NAV - Network Analysis Visualization

Meghan Allen* Peter McLachlan†

University of British Columbia
Department of Computer Science

ABSTRACT

In this paper we discuss the Network Analysis Visualization (NAV) project, a tool for investigating high level network events. Monitoring network activity has been a critical task for administrators since networks were first introduced, both for maintaining system security and tracking resource utilization. Traditionally, the tools used by both enterprise and home users display detailed packet information in a scrollable text list. Although these interfaces provide fine grained details, it is very difficult to quickly obtain a high level understanding of the nature of the traffic. The Network Analysis Visualization project attempts to address this problem by using information visualization techniques to display packet trace data from a common log format. Overview with details on demand is used to provide different levels of viewing data, brushing and linking allows the user to select information for more detailed investigation and log scales are used where possible to allow data to span large ranges. Dynamic filtering allows the scope of data being displayed to be modified in real time, enabling the visualization to scale to large data sets.

CR Categories: H.5.2 [User Interfaces]: Graphical User Interfaces (GUI); C.2.3 [Network Operations]: Network monitoring

Keywords: network traffic analysis, information visualization, system administration, brushing, trellis, security, networks

1 INTRODUCTION

Internet communication has undergone dramatic growth in the past decade; almost all organizations make use of networking technology and the reach of computer networks has grown into the home with as many as 64% [25] of Canadians having home internet access with nearly two thirds of them accessing the internet via broadband [5]. As the volume of traffic travelling over the network increases, traditional network monitoring tools can no longer provide a good overview of the traffic patterns. These tools typically display every packet which is transmitted or received in a textual list. Although these interfaces provide fine grained details and allow administrators to drill into the packet contents, useful information can be obscured by the technical nature of the format and the sheer volume of information. Providing a high level visual means of browsing and filtering the data set allows the user to more easily detect information that may be of interest and obtain further details on demand.

Monitoring network activity has been an important pursuit for administrators since networks were first introduced. Most administrators have at least a few different means of monitoring traffic both 'on the wire' and for analyzing packet capture log files. This is typically a suite of non-integrated tools which provides them with a variety of information. The use of a basic packet sniffer can typically be assumed; tcpdump [12] represents an example of the most primitive form of packet sniffer available, and is capable of capturing traffic from network devices, performing basic filtering activities, and outputting the data in ASCII format to the screen. Although tcpdump may be used to perform physical packet capture, typically administrators prefer a more flexible means to view and analyze network traffic in the form of a packet sniffer with a graphical user interface. There are a wide variety of sniffers on the market such as the open source Ethereal [6] or the commercial product Sniffer Pro [20], and high priced hardware sniffing devices such as Fluke [9]. However, as can be seen in the screenshot from Ethereal in Figure 1, these devices are still providing the same basic textual view of the data with a few added GUI features to make it more accessible. Typically they provide the administrator with the capability to apply additional static filters to the data set and offer a more readable, scrollable means to view packet contents. However, these types of applications do not include techniques to present an overview of the data set nor do they have the capability to visually 'pop out' important information. Although administrators find many of these tools useful, they are still left adrift in a sea of data with little context to assist with navigation.

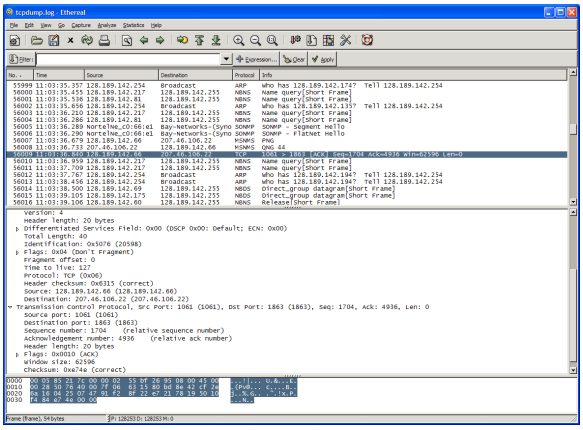


Figure 1: Ethereal is an open source network packet sniffer.

In addition to packet sniffers, an experienced administrator is also typically armed with a variety of auxiliary tools which are not integrated with the packet capture application. These include Intrusion Detection Systems (IDS) such as SNORT [23] that analyze packet payloads and port activity to determine when a network attack is taking place. The more simplified variety of these devices only detect port scanning activity, while more advanced variants provide alerts when one of a variety of known network attacks is taking place. These tools can typically be configured to log hostile payloads to a database. The database can then be monitored via a console such as ACID [1] which has a mechanism to send alerts to the administrator. A significant problem with these applications is that the considerable amount of background scanning and low level attacks performed by automated software running on compromised

*e-mail: meghana@cs.ubc.ca
†e-mail:spark343@cs.ubc.ca

hosts (also called 'zombies') potentially leads to a large number of false positives. Distinguishing a real directed attack from this background activity becomes challenging for the administrator; if too many alerts are received they may be ignored, but if the IDS is tuned to ignore certain attacks the administrator could miss important information. For example, a large scale increase in the number of below threshold hostile payloads might be an early warning that a directed attack was taking place, but might easily be ignored by an IDS tuned to discard notifications of that payload type.

The objective of the Network Analysis Visualization (NAV) project is to create a tool for visualizing network performance and connectivity data, with a secondary goal of integrating a facility to provide visual notification of hostile payloads for security awareness. Networking involves many layers of protocols, from the physical layer through to the application. The focus of NAV is the most commonly used protocols on the network and transport layer, with particular attention to TCP/IP and the services that run on this protocol suite. By distilling important traffic data NAV is intended to provide a useful complement to more typical sniffer interfaces, giving a higher level view of network activity.

2 Related Work

There are a number of projects in the information visualization space that have investigated the visualization of network data. Generally, these applications have focused on the security aspects of network computing by displaying such information to an administrator as when a port scan is in progress, when alarms have been triggered on an intrusion detection device or the source of hostile packets.

Visual Information Security Utility for Administration Live (VISUAL) [3] is a tool intended to help system administrators, particularly administrators of home systems, to rapidly perceive the security state of their network. VISUAL uses the concept of dividing network space into a local network address space and a remote network address space (the rest of the internet). It uses data from the log files of Ethereal [6] or tcpdump [12] to produce its visualizations. VISUAL provides a quick overview of the current and recent communication patterns in the monitored network. As you can see in Figure 2 a home and remote IP filter allows users to specify their home and remote IP ranges. The grid represents home hosts; by means of connection lines it is possible to see if a home host has received a large number of external connections at any given time. They can also see which single external host communicated with a large number of internal hosts, which may be relevant to detect network scanning activity. The amount of activity taking place relative to the local network by means of an external host is represented by the size of the external host box. The authors of the paper assert their solution can scale to a size of approximately 2500 home hosts and 10,000 external hosts.

PortVis [18] is a project that displays abstract security data. This project attempts to address the problem of giving outside consultants or security specialists access to corporate data, without entrusting them with information about the systems from which the data was gathered. The idea is to present these outside entities with data that is informative, but abstracted, to reveal as little information about the actual underlying systems as possible. In particular, information about the IP addresses of the hosts, labels, and network security alarm information is obscured. Basic summary information from each TCP port during a period of one hour is visualized by the PortVis tool, the goal of the project is to detect large scale security events while also allowing the identification of small scale events for further investigation. As you can see from Figure 3, several views of the data are made available to the user; key features include an overview located on the top right, a large window which represents port activity, and a number of detail windows to reveal

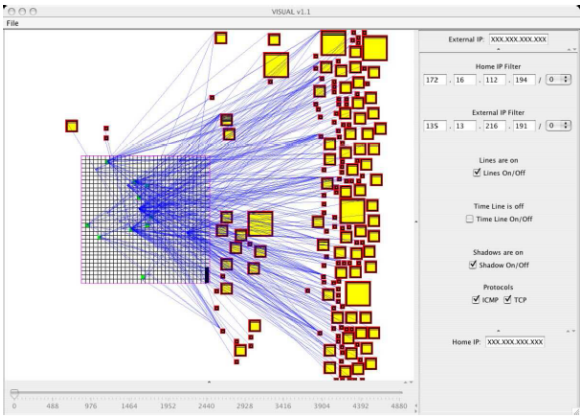


Figure 2: VISUAL is a tool to help administrators of home systems.

information about specific port activity. PortVis is focused on analyzing data from a single host at a time.

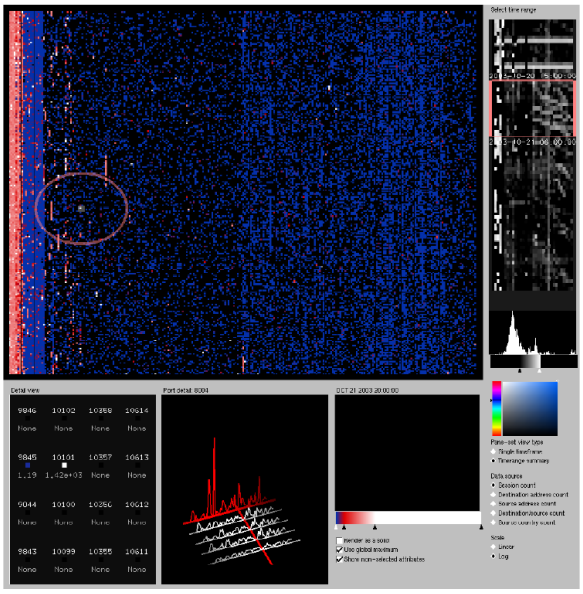


Figure 3: PortVis displays abstract security data.

The Spinning Cube of Potential Doom [16] is a network security visualization tool; the objective was to visualize sensor data from a large network and provide a complete map of internet address space indicating the frequency and origin of scanning activity. For this project a significant focus was to avoid creating another security visualization tool 'by security professionals for security professionals' and to give even a naive user some idea of the frequency and extent of network security threats. As you can see from Figure 4, the cube colours dots of incomplete connections using a rainbow colour map. Port scans appear in this visualization as lines, either horizontal if it is a scan across hosts or vertical if it is a single host's port space.

NVisionIP [15] is a visualization tool intended to improve the situational awareness of security administrators. Users are presented with a graphical representation of a class-B network and can choose from a number of different views of the data including a Subnet by Host grid based coordinate system, labeled clusters, and a treemap view showing individual hosts where size is determined by relative interest level of the given host.

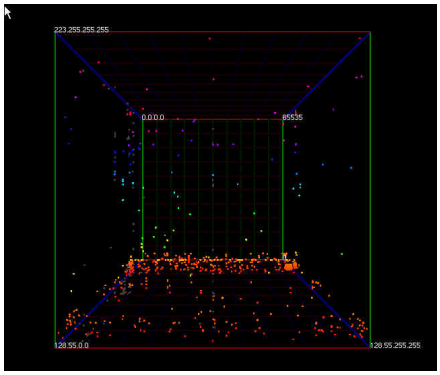


Figure 4: The Spinning Cube of Potential Doom.

Although security is a concern for home administrators, home firewalls such as ZoneAlarm [29], Linux routers or inexpensive hardware routing and Network Address Translation (NAT) [8] devices offered by a variety of manufacturers are very effective at reducing this threat. A greater concern may be determining which remote hosts are currently communicating with local machines as well as monitoring the use of bandwidth, especially given that many internet service providers have begun instituting bandwidth caps on home users and are either charging additional fees to users who exceed these caps or cutting off their access entirely.

3 Requirements

Requirements for our proposed system were generated based on the experience of the authors with home system administration and analysis of network sniffer traces in a corporate environment. These experiences provided a good working foundation for determining what tasks a visualization of network traffic needs to support. However, we also felt it was valuable to verify these intuitions with another experienced administrator and an interview was performed to get feedback on our ideas and to generate new suggestions.

3.1 User interview

We conducted a user interview to determine if our understanding of the information that our tool should visualize was correct. We interviewed a Computer Science professor with a background in distributed computing and networking, both as an academic researcher and as a Chief Technology Officer in industry. Our user interview helped support many of our intuitions about the data that our tool should visualize; it was established that per-service traffic flows were of the highest importance and that the user was less concerned about the actual packet internals. Our design sketches were refined from the feedback provided by this system administrator. Based on this interview we developed several new ideas for our design including using expanding bar graphs next to each IP address in the wall view to signify the amount of traffic sent to that host.

3.2 Home users

Most users who have used a home firewall product such as ZoneAlarm [29] are familiar with the number of unsolicited connections received from the outside world. Users of these products may also have been surprised by the number of connections initiated from their own computer to external hosts without their knowledge. This information is very useful to a home administrator for detecting malware or other malicious activity on their system. ZoneAlarm presents a single dialog which vanishes after user acknowledgement; however, this doesn't provide an easy means for the user to perform further investigation into the source and destination of this potentially surreptitious traffic. Building on our team's experience with these types of products, we felt an important requirement was to provide a view that explicitly connected local and remote traffic, distinguishing a strong boundary between the two. Further, it was our experience that quantity of traffic can also be a good indicator of a compromised host. Crackers frequently use compromised 'zombie' machines for a number of purposes such as Distributed Denial of Service (DDOS) type attacks, distribution hosts for illicit software, or relays for other illegal activity. All of these involve the use of considerable bandwidth; from this information it was felt that giving a user the means to detect large data transfers, particularly in time ranges where they might not be expected, is critical for security awareness.

A home user may wish to monitor their network to determine the amount of data being downloaded and uploaded, as well as what services are using the most network bandwidth.

3.3 Corporate users

The needs of a corporate administrator do not differ dramatically from the requirements of a home administrator although the volumes of traffic may be on different scales. Corporate administrators need to divide IP space into a 'local' vs. 'remote' topology, designated by the point of demarcation with the internet service provider (ISP). Since Classless Inter-Domain Routing (CIDR) [22] has become an Internet standard, small businesses can own as few as one internet routable IP address up to as many as tens of thousands. Often corporate users will only be interested in one subset of local addresses at a time; for example it is common for businesses to implement a 'demilitarized zone' (DMZ) on which servers which provide external services are placed. Although these servers are typically protected by a firewall, they are also segregated from the internal network by a separate firewall interface. Administrators might be interested in what traffic is flowing through the DMZ; in a corporation with a substantial internet presence or portal this may represent the majority of their traffic. Alternatively, administrators may be specifically interested in what traffic is not flowing through the DMZ, especially if large transfers are not expected from external hosts directly through the firewall to internal trusted machines.

Typically, an enterprise system administrator would like to know which machines in the corporate network are being accessed, what services on these machines are being used, and how much data is being moved. An administrator may also be interested in seeing patterns of accesses to local machines; for example, accesses from a single remote computer to a sequential group of ports on a local host may indicate a port scan is in progress. Usage spikes for unexpected services can be an indicator that a compromise has taken place; for example if a system typically dispenses web traffic but has a sharp spike in ftp-data transfers, it may be an indication that the system is being used as a cracker contraband distribution hub. Administrators actively monitoring the network may be interested in having certain information pre-attentively 'pop out', such as when packets are transmitted containing known hostile payloads, which would allow them to take immediate action against the originator.

4 System Design

We have determined that our solution should focus on the replay of log files; the ability to capture live traffic is discussed in section 9. Log files are obtained in a common format used by all applications

which have the open source pcap [26] library as their capture interface. As an example, log files generated by Ethereal [6] or tcpdump [12] can be used. Because it is a feature of the pcap library to provide a unified interface to reading in a log file or from a network interface, extending the application to accept live data in real time will require only a modest rewrite of some of the internal structures and can be considered as a future expansion to the project.

When the log file begins, users are requested to provide a number of pieces of information to NAV including the time range for the capture, an optional filter, and both the local network range and mask. The pcap library implements a bit packet filter which accepts a filter definition in the form of a string that our application passes from the user to the pcap library. More details about the filtering capabilities are available in section 4.1.5. The local network address and mask must be provided by the user in order for the NAV application to determine which addresses represent remote and local hosts; from a raw capture file there is no automated mechanism for making this distinction. The network address is specified in standard internet 'dotted quad' notation for IP addresses in the form V.X.Y.Z where each of {V,X,Y,Z} are a decimal range between 0-255. Internally each of these 'octets' is stored as an eight bit integer and taken together form a thirty two bit unsigned integer. The network mask is also provided in dotted quad format and when translated to binary represents a mask of ones followed by zeros that demarcate the bits specifying hosts on a subnet from the bits specifying the network address. The network address is then derived by taking any IP address inside of a subnet and applying the 'mask' for that subnet which involves performing a bitwise logical AND operation between the IP address and the netmask. Many system administrators are familiar with, and still use, the much simpler classful system of specifying internet addresses where the smallest address space available was a 'Class C' block of 253 usable addresses. However, Classless Internet-Domain Routing (CIDR) from RFC 1519 [22] has been the standard for some time now and we felt it was important to support more flexible IP range configurations at the expense of the added complexity of requiring the user to specify a full netmask. The user is free to specify an initial time range for the log file that exceeds the actual duration of the log file; providing the ability to select a time range in the open dialog is provided simply as a convenience if the user knows in advance a particular time frame they are interested in.

We intend to provide two primary 'views' into the data flow, a service and an IP view. The IP centered model involves the creation of two IP 'walls' of addresses with one side representing local addresses and the other remote hosts. The local address range is specified by the user of the application. Lines are drawn to connect the local and remote hosts, with further information such as traffic type encoded as line colour and line width indicating traffic volume.

Users can 'aggregate' ranges of IP addresses to reduce the number of line crossings. Dynamic queries are implemented with the use of a time slider to help users sort through the data. Filters are applied to both views simultaneously, which allows them to further reduce the number of crossings on the wall as well as simplify the graphs in the services view. Overall the visual focus of the IP wall view is to provide the capability to reduce edge crossings and provide a scalable visualization.

The services view is based on a trellis of 2D scatter plots or line graphs. The trellis view is inspired by R. A. Becker's The Visual Design and Control of Trellis Display [4]. Each graph represents a service; the x-axis is time and the y-axis is bytes/s of traffic. The IP and services views can be brushed to display the packet detail in the detail view. For example, if the user click and drags the HTTP line graph onto the detail view, the detail view will display all the HTTP traffic until the filter is reset or a different brushing operation takes place. Similarly, dragging an IP from the wall view onto the detail view ties the detail view to showing all packets from that specific IP. One scalability enhancement that can be made to this view is to allow portions of the time axis to 'stretch' or 'compress' through user interaction allowing them to gain details on demand. If this proves too problematic the user could be given the ability to 'pan' through the time axis by scrolling left and right on the graph.

The two primary views were implemented vertically because the 'wall' view was better suited to this orientation whereas the services view could be laid out either way. The detail view benefits from a horizontal orientation because it allows users to see all the details from each packet without having to scroll horizontally. The horizontal orientation of the detail view allows NAV to display packet details without using much of the space which is beneficial to our two main views.

4.1 Features

This section will discuss and describe all of the currently implemented features of NAV. Unimplemented features will be detailed in section 9.

Our default system displays the IP wall view on the left side, the services view on the right side and the detail view in a frame which spans the full width of the application frame across the bottom. Each of these views is collapsible or expandable with a single click. The proportion of the screen taken by each view is adjustable by means of split pane sliders. When a log file is opened, users are able to specify a filter at that time. We have selected 8 default Services to display, which are web traffic (http), SSL encrypted web traffic (https), FTP data (ftp-data), IRC (irc3), bittorrent activity (bittorrent), Microsoft Messenger (msnp), email in the pop3 format (pop3) and Kazaa file sharing (kazaa). These are user configurable, which will be discussed below in section 4.1.8.

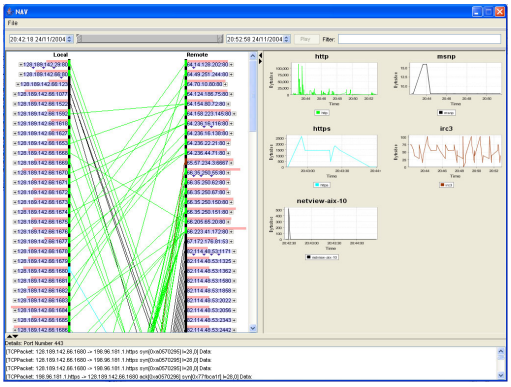


Figure 5: NAV Overview.

4.1.1 IP Wall View

The purpose of the IP wall view located on the center left frame of the application is to provide information to the user about what local and remote Internet Protocol addresses and ports are generating network traffic and to provide rapid visual feedback concerning how much traffic is being transmitted between which hosts. When the system is in playback mode it is possible to see new connections being formed between IP:Port pairs, as well as new local or remote addresses that are becoming active. The VISUAL project [3], as discussed in section 2 uses a similar local and remote distinction.

In the IP wall view local addresses are placed on the left side of the display and remote addresses are located on the right separated from one another by a pair of lines that represent 'the wall'. This

design visually forms a barrier between the two types of addresses. Connections are represented by coloured lines that connect a local address to remote hosts. Initially only ten default services are colour coded using a subset of Ware's [28] twelve recommended colours for use in colour coding (excluding black which is used to denote uncoded services and white which is the background colour of the display). The default services and colours are discussed in section 4.1.2 and more details on colour selection can be found in section 4.1.7. When a connection is established between an IP:Port combination the section of the 'wall' beside the IP label is also modified to show the colour of the connection. This helps to establish a visual link between the IP label and the connection.

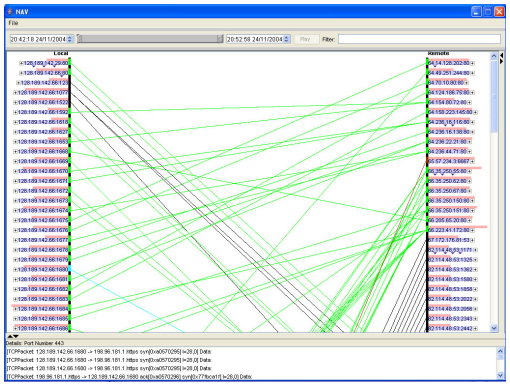


Figure 6: The IP wall view. The left list represents local addresses and the right represents remote addresses. Connections are represented by coloured lines. Widgets are provided to allow ranges to be collapsed and to disable visualization of connections associated with a given element or range.

The wall view contains a number of key features to reduce the number of edge crossing 'snarls' that occur when there is a large quantity of traffic being visualized. One of the most important features is the ability to visually 'collapse' a port or address range for both local and remote hosts. Address ranges can be collapsed to a single entry for a class A, B or C network address range, and individual hosts can have all of their ports collapsed to a single entry for that host. When a collapse takes place, all of the connections that originated from 'child' entries are moved to their parent representative. One consequence of collapsing a large range of ports or addresses is that multiple services are typically contained in a single entry, but this information can no longer be clearly coded. Possible solutions to this problem are discussed in the Future Work section.

When a user is not interested in seeing the edge connections for a given port, host, or a network class of hosts, the user can simply collapse up to the desired level and then click on a trigger on the 'outside' of the local or remote label to disconnect that address from the wall. The result of this disconnection is immediately visible, providing important feedback to the user. The IP address label slides to the left or right (depending which side it's located on) indicating it is disconnected from the wall, and all connections to that object vanish. However, the wall section colouring remains as a visual indicator to the user as to which service type was connected to that label.

The wall view is also intended to convey information about important events that have taken place on a connection between local and remote entries. Although this feature is not fully implemented by NAV at this time the underlying code to support this capability has been integrated into the code base and possible future directions are discussed in section 9.1.3.

Information about the amount of traffic between any two points is encoded in the form of a horizontal bar that displays just above the label for a given local or remote entry in the IP Wall View. This bar uses a base two logarithmic scale, where the largest bar indicates the IP that has transmitted the most traffic. Using a logarithmic scale enables huge disparities in transfer rates to be distinguished, while still permitting an accurate comparison of smaller transfers. If a linear scale were used a large download would result in smaller transfers becoming too small to visibly compare, possibly to the point of making them imperceptible. Users can tell at a glance which local or remote addresses are responsible for the most traffic on the network. When network ranges or ports are collapsed all of the transfer statistics for those entries are aggregated into their parent representative.

The IP wall view can quickly show users when a port scan has taken place; the port scan is visible as a series of connections from a single remote host to a series of ports on a local host. This visual technique allows NAV to show 'stealth' syn scans as well as regular port scans. During a stealth scan a tcp synchronize request is made but the port scanner only responds with an RST (reset) request which in many cases avoids a log being kept of the connection activity because a full connection is never established. An example of a port scan can be seen in Figure 7.

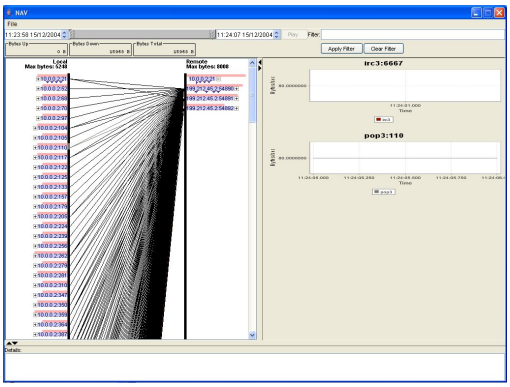


Figure 7: A portscan taking place in NAV, a single remote system is scanning a sequence of internal ports.

Although the wall view is provided in a scrollable region of the screen to allow large quantities of data to be displayed it is intended that users will use the ability to collapse and disconnect edges to focus their attention on connections of interest. The rendering code of the wall view is quite fast, with a worst case draw time bounded by a single O(log n) binary search required to locate the far side of a connection drawing, and of course by the speed of the Java draw code itself. The draw code does not draw IP labels or calculate traffic statistics for entries that are not currently visible on the screen. In several tests the draw code scaled to hundreds of addresses with thousands of connections with draw performance better than 40ms on a 1.4ghz P4.

4.1.2 Services view

The services view shows the traffic for particular services over time. The traffic corresponding to each service is shown in a separate graph. If there is no traffic in the log file related to a selected service, the graph is not shown. Currently, the services view only displays the default or user selected services, so there may be activity in the log file that is not displayed in the services view. The services view has the ability to show up to twelve graphs at once.

The default number of graphs is eight, but this can be changed on the Preferences dialog. The services that will be shown can also be changed in the Preferences dialog, further information on the Preferences dialog can be found in section 4.1.8.

Each service graph displays traffic to and from the local host(s) on a specific port. We label each graph based on our best guess of what service utilizes each port but as ports are not restricted to specific applications we cannot guarantee the accuracy of our labels. Each graph only displays the section of time in which activity occurred on that port. For example, if your log file was 20 minutes long but FTP traffic only occurred in the last 5 minutes, the x-axis of the ftp-data graph would only display the last 5 minutes.

The graphs in the services view can display either a linear, or log based date axis. They use a linear date axis by default, but this option can be toggled in the Preferences dialog. The log based date axis was implemented to allow users to view log files with a very large range of time values. The current implementation is a first pass at solving this problem, but further work could be done to improve the log based date axis.

Figure 8: The Service trellis graph view.

4.1.3 Open Dialog

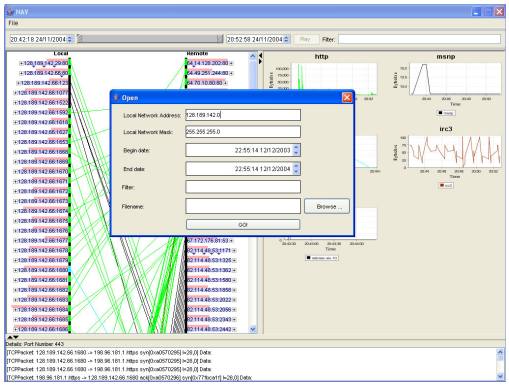


Figure 9: The open dialog.

As can be seen in Figure 9 the open dialog provides text fields and spinners to gather the minimum information necessary to begin visualizing the data. Users can specify a beginning filter and a time range at their option which may be able to perform an 'early cut' on the size of the data set being visualized. This information is optional; the filter can be left blank and the spinners default to a span of a year based on the current date. The local network address, netmask, and the file name of the log file to be opened are the only critical pieces of information in this dialog. Simple verification checks for format correctness are performed before the log file is loaded.

4.1.4 Detail View

NAV complies with Schneiderman's 'overview, zoom and detail' mantra [24] by providing a means to 'peer' into the actual packet data if they choose. Overview is provided by the IP Wall and services views, and the 'zoom' equivalent functionality is achieved by narrowing the time slice displayed using the time filter discussed in section 4.1.6. Details are available by means of the detail view that is located at the bottom of the screen. This view is initially empty, but users can drag and drop to the detail view from either the IP wall view or the services view to populate it. From the wall view, users can drag an IP address label to the detail view, which will cause the detail view to display all packets to or from that IP address. From the services view, the user can drag one of the graphs to the detail view which will cause the detail view to show all packets to or from that port number. The detail view contains a label which denotes the IP address or port number that the packets are from and the list of packets. For each packet, the detail view displays the source IP address, source port, destination IP address, destination port, the TCP or UDP flags and the data. We felt it was important to allow the users to see the details to supplement the two overviews we have provided. The detail view only shows data from either one IP address or port number at a time. Every time an item is dragged to the detail view, the current information is erased and only new information is displayed.

4.1.5 Textual Filters

We allow users to specify filters on the log file to limit the amount of information that is displayed. Users can filter the data by attributes such as IP address, IP address range, port number, time or protocol. This is just a small sample of the many different ways that data can be filtered. We simply pass the filter to the jpcap library which performs the filtering, so users have the full power of a bit packet filter. Complete reference documentation can be found on the tcpdump manual page available in UNIX or on the tcpdump site [12].

4.1.6 Time Filter

NAV has a double edged slider in the Filter Window that allows users to filter based on time. The slider is initially set to contain the full time period of the log file. As the user moves the slider both the IP wall view and the services view update accordingly. This allows users another way to narrow down what they are seeing, and concentrate on specific details that they find interesting. The slider is accurate to the millisecond, although feedback to the user is provided primarily by means of two 'spinners' that represent the beginning and ending time of the selected range which are only accurate to the nearest second. We surmise that this is sufficient accuracy for most purposes. The spinners can also be used to set the beginning and end time either through direct input or via a pair of arrow widgets. If the user chooses a subset of the capture time range using the slider or spinners the 'play' button becomes enabled. The user can click on this button and the range will advance one second at a time, allowing the user to replay events as though they were happening in real time.

4.1.7 Colour

Our ten default colours are selected from the twelve distinguishable colours listed by Ware [28]. We use black for services that are not assigned a specific colour and we do not use white because it is the background colour for the IP wall view. Our default colour selections have been checked with VisCheck [7] and they are discernible by people with both protanope and deuteranope colour blindness. Although cyan is distinguishable from white, it is difficult to see a cyan line on a white background for people with protanope colour blindness so we avoided using cyan for the most common services. The pink and orange colours we have selected are indistinguishable for people with tritanope colour blindness, but this is a very rare form of colour blindness. VisCheck only claims to be accurate when used with a calibrated monitor, but neither author had access to a calibrated monitor. We felt that using VisCheck on our own monitors was more useful than not using it at all.

Colour preferences are user selectable via a preferences dialog and these selections apply to both views of the data. Users are free to provide colours (using a colour picker) for as many service types as they choose; they are not limited to coding with only twelve colours. Although the academic literature suggests users may have trouble distinguishing colours above the limit established by Ware, we considered that it was important to allow the user to choose their own preferences and simply provide safe defaults. Because there are no constraints on the colours that users can select, we cannot make any guarantees about the discernibility of the colours once they have been changed.

4.1.8 Preferences

We have implemented a preferences dialog (Figure 10) which allows users to select the services that they wish to see in the services view. The UNIX services list was used to provide an exhaustive index of available services, which is presented in alphabetical order. Users can choose the order in which services will be displayed as well as select the maximum number of services they wish to display at once. The current choices are 6, 8, 10 or 12, although a possible future change could make this completely user configurable. The preferences dialog is where the users can associate a colour with a specific service. The chosen colour will stay associated with the service throughout the run of the application. Preferences are not saved between sessions at this time.

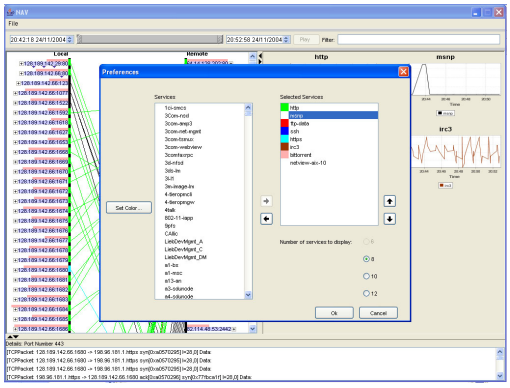


Figure 10: The preferences dialog.

4.2 Implementation

NAV is written in Java (compliant with version 1.4.2) using the Eclipse Platform. The graphs in the services view are created with jFreeChart [10]. We used jPCAP [14] as our network traffic packet capture library, which is a native library interface for the pcap[26] capture library written in C. The InfoVis Toolkit [13] was used to provide the double edged Time Filter slider.

4.2.1 Performance and Scalability

Although the views themselves scale effectively, limitations of the underlying packages and certain language features have created unavoidable performance problems. Java does not cope well with the bit and byte level analysis that is required for packets. One particular problem is its lack of support for unsigned primitive types; this often means that larger data types must be used to store standard values. For example, an IP address is typically stored in a single integer since it contains 32 bits of data, but only 31 bits are available in a Java integer and therefore it must be encoded as a long data type of 64 bits. Worse yet, java 'bytes' provide only seven usable bits because they are also signed. To represent eight bits it is necessary to use a pair of bytes at considerable extra overhead and expense. Java lacks many important low level functions to operate on bits; in fact it only provides a Bit class and not a primitive bit data type. There is no good means to create bit arrays for example, and the Java BitSet class lacks methods to convert to or from common data types. Writing convenience methods to perform these functions requires awkward and inefficient looping structures.

Data is retrieved from the capture file with reasonable performance; however, actually extracting data from the packets is an expensive operation. Simply retrieving a single field from a single packet can be on the order of 20-30 ms on a Pentium IV 1.4ghz. When there are many fields to be retrieved from each packet and tens of thousands of packets to be processed, this quickly makes the possibility of any real time processing of packets infeasible. Despite the fact that the inner loop to retrieve packet data is O(1) it remains infeasible to perform analysis of large data sets using NAV at this time.

During the performance tuning phase a number of optimizations were made to the source code. One major performance bottleneck rests in the IP wall view where every packet received must have its source and destination addresses extracted and compared against another linked list to determine if this packet is associated with a previous communication. The initial algoritihm for this check was a simple linear search and this section of code was the largest performance bottleneck. This problem was addressed by changing the linked list to a hashtable data structure which transformed from a O(n) worst case performance characteristic to an O(1) algorithm. Changing the sequence of packet processing methods was also important, for example, initially an O(n log n) sorting algorithm preceded another method that did not yet require an ordered list and provided additional filtering on the data set. By reversing the order of these methods, the expense of the sorting algorithm was considerably reduced. Because creating and recycling Java objects can be an expensive operation, static factory methods and freelists were used to manage frequently created and recycled objects. A freelist tracks objects that are no longer in use, and recycles them by clearing specific fields in the object before passing them out for re-use. This adds to code complexity because each object that is used must be explicitly retired to the freelist, bypassing Java's automated garbage collection, but when small user interactions can result in thousands of objects potentially being created or destroyed there are tangible benefits to recycling.

The draw code for the IP wall view has a worst case performance bound of O(log n) where n is the number of remote labels. This is because the number of labels grows relatively slowly in com-

parison with the number of packets; for example most hosts/ports communicate at least a few dozen or even a few hundred packets. Although the pre-processing time is quite high, the application was tested with a 10MB data set and once the data was displayed, collapsing, scrolling and redraws were found to function adequately with a redraw time on the order of 150ms on a Pentium IV 1.4ghz class computer.

5 Discussion

The authors of VISUAL [3] addressed a similar set of problems to NAV. There are many similarities in the design decisions made for NAV and Visual, and we feel that this supports many of our underlying assumptions. Both NAV and Visual divide the machines monitored into Local and Remote and both systems allow you to specify groups of local computers for monitoring. Both systems also display the ports and protocols used, as well as the amount of data transferred. Although VISUAL is a much more polished product, we feel that NAV is able to solve many of the same problems and visualize the same data in an effective manner.

6 Evaluation

Two cognitive walkthroughs were used to evaluate NAV. A cognitive walkthrough is a theoretically structured evaluation technique which involves asking specific questions regarding the interface [17]. A detailed description of the cognitive walkthrough process is available in Lewis et al.'s paper [17].

6.1 Cognitive Walkthrough 1

We developed a cognitive walkthrough that focused on the scenario of a user attempting to discern information about security related activity that might have taken place on their system overnight between midnight and eight am. Because the user was not using their system interactively during those hours, the only expected traffic was the large bittorrent and ftp downloads that were left running overnight. The walkthrough focused on each task the user would need to perform and the individual goals of the user at each step. From this walkthrough it was clear that to use the IP wall view effectively considerable filtering must be performed. The user was interested in discovering what was causing an unusual amount of upload traffic, and had to narrow the time field using the time slider and seek through the log information to realize their bittorrent client was also uploading data to other clients. Because the relevant hosts were farther down on the list the user was forced to scroll as well as 'collapse' elements of the wall. Scrolling may have a detrimental effect on the users context awareness. It was also determined that the user would need at least modest familiarity with networking terminology to use NAV effectively.

6.2 Cognitive Walkthrough 2

A second cognitive walkthrough was completed to determine whether users could easily use NAV to ascertain which services are being used on their machine and which service consumed the most network resources and whether their bandwidth limit was exceeded. As in the previous walkthrough, user goals were considered at each step of the process. It was determined that users will be able to extract this information from NAV, but a few questions were raised. First, in order to use NAV users must realize that they can log network traffic going to and from their machine and they must know how to log this data. NAV does not currently have any documentation and some documentation may be necessary for users who are inexperienced with networking. Users will be able to determine which services are being heavily used by scanning the IP wall view and the services view. However, since it is possible that some services from the IP wall view are not displayed in the services view, there is the possibility that a heavily used service would not be represented in the services view. This could cause the user some confusion. A third issue that arose during the cognitive walkthrough is that the y-axes in the service view graphs can all be different. Each axis's range is set based on the specific data in that graph. This detracts from the usefulness of having multiple graphs close to each other because they cannot easily be compared.

7 Lessons Learned

We learned several valuable lessons while working on the NAV project. An important goal of our project was that our solution be scalable to very large data sets; we discovered that implementing scalable graphical user interfaces in Java can be very difficult. Although Java is quite suitable for some tasks, we did not anticipate the added complexity and performance impact of having limited support for low level primitive operations on bits, bytes and unsigned integers. We also learned that the complexity of individual methods can be crucial to the performance and scalability of an application. Methods that are called during the drawing phase should be O(1) if possible, and O(n) at the worst case. Any algorithms in the critical path that exceeded these limitations caused a noticeable lag in our application's refresh rate. We initially implemented NAV without much consideration to the time complexity of the methods. Once we realized that performance was an issue we were able to redesign several methods to reduce their time complexity, and in some cases reduce the time complexity to O(1) from O(n) or in one case an $O(n^2)$ algorithm.

A considerable volume of software engineering and design literature describes the importance of involving users in the design process [2], and during the course of this project both of the authors felt the need for this requirement was reinforced. More user input throughout the design process would have been extremely useful when design decisions were being made. Due to the tight time constraints of the project, we did not feel we had time to gather further user data. In reviewing design decisions and considering that backtracking that was necessary on a number of occasions we must concede the possibility that we would have saved time overall if additional user feedback had been gathered and used to guide our development. Good design is an iterative process by nature.

Creating an effective model that provides both overview and details is challenging and requires good planning. It can be difficult to decide which dimensions are important enough to display in the overview. Even simple questions like how much space to devote to the overview and detail view can be challenging. We decided to allow the users to manipulate the relative sizes of our views but providing default values required intuition as well as trial and error.

The IP wall view visualization requires the user to scroll when there are too many labels to fit on the screen. Unfortunately scrolling violates guaranteed visibility [19] and can result in a loss of context. Another means of encoding this information or exploring alternatives to scrolling such as zoom/pan operations might be better.

There can be a tradeoff between documented information visualization techniques and allowing users to decide how they want the system to look or feel. In our case, we felt it was important to allow the users to select colours for their services, but by allowing them to do so, they may pick colours that are not distinguishable from each other. Despite this fact, we felt empowering users should be a priority design goal of any usable system. Perhaps an optimal approach would be to present the user with a visual warning when poor colour choices were made, or to provide a 'recommendation' wizard.

8 CHALLENGES

We ran into several challenges while working on the NAV project. Initially we were planning on using the InfoVis Toolkit [13], but both authors struggled to implement the necessary visual components using the Toolkit. Instead, we opted to use jFreeChart [10] for the services view and Java2D for the IP wall view with the InfoVis toolkit used only to provide the double edged slider for time filtering.

The scalability of our system is crucial because its usefulness depends on the capability of displaying large data sets. We did have challenges in obtaining a large log file to test the scalability of our system. After acquiring a large data set (on the order of 100MB) we spent a considerable amount of effort to optimize our algorithms. Ultimately, the underlying methods provided by the libraries we are using to access the data became the key bottleneck. Retrieving data from the packets is a slow operation; it involves obtaining a series of byte offsets inside of what is essentially a large byte array, and translating these into 'long' 64 bit integers, and from there into more useful forms of data such as text strings.

We were hoping to implement efficient dynamic queries for the filtering of data. Unfortunately, jpcap did not fully implement the functionality that we required; the library was designed as an interface for live packet captures and the support for reading log files is an auxiliary function. Once the data is read, there is no way to modify the filter to perform live changes on the data set. The only way to re-filter the data is to read the log file again from the beginning with a substantial up front cost in reprocessing the packets. The only alternative is to use a complete bit packet filter in Java, which our research suggests does not currently exist and was beyond the scope of this project to implement. First hand experience with Java's poor support for low level primitive types indicates it would not be worth the effort to perform this implementation. We have implemented the ability to change the filter at run time and reload the data, but this takes the form of a static, rather than dynamic, query. Another limitation of pcap is that though time stamp information is associated with each packet, the library does not provide the capability to filter based on a time range. Fortunately this is one area where it was relatively trivial to write support in Java to query the packet data to derive ranges based on time values.

Writing the necessary low level bit functions was time consuming and frustrating, particularly the lack of eight bit bytes which is a low level primitive type fundamental on most computer architectures. It is the belief of the authors that other programmers may have likely written utility classes to perform most of these functions but these don't seem to be available on the internet. Most Java code samples focus on higher level functions.

9 FUTURE DIRECTIONS

In this section we discuss features and capabilities we would like to see added to or expanded in the NAV system. As the project progressed it became clear that our desired feature list was growing much faster than the implemented feature list. We document here a few key ideas that we felt had the potential to dramatically improve the application.

9.1 Animation

Animation represents a valuable Information Visualization technique by helping provide users with context, feedback to input as well as being a pre-attentive cue to draw focus to a specific region of the screen. There are a number of areas where animation could be used more effectively in the NAV application.

9.1.1 IP Wall ghosting

In the current implementation of the IP wall, if the time slider is moved the IP wall view is immediately updated to display the changed information; however, the immediacy of this response comes at the cost of user context. We propose 'ghosting' as a solution to this problem where previous connection lines and labels do not immediately disappear but instead are blended with the background over some duration allowing the user to see connections that existed in previous time periods. When the user is in 'playback' mode, old connections would slowly fade out using the same mechanism.

9.1.2 Animated Connection Hiding

The current approach to 'disconnecting' an entry from the IP wall view is quite sudden and jarring. When a user clicks on the [+] next to the label, the label shifts inwards instantly and the connection lines to this object are removed. We propose using a smooth animation technique similar to what is proposed in the van Wijk paper on Smooth and Efficient Zooming and Panning [27]. Even though the path that is taken by the label is quite short, it could still benefit from the natural motion inherent in the accelerate/decelerate technique described in this paper.

9.1.3 Intrusion Detection

The underlying framework for providing a means to graphically display intrusions in the form of animation on the IP Wall view has been integrated into the code but the feature remains incomplete. Several different approaches were examined including causing connection lines to 'blink' or to change to a specific colour to alert the network administrator of hostile activity. Based on Ware [28] it may be best to provide some form of a glyph or icon that moves; this would help to work around change blindness as documented in visualization literature [21] and provide a pre-attentive means of drawing the users attention. The central idea is to provide a set of animated glyphs which would indicate a type of hostile payload. The glyph would be presented along the connection line between the two labels and would have a directional indicator to indicate the source and target of the attack, when the user clicked on the glyph the detail view would be updated with the specific packets involved in the hostile payload.

Beyond simply detecting hostile payloads it would also be helpful to implement a means for the user to define lists of hosts that do not normally accept connections from the internet, or IP's in the local range for which no hosts exist and highlight traffic that attempts connections to these hosts. Because no requests have been issued from these IP's, traffic directed to them from the internet is by definition unsolicited and almost certainly represents a network scan.

9.1.4 Direction of traffic

In the current implementation the IP wall view does not distinguish between incoming or outgoing traffic. This information could be crucial to a network administrator; if there is a large volume of traffic between a local and remote machine it may be imperative to know which direction that traffic is flowing. One possibility would be to animate the connections between remote and local machines to convey the direction of traffic.

9.2 IP Wall section colouring

Currently the IP wall view can only associate a single service colour with the section of wall immediately adjacent to the label. This serves well when the tree has not been collapsed since generally a

single IP and port will only be associated with a single remote service at any given time. However when the ports of an IP address are collapsed or when multiple IP ranges are represented by a single 'parent' label there may be connections for many service types encoded. These services are visible by the connection colourings when the label is connected to the wall; however, if the user disconnects this entry from the wall, information about all but one service is no longer visible to provide the user with context. A solution to this problem could be to make the section of wall into a grid of colours that can display a colour swatch of each service associated with that entry.

9.3 Tooltip support

There are a number of areas of the application that could benefit from the addition of tooltips. One key addition would be to provide tooltips when a user mouses over a label on the IP Wall View which provides DNS resolution for the specified IP Address. A lookup could also be performed on the service name associated with the specified port. This could be important in helping the user to identify remote hosts they don't recognize.

9.4 Live packet capture

The capability of providing live packet capture on the interface was explored but limitations of the underlying packet capture software library, which blocks waiting for packets, have made this challenging to achieve within the time limitations of the project. The framework for live traffic capture has been integrated into the code but some data structures and calls must be made thread safe before this feature is complete. Because the results of using this capability are currently unpredictable this feature has been disabled.

9.5 Expanded Preferences

Currently, the colours that users associate with services only persist while the application is running. It would be much more useful if colours could be stored in the registry or in some other manner so that users' colour selections are persistent across runs of the application. We would also like to allow users to specify default sizes of the views, or to indicate when they have the views aligned in the way in which they would like them to stay.

9.6 Unexpected Traffic

We would like to add the ability for users to set time ranges where substantial amounts of traffic are not expected and to provide a visual or audible notification, as well as log the information, when traffic exceeds the specified threshold during this time. The user would define the threshold amount of traffic at which this notification would occur.

9.7 Time Slider Critical Event Notification

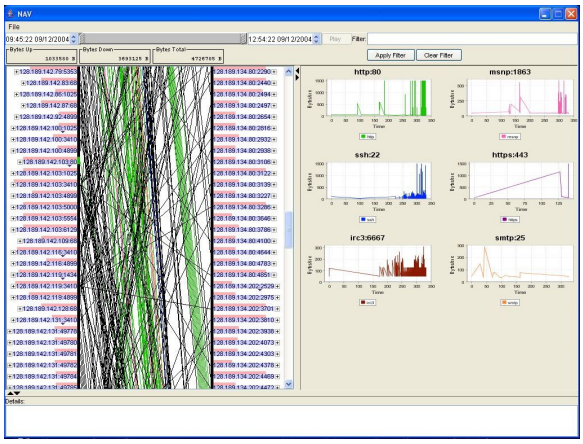
The time slider currently only conveys information about the size of the complete data set and the range of the currently selected subset. With modifications to the slider there is no technical reason it could not be used to encode additional information. For example, it could be integrated with the intrusion detection code in such a way that coloured vertical bars were visible at points along the slider where network security events such as port scans or known exploits have taken place.

10 Conclusion

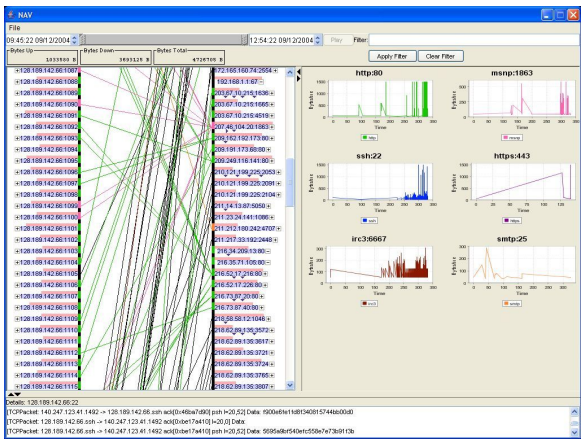
In this paper we have discussed our contribution to providing a high level network traffic visualization. Network traffic analysis remains an important problem due to the steadily increasing volume of network traffic and the inability of traditional network monitoring tools to provide a good overview of traffic patterns. Although these interfaces provide fine grained details, it is very difficult to quickly gain a high level understanding of the nature of the traffic and what services are involved; when standard network sniffers are used on large data sets the output quickly becomes unmanageable. Both home and enterprise users are becoming more interested in network traffic analysis and the current tools do not meet all of their needs. The Network Analysis Visualization (NAV) tool displays packet data from log files with IP address and service overviews accompanied by a detail view. NAV has textual as well as time based filtering and allows users selectable colours for services, as well as the capability to aggregate and remove connections among other features. We feel that we have addressed many of the key aspects of this problem with our solution, and that our visualization is sufficiently extensible to provide a wider range of capabilities in the future.

11 Acknowledgements

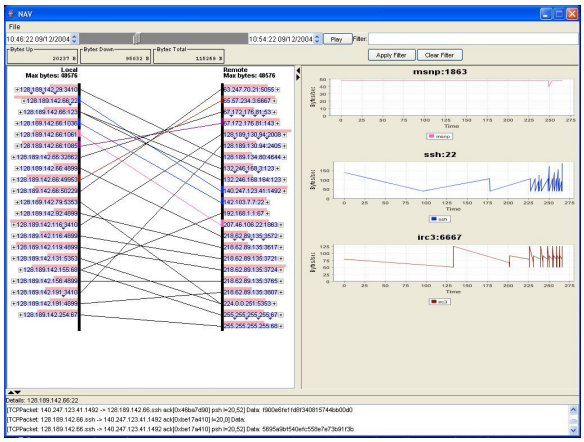
We are grateful to Norm Hutchinson for providing an interview to develop requirements and to Tamara Munzner for a number of helpful suggestions related to this project.



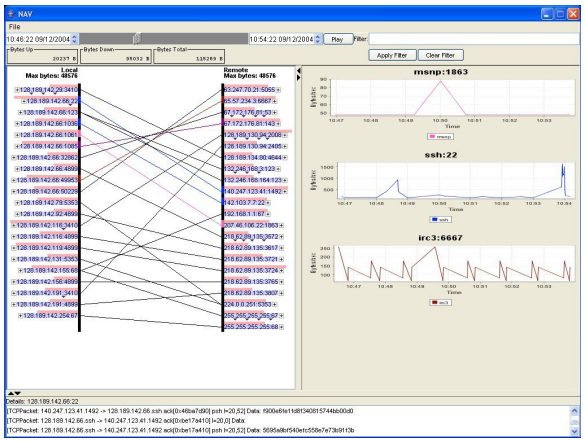
(a) NAV main screen showing the IP wall, and service view in logarithmic scale mode with a large unfiltered data set.



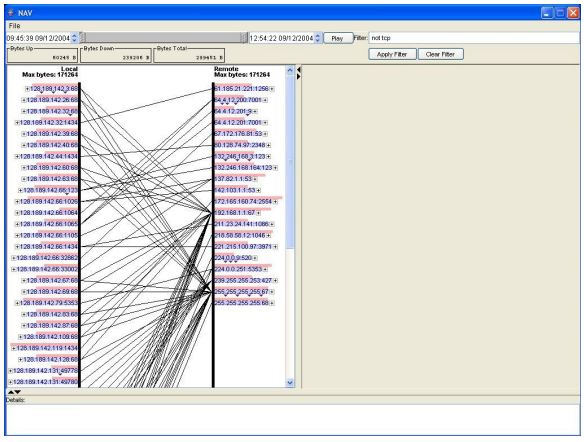
(b) IP Wall View with some elements collapsed and disconnected, service view with a logarithmic data set.



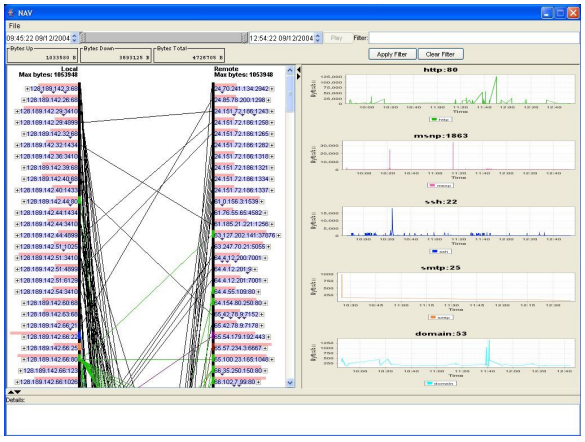
(c) NAV with a large data set and a narrow time range.



(d) Service view changed to linear time scaling.



(e) Filter applied excluding tcp packets.



(f) Added domain name service lookup monitoring, removed irc3 and https.

Figure 11: Screenshots from NAV

REFERENCES

[1] Analysis Console for Intrusion Databases (ACID). downloadable at: http://acidlab.sourceforge.net/, cited December 13, 2004.

[2] Ronald M. Baecker, William Buxton, Jonathan Grudin, and Saul Greenberg. *Readings in Human-Computer Interaction: Toward the Year 2000*. Morgan Kaufmann Publishers, second edition, 1995.

[3] Robert Ball, Glenn A. Fink, and Chris North. Home-centric visualization of network traffic for security administration. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 55–64. ACM Press, 2004.

[4] R.A. Becker, W. S. Cleveland, and M. J. Shyu. The Visual Design and Control of Trellis Display. *Journal of Computational and Statistical Graphics*, 5:123–155, 1996.

[5] April Bandwidth Report: Canadian Broadband Continues Record Growth. http://www.urlwire.com/news/042503.html, cited December 13, 2004.

[6] Gerald Combs. Ethereal. downloadable at: http://www.ethereal.com/, cited December 13, 2004.

[7] Robert Dougherty and Alex Wade. VisCheck. downloadable at: http://www.vischeck.com/vischeck/, cited December 13, 2004.

[8] K. Egevang and P. Francis. RFC 1631 - The IP Network Address Translator (NAT). http://www.faqs.org/rfcs/rfc1631.html, cited December 13, 2004.

[9] Fluke Networks. Fluke. http://www.flukenetworks.com/, cited December 13, 2004.

[10] David Gilbert. jFreeChart. downloadable at: http://www.jfree.org/jfreechart/, cited December 13, 2004.

[11] Jeffrey Heer. Prefuse. downloadable at: http://prefuse.sourceforge.net, cited December 13, 2004.

[12] Van Jacobson, Craig Leres, and Steven McCanne. TCPdump public repository. http://www.tcpdump.org/, cited December 13, 2004.

[13] Jean-Daniel Fekete. InfoVis Toolkit. downloadable at: http://ivtk.sourceforge.net/, cited December 13, 2004.

[14] jPCAP. downloadable at: http://jpcap.sourceforge.net, cited December 13, 2004.

[15] Kiran Lakkaraju, William Yurcik, and Adam J. Lee. NVisionIP: netflow visualizations of system state for security situational awareness. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 65–72. ACM Press, 2004.

[16] Stephen Lau. The Spinning Cube of Potential Doom. *Commun. ACM*, 47(6):25–26, 2004.

[17] Clayton Lewis, Peter G. Polson, Cathleen Wharton, and John Rieman. Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In *CHI '90: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 235–242. ACM Press, 1990.

[18] Jonathan McPherson, Kwan-Liu Ma, Paul Krystosk, Tony Bartoletti, and Marvin Christensen. Portvis: a tool for port-based detection of security events. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 73–81. ACM Press, 2004.

[19] Tamara Munzner, Francois Guimbretiere, Serdar Tasiran, Li Zhang, and Yunhong Zhou. Treejuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. *ACM Trans. Graph.*, 22(3):453–462, 2003.

[20] Network General. Sniffer Pro. http://www.sniffer.com/, cited December 13, 2004.

[21] Ronald A. Rensink. Internal vs. external information in visual perception. In *SMARTGRAPH '02: Proceedings of the 2nd international symposium on Smart graphics*, pages 63–70. ACM Press, 2002.

[22] Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy. http://www.faqs.org/rfcs/rfc1519.html, cited December 13, 2004.

[23] Marty Roesch. SNORT. downloadable at: http://www.snort.org/, cited December 13, 2004.

[24] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *VL '96: Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336. IEEE Computer Society, 1996.

[25] America Internet Usage and 2004 Population Statistics. http://www.internetworldstats.com/stats2.htm, cited December 13, 2004.

[26] Van Jacobson and Craig Leres and Steven McCanne. PCAP. downloadable at: http://www.tcpdump.org/, cited December 13, 2004.

[27] J. van Wijk and A. Nuij. Smooth and efficient zooming and panning.

[28] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, second edition, 2004.

[29] Zone Labs. ZoneAlarm. http://www.zonelabs.com/, cited December 13, 2004.