VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Visualising IP Network Performance with TCL/Tk

Tim Robertson and Robert Biddle

Technical Report CS-TR-96/9
April 1996

Abstract

Modern computer systems include both local and wide area networking as part of their functional structure. Accordingly, effective performance management requires understanding of the network performance. Networking technologies are not specific to vendors of either hardware or software, and so practical tools to assist network management must work with a range of hardware and software.

This paper describes a tool for visualising network traffic on IP networks by means of customisable dynamic displays of network activity. The display software is largely written in TCL/Tk, an interpreted language with an excellent flexible graphics toolkit. The work described in this paper is part of a larger project, CR, involving development of a performance management system that uses a "building block" approach to allow easily configurable monitoring of distributed heterogeneous systems.

Publishing Information

This was originally published as: Robertson, T., Biddle, R., "Visualising IP Network Performance with TCL/Tk", In *Proceedings of the 13th Uniforum NZ Conference*, Uniforum NZ, 1996.

Author Information

Tim Robertson is a Communications Analyst with Shell New Zealand Limited. His role involves investigating new technologies and undertaking long term development project work on the Shell New Zealand network infrastructure. He completed his Master of Computer Science degree at Victoria University of Wellington in 1995. The practical component of the Masters involved developing a graphical network monitor for the "Control Room" project.

Dr. Robert Biddle is a senior lecturer in computer science at Victoria University of Wellington. His main research is in software engineering, particularly in software reusability and in user interface design. He is also interested in distributed systems management, and has worked with Unix since 1974. Robert studied at the University of Waterloo in Canada, and the University of Canterbury in New Zealand. He also has experience in development of commercial computing applications, and in research and development in industry.

Visualising IP Network Performance with TCL/Tk

Tim Robertson
Shell New Zealand Limited
tim@shell.co.nz

Robert Biddle
Victoria University of Wellington
robert@comp.vuw.ac.nz

Abstract

Modern computer systems include both local and wide area networking as part of their functional structure. Accordingly, effective performance management requires understanding of the network performance. Networking technologies are not specific to vendors of either hardware or software, and so practical tools to assist network management must work with a range of hardware and software.

This paper describes a tool for visualising network traffic on IP networks by means of customisable dynamic displays of network activity. The display software is largely written in TCL/Tk, an interpreted language with an excellent flexible graphics toolkit. The work described in this paper is part of a larger project, CR, involving development of a performance management system that uses a "building block" approach to allow easily configurable monitoring of distributed heterogeneous systems.

1 Introduction

The Control Room (CR) [Biddle, Hine, and Zhang, 1993] is a system for monitoring and reporting on the performance of distributed computer systems. It brings together information about behavior and performance for easy inspection, ongoing monitoring, and review. CR is built on a framework that allows a modular approach to constructing monitoring tools, using shared memory extensively to pass values between components.

The current metaphor for information used by CR is "node" oriented, as all statistics are reported in terms of the individual devices attached to the network (see Figure 1). This is ideal for statistics gathered which relate to specific nodes, but it is generally unsuitable for network traffic. With network traffic, we are concerned with where traffic is coming from and going to, as well the volume and nature of the traffic.

This paper describes a project to design and implement a prototype display that graphically shows network connectivity and dynamically shows activity on the network between nodes. A new "network oriented" display is needed which enables the user to make comparisons more easily, and is able to present the large amount of information in an intelligible way. This display needs to present information associated with links between nodes, rather than individual nodes themselves.

To fit the current CR framework, the project is divided into two components: a network activity gatherer, and a network activity display. As the new components are to be part of the CR system, they must use the same interfaces to be compatible. However, it is not the intention of this project to replace the current CR components. In fact, the new components supplement the existing components.

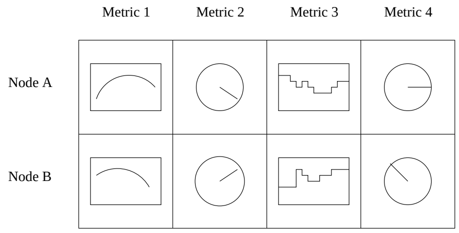

Figure 1: The Grid Model of the CR Display

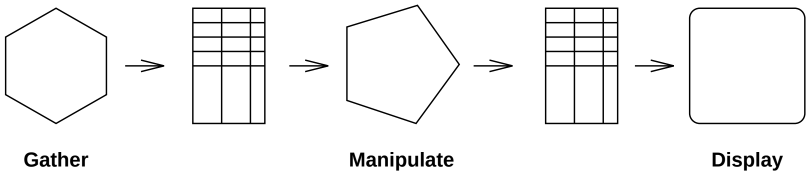

Figure 2: CR Basic Framework

For example, the existing display could be used to monitor the performance of each host, while the new display is used to monitor the network traffic between them.

2 Background

2.1 The Control Room Framework

The CR project is an attempt to bring the monitoring of distributed heterogeneous systems into one framework. The goal is to provide information about all devices in a consistent way, regardless of their type or manufacturer. While there are many tools for monitoring centralised systems, these provide an incomplete picture. CR was developed to integrate system and network information, as both are components of a single large system. The medium term goal of CR is to be able to anticipate performance problems before they severely impact system performance.

The key idea in the CR system is a flexible framework that allows for a modular approach to constructing monitoring tools. CR is a collection of components which support a standard interface for communication with each other. Components are added and removed depending on the required monitoring task. The standard interface chosen for communication between components was shared memory. The reasons given for its choice by Zhang [1993] were:

- It does not require any special files or communication channels.
- It has a low overhead and it is fast.
- It is easy to use with operations similar to file operations.
- Data is accessible to different processes, giving greater flexibility.

The only disadvantage of shared memory is that all the components of the CR system itself must run on the same machine.

There are three principle types of component in the CR framework (see Figure 2). They are:

- *Data Gathering Component* - Data gatherers retrieve data from data suppliers and deposit it into a shared memory segment. This data is called *raw data* and the memory segment is called the *raw shared memory segment.* Each type of data item is called a *metric.* Data gatherers can be run in parallel to feed the same memory segment. The CR system currently supports two gatherers; *Rstat* and *SNMP Gatherer Multiple* (SGM) [Flower, 1993]. The suppliers of the Rstat and SGM gatherers are Rstat daemons and SNMP daemons.

- *Data Manipulator Component* - Data from the gathering component is often unsuitable for display. The manipulator component allows the data to be processed before it is displayed. Gathering all the code which performs manipulation into one component helps to maintain consistency and reduce repetition. The manipulator takes the data from the raw shared memory segment, modifies it, and places it into another shared memory segment called the *cooked shared memory segment.* The current CR manipulator is written in C and requires re-compilation to change the manipulator's functions.

- *Data Display Component* - The display component takes data from the cooked shared memory segment and presents it to the user in a "useful" form. The current CR display is based on a grid model using a small window into a larger virtual display. The user can choose what information to display in the window. The grid model uses rows to represent nodes and columns to represent metrics (see Figure 1).

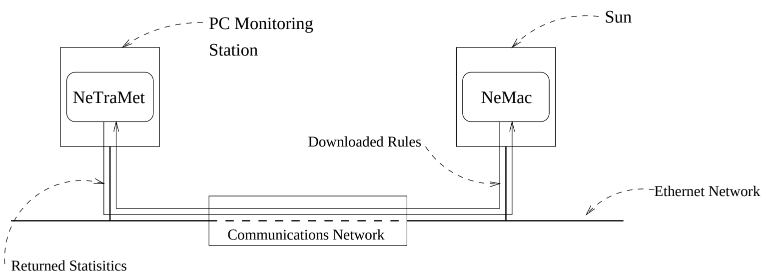

Figure 3: Operation of NeTraMet and NeMaC over a network

The terms raw and cooked are used to indicate whether the data has been modified from its original value in a particular memory segment. The principle components are typically organised as a sequence, starting with the gatherer and finishing with the display as illustrated in Figure 2. But as mentioned previously, flexibility is the key to the usefulness of CR; any number of components or shared memory segments are supported.

Other components developed for CR allow recording and playback of monitored information, and early detection of situations requiring attention (see [Gardner, 1994]).

2.2 Network Traffic

Understanding and monitoring *network traffic* requires some knowledge of what network traffic is and how it is transmitted. At the lowest level, network traffic is nothing more than pulses on transmission media. It is the structure imposed on these pulses by the source and destination nodes which allows them to be interpreted as information. The discussion which follows assumes that the transmission medium is *ethernet* using the internet protocols (IP).

The primary interest of network monitoring is what type of protocols are in use on the network and what network resources they are consuming. Common protocols in use on IP networks are:[1]

- TCP - Transmission Control Protocol
- UDP - User Datagram Protocol
- ICMP - Internet Control Message Protocol
- NFS - Network File System Protocol
- TELNET - Remote access Protocol

These protocols are encapsulated by the Internet layer and again by the Interface layer to eventually become packets on the network. We are interested in the total volume of the packets, their sources and destinations from the Internet layer, and the protocols they are associated with from the Transport layer.

2.3 NeTraMet and NeMaC

NeTraMet is an existing package which performs a similar function to that required for the new gatherer. It was the first implementation of the Internet Accounting Working Group's Internet Accounting Architecture [2] [Brownlee, 1994]. The version of NeTraMet used in this project was 1.2, which was released in January 1994. This release is composed of two separate components:

1. *NeTraMet* is a *meter* for network traffic which runs on a wide variety of hardware, including PCs. Using Nevil Brownlee's definition, meters are hosts which are attached to a network segment and measure traffic flowing on that segment. The hosts are usually PCs which are dedicated to the task.

2. *NeMaC* is a combined *collector* and *manager* which runs on the SunOS operating system. Again using Nevil Brownlee's definitions, a collector retrieves information from meters and a manager instructs meters which flows to measure.

[1]A more complete list is given in RFC-1061.
[2]RFC-1272 provides a more detailed introduction to internet accounting.

The NeMaC manager reads a set of rules which it downloads to the monitoring node running NeTraMet. NeTraMet then uses this rule set to determine which packets are of interest. It operates with the network interface card (NIC) in promiscuous mode to examine all the packet headers. If the rule set is large then there could be a large number of sequential matches which would be slow. NeTraMet performs an optimisation on the rules to speed the searches. Rules are organised into groups which test the same attribute using the same mask. Groups which have more than four rules are organised into hash tables which can be searched more efficiently. Packets which don't match any of the rules are discarded.

3 The New Gatherer

The new data gathering component must be responsible for collecting statistics from the network and placing them in the shared memory segment. It must be able to quickly, reliably, and efficiently collect any network statistic that the user is interested in.

As with other gatherers, we will actually need two programs working together: a supplier and a gatherer. A single program performing both the supplier and the gatherer roles is not possible, because it would restrict network information to that visible from the CR host. By splitting the supplier and gatherer role, suppliers can run on nodes around the network, and transmit information to a gatherer on the CR host.

3.1 Implementation Options

A possible implementation for the supplier is a daemon that gathers statistics and stores them for later collection. Several options exist for writing the daemon:

1. A modification of the SunOS (or another proprietary system) rpc.etherd daemon. This gathers statistics which are accessible through the rpc mechanism. The statistics gathered are the protocol type (TCP, UDP, ICMP, etc.) and the packet sizes. The modified version would also record statistics on the protocols used between specific nodes on the network.

2. A daemon based on the tcpdump facility. It is installed on most versions of Unix and is considered a standard utility. The source is available and the latest version includes a generic network interface component (but this was in alpha testing when this project was done).

3. A new daemon, possibly a kernel modification, to gather the statistics.

4. A modified form of *NeTraMet*.

Option 1 is the close to what is required but, as it is part of SunOS, the source was not available, and it would only work on a Sun. (Similarly for other proprietary software).

An important consideration in choosing the supplier is ease of portability, as new versions have to be written for each system type. NeTraMet and tcpdump include source for a number of architectures, including SunOS and PCs, which makes them suitable for the supplier. Option 3, the kernel modification, could be difficult to port across architectures as it would rely on system specific features.

The option which best fits the needs of a least a prototype is option 4, NeTraMet. It performs much of what is needed and is already portable across several architectures. To use NeTraMet with this project, the only modifications needed were to NeMaC to interface it with CR (see Figure 4).

3.2 Gatherer Design

One of the important criteria for the project was the integration of the new gatherer and display with the current CR model of shared memory. The CR shared memory segments use a conceptually two dimensional array of nodes and metrics. This structure is ideal for a node oriented system, but makes dealing with links more difficult as each link has at least two nodes involved. An ideal solution would have been to use a multi-dimensional shared memory segment with source and destination nodes on two axis and metrics on the remaining axis. While this would result in a simple and understandable structure, it doesn't fit with the existing CR capabilities. Changing CR to use a multi-dimensional structure would involve major architecture changes to other current CR components. Therefore any solution was necessarily two dimensional in nature.

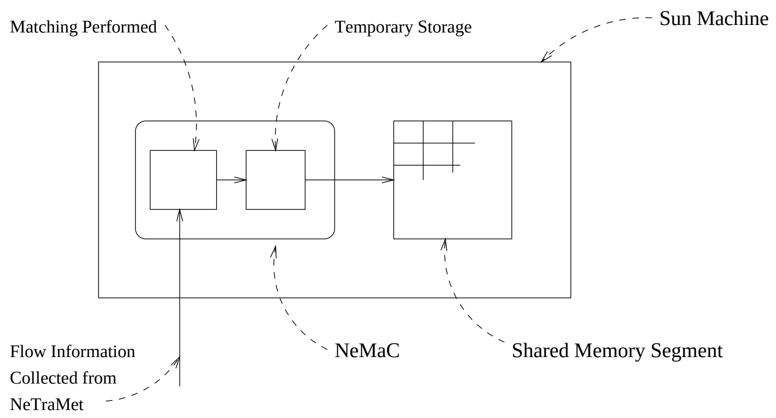

Figure 4: NeMaC and Shared Memory

The solution chosen for the project was to name the link rather than its end points. The link is then considered as a "node". Link names are the amalgamation of the source node name and the destination node name using an underscore to join them. For example the link from machine1 to machine2 is called

```
machine1.full.domain_machine2.full.domain    # link from 1 to 2
```

Full domain names are used as a link is logical; there does not need to be a direct link between the two nodes. Links are directional, the link above specifies traffic from machine1 to machine2. To specify a bidirectional link another entry is needed with the machine names reversed.

The metric, like the link name, is made up of two parts. The first part is the protocol used for the flow. Examples of protocols are TCP and UDP. The second part is the port number for the destination of the the flow. The metric name is made by joining the protocol and port names. For example, for Usenet type news using NNTP, the metric name is represented as follows:

```
tcp_nntp    # protocol = tcp, port = nntp (119)
```

Using the link name and the metric name as above, all the information needed is encoded into a two dimensional structure that fits well into the existing CR approach. This style of naming is used throughout these additions to the CR project.

4 The New Display

The goal of the display system is to show network activity without focusing on specific nodes. It should give an unambiguous indication of the type and amount of traffic flowing on the network. The user should be able to make comparisons directly from the display without needing to refer to other sources of information.

4.1 Implementation Options

The first decision was whether to base the display on an existing system or to write a new display from scratch. TkIned [Schonwalder, J., Langendorfer, H., 1993] was considered for modification. This is a network display tool with a considerable range of features for automatic network discovery and layout. It can also show limited network statistics gathered from the rpc.ether daemon using bar-graphs. However, TkIned provides node centered information, so it seemed unsuitable without substantial modification to provide a more network oriented display.

As there were no other suitable existing systems, the decision was taken to build a new display from scratch. This has the advantage of greater flexibility when different approaches are tried, and offers complete control over the code. The language chosen was TCL/Tk [Ousterhout, 1994] because of its flexible nature, fast development time for graphical applications, and existing use in the current CR display.

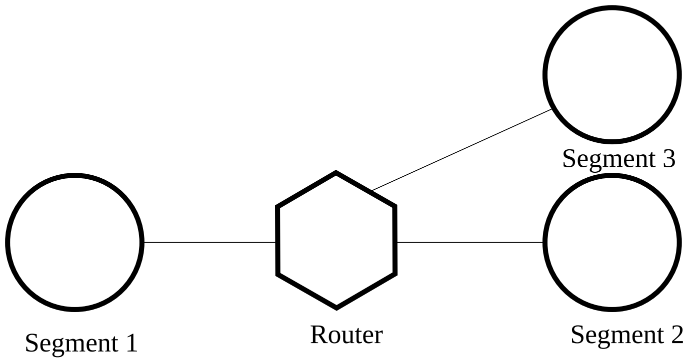

Figure 5: Example Network Diagram

4.2 Display Design

As the project focuses on the IP, the display is based on a visual metaphor for the internet architecture. The approach taken was to create a graphical display which presents a simplified view of the real underlying network. The new display is more graphical than the original CR display. It also makes greater use of colour to distinguish details. The basic philosophy of the display can be stated very simply: *show the traffic between two nodes by varying the thickness of a line connecting the two nodes in proportion to the amount of traffic.* A line on the diagram represents the flow from the previous collection interval, so the display shows *flow per unit time between nodes.*

There are two types of traffic that need to be displayed: traffic local to a segment, and traffic between two segments which needs routing. While both types of traffic could be shown on one large display, it was decided to break the display up into smaller pieces. This has the dual advantage of reducing information overload, and reducing the screen real-estate needed. The displays chosen were one for network traffic, and one each for local segment traffic.

Network Traffic

The network display is constructed from two types of nodes; nodes that correspond to routers and nodes that correspond to segments. Each node has an icon and is labeled with a textual name. Different node types are distinguished by colour and line thickness. The nodes are connected with lines which correspond to physical inter-network links. Figure 5 shows a small network consisting of 3 segments connected with a router.

In this prototype, there is no automatic detection of the network or network routing. Instead, the network topology is simply described statically in an input file, and both network topology and routing are determined from that description. In later versions of the software, the correct topology and routing should be determined automatically, as it is in TkIned.

Local Traffic

An accurate topological representation isn't used for the individual segments because an ethernet segment is essentially a broadcast medium. This means that all nodes on a segment receive all the packets sent on the segment. An individual node filters the packets, only retaining and taking to a higher software level those packets addressed to it. There is no correlation between the physical location of a node on a segment and the flow of packets. For this reason a circular segment shape was chosen as shown in Figure 6. It has a number of advantages:

- It is easily drawn with an arbitrary number of nodes equally spaced around the circumference of the segment. The textual names of the nodes can be easily placed.
- A link between any two nodes can be drawn easily in the form of a chord across the circle. This link doesn't cross any other node or node name.
- It gives a compact, tidy, and uniform display for each of the segments. This is important as it maximises the amount of information displayed for a given area of screen.

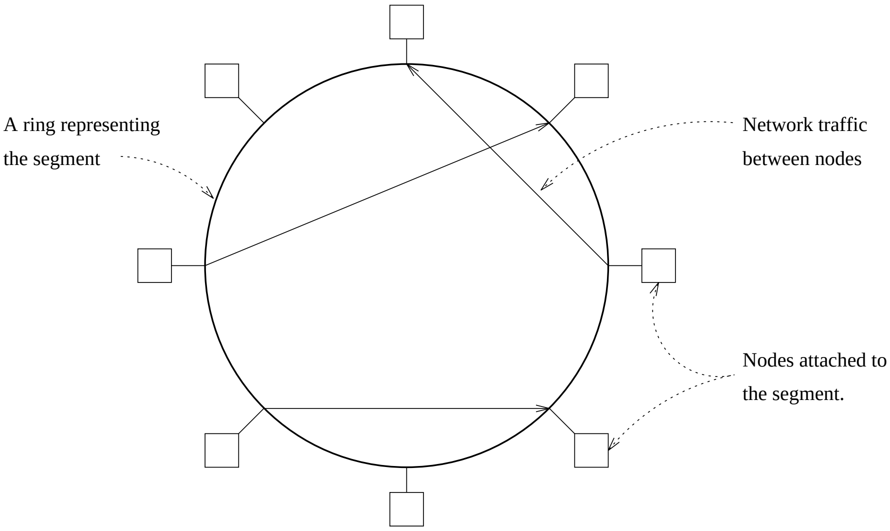

Figure 6: Example Segment Diagram

One remaining display issue is how network traffic that enters or leaves the segment display should be represented. In physical terms, the traffic comes through a router attached to the segment. A simple solution would be to include the router as a node on the local display. All traffic entering or leaving the segment could be displayed as lines to this node. This solution, however, would clutter the display with many lines if there was a lot of off segment traffic. It would make it harder to see the traffic which was local to the segment. The solution chosen was to display the traffic as radial lines away from the centre. This prevents the lines from interfering with the local traffic. This solution however, creates difficulties with the way multiple traffic flows are represented. There is not enough space between the node picture and the window border to use the multi-line approach (see below). The implementation in this project allows the lines to overlap.

Router Traffic

Routers on the network diagram are treated in a similar way to segments. They can be expanded to show the traffic between segments in the same way that traffic between nodes on the segment diagram is shown. This expanded view doesn't represent any physical entity, but gives a better understanding of how the router is routing traffic.

Representing Traffic

Traffic between nodes on each of the diagrams is represented in the same way. As the gatherer allows the metering of multiple metrics between nodes, the possibility exists that there may be more than one flow between two nodes active at one time. The lines representing different metrics would overlap when they were drawn. In the worst case, a thicker line drawn over a thinner line would totally obscure it. To prevent this possibility, the lines are adjusted in length to prevent overlap. As the number of lines increases, the average length decreases. The arrows indicate the direction of flow of the network traffic. In this case, the flow is from node 1 to node 2. Network flow in the other direction, i.e. from node 2 to node 1, could be represented using the same mechanism but with the arrow reversed. This is not done for two reasons: firstly, arrows pointing in different directions would be confusing, and secondly, if there are too many line segments then it is impossible to clearly see the width of the lines. The display uses 2 lines offset from each other, with each line representing traffic in only one direction.

The distance between the lines sets the maximum width that the line segments may obtain. If the the line segments' widths are not restricted, they will overlap, reducing the clarity of the display. In fact, the maximum width is slightly less than half the distance between the midpoints. This allows each line to grow to its maximum value and still not touch the other.

The other possibility of having multiple metrics between two nodes on the network or router diagram is that the same metric may occur more than once. For example, if we measure metric M between A and B as well as metric M between A and C, and flows from B and C follow the same path to A, then the path has two occurrences of metric M on it. To deal with this situation, the metrics are combined into one new metric (still called M), which is equal to the sum of the previous two.

The difficulty of using line segments to represent traffic is that the available line width limits the resolution possible. Network traffic may be a few bytes or many megabytes and representing this range is difficult with a small number of possible line widths. One approach is to use a logarithmic scale. This allows a wide variation in values to be represented but increases the difficulty in comparing values. However, a simple linear scale factor was chosen for this prototype. The scale factor for the line's width in relation to the actual traffic value was decided from observations of typical network traffic. Most flows were less than 200K bytes in size, so a line width of 1 represents 20K bytes. Traffic less than 20K bytes is represented with a line width of 1 and traffic greater than 200K bytes is represented with a line width of 10. Allowing the line width to continue to increase beyond 10 would create the problems of overlapping lines mentioned above. All traffic, regardless of protocol, is treated in the same way.

The actual numeric values for flow sizes are the most concise way to accurately represent traffic. For this reason, the values are shown, if required, in an additional window to the segment display.

4.3 The Graphical User Interface

The graphical user interface (GUI) for the display is designed to be intuitive and easy to use. Each display consists of a menu bar, a scrollable window, and a colour key for traffic identification. The menu bar varies between windows and contains options which are applicable to that window.

Clicking on a node in the network display pops up a new window that shows an expanded view of that node. Holding and dragging allows the node to be moved on the display to allow the manager adjust the display to a familiar orientation. Any links that are associated with the dragged node are automatically moved. The new positions of the nodes and links can be saved using the "save" menu item.

Clicking on a node in one of the expanded segment views pops up an information box which gives expanded information about that node and the current network traffic entering or leaving the node. The node's icon also varies according to what the node is. For example, servers, X terminals, and PCs are currently defined. The type of the node is currently not determined automatically, but simply read from the display configuration file.

Each of the pop-up segment displays has a colour coded key to distinguish between the traffic types. The colours associated with a particular type of traffic are read from the display configuration file.

The display is predominantly written in TCL/Tk with a small amount of C. The C code is used to extend *wish* (an interpreter for TCL/Tk) with functions to open and read values from shared memory as it is not possible to read the values directly from TCL. Extending wish in this way provides a seamless way to access the shared memory segments. The calls to the new functions are treated in exactly the same way as any other TCL function. Extensive use was made of the builtin TK widgets, particularly the canvas, for the graphical display. (See [Biddle, 1994]).

5 Using the Display

This section contains an example of the system working together to display network traffic. The snapshots presented here are representative of the actual operation of the gatherer and the display. Without the benefit of colour reproduction however, the protocols on the snapshots are hard to distinguish. Also, this example is limited to showing a single "snap-shot" in time. (The example uses our department network, where most nodes are named after Wellington landmarks.)

When the snapshots were taken, there were two suppliers running on the network. One was a PC (cable-car) attached to segment Comp8, and the other was a Sun machine (regent) attached to segment Comp59. This means that traffic can be monitored to or from nodes on either segment Comp8 or segment Comp59. If neither the source nor destination of the traffic is on these segments, then it cannot be seen. It also means that only on segment Comp8 and segment Comp59 can internal traffic be seen. For example, traffic between segment Comp5 and segment Comp6 cannot be monitored, nor can traffic within segment Comp6.

The gatherer, as well as the manipulator and the display, were run on the Sun machine lido, located on segment Comp8. In the example, a wide variety of protocols are monitored, including NFS, X windows, and NNTP.

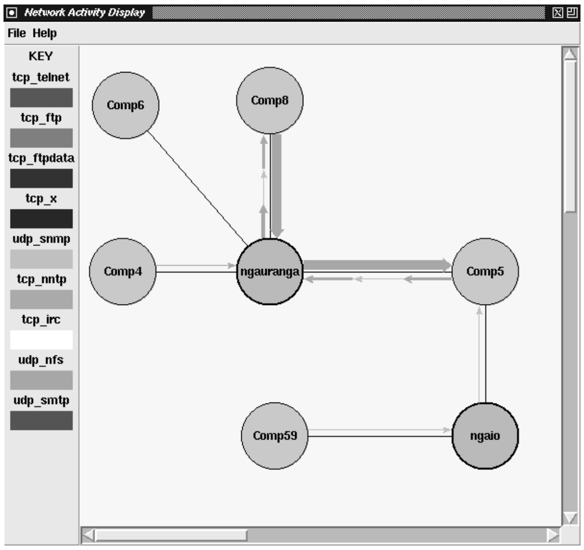

Figure 7: The network display

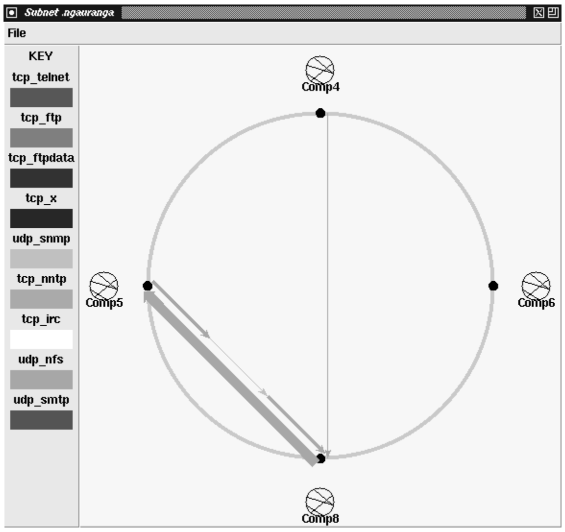

Figure 8: Expanded view of router Ngauranga

To enable the all snapshots to be taken at the same point in time, the manipulator was disabled. This stopped the transfer of information from the raw shared memory segment to the cooked shared memory segment. The display then continually redisplayed the same values from the cooked memory segment.

The first figure (see Figure 7), is the top-level network display. This shows the topology, as constructed from the network configuration file. In the figure, there is a large NFS flow from Comp8 to Comp5. There are also other smaller NFS flows on the network. An SNMP flow can be seen from Comp59 to Comp8. This is the NeMaC collection from the NeTraMet meter running on the Comp59 segment. This shows that the network activity gatherer monitors its own traffic.

One issue is not clear from this diagram: where is the NFS traffic from Comp4 going? To answer this question, and any other similar questions, the router diagram can be opened (see Figure 8). This shows an expanded view of the router named ngauranga. The user of the system can now see where the traffic is going; in this case, to the Comp8 segment. The other traffic between Comp8 and Comp5 is also displayed. As there was no monitored traffic to or from Comp6, there are no flow lines drawn.

To examine the traffic within a segment more closely, the user can click on the segment to get an expanded view. For example, Comp8 (see Figure 9). This shows the activity local to a segment between monitored nodes, i.e. the source and destination nodes are on the same segment, and traffic where only one of the source or destination nodes in on the segment. The diagrams shows that there is a lot of NFS traffic between the nodes debretts, bats, and lido, but that the amount of traffic is small. Once again, the SNMP flow from a supplier (cable-car) to the gatherer (lido) is shown. Other types of protocols present include NNTP and X windows. The X terminal salamanca was the display used when these snapshots were captured.

While the lines give an indication about the traffic flows, they do not give precise values. If the user is interested in the exact values of traffic flows they may click on a node to get a traffic window (see Figure 10). This window lists the sizes and protocols and flows into or out of a particular node. This window is particularly important where there are multiple flows on or off the segment. It allows the user to determine which flows are present, if the lines overlap.

Performance Limitations

The resolution of the display is limited by the speed at which the gatherer can provide new information. The best resolution obtained during the project was poor: 4 seconds. This means that new network flow information is only available every 4 seconds. Frequently, depending upon the network load at the time, it was longer. However, the main reason for the poor performance was the slow PC we used to run NeTraMet. With these delays, the display is no longer providing real-time network monitoring. What you are currently looking at occurred at least 4 seconds ago. The user currently has no way to know how old the information displayed is. This could be resolved by having the gatherer add a "time"

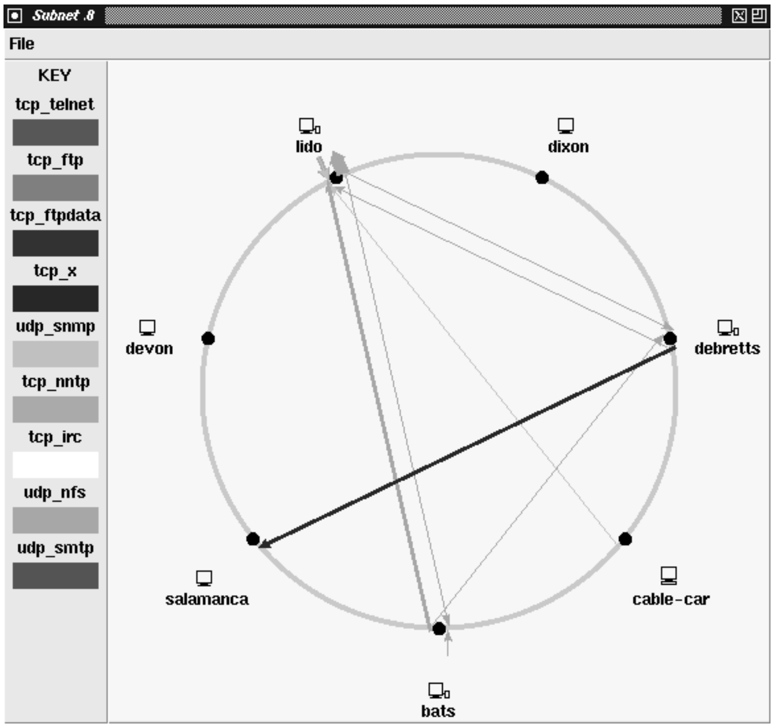

Figure 9: Expanded view of segment Comp8

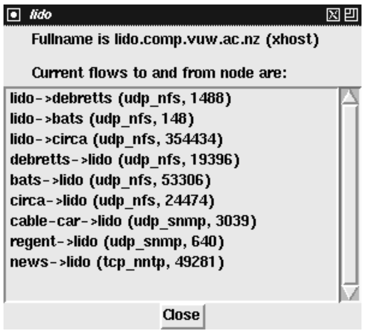

Figure 10: Network traffic to or from Lido

metric to the shared memory.

6 Conclusions

This was a project of two very distinct parts: a network activity gatherer using NeTraMet, and a network oriented display written in TCL/Tk. To consider the success of the project, the parts should be discussed individually.

6.1 The Network Activity Gatherer

The task of gathering network statistics, outlined in the introduction, has been accomplished successfully using the NeTraMet package. The new gatherer modifies NeMaC to identify and write network traffic flows to a CR shared memory segment. The NeTraMet program (running on the monitoring station) was used unchanged. The new gatherer meets the goals intended, as it is both portable and efficient.

The new gatherer also allows integration with the CR framework:

- The nodes to monitor are specified in a similar way to the other pre-existing CR components using configuration files.
- The new network statistics are stored in the existing node-metric oriented shared memory segment.
- The changes to NeMaC were localised to allow them to be easily incorporated into future versions of NeMaC.

NeTraMet was chosen as basis of the gatherer as it was an existing system with almost all the functionality required for this application. However, NeTraMet is in some ways unsuitable for the project. It seemed difficult to unite the NeTraMet rule file concept with the concepts of CR. As a result, some of the functionality of NeTraMet was wasted, consequently NeTraMet was less efficient than should have been, and flow from NeTraMet to the modified NeMaC was larger than it should have been.

In summary, however, the new gatherer performs the required monitoring task in accordance with the initially specified goals. The design could be improved to achieve gains in applicability, and consequent efficiency.

6.2 The Network Activity Display

The new display can graphically depict any protocol/port combination between any two nodes in the network. The display uses the standard CR framework, making it compatible with the other components of CR and limiting its dependency on any particular gatherer. The display comes close to meeting the goals intended:

- The display shows the network and flow topology. (Although in this prototype these are based on a topology read from a file, rather than detected automatically.)
- The display allows the user to selectively zoom on different parts of the network to receive more information. This was achieved using separate windows for each segment and router, and using a separate window to display the actual traffic values.
- Flows are displayed clearly by limiting their maximum line width. Limiting the width does not allow highly accurate network traffic representation; but it does allow network activity to be monitored and rough comparisons to be made. Scaling is used to give a reasonable range of values for the line's width across different protocols. Precise traffic flow information is available in an additional window and this represents a reasonable compromise between qualitative and quantitative information.
- Network diagrams may be loaded, manipulated, and saved to provide continuity between program invocations.

TCL/Tk was an ideal language to develop the display in. It provided an extremely rapid code-debug cycle due to its interpreted nature. Changes could be quickly tested and refined. The built in widget commands, particularly the canvas, made development more straight forward than it otherwise might have been. TCL's interpreted nature did affect the responsiveness of the display. This was due to the large number of mathematical calculations needed to place the lines representing traffic flow. The calculations were slow as TCL represents numbers as strings internally. However, the display is still fast enough to display information at the rate supplied by the network gatherer. The responsiveness could be improved by rewriting the calculation intensive code in C or some other compiled language.

We described earlier that all traffic is scaled in the same way. This was done to simplify the implementation of the project. Different protocols however, have different typical ranges for flow size. Applying the same scaling to different type of traffic, for example, telnet traffic and NFS traffic, may be totally inappropriate. But different scaling also has disadvantages. It limits the user's ability to make direct comparisons between different protocols, in terms of their respective bandwidth usage. The user is forced to refer to the actual traffic flow values to make precise comparisons.

The decision was explained in Section 4 to represent traffic on or off a segment using lines away from the segment. This however, leaves little room to draw lines and make lines hard to see if they overlap. This limitation makes the node traffic window very important to identify flows. The decision to not make the router a node on the segment should be re-evaluated in light of this. Alternatively, a new metaphor for displaying this type of traffic could be investigated. Another problem with router traffic occurs if there are two or more routers on the same segment: there is currently no indication which router the traffic is directed to.

In summary, the new display performs the required display tasks. It allows the user to select what they wish to see. The speed of the display is adequate to provide activity monitoring when used in conjunction with the network gatherer. It is limited primarily in ways relating to displaying traffic on or off a segment, and the lack of automatic network detection.

6.3 Project Design and the CR Framework

Currently both gatherer and display have limited flexibility. For instance, there is no feedback from the display to the gatherer, and a user cannot add another protocol or node to monitor. This is in large part due to current limitations of the CR framework. Basing the passing of values on indexed shared memory segments is quick and efficient, but not very flexible. There is a proposal to overcome this limitation: when changes to the nodes or metrics in the shared memory are needed, new index files would be written, and the CR components signalled to reset their indexes. They can then read the new index files and can continue with the updated indexes. Resetting the indexes is necessary due to the caching the components perform. The indexing must remain as it is the source of the speed of the CR system.

The design follows the format for the rest of CR in that individual nodes-metrics pairs are specified. This makes it easy to investigate small sets of nodes and metrics easily, but can be limiting for larger monitoring tasks. One solution would be to change the input format for the gatherer so it could automatically generate the required node-metric pairs. Another solution would be to create a high level language which compiles into the required node-metric pairs. More generally, the CR information could be expanded from two dimensional to a more general form.

This prototype for monitoring network performance been successful both in showing that network metrics can be integrated into the CR framework, and in showing where there are difficulties. In future work on CR, we plan to address these difficulties as discussed above.

References

Biddle, R., Hine J., Zhang, Z. *CR: A Monitor For Distributed Systems*, In Proceedings of the 10th Uniforum NZ Conference, 1993.

Biddle, R. *Graphic User Interfaces Made Easy? A TCL/Tk Tutorial*, In Proceedings of the 9th Uniforum NZ Conference, 1992.

Biddle, R. *Using the Tk Canvas Facility*, In Proceedings of the 11th Uniforum NZ Conference, 1994.

Brownlee, N. *NeTraMet and NeMaC*, Introductory Documentation, Auckland University, Jan 1994.

Flower, G. *SGM: SNMP Gatherer Multiple*, Diploma In Applied Science Report, Victoria University of Wellington, Feb 1994.

Gardner, F . *Automated Performance Management in a Distributed System*, MSc. Thesis, Victoria University of Wellington, 1994.

Reynolds, J., Postel, J. *Assigned Numbers*, RFC-1060, Mar 1990.

Mills, C., Hirsh, D., Ruth, G. *Internet Accounting: Background*, RFC-1272, Nov 1991.

Ousterhout, J. *TCL and the TK Toolkit* Addison-Wesley, 1994.

Schonwalder, J., Langendorfer, H., 1993 *How to Keep Track of Your Network Configuration*, In Proceedings of the 7th Large Installation System Administration Conference, The Usenix Association, 1993.

Zhang, Z. *CR: A Monitor For Distributed Systems*, MSc. Thesis, Victoria University of Wellington, 1993.