SANDIA REPORT
SAND99-2001
Unlimited Release
Printed August 1999

Applications Analysis: Principles and Examples from Various Distributed Computer Applications at Sandia National Laboratories, New Mexico

Dennis Bateman, David Evans, Dal Jensen and Spencer Nelson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (703) 605-6000
Web site: http://www.ntis.gov/ordering.htm

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND99-2001
Unlimited Release
Printed August 1999

Applications Analysis: Principles and Examples from Various Distributed Computer Applications at Sandia National Laboratories, New Mexico

Dennis Bateman, David Evans, Dal Jensen and Spencer Nelson
Helpdesk, Video Teleconferencing, & Applications Analysis Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0807

Abstract

As information systems have become distributed over many computers within the enterprise, managing those applications has become increasingly important. This is an emerging area of work, recognized as such by many large organizations as well as many start-up companies. In this report, we present a summary of the move to distributed applications, some of the problems that came along for the ride, and some specific examples of the tools and techniques we have used to analyze distributed applications and gain some insight into the mechanics and politics of distributed computing.

This page intentionally left blank.

Contents

HISTORY 6
The Way it Was 6
The Way it is Now 6
How did we get here? 6
Everybody's doing it! 7
At Sandia, we had to invent an application analysis process. 8
FIRST STEPS 9
Application Network Maps 9
Conversation Mapping 10
Bandwidth/Throughput Measurements 11
RESPONSE TIME 12
Response Time – Breakdown Measurements 12
Response Measurements and Auto-Scripting 15
Response Time - Modeling 16
Built-In Application Performance Measures 17
Application Response Time Successes 19
Response Measurements - Deployment Decisions 19
OTHER MEASUREMENTS 20
Packet Analysis 20
Network Load 21
Routine Traffic Monitoring 22
Thread Analysis 23
SUMMARY 23

Figures

Figure 1.The network, including communications, processors, and services is the computer! 6
Figure 2. Headlines from some recent trade publications. 7
Figure 3. An example application network map. 9
Figure 5. Achieved bandwidth example. 12
Figure 6. Response time breakdown for a database query. 13
Figure 7 and Figure 8 (below). Packet bounce diagram. 14
Figure 9. Web response sampled with a script. 15
Figure 10. Web response sampled with a script. 16
Figure 11. Measured and predicted response time. 17
Figure 12. Common Log Receiver and Display System 17
Figure 13. Built-in application performance measurement display. 18
Figure 14 . Application Response times for Various File Sources 19
Figure 15. Packet analysis showing requests for missing files 20
Figure 16. Network load versus time. 21
Figure 17. Samples of MRTG traffic plots. 22
Figure 18. Thread analysis annotated sequence. 23

HISTORY

The Way it Was

Just a few years ago, desktop applications usually involved a single PC and perhaps a local file server as well. The rule of thumb was that 80% of network traffic was local to the LAN and 20% went across the backbone. Programmers, networking personnel, server managers and desktop support people rarely spoke to each other unless it was to accuse each other of being the source of a particular problem. Each group stayed in its own realm and was interested only in solving "their own" problems.

The Way it is Now

We now have a new paradigm: Distributed Computing. The Old 80/20 rule is now 20/80, with 20% local traffic and 80% going across the backbone. There are multiple computers and network services behind each mouse click.

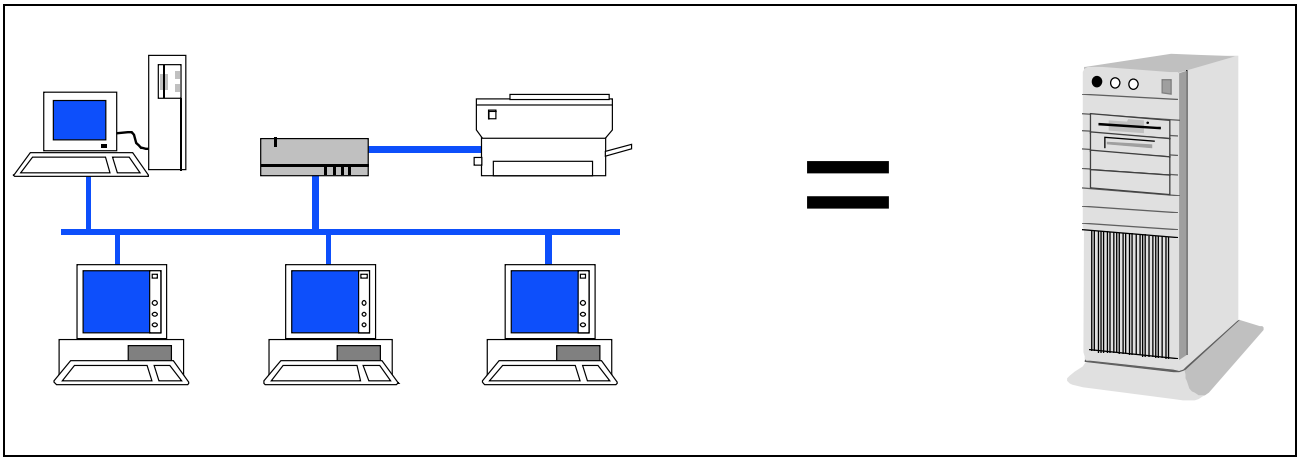


Figure 1.The network, including communications, processors, and services is the computer!

How did we get here?

In the seemingly prehistoric era of mainframes and batch jobs, systems were managed with great rigor. A small group of programmers and system managers were completely familiar with every aspect of the hardware and software. Once an application was put into production, change was minimal and tightly managed. This usually resulted in a very stable application that often couldn't produce all the information actually desired by the end user. Then in the early 80s, we got the personal computer and spreadsheets. Business users were ecstatic! Now they could slice and dice the company information by themselves, when and how they wanted. At last they were liberated from the tyranny of the IS department and the rigorous, snail-paced changes that were available on the mainframe. For a while, this worked out well. PCs really did enhance business worker's productivity. Soon, though, they found themselves wanting to share information with their co-workers. Sneakernet gave way to Ethernet, and department file servers led to corporate database servers. Systems and applications developed by Fred in HR or Susan in Finance for their own use were suddenly seen as Corporate Resources. People realized that they could be more effective if they had instant access to information from every segment of the enterprise, and client/server computing was born. Unfortunately, this was an unexpected child, and Fred and Susan really weren't up to caring for it. And so, distributed computing made its way back into the IS

Department. But the IS Department was going through its own changes. Staffed largely by people with no formal IT training, it was often divided into groups of specialists like *people-who write-programs* and *people-who-manage-servers* and *people-who-install-networks*. Most of them gravitated to the job during the PC revolution and are unfamiliar with the rigorous environment of their predecessors. They are running hard to keep up with the rapid changes occurring in their own area. Largely missing is someone whose sphere of concern *and* sphere of understanding encompasses all seven layers of the OSI model. Someone who can take the 10,000 foot view but understands the 10 foot view and can manage all aspects of an application.

Everybody's doing it!

We have discovered that many large organizations are realizing that application management is important. This is a new field, and the term 'Application Management' means different things to different people. The old saying, "When all you've got is a hammer, everything looks like a nail." comes to mind.

Large companies, trade publications and management consulting organizations all support the need for a standard approach to management of distributed applications. This area was a major theme of the Summit '98 Enterprise Management conference in August 1998 (http://www.summitonline.com). The detailed analysis of how an application works and tools to aid in this analysis is just one portion of the whole applications management topic.

Figure 2. Headlines from some recent trade publications.

At Sandia, we had to invent an application analysis process.

As distributed applications were deployed at Sandia, many application problems were initially directed to the network communications department for resolution. Usually, the problem's origin was discovered in the program itself or the configuration/availability of some network service or the method of deployment.

We realized that there was an expanded area of work. We needed to monitor and assure the total health of an application, rather than just the network infrastructure. This would allow us to be aware of potential problems, assist in resolution when problems occur, and plan for successful deployments.

The "Manufacturing" network with primary workers in one large building was chosen for an initial effort to develop procedures. The workers there use applications that access databases, work instructions, drawings, etc. that are physically located in many locations within the Sandia campus. This was a complex network with a requirement for high reliability and availability that had a history of problems. We formed a successful partnership with some funding from the Manufacturing organization and some funding from utility services. The core team met once a week to prioritize work and exchange information. Many of the techniques that were developed in that project have now been applied in other areas.

Over about a year of work, we have developed some general principles that try to answer the question, **Why are we doing application analysis?** The table below shows some of the reasons and where the information or analysis will be used in support of an application.

Table 1. Some reasons for application analysis.

Major area of benefit →	Resource Planning	Performance Improvement	Trouble-shooting
Provide basic information needed by the help desk and trouble shooting personnel.			✓
Identify network services used by the application and verify that the services are operated at compatible time schedules and response levels.	✓		
Assess the network communications load of the application to understand the prerequisites of deployment. eg, should executables reside on workstation or a server?	✓	✓	
Measure response times to be in a position to detect changes, to diagnose problems, and to let users know what to expect.		✓	
To allow for incorporation of built-in measures of key application responses and error conditions.		✓	

Following are some examples of the results of analysis work related to applications at Sandia New Mexico. Maps, operational diagrams, computer-to-computer conversation diagrams, and so on are constructed using information from many key individuals and the data from multiple monitoring tools.

FIRST STEPS

Application Network Maps

We discovered that it was very rare to find an individual who could give us a list of the specific computers and services involved in most corporate multi-computer applications. That deficiency made it slow and difficult to trouble-shoot a user's problem. It also made it difficult to understand what elements might be subject to improvement. A process was developed to create application network maps, identifying all network components of an application.

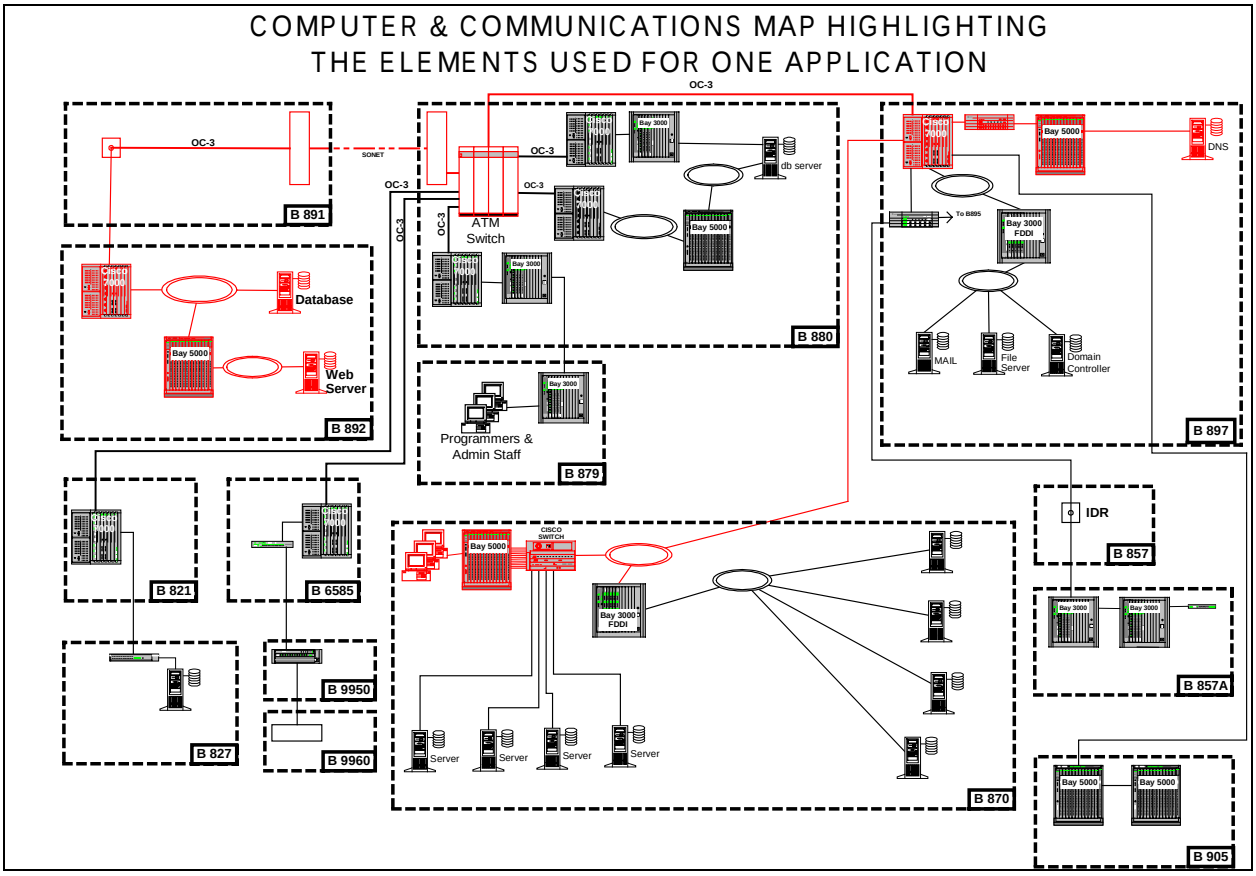


Figure 3. An example application network map.

Figure 3 is a network map showing most of the infrastructure and resources required for several distributed applications used by one division at Sandia. The highlighted elements are for one sample application from a particular client computer point of view. We find that many key

individuals involved with a distributed application are unaware of the total number of computers and transmission devices involved in the operation of their application. They all welcome a collected view of the parts as shown in a picture such as the one above.

In building the network map, the application developers and operators are interviewed and their documentation reviewed. The picture is then confirmed in various ways. Installation and machine registration documentation is referenced. Network management system discoveries are reviewed and compared to documentation. Packet captures are collected from the network, and conversation maps are built from these trace files. During the packet capture, an initial view of errors as viewed from network monitoring points can be seen. A statement about the basic health of the network from a data transmission point-of-view can also be made at this time.

Conversation Mapping

A capture from a "sniffer" network analyzer can be loaded into other application analysis tools. The packet traffic can be manipulated to display various views of the information. The following figure (figure 4) shows conversations and data quantity of each conversation while the client of interest, in the bottom center of the diagram, is being monitored. The numbers along the interconnecting lines represent the total number of bytes of data transferred between the computers. This particular conversation map shows us that the total traffic between the device labeled "File Server" and the "Client Workstation" was much greater than the traffic between the "Development Database Server" and the "Client Workstation". That type of information can help in determining where server or data communication bottlenecks might appear. It can help in predicting if there will be bottlenecks at a time when the application will be run by many users during the same time period.

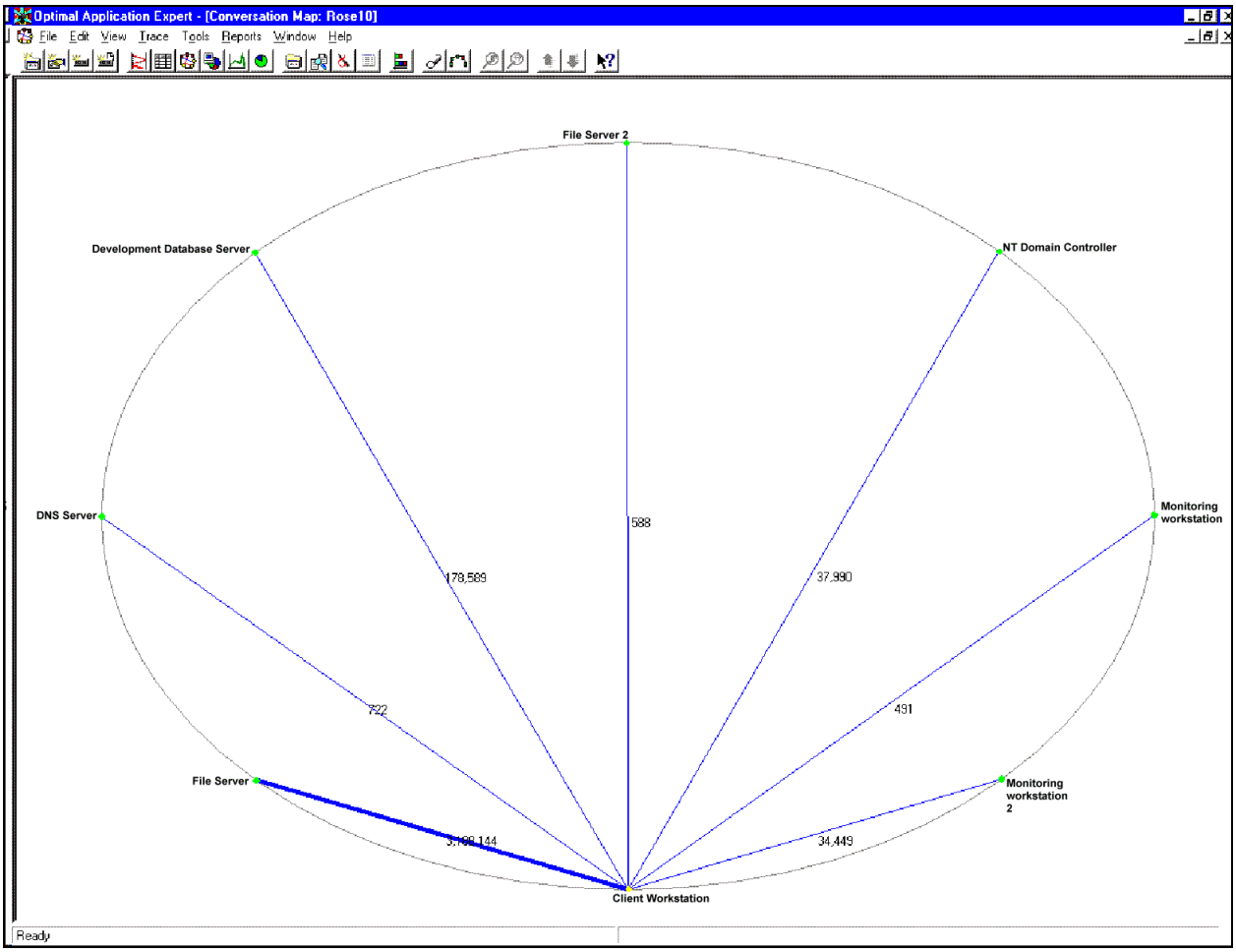
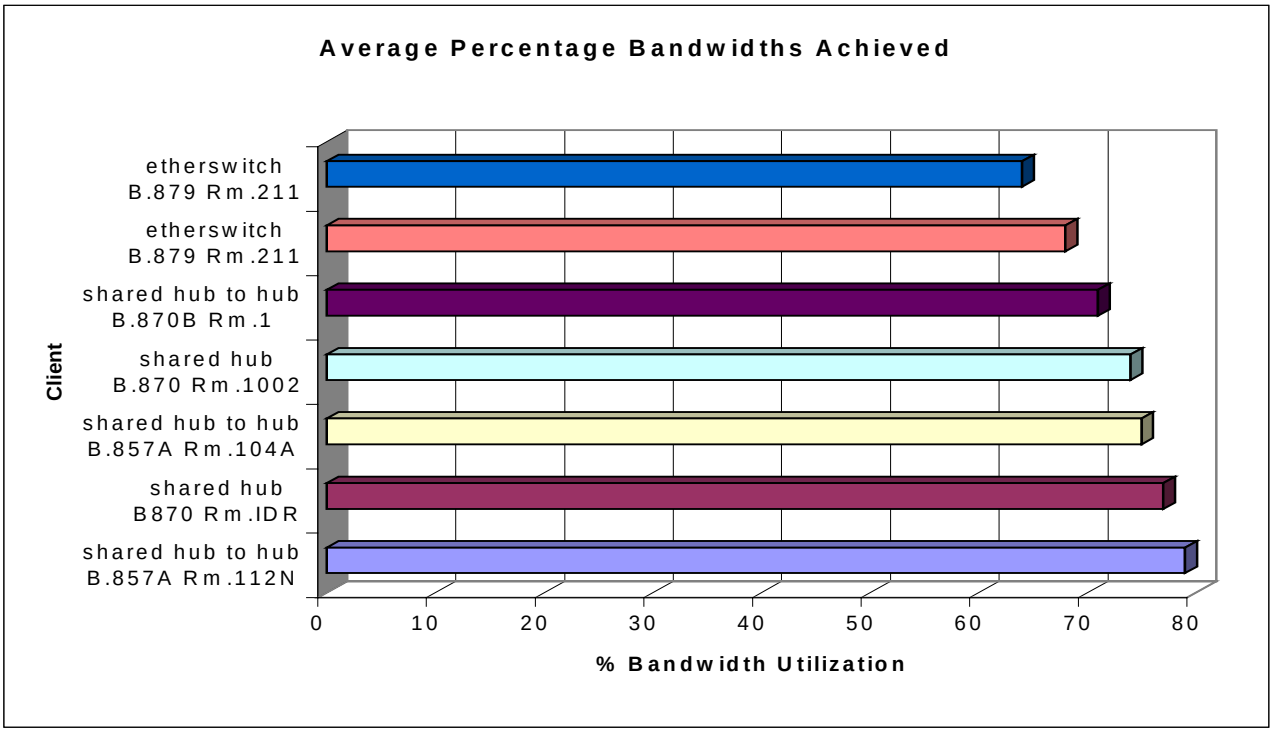


Figure 4. A conversation map.

Bandwidth/Throughput Measurements

After a network map and equipment inventory has been compiled, a TCP/IP throughput test identifies whether or not expected bandwidths to clients and servers can be achieved. Various methods can be used to accomplish this,including low-level tests like TTCP, observation using a network probe like the Sniffer, etc. When network interface cards, the attached communications equipment, and associated software are configured correctly, then bandwidths near 10 Mbps should be achievable on a standard Ethernet. Much higher bandwidths can be achieved during TTCP tests for 100Mbps paths; 50 Mbps is not unreasonable. If measured bandwidths are very much lower than expected, then some trouble shooting will be necessary to determine which devices are limiting throughput. Figure 5 shows bandwidths achieved for various devices within a data communications chain.

Figure 5. Achieved bandwidth example.



RESPONSE TIME

Response Time – Breakdown Measurements

It is important to understand where time is spent in order to identify the most likely areas of response improvement. In addition, we also found it is very helpful to tell users what response time to expect when they click the mouse. Then, they will know when to complain. In the following figure, the response time breakdown for an Oracle manufacturing query shows the response time seen by the user is due mostly to client processing. (It possibly indicates a slow client computer.)

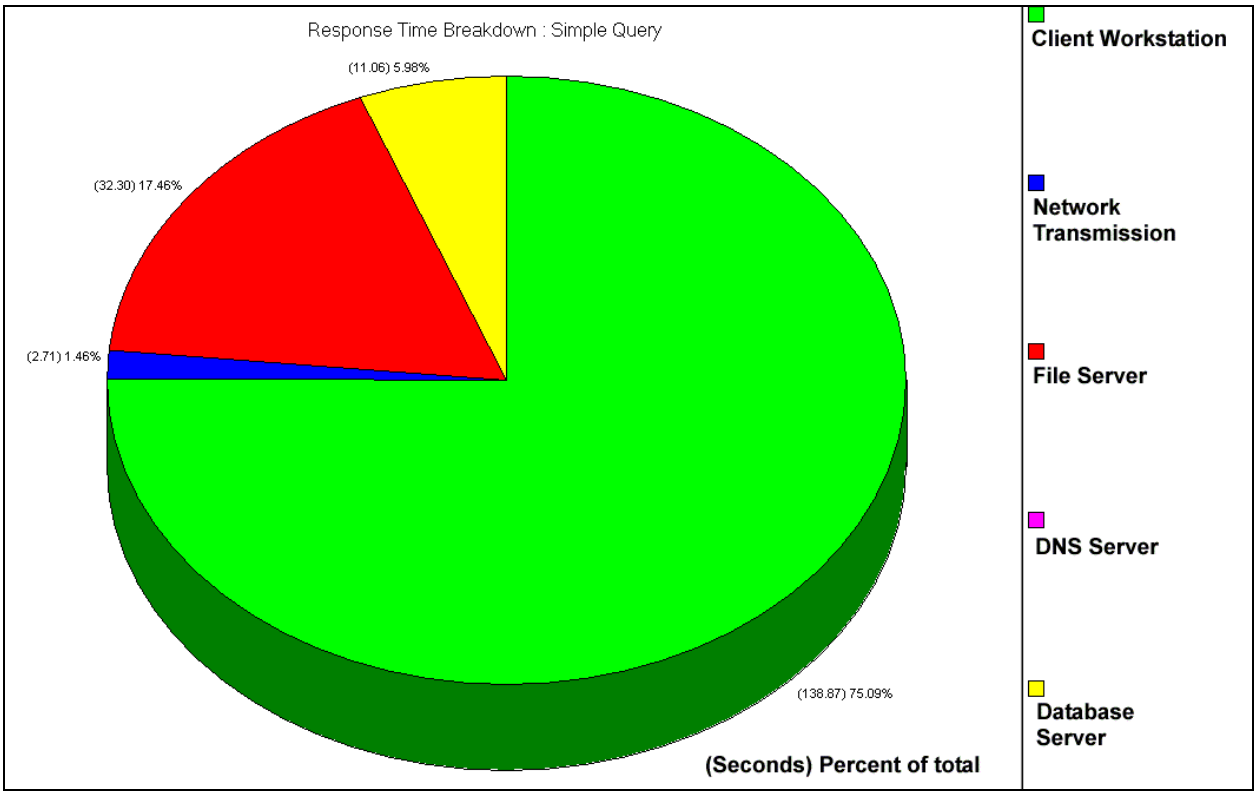


Figure 6. Response time breakdown for a database query.

For the purpose of improving an application's response, it is necessary to understand what each resource is contributing to the total response time. One way to depict the contributions of each item is to use a packet “bounce” diagram, relating traffic to specific events within the application. This gives developers and maintainers a clear idea of where changes in the application can produce changes in network traffic and response time. Supposed communications sequences can be compared to actual, captured sequences.

Figure 7 and Figure 8 (below). Packet bounce diagram.

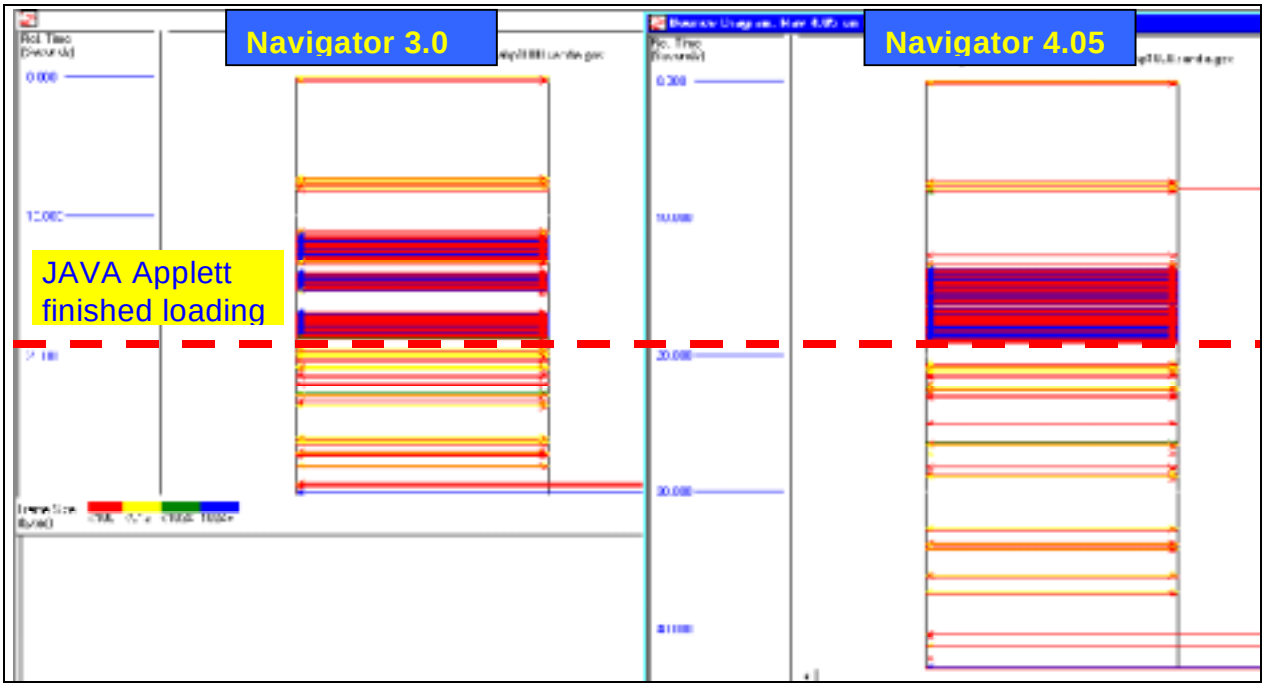


Figure 8 is a comparison of the variation in response time of a JAVA application executed by two versions of the same browser. Here we see that the JAVA applet loads in the same time, but the newer version of Netscape Navigator takes longer to execute it.

Response Measurements and Auto-Scripting

For another application, Product Data Management (PDM), response time sampling via synthetic transactions provides a metric for analysis. A script file, running on one computer, simulates a user doing three common functions: log-on, entering information and searching for a drawing set, and retrieving a viewing a single drawing. The script runs every fifteen minutes during the workday and the data is displayed on a graph (Fig. 9). The effect of changes to the systems supporting this application can then be seen from the "user" point of view. Note the more predictable response in the second graph after tuning the server (Fig. 10).

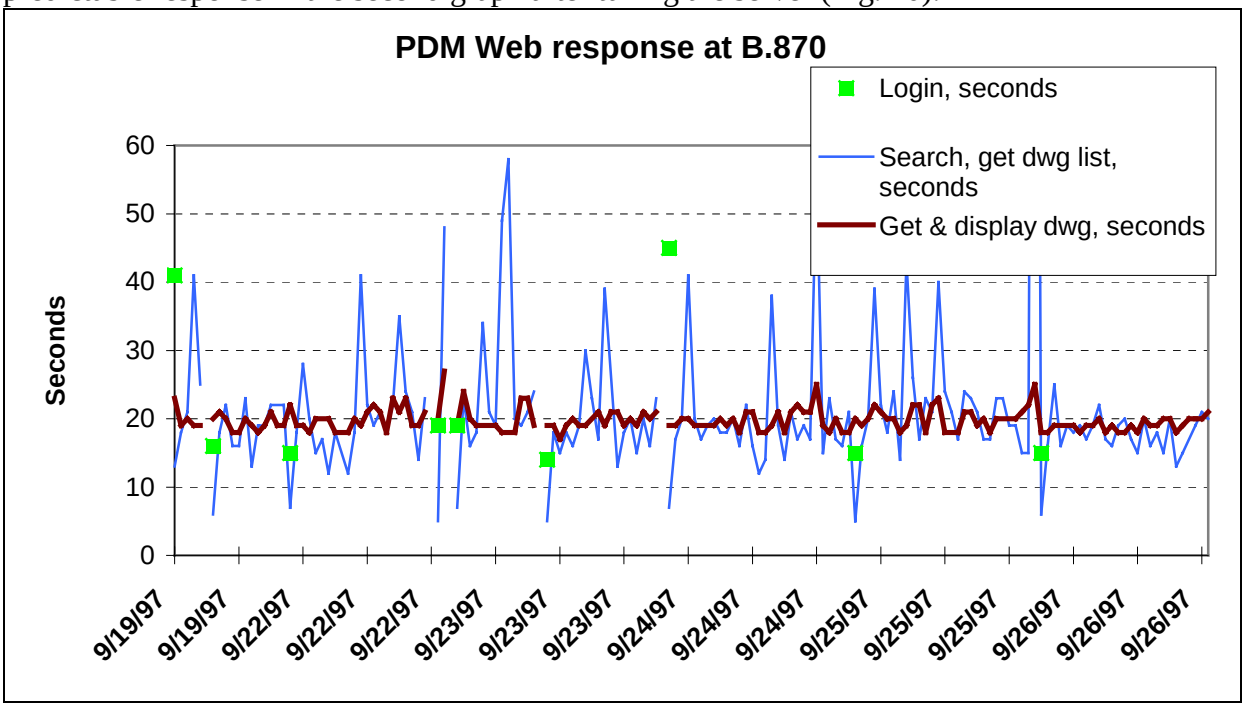


Figure 9. Web response sampled with a script.

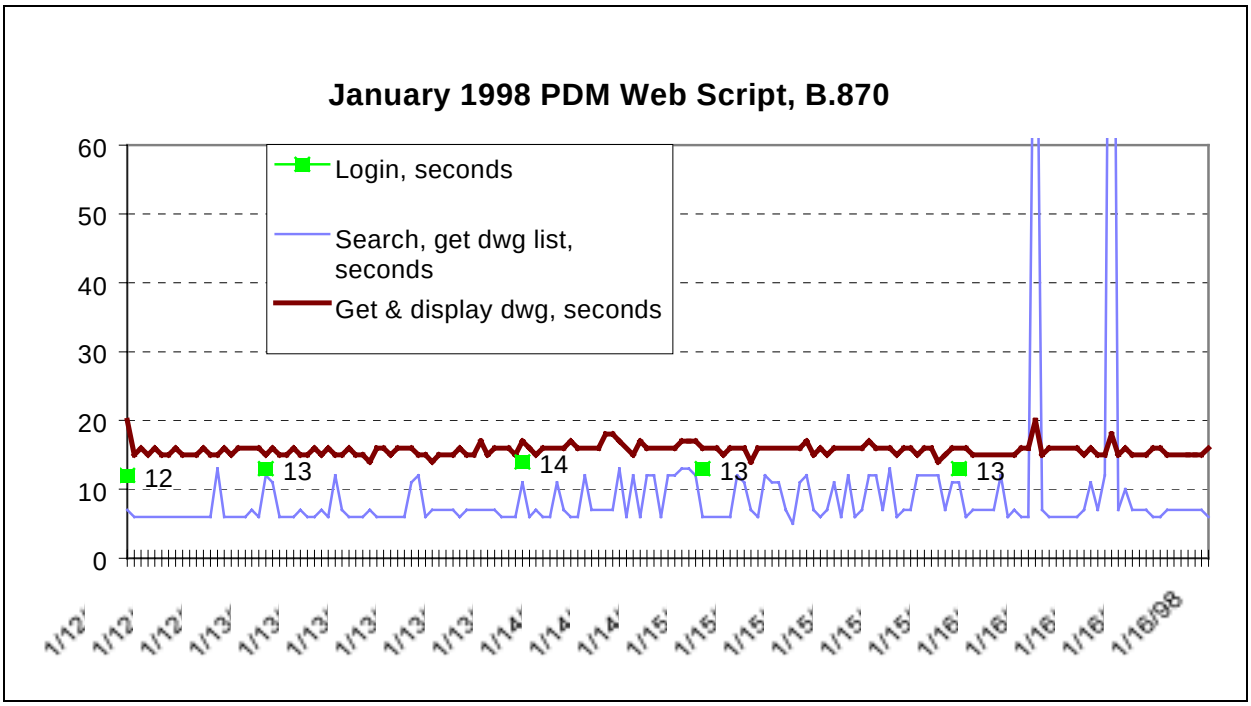


Figure 10. Web response sampled with a script.

Response Time - Modeling

Some simple programs can predict how response time will change if certain network parameters change. This is useful for planning new deployments and setting customer expectations. Figure 11 shows the application start-up time as measured on an Ethernet LAN (top), as predicted over a 28.8kbps modem connection (center), and as actually measured over a modem connection (bottom).

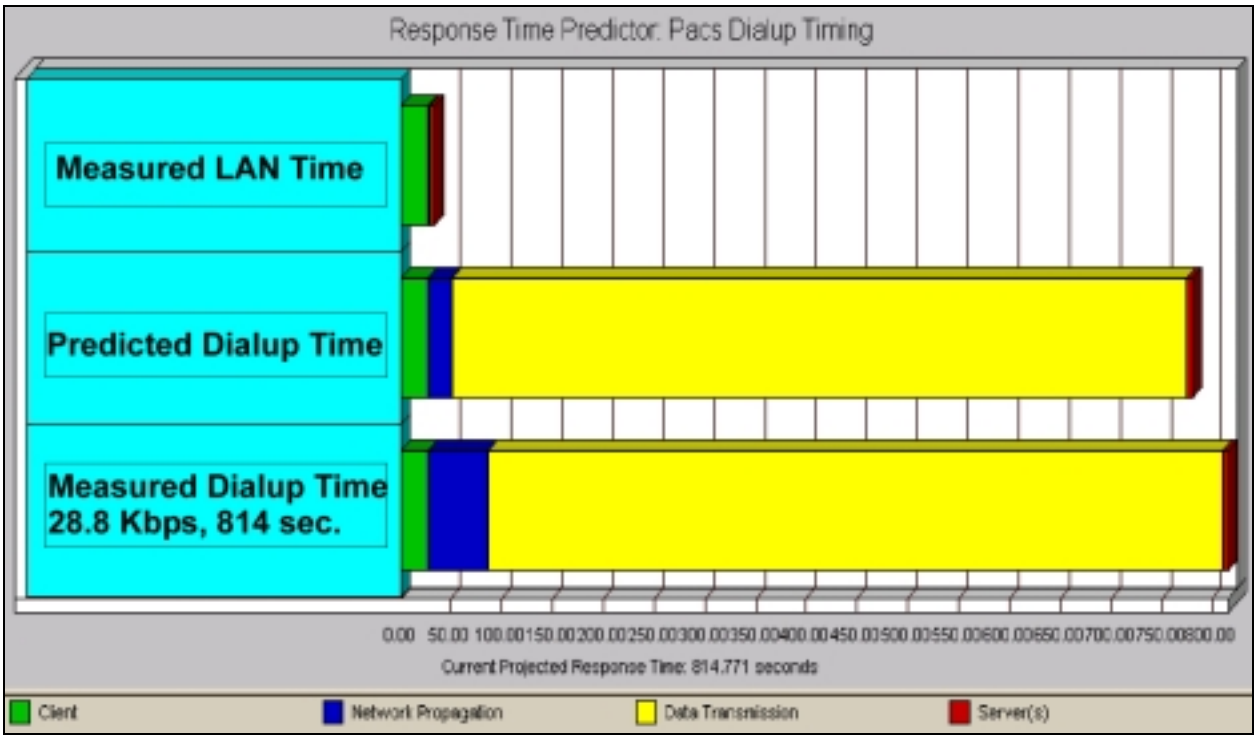
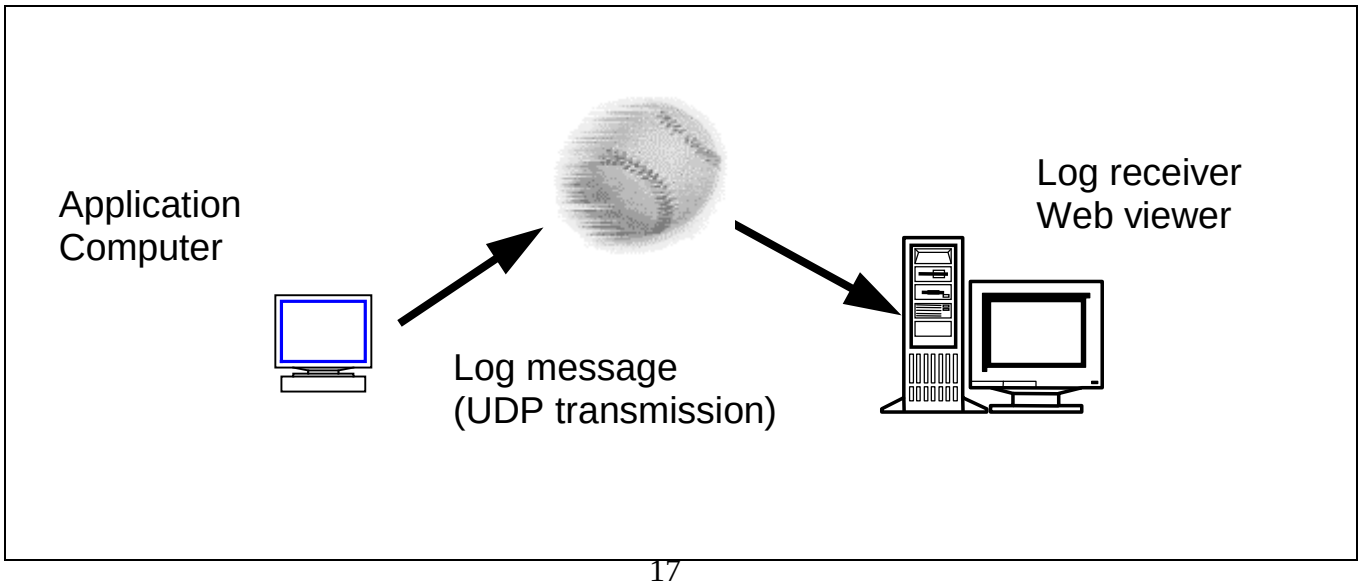


Figure 11. Measured and predicted response time.

Built-In Application Performance Measures

For built-in application performance measures, we are testing a common log receiver and display system. We believe that in the long run, applications with built-in response and error reporting will be essential. There is no standard way of accomplishing this now. The Application Response Measurement (ARM) API and the Application Management Specification (AMS) are just emerging as potential standards. Meanwhile, we are doing some internal experiments.

Figure 12. Common Log Receiver and Display System



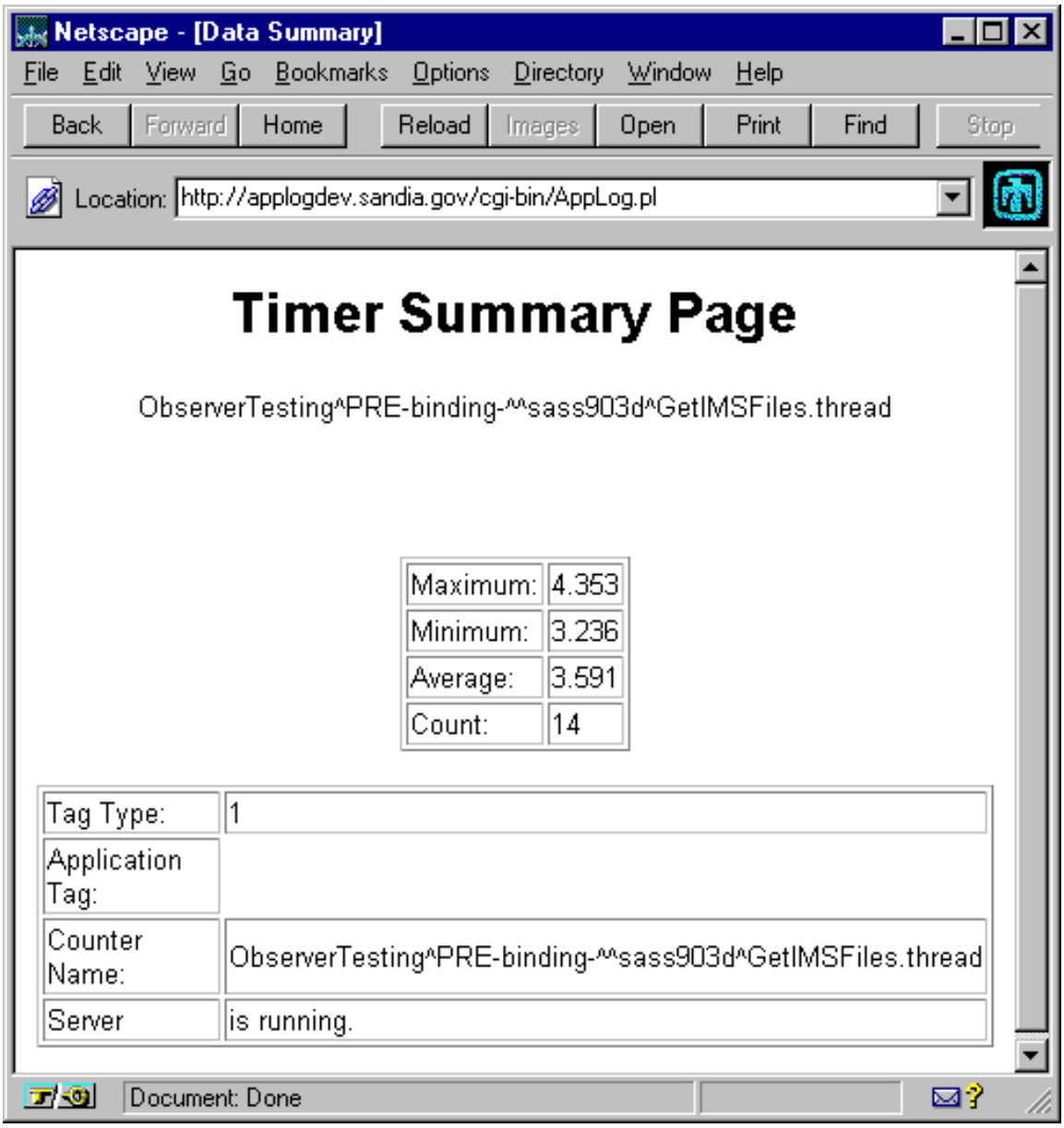


Figure 13. Built-in application performance measurement display.

Figure 12 depicts a method of sending a progress message to a logging receiver from an application on a server or client computer. The screen copy above (figure 13) shows that one function (the PRE-binding module) had an average response time of 3.59 seconds. In this case, each time this module operates, it sends a "start" message and an "end" message to the log receiver computer. The application on the log receiver computes the time between "start" and "end" for each pair and displays the result on a web page. By using UDP transmissions from the application, the application's operation will not be disturbed if the logging receiver or transmission path fails.

Application Response Time Successes

Careful analysis of applications can result in large successes. An examination of one application, Assembly Traceability Database (ATD) / Record Of Assembly (ROA), resulted in significant improvements for a function called current transmittals.

- Start-up and log-out times changed from minutes to less than 6 seconds.
- "Mysterious" network errors like IPX traffic and NFS requests to non-existent files have been eliminated.
- June 1998 "query all" time was 16 seconds, versus 26 seconds in April 1998.
- June 1998 "save" time was 20 seconds, versus 40 seconds in April 1998.

We discovered that no one happened to know the overall configuration of the ROA application. By interviewing and consolidating information from several "limited-area of responsibility" sources, we were able to construct a model that included the physical devices as well as the internal functions of the servers. This allowed the observed communications between the devices to be critically analyzed to pinpoint extraneous functions that could be eliminated.

Response Measurements - Deployment Decisions

Response measurements on the Production and Certification System (PACS) provide a basis for deployment decisions. Here we have a comparison of the average response time of various phases of an application when the executable files are located in three different places: Two remote file servers (880CAPP and 897CAPP), and the local hard drive. Times are in seconds.

Location of files	Startup Time (seconds)	Login Time	Retrieve traveler	Display traveler
880CAPP Server	18	8	3	1
897CAPP Server	17	7	3	1
Local Drive	2	4	1	1

Figure 14 . Application Response times for Various File Sources

OTHER MEASUREMENTS

Packet Analysis

Packet level analysis can identify improvement areas before application deployment. Here, we found that a test Oracle Manufacturing scenario made 8 file searches for each file found on the server; 240 file-not-founds in one “successful” run (Figure 15). The deployed application's search path and file locations were changed to improve this characteristic.

```
Relative
Time
Seconds  source  destination              packet type

1.797 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\FND75\forms\us\APPFLDR.plx

1.799 ds04snlnt. sang013e SMBReply:NT Create (ANDX )(Error:0x340000C0)

1.801 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\FND75\forms\us\APPFLDR.plx

1.802 ds04snlnt. sang013e SMBReply:NT Create (ANDX )(Error:0x340000C0)

1.805 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\FND75\forms\us\APPFLDR.plx

1.806 ds04snlnt. sang013e SMBReply:NT Create (ANDX )(Error:0x340000C0)

1.808 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\FND75\forms\us\APPFLDR.plx

1.809 ds04snlnt. sang013e SMBReply:NT Create (ANDX )(Error:0x340000C0)

1.812 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\AU10\res\us\APPFLDR.plx

1.813 ds04snlnt. sang013e SMBReply:NT Create (ANDX )(Error:0x340000C0)

1.815 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\AU10\res\us\APPFLDR.plx

1.816 ds04snlnt. sang013e SMBReply:NT Create (ANDX )(Error:0x340000C0)

1.818 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\AU10\res\us\APPFLDR.plx

1.820 ds04snlnt. sang013e SMBReply:NT Create (ANDX )(Error:0x340000C0)

1.822 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\AU10\res\us\APPFLDR.plx

1.823 ds04snlnt. sang013e SMBReply:NT Create (ANDX )(Error:0x340000C0)

1.826 sang013e ds04snlnt SMB:NT Create Pid(0xDE00) Root directory Tid(0803)FH(0000)
Tid(0803)\APPS10\AU10\res\plsql\APPFLDR.plx

1.827 ds04snlnt. sang013e SMBReply:NT Create Pid(0xDE00) Tid(0803)
```

Figure 15. Packet analysis showing requests for missing files

Network Load

Knowledge of the network load versus time helps us assess and model the network requirements of new application deployment.

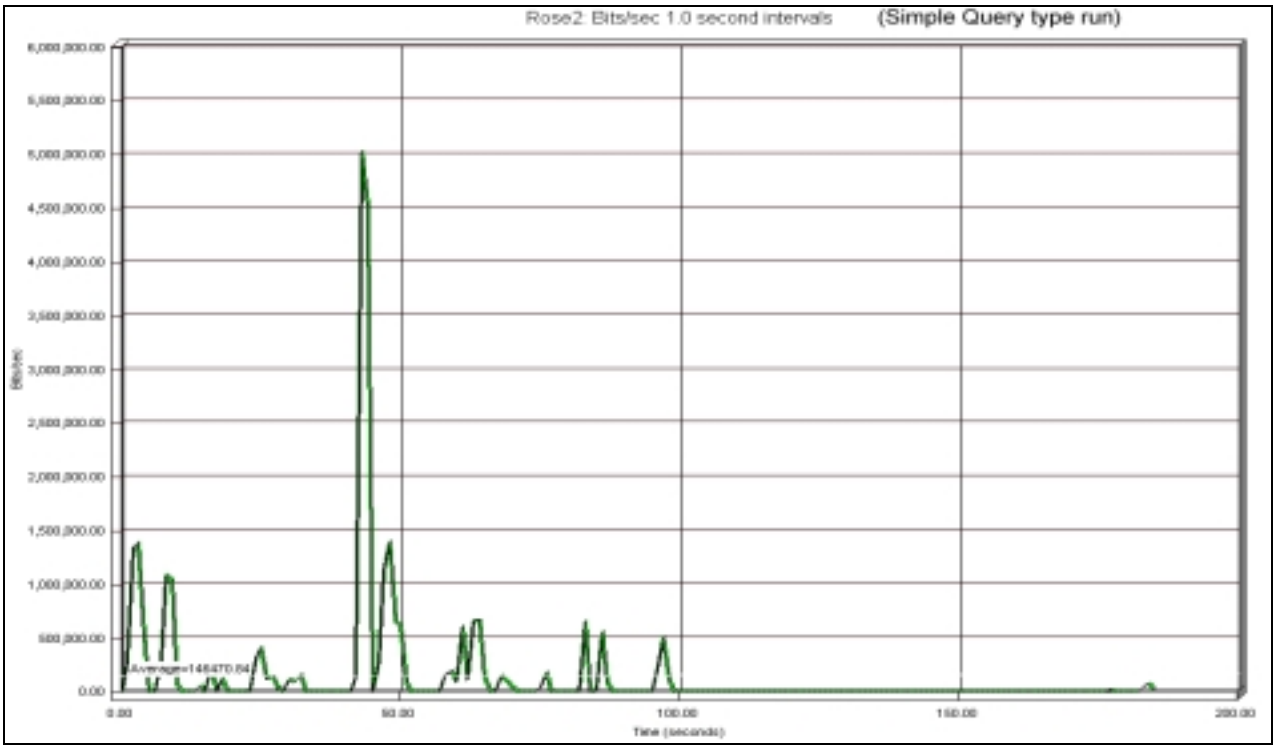


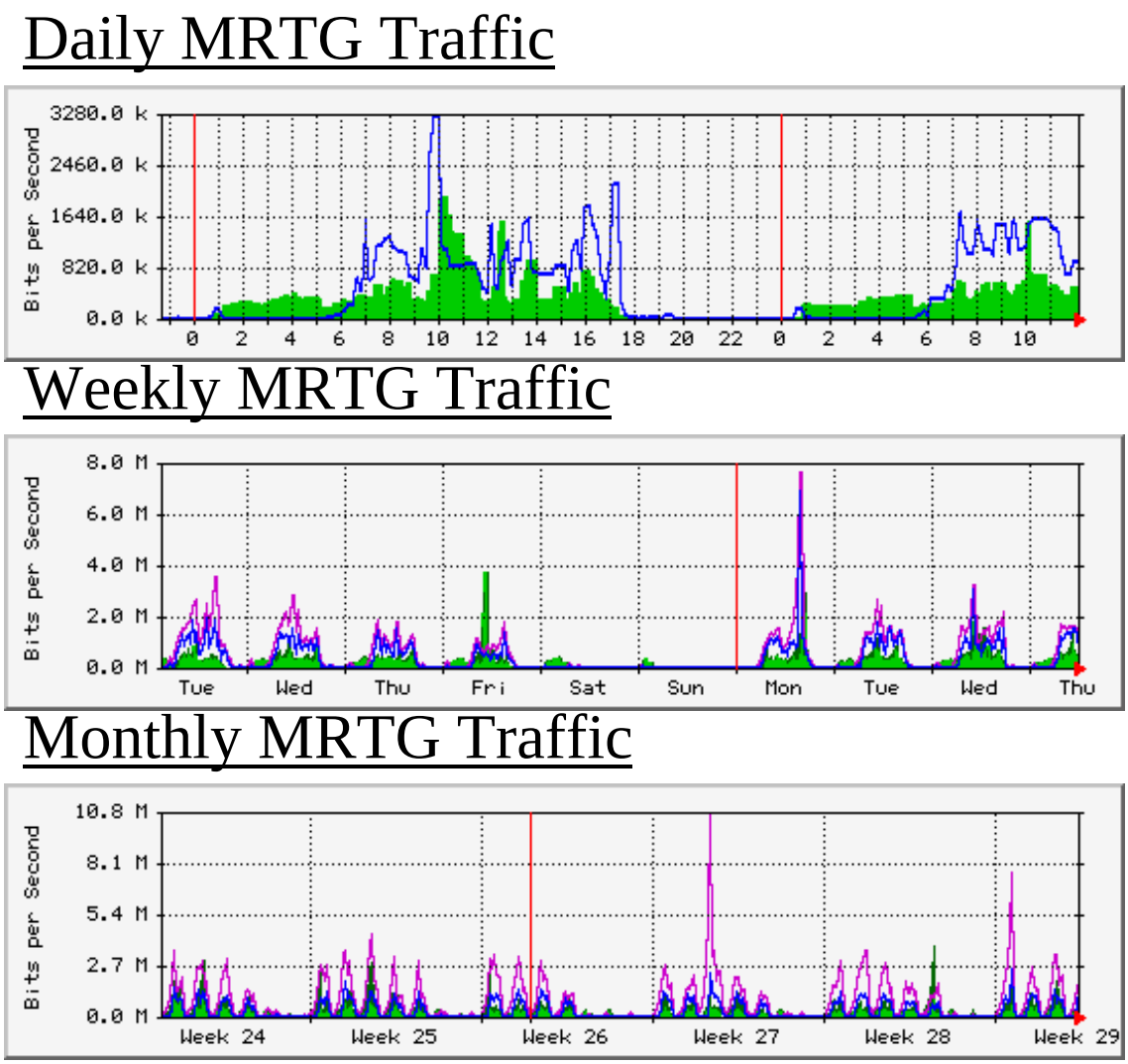
Figure 16. Network load versus time.

In Figure 16 above, the network load in bits per second versus time on the Ethernet line of a client computer are shown. From the timing and duration of the peak network activity it is possible to extrapolate the impact on the network of running the application with ten, fifty, or more users. With an assumption that each user's computer has a characteristic like the sample figure, overlaying multiple charts with expected variation in start times can allow a summation of total traffic; or a conclusion that there will be times of congestion that may slow an individual's response.

Routine Traffic Monitoring

Routine monitoring makes it possible to identify trouble spots in real time as well as displaying short term and long term trends in network traffic. We use a locally configured, public domain software package called MRTG (Multi Router Traffic Grapher) to interrogate SNMP network devices and plot traffic at key points and at key computers. Observation of the traffic at multiple points of the network is still a manual process requiring someone to look at the graphs to decide if there are trends requiring action.

Figure 17. Samples of MRTG traffic plots.



Thread Analysis

Frequently, packet captures from the network need to be analyzed to understand what is happening. The following is an example of a desktop PC's boot up conversation with a Domain Controller (DC). This particular "working packet capture" was documented and is used from time to time to compare with problematic desktops.

Figure 18. Thread analysis annotated sequence.

```
DNS uses (UDP port 137 NetBIOS-ns - name service)
<00> Join the network with a desktop unique name dbbate2:
C ID=32768 OP=REGISTER NAME=DBBATE2<00>
R ID=32768 STAT=OK
<00> Join the network as a desktop in the group named CSU880:
C ID=32770 OP=REGISTER NAME=CSU880<00>
R ID=32770 STAT=OK
Request the list of Domain Controllers:
C ID=32774 OP=QUERY NAME=CSU880<1C>
R ID=32774 STAT=OK  This list has 181.29, 181.29, 219.123, and 192.36
Notice 181.29 is listed twice, once as the PDC, and again a DC.
Now query every DC in the previous STAT=OK list:
   (include local broadcast and subnet broadcast)
Note the following eight (8) SMB transactions are (UDP port 138 NetBIOS-dgm - datagrams).
C Transaction \MAILSLOT\NET\NTLOGON  Call to Local Broadcast
C Transaction \MAILSLOT\NET\NTLOGON  Call to the first DC in the list.
C Transaction \MAILSLOT\NET\NTLOGON  Call to the second DC in the list.
C Transaction \MAILSLOT\NET\NTLOGON  Subnet broadcast
C Transaction \MAILSLOT\NET\NTLOGON  Call to the third DC in the list.
C Transaction \MAILSLOT\NET\NTLOGON  Call to the fourth DC in the list.
C Transaction \MAILSLOT\NET\GETDC323  (A rapid response from the first server!)
C Transaction \MAILSLOT\NET\GETDC323  (A rapid response from the second server!)
Notice the last two Source replies to the prior Commands (C) are also Commands, not
Responses (R)!
C ID=32776 OP=QUERY NAME=CSU880-PDC  Asking for the PDC's IP address.
R ID=32776 STAT=OK  Address is returned = 181.29
```

SUMMARY

The role of the Sandia Integrated Information Services (IIS) Applications Analysis Group is to understand the network impact of applications and to provide information to enhance application performance during the system life cycle: design, programming, testing, deployment, and production operation.

Rather than just taking remedial action we want to apply the process to new applications. We believe that an initial assessment should be done as a new application is planned and developed. Moreover, as an application is rolled-out and moved to production status, continuing measurements of availability and response time should be done. Our development of application analysis tools and procedures is going well. An application description includes a network-view diagram and message sequence chart. The techniques are being applied to various enterprise applications including DINERO, Foreign Travel, WinFrame/Business Objects, Oracle Manufacturing, and WebSIMS. The built-in application event log procedures are being tested with a development database project (IIM) and will be tried with other systems such as the OneScreen PACS/PDM code.

Only by instrumenting the network to provide insight into the application layer can IT organizations unite the technical and business perspectives, the top and the bottom of this model. External observation tools alone cannot solve today's problems. A thoughful and guided approach to this architecture empowers an IT organization to make the transition from reactive to proactive, and from managing the network to business-critical networking.

DISTRIBUTION

1	0629	Al C. Alvarado, 4800
1	0801	Melissa Murphy, 4600
1	0806	Michael Vahle, 0806
1	0806	Lawrence Tolendino, 4616
1	0803	L. Herbert Pitts, 4900
2	0807	Ivory Alexander, 4922
5	0807	Dennis Bateman, 4922
5	0807	David Evans, 4922
5	0807	Dal Jensen, 4922
5	0807	Spencer Nelson, 4922
1	0811	Thomas Ferguson, 4813
1	0812	Michael Sjulin, 4614
1	0863	Jorge Hernandez, 14407
1	9018	Central Technical Files, 8940-2
2	0899	Technical Library, 4916
1	0612	Review & Approval Desk, 4912 For DOE/OSTI