# Visual Correlation of Host Processes and Network Traffic

Glenn A. Fink[1]
Virginia Polytechnic Institute and State University

Paul Muessig[2]
Virginia Polytechnic Institute and State University

Chris North[3]
Virginia Polytechnic Institute and State University

**ABSTRACT**

Anomalous communication patterns are one of the leading indicators of computer system intrusions according to the system administrators we have interviewed. But a major problem is being able to correlate across the host/network boundary to see how network connections are related to running processes on a host. This paper introduces Portall, a visualization tool that gives system administrators a view of the communicating processes on the monitored machine correlated with the network activity in which the processes participate. Portall is a prototype of part of the Network Eye framework we have introduced in an earlier paper [1]. We discuss the Portall visualization, the supporting infrastructure it requires, and a formative usability study we conducted to obtain administrators' reactions to the tool.

**CR Categories**: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Prototyping, Screen Design, User-Centered Design; K.6.5 [Management of computing and information systems]: Security and Protection—Unauthorized access; General Terms: Design, Experimentation, Human Factors;
**Keywords**: Computer Security, Information Visualization, System Administration

## 1 INTRODUCTION

Over the past two years, we have been analyzing the security tool needs of system administrators [2]. Given the tremendous amount of host and network data that passes through today's systems every minute it is not surprising that visual tools for security awareness were among the top needs expressed by our user community. When we looked into how system administrators detect and investigate potential intrusions and other problems, we found a common process. First, the administrator becomes aware of an anomalous communication pattern. She may find out about the pattern by looking through log files, via an alert from an Intrusion Detection System (IDS), or (quite commonly) from a colleague whose machines are being pummeled by one of hers. Once she traces the communications to an end-point host, the administrator will look at the processes running on it, and make a determination about whether one or more of them are malicious.

The fundamental problem with this approach is that it requires a human to mentally correlate two sets of data: the communication patterns that first indicated the problem, and the set of processes that may be responsible for generating the patterns. Visualizations are natural solutions for helping humans detect patterns in data, and they serve as external memory [3] when correlating data sets.

e-mail: [1]finkga@vt.edu, [2]pmuessig@vt.edu, [3]north@cs.vt.edu

Street address for all authors: Department of Computer Science, 660 McBryde Hall (0106), Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA 24061

Workshop on Visualization for Computer Security
October 26, Minneapolis, MN, USA
0-7803-9477-1/05/$20.00 ©2005 IEEE.

We discuss the problem of simultaneously correlating host process data with network communications patterns, our implementation of a visualization system (called Portall) designed to solve the problem, and a usability evaluation. We conclude with a discussion of which parts of the problem a visualization *can* address, and which parts require extensive infrastructural support.

### 1.1 The Host/Network Divide

Currently, system administrators must integrate data gathered from several tools at multiple locations into a unified mental model of the situation. In our survey of the literature, we found no tool that could correlate both the network traffic and the process activity of the suspect host. This gap between host and network we call the *host/network divide*.

The underlying cause of this divide is in the core of modern operating systems. The Berkeley Software Distribution (BSD) networking stack, the *de facto* standard for almost all modern operating systems, enforces a division between host and network. To talk across a network, a process creates a communication channel (a socket) connecting to the remote destination. But the kernel does not track the processes that own and use the socket. Instead, processes provide a "callback address" for incoming network data to be sent to. Thus, while it may be possible to track processes to the packets they produce, it is impossible to make the reverse association.

In fact, there are two host/network divides that hamper investigation: the technical divide embedded within operating system kernels, and a cognitive divide in the human mind. In the minds of operating-system designers, a cognitive host/network divide based on separation of concerns gave birth to the technical divide. The technical divide now perpetuates a cognitive divide in the minds of system administrators as they attempt to mentally correlate host processes and network traffic.

Our Portall visualization project bridges these divides. Portall's visualization bridges the cognitive divide, by enabling system administrators to visually correlate host and network. The cognitive bridge is supported by an underlying technical bridge at the operating system level that is provided by our Network Eye infrastructure [1].

There are several common system administration scenarios where the bridging capability of Portall can provide critical insight to speed the tasks of understanding and resolving problems on networked hosts. First, users can more easily detect spyware and ad-ware, by correlating the offending network traffic with the responsible processes. If a user's computer is reporting on his activities, Portall can immediately present this information to the user visually, pinpointing the guilty process.

Second, Portall can help detect kernel-level covert communications channels (perhaps installed as part of a rootkit) by comparing communications activity seen on the network with process activity on the suspected host. Portall highlights network activity that cannot be attributed to any process, an excellent indication of compromise.

Third, Portall will also be useful for more accurate penetration testing, fine-tuning of firewalls, and better monitoring of clusters.

Each of these situations requires synthesizing network and process views from multiple sources simultaneously.

Portall lets users see at a glance what processes on the monitored host are communicating and with whom they are communicating. Administrators can use it for quick, accurate situational awareness. They may investigate further with other tools. Portall complements existing tools rather than replacing them. Portall supports three types of administrator tasks:

1. *Ambient*: When the user has no suspicion of malicious activity, she may use it to periodically check the security state of her machines.
2. *Directed*: When the user suspects malicious activity and is seeking to confirm it (*e.g.,* after an IDS alert), she may use Portall to rapidly obtain an accurate mental picture of the overall situation to focus further detailed search efforts using other tools.
3. *Forensic*: When the user has confirmed the malicious activity and is seeking to understand it, she might use the visualization to locate the processes, files, *etc*., responsible for the behavior.

We designed Portall primarily for *directed* search and secondarily to support periodic *ambient* monitoring. Although it could be used for *forensics*, we have not designed Portall as a forensic tool.

## 1.2 Existing Tools

All existing tools known at this writing provide a view of either host activities or network activities. None gives a correlated view of both simultaneously. In this section, we review selected tools from the literature and contrast them with Portall.

### 1.2.1 Host View Tools

Some tools such as the BSD netstat command (commonly available on Windows and Unix-like operating systems) correlate processes and sockets. However, netstat cannot bridge the divide because it does not correlate actual flows on the network with the host's processes. Netstat can poll certain kernel data structures at a user-defined rate, but rapid malware on the network can wreak havoc inside the host within a polling period. Similarly, processes and sockets may be created and destroyed within microseconds, leaving no trace of their existence for netstat.

ZoneAlarm [4] personal firewall for Windows is more powerful than netstat because it links communications to processes allowing the user to control connections. However, ZoneAlarm provides no visualization, nor can it provide remote monitoring of another machine.

Rivet: The Visible Computer [5] provides a visualization of every aspect of one or more computers but it is not practical for computer security awareness because it presents too much information, too slowly.

NVisionCC [6] is a compact dashboard for showing process status on large cluster computers. It focuses on the state of the processes monitored, excluding their communications activities. NVisionCC would need to be integrated with a packet sniffer to do what Portall can.

### 1.2.2 Network View Tools

Numerous visualizations of network data exist, from simple packet-header displays to visual IDS. The typical packet-header visualizations display source and destination IP addresses, source and destination TCP/UDP ports, and protocols using a two- or three-dimensional scatterplot. These include: [7], NVisionIP [8], VisFlowConnect [9], and PortVis [10]. None of these incorporate host-based data.

A visual IDS may incorporate host-based data by graphically plotting alerts from various IDS sensors by location, function, or criticality. Examples of Visual NIDS are the Spinning Cube of Potential Doom [11] and Secure Decision's Secure Scope [12]. However, visual IDS's provide automated diagnoses of potential anomalies, not correlations of process and packet data.

No other known tool correlates network packets to the machine processes that generate it. Some may integrate host information from multiple hosts on a network, and others may present network activity side by side with selected data from host logs, but none actually correlate each packet to a process on the machine that sent or received it. Thus, our approach is unique.

## 2 Implementation

Portall's design was derived to address needs expressed by administrators in our interviews and the conspicuous gaps in the literature we surveyed. We chose to apply information visualization techniques as we designed our displays, particularly Shneiderman's visual information-seeking mantra, "Overview first, zoom and filter, then details-on-demand." [13] Our users want to be able to view the big picture most of the time and drill-down to the packet dump level on demand. By presenting information visually, Portall relies on recognition rather than the slower and more cognitively challenging task of recall. The need for real-time analysis led us to design displays with preattentive features to speed pattern recognition.

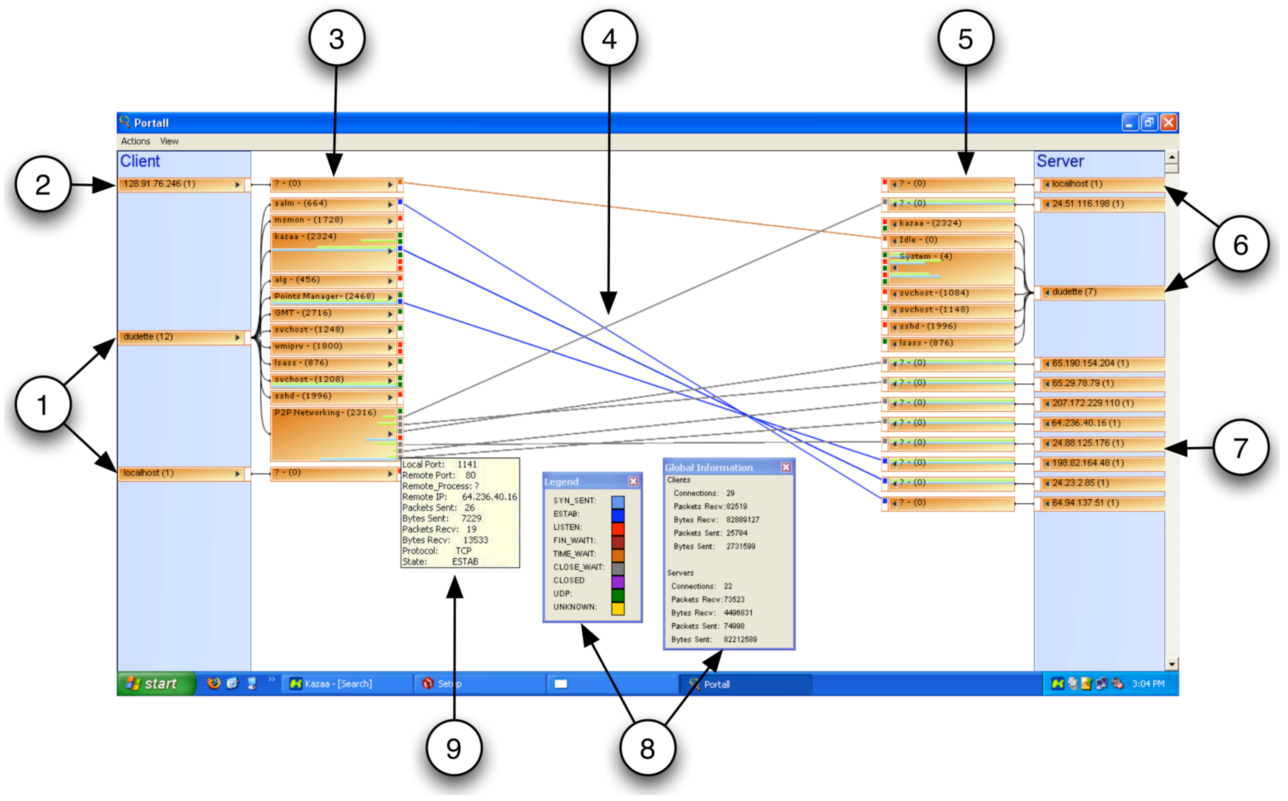
We intend Portall to reveal the overall communication situation in a single screen, leaving the intelligence and analysis to the human analyst. Artificial intelligence in any form is intentionally absent from the tool. We want to enable the user to discover the meaning behind the data rather than creating yet another automated intrusion detection system. Portall will provide a visual overview of host/network activity that is not available elsewhere. There are many fine tools available to our users, and we do not intend to try to persuade them to abandon their favorites.

A typical usage of Portall will be *directed* investigation after an IDS has raised an alert about a potential security incident. A user may conduct initial investigation with Portall to get a rapid and accurate mental model of what is happening on the affected machines. Then he may switch to more specialized tools to investigate in detail. An *ambient* usage would be to bring up Portall a few times a day to check the security state of critical machines. Running Portall's data collection processes continuously may be affect production system performance; thus, we envision users running it only periodically. This section discusses Portall (Figure 1) and its supporting data collection system.

## 2.1 Portall's Displays

Figure 1 shows what Portall looks like on our malware-infected system, Dudette. Close inspection reveals some suspicious client processes (salm, wmiprv, msmon, Points Manager) making external connections. Also, an external client machine (128.91.76.246) is making a connection to a service on Dudette (although Dudette was supposedly not providing any services).

The main window shows a client-server layout of communications from the monitored machines' perspective. On the left are the client machines, processes and ports. On the right are the server machines, processes, and ports. Client hosts are those with processes that initiate communication with other processes (via a TCP SYN packet). Server machines are hosts whose processes accept communications from clients (via a TCP SYN-ACK response). If the same machine has both client and server processes, icons for the host will appear on both sides, with its client processes on the left and its server processes on the right. User interviews led us to believe that this direct modeling of TCP's function would be clear, although our usability study revealed an unexpected reaction.



Local Port: 1141
Remote Port: 80
Remote_Process: ?
Remote IP: 64.236.40.16
Packets Sent: 26
Bytes Sent: 7229
Packets Recv: 19
Bytes Recv: 13533
Protocol: TCP
State: ESTAB

Legend

| | |
|---|---|
| SYN_SENT: | 🟦 |
| ESTAB: | 🟦 |
| LISTEN: | 🟥 |
| FIN_WAIT1: | 🟥 |
| TIME_WAIT: | 🟧 |
| CLOSE_WAIT: | ⬜ |
| CLOSED | 🟪 |
| UDP: | 🟩 |
| UNKNOWN: | 🟨 |

Global Information
Clients
Connections: 29
Packets Recv: 82519
Bytes Recv: 82889127
Packets Sent: 25784
Bytes Sent: 2731599

Servers
Connections: 22
Packets Recv: 73523
Bytes Recv: 4496831
Packets Sent: 74998
Bytes Sent: 82212589

Figure 1. Portall screenshot: (1) The monitored host on the client side, (2) an external client host, (3) client processes, (4) client-to-server connection lines, (5) server processes, (6) monitored host on the server side, (7) external server hosts, (8) information windows, and (9) pop-up connection details.

Overall, we used neutral, complementary colors and subtle shading gradients for the host and process markers. We selected contrasting, primary colors for the port boxes and connection lines because we wanted to draw the user's attention to the communication activity.

In Figure 1, item (1) points to icons for the local host. The upper icon represents the monitored machine, and its label is the host's DNS name. The lower icon is a termination point for traffic that seems to come from the local host but cannot be traced back to any visible process on the machine. Its label is the IP address "0.0.0.0." We consider any connections from this icon highly suspicious. Item (2) points to a remote machine that is a client of a server process on the monitored machine. This may be suspicious if no public services are running.

Item (3) points to a column of client process icons. These icons are connected to the machine where the processes are running by a black Bezier curve. Item (4) points to communication lines that link clients to servers and symbolize a TCP connection. Currently, we make communication lines only for TCP traffic. The line and port box color indicates the TCP connection state. Item (5) shows the list of server processes. On unmonitored hosts, we can give no information about such processes, so we label them "?". We include these icons because they prevent confusion caused by lines that otherwise could cross under the icons of known processes.

Item (6) shows the icons for the monitored host on the server side. Again, one marker has the DNS name of the monitored host for a label, while the other is labeled "0.0.0.0." As on the client side, the latter icon is for connections that are apparently to service ports on the monitored machine but that cannot be correlated to any visible process. Connections to this marker may indicate attack traffic, misdirected traffic, or possibly hidden services running on the monitored machine. Item (7) points to a column of icons for remote machines seen on the network as traffic destinations.

Item (8) shows the color legend and global information floating windows. The legend shows the colors used to indicate the connection's state. Seven colors are used to indicate the TCP state, one color indicates a UDP connection, and a final color indicates other communications. The global information box shows the number of client and server connections, and the packet and byte counts of all the connections monitored in the current session.

Item (9) is an information box that pops up when the mouse hovers over a port. The box tells the connection's local and remote port numbers and IP addresses, the packet and byte counts in both directions (since the communication was established or monitoring began), the protocol, and the connection state.

Not shown above is the packet-dump window that contains a table with packet-header fields from a connection. Right clicking on a port box shows all of the intercepted packets in the packet dump window for low-level analysis. We have also omitted the timeline window that allows users to visualize traffic and processes logged to the database at some point in the past via a horizontal slider bar.

Users can visually highlight communications lines of interest to bring them forward when there are many occluding lines, by simply clicking the communication lines, or their ports, or their process icons.

Figure 2 shows the detailed features of the process icons. Process icons contain a lot of information: item (1) is the process executable name and the process identifier (PID) number, item (2) shows bars indicating the relative activity of each port (green for incoming, blue for outgoing, and grey for overall), item (3) shows the connected communication lines, item (4) shows a port box (this one is a UDP port), and finally, item (5) shows the control that the user can click to show or hide the port list. Hiding the ports vertically shrinks the process icon and is useful if a process has many open ports that are not of concern. The icon's shading indicates whether the port list is shown or hidden.

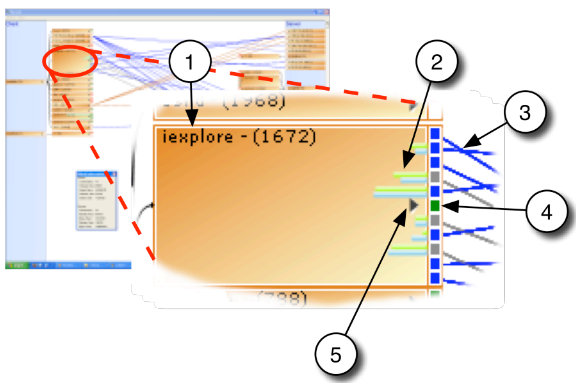


Figure 2. Detail of a process icon: (1) process name and process identifier (PID), (2) port activity bars, (3) communication lines, (4) port box, and (5) expand/reduce control.

## 2.2 Dynamic Behavior

One important challenge for Portall as a visualization of an active host is the need for dynamic layout. Although it would have been useful to arrange the boxes from top to bottom by IP address, PID, or name, doing this caused confusing movement as processes and connections appeared and disappeared. Instead we chose to place new markers where old markers had been removed. Our method may confuse some users and help others. Future work will include finding an optimal ordering algorithm.

We display new screen elements in green for their first display cycle to indicate the appearance of new connections, hosts, and processes. Recently active ports appear briefly as outlines. Newly opened ports appear double-sized for the first cycle.

## 2.3 Architecture

Portall is designed to be a distributed application to visualize data collected from remote sources. Figure 3 shows the distribution of these components on our three-machine demonstration network.

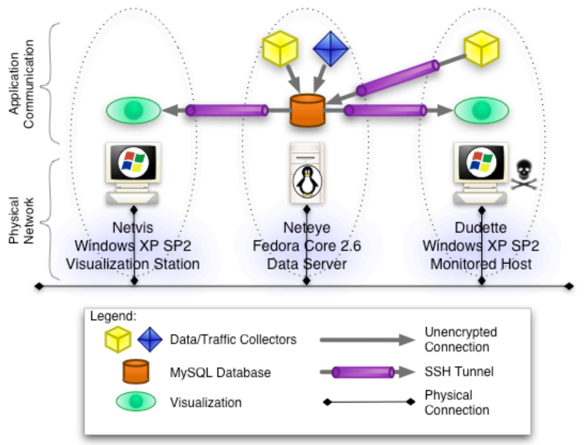


Figure 3. Physical and application layout for the Portall demonstration system.

The three machines on our demonstration network, each have a primary purpose:

1. *Netvis*: This Windows XP machine is our primary visualization platform. We are reasonably certain that it is free from malware. Portall runs on this machine and it may be used to visualize activities of any of the three machines.
2. *Neteye*: This machine is a Red Hat Linux (Fedora Core 3) machine running a MySQL server a traffic collector, and a process data collector. Neteye's primary job is to provide data for the visualization platforms.
3. *Dudette*: This host is running Windows XP and a large number of other malicious and questionable processes. Dudette's primary purpose is to serve as a monitored host although technically, it could visualize data from any of the machines.

We scanned Dudette with a variety of antivirus and spyware/ad-ware detectors and found that it has at least the following known malicious processes running on it: W32.Spybot.Worm, wpma36c.exe (unknown malware), W32/Rbot-WM worm, a variant of the GEMA.D trojan, and several executables classified by Norton Antivirus as a miscellaneous Backdoor.Trojan. NoAdware v3.0 found 180 files contributing to the following ad-ware installations: 180Search Assistant, AutoUpdater/Envolo, ISTbar/Powerscan, PeopleOnPage, AvenueMedia (DyFuCA), CashBack, NCase, Adware.SyncroAd.

Dudette is obviously a very distressed machine. In fact, the CPU is typically at 100% utilization, and frequently one cannot make any legitimate outgoing connections from it because it is so "busy." Unfortunately, problems with the tremendous amount of malware running on Dudette prevented us from reliably connecting to the database on Neteye. Thus, in the usability study we did not attempt to show remote visualization.

Currently, on Windows, the host and network monitor pieces are together, but this is not required. Soon, we plan to split these functions up according to our fully decentralized design. The Windows host monitor gathers process and port information via the IP Helper Application Programming Interface (IPHLPAPI). This interface retrieves all open TCP and UDP ports via a pair of undocumented Windows system calls. These two calls return statistics about each open port and what PID opened the port. The host monitor writes timestamped entries containing process and port information to the database.

The Windows network monitor collects and stores packet header data and timestamps to the database. We collect packets via the WinPCap packet capture library [14]. The network monitor also formats this data for display in the visualization's packet dump window.

From the host information stored in the database, we create two trees of hash tables that hold information about each monitored machine, process, and connection. One tree contains information on client hosts, and the other is for server machines. We populate the connection endpoints from packet-header information in the database. If the packet-header information shows connections that were not found among the known processes, they are assigned to the "0.0.0.0" host under the "?" process node.

Although the Windows data collection method is extensive, we must still rely on polling. Collecting packet information from the network gives an advantage over netstat, but we cannot correlate this information in real time. On Linux, the kernel is open source, and we have created a loadable kernel module and a modified kernel to collect and correlate the host data without missing any packets or processes. We plan to detail our implementation in a future paper [15].

Portall's data collection components can log information to a MySQL or Access database for later analysis and visualization. We connect to the remote database over an encrypted Secure Shell (SSH) tunnel that must be set up manually in advance. Portall can visualize current events without logging to a database, but without the database, we cannot review recent events.

## 2.4 Scalability

One of Portall's key design goals is to present a single-screen overview of the communications and processing activities of one or more networked hosts. Thus we must consider how well the visualization approach and architecture will scale as numbers of machines, connections, and processes increase. Currently, Portall does no aggregation, so there is a marker for each machine and process, and a line for each connection. Given a typical 1280x1024 pixel monitor there is vertical space for just 39 processes and hosts on each side of the display. A single process with multiple open connections (such as the Kazaa example shown in Figure 4) can easily use up this space. Follow-on prototypes could use automatic level-of-detail and aggregation to reduce the amount of space required.

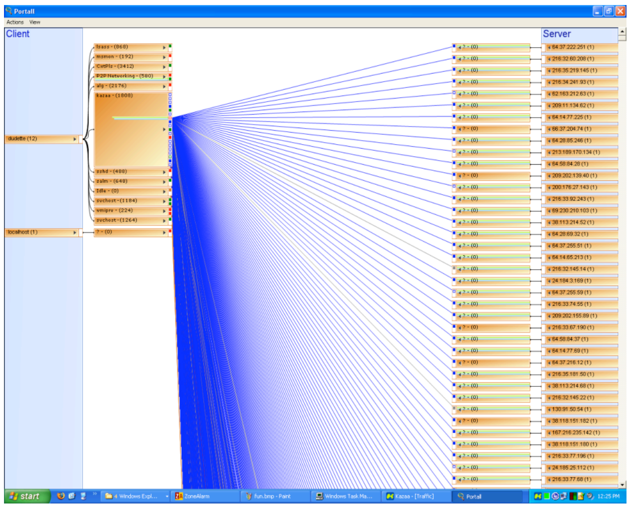


Figure 4: Remote connections from a single Kazaa process can exhaust Portall's display area. This shows the need for aggregation strategies.

In Figure 4, all the server markers could be automatically grouped into Classless Inter-Domain Routing (CIDR) blocks as is done in AutoFocus [16]. Such aggregations should only be done for unmonitored hosts where the process names are unknown. Focus+context techniques could reduce the size of nonfocal markers to mere lines a few pixels high.

Large numbers of connection lines may also be aggregated into a single line in some cases. Some of our users operate web servers where thousands of connections per minute are normal. It is possible to automatically aggregate connections to such servers from unmonitored hosts. However, an unusually large number of active connections, as Figure 4 shows, is a strong indication of a potential security breach. We believe it is best not to automatically aggregate connection lines until the user indicates that the activity is normal. Large numbers of crossing lines can be a source of confusion as well. Portall does not currently attempt to reduce the number of crossings, although an heuristic such as the iterated barycenter method [17] may prove effective in a future version.

Currently, Portall can visualize data from a single host and network connection, but the goal is to monitor multiple machines and potentially multiple networks from a single display (see Figure 5). Our decentralized deployment design calls for an overlay network of SSH connections linking monitored hosts (that serve internal host data) and network monitors (taps, etc. that serve connection data) to an aggregation engine on the local network. These aggregation engines will in turn serve data to local and remote visualization stations for monitoring. SSH will handle the authentication, confidentiality, and data integrity issues for the system. Portall is currently capable of limited decentralized deployment, but its displays cannot handle the capacity required for anything greater than a few machines.

We will need to investigate the impact of the SSH connections between monitored hosts, aggregators, and visualization stations in a decentralized deployment. Beyond the basic scalability issues, it is possible that the communications that support the overlay network may confound a user's view of the security states of the monitored machines. More prototypes and further research is required to investigate these scalability and decentralization issues.

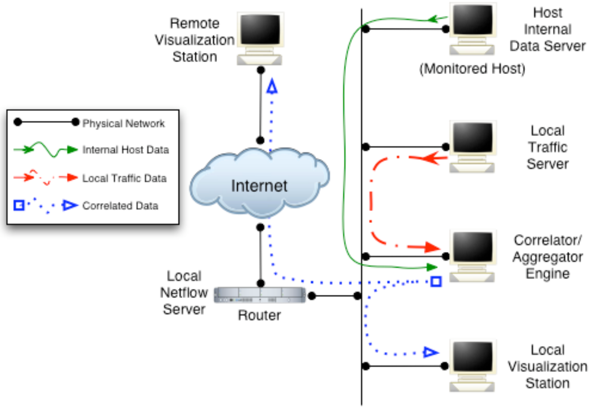


Figure 5: Eventually, Portall, as part of the Network Eye framework, will be able to display the activities of multiple hosts and network connections remotely via a hierarchy of data servers, aggregation engines, and visualization stations.

## 3 Usability Study

This section is a report of the usability study that we performed on Portall with a small group of experienced users. Our main research questions are:

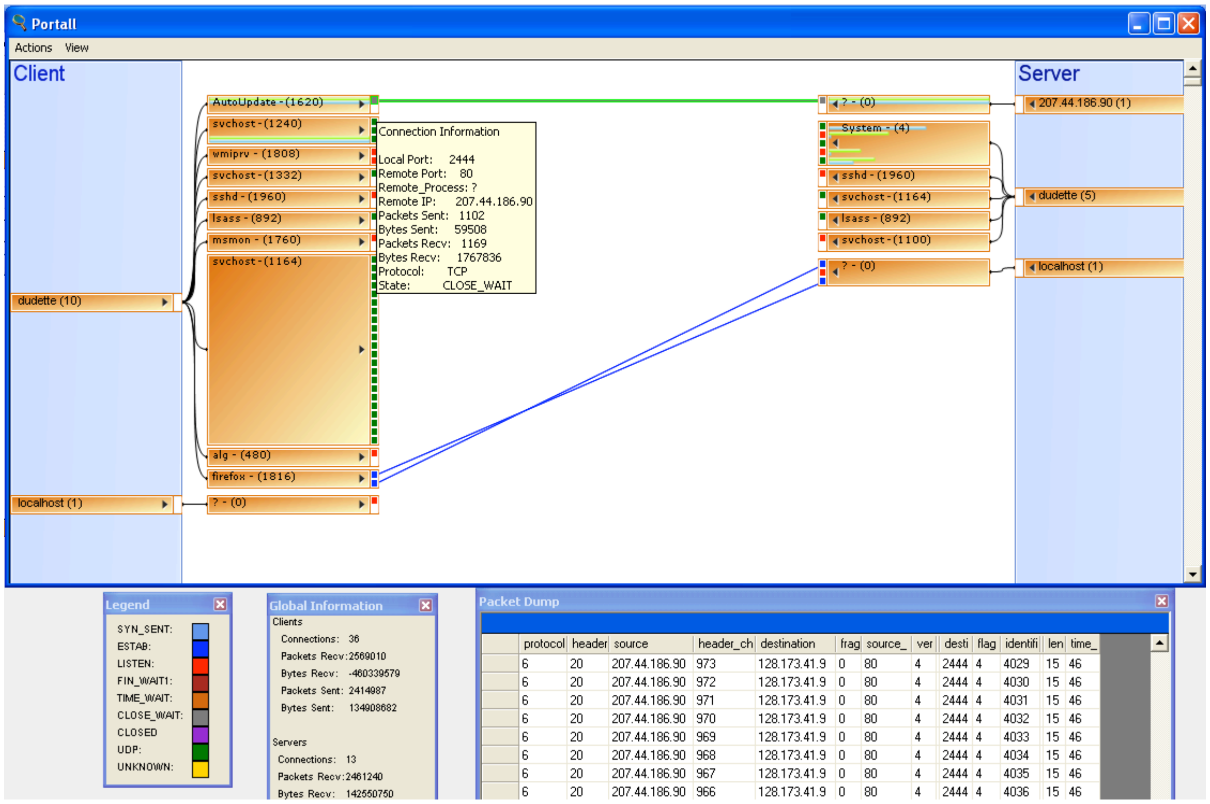
1. Does correlating network traffic with host process data provide useful insight to system administrators for investigating anomalies?
2. Is our visualization capable of presenting this correlation data in an understandable and effective way? In particular, we wanted to learn what kinds of filtering system administrators needed.

To answer these questions, we conducted a 45-minute usability evaluation of Portall with five subjects visualizing live data under realistic circumstances. We consider this a formative evaluation because we wanted detailed feedback to improve Portall's design, and we did not yet have sufficient information on the kinds of filtering tools needed for an actual investigation.

### 3.1 Study Subjects and Methods

We chose five system administrators known to be experienced in computer security to test our software and provide recommendations. We started with a venire of 17 administrators whom we selected because of experience we knew they had or because their posts to a campus-wide system administrator e-mail list indicated such experience. Five subjects volunteered from this group. Three of our subjects were from the group of system administrators who expressed tool needs in an earlier requirements-analysis study [2]. Two were new to our research. The study subjects were offered no compensation. All the subjects had worked as professional administrators for between 8 and 32 years (mean 18.2, stdev 9.8). We scored the subjects' expertise using a weighted average of five metrics (each on a five-point scale). We asked subjects to rate themselves on their expertise at Windows, netstat, and tcpdump use, and on their ability to detect intrusions. We asked how often they used security awareness tools to investigate an anomaly as an indicator of how often they might need to use Portall. The lowest frequency was 1-2 times per month and the highest was 4-5 times per day. From the composite expertise scores, we consider three subjects to be experts, and two to have moderate expertise.

Our usability study consisted of 28 questions, starting with a structured biographical information section, and concluding with a semi-structured evaluation. For our study, we prepared two equivalent workstations with Microsoft Windows XP service pack 2. Neither machine was running any public services. The first machine, Netvis, was loaded just for the study and was clean of any malware to the best of our knowledge. We obtained the second machine, Dudette, from a careless user and found it to be rife with malicious software of all sorts. After allowing the users to become familiar with the visualization on Neteye, we showed them the same visualization on Dudette. The interviewees were not told up-front that Dudette was hacked. We wanted to see how much they would notice by themselves first. If they gave up after a reasonable amount of time, we revealed the evidences of problems that Portall was showing about the machine (see Figure 6) and discussed them with the subjects. Thus, we planned that even users who were unfamiliar with Windows would be able to have a baseline of normality to compare with.



Connection Information

Local Port: 2444
Remote Port: 80
Remote_Process: ?
Remote IP: 207.44.186.90
Packets Sent: 1102
Bytes Sent: 59508
Packets Recv: 1169
Bytes Recv: 1767836
Protocol: TCP
State: CLOSE_WAIT

| | |
|---|---|
| SYN_SENT: | 🟦 |
| ESTAB: | 🟦 |
| LISTEN: | 🟥 |
| FIN_WAIT1: | 🟥 |
| TIME_WAIT: | 🟧 |
| CLOSE_WAIT: | ⬜ |
| CLOSED | 🟪 |
| UDP: | 🟩 |
| UNKNOWN: | 🟨 |

Clients
Connections: 36
Packets Recv: 2569010
Bytes Recv: -460339579
Packets Sent: 2414987
Bytes Sent: 134908682

Servers
Connections: 13
Packets Recv: 2461240
Bytes Recv: 142550750

| protocol | header | source | header_ch | destination | frag | source_ | ver | desti | flag | identifi | len | time_ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 20 | 207.44.186.90 | 973 | 128.173.41.9 | 0 | 80 | 4 | 2444 | 4 | 4029 | 15 | 46 |
| 6 | 20 | 207.44.186.90 | 972 | 128.173.41.9 | 0 | 80 | 4 | 2444 | 4 | 4030 | 15 | 46 |
| 6 | 20 | 207.44.186.90 | 971 | 128.173.41.9 | 0 | 80 | 4 | 2444 | 4 | 4031 | 15 | 46 |
| 6 | 20 | 207.44.186.90 | 970 | 128.173.41.9 | 0 | 80 | 4 | 2444 | 4 | 4032 | 15 | 46 |
| 6 | 20 | 207.44.186.90 | 969 | 128.173.41.9 | 0 | 80 | 4 | 2444 | 4 | 4033 | 15 | 46 |
| 6 | 20 | 207.44.186.90 | 968 | 128.173.41.9 | 0 | 80 | 4 | 2444 | 4 | 4034 | 15 | 46 |
| 6 | 20 | 207.44.186.90 | 967 | 128.173.41.9 | 0 | 80 | 4 | 2444 | 4 | 4035 | 15 | 46 |
| 6 | 20 | 207.44.186.90 | 966 | 128.173.41.9 | 0 | 80 | 4 | 2444 | 4 | 4036 | 15 | 46 |

Figure 6: The uppermost connection shown in this screen shot (a grey line highlighted in green) shows a half-open connection. Data is passing over the connection as shown in the Packet Dump window (bottom right) although it appears to be closed from the client's side. We believe this is a covert channel (innocuously named "AutoUpdate"). This is somewhat subtle, and users did not notice it independently.

After showing the subjects the visualization on both machines, we asked them questions in four areas: (1) initial observations and impressions of the visualization, (2) preferences for how this visualization should operate, (3) reflection on how the visualization might be improved, and (4) interest in the visualization (how they might use it in their work, and whether they would be willing to evaluate follow-on prototypes).

## 3.2 Discussion and Analysis

In this section, we discuss what we learned from the issues the subjects reported and present our key conclusions that will contribute to the future work. From the evaluation, we gathered 69 issues: 19 positive comments, 22 negative comments, and 28 enhancement requests. We have summarized the major issues in Table 1 (positive), Table 2 (negative), and Table 3 (enhancements). We classified each issue as one of the following:

- *Fundamental (F):* An essential issue with the visualization design (14 positive, 7 negative, 16 enhancements).
- *Incidental (I):* A prototype implementation issue not fundamental to the visualization design (1 positive, 11 negative, 8 enhancements).
- *Experiential (X):* An issue that was dependent on the user's experience or knowledge (*e.g.*, familiarity with Windows, etc.) (4 positive 2 negative, 0 enhancements).
- *Aesthetic (A):* Issues of personal taste (0 positive, 2 negative, 4 enhancements).

The Expert group tended to report more *Essential* negative features while the Moderate group reported more negative features that were due to their experience level. For enhancements and positive features, the expert group consistently generated more findings than the moderate group in every category except *Experiential*.

Table 1. Major positive issues.

| Issue type, # of users | Issue Reported or Comment Made |
|---|---|
| F, 5 of 5 | Users said that Portall's correlation of traffic and process data provides useful insight. |
| F, 5 of 5 | To get similar insight users said would require mental integration of information from several other tools. |
| X, 5 of 5 | Users would use Portall for ambient monitoring of machines or networks. |
| A, 4 of 5 | Made an unsolicited comment that the visualization was visually pleasing or preferable to textual tools. |
| F, 4 of 5 | Users quickly noticed differences between secured and hacked machines. |
| X, 2 of 5 | Users would use Portall for directed investigation. |
| F, 2 of 5 | Portall would speed up work practices. |

Table 2. Major negative issues.

| Issue type, # of users | Issue Reported or Comment Made |
|---|---|
| F, 4 of 5 | Connection lines were confusing (too many or too difficult to trace); the display seemed cluttered. |
| F, 4 of 5 | Marker for monitored host appearing on both client and server sides was confusing. |
| X, 4 of 5 | Unfamiliar with Windows process names. |
| F, 3 of 5 | Portall probably could not handle activity level of actual deployment. |
| I, 1 of 5 | Need an activity baseline to be absolutely certain that any activity was suspicious. |

Table 3. Key enhancements requested.

| Issue type, # of users | Issue Reported or Comment Made |
|---|---|
| F, 4 of 5 | Portall should let users filter out safe (or known) processes, IP addresses, protocols, and ports. |
| F, 3 of 5 | Provide an indication of CPU usage by process. |
| F, 3 of 5 | Let the user filter, minimize, or aggregate elements he has already examined. |
| F, 2 of 5 | Allow users to move machine and process markers. |
| F, 2 of 5 | Physically separate the markers for monitored machines from the unmonitored machines. |
| F, 1 of 5 | Let the user set notification thresholds on packets/bytes/connections, etc. |

### 3.2.1 Portall's Key Strengths

All of our subjects were enthusiastic about our visualization in spite of its current limitations. All said they would like to install a copy on their own machines, and all wanted to be included in future studies with more advanced prototypes. We found out later that several of the subjects had a reputation as "straight shooters" who would not have hesitated to "shred" our concept if they had not thought it was good. We were encouraged to hear that highly experienced users consider Portall a step in the right direction, and to know that they were not just being gracious when they made positive comments.

All of the subjects said that Portall's correlation of network traffic and process activity provides useful insight for their jobs. All agreed that the only alternative way they knew of to get this insight would be to mentally integrate the output of several other tools. Two users cited ways that Portall would speed up current work practices. One user said, "The only such 'tool' I would have [to do this correlation without Portall] would be extremely klunky, time-intensive attempts to do something similar manually."

All the users said they would use the visualization for ambient monitoring, and only two said they would use the program for directed investigation. This was unexpected since we designed Portall specifically for directed investigation. However, our study may have been biased toward an ambient environment. The users might have responded differently if we first gave them reason to be suspicious via an IDS alert. Since the administrators we interviewed typically reviewed their logs once or twice daily, perhaps they expect to use Portall in the same ambient manner. Since most of the subjects only noticed suspicious events on their jobs once a week or less, we believe they are saying that they would use Portall more often than we had expected them to. Thus, we must minimize the amount of computational resources required by Portall so that it can be used the way administrators desire.

Four of the five users quickly noticed important differences between the hacked and clean machines. Subjects noted large numbers of remote connections being made and dropped without any user activity, suspicious process names, and a half-open covert channel to a keystroke-logging server process on the local machine. However, only two of them would commit to saying they positively saw something "suspicious" without an accepted baseline. Although we first showed the subjects the clean machine, one of the expert subjects did not accept that machine's behavior as a baseline for the other since he did not know the administration history of either host. We think a study with the participants using the visualization in their own environments will help determine the true effectiveness of Portall.

Finally, four of the five users made unsolicited comments about how they preferred the visual approach to text-based tools or how pleasing they found the visualization. (e.g., "I like the visual approach. [Portall] is much more visual than other tools.") One user enthusiastically said that if he had Portall in his office, he "would be watching it all day long!" We believe this is another indication that we are on the right track to give users what they need.

### 3.2.2 Portall's Key Weaknesses

Probably the most important conclusion we may draw from the negative issues is that our subjects, regardless of expertise, seem to prefer the visualization to represent machines as concrete physical locations rather than by their role in TCP communications. Our target user population can be expected to understand the workings of TCP. Thus, we elected to base the visualization on TCP's client-server model. However, users did not understand why separate markers for Dudette would appear on both the client and server sides (even though they clearly understood that the machine was running both client and server programs). Even our most expert user who designs certification programs for a world-class computer security education organization had to ask three times whether the separate markers for the monitored machine on the client and server sides were for the same machine. We conclude that our users expect to see each machine's marker appear in only one place.

Similarly, subjects did not readily grasp the meaning of the mapping we used to color connections and ports according to their

TCP connection states. Although the users all understood TCP, they were not necessarily familiar with each state the protocol's connections can achieve. These results cause us to question whether a direct visual representation of TCP's abstract model is apt for our problem-space.

Another problem users had with Portall was that they expected it to be a notification system rather than an analysis tool. Some users wanted the color red to be used solely to draw attention to potential dangers. Others wanted Portall to send e-mail or call pagers when certain threshold quantities were reached. One user suggested keeping a list of "known trojans" (malware) and highlight these programs if they appeared. Of course, this direction is contrary to Portall's purpose—to enable *humans* to effectively locate potential security compromises. There are plenty of fine notification systems in existence, and we do not wish to copy them. However, these comments indicate the potential for integrating notification-system data into Portall to expand its use for combined ambient monitoring and directed analysis.

Several of the users wanted to have more control over the layout of the icons. They suggested interactions such as manually dragging the icons and automatically organizing the host icons into several groups by the degree of confidence the user has that the hosts are benign and well-managed. All these suggestions indicate that users would like to be able to impose some ordering on the markers to help them make sense of the visual scene.

Three of the five subjects believed that Portall was not space-efficient enough to handle the activity levels they typically see at their jobs. One subject said that the one-Hertz data refresh rate was too slow and would miss important traffic in production use. For the first issue, we plan to experiment with filtering and aggregation solutions proposed by the users in follow-on prototypes. Ultimately, kernel modifications are needed to solve refresh rate problem.

### 3.2.3 Key Enhancements Requested

A frequently asked question was, "which one of these icons represents this machine (the one being monitored)?" The users suggested various layout, coloring, etc. schemes to distinguish monitored machines from the others. They also provided lots of ideas about filtering capabilities and additional indicators that they would need to perform their jobs. Users wanted to be able to minimize, hide, or aggregate known safe machines, communications, ports, and processes so that the picture only showed risky behaviors. They wanted to see indications of bandwidth, CPU utilization, and file-usage by each monitored process. One user stated that, "It's not necessarily the number of connections, but the amount of traffic going through the connections that I would look at, personally."

Some requests seemed to be leading toward making Portall an "intelligent" IDS tool, something it was not intended to be. However, a few enhancement requests that bordered on providing artificial intelligence could become part of the visualization at some point. These involved allowing the user to specify threshold values for bandwidth, CPU usage, and other quantities per machine, process, connection, or port. If a threshold amount was exceeded, the visualization would change the color of the marker in the display.

## 4 Future Work and Contributions

Portall is a prototype of Network Eye's networked host view [1]. Network Eye will lay the groundwork for end-to-end visualization by allowing remote monitoring of multiple machines in a single display. End-to-end visualization is the ability to view communications between distributed processes across the network. As for Portall, we plan to make many improvements suggested by our users, including a detailed view of individual processes showing what resources (files, devices, etc.) they have open. We are working on a new version that will be much more concrete and less of a direct visualization of client-server connections. We plan to use Portall and follow-on prototypes to visualize what happens when an attacker compromises a honeynet.

Another use for Portall that our users suggested was that it may be a good tool to teach security monitoring to novice administrators who rely on the GUI. Conversely, experts may wish to retain the visualization but add a command-line interface to control it. Another area of future work involves investigating the scalability of our displays and of the underlying communications when multiple hosts and network connections are monitored simultaneously. Finally, we plan to conduct a situated study with experienced administrators using Network Eye in their real work environments to find out how well the framework handles real-world use.

Our study shows that visually fusing host and network data is useful for investigating security-related anomalies. But the real host/network divide lies within the kernel of every major operating system today. Our continued work on Network Eye will include bridging this technical divide with proof-of-concept implementations [15] and bridging the cognitive divide with compelling visualizations that free users from the restrictions imposed by today's tools and mindsets.

Our study makes the following contributions:

- We have identified two host/network divides and shown how they hamper investigation.
- We have designed, implemented, and tested a visualization that bridges the divides using the novel approach of correlating process and packet data.
- We have demonstrated Portall's practical value by showing how users can detect malicious behavior resulting from real malware infections.
- We have demonstrated that Portall is useful for both ambient security monitoring and for directed investigation of anomalies.
- We collected detailed information about the kinds of filtering and aggregation users need to perform security monitoring and investigation tasks.

We expect to make further improvements in both our visualization design and our understanding of our users' needs as we continue this study. Armed with visualizations that can simultaneously correlate host and network data into an integrated picture, we hope system administrators will be better prepared to defend their networks and machines.

## 5 ACKNOWLEDGMENTS

The authors would like to thank the system administrators at Virginia Tech and elsewhere who reviewed our prototypes and gave helpful suggestions. This research was supported in part by a National Science Foundation Integrated Graduate Education and Research Training (IGERT) grant (award DGE-9987586).

## References

[1] Robert G. Ball, Glenn A. Fink, Anand Rathi, Sumit Shah, and Chris North, "Home-Centric Visualization of Network Traffic for Security Administration," *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC 2004)*, ACM Press, New York, NY, USA, pp. 55-64, October, 2004.

[2] Glenn A. Fink, Ricardo Correa, and Chris North, "System Administrators and Their Security Awareness Tools." submitted for publication, 2005.

[3] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman, "Information Visualization," *Readings in information visualization: Using vision to think*, Card, S.K., Mackinlay, J.D. and Shneiderman, B.

eds., Morgan Kaufmann Publishers, San Francisco, Calif., pp.1-34, 1999.

[4] ZoneAlarm from ZoneLabs, http://www.zonelabs.com/, August, 2005

[5] Robert Bosch, et al., "Rivet: The Visible Computer," http://graphics.stanford.edu/projects/rivet/, August, 2005

[6] William Yurcik, Xin Meng, and Nadir Kiyanclar, "NVisionCC: a visualization framework for high performance cluster security," Proceedings of VizSEC 2004, ACM Press, New York, NY, USA, pp. 133-137, October, 2004.

[7] Hyogon Kim, Inhye Kang, and Saewoong Bahk, "Real-time visualization of network attacks on high-speed links," IEEE Network, IEEE Press, Washington, DC, pp. 30-39, 2004.

[8] Kiran Lakkaraju, William Yurcik, and Adam J. Lee, "NVisionIP: netflow visualizations of system state for security situational awareness." Proceedings of VizSEC 2004, ACM Press, New York, NY, USA, pp. 65-72, October, 2004.

[9] Xiaoxin Yin, William Yurcik, Michael Treaster, Yifan Li, and Kiran Lakkaraju, "VisFlowConnect: netflow visualizations of link relationships for security situational awareness," Proceedings of VizSEC 2004, ACM Press, New York, NY, USA, pp. 26-34.

[10] Jonathan McPherson, Kwan-Liu Ma, Paul Krystosek, Tony Bartoletti, and Marvin Christensen, "PortVis: A Tool for Port-Based Detection of Security Events Proceedings of VizSEC 2004, ACM Press, New York, NY, USA, pp. 73-81.

[11] Stephen Lau, "The Spinning Cube of Potential Doom," Communications of the ACM, 47 (6), ACM Press, New York, NY, USA, pp. 25-26, October, 2004.

[12] Secure Decisions, http://www.SecureDecisions.com, August, 2005.

[13] Ben Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations, CS-TR-3665, Department of Computer Science, University of Maryland, Human-Computer Interaction Laboratory, and Institute for Systems Research, 1996.

[14] WinPCap: The Windows Packet Capture Library. http://www.winpcap.org/, August, 2005.

[15] Glenn A. Fink, Vedavyas Duggirala, and Chris North, "The Host/Network Divide." submitted for publication, 2005.

[16] Cristian Estan, Stephen Savage, and George Varghese, "Automatically inferring patterns of resource consumption in network traffic," Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, ACM Press, New York, NY, USA, pp. 137-148, 2003.

[17] Kozo Sugiyama, S. Tagawa, and M. Toda, "Methods for Visual Understanding of Hierarchical System Structures", IEEE Transactions on Systems, Man, and Cybernetics, SMC-11 (1981), IEEE Press, Washington, DC, pp. 109-125, 1981.