# A UNIFIED AND COLLABORATIVE APPROACH FOR ANALYZING NETWORKS

BY

NAVEEN KUMAR KRISHNAPRASAD
B.E., Instrumentation and Control Engineering, University of Madras, Madras, India, 1998

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2001

Chicago, Illinois

To my parents,

Chandrika Prasad and

Krishnaprasad

## ACKNOWLEDGMENTS

I would like to start by thanking Jason Leigh for accepting me as a RA at EVL and at the same time, Satheesh Ganapathy for recommending me to him, because coming into EVL has been probably the most important thing that has happenned to me.

I am extremely thankful to my thesis committee - Andrew Johnson, Tom Defanti and Jason Leigh, for their non-stop support and faith in me, always believing that I would make it here, inspite of all the deadlines. I would also like to thank all the EVL staff and co-workers who have, and continue, to make this a great place to work.

In all the time I worked on this thesis, I had some continuous support flowing in from a bunch of people around me. In particular I thank, Brenda Silva for her artistic inputs into uCAN, Alan Verlo for spending late nights and always trying to make sure the networks worked for me, Lance Long for helping me out with the systems issues I had all along, Rajvikram Singh and Vivek Rajan for their help in setting up the final experiment, Shashank Khanvilkar and Kothonuzo Luruo for their valuable suggestions as network experts, and all the members of the CAVERN group who have lent their hand at every point.
I also thank Satheesh and Vivek for being great companions at EVL in times ranging from late night conversations to tight debugging situations.

I am grateful to my roommates for being there to understand and support me through many tough times during my masters and making my stay here as wonderful as it was.

Being a mentor is role that takes considerable effort and devotion. I am indebted, and will always owe a good portion of any future success to Jason, for being one.

## ACKNOWLEDGMENTS (Continued)

I finally dedicate these last few months of my efforts and whatever good that might turn out of it, to my family, who have tolerated, pushed and moulded me all my life.

**NKK**

# TABLE OF CONTENTS

# LIST OF FIGURES

## LIST OF FIGURES (Continued)

## LIST OF FIGURES (Continued)

# LIST OF ABBREVIATIONS

| | |
|---|---|
| uCAN | Unified Collaboratory for Analyzing Networks |
| CAVE | CAVE Automatic Virtual Environment |
| QOS | Quality Of Service |
| QoSIMoTo | Quality Of Service Internet Monitoring Tool |
| SNMP | Simple Network Management Protocol |
| EVL | Electronic Visualization Laboratory |
| NLANR | National Laboratory for Applied Network Research |
| CAIDA | Cooperative Association for Internet Data Analysis |
| MUD | Multi-User Domain |
| CAVERN | CAVE Research Network |
| API | Application Programmers' Interface |
| DSO | Dynamic Shared Object |
| NIMI | National Internet Measurement Infrastructure |
| ATM | Asynchronous Transfer Mode |
| ULM | Universal Logger Format |
| VNC | Virtual Network Computing |
| IP | Internet Protocol |

# SUMMARY

Analyzing fast-growing networks involves considerable complexity, with the number of networked applications increasing exponentially. Existing techniques to analyze networks make use of network data collection, measurement and visualization to understand the state of networks. However, performance problems in networks may be caused by specific interactions from applications, which may or may not be running within a local network controlled by one network manager. This calls for two primary capabilities not addressed by traditional approaches. Network state information has to be first viewed and understood in the context of performance of applications. In addition, the information that can thus be obtained has to be shared with other remotely situated network managers, hence taking advantage of group collaboration to accelerate problem solving. Working towards this goal, this thesis proposes an extendable, unified and collaborative approach for analyzing networks

The approach first proposes ubiquitous access to run active tests between machines on a network and basic network management queries, using the Simple Network Management Protocol(SNMP). A unified framework designed using a centralized server, allows simultaneous analysis through message passing between processes that monitor the results of a network test, and processes that monitor changes in a Management Information Base(MIB) entry of any network device. The framework also allows any new application written, to connect to the same centralized event server used, and send event messages concerning specific, possibly undesirable, conditions reached while execution.

The unified framework is complemented and merged with a collaborative framework to enable network managers to log-on to a group session and share the interfaces that control the network operations. Users can view results of operations initiated by others and also request control of interfaces that they see, to conduct their own tests on test-beds of other users in the session. The thesis demonstrates the result of such a unified and collaborative approach proposed by introducing a new type of network analysis tool called UCAN (Unified Collaboratory for Analyzing Networks) implementing the capabilities described.

UCAN (or uCAN) employs a three-dimensional visualization tool to analyze network performance of applications. A generic, configurable two-dimensional graphic visualizer is provided to monitor active tests and changes in MIB infomation using SNMP. uCAN also contributes utilities implemented as a reusable set of C++ classes built over the CAVERNsoft high performance networking toolkit(1) and the FLTK user interface library(2). The utilities are designed to facilitate building new interfaces that can be shared in a group session as described above. The goal is to help fit in new tools into the collabratory and give network administrators and users, a more complete view of an entire network.

# CHAPTER 1

# INTRODUCTION

**Analysis:** *"The resolution or breaking up of anything complex into its various simple elements; the exact determination of the elements or components of anything complex"*

- The Oxford English Dictionary

Revisiting this familiar definition of analysis gives an intuitive introduction to the context of analyzing networks. Networks are made of several components which interact and shape the performance of end applications that use it. The packets and streams that come out of an application travel a path riden by routers, hosts, switches, fiber and cables, crossing other subnetworks on the way to the destination. These components are managed by network managers and administrators of the corresponding domains. Hence, understanding and debugging network performance problems in applications requires an understanding what the networks, with all its components, does to the application. This has to be done by first measuring the network performance in an application in terms of standard parameters like bandwidth, delay, jitter and packet loss and then testing the network for those parameters.

Testing can be done by introducing new traffic to measure the network or using a portion of the existing traffic. Though testing helps to measure and verify results from a network, a large network cannot be managed by frequent human intervention, whenever problems occur, with many applications running at the same time. Hence automated network management becomes a necessary part of maintaining networks. Network management involves collection of data from devices on the networks like routers, switches and hosts and defining automated responses to the state information obtained from the data. In cases of big networks, detection of a bottleneck may not be entirely within the domain of a single network manager. In such cases, there is a need for communicating results of network tests and interpretations to others involved. Analyzing networks, therefore, has to be discussed in the context of all the components in a network and also the above techniques and issues involved in measuring and understanding networks.

## 1.1 Collaboratories

*Collaboratories* are defined(3) as systems that create integrated tool-oriented computing and communication systems to support scientific collaboration. The main goal of collaboratory projects has been to make use of communication and computational technologies to support collaboration and thus enhance contributions to technology. In line with this goal, this thesis also aims to explore the use of an integrated collaboratory of tools required for analyzing networks.

To explain the context of the thesis, the advantages and disadvantages of existing techniques and systems, and the need for a different approach are first discussed. The approach and the initial big picture are provided in chapter 3. Chapters 4 and 5 that follow, describe the approach in detail, in terms of the objectives and the design required to meet the objectives. Chapter 6 gives some sample results to demonstrate the usefulness of uCAN and describes the direction to work towards building a framework that helps to fit in other new network tools for collaborative analysis.

# CHAPTER 2

# RELATED WORK AND ANALYSIS

## 2.1 Network Analysis -Tools and Sytems

A number of network analysis systems and tools have been built in the past, using different approaches to measure and monitor networks. To understand and analyze them better, they should be classified broadly according to the methods they employ, namely,

1. Data Collection from Network Devices
2. Active Measurement
3. Passive Measurement
4. Network Visualization
5. Application Performance Monitoring

### 2.1.1 Data Collection from Network Devices

Collecting data from routers, hosts and other network devices is one of the least expensive and convenient methods of network analysis, since it uses existing equipment in a network and can be done by anyone.

SNMP is one of the most commonly used protocol to exchange management information between devices on the network. Several commerical and free tools have been developed that use SNMP and provide user interfaces and programming libraries that are simple to use. OpenView(4) by Hewlett Packard is a network management and monitoring tool that does real-time measurement and non-realtime data collection. It can also detect abnormal network events and indicate through alarms to the user, along with providing user defined automated responses to such events. Comparable to HP's OpenView in the Network Management category is CiscoWorks(5) by Cisco Systems. CiscoWorks has a bunch of tools including a real time fault detection manager and a flow traffic analyzer. It also provides low-level control of networks and allows network administrators to set Quality of Service (QoS) parameters to analyze traffic-shaping and performance. There are also plenty of public domain SNMP libraries and related network monitoring software, a list of which can be referred on (6) and (7). Some popular SNMP libraries are Scotty(8), the Multi Router Traffic Grapher (MRTG)(9) and the NET-SNMP(10) software libraries.

Although SNMP is a convenient way of network management, there is a compromise on the information about the individual flows in the traffic. Another standard called NetFlow(11), Cisco's technology for collecting router data, provides the metering base for a variety of applications including network traffic accounting, usage-based network billing, network planning, network monitoring and data also mining capabilities. NetFlow works by identifying different types of flows from its routers and switches which are instrumented to provide the relvant data required. Examples of tools available that collect and analyze NetFlow data from routers are cflowd(12) from the Cooperative Association for Internet Data Analysis (CAIDA) and FlowBoy(13) from the National Center for Supercomputing Applications(NCSA). A disadvantage of the NetFlow method however, is that it can only randomly sample a percentage of the actual link load.

### 2.1.2 Active Measurement

Active measurement tools work by generating network traffic to measure network parameters. Active measurement tools follow well standardized methods and are often used by network analysts to obtain information about a network's connectivity and the overall performance. Popular active testing tools used include the Netperf(14) benchmark tool for bandwidth measurements, the Ping(15) utility program for round trip time measurements and the Traceroute(16) utility for determining routes between machines on a network.

There have been several large scale projects in the past to measure and characterize the Internet. The Surveyor(17) and the Réseaux IP Européens (or RIPE) Traffic Test Project(18) are both examples of projects that measure one-way latencies and loss using the Internet Engineering Task Force (IETF) Internet Protocol(IP) Performance Metrics Standard(19). The National Laboratory for Applied Network Research (NLANR) is working on the Active Measurement Project(20) which consists of a distributed network of approximately 130 active monitors across the United States performing scheduled measurements between each other. The problem with the methodology in the above projects was that the infrastructure of tests were controlled by a single entity, thus not providing ubiquitous access to all participants. This need was addressed by the National Internet Measurement Infrastructure (NIMI) system(21). The NIMI consists of a set of measurement servers, configuration and control software running on a number of hosts in a network. NIMI aims on providing scalability to thousands of probes within the infrastrucure. Another important consideration by NIMI is the delegation of different control powers to different participants via cryptographically secure credentials.

### 2.1.3 Passive measurement and monitoring

Since active measurements retreive information by adding interference into a network, they dont give accurate indications of what the network does to an application while the application is running. Passive measurement systems overcome this problem by using splitters that dump packets from the network, with negligible interference. The OCxMon (22) is a set of monitoring tools that make use of passive Internet monitors built from PC hardware and ATM Network Interface Cards, employing optical splitters to snoop on fiber. The packets thus obtained can be collected and analyzed with the aid of tools like Coralreef(23) and the Network Traffic Flow Measurement Tool (NeTraMet)

Tcpdump(24) is another popular passive monitoring tool used to echo packet information up to and including payload content, to standard out or a file. The packet information is gathered from the local network interfaces after it has been placed in promiscuous mode.

### 2.1.4 Network Visualization

Despite the advancements in techniques for measuring and managing networks, there is always a need for an intuitive visual representation of the information gathered. Hence network visualization has been a very important part of analysis. Graphic visualization has been used to understand the nature of traffic flows as well as network topologies. The Cichild(25) 3D Visualization software developed at NLANR is a comprehensive tool written based on OpenGL and MesaGL graphics libraries, which can visualize arbitrary data coming from the network from any source. It provides 3D bar charts and topology maps which the user can interact with in real time. GraphViz(26) developed at the ATT Research Labs, is a package that provides a bunch of automatic graph generation tools. It addresses the problem of visualizing structural information by constructing geometric representations of abstract graphs and networks. GeoBoy(27) and VisualRoute(28) are examples of tools that provide different types of visual representations of traceroute results.

### 2.1.5 Application Performance Monitoring

All the different approaches for network measurement and analysis discussed above give us a wealth of network state information. But in most big networks today, there are applications interacting with each other over the network and also with the systems at their ends. So there can be a number of complex interactions which can increase exponentially with the number of applications and the number of network flows in them. Hence it is clear that in order to understand the interactions of the application with the network, the application has to be first instrumented for performance analysis. Though there have been systems that help to characterize end-end performance of distributed applications, with a focus on CPU utilization(29), there seems to be very few systems that characterize network performance. NetLogger(30) is one such system that proposes a methodology to understand behaviour of all elements in the application-to-application communication paths. It provides timestamped logs of interesing events at critical points of the distributed system. All tools in the NetLogger toolkit share a common log format called Universal Logger Format (ULM)(31) for logging and exchange of messages. All applications that run on a network can send their performance data to a server daemon that collects data to analyze it together. The Netlog library(32) developed at the National Center for Supercomputing Applications is a C language library that can linked to an existing network application to provide some instrumentation of network performance. It also comes with a visualization tool called Viznet that understands the log format from Netlog and provides a 2d graph visualization.

It can be inferred from NetLogger and Netlog that understanding performance needs a visualization tool to give meaningful intuitive representations of data.While NetLogger and NetLog come with useful 2d visualizations, in cases of complex networked applications like tele-immersion(33), there is a need to leverage 3D to facilitate simultaneous analysis of multiple network streams in the application. QoSIMoto(34) , developed at the Electronic Visualization Laboratory is a 3D visualization tool that understands ULM type of log data from a performance-server daemon and can be used for both real-time and non real-time analysis. The 3D graphs can be viewed in a **CAVE**$^{TM}$(35) like immersive virtual environment, in which the user can interact with the 3D visualization. QoSIMoTo also provides event based analysis, as proposed by NetLogger.

## 2.2 Analysis

### 2.2.1 Lessons learnt

A study of all the several types of network analysis tools indicates clearly that there have been many approaches, which were useful to people for different purposes. It can be observed that active testing has been very useful in cases where accuracy and convenience of testing was concerned. Though passive testing is the most suitable for analyzing the interactions between the network and the application, past experiences have shown(36) that passive testing is not standardized and is usually tough for everyone to conduct, owing to the apparent security and billing issues involved. Experiences with the NIMI system(21) indicate that ubiquitous access and subsequent delegation of control is an important feature not incorporated by many network measurement systems.

Comparing the methods for collecting router data, NetFlow is more suited for applications where knowledge of the flows and more low level control is needed. But a disadvantage of using NetFlow is that only a portion of the router traffic can be sampled and there exists an unnecessary load on the router for the traffic to be forwarded to a collector(11). SNMP is still widely used owing to its simplicity, convenience and the wide number of publically available tools and libraries. With respect to monitoring applications, NetLogger demonstrates the use of an event based analysis by instrumenting distributed applications. Also NetLogger's idea of using a server daemon to collect logged traffic from different machines in a network, for real time and non-real time analysis, is useful. It is also quite obvious that network visualization has to aid the information collected by network performance tools.

Hence, to summarize(visually), we can arrive at the basis for a new network analysis tool as follows:

### 2.2.2 The Missing Pieces

#### 2.2.2.1 The Case for Unified Analysis

The summary of different types of tools and techniques available for network analysis as discussed above, indicates that all types of approaches are useful to people in different circumstances. But most of the techniques available become really very useful and sufficient when one has narrowed down and defined the problem that exists in the network and the probable source of the problem. However in most circumstances in which several applications interact with a network at the same time, there are usually many interactions between components that are

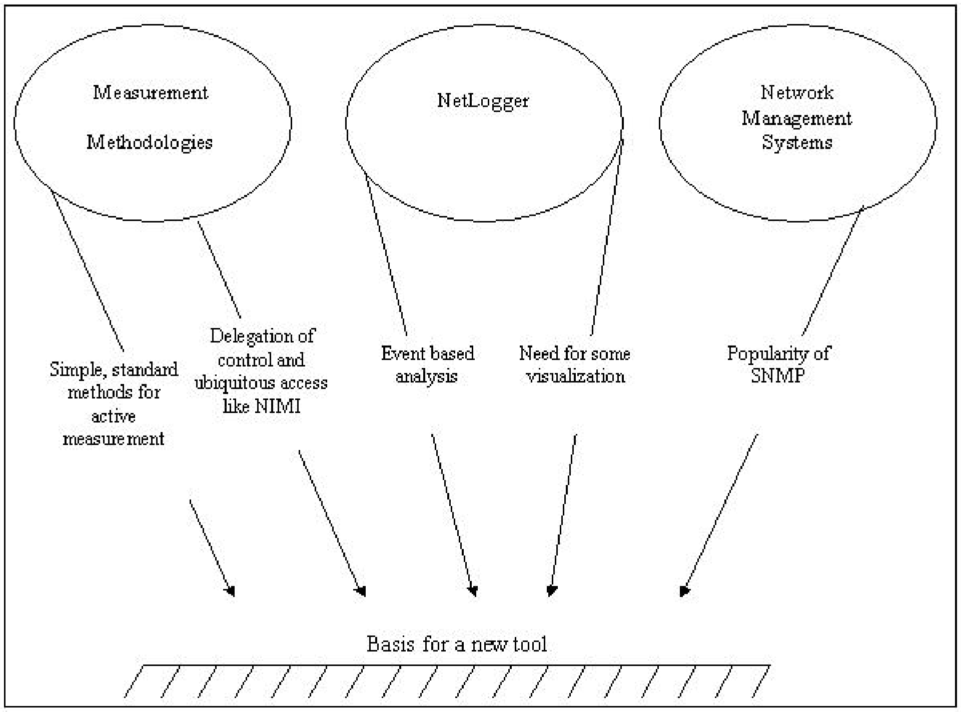

Measurement Methodologies
NetLogger
Network Management Systems
Simple, standard methods for active measurement
Delegation of control and ubiquitous access like NIMI
Event based analysis
Need for some visualization
Popularity of SNMP
Basis for a new tool

Figure 1: *The lessons learnt from past experience and the basis for a new analysis tool.*

not known ahead of time. Hence in addition to identification of a bottleneck in an application, simultaneous state analysis of the relevant network parameters also has to be done while an application is running. This leads to a need for a unified platform to analyze an application's performance simultaneously with the network state information obtained through tests or data collected from routers and other devices. This is a problem that has not been addressed by traditional approaches. NetLogger proposes an event based analysis of application flows along its path in a distributed system, but still assumes the identification of the network bottleneck after the identification of a performance problem.

The unified analysis approach proposed by this thesis starts with NetLogger's philosophy, taking events captured from the application to be viewed in the context of information about the backbone network, obtained from SNMP data and active tests. The goal is to help each different approach mentioned above, complement the others. The focus is on designing a framework by which other new network tools can be fitted in for such a unified analysis. Detailed accounts of the individual components of the framework design along with the implementation in uCAN, supplemented with screenshots are given in Chapters 3 and 4.

#### 2.2.2.2 The Case for Collaboration

Real-time collaboration helps problem solving when there is a need for sharing information between geographically seperated parties. Computer Supported Cooperative Work(CSCW) has shown that there is trend towards sharing information to share and ultimately gain intellectual expertise(37). Virtual Network Computing(38) and TeamWorkStation(39) are recent examples of shared desktop systems which demonstrate clearly how shared workspaces can aid in information sharing and group collaboration. Experiences also show that collaboration has been a very useful tool for exploiting different perspectives of participants in a shared environment(40) and in cooperative computer aided design(41)

It can be argued that in the case of small networks or private organizations, where control of a network might happen at one place, collaboration may not be needed. But for most big networks, network analysis deals with many situations where problems can be anywhere on the network. In most cases, network managers who control different sites and subnetworks are often geographically seperated and have to communicate and collaborate with each other for problem detection and maintenance in big networks. The case for collaboration is also conceivably strong in the context of research networks and consortiums like the Internet2(42) and the Science, Technology, And Research Transit Access Point(STARTAP)(43). These type of research networks involve tighter collaboration between participating institutions and deal with International partnerships with overseas couterparts. Hence in research networks as these, the corresponding network managers are more geograhically seperated and sometimes in different countries.

The United State's Department of Energy's Science Grid(44) aims to provide the advanced distributed computing infrastructure needed for the scientific community based on Grid middleware and tools. The aim of the Grid is to make better use of distributed computing and collaborative environments to facilitate computing and accelerate research. Working on the same lines, is another project funded by the Department of Energy, called Access Grid(AG), which has been gaining popularity in the recent past. The Access Grid(45) provides human interaction interfaces across the Grid like video/audio desktop conferencing tools to encourage group to group communication like virtual meetings between research collaborators and distance learning. The fast growing AG community has 60 domestic and international partners, including research institutions and universities. So, viewing network analysis in the context of this trend towards group communication and collaborative environments, one can appreciate that solving performance problems in big networks will need support for tighter collaboration than what exists currently, between the participants.

The thesis proposes a collaborative approach by which participants can connect to a collaborative session and share their network analysis tools. Remote participants can therefore view test parameters and graphical views of tests initiated by others. They can also request control over a test to reconfigure and control it from their ends. uCAN implements this approach and also aims to contribute a set of utilities to help build collaborative tools, including a generic interface for building shared network widgets using the FLTK(2) interface library and an API for building collaborative testing tools, by which users can also add their own tests to a test panel and make it collaborative. The goal is to encourage collaborative tool building and work towards a seamless shared workspace which gives a more complete view of the entire network, where both network information and performance profiles from applications of interest are shared amongst all the network analysts concerned.

A detailed account of the collaborative framework, its components and some screenshots to illustrate collaboration are provided in chapters 3 and 5.

# CHAPTER 3

# THE VISION OF A UNIFIED AND COLLABORATIVE APPROACH

This chapter explains why a unified and collaborative approach will be useful for problem solving and introduces the general solution proposed. The details of design and implementation are left to the following chapters.

## 3.1 The proposal

### 3.1.1 Why a Unified Approach

An analysis of most network measurement and monitoring techniques, indicates that every technique has its strengths in the context of certain type of problems and is required in different ways, for obtaining different kinds of information. A unified approach will make better use of these techniques by combining them in a way so that one can obtain more context information. For example, consider this rough scenario in which two networked applications communicate which each other, exchanging infromation through different types of flows. Hypothesize that the following information can be obtained:

1. A SNMP trap is set on the number of packets dropped by a router on an interface.
2. An application, if its instrumented to know its network performance, is programmed to identify any particular condition that caused an exception, while execution, like a missing piece of information not transmitted from another application on the network.
3. A non intrusive ping test is started to give an idea of the latency between two machines on which the applications are executed.

The idea is to put all these pieces of information together, understand the context better and identify a bottleneck faster. The source of the problem could be for example,

1. A congested condition at the router involved (which can be immediately detected by SNMP queries)
2. Irreponsible use of the network by applications (this can be verified by an event based analysis of performance data from applications)
3. An undesirable link condition on the path to or from a router along the path(this can be verified by active tests).

In the case of small networks or very simple applications, simple components of analysis would suffice. But this thesis aims at bigger networks and more complex networked appllications like tele-immersion(33), where several types of flows can be possible in an application and increasing the interactions with the network. Hence it can be clearly understood from the scenario that a unified approach to obtain information from both networks and the applications that use it, will help in a much better understanding and faster problem identification.

### 3.1.2 Why a Collaborative Approach

Collaboration helps when sharing information aids in understanding and solving problems. Network analysis is certainly an area complex enough to look for such ways to accelerate problem solving. The main motivation to consider collaboration is that big networks always involve management by more than one particular site. So in reality, there will be several network managers who have access to routers and hosts that occur on the path of a network, through which data from applications get transmitted. The fastest way of identifying bottlenecks is there fore through a real-time collaborative framework for group collaboration. Access to network operations is provided to everyone in a group, thus allowing all network managers to contribute by analyzing portions of networks that they control, and also being able to perform operations on other networks with the permission of the corresponding network managers.

Now for a reconsideration of the scenario mentioned in the previous section, but assuming that two network managers control two different portions (or sites on which the applications run) of a network. Assuming that it is possible to obtain the same information described earlier, but in this case, network managers contribute by sharing the SNMP and active test information, or the application's performance visualization from their corresponding sites. The result would be that both network managers involved can see results of operations initiated by each other. A sample set of the source of the problem could be,

1. A congested condition at the router involved, managed by the first network manager(which can be immediately detected by his SNMP queries, which can be shared with the second network manager)
2. Irreponsible use of the network by applications in the domain of the second network manager(this can be verified by an event based analysis of performance data from applications on his domain, shared with the first network manager)
3. An undesirable link condition on the path to or from a router along the path(both identification of a bottleneck in the link, and the verification can be done with active tests carried by both network managers).

There can be many permutations to the source of the problem and it can get much more complex, but it is evident that real-time collaboration would make it much faster to identify and solve a problem. Current means of collaboration, between network engineers/researchers require a considerable amount of initiative from participants to communicate and share results of their experiements. The collaborative approach proposed also includes role based access to operations. Users viewing operations started by others, would have to request control over the operation from the corresponding network manager. The grant of control of an operation to others is left to the discretion of the concerned administrator who initiates the operation.

The usefulness of such a unified and collaborative approach discussed, forms the foundation for a new type of networking tool that enables:

1. Monitoring applications for their performance and visualize information
2. Testing networks for its parameters
3. Managing networks, querying network devices
4. Collaborating with remote users, providing them views and access to all these operations

The tool implements the notion of a unified framework for these operations and also presenting them in a collaborative environment, thus, in short creating a *Unified Collaboratory for Analyzing Networks* (Figure 2)

## 3.2 A High level Description of Framework Required

### 3.2.1 The objectives of an event reflector

1. There should be a framework to exchange events between applications and network network state monitors.
2. It should be simple to send events from any new application which is not part of the uCAN framework.

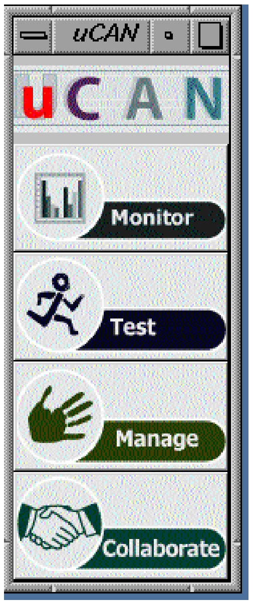


Figure 2: *The uCAN main menu. Conveying that, -u-can monitor an application for its performance, test the network to verify network conditions, manage your network and collaborate with others to analyze problems better.*

3. This internal event-control framework should also allow other messages to be passed other than events. This becomes useful in acheiving collaboration between the individual elements, as explained later in Section 6.2.2.

### 3.2.2 The Framework for Event Reflection

Figure 3 gives the general idea behind the *internal event reflector* and the clients. *Events* in this context refer to conditions reached in the network state -either a state value change in a network parameter to an abnormal condition, or the entry into a collaborative session, indicating the central collaboration server to connect to, or conditions in the state of an application of interest. A *reflector* is essentially a server that manages a list of clients that connect to it. An event message sent from one client is reflected to all other clients by the reflector. The

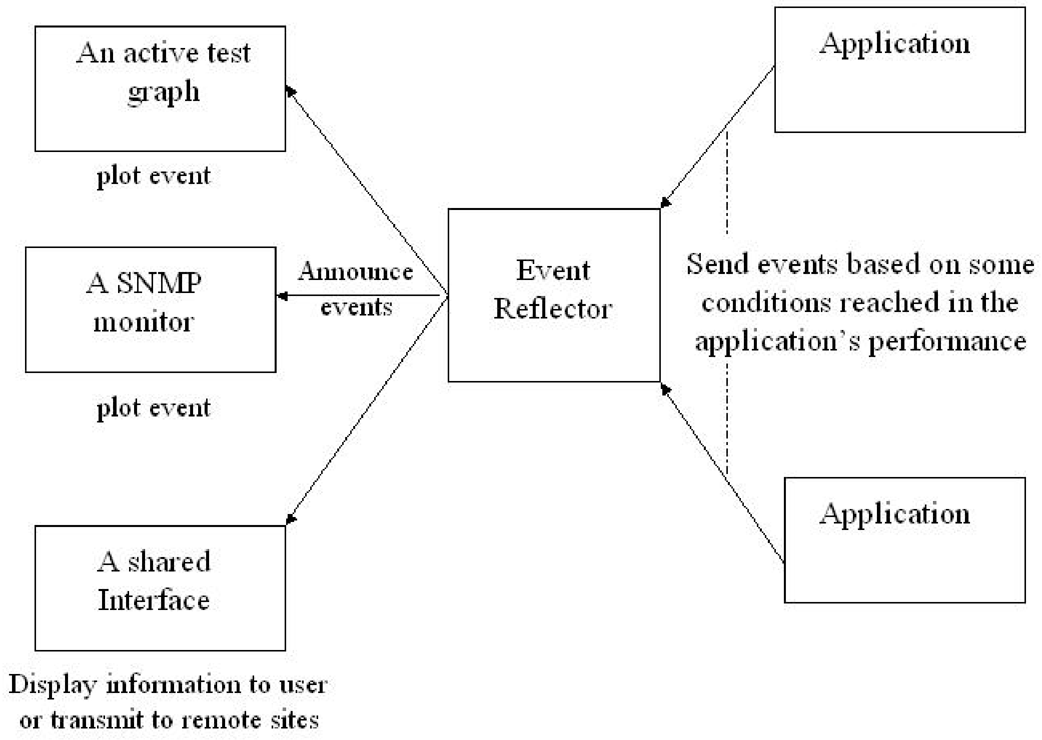
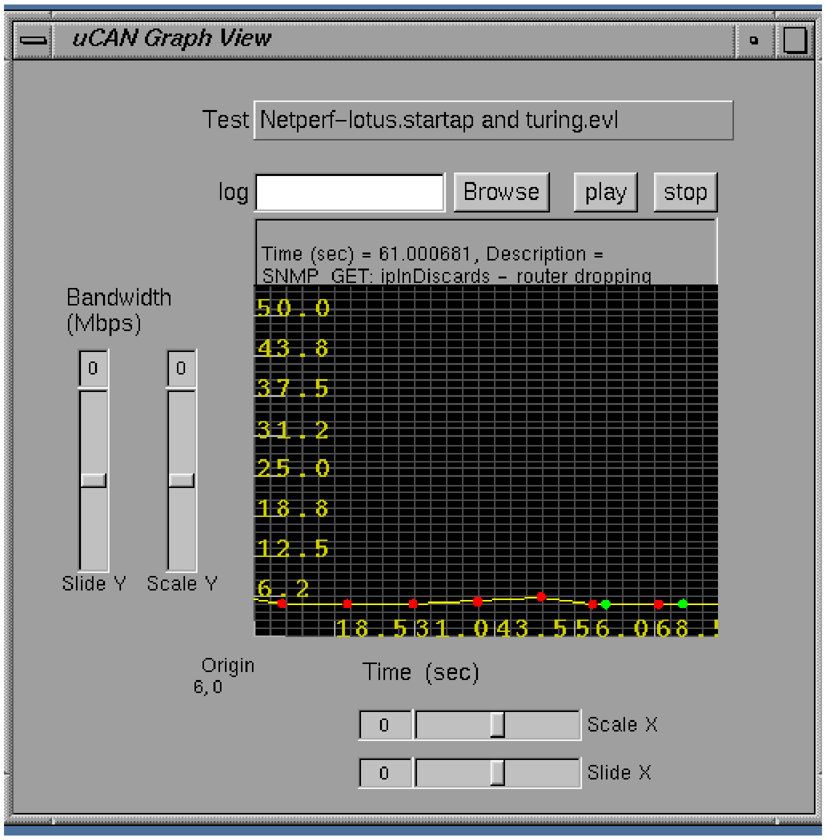

An active test graph
plot event
A SNMP monitor
plot event
A shared Interface
Display information to user or transmit to remote sites
Announce events
Event Reflector
Application
Application
Send events based on some conditions reached in the application's performance

Figure 3: *The diagram shows clearly how the events might be used and sent by processes. Events can also be sent from tests, snmp graphs and other network monitors to applications to indicate some significant changes in network state*

use of such an event-based analysis can be understood from Figure 4, which shows the two-dimensional graph visualizer provided by uCAN. The graph shows a sample bandwidth test, plotting the results of the bandwidth(as red dots) and at the same time marking events on the same graph, which can in this case be produced by a SNMP trap message(as green dots). This sample graph indicates when packets were dropped at a router, in the context of a bandwidth test that indicates a low bandwidth. Hence it can be clearly understood how simultaneous analysis based on events can be useful in identifying problems.



Time (sec) = 61.000681, Description = SNMP_GET: ipInDiscards - router dropping

Figure 4: *The graph in the diagram shows two type of points. The red points on the graph indicate data values of bandwidth for the netperf test. The green points on the graph represent events reflected. For example, In this case, the first green point refers to an event sent from an snmp interface when there was packet drop in the router.*

## 3.3 Defining the Requirements for Collaboration

The particular requirements for real-time collaboration in uCAN are:

1. The analysis operations should be shared with others -to start with, there is a need for a collaboration framework for sharing interfaces and graphs in a Multi User Domain.
2. The collaborators should be able to view as well as control remote interfaces that drive tests and network monitoring operations initiated by others.
3. The initiator of an interface should have the option of sharing his interface in a collaborative session
4. Since there has to be way of controlling the collaboration, roles have to be defined so that a participant who initiates an operation has the power to take back control and to delegate control to someone else at any time.
5. Participants need to have knowledge of who are the other collaborators in the session.
6. Interaction should involve quick small updates rather than bulk transfers
7. It will make a useful contribution to provide an extendable collaborative framework for building shared widgets, so that new interfaces can be built for new operations and can be made collaborative.

### 3.3.1 Sharing Interfaces

As per the requirements, a general shared interface scheme can be thought of as given in Figure 5.

Figure 6 and 7 give a sample collaborative session where a snmp query session is shared between two users. The user who has initiated the SNMP query, shown by the interface in Figure 6, can decide to share or not share his interface updates at any point. When the updates are shared, the remote interface that other users see(Figure 7) gets updated, letting everyone see what this user sees.

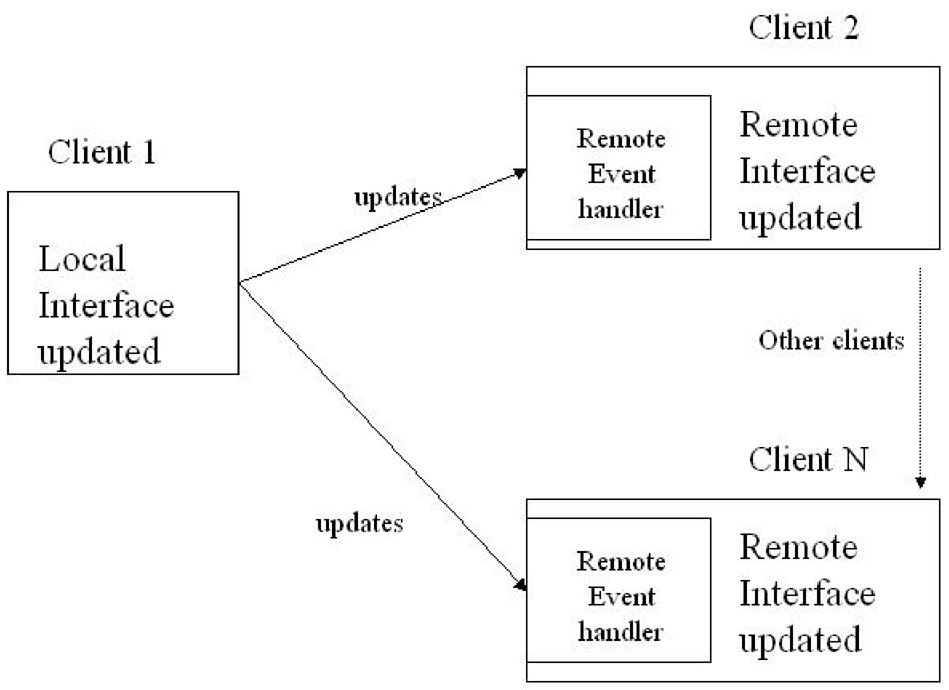
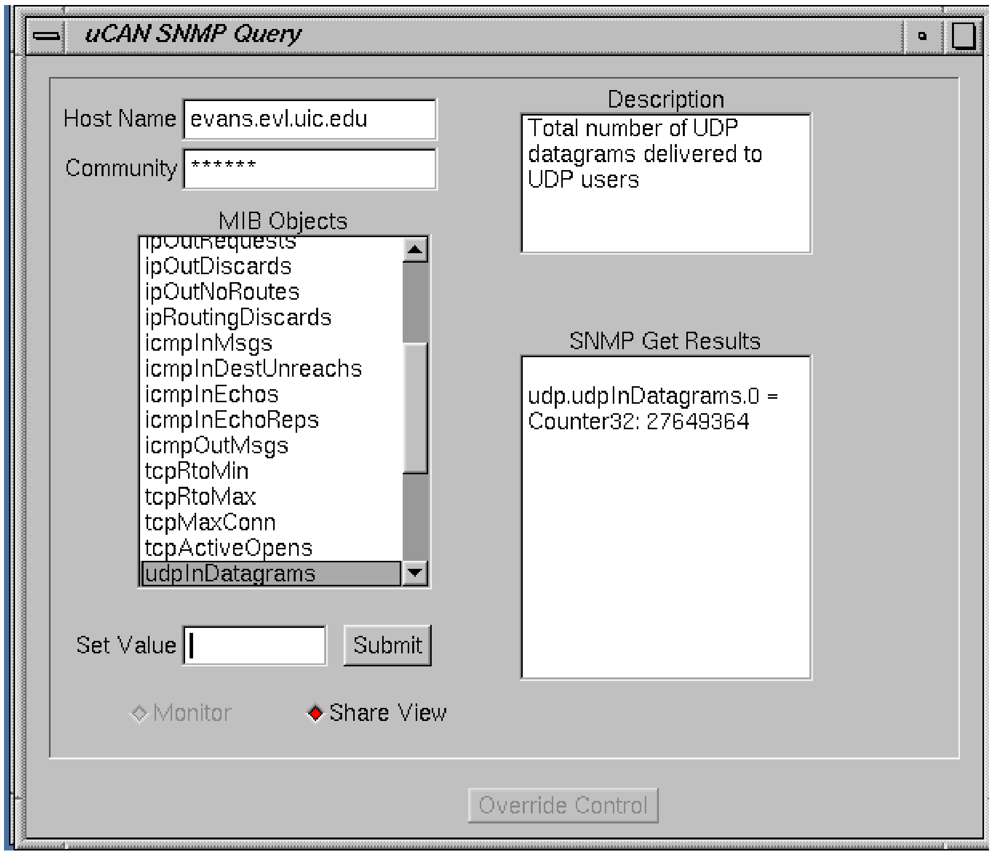


Figure 5: *The figure the basic idea behind a shared user interface in a collaborative session. Apart from local events, handles have to be written that uses high level events as recorded in user callbacks and handle events for interfaces on remote machines*



Host Name evans.evl.uic.edu

Community ******

Description

Total number of UDP datagrams delivered to UDP users

MIB Objects

ipOutRequests
ipOutDiscards
ipOutNoRoutes
ipRoutingDiscards
icmpInMsgs
icmpInDestUnreachs
icmpInEchos
icmpInEchoReps
icmpOutMsgs
tcpRtoMin
tcpRtoMax
tcpMaxConn
tcpActiveOpens
udpInDatagrams

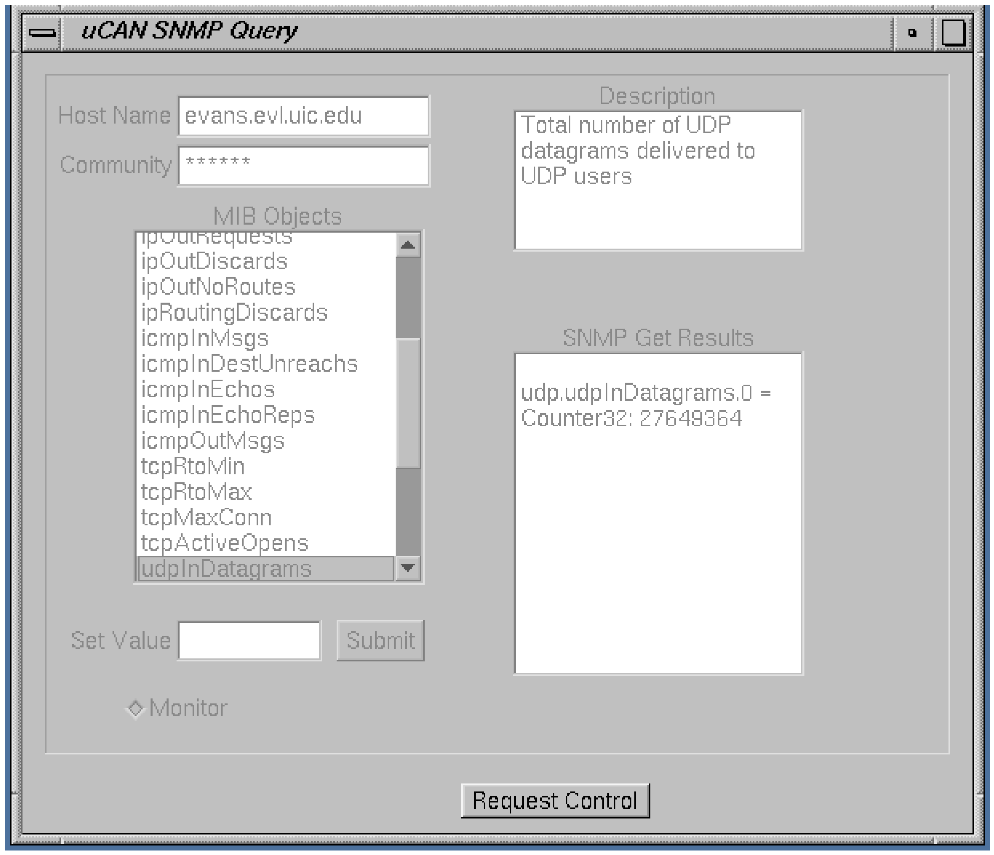
SNMP Get Results

```
udp.udpInDatagrams.0 =
Counter32: 27649364
```

Set Value

◇ Monitor ◆ Share View

Figure 6: *The figure shows an interface for snmp queries being shared with a remote user. The shareView can be toggled to share updates with others or not.*



Host Name: evans.evl.uic.edu
Community: ******

MIB Objects:
ipOutRequests
ipOutDiscards
ipOutNoRoutes
ipRoutingDiscards
icmpInMsgs
icmpInDestUnreachs
icmpInEchos
icmpInEchoReps
icmpOutMsgs
tcpRtoMin
tcpRtoMax
tcpMaxConn
tcpActiveOpens
udpInDatagrams

Description: Total number of UDP datagrams delivered to UDP users

SNMP Get Results:
```
udp.udpInDatagrams.0 =
Counter32: 27649364
```

Figure 7: *The figure shows the remote interface that appears for other users, when the query interface in Figure 6 is shared. The updates occur as is seen on the original interface. The user can also request control from the owner of the interface.*

# CHAPTER 4

# CAPTURING APPLICATION AND NETWORK STATE INFORMATION

This chapter encompasses the core components of the unified approach proposed. The focus is to explain in detail about the methodologies involved in obtaining performance data from an application and information about the network state. Details include how applications have been instrumented, performance monitoring can be done using an event based method, visualize results collaboratively, describe methodologies for ubiquitous SNMP queries and active tests. Each section defines the objectives of the concerned module and the design to acheive the objectives.

## 4.1 Application Performance Monitoring

High performance systems and other complex networked applications have several types of flows that interact with the network. To understand and tune the performance of application, a knowledge of the performance of the application is essential. So, most of the interactions of applications must be captured while an application is running. Apart from obtaining information, there should be an easy way to understand the informaion obtained and characterize performance. These needs help define the objectives of application performance monitoring in uCAN.

### 4.1.1 The Objectives

1. Instrumenting applications with performance monitoring routines should give all required information needed to understand performance of the different data streams in an application

2. The instrumentation should be implemented in a transparent manner and control should be handed to the application programmer to gather statistics anywhere in the application
3. There has to be a way of analyzing performance later in time
4. Finally, there is a need for visualizing performance from any convenient location

### 4.1.2 Instrumentation: What is Measured

A typical high performance application like tele-immersion has several streams and there is a need for a standard representation of performance across all streams and applications. uCAN measures performance using the Universal Format for Logger Messages (ULM)(31). The latency and jitter measurements require sending and receiving ends to be time synchronized, which may be done with a protocol like the Network Time Protocol(46). A sample log stream is given below:

```
TIME=955664372.441101 SELF_IP=131.193.48.163 REMOTE_IP=131.193.48.164 SELF_PORT=
9977 REMOTE_PORT= 1811 STREAM_INFO= 131.193.48.164_AVATAR_SERVER COMMENT=
TRACKER_UDP MIN_LAT= 0.000318 AVG_LAT= 0.002742 MAX_LAT= 0.069904 INST_LAT=
0.000734 JITTER= 0.001046 MIN_IMD= 0.000093 AVG_IMD= 1.902742 MAX_IMD= 480.669540
INST_IMD= 0.000715 AVG_RBW= 11.573473 INST_RBW= 44754.160720 AVG_SBW= 401.712774
INST_SBW= 733308.778808 BURSTINESS= 30118.629172 TOTAL_READ= 8258 TOTAL_SENT=
306604 PACKETS_READ=376 PACKETS_SENT=7860
```

A brief account of the parameters measured along with their units of measurement and the formulae used is given as follows:

$Latency(oneway) = T_s - T_r$

where $T_s$ is the Time recorded at the sender's end and $T_r$ is the time recorded at the receiving end. The unit of measurement is seconds. The minimum (MIN_LAT), average (AVG_LAT), maximum (MAX_LAT) and instantaneous (INST_LAT) latencies are calculated and provided for further analysis.

$Jitter = E[(L_i - E[L])]$

where $E$ is the Expectation of a data set, $L$ is the set of 100 most recent instantaneous latency samples and $L_i$ is the instantaneous latency. The unit of measurement is seconds.

$InterMessageDelay = T_{i+1} - T_i$

where $T_i$ and $T_{i+1}$ are instances of two consecutive messages received. The unit of measurement is seconds. Like latency, the minimum (MIN_IMD), average (AVG_IMD), maximum (MAX_IMD) and instantaneous (INST_IMD) inter message delays are calculated and recorded. The unit of measurement is seconds.

$Bandwidth = [\frac{\delta d}{\delta t}]$

where $\delta d$ is the data in bytes, received/sent over a time $\delta t$. The unit of bandwidth is bytes/sec. AVG_RBW, INST_RBW, AVG_SBW and INST_SBW represent the average and instantaneous values of read and send bandwidth, respectively.

$Burstiness = E[(B_i - E[B])]$

where $E$ is the Expectation of a data set, $B$ is the set of 100 most recent instantaneous bandwidth samples and $B_i$ is the instantaneous read bandwidth. The unit of measurement as in bandwidth is bytes/sec.

In addition to the performance parameters, some more variables that are useful in identification of streams and bottlenecks are provided. The time field (TIME), the IP numbers and the ports of the hosts between whom the data stream is sent(SELF_IP, SELF_PORT, REMOTE_IP and REMOTE_PORT), the information about the data stream instrumented(STREAM_INFO) and a special event-comment (COMMENT) field which can be used by the application developer to provide event information in the application. These events can also be further used, as illustrated in later chapters to help analyze network measurements in the context of the application.

### 4.1.3 The Architecture

The architecture of performance monitoring in uCAN is based on the architecture implemented in CAVERNsoft. CAVERNsoft provides performance monitoring routines that can easily turned on in an application using CAVERNsoft's networking modules. The performance data collected from applications can be used either for plain output for users, or logged to a file for further analysis. All CAVERNsoft networking modules have a sub-module that instruments network streams and measures network parameters. The applications can hence get performance information transparently, without having to instrument or include any measurement routines. The basic architecture is explained in Figure 8.

### 4.1.4 Visualizing Performance - QoSIMoTo

The log streams in the format as shown above have all the information a person would need to analyze performance. But going through several hundreds of lines of log entries and trying to form a mental picture of the performance becomes a very tough task. The analysis is highly enhanced with the use of QoSIMoTo, a CAVE based 3D dimensional tool developed at the Electronic Visualization Lab to visualize use of network Quality Of Service by high performance applications like tele-immersion. QoSIMoTo helps analyze multiple streams in an application simultaneously and provides the user interaction mechanisms to scale, play, stop

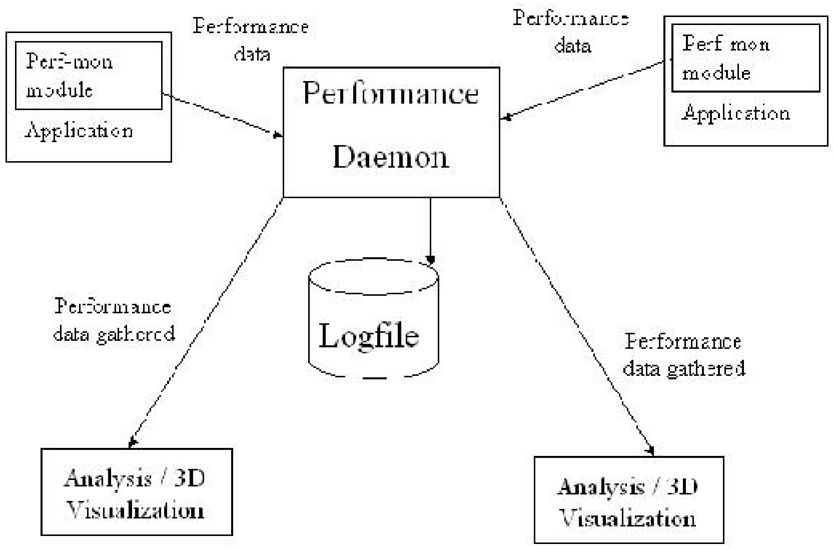
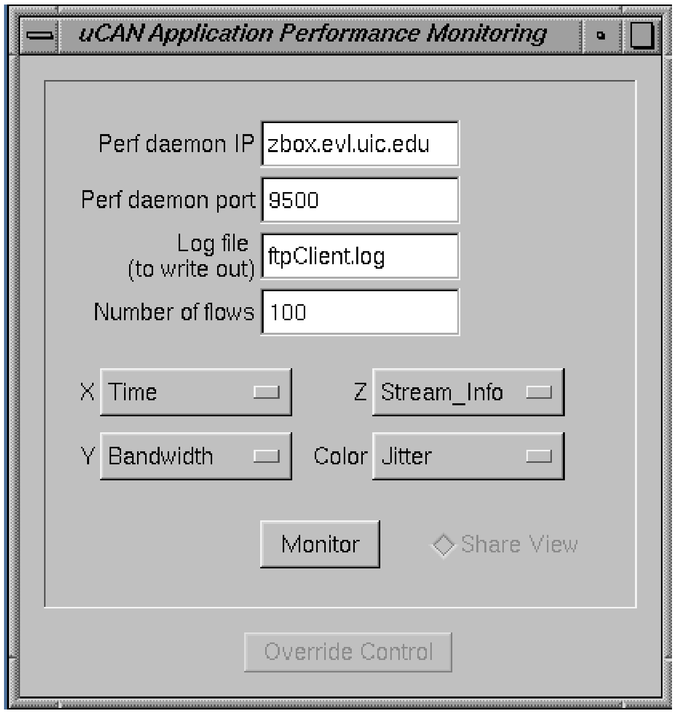


Figure 8: *The methodology for performance data collection and analysis. Any number of analysis and visualization clients can connect to the performance daemon and get the performance data gathered from applications.*

and rewind the visualization. Both real-time and non-real-time analysis can be done, as it can listen to data streams and visualize them as and when they are delivered by a performance daemon and also log the collected data for analysis at a later time.

The interface to configure QoSiMoTo and a sample visualization that QoSIMoTo can provide are given in Figures 9 and Figure 10


uCAN Application Performance Monitoring

Perf daemon IP zbox.evl.uic.edu

Perf daemon port 9500

Log file (to write out) ftpClient.log

Number of flows 100

X Time

Z Stream_Info

Y Bandwidth

Color Jitter

Figure 9: *The diagram shows the configuration interface for QoSIMoto. The location of the performance daemon, the logfile, the size of visualization and the axis parameters are the variables that can be controlled.*

## 4.2 A Framework for Active Testing

Active testing produces highly accurate results for measurements and there are many standardized techniques and tools publically available. Active tests form an important part of network analysis, since after the detection of a bottleneck in a network, accurate measurement of the network is always required to confirm the location of a performance problem in the network.

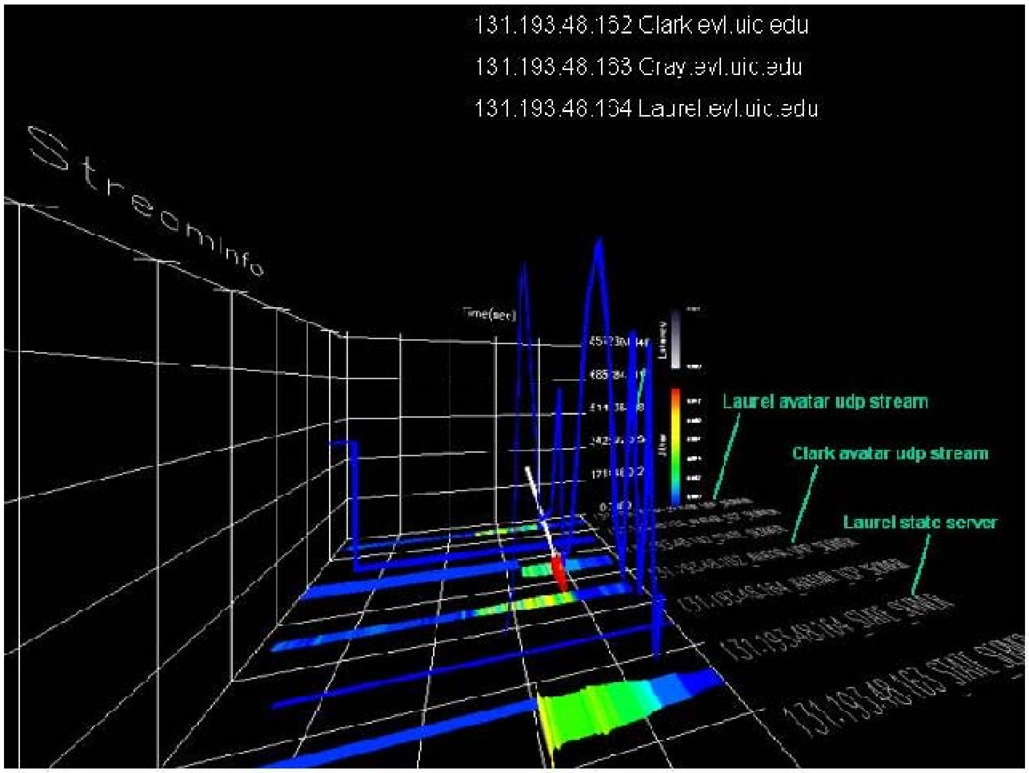


Figure 10: *QoSiMoto's 3D visualization of performance obtained by instrumenting a tele-immersion server with multiple streams, which can be visualized at the same time, leveraging 3D. The latency of each stream is plotted against time, while color is used to represent jitter*

### 4.2.1 The Objectives

1. Make testing possible from any machine, without having to run scripts manually on each machine
2. The test results should be shared by other clients who are interested too, bringing the test analysis to a multi-user domain
3. There has to be a way of analyzing measurement results later in time
4. There is a need for visualizing test results

5. The visualization should be configurable by the user, to adjust to the range of data measured

6. All of the above should be reusable standard features of a framework that a user can use again for a new test

### 4.2.2 Pieces of the Framework

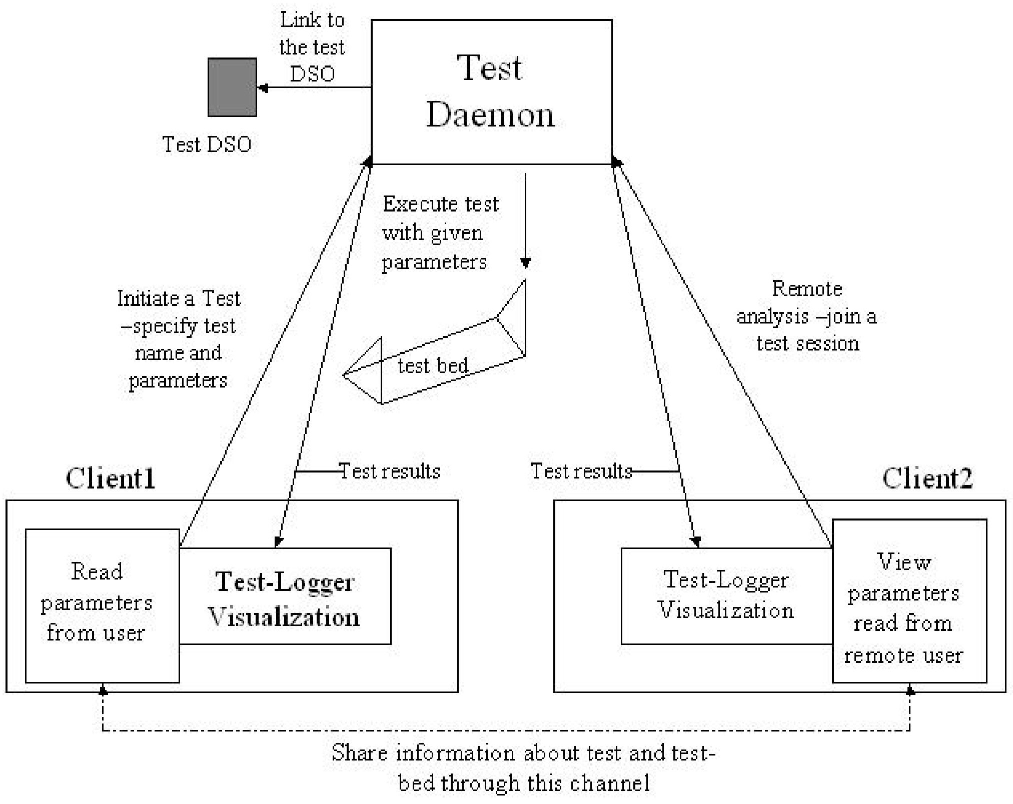
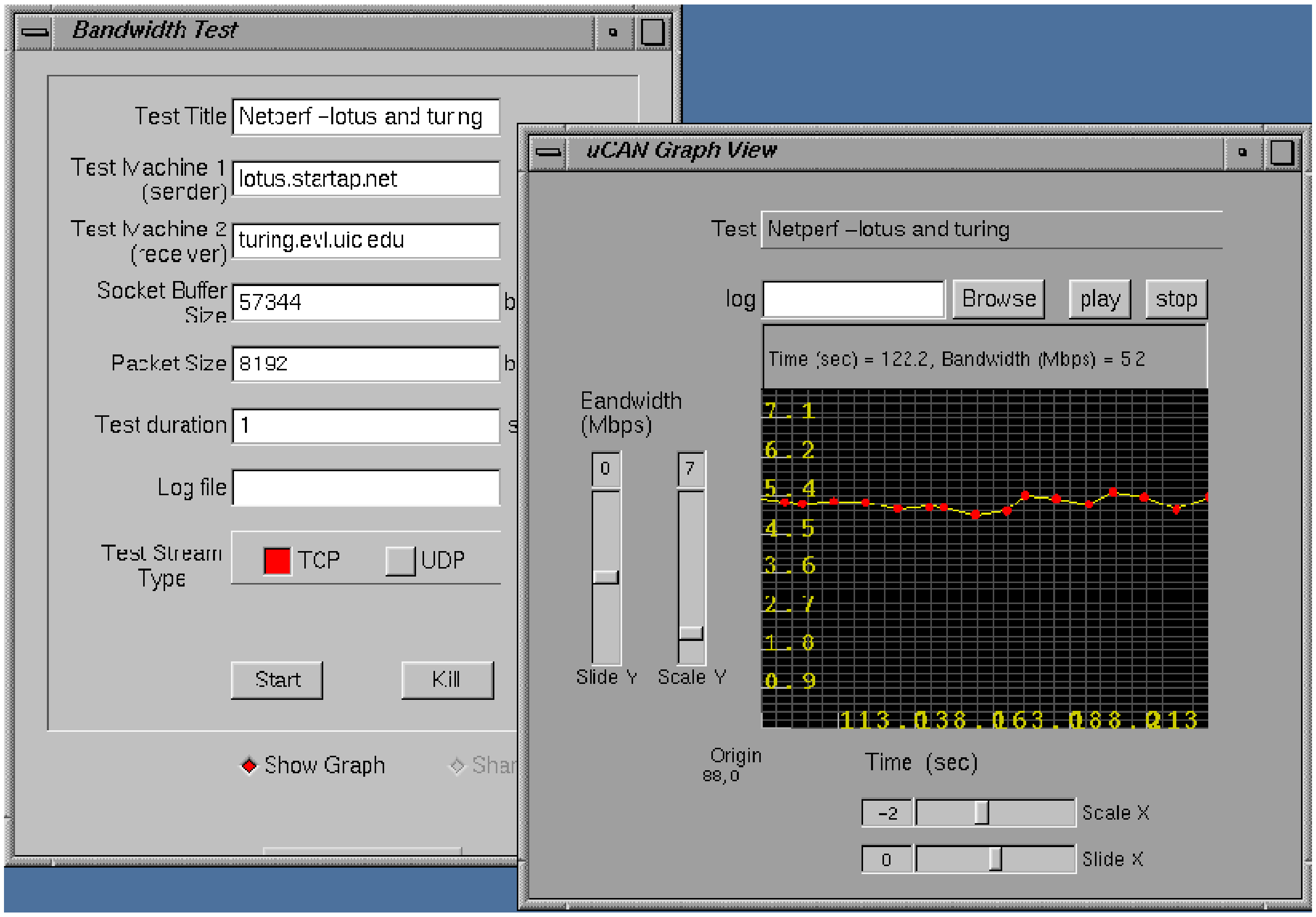

Link to the test DSO
Test Daemon
Test DSO
Execute test with given parameters
Initiate a Test –specify test name and parameters
test bed
Remote analysis –join a test session
Client1
Test results
Test results
Client2
Read parameters from user
Test-Logger Visualization
Test-Logger Visualization
View parameters read from remote user
Share information about test and test-bed through this channel

Figure 11: *The framework for testing - once a test is initiated, a test logger process is started to log and visualize the test. The test daemon executes the specified test on the test bed that should be preset. The results are sent to all clients*

The architecture of a typical uCAN test session is shown in Figure 11. The first part consists of the interface for the user to configure a test and initiate it. Once a test is initiated by the user, a seperate process for logging and visualization is spawned. The parameters sent to the test daemon include what type of test is to be run and the test parameters encoded as a set of strings. The test daemon links to the test which should be written as a Dynamic Shared Obbjet(DSO). Using Dynamic Shared Objects is a modern and popular mechanism to to build a piece of program code in a special format for loading it at run-time into the address space of an executable program. A good explanation of Dynamic Shared Objects and support specification is given in (47) This would help the new user to fit in his own tool reusing the existing framework and not worry about recompiling the uCAN distribution. The test daemon will load the test during runtime, given the name of the DSO to link to. The test results are packed and sent to all the clients connected. The test daemon iteratively executes the test and reflects the results to all clients connected, until a *KILL* command is read from one of the clients. The daemon also listens to new connections and dynamically adds them after each iteration of the test. So a new test client essentially joins a tests session to listen to the latest test results.

The test logger is an interface between the user and the test daemon. It reflects user's commands to the test daemon and visualizes the test results coming from the test daemon in real time. A generic 2-dimensional graph widget is provided along with set of utilities. The graph widget allows the user to plot any two parameters, adjust the scale values, shift the origin to meet a desired range of data, log the test results for later analysis and also provides data


Bandwidth Test

Test Title: Netperf -lotus and turng

Test Machine 1 (sender): lotus.startap.net

Test Machine 2 (receiver): turing.evl.uic.edu

Socket Buffer Size: 57344 b

Packet Size: 8192 b

Test duration: 1 s

Log file:

Test Stream Type: 🟥 TCP ☐ UDP

uCAN Graph View

Test: Netperf –lotus and turing

log:

Time (sec) = 122.2, Bandwidth (Mbps) = 5.2

Origin 88,0

Figure 12: *Figure showing a netperf test between two machines. The test can be configured using the interface. The test graph shows a plot in real time as the netperf test is executed on a remote test daemon.*

displays with interactive mouse over motions. Figure 12 shows an interface for driving netperf tests and the graph widget for visualizing results from such active tests.

## 4.3 Ubiquitous Network Management

Network management has become an important part of trouble shooting and maintaining most networks today. SNMP is the most popular protocol used for exchange of network management information between network devices, owing to its portability and ease of use.

### 4.3.1 The Objectives

1. Any node on the network should be able to access network management information, restricting access with the SNMP community string
2. Visualizing a changes in a variable that is monitored
3. Allowing ease of changing the Management Information Base(MIB) to a desired enterprise MIB in the future
4. Flexibility of using a different SNMP library

### 4.3.2 What can be Obtained

SNMP provides different types of information organized as variables and groups in the Management Information Base (MIB), a virtual database present in all machines on a network. MIB-II, defined in detail in (48), is the most standard MIB present on all devices. A description of SNMP's functions, practical aspects of implementation and the operations of the protocol are dealt in detail in (49). Highlights of each type of group and the type of information obtained from them can be listed as below:

1. The **systems** group consists of general information about the managed system -like its description, location, contact person and the amount of time the system has been up since it was last reinitialized.
2. The **interfaces** group consists of generic information about the physical interfaces of the entity, including configuration information and statistics on the events occuring at each interface. Examples of variables that can be obtained are the type of the interface, the protocol it follows, the physical address, operational status, current capacity in bits per

seond, the number of octets, multicast, unicast packets transmitted or discarded and the Maximum Transmission Unit(MTU) size, etc.

3. The **address translation** group consists of a single table, each row of which corresponds to one of the physical interfaces of the system. The network addresses and the corresponding physical addresses translated into are provided.

4. The **ip** group consists of information relevant to implementation and operation of IP at a node. Some variables that can be obtained include the IP routing table, the routing protocols for each entry, the number of packets forwarded, discarded, delivered, received, fragmented, etc.

5. The **icmp** group consists of information relevant to the implementation and operation of the Internet Control Message Protocol(ICMP), used for transmitting messages between routers and hosts, providing feedback about problems in the communication. The icmp groups gives various counter values of the different types of icmp messages sent

6. The **tcp** group consists of information relevant to the implementation and operation of TCP at a node like the values of timers, number of segemented, retransmitted packets, a tcp connection table with the details of each coninection, etc.

7. The **udp** group consists of information relevant to the implementation and operation of udp at the node like the number of input and output datagrams, errors and a table with the addresses and ports of connections made.

8. The **egp** group has information relevant to the implementation of External Gateway Protocol (EGP), with a table containing information about each of the neighbour gateways known to this entity.

9. The **snmp** group consists of information relevant to the implementation and operation of snmp at the device, giving the nunmber of requests coming in, the errors, the number of traps set, etc.

The MIB variables and their Object Identifiers are written in a configuration file, that is loaded by a SNMP query client. The configuration file can be changed to load variables from another MIB other than MIB-II(which is currently the only MIB supported in uCAN), thus allowing flexibility of using the same interface for different MIB's. This can be a very useful feature since in many circumstances, enterprise MIB's other than MIB-II come in handy to obtain specific details about network devices and interfaces. uCAN currently has a database of a sample set of the variables from each group in MIB-II. Figure 14 shows a SNMP query interface with a query being made to obtain the value of the system description from MIB-II. The database can be easily extended to support all other MIB-II variables.

### 4.3.3 Ubiquitous SNMP Query

This section explains the unique feature of uCAN's SNMP query. Traditionally SNMP queries are made from the machine that the user runs his query program. Network managers will have to authorize every machine that should be allowed to query a particular device such as the router. This may not be very practical since it can be a potential security threat to distribute access to several machines. Instead if one machine can be made to query all machines on a network, a query daemon can be run on that machine to reply to queries coming from other machines. This is acheived in uCAN by implementing the SNMP query module as a Remote Procedure Call(RPC) server and any other machine using uCAN can make a query to the RPC server, which acts as a bridge. So, the SNMP query interface in uCAN on any machine reads the MIB variables and the corresponding Object ID's from a configuration file and sends the query as a rpc call to the uCAN SNMP Daemon. The uCAN SNMP Daemon interacts with the SNMP Daemon that resides inside the network device thats is queried and returns the result to the user. This transaction remains transparent to the user, when he submits a SNMP query. The above description can also be understood clearly from Figure 13.

CAVERNsoft's RPC modules have been used for the implementation of the SNMP Query Daemon. The SNMP Get method is managed as a call on the CAVERN RPC server, which can be remotely called by SNMP query clients. The implementation of SNMP queries was done using the NET-SNMP library(10) which is an open source library also available as a standard package provided with RedHat Linux distributions.

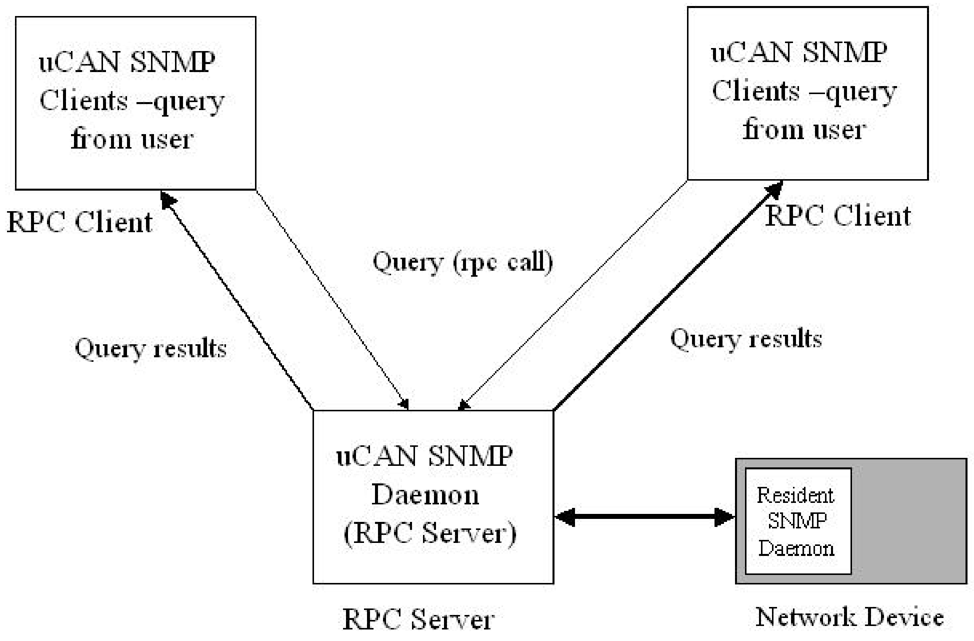
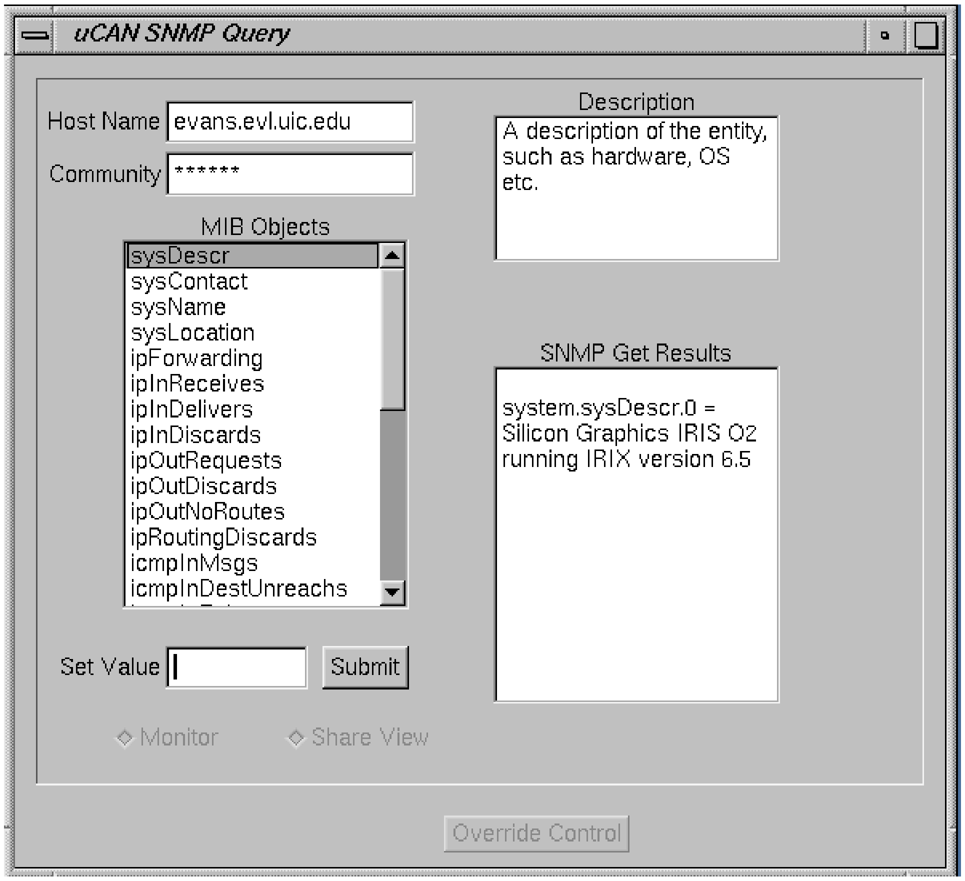


Figure 13: *The framework for SNMP Queries. Query clients can be on any machine and the actual query is made by the uCAN SNMP Query Daemon, which is transparent to the user.*

## 4.4 Summary

The core network analysis techniques implemented in uCAN has been described. This chapter's goal was to cover the objectives of testing, network management and performance monitoring modules, the architecture and the unique features of each module. The next chapter introduces a new capability that would add a new dimension to all these capabilities explained so far.



Host Name: evans.evl.uic.edu

Community: ******

Description: A description of the entity, such as hardware, OS etc.

MIB Objects:
- sysDescr
- sysContact
- sysName
- sysLocation
- ipForwarding
- ipInReceives
- ipInDelivers
- ipInDiscards
- ipOutRequests
- ipOutDiscards
- ipOutNoRoutes
- ipRoutingDiscards
- icmpInMsgs
- icmpInDestUnreachs

SNMP Get Results:

```
system.sysDescr.0 =
Silicon Graphics IRIS O2
running IRIX version 6.5
```

Set Value:

Figure 14: *The interface for making SNMP queries. The system description is being queried (query submitted with a single click on the variable) The Query interface itself can be used on any machine to query any other machine on the network. Queries are bridged through a query dameon. The interface shows a sample set of queries for MIB-2 variables.*

# CHAPTER 5

# COLLABORATING IN A VIRTUAL WORKSPACE

As operations are usually driven by user interfaces, uCAN aims to help with the process of helping build shared interfaces suitable for the collaborative environment. This chapter follows the requirements of collaboration described in chapter 3, describing the framework design for building shared interfaces, the event sharing methodology involved and how roles can be defined in a collaborative environment for analyzing networks.

The objective of collaboration is to build a virtual desktop space for people involved in anlayzing networks. Amongst other similar systems, VNC(38) provides higher level solutions for desktop sharing, providing a more general solution. VNC still does not have Multi User Domain support and has high bandwidth requirements since transfer of frame buffer information requires very high throughput. But the particular requirements of collaboration in uCAN include multi user support, quick small updates and also the ability to exchange control between participants, so that only one person remains in control over an interface at a time. For these reasons, VNC did not qualify as an ideal tool for this type of collaboration.

## 5.1 Building Collaborative Interfaces

In general controlling network tests and monitoring through user interfaces is more convenient than using scripts and manually editing options each time. Among user interface libraries available for programmers, FLTK(2) has been a popular open source library for building user interfaces very quickly. uCAN uses FLTK to build interfaces for controlling tests, adding new tests, configuring visualization parameters for performance monitoring in QoSIMoTo and also for SNMP queries, as can be seen from snapshots in the previous chapter.

The requirements of collaboration as discussed include providing access to analysis operations to collaborators on a remote site. The first step was implementing collaborative interfaces that can be shared and controlled in a Multi User Domain. Sharing interfaces means letting a remote person control an operation with the permission of the user who initiated the operation.

The sharing of interfaces leads to a need for a event management system to transmit events from one interface to another and still give the user, control over when the events should be passed. A challenge for implementing such a scheme was to design the event management framework so that it can be re-used for building new interfaces for sharing in a collaborative environment.

### 5.1.1 Networking and Database Architecture for Event Sharing -Implementation

The networking and database architecture is based on CAVERNsoft's object-oriented design for building collaborative environments(1). Figure 15 gives a simple diagram of the architecture for sharing events in a multi user domain. The interfaces and graphs used for analysis in a collaborative session form the event clients. The user in control sends updates of the changes in widget values to the event server in terms *keys*, which correspond to an entries in the databases of event-clients and the event-server. A collection of keys are stored under a *path* in the corresponding databases. uCAN makes use of this architecture to store widget updates as keys inside a unique path for each interface. The event server reflects the incoming key updates to all other event database clients, thus updating their local databases for the widget values.

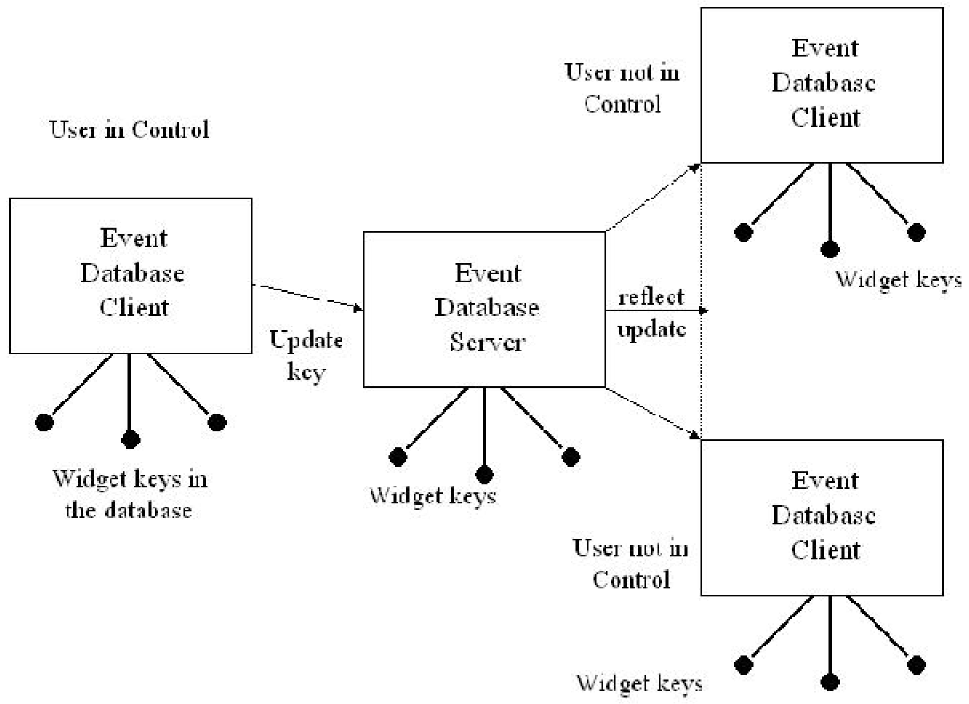


Figure 15: *The networking and database framework for event sharing -the updates in each widget from an interface in control(an event client), is reflected to other clients, through the event server*

The user in control of the interface has control over sending the event updates whenever he decides to. The simplest way to send updates will be during callbacks set for the widgets. This can be done by calling this api

**eventClient** $\rightarrow$ **put(pathname, widgetname, data, datasize)**

where *eventClient* is the event database client used for collaboration, *pathname* is the name of the path for a particular interface, *widgetname* is the name of the widget which is used to register a widget with the database (this is explained in the following section) and the *data* of size given by *datasize*, is sent as the encoded event update. For example, input widgets can send the string in the widget as data, while buttons can send a SET or an UNSET state as defined in the event client C++ class.

Sharing an interface is a choice left to the user. Collaborative interfaces have a *share* state which can be toggled. If the *share* state is turned on, all updates from the client reach the server. Otherwise only local updates are done. (The *share* state is represented in Figure ¡¿ by the *share view* check button which can be toggled)

The updates from remote clients, coming from the server, can be read by calling this api

**eventClient** $\rightarrow$ **process**()

To understand the implementation of the event client, a knowledge of how the event server works is helpful. The algorithm in Figure 16 explains the inner-workings of the event-database server, which is an implementation of the CAVERNsoft's database server. The event-database server has a database look up of different keys for the widgets registered for each interface. Any new update from a client is reflected to all other clients. Clients can also request for the latest value of a key(usually state value of a widget). The event clients listen to messages which have encoded information about an event from a remote interface. The event handler is triggered to react to the event(which is detailed in the next subsection). Another implementation issue is, giving a late joiner in a collaborative session, all current values of the keys corresponding to the values of widgets in an interface. This is done by having a key giving the list of keys(for the widgets) in an interface and updating that list key when a widget is updated for the first time. A new client will therefore fetch the value of this list key when it enters a collaborative session.

**Implementation algorithm for the event-database server**

**Given: A list, L of N event clients**

**Objective : To handle incoming event updates and requests and manage new clients**

```
// PUT_MESG is the command to reflect the update to all clients
// FETCH_MESG is the command to obtain the latest value of a key
in the database of the server
```

```
01. while (forever) do
02. begin
03.     check for any new clients
04.      if a new client has joined
05.         add the new client to L
06.      endif
07.     for i ←1 to N do
08.     begin
09.         Read from the client_i for any new message
10.         if there is a new message and if it is a PUT_MESG
11.            then update database, reflect updates to other clients
12.         endif
13.         else if the new message is a FETCH_MESG
14.            return the latest value of the corresponding key
15.         end elseif
16.     end
17. end
```

Figure 16: *Algorithm explaining the implementation of the event server in a collaborative session*

### 5.1.2 Event Handling, A Subject-Observer Design Pattern

The subject-observer design pattern is used in object-oriented design to establish an one-to-many dependency between objects so that when one object(the subject) changes state, all its dependents are notified and updated automatically(50). The event handler in uCAN is implemented as an observer that is attached to the event client(the subject). Any new update about a key that the event client receives is given to the event handler for a reaction on the local interface. Figure 17 explains the subject-observer relation for the event handler and Figure 18 describes the way in which the event handler updates an interface of events.

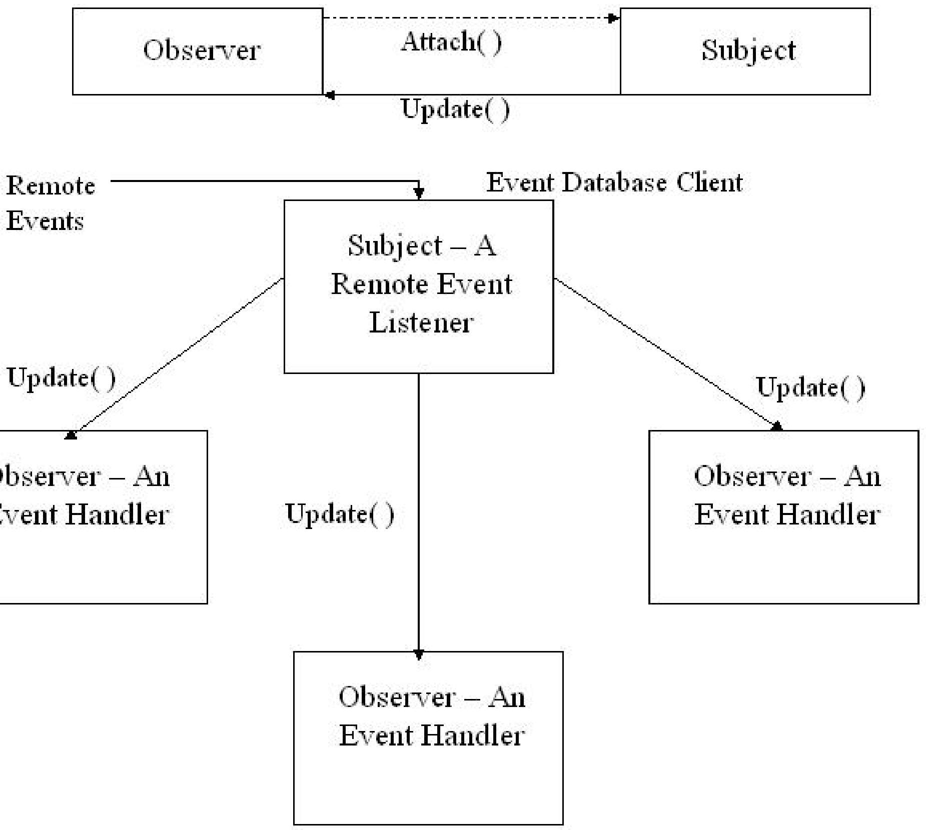
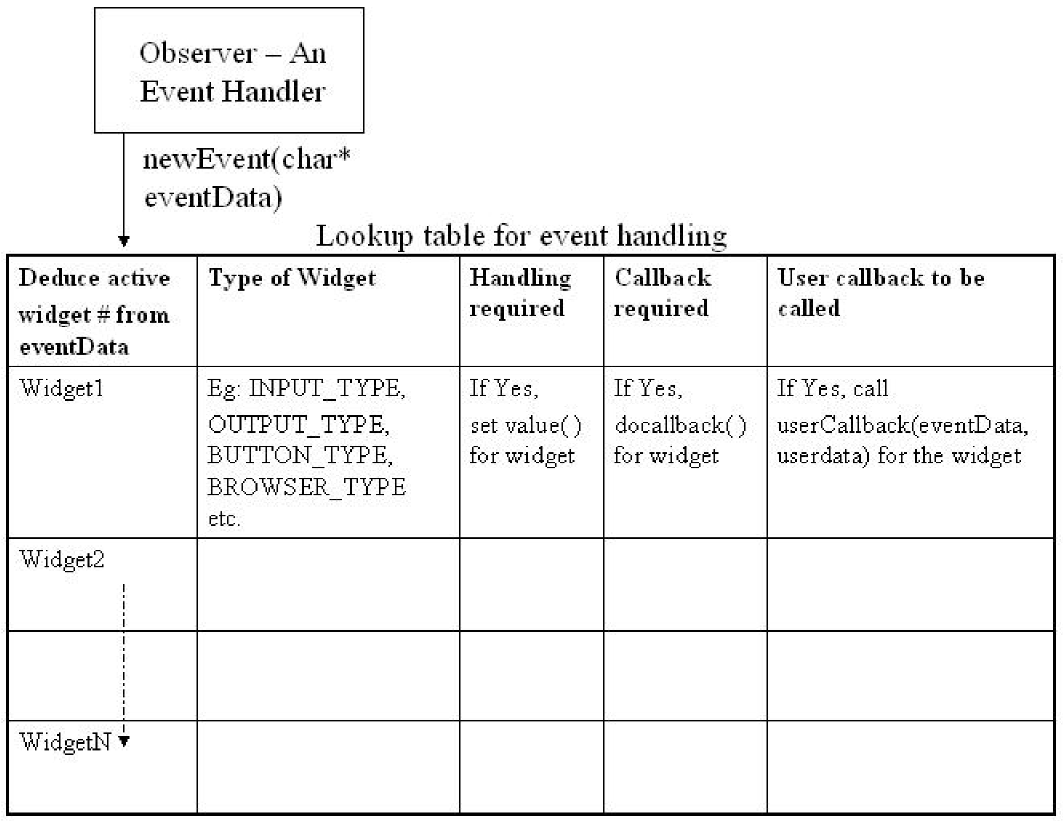

Observer
Attach( )
Subject
Update( )
Remote Events
Event Database Client
Subject – A Remote Event Listener
Update( )
Update( )
Update( )
Observer – An Event Handler
Observer – An Event Handler
Observer – An Event Handler

Figure 17: *The event handler and event client are implemented using the subject-observer design pattern. A new event read by the event client automatically triggers the update() call of each observer(event handler)*

A widget can be added to the lookup table of the event handler by the following api:

**eventHandler → manage(keyname, widget, handlingRequired,**


Observer – An Event Handler
newEvent(char* eventData)
Lookup table for event handling

| Deduce active widget # from eventData | Type of Widget | Handling required | Callback required | User callback to be called |
|---|---|---|---|---|
| Widget1 | Eg: INPUT_TYPE, OUTPUT_TYPE, BUTTON_TYPE, BROWSER_TYPE etc. | If Yes, set value( ) for widget | If Yes, docallback( ) for widget | If Yes, call userCallback(eventData, userdata) for the widget |
| Widget2 | | | | |
| | | | | |
| WidgetN | | | | |

Figure 18: *The diagram describing the lookup table in a event handler and how updates are done whe new data comes in to the client*

**typeOfHandling, do_callback, userData, userCallback**)

The *manage* api adds the widget instance(*widget*) with the corresponding(*keyname*) in a lookup table. The type of handling(*typeOfHandling*) depends on the nature of the widget. There are predefined types for standard widgets like the Input, Output, Button, Check Button, Radio Button, Browser types and Choice Menus. *handlingRequired* is a flag that should be turned on if the handling is to be done by the event handler(which manually sets values for these widgets, by reading the data assuming its in a default format). The callback of the widget can also be called from the event handler. This is made possible by the good object oriented design of the FLTK library, wherein all types of widgets are inherited from an abstract widget class, where most functions are virtual and the appropriate methods of the derived types are called. The user is also given freedom to implement his own handling to an event, by providing a *userCallback* which is called with the *userData* provided and the data(encoding the event update) got from the client.

As for graphs in a collaborative environment, they are initiated for monitoring at one site and are generated at remote sites in the same way. So each client is in control of the graphs at their end and can interact with the visualization without any control transfer.

## 5.2 Implementing Roles for Control in Collaboration

The previous section deals with how collaborative interfaces have been implemented. Since network analysis can be a sensitive issue in terms of control, there has to be a way of allowing the person initiating a network operation have control over the operation at anytime. Hence control states and roles have to be defined.

A participant in a collaborative session who initiates (or had initiated the interface before entering a session) is termed as the *owner* of the interface. Any participant in a session can request control over an interface from the owner and control can be granted or denied by the owner. The owner of the interface, if he grants control to another participant, can override others and take back control at any time. Hence this places the control of an operation to the person who initiates it, while at the same time, the owner can also let the operation be controlled by others and possibly gain from the expertise of the collaborator.

### 5.2.1 Implementation

uCAN has an inbuilt default control exchange in the event handler of the uCAN event client. This can be used by setting a widget (usually a button to request control as shown in Figure 14)as a *CONTROL_TYPE* widget. In general all shared interfaces in uCAN have two modes - *IN_CONTROL* and *NOT_IN_CONTROL*. As they might suggest, the user who initiates an operation is by default *IN_CONTROL* and when an interface is shared, it appears on remote sites in the *NOT_IN_CONTROL* mode, where widgets are disabled and other users have no control over the interfaces. FLTK allows a collection of widgets to be used and referred to as a group. uCAN makes use of this to defined a *activeGroup* and an *inactiveGroup* of interfaces in the event handler. So, the freedom is left to the user to set any number of widgets as part of an *activeGroup*, which is inactivated for users who are not in control or vice versa by defining widgets as part of the *inactiveGroup*. The *setActiveGroup* and *setInactiveGroup* api's are to be used for this.

The control transfer can be represented as an event diagram as in Figure 19.

Figures 20 and 21 illustrates the control transfer in a collaborative session

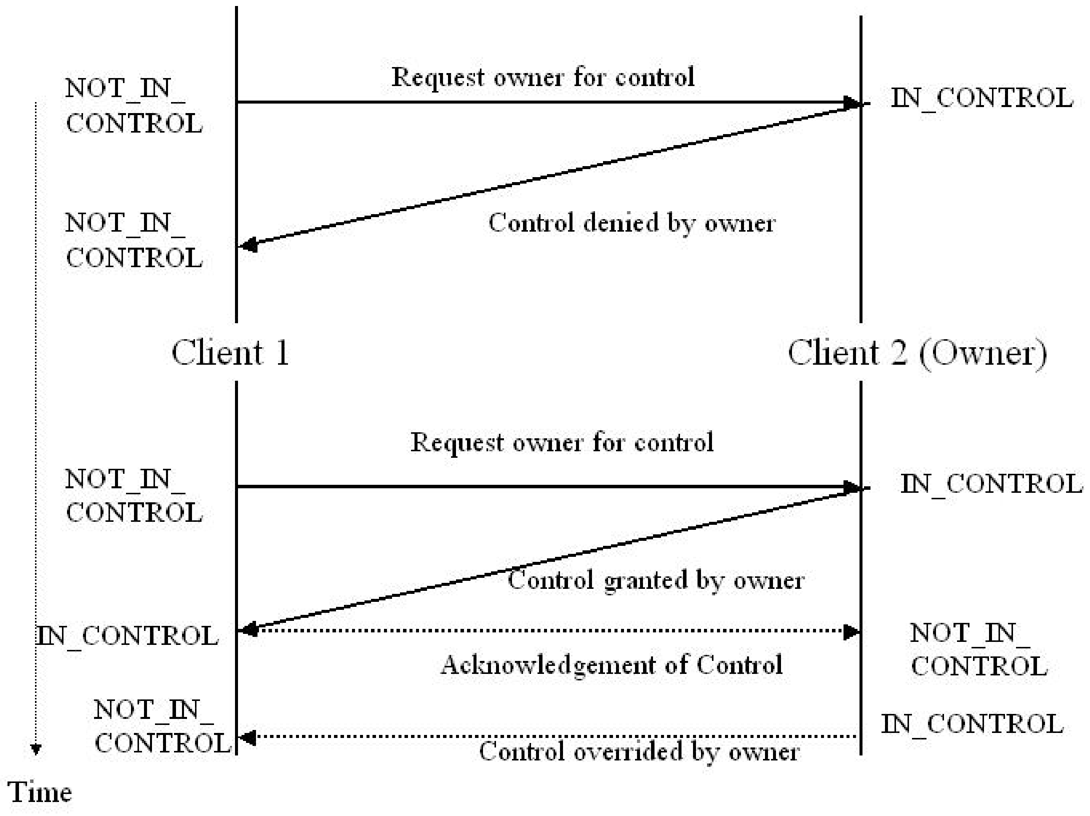
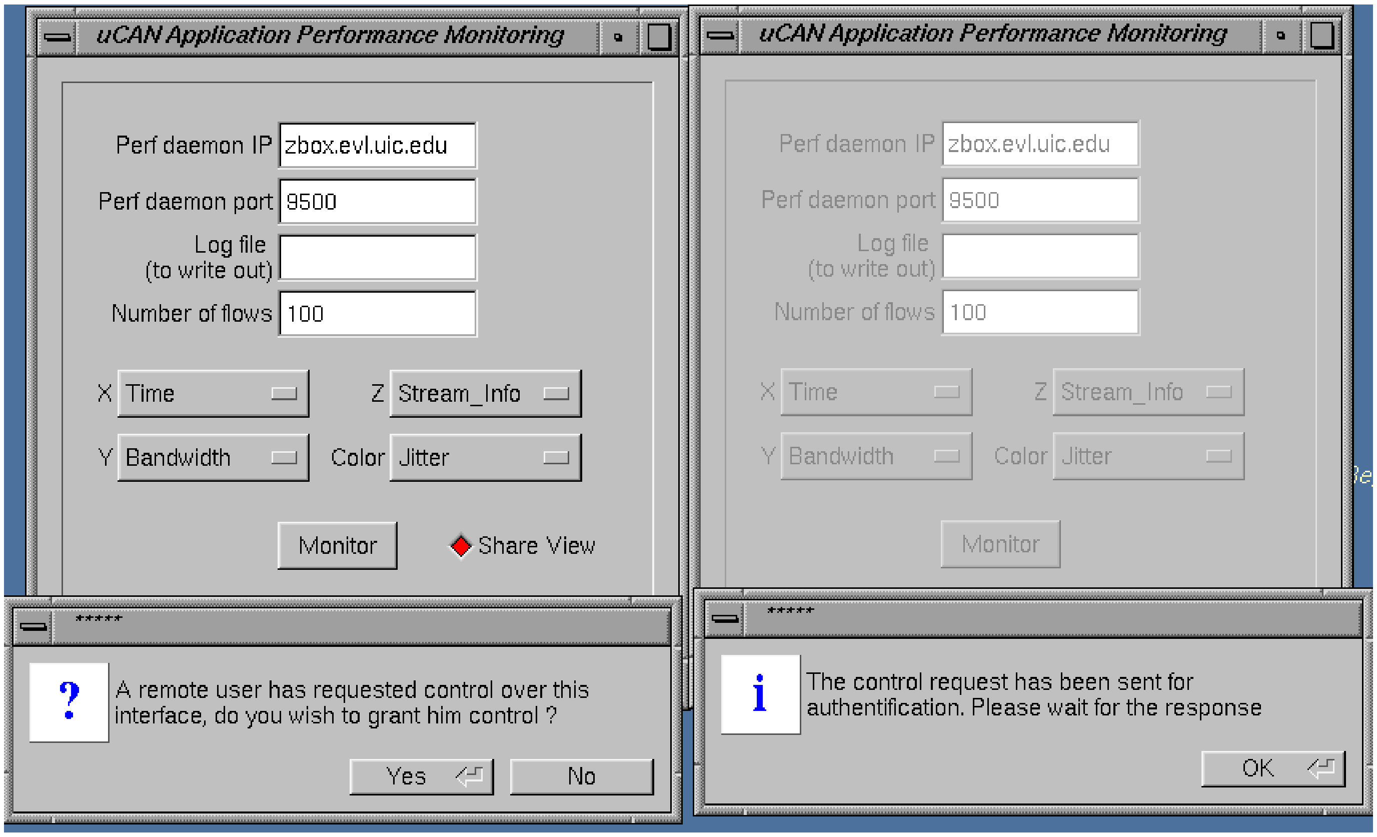


Figure 19: *A control flow diagram represented as events - two clients are represented and both control denial and grant cases are shown. In the diagram, the two states - IN_CONTROL and NOT_IN_CONTROL are activated after the corresponding message reaches the client.*


uCAN Application Performance Monitoring

Perf daemon IP: zbox.evl.uic.edu
Perf daemon port: 9500
Log file (to write out):
Number of flows: 100
X: Time
Z: Stream_Info
Y: Bandwidth
Color: Jitter

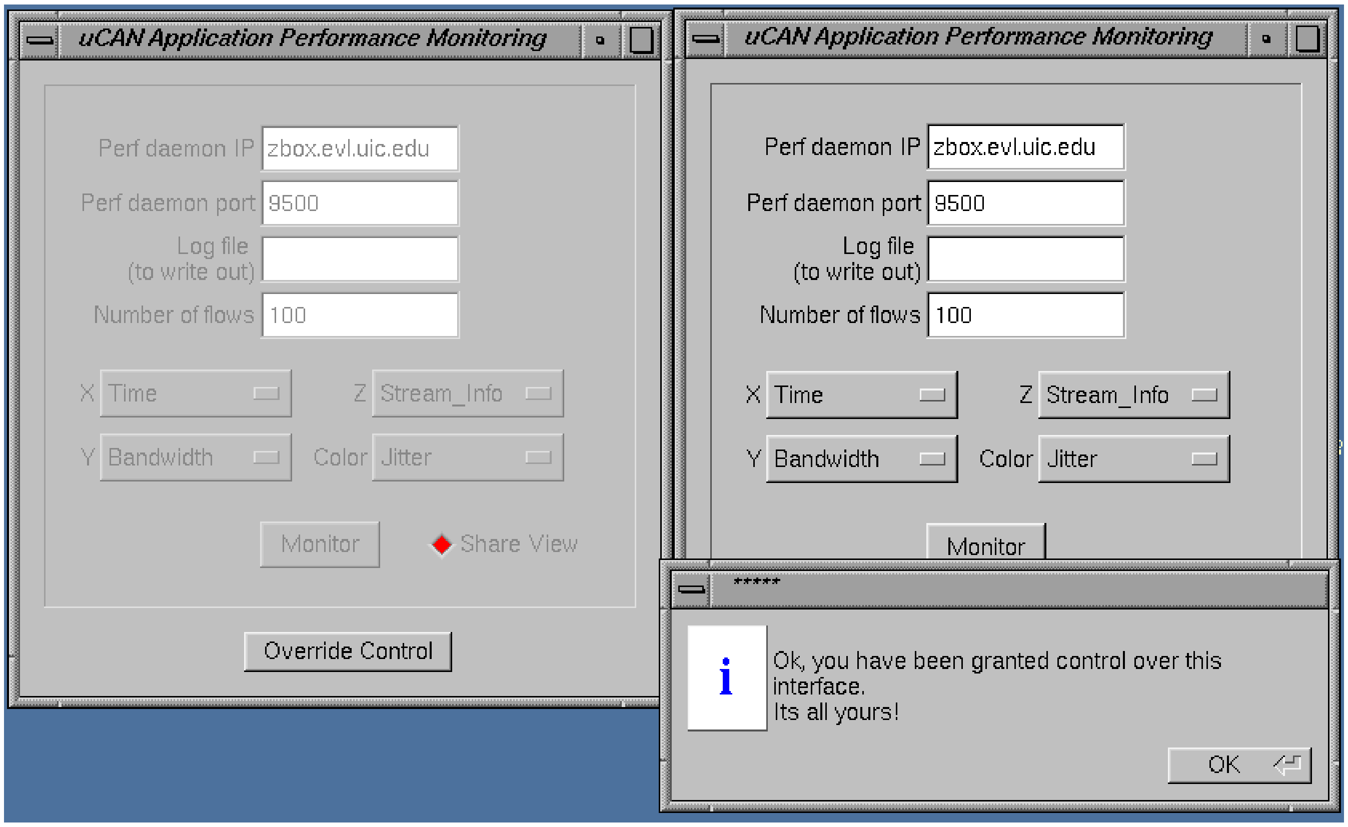
A remote user has requested control over this interface, do you wish to grant him control ?

The control request has been sent for authentification. Please wait for the response

Figure 20: *The figure shows a request from the user on the right to the owner of the interface(left), for granting control. The owner will lose control once he grants control to the remote user*

## 5.3 Building a New Shared Interface

The networking and database architecture and the event handling, including defined control transfers are reusable portions of uCAN's framework. A new interface designed by the user to configure a new kind of test or visualization, can also use the above features and make their interface shared like other uCAN interfaces. To do this, the shared interface must be written as a C++ class inherited from $uCAN\_sharedInterface\_c$ and they automatically inherit the


uCAN Application Performance Monitoring

Perf daemon IP zbox.evl.uic.edu

Perf daemon port 9500

Log file (to write out)

Number of flows 100

X Time

Z Stream_Info

Y Bandwidth

Color Jitter

Monitor

Share View

Override Control

uCAN Application Performance Monitoring

Perf daemon IP zbox.evl.uic.edu

Perf daemon port 9500

Log file (to write out)

Number of flows 100

X Time

Z Stream_Info

Y Bandwidth

Color Jitter

Monitor

*****

Ok, you have been granted control over this interface.
Its all yours!

OK

Figure 21: *The figure shows the grant of control to the remote user. The remote user is in full control over the interface and can perform operations as he can from his own interface. The owner can override control at anytime.*

functionality of an event client for sharing with remote interfaces and a event handler to listen to events defined by the user, as shown in Figure 18.

Here's a sample usage of the api's to make a new interface, say $newSharedInterface$ collaborative:

**class newSharedInterface : public uCAN_sharedInterface_c** *//Inherit from the base type*

**eventClient = new uCAN_eventDBClient_c** *//Creates a new event client*

**eventClient → shareMyStuff()** *//Connects to a collabortive session*

**eventClient → triggerThisUI(this)** *//delivers the updates to the widgets managed*

The widgets in the interface should be managed by the *manage*() api described in the previous section. The events to the managed widgets are delivered as specified. The user can control the updation of the interfaces by using the *eventClient* → *process*() api.

Thus a combined view of these three sections gives a good picture of how collaboration is implemented in uCAN to share interfaces and how the framework can be extended to build new shared interfaces.

## 5.4 Summary

The design and implementation of collaborative services has been discussed in detail. The chapter's aim was to explain breifly, the issues in creating a shared workspace type environment and to define needs of collaboration in the domain of network analysis. A treatment of the design has been given without too many implementation level details, but with just the detail required to understand the basic functionality.

So far, the emphasis has been on exploring the core capabilities of network analysis, concentrating on the depth of each area. The next chapter describes the missing pieces needed to come to the final picture of an extendable unified collaboratory.

# CHAPTER 6

# AN EXTENDABLE UNIFIED COLLABORATORY

This chapter describes the implementation of the framework in uCAN, focussing on the areas that have been designed to be reused and extended. A results section has been provided, giving some sample results from an experiment.

## 6.1 Unification of tools: The Event Reflector

The use of multiple tools in a collaboratory is highly enhanced when they can be made to complement each other's capabilities. This is the motivation behind using an event reflector to transmit event messages between processes. The objectives of the event reflector include providing simple methods to pass events and allowing an external application to connect to the event reflector to pass events. The implementation of these features is explained in the following section.

### 6.1.1 Some API's

An event can be sent can be sent to the event reflector using the following api's:

**uCAN_internalClient_c $*$ internalClient**

**internalClient $\rightarrow$ init()**

**internalClient $\rightarrow$ sendEvent(description)**

where the *init*() api connects the internal event client(*internalClient*) to the internal event reflector on the local host and *sendEvent*() sends a timestamped event with the *description* string that describes the event to all other connected clients(processes which include graph widgets used for active tests, snmp queries and shared widgets)who are listening to events from other processes.

The shared interfaces that wish to listen to events from the event reflector should implement the following api:

**announceEvent(timeStamp, description, typeOfEvent)**

where the *timestamp* corresponds to the time when event was sent from a process, *description* describes what the event is and the *typeOfEvent* can be a default event or a specific event that maybe understood and implemented by this process, either to return a particular value or do something after checking for a condition. This can be conceivably very useful in the context of applications requesting for values of network parameters from tests or from a snmp variable.

## 6.2 Joining the pieces for a collaborative session

This section explains the management of a collaborative session, including logging into a new section and managing new interfaces in a collaborative session.

### 6.2.1 The Objectives

1. Once an user is logged into a new collaborative session, all other interfaces should be updated about the central server and can connect to it at anytime
2. When a new interface is shared from a remote end, it should be run on the local host in the inactive mode.
3. Each participant should know the list of participants in a collaboratorive session
4. A white board feature will be useful for exchanging messages between collabroators

### 6.2.2 The Connection Manager

The connection manager module in uCAN coordinates between shared widgets and is built to meet the objectives as mentioned. A login interface used is shown in Figure 23. Participants

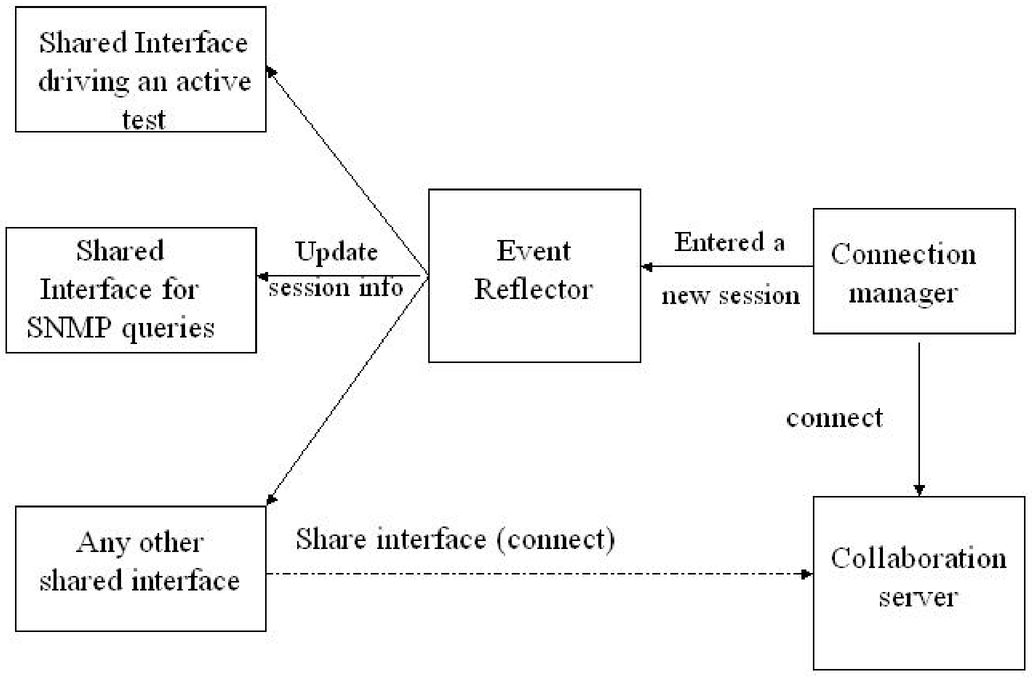
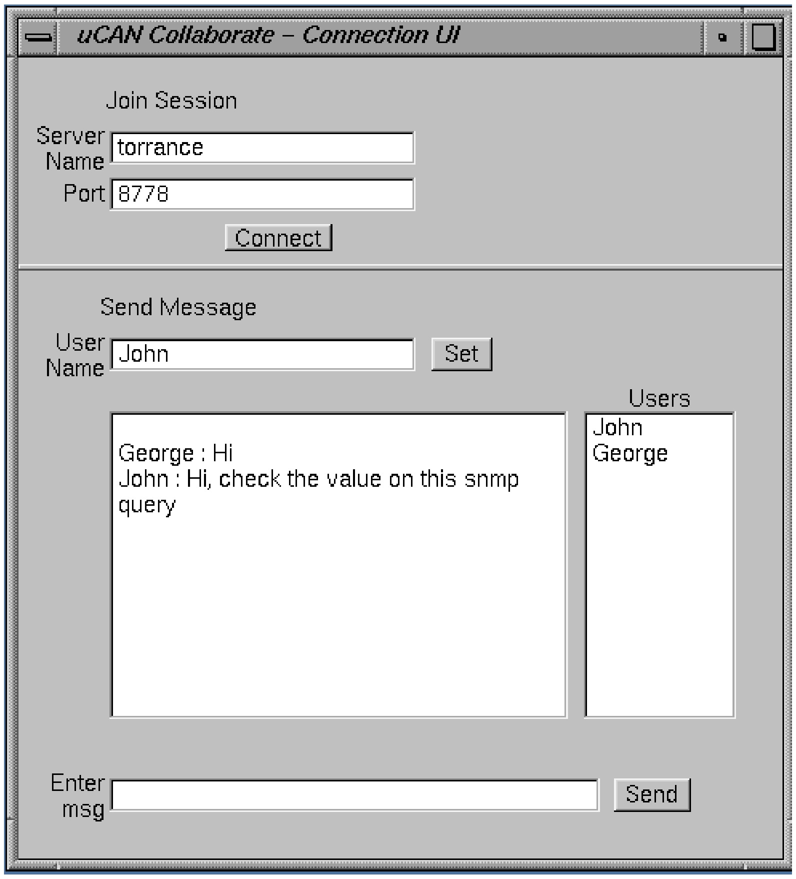

Shared Interface driving an active test
Shared Interface for SNMP queries
Any other shared interface
Update session info
Event Reflector
Entered a new session
Connection manager
connect
Share interface (connect)
Collaboration server

Figure 22: *The diagram explains how a collaborative session is managed with the help of a connection manager. Once the user logs into a new session, information about the session is transmitted to all his internal operations -so that each interface can connect itself to the session*

can log on to a known collaboration server to join others in a session. Once a connection is established, the location of the server and the port is reflected to all other shared interfaces on the local host, via the internal communication channel (through the internal event reflector described in the previous section). This is explained in Figure 22. A interface, once shared, connects to the collaboration server and is a new client in the session. For each new interface shared, a new remote interface is created on other sites by the respective connection managers.



Server Name: torrance
Port: 8778
User Name: John

George : Hi
John : Hi, check the value on this snmp query

Users: John, George

Figure 23: *The figure shows a login interface, which is used to enter a collaborative session. The users can conference using the whiteboard. Once a collaboration session is started, all other processes get updated about the session information.*

An implementation issue, in creating remote interfaces dynamically, is to keep track of the number of interfaces and to start remote interfaces for each interface shared from other machines (it should be taken care that interfaces on the local host are not duplicated). This is implemented by maintaining a list of interfaces on the collaboration server, that is updated each time a new interface enters a session. The list entry consists of the identification of the interface(essentially a unique path for the interface) and the host name of the machine on which the interface (and potentially, a corresponding network operation) was initiated.

## 6.3 Moulding into a Collaboratory

In the trend towards collaborative computing and tool building, there exists a need for collaboratories that integrate different tools and encourage new tools to be added to the repository of collaborative tools. The final vision of uCAN is to attempt to mould a system along these lines, by designing a framework to add new tools and make it collaborative. An implementation shows a collaborative test repository and a way to configure new tests and add to the collaboratory. Figure 24 shows the configuration tool for tests. Though the framework is meant for tests, most parts of it can be reused to use it in the future for a more generic collaboratory for network operations.

### 6.3.1 Writing DSO's -Some API's to Use

Active tests are implemented as Dynamic Shared Objects(DSO's). The DSO written should be made to extend a generic collaborative test C++ class in uCAN, namely uCAN_collab_test_c. This has a few API's that help run a new type of test easily. Once the parameters for a test has been supplied, the test is automatically run a test machine that the user has specified. This is acheived by passing the name of the DSO to link to. A test daemon should be run on that machine that would link to the corresponding DSO at run time, assuming it has been compiled and available on that machine. The Figure 11 given in chapter 4 explaining the architecture of the test daemon may be referred in this context. Some API's that are useful for fitting a new tool are given as follows:

**initTest(dsoName, hostName, ifScript, logfilename, parameterSize)**

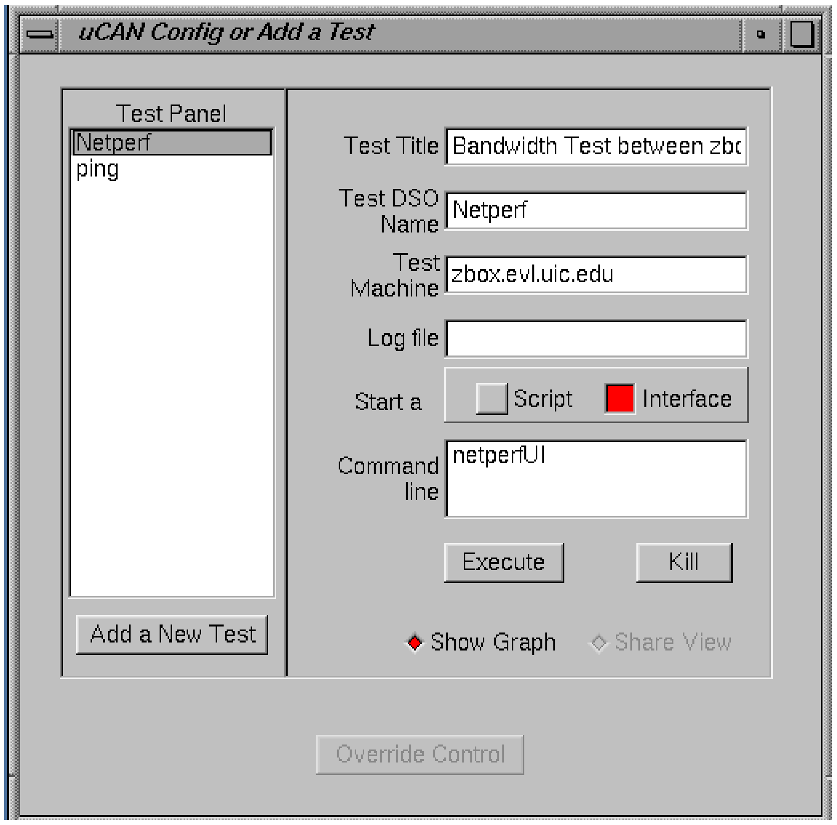


Figure 24: *The configuration interface shows how a new test can be added or an existing test can be reconfigured. The configuration is saved for future sessions. The user can either run a script or have the tool start another executable for his own interface. DSO names of the test are used to refer to the test. In this interface, the Netperf test has to be implemented in a Netperf.so.*

This should be done to initialize a test. The parameters needed are, the name of the DSO that should be used for the test($dsoName$), the name of the test machine($hostName$),the name of the logfile -if the test has to be logged, the parameter size of the test parameters that will be passed and whether the program is a script or an interface. By default the function will be mostly used by programmers who design their own interface. Scripts are dealt differently, as explained later in this section.

The test to be run is encoded in a set of strings(say *argv*). If *argc* indicates the number of these strings which form the command, a test can be triggered by the user with the following API:

**runThisTest(argc, argv)**

The test command thus provided is packed and sent to the test daemon. The rest daemon links to the appropriate DSO for the test and executes the test using a

**executeTest(argc, argv)**

which should be implemented in the new test defined.

The results of the test have to be deciphered by the user. and the value to be packed to be sent for analysis. The buffer is packed at the testdaemon using a $packTestResults(buffer, size)$ api and unpacked at the user end by an $unpackTestResults(buffer, size)$ api. The results of the test thus packed are automatically sent to all clients interested in receiving the resulting test data.

The results of the test will be plotted on a graph(unless the user decides to disable the graph option) The coordinates of the graph, the scale and origin are configurable.

Most network researchers who carry tests on networks, use scripts, usually tailored to meet their needs. It becomes important to support scripts to be run directly without minimum extra effort. For a script to run on a remote test machine, the user will have to specify the script command line with the configuration tool as showin in Figure 24. The script will be run in a seperate shell and the user will just have to implement the pack and unpack methods for interpretation of the tests.

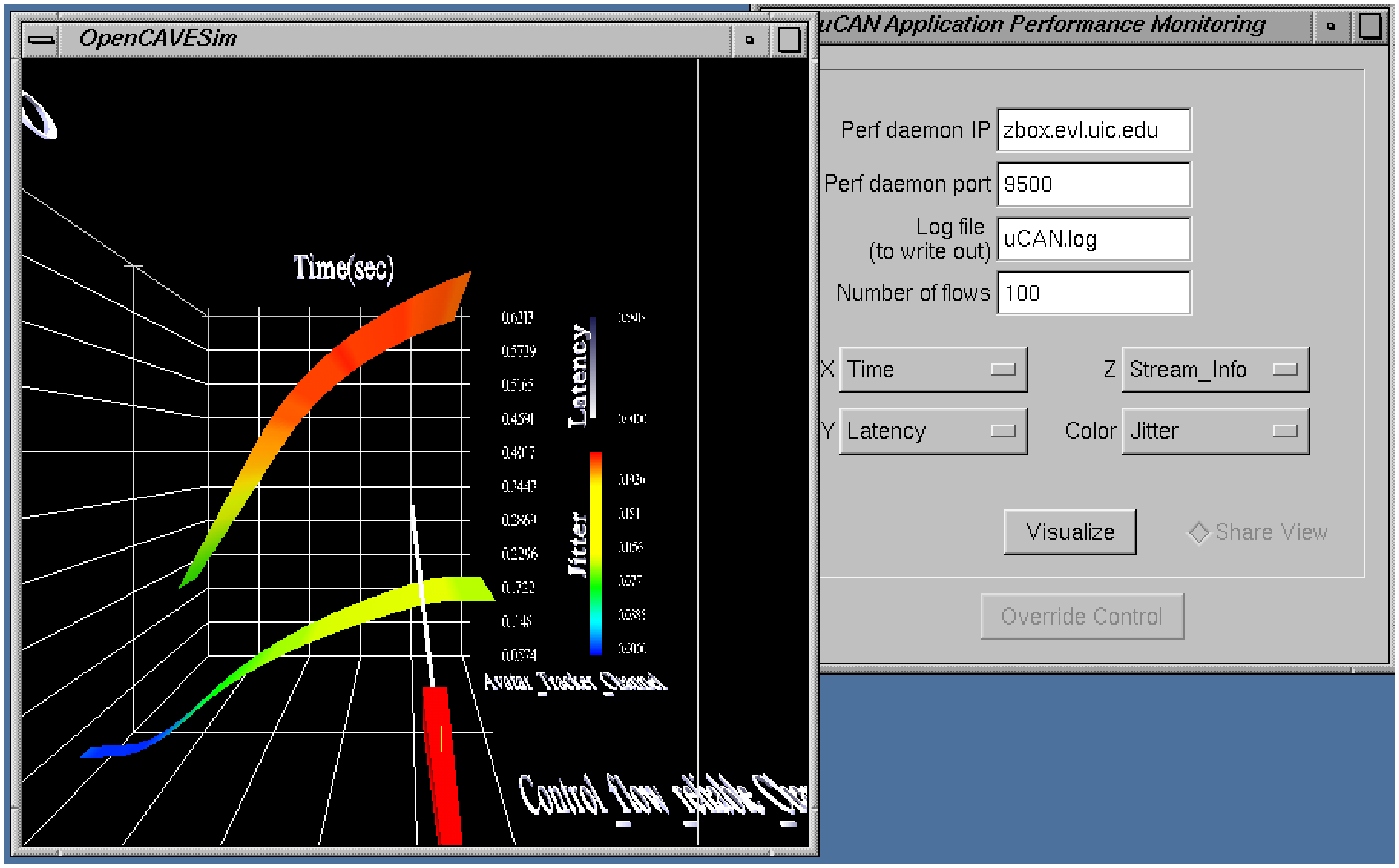
The main idea behind this framework is that the new test thus added using the configuration tool is automatically made part of the integrated and collaborative framework of existing tests. So, if the user decides to share the results of a test, the test panel for others in the collaborative session will be updated to add the new test and a graph will be started that will be automatically updated with the results of the tests, assuming the DSO for the test exists on all machines. Also the graph plotted will be enabled to receive events from the event reflector to help in simultaneous analysis.

## 6.4 Results and Analysis

### 6.4.1 An experiment

An experiment was conducted between an SGI Onyx2(sender) and a SGI O2(receiver), both with 100 Mbps ethernet cards connected to a Cisco 7507 Diffserv enabled router. The aim of the experiment was to demostrate utility of uCAN to quickly arrive at a possible solution. The scenario explained in this case is very simple. But the thesis is that in the case of more real complex networks, another tool required to gather more information, can be used to fit into the collaborative and unified framework of uCAN.

The machines were used to run a simulated, scaled down version of a tele-immersion application, using the sender as the client and the receiver as the server. The application is multi-processed, with each process consisting of a different type of flow. The performance information collected from the two (reliable and unreliable) channels instrumented, was visualized using QoSiMoTo (Figure 25). The first process consisted of a reliable communication loop, in which TCP was used to send some data continuously. The flow corresponding to this process is labelled as *Control_flow_reliable_channel* in Figure 25 . The second process consisted of an unreliable message communication loop using UDP. Such a channel is typically used for sending traced positions of users in applications like tele-immersion. This flow is named as *Avatar_Tracker_channel* in Figure 25. The third process consisted of a file download loop which was left to run in the background. The performance visualization of the streams, with latency and jitter versus time, of multiple flows, is seen in Figure 25.



Perf daemon IP: zbox.evl.uic.edu
Perf daemon port: 9500
Log file (to write out): uCAN.log
Number of flows: 100
X: Time
Z: Stream_Info
Y: Latency
Color: Jitter

Figure 25: *The figure shows a qosimoto visualization, with the configuration interface in uCAN, visualizing two different streams used in the experiment. Latency(seconds) is plotted versus time(in seconds) with the two streams along the Z-axis. Color is used to represent jitter. As traffic conditions on the router changed, the latency showed a sharp rise in the streams as shown.*

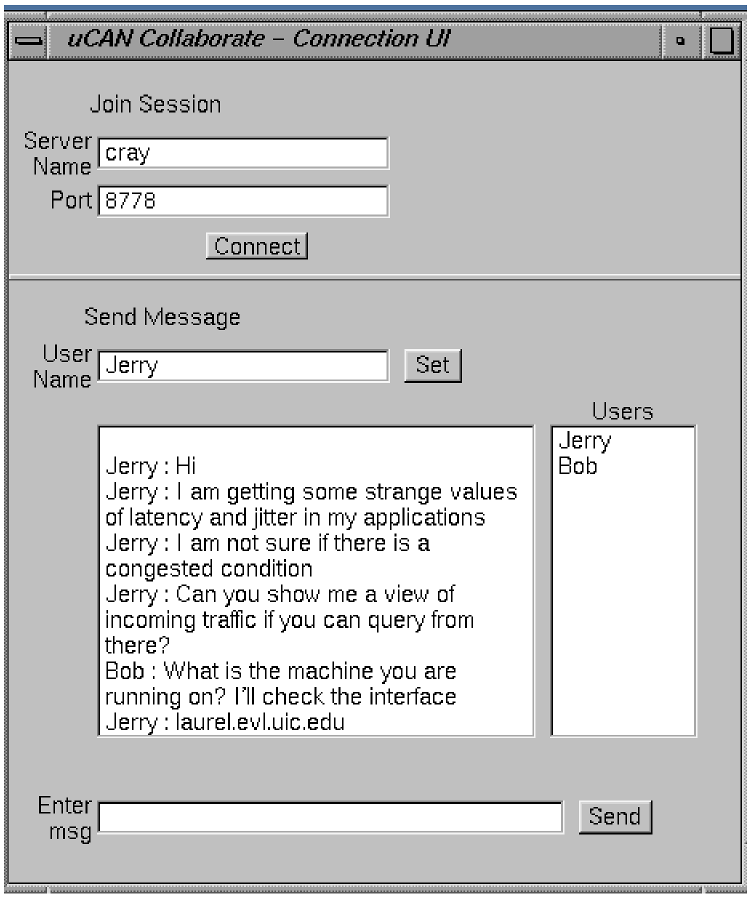
Now background test traffic, which consisted of jumbo UDP packets at rates starting from 20 Mbps to 100 Mbps was sent in the same direction -from the sender to the receiver used for the above application.

A sequence of steps listed below, is hypothesized, simulating a real environment (to indicate that this sequence of steps is possible) to demonstrate the use of uCAN in an application.

1. The QoSiMoTo visualization shows a sharp rise in latency in the application flows.
2. The application developer/user contacts the network manager who handles the router management and use his help in identifying what could be the source of the traffic situation.
3. The manager brings up a SNMP monitoring tool to query the incoming traffic at the sender machine(*laurel* in Figure 27). He shares the query interface with the user, and the user sees the graph shown in Figure 27.
4. The user and the manager interact to find out from other interfaces to see if any activity indicates a relation with the condition in the receiver's interface. After this, the user requests control from the manager and decides to make his own query, querying the sender's interface (*turing* in Figure 28).
5. The traffic pattern matches in the output traffic from turing and input traffic on laurel indicate a possible relation, thus helping in narrowing down possible sources of bottlenecks.

### 6.4.2 Analysis

The sequence of the experiment demonstrates very clearly that use of an unified, collaborative tool like uCAN can accelerate arriving at the source of a problem. However in the



Server Name: cray
Port: 8778
User Name: Jerry
Enter msg:

```
Jerry : Hi
Jerry : I am getting some strange values
of latency and jitter in my applications
Jerry : I am not sure if there is a
congested condition
Jerry : Can you show me a view of
incoming traffic if you can query from
there?
Bob : What is the machine you are
running on? I'll check the interface
Jerry : laurel.evl.uic.edu
```

Users:
Jerry
Bob

Figure 26: *The connection interface - an user logs into a collaborative session to collaborate with a network manager to 'borrow' some information about incoming traffic related to his host machine*

case of more complex networks, several interfaces exist in a router and each one with several suibnetworks, passing traffic to many such machines. In such cases, the use of a passive measurement visualizer to identify flows in the traffic will be very useful. Also a map of the topology of the network or of bandwidth utilization of several interfaces on a router, will also help in identifying locations of problems that was trivially done in this experiment. The thesis of the experiment again stands to prove that such a unified and collaborative approach is useful and gives a new powerful platform to use other such new tools in future to identify more realistic traffic problems.

## 6.5 Summary

The focus of the chapter has been on describing the unification of tools in uCAN and the methodolgy to add a new tool into the collaboratory. The results section demonstrates the advantage of using a unified and collaborative approach in understanding network problems related to applications.

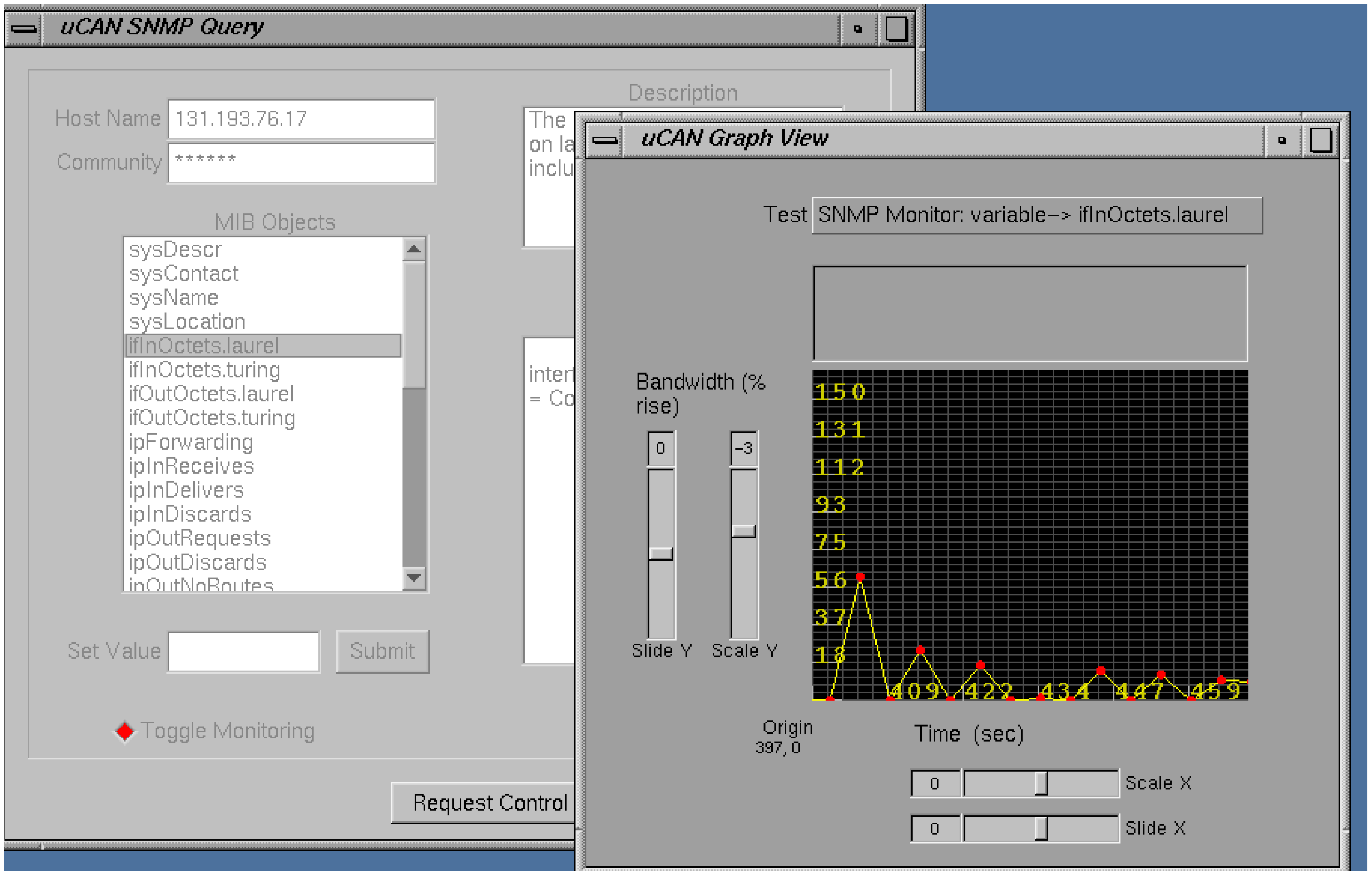


Figure 27: *The network manager situated remotely brings up a graph to monitor traffic inflow at the interface to which the user's host is connected. The graph shows a plot of percentage increase in the bandwidth (the percentage of the new input octets with respect to the total number of octets at the interface) versus time. The query interface is controlled by the manager here, while the results of the query can be seen by both of them. The bandwidth rise showed a sharp increase, when traffic was first introduced and the SNMP database was updated.*

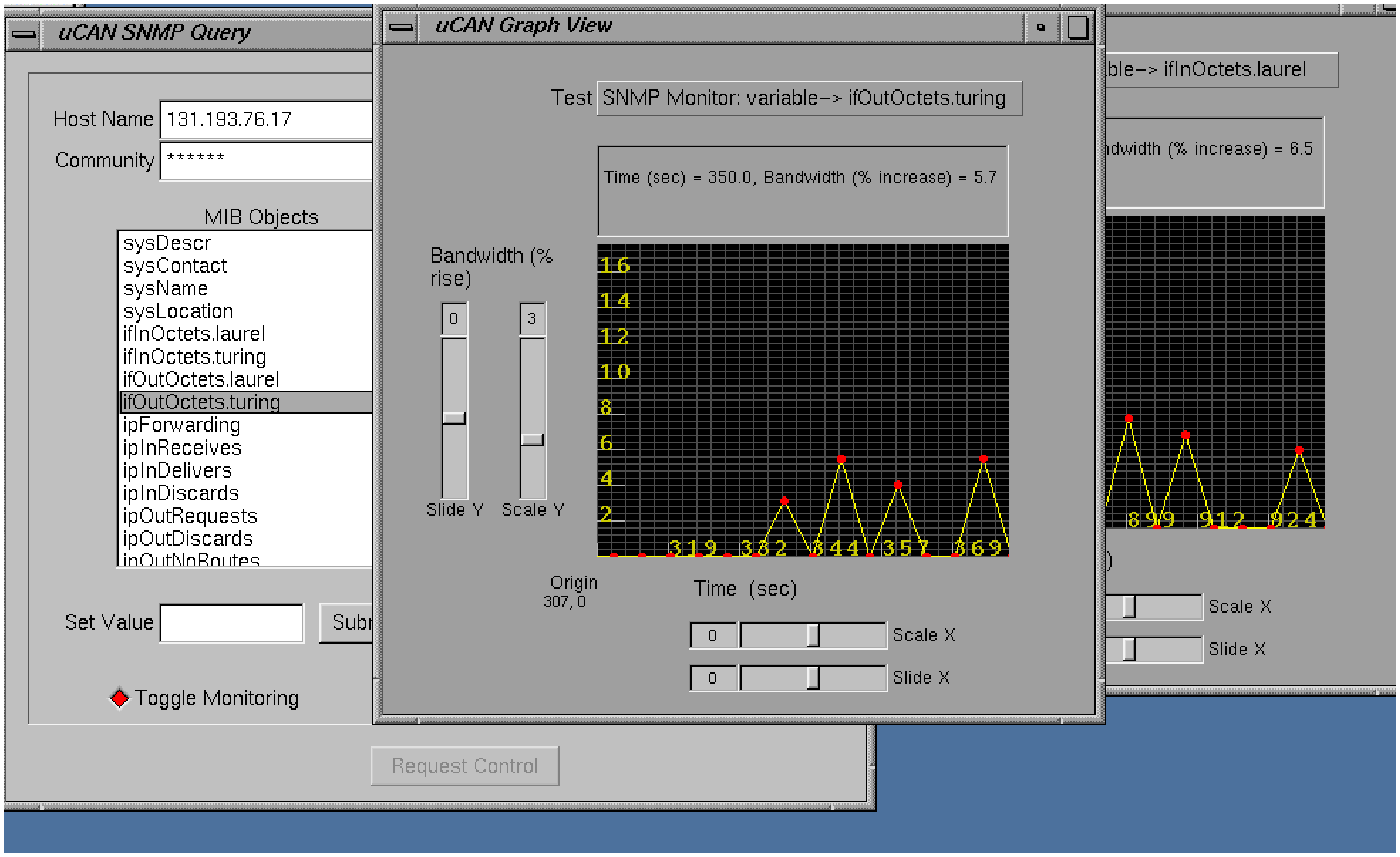


Figure 28: *The user then requests the network manager control over the SNMP query interface to make his own query to find out possible areas from which traffic might arise from. The Figure shows that the user is in now in control and has started a SNMP monitor on output traffic from turing - the sender in the experiment. A comparison with traffic patterns in laurel indicated that there could be a possible mapping between the two*

# CHAPTER 7

# CONCLUSION AND FUTURE WORK

## 7.1 Conclusion

On a concluding note, a second look at the formal definition of analysis helps to summarize the context and focus of this thesis. The purpose of a unified and collaborative approach is mainly to understand diferent perspectives of analyzing networks, exploring how they can merge together to work hand-in-hand and aim towards building a collaboratory of network tools that can be shared in a group environment. A detailed treatment of the active testing, network management and performance monitoring capabilities has been provided. The focus has been to stress on unique features and reusable framework design at places applicable.

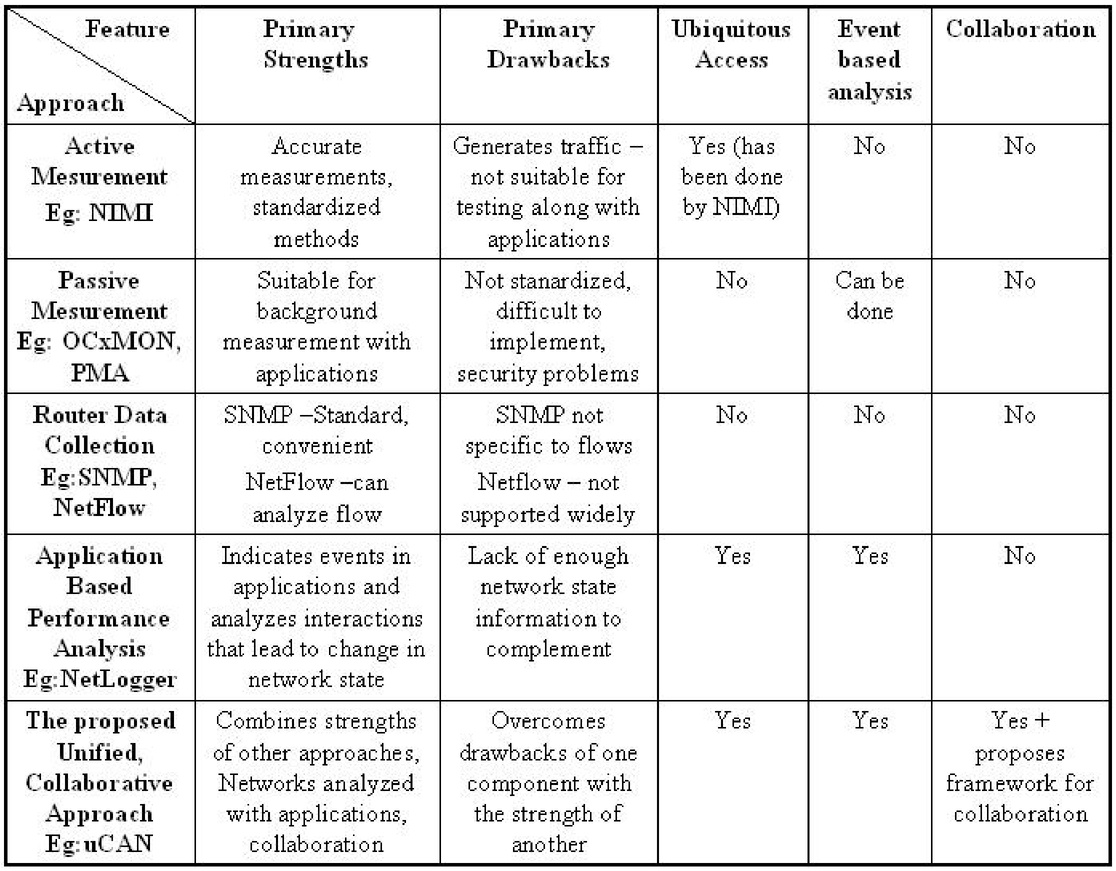
### 7.1.1 Contributions

The main contributions of this thesis can be listed as follows:

1. Provides ubiquitous active testing capabilities
2. Proposes an active testing framework for adding new testing tools
3. Proposes ubiquitous network management in replacement for traditional management
4. Provides an event based analysis methodolgy for applications and a 3D visualization capability using an existing tool
5. Proposes a unified methodology for analyzing network events along with events from applications
6. Proposes collaboration and a role based remote acess control for analyzing networks

7. Proposes a collaborative framework, with utility tools for building collaborative applications

A comparison of such a unified and collaborative approach with other traditional approaches and techniques can be made as given by the following table. As the table indicates, the strongest point of the approach is that, it combines the strengths of other approaches and aims to explore more ways of using one approach to complement the others.



| Feature / Approach | Primary Strengths | Primary Drawbacks | Ubiquitous Access | Event based analysis | Collaboration |
|---|---|---|---|---|---|
| **Active Mesurement** Eg: NIMI | Accurate measurements, standardized methods | Generates traffic – not suitable for testing along with applications | Yes (has been done by NIMI) | No | No |
| **Passive Mesurement** Eg: OCxMON, PMA | Suitable for background measurement with applications | Not stanardized, difficult to implement, security problems | No | Can be done | No |
| **Router Data Collection** Eg:SNMP, NetFlow | SNMP –Standard, convenient NetFlow –can analyze flow | SNMP not specific to flows Netflow – not supported widely | No | No | No |
| **Application Based Performance Analysis** Eg:NetLogger | Indicates events in applications and analyzes interactions that lead to change in network state | Lack of enough network state information to complement | Yes | Yes | No |
| **The proposed Unified, Collaborative Approach** Eg:uCAN | Combines strengths of other approaches, Networks analyzed with applications, collaboration | Overcomes drawbacks of one component with the strength of another | Yes | Yes | Yes + proposes framework for collaboration |

Figure 29: *Comparison with other approaches for Analyzing Networks, the relative strengths and drawbacks and some missing criteria. The rows of the table show the various approaches and the columns of the table show the criteria of comparison.*

## 7.2 The Future

The thesis has been an attempt to define an approach and formulate a framework for using the approach further to use different tools. Therefore, the focus will be on building the toolset and experimenting more with different type of tools. Users' perspectives to the tool will be very vital to make the design more generic and acceptable to new tools. A primary focus is therefore to help analyze performance in a research network between EVL and another institution.

The unified framework has inevitably led into the other side of analysis -helping an application react to performance problems that have been detected. The event reflector scheme that has been discussed should include more features to help applications query values of network parameters that interest the application and set trap-like events that give an automatic indication back to the application when a network parameter or a monitored MIB variable crosses an acceptable limit.

In the case of large collaborative environments, the scalability of the collaboration server might become an issue. Though the performance scales fairly well for around 10 clients, higher numbers might benefit from a peer-peer selective update method or use of multicast support for managing clients.

Considering the sensitivity of network information and the possible concerns this approach might raise in the networking community, an implementation of a cryptographically secure server for collaboration and also extending it for access to SNMP queries, tests etc, should be an important step.

Finally, experiences with uCAN will lead to placing it as a Grid tool to aid acceleration in understanding networks, which will ultimately help networks and applications work together smoothly, in a better understood context of each other.

## CITED LITERATURE

1. Leigh, J. et al.: Cavernsoft g2, a toolkit for high performance tele-immersive collaboration, http://www.openchannelsoftware.org/projects/cavernsoft_g2.

2. Spitzak, B. et al.: The fast and light toolkit, http://www.fltk.org.

3. Wolf, W. A. et al.: National Collaboratories: Applying Information Technology for Scientific Research. National Academy Press, 1993.

4. The HP OpenView Homepage, http://www.openview.hp.com.

5. The CiscoWorks 2000 Homepage, http://www.cisco.com/warp/public/cc/pd/wr2k/index.html.

6. Cottrell, L.: A list of network monitoring tools from the stanford linear accelerator center, http://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html.

7. The simpleweb, list of network management software, http://www.simpleweb.org/software.

8. Schonwalder, J. and Braunschweig, T.: Scotty, Tcl Extensions for Network Management Applications, http://wwwhome.cs.utwente.nl/ schoenw/scotty.

9. Oetiker, T., Rand, D., et al.: The Multi Router Traffic Grapher network monitoring tool, http://mrtg.hdl.com/mrtg.html.

10. The NET-SNMP Project Homepage, http://net-snmp.sourceforge.net.

11. White Paper on NetFlow Services and Applications.

12. cflowd: Traffic Flow Analysis Tool, http://www.caida.org/tools/measurement/cflowd.

13. Haberman, M. et al.: flowboy, an object-oriented framework for generic network flow management. In Proceedings of the Passive and Active Measurements Workshop, Hamilton, New Zealand, 2000.

14. Jones, R. et al.: The public netperf homepage, http://www.netperf.org/.

15. The ping page, http://www.ping127001.com/pingpage.htm.

16. The traceroute page, http://www.traceroute.org.

17. Kalidindi, S. and Zekauskas, M. J.: Surveyor: An infrastructure for internet performance measurements. In Proceedings of INET'99, 1999.

18. G, F. et al.: Providing active measurements as a regular service for isp's. In Proceedings of PAM 2001, 2001.

19. Almes, G. et al.: A One-way Delay Metric for IPPM, RFC 2679, September 1999.

20. Hansen, T., Otegro, J., McGregor, T., and Braun, H.-W.: Active measurement data anlaysis techniques. In The Proceedings of the International Conference on Communications in Computing (CIC'2000), Las Vegas, Nevada, June 2000.

21. Paxson, V., K.Adams, A., and Mathis, M.: Experiences with nimi. In Proceedings of PAM 2000, 2000.

22. Apisdorf, J. et al.: Oc3mon: Flexible, affordable, high performance stat istics collection. In Proceedings of the USENIX LISA X, 1996.

23. The Coralreef Measurement software, http://www.caida.org/tools/measurement/coralreef.

24. The tcpdump public repository, http://www.tcpdump.org.

25. Brown, J.: The Cichild Visualization Software, http://moat.nlanr.net/Software/Cichild.

26. Ellson, J. et al.: GraphViz Home Page, http://www.research.att.com/sw/tools/graphviz.

27. The GeoBoy2 Homepage, http://ndgsoftware.com/ ndgprod/page15.html.

28. The VisualRoute Hompage, http://www.visualware.com/visualroute.

29. Browne, S., Dongarra, J., and London, K.: Review Of Performance Analysis Tools for MPI Parallel Programs, http://www.cs.utk.edu/ browne/perftools-review, !997.

30. Gunter, D. et al.: Netlogger: A toolkit for distributed system performance analysis. In Proceedings of the Eighth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000.

31. Abela, J. and Debeaupuis, T.: Universal Format for Logger Messages, IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-abela-ulm-05.txt.

32. Gates, M., Warshavsky, A., and Welch, V.: The netlog library libray for performance monitoring, http://dast.nlanr.net/projects/netlog.

33. Leigh, J., Johnson, A., Defanti, T., et al.: A review of tele-immersive applications in the cave research network. In Proceedings of IEEE VR'99, Mar 1999.

34. Park, K.: Qosimoto 3d visualization tool, http://www.evl.uic.edu/cavern/qosimoto.

35. Cruz-Neira, C. et al.: The cave: Audio visual experience automatic virtual environment. Communications of the ACM, 35:65–72, June 1992.

36. Miller, G.: A perspective on passive measurement -a presentaion at the caida, http://www.caida.org/outreach/isma/9901/slides, 1999.

37. Azabu, M. and Minato-ku: From wealth to wisdom: A change in social paradigm. In Proceedings of CSCW 92, 1992.

38. Richardson, T. et al.: Virtual network computing. IEEE Internet Computing, 2:33–38, Jan/Feb 1998.

39. Ishii, H.: Teamworkstation: towards a seamless shared workspace. In Proceedings of CSCW '00, 2000.

40. Park, K. et al.: Exploiting multiple perspectives n tele-immersion. In Proceedings of IPT 2000, 2000.

41. Shu, L.: Groupware experiences in three-dimensional computer-aided design. In Proceedings of CSCW '92, 1992.

42. The internet2 website, http://www.internet2.edu.

43. The science, technology, and research transit access point, http://www.startap.net.

44. The doe science grid , http://www-itg.lbl.gov/grid/.

45. The access grid project homepage, http://www-fp.mcs.anl.gov/fl/accessgrid/default.htm.

46. The network time protocol distribution, http://www.eecis.udel.edu/ ntp/.

47. Engelschall, R. S.: Apache DSO Support, http://httpd.apache.org/docs/dso.html.

48. McCloghrie, K. et al.: Management Infomation Base for Network Management of TCP/IP-based internets, March 1991.

49. Stallings, W.: SNMP, SNMPv2, SNMPv3, and RMON 1 and 2. Addison Wesley Longman, Inc., third edition edition, 1999.

50. Gamma, E. et al.: Design Patterns, chapter 5, pages 293 – 393. Addison-Wesley, 1995.

# VITA

| | |
|---|---|
| NAME: | Naveen Kumar Krishnaprasad |
| EDUCATION: | B.E., Instrumentation and Control, University of Madras, Madras, India, 1998 |
| | M.S, Computer Science, University of Illinois at Chicago, Illinois, USA, 2001 |
| EXPERIENCE: | Graduate Assistant, Clinical Resource Management, Fall 1998 - Fall 1999<br>Graduate Assistant, College of Business Administration, Fall 1999 - Spring 2000<br>Research Assistant, Electronic Visualization Laboratory, University of Illinois at Chicago, Spring 2000 - Fall 2001 |
| PROJECTS: | |
| | Fault tolreant distributed system for ray casting |
| | Visualization of network performance in a virtual environment |
| | Shared Distributed document editor |
| | Statistical detection of network congestion using the Data Space Transfer Protocol |
| | Compression and Uncompression of files using the Huffman encoding algorithm |
| | PC based distributed control system, involving design of hardware interface to the PC and simple visualization of the control of chemical processes |
| REFERENCES: | Available on request |