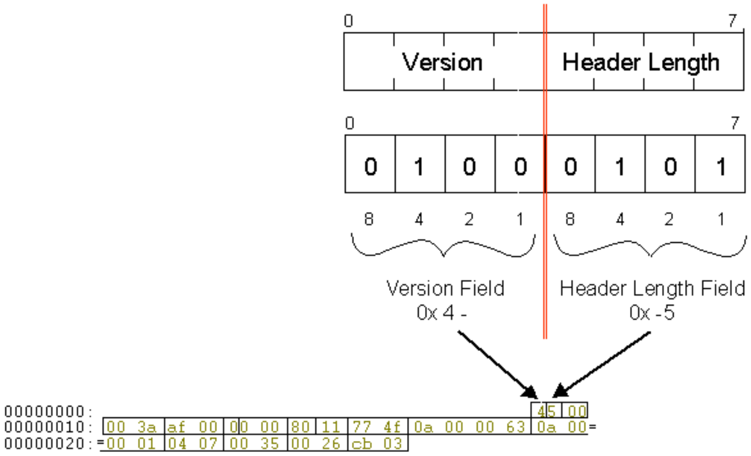
NOTE

A great reference on recursive and iterative queries is “DNS and BIND,” by Paul Albitz and Cricket Liu. Published by O’Reilly - ISBN Number 1-56592-512-2. -- Laura

Clear as mud? Eh? Let’s take a look at another bit-level decode.

Bit-Level Decode of the IP Version and Header Length Fields

IP headers start with the IP Version and Header Length fields which each consist of four bits, as shown in Figure 4-12.



```
00000000:                                                  45 00
00000010: 00 3a af 00 00 00 80 11 77 4f 0a 00 00 63 0a 00=
00000020:=00 01 04 07 00 35 00 26 cb 03
```

FIGURE 4-12. **The first byte of the IP header consists of two separate fields.**

In order to interpret 4-bit fields, we must apply the proper values to the bit positions. These values (8, 4, 2, and 1) are shown in Figure 4-12.

NOTE

On 8-bit (single byte) fields, the bit values are 128, 64, 32, 16, 8, 4, 2, and 1. --Laura

The IP Version Field

The value 0x 4- in the IP version field clearly indicates that this is IP version 4 in use. When we migrate to IPv6 someday, the value in this field will be 6, or 0110 in binary.

The Header Length Field

Pursuant to the RFC 791 which covers IP communications, the Header Length field indicates a value that should be multiplied by 4 bytes to obtain the length of the IP header. This field is required because the header can have options that extend the length in 4 byte increments. The value 0x-5 in the Header Length field indicates that the IP header is 20 bytes long (5 times 4 bytes).

NOTE

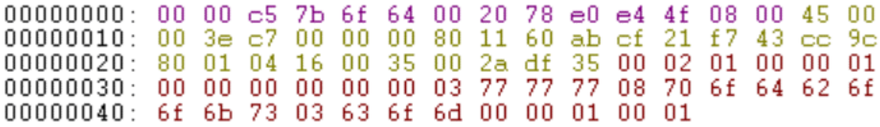
Some analyzer manufacturers may help you out by building or ‘sneaking out to you’ some extra decodes for their products. You could, in many instances, also build your own set of decodes -- check with your manufacturer to find out if they offer a Software Developer Kit (SDK) for decode construction.

Want to try one yourself? Ok... Check out the chapter test -- it contains numerous manual decode questions.

Chapter 4 Quiz

Spend a few moments reviewing this quiz. Review the chapter as needed. The answers are contained in Appendix A, “Answers to Chapter Quizzes.”

Question 4-1: Examine the packet shown in Figure 4-13. Answer the questions that follow.



```
00000000: 00 00 c5 7b 6f 64 00 20 78 e0 e4 4f 08 00 45 00
00000010: 00 3e c7 00 00 00 80 11 60 ab cf 21 f7 43 cc 9c
00000020: 80 01 04 16 00 35 00 2a df 35 00 02 01 00 00 01
00000030: 00 00 00 00 00 00 03 77 77 77 08 70 6f 64 62 6f
00000040: 6f 6b 73 03 63 6f 6d 00 00 01 00 01
```

FIGURE 4-13. **Manual decode practice.**

What is the destination MAC address?

____

What is the source MAC address?

____

Is there a type or length field in the MAC header?

____

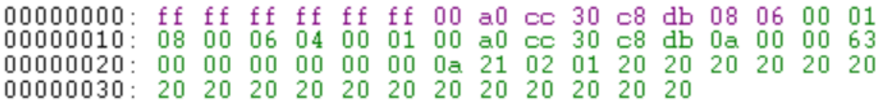
What follows the MAC header?

____

What is the purpose of this packet?

____

Question 4-2: Examine the packet shown in Figure 4-14. Answer the questions that follow.



```
00000000: ff ff ff ff ff ff 00 a0 cc 30 c8 db 08 06 00 01
00000010: 08 00 06 04 00 01 00 a0 cc 30 c8 db 0a 00 00 63
00000020: 00 00 00 00 00 00 0a 21 02 01 20 20 20 20 20 20
00000030: 20 20 20 20 20 20 20 20 20 20 20 20
```

FIGURE 4-14. **Manual decode practice.**

What is the destination MAC address?

____

What is the source MAC address?

____

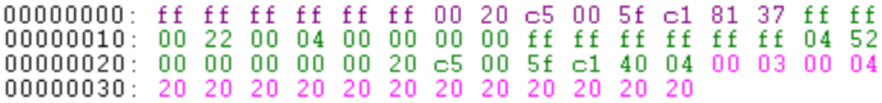
Is there a type or length field?

____

What follows the MAC header?

____

Question 4-3: Examine the packet shown in Figure 4-15. Answer the questions that follow.



```
00000000: ff ff ff ff ff ff 00 20 c5 00 5f c1 81 37 ff ff
00000010: 00 22 00 04 00 00 00 00 ff ff ff ff ff ff 04 52
00000020: 00 00 00 00 00 20 c5 00 5f c1 40 04 00 03 00 04
00000030: 20 20 20 20 20 20 20 20 20 20 20 20
```

FIGURE 4-15. **Manual decode practice.**

What is the destination MAC address?

____

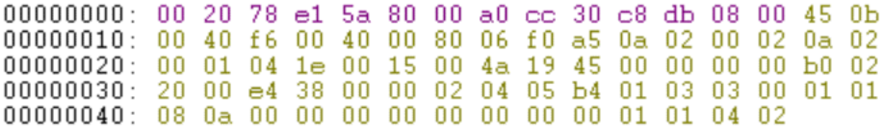
Is there a type or length field?

____

What follows the MAC header?

____

Question 4-4: Examine the packet shown in Figure 4-16. Answer the questions that follow.



```
00000000: 00 20 78 e1 5a 80 00 a0 cc 30 c8 db 08 00 45 0b
00000010: 00 40 f6 00 40 00 80 06 f0 a5 0a 02 00 02 0a 02
00000020: 00 01 04 1e 00 15 00 4a 19 45 00 00 00 00 b0 02
00000030: 20 00 e4 38 00 00 02 04 05 b4 01 03 03 00 01 01
00000040: 08 0a 00 00 00 00 00 00 00 00 01 01 04 02
```

FIGURE 4-16. **Manual decode practice.**

What is the destination MAC address?

Is there a type or length field?

What follows the MAC header?

CHAPTER 5

The Master Analyst’s Toolkit

Protocol analysis is primarily performed with a packet sniffing system, such as Sniffer or EtherPeek. There are, however, many other tools that you should have at hand when you are analyzing network communications.

These tools include, but are not limited to the following:

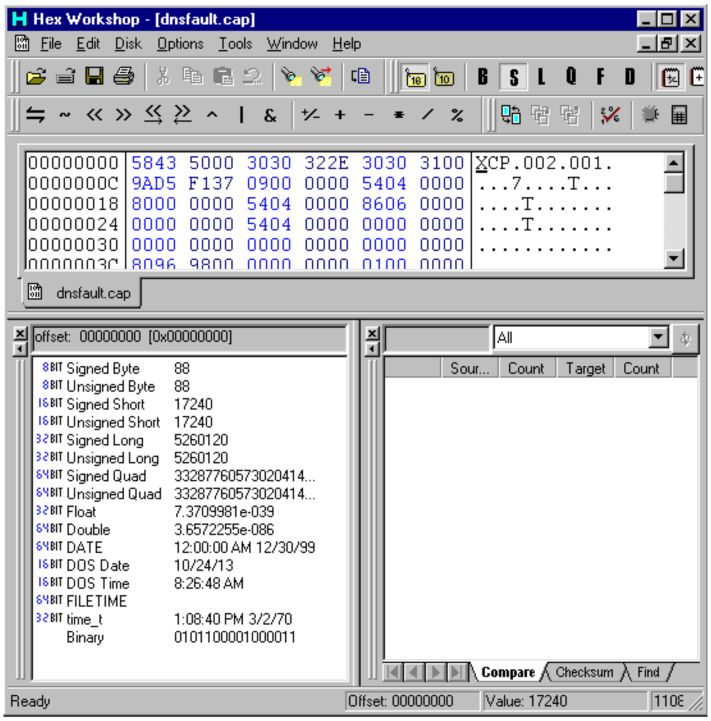
- Hex reader/editor
- Packet sanitizer
- General route tracing tool
- All purpose TCP/IP utility software
- Screen capture utility

In this chapter, we’ll examine how each of these tools can be used in network analysis.

Hex Editor

Hex editors enable you to view code in hexadecimal format. These editors enable you to spot malicious code and even do 'total trace- random offset searches,' which I'll show you later in this section. Finally, these hex editors can be used to 'sanitize' your traces and remove any confidential or sensitive materials from the trace before sending them to the outside world.

Hexadecimal numbering is a base-16 number system, that consists of 16 symbols (numbers 0 to 9 and the letters A to F). Hex is used to represent every byte (8 bits) as two consecutive hexadecimal digits which is easier than reading binary - for sure!


Hex Workshop - [dnsfault.cap]
File Edit Disk Options Tools Window Help
B S L Q F D
dnsfault.cap
All
Compare Checksum Find

```
00000000 5843 5000 3030 322E 3030 3100 XCP.002.001.
0000000C 9AD5 F137 0900 0000 5404 0000 ...7....T...
00000018 8000 0000 5404 0000 8606 0000 ....T.......
00000024 0000 0000 5404 0000 0000 0000 ....T.......
00000030 0000 0000 0000 0000 0000 0000 ............
0000003C 8096 9800 0000 0000 0100 0000
```

offset: 00000000 [0x00000000]

8BIT Signed Byte	88
8BIT Unsigned Byte	88
16BIT Signed Short	17240
16BIT Unsigned Short	17240
32BIT Signed Long	5260120
32BIT Unsigned Long	5260120
64BIT Signed Quad	33287760573020414...
64BIT Unsigned Quad	33287760573020414...
32BIT Float	7.3709981e-039
64BIT Double	3.6572255e-086
64BIT DATE	12:00:00 AM 12/30/99
16BIT DOS Date	10/24/13
16BIT DOS Time	8:26:48 AM
64BIT FILETIME	
32BIT time_t	1:08:40 PM 3/2/70
Binary	0101100001000011

	Sour...	Count	Target	Count

Ready | Offset: 00000000 | Value: 17240 | 110E

FIGURE 5-1. **Hex Workshop shows the trace file in hex format.**

I use Hex Workshop by BreakPoint Software, Inc., which is shown in Figure 5-1. With Hex Workshop you can edit, cut, copy, paste, insert, and delete hex, print customizable hex dumps, and export to RTF or HTML for publishing. Additionally you can “goto”, find, replace, compare, calculate checksums and character distributions within a sector or file.

Hex Workshop supports drag-and-drop and is integrated with the Windows Explorer so you can hex edit from your most frequently used workspaces. Hex Workshop can convert between hex, decimal, and binary and supporting arithmetic and logical operations.

The primary window that interests us, as analysts, is the top ‘hex dump’ window that includes the ASCII interpretation of the hex characters. You can get rid of the Data Inspector and Results Window that clutter up the display, if desired.

Let’s look at how you can use Hex Workshop.

Sanitizing Trace Files

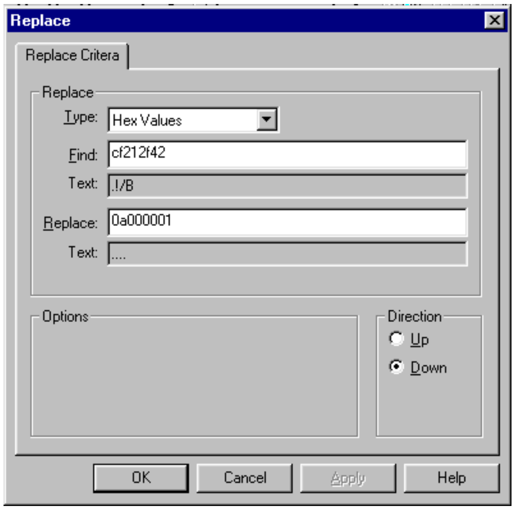
You may want to remove any sensitive or confidential information from your trace files before you share or publish them. For example, perhaps you captured an FTP login sequence and you have questions about the TCP processes you witnessed. What if a username and password crossed the wire? Whoops!

NOTE

There are now some tools specifically designed to sanitize trace files, such as PacketScrubber (recently released by Scott Haugdahl’s Net3Group - which... just before we went to press was acquired by WildPackets -- check out www.wildpackets.com). -- Laura

In Figure 5-15, I have selected to replace IP address 207.33.47.42 (which is converted from hex value cf.21.2f.42) with 10.0.0.1 (which is equivalent to hex value 0a000001).

When you sanitize your file, don't forget to look at the source IP address, the destination IP address, any passwords or user names or service names that may have crossed the wire.



Type: Hex Values

Find: cf212f42

Text: .!/B

Replace: 0a000001

Text: ....

FIGURE 5-2. **Sanitizing an IP Addresses**

You may want to replace just the network address portion throughout the trace file. That would allow a private address to be used in place of your public address. For example, if your network address is 222.33.0.0, you could replace the 222.33 value with the private network address of 192.168.0.0 throughout the file. This should handle all instances of your network address, regardless of whether that addressed was used in the IP header, DHCP data area, or other portion of a packet.

NOTE *Consider replacing your network practices with the default private addresses whenever possible. The default private addresses are defined in RFC 1918. -Laura*

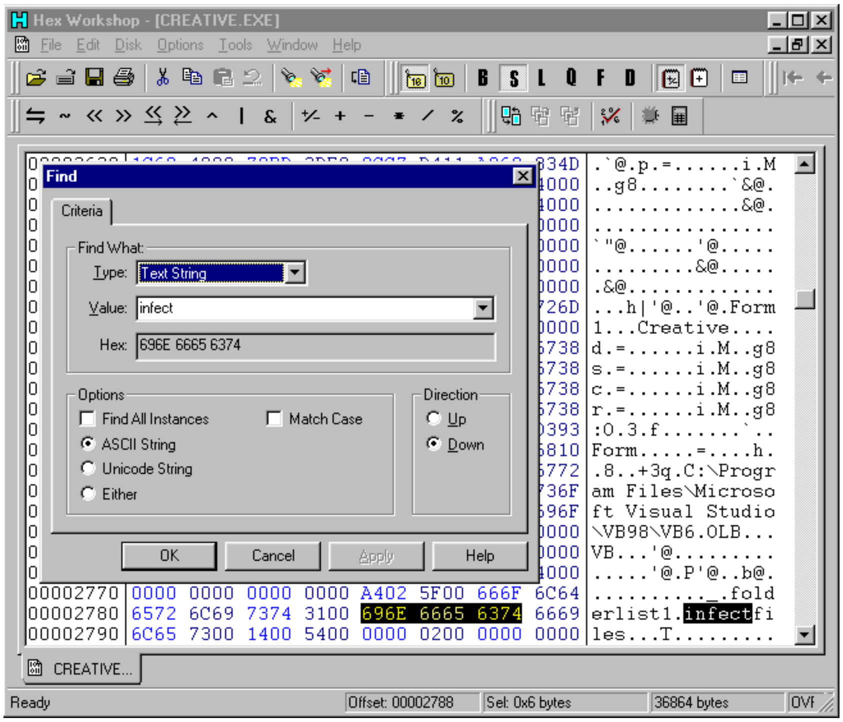
I have set up the system that enables me to clearly identify which files have been sanitized and which files have not. I save all sanitized files with the value '-x' at the end of the filename, before the extension. For example, if I open up a file named 'ftp-connect.cap,' after I sanitize file, I will save it with the name 'ftp-connect-x.cap.' This ensures that I will not mistakenly send out a file that has not been sanitized.

Searching for Text Strings

You can use a hex editor to find a text string anywhere within a trace file. For example, perhaps you'd like to check a trace file for a specific name or word.

You can also use a hex editor to open up a file, typically an executable file, to find out if there may be any malicious commands defined within the file.

One example of this is shown in Figure 5-3. Someone sent us the Creative Virus as an email attachment. We don't open up attachments from unknown sources as a rule here. I opened the file and did a search on the word 'infect" and BINGO! There it was. Now, there's no guarantee that you'll hit the word the exact way it is used in the file, so I recommend you scroll through the file in hex to see what you can see!


Hex Workshop - [CREATIVE.EXE]
File Edit Disk Options Tools Window Help
Find
Criteria
OK
Cancel
Apply
Help
CREATIVE...

Find What:
Type: Text String
Value: infect
Hex: 696E 6665 6374

Options
☐ Find All Instances
☐ Match Case
● ASCII String
○ Unicode String
○ Either

Direction
○ Up
● Down

```
00002770 0000 0000 0000 0000 A402 5F00 666F 6C64 .........._.fold
00002780 6572 6C69 7374 3100 696E 6665 6374 6669 erlist1.infectfi
00002790 6C65 7300 1400 5400 0000 0200 0000 0000 les...T.........
```

Ready | Offset: 00002788 | Sel: 0x6 bytes | 36864 bytes | OVR

FIGURE 5-3. **Bad news by mail - the Creative virus.**

NOTE

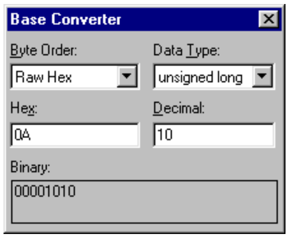
Another example of this, is the hex viewing of the WinSatan trojan performed by the IT Security Lab Group from the Carlos III University of Madrid. (See the June 2000 issue of ;login: The Magazine of Usenix & SAGE.) When the group opened the WinSatan Trojan, they found a large number of plaintext messages. Some of the messages were there are messages (not uncommon in executables). Some of the plaintext defined a strange list of active IRC servers and their respective IP addresses. The team was even able to determine what program WinSatan was built with.

It is always the good idea to scroll through the trace file to see what plaintext is visible. You will be able to see entire packets in their hex format. If there is a specific word, phrase or name you are concerned about, you can do a search for that text. For example perhaps you captured email communication from your boss. You could do a search for your boss’ name to see if it showed up at any point into the trace file.

Converting Hex to Decimal to Binary

There are several conversion utilities available for analysts -- in fact, the Windows calculator (once you put it in scientific mode) can do the conversion job nicely. I really like the conversion capability included in Hex Workshop, however, because you can get see the Hex, Decimal and Binary value of data in a single window. (Man... does that sound geeky?!)

Figure 5-4 shows the results of converting the hex value 0A to decimal and binary. You can also enter in decimal values to convert them to hex and binary.



Byte Order: Raw Hex
Data Type: unsigned long
Hex: 0A
Decimal: 10
Binary: 00001010

FIGURE 5-4. **Hex Workshop's Base Converter is simple.**

LINK

Hex Workshop is available at the retail price of $49.95 US (with volume discounts and site licenses also available). For more information on Hex Workshop, visit the Hex Workshop web site at http://www.hexworkshop.com. -- Laura

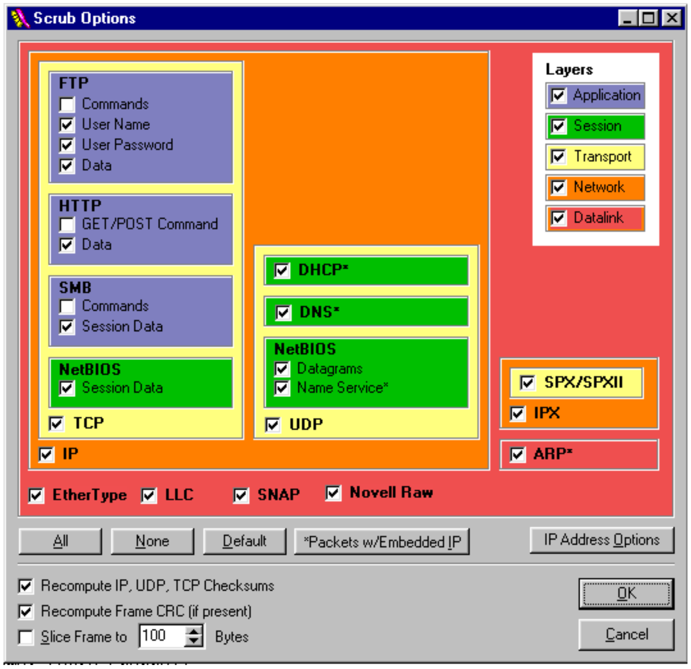
Packet Sanitizer

There are a few programs designed specifically for sanitizing trace files. Check out OptiEdit (Optimized Engineering) and PacketScrubber (by Net3Group... well, actually WildPackets -- Net3Group was acquired by WildPackets, which used to be AG Group... confused? Just go to www.wildpackets.com).

NOTE

I must admit... I'm an old fashioned girl... I have a tendency to use my hex editor to perform sanitization -- I will admit, however, that I am starting to migrate to automated tools -- of course, I do check and recheck the sanitized packets in my analyzer before sending them on. --Laura

Figure 5-5 shows PacketScrubber's Option screen.



FTP
☐ Commands
☑ User Name
☑ User Password
☑ Data

HTTP
☐ GET/POST Command
☑ Data

SMB
☐ Commands
☑ Session Data

NetBIOS
☑ Session Data

☑ TCP

☑ DHCP*

☑ DNS*

NetBIOS
☑ Datagrams
☑ Name Service*

☑ UDP

☑ IP

Layers
☑ Application
☑ Session
☑ Transport
☑ Network
☑ Datalink

☑ SPX/SPXII
☑ IPX
☑ ARP*

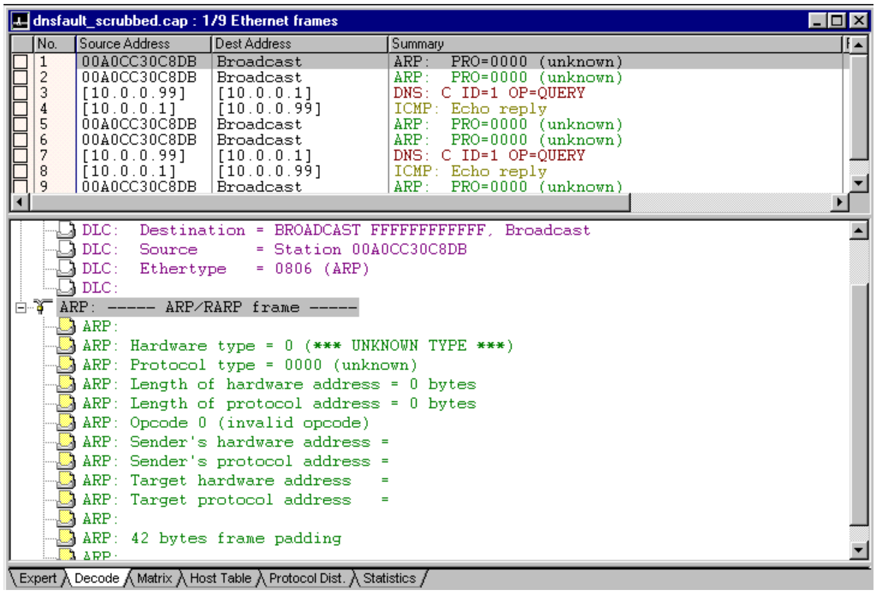
☑ EtherType ☑ LLC ☑ SNAP ☑ Novell Raw

☑ Recompute IP, UDP, TCP Checksums
☑ Recompute Frame CRC (if present)
☐ Slice Frame to 100 Bytes

FIGURE 5-5. **Packet Scrubber by Net3Group.**

As you can see, PacketScrubber provides numerous options for sanitization. You can choose to sanitize just at the IP layer, or you can move up the stack to sanitize all the way up through the application layer.

Be forewarned -- you may find that you ‘over sanitize’ a file, which makes it almost unusable, as shown in Figure 5-6.


dnsfault_scrubbed.cap : 1/9 Ethernet frames

	No.	Source Address	Dest Address	Summary
□	1	00A0CC30C8DB	Broadcast	ARP: PRO=0000 (unknown)
□	2	00A0CC30C8DB	Broadcast	ARP: PRO=0000 (unknown)
□	3	[10.0.0.99]	[10.0.0.1]	DNS: C ID=1 OP=QUERY
□	4	[10.0.0.1]	[10.0.0.99]	ICMP: Echo reply
□	5	00A0CC30C8DB	Broadcast	ARP: PRO=0000 (unknown)
□	6	00A0CC30C8DB	Broadcast	ARP: PRO=0000 (unknown)
□	7	[10.0.0.99]	[10.0.0.1]	DNS: C ID=1 OP=QUERY
□	8	[10.0.0.1]	[10.0.0.99]	ICMP: Echo reply
□	9	00A0CC30C8DB	Broadcast	ARP: PRO=0000 (unknown)

```
DLC:  Destination = BROADCAST FFFFFFFFFFFF, Broadcast
DLC:  Source      = Station 00A0CC30C8DB
DLC:  Ethertype   = 0806 (ARP)
DLC:
ARP: ----- ARP/RARP frame -----
ARP:
ARP: Hardware type = 0 (*** UNKNOWN TYPE ***)
ARP: Protocol type = 0000 (unknown)
ARP: Length of hardware address = 0 bytes
ARP: Length of protocol address = 0 bytes
ARP: Opcode 0 (invalid opcode)
ARP: Sender's hardware address =
ARP: Sender's protocol address =
ARP: Target hardware address   =
ARP: Target protocol address   =
ARP:
ARP: 42 bytes frame padding
ARP:
```

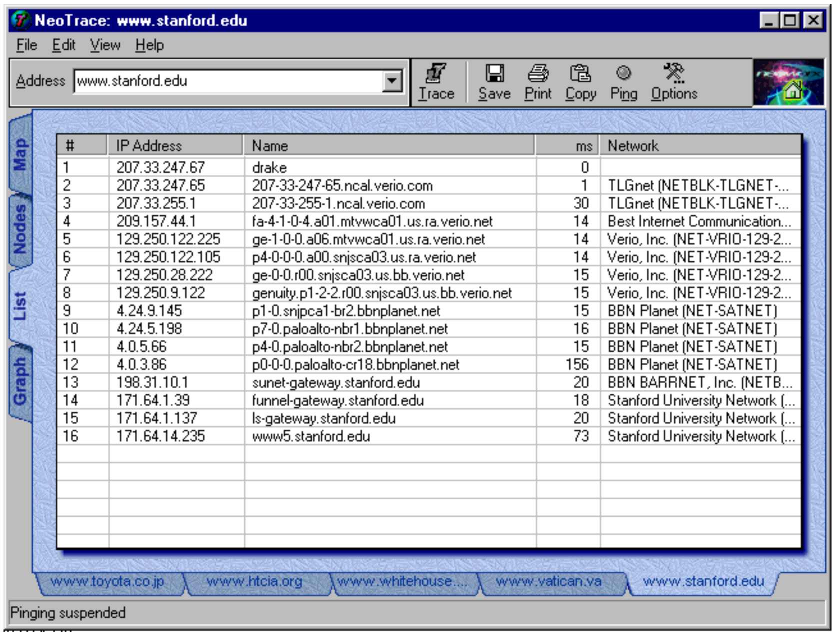
FIGURE 5-6. **A sanitized packet -- a little overdone.**

LINK

PacketScrubber was released in late 2000, not long before I finished this book -- keep an eye out for revisions and updates to the product. PacketScrubber can be purchased online at www.net3group.com or www.wildpackets.com for $169 US. - Laura

General Route Tracing Tool

There are numerous tools and web sites out there that allow you to trace a path from one device to another. I prefer NeoTrace (by NeoWorx). Lovely screens, nice beeping and just plain fun to use. Figure 5-7 shows the NeoTrace path information to reach Stanford University's website (www.stanford.edu).



Address: www.stanford.edu

#	IP Address	Name	ms	Network
1	207.33.247.67	drake	0	
2	207.33.247.65	207-33-247-65.ncal.verio.com	1	TLGnet (NETBLK-TLGNET-...
3	207.33.255.1	207-33-255-1.ncal.verio.com	30	TLGnet (NETBLK-TLGNET-...
4	209.157.44.1	fa-4-1-0-4.a01.mtvwca01.us.ra.verio.net	14	Best Internet Communication...
5	129.250.122.225	ge-1-0-0.a06.mtvwca01.us.ra.verio.net	14	Verio, Inc. (NET-VRIO-129-2...
6	129.250.122.105	p4-0-0-0.a00.snjsca03.us.ra.verio.net	14	Verio, Inc. (NET-VRIO-129-2...
7	129.250.28.222	ge-0-0.r00.snjsca03.us.bb.verio.net	15	Verio, Inc. (NET-VRIO-129-2...
8	129.250.9.122	genuity.p1-2-2.r00.snjsca03.us.bb.verio.net	15	Verio, Inc. (NET-VRIO-129-2...
9	4.24.9.145	p1-0.snjpca1-br2.bbnplanet.net	15	BBN Planet (NET-SATNET)
10	4.24.5.198	p7-0.paloalto-nbr1.bbnplanet.net	16	BBN Planet (NET-SATNET)
11	4.0.5.66	p4-0.paloalto-nbr2.bbnplanet.net	15	BBN Planet (NET-SATNET)
12	4.0.3.86	p0-0-0.paloalto-cr18.bbnplanet.net	156	BBN Planet (NET-SATNET)
13	198.31.10.1	sunet-gateway.stanford.edu	20	BBN BARRNET, Inc. (NETB...
14	171.64.1.39	funnel-gateway.stanford.edu	18	Stanford University Network (...
15	171.64.1.137	ls-gateway.stanford.edu	20	Stanford University Network (...
16	171.64.14.235	www5.stanford.edu	73	Stanford University Network (...

Pinging suspended

FIGURE 5-7. **The nodes on the way to Stanford.**

When you look at the trace of how this works, you see the ICMP Time Exceeded messages along the way and the WHOIS lookups to port 43.

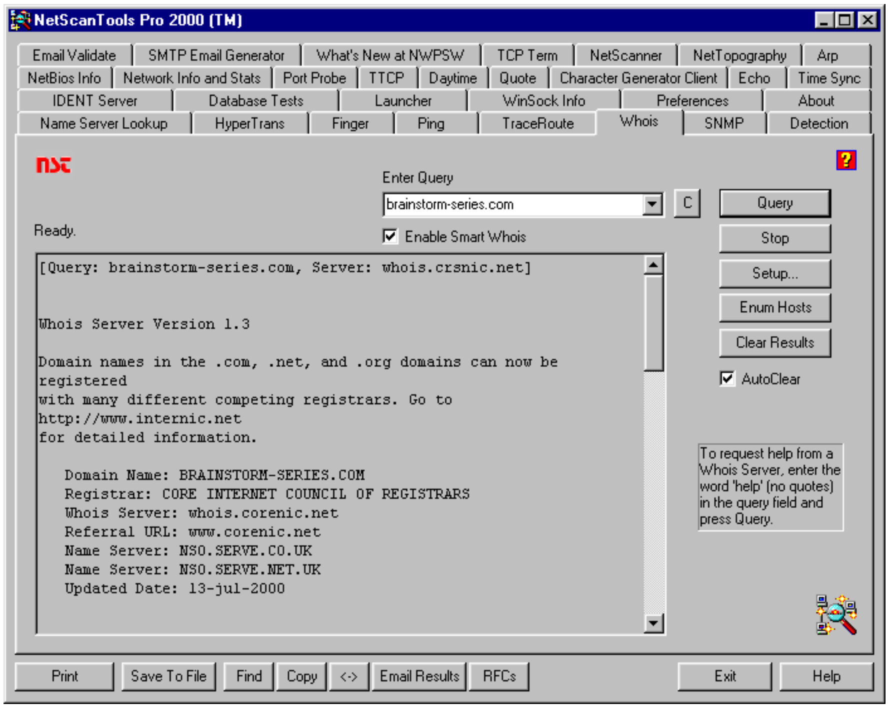
LINK

Note: NeoTrace (v2.12a) is available for Windows 95, 98, NT or 2000. The single user price is $29.95. There are 5, 10, 25 and 50 user licenses available at www.neoworx.com. At the time we were going to press, NeoTrace was working on their beta of NeoTrace Pro - which looks fantastic. -Laura

Another great route tracing tool is VisualRoute. VisualRoute is available for Windows, Sun Solaris, Linux and FreeBSD. For more information on VisualRoute, see www.visualroute.com.

All Purpose TCP/IP Utilities

To effectively troubleshoot your TCP/IP network, you need to be able to do DNS lookups, WHOIS queries, port probes, and more! You must have NetScanTools Pro by Northwest Performance Software. Figure 5-8 shows the main NetScanTools Pro window and the results from a WHOIS operation.



Enter Query: brainstorm-series.com

Ready.

☑ Enable Smart Whois

☑ AutoClear

To request help from a Whois Server, enter the word 'help' (no quotes) in the query field and press Query.

```
[Query: brainstorm-series.com, Server: whois.crsnic.net]


Whois Server Version 1.3

Domain names in the .com, .net, and .org domains can now be
registered
with many different competing registrars. Go to
http://www.internic.net
for detailed information.

   Domain Name: BRAINSTORM-SERIES.COM
   Registrar: CORE INTERNET COUNCIL OF REGISTRARS
   Whois Server: whois.corenic.net
   Referral URL: www.corenic.net
   Name Server: NS0.SERVE.CO.UK
   Name Server: NS0.SERVE.NET.UK
   Updated Date: 13-jul-2000
```

FIGURE 5-8. NetScanTools Pro is another 'must have' tool.

See "TCP/IP Analysis and Troubleshooting" from *podbooks.com* for more information on NetScanTools Pro.

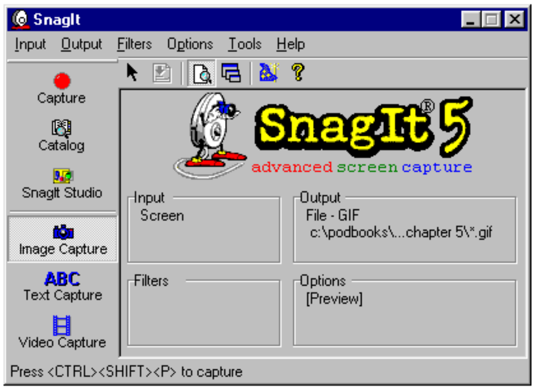
LINK

Note: NetScanTools Pro is available online at www.nwpsw.com for $150.00 US. - Laura

Screen Capture Utility

I have tried lots of screen capture utilities over the years and have standardized on SnagIt by TechSmith Corporation. I like this screen shot utility because it doesn't conflict with any software I've found and it can scroll through an complete packet, even if it's not entirely visible on the analyzer screen.

You can define various input types (and even include the cursor) and various output formats. The current version (version 5) is wonderful -- use it to create network analysis reports or capture a quick view of your analyzer statistics for the boss.



Input
Screen

Output
File - GIF
c:\podbooks\...chapter 5*.gif

Filters

Options
[Preview]

Press <CTRL><SHIFT><P> to capture

FIGURE 5-9. **SnagIt 5 will help you create awesome reports.**

LINK

You can download SnagIt for $39.95 from www.techsmith.com.

Ok... now you have the tools and the knowledge -- go out and have some technical fun! If there are some tools I should cover in the future, please let me know -- lchappell@packet-level.com. Thanks!

APPENDIX A

Answers to Quizzes

The appendix provides the answers to all the chapter quizzes.

Chapter One Answers

Question 1-1: Why would an analyzer drop packets?

If the traffic is particularly heavy or the analyzer is busy handling other processes, it may drop packets. Also... if you left your filter on and then restarted the analyzer, it will drop packets.

Question 1-2: Your broadcast alarm has been triggered again and again over the night. Since it's sending messages to Joe, you really aren't that put out, however, Joe is certainly a cranky little brat the following morning. What can you do to reduce the broadcast alarms?

Check your broadcast traffic patterns to see if the alarms indicate a problem or are just set too low for your network traffic rates.

Question 1-3: What devices need to process packets addressed to 0xFF-FF-FF-FF-FF-FF? What devices need to process packets sent to a multicast address?

All devices need to process packets addressed to 0xFF-FF-FF-FF-FF-FF - regardless of the transport or network protocol in use. Only devices that recognize the multicast address should process multicast packets. Devices in promiscuous mode should process them as well.

Question 1-4: What is the difference between the delta (interpacket) timestamp and the relative timestamp?

The delta timestamp indicates the time between each packet. The relative timestamp indicates the time from the first packet in the trace buffer (or the marked packet) and the current packet.

Question 1-5: When is the communication pattern request-reply-request-reply acceptable? Give an example of a communication that might use this pattern.

This pattern is acceptable for commands. For example, when you issue an FTP command, you receive a single reply back.

Question 1-6: What type of communication pattern does IPX burst mode use?

IPX burst mode uses the request - reply - reply - reply pattern, if it is working properly.

Question 1-7: What type of communication pattern does IP RIP use?

IP RIP uses reply - reply - reply pattern.

Question 1-8: What type of communication pattern is often seen from database queries?

Often, because of the non-linear read pattern of databases, you will see the request - reply - request - reply pattern.

Question 1-9: What is a false positive? How can you reduce false positives?

A false positive is indicated when you receive an alarm due to normal activity on the network. You can reduce some false positives by adjusting your alarm settings based on your network traffic patterns.

Question 1-10: What is a false negative? How can you reduce false negatives?

A false negative is indicated by a lack of alarms when the network is experiencing abnormal activity.

Chapter Two Answers

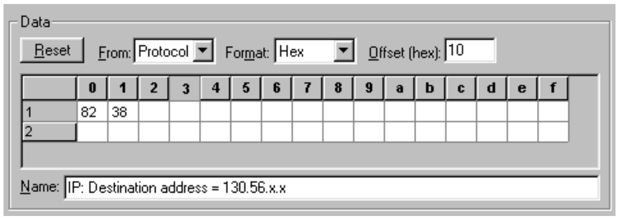
Question 2-1: What is the purpose of a boolean filter?

Boolean filters are used to build more complex and exact filters using boolean operands (OR, AND, AND NOT).

Question 2-2: What portion of the packet could you build a filter on to capture all traffic that passes through a specific router?

You need to be on a segment connected to the router and filter on the router's MAC address in the source or destination MAC header. If you filter on the router's IP address, you'll only get traffic that originates from the router (such as routing updates), but you won't get forwarded packets.

Question 2-3: Fill out the following filter window to capture all traffic to network 130.56.x.x.



From: Protocol
Format: Hex
Offset (hex): 10

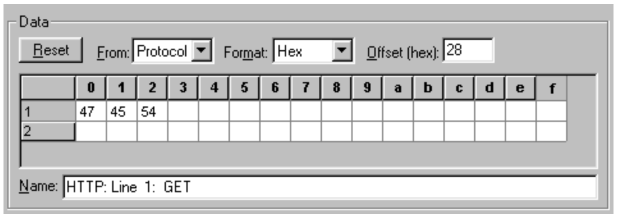
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	82	38														
2																

Name: IP: Destination address = 130.56.x.x

Question 2-4: Where do you obtain the latest UDP/TCP port number assignments?

Obtain the latest UDP/TCP port number assignments at www.iana.org (Internet Assigned Numbers Authority) under the Protocol Number Assignment section.

Question 2-5: Fill out the following window to create a filter to catch all HTTP GET traffic. Refer to the RFC if necessary.



From: Protocol Format: Hex Offset (hex): 28

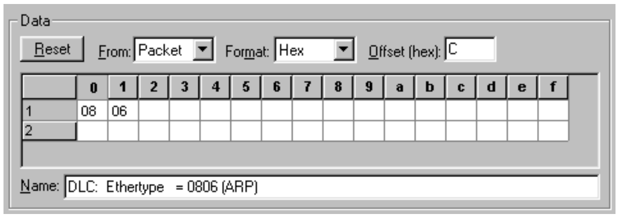
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	47	45	54													
2																

Name: HTTP: Line 1: GET

Question 2-6: How can you use filters to identify hacker attacks?

Build filters based on the ports and data sequences known to be used by hackers. Refer to Dave Dittrich’s site and www.cert.org. Also be prepared to capture excessive pings or port scans.

Question 2-7: Fill out the following window to create a filter for all ARP communications (requests and replies).

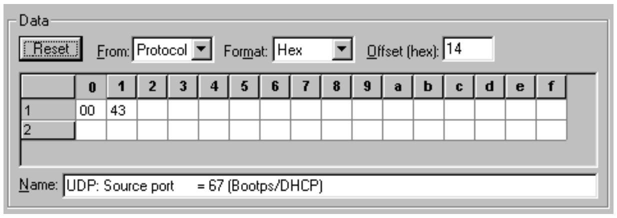


From: Packet Format: Hex Offset (hex): C

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	08	06														
2																

Name: DLC: Ethertype = 0806 (ARP)

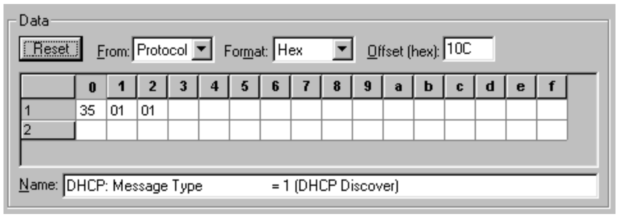
Question 2-8: Fill out the following filter window to create a filter for all DHCP Discovery packets, but no other DHCP packets.



From: Protocol Format: Hex Offset (hex): 14

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	00	43														
2																

Name: UDP: Source port = 67 (Bootps/DHCP)



From: Protocol Format: Hex Offset (hex): 10C

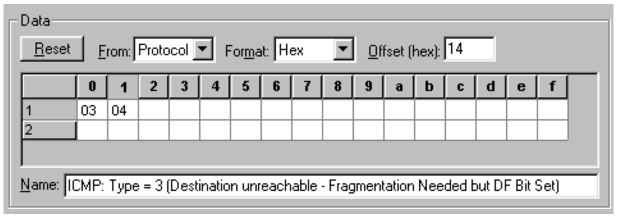
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	35	01	01													
2																

Name: DHCP: Message Type = 1 (DHCP Discover)

Question 2-9: What type of filter can be used to create a filter for all DNS communications over UDP and TCP?

To filter on all DNS communications over UDP or TCP, look for packets that contain the value 0x0035 at protocol offset 0x16. The TCP and UDP port offsets are the same.

Question 2-10: Fill out the following window to create a filter to capture all ICMP 'fragmentation required, but don't fragment bit set' messages.



From: Protocol

Format: Hex

Offset (hex): 14

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
1	03	04														
2																

Name: ICMP: Type = 3 (Destination unreachable - Fragmentation Needed but DF Bit Set)

Chapter Three Answers

Question 3-1: When is the best time to analyze an application?

The best time to analyze an application is before you deploy it.

Question 3-2: What is the most common time that applications are analyzed?

Most commonly, applications are analyzed after they are deployed and after they have made an ugly mess of the network communications. Ugh.

Question 3-3: List at least three characteristics you should document when analyzing an application.

Some of the characteristics you should document when analyzing an application include:

- Number of packets
- Supplemental files referenced
- Process time
- Failures or errors during the process
- Typical packet sizes

Question 3-4: Should you use a MAC address or network address filter when applying a filter on a test device?

If the device is local to the analyzer, you should filter on the device's MAC address to catch all traffic regardless of protocol. In the case of DHCP clients, you must only filter on the MAC address since there's no guarantee the client will get the same IP address each time it boots up. If, however the device is on another network than the analyzer, you should filter on the network address.

Question 3-5: You are analyzing an application that sends 20 MB of data on the wire during the key operations. How do you ensure that you capture all application traffic during the test?

To ensure you capture all application traffic during the test, set up a capture buffer that is greater than 20 MB

Question 3-6: How do you set up your analyzer if you know you can't fit all the packets into your buffer?

If you know you can't fit all the required packets in your trace buffer, set up the filter for packet slicing.

Question 3-7: Why should you track the start packet and stop packet count during an application analysis session?

Track the start packet and stop packet count during an application analysis session to know how many packets are involved in each process and easily find the start and stop point of the processes in the trace files.

Question 3-8: How do you identify dependent web sites that are located during an HTTP session?

Filter on DNS traffic to identify dependent websites that are located during an HTTP session. Remember, however that if the client has a web site’s IP address in a host table, they won’t send a DNS query. In that case, you might want to look for the TCP SYN packets to see who they make a connection to.

Question 3-9: Which window should you examine to find out if unencrypted data is crossing the network?

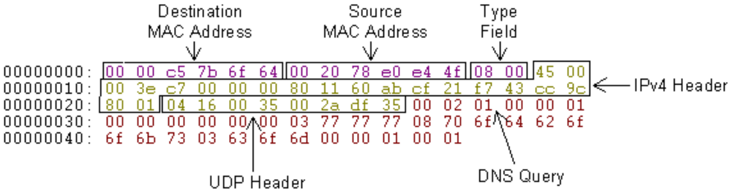
Examine the hex dump window to find out if data is crossing the network unencrypted.

Question 3-10: You’ve captured all FTP communications to and from your test station. How can you compare what you see in the trace to what is considered ‘by the spec’ on the application’s behavior?

To compare your traffic with ‘by the spec’ traffic, go get and read RFC 959.

Chapter Four Answers

Question 4-1.



```
00000000: 00 00 c5 7b 6f 64 00 20 78 e0 e4 4f 08 00 45 00
00000010: 00 3e c7 00 00 00 80 11 60 ab cf 21 f7 43 cc 9c
00000020: 80 01 04 16 00 35 00 2a df 35 00 02 01 00 00 01
00000030: 00 00 00 00 00 00 03 77 77 77 08 70 6f 64 62 6f
00000040: 6f 6b 73 03 63 6f 6d 00 00 01 00 01
```

FIGURE 4-1. **Manual decode practice.**

The destination MAC address is 0x0000c57b6f64*.*

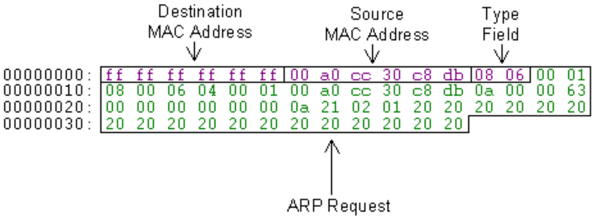
The source MAC address is 0x002078e0e44f

There a type field value of 0x0800 in the MAC header.

An IP header follows the MAC header.

This packet is a DNS query for the web site www.pod-books.com.

Question 4-2.



```
00000000: ff ff ff ff ff ff 00 a0 cc 30 c8 db 08 06 00 01
00000010: 08 00 06 04 00 01 00 a0 cc 30 c8 db 0a 00 00 63
00000020: 00 00 00 00 00 00 0a 21 02 01 20 20 20 20 20 20
00000030: 20 20 20 20 20 20 20 20 20 20 20 20
```

FIGURE 4-2. **Manual decode practice.**

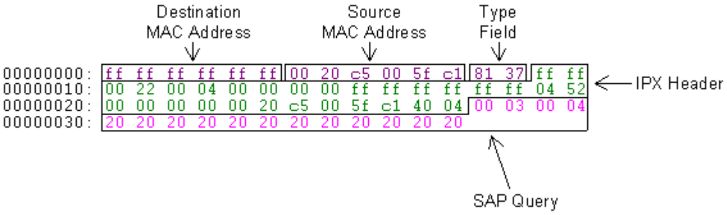
The destination MAC address is 0xFFFFFFFFFFFF (broadcast).

The source MAC address is 0x00a0cc30c8db.

The type field contains the value 0x0806 (indicating this is an ARP packet).

This packet is an ARP request to resolve the IP address 10.33.2.1.

Question 4-3.



```
00000000: ff ff ff ff ff ff 00 20 c5 00 5f c1 81 37 ff ff
00000010: 00 22 00 04 00 00 00 00 ff ff ff ff ff ff 04 52
00000020: 00 00 00 00 00 20 c5 00 5f c1 40 04 00 03 00 04
00000030: 20 20 20 20 20 20 20 20 20 20 20 20
```

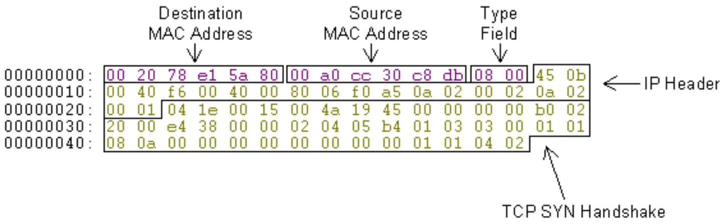
FIGURE 4-3. **Manual decode practice.**

The destination MAC address is 0xFFFFFFFFFFFF.

The type field value is 0x8137 (NetWare).

An IPX header follows the MAC header. This is a NetWare SAP packet (socket 0x452). See “Novell’s Guide to LAN/WAN Analysis: IPX/SPX” for the IPX and SAP packet breakdowns.

Question 4-4.



```
00000000: 00 20 78 e1 5a 80 00 a0 cc 30 c8 db 08 00 45 0b
00000010: 00 40 f6 00 40 00 80 06 f0 a5 0a 02 00 02 0a 02
00000020: 00 01 04 1e 00 15 00 4a 19 45 00 00 00 00 b0 02
00000030: 20 00 e4 38 00 00 02 04 05 b4 01 03 03 00 01 01
00000040: 08 0a 00 00 00 00 00 00 00 00 01 01 04 02
```

FIGURE 4-4. **Manual decode practice.**

The destination MAC address is 0x002078e15a80.

The type field contains the value 0x0800.

An IP header follows the MAC header.

If you decode this entirely, you will find that this is a TCP SYN packet.

APPENDIX B

Switched LAN Analysis

The appendix provides a slew of ideas on how to analyze a switched network. Some of the solutions are simple and require minimal investment in money and reconfiguration. Others require a bit o’ cash and some sophisticated switches.

The Problem with Switches

Switches are great for controlling traffic on the network. They break up a single shared media to allow multiple simultaneous but separate communications between devices.

When you tap into a switched network, however, you will be blinded to the general network traffic because that switch isn't going to send all the data down your port. You'll only see broadcasts and multicasts and packets that the switch doesn't have a known port for yet.

Oh, sure... this is a great way to quickly view all the broadcast/multicast traffic on your network, but it doesn't do you much good when you're troubleshooting a slow login process.

In this appendix, we'll focus on a few methods possible for tapping into and understanding the communications on a switched network. These methods include:

Hubbing Out
Port Redirection
RMON (Remote Monitoring)

Let's check out these options one at a time.

Hubbing Out

Hubbing out is a great, inexpensive solution if your switch doesn't support spanning or mirroring (covered later in this appendix). Basically, hubbing out takes advantage of the shared media design offered by a hub.

As shown in Figure B-1, the hub separates a key device from the switched network. The analyzer and the key device are connected to the hub. The analyzer can now see all traffic going to and from the key device (a server in this example). Use a crossover cable to connect the hub to the switch.

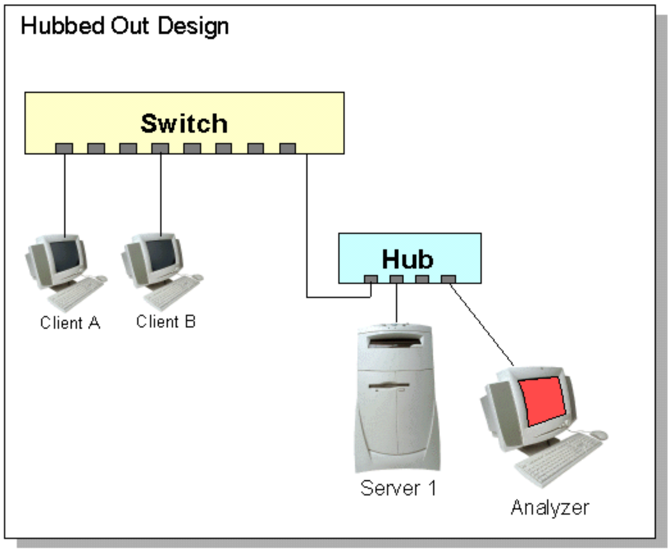


FIGURE B-1. **Hubbing out enables you to analyze switch communications when the switch doesn't support port redirection.**

Port Redirection

Quality switches now include the capability to copy data traveling down one port (or ports) to another port for the purpose of analyzing the traffic. The term ‘port span’ (which I think was coined by Cisco) is actually an acronym for ‘Switched Port ANalysis.” Other vendors, such as Nortel, use the term ‘port mirroring’ to describe this analysis capability.

NOTE

I personally hate the term ‘port redirection’ -- it implies that we are redirecting the original traffic, not just copying it. In this appendix, I will use the term ‘port spanning,’ and ‘spanned port’ throughout. - Laura

Spanning or mirroring only enables you to see good packets, however. The switches will not forward bad packets, however. That’s because decent switches act as store-and forward devices -- even when they are acting as spanning devices, the switches should check the integrity of each packet before forwarding it.

If your analyzer doesn’t support port spanning, I suggest you dump it or donate it. You cannot effectively analyze and troubleshoot a switched network that does not support port spanning. You’d be stuck doing hubbed out analysis everywhere - not a very graceful analysis option.

There are five types of spanning available with a variety of switches.

Static Spanning - Single Port
Static Spanning - Multiple Port
Remote Spanning
VLAN Spanning

Refer to your switch documentation for details on what your switch supports.

Static Spanning - Single Port

When you set up your analyzer to view data flowing through a specific port, you have configured a 'static span.' For example, if your email server is connected to module 6/port 6, you might want to set up a static span to capture all data sent down to module6/port 1, as shown in Figure B-2.

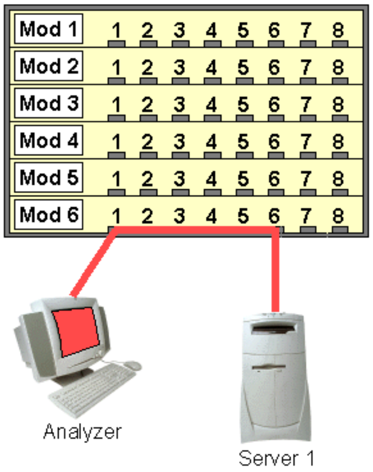


FIGURE B-2. **Single Port Scan Configuration.**

Most scans performed are static single port scans.

The following table illustrates the port scanning commands used on Cisco CAT 4000, 5000 and 6000 series switches and the 3Com 9100 switch.

Table B-2: Single Port Span Commands

Cisco CAT 4000, 5000, and 6000 Series Span Configuration Commands

```
switch (enable) set span <src_mod/src_port>
               <dest_mod/dest_port>
```

Example - redirecting traffic from module 6/port 6 to the analyzer port on module 6/port 1.

```
      set span 6/6  6/1
```

3Com 9100 Switch Span Configuration Commands

```
enable mirror to <port>
config mirror add port <port>
```

Example - Analyzing traffic from port 6; analyzer on port 1:

```
      enable mirror to port 1
      config mirror add port 6
```

Static Spanning - Multiple Ports

Some switches can be configured to copy the data from multiple ports to the analyzer port. In this case, you can gather data from numerous ports at one time. For example, perhaps you want to capture traffic that goes to three of your web servers at the same time so you can compare the traffic patterns, as shown in Figure B-3.

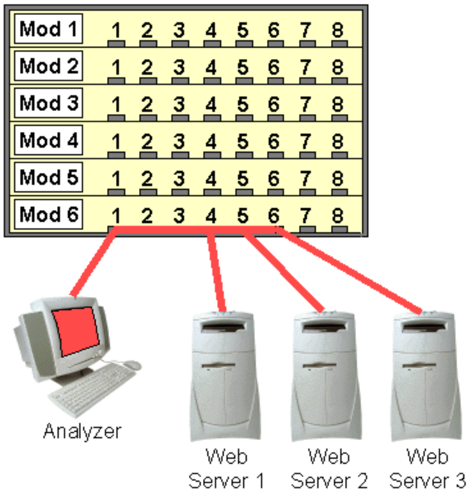


FIGURE B-3. **Multiple port span configuration.**

Both the Cisco 4000, 5000 and 6000 series switches and 3Com 9100 Switches support multiple port scanning. Refer to the table below for multiple port scan commands.

Table B-2: Multiple Port Span Commands

Cisco CAT 4000, 5000, and 6000 Series Multiple Port Span Commands

```
switch (enable) set span <src_mod/src_port>,
                <src_mod/src_port>, <src_mod/
                src_port> <dest_mod/dest_port>
```

Example - Redirecting traffic from module 6/port 4, 5, and 6 to the analyzer port on module 6/port 1.

```
      set span 6/4-6 6/1
```

There can only be one destination port number and it always follows the source port number.

3Com 9100 Switch Multiple Port Span Commands

```
config mirror add port <port>
```

Example - Analyzing traffic from port 6; analyzer on port 1:

```
      enable mirror to port 1
      config mirror add port 4
      config mirror add port 5
      config mirror add port 6
```

For more information on switch configurations, refer to the manufacturer's configuration guides.

Remote Spans

Remote spanning enables you to gather statistics from a switch that is remote to the analyzer. The packets are sent to the analyzer through a special VLAN.

Remote spanning works exactly like a regular span except that the data is sent across the wire to the analyzer on a special VLAN.

NOTE *Cisco offers a remote span capability on their CAT6000 family. They open up an RSPAN VLAN for transferring the data to the analysis system. Interestingly, you can span to multiple ports using their RSPAN (Remote Span) technology. -Laura*

VLAN Spans

Virtual LANs (VLANs) are quite popular among the switched network folks -- on some switches, you can define your multiple port spans based on the VLAN number instead of the port number.

Figure B-4 illustrates a switched VLAN environment.

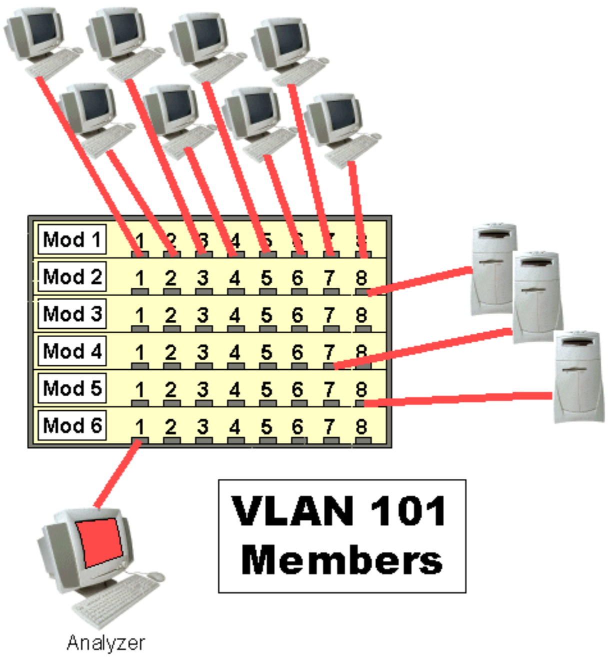


FIGURE B-4. **VLANs are common in switched networks.**

The following table illustrates how you can configure the Cisco Catalyst 4000, 5000 and 6000 switches and the 3Com 9100 switch.

Table B-2: VLAN Span Commands

Cisco CAT 4000, 5000, and 6000 Series VLAN Span Commands

```
switch (enable) set span <src_VLAN> <dest_VLAN>
```

Example - redirecting traffic from VLAN number 2 to the analyzer port on module 6/port 1.

```
      set span 2  6/1
```

Example - redirecting traffic from both VLAN 2 and VLAN 4 to the analyzer port on module 6/port 1.

```
      set span 2,4  6/1
```

3Com 9100 Switch VLAN Span Commands

```
enable mirror to <port>
config mirror add VLAN <vlan_name>
```

Example - Redirecting traffic from VLAN ‘bldg2’ to the analysis port number 1.

```
      enable mirror to port 1
      config mirror add VLAN bldg2
```

Example - Redirecting traffic from port 1 and port 3 of VLAN ‘bldg2’ to the analysis port number 1.

```
      enable mirror to port 1
      config mirror add VLAN bldg2 port 1
      config mirror add VLAN bldg2 port 3
```

Choosing a Hot Port

Most troubleshooting and general analysis tasks can be done by focusing on a single port. If you need general statistics, such as packets/second in and out of each port, check out the switch statistics information. All decent switches offer up some ‘per port’ statistics, such as bytes in/out, packets in/out and errors.

When you are looking at troubleshooting a conversation, however, you can focus on either the server, client or router hot ports. Select the most appropriate one so you don’t need to wade through irrelevant traffic.

It may seem to you that choosing the right port is a no-brainer -- but consider some subtle differences in a device’s bootup sequence when you view the communication from the server port and the client port. From the client side you can see the DHCP startup sequence. If you spanned the server port, you would miss that DHCP communication (unless the server offers DHCP services to the client, of course).

Check out Figure B-5. This network consists of clients, servers, routers, VLANs, and more. We’ll use this network topology in our example of which ports to span.

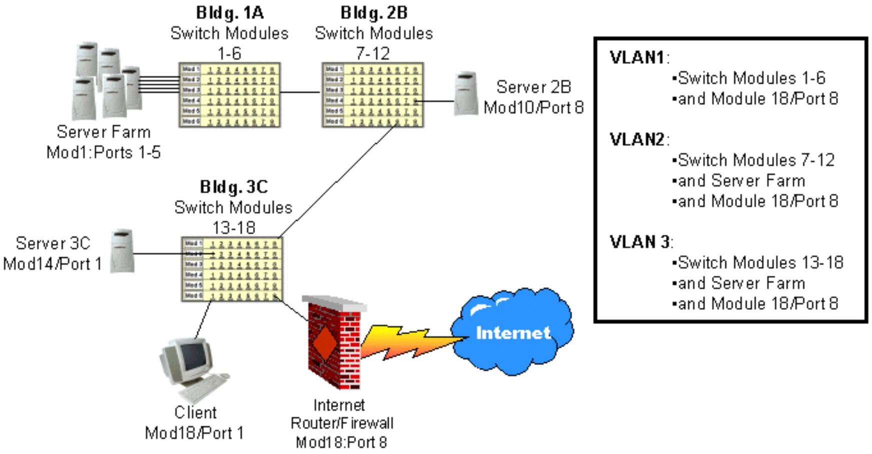


FIGURE B-5. The switched network.

Spanning a Server Port

When you are focused solely on the performance of the server, use the server's port number as the source span port. You should be able to see who is talking to the server, the type of protocols the server supports and the type of periodic broadcasts sent by the server (if any).

In Figure B-5, you might consider spanning Module 1 ports 1 through 5, Module 10/Port 8 and Module 14/Port 1 to the analyzer's port. Unless your switch supports remote spanning, you should put your analyzer near the server ports that you are going to span.

Spanning a Client Port

When you are looking specifically at the communications between one client and a server, I recommend that you span the client's port (not the server's port). You might consider filtering on the server's port because you figure you'd catch all the client/server communications -- wait though -- you may miss some of the client's traffic that is not sent to the server such as broadcasts and multicasts).

To capture a bootup sequence of the client shown in Figure B-5, you would span Module 18/Port 1 to the port where you put your analyzer - somewhere near the client's port.

Spanning Router and Firewall Ports

It's always interesting to watch the traffic coming from routers and firewalls. By spanning the router's port, you can see the traffic that has been forwarded and the periodic router updates that are sent on the network.

In the case of a firewall, it is always interesting to see what traffic comes from the firewall. Check the outbound traffic to see if any traffic that should be filtered is coming through the firewall. For example, you could look at the firewall to see if any TCP handshake (SYN) packets come through it if such traffic should be blocked.

In Figure B-5, you might consider spanning Module 18/Port 8 to the analyzer's port. Naturally, unless your switch supports remote spanning, you should put your analyzer on a port near the router/firewall's port.

RMON (Remote Monitoring)

Distributed analysis uses both SNMP (Simple Network Management Protocol) and (RMON) Remote Monitoring.

RFC 1757 defines RMON as follows:

> Remote network monitoring devices, often called monitors or probes, are instruments that exist for the purpose of managing a network. Often these remote probes are stand-alone devices and devote significant internal resources for the sole purpose of managing a network. An organization may employ many of these devices, one per network segment, to manage its internet. In addition, these devices may be used for a network management service provider to access a client network, often geographically remote.

There are four RFCs that deal with RMON: RFCs 1513, 1757, 2021 and 2074.

Distributed analyzers offer probes that attach to various switches or even, in some cases include RMON probes that are built into the switches.

NOTE *Distributed analyzers are pretty expensive... last time I looked, you had to pay up with a lung to get one of ‘em. Well, that’s just my opinion... -Laura*

The RMON MIB (Management Information Base) consists of the following groups:

The RMON MIB objects are arranged into the following groups:

- ethernet statistics
- history control
- ethernet history
- alarm
- host
- hostTopN
- matrix
- filter
- packet capture
- event

If you are really interested in how this whole RMON thing works, learn the basic SNMP operations and then focus in on the RFCs.

Overloading an Analyzer

How easy is it to overload an analyzer when you are performing switched port analysis? Easy. The reason for this is simple -- once you start spanning multiple ports to the analyzer's port, you could have a combined traffic level that is equal to or greater than the analyzer's capability for receipt.

In other words, if your analyzer is hanging off a 100 Mbps port, you could actually see 100 Mbps coming down that spanned port.

You should be aware of the possibility of overloading an analyzer and look for signs of dropped packets. Most analyzers have a way to indicate dropped packets -- pay attention to it!

APPENDIX C

Resources for Analysts

Network Analysis Articles

· 10 Tips for Creating a Network Analysis Report (Feb. 2000, pp.22-27)

· A NetWare Tool Time. (Sept. 1996, pp. 58-60)

· Cyber Crime: It Could Happen to You (April 2000, pp. 36-40)

· Ethernet Switches: Faster Than a Speeding Hub. (Aug.1996, p. 50-52)

· Fragmentation Good, Bad and Downright Ugly. (March 2000, p32-)

· Houston NetWare Conference and Exhibits. (May 1996, pp. 61-62)

· IntranetWare Over TCP/IP. (June 1997, pp. 30-39)

· Inside DHCP. BrainShare '99 Conference Daily; March 25, 1999

· IPX-IP Gateways: Find Internet Solutions at a NetWare Conference and Exhibits. (July 1996, p. 60-64)

· Is your Network Connected? Finding Out Which Network Devices Are Responding. (March 1998, pp. 16-24)

· Is Your Network DOOMed (Jan./Feb. 1996, pp. 6-18)

· ManageWise 2.0: Novell's Management Platform Comes of Age with D. Hakes (Nov./Dec.1995, pp.51-56)

· Migrating to Pure IP: With NetWare 5. (September 1998, pp. 34-36)

· Mobile IPX: Unshackle Your Computer. (Aug. 1996, p. 24-32)

· Multimedia: It's Not Just for Video Games Anymore with Dan. Hakes (Sept./Oct.1994, pp.8-24)

· Multimedia: Making It Work on the Network with D. Hakes (Nov./Dec. 1994, pp. 34-42)

· NetWare Link Services Protocol: Building a Link State Database. (Sept. 1997, pp. 38-45)

· NetWare Link Services Protocol: Interoperating With RIP and SAP. (Nov. 1997, pp. 36-39)

· NetWare Link Services Protocol: Updating the Link State Database. (Oct. 1997, pp. 34-39)

· Novell's NetWare Web Server: An Easy Way to Stake Your Claim in Cyberspace (June 1996, pp. 6-18)

· On the Road Again: Technical Education at NetWare Conferences. (June 1996, p. 51)

· On the Wire Again... (July/Aug. 1995, pp. 34-43)

· Onsite Network Analysis. (April 1999, pp. 28-31)

· Onsite Network Analysis. BrainShare '99 Conference Daily; March 21, 1999, pp. 3-9.

· Preserving WAN Bandwidth NetWare Connections. (Aug.1997, pp. 32-37)

· Pure IP Architecture. BrainShare Conference Daily, April 1998, pp. 3-7

· Service Location Protocol: Discovering Services in a Pure IP Environment. (July 1998, pp. 32-36)

· TCP/IP Troubleshooting Tools. (January 1999, pp. 20-28)

· Troubleshooting: Analyzing and Optimizing NetWare Communications. (Mar./Apr. 1994, pp. 12-16)

· Troubleshooting: Identifying and Eliminating Problems on Ethernet Networks. (Nov./Dec. 1993, pp. 32-34)

· Troubleshooting: Identifying and Eliminating Problems on Token Ring Networks. (Jan./Feb. 1994, pp. 44-48)

Recommended Books and Periodicals

· Albitz, P., Cricket, L. DNS and BIND, 3rd ed. O'Reilly & Associates, Inc.; 1998.

· Amoroso, E., Intrusion Detection. An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response. AT&T Inc.; 1999.

· Bace, R., Intrusion Detection. Macmillan Technical Publishing, 2000.

· Bradner, S., Mankin, A., IPng Internet Protocol Next Generation, Addison-Wesley Publishing Company; 1996.

· Breyer, R., Riley, S. Switched and Fast Ethernet: How It Works and How to Use It. Ziff-Davis Publishing; 1995.

· Buchanan, Jr., R.W. The Art of Testing Network Systems. John Wiley & Sons, Inc.; 1996.

· Chappell, L. Introduction to Network Analysis. Podbooks.com; 1999.

· Chappell, L. TCP/IP Analysis and Troubleshooting. Podbooks.com; 2000.

· Chappell, L. Cisco Internetwork Troubleshooting. Cisco Press and Macmillan Technical Publishing, Inc.; Publication pending.

· Chappell, L. Novell's Guide to LAN/WAN Analysis: IPX/SPX. IDG Books Worldwide, Inc.; March 1998.

· Cheswick, W.R., Bellovin, S.M., Firewalls and Internet Security. Repelling the Wily Hacker. AT&T Bell Laboratories Inc.; 1994.

· Comer, D., Stevens, D. Internetworking with TCP/IP, Vol. II; Design, Implementation, and Internals. Prentice Hall; 1991.

· Graham, B., TCP/IP Addressing. Academic Press; 1997.

· Hein, M., Griffiths, D., Switching Technology In the Local Network: From LAN to Switched LAN to Virtual LAN. International Thomson Computer Press; 1997.

· Huitema, C. Ipv6 The Internet Protocol, 2nd ed. Prentice Hall PTR; 1998.

· IEEE Technical Committee. Local and Metropolitan Area Networks. 802.1D; Media Access Control (MAC) Bridges. Institute of Electrical and Electronics Engineers, Inc., March 8, 1991.

· IEEE Technical Committee. Local and Metropolitan Area Networks. 802.1E; System Load Protocol. Institute of Electrical and Electronics Engineers, Inc., March 8, 1991.

· IEEE Technical Committee. Local and Metropolitan Area Networks. 802.3 Supplements; Layer Management (Section 5). Institute of Electrical and Electronics Engineers, Inc., March 8, 1991.

· IEEE Technical Committee. Local and Metropolitan Area Networks. 802.5 Supplements; Practice for Use of Unshielded Twisted Pair Cable (UTP) for Token Ring Data Transmission at 4 Mb/s. Institute of Electrical and Electronics Engineers, Inc., October 18, 1991.

· IEEE Technical Committee. Local and Metropolitan Area Networks. 802.5 Supplements; Recommended Practice for Dual Ring Operation with Wrapback Reconfiguration. Institute of Electrical and Electronics Engineers, Inc., September 18, 1991.

· IEEE Technical Committee. Local and Metropolitan Area Networks. 802.5; Token Ring Access Method. Institute of Electrical and Electronics Engineers, Inc., 1989.

· IEEE Technical Committee. Local and Metropolitan Area Networks. 1802.3 Conformance Test Methodology; Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. Institute of Electrical and Electronics Engineers, Inc., September 20, 1991.

· Kwok, T. ATM The New Paradigm Form Internet, Intranet & Residential Broadband Services & Applications. Prentice Hall PTR, A Simon & Schuster Company; 1998.

· McClure, S., Scambray, J., Kurtz, G., Hacking Exposed, McGraw Hill Companies; 1999.

· Northcutt, S., Network Intrusion Detection. An Analyst's Handbook. New Riders Publishing; 1999.

· Perlman, Radia. Interconnections Bridges and Routers. Addison-Wesley Publishing Company, Inc., 1992.

· Peters, Dr., Ruth, It's Never Too Soon To Discipline, St. Martin's Griffin; 1998.

· Raymond, Eric S., The New Hacker's Dictionary, The MIT Press; 1998.

· Rose, M.T. The Simple Book; Prentice-Hall Inc.; 1991

· Sacket, George C., IBM's Token-Ring Networking Handbook. McGraw-Hill, Inc.; 1993.

· Sherman, K. Data Communications, A User's Guide, 3rd ed. Prentice-Hall, Inc.; 1990.

· Sidhu, G., Andrews, R. Oppenheimer, A. Inside AppleTalk, 2nd ed. Apple Computer, Inc.; 1990.

· Spock, M.D., B., and Rothernberg, M.D., M.B. Dr. Spock's Baby and Child Care. Simon & Schuster; 1992.

· Stallings, Ph.D., W., Data And Computer Communications, 3rd ed. Macmillan Publishing Company; 1991.

· Stallings, Ph.D., W., Local Networks; Macmillan Publishing Company; 1990.

· Stevens, R. TCP/IP Illustrated, Vol. 1 The Protocols; Addison-Wesley Professional Computing Series; December 1996

Web Sites

Standards

www.ieee.org	MAC-layer specifications
www.iana.org	Protocol number assignments
www.ietf.org	RFCs and working groups

Security

www.sans.org	Security information
www.htcia.org	High tech crime group
www.cert.org	Carnegie-Mellon emergency response team
staff.washington.edu/dittrich/misc/ddos	Dave Dittrich's DDoS site
www.l0pht.com	L0pht Heavy Industries
www.insecure.org	Nmap security scanner
www.2600.com	Hacker's Quarterly

Protocol/Network Analysis

www.packet-level.com	Protocol Analysis Institute
www.optimized.com	Network analysis info
www.nai.com	Sniffer (Network Associates)/VirusScan info
www.net3group.com	Trace file conversion utility (just purchased by Wildpackets)
www.wildpackets.com	EtherPeek, TokenPeek and more

APPENDIX D

Application Analysis Form

This appendix includes the *Application Analysis Form* referred to inside Chapter 3, “Application Analysis.” You can get a softcopy of this form online at www.packet-level.com/resources/appform.pdf.

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ______________________________________ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ______________________________________ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ______________________________________ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ______________________________________ Packet Count: __________________ Process Time: ______________ Notes: ____	________

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________

Test Name: ______________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: ___________________ Process Time: _______________ Notes: ____	________
________	Process: ____ Packet Count: ___________________ Process Time: _______________ Notes: ____	________
________	Process: ____ Packet Count: ___________________ Process Time: _______________ Notes: ____	________
________	Process: ____ Packet Count: ___________________ Process Time: _______________ Notes: ____	________

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________
________	Process: ____ Packet Count: __________________ Process Time: ______________ Notes: ____	________

Test Name: ______________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: ________________ Process Time: ____________ Notes: ____	________
________	Process: ____ Packet Count: ________________ Process Time: ____________ Notes: ____	________
________	Process: ____ Packet Count: ________________ Process Time: ____________ Notes: ____	________
________	Process: ____ Packet Count: ________________ Process Time: ____________ Notes: ____	________

Test Name: ________________________________

Start Packet	Process Description	End Packet
________	Process: ____ Packet Count: ___________________ Process Time: _______________ Notes: ____	________
________	Process: ____ Packet Count: ___________________ Process Time: _______________ Notes: ____	________
________	Process: ____ Packet Count: ___________________ Process Time: _______________ Notes: ____	________
________	Process: ____ Packet Count: ___________________ Process Time: _______________ Notes: ____	________

Index