Designing Networks for Tomorrow's Traffic

by

Henry H. Houh

S.M.E.E. Massachusetts Institute of Technology (1991)
B.S.Phy. Massachusetts Institute of Technology (1990)
B.S.E.E. Massachusetts Institute of Technology (1989)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1998

Signature of Author .....
Department of Electrical Engineering and Computer Science
January 26, 1998

Certified by .....
David L. Tennenhouse
Senior Research Scientist
Thesis Supervisor

Certified by .....
John V. Guttag
Professor and Associate Head, Computer Science and Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Designing Networks for Tomorrow's Traffic

by

Henry H. Houh

Submitted to the Department of Electrical Engineering and Computer Science
on January 26, 1998, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Today, information transfer on the Internet has rapidly moved to a client/server model. This shift has happened rather dramatically. But, descriptions and thought processes regarding traffic flows and patterns within the Internet have not shifted as rapidly; they have been mired in terminology reflecting the old peer-to-peer model. All prior analyses of Internet traffic use symmetric endpoint flows to describe traffic patterns. Advanced flow-routing methods such as IP switching and flow switching still base their techniques on the identification of symmetric endpoint flows.

In this thesis, we examine the asymmetries of network traffic, which are particularly severe near servers and identify asymmetric endpoint flows as a means of dealing with this traffic. We design a network infrastructure with the flexibility to integrate new flow switching techniques with our new asymmetric endpoint flows. Our challenges are to achieve better methods of flow classification and reduction in the capture delay for packets, i.e., the time taken to recognize packets as part of an existing flow. We demonstrate that asymmetric endpoint flows address both these challenges.

In order to analyze the capacity gains realized from integrating asymmetric endpoint flows, we create a model for IP switching which uses a small number of variables. We simulate IP switching using real network traces of wide area and gateway traffic to show the increase in routing capacity realized. We have also built a prototype system that implements the core concepts.

We use our model to optimize the parameters of the IP switch to achieve optimal performance. In an efficient IP switch implementation, our asymmetric endpoint flows and our techniques of IP switch optimization can produce a 24-fold increase in routing capacity for gateway traffic.

Thesis Supervisor: David L. Tennenhouse
Title: Senior Research Scientist

Thesis Supervisor: John V. Guttag
Title: Professor and Associate Head, Computer Science and Engineering

Acknowledgments

I gratefully acknowledge the contributions of the following people and organizations without whose support this thesis would have been impossible:

- David Tennenhouse for all his years of guidance, inspiration, help, support and patience.
- John Guttag for his support and guidance in helping me to collect my thoughts together and for his help in pulling together the final document.
- Thesis reader Steve Ward, for his help and support throughout my undergraduate and graduate years.
- Thesis reader Anant Agarwal for his helpful and insightful comments and suggestions.
- All past and present members of the Telemedia Networks and Systems Group, now Software Devices and Systems. They have made it a fun experience to be at MIT for the past 8 years. I especially would like to thank Joel Adam, my former officemate, for fun times during the VuNet work and for introducing me to my wife Lisa; Chris Lindblad for being a great resource and for his helpful discussions; Bill Stasior for lots of late-night discussions; Vanu Bose, Mike Ismert and David Wetherall have all been extremely helpful; Ulana Legedza for helpful discussions and for taking over my group support responsibilities.
- All my friends who encouraged me over the years.
- My best friend Derek Chiou for reading my thesis and for being there in good times and bad.
- My sister Emily.
- My parents for their continuing support of my education and for giving me endless opportunities.
- My wife Lisa for her understanding, comfort, support and patience, who is as happy as I am for finally completing my thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In just a few years, the Internet has become widely accepted as a medium for information delivery. This has lead to tremendous growth in the number of users and the amount of data they transmit and receive. It has also lead to the introduction of new types of real-time services not available over the network just a few years ago.

This traffic presents problems to current connectionless IP routed networks as the number of packets begins to exceed network routing capacity. In IP networks, routers must determine on a hop-by-hop basis the proper next hop. These routers incur a minimum fixed overhead cost for each packet processed and forwarded, limiting their performance. Some service providers that host large servers already have had to make alternative arrangements for receiving traffic, as the main routers serving them can handle no more traffic [Week, 1997].

New methods of designing networks can lead to much greater capacity. High-speed switches working in conjunction with IP routers can provide fast paths that bypass per-packet processing by the routers. Recognizing that servers aggregate tremendous amounts of traffic and providing special network circuits for them can also reduce the amount of overhead incurred for delivering this traffic.

In this thesis, we design and implement a flexible connection-oriented network with the ability to manage connections from attached hosts. We take advantage of this flexibility to incorporate various types of high speed network services – video to the application, static meshed workgroup connections, and even connectionless services – on top of our network. By further utilizing our flexibility, we extend the IP over ATM model to designate and switch special server based flows which greatly reduces the amount of resources devoted to providing IP routed services.

1.1 Traffic Growing Rapidly

The Internet began in the late 1960's as the ARPANET, and by July 1997, it was estimated that almost 20,000,000 computers worldwide were connected [Zakon, 1997]. The early 1990's marked a turning point for the Internet, which rapidly changed from a network utilized by educators, students and researchers to a mass-market medium.

The World-Wide Web (WWW) has been a primary driving force behind this transformation; the clients, or browsers, provide the user interface to the growing amounts of information available on the Internet. Today, Web traffic is dominated by text, image, and file retrieval, but we are beginning to see real-time audio and video. As this multimedia traffic makes its way into the wide area, it is becoming a rapidly growing burden on backbone networks. In many cases, the traffic growth has outstripped the pace of increase in network and router capacity. These problems must be addressed before today's rapidly growing multimedia data delivery can be well handled by the network backbones.

1.2 Today's Traffic: Server Based

Network data is also increasingly being delivered by servers, since the primary mode of information retrieval on the Internet is the use of a WWW client.

Data taken from gateways and backbones indicate that roughly 50-55% of all packets are Hypertext Transport Protocol (HTTP, or WWW) packets, and in some wide area traces, almost 25% of all packets are Domain Name System (DNS) packets. In both cases, these packets are destined to or generated by WWW and DNS servers. The number of WWW servers has increased from 163 in 1993 to an estimated 1,269,800 in August 1997, while the number of domains have grown from 3,900 in 1989 to 1,301,000 in July 1997.

The data flow to WWW servers is highly asymmetric since the servers are primarily delivering information to the clients. The number of packets flowing in each direction is roughly on par, since TCP requires an acknowledgment packet for every data segment sent so that sender can release the buffer and send more data. This leads to a high degree of asymmetry in the number of bytes per packet flowing away from the server as opposed to flowing toward the server. Since the work in routing packets toward and away from a server is roughly equivalent for any given TCP session, the amount of work done by the network is much greater per byte – by a factor of 10 to 1 or more – for the little amount of information flowing toward the server.

For example, in a sample web browsing session at www.cnn.com that retrieved the main page and a roughly six news stories, 676 packets were exchanged between the client and the server. 373 packets containing 421119 bytes of data (not including protocol headers) were sent from the server to the client, with almost two-thirds of these packets of size 1460

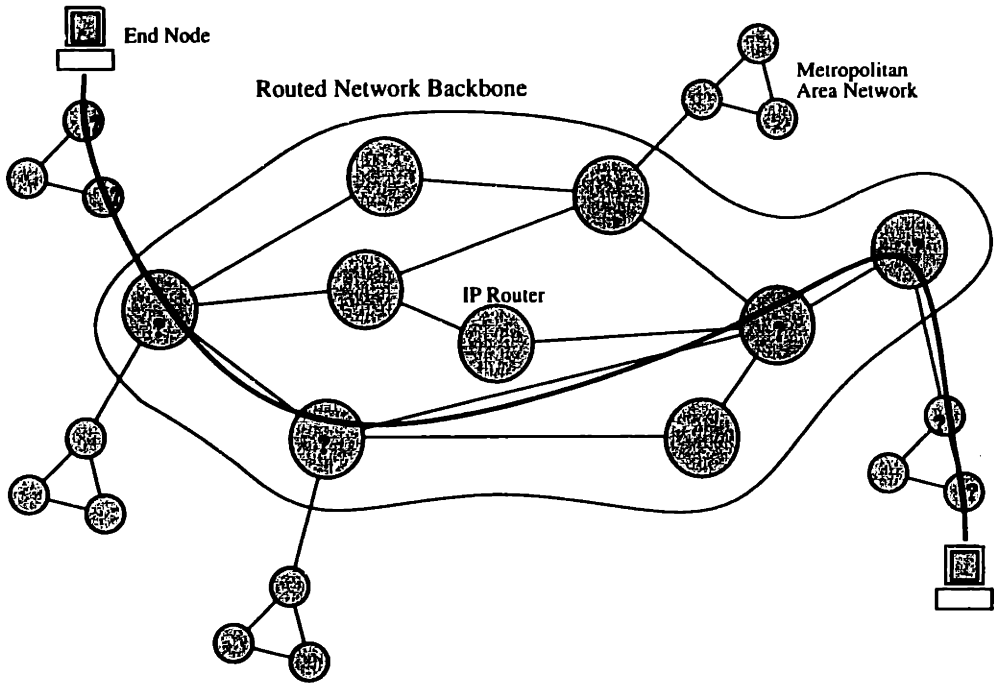

Figure 1-1: High performance routers route individual IP packets throughout a IP network backbone. Packet headers must be examined and acted upon at each router for all packets traversing the backbone.

bytes. 303 packets containing 16881 bytes of data were sent back to the server, with 242 of these being empty protocol acknowledgement packets of no data. The remainder of the packets were smaller than 324 bytes.

1.3 Today's Architecture: IP

Most of today's network is based upon a connectionless IP model. A connectionless model is one in which packets are sent to one or more destinations independently of each other, with each packet's header containing addressing information. IP routers forward packets based on their universal destination addresses. Routers collect the information required for them to determine the best next-hop for any given packet. No unrecoverable connection related hard state resides in the network.

Figure 1-1 diagrams the Internet, where high-performance routers within the network backbone route traffic between Metropolitan Area Networks. Currently, the fastest routers handle roughly 500,000 packets per second, and these backbone routers often strain from the loads presented at peak periods.

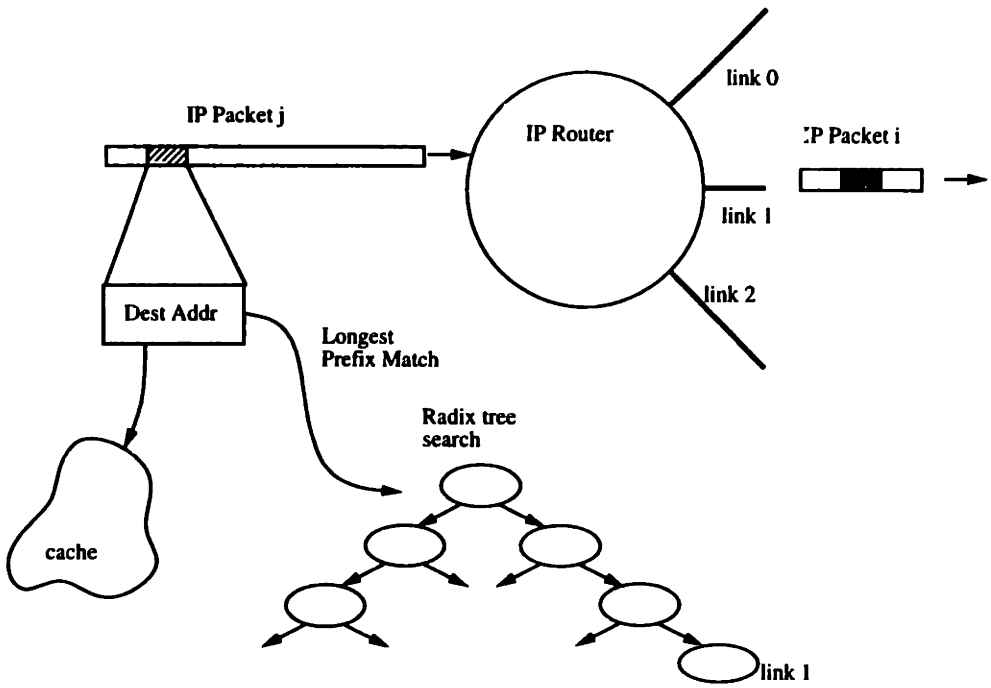

Figure 1-2: Each router between the source and destination must examine the IP header in order to determine the proper outbound link. The routing table lookup uses a radix tree search to find the longest prefix match.

IP routing is time consuming. All routers perform a rather simple function: they determine a specific outbound interface based on the destination IP address contained in every packet. However, because IP addresses are long (32 bits[1]), it has been impractical to provide simple table lookups on IP addresses at network speeds. Routers resort to building much smaller routing tables – tens of thousands of entries as opposed to billions – that are efficiently search-able. In addition, if an exact match is not found, then the "best" match must be returned, with the "best" match defined as the entry having the most number of prefix bits in common with the destination address.

1.3.1 Connectionless vs. Connection-oriented

An IP network is connectionless, that is to say, information contained in each packet determines the destination without relying on any hard state contained within the network. Packets from the same source to the same destination may take different paths through the network, and may even be delivered out-of-order.

[1]IPv6 will move to 128 bit IP addresses.

In a connection-oriented network on the other hand, a specific end-to-end circuit is assigned for communication between two users, and this circuit's setup must take place before any information can be exchanged. This "virtual circuit" determines the route at the time that the circuit is set up that all future packets transmitted on the circuit. All traffic follows the same path and does not get reordered. While this is inefficient for short-lived flows, packets in long-lived flows experience lower latency and need not be processed at every hop. Header sizes may be smaller, as there is no need to add universal address information to every packet once an end-to-end circuit is established.

Both types of networks have their advantages. In a connectionless network, new hosts can be more easily added. Once a host is configured locally, it may communicate globally, without any lengthy delays to set up connections. The overall network is able to recover from internal router failures.

Connection-oriented networks can more easily provide service guarantees. Once a connection is established, packets traverse the network without being examined at every node along the way. Also, since the packets are generally switched at the hardware level, the latency for packets is much lower than in a connectionless network.

On the other hand, both types of networks have their disadvantages. In a connection-oriented network such as the telephone network, when an internal switch fails, a user's call is dropped. The state of the circuit in this case is usually encoded into the switching hardware, and is not generally dynamically re-allocated. And for short lived connections, the circuit setup overhead is quite large compared to the data transfer time. Even long lived flows must wait for circuits to be set up.

Though a connectionless network may transmit packets without lengthy circuit setup, every packet must be examined at every router. This is one of the largest costs of maintaining the connectionless IP network. This involves searching the routing table, which may total to tens of thousands of entries for networks, subnetworks, and hosts. While hosts are often grouped hierarchically into networks, the process of determining the proper outbound interface can be lengthy – even a high end Cisco backbone router can only route 10,000 packets per second when each packet needs a table lookup, while low end routers handle around 1,800 [Kaeo, 1996].

Searching methods have been streamlined to use the best algorithms, and entries are also cached so that packets in a flow that are local in time generally experience less processing delay. However, routers must still process every single packet flowing between machines, even in a flow that is well established. While the first packet may take a relatively long time to route, all succeeding packets still must be processed.

Thus, it is difficult for routers to increase capacity due to the overhead of the basic routing function.

1.4 Tomorrow's Network: A Flexible Hybrid

As traffic continues to grow, current IP networks will become stressed. There are already signs of this, and network pundits have predicted major outages in the near future. Thus a new network design for handling the growth in traffic is needed.

1.4.1 Routers and Switches and Flows, Oh My!

Before continuing, we will define terms that will be used throughout this thesis. The terminology is confusing because we will be describing the joint operation of devices which ordinarily would be functioning in distinct layers within the network.

Our definitions are as follows:

- **Router:** A device which is able to forward a packet from the input port to one output port that guides the packet towards the destination whose address (with global scope) is contained in the packet's header. An IP router is a router for packets with IP protocol headers. The determination of the proper output port generally requires a complex lookup function, and the packet's header remains unchanged.[2] Routers are considered slow devices.
- **Switch:** A device which is able to forward a packet from the input port to one output port that guides the packet towards the destination based on a label (with only local scope) contained in the packet's header. The determination of the proper output port is a simple lookup function, and the label is replaced with a new label. Switches are considered fast devices.
- **Label-Swapping:**[3] The process of replacing an identifier in a packet header with a new identifier, based on a rapid table lookup.
- **Flow Switch:** A general class of new device which is the combination of both a router and a switch operating in conjunction.
- **IP Switch:** A specific type of Flow Switch.
- **IP Switch Router:** The router portion of an IP switch.
- **ATM Switch:** The switch portion of an IP switch. Though an ATM switch is just one specific type of switch, in this thesis, we will be only dealing with IP switches that use ATM switches.

[2]This is not strictly true, but for our purpose it suffices since the addresses in the protocol header remain untouched.

[3]Labels are sometimes referred to as Tags.

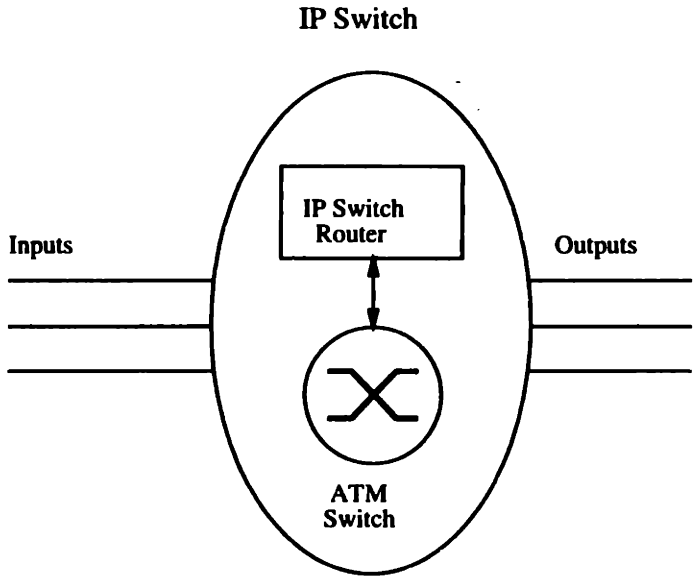

Figure 1-3: An IP switch provides same external functionality of a router. The IP switch consist of two parts, the IP switch router and an ATM switch.

We will also be dealing with network traffic. In order to talk about the characteristics of this traffic, we introduce the following definitions:

- **Host:** A network-connected computer.
- **Endpoint:** Either a specific host or a specific process running on a specific host.
- **Host Endpoint:** An endpoint which is a host.
- **Process Endpoint:** An endpoint which is a process.
- **Flow:** A one-way train of packets traveling from a source endpoint to a destination endpoint.

1.4.2 IP Switching: Connectionless/Connection Hybrid

A flow switch is a new type of network device which have been introduced recently. One type of flow switch is an IP switch [Newman *et al.*, 1996b], which incorporates both a router and a connection-oriented ATM switch, as shown in Figure 1-3. Initially, all packets arriving at the switch are delivered to the IP switch router. The IP switch router performs its standard routing table lookup (as a normal router does) and forwards the packet to the appropriate outbound switch port.

However, when the IP switch router is able to identify a long lived flow, it pushes the IP

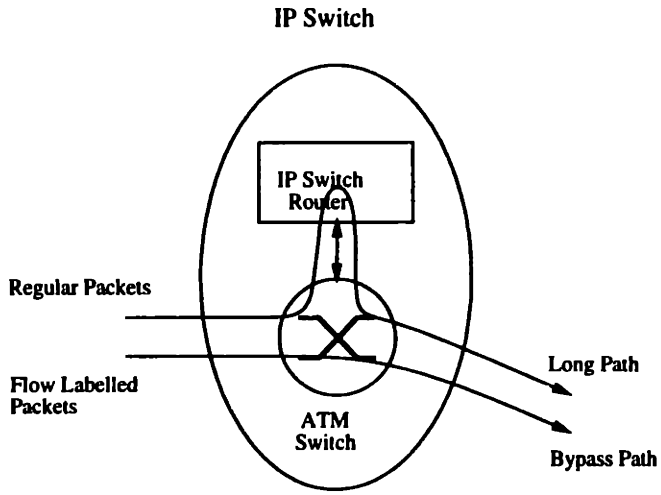

Figure 1-4: Packets on a labeled flow take a bypass path through the ATM switch, thereby avoiding the IP switch router.

routing table lookup result directly into the ATM switch using a newly allocated ATM switch label, thereby configuring a cut-through circuit through its own ATM switch that bypasses the IP switch router. Future packets belonging to the long lived flow bypasses the IP switch router.

The peer IP switch immediately upstream is notified that a new circuit exists on the new label for packets within this flow, and begins to forward packets on this flow to the downstream IP switch using the label allocated. So, as shown in Figure 1-4, packets in identified flows are able to take a fast path through the ATM switch, thereby bypassing any processing incurred in the IP switch router once the connection is set up.

Unlike a normal router, the routing decision is no longer cached in the IP switch router, but is cached in the ATM switching hardware. This is possible because the number of label entries in the ATM switch's lookup table is much smaller than the IP address space. Label-swapping, a fast mapping from the switch circuit identifier to an output port and new identifier, is thus a simple direct-mapped table lookup, as opposed to a lengthier tree search.

An IP switched network thus provides the routing capabilities of an IP routed network while eliminating the need to process all the packets. It dynamically allocates direct switched connections to packets that match the characteristics of a long-lived session. It combines the strengths of connectionless and connection-oriented networks while addressing the weaknesses of each. By combining an IP switch router with an ATM switch, it provides a means to dynamically recover from switch failures while taking advantage of switching hardware while eliminating individual processing of IP packets for long lived connections.

These new IP switches claim to be able to handle many times the capacity of IP routers. However, the types of flows they can handle are limited to those which connect specific machines.

Flexibility in this approach is necessary so that introducing methods of detecting and aggregating flows can lead to even greater overall capacity of an IP switched network.

1.4.3 Challenges to IP Switching

In a IP switched network, long lived or other specific types of flows would be switched directly. However the identification of such flows and how soon the circuit is set up determines how efficient the process can become. These are two of the challenges that we address in this thesis: flow identification and flow capture delay.

Flow identification is the process by which information contained in a packet used to classify it into some flow, and **flow capture delay** is the amount of time necessary to make the decision to switch a flow.

In the current IP switching model, flows are defined with symmetric endpoints, as traffic from one host endpoint to another host endpoint (**Machine flow**), or from one process endpoint to another process endpoint (**TCP flow**[4]).

The difference between these two types of symmetric endpoint flows is that any number of TCP flows might be subsets of a Machine flow. However, providing flows for processes allows separate service guarantees since they can be distinguished from other kinds of traffic between the two hosts.

Using Machine flows to identify traffic, though, eliminates the flow capture delay of successive TCP connections since a Machine flow will already be in place after the first TCP connection from the same source host endpoint is initiated.

When accessing a web server, for example, one Machine flow would be set up to deliver packets to the client, and another separate Machine flow would be set up to deliver packets from the client to the server. The web session example shown in Section 1.2 demonstrated the high degree asymmetry of actual data transfer on the forward and reverse flows.

This suggests that the symmetric Machine and TCP flows are might not be the best method of identifying packets.

[4]This is somewhat of a misnomer, as not all processes communicate using TCP, but it we use it because it conveys the right idea.

1.4.4 An Asymmetric Endpoint Flow Approach

The asymmetry mentioned above and the aggregation of traffic around servers suggests that we might deal with traffic flowing to a server and away from a server in a different manner, by using flows which have asymmetric endpoints.

Thus we introduce two new types of asymmetric endpoint flows: the **Session flow**, which is identified by a source (server) process endpoint and destination host endpoint, and a **Server flow**, which is identified only by a destination (server) process endpoint.[5]

A Session flow behaves much like a Machine flow: it eliminates the capture delay of all TCP connections after the first. It also is a subset of a Machine flow. However, the Session flow retains many good characteristics that are lost with Machine flows, as will be discussed in Section 4.1.3. We hypothesize that Session flows will provide roughly the same performance increase as Server flows.

Let us separate the notion of a Server flow from the instantiation of a Server flow. The notion of a Server flow is that all packets from any host, located anywhere, destined for a server process endpoint is considered to be in the same Server flow. So, the Server flow itself has a global scope, much like the IP address and TCP port number that identifies the server globally.

The difference between the globally allocated Server flow and the globally unique server identifier, though, is that packets in the Server flow can take a fast path through the entire network, while IP packets must still be routed individually.

In a single instance of a Server flow, different flow labels will be allocated to different IP switches. These labels have a local scope identifying the next hop towards the server, and through label-swapping, define a global flow. Packets merge into a single label as they flow towards a server, and the labels change from hop to hop.

All packets within the network destined for this server are eventually multiplexed at the hardware level onto a single Server flow – a type of reverse-multicast tree. Many fewer connection setups are required to handle all this inbound traffic. The Server flows reduce the capture delay by allowing any other host's packet to initiate a Server flow into which a packet can fuse. The packet is captured into a Server flow when it joins up at an IP switch which has the same Server flow.

Server flows allow the aggregation of traffic bound for servers into a single switched circuit tree rooted at the server. If a large web server receiving 100 million hits/day generates 20 packets/hit, then 2 billion packets are going to or from the server. All these packets are handled by all the routers between the server and the clients. If we assume that 40% of the packets are inbound, then 800 million packets would be handled by a single inbound Server

[5]We realize that this is beyond our strict definition of a flow. However, we broaden the types of endpoints here to include the multiple unspecified source endpoints of a Server flow.

flow. This would be much more efficient than generating separate circuit setups for inbound TCP or Machine flows from millions of different hosts. Also, since Server flows reduce the amount of flow resources needed to track flows within the network, they help avoid label "thrashing" due to limits of label space.

Broadening the types of flows to include asymmetric endpoint flows leads to better choices for flow identification, and both Server and Session flows fuse TCP connections to reduce the capture delay. Session flows fuse TCP connections of specific server types between hosts, and Server flows fuse TCP connections of specific server types both between hosts and laterally across hosts.

We hypothesize that our asymmetric endpoint flows will produce better performance than symmetric flows.

Asymmetric endpoint flows in an IP switched environment enables better use of resources, leading to greater network capacity. We implement this network using flexible components that are managed from attached hosts. We provide an analysis of the work to be done by the network routers with and without the Server flows, showing that Server flows increase the capacity of an IP switched network.

1.5 Summary of Major Contributions

The main contributions of this thesis have to deal with using asymmetric endpoint flows in an IP switching platform. We will provide new insights and analysis on: the identification of asymmetric traffic patterns, the implementation of an IP switched system, the development of a work model of IP switching, and the detailed analysis of wide-area and gateway network traces.

We summarize our contributions as follows:

- **New Traffic Analysis** We identify and characterize asymmetries of network traffic by using parameters that others have not scrutinized.
- **Design of a Flexible ATM Infrastructure (VuNet):** We have designed an ATM network infrastructure in which functions such as connection management and packet segmentation and reassembly are cleanly decoupled from the switch itself, providing a great deal of flexibility. The VuNet provides direct, high bandwidth network circuits for user applications. The flexibility allows multiple co-existing approaches to connection management and integration with existing protocols and allows us to respond to changing traffic types and traffic patterns.
- **Integration of Asymmetric Endpoint IP Switched Flows into the VuNet:** Using the flexibility and unique features designed into the VuNet components, we

incorporate an IP switching implementation and provide a proof-of-concept of asymmetric endpoint flows into our system.

- **Development of an IP Switching Work Model:** In order to quantify the capacity gains made by introducing IP switching with symmetric endpoint and asymmetric endpoint flows, we develop a model for IP switching that reduces the variables to a manageable number. We also show how to derive all necessary parameter for a detailed analysis from network traces.
- **Detailed Analysis of Flow Traces:** Using the model we develop, we analyze several network traces. We show how to maximize performance by optimizing the flow switching threshold, and show how this switching threshold can be tuned for different types of network services. We also provide a comparison to the best possible performance increase.

1.6 Roadmap

In the next Chapter, we discuss the various factors that have driven recent traffic growth. We identify and characterize the large byte asymmetry in the new client-to-server data flow. Using our analysis of existing network traces, we identify opportunities for gaining more benefit from a flexibly designed IP switched network.

Chapter 3 lays out the various components of the VuNet. We lay out our design goals and describe the various VuNet components. The network components are designed only for bit transport, and all the network devices are designed with hardware simplicity in mind. We discuss how we push complex functions into the attached hosts at the edge of the network. This enables a great deal of flexibility through host software which we use in our asymmetric endpoint flow implementation.

Chapter 4 discusses IP switching in more detail, and shows how we use VuNet features to integrate IP switching into the VuNet. It is also important to have a high degree of flexibility in identifying flows, as this flexibility provides the largest opportunity for better flow identification and reducing capture delays. Section 4.2 discusses our new asymmetric endpoint flows and the services which will benefit from their addition.

In Chapter 5, we discuss our work model for IP switching and IP routing and our network trace simulations. Section 5.2, we develop a work model for IP switching, in which a small number of parameters can be used to determine the cost savings. We then discuss the simplifications to our model to reduce the number of parameters needed to a manageable level. Section 5.5 details the simulation results showing performance increases and shows how to adjust parameters to achieve optimum performance.

Finally, we conclude with some observations and suggestions for future work.

Chapter 2

Understanding Traffic Patterns

In this chapter, we set forth many of the factors causing rapid traffic growth in the Internet. Many of these are apparent to those who have followed the Internet over the past few years. However, one must dig beneath the obvious to discover interesting information about the impact of this growth – particularly how it affects packet sizes and data in flows within local and wide area networks. Network planners would be better off in planning future expansion if they had components available that could take advantage of this information.

We examine recent network traces and compare these to existing analyses, to determine how today's traffic is distributed among various services. Furthermore, we examine the directionality of packet flows to study the byte asymmetry of most data flows, particularly around servers. Later, we use this information to show how bundling many of these data flows can greatly reduce the work done by the network and increase its routing capacity.

2.1 Factors Driving Traffic Growth

Many factors have driven the rapid growth of network traffic in recent years. The largest increases can be tied to the incredible growth in the number of users. In addition, new applications and services enable these users to download more and more data during their sessions. We discuss these and other factors here.

2.1.1 Growth in the Overall Number of Users

The rapid growth in the number of users followed the introduction of the World-Wide Web. Prior to this time, for most people, retrieving data from the Internet required using text-based command-line interfaces. Thus, it was tedious to retrieve information from a number

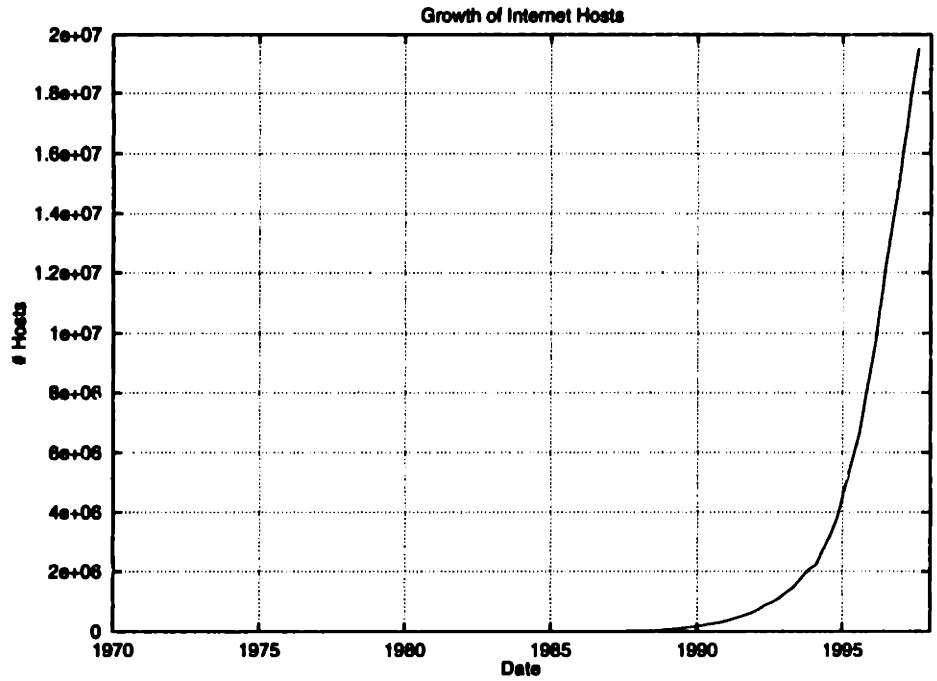

Figure 2-1: The number of Internet hosts has grown explosively since the introduce of the World-Wide Web in 1991. Source: The Internet Society

of different sites. The introduction of hypertext and a graphical interface to network-based documents made these information sources much easier to use.

Figure 2-1 shows the growth of the attached hosts to the Internet since its inception. Over the past 10 years, this growth has averaged around 90% per year. On today's scale, the hosts in the first two decades of existence are barely even visible.

In 1997, it was estimated that the number of hosts approached 20 million. In many cases, several to hundreds of users share a host. The number of actual Internet users in 1997 was estimated at 50 million users over the age of 16 in the US and Canada alone [Research, 1997]. The growth in the number of users has followed the growth curve of Internet hosts.

Most of this growth has come from a booming PC market. This market is driven by both business and home sales. Since 70% of Internet users are estimated to access the Internet from home, growth in home PC sales will have a bigger effect on overall traffic due to a compounding of effects described in the next section.

2.1.2 Effect of Home Users on Internet Usage

Multiple factors acting on home markets will greatly affect the amount of traffic on the Internet. The number of Internet-capable home PCs has been growing at a high rate. Also,

the access speeds available from homes is increasing rapidly with the introduction of faster modems and cable modem services. The rate of increase in the number of users outside of the US is also estimated to be much higher than US user growth. Finally, as users get high-speed download capability, they will begin to download larger amounts of data.

Growth in the Number of Home PCs

The home PC industry, as measured in units, continues to grow at rates exceeding 10% annually [McCarthy and Bernoff, 1996], with PCs in 38% of all US households in October 1996; the penetration is expected to grow to 53% of households by 2001. Also, modem penetration in these homes is expected to grow from 30% of the computer households in 1994 to 57% of the computer households by 1999 [Eland, 1996b], an average rate of 16%. Combined, this represents an average rate of increase of Internet-capable home computers of roughly 28% annually.

Also, while users in the US represent 73.5% of all WWW users [Kehoe and Pitkow, 1996] in 1996, it is estimated that US users will represent only 50% of all users by the year 2000 [Eland, 1996a]. This means that the growth in number of users outside the US – estimated at 42% annually – will exceed by 50% the growth in the number of US users.

Increase in Download Bandwidth from the Home

Home download rates are expected to increase in the next few years as better services are deployed. As of mid-1996, 40% of all network users connected via 28.8k modems, 30% with 14.4k modems, and the remaining 30% connected with T-1 or faster lines. Presumably, almost all modem users are home users. 56k modems have become available since this time, but the widespread deployment of these modems has been slowed by a lack of standard.

One recent service introduction that may prove popular is cable modem access. These cable modems provide 1.5 Mbit/sec download and 300 Kbit/sec upload speeds, with pricing about twice the cost of standard Internet access. Recent reports indicate that number of users doubled from July 1997 to October 1997, and is expected to double again by January 1998 [Pelline, 1997]. While this may represent just 200,000 users in 1998, by 2000 there are expected to be 1.6 million users – an annual growth rate of 183% until that time. The number of cable modem users is expected to double to 3.2 million by 2002, with growth slowing to just 41% per annum.

A study conducted in August 1997 measured the average page download times from web sites on 34 Internet backbones to 30 measurement locations [Keynote, 1997]. It was found that the average download speed was 40,000 b/s. It was also suggested that this was a limitation of the network, and not the servers. As users rapidly adopt access methods whose bandwidth exceeds this average download speed, the Internet will become more strained.

Growth in Retrieved Object Sizes and Number of Pages Retrieved

As higher speed services are adopted by the tens of millions of home users, the amount of information downloaded is bound to go up. For example, users on high speed networks generally download all of the images on a page where modem users may selectively download images.

At the Web's inception, many were concerned about the download speed of their pages, and kept images to a minimum. Today, graphically rich sites abound. Technologists have been pushed aside as the primary designers of web pages, and graphical designers generally design popular sites now. Many web designers think nothing of including many large in-line images in their web documents.

An informal survey of a few sites showed that downloading all embedded images resulted in retrieving three to eight times more bytes compared to just downloading the web page without images. When waiting three to eight times longer is fast enough to be below the users' wait tolerance threshold, they will begin to download more data. At any rate, it is likely that speed gains in user access will also affect web page designs – quite possibly leading to ever larger web pages.

Also, the ability to download data more quickly combined with the increase in graphics will undoubtedly lead to the retrieval of more documents as the browsing experience becomes more pleasing.

2.1.3 Higher Utilization

Applications are emerging which utilize idle user bandwidth. These types of services retrieve data when the user is not otherwise actively requesting it. Examples of current applications include news-displaying screen savers such as Pointcast. Pointcast retrieves information in broad categories chosen by the user. This information is downloaded on a regularly scheduled basis, or when the user is not using the network for other services. The information will be retrieved whether or not the user is actually going to read any of it.

Others have proposed methods of making web browsing appear faster to the user, such as pre-fetching pages that a user is likely to request after viewing the current page. New browsers even have the ability to download portions of web sites for off-line viewing, allowing the user to download all pages that are a certain number of links away from pages preselected by the user.

Other optimizations to existing applications, for example, might include email clients that retrieve email attachments while the user is reading another piece of email.

2.1.4 New Services

New types of services will also affect the kinds of traffic seen. Currently, real-time streamed audio and video is becoming more popular, since it need not be downloaded in entirety before viewing.

These real-time services increase traffic in several ways. First, those that were discouraged from downloading large audio or video files before will be more encouraged to do so, since there is no wait involved before beginning to view the file. Note that the same amount of data is likely to be transmitted, but when streaming is used, the listening or viewing time overlaps the download time. Users will be more likely to view streamed clips than non-streamed clips, leading to a larger amount of data downloaded per user.

Also, as users begin to request these types of services, they will begin to proliferate, making much more new real time information available.

2.1.5 Traffic Growth Demands More Routing Capability

If routing capacity increases according to Moore's Law, it will barely be able to keep up with the growth in the number of users, which is expected to triple in the next three years. This even assumes that access rates and bytes downloaded do not grow.

One seemingly simple solution might be to increase the number of routers along congested links, thus providing a selection parallel routes for traffic. For example, if a particular router is overloaded, one might replace it with two routers connected to all the same peer routers while maintaining the logical topology.

The problem with this solution is that increasing the number of routers increases the work required in the exchange of information between peer routers. In fact, the routing algorithms used grow as $O(N^2)$ where N is the number of routing peers. So increasing the number of routers is not a simple scalable solution, and breaks down rapidly with growing parallelism.

It is clear that the current methods of routing IP traffic cannot meet this demand. In the next section, we will see what kinds of traffic can be targeted by routing architectures for special treatment which requires less overall work.

2.2 Analysis of Existing Traffic Traces

Analyses of existing network traces at various points throughout the network reveal the types of services used on the Internet.

Various groups have used their analyses to motivate the types data sessions, or flows, to be identified for special treatment by routers. These traffic analyses are the basis for models

and metrics of flow classification.

In this section, we discuss the various existing trace analyses used to justify that using two types of symmetric endpoint flows in the current IP switching reference implementation is beneficial. The two existing flow types are Server flows and Machine flows.

To our own analysis, we add an extra parameter: the directionality of a flow, whether it be to or from a server. In this framework, we discuss our analysis of the Telemedia, Networks and Systems (TNS) server network trace, the Laboratory for Computer Science (LCS) gateway trace, and publicly available wide area network traces.

2.2.1 Existing Flow Analyses

Several prior works use network analysis of network traces to examine the distribution of traffic among various services. Each introduced new concepts to traffic analysis and flow classification.

Prior to [Claffy *et al.*, 1995], a single TCP session was generally considered a single flow in each direction. That work introduced the notion of a flow timeout, where packet spacings within a TCP session that were greater than some flow timeout interval increased the number of flows created. The main notion introduced was that flow tables should more dynamic, by equating flows to some minimum level of packet exchange between hosts instead of TCP sessions.

In [Newman *et al.*, 1996b], another wide area trace was analyzed using similar methods. This data was used to determine which services were worthwhile to switch using the symmetric endpoint flow switching model proposed by Ipsilon [Newman *et al.*, 1996b]. In selecting flows for special treatment in IP switching, those flows that were shown to have more than 40 packets in 20 seconds were selected for IP switching. Those services in which the average was found to be below this threshold are never switched. DNS traffic, which is far below the threshold, is thus never switched, but other types of traffic which are just below the threshold are also not switched.

Table 2.1 illustrates some results of gathering information on TCP flows. These results are consistent with prior analyses: HTTP traffic dominates in packet and byte percentages, and DNS flows are a disproportionately larger number of flows in comparison to their fraction of packets and bytes.

The reasoning for not classifying DNS traffic into switched flows is that traffic such as DNS generates a small, though not insignificant, number of packets, but would require a disproportionately large fraction of flow setup resources and flow tables space to be tracked. Specifically, in the Claffy trace from 1995, DNS consisted of only 2% of the total number of packets while representing 27% of the number of flows, while in the 1996 Ipsilon trace, DNS consisted of 5.6% of the total number of packets while representing 45.3% of the number

Protocol	port	%flows	%pkts	%bytes	sec/flow	B/flow	pkts/flow	bytes/pkt
TCP ftp-data	20	0.95	7.73	11.57	68.4	56580	164	345
TCP ftp-cntrl	21	1.77	0.87	0.30	66.3	780	10	78
TCP telnet	23	0.46	5.28	1.72	219.1	17475	233	75
TCP smtp	25	6.04	3.89	2.69	33.8	2067	13	159
UDP dns	53	22.05	2.19	1.04	49.0	220	2	110
TCP http	80	55.00	40.88	43.84	41.6	3705	15	247
TCP X-11	6000	0.02	0.10	0.06	204.4	15609	121	129
Other		13.72	13.62	38.79	114.5	13143	57	229

Table 2.1: A flow analysis from Cisco systems, taken over a 40 hour period in 1996, illustrates the large proportion of HTTP and DNS flows.

of flows. In 1996 data from Cisco systems, DNS consists of 2.2% of the packets but 22.0% of the flows. In a more recent trace from January 1997, DNS traffic at one backbone node shot up to over 24% of the packets representing 11% of the total bytes and 46% of the total flows.[1]

In contrast, TCP-based traffic such as HTTP or FTP constitutes the bulk of packets and data and are thus the prime candidates for flow classification and switching, since the work involved is setting up a flow can be amortized over a larger number of packets.

While earlier analyses looks at service types, they do not distinguish the directionality of a flow, whether towards or away from the server providing that type of service. Because we know that the data flow in one direction versus the other is asymmetric in byte volume, we suggest that the large server-originated packets are even better candidates for TCP or Machine flow switching than might be deduced from the earlier data, on a work per byte basis. Conversely, the server-destined packets associated with individual clients may be poorer candidates for TCP or Machine flow switching than previously thought.

Also, the current IP switching reference model chooses to ignore flows of certain service types as candidates for flow switching. Based on the reference TCP and Machine flow types, this makes sense as it may consume more resources to switch these flows than to route them. However, as we will see from our server-based analysis of DAN/LAN traffic, we will introduce new flow types that make DNS flows very good candidates for flow switching.

These prior network trace analyses are a good starting point in identifying flows for special treatment, but they do not go far enough in that they overlook the large asymmetry of traffic flow by byte volume and the concentration of traffic around servers.

What we propose differs in two main ways: we will use asymmetric endpoint flows that take

[1]This could be due to a variety of reasons. DNS registrations and Domain usage shot up in this time, or the network trace's location may have more nearby DNS servers to which queries are concentrated.

Protocol			%flows	%pkts	%bytes	flows/s	pkts/s	pkts/flow	bytes/pkt
TCP smtp	inbound	25	0.15	0.06	0.04	0.00	0	17	261
	outbound		0.15	0.06	0.01	0.00	0	15	75
	local		0.07	0.04	0.01	0.00	0	22	66
UDP dns	inbound	53	4.44	0.54	0.21	0.04	0	5	164
	outbound		4.78	0.61	0.06	0.04	0	5	41
	local		0.01	0.00	0.00	0.00	0	1	95
TCP http	inbound	80	37.47	16.74	0.73	0.33	6	17	19
	outbound		37.44	20.46	19.74	0.33	7	21	411
	local		0.33	0.06	0.04	0.00	0	7	320
UDP nfs	inbound	801	0.01	0.00	0.00	0.00	0	28	594
	outbound		0.00	0.00	0.00	0.00	0	31	96
	local		0.10	16.00	38.42	0.00	5	6378	1022
TCP X-11	inbound	6000	0.46	1.07	0.02	0.00	0	90	10
	outbound		0.48	1.70	4.56	0.00	1	136	1142
	local		0.04	2.39	0.45	0.00	1	2319	79
Total	inbound		42.99	37.84	14.55	0.38	13	34	164
	outbound		43.21	27.78	26.42	0.39	10	25	405
	local		13.80	34.38	59.02	0.12	12	96	731

Table 2.2: **Local Area Traffic**: Flow traces from the TNS group's web server network, broken down for directionality.

advantage of the asymmetry we identified, and we will include more types of traffic classes in those that are eligible to be switched.

2.2.2 Interesting Features of Network Traces

In order to highlight the byte volume asymmetry of traffic flow, we analyzed traffic with the additional parameter of directionality. We use the results to support our assertion that much traffic well-suited for IP switching is also highly asymmetric in byte volume and is concentrated in servers. This leads to our identification of new types of asymmetric endpoint flows – the Server and Session flows.

As we will see in Chapter 4, adding these asymmetric endpoint flows to the IP switching model bundles flows that were previously not worthwhile to switch and moves them into the region where they are very worthwhile to switch. This is just one of the benefits of being able to exploit the asymmetry and concentration of traffic around servers.

In the following three sections, we show results of our network analyses of Workgroup, LAN, and WAN traffic traces, and point out the most interesting features of each.

Workgroup Traffic

In analyzing traffic on our local ethernet, in addition to the direction of the flow, we also kept track of whether or not the traffic stayed entirely within the local area network comprised of multiple ethernet segments within our workgroup.

In a 20-hour period in April 1996, we counted packets on the ethernet segment on which our group's web server resides. We counted over 2 million packets entering and leaving this segment, of which 407,565 (18.5%) were packets destined for the web server port, and 498,355 (22.7%) were replies from the web server. The fraction of our traffic that was web traffic (41.2%) closely matched the 40.21% reported by Ipsilon at that time. Both data sets were gathered within one month of each other.

Of the 407,565 packets sent to the web server, 364,924 of these, or close to 90%, were TCP protocol acknowledgement or server request packets. These packets came from at least 3220 distinct clients and their total byte volume was slightly over 10 MB (including packet headers). In contrast, the server delivered over 205 MB to the clients.

If one were to average the bytes per packet over all the packets, both inbound and outbound, as was done in prior analyses, one would get a distorted view of the traffic. Producing averages of bytes over all flows produces a similarly distorted picture.

One of the most notable feature of the Workgroup trace is that internal traffic patterns of desktop hosts are also noticeably different than traffic patterns of the network servers. In general, internal flows are much longer in duration than external flows.

From Table 2.2, we point out a few interesting points. In total, local workgroup-only flows represented 13.8% of the flows but almost 60% of the bytes. This represents good value for any work required to account for and setup internal workgroup flows.

The remaining flows were roughly equal for inbound and outbound flows. But, the inbound number of bytes was half the outbound number. This asymmetry is much more dramatic for HTTP packets – outbound bytes outnumbered inbound bytes by almost 30 to 1, even when the number of flows were almost identical and the number of packets were roughly on par.

LCS Gateway Traffic

Our analysis of the LCS Gateway traffic also yielded interesting results. Over 7.8 million packets containing almost 2.8 GB were counted over a one-hour period in September 1997.

Fully 50% of the bytes and 50% of the packets were web based. One point of interest is while 203 web servers within LCS provided information, the bulk of the data delivered was provided by just a few large servers. Also, the inbound DNS packets were handled by an even fewer number of DNS servers. Server flows that are set up for these DNS servers

Protocol		port	%flows	%pkts	%bytes	flows/s	pkts/s	pkts/flow	bytes/pkt
TCP ftp-data	inbound	20	0.14	4.27	1.98	0.0	88.6	2382.8	163.6
	outbound		0.14	6.14	9.16	0.0	127.4	3425.3	525.2
TCP ftp-cntrl	inbound	21	0.32	0.15	0.01	0.1	3.1	35.8	15.4
	outbound		0.32	0.15	0.02	0.1	3.2	36.2	37.0
TCP telnet	inbound	23	0.32	1.52	0.06	0.1	31.6	365.2	13.4
	outbound		0.31	1.47	0.11	0.1	30.6	361.5	25.8
TCP smtp	inbound	25	4.01	1.88	0.34	1.1	38.9	35.3	63.7
	outbound		4.25	2.40	1.59	1.2	49.8	42.7	232.9
UDP dns	inbound	53	27.56	0.87	0.28	7.6	18.0	2.4	50.5
	outbound		28.00	0.96	0.20	7.7	19.8	2.6	75.5
TCP http	inbound	80	8.63	21.62	3.10	2.4	448.7	189.4	50.5
	outbound		8.61	28.51	46.81	2.4	591.7	250.2	577.8
TCP pop-v3	inbound	110	0.05	0.12	0.00	0.0	2.5	191.6	11.1
	outbound		0.05	0.16	0.15	0.0	3.2	251.3	345.5
TCP authent	inbound	113	1.87	0.07	0.00	0.5	1.5	3.0	1.6
	outbound		1.94	0.08	0.00	0.5	1.7	3.1	1.7
TCP nntp	inbound	119	0.05	4.92	11.94	0.0	102.1	8237.3	853.9
	outbound		0.05	7.08	2.67	0.0	146.9	11850.9	132.5
UDP ntp	inbound	123	0.10	0.03	0.00	0.0	0.5	20.5	0.0
	outbound		0.09	0.03	0.00	0.0	0.5	21.8	0.0
UDP nfs	inbound	801	0.11	0.92	0.41	0.0	17.1	572.9	173.7
	outbound		0.11	2.93	8.29	0.0	60.7	2001.9	997.8
TCP X-11	inbound	6000	0.08	1.41	0.84	0.0	29.3	1405.8	208.7
	outbound		0.08	1.45	1.26	0.0	30.0	1422.3	307.6
TCP SSH	inbound	7000	0.07	0.03	0.07	0.0	0.7	37.2	741.9
	outbound		0.07	2.53	0.82	0.0	52.6	2769.9	114.2
TCP SSH	inbound	7001	0.16	3.14	5.65	0.0	65.2	1516.0	633.4
	outbound		0.04	0.04	0.06	0.0	0.8	74.0	526.9
Other	inbound		4.66	1.40	0.63	1.3	29.1	22.7	157.3
	outbound		4.74	1.95	2.42	1.3	40.5	31.2	437.1
Total	inbound		49.8	43.2	25.4	13.7	895.8	65.6	207.0
	outbound		50.2	56.8	74.6	13.8	1179.3	85.6	462.1

Table 2.3: Analysis of LCS gateway traffic shows the dominant traffic types and a high degree of asymmetry in services such as HTTP.

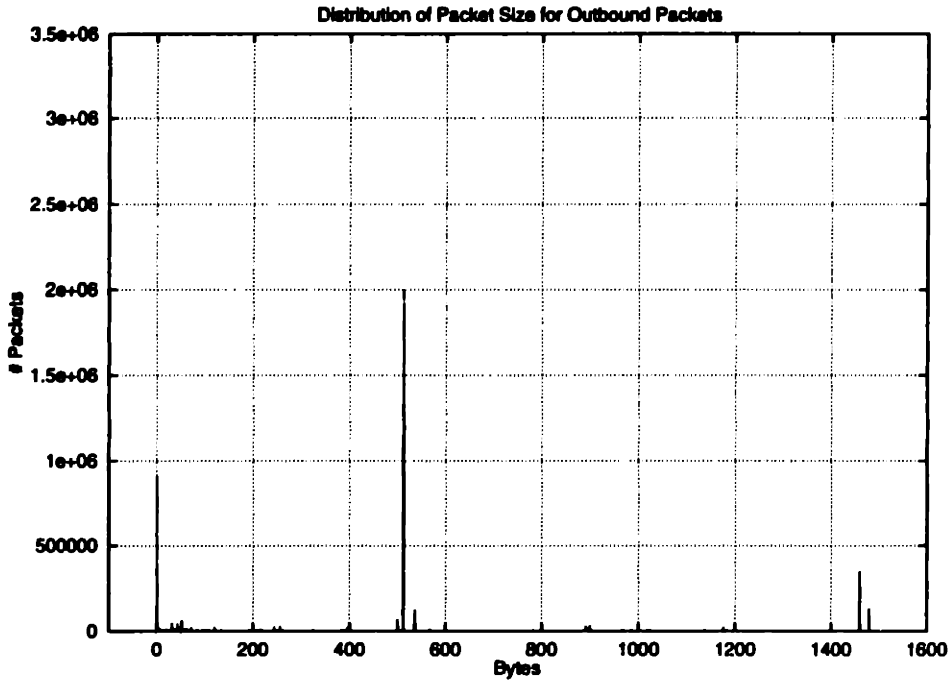

Figure 2-2: Distribution of packet sizes for packets leaving LCS.

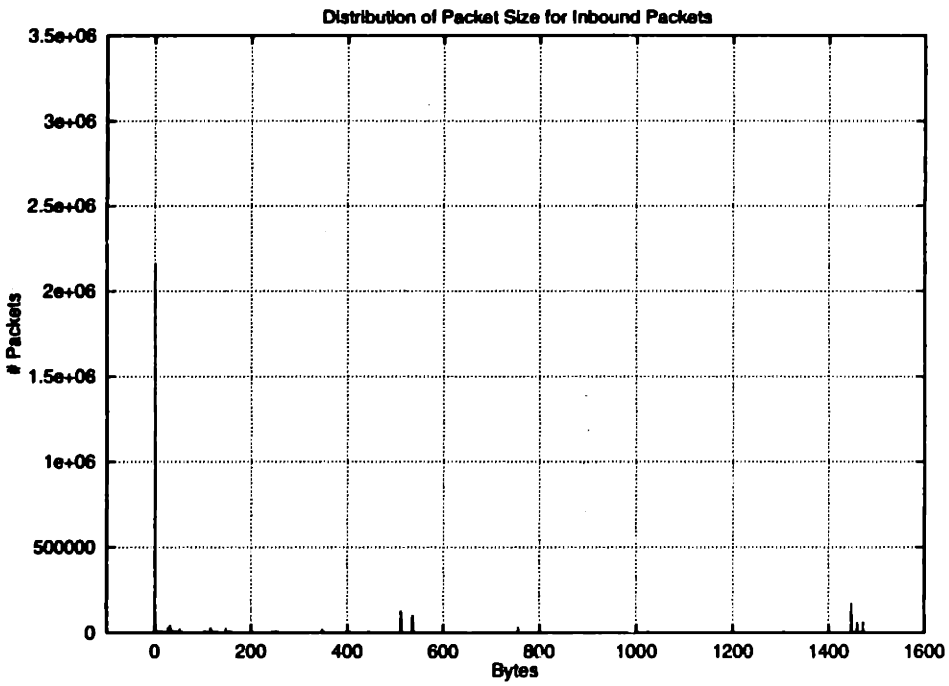

Figure 2-3: Distribution of packet sizes for packets entering LCS.

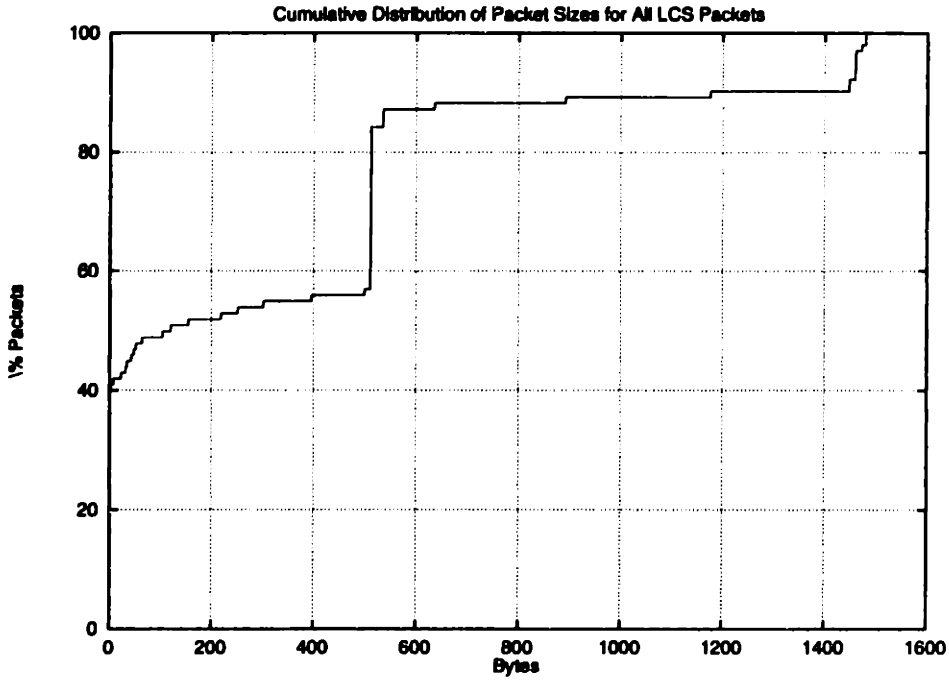

Figure 2-4: Cumulative distribution of packet sizes for all LCS packets.

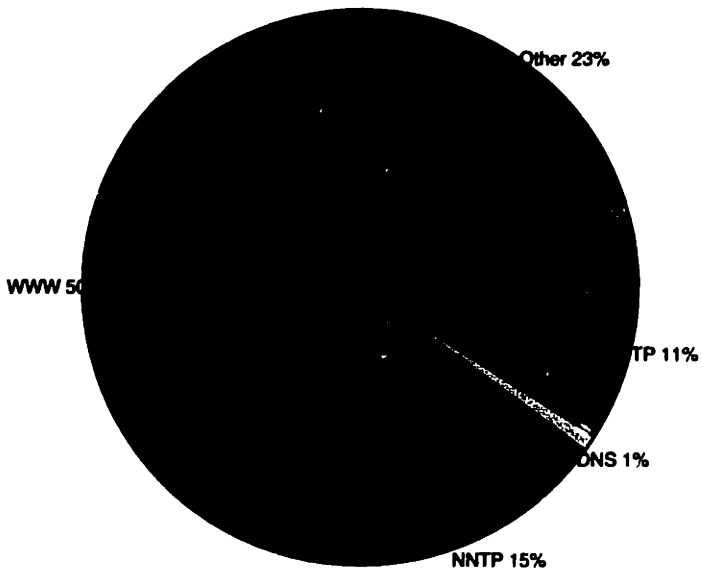

Figure 2-5: Total bytes traversing LCS gateway, by service.

would be very heavily used compared to the 40 packets in 20 seconds rate used to justify the existence of most flows in [Newman *et al.*, 1996b]. This concentration of traffic is important and is at the heart of the asymmetric endpoint flows we propose.

Figures 2-2and 2-3 show the distribution of packet sizes for outbound and inbound packets, respectively. This again emphasizes the asymmetry of flows by byte volume based on the directionality of traffic. Another interesting feature is that over 40% (see Figure 2-4) of all packets are completely empty TCP/IP packets with just 40 bytes of header information. The enormous number of small 40-byte TCP acknowledgement packets suggests that the ATM cell data size, which many consider to be "too small" for IP over ATM, is perhaps not too small after all.

Wide Area Traffic

We also looked at existing traces from the FIX-West [2] backbone node. FIX-West is the Federal agency Inter-eXchange point between NSFNET and ARPAnet on the west coast. FIX-West and FIX-East were major top level national network interconnections for quite some time, but their main function is now supplemented at Network Access Points that were introduced to the network backbone in 1995. The latest trace available was from January 1997, and another trace from February 1996 was also examined.

Regardless, we see the same features in the WAN trace that we saw in the others: traffic is concentrated in Server flows, and in most cases, the predominant direction of data flow by byte volume is away from servers.

One very surprising item which emerges from the latest trace is that DNS traffic at this traffic interchange has exploded between the traces. It is difficult to make generalizations without more traces from throughout the network, but if DNS traffic has scaled equally throughout the Internet, we may be able to attribute it to an evolution in the design of large web sites and the explosion of domain registrations. Or, DNS traffic could have become concentrated in non-commercial exchanges due to the nature of the DNS hierarchy.

2.3 Summary

It is clear from all of the traces that TCP/IP is the dominant protocol used all across the Internet. Also, traffic which leaves the local workgroup is dominated by server-based services, whether it be WWW, DNS, SMTP or NNTP. The trends indicate that traffic will continue to be concentrated in these types of server-based services.

[2] FIXWEST/NLANR traces can be retrieved from http://www.nlanr.net/Traces/FR+/

Protocol		port	%flows	%pkts	%bytes	sec/flow	B/flow	pkts/flow	bytes/pkt
TCP ftp-data	to	20	0.19	2.73	3.02	28.1	54745	174.4	313.9
	from		0.14	2.47	5.07	29.7	124890	213.9	583.9
	total		0.33	5.20	8.08	28.8	84473	191.2	441.9
TCP ftp-cntrl	to	21	0.17	0.15	0.03	20.3	551	10.9	50.6
	from		0.15	0.12	0.04	18.1	886	9.7	90.9
	total		0.31	0.27	0.06	19.3	709	10.3	68.5
TCP telnet	to	23	0.19	0.86	0.13	36.6	2357	55.2	42.7
	from		0.16	0.57	0.23	34.3	4915	42.9	114.5
	total		0.35	1.43	0.36	35.5	3533	49.6	71.3
TCP smtp	to	25	0.94	2.01	3.06	20.0	11181	25.8	433.5
	from		0.83	1.24	0.23	16.7	945	17.9	52.8
	total		1.77	3.24	3.29	18.4	6369	22.1	288.4
UDP dns	to	53	0.58	0.18	0.04	7.7	235	3.8	62.5
	from		0.34	0.90	0.60	11.4	6112	32.5	188.3
	unknown		45.16	22.92	11.03	16.8	837	6.1	136.8
	total		46.08	24.00	11.67	16.6	868	6.3	138.1
TCP finger	to	79	0.01	0.00	0.00	3.9	273	6.6	41.4
	from		0.01	0.01	0.01	4.3	1404	6.7	209.2
	total		0.02	0.01	0.01	4.2	990	6.7	148.5
TCP http	to	80	12.28	10.53	2.65	11.6	739	10.3	71.5
	from		15.92	14.74	31.86	10.7	6858	11.2	614.3
	total		28.21	25.27	34.51	11.1	4193	10.8	388.1
TCP pop-v3	to	110	0.08	0.08	0.01	12.1	518	11.7	44.1
	from		0.05	0.05	0.04	11.1	2849	11.8	241.6
	total		0.13	0.13	0.05	11.7	1402	11.8	119.2
TCP authent	to	113	0.23	0.03	0.00	1.6	67	1.5	43.7
	from		0.24	0.03	0.00	0.3	59	1.4	43.1
	total		0.47	0.06	0.01	0.9	63	1.4	43.4
TCP nntp	to	119	0.15	1.47	2.10	43.4	48773	119.7	407.3
	from		0.14	1.15	0.46	41.1	11467	100.4	114.2
	total		0.29	2.62	2.56	42.3	30752	110.4	278.6
UDP ntp	to	123	0.04	0.00	0.00	0.4	109	1.4	76.0
	from		2.76	0.38	0.10	1.9	125	1.6	76.0
	total		2.80	0.38	0.10	1.9	125	1.6	76.0
TCP netbios samba??	to	139	0.01	0.00	0.00	17.7	885	8.9	99.1
	from		0.02	0.01	0.01	21.6	1141	8.9	128.0
	total		0.02	0.02	0.01	20.6	1076	8.9	120.7
UDP snmp	to	161	2.29	0.25	0.07	0.4	106	1.3	80.6
	from		0.05	0.02	0.01	12.6	467	4.9	94.7
	total		2.33	0.27	0.08	0.7	113	1.4	81.6
TCP login	to	513	0.03	0.16	0.02	29.9	2613	64.1	40.7
	from		0.05	0.18	0.10	22.7	6682	42.6	156.7
	total		0.08	0.34	0.12	25.4	5165	50.6	102.0
TCP cmd	to	514	0.01	0.03	0.05	27.2	24052	62.0	388.0
	from		0.00	0.03	0.03	12.2	45123	153.1	294.8
	total		0.01	0.07	0.08	23.1	29864	87.1	342.8
TCP audio	to	1397	0.01	0.01	0.02	9.0	7473	11.4	656.8
	from		0.00	0.00	0.00	21.1	876	19.6	44.8
	total		0.01	0.01	0.02	10.2	6821	12.2	559.7
TCP X-11	to	6000	0.02	0.16	0.13	34.0	21093	88.6	238.0
	from		0.02	.016	0.05	33.2	8192	99.1	82.6
	total		0.04	0.32	0.18	33.6	14917	93.7	159.3
Other	unknown		16.29	35.19	37.80	6.2	7953	26.0	305.3
All Flows			100	100	100		3934.5	19.9	197.5

Table 2.4: **Wide Area Traffic**: Analysis of January 1997 traffic trace from FIX-West Backbone with a 60 second flow timeout, taking into account directionality.

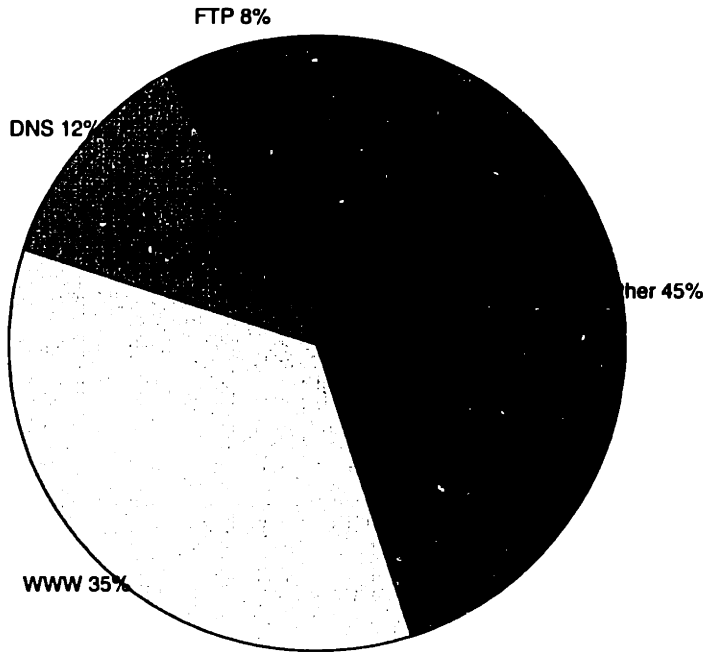

Figure 2-6: Total bytes in FIX-West trace, by service.

While the wide area traffic traces only hint at flow aggregation around servers due to the number of packets from various sources, the gateway and local traces demonstrate very clearly the aggregation of flows around these types of servers. Furthermore, a small number of the set of accessed web servers provide a dominant amount of the data, and the number of active DNS servers is even smaller than the number of active web servers.

Our traffic analyses, in which we introduce flow directionality with respect to servers, motivates the need to distinguish this server traffic from other types of traffic for flow classification purposes. If such flows can be set up, there is an immediate gain of almost 100%, doubling the effective sizes of flow tables within the network and halving the amount of work required to route flows generated by servers. As we will see later, applying other methods of flow classification can build on these efficiency gains.

Asymmetric endpoint flows may be more easily recognized at this local level, but the benefit of using asymmetric endpoint flows extends far into the wide area network.

Chapter 3

Network Design - A Flexible Approach

The VuNet, our high speed network implementation, was designed to support machines within the "Desk Area." In a Desk Area Network (DAN), network peripherals attach directly to the network. Machines within the DAN, all connected at high rates, access network peripherals directly over the network and coordinate the movement of information streams among the different parts of the system. The VuNet is illustrated in Figure 3-1.

When we set out to design and build the VuNet we had several high-level goals. The network was targeted towards a working environment where we would be able to bring high bandwidth to the application level. 10 Mb ethernet was the deployed standard for network connectivity when we began the project. We did not consider this to be sufficient for the environment we envisioned, where multimedia data needed to be easily shared throughout the network among specialized devices and hosts.

Our goal was to deliver over 100 Mb/s of data to the application level of the attached hosts. This represents the data rate of a full frame, full resolution, full color, uncompressed video stream. Bringing such high speed network services to the desktop would enable a variety of new host applications [Lindblad, 1994, Wetherall, 1994, Stasior, 1997, Tennenhouse *et al.*, 1996].

Secondly, we demanded flexibility in the ability to manage our resources. A chief concern was the ability to manage network connections in a connection-oriented environment. We also needed to integrate services using existing protocols, and also wished to take advantage of the asymmetries, identified in the previous chapter's traffic analysis, to improve the management methods of network services.

Finally, our goal was to make the designs simple and straightforward so that we could rapidly prototype and build the network components. Both the components and the com-

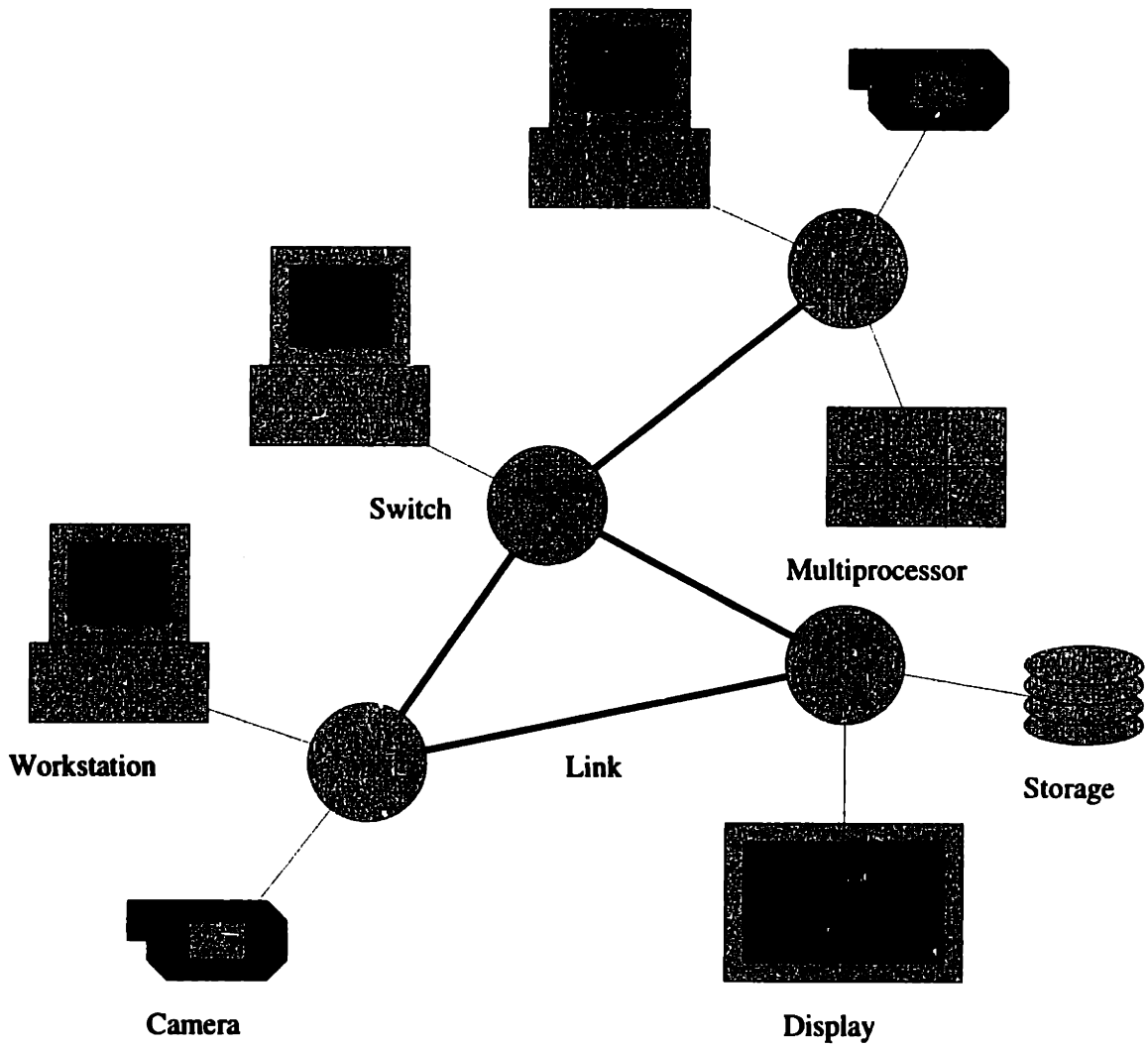

Figure 3-1: A Desk Area Network where devices are taken out of the workstation and attached directly to the network. The workstations coordinate the flow of data among devices.

ponent interfaces must share this trait. This was reflected in the VuNet lacking traditional network-based functions which add complexity at the hardware level such as complex queue management, link-level flow control, and link-state information exchange for routing protocol use. We made a clean separation between network components which would move bits and network management functions which we did not include in the network hardware.

The different VuNet components, comprised of switches, links, host interfaces and peripherals, formed a networking toolkit that enables high speed services and data to be delivered to the desktop application from the desk area.

This chapter discusses the first phase of the VuNet (VuNet I). We describe the decisions we made in order to achieve our high-level goals. We also discuss other major design decisions, and then describe the components of the VuNet and their operation. We show how we integrated the VuNet services using existing protocols, and finally, we discuss the consequences of our decisions and our performance results.

The second phase of the VuNet (VuNet II) incorporates the novel aspect of switching flows. The VuNet II will be discussed later in Section 4.3.

3.1 VuNet I Architecture

Our high-level design goals lead to some choices which affect the overall architecture of the first phase of the VuNet. In order to meet the requirements for simplicity of the network components, we selected Asynchronous Transfer Mode (ATM) as the transport layer protocol. Also, the flexibility and simplicity lead us to push functions from the network to the edges and even into the host software.

3.1.1 ATM

ATM is a connection-oriented system based on fixed-sized packets, called "cells." We chose ATM because fixed-size cells made the hardware design simpler, and furthermore, small cells lead to low switching latency. At the time we began the project, there were no commercially available ATM solutions. Each ATM cell contains a header in which a Virtual Circuit Identifier (VCI) determines the path through the ATM network. This VCI is the basis for label swapping when cells pass from switch to switch.

We take the simplified hardware approach a step further and push functions found in most ATM switches into the hosts and links. Many switches (ATM switches included) provide complex port controllers to manage queues and connections. In eliminating all of these functions from our switch, we incorporate the table lookup and label swapping functions in the links. These functions are generally found in a switch's port controllers; since it is necessary only when data moves from switch to switch over a link, there is no loss in

functionality when shifting this to the links. It is the method by which the tables are managed that changes.

Our ATM switch provides a straightforward interface that simplifies the design of client hardware. Since the timing of the client side of the port FIFOs is decoupled from the internal timing of the switch matrix, there is no need for clients to synchronize their operation with the internal switch clock, as is often the case with traditional switch designs. Furthermore, both the transmit and receive blocks can be operated concurrently. The FIFOs also serve to buffer cell bursts, an important factor in our simplified network. Finally, the data bus width to the port is selectable, either 32 or 64 bits, allowing a simple mapping to workstation and processor buses. All these factors make it easy to design devices which connect directly to the switch, making the switch clients simple to design.

3.1.2 Separating Network Mechanism and Policy

The demand for simple components and the necessity for flexibility in connection management lead to a software intensive philosophy where our design favors software-based solutions. This reduces the complexity of network hardware by separating the network transport functions in the hardware with the network management functions which we placed into the software of the end nodes. Not only can "out-of-band" network functions such as link table setup and topology discovery be software-based, "in-band" functions such as ATM adaptation layers can be performed in end-node software as well. The software-intensive approach also increases the flexibility of the system since issues such as resource scheduling and the choice of adaptation protocols can be tailored to specific application needs on these same end-nodes.

In a sense, we share the resources of the host in order to perform our network functions. We have access to a larger pool of resources such as an abundance of memory, idle processor cycles, and processing power which increases greatly over time when machine are upgraded.

This flexibility paid off when integrating our network using our extensions to IP switching techniques. Other systems, even experimental systems, did not have the flexibility for such extensions to be possible.

One of the most important consequences of moving network functions to the edges is that we must manage the network's virtual circuit identifier (VCI) tables from the hosts. We can do this for our network because of the important observation that the trust boundaries in our desk area network are different from those of the local and wide area networks. Large networks need protection from hostile or faulty clients. Such protection includes access control and fairness policies in order to prevent greedy users from taking bandwidth away from well-behaved users.

Networks such as ours, whose end nodes are dedicated to one work environment, can be controlled more easily. The end nodes have the ability to configure and change routes, and

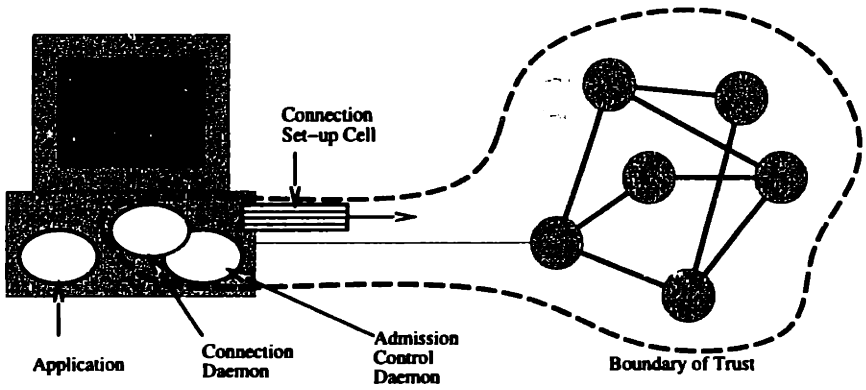

Figure 3-2: The workstation performs some of the high level networking functions such as topology discovery, connection setup, and possibly even admission control. Hence, the network "boundary of trust" extends into the workstation.

are trusted to only manage routes for which they are entitled. As illustrated in Figure 3-2, the host workstation software can provide network functions such as routing, topology discovery, and admission control. In addition, many of the protections and automatic reconfiguration issues found in larger networks can be moved into the end nodes. Such small trusted networks greatly reduce the complexity of our network components.

Note that separating the network transport with the management functions does not dictate the structure of the management functions. Functions such as connection setup and topology discovery can be distributed, or they can be performed by one or a small fraction of the end-nodes.

3.1.3 Other Design Decisions

The three design decisions listed above arise from our goals of flexibility, speed and simplicity. In this section, we discuss other major design decisions.

Software Segmentation and Reassembly

While the decision was already made to incorporate network control into the end hosts, the issue of segmentation and reassembly (SAR) of large packets into ATM cells is orthogonal to the issue of network control.

The SAR issue pits our high-level goals against each other: speed vs. simplicity and flexibility. Most ATM network interface cards provide the SAR function in hardware, providing the hosts with full packets to process. We make the decision to provide SAR as a host-

based software function. While this may degrade overall host performance, it provides us the flexibility to allocate different SAR methods for different VCIs, which is important for merging VCI streams.

Section 3.5 will discuss how this impacted the overall performance of the VuNet.

In-Band Link Management

As stated previously, a link's VCI re-mapping tables are managed from the hosts. However, host management does not specify the channel through which the tables are modified. We made the decision to use in-band processing to modify the tables, as opposed to out-of-band methods used by most others. We chose a special "magic" VCI for this purpose.

The setup cells can be pre-pended to any transmission needing route setups, without the need for the latency usually required in a connection-oriented network.

Collapse of OSI Layers

Unlike the 7 layer OSI model used in the design of most network components, our network design collapses some of those layers together for increased efficiency. Specifically, the ATM network hardware directly incorporates a hardware mapping function into the links, allowing routing table lookups to be cached in the network hardware. This forces the layer 2 (switching) and layer 3 (routing) to be collapsed. An IP switching approach reference implementation provides the control of these tables and the message passing between IP switch peers.

3.1.4 VuNet Influence on Other ATM Equipment

At the time we began the project, there were no commercially available solutions for ATM. The standard inter-networking products at the time were based on 10 Mb Ethernet. As the standards evolved and commercial products became available, they took two paths: one adhering the complex standards and one taking a simpler approach much like our VuNet.

For example, much of the commercial ATM switches from Fore, Bay Networks, Cisco and other vendors that are now available adhere to the ATM Forum's standardized methods of connection management. These include end-to-end connection setup solutions that require high overhead for short transactions. These solutions have been deployed first for backbone connectivity, as these connection management methods serve well to provide long-lived connections across backbones which bundle all traffic from one backbone access point to another.

We continued on our path of simpler connection management schemes, and were joined by Ipsilon, who threw out the ATM Forum standards in favor of a much simpler scheme that we will discuss in Chapter 4. It is possible that our simple approach to ATM influenced others to take a similar approach. Ipsilon's scheme also collapses the OSI layers, but differs for the VuNet in that the connection management is still out-of-band. However, the Ipsilon and VuNet methods of connection management are much more suitable for the local areas. We show that using the VuNet, we can provide different connection management schemes for different regions of the network, including the backbone, to increase the routing capacity of the network.

3.1.5 Summary

Many of the decisions we made took us toward separating the policy and mechanism of network transport. Because of the research nature of the VuNet, these design decisions often favored flexibility over speed. However, this was not a bad choice, as we demonstrate that the reliance on end-node software for many of the complex network functions can still produce excellent performance. In addition, we gain the flexibility to experiment with various connection management methods that greatly enhance overall network performance.

By taking advantage of the flexibility, the VuNet supports a myriad of access methods: standard IP datagrams through a fixed virtual circuit mesh to other VuNet hosts, approaches where individual connections are set up at the application level, and support for router emulation via IP switching methods.

3.2 VuNet Hardware Implementation

This section describes the hardware design of various VuNet components. The network infrastructure consists of two main components: switches and links. Switches, in whose design I participated, provide high-speed cell forwarding ability, and the links, which I designed and built, provide a means of interconnecting switches and a mechanism for hop-by-hop routing. In order to connect hosts to the network, I also designed and built a network interface card.

The VuNet hardware consists of four classes of hardware: switches, links, host interfaces, and network peripherals. The application of the design principles for the switches, links, and host interfaces is discussed in this section.

3.2.1 Switches

The VuNet switch provides a simple mechanism for the exchange of cells, without regard to their ATM header or payload. In contrast to other ATM switch fabrics, the responsibility

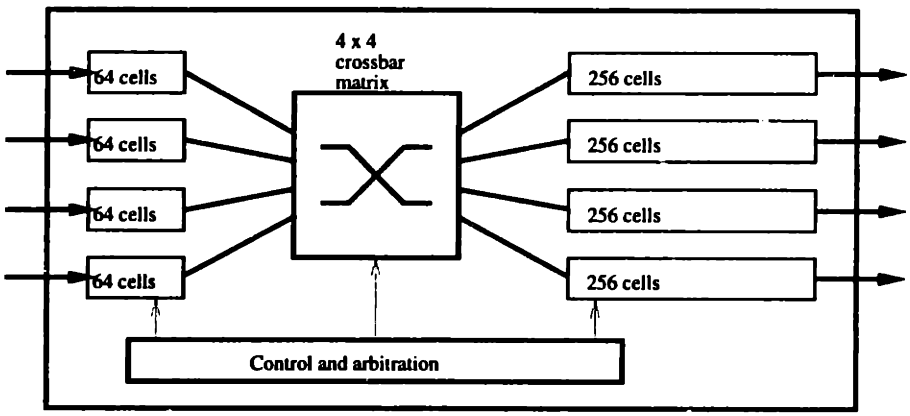

Figure 3-3: The VuNet switch is a 4 or 6 port non-blocking crossbar switch, with both input and output port buffering. It has a per-port throughput of 700 Mbps.

for ATM functions, such as VCI mapping, has been teased out of the switch fabric and assigned to the devices that plug into the switch. This is again an important separation of functions to push them to where they are best suited. In the case of VCI remapping, the only case where it is needed is when cells pass between switches. In the VuNet, this only occurs at links. This contrasts with other switch designs which include this function in all the input or output port controllers.

The switch's transmit and receive ports are very simple, containing only buffering and rudimentary arbitration circuitry. The buffering allows ATM cells to be read and written at any data rate up to the maximum allowed by the FIFO memories and their associated control circuitry. Each of the transmit and receive ports operates independently. They have been tested at 1.5 Gbps per port. Each port has substantial output port buffering and limited input port buffering. Input port queuing is somewhat unique to the VuNet switch, and is used to decouple the timing.

In order to transmit a cell across the switch, the client presents both the data and the destination port address to the switch. These are stored in the input FIFOs until a full cell is transmitted to the switch. Because of the input buffering, the portions of the cell need not be transmitted at regular intervals.

As shown in Figure 3-3, the input FIFOs of each port feed into the crossbar matrix; control circuitry configures the crossbar matrix for the duration of a cell transmission while cells are moved from the input FIFOs to the output FIFOs.

Multiple cells are handled by the crossbar matrix simultaneously. If there is output port contention, round robin arbitration on the input ports is performed. In this case, the cell that was not selected during this arbitration cycle blocks cells behind it.

The entire process is pipelined so that arbitration and crossbar configuration for the next

cell interval is performed while transfers are in progress.

Current versions of the VuNet switch have either four or six ports. The crossbar matrix is clocked at a rate of 700 Mbps per port.

Switch Buffering

VuNet switch buffering is necessary for a variety of reasons. On switch inputs, the purpose of buffering is to decouple the switch timing with the timing of the device. Also, a secondary use of the input buffers is to hold cells during high usage when there is output port contention. The buffering need not be very big for these purposes.

Buffering on the output performs a much more important function. Since the hosts directly connect to the network, the switch's output buffering must be sufficient so that the hosts can manage their memory and DMA buffers adequately. Since our architecture depends on large amounts of host memory for reassembly, the switch output buffers must be large enough so that when the host is managing these buffers and not accepting cells from the network, no data is lost.

In the case of the VuNet host, the device driver has been carefully designed to minimize this down time. An extra set of buffers are allocated in workstation memory so that the only downtime would be during the time required to disable the host interface, swap incoming data buffers, and then re-enable the host interface. The host interface and device driver will be discussed in more detail in Section 3.4.

A lot of tuning was required to determine the proper output buffering size required in the switch. Initially, the output buffers held just 64 cells. This was determined to be inadequate from the loss of data when receiving video-frame sized bursts from video applications, so the size of the output buffers was increased.

3.2.2 Links

The VuNet link implements two key functions, cell transfer and VCI mapping. To support the inter-office separation of switches, it is constructed in two parts that are inter-connected by high speed channels as illustrated in Figure 3-4. The channels use single mode optics and serial coding based on the HP G-Link chipset. The links operate at 500 Mbps.

The link tables are large enough to remap the entire 64K (16-bit[1]) VCI space to 64K new VCIs and four bits of outbound switch port information. All other bits in the ATM header remain untouched. [2]

[1]The standard has since been set to provide 24-bit VCIs; however more modern memories have also increased in size.

[2]The VuNet does not support the Header Error Check (HEC) function on a hop-by-hop basis, as we do

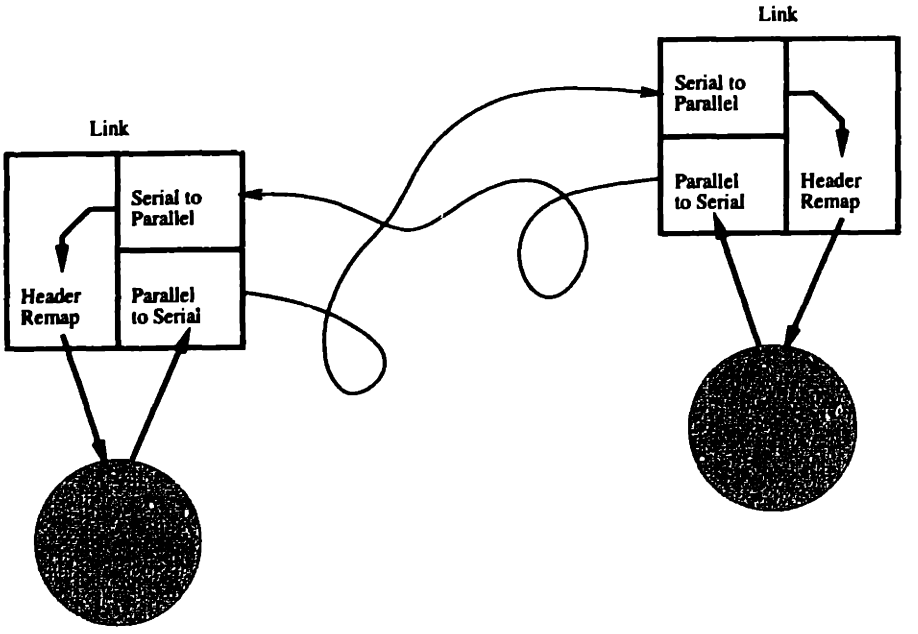

Figure 3-4: The links in VuNet perform header remapping and next output port lookup for hop-by-hop cell routing.

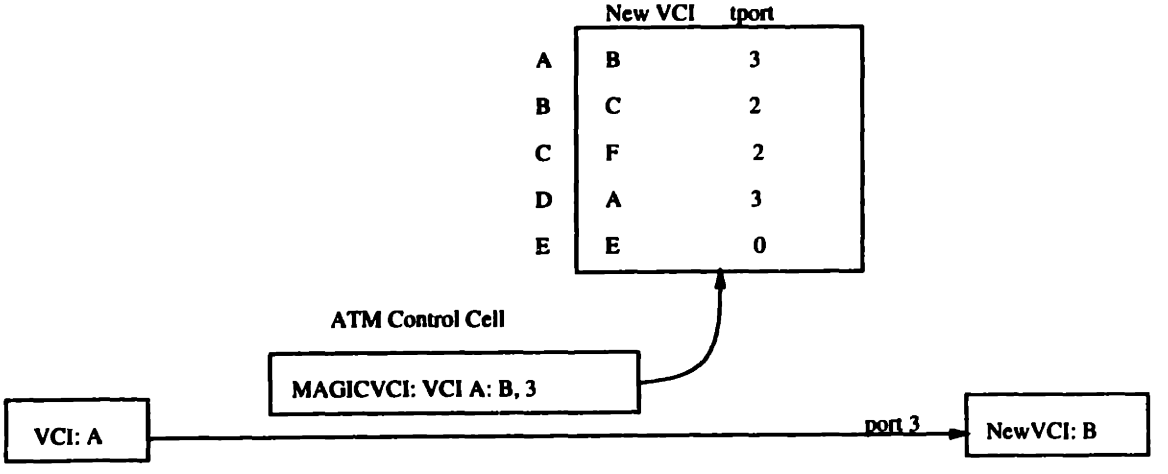

Figure 3-5: Cells sent on the reserved VCI are processed by the link to update its lookup tables. The cell immediately following the setup cell will be remapped to the new VCI and port.

Management of the link tables is performed using control cells sent with a reserved ATM VCI, as shown in Figure 3-5. Cells received by the link on these special VCIs cause the link to re-write entries in its header lookup table. They can also cause the link to emit a cell containing table entries, so link tables can be read back.

The selection of the link table management in this manner means that some single host or distributed set of hosts must manage the link tables throughout the network, and thus the network routes, by transmitting cells into the network to manipulate the link tables.

3.2.3 Host Interface

In designing our host interface, known as the VudBoard, we carefully selected the point of attachment between the host and the VuNet. Alternative approaches include a coprocessor style interface and direct connection to the cache or memory systems. We chose our attachment point from our experience with coprocessor style network interfaces [Gautam, 1993] and our desire not to interfere in the cache/memory hierarchy. We rely on the host's memory system to augment the switch's burst absorption capacity.

Many host interfaces in other systems provide support for packet reassembly [Davie, 1993, Kanakia, 1988] and streaming [Davie, 1993, Traw, 1992] in the hardware of the interface itself. In many of these host interfaces which support packet reassembly, the interface provides buffering on a per-VCI basis and only interrupts the processor when full packets are received. This approach is similar to the approach many have taken with providing video services – that is, to provide specialized hardware support for specific functions.

The VuNet interface (known as the VudBoard) is a simple DMA interface, allowing programmable length DMA bursts both to and from main memory into the board's FIFOs, which can hold one cell in each direction. The bus from the interface to the network is shared by the receive and transmit circuitry, so arbitration is done with priority given to incoming cells.

Incoming cells are packed into processor memory, where the device driver reassembles packets and delivers them to the proper application's socket, and eventually the data reaches the application memory space. This allows the interface to have very little onboard memory, essentially sharing the memory resources of the host.

Buffering at hosts is important because hosts generally have their own tasks to complete and are not slaved to network traffic. Thus, data must not only be reassembled from ATM packets into application-level data chunks, but also must be held for their processes while others are running.

---

not believe that HEC is an essential function for desk area or even local area operation. For compatibility purposes a cell source can pre-compute the HEC that is expected by the destination.

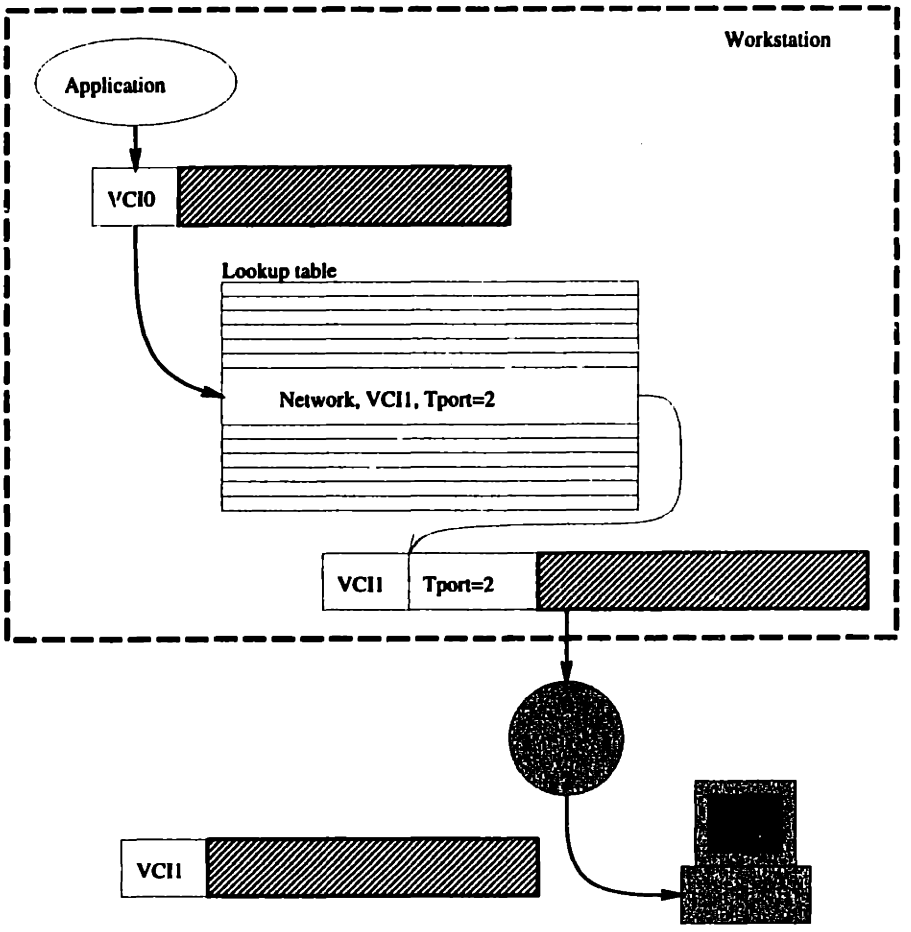

Figure 3-6: An application which needs to communicate to another workstation on the network generates data with an internal identifier. This identifier is remapped using a table look-up to generate the network VCI and first hop switch output port (tport). This information is passed to the switch, which forwards the data to the workstation on port 2.

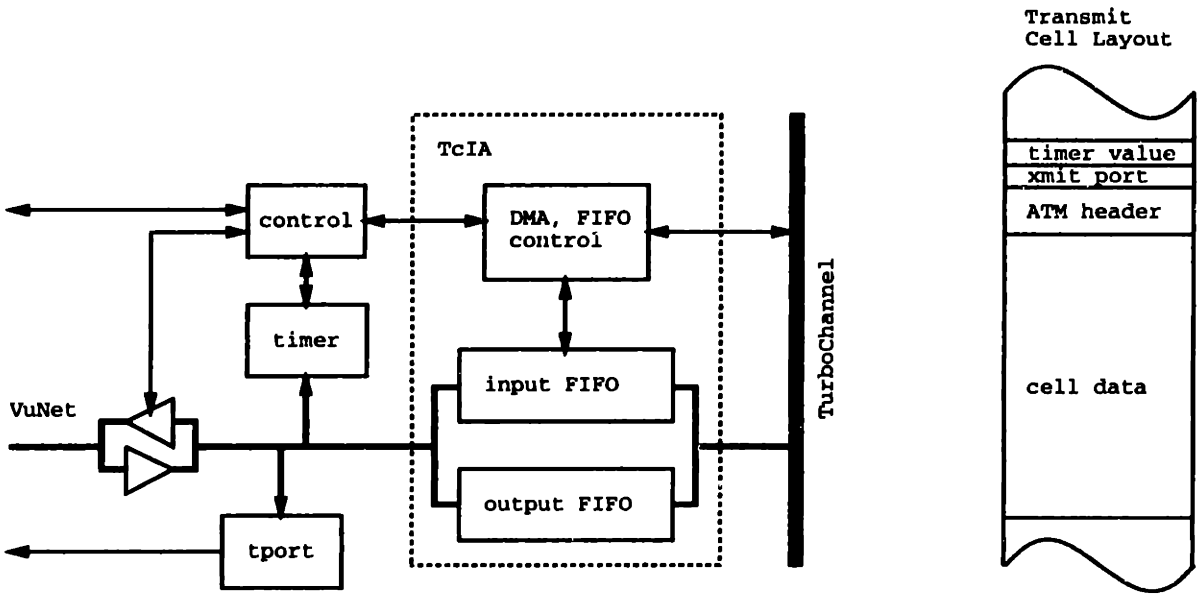

Figure 3-7: A block diagram of the VudBoard and the required cell layout in memory.

This is important in that many other host interface designs place a large amount of memory on the interface in order to buffer incoming cells while reassembling packets. Their placement of the reassembly function on the host interface necessitates this. However, hosts generally have a larger amount of memory, and using the host to reassemble cells allows the larger host memory to serve as a buffer. In this case, the only on-board and output port memory that is required is to buffer instances when the host has temporarily turned off DMA in order to manage its receive buffers.

Outbound cells are packed in main memory with routing information and timing information (outgoing cells can be paced on a per-cell basis), and are transferred to the interface using block DMA transfers.

The interface connects to a DEC Alpha workstation's peripheral bus, using the DEC TurboChannel Interface ASIC. It is capable of bursting at the full rate of the TurboChannel, which is nominally 800 Mbps. However, because of the small size of the DMA transactions and the DMA latency, the maximum achievable inbound DMA rate is 400 Mbps.[3] This is comparable to the Bellcore Osiris interface [Davie, 1993], which can achieve a maximum input throughput of 463 Mbps.

Figure 3-7 shows a block diagram of the VudBoard. The cell layout in memory is shown to the right, with the ATM cell preceded by the timer value and transmit port number for the cell. The TurboChannel Interface ASIC (TcIA) serves as the DMA engine to retrieve cells

[3] A more sustainable burst rate is 230 Mbps; performance at other levels is given in Section 3.5

from memory and write cells to memory. After initiating a DMA transfer from memory, the data is stored in the output FIFO, which cannot hold a full sixteen word cell. When the output bus is free, and the previous cell's time count has expired, the VudBoard controller reads the timer value and the transmit port (`tport`), and then transmits the ATM cell to the switch.

The hardware design for the VudBoard relies on the host to pre-segment the data for transmission and pack the cells' structures with the appropriate timer and tport values. The hosts thus have the ability to shape the traffic flowing into the VuNet. By properly selecting inter-packet timer values, the host can perform simple pacing or even more complex shaping, such as dribbling out the initial cells and then ramping up the cell rate, or sending out cells in brief bursts.

On input, the VudBoard's input FIFO serves as a way for the TcIA to monitor the status of the incoming data. When the input FIFO fills above a certain level, the TcIA requests the bus, and begins to transmit the data over the bus. While the FIFO continues to stay at or above a certain level, the bus transfer continues, unless the maximum transfer size has been reached. Hence the role of the input FIFO is to provide the appropriate monitoring of the input stream to make decision for the bus transfer.

3.2.4 Peripherals

In the VuNet, peripherals form a special class of devices. Unlike a host, a peripheral may not have the ability to perform all of the network-based functions that are required in the hosts. But since we have pushed first hop computation out of the switches, all peripherals must have the ability to provide the switches with the first-hop information it needs. A peripheral must be able to be told where to send its data, making it slaved to a host.

The Vidboard [Adam and Tennenhouse, 1993], a network-based peripheral which performs video capture, is one example of a VuNet host slave. The Vidboard is designed to capture video and will transmit single video frames on demand. It handles the ATM encapsulation and framing of the video and other data, and transmits the video data to any host that instructs it to. The Vidboard is told which outbound VCI and output port to use, and it uses that exclusively. It cannot manage or decide to switch VCIs at any point.

3.3 VuNet Protocols and Their Implementation

We detail here the methods by which the VuNet is used, and the various functions that are required to initialize connections between machines.

3.3.1 VudBoard Device Driver Architecture

The network interface device driver was written to work in a standard UNIX-based operating system. The driver for the VudBoard provides segmentation and reassembly functions in host software on a per-VCI basis. In addition, an address family implementation for raw VuNet virtual-circuit sockets was developed. By working within the existing sockets framework, we were able to easily provide TCP/IP connectivity as one of the protocols used throughout the VuNet. The address family implementation provides support for mapping VuNet VCIs to UNIX sockets. Thus, two types of services are offered: direct access to VCIs from user space, and IP services over existing VCIs.

3.3.2 Network Functions

This section discusses some of the network management functions that are assumed by the hosts.

Topology Discovery

When a host is started, it sets up a default input VC for VuNet control messages. First, the host tries to discover to what port it is attached by sending control messages to each port on the switch. Each message contains a probe cell identifier, a probed port identifier, and an indentifier uniquely identifying the probing host.

Cells received by the host on the VuNet Control VCI are examined. Any cells which contain a host's identifier will also contain the port number to which the host is attached. This information is stored and used for future reference.

When the host's port is determined, the host broadcasts to each of the other ports on the switch its own discovered information.

Then, the host probes for other network peripherals on the switch, and then uses the connection setup facilities to probe for links. Because the links cannot respond directly, any link must be inferred from data received from other machines, or from itself.

VCI and Connection Management

Each application in the VuNet is responsible for opening, maintaining, and closing its own connections. This is done in a "wormhole" fashion by way of ATM control cells embedded in the cell data stream. Applications use library functions that access a shared file and execute an allocation algorithm that prevent nodes from stealing other nodes' VCIs. Background processes can be used to verify link tables and refresh connections when necessary, such as in the case where a link card has been power cycled. In order to maintain table consistency

when the network is reconfigured, it is possible to run topology daemons that allow each host to discover the network topology.

Within a desk area network, it is appropriate for hosts to perform their own VCI allocation. This method does not scale well to large number of hosts. In order to interconnect with larger networks, a connection allocation server can be used to allocate inter-domain VCIs on a dynamic basis, or methods described in Chapter4 can be used.

Note that host-based connection management through wormhole tunneling implies that individual hosts must "own" portions of each link's VCI tables, since no single entity is allocating space for the individual connections of other hosts. This implies that hosts within the desk-area are trusted not to alter table entries which it does not control. Since the desk-area is limited in number of hosts, this is not a problem. Connection management for larger spaces outside of the DAN are discussed in Chapter 4.

On our DAN, each host owns chunks of every link's tables. The links do not check whether route update cells are altering table entries that the hosts should not be updating. Hence in the DAN, we must trust all hosts to do the right thing. There is no overlapping VCI space in our current VCI allocation algorithm.

We will see later that by restricting the trusted hosts only slightly, these techniques can be applied to metropolitan and wide area networking.

Opening VuNet Sockets

Raw VuNet virtual-circuit sockets can be opened by an application program. An application program can specify an output VCI for a socket with the `connect` system call, and and input VCI with the `bind` system call. Applications can also use `ioctl` to specify the format used for segmentation and reassembly, and the cell transmission pacing parameters of burst-size and burst-delay. Packets output by applications with `send` are segmented into cells and then transmitted on the virtual-circuit for the socket. Cells received from a virtual-circuit are reassembled into packets and queued for input from the corresponding socket with `recv`.

IP connectivity over the VuNet

IP connectivity is provided over the VuNet by assigning static VCIs for IP traffic between hosts. A special sockets `ioctl` is provided to arrange for input packets from certain VCIs to be queued by the VuNet reassembly code into the IP input queue, instead of a raw socket input queue. Output VCIs are specified by permanent entries in the ARP table.

A dense mesh is created at boot time in which all hosts have static VCIs to all other hosts and peripherals in the VuNet. This is a widely used practice in setting up ATM connections between machines on an ATM network. However, as the number of hosts and peripherals

N scales, the number of static connections grows as N^2. This obviously works for small networks, but does not scale to very large networks very well.

While each machine stays connected via its ethernet adapter to the rest of the world, the high speed route to other VuNet machines and peripherals is over the VuNet.

Protocol Integration

As shown in Figure 3-8, the driver is organized into several major levels. At the highest level are the application processes, which receive data over the UNIX socket interface. Below this are the kernel side of the socket interface level, the reassembly and protocol processing level, and the buffer and device management level. The details of the driver implementation will be discussed in the Section 3.4[4].

Reassembly Level

With the VudBoard host interface, segmentation and reassembly is performed in software by the kernel. The device driver is designed so that this segmentation and reassembly occurs at software interrupt priority level, instead of device interrupt priority level, to allow the kernel to be interrupted while performing segmentation and reassembly.

The reassembly of packets was designed to be flexible enough to provide for a variety of needs. Reassembling video data in which errors are unimportant use different routines than data for which errors are important. The flexibility to allocate this on a per-VCI basis adds great flexibility to our system – and allows us to easily address problems for which in other systems there may be no solution.

An application processing input data could result in code being executed at three interrupt priority levels. At the lowest priority level would be running the code of the application itself. At software interrupt priority level would be the VuNet network code reassembling cells into input packets. Finally, at device interrupt priority level would be the the code which handles device interrupts. Between each of these levels, buffer queues provide for transfer of data.

Socket Level

The UNIX socket mechanism is used as the data delivery mechanism for the applications that use our network data. Once the data is delivered to the socket level, it can be copied out to user space on user reads, completing the delivery of data to the application.

[4]Although an earlier version of the device driver was written by Chris Lindblad, I have substantially improved and instrumented it.

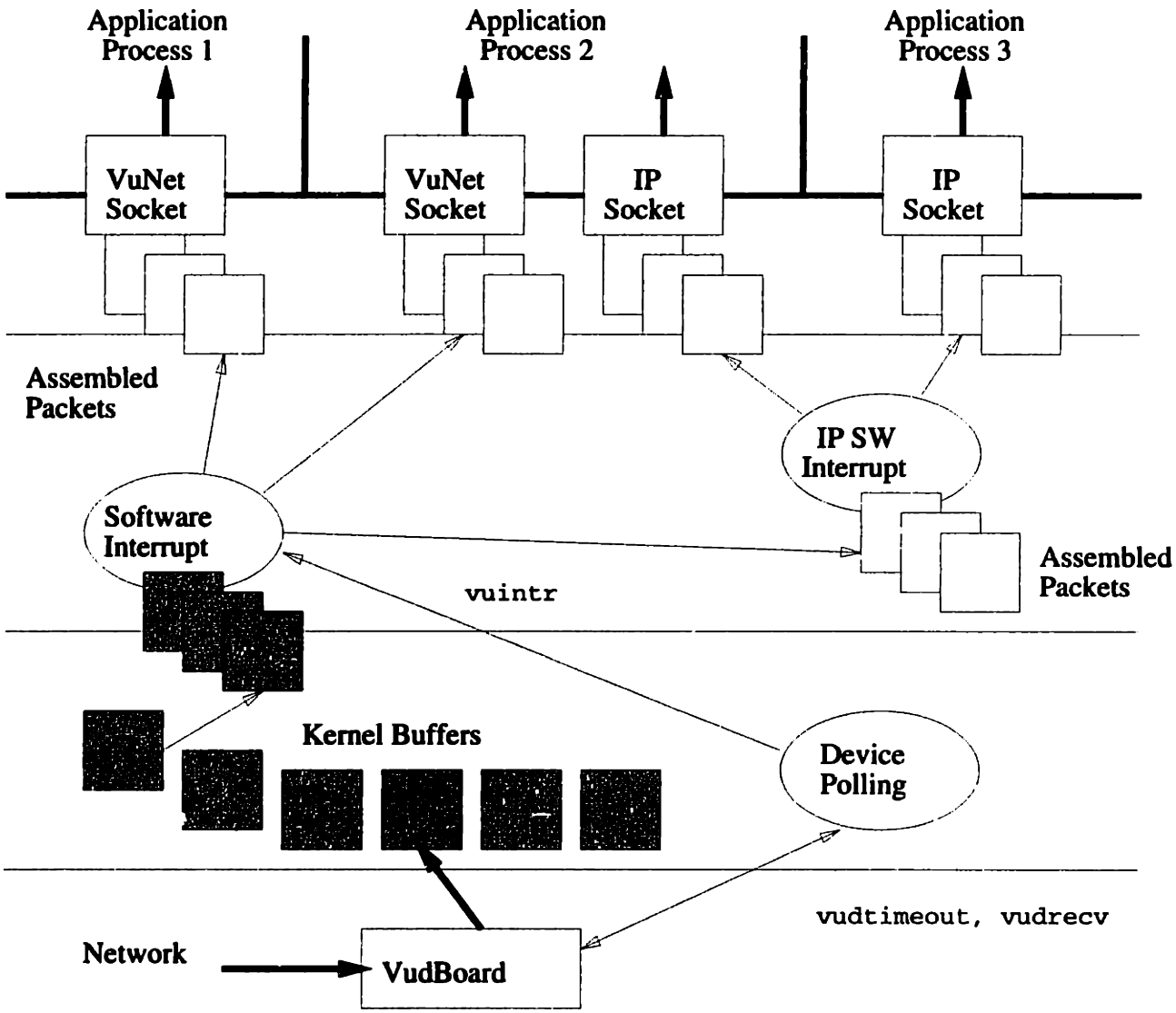

Figure 3-8: Cells are read from the network at and passed to the software reassembly process at regular polled intervals. These assembled packets are then passed to the IP handler or directly to a VuNet socket.

While using UNIX sockets adds inefficiency to the data delivery process through the introduction of extra memory copies, there is other ongoing research which makes operating systems more streamlined with respect to data delivery from the network [Montz *et al.*, 1994, von Eiken *et al.*, 1995]. Our system and architecture can only stand to benefit as these principles become widely deployed. Appropriate software could avoid the copy required from operating system memory to user memory currently required in most UNIX flavors, including NetBSD. To reduce the required copying even more, the host interface could use the VCI to write data directly into the appropriate socket's buffer.

3.4 VuNet Device Driver Data Paths

Each interface configured into the system is referenced by its corresponding `vud_softc` structure. This structure contains required system structures, and other pointers necessary for operation such as: device register pointer, receive/transmit buffer pointers, swap receive buffer pointer, scatter/gather table pointer, transmit mbuf queues, data counter, and pointers to control information for each possible receive/transmit VCI.

3.4.1 Input Polling

In many systems, interrupts are used to start packet processing when a packet arrives from the network. Alternatively, the interface may be polled at regular intervals in order to determine the arrival of packets.

Interrupt processing requires a fair amount of processing overhead. For the reception of a large number of small packets, the context switching overhead is significant but amortized over a burst of cells, while the per-cell processing is constant.

However, the tradeoff between regular polling and interrupt processing on reception is the packet latency. In polling methods, the latency can usually be fixed at some minimum, but the latency to begin processing a packet using interrupts may vary significantly.

We have experimented with these different styles of cell reception handling and our methods are described below. Instead of device interrupts, periodic timer callouts are used to poll the VudBoard. This scheme is similar to clocked interrupts discussed in [Smith and Traw, 1993].

When a timer callout executes, the status of the VudBoard is polled. If any cells have been received, the device driver suspends input DMA, swaps the filled input buffers with empty buffers, and restarts input DMA. It then queues for reassembly the input buffers that contain the cells. If there is any output buffers to be transmitted, an output DMA is also started. Finally the device driver schedules another timer callout.

The timer callout interval is chosen independent of data input rate. By allocating enough raw input buffers, we can ensure that between timer callouts the allocated buffer memory can never completely fill, so no cells can be lost at this point. Instead, the timer callout interval rate is chosen by trading off network round-trip time and CPU utilization. Longer timer values cause higher network round trip times, and shorter ones can cause excessive context switching. Currently, the timer callouts are executed every two milliseconds.

3.4.2 Device Input

If cells have been received, the driver temporarily halts the VudBoard in order to point it to a new kernel buffer, freeing up the other buffer to be passed to `vudrecv`, which queues it for reassembly. New buffers are then pre-allocated in order to minimize the time the host interface is "deaf" when they next need to be swapped.

`Vudrecv` places the buffer on a different queue, depending on whether the device is being directly read from/written to, or whether the device is in normal operation mode. In the latter case, the interface's `vud_softc` address is added to the front of the mbuf chain in order to distinguish data received from different interfaces, and a `vuintr` software interrupt is signaled.

3.4.3 Packet Reassembly

The buffers that are queued by the previous level may contain interleaved ATM cells over many different VCIs. The reassembly process must then look up the VCI to determine the type of connection, and reassemble the cell payloads into packets, stripping out the ATM header information and other related information. This is done by copying the data into different buffers reserved for reassembled packets.

When the software interrupt `vuintr` executes, the mbuf is dequeued and passed to the `vuinput` routine along with the pointer to the receiving interface's vud_softc. `Vuinput` allocates an mbuf for reassembly, grabs the first cell's VCI, uses that to grab the VCI's Input VCI Control Block (IVCB), and calls the procedure in the IVCB's `ivcb_proc` entry with pointers to the beginning and end of the mbuf's data for each mbuf in the chain. In the case where cells from different VCIs are in the same mbuf, the IVCB's reassembly procedure will return after reassembling all the cells that it can, and `vuinput` will call the next VCI's reassembly procedure using that VCI's IVCB.

Whenever it processes an end-of-frame cell, the length and checksum of the appropriate input packet is verified if required by the reassembly code segment, and the input packet is queued in the input queue appropriate for the VCI via a call to `vudeliver`.

The reassembly procedures can differ for different VCIs. Thus, cell reassembly is completely flexible for any need that may arise.

Type of read	# of copies
Raw device read	1
VuNet Cell Protocol	2
VuNet Frame Protocol, no checksumming	2
VuNet Frame Protocol, byte checksumming	2
VuNet Frame Protocol, longword checksumming	2
IP Protocol	2

Table 3.1: Number of copies for data getting from the network to user space. DMAs are not considered copies, but the data does traverse the system and I/O busses.

3.4.4 Packet Delivery

Depending on the kind of packet reassembled, packets are passed directly to the VuNet application socket, or may be passed to the IP Packet Handler. For IP packets, `vudeliver` places the packets into the IP input queue and schedules an IP software interrupt for additional protocol processing and further demultiplexing. Otherwise, through a series of calls, `vuso_usrrcv` is called, which invokes the proper system calls to write the data to the proper VuNet socket.

The total number of system copies required for reception are shown in Table 3.1. Note that the initial DMA from host memory to the device is not considered a copy, but the data does traverse the system and I/O busses.

3.4.5 Data Output

There are several ways to transmit data into the VuNet. Writing directly to the raw device puts the memory pages containing the data to be transmitted directly onto the device's transmit queue, immediately after being copied into kernel memory. This is not the standard method of writing to the network, however.

In order to write cells or frames to the network, the user process opens up a VuNet socket using the standard socket open and write calls. When a VuNet socket is opened, the type of VuNet socket is passed in (cell or frame mode), and for frame mode, the type of checksumming (none, byte, or longword) is specified. Any data written to the VuNet socket gets copied into kernel space, where the proper segmentation is performed and the ATM headers appended. This function requires another copy into a new buffer. This buffer is enqueued on the transmit queue.

IP traffic can also be transmitted over the VuNet. If the IP address of the destination host is a VuNet address, the device's output routines are invoked. The VCI, previously encoded within the VuNet host's MAC address, is determined by performing an *arp* operation on

Type of write	# of copies
Raw device write	1
VuNet Cell Protocol	2
VuNet Frame Protocol, no checksumming	2
VuNet Frame Protocol, byte checksumming	2
VuNet Frame Protocol, longword checksumming	2
IP Protocol	2

Table 3.2: Number of copies for data getting from user space to the network. DMAs are not considered copies, but the data does traverse the system and I/O busses.

the address. This VCI is used to determine the proper framing, enabling the segmentation of the IP packet into ATM cells via the proper segmentation routine.

During segmentation, information is added before each cell header. These are the transmit port and timer value for the cell. The segmented cells' memory page is enqueued on the transmit queue. Any pages placed on the device's transmit queue are set up so that the VudBoard reads directly from those pages for transmission into the network.

The total number of system copies required for transmission are shown in Table 3.2. Note that the final DMA from host memory to the device is not considered a copy, but the data does traverse the system and I/O busses.

3.5 VuNet Performance and Discussion

In this section, we first discuss the overall performance of the VuNet when used in conjunction with multimedia data for which it was designed. Then, we discuss how design decisions affected the overall performance and any other aspects of using the VuNet.

3.5.1 VuNet Performance

We conducted experiments to determine the maximum sustained data rate which could be achieved between the Alpha workstation hosts and the Vidboard. Frame rates for particular types of video are limited by the Alpha host, while others are limited by the Vidboard. For example, the frame rate of a full resolution, full color video stream is limited by the rate at which the Alpha can read data from the network, process and display it. The maximum video bit rate which can be displayed in the context of this system is about 73.7 Mb/s. The Vidboard is the limiting factor in the case of color streams, since the color-dithering algorithm implemented on it is computationally intensive.

	Bandwidth (MHz)
Host Interface Bursts	230.0
Continuous Through Host Interface	123.9
Continuous Through Software Reassembly	111.0
Continuous Delivered to Application Space	100.4
Continuous Displayed Video	73.7

Table 3.3: Bandwidth at various points in the VuNet.

When data arrives at the host interface, it must contend for the TURBOchannel bus to transfer its data into host memory. Due to bus arbitration, grant latency, and a wait period between successive transfers, the host interface is able to write to memory at a rate of roughly 230 Mb/sec. However, since the host cannot process data at this rate, it cannot be sustained indefinitely. The host memory is thus used specifically to absorb such high rate bursts.

When host processing is factored in and the data is reassembled and delivered to the application level, we find that we can indefinitely sustain a data rate of 100.4 Mb/sec of assembled data packets delivered to the application memory space. The device driver must process and reassemble raw ATM data at 111 Mb/sec in order to sustain this packet data rate to the application level.[5]

While the host interface is capable of a throughput of over 230 Mb/s, only 100 Mb/s is achieved because the host is running a full operating system with many other processes, including those that transmit into the network. Throughput measurements are made while running the full suite of ViewStation software within the UNIX operating system, using the X-window driver to display the video windows.[6]

While using a software-oriented approach buys extreme flexibility, the price incurred is an additional amount of processing overhead. Operating system "firewalls" impose a minimum on the number of times the data is moved over the system bus, limiting throughput.

3.5.2 Discussion

The consequences of this operating system overhead combined with software-based SAR is less than previously thought. The software SAR adds an additional operating system copy to the data paths (see Figure 3.1. However, we are still able to deliver our goal of 100 Mb/s to the application level on a DEC Alpha 3000/800 which is at least one generation old at this writing.

[5]The actual data transfer rate over the TURBOchannel bus is actually 123 Mb/sec, but this reflects transfer of 64 bytes per ATM cell.

[6]Video was not displayed when measuring top rate of data to the application.

Our method of VCI table management of the links leaves the link's tables vulnerable to corruption or tampering. While this security issue is not an issue in the desk area for which it is designed, it does provide us great flexibility in how we manage the tables. Chapter 4 details how we integrated a wide-area connection management scheme into the VuNet.

3.6 Summary

The design goals and hardware described in the chapter lead us to an interesting design for the network hardware and management software. By applying these towards the design of the VuNet ATM network, we are left with a network whose sole function is to transport bits from one host or peripheral to another. The only functions built into the network hardware are those necessary to transport the cells. All the connection management and other higher-level functions are handled by attached hosts. We thus provide a clean separation between the network transport and the network management.

The VuNet hardware, while designed to provide a flexible platform for experimentation, demonstrates that using this approach can make the network components fast and cheap. The VuNet hardware was an ideal platform to demonstrate how network control functions can be performed by the end nodes.

By pulling the segmentation and reassembly functions into the host software, we can reduce the complexity of the host interfaces and eliminate outboard memory needed to support these functions. This in turn buys flexibility which will be demonstrated in Section 4.3.1.

Also, by pulling some functions out of the switch port controllers and into the links, we allow the the network bit transport functions to become teased apart from the higher level network management functions. End nodes thus have better control and can handling specific situations, allowing this host-software based network to provide much better network performance.

Chapter 4

IP Switching

In this Chapter, we discuss how we integrate asymmetric endpoint flows into IP switching. IP switching [Newman *et al.*, 1996b] is an enabling technology that allows caching of routing information into network-level ATM switches and binding them to flow characteristics. It utilizes the high capacity and scalability properties of ATM and the network layer services provided by IP. An IP switch consisting of an IP switch router attached to ATM switch, as shown in Figure 4-1, routes packets as any other router, but also has the ability to cause packets in identified flows to bypass the IP switch router by utilizing the ATM switch's virtual circuit forwarding mechanism to take a fast path through the switch.

The key to the IP switching approach is the ability to identify the traffic flows that can benefit from taking the fast path through the ATM switch.

The IP switching approach also integrates hosts, which makes it ideal for implementation within the VuNet. Hosts participate in flow identification by using ATM circuits for long lived flows, just as a router in an IP switch would.

In this chapter, we will first give an overview of IP switching, which was developed by Ipsilon [Newman *et al.*, 1996b]. We discuss how the flows in Ipsilon's reference implementation have deficiencies in security, quality of service ability, and are too fine grained. We use the Ipsilon framework for setting up flows with our own types of flows, which take advantage of the asymmetric nature of traffic that we identified.

4.1 Overview of IP switching

This section describes the normal operation of an IP switch on data flows within a network. IP switching was developed elsewhere, but we use their framework to manage flow switching for new flow types that we identify.

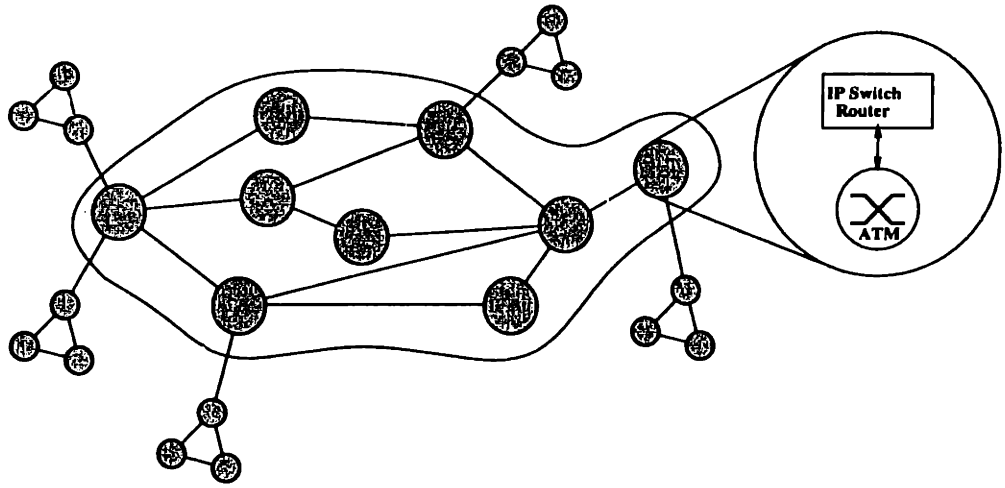

Figure 4-1: A router within the network is replaced by an IP switch - a combination of an IP switch router and an ATM switch. This combination provides the same exterior functionality as a standard router.

An IP switch functions as a router. By default, all traffic incoming to the switch is directed to the attached IP switch router and then out the appropriate link, behaving just like the IP router that this combination replaces.

However, the IP switch router has the ability to configure the attached ATM switch ports' entries for different VCIs. It thus can set up a special circuit that cuts directly through the ATM switch to the appropriate output port, thereby bypassing the IP switch router.

By default, only one inbound and one outbound circuit is established. This circuit serves as the control circuit over which other connections, such as the default IP circuit or a redirected flow's circuit are negotiated and set up. This channel is also used to establish adjacency between IP switches. If a default IP circuit is not yet initiated, IP packets can be processed on the control circuit.

All IP traffic initially flows over the established default IP circuit. When an IP switch router decides that a flow is worth switching, it uses the control channel to relay the information necessary for establishing the switched flow.

Figure 4-2 shows IP packets being forwarded to the the IP switch's router over the default IP VCI. When the IP switch router detects a flow and decides to set up a cut through circuit in its ATM switch, it sets up the special bypass in its switch, and signals a "redirect" to the upstream IP switch to redirect traffic on that flow to a new VCI, thereby bypassing the IP switch routing stage at the next hop.

What constitutes a flow and what information is needed upstream is described in the next section.

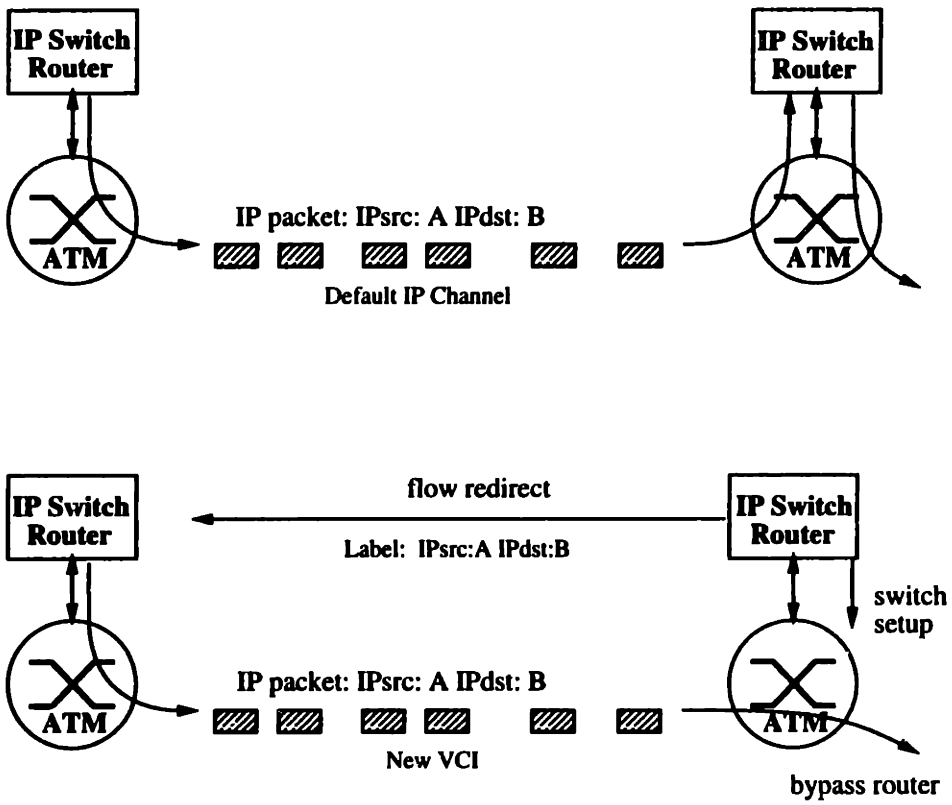

Figure 4-2: The IP switch router routes packets normally until it receives a redirect message from the downstream IP switch. It then forwards all packets matching the label on a new VCI, causing the packet to bypass the downstream IP switch router.

Flow type	Flow Label Identifying Parameters			
	Source IP	Source Port	Dest IP	Dest Port
TCP	X	X	X	X
Machine	X		X	

Table 4.1: Two types of flows are defined in the reference implementation. The *X*'s indicate identifying characteristics for a flow. Note that these flows are symmetric in nature; when a source process endpoint is used to identify a flow, a destination process endpoint is also used. When only the destination IP address is used, only a source IP address is used.

4.1.1 Flow Detection and Classification - A Symmetric Approach

The two types of symmetric endpoint flows in the IP switching reference implementation are Machine flows and TCP flows. The characteristics that identify these flows are marked in Table 4.1. Process endpoints define TCP flows and host endpoints define Machine flows. They are symmetric because both source and destination must be the same type of endpoint - either process or host. This symmetry is an important characteristic of these flows, as it places limitations on the scope of the flows.

4.1.2 IP Switching Mechanics

When a flow is detected, a new VCI is allocated and the attached switch is programmed to direct cells on this VCI to a specific port – the same port to which the router would have decided to route the packet. Then, a "redirect" signal is sent to the upstream IP switch. This redirect contains the flow label identifying parameters (see Table 4.1) and flow type. The upstream IP switch uses the flow label to create a new flow, which is added to that IP switch's flow tables stored in the IP switch router. Inbound packets on the default IP circuit are compared against all active flows of a determined type or types, and if a match is found, the flow type is used to determine which output VCI to use and how the TCP/IP header will be compressed.

The packet's header is compressed because the flow label that is stored in the flow tables of the upstream IP switch identifies the information used to identify the flow and hence some of the information in the TCP/IP packet header becomes redundant. This is not as important in conserving space as it is in preserving security. The header could be sent in uncompressed form, but care must be taken not to use this information at the destination. This security problem is discussed more fully in the next section.

The redirected flow is now bound to a specific VCI, which in turn is bound to a flow label. When a packet needs to be uncompressed, the flow label information is used to fill in the gaps in the packet. Table 4.2 shows the TCP/IP header information that is still sent with the packet. Note that TCP flows do not contain the Type of Service, Time to Live, and

Protocol fields, as they are generally considered non-variant over the life of a TCP flow. The same does not apply to Machine flows, however.

4.1.3 Drawbacks of Symmetric Flows

The symmetric endpoint (TCP and Machine) flows defined in the current IP switching reference implementation have some drawbacks. The limitations of both TCP flows and Machine flows stem from their symmetric definitions.

TCP Flows

The average Wide-Area TCP flow, as shown in Table 2.4, has just 20 packets. These TCP flows are too fine grained, causing many flows to be set up and torn down when performing routine network accesses to web servers, for example. Such a small number of packets per flow means that there would not be tremendous savings of IP switching TCP flows over routing the packets in the flow.

The TCP flow, defined by its symmetric inclusion of both IP addresses and TCP port numbers, greatly restricts the definition flows.

Machine Flows

While the Machine flows were probably introduced to address the graininess issue, the primary drawback of Machine flows is that they allow back-door methods of accessing services on machines behind firewalls.

An IP switch which also functions as a firewall should allow only certain classes of traffic to be switched through at the ATM level. For example, if a web server were behind an IP switch firewall, one would like to allow WWW traffic but block all other traffic from traversing the IP switch. However, this kind filtering can only be done with TCP flows, since flow labels for TCP flows contain the TCP port numbers.

If, in this example, a WWW flow is allowed to be switched onto a Machine flow, then the firewall has allowed a back door attack. The source endpoint of the Machine flow can now send packets of other services (such as telnet) on the allocated Machine flow. These packets will bypass the firewall.

Another drawback of Machine flows is that by bundling all the traffic between machines, quality of service cannot be allocated by service type. If a web server provides web pages and streaming audio, then any service guarantees that are allocated on a per-Machine-flow basis would necessarily bundle the two types of traffic, when guarantees are required for only part of the flow.

Flow type	Compressed Headers Contain:											
	IHL	TOS	Len	ID	Off	TTL	Proto	Cksum	SrcIP	DstIP	SrcPort	DstPort
TCP			X	X	X			X				
Machine	X	X	X	X	X	X	X	X			X	X

Table 4.2: When flows are redirected, the headers can be compressed, since the packets are bound to a VCI for which a flow label with additional information exists.

Flow type	Flow Label Identifying Parameters			
	Source IP	Source Port	Dest IP	Dest Port
TCP	X	X	X	X
Machine	X		X	
Session1	X		X	X
Session2	X	X	X	
Server			X	X

Table 4.3: The new asymmetric endpoint Session and Server flows exploit the aggregation of traffic around servers and the asymmetry of traffic flow to and from servers.

Packets sent on Machine flows would still be demultiplexed using the existing TCP/IP stacks and any network service guarantees would not be carried into the host operating system.

While Machine flows increase the number of packets per flow, the price is the loss of flow switching through security boundaries, the loss of ability to provide quality-of-service guarantees for particular service classes, and the loss of binding network connections to applications.

We address all of these concerns by taking an asymmetric approach to defining flows. Our solutions will be described in the next section.

4.2 Asymmetric Endpoint Flows

As discussed in Section 2.2, the traffic near servers is very asymmetric. The asymmetry in packets per flow suggests taking a similar asymmetric approach in detecting flows.

To exploit asymmetries, we introduce the notion of asymmetric endpoint flows. The first type of new flow, a Session flow, indicates a particular source process endpoint and destination host endpoint, or a source host endpoint and destination process endpoint. The effect of this type of flow is to bundle multiple TCP connections from a client to a server service, or from a server service to a client.

For example, a web client which downloads a page with images initiates a new TCP connection for each individual file, which translates into a TCP-style flow for each file. Using Session flows bundles all of the TCP connections into a single flow.

The second type of new asymmetric endpoint flow is the Server flow, which identifies only a destination process endpoint without respect to any client information.[1] The effect of this flow is to bundle all packets going to a server into a single flow. In the case of a web server, all inbound document requests end up on the same flow, as are all inbound TCP acknowledgments.

The introduction of these flow types can have important ramifications for the capacity of IP-switched networks and other similar tag-switched networks.

Assuming that a significant load on a fully loaded IP switch is that of forwarding un-labeled packets and setting up new flows, the reduction of the number of flow setups saved by using Server flows can potentially reduce the workload and resources required by a factor of two or more. We will compare the costs savings in Chapter 5.

4.2.1 Asymmetric Endpoint Flow Benefits

Asymmetric endpoint flows address the drawbacks of symmetric endpoint flows. The Session flows maintain their binding to a particular service, so that service guarantees can still be allocated based on the demands of only that Session flow. A server that delivers documents and streamed audio offers two distinct services for which service guarantees can be different based on using Session flows.

In addition, Session flows do not have the security problems that Machine flows do, since it is not possible to use a Session or a Server flow to attack another service on the destination machine. And, service guarantees can be carried into the server's operating system on a per-flow basis.

Furthermore, Session flows result in approximately the same amount of work as using Machine flows, as shown in our work simulations in the next chapter, The reason is that often, only one type of service is used on any particular server machine, making a Machine flow equal in work to a Session flow for that server.

Server flows bundle all incoming traffic to a particular service. Many client-server applications are bursty by nature, and inbound traffic streams that gets bundled behave better on a statistical basis and are also easier to predict. Service guarantees can be allocated on a Server flow basis for all traffic inbound, reducing overhead required and wasting fewer reserved resources.

[1] We realize that this is beyond our initial strict definition of a flow. However, we broaden the types of endpoints here to include the multiple unspecified source endpoints of a Server flow.

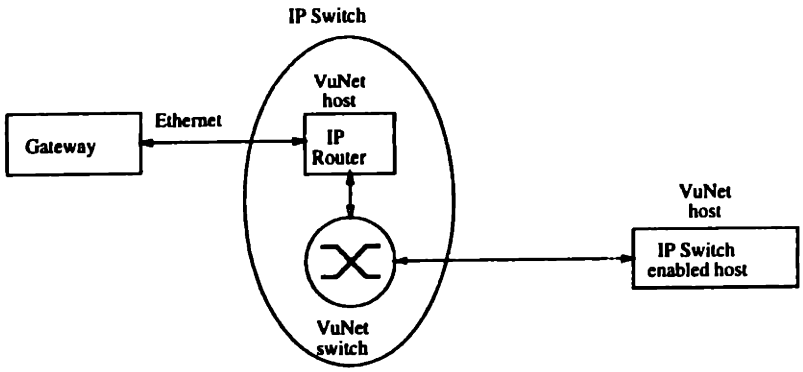

Figure 4-3: VuNet hosts and switches are converted for use in an IP switching environment.

However, while Session flows are straightforward to implement with the IP switching framework, Server flows pose problems from merging packets from different VCIs into a single VCI. We will see in the next section how this is addressed in the VuNet.

4.3 VuNet II: Integrating IP Switching

In the first phase of the VuNet, as we have seen in the preceding chapters, hosts direct traffic flow throughout the network and set up connections throughout the local trusted area. In essence, this is very similar to the IP switching style of traffic management. The "host" in the IP case is the IP switch router which has the ability to set up connections in the local ATM switch. This host also makes decisions based on the best use of network versus host resources, and cuts data through the switch whenever it deems necessary.

Because the method of control for VuNet hosts and IP switch routers, and the flexibility of the VuNet, the integration of the IP switching into the VuNet was straightforward. We call the integration IP switching with the earlier hardware the VuNet II. The novel aspect introduced in the VuNet II is the ability to identify and switch flows.

We have integrated the VuNet driver and the Ipsilon host reference software into the NetBSD kernel, and are currently using two NetBSD machines running the Ipshost software as an IP-switched network with an IP-switching enabled host, as shown in Figure 4-3. This network, along with simulations, was used to test and evaluate our flow design.

The VuNet II IP Switching implementation is designed around an Ipsilon IP Switching module attached to the VuNet device driver. The Ipsilon module was designed to work with multiple instances of an Ipsilon interface using loadable device drivers. However, in our NetBSD implementation, there is only a single instance of an Ipsilon module running in conjunction with a single instance of our VuNet device driver.

Each port on the switch to which the host is attached is assigned a separate incoming control VCI, a separate outgoing control VCI, and a separate default IP channel VCI. In this way, the inbound and outbound VCI space can be managed among the different outbound ports. This provides a more efficient use of resources than approaches that distribute the VCI space to each port.

4.3.1 The VCI Fusing Problem

One of the difficulties in implementing Server flows is that merging traffic from two VCIs into a single VCI is not a straightforward task. The problem is illustrated in Figure 4-4. When the cells from two VCIs are simply merged onto a single VCI, it is impossible to distinguish the cells to reassemble the packets properly.

The reason for this is that the methods for segmentation and reassembly included in the standard ATM Adaptation Layers (AALs) [Bellcore, 1992] did not anticipate the possibility of this kind of VCI merging, and hence are insufficient to solve this problem.

In AAL5 segmentation [ITU, 1996b], a trailer is added to the packet and then the entire packet is fragmented into ATM cells. A bit in the ATM cell header indicates the last cell of a packet. Thus, the packet can be reassembled by using all ATM cells that arrive in sequence from the first ATM cell in the packet to the last ATM cell (as indicated by the header bit). If cells from multiple AAL5 streams are multiplexed onto a single circuit, the proper reassembly of cells is lost, since AAL5 depends on sequential cells on a circuit forming a single packet.

In AAL3/4 segmentation [ITU, 1996a], part of every cell payload is used for a cell sequence number and multiplexing identification (MID). The receiver can use this information to reassemble packets, even if they are multiplexed and interleaved onto a single VCI. However, MIDs must be dynamically allocated which leads to increased overhead. Furthermore, the MID space is not large enough to support the many segmentation points that are possible with Server flows.

Some have proposed hardware methods which have knowledge of the segmentation process to block the introduction of a cell from a new packet into the middle of a cell stream of another packet [Turner, 1996]. While this solves the problem, it is not scalable, and unnecessarily complicates the switch, which should not have to know about how packets are segmented.

In the VuNet, we can use the flexibility of our software-based segmentation and reassembly to solve this problem. Because we can assign SAR routines on a per-VCI basis, SAR for Server flows that need to be merged can introduce the point of segmentation's unique IP address into each cell in addition to marking the end of each packet in the ATM cell header. As long as each segmentation point processes packets sequentially (i.e. it does not interleave

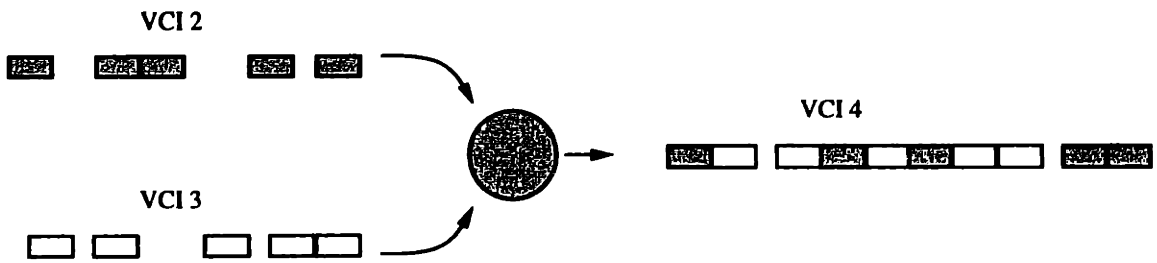

Figure 4-4: When cells from two VCIs are simply merged to form a single VCI, it is impossible to distinguish the cells to reassemble the packets properly. Other techniques must be used.

cells from different packets), the final reassembly point can distinguish the unique source of all incoming cells and properly reassemble packets.

Downstream IP Switch VCI Table Management

The standard method of IP switching requires that the IP switch router manage the VCI tables of all inbound links. The reason for this is that the VCI tables are generally associated with the inbound port controllers of an ATM switch. For standard ATM switching equipment, it is much more difficult, if not impossible, to manage the inbound port controller of another ATM switch directly.

In the VuNet, hosts are able to manage the link tables of any link within the system. An IP switch router in the VuNet has the ability to manage the inbound VCI tables of all downstream switches. In fact, in the VuNet, this is easier than managing the inbound VCI tables of the ATM switch to which it is directly attached. We call this method of IP switching "Preemptive IP Switching," which implies that the flow cutover action takes place earlier than in regular IP switching.

4.4 Summary

The data transmitted by a web server are good candidates for IP switching of TCP flows. However, the requests and acknowledgments are better candidates for asymmetric endpoint flow switching as opposed to symmetric endpoint flow switching. This introduces problems of merging cells onto a single VCI, but we can solve this problem using the flexibility of the VuNet.

We have shown how the asymmetric endpoint flows can be implemented in the VuNet. In the next section, we present our simulations and work model to show how good asymmetric endpoint flows are as compared to symmetric endpoint flows.

Chapter 5

IP Switching Model and Network Trace Simulations

In the previous chapters, we have identified the existence of asymmetric endpoint flows. We have also hypothesized that using asymmetric endpoint flows in an IP switched environment would lead to greater routing capacity than using symmetric endpoint flows, which in turn has greater capacity than a standard router.

We have implemented a limited asymmetric endpoint flow IP switched environment on the VuNet. However, the VuNet is not of a sufficient scale to validate that asymmetric endpoint flows perform better under real traffic loads. Thus, we turn to simulations to show how much better asymmetric endpoint flows are. We can "play back" publicly available network traces from the wide area to simulate the load on an IP switch.

We created an IP switch simulator to analyze network traces from the local and wide areas. Using a work model of IP switching that we developed, we compared the work required to a work model of IP routing to determine the cost savings. In our simulations, our network with extended flow types processed the packets in the traces with one-fourth the total work required as compared to the reference IP switch implementation, and one-sixteenth the amount of work required to route the traces.

In this chapter, we first build a model of IP switching based on functional modules. Then, we attach work functions to these modules. We describe the simulator which was used to analyze network traces and the output that the simulator produces. By reducing the simulator data to just a few parameters, we see how the cost savings from IP switching varies based on IP switching parameters and module costs.

Using the simulator output, we study how our asymmetric endpoint flow types affect the total work done by an IP switch. We then discuss the parameters of the model which provide the greatest impact towards increasing the capacity of an IP switch.

Figure 5-1: The functional description of a router is composed of Input, Fowarding and Output modules.

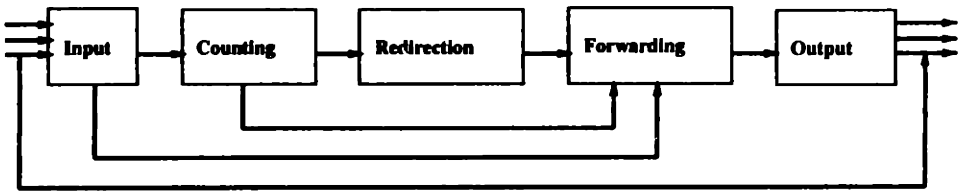

Figure 5-2: The functional description of an IP switch is composed of modules which handle flow tracking and setup in addition to those that compose a router.

5.1 Functional Model for Routing and IP Switching

An IP Switch can be broken up into several functional modules. Each packet passes through some subset of these modules. These modules can be used to model both a router and an IP switch. The modules and their functional description are as follows:

- **Input:** Recovers framed data packets from network and delivers them to the upper layers.
- **Counting:** Searches for and updates flow information associated with a packet.
- **Forwarding:** Decides the proper output route for a packet.
- **Redirection:** Sends out control information to the peer IP Switch in order to manage flows. This includes setup as well as tear down work for each flow.
- **Output:** Segments and/or frames an IP packet and delivers it to the network.

Figure 5-2 represents the IP switch. Note the paths which bypass sets of modules. The first bypass path is for packets which are switched through at the ATM switch level. The second bypass path provides a function which is identical to the router. This path is for packets which are always forwarded without keeping any statistics on flows. The inner bypass path represents packets which may belong to a flow but do not trigger a redirection action.

5.2 IP Switching Work Model

In this section, we develop a model for the amount of work an IP switch demands as compared to a router.

First, we state our assumptions that we use to simplify our work model. We show how packets in a network trace can be categorized as one of five types. Then, we show how the model can be reduced to a few parameters that can be easily collected from network traces.

5.2.1 Assumptions

In order to simplify our work model, we make some assumptions that we will revisit later. First, we assume that the switch capacity at the ATM level of the IP switch is such that it will never be a bottleneck so it is essentially free. Therefore, all the work is done by the IP switch router.

Next, we assume that each of the modules shown in Figure 5-2 can be associated with a work function, and that this work function does not vary from packet to packet.[1]

Finally, we also assume that we are not bound by memory resources in the IP switch and that we can track and account for all active flows, and that the cost of accounting for a flow is not related to the number of active flows.[2]

5.2.2 Packet Classes

The primary role of the simulator is to collect information regarding each packet in the network trace. In order to model the IP switching process, each packet passing through an IP switch can belong to one of several non-overlapping classes, based on the types of flows being used in the simulation. The different classes are described as follows:

- **Observe:** A packet which may be part of a flow if enough packets arrive with same label.
- **Setup:** A packet which triggers a flow setup after the IP switch has decided that it has enough information to do so.
- **Straggler:** A packet arriving on the default IP channel which belongs to a flow in which the setup has been initiated but not yet completed.
- **Bypass:** A packet which is switched at the ATM level.

[1] This implies that the router is limited by packet processing and not bit throughput.

[2] However, we may be bounded by the size of the label space.

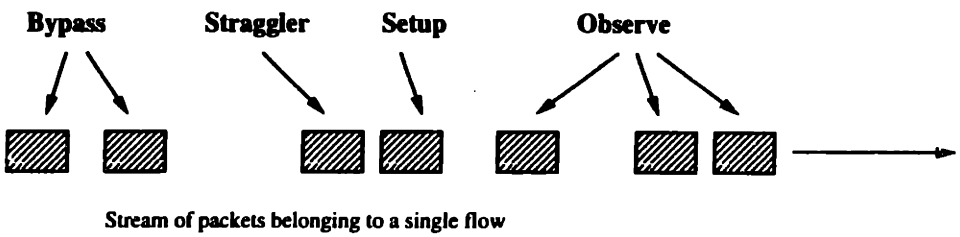

Figure 5-3: Each packet in a flow can be classified as one of five classes. Each of these class of packets has a different work function.

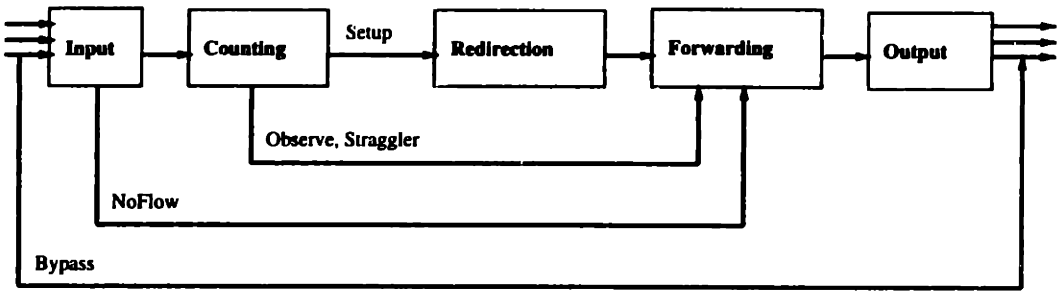

Figure 5-4: Each type of packet takes the same path through the various functional modules of an IP switch.

- **NoFlow:** A packet which never will be switched.

The first four packet classes (**Observe, Setup, Straggler, Bypass**) deal with packets that belong to identified flows. The **NoFlow** class is for packets which the IP switch will always route individually.

Figure 5-3 shows how packets in different classes can be classified. The packets at the beginning of the flow cause the IP switch to begin **observing** packets and tracking the flow's statistics. After a certain threshold is reached, the next packet triggers a flow **setup**. Some of the packets may continue to arrive on the default channel before the flow setup has been completed; these are the **stragglers**. Finally, all the other packets in the flow **bypass** all of the modules and are cut through at the ATM switching level. Figure 5-4 show the various paths that are taken through an IP switch for each of our packet classes.

Since packets must belong to one of these five flow classes, the probability of belonging to any one of these classes is conserved:

$$P_{Observe} + P_{Setup} + P_{Straggler} + P_{Bypass} + P_{NoFlow} = 1 \tag{5.1}$$

Furthermore, each class of packet has an associated work function W. If we pick a random packet from the trace, the expected work required to process packet p:

$$E(W_p) = P_{Observe}W_{Observe} + P_{Setup}W_{Setup} + P_{Straggler}W_{Straggler} + P_{Bypass}W_{Bypass} + P_{NoFlow}W_{NoFlow} \quad (5.2)$$

where the probability of belonging to a particular class depends strongly on a number of factors which will be discussed.

5.2.3 Module Work Models

The simulator must be able to model the IP switch's method of determining when to switch flows. Since the IP switch cannot know how long a flow is going to last, a threshold, $N_{Threshold}$, is generally used. When this threshold is reached for a particular flow, the decision is made to redirect the flow. The exact optimal threshold must be determined carefully, and is a function of how much work it costs to handle a flow relative to the cost of routing the packet.

Hence the simulator must be able to capture this kind of information. The simulator records the number of packets per flow for every flow and builds a histogram of flows versus packets. Using this histogram, it is possible to derive the total work required to handle a flow based on the packets per flow and the work functions.

We assign costs to each of the different packet classes listed in Section 5.2.2. Each packet class is associated with a single path through the functional model shown in Figure 5-2 and described in Section 5.1. Thus, the work for each type of packet can be derived:

$$W_{Observe} = W_{Input} + W_{Counting} + W_{Forwarding} + W_{Output} \quad (5.3)$$
$$W_{Setup} = W_{Input} + W_{Counting} + W_{Forwarding} + W_{Redirect} + W_{Output} \quad (5.4)$$
$$W_{Straggler} = W_{Input} + W_{Counting} + W_{Forwarding} + W_{Output} \quad (5.5)$$
$$W_{Bypass} = 0 \quad (5.6)$$
$$W_{NoFlow} = W_{Input} + W_{Forwarding} + W_{Output} \quad (5.7)$$

Note that the cost of routing a packet is the same as the cost of a **NoFlow** class packet:

$$\begin{aligned} W_{Routing} &= W_{Input} + W_{Forwarding} + W_{Output} \\ &= W_{Noflow} \end{aligned} \tag{5.8}$$

Using these equations, we can derive the work necessary to switch any particular flow with a known number of packets N^i. By using Figure 5-3, we can easily determine that for the i^{th} flow with N^i packets, the total work W^i_{Flow} required for switching the flow is:

$$\begin{aligned} W^i_{Flow} &= N^i_{Observe} W_{Observe} + W_{Setup} + N^i_{Straggler} W_{Straggler} \\ &\quad + (N^i - (N^i_{Observe} + 1 + N^i_{Straggler})) W_{Bypass} \\ &= (N^i_{Observe} + N^i_{Straggler}) W_{Observe} + W_{Setup} \\ &= (N_{Threshold} + N^i_{Straggler}) W_{Observe} + W_{Redirect} \end{aligned} \tag{5.9}$$

for $N^i >= (N_{Threshold})$. $N_{Threshold}$ is defined by the IP switch's configuration as the threshold number of packets before switching a flow and is not dependent on any flow.

Since the threshold is an IP switch configuration parameter, by definition:

$$N^i_{Observe} = N_{Threshold} - 1 \tag{5.10}$$

for all switched flows i.

If $N^i < (N_{Threshold})$, then the total work to process this flow is given by:

$$W^i_{Flow} = N^i W_{Observe} = N^i W_{Routing} + N^i W_{Counting} \tag{5.11}$$

Time is spent counting up statistics for this short-lived flow, but since the threshold is never reached, the work is wasted. For routing a flow, the work is:

$$W^i_{Routing} = N^i W_{Routing} \tag{5.12}$$

This states that for short-lived flows, the cost of IP switching is greater than that for routing the packets in the flow individually.

It makes sense to compute a ratio of the amount of work it takes to switch a flow i compared to the amount of work it takes to route the same flow i. This ratio is:

$$R^i = \frac{W^i_{Flow}}{W^i_{Routing}}$$

We can find the total work ratio by summing R^i over all flows i.

However, we can achieve the same result if we use the probabilities introduced earlier in Equation 5.2. We find that:

$$\frac{W^{total}_{Flow}}{W^{total}_{Routing}} = \frac{P_{Observe}W_{Observe} + P_{setup}W_{Setup} + P_{Straggler}W_{Straggler} + P_{NoFlow}W_{NoFlow}}{W_{Routing}} \tag{5.13}$$

Using Equations 5.2 and 5.3-5.7, this equation reduces to:

$$\begin{aligned}\frac{W^{total}_{Flow}}{W^{total}_{Routing}} &= 1 - P_{Bypass} + P_{Setup}\frac{W_{Counting} + W_{Setup}}{W_{Routing}} + \\ &\quad P_{Observe}\frac{W_{Counting}}{W_{Routing}} + P_{Straggler}\frac{W_{Counting}}{W_{Routing}} \\ &= 1 - P_{Bypass} + P_{Setup}\frac{W_{Setup}}{W_{Routing}} + (P_{Observe} + P_{Setup} + P_{Straggler})\frac{W_{Counting}}{W_{Routing}}\end{aligned} \tag{5.14}$$

Equation 5.14 is arranged from largest term to smallest term.

5.3 Simplification of Work Model

While we have an equation that will tell us how to compute the cost savings from IP switching, it is still a bit cumbersome. In this section, we discuss how we reduce the number of parameters that need to be collected from network traces.

5.3.1 Flow Setups

In an IP switching implementation in equilibrium, every flow setup requires an additional three message exchanges: one to send the flow setup, one to tear down the flow, and one to

acknowledge the flow deletion.

We assume that each of these messages costs $W_{Routing}$ to handle. This is a realistic and even conservative assumption. Our messaging packets are likely to be smaller than the average sized packet; they may even be smaller than the TCP header. In addition, we assume that the average work spent in looking up or creating a flow entry is on par or less than the work spent in looking up a routing table entry $W_{Forwarding}$. We will discuss this assumption in more detail in Section 5.4.5.

Hence, the minimum value of the ratio of $\frac{W_{Setup}}{W_{Routing}}$ possible to set up a flow to the work required to route a packet is roughly 4. Equation 5.4 embodies this high ratio in the $W_{Redirect}$ term. This term includes all the work necessary to both setup and tear down the flow.

To simplify our model, we will assume that:

$$W_{Redirect} \approx 4W_{Routing} \tag{5.15}$$

Thus,

$$W_{Setup} \approx O_{Setup}W_{Routing} + W_{Counting} \tag{5.16}$$

where $O_{Setup} = 4$ represents the setup messaging overhead.

We also assume that there is only one setup per active defined flow. This may not be true for conditions where a flow does not cut over within a short amount of time, for example, when the upstream IP switch's flow tables are full. In that case, more redirects are eventually generated. This is taken to be an error condition and we do not account for it here.

Note that this might be similar to a cache optimization problem were it not for the high value of O_{Setup}, the cost of setting up a flow. With such a high value of setup overhead, we must carefully consider whether or not we want to "cache" a flow.

5.3.2 Stragglers

While the number of stragglers in a particular flow $N^i_{Straggler}$ can be derived from an examination of the network traces, we can simplify our model if we assume that the number of stragglers that arrives per flow is relatively constant, assuming a given setup latency and a particular point within the network. We also assume that the number of stragglers per flow is independent of both the setup time of the flow and the switching threshold.

We measured the exact number of stragglers by collecting the delay since the very first packet in the flow. This tells us how many of the packets will be stragglers based on the

Flow Types	LCS	WAN
TCP	0.30%	3.95%
Machine	0.27%	2.76%
Session	0.28%	3.81%
Server	0.20%	3.35%

Table 5.1: $P_{Straggler}$ with a setup delay of 25 ms for the LCS and WAN traces, for various flow types.

Flow Types	LCS	WAN
TCP	.080	0.484
Machine	.123	0.545
Session	.116	0.617
Server	.136	0.892

Table 5.2: $\frac{P_{Straggler}}{P_{Setup}}$ with a setup delay of 25 ms for the LCS and WAN traces, for various flow types. This can also be read as the expected number of straggler packets per switched flow.

types of flows used. In our simulator, we assume that the flow setup delay is 25 ms.[3]

This represents the **one-way** propagation delay from the originating IP switch to the upstream neighbor. Only a one-way delay is used because the only packets that will be stragglers are those sent on the default channel from the time the redirect message is sent upstream to the time it is received and processed upstream. We can then determine $P_{Straggler}$.

Our 25 ms flow setup delay is a conservative estimate, representing a small amount of processing and a large propagation delay. In 20 ms, a signal travels approximately 4000 km through an optical fiber.

We realize that many factors affect the counting of the stragglers. Foremost is that we count stragglers from the first packet in all our flows, which in reality is starting in the middle of many of the flows recorded, so our profiling does not actually give the true number of stragglers from the true beginning of all flows. However, we believe that the probabilities measured and used will be characteristic of the true number of stragglers for a particular collection point.

Also, the number of stragglers should be dependent on the bandwidth of the incoming link. More packets can be packed into a certain time interval if the bandwidth is higher, and

[3]We only deal with Stragglers due to the first order effect of switching and not higher order effects of chained IP switching.

the time interval cannot be reduced. This can be partly seen in the difference in the WAN traces straggler probabilities and the LCS trace straggler probabilities shown in Table 5.1.

Once we determine the number of stragglers per flow, we can relate this number back to the number of Setup packets to simplify Equation 5.14. Table 5.2 shows the fraction of straggler cells per switched flow measured in our traces. Thus we can eliminate the $P_{Straggler}$ term from Equation 5.14.

Downstream IP Switch VCI Table Management

In Section 4.3.1, we showed how we can shift the VCI table management from the inbound ports to the outbound ports in the VuNet. This requires no additional messaging overhead because VCI table management can be done by prepending a connection management ATM cell to any packet.

If the IP switch router manages the outbound VCI tables, then there are no stragglers.[4] This will provide a reasonable increase to an IP switch's performance, particularly in the wide area. Note that a message still must be sent to an upstream IP switch router in order to allow upstream IP switches to redirect traffic onto the VCI selected by the IP switch router, and that two more messages still must be sent regarding the flow deletion.

In the wide area, the elimination of stragglers increases the performance by roughly 15% when using server flows; for the gateway trace, the performance increase is more modest at roughly 2%. However, this method of VCI table management is more suited for the local are where the administrative boundaries are clear.

5.3.3 Deriving Costs as a Function of Switching Threshold

In Section 5.2.3, we have derived an equation for determining the increase in performance given probabilities which can be derived from a flow trace. In Section 5.3.1 we simplified the model by relating the setup cost W_{Setup} to other parameters of the model. In Section 5.3.2 we showed how to eliminate the $P_{Straggler}$ term.

Combining all these, we have:

$$\frac{W_{Flow}^{total}}{W_{Routing}^{total}} = 1 - P_{Bypass} + O_{Setup}P_{Setup} + (2+S)P_{Setup}\frac{W_{Counting}}{W_{Routing}} + P_{Observe}\frac{W_{Counting}}{W_{Routing}} \tag{5.17}$$

[4] Again, we ignore higher-order stragglers from chained IP switching.

where $O_{Setup} = 4$ is the setup messaging overhead and S is the expected number of stragglers per flow setup packet, as shown in Table 5.2.

Of the remaining probability terms, P_{Bypass} is dependent on P_{Setup}, $P_{Observe}$, and P_{NoFlow}. P_{NoFlow} is dependent only on the types of flows that are never switched is constant. P_{Setup} and $P_{Observe}$ are dependent on the switching threshold $N_{Threshold}$.

If we know the distribution of packets per flow, then:

$$P_{Setup}(N_{Threshold}) = \frac{1}{N_{Packets}} \sum_{i=N_{Threshold}}^{\infty} Numflow_i \tag{5.18}$$

$$P_{Observe}(N_{Threshold}, N_{Threshold} = 1) = 0 \tag{5.19}$$

$$P_{Observe}(N_{Threshold}, N_{Threshold} > 1) = P_{Observe}(N_{Threshold} - 1) + P_{Setup}(N_{Threshold} - 1) \tag{5.20}$$

where $Numflow_i$ represents the number of flows with i packets, and $N_{Packets}$ is the total number of packets in the trace and can be derived from the flow distribution:

$$N_{Packets} = \sum_{i=1}^{\infty} i * Numflow_i \tag{5.21}$$

These equations can be reasoned out using common sense. If flows are switched immediately ($N_{Threshold} = 1$), then no packets will be observed, and all flows that exist will generate a flow setup. If $N_{Threshold} = 2$, then we will setup a flow for every flow except for flows of length one packet, and we will observe one packet for every flow. If $N_{Threshold} = 3$, then we will setup a flow for every flow except for flows of length three or more packets, and we will observe all the packets that were observed when $N_{Threshold} = 2$, plus all the packets that were setup packets when $N_{Threshold} = 2$ now become packets that are observed.

Equation 5.20 can be rewritten as:

$$P_{Observe}(N_{Threshold}, N_{Threshold} > 1) = \sum_{i=1}^{N_{Threshold}-1} P_{Setup}(i) \tag{5.22}$$

which is less intuitive but perhaps easier with which to work.

We can now use the distribution of packets per flow to show how the total work varies according to just two parameters: the switching threshold $N_{Threshold}$ and the counting

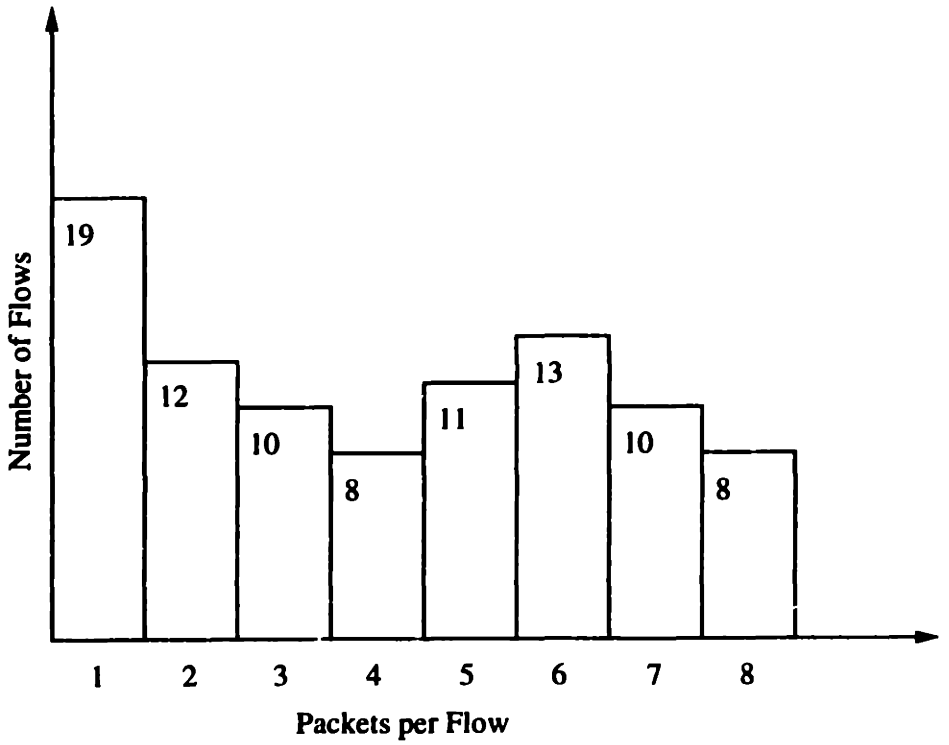

Figure 5-5: The number of Setup and Observe packets in a trace can be derived from the distribution of packets per flow.

overhead $\frac{W_{Counting}}{W_{Routing}}$.

For example, Figure 5-5 shows a distribution of packets per flow. 91 flows represent 372 packets. If the switching threshold is 1, all 91 flows get set up immediately for $P_{Setup} = 91/372 = 24.5\%$. No packets are observed. If the switching threshold is 3, then only 60 flows are set up: $P_{Setup} = 60/372 = 16.1\%$. However, $P_{Observe} = 91/372 + 72/372 = 43.8\%$.

5.4 Discussion of Work Model and IP Switching

Equation 5.14 reveals some interesting aspects. In this section, we discuss the meaning of the various terms in Equation 5.14 and the impact of various parameters on the throughput of the IP switch. First, we discuss the fairness of our model and compare it to other models of IP switching.

5.4.1 Fairness

We have made some assumptions in our model that need some discussing here. First, we assume that our costs are constant over all packets, or, at the very least, that we use average costs over all packets. Note that Equation 5.14 contains only three work terms: the cost of routing a packet, the setup cost of a flow, and the counting cost of updating flow information.

Since packet sizes are variable, the work required to route a packet may be dependent on the length of the packet, because the cost to read in and send out a packet is proportional to the packet length. However, advanced routers only transfer packet headers in to the routing subsystem, eliminating the need to read whole packets – only packet headers need be read.

This router optimization can be applied to IP switches since no more information is required to switch a flow than is contained in the packet's TCP header. While the performance of a router with this optimization increases because the average work to route a packet drops, an IP switching implementation can take advantage of the same optimization.

We also use real traces in our simulations, so that the packet loads and the flow endpoints represent real loads and real endpoints. While we can derive an overall performance increase for the whole trace, the performance increase may not be uniform over the duration of the entire flow. Equation 5.14 does not specify any time interval, so the probabilities can be broken down on a second-by-second basis if need be.

No other models of IP switching exist. An informal analysis offered in [Newman *et al.*, 1996b] states that in an efficient implementation, one only needs to take the equivalent amount of work needed to setup flows and add to it the work required to route the non-flow packets. While our model approximates this statement, we take into account more implementation parameters such as the counting overhead, the setup messaging overhead, and the switching threshold.

5.4.2 Flow Timeouts

When flow expirations are factored into flows, flows which were previously one flow may now become two or more flows, since the process to set up a redirection is started again after the flow entry is timed out. The simulator can time out flows every so often. In order to determine what an appropriate timeout would be, we can collect flow information for flows which never expire. We can examine the packet inter-arrival times for these flows to estimate the best flow life time.

Increasing the timeout interval decreases the probability that a flow will be timed out. However, it causes more flows to be active at any given time, and does not decrease P_{Setup} or $P_{Observe}$ by much. We select the timeout interval by taking a point beyond the knee of

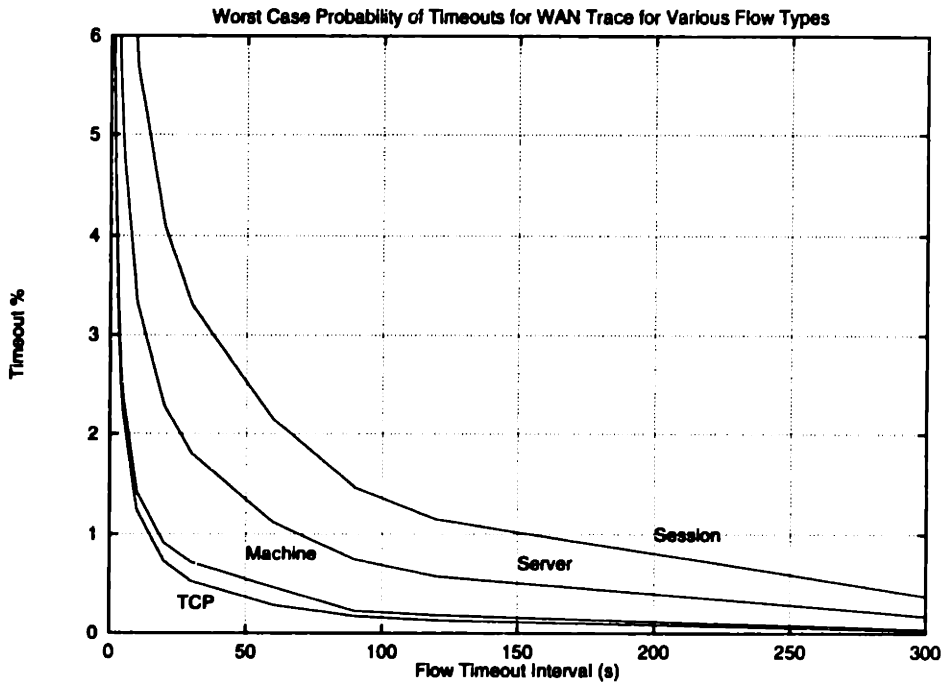

Figure 5-6: Worst Case Probability of Flow Timeouts for WAN Trace for Various Flow Types

the curve in Figures 5-6 and 5-7. Our 90 second timeout interval used in our later analysis corresponds closely to proposed values of the timeout interval by others [Newman *et al.*, 1996b, Lin and McKeown, 1997].

5.4.3 Balancing the Switching Threshold

If the switching threshold $N_{Threshold}$ is the only adjustable parameter, we see from equations 5.18-5.20 that the $P_{Observe}$ and P_{Setup} terms work against each other and will not both decrease as the threshold is adjusted. If we reduce the $P_{Observe}$ by reducing the $N_{Threshold}$, P_{Setup} increases since we begin to set up flows for shorter-lived connections. There is an optimal threshold for flow cutover for various sets of module work parameters. There are even different optimal thresholds for varying service types. This will be discussed in more detail in Section 5.5.3.

5.4.4 Asymmetric Endpoint Flows

Finding the optimal switching threshold only affects the overall work modestly. A better strategy to decrease the total work load is to decrease both P_{Setup} and $P_{Observe}$ simultaneously. This cannot be done by adjusting the switching threshold.

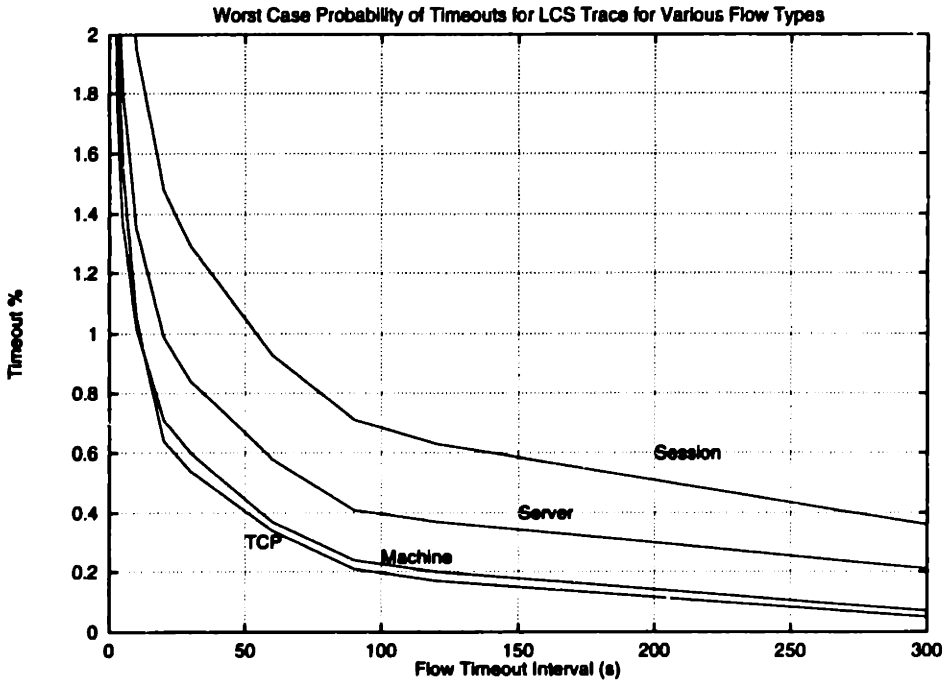

Figure 5-7: Worst Case Probability of Flow Timeouts for LCS Trace for Various Flow Types

But, we can do this by broadening the definition of a flow. This is what occurs with our asymmetric endpoint Server and Session flows. TCP flows now become bundled, with more packets bypassing the switch and fewer packets being counted and triggering flow setups. The effect of session and Server flows is to increase P_{Bypass} and decrease both P_{Setup} and $P_{Observe}$. We will see the detailed comparison of asymmetric endpoint flows to symmetric endpoint flows in Section 5.5.3.

5.4.5 Reduce Flow Accounting Costs

Another strategy to reduce the total workload is to decrease the ratio of $\frac{W_{Counting}}{W_{Routing}}$. If flow tables are organized like routing tables, then when there are a large number of active flows, $W_{Counting}$ is dominated by the time required to search for and find the flow entry, rather than by the time necessary to update this entry.

The major difference between looking up a flow table entry and a routing table entry is that a routing table entry must return the entry with the longest prefix match as the destination address. There may be several entries in the routing table that match somewhat, but the best match must prevail.

Since flow labels are unique, strategies such as MD5 hashing [Rivest, 1992] on the flow label can be used to reduce the time required to find the flow entry to a cost that is less than

that required to retrieve a routing table entry. There is additional work required to update the flow statistics, but the work is minimal.

It turns out that realistic ranges of the counting overhead can cause dramatic differences in whether or not one should switch flows that have been identified by others as "short and not worth switching." This will be discussed in more detail in Section 5.5.3. In fact, the flow accounting work is an important part of our model because it helps to yield surprising results that contradict earlier, simpler models [Newman *et al.*, 1996b].

5.5 Simulation Results and Analysis

Packet traces from the network can be recorded for later analysis. These traces give detailed information about packets which pass the collection point, including timestamp, and full TCP/IP headers. Packet traces can be recorded with the LINUX utility `tcpdump`.

Our data analysis used several packet traces which we recorded from our laboratory's main gateway. We also used publicly available traces from wide-area access points. The wide-area traces are derived from tcpdump traces; however these dumps have been sanitized for privacy purposes and thus the raw trace format differs from those derived from tcpdump traces.

Using the work model described in this chapter, we analyzed the simulator output and produced parameterized graphs of the work ratio expressed in Equation 5.14.

5.5.1 Using our model with the simulator

Based on a few assumptions, we have reduced the number of variables in our IP switching model to just two numbers: the switching threshold $N_{Threshold}$ and the counting overhead $\frac{W_{Counting}}{W_{Routing}}$.

From the simulator, we need just two items: the total number of packets and the distribution of packets per flow. With this information it is possible to determine the increase in capacity as a function of the two variables.

This may be somewhat surprising in that we need not run the simulator for every different switching threshold. It is possible to determine P_{Setup} and $P_{Observe}$ exactly using the distribution of packets per flow as discussed in Section 5.3.3. The only other variable factor is $P_{Straggler}$; we discuss why this can be approximated in section 5.3.2.

Module	Time Cost (μsecs)
Input	40.8
Counting	181.4
Forwarding	20.2
Redirection	32.2
Output	63.8

Table 5.3: Cost of the various IP switching modules on a DEC Alpha 3000/400.

Packet Type	Time Cost (μsecs)
Observe	306.2
Setup	680.6
Straggler	306.2
Bypass	0
NoFlow	124.8

Table 5.4: Cost of the various packet classes on a DEC Alpha 3000/400.

5.5.2 Values of Parameters Used

We have measured the costs of the various modules on our IP switching implementation on a DEC Alpha 3000/400 with a 133 MHz alpha processor running NetBSD. The costs of the various modules in our implementation are summarized in Table 5.3.

Our values produce a counting overhead ratio of $\frac{W_{Counting}}{W_{Routing}} = 1.45$. Our IP switching implementation is not efficient in the methods used to retrieve flow records and the extra overhead necessary as our IP switching modules are not tightly integrated into the routing path. As discussed in Section 5.4.5, in an efficient implementation, this ratio most likely can be reduced to a value closer to $\frac{W_{Forwarding}}{W_{Routing}}$, which has the value of 0.16 in our implementation. We use this value to produce our results.

5.5.3 Overall Capacity Gains Using Asymmetric Endpoint Flows

LCS Gateway Trace

Figure 5-8 shows the results of the simulator for the LCS trace, using a 90 or 180 s flow timeout, setup overhead of 4, counting overhead of 0.16, and stragglers per flow of 0.12.

First, we note that the best performance comes from using Server flows, as we hypothesized.

Machine flows and Session flows mirror each other closely, validating our hypothesis that Session flows are not much worse than Machine flows because most clients use only one

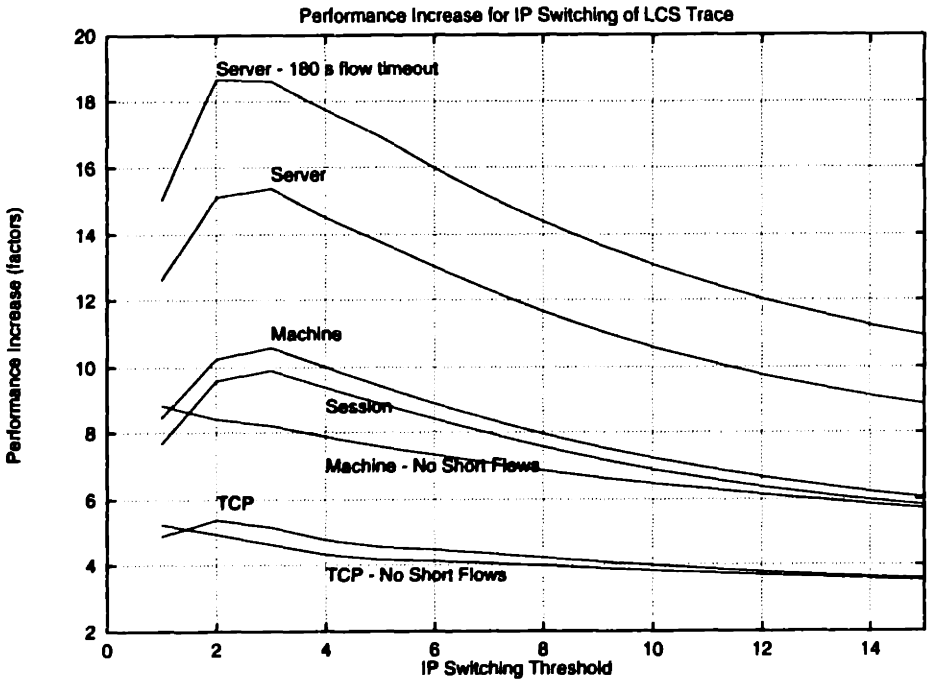

Figure 5-8: Performance increase using various flow types as a function of IP switching threshold for LCS trace. The flow timeout interval is 90 s except where indicated, the flow setup overhead is 4, and the counting overhead is 0.16, and the expected stragglers per flow is 0.12.

service per server. The overall performance loss in using Session flows instead of Machine flows is roughly 6.5% reduction in overall performance, and a 10% increase in the number of average active flows per second.

To Switch Short Flows or Not to Switch Short Flows?

Now we come to a rather surprising result. Though it has been suggested that short flows such as DNS, SMTP, and others should never be switched because they are likely to be more trouble than they are worth [Newman *et al.*, 1996b], our simulations show that using the optimal switching threshold shows that this presumption is false.

In Figure 5-8, the line labeled "Machine" represents the switching of all flows, and the line labeled "Machine - No Short Flows" represents a simulation where the flow types identified in [Newman *et al.*, 1996b] are not switched (specifically, FTP-control, SMTP, DNS, POP-v3, AUTHENT, NTP, and SNMP).

At a switching threshold of 1, not switching short flows is indeed better than switching short flows. But when the switching threshold is increased, switching all Machine flows becomes better and switching only selected flows becomes worse. At a switching threshold of 3, the performance is almost 20% better than the optimal performance of "Machine - No Short

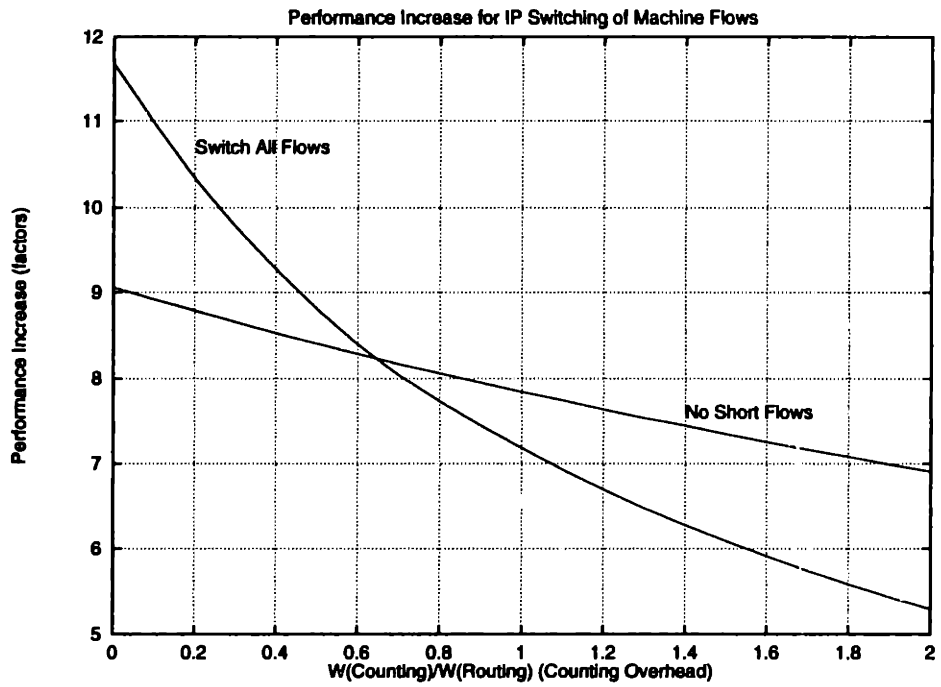

Figure 5-9: Performance increase using Machine flows for not switching flows of pre-identified services and switching all flows, versus the counting overhead $\frac{W_{Counting}}{W_{Routing}}$.

Flows" with a switching threshold of 1.

Similarly, for TCP flows, it is better to switch all flows, though the difference becomes much less.

Examining this in more detail, we find that this surprising result arises out of our assumption of an efficient counting mechanism and a long tail on the packet distribution. We argued that, in an efficient implementation, finding flow entries and updating them should take about as much work as finding a routing table entry, but that the setup overhead required the exchange of messages and setups remain expensive. In fact, if the counting overhead is increased, not switching the short flows is indeed better, as shown in Figure 5-9. The cutover where performance is roughly equal is for $\frac{W_{Counting}}{W_{Routing}} \approx 0.65$, but we believe that in an efficient implementation, as discussed in Section 5.4.5, this ratio should be well below the crossover point.

Earlier IP switching models [Newman *et al.*, 1996b] do not account for this counting overhead – they are based only on flow setup overhead. Their assumptions are that counting the packets is free. Note using a counting overhead ratio of zero yields almost the exact opposite result as we have found. It is only until we reach a high counting overhead that, using our model, we would draw the same conclusions with respect to switching or not switching these "short" flows as the earlier works.

Flow Type	Average number of active flows per second	Performance Increase
Server - 180 s timeout	4536	18.66
Server - 90 s timeout	2773	15.38
Session	4525	9.90
Machine	4110	10.59
Machine - No Short Flows	1569	8.84
TCP	7067	5.38
TCP - No Short Flows	4221	5.25

Table 5.5: Number of active flow per second, on average, and the performance increase for the LCS trace. Flow timeouts are 90 s unless otherwise indicated.

Using the Same Flow Resources

Now, let us examine the flow resources that are used by the each of the various flows. Table 5.5 lists the average number of flow entries per second when tracking the various types of flows. The fewest flow resources are used by Machine flows when not tracking short flows. We have shown that the performance is worse than other schemes, though.

If we look at the TCP flows, we find that switching all flows costs us almost 70% more flow resources for roughly the same performance gain. Thus, it may make sense to not switch some short flows in this case. For Machine flows, the increase in flow resources used is 162%, much higher. However, there is performance boost of almost 20%. As discussed in Section 4.1.3, there are other drawbacks to Machine flows which may justify not using them at all.

Using Server flows with a 90 second timeout is an efficient use of flow resources - there are just 2773 active flows on average. However, if we use the same amount of flow resources as TCP flows without short flows or all Machine flows, we can boost performance from a 15-fold capacity increase to an over 18-fold increase, by increasing the flow timeout interval by 100%. The average number of active flows per second is shown in Figure 5-11. So, for the same amount of flow resources, Server flows perform over 3.5 times better than TCP flows.

Figure 5-10 shows the performance gains at 1 second intervals. On average, the performance increase is 1766%. However, note peaks near 8000%. These are not due to lulls in traffic, but to brief periods where there were about the same number of packets, but far fewer flow setups per second.

Different Switching Thresholds for Different Flows

Figure 5-8 shows the optimal switching threshold for all traffic. However, we may be able to optimize the performance by using different switching thresholds for different classes of

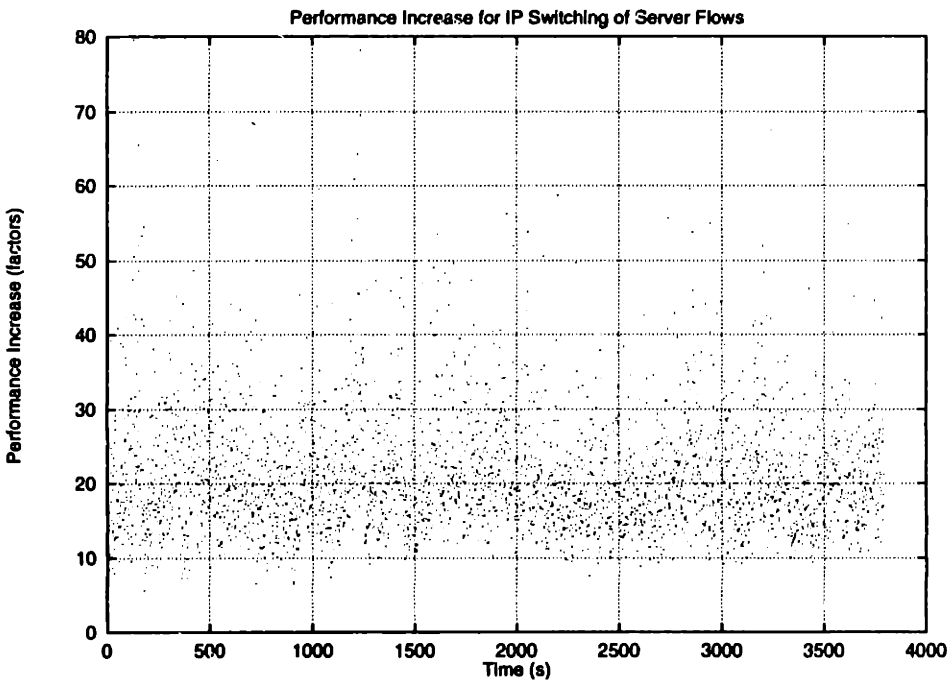

Figure 5-10: Performance increase versus time for Server flows with a switching threshold of 2 packets and a flow timeout of 180 seconds using the LCS trace. The number of active flows per second are within 10% of the number of active flows per second of TCP flows using a 90 second timeout.

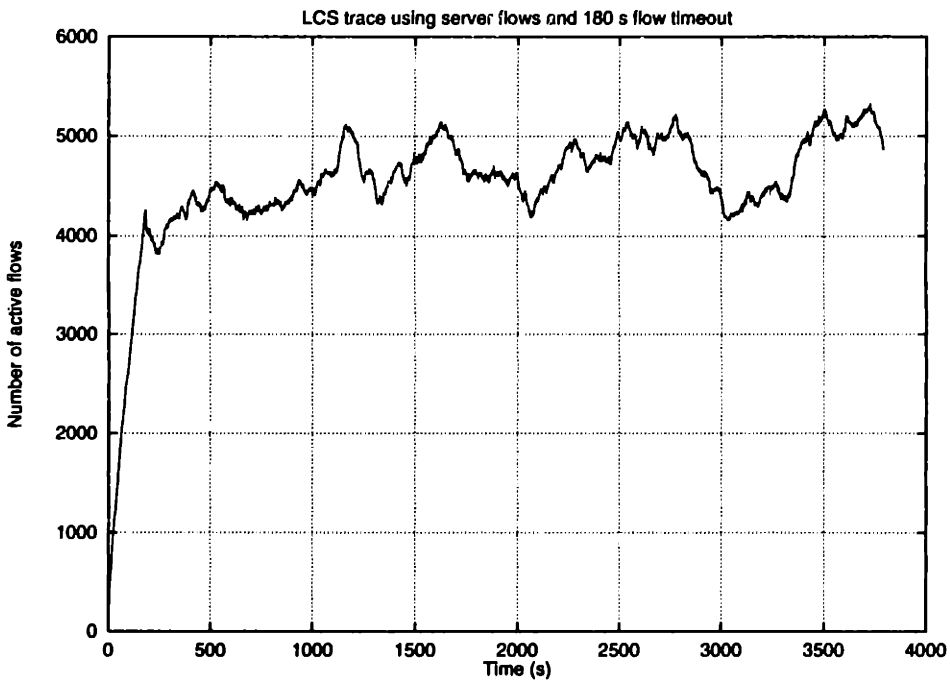

Figure 5-11: Number of active flows per second for LCS trace using Server flows with a 180 second timeout. The average number of flows is within 10% of the average number of flows when using Machine flows and TCP flows without short flows.

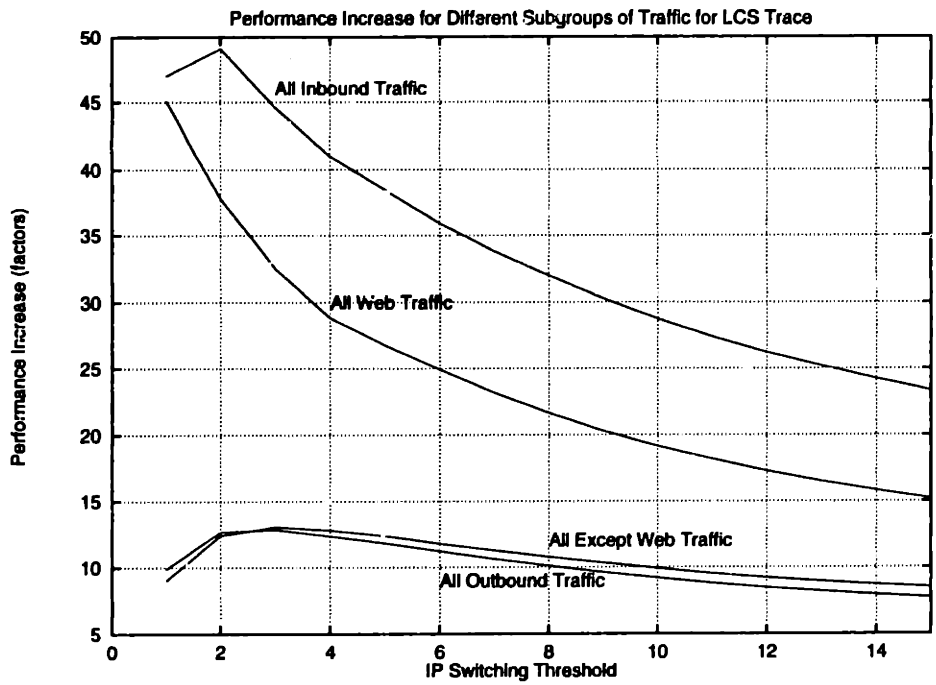

Figure 5-12: Performance increase versus switching threshold for Server flows with a flow timeout of 180 seconds using various parts of the LCS trace. The optimal switching threshold for outbound is 3; for inbound traffic is 2; for web traffic is 1; for all but web traffic is 3. We can increase our capacity by almost 10% by using web/non web switching threshold discrimination.

traffic - whether it be inbound versus outbound traffic or web traffic versus all other traffic. Figure 5-12 shows the performance gains for Server flows based on these characteristics.

We can get a 50-fold increase in capacity if we use a switching threshold of 2 for all inbound traffic. If we use the optimal switching threshold of 3 for outbound traffic, the overall performance increase is 18.9, or just 1% over our previous best. However, if we use the optimal switching threshold of 1 for web traffic and a switching threshold of 3 for all other traffic, we get a 20.2-fold increase, or almost 10% better than previously.[5] A simple change in strategy can produce quite a handsome performance gain.

When we use optimal switching thresholds for flows when we discriminate both service type and directionality, we get a 23.3 fold increase. If we do not switch the subset of these flows that are not worth switching, then we can get a 24.2 fold increase in performance, which represents a 30% increase over using the same threshold for all flows.

[5]The reason using inbound/outbound discrimination looks like it should be better than using web/noweb discrimination but in fact is not is that web traffic represents a larger fraction of the total traffic than inbound traffic does.

Flow Type	Flow Timeout (s)	Max Possible Increase	Increase Using Optimal Threshold	Increase/Max
Server	180	32.8	18.7 (20.2) [24.2]	0.57 (0.62) [.74]
Server	90	26.9	15.4	0.57
Session	90	16.9	9.9	0.58
Machine	90	18.3	10.6	0.58
TCP	90	9.1	5.3	0.58

Table 5.6: Comparison of results to maximum possible performance increase using all-knowing IP switch. The Server flow using 180 s timeout shows results for use of an optimal threshold for all web traffic in parentheses, and the bracketed number represents gains from using optimal thresholds for all types services and direction, and not switching selected flows.

Comparison to the Maximum Possible Gains

Assume that we were to know how in advance how many packets would be in a flow. Then, we would route all the packets in flows not worthwhile to switch and switch flows immediately for longer flows. In our ideal model, we need not keep any flow statistics and there are no stragglers. Since we have assumed a messaging overhead of 4 packets, flows with 4 or fewer packets are not worthwhile to switch and flows of 5 or more packets are worthwhile. Using this simple guide to the maximum performance gain possible, we compute the maximum possible performance increase for the various possible types of flows. The results are shown in Table 5.6.

Wide Area Trace

Figure 5-13 shows the results of the simulator for the WAN trace. The results are not as good for several reasons. Primarily, the trace is too short to really see the gains won from using Server flows. In general, Server flows have a much longer lifetime than five minutes and we only see small initial benefits from such a short trace.

Many of the other observations that we made with the LCS trace still hold true. Server flows are the best, switching all flows still gets better performance. However, one observation that changed was that the number of active Server flows was on par with the number of TCP flows without short flows. This we can again attribute to the very short duration of the trace (18 minutes).

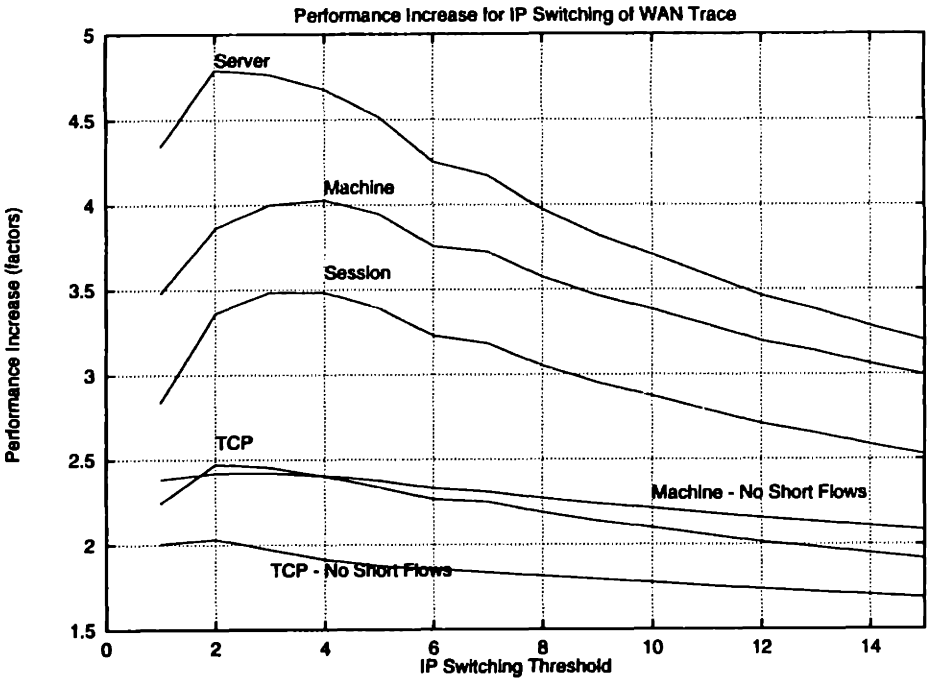

Figure 5-13: Performance increase using various flow types as a function of IP switching threshold for WAN trace. The flow timeout interval is 90 s, the flow setup overhead is 4, and the counting overhead is 0.16, and the stragglers vary by flow type used as shown in Table 5.1.

Optimizing Switching Thresholds for Location

We can use our model to optimize the switching threshold of an IP switch based on location in the network, by gathering a characteristic packets-per-flow distribution for any particular point in the network. These distributions will vary drastically as we near WWW server farms versus when we are near DNS servers. They will differ if we are near modem banks. Also, by selecting specific characteristics of traffic (WWW versus non-WWW, inbound versus outbound) we can further optimizer performance.

5.6 Summary

In this chapter, we have developed a model of IP switching which we use as a basis for analyzing the performance gains for IP switching using different types of flows.

Our model reduces the parameters needed from an IP switched trace down to a manageable number. We show how the probabilities of packets belonging to certain classes can be used to derive the overall cost savings. We also use the interdependence of these probabilities to further reduce the number of variables.

We then show how to use the distribution of packets per flow to derive all the trace-dependent probabilities necessary to fully analyze the trace.

The best method of reducing the amount of work done by the IP switch is to increase the chances of packets bypassing the router. Our asymmetric endpoint flows do this, enabling the bundling of more packets into a single flow. We also show how using different thresholds for different flow types can further boost performance.

Using our work model, we show that using asymmetric endpoint flows on the LCS trace yields a 20-fold improvement over routing, and a 4-fold increase over using symmetric TCP flows.

Chapter 6

Summary, Contributions and Future Work

Today, for the most part, data, voice, and video are carried on networks designed to handle that specific kind of traffic. Now, we are beginning to witness the convergence of services offered over the various infrastructures. Voice and data have long been handled by the telecommunications network, pushing the average call durations higher, which have forced voice network designers to adapt. More recently, video and data are being carried by the cable television companies, although on separate channels. Voice and video, though in primitive form, are being carried by data networks. Even power companies want to begin delivering these types of services to end users.

Each of the networks has been slowly adapting to handling traffic traditionally carried by the other networks. The most rapid shift, though, has been in the data networking side. Its enormous growth and highly competitive environment promotes rapid introduction of new technologies: 14400 baud modems to 57600 baud modems, 10 Mb ethernet to 1 Gb ethernet, shared wires to hubs to switches, and perhaps even routers to flow switches.

Data networks are made up of a melange of different equipment: hosts of various speeds and types and routers of vastly different capacities. The network interconnections also vary, with links of vastly different capacities, and links of all types: copper wire links, optical fiber links, point-to-point microwave links, and other wireless links including satellite links.

Though a motley mixture of equipment, the Internet has continued its ability to deliver data packets from host to host. However, the Internet was originally designed for an environment where client-to-client communication was the dominant style of information exchange. This led to network backbone routers, which were designed to handle each packet individually.

Proposals for handling the growing traffic propose fast path switching for network flows - the trains of packets traveling from a source endpoint to a destination endpoint. However,

the types of flows identified in these earlier efforts are symmetric endpoint flows. Our thesis concentrates on the effect of asymmetric endpoint flows on these new models.

6.1 Summary

In this thesis, we identified a large shift in traffic patterns in data networks over the past few years, driven by the growth in the number of users and the growing ease of information retrieval. We identified a large asymmetry in the new client-to-server data flow.

This asymmetry suggests that using asymmetric endpoint flows would be beneficial. To this end, we proposed two new types of traffic flows: the Session flow and the Server flow. We hypothesized that IP switching based on asymmetric endpoint flows would be better than IP switching based on symmetric endpoint flows, which in turn would be better than plain routing. We also provided evidence that supports this hypothesis.

We developed a work model for IP switching, in which a small number of parameters is used to determine the cost savings. This model is normalized to the work required for routing, so any routing capacity gains from IP switching always come on top of gains from improved methods of plain routing.

To provide a proof-of-concept of asymmetric endpoint flow switching, we integrated asymmetric endpoint flow IP switching into the VuNet, our own high speed ATM network. The VuNet was designed for flexibility in resource management and for delivery of high bandwidth to the application. The features that we designed into the VuNet allowed us overcome problems with creating asymmetric endpoint flows that other ATM systems have difficulty overcoming.

Since we cannot replace backbone and gateway routers to validate our hypothesis, we developed a simulator to analyze wide-area network and gateway network traces. We used the output from the simulator combined with our work model to determine the increase in routing capacity when IP switching is used with asymmetric endpoint flows. Our simulations indicate that using asymmetric endpoint flows at LAN gateways can produce a eighteen-fold or more increase in capacity over a normal router, assuming an efficient implementation. In the wide-area, we can see gains of five-fold or more.

In proposing asymmetric endpoint flows, we retain important qualities of TCP flows: maintaining security, the ability to provide network circuits for specific applications, and the ability to provide quality of service for particular services. Our asymmetric endpoint flows are still based on service type, so that quality-of-service guarantees can be provided on a per-service basis. Furthermore, Server flows aggregate a large amount of traffic so that quality-of-service can be allocated on the aggregated traffic, whose bandwidth requirements are much more predictable than that of individual TCP and Machine flows. In addition, security aspects can be maintained, and advanced operating systems may utilize the allocated

network circuit to bypass any protocol stack demultiplexing.

Not only do asymmetric endpoint flows produce solid performance gains, but they retain the beneficial properties of TCP flows while introducing a new beneficial feature: Server flows allow quality of service to be allocated to aggregated traffic.

6.2 Contributions

This dissertation has shown that using an asymmetric approach to identification and handling of traffic flows within the Internet can produce solid performance gains. In support of this thesis, we have achieved the following:

- **Identified Asymmetric Endpoint Flows:** By incorporating the direction of traffic relative to servers (or relative to a gateway) into our traffic analysis, we were able to provide the first clear look at the great asymmetry of bytes per packet in inbound server traffic versus outbound server traffic.
- **Designed a Flexible ATM Infrastructure (VuNet):** We designed an ATM network infrastructure in which we make a clean separation between network bit transport functions and more complex network management functions. We push functions such as connection management and packet segmentation and reassembly into the hosts (or devices) at the edges of the network. This provides flexibility and simplifies the hardware design. The flexibility allows multiple co-existing approaches to connection management and integration with existing protocols and allows us to respond to changing traffic types and traffic patterns. The VuNet approach also emphasizes network circuits for individual applications.
- **Demonstrated Asymmetric Endpoint IP Switched Flows in the VuNet:** Using the flexibility designed into the VuNet components, we incorporated an IP switching implementation that integrated asymmetric endpoint flows into our system. We took advantage of the unique aspects of the VuNet in order to merge flows from multiple circuits into a single outbound circuit. The software segmentation and reassembly allowed us to use an adaptation layer which uniquely defines the segmentation point within the network so cells can be properly reassembled at the receiver.

  We showed that using a flexible network approach enables a network to provide not only fast-path circuits for asymmetric endpoint flows, but the ability to manage the connections depending on the network region.
- **Integrated Asymmetric Endpoint Flows into IP Switching:** We used the mechanism of IP switching to integrate our asymmetric endpoint flows as a new way to identify packets. Our asymmetric endpoint flows aggregate packets from various hosts, reducing the capture delay of the packets. We implemented IP switching on

the VuNet and integrated our new asymmetric endpoint flow types, which leverage the unique features of the VuNet.

Asymmetric endpoint flows reduce the amount of multiplexing found in network stacks, and allow service guarantees to be attached to a server's aggregated inbound flows. Hosts can deal with these flows separately from other traffic, enabling them to be handled in a fast path unlike current network stacks. Furthermore, the aggregated traffic in Server flows behave better statistically, so that service required can be better predicted and handled in the network.

- **Developed an IP Switching Work Model:** In order to quantify the capacity gains made by introducing IP switching with symmetric and asymmetric endpoint flows, we developed a model for IP switching that reduces the variables to a manageable number.

  The variables capture the overhead necessary to track flows, and the switching threshold, which can be optimized for various types of flows. Optimization of the switching threshold for specific services can include characteristics of the flows such as whether the flow is inbound to a server or outbound from a server.

- **Performed Detailed Analysis of Flow Traces:** Using our model, we analyzed several network traces. When an optimal switching threshold is found for the flow or network regime, we can see how the IP switching overhead affects the capacity of the IP switch. When the flow tracking overhead is high, flows that are considered too much work to switch are better handled by routing the packets in these flows. The service type (SMTP, DNS, etc) is used to infer the average length of a flow. Those services known a priori to have generally short flows are not switched.

  However, when the IP switching flow tracking overhead is low, it is incorrect to assume that it is not worthwhile to switch services with flows that are short on average. The reason is that while the flows of a particular service may not last long on average, some of these flows last long enough such that switching all of them yields a net overall gain in performance.

  Simulations of various flow traces indicate that by providing a broader class of flow types, we can reduce the capture delay to increase the capacity of the router by a factor of twenty or more.

We envision a network of asymmetric endpoint flow IP switches and a heavily server-based wide area traffic access pattern. In this scenario, a single global server incast circuit exists for each heavily used server[1]. In this scenario, all the work required to set up the inbound flows is virtually eliminated. The "branches" of the incast "tree" shift as the active hours of the day shifts from geographic region to region.

[1]Yahoo, Microsoft, Netscape, Excite, and Infoseek were the top five most popular servers in October 1997.[Knowledge, 1997]

A key strength of our approach is that it promotes network efficiency, while retaining the full flexibility of the IP architecture This is an important characteristic given that, over time, new types of services may come to dominate the Internet. We must maintain the ability to identify and adapt these new types of services, which may show drastically different traffic patterns.

6.3 Future Work

6.3.1 Active Flow Switches

Not all types of flows are created equal. Others have recognized this with the identification of types of services that were originally considered a net loss to switch. We have taken the analysis a bit further incorporating directionality, and have shown that even flows of the same class of service differ radically when they are outbound from a server versus inbound to a server.

And of course, the volume and distribution of traffic varies greatly even for a particular type of service and direction.

Other new approaches might be integrated to deal directly with such differences. Active networks [Tennenhouse and Wetherall, 1996] is an approach where code fragments can be distributed among network routers. An active network approach to IP switching could allow the high volume servers to inject customized code fragments that aid in flow identification and threshold tuning into the IP switches.

6.3.2 Analysis of More Wide Area Traces

We have examined the publicly available wide area traces to reach our conclusions. Several questions are raised in our analysis. First, the fraction of DNS traffic in this trace is very high compared to our LCS gateway traces. It is not clear that this is widespread throughout the network or if this particular collection point has a much higher fraction of DNS packets.

Secondly, the curious DNS traffic raises the point that traces from different areas of the backbone should be analyzed to see how traffic pattern differ at various locations in the network. Some points might be closer to large web server farms, and other points might be closer to large DNS servers, for example. If this is the case, it bolsters our assertion that traffic should be handled differently in various network regions, a capability provided by the VuNet approach to switch design.

Other wide area traces are generally not available, however. If others can be convinced to provide slightly more data than they previously have reported, we can apply our model to determine the cost savings for the types of flows on which they report.

6.3.3 Change the Way People Report Information on Network Traffic

Recently, in [Thompson *et al.*, 1997], a detailed analysis of traffic on the MCI backbone was reported. In this work, it was hinted that asymmetries in traffic exist. However, the analysis was performed on the basis of symmetric TCP flows. A histogram of packets per flow was presented. With this data, we can determine how beneficial TCP flows are in the wide area.

We would like to encourage others that report on traffic characteristics to report the number of packets per flow based on the service type and using both symmetric and asymmetric endpoint flow types.

As the world moves from thinking about routing packets to switching flows, this kind of information will be essential in estimating the performance of the new network designs.

6.4 Conclusions

Today, information transfer on the Internet has rapidly moved to a client/server model. This shift has happened rather dramatically. But, descriptions and thought processes regarding traffic flows and patterns within the Internet have not shifted as rapidly. Discussions and proposals regarding traffic flows have been mired in terminology reflecting the old client-to-client model. All prior analyses of Internet traffic use symmetric endpoint flows to describe traffic patterns. Advanced flow-routing methods such as IP switching and flow switching still base their techniques on the identification of symmetric endpoint flows.

We believe the underlying networking equipment should shift its focus from packet-by-packet routing to flows switching. However, the flow switching engines must retain the flexibility to deal with new shifts in traffic patterns. This is where we believe our major contributions lie: in the introduction of new flow types that better suit today's networks, and in the ideas present in our VuNet, which enable the flexible management of connection resources in the workgroup, local and wide areas.

Bibliography

[3Com, 1996] 3Com. Evolutions in high-speed networking, September 1996.

[Adam and Tennenhouse, 1993] J. F. Adam and D. L. Tennenhouse. The Vidboard: A video capture and processing peripheral for a distributed multimedia system. *Multimedia Systems Journal*, 2:150–156, 1993.

[Adam *et al.*, 1993] J. F. Adam, H. H. Houh, and D. L. Tennenhouse. Experience with the VuNet: A Network Architecture for a Distributed Multimedia System. In *Proceedings of the IEEE Conference on Local Computer Networks*, pages 70–76, Minneapolis MN, September 1993.

[Anderson *et al.*, 1993] Thomas E. Anderson, Susan S. Owicki, James B. Saxe, and Charles P. Thacker. High Speed Switch Scheduling for Local Area Networks, April 1993.

[Andrews, 1996] Eric Andrews. MPOA Ties It All Together. *Data Cummunications Magazine*, April 1996.

[Bellcore, 1992] Bellcore. Asynchronous Transfer Mode (ATM) and ATM Adaptation Layer (AAL) Protocols Generic Requirements. Technical Report 1, Bellcore Technical Advisory, August 1992.

[Bertsekas and Gallager, 1987] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[Biagioni *et al.*, 1993] Edoardo Biagioni, Eric Cooper, and Rober Sansom. Designing a Practical ATM LAN. *IEEE Network*, pages 32–39, March 1993.

[Bradner, 1994] Scott Bradner. The Exclusive Bradner Report, September 1994.

[Cheriton, 1989] David R. Cheriton. Sirpent: A High-Performance Internetworking Approach. In *SIGCOMM '89 Symposium on Communications Architecture and Protocols*, pages 158–169, Austin, TX, September 1989. ACM.

[Claffy *et al.*, 1995] Kimberly C. Claffy, Hans-Werner Barud, and George C. Polyzos. A Parameterizable Methodology for Internet Traffic Flow Profiling. *IEEE Journal on Selected Areas in Communications*, 13(8):1481–1494, October 1995.

[Clark and Tennenhouse, 1990] D. D. Clark and D. L. Tennenhouse. Architectural Considerations for a New Generation of Protocols. In *SigComm Symposium*. ACM, September 1990.

[Clark *et al.*, 1989] David D. Clark, Van Jacobson, John Romkey, and Howard Salwen. An Analysis of TCP Processing Overhead. *IEEE Communications*, 27(6):23–29, June 1989.

[Cohen and Finn, 1992] Danny Cohen and Gregory G. Finn. The Use of Message-Based Multicomputer Components to Construct Gigabit Networks. USC/Information Sciences Institute, June 1992.

[Comer, 1991] Douglas E. Comer. *Internetworking with TCP/IP*, volume I. Priciples, Protocols, and Architectures. Prentice-Hall, Englewood Cliffs, NJ, second edition, 1991.

[Davie, 1993] Bruce S. Davie. The Architecture and Implementation of a High-Speed Host Interface. *IEEE Journal on Selected Areas in Communications*, 11(2):228–239, February 1993.

[DEC, 1996] DEC. GIGAswitch/IP solution, fact sheet, November 1996.

[Druschel and Peterson, 1993] Peter Druschel and Larry L. Peterson. Fbufs: A high-bandwidth cross-domain transfer facility. *Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles*, pages 189–202, December 1993.

[Druschel *et al.*, 1994] Peter Druschel, Larry L. Peterson, and Bruce S. Davie. Experience with a high-speed network adaptor: A software perspective. *Proceedings of the SIGCOMM '94 Symposium*, pages 2–13, August 1994.

[Eland, 1996a] Eland. eStats, 1996.

[Eland, 1996b] Eland. eStats Modem Growth, 1996.

[Finn, 1992] Gregory G. Finn. An Integration of Network Communication with Workstation Architecture. *Computer Communication Review*, 21(5):18–29, October 1992.

[Gautam, 1993] Nikhil C. Gautam. Host Interfacing: A Coprocessor Approach. Master's thesis, MIT, 1993. LCS-TR-625.

[Hayter and McAuley, 1992] Mark Hayter and Derek McAuley. A Desk Area Network. *ACM Transactions on Operating Systems*, pages 14–21, October 1992.

[Hayter, 1993] Mark David Hayter. *A Workstation Architecture to Support Multimedia*. PhD thesis, University of Cambridge, 1993. TR319.

[Heinanen, 1993] J. Heinanen. Multiprotocol encapsulation over ATM adaptation layer 5. Technical report, IETF RFC 1483, July 1993.

[Houh *et al.*, 1995] H. H. Houh, J. F. Adam, M. Ismert, C. J. Lindblad, and D. L. Tennenhouse. The VuNet desk area network: Architecture, implementation, and experience. *IEEE Journal on Selected Areas in Communications*, 13(4):710–721, 1995.

[ITU, 1996a] ITU. B-ISDN ATM Adaptation Layer spefification: Type 3/4 AAL. Technical report, International Telecommunication Union, August 1996.

[ITU, 1996b] ITU. B-ISDN ATM Adaptation Layer spefification: Type 5 AAL. Technical report, International Telecommunication Union, August 1996.

[Kaeo, 1996] Merike Kaeo. Router Architecture and Performance Analysis. Technical report, Cisco Systems, May 1996.

[Kanakia, 1988] H. Kanakia. The VMP Network Adaptor Board: High-Performance Network Communication for Multiprocessors. In *Proceedings of the SIGCOMM '88 Symposium*, pages 175–187. ACM, 1988.

[Kehoe and Pitkow, 1996] Colleen M. Kehoe and James E. Pitkow. Surveying the Territory: GVU's Five WWW User Surveys. *The World Wide Web Journal*, 1(3), Summer 1996.

[Kermani and Kleinrock, 1979] Parviz Kermani and Leonard Kleinrock. Virtual Cut-Through: A New Computer Communication Switching Technique. *Computer Networks*, 3:267–286, 1979.

[Keynote, 1997] Keynote. How Fast is the Internet?, October 1997.

[Knowledge, 1997] Relevant Knowledge. The Web Report, October 1997.

[Lampson, 1992] Butler Lampson. Seminar: AN2: A Self-Configuring Local ATM Network, March 1992.

[Lane, 1992] Jim Lane. *Asynchronous Transfer Mode: Bandwidth for the Future.* Telco Systems, Norwood, MA, first edition, 1992.

[Lin and McKeown, 1997] Steven Lin and Nick McKeown. A Simulation Study of IP Switching. *Computer Communication Review*, 27(4):15–24, October 1997.

[Lindblad, 1994] Christopher J. Lindblad. *A Programming System for the Dynamic Manipulation of Temporally Sensitive Data.* PhD thesis, MIT, August 1994. LCS-TR-637.

[McCarthy and Bernoff, 1996] Jphn C. McCarthy and Josh Bernoff. The Forrester Report: People and Technology Strategies. Technical Report 6, Forrester, October 1996.

[Montz *et al.*, 1994] A. B. Montz, D. Mosberger, S. W. O'Malley, L. L. Peterson, T. A. Proebsting, and J. H. Hartman. Scout: A communications-oriented operating system. Technical Report 94-20, Department of Computer Science, University of Arizona, June 1994.

[Networks, 1997] WH Networks. Gigabit packet switching routers (best effort), July 1997.

[Newman and Lyon, 1997] Peter Newman and Tom Lyon. IP Switching and Gigabit Routers. In *IEEE Communications Magazine*, January 1997.

[Newman *et al.*, 1996a] Peter Newman, W. L. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G. Minshall. Ipsilon Flow Management Protocol Specification for IPv4. Technical report, IETF RFC 1953, May 1996.

[Newman *et al.*, 1996b] Peter Newman, Tom Lyon, and Greg Minshall. Flow Labelled IP: A Connectionless Approach to ATM. In *Proceedings of IEEE Infocomm*, volume 3, pages 1251–60, San Francisco, CA, March 1996. IEEE.

[Newman, 1992] Peter Newman. ATM Technology for Corporate Networks. *IEEE Communications Magazine*, 30(4):90–100, April 1992.

[Owicki and Karlin, 1992] Susan S. Owicki and Anna R. Karlin. Factors in the Performance of the AN1 Computer Network. Technical Report DEC/SRC TR-88, Digital Equipment Corp. Systems Research Center, June 1992.

[Pelline, 1997] Jeff Pelline. Cable modem users growing. *CNet Online*, October 1997.

[Prycker, 1991] Martin De Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN.* Ellis Horwood, Chichester, England, 1991.

[Rekhter *et al.*, 1995] Yakov Rekhter, Bruce Davie, Dave Katz, Eric Rosen, and George Swallow. Cisco Systems' Tag Switching Archtecture Overview. Technical report, IETF RFC 2105, February 1995.

[Research, 1995] Nielsen Media Research. CommerceNet/Nielsen Internet Demographics Survey. Technical report, CommerceNet/Nielsen Media Research, Fall 1995.

[Research, 1997] Nielsen Media Research. CommerceNet/Nielsen Internet Demographics Survey. Technical report, CommerceNet/Nielsen Media Research, Spring 1997.

[Rivest, 1992] R. Rivest. The MD5 Message-Digest Algorithm. Request For Comments: 1321, April 1992.

[Rodeheffer, 1993] Thomas L. Rodeheffer. Experience with Autonet. *Computer Networks and ISDN Systems*, pages 623–629, 1993.

[Schroeder *et al.*, 1990] M.D. Schroeder, A.D. Birrell, M. Burrows, H. Murray, R.M. Needham, T.L Rodeheffer, E.H. Satterthwaite, and C.P. Thacker. Autonet: A High-speed, Self-configuring, Local Area Network using Point-to-point Links. Technical Report DEC/SRC TR-59, Digital Equipment Corp. Systems Research Center, April 1990.

[Smith and Traw, 1993] Jonathan M. Smith and C. Brendan S. Traw. Operating Systems Support for End-to-End Gbps Networking. *IEEE Network*, July 1993.

[Stasior, 1997] William Stasior. *An Interactive Approach to the Identification and Extraction of Visual Events.* PhD thesis, MIT, July 1997. LCS-TR number to be issued.

[Tanenbaum, 1989] A.S. Tanenbaum. *Computer Networks.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Tennenhouse and Wetherall, 1996] David Tennenhouse and David Wetherall. Towards an Active Network Architecture. *Proceedings of Multimedia Computing and Networking 1996*, January 1996. Appears later in Computer Communications Review, Vol. 26, No. 2, April 1996.

[Tennenhouse *et al.*, 1995] David Tennenhouse, Joel Adam, David Carver, Henry Houh, Michael Ismert, Christopher Lindblad William Stasior, David Wetherall, David Bacher, and Teresa Chang. The ViewStation: A Software-Intensive Approach to Media Processing and Distribution. *Multimedia Systems Journal*, 3:104–115, 1995.

[Tennenhouse *et al.*, 1996] David Tennenhouse, Thierry Turletti, and Vanu Bose. The SpectrumWare Testbed for ATM-based Software Radios. *ICUCP'96 Proceedings*, September 1996.

[Tennenhouse, 1993] David Tennenhouse. A Software-Oriented Approach to the Design of Media Processing Environments. Technical Report The VuStation Collected Papers: TR-590, MIT LCS, 1993.

[Thompson *et al.*, 1997] Kevin Thompson, Gregory J. Miller, and Rick Wilder. Wide-Area Internet Traffic Patterns and Characteristics. *IEEE Network*, 11(6):10–23, November/December 1997.

[Traw, 1992] C. Brendan S. Traw. A High-Performance Host Interface for ATM Networks. Master's thesis, University of Pennsylvania, 1992.

[Turner, 1996] Jon Turner. Dynamic Sharing of Many-to-Many Virtual Circuits. In *Proceedings of Washington University Workshop on Integration of IP and ATM*, St. Louis, MO, November 1996.

[von Eiken *et al.*, 1995] Thorsten von Eiken, Anindya Basu, Vineet Buch, and Werner Vogels. U-Net: A User-Level Network Interface for Parallel and Distributed Computing. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP)*. ACM, December 1995.

[Week, 1997] Interactive Week. Article. *Interactive Week*, 4(27), August 1997.

[Wetherall, 1994] David Wetherall. An Interactive Programming System for Media Computation. Master's thesis, MIT, 1994. LCS-TR-640.

[Zakon, 1997] Robert Hobbes Zakon. Hobbes' Internet Timeline v3.1, July 1997.

[Zegura, 1993] Ellen Witte Zegura. Architectures for ATM Switching Systems. *IEEE Communications Magazine*, 31(2):28–37, February 1993.