MARCH 1990

NSL Research Report RR-1

Local-Area Internetworks: Measurements and Analysis

Glenn Trewitt

digital **Network Systems Laboratory** 250 University Avenue Palo Alto, California 94301 USA

The Network Systems Laboratory (NSL), begun in August 1989, is a research laboratory devoted to components and tools for building and managing real-world networks. Current activity is primarily focused on TCP/IP internetworks and issues of heterogeneous networks. NSL also offers training and consulting for other groups within Digital.

NSL is also a focal point for operating Digital's internal IP research network (CRAnet) and the DECWRL gateway. Ongoing experience with practical network operations provides an important basis for research on network management. NSL's involvement with network operations also provides a test bed for new tools and network components.

We publish the results of our work in a variety of journals, conferences, research reports, and technical notes. This document is a research report. Research reports are normally accounts of completed research and may include material from earlier technical notes. We use technical notes for rapid distribution of technical material; usually this represents research in progress.

Research reports and technical notes may be ordered from us. You may mail your order to:

Technical Report Distribution
Digital Equipment Corporation
Network Systems Laboratory - WRL-1
250 University Avenue
Palo Alto, California 94301 USA

Reports and notes may also be ordered by electronic mail. Use one of the following addresses:

Digital E-net:	`DECWRL::NSL-TECHREPORTS`
Internet:	`NSL-Techreports@pa.dec.com`
UUCP:	`decwrl!nsl-techreports`

To obtain more details on ordering by electronic mail, send a message to one of these addresses with the word ‘‘`help`’’ in the Subject line; you will receive detailed instructions.

Local-Area Internetworks: Measurements and Analysis

Glenn Trewitt

March, 1990

Abstract

The effectiveness of a local-area internetwork is evident in its end-to-end performance and reliability. The dominant factors limiting end-to-end performance are router delay and the ‘‘fit’’ of traffic patterns to the topology. Internetwork reliability may be enhanced by conservative topological design, even in the face of unreliable network or router technology. These issues are crucial to network design.

This dissertation describes the results of studying the Stanford University Network, a large internetwork consisting of interconnected, local-area networks. To prove this thesis, I made measurements of router delay, local traffic characteristics, network topology, and traffic patterns in the Stanford internet.

This report reproduces a dissertation submitted to the Department of Electrical Engineering and the Committee on Graduate Studies of Stanford University in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

digital **Network Systems Laboratory** 250 University Avenue Palo Alto, California 94301 USA

Acknowledgments

I would like to thank John Hennessy, my advisor, for his support for this research and his patience with the many detours I've taken along the way.

David Boggs, my second reader, gave countless suggestions that made the prose and figures more direct and clear. His emphasis on directness and clarity helped focus the research as well.

This research would not have been possible without the equipment, encouragement, and conference admissions provided by the Networking Systems group at Stanford. Bill Yundt, Ron Roberts, and Jim Powell were notably helpful. Closer to home, I owe Charlie Orgish a tremendous debt for being ready to provide help and equipment on short notice.

My computing environment was greatly enriched by the tools provided by Mark Linton's *InterViews* project here at Stanford. John Vlissides and John Interrante's *idraw* program made drawing the figures a pleasant task.

Craig Partridge's partnership with me in network management standards activities helped shape many of my ideas about the tools and methods required for network management. This "distraction" provided me with many helpful contacts in the "real world" of networking.

Lia Adams, Harold Ossher, and John Wakerly were always ready to listen to my ideas and provide suggestions and encouragement. The many friends in my church and choir

helped remind me that there are other things in life besides research and dissertations.

Most of all, I thank Lucy Berlin, my wife. Her love and affection add joy and delight to my life.

The research reported in this dissertation was supported by the Stanford Computer Systems Laboratory.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This dissertation argues that router delay and topology are the two key points of local-area internetwork design. End users perceive two aspects of network performance: speed and availability. Speed is evident in the network as the end-to-end packet delay or as data throughput. Delay through routers is the major component of end-to-end delay. Network availability ("can I get from here to there, *now*") may be enhanced by conservative topological design, even if router or network technology is failure-prone. These two factors, more than anything else, determine the end user's perceptions of how "good" the network is.

This dissertation describes the results of studying the Stanford University Network, a large internetwork consisting of interconnected, high-bandwidth, local-area networks. To prove the above thesis, I made the following measurements:

> Router delay was measured, *noninvasively*, for *all* traffic through the router. Although the end-user perceives end-to-end delay, it is difficult to measure end-to-end delay for all types of traffic, since instrumentation can't monitor all traffic in the internet simultaneously.
>
> Local traffic characteristics, such as packet sizes and interarrival times.

Topological characteristics of the internetwork, such as redundancy and robustness.

Traffic patterns through the internetwork.

Many of the tools developed for this research turned out to be valuable network management tools, and continue to be used by the Networking Systems group at Stanford for day-to-day management of the internetwork.

1.1 Local-Area Internetworks

Local-area networks (LANs) are increasingly popular. Even the smallest workstations may have a network interface, and many rely on a network for essential services such as file transfer and mail. Ethernet (10 MBit/Sec) is currently the dominant technology used at Stanford[1]; parts of the network will soon be upgraded to optical fiber technology (100 MBit/Sec FDDI). One common characteristic of local-area networks is that they are limited in size, both in the number of nodes that may be connected and in their maximum physical size. The high speed and low delay characteristics of LANs are a direct result of their limited size.

The solution to this size limitation is to connect LANs together into a *local-area internetwork*. The connections between networks can be made with either *routers* (often referred to as *gateways*) or *bridges*. The choice between the two has been the topic of heated debate in networking circles, and both have valid applications. For the purposes of this dissertation, they are equivalent — they both selectively forward packets from one network to another and take a non-zero amount of time to process each packet. In this dissertation, I will primarily discuss routers, but most points, unless otherwise indicated, apply equally well to bridges.

With the size limit of individual local-area networks broken, it is common to see local-area internets with dozens of component networks, spanning several buildings or entire

[1]Token rings are not widely used at Stanford. Most of the results presented in this dissertation are independent of the particular technology.

campuses. Wide-area network technology, such as dedicated telephone trunks or satellite links, can be used to interconnect local-area networks to span cities or continents[Arm88]. This dissertation is primarily concerned with local-area internetworks.

Once a large local-area internetwork is in place, it quickly becomes indispensable to its users, who come to regard it as a utility, like electric power or mail service. As an internet pervades an organization and is used for more and more applications between more groups of people, its reliability becomes increasingly important.

The bad side of the increased physical reach of an internetwork is that the low-delay, high-speed, and reliability of a simple LAN is given up. Packets must pass through several network segments and routers, each of which adds delay. Even though the throughput of the individual networks and routers may still be as high as that of a single LAN, the added delay will reduce end-to-end throughput of many protocols. Reliability may be decreased as well because all of the intermediate networks and routers must be working to send a packet.

Most large local-area internetworks installed to date have been built in an *ad-hoc* fashion, installing links where there was conduit in the ground and adding routers or bridges when distance limitations were close to being exceeded. Some systematic design has been done, such as backbone topologies. In many cases, however, topologies have been determined by geographic factors or historical accident[Arm88, MKU87]. This is not to say that such internets are bad, merely that there is little engineering knowledge about what makes one local-area internet design better than another. This dissertation examines the structural and operational characteristics that determine local-area internetwork speed and reliability.

1.2 Wide-Area Internetworks

Wide-area packet-switched networks (WANs) have been around since the late 1960's and differ in many significant ways from LANs. The principal differences are network bandwidth and delay. Until recently, most long distance communications links were

56 kBit/Sec point-to-point circuits.[2] At such a low speed, a 1000-byte packet would take 140 *m*Sec to transmit, compared to .8 *m*Sec for 10 MBit/Sec Ethernet. More serious, however, is the propagation delay across long distances — 24 *m*Sec across North America vs. 5 *u*Sec across a typical 1 kM LAN. Finally, many WANs, such as the ARPANet, have internal switching systems that further delay packets.

This dissertation focuses on local-area internets, rather than wide-area internets. The speed-of-light delay across a WAN is much larger than the delay through most available routers. Because of this, simply improving router performance will not have a major effect on end-to-end delay, assuming that the router isn't saturated. This is not to say that router performance isn't important; it is absolutely critical that routers be able to handle the offered load and do reasonable things when overloaded [DKS89]. The important fact here is that the difference between a 20 uSec router and a 200 uSec router is unimportant if there is a 20 mSec speed-of-light delay across a WAN.

1.3 Model and Terminology

There are many components that make up an internetwork. Because many isolated groups have developed network technology, for different purposes, and in different cultures, there are many sets of terms in use to describe networks. Many different terms are used to describe the same components, and some terms are used in contradictory ways by different groups.

The following terms will be used in this thesis. This is not an attempt to define a "correct" set of terms, but merely a set that is self-consistent. Most of these terms are well-understood in the ARPA-Internet community, and were drawn out of experience in that environment. They are presented here in two groups. The first group consists of generic terms used for describing any network, and the second group consists of terms that are specific to IP (ARPA-Internet) networks.

[2] 1.5 MBit/Sec (T1) links are now common, and 45 Mbit (T3) links will soon be in use. This will increase bandwidth, but delay will remain high, limited by the speed of light.

Node A piece of equipment connected to a network. Examples of nodes are host computers, workstations, network servers, and routers.

Network A communication system that delivers packets from one node to another. In this thesis, "network" refers to the link layer (layer 2 in the OSI reference model). Examples of link-layer networks are Ethernet, token ring, and the Arpanet (which is accessed with either X.25 or the DDN 1822 protocol).

Local-area Network A network capable of covering only short distances — from a few dozen meters to a few kilometers. Examples include Ethernet, token ring, and FDDI fiber optic networks. Local-area networks are usually characterized by high data rates (greater than 1 MBit/Sec) and well bounded transit times.

Wide-area Network A network capable of covering long distances — thousands of kilometers or more. Examples include the Arpanet and satellite networks. Wide-area networks usually have lower data rates and longer transit times.

Data-link Layer (or simply *link layer*.) The set of services and protocols that provide the lowest level of communications visible from the outside. This is often the hardware layer (in the case of Ethernet and token ring), but may actually be implemented with other, invisible, layers supporting the services. This is the case for the Arpanet, which provides data-link services when viewed from the outside, but is actually implemented with packet switches and point-to-point links.

Link-layer Address An address that is used to name a node connected to a (link-layer) network. A link layer address is only meaningful within the scope of a single network. That is, only nodes connected to the same network may use link-layer addresses to name each other. The characteristics of this address depend upon the network technology and are different for Ethernet, token ring, telephone dial-up networks, etc. The important point here is that a particular link-layer address is only meaningful within the context of the network that the node is connected to.

Interface The connection between a node and a network. If a node is connected to more than one network, it will have multiple interfaces. Each interface has a (possibly

different) link-layer address on the network to which it is connected.

Internetwork (or simply *internet*.) A collection of networks, connected together by routers. In the OSI reference model, internetworks provide services corresponding layer 3, the "network" layer. The best-known example of an internetwork is the ARPA-Internet, which which has as its backbone the Arpanet.

Internetwork Address An address that identifies a node in an internetwork. It is conceptually a net,host pair, identifying both a network connected to the internet and a host connected to that network. Because of the correspondence to the OSI network layer, these are often referred to "network addresses".

Network Segment A portion of a network containing no routers or other switching devices that may store a packet for later forwarding. A network segment may contain other active devices such as amplifiers or repeaters (which do not store, and therefore, delay packets). The size of a network segment is usually limited by constraints of its underlying technology.

Router A device which accepts packets from one or more sources and sends them toward their destination address, given in the packet header. In common usage, and in this dissertation, this address is assumed to be a network layer (internet) address. Routers connect several *networks* together into an *internetwork*. The term is sometimes used more generally to include devices that route based upon addresses at any layer.

Gateway Generally, the same thing as a router. However, the term is also used to describe systems that translate higher-level protocols between systems. For example, an RFC-822 to X.400 mail gateway. In this dissertation, we will avoid the term gateway and use "router" instead.

Bridge A device that routes packets based upon link-layer addresses. Bridges are also known as "layer-2 routers" in the OSI community. Bridges connect several *network segments* together, forming one *network*. This connection is completely transparent

to packets on the network — except for additional delay, it is as if the aggregate is just one large Ethernet, token ring, or whatever.

Hops The number of networks between two nodes in an internet. Hosts on the same network are one hop apart. A pair of hosts with two routers (and three networks) separating them would be three hops apart.

Redundancy The amount of extra "stuff" in an internetwork beyond the minimum that is needed to connect the pieces together. I will discuss ways to measure this in Chapter 3.

Robustness The ability of an internetwork to withstand failures in its components and still function. Redundancy is often built into an internet to provide robustness, but some additions do better than others. I will discuss ways to measure robustness, and how it relates to redundancy, in Chapter 3.

Object	May Contain	Addressed by
Network Segment	"wires", repeaters, amplifiers	link-layer address
Network	network segments, bridges	link-layer address
Internetwork	networks, routers	internet address

Table 1.1: Hierarchy of Network Objects

The various layers described above fall into a hierarchy, from smaller to larger objects, which is summarized in table 1.1.

These terms are confusing, but most are rooted in history. Attempts to create rational sets of terminology seem to create even more confusion because they inevitably re-use old terms for different things. There are equivalent OSI terms for most of these. Most of them are several words long; they will be avoided in this dissertation.

While we are primarily concerned with analyzing local-area internetworks, it is important to remember that they are often connected to wide-area networks. The ARPA-Internet is the largest such packet-switched network in existence today, and has greatly influenced the development of packet-switched networks.

Arpanet A wide-area network, developed in the 1970's. It is built out of point-to-point links and custom-made packet switches called "IMPs" (Interface Message Processors) which provide connections to hosts and other networks.

ARPA-Internet A very large internet, with thousands of component networks, spanning most of the United States. It is often referred to simply as "the Internet" (note capitalization). Until recently, the backbone of the ARPA-Internet was the Arpanet, which provided most of the long distance communications services. That function has since been taken over by NSFNet[3] and various regional networks such as BARRNet[4].

IP "Internet Protocol", the underlying datagram protocol used in the ARPA-Internet. It provides 32-bit internetwork addresses, with variable boundaries between the network and host part of the address. The network number may be 8, 16, or 24 bits long. The rest of the 32 bits is the host address. IP packets are unreliable; reliable transport is provided by reliable end-to-end protocols.

IP Network The set of IP addresses with the same network number. IP networks are defined as class A, B, or C, depending upon whether the network number is 8, 16, or 24 bits long. This notion of "network" usually bears only a loose correspondence to the notion of a single link-layer network, as defined earlier. An IP network is a logical collection of nodes and may be realized with a single link-layer network or with a complex internetwork. In the later case, each component link-layer network is termed a *subnetwork.*

Subnetwork A term used in the ARPA-Internet community that refers to a scheme for partitioning a single logical IP network into several physical link-layer networks. With subnet addressing, the host part of an IP address is further subdivided, yielding a network,subnet,host triple. Subnetting allows an entire internetwork to be viewed as one IP network. Subnetting is primarily an addressing convenience. By giving additional structure to an internet address, routers "inside" an IP network

[3]National Science Foundation Network
[4]Bay-Area Regional Research Network

can route based upon the subnet number, while routers outside the IP network can ignore its internal topology.

1.4 Model of Internetworking

With the above terms in mind, the model of networks and internetworks used in this dissertation is as follows:

An internetwork is composed of networks connected together by routers. Different networks may use different technologies, *e.g.* Ethernet, token ring, FDDI, etc.

Each network consists either of one network segment, or several network segments connected together by bridges. All network segments in a network will usually be of similar kind. This is because bridges are usually not capable of translating link-layer addresses for one network type (*e.g.* Ethernet) into link-layer addresses for another type (*e.g.* token ring).

This model is sufficient to describe existing local-area networks, such as Ethernet and token ring networks. It can be extended to include point-to-point networks by treating each link as a single network.

1.4.1 Internetwork Maps

"A picture is worth a thousand words," is an appropriate adage for internetworks. An internet can be quite complex and can't be be described concisely with only words. The best picture of an internetwork is a map showing the component networks, network segments, routers, and bridges in a compact form. Throughout this dissertation, maps will be used to illustrate important points.

Figure 1.1 shows a schematic of a simple internetwork composed of a "backbone" network and several "leaf" networks. Routers (Rt) and a few hosts (H) are shown as examples.

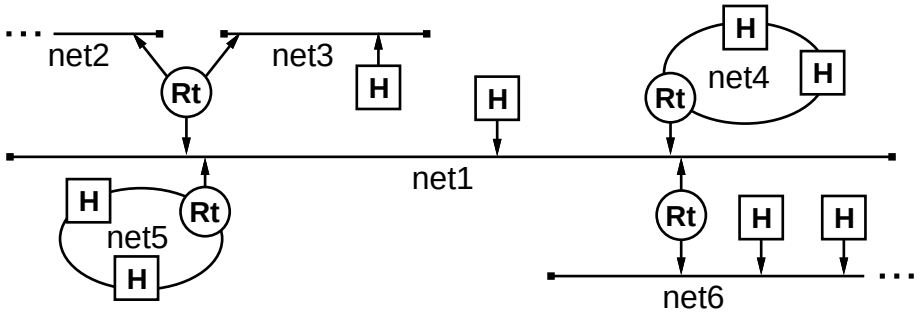

Figure 1.1: Simple Topology Schematic

Note that some networks are Ethernets and some are rings. Most of the area is taken up by the networks, which clutter the picture.

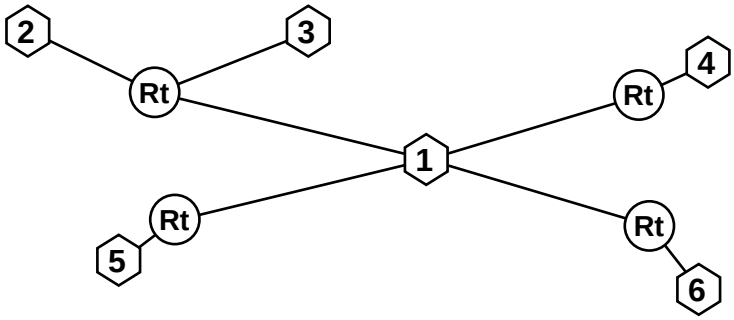

Figure 1.2: Simple Topology Map

The corresponding map for this topology is shown in figure 1.2. Routers are indicated by circles and networks are indicated by hexagons. Note especially that the hosts are not shown on this map. These maps have some interesting topological characteristics that are discussed in Chapter 3.

Figure 1.3 shows the same network, except that the backbone network is implemented with three network segments and two bridges (Br). None of the hosts or routers need know (or, in fact, can detect) that the bridges are there, because they pass the packets without modifying them.

Figure 1.4 shows the corresponding map. Even though this is a complex internet, the relationships between the components are quite clear. The lines show connections between

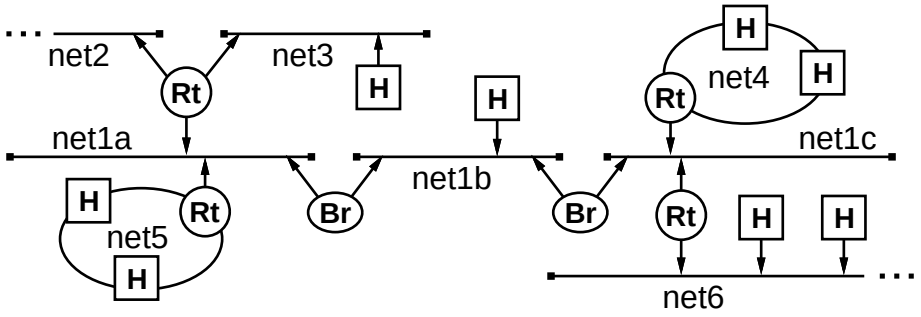

Figure 1.3: Topology Schematic With Bridges

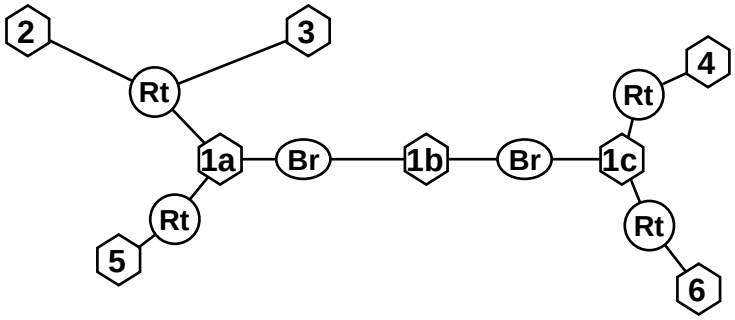

Figure 1.4: Topology Map With Bridges

routers (or bridges) and networks (or network segments), and therefore correspond directly to network interfaces. This correspondence can be exploited to display additional information on the map. Figure 1.5, for example, shows relative traffic levels on the router interfaces by varying the thickness of the corresponding lines. This single map

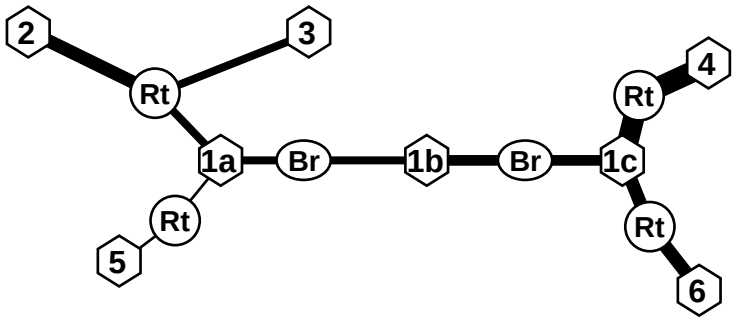

Figure 1.5: Topology Map Showing Traffic Levels

shows 40 pieces of information in an easy-to-understand manner. An annotated map like this will be used to illustrate traffic patterns in Chapter 4.

1.5 Synopsis

My thesis is that the key parameters affecting local-area internetwork "goodness" are router delay, internetwork topology, and traffic patterns. These are discussed in separate chapters.

Chapter 2: Router Delay. Although there are many possible causes of lower-than-desired throughput, most of them can't easily be changed. One parameter that can have a significant impact and that is also subject to improvement is router delay. Router delay is investigated by devising a queuing network model for a router, evaluating it, and comparing its predictions with delay measurements. The simple queuing network model does a very poor job of predicting delay. To reconcile these results, the assumptions and details of the model are investigated in depth.

Chapter 3: Internetwork Topology. The Stanford internetwork started as an ad-hoc collection of Ethernets, constructed with a little funding by a few academic departments. When a university-wide networking group was formed to create a more comprehensive, structured network, the topology was gradually improved to incorporate a redundant backbone spanning the entire campus. These changes were monitored and recorded over a period of two years. Methods for evaluating redundancy and robustness were devised and then tested against the changing topology of the Stanford internetwork.

Chapter 4: Traffic Patterns. The traffic in the Stanford internet was monitored to determine which subnetworks communicate the most and which paths are most heavily used. A few subnets account for most of the traffic in the internet, leaving large portions relatively unloaded.

Chapter 5: Conclusions. An important meta-issue for all of this research is the design of networking systems that support monitoring. Besides providing data for analysis, network monitoring facilities are critical to the operation and maintenance of any large internetwork. In this chapter, I discuss some of the capabilities which must be provided by a network monitoring system. I conclude with a summary of the research results presented in the dissertation.

Chapter 2

Router Delay

There are many interrelated factors that determine the speed of data transmission across an internetwork. As network data rates increase, the delay through routers becomes the significant limiting factor to network performance, in a local-area context. Intrinsic network delay, although heavily studied [MB76, AL79, TH80, SH80, Gon85, BMK88], is a non-issue when routers are added to the system. As we show in this chapter, router delay is, by far, the largest delay component in the Stanford internet. Furthermore, these packet delays have a large variance that cannot be attributed to any cause. Such large variances will adversely affect transport protocols that don't include variance estimates in their round-trip time calculations. Router behavior must be carefully examined as network speeds increase. Our goal is to determine what factors contribute to delay through a router, and how they can be used to predict actual performance.

2.1 Why are Internetworks Slow?

High delay and limited bandwidth are the two factors that limit a network's performance in a particular application. Delay may be more critical in some applications (remote procedure call) and bandwidth more important in others (bulk data transfer), but these

two parameters suffice to characterize a path between two points in an internet. The two substantial contributors in a local-area context are intrinsic router delay and delay due to congestion.

Delay becomes evident when some processing must wait for a response from the other end before continuing. Most stream protocols do not suffer from this because the "next data" is known ahead of time and can be sent without waiting for acknowledgment of the preceding data. Other applications, such as transaction processing or swapping suffer badly if there is even moderate delay, because some process must wait for one round-trip for the data to come across the net. These types of applications suffer precisely because they *can't predict what data they will need ahead of time* and must wait one round-trip to find out. In the absence of better prediction, the only path toward improvement is to reduce the round-trip time, either by bringing the data closer to the client (fewer hops) or reducing the delay across each hop (faster routers).

The factors affecting delay and bandwidth are summarized in the following sections.

2.1.1 Speed-of-Light Delay

Electrical signals (in coaxial cable) and optical signals (in optical fibers) propagate at 60-70% of the speed of light, or about 200 meters/*u*Sec. Putting the speed of light into these units illustrates how significant this delay can be, because one microsecond is a significant amount of time to a modern computer. A typical local-area internetwork that spans 2 kM has a speed-of-light delay of about 10 *u*Sec. The delay across the continental United States (about 3000 miles) is about 24 *m*Sec.[1] Even with current technology, 24 *m*Sec is a significant fraction of the total end-to-end delay across the US Internet . In just a few years, as router performance improves, this immutable delay will be the dominant delay in wide-area-internetworks.

Speed-of-light delay is almost totally immutable. The only way to change it would be

[1] 24 *m*Sec is the theoretical minimum, assuming no repeaters, etc. in the path. Long distance companies currently guarantee about 60 *m*Sec between "points of presence", transcontinental.

to use a different medium. Except for sending through air or vacuum (microwave or satellite links), there are no faster mediums available.

2.1.2 Link Bandwidth

Link bandwidth is the rate at which data is sent across a link. Since there may be several links in a path, possibly with different bandwidths, the smallest bandwidth limits the bandwidth through the path. Bandwidth affects some network usage more than others. It adds more delay to larger quantities of data, so interactive terminal or transaction traffic, requiring small packets of data, won't notice much difference between a 1 MBit/Sec link and a 10 MBit/Sec link. File transfer, requiring the movement of hundreds or thousands of kilobytes of data, will notice the full order-of-magnitude difference.

Link bandwidth is hard to change, once facilities are installed. Even starting from scratch, there are few choices. Today, in 1990, there are solid standards for what cable to install for 10 or 100 MBit/Sec, but standards do not exist yet for 1 GBit/Sec networks, even though such technology is expected to be available in the early 1990s.

2.1.3 Protocol Definition

The definition of a protocol strongly influences its performance in some circumstances. As a trivial example, a stream protocol that only allows one outstanding packet at a time will be vastly inferior to a sliding window protocol, at all but the slowest speeds. Therefore, one sees such simple protocols used for low-speed applications such as file transfer over slow telephone lines, but rarely over Ethernets. (Deliberately simple boot load protocols, such as TFTP, are a common exception.)

There are two different potential sources of delay in most protocols today. First, there is the delay experienced by each packet, every time it passes through a packet switch, associated with examining and routing that packet. This includes the time taken to do a checksum, interpret the header, and make a routing decision. More complicated headers,

address formats, and routing mechanisms all add to this time. Virtual circuit protocols have less of this overhead and datagram protocols have more.

Second, there is the delay associated with setting up a connection, which may manifest itself at different levels. In most packet stream protocols, several packets (at least 3) must be exchanged to establish a connection and get *any* data across. After the connection is established, it may take several round-trips to "fill up the pipe" and ramp up to full bandwidth.[Jac88] As bandwidths increase, per-connection delays represent larger and larger opportunity costs, when measured in bits. For a 1 GBit/Sec, coast-to-coast network, the single roundtrip required just to set up connection will be at least as great as the time required to send 10 MByte of data! Connection set-up and tear-down is more expensive for virtual circuit protocols than for datagram protocols, although this difference may change as bandwidths increase and bits (but not round-trips) get cheaper.

These are not insurmountable problems for high-bandwidth, long-distance networks. As bits become cheap and time becomes precious, trade-offs in protocol design can be made to get data there early, as well as to quickly estimate the "size of the pipe" so as to be able to ramp up transmission rate quickly.

Unfortunately, protocol definitions are difficult to change. A large part of the utility of any network protocol comes with widespread use. So, the most popular protocols, which might offer the biggest payoff from improvement, are the most difficult to change.

2.1.4 Protocol Implementation

Even the best protocol may not perform up to its potential if the implementation is poorly done. By design, sequenced packet protocols are able to recover from failures in the underlying layers, such as lost packets. Because of this resilience, they are often able to recover from errors in protocol implementations, although performance will almost certainly suffer. (In fact, this resilience makes it unlikely that subtle errors will be detected by casual observation.) Some implementations may have actual errors, such as sending duplicate packets or setting flag bits incorrectly once and getting them right on

the next try. More common are subtle tuning errors such as badly set retransmission timers, that, nevertheless, can wreak havoc in a network.[Jac88]

Protocol implementations can usually be fixed, although it may take some time for fixes to get distributed. A sufficiently motivated network administrator can make a significant difference in how quickly such upgrades happen.

2.1.5 Routing Delay

As described above, routers delay packets in several processing steps: packet reception, header decoding, routing decisions, and packet transmission. If we take the complexity of these tasks (due to protocol definitions) as a given, there are many different tradeoffs that may be made when designing a router. High performance may be achieved by using faster and/or more complex hardware, such as fast CPUs, shared memory for packet buffers, and multiple processors. Performance may also be improved with clever software techniques that use routing caches or process packets at interrupt time. *Cut-through* is a technique that starts sending a packet before it has been completely received, once a routing decision has been made, but this requires even more specialized hardware architectures, in addition to faster components.

Otherwise desirable features, such as security facilities to filter packets based upon source or destination addresses, add overhead and delay to the processing of each packet. Other features, such as network management facilities and sophisticated routing protocols add a background load that can also affect delay.

One important fact to keep in mind is that if a packet must be completely received before it can be transmitted (as is usually the case, except with cut-through), then a router will always delay packets by the time it takes to receive a packet, in addition to any computation that the router must perform. This is a significant delay even for “fast” networks such as Ethernet; a 1500-byte packet takes 1.2 *m*Sec to receive. This delay is experienced at *each* router or bridge in the path and is usually referred to as *store and forward delay*.

So, performance (delay) can be traded off against complexity and therefore cost. Routers are available now over a broad price and performance range, up to 80 or 100 MBit/Sec. Partridge [Par89] argues that 1 GBit/Sec routers will be achievable very soon.

2.1.6 Network Bottlenecks — Congestion

When more traffic is presented to an internetwork than it can handle, congestion results. This will produce delay, as packets pile up in routers, where they wait until other traffic clears. To a single packet stream it will appear that the link's bandwidth is lower, because it is now sharing the link with other traffic. The bandwidth reduction is inevitable, but the delay will be less severe if good congestion control is designed into the network protocols and routers. [RJ88, Jac88, DKS89]

Although traffic patterns are not instantaneously controllable, a network manager should keep an eye on what the traffic is doing over the long term. A manager can usually take some steps to compensate for "hot spots" in the internet, such as rearranging the topology. If the whole internet is overloaded, that is a different problem — new capacity must be added, perhaps in the form of higher bandwidth links or better routers.

2.1.7 So What Can We Control?

Many of these parameters are not under the control of the network designer. Speed-of-light delay and protocol design are certainly not, and the designer has only oblique influence over protocol implementation. The distance over which the network is to operate is probably part of the job description. So, the areas of flexibility are: link bandwidth (technology selection), router delay (product selection) and topology design.

The following tables show breakdowns of end-to-end delay for two scenarios: a current-technology local-area internetwork and a future-technology 1 GBit/Sec wide-area internetwork.

	Local-Area	Wide-Area, High-Bandwidth
distance	2 kM	4000 kM
bandwidth	10 MBit/Sec	1 GBit/Sec
avg. packet size	100 bytes	500 bytes
number of routers	2	6
router delay	1 *m*Sec	2 *u*Sec

Table 2.1: Two Sample Internets

The figures for the number of routers in the path are typical of current local-area internet [Arm88] installations and the NSFnet wide-area internetwork and are on the conservative side. The local-area packet size is typical of what was seen in the course of this research. The high-bandwidth packet size is an extrapolation based on trends toward longer maximum packet sizes (MTU) in faster networks[Par89], as well as observations of packet size increases at Stanford. In the past year, the average size has increased from slightly over 100 bytes to over 200 bytes at many routers. At least some of this increase is due to the increased network file system traffic across routers. The high-bandwidth router delay of 2 *u*Sec is based upon estimates of what will be attainable in the near future with fast processors and specialized architectures, as well as what will be required to be able to switch 500,000 Pkts/Sec.

The total end-to-end delay, not counting delay in the end-nodes, will be:

$$
\begin{aligned}
\text{TotalDelay} \quad & \left(\frac{\text{Distance}}{\text{Velocity}}\right) && \text{Propagation Delay} \\
& + \text{Hops} \times \text{RouterDelay} && \text{Router Delay} \\
& + \left(\text{Hops} + 1\right) \times \left(\frac{\text{PktSize}}{\text{Bandwidth}}\right) && \text{Store \& Forward Delay}
\end{aligned}
$$

	Local-Area	Wide-Area
Propagation Delay	10 *u*Sec	20 *m*Sec
Router Delay	2 *m*Sec	12 *u*Sec
Store & Forward Delay	240 *u*Sec	28 *u*Sec
Total Delay	2.25 *m*Sec	20 *m*Sec

Table 2.2: Breakdown of End-to-End Delay

Table 2.2 shows a breakdown of the end-to-end delay for the two sample internets. Each component is totaled over a packet's entire route. In the local-area internet, router delay

is the principal component of end-to-end delay. Reducing router delay below 100 *u*Sec would be less productive, because at that speed transmission delay starts to dominate. Cut-through then becomes more attractive, although only at significant hardware cost.

In a wide area network, propagation delay swamps all other factors, even if relatively slow routers are used. In the wide area, there is little point in reducing router delay just for the sake of reducing end-to-end delay, although delay will certainly be reduced as routers are made faster to handle the large volume of traffic in higher-speed networks.

2.1.8 End-to-End Delay Variance

The previous section described all of the sources of delay in an internet, and concluded that, in a local-area context, routers are the major fixed source of delay. Besides the need for low end-to-end delay, the variance of the delay is also an important factor, especially for transport protocols.

Transport protocols usually maintain an estimate of the round-trip time and use that to decide how long to wait before retransmitting a packet. One common choice for the retransmit timer is just some fixed multiple of the average round-trip time, which implicitly assumes a fixed delay variance. In this case, a high actual delay variance will cause some packets to be needlessly retransmitted. The solution to this problem is to estimate the round-trip time variance as well as the delay.

Another possible consequence of high variance is that some packets may be delayed enough that newer packets overtake them, resulting in out-of-order packet delivery. Although transport protocols are designed to cope with out-of-order packets, many interpret an out-of-order packet as a sign that the previous packet was lost, causing more retransmissions and more delay.

2.2 Factors Affecting Router Performance

There are three classes of influences on router performance: characteristics of the router, characteristics of the routed traffic, and external events (besides routed traffic) that demand the attention of the router. The two routers that I studied are quite different in terms of these characterizations.

The baseline speed of a router (given the most favorable environment) is determined by the intrinsic characteristics of the router itself. A router is a complex hardware/software system. One straightforward characteristic that can be measured is the speed of the hardware — CPU, backplane or bus, and communication interfaces. The software code to be executed for each packet is determined by the quality of the implementation and the complexity of the protocol and routing architecture. The code may also be more or less constant. If it is constant and small enough to fit into cache memory (if present), execution speed will be dramatically enhanced. Finally, most routers are complex enough to require a multi-tasking operating system to handle the variety of tasks the router is called upon to perform. The selection of the scheduling algorithm, and minimization of blocking due to critical sections can have a dramatic impact on routing delay. For example, if a packet to be routed has to wait for an in-process routing-table update to finish, that packet will likely experience a much longer delay than the average case.

The characteristics of the traffic to be routed also contribute to the delay experienced by individual packets. Average packet rate and size are important and easy-to-measure characteristics, but don't nearly tell the whole story. If packets arrive in bursts, they will be delayed more than evenly-spaced packets. Burstiness is, unfortunately, hard to quantify. A useful reference is the Poisson density function, which results in a exponential distribution of interarrival times. Analytical models often assume Poisson distributions, providing a baseline for comparison of other results. Unfortunately, real networks do not exhibit Poisson characteristics [SH80, Llo86, PDA86]. [Llo86] and others have proposed ways to measure burstiness which, unfortunately, require somewhat subjective parameters to define what looks "bursty". In section 2.5.4, I propose a simple technique that allows direct comparison to Poisson results.

2.2.1 Environmental Factors

The environment around a router can also affect the routing delay. Routers are an integral part of a larger system, and therefore must interact with the rest of the world, besides doing the primary work of routing packets. Two typical tasks are routing table maintenance and address resolution. Routing tables are updated by routing update packets, which are (usually) exchanged between routers. A single router must receive updates and use them to modify its routing table, as well as send out updates when something significant changes. A router that communicates directly with many others will spend a lot more time processing routing updates than a relatively isolated router.

Address resolution is one example of broadcast traffic that routers must handle. In the Internet, the Address Resolution Protocol (ARP) is used to find Ethernet addresses, given an IP address on the same network. (ARP is a specific case of address resolution; almost all internet architectures have some protocol to fill this niche.) A host broadcasts an ARP request and the host with that IP address responds. The requesting host caches the response. Some hosts, that aren't clever enough to do "real" routing, rely on a technique known as *Proxy ARP*, where a host blindly sends an ARP request, even if the target host isn't on the same network. If a router supports Proxy ARP (most do), then it responds to the ARP request as if it was the target host (with certain restrictions, mostly that the router is on a direct path between the two hosts). The originating host then sends packets destined for the target host to the router, which dutifully forwards the packets. The result, once the ARP process is done, is exactly the same as if the originating host knew how to find the appropriate router in the first place.

Proxy ARP works well, except for one thing: participating routers must listen to all ARP packets on the network and look up each IP address in their routing tables to determine whether and how to respond. This burden is normally small, unless there are protocol implementation errors that result in large numbers of ARP requests. Unfortunately, there are many subtle bugs, in many protocol implementations, that have persisted for years that produce "storms" of ARP requests[BH88]. Routers are especially vulnerable to such misbehavior.

Finally, other traffic on the networks to which a router connects will interfere with and delay the transmission of packets. Delay due to contention for the network has been investigated thoroughly for Ethernets [MB76, AL79, TH80, SH80, Gon85, BMK88]. For situations not involving overloads, this contribution to the total delay is quite small (typically less than 1 *m*Sec).

2.2.2 Case Studies

Measurements were made on two different routers, with significantly different characteristics.

	"Slow" Router	"Fast" Router
Processor	10 MHz 68000	24 MHz 68020
Routing Scheduling	via process	at interrupt time
Routing Table	table lookup	routing cache
Packet Copying?	Yes	No
Typical Delay	2 *m*Sec	100 *u*Sec

Table 2.3: Router Characteristics

Both of these routers use a non-preemptive, priority-based process scheduler. In the slow router packets are forwarded by a scheduled process (highest priority) which looks up the destination address in the main routing table. In the fast router, packets are forwarded by the interrupt handler which looks up addresses in a routing cache. Only cache misses have to wait for the scheduled process to do a full routing-table lookup. The other major difference is that packets are copied across the system bus in the slow router, while the fast router's two Ethernet interfaces share common packet buffer memory. It is clear from the delay figures that these techniques pay off well — even discounting the increased processor speed, the fast router is an order of magnitude faster than the slow one.

2.3 Queuing Network Model of a Router

The goal of this chapter is to understand the characteristics of a router, specifically, to model how it delays packets as it routes them. I chose to use a queuing network model because it is easy to analyze and requires few inputs. I initially use the simplest type of queuing network model, with one service center, fed by a queue, whose workload consists of a stream of packets to be routed, as shown in Figure 2.1. After comparing the predictions of the model to measurements, I identify the significant differences and attempt to determine their causes.

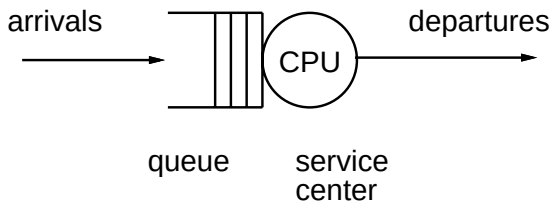

Figure 2.1: Simple Queuing Network Model

This queuing network model requires one input parameter, the average *service demand* at the CPU, which is the amount of time that the CPU must spend to process one packet, independent of how the time is divided up. That is, it doesn't matter to the model if a packet requires two slices of .5 and 1.5 *m*Sec or one slice of 2 *m*Sec. The service demand is 2 *m*Sec in both cases.

The actual service demand for routing packets was obtained from statistics kept by the router for the various processes it runs. Figure 2.2 shows an excerpt of the statistics for the slow router. Statistics are kept on the cumulative runtime and the number of invocations for each process. The average runtime per invocation is also computed, although it appears to be severely rounded down in this table. The **IP Input** process is the one that does the actual routing in the slow router. (In the fast router, most routing is done at interrupt time; the **IP Input** process only handles cache misses.)

```
PID Runtime (ms)     Invoked   uSecs Process
  5     14513508     5754595    2000 IP Input
  6        67452       62373    1081 IP Protocols
  9        74360       50021    1486 ARP Input
 12      2996704      358977    8000 GWINFO Router
 14        99964      272581     366 PUP Input
 15        64068      271825     235 PUP Protocols
```

Figure 2.2: Excerpt of Process Statistics

It isn't sufficient to just look at the average runtime, because when the **IP Input** process gets run, there may be several packets in the input queue, which will all be routed on this invocation. For this reason, the number of invocations will be less than the number of packets routed. The number of routed packets is obtained from other statistics kept by the router.

The model inputs we obtained from the router's statistics were 2200 *u*Sec for a working router and 2050 *u*Sec for an isolated test router. There is some variation with load, but not much. We also measured all of the other routers in the Stanford internetwork. Those running on comparable hardware had estimated service demands between 2000 *u*Sec and 2200 *u*Sec — a reasonably tight range.

2.3.1 Results of the Simple Model

The simple model shown in 2.1 has one parameter, the service demand ( ), and one independent variable, the *offered load* ( ) — the arrival rate of packets to be routed. The closed-form solution to this model[LZGS84, page 109] is quite simple, and produces three outputs, in terms of the offered load: the saturation packet rate, the average delay of a packet through the router, and the average number of packets in the system (CPU and queues):

Saturation rate $_{sat}$ 1

Residence time delay 1

Number in system

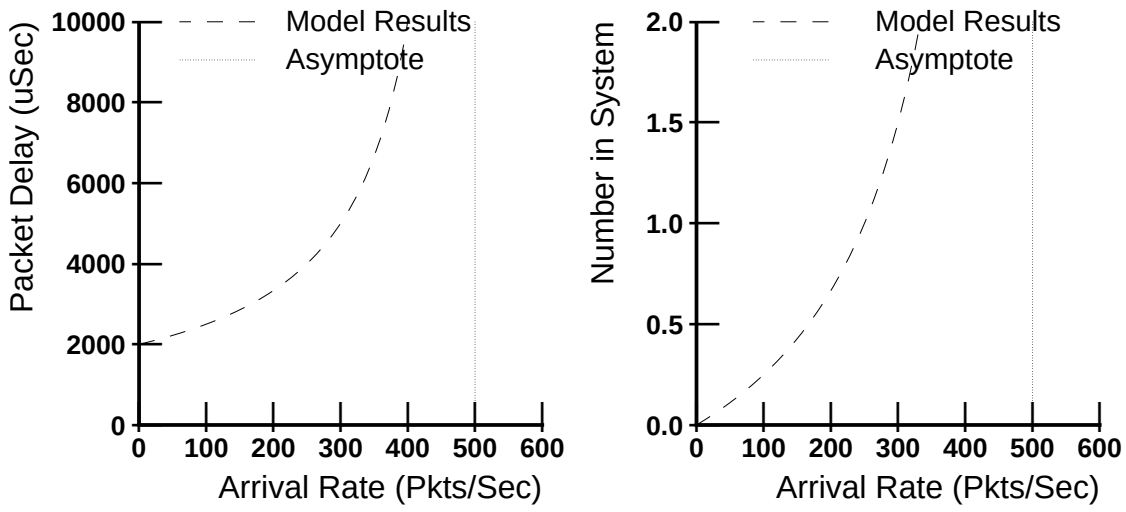

Figure 2.3: Delay and Number in System vs. Offered Load

These results are shown in Figure 2.3 for an example service demand of 2000 *u*Sec. The two graphs are fundamentally related by *Little's law*[LZGS84, page 42], which states that the average population in the system is equal to the product of the rate at which they arrive (and depart) and how long each stays in the system: . Little's law requires very few assumptions, so it applies to the actual system as well as this simple model. Therefore, both of these graphs contain exactly the same information. From here on, we will focus only on the packet delay relationship. If needed, customer populations can be easily calculated from that.

2.3.2 A More Complete Model

Although the simple model gives intuitively pleasing results for the delay through the router, it is important to keep in mind the ways that the model and its assumptions deviate from actual router behavior. Queuing network models, in general, make a number of assumptions about the characteristics of the system being modeled and its workload. The important ones that may be violated by the router's behavior are:

Service demand is independent of time. If some aspect of the processor or the packet load varies with time, such that packets at one time take longer to process than packets at another time, the model will give poor results. For example, the model will give bad predictions if the packet size has a strong influence on service demand and the average packet size varies a lot over time.

Service demand is independent of the load. If several packets arrive close together, they may be processed by the same invocation of the routing process. All but the first will be spared the scheduling overhead, reducing the service demand for such high-load packets.

Packet arrivals have a Poisson distribution. The distribution of packet arrivals (for a given arrival rate) is a characterization of how much packets bunch up and determines how much they get in each other's way, causing queuing delay. Models that assume a Poisson distribution (such as this one) tend to have simple solutions. It is well-documented that packet arrivals in packet-switched network are not Poisson[SH80, Llo86, PDA86]. Our measurements confirm this. The fact that an actual distribution deviates from Poisson does *not*, however, say whether there will be more or less queuing.

In addition, the simple model doesn't incorporate all of the potentially significant characteristics of the actual router:

The router does non-preemptive scheduling. Although the total amount of time the processor spends doing other (non-routing) tasks is relatively small, newly-arrived packets must wait for the current process to finish its task before they will be processed. Therefore, some non-trivial amount of extra delay *may* may be experienced by packets that arrive while the CPU is busy with another task.

Contention delay on output to Ethernet. Packets must wait for the Ethernet to be idle, and, if the Ethernet is busy, may have to wait for collisions to resolve before they can be sent. For a lightly-loaded Ethernet, collisions are unlikely, but the

random chance of having to wait for an Ethernet packet to clear can add up to 1.2 *m*Sec to the delay.

Variation in packet characteristics. Any packet characteristic that may make the router take longer to process the packet. This includes packet size, since it takes some time to copy the packet when receiving and transmitting. More delay will be introduced if the packet has to be copied over internal data paths. For Ethernet, just transmitting the packet can take 1.2 *m*Sec for a maximum-size packet. Packet destination is another possibility, if routing takes longer for some destinations than for others.

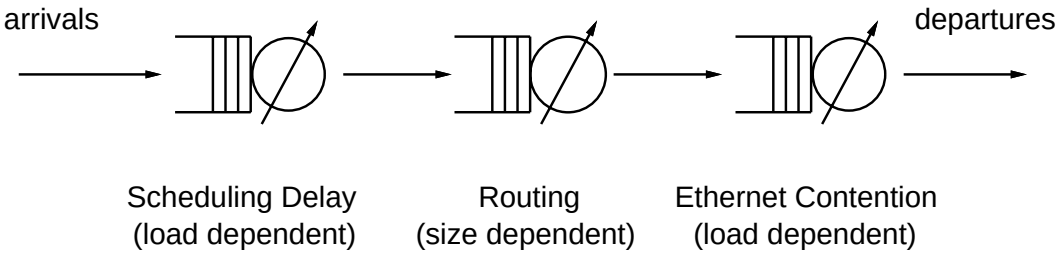

Figure 2.4: More Realistic Queuing Network Model

Figure 2.4 shows a more realistic model that incorporates these differences. In general, load-dependent and traffic-dependent variations in service demand are difficult to incorporate into a queuing network model. If these variations have a significant effect on delay or queuing, then the model does not have a closed-form solution; it may require more sophisticated analysis or simulation.

2.4 Router Measurements

We made measurements of delay through a loaded router in regular service in the Stanford network. We used hardware that was identical to the slow router, with two Ethernet interfaces connected on either side of the router under test. The measurement software

captures all packets on both subnetworks and matches packets departing from one interface of the router with packets that had arrived at the other interface. The data is recorded and reported to a fileserver via a simple file transfer protocol (TFTP) in real time. Additionally, the packet trace can be run through a simple simulator, also in real time, to compare the delay and queueing for each packet in the trace.

Two types of data are returned to the fileserver: packet traces and periodic statistics reports. For each routed packet, the following information is reported:

Arrival time.

Packet size.

Time since the last routed packet arrived.

Delay through the router, from one network to the other.

Source and destination IP addresses.

Direction through the router.

Information needed to compute the amount of traffic present on the destination network at the time this packet was transmitted: the number of packets and bytes since the last routed packet was sent on that network.

In addition, routing update packets and ARP packets (both incoming and outgoing) are timestamped and reported, to help get a picture of what's going on inside the router at particular times.

Besides the packet trace, statistics are reported periodically, typically every one or two seconds. Most of the statistics are concerned with the volume of traffic (packets/second and bytes/second) for different types of traffic. From this data, the packet and byte rates, average size, and average network utilization is computed. Statistics are reported on:

Total traffic volume on each connected network.

Traffic into and out of the router, for comparison against forwarded traffic.

Traffic forwarded by the router. Average routing delay is also reported here.

Statistics about interface behavior, to help estimate the number of packets not received because the interface was not ready.

Simulation results, which include statistics about forwarded traffic (as above) as well as simulated delay and queue length distributions.

A block diagram of the measuring setup is shown in figure 2.5. The system watches all packets on both networks and collects the ones that interact with the router. At interrupt time, received packets are timestamped, essential information is extracted and queued for processing. The core of the software is the *matching queue*, which holds records of packets arriving at the router for later comparison with departing packets. The queue is limited to a maximum of 50 unmatched packets. At this length, unmatched packets take between 0.1 and 1 second to be discarded, so this is not a likely source of missed packets. The general procedure is as follows:

1. ARP broadcasts and routing updates are reported.
2. Forwarded packets are distinguished by the fact that their source or destination Ethernet address belongs to the router, but the corresponding IP address belongs to some other host. These packets are classified as either arriving at or departing from the router and ...
3. Packets arriving at the router are put in a matching queue (one for each direction).
4. Departing packets are matched against packets in the queue. They are matched based upon source IP address, IP identifier, and IP fragment information. Each match generates one record of a forwarded packet, with delay computed by subtracting the timestamps.
5. Forwarded packet records may be reported.

6. Alternatively, forwarded packets may be processed through a simulator that models the router queues and computes average delays and queue lengths:

   (a) The forwarded packet records are sorted by arrival time (They were generated based upon departure time.)

   (b) The packet records enter a simulator that models queue length inside the router.

   (c) At regular intervals (typically selected as one or two seconds), the average delay and queue length distribution over the interval is computed and reported along with other statistics to give a picture of the workload.

This may seem like a lot of trouble to go to just to get this data. The obvious alternative is to instrument the router code and have the router itself report the data. There are two fundamental reasons for not instrumenting the router. The most basic reason is that the added instrumentation would be additional work for the router to do, probably adding delay and variance. The most compelling reason is that the source code was not available. Even if it had been, adding the instrumentation would certainly cause some router crashes, interrupting network service to a very compute-intensive building. The administrators of the network did not want such disruption; neither did I. The other alternative was to build an instrumented router and use it to connect a test network, relying on artificially-generated traffic to provide a workload. However, since the characteristics of "natural" traffic are one of the unknowns, the realism of the test would suffer.

2.4.1 Measurement Results

Figure 2.6 shows the results of the measurements, superimposed with the results of the queuing network model. The queuing network model is evaluated with a service demand of 2200 *u*Sec, which is indicated in the key. Samples were taken every 2 seconds over a period of 14 minutes; a typical sample has about one hundred packets. Even with this large sample size, there is a large variation in average delay for similar workloads.

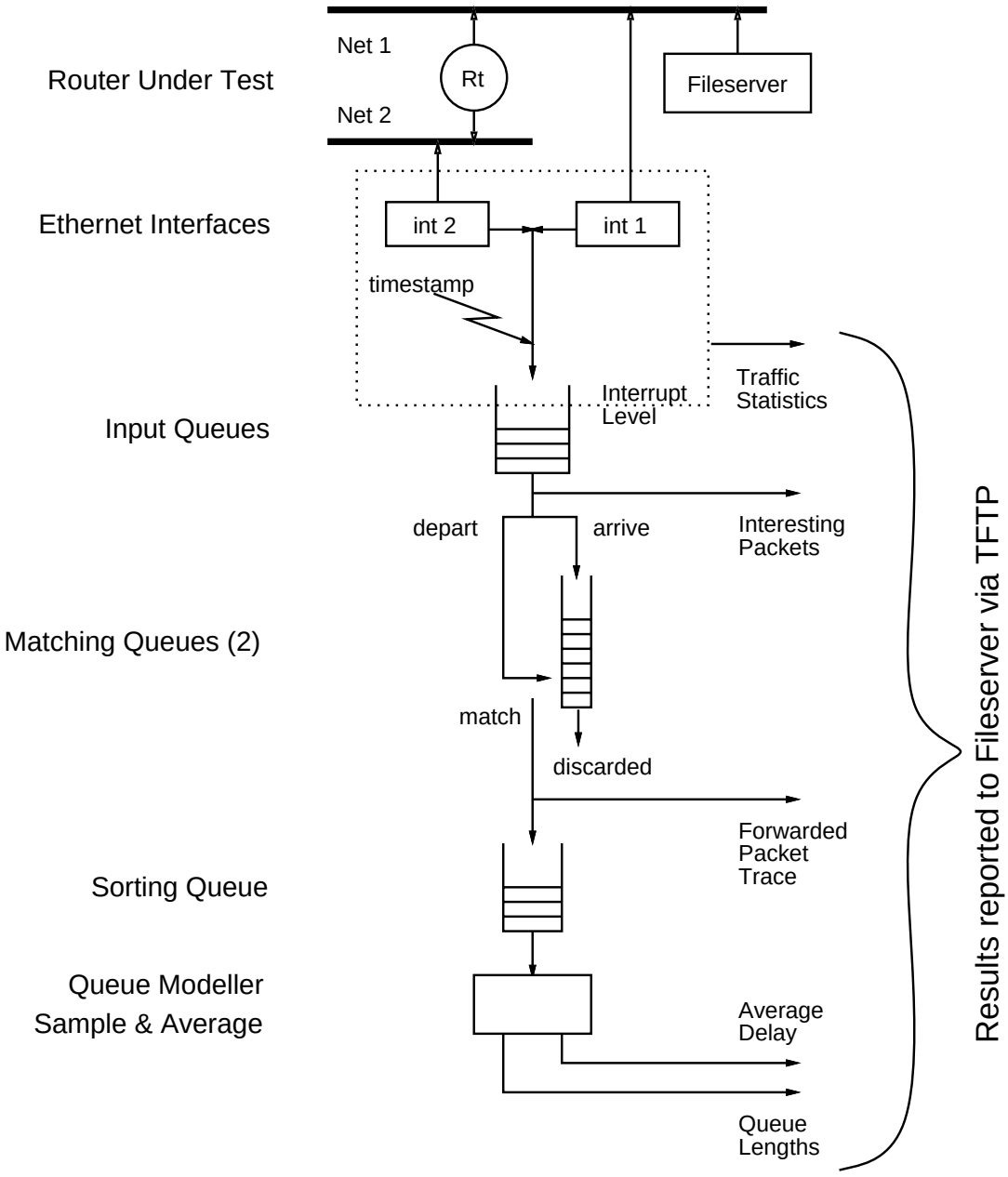

Figure 2.5: Router Measurement Setup

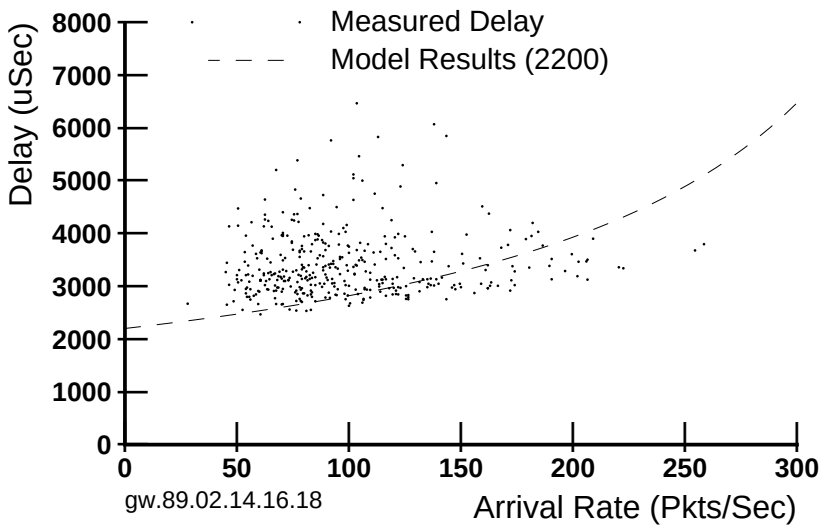

Figure 2.6: Delay Measurements — 2 Sec Samples

There is a fairly distinct two-tail pattern, with one tail exhibiting large delay for even small arrival rates, and the other tail showing relatively low delay, even at high arrival rates. Both of these tails are at variance with the model, in opposite directions. About the best thing that can be said about the model's prediction is that its line passes through the cloud of points — and then, not even through the middle.

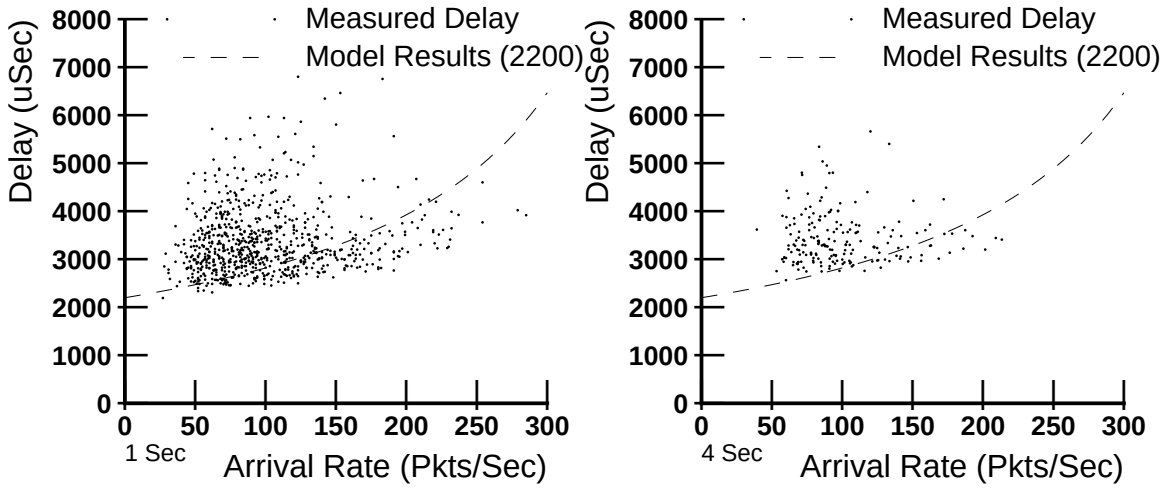

Figure 2.7: Delay Measurements — 1 and 4 Sec Samples

Changing the sample size has little effect on the general shape of the cloud. Figure 2.7

shows the results of changing the sample size to 1 and 4 seconds. With 1 second samples, the variations are more extreme. The opposite effect is observed for 4 second samples. These are the expected results when changing the sample size.

Both the too-high and too-low variations are of interest, for different reasons. For the first tail, the measured delay is as much as two to four times the delay predicted by the model. If the delay was caused by underestimating the service demand, the points should still fall on a line that resemble the model prediction. Instead, they form a diffuse cloud. This suggests either that something else was going on during those intervals to cause the extra delay, or that there is a lot of inherent variation in packet delay. This hypothesis can be tested by looking for other work that the router might be doing by looking for specific indicators of other activity (such as routing packets) and by finding bounds on how much time the router might be spending doing other things.

For the second tail, the only explanation suggested by the model is that actual packets get in each other's way less often than the assumed Poisson distribution would cause. This hypothesis can be tested by comparing the actual arrival pattern against a Poisson distribution.

2.5 Reconciling the Measurements and the Model

The preceding section outlined the general approaches used to reconcile the measurements and the model's predictions. I will now describe the methods used to test the hypotheses. The contributors to delay that are not included in the model (scheduling delay and Ethernet contention delay) were estimated based upon traffic statistics and information about the router's other activities. To test for inherent variability in packet delay and the effect of packet size on delay, I set up a test router and sent artificially-generated traffic through it. The arrival time distribution, which determines the amount of queuing delay through the router, was directly compared against a Poisson distribution. Finally, the analytic solution of the model (from section 2.3.1) was replaced by a simulation based upon arrival times and fed by the packet trace, completely eliminating the arrival distribution assumption.

2.5.1 Scheduling Delay

Scheduling delay is not modeled at all in the queuing network model solved in section 2.3.1 and represents a potentially large source of extra delay if there is a significant fraction of non-routing activity in the router. The router in question does priority scheduling, with routing as the highest-priority task. Scheduling is *not* preemptive, however, so it is important that lower-priority tasks not hog the CPU. The effect of scheduling delay is dependent upon two things: the probability that an incoming packet will arrive when the CPU is busy with another task, and the length of time the CPU is tied up with the pre-existing task.

The best-case situation would be if no incoming packet ever arrives when another task is running, which is what was modeled. The worst-case situation would be if incoming packets arrive exactly when the CPU starts some other task, forcing them to wait for the tasks to finish. This can be modeled by lumping all of the low-priority tasks' service demands together and adding them to the known packet forwarding service demands.

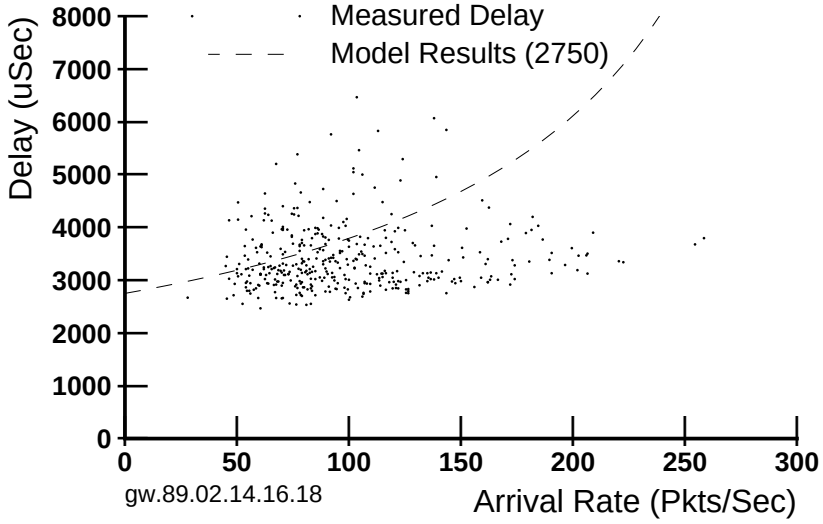

Figure 2.8: Delay vs. Offered Load, Worst Case

In the situation measured, 80% of the router's activity was devoted to forwarding packets and most of the other 20% was spent processing routing table updates. Modeling the worst-case situation discussed above would increase the service demand by 25%, to

2750 *u*Sec. The effects of this are shown in Figure 2.8. Although the line now passes close to the center of the cloud of points, it doesn't come anywhere close to predicting the high delays seen near 100 packets/second and overestimates delay by a factor of two at high loads.

This worst-case scenario is also very unrealistic. The biggest, longest-running task after routing is routing table update, which is invoked about 70 times per minute (due to about 31 routers broadcasting updates every 30 seconds), evenly distributed, running for an average of 10 *m*Sec per invocation. Thus, this task runs for about 12 *m*Sec per second, giving a mere 1.2% chance that a particular packet will get lucky, compared to the 100% assumption in the worst-case scenario analyzed above.

2.5.2 Ethernet Contention

The queuing network model does not include the time spent transmitting the packet. This time may include collisions, exponential backoff delays, and retransmissions. The amount of time spent waiting to transmit is dependent upon the traffic on the Ethernet. For loads of less than 50%, the average delay is less than 300 *u*Sec.[AL79, page 76] The peak load seen in the measurement interval was 37%, averaged over 2 seconds. The average delay attributable to Ethernet contention is slightly over 200 *u*Sec. This does not explain the discrepancies.

2.5.3 Controlled Workload Experiments

One possible source of delay was simply an inherent variability of delay in the router. We investigated this by connecting a router to an empty network and sending a controlled load of packets through it. Several types of load were used:

> Varying packet rate, constant-size packets, but packets well-separated. The router showed constant delay, except at very high rates, when the packet source's inherent

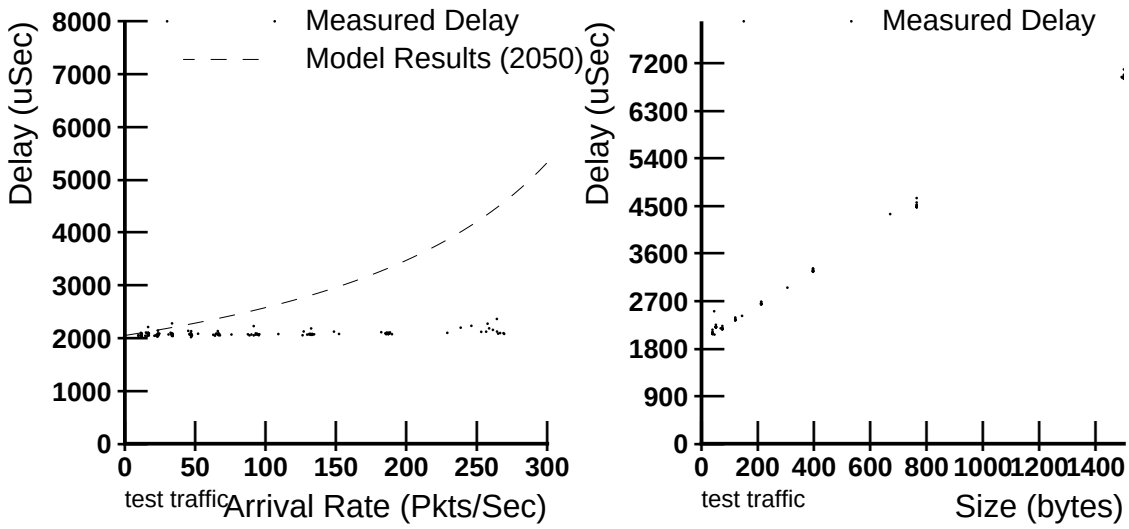

Figure 2.9: Delay vs. Rate and Size for Isolated Router

variation caused some packets to be transmitted too close together. See figure 2.9.

Variable-sized bursts of constant-size packets. Packets were delayed in proportion to the size of the burst. The measurements were not particularly accurate for larger bursts, because the measurement setup lost some of the later back-to-back packets.

Variable-size packets. Packets were delayed an additional 3.4 *u*Sec per byte of data. This will produce a difference in delay of 4.8 *m*Sec between minimum (64 byte) and maximum (1500 byte) size packets. See figure 2.9.

These results were generally as expected. Unaccounted-for variations in delay were very small — on the order of the resolution of the test setup (250 *u*Sec). One unexpected result was that, at high packet rates (250 Pkts/Sec), every one-tenth second, a packet would be delayed by a few milliseconds, as shown in figure 2.10. Occasionally, this delay was great enough (*e.g.* near 28.3 seconds) to delay several packets.

The packet-size contribution to delay, although expected, was surprisingly large. The magnitude of the delay was comparable to the unexpected delays observed in section 2.4.1. Whether this is, in fact, the explanation is examined in section 2.5.6.

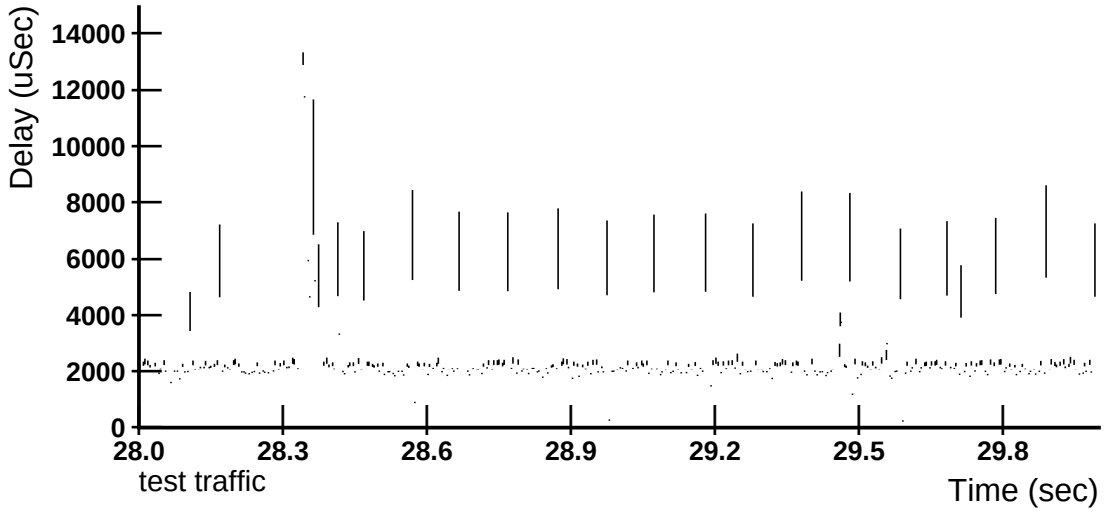

Figure 2.10: Periodic Delays Through Test Router

2.5.4 Arrival Time Distribution

The arrival time distribution determines how likely a packet is to arrive while the router is busy processing a previous packet. A Poisson arrival time distribution is a good model of many natural physical distributions, such as particle densities, and is assumed in the queuing network model analyzed in section 2.3.1. However, it has been well-documented that packet arrival distributions on an Ethernet are *not* Poisson[SH80, Llo86, PDA86], so that must be checked. This study is different from earlier studies, because we are only interested in the traffic arriving at the router, rather than all of the traffic on an Ethernet.

Figure 2.11 shows the arrival distributions for the sample trace, sampled with 10 and 100 *m*Sec bins. Although the distribution viewed with 10 *m*Sec bins looks very close to Poisson, the distribution with 100 *m*Sec bins is a very poor match. So, just what is it we are looking for, anyway? As stated above, what we are trying to measure is the tendency for packets to get in each other's way at the router. Comparison against a Poisson distribution has several drawbacks:

It requires the (somewhat arbitrary) selection of a bin size. Clearly, this choice affects the outcome of the Poisson test.

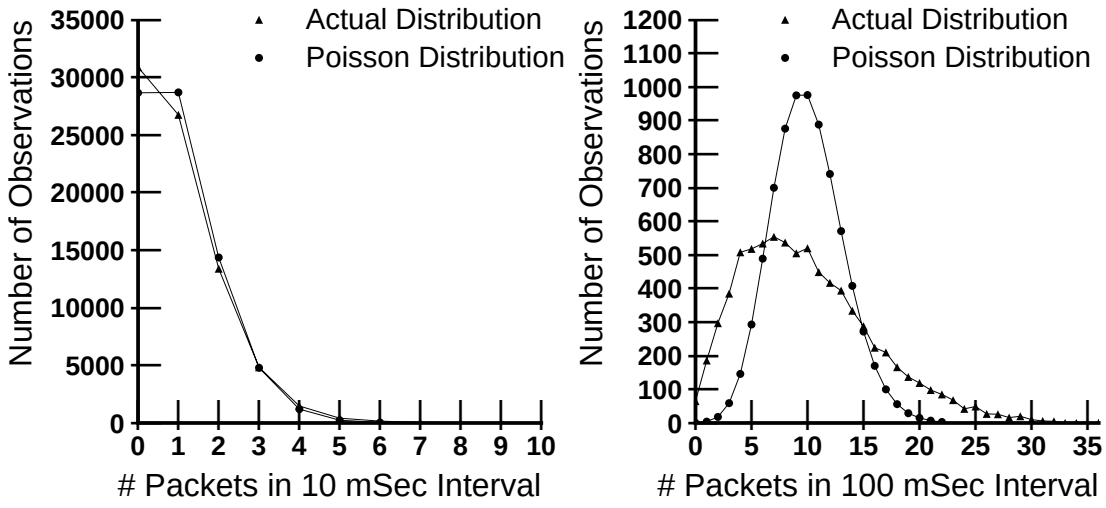

Figure 2.11: Arrival Time Distributions

The result, either a pair of graphs or a chi-square confidence level, doesn't really say anything at all about what to expect from the model. It does give a rough feel for the accuracy of the model's assumption, but says nothing in terms of how the model's prediction will relate to the real world.

Doing the test is non-trivial. A program must divide time into bins, examine each arrival, and count up bins. Although this is only an operation, it is more complicated than just adding up numbers.

Given these difficulties, I avoided the problem altogether by simulating the queuing network model and driving it with an actual arrival distribution. The simulation is quite simple, and consists of keeping track of when the CPU will next be idle, and adding the estimated service demand to that whenever a new packet arrives, yielding the exit time of that packet. The algorithm is given in Figure 2.12. It is very simple — much simpler than testing the arrival distribution against a Poisson distribution. The results are shown in Figure 2.13.

The simulation shows that the traffic would experience significantly less queuing delay than the queuing network model (with Poisson assumption) predicts at most loads. At

```
NextIdle = 0                                    Initialize
TotalDelay = Packets = 0
foreach Pkt do
        if (NextIdle     Pkt.Arrive)            We were idle
                NextIdle = Pkt.Arrive
        NextIdle = NextIdle + ServiceDemand     When this packet will finish
        Delay = NextIdle - Pkt.Arrive           Delay for this packet
        Packets = Packets + 1                   Count packets
        TotalDelay = TotalDelay + Delay         and total delay
end
AverageDelay = TotalDelay / Packets             The result
RelativeDelay = AverageDelay / ServiceDemand    1.0 = no queuing
```

Figure 2.12: Simulation Algorithm

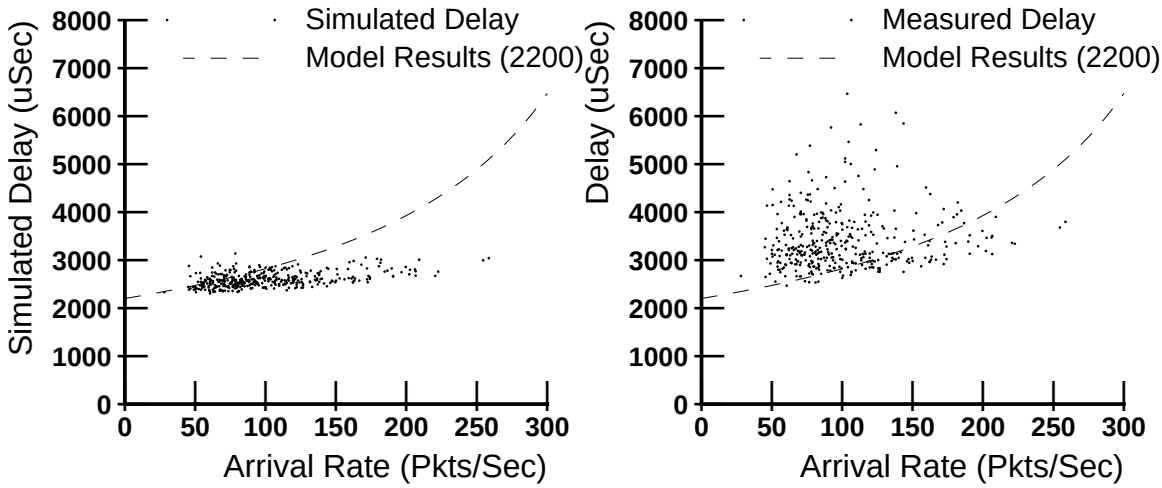

Figure 2.13: Simulated and Actual Delay vs. Arrival Rate

low loads, some samples show larger delays than predicted by the model, but not nearly enough to explain the observed delays.

At high loads, the lack of interference between packets is striking — delay due to queuing is 800 *u*Sec in the simulation vs. 2500 *u*Sec predicted by the queuing network model using the Poisson assumption. This is certainly the reason that the observed delay at high loads is so low. Judging by its effects on queuing delay, the arrival distribution is

clearly not Poisson, but it is not Poisson in a "nice" way — queuing delays are lower than expected.

2.5.5 Using Simulation to Measure Burstiness

A side benefit of doing this simulation is that it gives a way to measure a particular traffic load and get an idea of how bursty it is. Others have tried to measure *burstiness*[Llo86, page 85], based upon an intuitive notion of what "bursty" means. The intuitive notion of burstiness is not what we need to measure. We need a number that characterizes a traffic stream that can be used to predict the effect of presenting that traffic to a real system. The characterization can't be done in a vacuum — it doesn't make any sense to define a measure of burstiness outside of the context of the system "detecting" the burstiness.

For example, assume traffic that tends to have bursts of high activity (500 pkts/sec) against a typical background of 50 pkts/sec. A human observer would probably say that this traffic is quite bursty. On the other hand, a router that needs 10 *u*Sec to process each packet won't even notice the difference because it's only loaded between 0.5% and 0.05% — the "burstiness" is irrelevant. A router that needs 1 *m*Sec will probably notice a lot of interference during the bursts, but be only 5% loaded otherwise. A 5 *m*Sec router, however, will be saturated and probably drop packets during the bursts. So, is this assumed traffic bursty?[2]

To be relevant, the test must take into account the speed of the system that's expected to handle the load. The test I propose here is very simple: model this hypothetical system as an open queuing network model with one service center and feed it the arrival trace to be tested for burstiness. The service demand corresponds to the interarrival time that is considered "bursty". If packets arrive closer together than this, they will experience queuing and delay, and the average delay will increase. Packets that are spread out more will be delayed by exactly the service demand, with no queuing delay. The ratio between

[2]Readers who are familiar with digital logic design may recognize a similarity to designing a logic circuit to "de-bounce" switches, based on timing alone. A similar decision has to be made to differentiate mechanical noise from a rapid key press.

the simulated delay and the service demand is the measure of the burstiness of the traffic, *relative to the given service demand.*

2.5.6 Incorporating Packet Size Into the Model and Simulation

In section 2.5.3 we found that router delay was dependent upon packet size. It takes 3.4 *u*Sec for each additional byte of data in a packet, causing variations of up to 4.8 *m*Sec between minimum- and maximum-size packets. This contribution was incorporated in the simulation of section 2.5.4 by increasing by the size-dependent delay. With this modification, each packet's service demand incorporates both the fixed (average) costs of processing and the variable (size-dependent) costs. The actual values used for this simulation are 2200 *u*Sec and 3.4 *u*Sec, respectively.

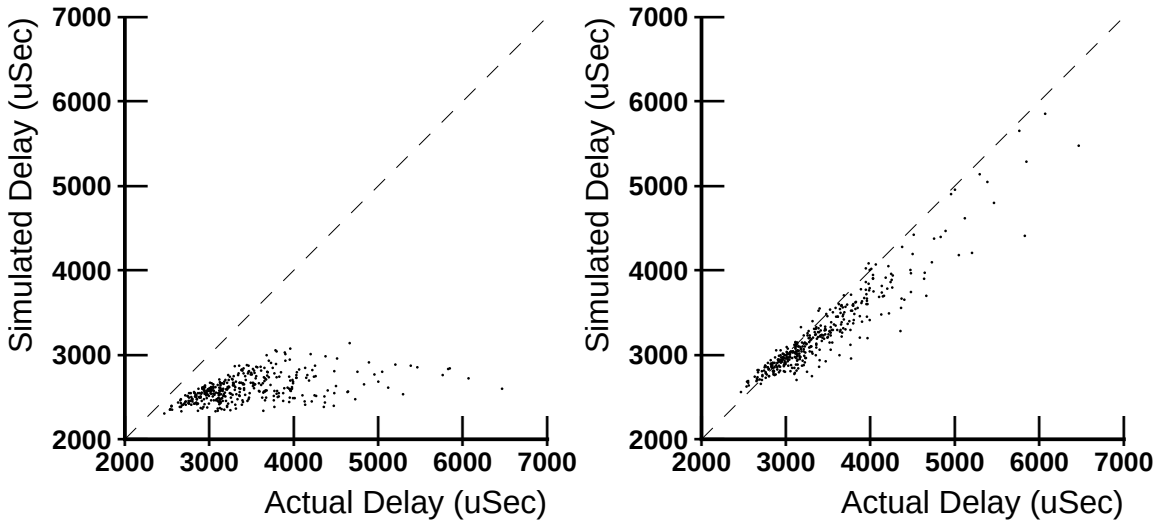

Figure 2.14: Original and Modified Simulation vs. Actual Delay

Figure 2.14 shows the original and modified simulation results, plotted against the actual average delays. The improvement is dramatic, with the modified simulation and actual delay agreeing with a correlation of 0.95. Most predictions are within 10%, with a few as far off as 30%. These results could probably be improved by tweaking the parameters, but that is not the point here. The goal is to determine what factors contribute to delay

through the router, and how they can be used to predict actual performance.

By incorporating both interarrival time and packet size in the simulation, we have achieved a good match to the measured delay. Two questions remain:

Why does the size dependency have such an impact?

Can these findings be incorporated into the queuing network model to produce a more accurate model? Such a model would be much easier to evaluate than the simulation.

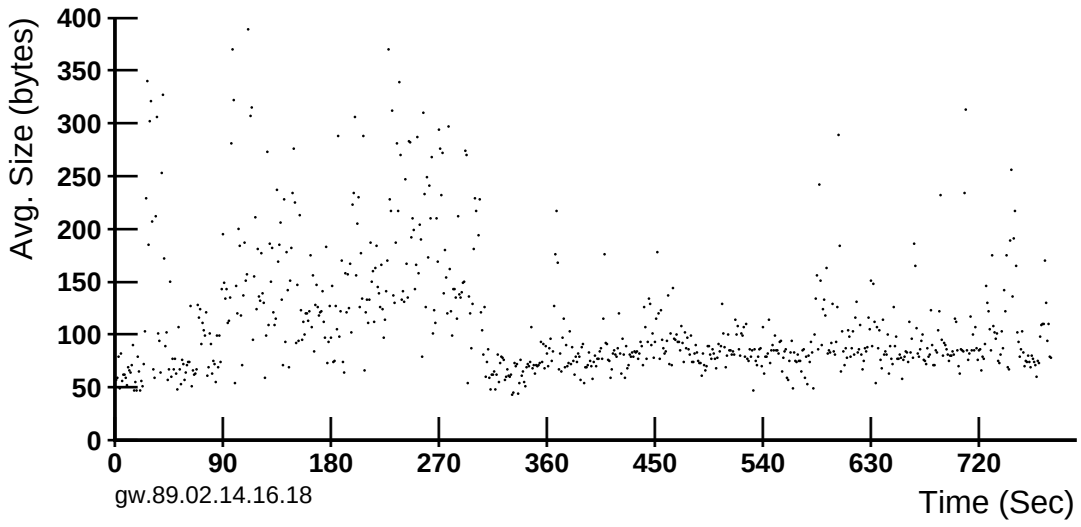

Figure 2.15: Average Packet Size vs. Time

Figure 2.15 shows the variation of IP packet size (not including the 14-byte Ethernet header) with time, with one-second samples. Even with 50 to 200 packets in each sample, there is a large variation in average size. Some variations are sustained, such as between 90 and 300 seconds, while some are short, such as near 40 or 600 seconds. But even these “bursts” of large packets include samples that are near the baseline of 70-100 bytes.

So, the large delays are a direct consequence of varying packet size. Given this, it makes sense to try to modify the queuing network model to take average packet size

into account. This modification is the same as for the simulation — for each sample, the service demand is computed from the baseline service demand (2200 *u*Sec) plus the size-dependent service demand (3.4 *u*Sec per byte). The difference here is that "size" is the average size over the sample rather than the size of one packet.

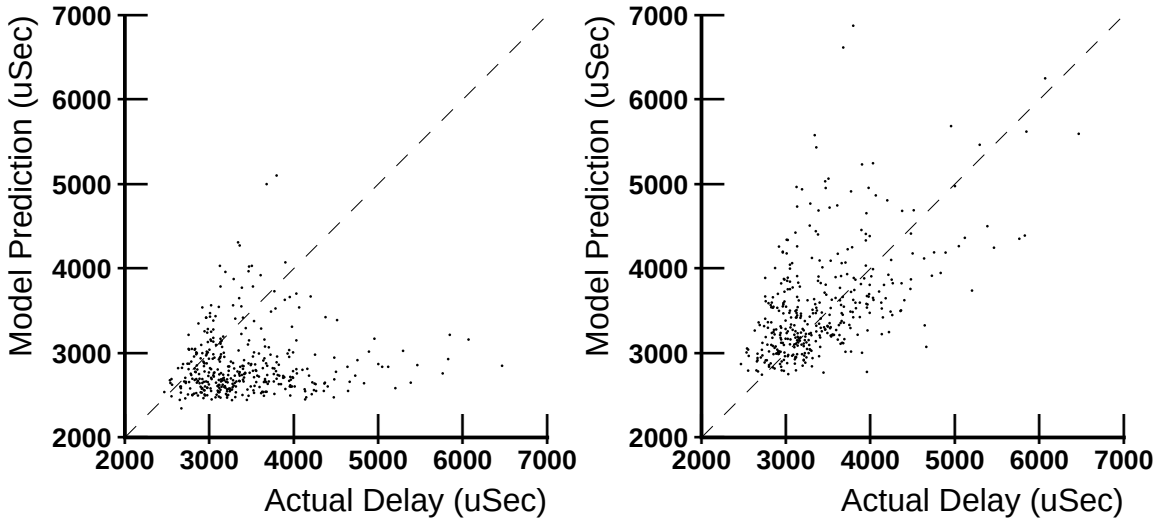

Figure 2.16: Original and Modified Model Prediction vs. Actual Delay

Figure 2.16 shows the results of the original model and this modified model, compared against the actual delay. As observed before, the original model had fairly poor predictive powers. The modified model isn't much better — the major effect has been to move the cloud so that the deviations are more centered. Clearly, the actual arrival pattern *must* be taken into account to get good results. The simple queuing network model solution doesn't do this, and so gives poor results. The modified simulation (section 2.5.6) uses both size and arrival pattern, and gives good results.

2.6 Results for "Fast" Router

The "fast" router has improved hardware and software over the "slow" router, which has been discussed up to now. The significant differences, as well as their expected impact, are:

Faster CPU. The CPU is a 24 MHz 68020, rather than a 10 MHz 68000. Most aspects of performance should improve by about a factor of two.

Improved Ethernet interfaces. The Ethernet interfaces are faster and are less likely to drop packets. In addition, pairs of interfaces share packet buffer memory, so packets routed between interfaces on the same card don't need to be copied over the relatively slow bus. This should reduce packet copying time to zero for those packets.

Routing Cache. A cache that maps destination IP addresses into Ethernet addresses is used to route packets at interrupt time. This isn't a cache in the usual sense of the term — it is allowed to grow very large. Entries are removed only when the routing table changes, but not because the table gets full. In most cases, this should eliminate scheduling delay and the cost of a full-blown routing table lookup.

We ran two sets of measurements, one with the routing cache disabled and one with it enabled, to get an idea of how much the cache improves performance, independent of other differences.

2.6.1 Results with Cache Enabled

The fast router with the routing cache enabled is much faster than the slow router — less than 250 *u*Sec for most packets vs. 2200 *u*Sec. Because the delay is reduced by nearly a factor of 10, but the load is the same, the fast router is much less busy. Packets are now very unlikely to interfere with each other. This expectation is validated by the measurements.

One significant problem in these measurements is that the best resolution available from the measurement set-up is 270 *u*Sec between packet arrivals at the monitoring station. At least 55% of the packets observed had delays reported at or close to this minimum. The remaining packets' delays were large enough to rule out the possibility that they, too,

were smaller than measurable. Given this, it is clear that the minimum delay through this router is less than 270 *u*Sec. The manufacturer claims 100 *u*Sec delays for this router.

We have no way of compensating for this error, or even determining exactly how much the averages are affected by it. We can, however, get a reasonable bound on the error: If we believe that the manufacturer's claim of 100 *u*Sec is a lower bound on delay, then the maximum error for a packet is 270 100 170 *u*Sec. Since about 55% of the packets appear to be subject to this error, the average error in a sample will be less than 94 *u*Sec. This calculation makes the worst-case assumption that all packets whose delay was measured as 270 *u*Sec actually had a delay of 100 *u*Sec. This error bound should be kept in mind in the remainder of this section. To model this error, we simply set the service demand to 270 *u*Sec. This will have disastrous effects if the system actually has any queuing. As it turns out, this isn't a problem — there is essentially no queuing in the system at the low utilizations observed.

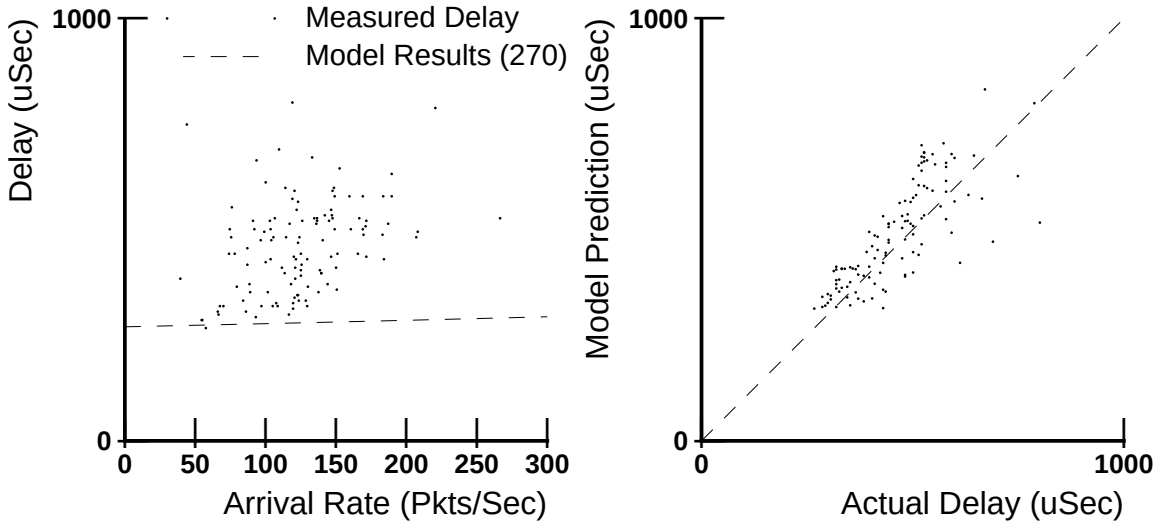

Figure 2.17: Simple Model and Modified Model Results

Figure 2.17 shows, on the left, the queuing network model prediction for the fast router, along with the average measured delay for a traffic sample. At the packet rates observed, there is no queuing, so the curve for predicted delay is flat. The measured delay disagrees with the model in a similar fashion as in Section 2.4.1. The reason for the disagreement

is also the same: variation in average packet size. Even though there is no packet copying within the router, it takes 0.8 *u*Sec per byte to receive (or transmit) a packet at 10 MBit/Sec.

In section 2.5.6, we found that it was necessary to simulate the entire packet trace, incorporating both packet size and arrival pattern, to get results that matched the measurements. In this case, it is sufficient to modify the model to incorporate packet size; the arrival pattern is not a significant factor. The right side of Figure 2.17 shows the correlation between the modified model results and the measured delay. These results are good, but only due to the fact that this is a trivial case, with little or no queuing.

2.6.2 Results with Cache Disabled

We disabled the routing cache on a fast router to obtain an intermediate measurement between the fast router with cache and the slow router. Our expectation had been that disabling the routing cache would eliminate the speed gain due to routing at interrupt time, and conserve the advantages due to CPU speed and lack of packet copying. From our measurements, however, it appears that packets are copied across the bus when the cache is disabled. The only speedup seen is due to the increased CPU speed.

Measurement set-up speed was not a problem in this case — only one percent of the observed packets had delays near the threshold of measurability. This is consistent with our estimate of the service demand — 1100 *u*Sec.

Figure 2.18 shows, on the left, the queuing network model prediction, along with the average measured delay for a traffic sample. Except for the changed parameters, these results are very similar to those reported for the slow router and the fast router with the route cache enabled. The service demand used is 1100 *u*Sec(about half of that found for the slow router) and the time to copy one byte is 2.6 *u*Sec (about halfway between the figure for the slow router and the fast router with caching). These intermediate values reflect the speedup due to the fast CPU, only.

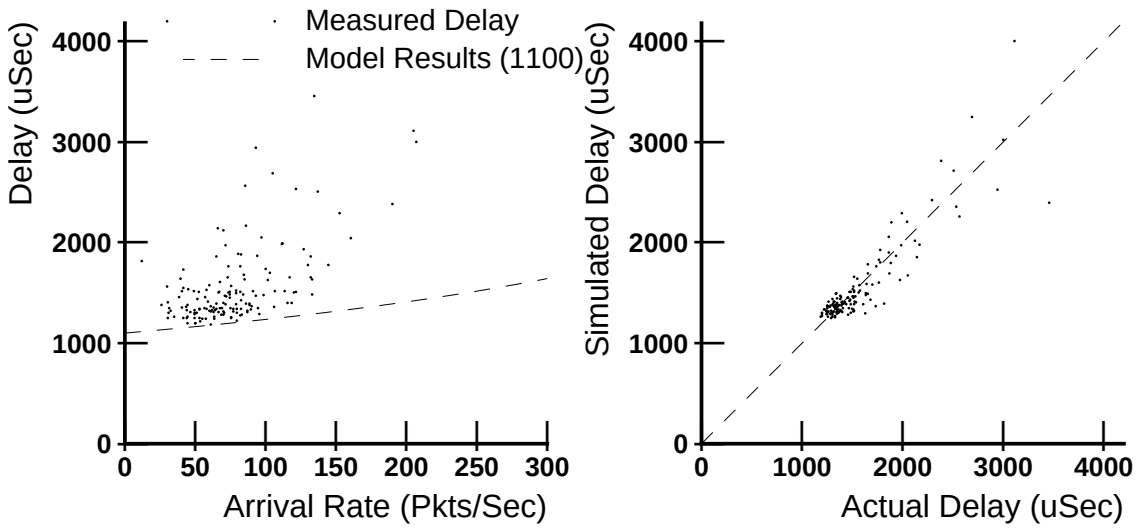

Figure 2.18: Simple Model and Modified Simulation Results

Because the service demand is so much larger than with the routing cache enabled, the utilization is higher and there is more queuing. So, as found for the slow router (Section 2.5.6), the modified simulation (not the queuing network model) that incorporates packet copying time must be used to accurately predict router delay. The right side of Figure 2.18 shows the correlation between the simulation results and the measured delay.

2.7 Conclusions

Table 2.4 summarizes the router characteristics and model inputs for each of the three situations studied. For each of these routers, the simulation produced accurate results using only the service demand and packet copying overhead as inputs.

Router delay is the most significant component of end-to-end delay in local-area internetworks. Of the factors affecting router delay, we found that the interarrival time distribution and size-dependency of delay were the key items that caused variation in router delay. The interarrival time distribution was (as expected) not Poisson. However, the non-Poisson nature of the distribution caused *less* queuing delay, rather than more,

	Slow Router	Fast Router no cache	Fast Router with cache
Processor	10 MHz 68000	24 MHz 68020	24 MHz 68020
Routing Scheduling	via process	via process	at interrupt time
Routing Table	table lookup	table lookup	routing cache
Packet Copying?	Yes	Yes	No
Baseline Service Demand	2200 *u*Sec	1100 *u*Sec	270 *u*Sec
Packet Copying Overhead	3.4 *u*Sec/byte	2.6 *u*Sec/byte	0.8 *u*Sec/byte

Table 2.4: Summary of Router Analysis

leading to substantially less delay than predicted by a Poisson-based model. Variations in average packet size were quite large, even for large samples. These variations were amplified by the extra time taken to process large packets, presumably because of packet copying inside the router, leading to router delay variations over a 3:1 range. Router architectures that eliminate internal packet copying offer much improvement here.

The queuing network model's simple solution proved to be an unacceptable tool for predicting any of this behavior. Even including the size-dependent service demand in the model did not significantly improve accuracy. Going back to the basic model and using a simple simulation of router activity produced much more accurate results, at essentially the same cost as accumulating statistics about the traffic.

Router delay is large enough that the performance of the local-area network is a non-issue. For the routers we studied, contention delay (time required to access the Ethernet) was an insignificant fraction of router delay.

In all cases, we found that there was non-trivial variance in router delay. Protocol implementors should take this into account when designing retransmission strategies. Even though many stream protocols (*e.g.* TCP) do this properly, datagram-based protocols (network file systems or name servers) may need to be concerned about delay variance.

2.8 Future Measurement Work

More work needs to be done in several areas. The test set-up, which is based on the same hardware as the "slow" routers, is just barely fast enough to keep up at high packet rates and really isn't capable of accurately timing packets routed through the "fast" router. Fortunately, fast Ethernet interfaces with on-board processors and large buffer memories should be more than capable of handling this load. The greatest difficulty, in fact, was obtaining an accurate, high-resolution clock for timestamping packets. Adding a high-resolution (48- or 64-bit) hardware clock would improve both speed and accuracy.

It would be interesting to know what it is about the traffic that causes it to be less bursty than Poisson. One possible explanation is that, if the traffic through the router coming from the backbone (about half of the traffic) comes from only a few other routers, than those routers will have conditioned the traffic by their own maximum packet forwarding rate. That is, packets from a similar router will arrive at this router at exactly its processing rate. Packet trains[JR85] have been suggested as a possible model of network traffic. This (hypothesized) conditioning effect of upstream routers would tend to group otherwise unrelated packets from the same subnet together into trains. Furthermore, because of the spacing provided by upstream routers, *these* trains are especially suited to go through a router with little or no queuing delay.

The simulation code built into the measurement set-up produced useful data and reduced the amount of data to be transferred. However, packet traces were absolutely essential during the development of the model so that they could be analyzed in detail using a variety of methods. At the packet rates encountered, the data rate required for transferring the trace back to the workstation was not large enough to affect the measurements. This may not always be true, so local disk storage or a very large memory buffer would be another useful addition.

High-level tools were essential for interpreting the data. We used an array of data-extraction tools, scripts, and plotting templates for this, most of them home-made. An integrated system that would allow database-like extraction of data, custom manipulation,

and graphic presentation would be a big help for speedy data interpretation and testing of theories. Such tools are becoming available for personal computers, but they are not yet capable of handling the large volume of data (data sets larger than one megabyte) and do not offer flexible enough data manipulation capabilities. Most tools offer the ability to create derived variables, but only within a single case. That is, one could create a derived variable for network utilization from variables for packet rate and byte rate *for each individual sample*. What is lacking in most data analysis tools is a more general capability that would allow variables from other cases to be used as well. With this, one could compute interarrival times by subtracting the timestamps of the current and previous packet. The queuing simulation could be done easily by creating a derived variable for the next idle time. Spreadsheets have this computational capability, but don't offer the data analysis and presentation capabilities of statistics and data visualization tools. As these tools mature, they should be more suitable for this type of application.

Chapter 3

Internet Topology

The topology of an internetwork — how the component networks and routers are interconnected — is an important factor in determining how well it will perform when there are failures in its parts. Analysis of the topology can point out critical components or paths, suggesting modifications to improve overall reliability. Proposed changes can also be investigated before implementation, providing a useful planning tool.

The topology is also a basis for comparison of internetworks. Comparing the same internet at different times demonstrates the impact of systemic changes. By comparing different internetworks, implementors can learn from experience gained elsewhere.

We are primarily concerned with local-area internetworks, because they often have a richer, more dynamic topology than relatively static wide-area internetworks[Arm88]. The methods outlined here apply equally well to wide-area internets, however. For any given internetwork topology, it is possible to do an analysis by hand that produces the same results we describe here[MKU87, pages 8-9]. Because local-area internet topologies change relatively frequently (see section 3.6), such manual analyses would be very tedious and time-consuming to do on a regular basis.

Our goal is to replace the expert intuition required for *ad hoc* analysis with precise definitions of the important characteristics of an internetwork topology. Working from

these definitions, we have built a set of tools to do topological analysis and coupled them with automated network monitoring tools that extract the topology from the running network. This automation has been increasingly important as the Stanford internet has grown to the point where no one person knows all of its details[Arm88, pages 151-156].

One of the most interesting characteristics of an internetwork's topology is its resistance to partitioning by failures. We refer to this characteristic as *robustness*. Robustness is usually achieved by adding *redundant* components to an internet, usually both routers and communications links. We first define several basic topological measures and use them to quantify the amount of redundancy in a topology. We then define techniques for measuring robustness that indicate how resistant a topology is to being partitioned by a failure.

Finally, these techniques are used to analyze changes made to the Stanford University Network over a period of 21 months. We find that there have been significant changes in the robustness, and that some very small topological changes have had a large effect on the robustness of the internetwork.

3.0.1 Common Sense and Assumptions

The analytical tools described here operate on abstract models of internetwork topologies. Real topologies are usually constrained by technical limits, such as limits on the size of a network or number of interfaces in a router, by practical limits, such as restrictions on how cables may be run between or within buildings, and economic considerations. As with any tool, common sense must be applied as well.

It is assumed that the routing algorithms used by the routers or bridges can detect changes and recover in a timely fashion (before connections time out). This assumption is *false* for some implementations of some algorithms and, for those systems, failures that would otherwise go unnoticed may be felt as a temporary loss of connectivity, severe enough to disrupt some connections [BH88].

3.1 Measures of a Topology

There are many different ways to measure topologies for comparison. Numeric *metrics* are especially useful. The most obvious metrics are the bulk sizes:

Networks. The total number of component networks. This a good measure of the size of the internet, since it bears some relation to the number of hosts that may be connected to the internet, as well as the geographical span of the internet. In local-area networks, there is usually some limit on the physical size of a network, as well on how many hosts may be connected to one network. The number of networks usually increases when the internet expands to an additional physical location, or when an existing network grows too large and must be partitioned into separate networks.

Routers. The total number of routers in the internetwork.

Interfaces. The total number of router interfaces, represented in the graph as the number of edges between routers and networks.

We will refer to these metrics as , , and , where is a topology. When it is understood what topology is being referred to, these will be abbreviated as , , and .

Some related metrics that are related to the redundancy in an internet are the ratios of the above quantities:

Interfaces/Router, . The ratio of the number of edges in the graph to the number of routers. This is the average number of interfaces per router. The lower bound on this ratio is two, since a router must have at least two interfaces.

Interfaces/Network, . The ratio of the number of edges in the graph to the number of networks. This is the average number of routers connected to each network. For the boundary case of one network and no routers, this ratio doesn't exist. Other

than that, the lower bound is one, since each network must be connected to a router in order to communicate with the rest of the internet.

Based upon observation, this ratio is usually close to one. This is because the main goal of an internetwork is to attach leaf networks, each of which is usually connected by only one router. Each leaf network contributes one interface to the total, pushing the ratio towards one.

represents the average spreading factor at routers, while represents the average spreading factor at networks. For both of these, larger ratios for similar internetworks indicate increased redundancy. But they do not give a complete picture. It is possible to create a topology with arbitrarily large values for *either* ratio, without increasing redundancy (*i.e.* the internetwork can still have a tree structure).

Attempting to increase both and , however, will increase redundancy, because the added edges will tend to make the graph fold back in on itself. We make a more precise characterization of the behavior of these ratios in the next section.

3.2 Internetworks With No Redundancy

Any internet can be represented by a fully-connected, undirected graph, with no duplicate edges. An internet with no redundancy will also be acyclic — there will be exactly one path between any two nodes. For an acyclic graph, there is a simple relation between the number of nodes and the number of edges: 1.

This relation is easily demonstrated by repeatedly removing one leaf node along with the edge that connects it to the rest of the graph. After all leaves have been removed, one node (with no connected edge) will be left. An example of such a decomposition is given in Figure 3.1.

As a result of this decomposition, each router or network is paired with one interface, with one network or router left over. In terms of our metrics, the equation becomes

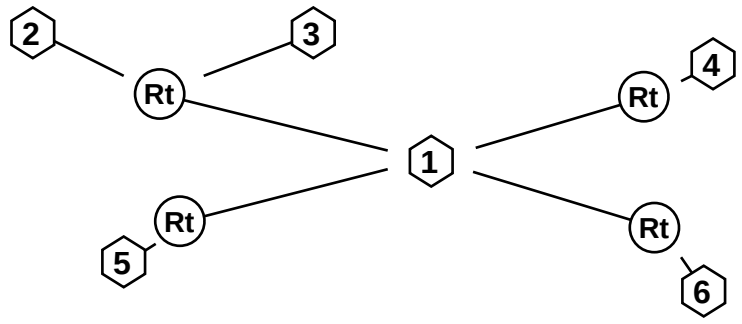

Figure 3.1: Decomposition of Example Topology

1.

If the number of interfaces in a non-redundant internet is fixed, then each choice of implies a particular . Since the ratios and are defined in terms of these three numbers, there must be a fixed relation between the two ratios, for a particular value of . This relation can be used to identify the region that can lie in, which turns out to be quite small. The derivation of this relation follows:

$$1$$

$$1$$

$$\frac{}{\frac{}{1}}$$

$$\frac{}{\frac{}{1}}$$

$$\frac{}{}$$

$$1$$

$$\frac{}{1} \qquad \frac{}{}$$

For a constant number of interfaces in an internet, this equation prescribes an inverse relation between and . Figure 3.2 shows this curve for equal to 2, 4, 8, 16 and the asymptote as .

Each of the dashed curves corresponds an internet with a fixed number of interfaces.

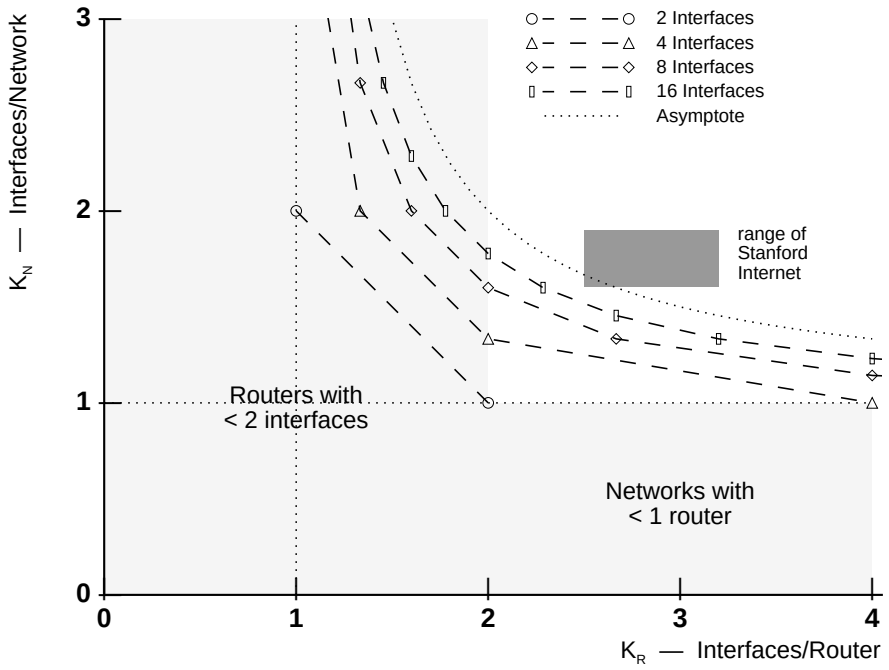

Figure 3.2: vs. for Varying Interfaces

Different possible combinations of networks and routers (adding up to 1) are represented by each triangle. For real internets, will be at least 2, corresponding to the right-hand portion of the graph. For example, for 4, it is possible to have a network with combinations of 4 1 (not visible), 3 2 , 2 3 , and 1 4 . The first two, however, represent internets with 2. That part of the graph is shown shaded and will not be considered any further.

The dotted line is the limit of this family of curves as the number of interfaces goes to infinity (—1 1). Measurements of ( ) for the Stanford internetwork all lie inside the small dark gray rectangle, which is enlarged in Figure 3.10.

For internetworks with no redundant paths, ( ) will lie exactly on the curve corresponding to the number of interfaces in the internet. If some interfaces are missing (a partitioned internet), the point will lie below the curve. If there are additional interfaces beyond the minimum required for full connectivity, the point will lie above the curve. The greater the redundancy, the farther from the curve the point will be.

To measure the redundancy, we define the function to be:

$$\frac{\quad}{\quad}\ \frac{1}{\quad}$$

Values of equal to 1 lie on the curve, indicating no redundancy, and values greater than 1 lie above the curve, with larger values of indicating more redundancy. The exact location of indicates whether internetwork "fan-out' happens mostly at routers or networks, and is indicated clearly by the location of on the graph.

can be computed directly in terms of , , and , by substituting the definitions for and :

$$\frac{\quad}{\quad}\ \frac{1}{\quad}$$

$$\frac{\quad}{\quad}\ \frac{1}{\quad}$$

$$\frac{1}{\quad}$$

This formula provides another interpretation of : it is (essentially) the number of interfaces present in an internetwork divided by the number required to minimally connect all of the routers and networks in the internet.

captures some information about the redundancy present in an internetwork, and is very simple to compute, but it doesn't unambiguously indicate how well the internet will function when a failure occurs. Consider the case of the internetwork shown in figure 3.3(a) and the two possible improvements in (b) and (c). Both consist of adding one router and two interfaces to the original internet. Both produce the same improvement in , raising it from 1 to 1.1. However, adding the router between two nearby networks only protects against the failure of two routers or their interfaces. Adding the router so that the internet becomes a ring, however, protects against any single failure.

To measure differences in robustness, it is necessary to examine the topology in detail. The next section discusses such measurements, which are based upon how an internet is partitioned by failures.

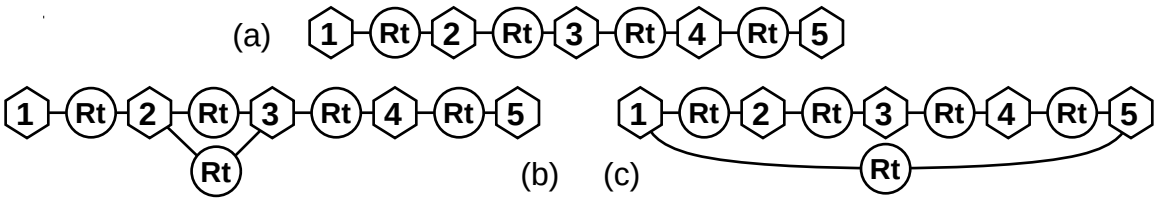

Figure 3.3: Sample Internet and Two Possible Improvements

3.3 Measuring Robustness

To find a way to measure the robustness of a topology, we must first define exactly what we want to measure. The primary purpose of an internetwork is to allow individual networks (and therefore the hosts, users, and processes on them) to communicate. The underlying measure of robustness, then, is the amount of disruption caused by failures. *e.g.* When part of an internet fails, and no one notices, then there is "good" robustness. When some other part fails, and two-thirds of the connections in progress fail, there is "poor" robustness.

In this section, we present a framework for analyzing an internetwork's robustness. The analysis identifies the weak points in an internet and provides an overall robustness measure that may be used to compare different internetworks. We define two practical metrics based on this framework and show ways that the computational demands of the analysis may be reduced. Finally, we extend the framework to handle double faults.

The first step is to quantify the effects of a failure. We use the term "failure" to mean one or more faults in the internetwork. Each fault is represented in the topology by the removal of a single node or edge. The techniques presented here are most easily applied to single faults, but the underlying concepts are also applicable to multiple-fault failures.

We define a *failure metric* as a function that describes how well the internet survives a particular failure. For example, one failure metric might measure the percentage of networks able to communicate with each other. It might have a value of one if all networks could communicate, and zero of none could communicate.

The individual failure metrics can be combined by taking a weighted sum over some set of failures. The resulting *robustness metric* gives an overall measure of the ability of the internetwork to continue to function after a failure. For the above example, the robustness metric will be close to one if most networks can continue to communicate after a failure, and close to zero if failures tend to disconnect a large fraction of the networks.

There are three choices made when defining a robustness metric using the above model:

Failure Metric, . The measure of the effect of the failure . The choice of determines what attributes of network performance are classified as important. If, for example, only measures the percentage of terminal server traffic disrupted by a failure, then other important effects may be ignored. This is an extreme example, but many different metrics are possible, and if there is any difference between the resulting values for the same failure, then they *must* measure different things.

Weights, . Represents the contribution of the corresponding failure's metric value to the overall robustness metric. It might be selected to correspond to the probability of the failure, the expected time to repair, the number of phone calls likely to be triggered by the failure, etc.

Failure Set, . The set of failures considered when computing the robustness metric. This choice may be constrained by limitations of the data available or the cost of computing over the entire set. If devastating failures are left out, or large classes of irrelevant failures are included, then the robustness metric will not provide an accurate picture. Inclusion in is related to the choice of , since exclusion from corresponds to 0.

The choice of these three components makes a statement about what is considered important in the internetwork — how the "correct operation" of the internet is to be judged.

In terms of the above components, a general formulation of a robustness metric is:

There are many possible weightings for robustness metrics, depending upon what is considered important. Here are a few factors that the weighting might incorporate:

Consider what percentage of the traffic in the internet is disrupted by a failure. "Traffic" might be measured in terms of bytes, packets, or connections.

Weight the failures by their probability of occurrence.

Weight the failures by their probable duration.

Weight the disrupted traffic by its importance, however that might be defined.

A potentially serious problem with these metrics is the amount of data that may be needed to compute them. At a minimum, the topology of the internet must be known, in order to judge the effects of a failure. Traffic volume may be available and would probably be fairly accurate. Failure probabilities are, at best, difficult to obtain with any accuracy, and, at worst, are very subjective. Rating the importance of traffic is completely subjective.

We will consider a robustness measure based on one failure metric, the *Partition Fraction*. It is based strictly upon the topology, and weights failures by the number of networks still able to communicate with each other in the presence of a failure.[1] We will also discuss ways to reduce the size of the failure set, without adversely affecting the validity of the robustness metric.

3.3.1 Partition Fraction

The first failure metric is the *Partition Fraction*, , the fraction of network pairs in that can still communicate with each other after a particular failure, has occurred.

[1] Although we have a considerable amount of traffic volume data, we haven't studied a metric that weights failures by the amount of traffic disrupted by the failure because this data only covers about eighty percent of the gateways and none of the bridges. In addition, it turns out to be difficult to infer what fraction of the traffic through a gateway or network could *not* find alternate routes when that component fails — there many be many routes in use through the failed node, and alternate paths may exist only for some of them.

It can be computed as follows: For an internet with networks, each network can communicate with 1 other networks, for $\frac{1}{2}$ pairs (counting each pair once rather than twice). In addition, each network can also communicate with itself. The number of communicating network pairs is therefore

$$\frac{1}{2} \qquad \frac{1}{2}$$

When an internet is partitioned, the number of communicating network pairs is the sum of the numbers for the partitions. So, for a given failure, all that needs to be known to compute this metric is the sizes of the partitions produced by the failure. For the failure , let the number of networks in each fragment be , where identifies the fragment. The partition fraction for this failure is

$$\frac{1}{1}$$

The sum of the is not necessarily equal to N, since the failure may be in one of the networks, in which case that network will not be included in any of the fragments. Therefore, will be slightly less than one if included a failed network, even if was not partitioned.

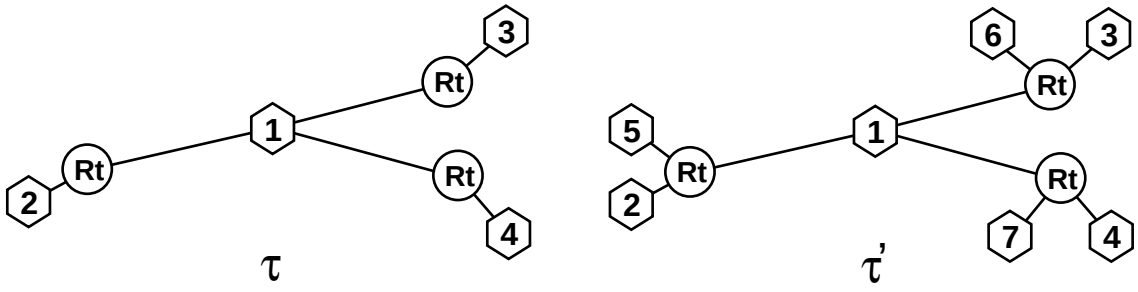

Figure 3.4: Two Sample Topologies

As an example, we will compute for all possible single failures of the topologies shown in Figure 3.4. Both topologies are symmetric around network 1. Therefore, there are five classes of single failures to be analyzed:

The hub network, 1.

Each of the leaf networks.

Each of the routers.

An interface connecting a router to network 1.

An interface connecting a router to a leaf network.

is computed by first determining the set of for the failure, and then applying the formula above. Table 3.1 gives values for the and the resulting for the example topology, , and for a similar topology, , where each gateway connects two leaf networks instead of just one.

Failure				
Network 1	1, 1, 1	.3	2, 2, 2	.321
Any other network	3	.6	6	.750
Any router	1, 3	.7	5, 1, 1	.607
Interface to net 1	1, 3	.7	2, 5	.643
Other interface	1, 3	.7	1, 6	.786

Table 3.1: Analysis of Topology in Figure 1.2

As we have defined it here, the partition fraction only has a boolean concept of connectivity. *i.e.* nodes are either connected or they aren't. It would be useful to extend this model to take into account the available bandwidth between two nodes. In widely dispersed local-area-internetworks, communication links may vary in bandwidth by several orders of magnitude, and some faults may decrease the available bandwidth from, for example, 10 Mb/sec to 56 Kb/sec. We leave this as future work.

The partition fraction measures how well an internetwork will survive failures in its components, using a model of connectivity that has a solid practical basis and is intuitively clear. It is reasonable to compute for all possible single faults in an internet, and from that obtain an average over all single failures. This may be used to directly compare the robustness of different internetworks. Besides providing an overall metric, the analysis can be used to identify faults that have especially serious effects. This information can be used to guide modifications to the topology.

3.4 Failure Set Trimming

The simplest choice of a failure set is to include all nodes and edges, or perhaps just all nodes, in the topology. This set includes many faults that aren't very interesting, because their effects are minor and very predictable. The resulting robustness metric will require more work to compute, and will be diluted by the effects (or, rather, the non-effects) of the uninteresting failures.

The prime candidates for exclusion from the failure set are the leaf nodes. In a local-area internetwork, these are networks, and the effect of one failing is entirely predictable: other than that network, nothing else is affected. *Near-leaf* routers that are connected only to leaf networks plus one link to the main body of the internet have similarly predictable effects. In the internetworks that we studied, over half of the networks were leaves, and almost half of the routers were near-leaves.

It would be possible to further extend this method, and eliminate from the failure set any trees whose root is attached to the main body of the internet. At some point, however, the pieces eliminated would be large enough that the fault effects would no longer be small enough to be ignored. Also, there may not be much additional savings: in the internets that we studied, only about ten additional nodes would have been eliminated, compared to about sixty leaves and near-leaves.

3.5 Double Faults

The general framework given above for computing robustness measures works well for single faults, but there are several problems to be overcome when failures consisting of multiple faults are considered. As we will see below, most of these problems can be overcome for double faults in internetworks of moderate size. The complications introduced by multiple faults include:

There are possible -multiple faults in a network with possible single faults. The large size of the failure set, even for small , may make computation of an overall robustness metric too expensive.

Multiple physical faults are often linked in unexpected ways, making the problem not one of multiple independent faults, but multiple, correlated faults. The usual cause for this is shared facilities such as power or conduit space.[2]

Because of these linkages, joint probabilities are needed to weight the failures. For such failures, it is difficult to even imagine the range of problems that would cause them, much less assign probabilities to them. The problems usually appear to be one-shot events — a backhoe cutting a bundle of cables, power failure in one building, building air conditioning failure, etc. — and the same failure rarely recurs. By comparison, probabilities for single failures are easy to obtain.

Because many failures are linked, it is important to understand what multiple faults are likely to occur in a particular internet topology, and how those fault sets will affect connectivity. It is usually impractical to take a "shotgun" approach and compute a robustness metric for all possible fault sets — most of those fault sets won't ever happen, and the ones that are likely can only be determined by looking at factors not readily visible in the topology, such as shared resources.

Because of these factors, the concept of an overall robustness metric is not very useful — it would either be too expensive to compute or would require input data that isn't available. A better approach is to search for multiple faults whose effects are interesting, and present them for manual analysis. We will consider only double faults. These techniques extend directly to higher-order faults, but get much more expensive to compute.

[2]One of the more spectacular examples of how multiple, apparently independent, systems were disabled by a single event occurred on December 12, 1986, when a construction crew dug through a fiber optic cable running between Newark, New Jersey and White Plains, New York. As it turned out, seven Arpanet links, including all of the cross-country links into the Northeast US, went through this cable, causing a major portion of the Arpanet to be disconnected. It was not obvious, from looking at a geographic map of the sites connected by those links, that they shared a common point of failure. In fact, no one really knew where the physical communications channels went, because the Arpanet links were just leased from the telephone company. This partition lasted for 11 hours, the time it took AT&T to restore service[HMC 86].

The number of fault pairs to be analyzed can be greatly reduced by trimming nodes out of the failure set, as described in section 3.4. In this case, a double fault is not considered if *either* of the component faults is a leaf or near-leaf. In the topologies that we studied, this reduced the number of fault pairs to be considered by a factor of 7 to 9.

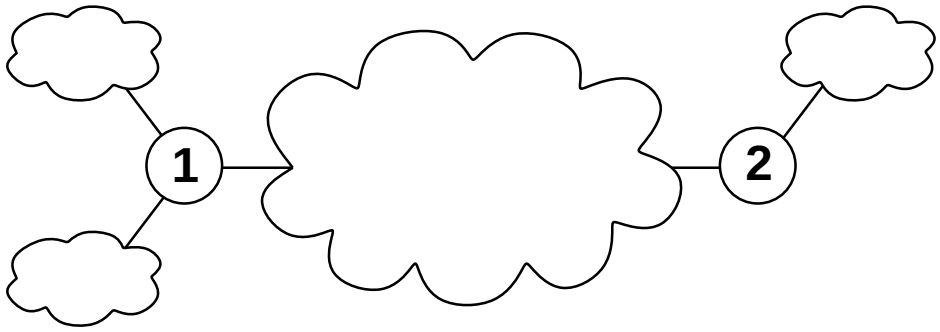

Figure 3.5: An Independent Fault Pair

Most possible fault pairs are *independent* — their effects do not interact. Consider figure 3.5, which shows two possible faults, labeled 1 and 2. If either fault occurs by itself, the internetwork is broken into three or two fragments, respectively. If both faults occur, the internet is broken into four fragments. The effects of the two faults are just additive. Independent fault pairs are not particularly interesting because nothing "new" has happened just because both faults occurred at the same time.

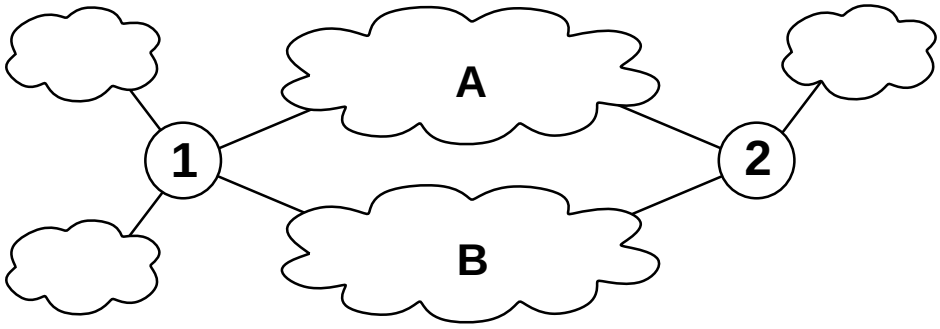

Figure 3.6: A Dependent Fault Pair

Dependent fault pairs are more interesting, however. Consider figure 3.6, which shows a similar internetwork. In this case, the effects of either fault by itself is the same as before, because the large fragments (A and B) are still connected by the non-failed node. If both

faults occur together, however, the two fragments are disconnected, and the effects of the faults have been compounded.

Dependent fault pairs can be detected very easily. If two faults are independent, then the number of fragments caused by the fault pair will be one less than the sum of the number of fragments caused by each of the faults alone. If the faults are dependent, the number of fragments produced by the fault pair will be greater. In the topologies that we studied, the number of dependent fault pairs was about two orders of magnitude less than the number of possible pairs, leaving between 20 and 50 fault pairs to be inspected by hand.

Many of these fault pairs are not interesting because the two faults are not causally linked. As pointed out above, however, determining such linkages requires extensive physical knowledge and intuition. Given the relatively small number of dependent faults, manual inspection is reasonable.

3.6 Measurements of the Stanford Internetwork

In this section, we present the results of applying the tools discussed in the previous sections — measuring the redundancy and evaluating the partition fraction-based robustness metric. In order to understand these results, it is first necessary to understand the context — the topology and changes that were going on in the Stanford internet.

3.6.1 Stanford Networking Environment and Evolution

Stanford first started using local-area networks in 1979, when a grant of experimental 3 Mbit Ethernet equipment was received from Xerox. The single network was initially confined to the Computer Science Department, but was soon extended to the Electrical Engineering Department, in another building. 3 Mbit Ethernet segments can be up to 1000 meters long (compared to 500 meters for 10 Mbit), making it suitable for links between

buildings. This network was extended further with repeaters, until it had 4 network segments connecting computers in 5 buildings. At this point many of the experimental Ethernet's physical limits were reached, forcing other alternatives to be considered for extending the network.

Ethernet bridges were first built at Stanford in 1981, using 68000 processors and home-made 3 Mbit Ethernet interfaces. They worked well enough for connecting a few network segments together, but were not pursued for several reasons:

> They would not work for connecting 3 Mbit and 10 Mbit Ethernet networks together, and it was clear that 10 Mbit Ethernet was on the way.
>
> The existing implementation could not deal with loops in the topology, preventing any attempt to add redundancy.
>
> There was concern that a very large network built out of bridges would present enough traffic to saturate the bridges. Saturation was never reached on the existing network, but there were never more than 3 bridges in use.

Although no new 3 Mbit bridges were added to the network after 1982, they were very reliable, and one remained in service until 1985, when it was replaced with a router so that a 10 Mbit network could be added nearby.

The first Ethernet router was built in mid-1981, to connect the medical school to the CSD/EE network. That implementation was based upon very resource-poor hardware (PDP-11/05) and could only connect 3 Mbit Ethernets. It was never copied and was retired in 1985.

The next-generation router was built in mid-1983, based upon an 8-MHz 68000 and appropriate network interfaces. It was able to connect both 3 Mbit and 10 Mbit networks. With the availability of a "real" router, the growth of the Stanford internet began in earnest. By the end of 1984, there were 6 routers connecting about a dozen networks. The original 3 Mbit network formed the backbone of the internet. It was still overextended, and remained so, despite continual efforts to prune it.

By the end of 1984, the network had grown large enough, and was changing so rapidly, that the manually-maintained map of the internet was abandoned.

By early 1985, the 3 Mbit network was becoming very difficult to maintain and expand, due to lack of additional interfaces and transceivers. Fortunately, by that time, most equipment could be converted to use 10 Mbit hardware. The major exceptions were a fileserver and a network printer that were part of the original Xerox grant. The extra length of the 3 Mbit technology was also needed for 3 inter-building network segments.

In early 1986, work was begun on a large campus backbone network built of 10 Mbit Ethernet segments, repeaters, and commercially-available Ethernet bridges. It was initially intended to provide backup communication paths to supplement the increasingly unreliable 3 Mbit networks. By late 1986 it was clear that this technology was very reliable and it was further expanded to become the central backbone network. During 1987, additional network segments and bridges were added to provide redundancy in the backbone.

The original 3 Mbit backbone was essentially taken out of service in November, 1987, when the building housing a central splice for it was demolished.

Figures 3.7, 3.8, and 3.9 show the topology of the Stanford internet during the later part of this history. The growth of the 10 Mbit backbone is the most obvious feature. These maps will be useful for understanding the impact of individual topological changes on the partition fraction robustness metric, which are presented in Section 3.6.3.

The maps are laid out geographically, with routers and bridges placed close to their true positions. South is approximately at the top of each figure. Each map also includes a table giving the number of routers, bridges, and networks in the internet, as well as calculations of , , and .

We monitored the topology of the Stanford internet from August, 1986, to April, 1988. During this period, 54 long- or medium-term changes were made to the topology of the internetwork. Many other short-term or transient changes were observed, but these are ignored for the purposes of these analyses. Several special-purpose or experimental

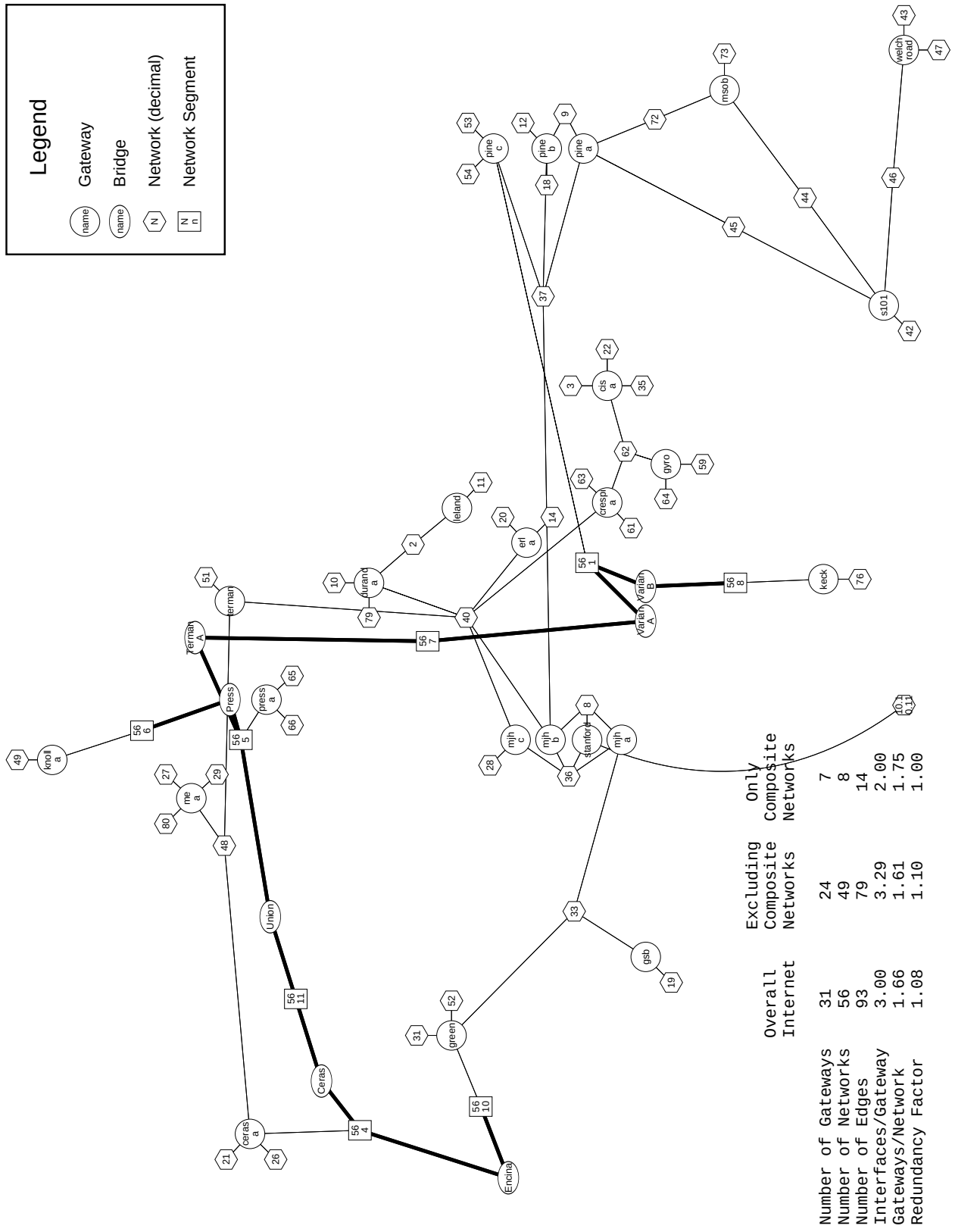

Figure 3.7: Stanford University Network of Aug 6, 1986

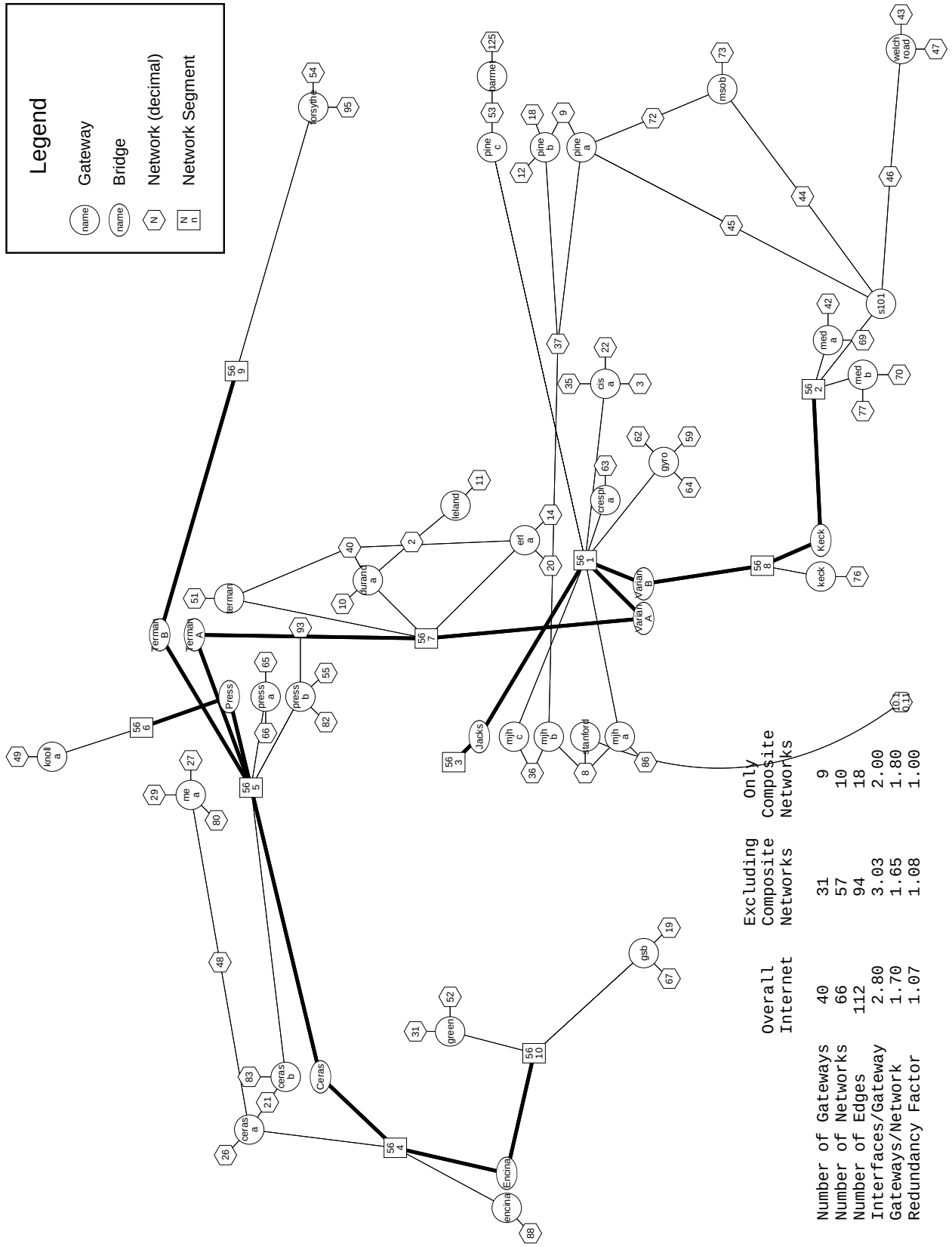

Figure 3.8: Stanford University Network of Nov 10, 1987

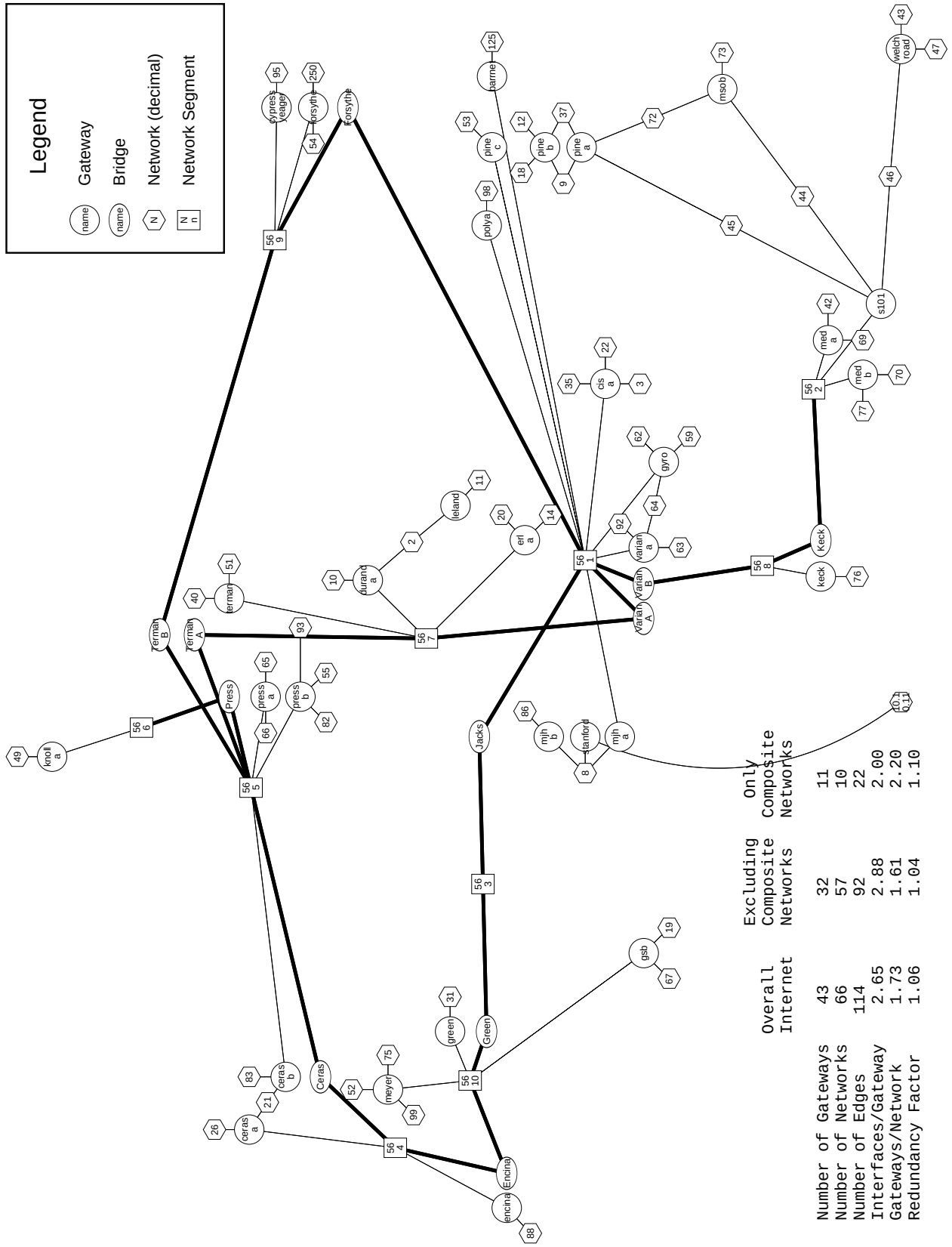

Figure 3.9: Stanford University Network of Apr 29, 1988

routers and networks are consistently omitted from the topology in order to provide a stable baseline for comparison.

During this period, the network topology moved from a decentralized topology built with 3 Mbit Ethernet toward a backbone topology built with 10 Mbit Ethernet. This can be seen in the chronology in two trends:

> The *link-net*, a 10 Mbit Ethernet backbone built out of Ethernet bridges was expanded to cover most of campus. The bridges are shown on the map as squares connected together by heavy lines. This is the "composite network" mentioned in the table printed with the maps.
>
> Several older experimental 3 Mbit Ethernets were taken out of service or severely pruned. These were nets 33, 37, 40, and 48.

We now analyze the impact of these changes using the techniques described earlier in this paper.

3.6.2 Redundancy Results

Figure 3.10 is a plot of for the Stanford internet during this period. The three curves are reference lines indicating different amounts of redundancy. The solid line traces the actual changes in . Each point, marked with a '+', represents one distinct value that took on. The state as of August, 1986 is marked with a circle, and the state as of April, 1988 is marked with a diamond.

shifted quite a bit during the period. The most dramatic change is that went down from 3.0 to 2.7, with a peak of 3.15 in between. This change is significant because the lowest value that can have is 2.0. Some of this change is due to the addition of four bridges, each of which has two interfaces. The rest of the change isn't attributable to any one cause, and represents a general trend toward routers with fewer interfaces.

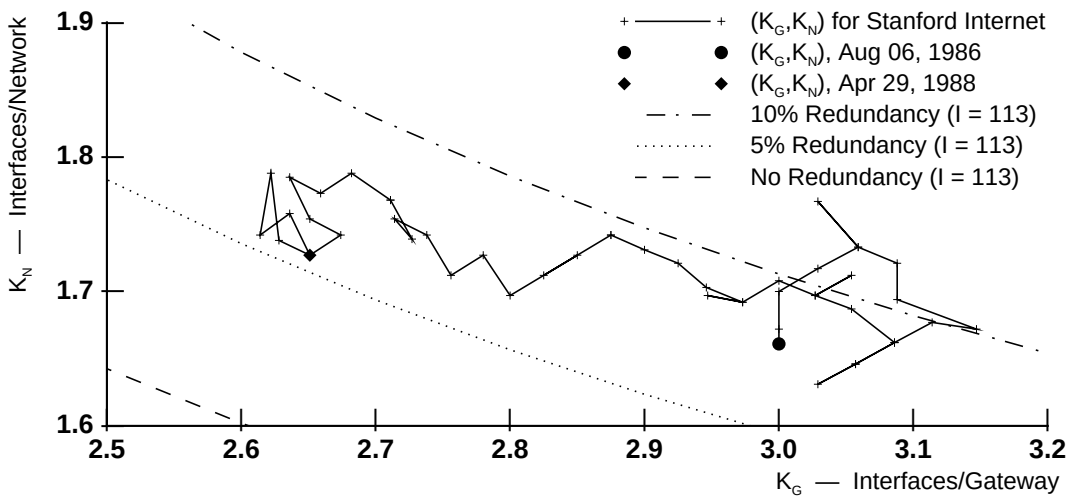

Figure 3.10: Trace of for Stanford Internet

The redundancy factor shows a general downward trend, with a peak of 12 percent redundant connections in late 1986, ending at about 6 percent in May, 1988. This is a result of the general trend toward a backbone topology from a less structured mesh. Notice that, for any single change in the topology, the shift of is very slight. This is to be expected, since most changes consisted of the addition or deletion of a single interface, router, or network. In the next section, we will show that changes in robustness, for the same small changes in topology, were very dramatic.

3.6.3 Partition Fraction Results — Single Faults

Studying the behavior of the partition fraction produces more insight into the characteristics of the internet. Figure 3.11 shows the robustness metric based on the partition fraction averaged over two single-fault failure sets: all non-leaf networks and all non-near-leaf routers. In addition, the raw partition fraction values for the ten most disruptive single faults are shown.

The vertical axis shows 1 , so “no effect” is at the bottom and “full partition” is at the top. The ten worst values of are shown shaded, with the most disruptive shaded light gray and the least disruptive shaded dark gray. The interpretation of this is that any

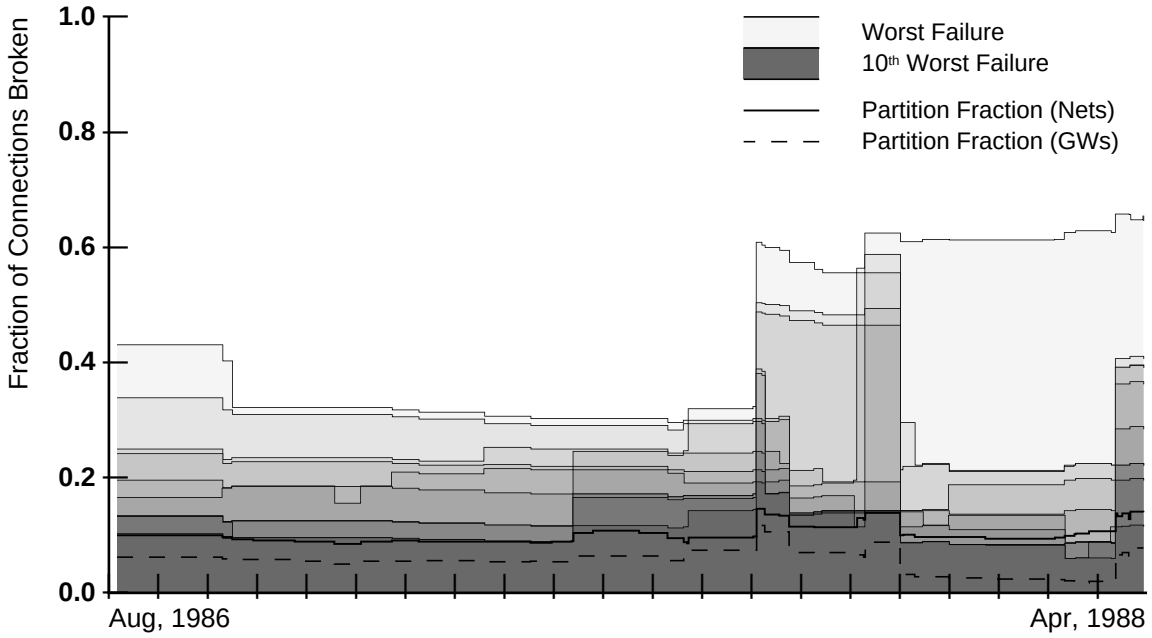

Figure 3.11: for Ten Worst Single Faults vs. Time

of ten faults can produce a in the dark-gray region, while only one fault can produce a in the light-gray region. The shaded regions do not track the same faults across the graph — they only indicate the ten highest values of at any time, so different faults occupy different positions at different times.

The two robustness measures track each other closely, with the average over network failures consistently worse than the average over router failures. The reason for this is that, in this internetwork, very few router failures produce bad partitions; most only disconnect a few leaf networks. *Non-leaf* network failures, on the other hand, usually disconnect at least one near-leaf router, for an effect at least as bad as that of any failed router. In addition, networks are more prone to produce a significant partition when they fail. We have no reason to believe that this is a general result. Other styles of topologies, where networks tend to be more point-to-point, for example, could have an opposite characteristic.

There are several sudden shifts in , brought on by the following changes:

Oct 16, 1986. A link was added between link-net (56-7) and *durand-a*. This kept that fragment connected to the main body of the internet when net 40 fails. Up until this time, Net 40 had had the worst .

May 13, 1987. The link from net 33 (3 Mbit) to *mjh-a* was removed, allowing all of GSB, Green, Encina, and Ceras to be detached by single failures in that portion of link-net.

Sep 4, 1987. The link from net 48 (3 Mbit) to *terman* was removed, allowing even bigger fragments to be detached by even more single faults, especially 56-5. This produced a dramatic worsening of the robustness measures.

Nov 5,10, 1987. Removed many links from net 40 (3 Mbit) to *mjh-b*, *mjh-c*, and *crespi-a*, reconnecting them to link-net (56-1). This made 56-1 and 56-7 single points of failure that could split the internet cleanly in half.[3]

Dec 1, 1987. *Jacks* (a bridge) was added, completing the large loop through Green, Ceras, etc. This reduced the effect of many failures back to the levels before Sep 4, 1987 and caused a corresponding improvement in the robustness measures.

Apr 12, 1988. Disconnected net 37 from *mjh-b*, removing the only redundant path to the cluster including *pine-a*, *pine-b*, *med-a*, and *med-b*. This cluster can now be disconnected by any of six single faults.

3.6.4 Partition Fraction Results — Double Faults

Figure 3.12 shows the the raw partition fraction values for the twenty most disruptive double faults. Only *dependent* faults are shown, and double faults involving leaf nodes and near-leaf nodes are not included. These faults have *not* been evaluated to see if there is any possible causal relation between the components involved in the double fault.

[3]The motivation for this major change was not to retire more 3 Mbit equipment, but instead the impending demolition of the building housing two bridges and a router. The equipment was moved and the other (non-3 Mbit) networks were re-routed.

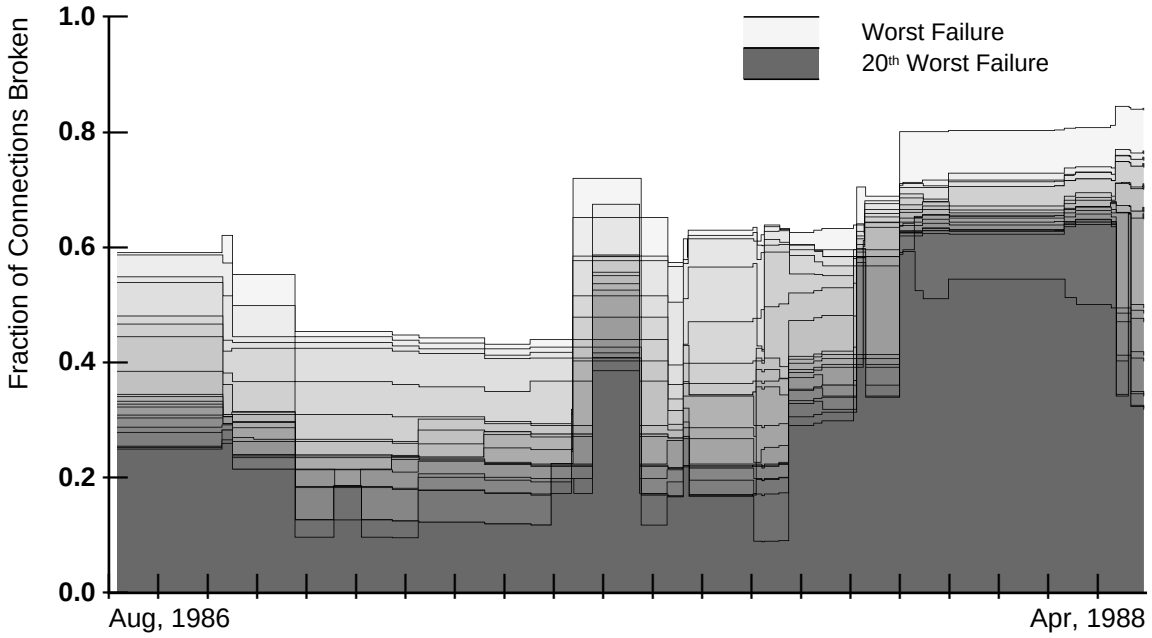

Figure 3.12: for Twenty Worst Dependent Double Faults vs. Time

In looking at this graph, it is important to remember the characteristics of dependent double faults: a double fault is only included if its effect is worse than the combined effect of its two component faults. An internet with no redundancy, will have *no* dependent double faults, because no components in the topology are redundantly connected. Closing a large loop in such a topology will create several dependent double faults, because two failures will now be required to break the loop apart.

So, it is possible (even likely) that adding redundancy that protects against single faults will increase the number of dependent double faults. For example, after the Dec 1, 1987 change, the Stanford internetwork's vulnerability to single faults was greatly improved by closing the large loop through Terman, Ceras, Green, etc. This change produced an even greater *increase* in the number and severity of dependent double faults, as seen in figure 3.12.

Besides the changes listed above, there are a few other sudden shifts in , brought on by the following changes:

May 25, 1987. Durand-a-router's interface to net 40 was removed for testing and forgotten. Up until this point, there were parallel paths running from Ceras to net 56-1. These two paths were connected at Ceras, 56-1, and 56-7 to net 40 (through durand-a). These three redundant connections gave an extra measure of stability to that area. When the link from durand-a to net 40 was removed, many components that had been doubly backed up now became part of dependent double faults.

Jun 23, 1987. The above interface was re-installed.[4]

Sep 4, 1987 - Apr 29, 1988. Almost every change in the single-fault graph is reflected *inversely* in the double-dependent-fault graph.

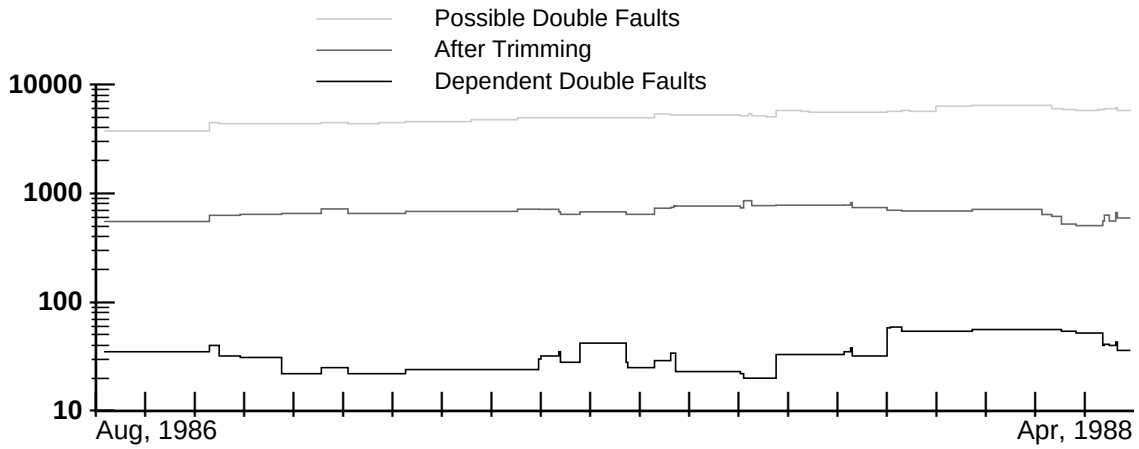

Figure 3.13: Savings for Double Fault Calculations

Figure 3.13 shows the relationships between the number of possible double faults, the number for which partition fractions actually had to be calculated (after trimming leaves and near-leaves), and the number of dependent double faults. The first two steadily increase as the size of the internet increases. The number of dependent faults, as observed above, is very sensitive to changes in the topology.

[4]A similar change can also be observed on Oct 16, 1986, when the durand-a interface to 56-7 was originally added.

3.6.5 Summary of Analysis

The Stanford University Network has been undergoing gradual change from a loosely coupled “mesh” topology to a more structured “backbone” topology. As changes were made, the internetwork became more or less vulnerable to failures, depending upon what the change was.

The measures of robustness, , , and were somewhat useful for spotting trends in the composition of the internetwork, but were not useful for determining its robustness.

The partition fraction, , and its derived robustness measure were both useful for pinpointing changes that significantly affected robustness. The robustness measures, by themselves, only indicated *when* a significant change had been made. Examination of the failures with the greatest impact provided a more dramatic indication, as well as pinpointing weak spots.

Double-fault analysis also pointed out weak spots, but since the Stanford internet had only just reached the point where most serious single faults had been fixed, there were many potentially serious double faults identified by the analysis. Additional information is required to determine if particular double faults actually represent a real threat, because most fault pairs are physically independent.

3.7 Summary and Conclusions

We have presented several analytical tools that can be used to analyze the topology of an internetwork and provide a basis for comparison of topologies. The topology is represented as a bipartite graph, which provides a useful fault model. The tools are practical to use and provide results that can point out specific potential problems in the internetwork.

We analyzed the effects of changes to the Stanford internetwork’s topology and found

that the tools were able to provide excellent detail. In several cases, the tools revealed problems that a casual inspection had missed. We were able to understand both the effects of a change and why the change produced the effects it did. By providing a better understanding of the problems, we were better able to find ways to fix them.

Throughout all of these measurements, the existence of an automated network monitoring facility was invaluable. The data could not have been obtained any other way. Besides providing data for this research, the data has been used in the day-to-day operation and maintenance of the Stanford Network. Good network management facilities are essential just to be able to keep track of what is going on in the internetwork, and even more important when something is broken.

Chapter 4

Traffic Patterns

The previous chapters dealt with how local-area internetworks perform and how they are built. This chapter discusses how they are actually used, by looking at the traffic patterns in an internet. We use two different tools to study traffic patterns. Maps, annotated with traffic volume, show traffic intensities and usage patterns in the internet. Traffic matrices show packet rates between each source-destination pair in the internet. They highlight heavily-communicating pairs and are useful to identify communities of interest. Finally, we discuss how we deal with inadequate and partial data.

Both of these tools are useful for network management. They provide information necessary for planning network upgrades and understanding how the internet is used. The latter point is very important. Without knowledge about what's going on in the internet, there is no way to tell:

Who are the real users of the network?

Where are the "communities of interest"?

Who uses services associated with a particular network? A computer center and external networks are examples of such resources.

In many cases, this information could be obtained from accounting data. At present, however, very few (if any) internetworks provide any more accounting data than just simple packet counts, which doesn't give any information about traffic patterns. Most of the usage information presented in this chapter was gleaned from traffic statistics and routing tables.

Even if we had full control over the statistics collected by routers and end-nodes, it is not practical to collect and process all possible information over an entire internet. At one extreme, this would mean collecting a full packet trace, or at least the headers. This is clearly too much information to collect, transmit, and store. Even if the data came for free, most of it wouldn't amount to any more than the sum of its parts — sixty networks worth of interarrival time or size distributions, for example. Such data would be useful for comparative studies, between networks, but doesn't say much about the behavior of the internet as a composite entity.

So, if we can't have it all, and "everything" isn't that interesting, we must decide which data are important and figure out how to present and interpret it. In most cases, it wasn't possible to get all of the data that we would have liked. Instead, we collected what was available and used that to experiment with presentation techniques, keeping in mind where the data was lacking.

4.1 Measurement Technology and Methodology

For the delay measurements in Chapter 2, we used dedicated hardware to collect very detailed information about one router in the internet. It was not practical to collect traffic data for the internetwork with special hardware distributed over its entirety. Fortunately, the routers themselves collected almost all of the data that we needed. Two diagnostic reports were used, one giving interface statistics and the other displaying the routing table. In addition, we can find out from the router how long it has been up, and therefore detect system restarts.

The bridges that comprise the Ethernet backbone collect similar statistics, although their forwarding tables[1] don't provide useful route usage information. Fortunately, there are only two hosts on the backbone, so most of the backbone traffic originates on one of the leaf networks and passes through a router, where it is counted.

4.1.1 Interface Statistics

```
Ethernet 0 is up, line protocol is up
  Hardware type is Type 2 Ethernet, hardware address is 0207.0100.33E4
  Internet address is 36.56.0.10, subnet mask is 255.255.0.0
  MTU 1500 bytes, bandwidth 10000 Kbit, delay 1000 usec, rely. 255/255
  Encapsulation is ARPA, loopback is not set, keepalive is set
  ARP type: ARPA
  Last input 0:00:00, output 0:00:00, output hang never
  Output queue length is 0 (maximum 100)
  Five minute input rate 41176 bits/sec, 54 packets/sec
  Five minute output rate 57012 bits/sec, 84 packets/sec
     29822983 packets input, 2951937976 bytes, 0 no input buffers
     Received 3644191 broadcasts, 49159 runts, 0 giants
     50541 input errors, 4839 CRC, 43174 frame, 201 overrun, 255 ignored
     42836313 packets output, 2762746384 bytes, 0 congestion drops
     1 output errors, 207117 collisions, 5 interface resets, 5 restarts
```

Figure 4.1: Statistics for One Router Interface

The router keeps statistics for each of its network interfaces. The report for one interface is shown in Figure 4.1. The first part gives configuration information such as Ethernet address. The remainder gives statistics about the traffic sent and received, including error counts. Most of the routers have very low error counts, so those can be ignored. By subtracting two sets of statistics taken at different times, we can find the packet and byte rates for both input and output. Since an interface is the connection between a router and a network, these statistics reflect *all* of the traffic flowing between the two.

The transmitted and received traffic counts include all traffic sent or received at the interface, which includes non-IP and non-routed traffic such as ARP requests/responses, routing table updates, and broadcasts. For nontrivially-loaded routers, these only amount

[1] Bridges keep a sort of routing table, called a *forwarding table* that tells which side of the bridge each host (as known by its link-layer address) is on. Since forwarding tables only know about hosts, not networks, they don't provide the sort of structural information about the internetwork that can be gleaned from a routing table.

to a few percent of the routed traffic, so the router statistics accurately reflect routed traffic in the internet.

4.1.2 Routing Table

Figure 4.2 shows a small excerpt from the same router's routing table. Each entry gives information about one route:

Type	How the entry was obtained, *e.g.* statically configured, directly connected, which routing protocol, etc.
Destination	Target network
Routing metric	Capacity and distance
Next hop	Next router on the path
Age	Time since last routing update showing this route
Use count	Number of packets sent via this route
Interface	To transmit on

Entries for directly-connected networks only give a use count. Some destinations have multiple routes and therefore multiple entries (*e.g.* 36.64.0.0).

The use count is the most important for our purposes. By subtracting two snapshots and dividing the use count and uptime differences, we get the packet rate through each route. There are two events that interfere with the accuracy of this calculation: router reboots and route resets.

We detect router reboots directly by noticing that the change in uptime differs significantly from the change in elapsed time. After a reboot, a valid rate can usually be obtained by dividing the current count by the current uptime, ignoring the old snapshot.

```
S Net 129.140.0.0 via 36.125.0.1, 70099 uses
S Net 132.154.0.0 via 36.8.0.136, 0 uses
C Net 36.0.0.0 is subnetted (mask is 255.255.0.0), 71 subnets
G     36.53.0.0 [160/1] via 36.56.0.5, 14 sec, 21469 uses, Ethernet0
G     36.54.0.0 [160/1] via 36.56.0.2, 24 sec, 499734 uses, Ethernet0
G     36.55.0.0 [160/1] via 36.56.0.9, 24 sec, 2 uses, Ethernet0
C     36.56.0.0 is directly connected, 49867 uses, Ethernet0
G     36.59.0.0 [160/1] via 36.56.0.18, 16 sec, 962 uses, Ethernet0
G     36.63.0.0 [160/1] via 36.56.0.64, 5 sec, 0 uses, Ethernet0
G     36.64.0.0 [160/1] via 36.56.0.18, 16 sec, 401 uses, Ethernet0
                        via 36.56.0.64, 5 sec, 401 uses, Ethernet0
G     36.65.0.0 [160/1] via 36.56.0.10, 18 sec, 666 uses, Ethernet0
G     36.66.0.0 [160/1] via 36.56.0.10, 18 sec, 1348 uses, Ethernet0
S Net 192.26.25.0 via 36.8.0.136, 0 uses
```

Figure 4.2: Excerpt from Routing Table

When a destination network becomes unreachable, perhaps due to a network failure or an extended router outage, the routing table entry for that destination will be discarded. In fact, it is most likely that the same entry will disappear from *all* routing tables in the network, because the routers are all exchanging routing information. When the network becomes reachable again, the routing table entries will reappear, with the counts starting at zero. This could lead to inaccurate data — the difference will either be negative or too small. Fortunately, route resets are infrequent and we can usually deduce when they happen.

From observing the routing tables, routes in the Stanford internet are generally stable — route resets are infrequent events. Because of this, routing table entries have a lifetime of many days. Since we sampled the routing tables several times a day, the most likely result of a route reset is that the new count (a few hours old) will be less than the previous count (several days old). This situation is easily detected, and that data is ignored. In addition, the route resets that were observed were all for little-used networks, so even if a too-small number did get past, it would have very little impact on the data, overall.

A more serious problem is that the fast routers (see Section 2.2.2 use a cache of routing table entries and only increment the use count on cache misses. At the time these measurements were made, 6 out of 32 routers had the hardware necessary to support route caching. Three of these were the most heavily-loaded routers in the internet, two were moderately loaded, and one was essentially unloaded. The routing cache for all of these routers was turned off while these measurements were made.

4.1.3 Bridge Statistics

The bridges that comprise the Ethernet backbone collect much fewer statistics than the routers. They keep packet counts for their interfaces, but not byte counts, so it isn't possible to get statistics for bandwidth through the bridge, only packet counts. They do not keep use counts for routing table entries, which is fine, because a bridge's routing table contains only Ethernet addresses, which cannot be used to deduce location in the internetwork.

```
Line counters for Line 1  as of    2-NOV-1989 09:43:09
Bridge   CERAS,   Address   08-00-2B-03-1F-F3

Bridge seconds:                              259953
Invalid protocol messages:                        0
Filtered frames:                            1603815
Bad frames:                                       1
Lifetime exceeded frames:                         0
No forwarding entry frames:                       0
Forwarding transition count:                      1
Collision Presence Test errors:                   0

                           Transmitted                        Received

Frames:                       15998447                        13942359
Bridge frames:                  259580                               1
Frames lost:                         0                               0
```

Figure 4.3: Statistics for One Bridge Interface

Figure 4.3 shows the statistics for one interface in a two-interface bridge. *Received frames* includes all packets on the network segment, because bridges operate in "promiscuous mode" and receive all packets. *Filtered frames* is the count of frames that were discarded, either because the destination hosts was on the same side of the bridge as the source host or because of an administrative rule, such as restricting broadcasts or certain packet types. Omitting errors and frames originating at the bridge (both of which are minor contributions), the relation between these three major counts is $_2 \quad _1 \quad _1$ and $_1 \quad _2 \quad _2$. If a bridge were to have more than two interfaces (none in the Stanford backbone do), then the count of packets forwarded from one interface to one or more others in the bridge would be .

As with the router interface statistics, these include other packets in addition to routed

traffic. Since the bridges do not operate at the IP layer, there is no way to confirm how large a percentage of the traffic is IP rather than some other protocol. On the other hand, at the present time, the routers are all IP routers, so the proportions of traffic seen at the routers should be reflected on the backbone as well.

One advantage that the bridge statistics offer is that, since they receive *all* packets on the network segment, the statistics should show the total traffic volume. The received traffic counts do not include packets transmitted by the bridge, so the total traffic is actually the sum of the transmitted and received packet rates. We tested this assertion by comparing these sums for all bridges connected to a segment, for all segments. All rates agreed to within a few percent, and in many cases to within less than one percent.

4.2 Network Map Annotated With Traffic Volume

We used the packet rates obtained from the bridge and router interface statistics to annotate a network map and give a graphical interpretation of the traffic intensity throughout the Stanford internet, shown in Figure 4.4. The routers in the Medical School (in the lower-right corner) could not be monitored, so no traffic volumes are shown, just dotted lines. Two of the bridges (Forsythe and Ceras[2]) are automatically disabled by the bridges' spanning-tree algorithm and so show no traffic.

When assigning a value for the traffic volume "through" a router interface, we include traffic in both directions (transmitted and received). To be consistent with this scheme, bridge interfaces are assigned the volume $_1$ $_1$ $_1$. For a two-interface bridge, this is equal to $_1$ $_2$, which is the same value for both interfaces, which makes perfect sense.

The map shows very clearly which links are heavily loaded, as well as which leaf networks have heavy traffic to the backbone. It is clear that large portions of the internet (and

[2]The choice of which bridge to disable seems to change with time. At most times, Encina has been disabled rather than Ceras.

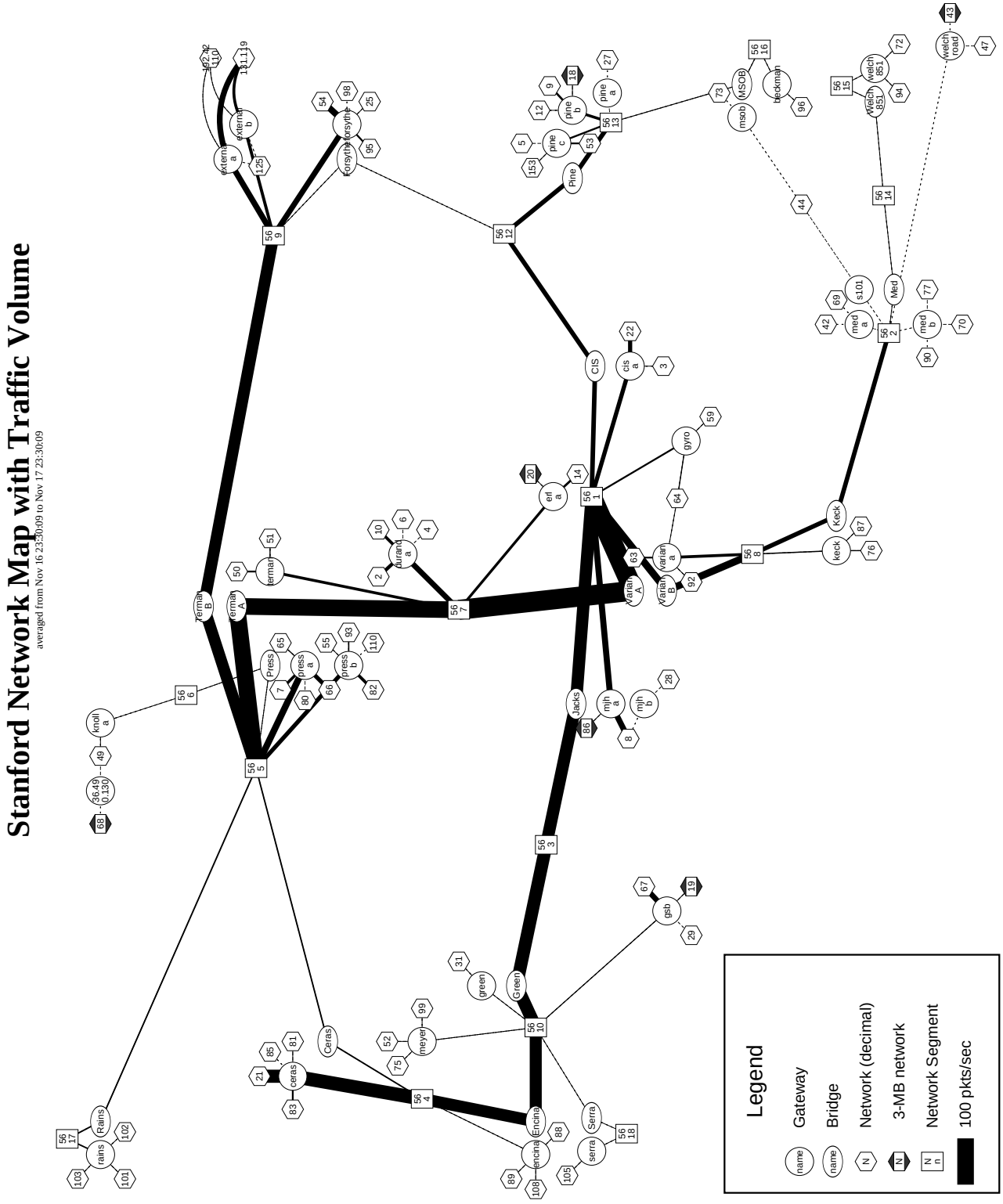

Figure 4.4: Stanford Network with Traffic Volumes

backbone) are very lightly loaded, while other areas have significant load due to a few major sources. Some of these sources are listed in table 4.1.

Network Number	Resources
21	Undergraduate computing center
7	Undergraduate terminal cluster
131.119, 192.42.110	External wide-area networks (BARRNet/NSFNet)
54	Campus computing facility incl. on-line library catalog
8	Computer Science
2, 10, 14	Electrical Engineering
22	EE and CS
66	Dormitory network
82	University administration

Table 4.1: Networks With Heavy Internet Loads

Although it is informative and easy to interpret, the map doesn't show the correspondence between sources and destinations. For example, it is obvious that network 21 (the undergraduate computing center) has the heaviest traffic of any of the leaf networks, and that that traffic flows through ceras (a router) and Encina, Green, and Jacks (bridges) to get to network segment 56-1. From there, however, it is unclear where this traffic goes; segment 56-1 is the major nexus for traffic in the backbone[3]. Questions about source-destination pairs require knowledge of route usage in the internet.

4.3 Routing Matrix

To get a better picture of who's talking to whom in the internet, we looked at the routing tables of all of the routers, as described in Section 4.1.2. Figure 4.5 shows a matrix of source-destination pairs, with the volume from one subnet to another represented by the area of the square at the intersection. The rows and columns are sorted by total

[3]Unhappily, besides having the highest traffic load, segment 56-1 is also the weakest link in the internet — if it fails, it will disconnect the Medical Center, Computer Science, and much of Electrical Engineering.

destination traffic. For example, 0.0.0.0 is the most popular destination, 36.21.0.0 is the second-most popular, etc.

There are a few things to be aware of when interpreting the matrix:

Rather than try to show all 75 subnetwork numbers, the subnets are aggregated by router. For example, the row/column for 36.21.0.0 includes traffic for subnets 81, 83, and 85, as well as net 21. The row/column labels used are for the most heavily-loaded subnets attached to the router.

The external wide-area network connections through the external-a and external-b routers are shown as "0.0.0.0".

As in the network map, routing tables could not be obtained from all of the routers in the internetwork. Ordinarily, such missing data would cause the source columns for these routers to be empty. However, these routers' associated subnets show up as destinations in other routing tables, so we do have *some* data about them. Rather than show no data at all, we filled in these columns with the data from the corresponding row, making the assumption that traffic is symmetric. This fudged data is shown in light gray. This symmetry assumption seems justfied; however, it is important to remember that traffic *among* these silent routers is not reported at all. These routers comprise the network for the Stanford Medical Center, so it is believable that there is significant traffic going unreported.

The human eye is not very good at judging proportions of area; study the legend carefully. I picked area (rather than length) for the representation so that the density of the matrix (which is a characteristic of area) would highlight the heavily-used subnets.

The data for this matrix was gathered at a different time than the data for the map, so they do not reflect the same traffic patterns.

As seen in the map, relatively few of the leaf networks account for the bulk of the traffic. Quite a few things can be learned from careful study of it:

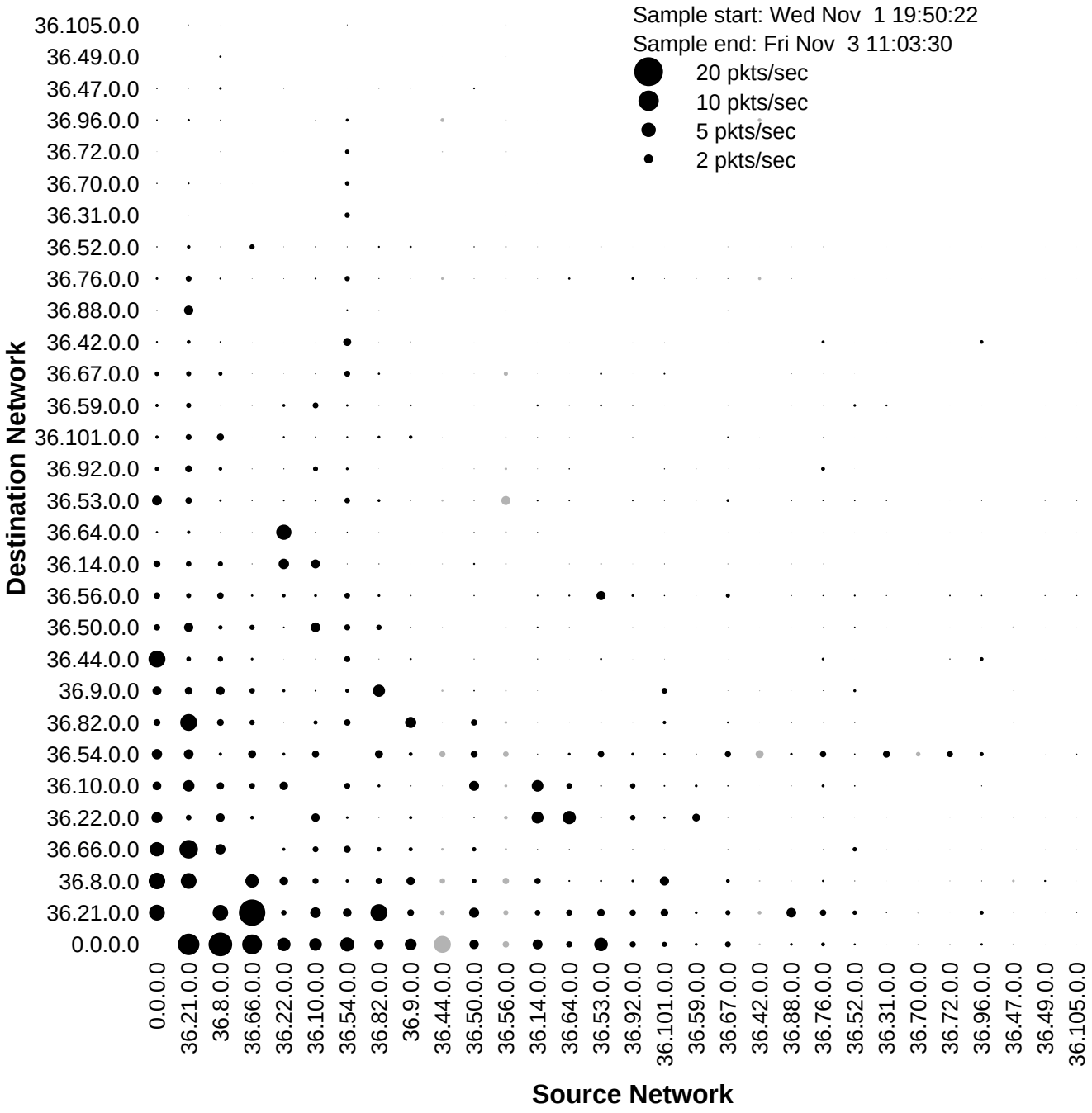

Figure 4.5: Traffic Source-Destination Matrix

Several networks (0.0, 21, and 54) are used by just about every subnet. That is not surprising, considering the resources on these nets. (See table 4.1.)

As one would expect, the matrix is roughly symmetric about the diagonal.

A few networks are subordinates of others. For example, net 64 communicates almost exclusively with net 22. (Although, not visa-versa.)

The most popular "network" in this sample was the connection to the external networks.

A large portion of the traffic from the dormitory networks (net 66) was to the external networks. This is puzzling, because the majority of the hosts on net 66 are terminal concentrators; no systems exchanging mail and probably few computers capable of doing file transfers. Is *all* of this telnet traffic? If so, where do these students (most of them undergraduates) have accounts?[4]

Traffic is obviously concentrated in a few source-destination pairs. Figure 4.6 shows the distribution of traffic intensities. The symmetric pairs of traffic intensities shown in Figure 4.5 were summed and sorted from largest to smallest. Their cumulative sum is plotted against the number of pairs included in the sample. This graph only shows the first 100 communicating pairs, out of a possible total of 435 (for 30 subnet groups). Fifteen of the source-destination pairs account for half of all traffic in the internet, and 65 account for 90%. For reference, an exponential with the same area is plotted. The fact that the distribution is close to exponential demonstrates that, while traffic is heavily concentrated within a few pairs, there is no sharp cutoff point where one can say, " subnets use the network, the rest are inconsequential". The selection of such a cutoff point would be arbitrary.

[4]We don't suspect any nefarious activity here, but the usage is surprising — most administrators had expected that the bulk of the usage would be to the undergraduate computers on net 21.

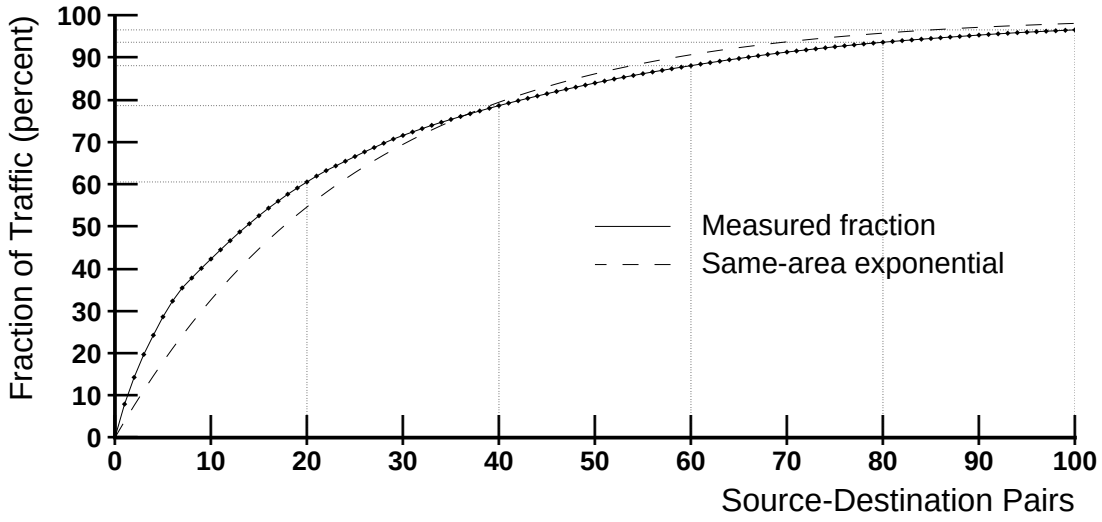

Figure 4.6: Cumulative Traffic Distribution

4.3.1 Other Ways to Collect Routing Data

There are some problems with using raw routing table statistics to construct the routing matrix. In this section, we discuss the problems and ways to overcome them. We used the routing table because it was the only data available. This method has the following problems:

Incomplete Data. When a packet is counted in a router, we only find out where it was headed, not where it came from. If a router connects two similarly-loaded networks to a backbone, there is no way to determine, from the routing table data, which traffic onto the backbone belongs to which leaf network. If the two networks actually talk to completely disjoint sets of networks, the data will not reflect it. This inaccuracy was acceptable in our studies because almost all routers had only one loaded network connected; the others were essentially idle. In addition, we accepted the imprecise routing matrix, where all leaf networks connected to a router were lumped together. This completely avoids the problem.

Multiply-Counted Packets. If a packet passes through a longer path more than just the two backbone routers that are between most leaf networks, it will be counted

additional times at any additional routers. In the Stanford internet, there were only three routers not connected directly to the backbone that would cause packets to be counted multiple times. Two of those routers could not be monitored, and the other was essentially idle.

Lost Data. Route resets cause accumulated packet counts to be thrown away. A solution to this would be for the router to keep all entries and simply mark them as bad. This increases the amount of trash in the routing table, especially in a large, dynamic internet.

No Size Information. Obviously, only packets, not bytes, are counted. Any system, independent of its other attributes, need only keep a byte counter in addition to the packet counter to report actual character throughput. Because this solution can be applied to any monitoring system, it will not be discussed any further.

The "perfect" solution would be to have the routers keep statistics independently of the routing table. Rather than a single list of packet counts, the router would keep one list of counts for each directly-connected network (*i.e.* one per interface). The table would be an matrix, with equal to the number of interfaces and equal to the size of the routing table.

When a packet arrives at the router, its incoming interface is noted and the source address is checked to see if it originated on the directly-connected network. If it did, then it will be counted when it is routed toward its destination. The counter will be incremented. A routing matrix over the entire internet is obtained by accumulating statistics from all of the routers. This scheme avoids multiple counting because packets are only counted when they arrive at the first router in their path. It gets more complete data because it differentiates between source networks when it counts packets.

The obvious disadvantage is that this scheme adds additional work for the router, which may not be worth the knowledge gained. An alternative will be available shortly when routers that collect detailed accounting information are introduced. One vendor's latest

router[5], at least, collects packet and byte counts on a source/destination pair basis. This data can be fed into a variation of the above scheme and be used to accumulate a source-destination matrix over the entire internet.

4.4 Conclusions

Both the annotated network map and the source-destination matrix provide valuable information about traffic flow in an internetwork. They provide slightly different perspectives, however. The annotated network map provides a picture of how the internet, in its current configuration, is utilized. The traffic matrix shows where the traffic sources and destinations are in the internet. In essence, it explains *why* the traffic flows as it does in the map. The distribution of traffic sources and sinks across the internet is not uniform — a few networks account for most of the traffic and many networks are essentially idle. Surprisingly, this distribution is very smooth; almost exponential.

[5]This software version was only recently released, and is not yet in operation at Stanford, so it was not used to collect the statistics reported here. The work would have considerably easier had it been available.

Chapter 5

Conclusions

The measurements described in this dissertation have added a great deal to our understanding of the behavior of the Stanford internetwork, as well as local-area internetworks and their components in general.

Delay through a local-area internetwork is determined almost entirely by the delay experienced by packets going through routers. Most interestingly, Ethernet contention delay, which has been the subject of much research in the past, was not a significant factor in the overall end-to-end delay. We found that the largest component of end-to-end delay was delay through routers, a topic that has received little research. When we measured the router, we found that delay was considerably higher than predicted by a single-service center open queuing network model. Queuing delay, caused by bursty packet arrivals, was not the major cause of this delay, however. This result was surprising, since other research has shown network traffic to be highly bursty, which would be expected to cause more queuing than predicted by the queuing network model, which assumes a Poisson distribution. What we found was that packet arrivals, although not Poisson, do not occur in dense enough bursts to cause excessive queuing.

The discrepancy between actual and predicted delay was caused by size-dependent tasks in the router — long packets could take three times as long to process as short packets.

In addition, we found that average packet size varied by factors of 4 or 5 over short intervals. These two facts combine to violate a key assumption of the queuing network model. The simulation, however, produced very accurate results for all of the routers tested when we incorporated the size-dependent delay.

The topology of the Stanford internetwork has changed considerably since its beginnings as a loosely-connected mesh of networks. It has grown much larger in all ways: number of connected subnetworks, number of hosts, and utilization. Because of this increased usage and users' dependency upon the network for vital services, network reliability is essential. In the early days, a cross-campus link might be down for a few hours without much complaint. There were few users, and most were appreciative of whatever service was available — even with outages, it was far better than previous methods, such as carrying a tape across campus. Now, however, with coursework, research, and administration dependent upon the proper functioning of the network, reliability is very important. Users usually notice outages and report them within a few minutes.

We devised a set of monitoring and analysis tools to study the topology of the Stanford internet. We monitored the topology of the Stanford internetwork for 18 months, as it was restructured from its initial *ad hoc* design to a more structured design incorporating a redundant backbone. We analyzed the impact of each of the changes on the robustness of the internet and found that, while most changes were innocuous, some had a profound impact on robustness. During this transition, some redundant links were temporarily removed and not replaced for several months, greatly increasing vulnerability to single-point failures.

Surprisingly, the backbone topology, although larger and more structured, is not significantly more robust than the old topology. There are still network segments and routers that will disconnect large sections of the network if they fail.

As more people use the Stanford network, it has become more difficult to make categorical claims about who's using what — which users, located where, are using what resources. It is important to know where the traffic is concentrated in the internet to plan for future needs. We used two techniques to measure and display the traffic patterns. First, we

measured packet rates on the links connecting routers and networks. By annotating a topological map with these packet rates, they give a very clear picture of where traffic is concentrated in the internet. Unfortunately, it is usually not possible to determine ultimate sources and destinations because the traffic blends together in shared links, especially the backbone. Second, we sampled routing table usage counts over the entire internet to construct a source-destination matrix. This method clearly shows the endpoints of traffic, whereas the first shows the paths used.

These measurements have greatly enhanced our understanding of how the Stanford internetwork is used. With this knowledge, we can answer such questions as: where is additional capacity needed; are any routers operating at or near capacity; where are potential trouble spots in the network; and where are network users concentrated and what network services do they use.

5.1 Future Work

Additional work is still needed in all of these areas. Especially, the data gathering tools need to be improved to make them more flexible and easier to use. This is a broad topic, which I will discuss in the next section.

Additional research should be done to test other routers' delay characteristics against the model used in Chapter 2, especially routers employing other architectures. Most routers have multiple CPUs, but employ them differently. Examples include:

I/O processors on interface cards that handle low-level operations.

One or more additional processors to handle ”overhead” tasks such as routing table update or local communications such as network management traffic.

Processors dedicated to each interface that actually handle routing tasks, requiring little intervention by the central processor.

As noted in Chapter 2, faster measuring tools are needed to get accurate results with the current generation of high-speed routers.

Other router measurement work has been done[Bra89], but has been concerned with measuring the limits of router performance, such as maximum throughput under various conditions, responses to overload, and maximum number of back-to-back packets permitted without dropping any. These measurements are very useful for comparing routers, but differ from our experiments in two significant ways. First, they use artificially-generated traffic, which may not be representative of real traffic. In our experiments, the variation in average size over time was the key to why delay differed so much from the simple model's prediction. Second, the results are just measurements; no attempt is made to define an underlying model that would be useful for predicting performance in other situations. On the other hand, Bradner has measured routers from four different manufacturers; we've only measured two routers from one. The reason for this is that it is difficult and risky to bring up a new and different router in an operational environment (with real traffic), just to make measurements. Router configuration is often tricky, and not all routers support the same protocols. A common problem is that routing protocols are not standardized (yet), so many companies use a proprietary protocol.

Traffic pattern measurements could benefit from better data gathering tools, but the results are mostly complete. What is really lacking are good visual presentations of the data. The annotated map does a nice job as far as it goes, but, in its current form, cannot be used for showing the sources and destinations of the traffic. The routing matrix shows a lot of data, but is hard to interpret. Data can be extracted from it, but it doesn't present a clear overall picture like the map.

5.2 Network Monitoring Lessons

Much of the work described in this dissertation was dependent upon the ability to gather, process, and present data from devices on the internetwork. These three activities constitute what I refer to as *network monitoring*, and each needs improvement. Some improvements require better tools, which must be well-designed to suit the task — they can be thought of as refinements to previous work. Other improvements, especially in the data-presentation area, require creative innovation to devise a better system. Network monitoring is a portion of the considerably broader activity of *network management*. Many of the issues discussed here are also relevant to network management.

For each of these three activities, I describe how I did network monitoring for this research and discuss how it can be improved.

5.2.1 Data Gathering

I obtained the data presented in this thesis by four methods. Each required a significant programming effort, with considerable repetition of some work.

Routing tables for topological maps were obtained by sending routing table requests to the routers and using an expanding-ring search[Bog83, page 74] to find all routers in the internet.

Router delay was measured using the measurement system described in Chapter 2.

All other router data — interface, routing table, and process statistics — were obtained by using the remote console mechanism built into the routers, sending a script of requests, recording the responses, and extracting the interesting data.

Bridge data were obtained using the bridges' native management protocol. This would seem to be an ideal situation but, in fact, required many contortions to get the data into a usable form. Because the management protocol is proprietary,

queries could only be sent from a computer running an operating system to which I had no access. So, the actual procedure was to arrange with someone on the other computer to set up a periodic task to query the bridges and send the resulting text file to a ”drop box” on my computer. Later, my programs would extract the data from the file.

Except for the router delay measurement system, all of these measurements could have been made using a common network management protocol, if one had existed. As it was, only *ad-hoc* solutions were available, and these were slow and cumbersome, required extensive data conversion between formats, and depended upon knowledge of system quirks.

Recently, there has been considerable interest, both among vendors and standards bodies, in common management protocols and data representation. In the Internet community, these are the High-level Entity Management System (HEMS)[PT87], the Simple Network Management Protocol (SNMP)[CFSD89], and the Common Management Information Services/Protocol over TCP/IP (CMOT)[WB89]. Within the last year, SNMP support has appeared in products on the Stanford internet. At the present time, it would be possible to obtain all of the router data except delay via SNMP. It would be interesting to study the relative amount of effort required for such an implementation and the resulting speed improvement over the remote console method.[1]

Not all of the data used in this research is defined in the standard ”Management Information Base” (MIB) used by both SNMP and CMOT. In particular, the process statistics and routing statistics are only available as ”vendor-specific” data. This is not surprising, since these data are tightly coupled to the router’s architecture. The implication of this is that other vendors may keep different statistics. Even if the statistics are directly comparable, they may not be available through the management protocol, and even then they will

[1]The remote console scheme requires several programs running in concert to send the script to each router in turn, decode the resulting trace, and massage the data into a usable form. Querying all 30+ routers typically takes 4-5 minutes, with the programs running full tilt. In contrast, the expanding-ring search, which operates on a much lower level, takes about 2 minutes, with most of that time spent waiting to be sure that all replies have been received.

almost certainly have different ”names” in the vendor-specific area of the MIB.

5.2.2 Data Processing and Storage

Once the raw data has been collected, it must be stored and processed. All of our methods followed the following general pattern:

1. Collect data from the network to form a snapshot of the network state at the current time.
2. Subtract the new data from the previously-recorded network state to obtain the data over the most recent interval. Usually, the difference between counter values is divided by the elapsed time to get a rate.
3. Update the ”current state” with the new snapshot. Generally, this is simply replacing an old snapshot with the new one. However, if it is necessary to deal with missing data in a snapshot (*e.g.* because a router was down or inaccessible), it is necessary to merge the new snapshot with the stored network state.

Rates over an interval, which were often the primary result of the monitoring, were determined by dividing the difference between snapshots by the interval between them. An important exception happens when, for whatever reason, a counter gets reset — the difference will be meaningless. This would often be caused by a crash and reboot of the entire system, but may be caused more locally by, for example, a routing table entry disappearing and reappearing. All counters have an associated *age* — how long since it started counting — which must be monitored in addition to the counter value itself. Knowing a counter’s associated age allows resets to be detected and provides the time reference needed to obtain a rate from the counter values.

The pattern outlined above was repeated for all of the data collected. This task would have been greatly simplified if a network monitoring ”toolkit” had been available. Such a toolkit should:

Provide a database system for storage and retrieval of network state snapshots and/or the data resulting from subtracting two snapshots. It must support the addition of fields to an existing database, as knowledge about what needs to be monitored is refined.

Provide a ”shell” for doing subtraction and state updates. This shell might even drive the programs that do the data collection.

Understand the concepts of counters (*e.g.* number of packets transmitted or current time) that may be subtracted, and static data (*e.g.* interface address or process name) that may not be subtracted.

Understand ages and how to handle them. There will certainly be a different age for each monitored entity. There may be several other state variables that must receive similar treatment —

Be able to associate counters with the appropriate ”age” state variable, so that subtractions and rate calculations can be determined automatically from configuration information.

Such a toolkit would make the data processing aspect of network monitoring much easier and more flexible. The added flexibility would, by itself, benefit research by making it easier to add new data to be collected to check out a hunch or deal with a short-term monitoring need.

5.2.3 Data Presentation

Presentation of large volumes data in an understandable form was critical to this research. The packet traces discussed in Chapter 2 were typically between 400 Kbytes and 2.2 Mbytes in size, containing thousands of packet records. Manual inspection of the data was out of the question, and conventional statistical methods of plotting distributions, calculating means, etc. were not going to give any indication of the actual processes that

were going on. Data visualization was critical to be able to make sense out of this volume of data. There are two types of visualization used in this dissertation. Both produce a meaningful condensation of the data, but go about it in different ways.

The first method relies on having a model to test the data against, and using that to condense the data down to quantities that could be directly compared. This is the basis of the analysis and presentations in Chapter 2. There's nothing especially new here. Tools for rapid prototyping and testing of the models were a big help and made it relatively painless to test the models.

The second method relies on innovative data presentation; examples appear in the later part of Chapter 3, Figures 3.11 and 3.12 and in Chapter 4. The emphasis here is in incorporating large amounts of data into a single, meaningful graphic, without simply jamming a bunch of data together. Tufte[Tuf83, Chapters 1,7] gives many examples of such graphics as well as guidelines for constructing them. Network management tools, in general, deal with large quantities of data and could benefit greatly from well thought-out graphical display methods. The need is especially acute in network management workstations where, besides needing lots of data clearly visible, anomalous data must stand out clearly.

Bibliography

[AL79] Guy T. Almes and Edward D. Lazowska. The behavior of ethernet-like computer communications networks. In *Proc. 7th Symposium on Operating Systems Principles*, pages 66–81, 1979.

[Arm88] Caroline Arms, editor. *Campus Networking Strategies*. Digital Press, 1988.

[BH88] Leonard Bosack and Charles Hedrick. Problems in large lans. *IEEE Network*, 2(1):49–56, January 1988.

[BMK88] David R. Boggs, Jeffrey C. Mogul, and Christopher A. Kent. Measured capacity of an ethernet: Myths and reality. In *Proc. SIGCOMM '88*, pages 222–233, Stanford, California, August 1988.

[Bog83] David Reeves Boggs. *Internet Broadcasting*. PhD thesis, Stanford University, 1983. Also published as Xerox Techinchal Report CSL-83-3.

[Bra89] Scott Bradner. Tcp/ip router tests, v2.5. Electronic Document; FTP from husc6.Harvard.EDU in pub/rtests; see 'README' and 'posting-tcp-ip' files., 1989. This is ongoing research and will probably be updated and eventually published.

[CFSD89] J.D. Case, M. Fedor, M.L. Schoffstall, and C. Davin. Simple network management protocol (snmp). RFC 1098, Network Information Center, SRI International, April 1989.

[DKS89] Alan Demers, Srinivasan Keshav, and Scott Shenker. Analysis and simulation of a fair queuing algorithm. In *Proc. SIGCOMM '89*, Austin, Texas, September 1989.

[Gon85] Timothy A. Gonsalves. Performance characteristics of 2 ethernets: an experimental study. *Communications of the ACM*, 28(5):78–86, May 1985.

[HMC 86] Charles Hedrick, Andrew Malis, Vint Cerf, Jack Haverty, Ken Pogran, and *et alia*. Arpanet outage. In "TCP-IP" mailing list, tcp-ip@sri-nic.arpa, 1986. Archive available from sri-nic.arpa in file TS:¡TCP-IP¿tcp-ip.1986D, under subject "Arpanet outage", beginning 15 Dec 1986.

[Jac88] Van Jacobson. Congestion avoidance and control. In *Proc. SIGCOMM '88*, pages 314–329, Stanford, California, August 1988.

[JR85] Raj Jain and Shawn Routhier. Packet trains: Measurements and a new model for computer network traffic. MIT/LCS TM-292, Laboratory for Computer Science, Massachusetts Institute of Technology, November 1985.

[Llo86] Peter John Lloyd. *End-To-End Protocol Performance Over Linked Local Area Networks*. PhD thesis, University College London, November 1986. Also available as UCL-CS TR 123.

[LZGS84] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. *Quantitative System Performance*. Prentice-Hall, Englewood Cliffs, New Jersey, 1984. Computer Systems Analysis Using Queuing Network Models.

[MB76] R.M. Metcalfe and D.R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976. Also CSL-75-7, Xerox Palo Alto Research Center, reprinted in CSL-80-2.

[MKU87] Robert Michaels, Norman Kincl, and Walter Underwood. Designing and installing an international tcp/ip network. STL 87-19, Hewlett-Packard Laboratories, December 1987.

[Par89] Craig Partridge. How slow is one gigabit per second? Report 7080, BBN Systems and Technologies Corporation, June 1989.

[PDA86] *et. al.* Paul D. Amer. Local area broadcast network measurement: Traffic characterization. Technical Report 86-12, University of Delaware Dept. of Computer and Information Sciences, January 1986.

[PT87] Craig Partridge and Glenn Trewitt. High-level entity management system (hems). RFC 1021, Network Information Center, SRI International, October 1987.

[RJ88] K. K. Ramakrishnan and Raj Jain. A binary feedback scheme for congestion avoidance in computer networks with a connectionless network layer. In *Proc. SIGCOMM '88*, pages 303–313, Stanford, California, August 1988.

[SH80] John F. Shoch and Jon A. Hupp. Measured performance of an ethernet local network. *Communications of the ACM*, 23(12):711–721, December 1980. Also in CSL-80-2, Xerox Palo Alto Research Center.

[TH80] F. A. Tobagi and V. B. Hunt. Performance analysis of carrier sense multiple access with collision detection. *Computer Networks*, 4(5):245–259, Oct/Nov 1980.

[Tuf83] Edward R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 1983.

[WB89] U. Warrier and L. Besaw. Common management information services and protocol over tcp/ip. RFC 1095, Network Information Center, SRI International, April 1989.