

School Daze

Analyzing a School Network

Editor's Note. This article assumes that you have a basic understanding of protocol analysis. For more information about protocol analysis, see "Basic Packet Filtering Using Your Network Analyzer," Novell Connection, January 2001, pages 40–41. You can download this article from www.ncmag.com/past. You can also visit www.podbooks.com.

Protocol analysis is, by definition, hands-on. To perform a complete analysis of a customer's network, you must visit that network. The unspoken purpose of this onsite visit is to protect the customer from anything that may harm that customer's network, data, and infrastructure. "Anything" may include a bad application, poor network design, hackers, or even a lousy network administrator.

Of course, the customer usually has a more specific reason for bringing a protocol analyst onsite. I call this reason the *primary directive*, which essentially helps clarify the customer's expectations for an onsite visit. In general, people request my expertise as a protocol analyst for one (or more) of the following reasons:

- Troubleshoot a specific problem
- Examine overall network performance (sort of a network health check)
- Optimize network communications and design
- Look for security flaws or openings on the network
- Train internal staff on analysis

Unfortunately, sometimes the customer does not seem to have a clear reason for the network analysis. For example, the customer may say, "We just need you to look at our network and give us a general idea of what's going on."

In this case, I suggest a primary directive to the customer. For example, does the customer want me to examine and baseline key communications, or does the customer want me to look for any unusual communications that may cause erratic or poor network performance? Suggesting a primary directive may help the customer determine a primary directive.

Although I try to have the customer articulate a primary directive before I go onsite, I sometimes don't discover the real primary directive until I begin my work on the customer's network. In some instances, the CEO who hired me may not be technical enough to define the problem, and the technical staff may not have wanted to list its questions for the CEO to



see in case that CEO felt the staff members were not doing their jobs properly. As you undoubtedly know, company politics can create many sticky situations.

Note. For more information about preparing for an onsite visit, see "Onsite! Case Studies" at www.packet-level.com.

SCHOOL DAYS AIN'T FUN FOR NETWORK ADMINISTRATORS

I have been asked to analyze networks for numerous organizations, but I have found that one of the most difficult types of networks to manage, troubleshoot, and protect is a school network. The equipment is often subpar with donations, hand-me-downs, and just plain junk.

School networks may include portable networks that travel the hallways from one computer lab to another. They may also include complete classrooms without a single mouse ball in sight. Not surprisingly, school networks may also be riddled with viruses and worms. (These viruses and worms may have even been developed in the school's computer labs, only to be tested on the school's network itself.)

Of course, the IS team is handed this network of aging equipment and chaotic layout and is then charged with setting up security. Not only does the IS team have to protect the network from outside hackers, the team must pay particular attention to inside hackers (uh, er—I mean the students). Over the past few years, I have been fortunate to present several Cybercrime and Advanced Networking seminars for Classroom Connect (www.classroom.com). During that time, I have heard some wicked stories of what students do on school networks.

School networks are truly unlike any other network in the world. The typical rules of network development and protection don't apply directly to school networks.

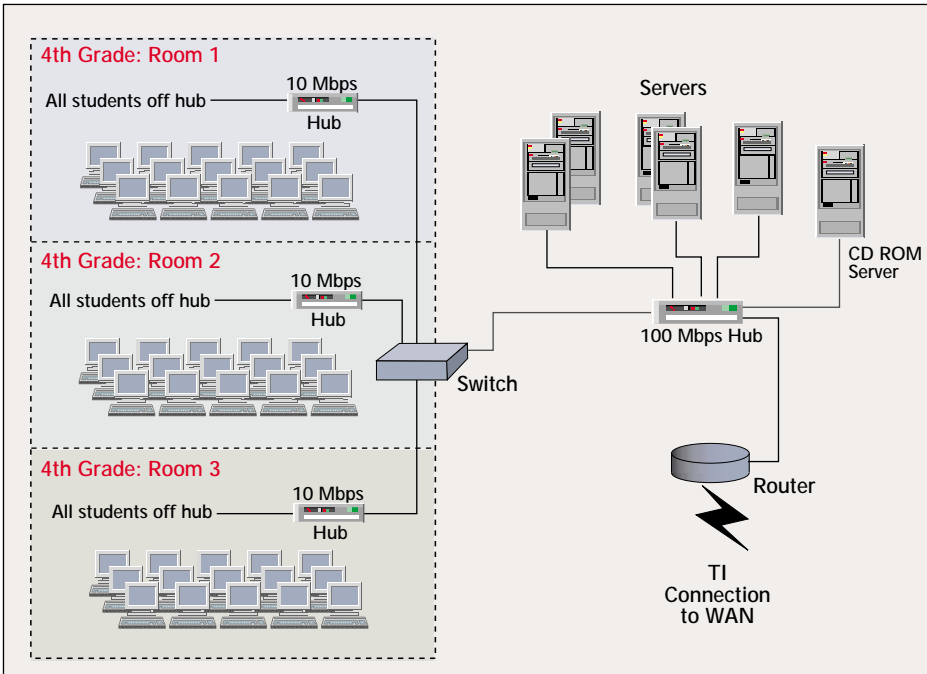


Figure 1. The school's tiered network tied together computers with 10 Mbps hubs.

The remainder of this article describes an onsite visit that illustrates the unique needs of schools. For this onsite visit, I was asked to analyze the performance of a CD-ROM-based application, which I'll call *BUZZING BEE*. Although the school district had paid for the development of this application, it would not run on the school's network. The school thought the cause of the problem was at the cable level. (To ensure confidentiality, I have omitted both the name of the school and specific details about the school.)

ONSITE WITH PROTOCOL ANALYSIS TOOLS

To perform this network analysis, I used the tools of my trade: Sniffer Pro from Network Associates (www.nai.com) and EtherPeek from Wild Packets Inc. (www.wildpackets.com). I also examined the network design at this location. As you can see in Figure 1, the school has a tiered network that ties together the classroom computers with 10 Mbps hubs.

The servers (including the CD-ROM server) are connected to a 100 Mbps

hub. T1 links connect each classroom to the district WAN (which provides Internet access centrally). The district network covers 12 separate campuses with a total of 40+ NetWare servers, 12+ Windows NT and 2000 servers, and 4,000 users. This network is in a constant state of growth.

BUZZING BEES

To accomplish the aims of my primary directive, I identified several tasks. First, I needed to look for the general traffic patterns and identify any datalink errors that might be occurring. I wanted to look for any sign that this network had lower layer errors. How was the infrastructure integrity holding up? When I checked out the general network communications, there appeared to be no datalink errors that indicated LAN driver or network interface board problems.

Next, I wanted to look specifically at the application as it ran across the network. I had been told that *BUZZING BEE* ran falteringly. When the students in the class ran the application, the application would eventually lock up.

I asked to see a class run the *BUZZING BEE* application. I placed my network analyzer off one of the empty ports in Room 1. The analyzer charts and graphs are pretty revealing—showing what happens when the application starts up. (See Figure 2.)

The first thing I noticed is that three students running the application brought the network utilization up to almost 80 percent of the 10 Mbps hub. The interesting thing here is that the school staff wanted to run the application off a CD-ROM server to avoid having CDs in each of the workstations. The staff was concerned that the students would steal the CDs (just as they had stolen the mouse balls).

Looking at the initial three-student start-up process made it evident that this application didn't work as a network-based application. It consumed all of the bandwidth available through all common paths from the CD-ROM server to the classroom that was running the application.

As you analyze the communications on your own network, you may wonder how to determine how much bandwidth an application requires. The next section provides the Bandwidth Calculation, a simple process you can use to determine this information. You can

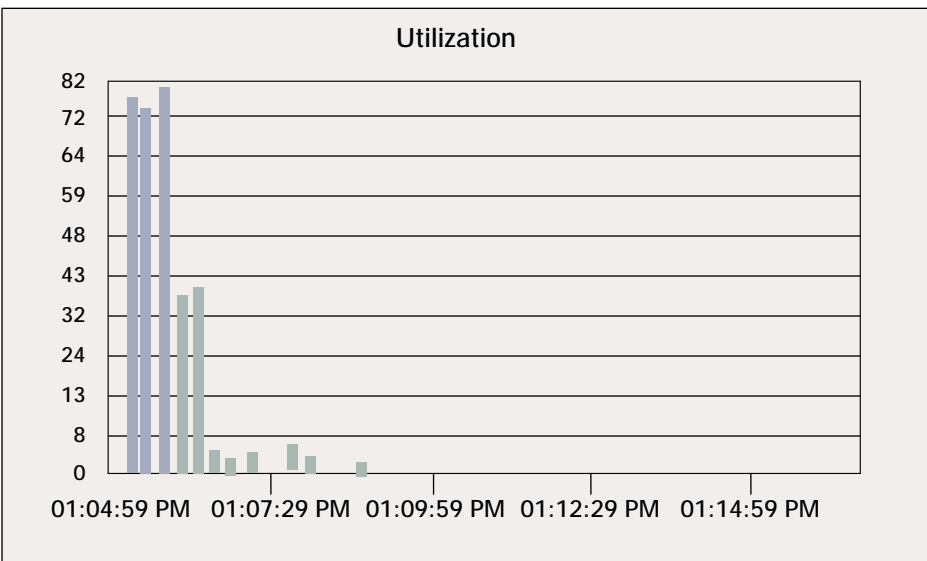


Figure 2. Three students running the *BUZZING BEE* application brought the network utilization up to almost 80 percent.

use it to determine how much bandwidth is left after you run a specific network application.

Calculating Bandwidth Use

Two separate steps are required to determine the amount of bandwidth that an application requires:

1. Determine how much bandwidth is available in bytes per second.
2. Determine the average utilization required by an application (again, in bytes per second).

First, start with the available throughput in bytes per second. After all, network analyzers give you throughput in bytes per second, not bits per second.

1. Start with the bit rate that you know (such as 100 Mbps, which is 100 million bits per second).
2. Divide this bit/second rate by 8 to determine the byte per second rate. For example, on a 100 Mbps network, you would calculate as follows:
 $100,000,000 / 8 = 12,500,000$ bytes/second.

To make sure that you understand how to perform this calculation, figure out the bytes/second rate for the following:

- a. A 10 Mbps Ethernet network
- b. A 1.544 Mbps T1 link

The answers are 1,250,000 bytes/second and 193,000 bytes/second, respectively.

After you know how many bytes per second your company's network can handle, you need to know how many bytes per second the application sends across the network. To do this, make sure you have enabled the Cumulative Bytes column of your network analyzer. Then complete the following steps:

1. Capture all of the traffic to and from a single workstation that is running the test application.
2. In the decode summary window, mark one of the packets at the beginning of a file transfer or information transfer section.
3. Follow the relative timestamp field down to one second later. What is the value in the cumulative byte field? This is the value of the bytes/second rate of the application. For example, if

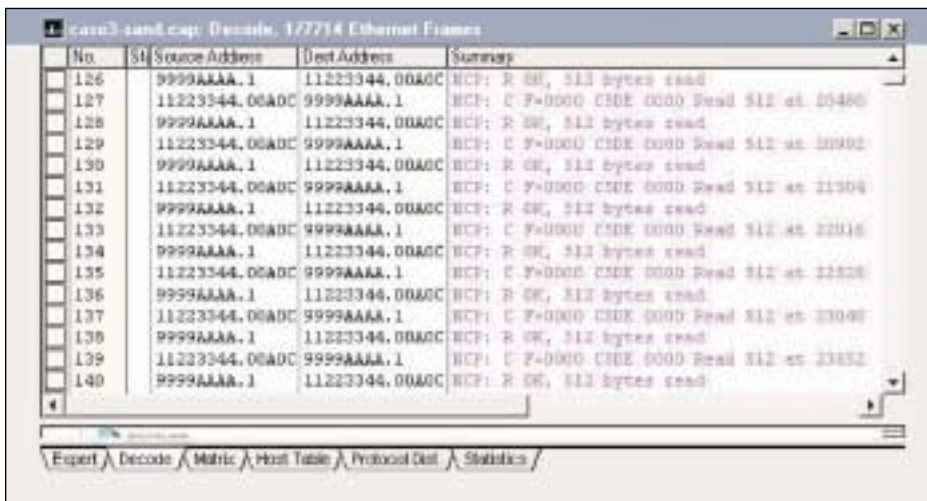


Figure 3. The BUZZING BEE application read data in small, 512-byte packets.

the Cumulative Bytes field reads 28,000, the application is sending at a rate of 28,000/bytes per second.

If possible, consider capturing more than one second of data and dividing the cumulative bytes by the number of seconds you captured. This provides a better average value than just capturing a single second's worth of data.

4. Divide the possible bytes/second rate by the application rate to find out the bandwidth usage of the application.

When I looked at a single instance of BUZZING BEE, I noted that it was transferring data at a rate of approximately 475,000 bytes per second. A 10 Mbps link can support 1,250,000 bytes/second. As you can see, this school's network couldn't support an entire classroom accessing the application at one time. The network needed to be upgraded to higher Mbps rate, or the CDs needed to be run locally. The current setup wouldn't work.

Note. The most highly utilized network I've ever seen was a school network. It was unbelievable how much garbage was crossing the wire—especially when Napster was in its prime.

Coyote-Ugly Apps

You probably remember the old "coyote-ugly" saying. It applies to the networking world as well. When an application is just too ugly to put on a network, wouldn't you rather gnaw off your arm than insert that CD to install the junk on your network? I sure hope you said "yes."

What constitutes a lousy application? Here are some examples of signs that an application is brain-dead:

- The application asks for the same information over and over and over. . . . You get the idea. (The application has obviously never heard of cache!)
- Although the network maximum transmission unit (MTU) and the protocol stack and datalink layers allow for a larger packet size, the

Visit our advertiser,
Beginfinite at
www.beginfinite.com.

Visit our advertiser, Novell eDirectory at
www.novell.com.

Visit our advertiser, Novell eDirectory at
www.novell.com.

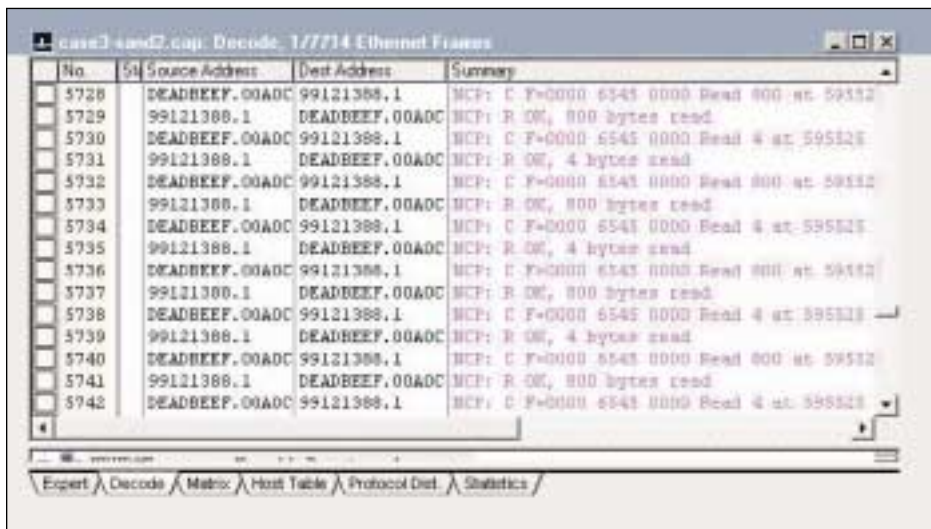


Figure 4. When the BUZZING BEE application experienced idle time, it went into a loop that saturated the cabling with unnecessary traffic.

Disgusting! This application was obviously never made to run across a network.

MORE STUFF HAPPENS

What else plagues school networks? Perhaps the question should be, what doesn't plague school networks? Security is obviously a big concern. Whenever I do an onsite visit at a school, I pay particular attention to security. I look for any types of scans or scams on the cable.

At this onsite visit, I found something else adding to the overhead on the cabling system. Check out the trace in Figure 5 to see what was happening.

You're right; that is TCP/IP traffic on the network. Hey! It looks like a little hacker-in-training! This sequence of packets is often associated with OS finger-printing, a technique defined by Ofir Arkin. (See "ICMP Usage in Scanning" or "Understanding Some of the ICMP Protocol's Hazards" at www.sys-security.com.)

Remember that any time you see a lot of Internet Control Messaging Protocol (ICMP) packets on your company's network, you need to focus on what those ICMP packets are used for—especially in a school network that is often loaded with script kiddies that will ping and port probe before they attack! (For more information about ICMP, see *TCP/IP Analysis and Troubleshooting*, which provides examples of ICMP traffic issues. Also check out *Packet Filtering: Catch the Cool Packets!* at www.packet-level.com.)

CONCLUSION

This school's network worked fine after the school ran the lousy network application locally. The school also put in place some serious ICMP filtering on the network router and upgraded the central switch to build in more filtering.

Finally, this school established some rules for network use. These rules helped decrease the number of ICMP packets on the wire, but they didn't eradicate them. The IS staff now has some really hot filters to help detect all types of scans and OS-fingerprinting sessions across the wire.

Laura Chappell performs onsite network analysis sessions for troubleshooting, optimization, and security checks. She also teaches hands-on courses on protocol analysis. For more information about how Laura Chappell can help your network run more efficiently and securely, visit www.packet-level.com. ●

application just can't read and write information in larger chunks. (The application has a very little brain.)

- The application doesn't recover from file location failures very well. The application just keeps going back over and over and over. . . . Again, the application is not the brightest bulb.

These are just a few examples of how applications can be too ugly to install. Speaking of which, Figure 3 shows the BUZZING BEE application as it ran across the network. (See Figure 3 on p. 33.) Wow, what a putrid application. The BUZZING BEE application read data in 512-byte packet sizes. Whether an application runs over TCP/IP or IPX, it should never use such pathetically small packets.

BUZZING BEE is really a lousy look-

ing application. But wait—there's more. As I analyzed the application further, I found that it never shut up. When the student finished a section of the application, the application appeared to be in an idle state, or so I thought. When I looked on the wire at the application performance, however, I saw something really upsetting. The application had a keep alive process that ran between every task that the student performed.

BUZZING BEE went into a loop whenever there was idle time. These loops literally saturated the cabling with unnecessary traffic. As you can see in Figure 4, the loop consisted of two file read operations:

- Read 800 bytes at offset 59552
- Read 4 bytes at offset 595525

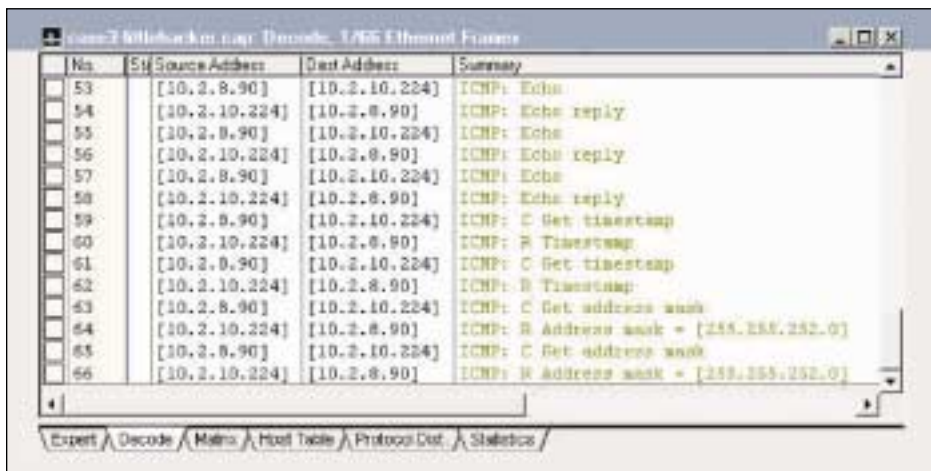


Figure 5. The network had a lot of TCP/IP traffic, indicating possible security threats.