An Architecture for Discovering and Visualizing Characteristics of Large Internets

Michael F. Schwartz
David H. Goldstein
Richard K. Neves
David C. M. Wood

CU-CS-520-91

An Architecture for Discovering and Visualizing Characteristics of Large Internets[1]

Michael F. Schwartz
David H. Goldstein
Richard K. Neves
David C. M. Wood

CU-CS-520-91 February 1991

Department of Computer Science
University of Colorado
Boulder, Colorado 80309-0430
(303) 492-7514
electronic mail contact: schwartz@latour.colorado.edu

Abstract

In this paper we present an architecture for discovering characteristics of large internets, such as topology, congestion, routing, and protocol usage. Our approach uses a very loosely coupled architecture that does not require global agreement on a particular network management standard, such as the Simple Network Management Protocol. Instead, we use a number of different network protocols and information sources to derive information about networks, cross-correlating this information when necessary to determine important characteristics or to uncover inconsistent information. This approach recognizes that different sources of network information have different characteristics with respect to timeliness of discovered information, expense, danger of generating network problems, and completeness of discovered information. Our architecture gives the network administrator control over which discovery protocols are used, and how frequently each is scheduled. Moreover, the architecture focuses on supporting network management in large scale internets, such as the global TCP/IP Internet. We have built a prototype implementation that can collect network information using a few network protocols, and display this information graphically.

1. Introduction

For the past three years, the Networked Resource Discovery Project at the University of Colorado, Boulder has explored a number of experimental means by which users can discover the existence of resources in a large internet environment [Schwartz 1991]. Example resources include network services, documents, retail products, current events, and people. The project focuses particularly on very large, administratively decentralized environments, spanning national or international networks. This general problem has taken a number of specific forms, including Internet "white pages" [Schwartz & Tsirigotis 1991], support for distributed collaboration [Schwartz & Wood 1990], support for probabilistic "yellow pages" management [Schwartz 1989, Schwartz 1990], and support for mapping and discovering public resources reachable via the Internet [Schwartz et al. 1991]. A common theme of these projects is the ability to accommodate heterogeneity of upper level network protocols, information formats, and organizational structures. While standards are helpful in this regard, it is difficult to specify standards that are both globally adopted and technologically current.

In the current paper, we present an architecture that applies resource discovery techniques to the problem of network management, and a prototype implementation of this architecture. The architecture supports a visual interface to data discovered about networks, such as topology, congestion, routing, and protocol usage. Three characteristics differentiate our work from related projects, such as Digital Equipment Corporation's DECmcc system [Malamud 1991], Sun's SunNet manager [Sun Microsystems 1990], Silicon Graphics' NetVisualyzer [Silicon Graphics Computer Systems 1990], Reissig's network management system [Reissig 1990], NASA's Lisa IV system [Kislitzin 1990], and various other tools [Stine 1990]. First, our approach uses an unusually loosely coupled architecture, which does not require global agreement over a particular network management standard (such as the Simple Network Management Protocol (SNMP) [Case et al. 1989]). Instead, we use a number of different network protocols and information sources to derive information about networks, cross-correlating this information when necessary to determine important characteristics (such as the location of gateways), or to uncover inconsistent information (such as different subnet masks on individual nodes within a particular IP subnet). Second, our approach provides mechanisms to allow passive monitoring and active probing to be scheduled in different ways in various segments of an internet, and for information to be cached at various points around the network. These considerations enhance scalability, for use in large internets. Third, our architecture supports multiple graphical representations of network information, in recognition of the fact that different types of information and different scales of network can best be represented using different graphical representations.

An explicit component of our approach is the recognition that different sources of network information have different characteristics with respect to timeliness of discovered information, discovery expense, danger of generating network problems (such as broadcast storms), and completeness of discovered information. In contrast, network management systems based on a single standard are limited to the characteristics provided by that standard. Usually the standard focuses on a particular perspective of network management, which therefore limits its scope. SNMP, for instance, takes the perspective that the network is essentially a collection of devices that can be instrumented and measured, to determine packet flow rates, routing table contents, etc. A different set of characteristics may be detected if passive packet monitoring servers are installed in the network (which would allow broadcast storms to be detected without imposing added processing load on a gateway, for example), or if the network management system supports directed probes into a network (which would allow networks to be examined even if no monitoring server were installed; this might be useful for network management when it is infeasible to install monitors on all network segments). Our architecture supports all of these perspectives.

As with our earlier Internet white pages work [Schwartz & Tsirigotis 1991], the current architecture makes use of multiple protocols and information sources. In contrast, however, the current architecture targets an environment that provides significantly more information sources that can support discovery. In the case of the white pages tool we used four sources, while the current architecture may use dozens of sources, particularly in multi-protocol networks. While the current prototype only uses a few of these sources, our longer range goal is to incorporate many more of the available sources. Essentially, our high level goal is to explore the extent to which one can integrate a heterogeneous, administratively decentralized network by viewing it as an instance of a resource discovery problem, using the techniques we have developed in other resource discovery contexts to support network management.

The remainder of this paper is organized as follows. In Section 2 we introduce the network visualization architecture. In Section 3 we overview the set of discovery protocols and information sources that can be used by our system, indicating the implications of each. In Section 4 we discuss a prototype implementation of the important parts of the architecture. Finally, in Section 5 we offer a summary, and discuss continuing efforts and future work.

2. Architecture

In this section we discuss the general architecture to support scalable network discovery and visualization. The full architecture involves a number of elements that have not yet been implemented. In Section 4 we will discuss our prototype implementation.

2.1. Overview

The macroscopic architecture for supporting internet discovery and visualization is illustrated in Figure 1. The architecture uses a collection of servers distributed around portions of the internet being instrumented. Each server periodically executes a set of discovery protocols to maintain information about certain segments of the network. Typically, a server will reside on a particular segment of a local internet, and passively gather information about that segment (monitoring routing table update traffic on a broadcast LAN, for example). In the figure, passive monitoring is indicated by thin arcs with arrowheads directed into the discovery server. However, the architecture also allows active probes, where queries are sent to remote LANs to collect information. This capability can be used by discovery servers to monitor multiple network segments, to reduce the number of discovery servers that must be installed to instrument an internet. Active server probes are indicated in the figure by thin arcs with arrowheads directed out from the discovery server. The scope of a discovery server is specified in terms of a set of networks or subnets to probe, as indicated by the dashed cloud surrounding three networks in the figure.

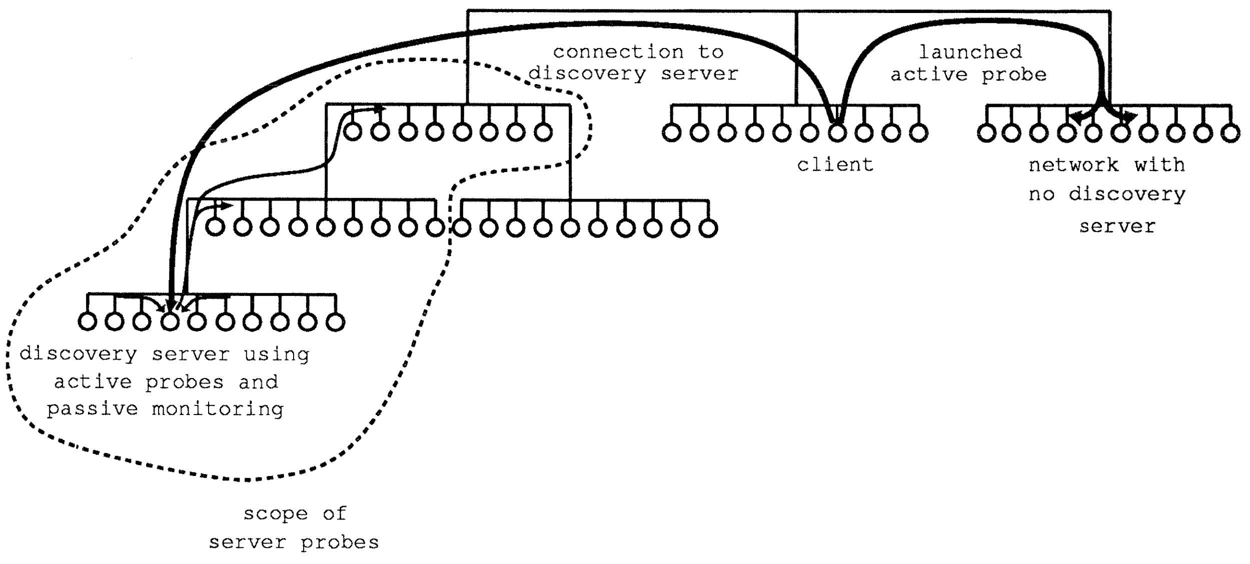

Figure 1: Macroscopic Network Discovery Architecture

This active probe capability can also be used by clients to "launch" a probing network discovery operation to a portion of the internet where no discovery server has been installed. In this fashion, a network manager could run servers on only those network segments of frequent concern (such as heavily loaded segments or critical backbones), but can still check into problems that arise on the parts of the internet that have not been instrumented.

Client requests are indicated by thick arcs in the figure. Of course, the ability of clients to launch active probes raises security issues, similar to those raised by SNMP's community name model of network instrumentation.[1]

[1] The SNMP standard currently provides no means for authenticating users who connect to servers. Instead, SNMP divides its information space into "communities", and any user who knows the name of a community maintained by a particular server can access the information in that community on that server.

Remotely launched probes may also generate more network load (and load that is less isolated onto network segments) than local discovery protocols. Clearly, the decision to use remotely launched probes must be made with some understanding of these issues. It might be prudent, for example, for a network administrator to configure a client that will not allow remotely launched probes, for use by non-network sophisticates.

In addition to the ability to use both passive and active probes, the architecture provides a discovery protocol scheduler and state manager that allows a system administrator to specify which discovery protocol modules will be scheduled and how frequently, according to his/her perceptions about the relative importance of the various protocol characteristics (timeliness of discovered information, etc.). Network protocols can be specified in a table that describes each protocol, where the executable code to use it resides, and its characteristics. In this fashion, the set of network protocols that are supported is easy to change without rebuilding the system.

We consider the ability to tailor the discovery mechanism to be an important aspect of our network discovery architecture. The range of protocols under consideration will be discussed in Section 3.

2.2. Scheduling, State Management, and Caching

The heart of the network discovery architecture is the scheduling and state management mechanism. This mechanism determines which discovery protocols will execute, and how frequently each protocol will execute. It maintains an in-memory representation of information associated with each network interface, host, router, and network link, and periodically dumps this information to a cache file that may be retrieved by a remote client. (Clearly, the scope of discovery must be limited for any particular server, so that the internal state it maintains does not become unmanagably large.)

Since some protocols do not provide sufficient information for determining all characteristics of interest in a network (e.g., it is not possible to determine which hosts are gateways using Internet Control Message Protocol Echo messages alone), the state manager must also maintain partial information that can later be filled in by correlating information obtained from multiple protocols. This mechanism can also detect conflicting information from different protocols. It may be useful to bring such inconsistencies to the attention of a system administrator, to flag potential problems.

The architecture supports two different modes for scheduling discovery protocols. *Explicit* mode allows a system administrator to specify explicitly which protocols will run, and how frequently each will run. *Implicit* mode allows a system administrator to specify his/her perceptions of the relative importance of various characteristics of the discovery protocols (timeliness, etc.). From this specification and a table of the protocol characteristics (to be discussed in Section 3), the scheduling and state management module chooses a schedule. In general the scheduling algorithm could take dynamic characteristics such as current network load into consideration.

To support scalable operation, the architecture includes a mechanism to cache discovered network information. In addition, queries may specify predicates, to avoid retrieving duplicate information. For example, a query may request remote retrieval of information about all of the hosts on a particular network segment that have been discovered since the most recent entry in the cache. In addition to enhancing scalability, this mechanism can improve the responsiveness of browsing operations, for the case where users are browsing parts of the internet that have been viewed recently.

Cache consistency is handled using a simple scheme involving two timestamps and a counter for each interface in the network. The first timestamp indicates when the element last changed status (e.g., the last time its Media Access Control layer address changed). The second timestamp indicates the date and time when the most recently recorded values for the interface were confirmed. The counter indicates the number of failures that occurred since the last confirmation. Combining these pieces of information allows the system to distinguish between an element that has not been observed recently but is still around (such as a node that crashes for a few days) vs. one that has probably been removed permanently vs. one that has changed characteristics (such as a host being replaced by a newer model with a different Ethernet address, whose name and IP address have not changed).

2.3. Network Representation

The user interface for this architecture allows a user to browse the state of the internet graphically, and display the discovered information. The user need not manually specify how to display the information. Instead, the user specifies the type of network state to be viewed (topology, protocol usage, etc.) and some indication of the scope of the internet to be browsed. The interface then connects to discovery servers in sequence around the internet, retrieves the necessary information, and displays it according to some display representation.

An important part of the user interface is the network representation. Our eventual goal is to allow the user to chose among several different representations, depending on their needs. In a wide area network, the representation could position nodes according to their geographical location. However, for some network management functions (such as detecting routing problems), this may not be the most appropriate representation. Another mechanism could place nodes on the screen with the appropriate links drawn between them, and allow users to interactively modify the on-screen layout. We favor a more automated approach, so that the network topology is always clear. A number of commercial network management stations force users to infer network topology from a list of host pairs (or a similar representation). We feel this task should be handled by the visualization service, with a mechanism provided to allow users to move nodes around and abstract away uninteresting detail, for example by specifying nodes on a particular network segment to be collapsed into a single display node.

Once information is displayed, the user uses the mouse to click on a node or link, to open a scrollable window of relevant information. Self describing fields are highlighted and viewable. For example, clicking on a link might display the type of link (e.g., leased line, satellite link, or fiber) and its bandwidth. Clicking on a host may display its network and Media Access Control layer addresses, and network protocol specific information (e.g., the subnet mask for IP networks).

3. Discovery Protocols and Implications

A wide variety of network protocols can be used for network discovery. In discussing these protocols, we assume the reader has basic familiarity with the Internet Protocols in general, and ICMP, UDP, ARP, and SNMP in particular. Details of these protocols are available in RFC's [Case et al. 1989, Plummer 1982, Postel 1980, Postel 1981a, Postel 1981b] and in Comer's TCP book [Comer 1988].

Network discovery protocols can be classified according to several different criteria. Some important criteria are:

- Are special privileges (e.g., UNIX[2] "root") or private information (e.g., SNMP community names) needed?
- Does the discovery protocol involve listening only to packets sent to the monitoring system, or does it require that all (or a large number) of packets be monitored from the attached network?
- How much load does the discovery protocol put on the host being used for discovery?
- Does the discovery protocol generate packets on the network? How much load does the discovery protocol generate?
- Does the discovery protocol work on Local Area Networks? Does it work on Wide Area Networks?

We now discuss a number of protocols for use in network discovery.

3.1. Simple Network Management Protocol

The most obvious protocol under consideration is the Simple Network Management Protocol (SNMP). It provides the most complete set of information from the widest number of routing devices. From SNMP one can obtain complete routing table information, including next hop, metric, and routing protocol for known routes. One can also obtain other management information such as queue length (indicating congestion), packet and byte counts (indicating utilization), and error counts (which aid in finding malfunctioning facilities).

A current operational problem is that SNMP is not running on many nodes around the Internet, although it probably has the widest distribution of any of the management protocols. More importantly, it requires knowledge of the SNMP "community name" in order to access the agent running on any particular router. This last issue is the main reason why we do not treat SNMP as a pivotal part of our discovery protocols.

3.2. Internet Control Message Protocol

The Internet Control Message Protocol (ICMP) provides several discovery mechanisms that can be used in discovering characteristics of a network. The biggest drawback to its use is that normal users on most UNIX systems

[2] UNIX is a trademark of AT&T Bell Laboratories.

are not permitted to generate ICMP packets, largely because ICMP packets can cause problems with network and host loading if not used carefully.

The ICMP Echo and Echo Reply messages can be used in several ways to determine information about a network. For example, the UNIX "ping" utility sends a sequence of ICMP Echo requests to a host. Properly configured hosts will send back ICMP Echo Reply packets. We have created a variation of this mechanism to send a single ICMP Echo Request to the broadcast address for the local network, and wait for responses from all of the machines on the addressed network, to discover all hosts on that LAN. The main problem with this approach is that it can cause a sufficient number of collisions on an Ethernet that some of the replies are lost. In addition, on networks that already have problems with "broadcast storms", this may cause severe storms. This problem can be reduced by setting the Time-To-Live (TTL) field in the packet header to a small value, so that broadcast storms last at most a short time. An alternative is to send an ICMP Echo Request individually to each machine in a range of addresses. For example, one could use this technique to discover the machines in a Class C network by cycling through 254 network addresses (all 2^8 addresses except 0 and 255, the broadcast address).

Van Jacobsen's "traceroute" utility also uses ICMP Echo Request packets, but sequentially increments the TTL through the values 1, 2, 3, etc., causing the packet to be returned to the originating host with an ICMP "TTL Expired" message by each router along the chain from source to destination. This allows the source host to discover the route that a normal packet would likely take from the originating machine to the ultimate destination (assuming that routes do not change frequently). This is one of the best mechanisms for determining network structure because it is implemented in a majority of routers and because it does not require knowledge of "private" information on the router.

The "pong" utility uses ICMP Echo Reply packets combined with "loose source routing" to specify that the packet must pass through the "destination" node on the way to the ultimate destination, which is the originating node. This tool uses the "Route Record" option of IP to record the addresses of the routers through which the packet travels from the source to the "destination" and back to the source. This tool suffers from the fact that many systems do not implement the "Route Record" IP option, and that many also do not handle "loose source routing". Moreover, the number of slots reserved in the IP packet header for recording route hops is smaller than the current diameter of the Internet.

The Subnet Mask Request and Subnet Mask Reply messages are very similar to the Echo Request and Echo Reply messages, except that they provide additional information from the sampled machine. This is one way to determine the subnet mask of a network.

3.3. Address Resolution Protocol

The Address Resolution Protocol (ARP) can be used to determine the presence of a machine on a network, as well as the machine's MAC layer address. On an Ethernet, the ARP protocol is used to find the Ethernet address for a particular IP address on that Ethernet. This information is normally kept in a cache file which is accessible by a user program.

Given the Ethernet address of a machine, it is possible to determine the make of the network interface, because each manufacturer has an assigned range of Ethernet addresses.

A variation of Paul McKenney's EtherHostProbe program can be used to check for the presence of an entry in the ARP table. If no entry is found, and if the destination node is on the same subnet, then EtherHostProbe will attempt to send a UDP Echo packet to the destination node. Any reply is ignored, but the ARP table is checked again for an appropriate entry. This process is repeated for a range of hosts and, while slower than a "broadcast ping", produces reliable results. Clearly, it is only feasible to sequence through a fairly small range of hosts (perhaps on the order of hundreds).

One can also implement a discovery protocol that simply monitors broadcast ARP requests on a network. This imposes very little overhead on the network, yet it eventually discovers most of the machines that are sending packets onto the network (except, for example, if the monitoring machine cannot keep up with the network traffic). One can also monitor ARP replies, to discover all of the receiving hosts on a local subnet. However, this requires that the Ethernet controller be operated in "promiscuous" mode, receiving every packet transmitted on the Ethernet.

3.4. User Datagram Protocol

The User Datagram Protocol (UDP) can be used to send packets with little overhead and, depending upon the port number, without specific user privileges. The Routing Information Protocol and the RWHO protocol both broadcast UDP packets to advertise their information. Individual users can use the Echo, Daytime, Time, Systat, Netstat, and other ports [Postel 1979] to get information from remote machines, but not all machines enable all these services, so they are not relied upon in our discovery protocols.

3.5. Transmission Control Protocol

The Transmission Control Protocol (TCP) can also be used for discovering information about remote machines. It does, however, present higher overhead than UDP. For example, one could make a TCP connection to the port for the Simple Mail Transfer Protocol (SMTP). Most machines providing mail service will respond with their host name. Other services (such as Daytime and Time) may also be used, but they are less commonly accessible.

3.6. Domain Naming System

The Domain Naming System can be used to determine a large amount of information about a network, including some topological information in certain cases. In particular, the DNS can be used to discover all registered hosts on a network.[3] Multiple addresses for the same name can indicate likely routers. Multiple names for the same address may also indicate possible routers, but with less probability. Host names that include "-gw" or "-rt" usually indicate routers. The subnet mask can frequently be deduced from a list of all registered hosts.

3.7. Routing Protocols

Routing information is the most likely source of information related to the structure of a network. This information is frequently available in a host's own routing tables (as can be seen using the UNIX "netstat -r" command).

The most common routing protocol is the Routing Information Protocol (RIP). RIP information can be acquired from the network by monitoring RIP packets that are broadcast onto a local Ethernet. Because they are broadcast packets, this does not require setting the interface into promiscuous mode. It is necessary to perform much the same function as would be done by a routing server because many systems readvertise onto the Ethernet routes that were learned from that same Ethernet, thus providing false indications that they have a route to the advertised networks.

RIP information can also be acquired from RIP sources by issuing RIP Request/Poll packets, as is done in the UNIX RIPQUERY program. This avoids the task of continually monitoring the network for RIP broadcasts, but has the same problem of discerning the real routers from those hosts that are just passing on hearsay evidence.

Other routing protocols include the Exterior Gateway Protocol (EGP) [Rosen 1982], the Border Gateway Protocol (BGP) [Lougheed & Rekhter 1990], and the HELLO protocol [Mills 1983]. However, these protocols are not generally accessible to the average host, and are not in use on most Ethernets.

Table 1 summarizes the network discovery protocols under consideration. Double lines indicate grouping as per the subsections above.

4. Prototype

The current prototype is implemented in C on top of UNIX, and runs on the University of Colorado local area internet. While the current set of discovery protocols are based on IP, they may be extended for use in a multiprotocol environment. We used a simple ASCII representation of data to aid debugging.

[3] Except on networks where Domain "zone transfers" have been disabled.

Protocol	Requirements	Information	Prevalence of Implementation	Problems
SNMP	community name	topology, load	broad	varying community names, network load, privacy
Ping	root permission	host, delay	very broad	network load
Broadcast Ping	root permission	multiple hosts	broad	collisions
Sequential Ping	root permission	multiple hosts, delay	very broad	slow speed, network load
Traceroute	root permission, ability to modify TTL	route to specified address, delay	medium	slow, network load
Pong	root permission, loose source routing, route record	route to/from specified address, delay	narrow	slow, network load, limited implementation
SubnetMask	root permission	host, subnet mask, delay	medium	slow, network load
ARP table	-	communicating hosts, manufacturer	very broad	only see hosts to which this host has sent packets on this LAN
ARP requests	read access to broadcast traffic	communicating hosts, manufacturer	very broad	only see hosts that are sending packets on this LAN
ARP replies	read access to all network traffic	communicating hosts, manufacturer	very broad	only see hosts that are sending or receiving packets on this LAN
EtherHostProbe	-	hosts, manufacturer	very broad	only see hosts that are currently answering ARP requests on this LAN
UDP	-	host, delay	narrow	slow, network load, limited implementation
TCP	-	hosts, host name if SMTP host, date/time	very narrow	-
DNS	-	most host names and addresses in a domain, perhaps routers	broad	requires sifting through much information
"netstat -r"	login on host	routers, known networks, metrics	broad	hearsay, fixed metrics
RIP watch	read access to broadcast traffic	routers, known networks, metrics	broad	hearsay, fixed metrics
RIPQUERY	-	routers using RIP, known networks, metrics	medium	hearsay, fixed metrics, must look for routers, miss other protocols
EGP, BGP, HELLO	a place to monitor protocol traffic	routers, known networks, metrics	narrow	not widely accessible

Table 1: Characteristics of Discovery Protocols

We discuss various parts of the prototype in more detail below.

4.1. Control Flow

The modular decomposition and control flow of the system is illustrated in Figure 2. In this figure, directed arcs indicate control flow. Data flows in both directions of each arc. We discuss the various modules and control flows below.

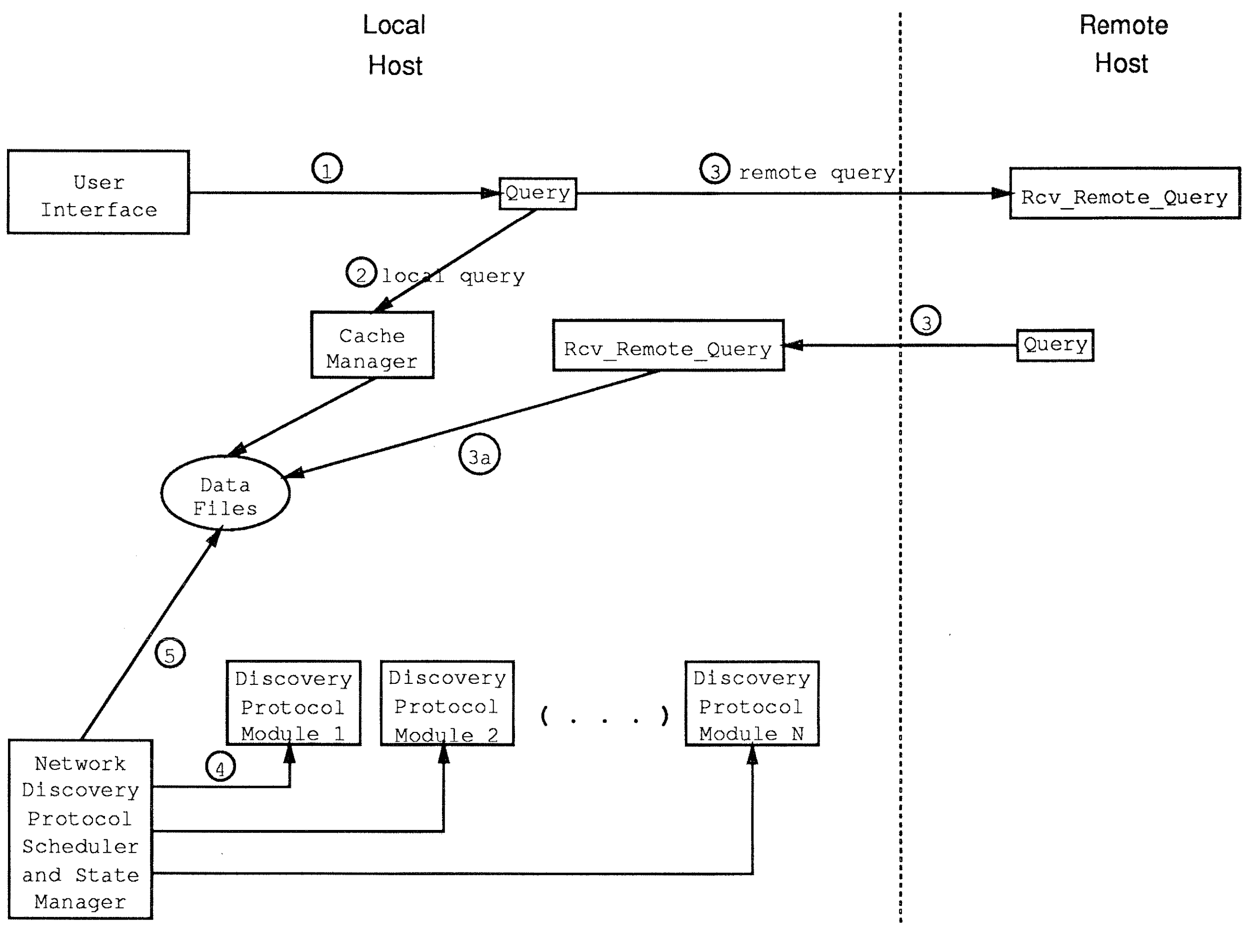

Figure 2: Modular Decomposition and Control Flow of System

The query mechanism presently consists of two major executable components, called *Query* and *Rcv_Remote_Query*. *Rcv_Remote_Query* is a server that runs continuously on each host running a network discovery suite. This component accepts TCP connections from clients, and transfers the data for a particular type of discovered resource information over these connections. In general, a number of different types of information could be discovered using the architecture described in Section 2 (including, for example, local printers). In the current prototype, only network and host information is discovered.

Query is a stand-alone executable invoked by the interface module, as illustrated in arc 1 of Figure 2. *Query* has four command line arguments: the name of the host to query, the type of resource, the predicate query, and the name of a file to which the results of the query should be written. If the host to query is not the local host and the data is not available in the local cache, a TCP connection is made with *Rcv_Remote_Query* running on the remote host, as illustrated from both perspectives in the arcs labeled 3 of Figure 2. On the remote host, *Rcv_Remote_Query* retrieves the needed data file, as illustrated in arc 3a of Figure 2. The remote data file is then transferred across the connection and saved on the local machine in the cache directory under the name *RemoteHostName_ResourceType*, as illustrated in arc 2 of Figure 2. If the host to query is the local host, the local cache file is used to satisfy the query, and the TCP connection/file transfer sequence is not necessary.

Once the needed data file has been obtained, the query is computed, and the result is written to the output file. Query predicates currently support only the AND of simple dyadic operators (=, <, and >). The query evaluation mechanism applies no interpretation to the data. Instead, query evaluation is based on simple string comparison operations. This has the advantage that one query mechanism can serve for all query types. The disadvantage of this approach is that field names used within the query must correspond exactly to those used in data files, or comparisons will fail. This problem will eventually be eliminated by generating queries in a semi-automatic fashion, by having the user select query fields and operators from a menu, and then having the interface form the specific query before passing it to the query module.

Arc 4 of Figure 2 represents the protocol scheduler and state manager scheduling one of the network discovery protocols. Arc 5 represents the memory-resident state representation being dumped to the cache files on disk.

4.2. Discovery Protocol Suite

We have implemented discovery mechanisms based on three different sets of discovery protocols. The first set of protocols uses EtherHostProbe to sequence through a range of machines on a local network segment, to discover the Ethernet addresses of each node on the network, and then probes the ARP cache to determine the Internet address associated with each of these hosts. It then issues a Domain Naming System lookup, to determine the host name associated with each node.

The second protocol set uses RIPQUERY to probe the network with RIP request packets on a local network segment, to determine the gateway topology of the network.

The third protocol set uses broadcast "ping" to discover the hosts on a remote network segment. To reduce the problem of broadcast storms, we are modifying this protocol to use a sequentially incremented Time-To-Live mechanism similar to Van Jacobson's traceroute program, so that a broadcast directed at a remote network will reach that network with TTL = 1.

Currently the various parts of the prototype are not integrated. The user interface can cause data files to be transferred from a cache, but this cache must be filled by running discovery protocol modules manually. Moreover, there is no mechanism to manage partial state as it is accumulated from the various discovery protocols. We are currently completing the integration of these pieces.

4.3. User Interface and Visual Representation

The user interface for the network visualization tool is implemented on top of the X windowing environment, using the OpenLook-based XView toolkit. Using this interface, users select the type of information to be discovered/displayed, and the remote network to observe. Currently the user must simply know which remote networks support discovery servers. In the future, the system will support an automated means for discovering this information.

The prototype currently supports a single network representation that is oriented towards allowing users to "zoom" in and out on multiple parts of a large internet. Hosts and links are arranged in a series of concentric rings around a central distinguished host, and the name of each host is displayed in an oval. Each ring indicates an additional network routing hop from the distinguished host. For example, a node two rings away would indicate that there are two gateways between that node and the distinguished node. In a network that has multiple paths between nodes, the shortest link path (as opposed to the current route) between nodes is used for chosing which ring each node is placed on. However, all links are shown, including links for longer routes. The distinguished host can be changed at will, providing a different perspective on network topology. The reason for chosing this representation is to make effective use of the screen space. It is essentially an *n*-ary tree mapped on to a series of rings.

Currently, there is no way for a user to specify that uninteresting information be removed from the display. We will do this by allowing the user to select a particular network and collapse all of the nodes in that network into a single display node. Similarly, users will be able to collapse networks. In this way, the user will be able to "zoom" in on several parts of a large internet simultaneously. Abstraction will be done automatically when there are too many hosts or networks for a given ring.

The output of the tool is shown in the screen dump from running the tool on part of the University of Colorado local area internet, in Figure 3. This figure only shows a subset of the hosts in the internet, to reduce crowding on the screen. For the present this effect was achieved by manually editing the cache file entries, but the same effect will eventually be achieved by using the abstraction mechanism described above. In this figure, the central node (piper) is a gateway, and the nodes above and below this node represent hosts on the two networks between which this gateway exists.

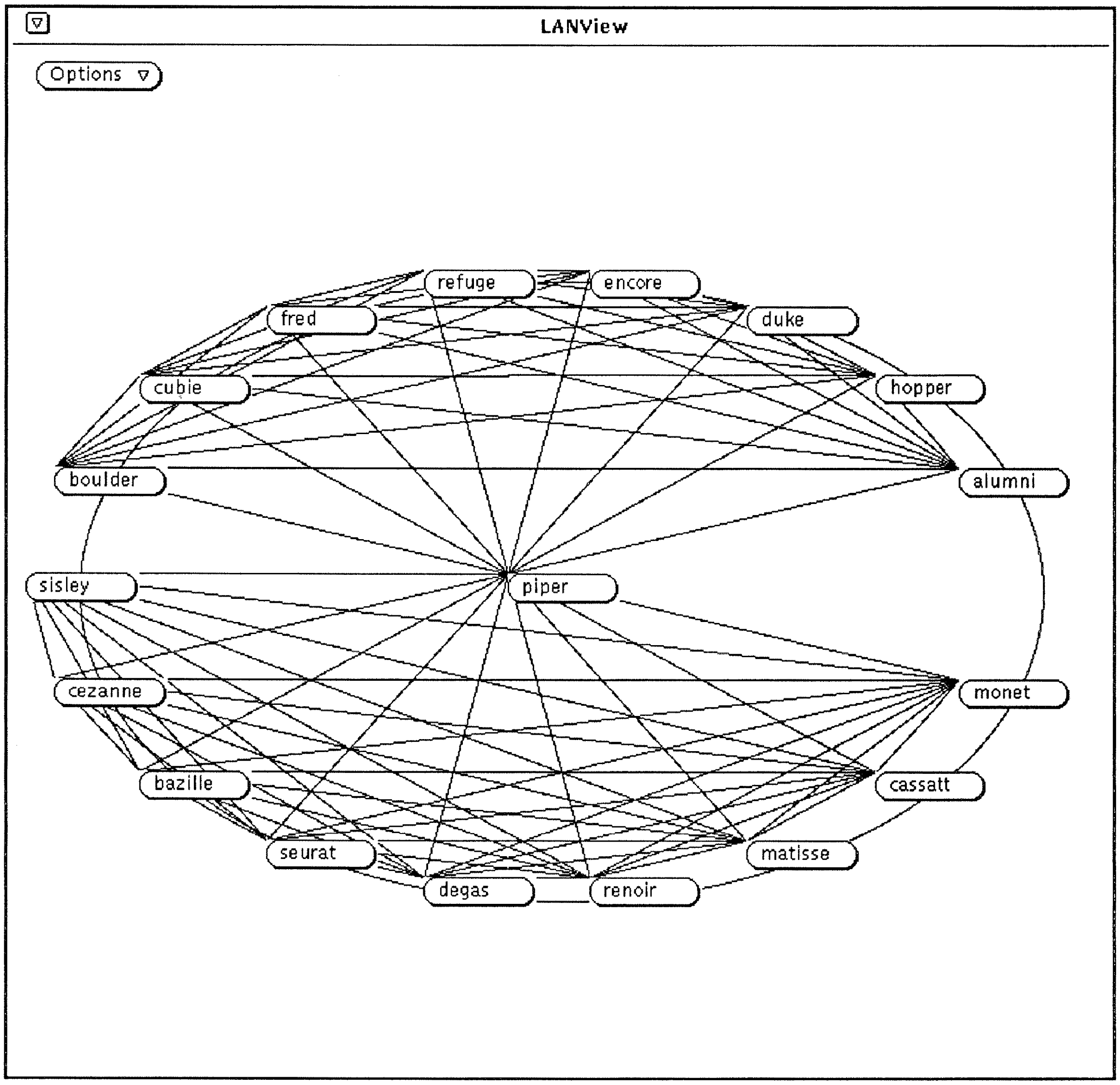

Figure 3: Screen Dump From Example Network Discovery Session

5. Summary and Future Work

In this paper we have presented an architecture for discovering characteristics of large internets, and displaying this information to a user visually. Our approach uses a very loosely coupled architecture that does not require global agreement over a particular network management standard, such as SNMP. Instead, we use a number of different network protocols and information sources to derive information about networks, cross-correlating this information when necessary to determine important characteristics or to uncover inconsistent information.

An explicit component of our approach is the recognition that different sources of network information have different characteristics with respect to timeliness, expense, danger of generating network problems, and completeness. Our architecture gives the network administrator control over which discovery protocols are used, and how frequently each is scheduled, to allow the system to be tailored to local perceptions of the relative importance of the various discovery protocol characteristics. Moreover, our architecture focuses on supporting network management in large scale internets. The support here takes the form of caching and predicate queries to reduce traffic generated by network visualization, as well as a user interface that allows differing network representations depending on the type of information and scale of network being monitored.

Our high level goal is to explore the extent to which one can integrate and manage a heterogeneous, administratively decentralized network by viewing it as an instance of a resource discovery problem. Moreover, our use of multiple protocols to support network visualization indicates an interesting light in which to view resource discovery. In a sense, resource discovery acts as a "glue" to allow non-network sophisticates to understand important characteristics of a complex network environment, by correlating and extracting relevant information from a multitude of protocols, and presenting this information visually.

The architecture we have presented can be used to discover a number of different types of information. For example, in addition to network-related information such as topology and protocol usage, the architecture could be used to discover the existence of printers and other shared devices.

We have built a prototype implementation of this architecture that can collect information using a few network protocols, and display the information using a particular network representation. We are currently extending this prototype to incorporate other discovery protocols and network representations; to manage partial state as it is accumulated from the various discovery protocols; and to support explicit as well as implicit protocol scheduling modes. Once the prototype is more mature, we will deploy a collection of servers around the global TCP/IP Internet, in an experimental effort to determine the extent to which this paradigm can successfully support network management and network integration functions in such a large, decentralized environment.

Acknowledgements

We would like to thank Trent Hein for implementing one of the discovery protocols used in this system. We thank Carl Malamud, Evi Nemeth, Mike Smith, Curt Stevens, and Alex Waterman, who provided feedback on an earlier draft of this paper.

This material is based upon work supported in part by NSF cooperative agreement DCR-8420944, and by a grant from AT&T Bell Laboratories.

6. Bibliography

[Case et al. 1989]
J. Case, M. Fedor, M. Schoffstall and C. Davin. A Simple Network Management Protocol (SNMP). Req. For Com. 1098, Apr. 1989.

[Comer 1988]
D. E. Comer. *Internetworking with TCP/IP: Principles, Protocols, and Architecture.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[Kislitzin 1990]
K. Kislitzin. Network Monitoring by Scripts. *Lisa IV*, Oct. 1990.

[Lougheed & Rekhter 1990]
K. Lougheed and Y. Rekhter. A Border Gateway Protocol (BGP). Req. For Com. 1163, June 1990.

[Malamud 1991]
C. Malamud. *Analyzing DECnet/OSI Phase V.* Van Nostrand Reinhold, New York, NY, forthcoming, 1991.

[Mills 1983] D. L. Mills. DCN Local-Network Protocols. Req. For Com. 891, Dec. 1983.

[Plummer 1982]
D. C. Plummer. An Ethernet Address Resolution Protocol -- Or -- Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. Req. For Com. 826, Nov. 1982.

[Postel 1979]
J. Postel. Assigned Numbers. Req. For Com. 758, USC Information Sci. Institute, Aug. 1979.

[Postel 1980]
J. Postel. User Datagram Protocol. Req. For Com. 768, USC Information Sci. Institute, Aug. 1980.

[Postel 1981a]
J. Postel. Internet Control Message Protocol. Req. For Com. 792, USC Information Sci. Institute, Sep. 1981.

[Postel 1981b]
J. Postel. Internet Protocol - DARPA Internet Program Protocol Specification. Req. For Com. 791, USC Information Sci. Institute, Sep. 1981.

[Reissig 1990]
W. C. Reissig. Dynamic Network Management Using the Simple Network Management Protocol (SNMP). Tech. Rep. 90-08-04, Comput. Sci. Dept., Univ. Washington, Seattle, WA, 1990. M.S. Thesis.

[Rosen 1982]
E. C. Rosen. Exterior Gateway Protocol (EGP). Req. For Com. 827, Bolt Beranek and Newman Inc., Oct. 1982.

[Schwartz 1989]
M. F. Schwartz. The Networked Resource Discovery Project. *Proc. IFIP XI World Congress*, pp. 827-832, San Francisco, CA, Aug. 1989.

[Schwartz & Wood 1990]
M. F. Schwartz and D. C. M. Wood. A Measurement Study of Organizational Properties in the Global Electronic Mail Community. Tech. Rep. CU-CS-482-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Aug. 1990. Submitted for publication.

[Schwartz 1990]
M. F. Schwartz. A Scalable, Non-Hierarchical Resource Discovery Mechanism Based on Probabilistic Protocols. Tech. Rep. CU-CS-474-90, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, June 1990. Submitted for publication.

[Schwartz & Tsirigotis 1991]
M. F. Schwartz and P. G. Tsirigotis. Experience with a Semantically Cognizant Internet White Pages Directory Tool. To appear, *J. Internetworking Research and Experience*, 1991.

[Schwartz et al. 1991]
M. F. Schwartz, D. R. Hardy, W. K. Heinzman and G. Hirschowitz. Supporting Resource Discovery Among Public Internet Archives Using a Spectrum of Information Quality. To appear, *Proc. 11th IEEE Int. Conf. Distrib. Comput. Syst.*, Arlington, TX, May 1991.

[Schwartz 1991]
M. F. Schwartz. Resource Discovery and Related Research at the University of Colorado. Tech. Rep. CU-CS-508-91, Dept. Comput. Sci., Univ. Colorado, Boulder, CO, Jan. 1991. Submitted for publication.

[Silicon Graphics Computer Systems 1990]
Silicon Graphics Computer Systems. NetVisualyzer Application Guide. Technical product documentation, July 1990.

[Stine 1990] R. Stine, editor. FYI on a Network Management Tool Catalog: Tools for Monitoring and Debugging TCP/IP Internets and Interconnected Devices. Req. For Com. 1147, SPARTA, Inc., Apr. 1990.

[Sun Microsystems 1990]
Sun Microsystems. *SunNet Manager — Installation and User's Guide.* Sun Microsystems, Inc., Mountain View, CA, Oct. 1990. Part No: 800-5512-05.