RICE UNIVERSITY

The Axon Ethernet Device

by

Michael Foss

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

[illegible]

Scott Rixner, Chair
Associate Professor of Computer Science
and Electrical & Computer Engineering

Alan L. Cox

Alan L. Cox
Associate Professor of Computer Science
and Electrical & Computer Engineering

[illegible]

T. S. Eugene Ng
Assistant Professor of Computer Science
and Electrical & Computer Engineering

HOUSTON, TEXAS

MAY, 2010

ABSTRACT

The Axon Ethernet Device

by

Michael Foss

Data centers are growing in importance since computation is moving from personal computers to the Internet. Data centers often use Ethernet as the network fabric; however Ethernet presents fundamental limitations to scalability.

This work examines the design, implementation, and characterization of the Axon, a network device that overcomes Ethernet's scalability limitations while maintaining the simplicity of such devices. Axons use cut-through routing to reduce the latency of communication and source-routing both to eliminate the the spanning tree and to reduce state within the network.

Using just one redundant link in small network has been shown to give a 96% increase to UDP bandwidth and a 63% increase to TCP bandwidth. Experiments confirm that an Axon's latency is an order of magnitude faster than that of a store-and-forward switch in an uncongested network, thereby increasing the potential diameter and improving the scalability of an Ethernet network.

Acknowledgments

First I would like to thank my advisor, Scott Rixner for all of his patience and guidance throughout my studies at Rice. I would not have been able to complete this work without his support and insights.

I would also like to thank the other members of my thesis committee, Alan Cox and Eugene Ng, for their helpful suggestions and support for this work.

I am grateful to my colleague Jeff Shafer for all of his help throughout the process of developing the Axon and especially for his feedback and help throughout my graduate studies at Rice. Thanks to Brent Stephens for his work on the CAM in the Axon and for his insights into the Axon design and development.

If it weren't for my family, I would never have had the opportunity to be at Rice. I thank them for all their support throughout my tenure at Rice.

Finally I would especially like to thank my fiancee Susan White for her help and support throughout graduate school.

Contents

Illustrations

Tables

Chapter 1

Introduction

Data centers are becoming increasingly more important to the utility of Internet. Users expect that a query searching the entire Internet responds on the order of one second. Users expect to be able to browse shops and purchase products online without any noticeable delay. As netbooks become more common, users are performing more compute-intensive tasks on the Internet that they had previously done locally (e.g. word processing, photo editing, and gaming among others). Data centers are typically the place where these tasks are physically computed.

The network architecture is a critical component of the data center. In order to reduce costs, data centers are typically composed of many commodity computers connected to each other. The communication between these computers is what gives the data center its power to handle high loads of computation. In order to reduce the amount of time to process a request from a user outside the data center, a request may become highly parallelized, as in the case of map/reduce. Hosts inside the data center need high bandwidth to other hosts inside the data center in order to fulfill these requests most efficiently.

Indeed there are specialized network fabrics for data center networking; however, these come at a high cost. One major goal of the data center is to maximize the computing power to cost ratio. Since Ethernet is so widespread, it has become cheap relative to other network fabrics and is thus the most commonly used data center network fabric.

It is well known that switched Ethernet does not scale well to large numbers of hosts [1]. In fact, the very mechanisms that make switched Ethernet easy to manage also hinder its scalability. As the network size increases, dynamic address and location discovery using broadcast packets and packet flooding become prohibitively expensive for both switches and hosts connected to the network.

As a consequence of switched Ethernet's limitations, the current practice is to break the network into subnets and use IP routing between the subnets. Each subnet can then be its own independent Ethernet network. In effect, IP routers, which originally existed at the edge of the data center network, now form the core. However, this creates additional management overhead. Furthermore, the route computation and storage needs of a network device in this architecture scale with the amount and type of traffic that traverses the device.

1.1 The Axon

This thesis introduces the *Axon*, an Ethernet-compatible device for creating large-scale, local-area networks. Specifically, an Axon is an inexpensive, practical device that replaces an Ethernet switch. In fact, to a directly connected host, an Axon appears to be an Ethernet switch because the Axon and host communicate using the standard Ethernet datalink and physical layer protocols. Moreover, Axons provide the same ease of management as Ethernet switches. A host can be connected to an Axon without manual network configuration. However, among themselves, Axons use *source-routed Ethernet*, a new datalink layer protocol.

Source-routed Ethernet has two advantages over switched Ethernet. First, it stores all network and routing state needed by a host in the *local* Axon—the Axon to which the host is directly connected. Therefore, regardless of the network's scale, the route computation and storage needs of a single Axon are proportional to the demands of its locally connected hosts. Axons in the core of the network can thus handle a much larger number of traffic flows, as no routing computation or storage resources are necessary for flows that traverse the Axon. Second, source-routing allows for arbitrary network topologies. Switched Ethernet requires that the network topology have no cycles, thus limiting the bandwidth between hosts. Because no cycles are allowed, switched Ethernet also creates the conditions for oversubscription of links at the root of the tree. By definition, source-routing allows a packet to take any available path to reach its destination.

Others have proposed architectural modifications to Ethernet switches to enable large-scale Ethernet networks [2, 3, 1, 4, 5, 6, 7, 8, 9]. These prior techniques move the responsibility for routing among hosts from the IP routers to the Ethernet switches. This requires the switches to maintain routing tables and other network state for all traffic flows that traverse the switch. This effectively replaces lightweight Ethernet

	Standard Ethernet	**Axon Network**
Route Storage	Since this network uses destination-based routing, switches must have state for each flow that uses the switch.	Since a source route needs only to be determined initially, Axons only need state for flows originating from hosts directly connected
Topology	The only active topology is a tree.	Axons may employ arbitrary topologies
Redundant Links	Redundant links are used only for protection from network failure.	All redundant links in the network may be used.

Table 1.1 : Key benefits of an Axon network over a standard Ethernet network.

switches with heavyweight "Ethernet routers". While these techniques do reduce the management overhead of IP routing and subnetting, they do not consider the practicality, complexity, and cost of the required network devices.

An Axon is a simple, practical device because it uses source-routing to forward network traffic. As a consequence, Axons only need to store routing state for locally connected hosts. Source-routing has the additional benefit beyond other scalable Ethernet approaches in that it allows much more control over bandwidth provisioning. Moreover, by using source-routing, the initial Axon along a path is the only Axon that needs to consult a large hardware table to determinescalability a route through the network. In contrast, every Ethernet switch and IP router must always perform a route lookup in a large hardware table for every packet that traverses the switch or

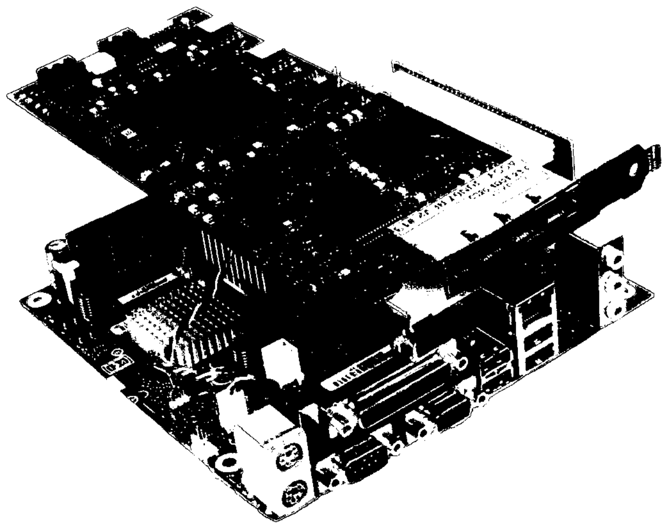


Figure 1.1 : Prototype of the Axon Ethernet Device

router. Furthermore, the switching fabric inside the Axon uses cut-through routing to minimize latency. These key features of the Axon device—local resource management, source-routing, and cut-through routing—combine to create a powerful building block for future Ethernet networks.

A prototype Axon device, shown in Figure 1.1 demonstrates the practicality of source-routed Ethernet. Table 1.1 shows some of the key benefits given by the Axon architecture. The prototype Axon achieves latencies of less than 1us per hop, compared with 7–28us per hop in switched Ethernet. Furthermore, the prototype Axon is able to saturate 1Gbps Ethernet links and fairly distribute bandwidth among competing traffic flows in the face of congestion. These characteristics result in demonstrable improvements in performance for network-intensive applications. For example, Axon devices improve the performance of PostMark, a file server benchmark, by 20–77%

for clients connecting to an NFS file server over the network.

1.2 Contributions

This thesis makes the following contributions.

- The first contribution of this thesis is the design of the Axon network device. The design proposes using transparent source routing to set up source routes in the Axon network. By using transparent source routing, redundant links in the network may be utilized in the network topology, and no host needs to be modified. Furthermore, routes are stored on the local Axon, which is an improvement over standard Ethernet where each Ethernet switch must maintain the state of all flows using the switch.

- The second contribution of this thesis is the implementation of the Axon. This implementation verifies that the design is functional and hosts can indeed take advantage of source routing without modification. We verify that the Axon can transparently replace an Ethernet switch.

- The third contribution is the characterization of the Axon based on the prototype implementation. We show that a user application on a host can benefit from the reduced latency of cut-through routing. Furthermore, we show that an Axon network can take advantage of redundant links in the network and give a bandwidth increase of 96% for UDP traffic and 63% for TCP traffic. These measurements show that the Axon is indeed well-suited for being the substrate of a data center network.

1.3 Organization

This thesis is organized in the following way. Chapter 2 gives the relevant background information on the operation of Ethernet and IP, with particular emphasis on the scalability limitations of Ethernet in the data center. Some of the related work on scaling Ethernet and the limitations of these approaches are discussed. Chapter 3 describes the Axon's design and architecture. The following chapter compares the performance of Axon networks to that of standard Ethernet networks. Finally, chapter 5 concludes this thesis.

Chapter 2

Background

This chapter will present an overview of the operation of Ethernet and how it is used in the data center. It also points out some of the problems with scaling Ethernet. Section 2.1 explains how Ethernet works and gives some of its problems with scaling. Section 2.2 explains how the DHCP and ARP protocols use broadcast packets to communicate with other IP hosts on the same subnet. Section 2.3 explains some of the drawbacks of using the IP layer to solve the scalability problems of Ethernet. Section 2.4 explains how Ethernet is typically used in the data center. Finally, section 2.5 presents some of the work others have done to combat Ethernet's scalability problems.

2.1 Ethernet

Switched Ethernet is a truly ubiquitous technology. Ethernet interfaces are standard equipment in a wide range of computer systems, from embedded devices to mainframes. Moreover, switched Ethernet is deployed in a variety of environments, including home networks, office networks, data center networks, and campus networks.

A key reason for switched Ethernet's wide-spread deployment is its ease of operation. First, Ethernet equipment will operate with little or no manual configuration.

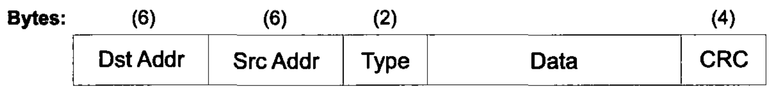


Figure 2.1 : Ethernet packet format.

Interface addresses, known as Media Access Control (MAC) addresses, are simple globally unique identifiers that are assigned by hardware manufacturers, and packet forwarding is set up automatically. Second, switched Ethernet is self healing. It can automatically take advantage of redundant network connectivity to recover from network failures.

Switched Ethernet's ease of operation derives in large part from its ability to flood packets throughout the network. Specifically, flooding enables a packet to reach the destination host's interface without any configuration of that interface or the network, regardless of the interface's location in the network topology. However, since Ethernet packets do not have a time-to-live (TTL) field (see Figure 2.1), the network topology must not have any cycles. Otherwise, flooded packets will circulate endlessly inside network cycles. Even worse, flooded packets will be duplicated at the intersection of two network cycles.

Prohibiting cycles in the network topology does, in principle, eliminate the possibility of redundant links in the network. However, switched Ethernet permits redundant links because their use is severely restricted. Specifically, they are only used to heal the network after a failure. They are never used to provide additional bandwidth. Ethernet switches employ a distributed algorithm, such as the Rapid Spanning Tree Protocol (RSTP), to reach agreement among the switches on a cycle-free *active topology* that is a spanning tree for the network topology. Furthermore, the Ethernet switches only use the active topology when forwarding packets to their destination. In effect, redundant network links are disabled. However, in the event of a network failure, the switches may selectively reactivate one or more disabled links and reconfigure the active topology to use those these links, thereby healing the network. Under ideal conditions, RSTP achieves a reconfiguration time that is on the order of the maximum communication delay across the network. During this time, packets may be lost, duplicated, or otherwise routed incorrectly.

Because Ethernet does not allow an active topology with cycles, the cross-section bandwidth is effectively limited to that of one link. Furthermore, even though two hosts may be connected by a physically shorter path through the network, packets are restricted to travel along the active topology.

The Ethernet packet format, as shown in Figure 2.1, is very simple. The header contains the destination address of the target host, the source address of the originating host, and a protocol type field, which describes the encapsulated data. The remainder of the packet is the encapsulated data, such as a TCP/IP packet, and a CRC for verifying the integrity of the packet.

To reduce packet flooding, Ethernet switches automatically perform *address learning*, which is the process of mapping the locations of interface addresses within the active topology. Specifically, whenever a packet is received by a switch port p, a mapping is created from the packet's source address a to the port p. Typically this mapping is stored in a content-addressable memory (CAM). Later, if a packet with a destination address a is received on a port other than p, instead of flooding the packet on all ports it is forwarded only to port p. Eventually, if the mapping of address a to port p is not reused, it will be discarded. Thus the number of hosts on an Ethernet segment is effectively limited to the size of the Ethernet switch's CAM; segments with more hosts may see excessive flooding due to packets overflowing the switch's CAM.

2.1.1 Link Aggregation Control Protocol

Link Aggregation Control Protocol (LACP) [10] is often used in Ethernet networks to increase the bandwidth between two devices by connecting them together with multiple Ethernet links. However, LACP's original intent was to provide additional network redundancy rather than additional bandwidth. One MAC address is chosen as the logical source MAC of all aggregated links. A *conversation* is some set of frames to be transmitted from one device to another where it is important that ordering between the frames remains consistent, as in a single TCP flow. The definition of a conversation is purposely vague to allow for different implementations.

The *distribution function* is in charge of assigning which output port a packet takes when it leaves on one of the aggregated links. Typically, network devices implement the distribution function as a hash of several fields within the network packet. The main requirement is that within a conversation no packets are duplicated or reordered. In this way, the link aggregation is transparent to the higher network protocols. A single conversation, however, is still limited by the maximum throughput of a single link. While LACP can help increase the bandwidth over a busy link in the Ethernet network, it is generally limited in that it must use a hash for the distribution function, which is not guaranteed to disseminate the traffic optimally.

2.2 Broadcast Packets in the Data Center

For hosts communicating via IP, the main two types of packets that are broadcast over an Ethernet segment are DHCP and ARP packets. These comprise the Ethernet broadcast traffic most relevant to the data center.

2.2.1 Dynamic Host Configuration Protocol

The Dynamic Host Configuration Protocol (DHCP) allows hosts to join a network without any manual configuration. For example, a host can use DHCP to learn its IP address and its default IP router.

In the general case, a host will broadcast a DHCP discovery message over the Ethernet network. This message will therefore be received by all of the DHCP servers in the network. Any number of these servers may respond with a DHCP offer message, which is unicast to the host. The host chooses one of these DHCP offers and broadcasts a DHCP request message indicating its choice, thereby simultaneously informing all of the servers. Finally, only the chosen server responds with a DHCP acknowledge message, which is unicast to the host.

The host's "lease" on the configuration provided by the DHCP server also has an expiration time. Before the lease expires, the host will attempt to renew its configuration lease with the server from which it came. In this case, the host will skip the first phase of the protocol, and begin with a unicast DHCP request to the server. If the host does not receive a DHCP acknowledgement renewing the lease before its expiration, the host falls back to the general case and broadcasts a new DHCP discovery message.

2.2.2 Address Resolution Protocol (ARP)

When an IP host sends an IP message to another IP host residing on the same Ethernet network, it sends the message directly without the involvement of a router. However, before the message can be sent, the sending host must discover the receiving host's MAC address. To discover the receiving host's MAC address, the sending host uses the Address Resolution Protocol (ARP). In this case, the sending host broadcasts a message containing an ARP request on the Ethernet network. The ARP request specifies the IP address of the host whose MAC address is sought. Since the ARP request is broadcast, every host on the Ethernet network will receive it; hosts residing on large Ethernet segments will see much irrelevant ARP traffic. When a host receives an ARP request specifying its IP address, it responds with an ARP reply specifying its MAC address. The ARP reply is unicast to the requester. When the ARP reply is received by the requester, the translation from IP address to MAC address is cached, and the IP message is unicast using the discovered MAC address.

In addition, all hosts in the network take advantage of the fact that ARP requests are broadcast to avoid future ARP requests of their own. Specifically, the receiver of an ARP request learns the mapping from the requester's IP address to its MAC address, and caches this information. In fact, a host may spontaneously send an ARP request specifying its own IP address as a way of announcing its MAC address. In this case, no host is expected to respond.

2.3 IP

As mentioned in the previous section, Ethernet has several inherent limitations to scalability. Because of this, often the internet protocol (IP) layer is used to scale networks. Each host in an IP network has a unique IP address (which may be assigned by a DHCP server). Unlike Ethernet addresses, IP addresses are related to the host's location in a network. Therefore, network architects must carefully plan the network IP topology (in data centers, campus networks, etc.) in order to allow for future growth and the most efficient communication between active hosts.

IP routers are responsible for moving incoming IP packets closer to their ultimate destination in the IP network. Assuming that the IP network is layered on top of multiple Ethernet networks, each port of the IP router connects to a different

Ethernet segment and performs the routing required to transfer packets from one Ethernet segment to another. When a packet arrives at a router, its IP address is inspected against a *routing table.* This table maps a given range of IP addresses to a specific port on the router. The range of IP addresses is specified by the most significant bits in the IP address, or, the prefix. The router examines the destination IP address of an incoming packet and determines the output port based on the longest matching prefix of its routing table. Typically this lookup is performed by a ternary CAM (TCAM), which takes up a larger area and more power than the CAM that is found on an Ethernet switch.

The operation of the routing table is what ties an IP address to a host's location in the network. Routing tables in an IP network must be globally coordinated in order for packets may reach all available destinations on the network. Thus, IP networks require further management in a way that Ethernet networks do not. Furthermore, routing tables must be modified each time a change is made to the network topology. Several protocols exist in order to automatically update the routing tables in a network, but enforcing network policies (such as a firewall) may increase the complexity required to maintain consistent routing tables across the IP network or even require setting the routing tables manually.

2.4 Data Centers and Ethernet

A data center is an aggregation of thousands or even hundreds of thousands of servers. These are particularly beneficial for cloud computing, where one uses an Internet service to do work as opposed to one's own host machine. The applications of interest here are office-type documents, Internet search, or cloud computing services, such as Amazon's EC2. These services require extensive network communication within the data center.

It is desirable to use Ethernet in the data center because it is ubiquitous and is thus cheaper than other technologies. Also, using Ethernet allows servers from different generations to be used side by side. Finally, as previously discussed, Ethernet is a simple protocol and needs little or no management.

Data centers would benefit in many different ways if it were possible to have one large, flat Ethernet network connecting all hosts. First, the network architects would not need to segment the IP address space. This would make it easier for data centers to construct and to grow . Second, the data center's network would be cheaper to build, as it would not require IP routers to handle internal traffic. IP routers are generally more expensive than Ethernet switches due to their additional capabilities and more expensive route lookup function. Third, any virtual machine on the network could migrate to any physical host while maintaining the same IP address. This capability would allow data centers to run more efficiently as it enables arbitrary virtual machine migration. Fourth, the management overheads of managing the IP network would be reduced or even diminished.

2.5 Related Work

The benefits of Ethernet networking are widely recognized, which is why it is so popular today. However, the scalability limitations of Ethernet are also well known. These previous observations have all led to novel approaches to scaling Ethernet. Many of them, however, still involve network-wide dissemination of topology and/or routing information [4, 5, 8, 9]. More recent approaches have further reduced the overhead of scaling Ethernet [2, 3, 6]. However, in these schemes, each device still must provide storage and computation resources for all flows that traverse the device. The computation, but not storage, requirements can be reduced by offloading the route calculation to either a distributed control plane [1] or a wafer-thin control plane [7].

Identity-based routing can also be used to improve the scalability of local area networks [11, 12, 13]. These prior techniques, however, do not route directly to the destination. Instead, they determine paths using a hash of the destination identifier. These paths limit the route that a packet may take to reach a destination in the network.

OpenFlow switches can be programmed to identify flows, process packets, and forward packets in a flexible manner [14]. While OpenFlow switches allow modifications to the Ethernet protocol, they are still closely tied to the existing switched

Ethernet architecture. This means that flow data must be disseminated to all OpenFlow switches in order to forward packets correctly, thus limiting the scalability of an OpenFlow network.

Others have recognized that there are many redundant links that Ethernet switches never use. Many have proposed new ways to manipulate the spanning tree [4, 9]. However, [9] requires software to be run on each host in the network. [4] does not allow for arbitray paths to be taken via the redundant links in the network.

Section 3.5 will revisit some of the above proposed solutions to scaling Ethernet and will show how the Axon overcomes many of the problems that these techniques introduce.

Chapter 3

Axon Network Device

3.1 Axon Design

The Axon device is a direct replacement for Ethernet switches that can be used to create a highly-efficient local-area network. Hosts* can be directly connected to an Axon without modification. Hosts continue to transfer packets as if connected to a traditional Ethernet switch. Axons transparently use source-routed Ethernet to transfer data through a network of Axons. In order to implement source-routed Ethernet, Axons utilize a different packet format and different routing mechanisms than conventional switched Ethernet.

3.1.1 Axon Packet Format

Axon packets are transmitted over conventional Ethernet cables using standard Ethernet media access control and physical transceivers. However, within a network of Axons, packets have their own header type, as shown in Figure 3.1. The Axon header includes a type, length, and source-routing information. Although Axon packets do

*In this thesis, the term *host* means any non-Axon device that communicates via the Ethernet protocol, including a standard Ethernet switch, an IP router, or a host computer. *Switch* means a device that strictly adheres to the layer 2 Ethernet protocol

Bytes:	(0.5)	(2)	(1.5)	(1.5)	*(0.5 per hop)*	*(0.5 per hop)*	(0 - 0.5)		(4)
	Type	Length	Fwd Hop Count	Rev Hop Count	Fwd Hops	Rev Hops	Padding	Data	CRC

Figure 3.1 : Axon Packet Format

not have an Ethernet header at the beginning of the packet, standard Ethernet physical transceivers are still able to send and receive them.

The type field in the header indicates the type of packet that is encapsulated in the Axon packet. For normal traffic between hosts, Ethernet packets are encapsulated in Axon packets. For traffic between Axons, non-Ethernet control messages are encapsulated in Axon packets. The type field can also indicate other special packet types. The length field contains the length of the Axon header.

The forward and reverse hop counts indicate the number of remaining forward hops and the number of hops the packet has already taken. The forward path indicates the port numbers that should be followed for subsequent hops and the reverse path indicates the port numbers that should be followed to return to the packet's source. Each hop in the path is a 4-bit value that indicates an output port number. A hop with the value '0xf' indicates the packet should be forwarded to the control plane, rather than an Ethernet output port. If there are an even number of hops, the header will be padded with 4 bits so that it ends on a byte boundary.

In order for commodity Ethernet MACs and PHYs to transmit and receive Axon frames, they must support frames larger than standard Ethernet frames and they must not reject frames based on MAC address. Fortunately, modern Ethernet MACs support both of these features. First, Ethernet MACs support *jumbo frames*, which, as the name implies, are oversized Ethernet frames. With this feature enabled on the Axon MAC units, a maximum-sized Ethernet frame can safely be encapsulated within an Axon packet. Second, Ethernet MACs support *promiscuous mode*, which disables any checking of the destination MAC address. With this feature enabled, the receiving MAC will accept frames whose first 6 bytes do not match the receiver's MAC address, allowing the first 6 bytes to be part of the Axon header.

Finally, from the perspective of the Ethernet MACs and PHYs, an Axon packet is just an oversized Ethernet frame, the Ethernet CRC can be used to detect transmission errors. As with conventional Ethernet frames, the transmitting MAC will compute and append a CRC to every Axon packet and the receiving MAC will compute and verify the CRC of every Axon packet.

3.1.2 Axon Packet Routing

The source-route in the Axon header allows an Axon to determine the output port for an Axon packet after receiving the sixth byte of any Axon packet, as the sixth byte of the header contains the next forward hop. At this point, the packet can immediately be forwarded to the appropriate output port. The paths in the Axon header are updated for the next hop as the packet is forwarded to the output port.

Axons perform cut-through routing, but its implementation differs from conventional high performance interconnection networks. An Ethernet network has two key differences from a conventional interconnection network. First, Ethernet MACs and PHYs only allow entire Ethernet packets to be transmitted over a wire. No backpressure can be applied for flow control in the middle of a packet transmission. This means that buffering for entire packets must be provided in order to deal with collisions and congestion. Second, unlike most interconnection networks, Ethernet networks do not guarantee packet delivery. This means that once the buffers fill up, the Axon can safely drop packets. If necessary, higher level network protocols will then throttle their transmission rate and properly resend these packets.

3.1.3 Interface with Conventional Ethernet Devices

Axons present themselves as a conventional Ethernet switch to conventional Ethernet devices. Hosts that are connected to an Axon send and receive normal Ethernet frames, not Axon packets. In order to present this interface, all packets that are transferred between an Axon and a conventional Ethernet device must be converted between Axon packets and Ethernet frames.

Axons use a bootstrap protocol to determine whether each port is connected to another Axon or to a traditional Ethernet device. When connected to another Axon, packets are simply forwarded as described in Section 3.1.2. Otherwise, packets received from an Ethernet device are encapsulated in an Axon header and packets sent to an Ethernet device are stripped of their Axon packet header.

Locally connected hosts will broadcast DHCP and ARP requests and expect replies from the appropriate hosts. The local Axon intercepts these address and location discovery messages. An Axon can respond to DHCP requests directly with lease offers, or the requests can be forwarded to a known DHCP server. Once the host has been configured with an IP address, it will use ARP to find other hosts. The local Axon is responsible for collaborating with the Axon connected to the target host in order to set up routes to allow communication in both directions between the hosts (further described in section 3.2.3).

When the next forward hop of an Axon packet indicates a port which is connected to a traditional Ethernet device, such as a host, the Axon header will be stripped from the packet, leaving a normal Ethernet packet. The packet is then forwarded to the host, which will never know that the packet had previously been encapsulated in an Axon packet.

3.2 Axon Device Architecture

Figure 3.2 shows the overall architecture of the Axon Ethernet device. An Axon includes both a hardware switching fabric—the *data plane*—and a processing element to perform control operations for locally connected hosts—the *control plane.* As the figure shows, the data plane is implemented in hardware for performance and the control plane is implemented in software for flexibility. While IP routers bear some similarity to this high-level architecture, the Axon device is much simpler, and therefore can be faster and more cost effective. This section describes the data and

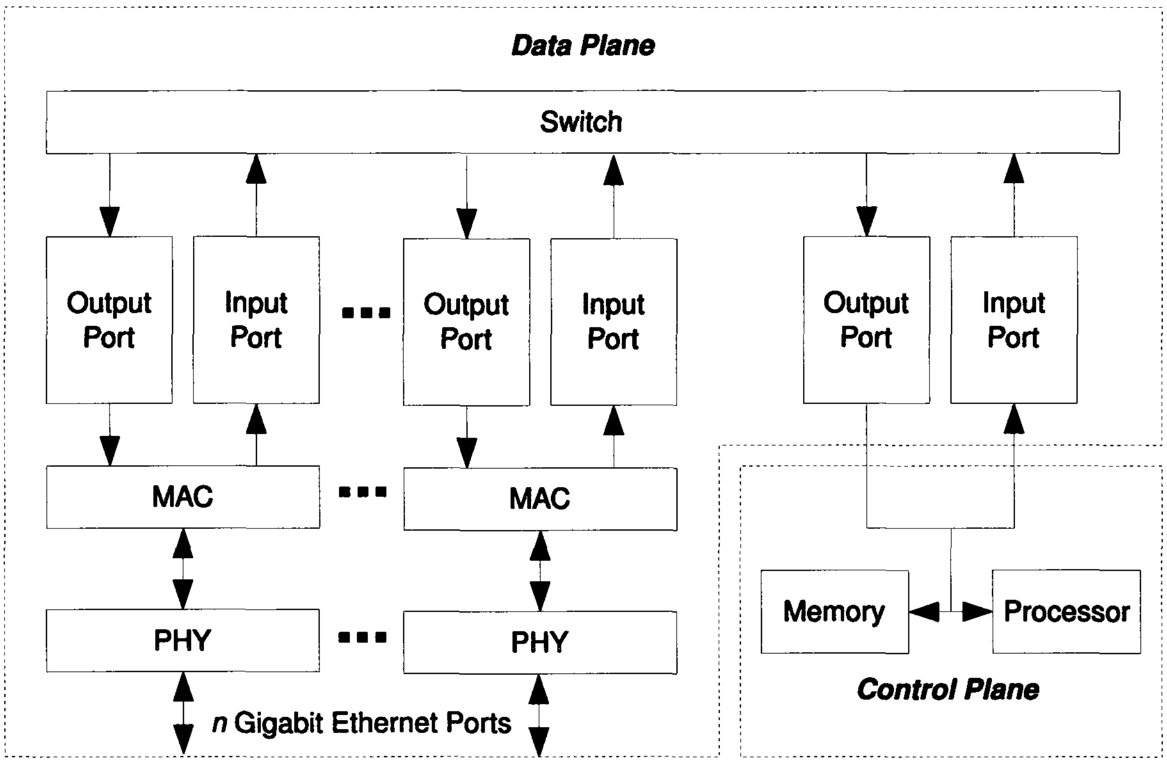


Figure 3.2 : Axon Architecture

control plane architecture in detail.

3.2.1 Axon Data Plane

Each Ethernet link is connected to the Axon via an *Ethernet port* in the Axon data plane. Figure 3.3 shows the architecture of an Ethernet port, which consists of an input port and an output port. The input and output ports are all interconnected via a switch.

The control plane configures each Ethernet port as an *Axon port* or a *host port.* A host port is connected to a conventional Ethernet device, so all packets crossing its

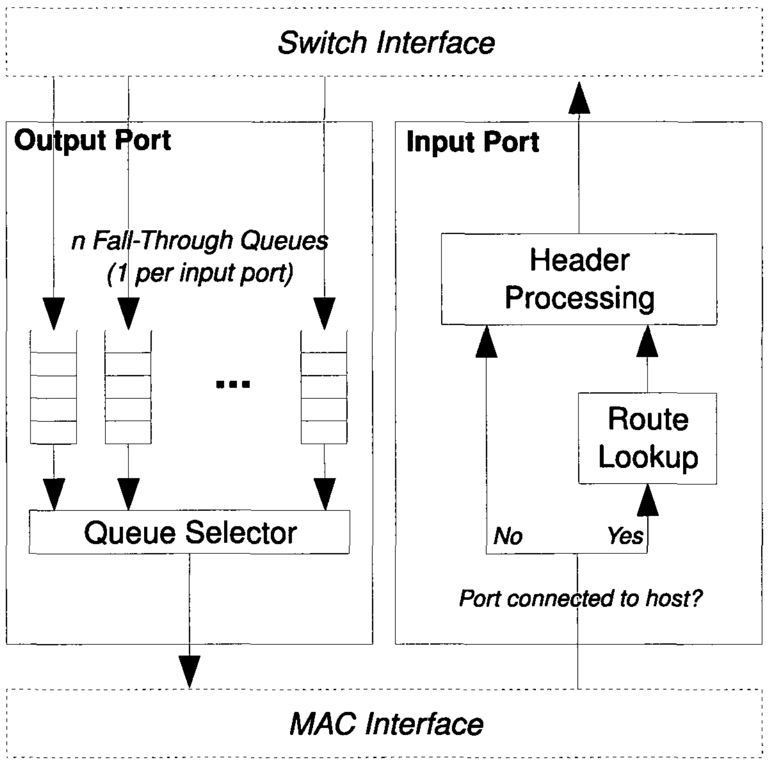


Figure 3.3 : Axon Data Plane Ethernet Port

Ethernet link are standard Ethernet frames. In contrast, an Axon port is connected to another Axon, so all packets crossing its Ethernet link are Axon packets. The control plane is also connected to the data plane as if it were an Ethernet link. The control plane configures its port as an Axon port and injects and receives only Axon packets.

Input Port

When a packet is received over an Ethernet link, the packet is first processed by the input port connected to that link. As Figure 3.3 shows, packets received on a host port are processed by the route lookup module and then the header processing module. Packets received on an Axon port are only processed by the header processing module.

Route Lookup. Packets sent by a host connected to an Axon will always be Ethernet frames. These frames cannot be routed through the Axon network. Instead, they must first be encapsulated in an Axon packet. The *route lookup* module uses the destination MAC address of the received Ethernet packet in order to find the correct route in route memory.

The Axon uses one of two mechanisms to determine the source route to be prepended to incoming data packets. In the first case, the destination MAC address maps to a source route via a content-addressable memory (CAM) entry in the Axon. If the CAM is not large enough to store all the mappings for active routes, the Axon may provide the host a fake destination MAC address when it first sends an ARP request for the destination. The host then maps the destination's IP address to this masqueraded MAC address, which is a direct index into route memory. The Axon sets the locally administered bit in the masqueraded MAC address returned to the host. In this way the mapping function is pushed to the host as opposed to the Axon when the Axon's CAM is exhausted.

Each Ethernet port has its own private route memory that is only used by host ports. This route memory contains Axon headers with source-routes that have been configured by the control plane in response to ARP requests made by the host. Each Ethernet port has its own route memory in order to allow network security policies that require isolation among the hosts.

The route lookup module therefore uses the destination MAC address of the incoming packet to determine the address of the Axon header in route memory. If the locally administered bit is set, then the Axon uses the MAC address as the index directly; otherwise it looks up the address by using the CAM. The header is then retrieved from route memory and prepended to the Ethernet frame, creating an Axon packet. Since the destination may be expecting the source to have a different MAC address (in case it was given a masqueraded MAC address),the source and destination addresses of the Ethernet packet are also replaced with the MAC addresses that the target is expecting–these are are stored in the route memory immediately following the Axon header. If the CAM lookup fails or if the locally administered bit is set but the destination MAC address does not encode a valid route index, then a default Axon header is prepended to the frame with a single forward hop that targets the local control plane. A lookup failure which causes a packet to be forwarded to the control plane is not necessarily an error. This is how broadcast traffic is captured and sent to the control plane, for instance.

Once an Ethernet frame has been processed by the route lookup module, it has become a valid Axon packet like any other Axon packet and can be processed by the header processing module. Note that regardless of the length of the route a packet must traverse, this initial route lookup on a host's local Axon is the only time any route lookup will be performed. For all subsequent hops, the packet will arrive at an

Axon port and skip the route lookup module.

Header Processing. Packets arriving on an Axon port or packets that have been processed by the route lookup module are processed by the header processing module. This module uses the source-routing information to determine to which port to forward the packet and then modifies the header appropriately.

The header processing module first reads the next forward hop to determine the correct output port to which the packet should be forwarded. If there are no remaining forward hops, the packet type is changed to an error packet and it is forwarded to the control plane.

If the output port is an Axon port, the header is then modified for the next hop through the network. First, the forward hop count is decremented and the reverse hop count is incremented. Second, the first forward hop is removed from the header, and subsequent hops are shifted forward. Finally, the input port number is inserted as the first reverse hop. These modifications to the header can be made as it is sent to the output port over the switch.

If the output port is a host port, the Axon header is completely removed from the packet, leaving a valid Ethernet packet. However, if the packet type is not encapsulated Ethernet, then the packet type is changed to an error packet and it is forwarded to the control plane.

Axon packets that are destined for the control plane are treated as any other packet. In their source-routes, their next forward hop will be '0xf'. They will be sent over the switch to the control plane's output port, which will then "send" the packet to the control plane.

Switch

The non-blocking switch within the Axon data plane provides connectivity between all of the input ports and all of the output ports. Each input port can inject a packet into the switch at any time. Before injecting a packet into the switch, the input port prepends a one byte header which indicates the type of packet and the destination output port. Each output port can simultaneously receive a packet from every input port. The output port must either accept packets that are sent to it or it must send a negative acknowledgement (NACK) back to the input port. An output port will only NACK a packet if the output port is not able to buffer the packet or immediately transmit it over the Ethernet link. When an input port receives a NACK, it can either drop the packet, retry sending it to the output port later, and/or send a congestion message back to the previous Axon in the path to slow down the incoming packet stream.

Output Port

The output port receives packets that have already been processed by the header processing module in an input port. This means that packets received by the output port are immediately ready to be sent out over the Ethernet link with no further modifications.

The output port contains one buffer per input port in the data plane. This allows the output port to simultaneously receive packets from an arbitrary number of input ports. This eliminates head-of-line blocking problems at the input ports and simplifies the task of fair bandwidth allocation of each outbound link among input ports. Even though all packets received by the output port are buffered, the buffers all support *fall-through* operation. So, the entire packet does not need to arrive at the output port before it can be sent over the Ethernet link. When the output port is not congested, packets are sent immediately from the switch to the Ethernet link. This allows the Axon to operate as a cut-through router, where the Ethernet packet can begin to leave the output port before it has even been fully received on the input port.

3.2.2 Probe Packets

In order to determine the best source route for a flow, an Axon may wish to test out different candidate routes. In order to do this, an Axon can send a *probe packet* out along an arbitrary path that ends at the Axon that originally sent the probe packet (forming a circuit). Probe packets are transmitted like other Axon packets except that Axons along the path add data to the tail of these packets depending on which probe option bits are set in the packet. A probe unit sits on both the input port and the output port in each Axon. If a probe packet is seen and the timestamp options are set, then the input and output units will attach to the end of the packet the time that the packet was first seen by that unit. Timestamps cannot be synchronized across devices, but the timestamp from when a probe packet is received by a device can be compared to the timestamp from when that probe packet was originally sent by the same device to compute the intervening latency. In this way, Axons can determine the raw latency of different paths across the Axon network and possibly determine which Axons may be congested since timestamps are attached both at the input and output ports of each Axon on the path. In general, probe packets offer a way for one Axon to gather data about another Axon's hardware status.

While in the above scenario probe packets are used to gather latency and congestion data about different paths within an Axon network, a probe packet may also measure the latency of external Ethernet networks. This kind of probe packet is referred to as an *ether probe*. In this case a single Axon specifies an source port and a destination port, which are both assumed to be connected to the same external Ethernet network. First, the Axon sends a packet from the destination port to the source port to ensure that the external network performs address learning. Next, it sends a packet from the source port to the destination port, and the timestamps on the Axon from when it left and returned are compared to determine the amount of time the packet spent in the external network.

3.2.3 Axon Control Plane

The primary responsibility of the Axon control plane is to handle packets that cannot be directly switched by the hardware data plane. To allow flexibility, the control plane is implemented in software running on a low-power embedded processor in the Axon. The control plane has a port into the switching fabric of the data plane, as shown in Figure 3.2. This allows the control plane to receive packets from and inject packets into the data plane seamlessly. The data plane forwards to the control plane all Ethernet packets from a host with a destination address that does not indicate a valid source-route in that host port's route memory. This includes all broadcast traffic. The most common task of the control plane is to handle Ethernet broadcast packets transmitted by local hosts, such as DHCP and ARP.

DHCP

As described in Section 2.2.1, DHCP enables a host to dynamically discover its IP address. The initial DHCP discovery message is broadcast onto the Ethernet by the host. As with all Ethernet broadcast traffic, DHCP discovery messages will be forwarded to the control plane. The control plane can then either forward the DHCP traffic to a conventional DHCP server or act as a DHCP server itself and respond to the host directly. In the latter case, each Axon can immediately assign any of its local hosts an IP address from a locally reserved pool.

All communication between the host and the DHCP server is guaranteed to be forwarded to the control plane. The broadcast DHCP discovery and request messages from the host will always be sent to the control plane. The Axon will use a source Ethernet address that will never correspond to a valid source-route in the DHCP offer and acknowledgement messages it returns to the host. When the host subsequently tries to renew its lease with a unicast message, it will use that Ethernet MAC address. The data plane will then forward the message to the control plane, as the address does not correspond to a valid source-route.

Transparent Route Creation

As described in Section 2.2.2, hosts use the ARP protocol to determine the location of other hosts on the Ethernet network. The control plane then uses the *Axon-ARP* protocol to satisfy the request. The Axon-ARP protocol involves two Axons: the *source* Axon, which is connected to the source host making the request, and the *target* Axon, which is connected to the host that is the target of the request. The source and target Axons must communicate in order to setup routes in both directions between the source and target hosts.

Upon receiving an ARP request, the source Axon first reserves sufficient space in the source host's input port's route memory to hold an Axon header containing a route from the source host to the target host. Then the source Axon sends an "Axon-ARP request" to the control plane of the target Axon. This request includes the source host's real MAC address (taken from the ARP request), the MAC address corresponding to the allocated route memory in the source host's input port, the IP address of the source and target hosts, and the Axon ports to which the source and target hosts are connected.

Note that the Axon cannot reserve space in the route memory or send an Axon-ARP request if it does not already know of a path to the target host and target Axon. Therefore, additional communication may be required to determine such a route. For the prototype Axon implementation, the network topology and route selection is set statically via a configuration file. This allows the exploration of other aspects of the Axon architecture, but is clearly not reasonable for a realistic deployment. Others have already proposed mechanisms to more realistically determine routes. The two main ideas are to use a distributed hash table [6] or a central controller [15]. In principle, the Axon control plane architecture can support either method.

When the target Axon receives an Axon-ARP request, it also reserves space in the target host's input port's route memory to hold an Axon header containing a route back from the target host to the source host. Note that the target Axon will always know a route back to the source because it is encoded in the reverse path of the Axon header of the request. The target Axon then sends a standard ARP request to the target host. The source MAC address used in this request depends on whether there is room in the CAM on the target Axon. If there is enough room, then the source host's real MAC address is given as the source MAC address; otherwise, the source MAC is masqueraded and corresponds to the allocated route memory in the target host's input port. The masqueraded MAC address has the locally administered bit set. Regardless of whether the real MAC or masqueraded MAC is used, when the target host sends back an ARP reply to the MAC address, the data plane will forward it to the control plane, since the reserved route memory is not yet marked as valid.

When the target Axon receives the ARP reply from the target host, it can then install a route to the source host. The route's Axon header will include the route back to the source host, which is a combination of two paths: 1) the reverse path from the Axon-ARP request header (to get to the source Axon) and 2) the source host port in the Axon-ARP message (to make the final hop to the source host). The control plane also stores destination and source Ethernet MAC addresses in the route memory. These addresses will be used by the input port to modify the Ethernet header so that it will match what the source host expects when a packet from the target host arrives at the source host. The destination MAC address is always the real MAC of the source host, but the source MAC address is real MAC address of the target host only if the source Axon used the CAM to map to the route in route memory. Otherwise, the source Ethernet address is the masqueraded MAC address that directly indexes to the allocated memory in the source host's input port.

After installing a valid route, the target Axon can respond to the control plane of the source Axon with an "Axon-ARP reply". This reply is similar to the request, in that it includes the target host's real MAC address (taken from the ARP reply), the MAC address that corresponds to the allocated route memory in the target host's input port, and the IP addresses and Axon ports of the source and target hosts.

When the source Axon receives the Axon-ARP reply, it can complete the route setup for the source host. It uses the MAC addresses in the reply to place the Axon header in the previously allocated source host's route memory. Finally, the source Axon can respond to the source host with a normal ARP reply. This ARP reply will use as the source MAC either the real MAC of the target host (if the CAM had room to store the mapping) or a masqueraded source MAC address that corresponds to the allocated route memory in the source host's input port. When the host receives the ARP reply, the source and target can begin sending packets directly between each other. The input ports will find a valid route in each direction and will therefore forward the packet along the appropriate path with no further intervention from the control plane.

The route from the target to the source is valid before the route from the source to the target is valid. In the unlikely event that the target sends a packet to the source during that time period (recall that the source is requesting a path to the target, so is likely the one to initiate any communication), it will arrive correctly at the source. If the source then responds before the source Axon has validated the route, the hardware will simply forward that packet to the control plane. At that point, the control plane can safely drop the packet, as this should affect only a small amount of traffic until the source Axon installs the correct route, and higher level network protocols should retransmit those packets.

3.3 Other Ethernet Concerns

3.3.1 Link Error Detection

As discussed in Section 2.1, a 4-byte CRC is used to detect Ethernet link errors. However, this CRC does not protect against errors that occur within a device, as it is calculated as an Ethernet frame is transmitted over a link and is verified and discarded when it is received. A conventional store-and-forward Ethernet device will discard any received packets with an invalid CRC. Since Axons employ cut-through routing, use of the CRC is a bit more complicated. By the time the receiver is able to verify the CRC, the head of the packet may already have been transmitted by its next hop output port. In this case, the output port would already be in the process of computing a new CRC on the invalid data. So, when the input port receives an invalid CRC, it must notify the output port so that it can append an invalid CRC to the end of the frame. This will ensure that the subsequent device will know that it has received an invalid frame. This may continue on each hop of the path until the frame is finally fully buffered, either in a congested Axon or a conventional Ethernet device. At that point, it will then be dropped due to the invalid CRC.

3.3.2 LACP

As mentioned in section 2.1.1, standard Ethernet may increase the bandwidth between two devices via link aggregation control protocol (LACP). Any network device is allowed to connect to a single Axon via multiple aggregated links. In fact, one device could even have aggregated links that connect to separate Axons. As long as a given conversation deterministically enters the Axon network along a single link, separate Axons (or multiple ports of the same Axon) would be able to continue to provide the source route for incoming packets within the Axon network. The ability to transparently split the aggregated links is called *multipoint aggregation* and is not possible in a standard Ethernet network.

Increasing the bandwidth between any two Axons within the Axon network is trivial and does not require LACP. One would simply create the desired topology, including extra links. Then when assigning source routes, the mechanism could choose which link to use, perhaps based on the current amount of traffic observed along the links in common with the two devices.

Similarly, an Axon may send packets to another network device that is connected to multiple ports via LACP. In this case, the Axon network would be in charge of the distribution function. Since all source routes are based on source and destination Ethernet addresses, they would essentially go through the distribution function at the time of source route creation. Further, the source route could be changed dynamically by updating the source route on the source Axon.

That the Axon does not require a hash for the distribution function is a fundamental advantage of the Axon network over a standard Ethernet network. This allows the Axon to have arbitrary flexibility in assigning conversations to links, which could aid in balancing the traffic between the links.

3.3.3 Self-Congestion

Since the Axon prepends an Axon header to each packet originating from a host, each packet that leaves an Axon is larger than when the host sent it. When packets from a host enter the Axon at the maximum rate for some time, the extra data due to the new Axon headers may fill up the buffer on the output port, even in an otherwise quiescent network. If left unchecked, the buffer may eventually drop a packet. We call this phenomenon *self-congestion*.

Currently, the Axon solves the self-congestion problem by using *pause frames*, an Ethernet-based flow control mechanism. When one device sends a pause frame out of an Ethernet port, the connected device must refrain from sending any packets on that port for the duration specified by the pause frame. The Axon currently sends a pause frame to a host after the host has generated a maximum-sized packet's worth of Axon header data (which occurs once every several hundred packets when there are only a few hops to the destination). This pause frame instructs the host to stop transmitting for as long as it would take to transmit a maximum-sized packet. In this way, the buffer in the Axon may drain any excess data due to Axon headers and will not drop a packet due to self-congestion. If the packet were lost, a higher-layer protocol such as TCP might need to retransmit the packet, but potentially at the cost of reducing its bandwidth.

3.4 Axon Limitations

3.4.1 Lack of Flooding

One way that the Axon combats the challenges of scaling Ethernet is by limiting the types of packet flooding. The Axon intercepts all packets to be broadcast on the network, but it only handles ARP requests and DHCP requests. This will take care of most networks' requirements, including that of typical data centers; but other networks may need to use Ethernet's broadcast mechanism for different protocols. The Axon currently does not support this, but if a central controller were to manage a network of Axons, it could intercept these broadcast packets and perhaps handle them on a protocol-by-protocol basis.

3.4.2 Fault Tolerance

We envision that a higher-level control system (such as Tesseract [15]) would manage faults that may occur in an Axon network. When a fault occurs, surrounding Axons will know either by a link failure or by receiving many packets with invalid CRCs. Once one Axon detects the fault, it could notify a central controller, which would then reassign source routes going through the faulty Axon.

3.4.3 Dependence on ARP/IP

As currently designed, the Axon is dependent on intecepting an ARP request in order to create a source route. It "knows" where the target IP address lives and then handles the Axon-ARP mechanism from there. This works well in a network where the Axon is responsible for assigning IP addresses, but this may be troublesome when protocols other than IP are used. One solution would be to wait until a host sends out a packet, and then the Axon could learn its Ethernet address by examining the source MAC address, just like a standard switch does. However, if an end host is silent on this network, another mechanism may need to be developed in order to find the host.

3.5 Axon Benefits

The following subsections outline some of the benefits that data centers can enjoy by using an Axon network.

3.5.1 Local Route Lookup

The Axon network architecture represents a fundamental departure from conventional network architectures as all state is stored at the edge of the network. In current networking technologies (i.e., Ethernet switches, IP routers, etc.), routing information is required on all devices along the path from the source to the destination. This often requires every device to store at least partial routing information for every destination in the network, and this state is accessed by every packet traversing the device. Even proposed network devices, such as OpenFlow switches [14], still require access to routing state on every hop through the network. Storing state on every device for all traffic flows traversing the device is not scalable. In contrast, Axons only store routing state on behalf of hosts connected directly to that Axon. Therefore, the required state scales only with the number of hosts that are connected to an Axon, not with the diversity of traffic that flows through an Axon. An Axon does not need to have any routing or topology information for traffic that it is forwarding. Fundamentally, this is a far more scalable network architecture than the state-of-the-art.

3.5.2 Arbitrary Paths

The Axon network is not constrained to use any particular path for a traffic flow. When necessary, shortest-path routing can be used, but in other instances different routes could be chosen. For example, a longer path may be chosen if the shortest path uses a link that is congested. Arbitrary paths are allowed, and they remain transparent to end hosts.

3.5.3 Security

By controlling access to the network at the source, transparent source routing also enables efficient network virtualization. The existence, or lack thereof, of a source route to the intended destination determines whether that host is allowed to communicate with that destination. There is no need for inter-VLAN routing. Instead, the control planes across the network provide distributed access control among hosts throughout the network.

3.5.4 Virtual Machine Migration

That the entire network is one Ethernet segment yields advantages for both manageability and virtual machine (VM) migration. Administrators need not worry about how the IP address is segmented because any host on the network can have an arbitrary IP address and may still communicate with all other hosts. Furthermore, VMs are usually restricted to migrate within their subnet because otherwise the VM would need to change its IP address. In an Axon network, any VM can migrate to any host and retain its IP address, provided that the control plane reconfigures the appropriate source routes.

3.5.5 Efficient Use of Redundant Links

The use of source routing also frees the network from any topology constraints, unlike some other approaches to scaling Ethernet [16] [5]. The Ethernet spanning tree disables all redundant links from the network. These redundant links are only utilized in response to link failures. Typically, link aggregation is used to prevent bandwidth bottlenecks because of the lack of redundancy in the spanning tree. Since flows are distributed across aggregated links using hashing, link utilization can easily become unbalanced, leaving the available capacity underutilized. In contrast, the use of source-routing allows an Axon network to exploit redundant links effectively. These links can be used to easily increase network bandwidth, respond to link failures, and avoid congestion.

3.5.6 Manageability

The Axon device does not in itself make a case for how the network should be managed; rather, it provides a flexible, scalable network primitive: the source route. Other proposed mechanisms can be used to provide control for the network and set up the source routes [15].

3.5.7 Benefits Over Myrinet

Similar to the Axon device, Myricom's 10Gbps network devices, Myri-10G, all use commodity Ethernet physical interfaces [17]. This means that Myri-10G adaptors and switches can interoperate with conventional 10Gbps Ethernet adaptors and switches. However, to enable this functionality, the switches must be equipped with special network processors to convert Ethernet packets into Myricom packets. Furthermore, Myrinet networks also use source-routing for performance. However, the source-routing is controlled by the Myri-10G adaptors, not the Myri-10G switches. Therefore, in contrast to the Axon device, which allows commodity systems to obtain higher network performance, Myri-10G switches only provide improved network performance when the host systems also use Myri-10G adaptors. The Axon architecture will therefore be able to provide better network performance for commodity host systems.

Chapter 4

Axon Performance

In this chapter we evaluate the Axon architecture. This includes a description of the platform we chose and how the Axon provides better latency and bandwidth efficiency than standard Ethernet networks.

4.1 Implementation

Figure 1.1 shows the prototype Axon device used to evaluate source-routed Ethernet. In the prototype, the hardware data plane is implemented on Stanford's NetFPGA platform [18], and the software control plane runs on an Intel Atom processor in a D945GCLF mini-ITX motherboard. Communication between the data and control planes takes place over the PCI bus in the prototype.

The NetFPGA platform is a 32-bit/33MHz PCI card that includes four Gigabit Ethernet ports, a Virtex-II Pro 50 FPGA connected to those ports, several memories, and other essential components (Ethernet PHY, PCI interface, etc.). The data plane is entirely implemented with the Virtex-II Pro FPGA on the prototype. While a true Axon device would likely have more Ethernet ports, the NetFPGA effectively limits the prototype to four. However, four Ethernet ports are sufficient to demonstrate the viability of the Axon device.

The Intel Atom is a low-cost, low-power, hyperthreaded x86 processor that is well suited to embedded network devices. The prototype runs a standard x86 Linux kernel on the Atom, allowing the control plane to be implemented as a user-level application. Creating a user-level application as the control plane simplified its development and ensured its portability. The NetFPGA Linux device driver presents NetFPGA as four regular Ethernet network interfaces. The control plane software communicates with the data plane using Linux raw packet sockets over one of these interfaces. The bandwidth between the control and data planes is limited by the available PCI bus bandwidth in the prototype. In practice, there is more than enough bandwidth for the address and location discovery tasks currently performed by the control plane.

4.1.1 Why Use a Prototype?

There are many ways to evaluate the Axon architecture. Some possibilities include simulating the Axon, implementing it on a software-based routing platform such as XORP or the Click Router, or implementing it on an FPGA. We chose to implement the Axon on NetFPGA because we felt that this would be the most convincing way to demonstrate the viability of the architecture. We have shown that we can manipulate the Ethernet headers while remaining compatible with all standard Ethernet devices. It is often easy to overlook subtle problems that may arise in the real hardware when simulating a device. By creating an actual hardware device that runs in an actual network, we have proven that it is viable.

One unexpected problem that arose in testing the Axon is that a source host will sometimes send a unicast ARP packet to a target host that it is communicating with. If, however, the target host believes that the MAC address of the source host is a different one (due to MAC masquerading), then it will reply to a MAC address that does not map to a source route in the connected Axon; thus the ARP reply will not be received by the source host. In order to fix this, we modified the Axon to intercept *all* such ARP requests and reply to them. If we had strictly run the Axon in a simulator, for instance, we may not have discovered this problem.

The main argument against implementing a hardware prototype is that it would be difficult to investigate the scalability of the Axon network. While it is true that our Axon network is currently limited by the number of physical Axons we can construct, we felt it was first more important to persuasively demonstrate that the Axon is a viable architecture. The rest of this chapter will show that the Axon is indeed viable; furthermore it could be the substrate for network architectures already demonstrated to be scalable in simulation (Tesseract, for instance [15]).

4.2 Functionality

After implementing the prototype Axon, we verified its functionality by using it as a replacement Ethernet switch for different hosts. We connected a Mac host, a Windows host, a FreeBSD host, and a Linux host to verify functionality across different operating systems. We also connected our Axon to a Cisco router as well as a wireless

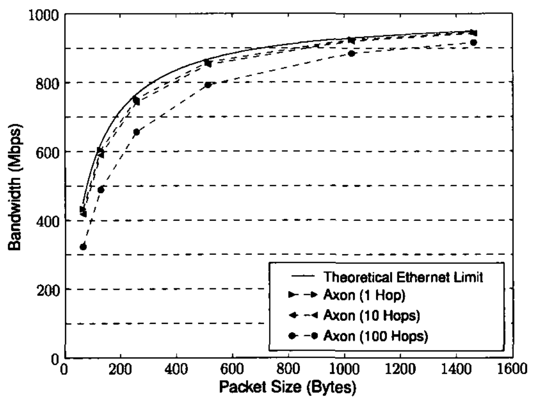


Figure 4.1 : Maximum TCP Bandwidth for Axon and Ethernet Packets

router to verify that it would work properly when connected to other network devices.

4.3 Bandwidth

4.3.1 Axon Header Overhead

A disadvantage of source-routing is that it reduces the effective bandwidth available at the physical layer. In effect, application layer data is displaced by the source-route. Figure 4.1 presents the maximum theoretical TCP bandwidth that can be achieved over a 1Gbps physical link with Axon packets of varying sizes containing source-routes of 1, 10, and 100 hops. The theoretical Ethernet limit shows the effective bandwidth over an Ethernet link given the Ethernet interframe gap and protocol overhead at the datalink, IP, and TCP levels. As the figure shows, the overhead of source-routing on Axon network bandwidth is negligible for packets with routes containing 1 or 10 hops. For 100 hop routes, the maximum bandwidth only decreases by 3% at the

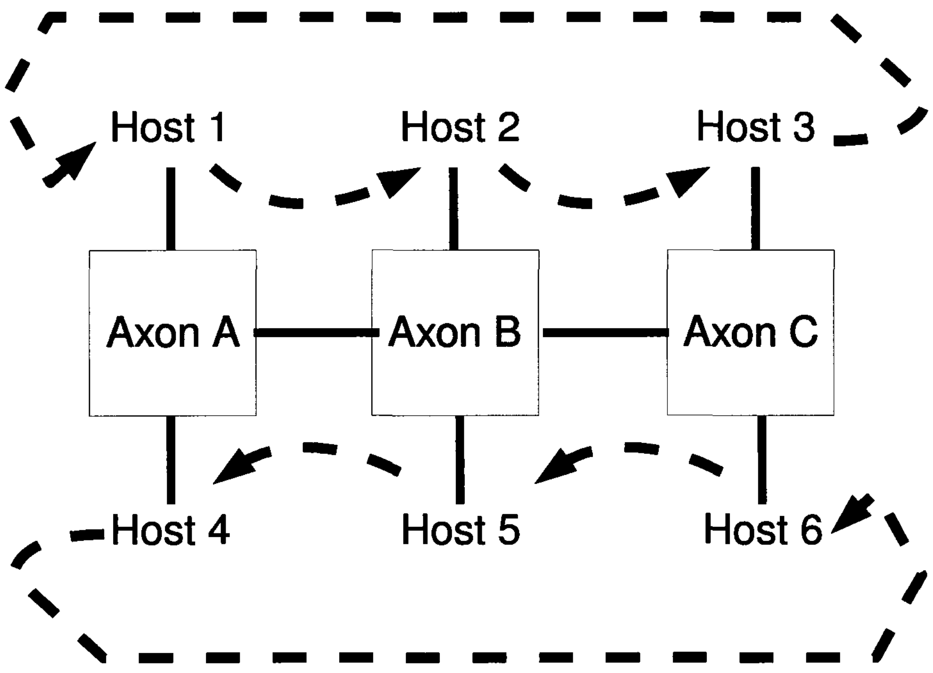


Figure 4.2 : Line topology and flows used in Table 4.1

largest packet size.

4.3.2 Improvement Over Spanning Tree

One major advantage of Axons over standard Ethernet networks is that it allows for arbitrary topologies, which may include cycles. Figure 4.2 shows a network of three Axons and six hosts connected in a *line*. This resembles the simplest Ethernet network that is restricted by using a spanning tree. Figure 4.3 shows a similar network of three Axons and six hosts connected in a *ring*. The ring is the simplest network with a cycle and shows how the Axon network may take advantage of increased bandwidth.

Table 4.1 shows the bandwidths of the six flows between hosts for each topology for

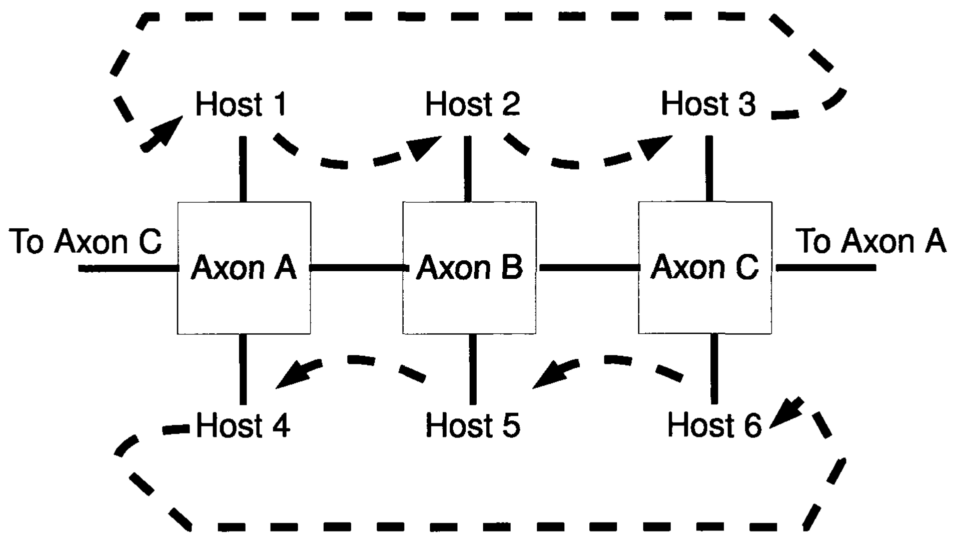


Figure 4.3 : Ring topology and flows used in Table 4.1

the UDP and TCP protocols. These bandwidths were measured using netperf; they include only the application-perceived bandwidth, so they do not include protocol overhead. In the line topology, there is network contention over the links connecting the three Axons because each of the six flows try to produce 1Gb/s of traffic, but there are only 4Gb/s available in the network (accounting for the bidirectional rate). In the ring topology, we can see a dramatic improvement in the bandwidth. Again, each flow attempts to utilize about 1Gb/s of bandwidth, but this time the network can provide 6Gb/s due to the extra link.

All individual flows see a marked improvement when using the ring topology over the line topology. The ring topology gives a 96% improvement in UDP's aggregate bandwidth and a 63% improvement in TCP's bandwidth over the line topology. The improvement in UDP's bandwidth is more pronounced since there is traffic in exactly

Flow	UDP		TCP	
	Line	**Ring**	**Line**	**Ring**
Host 1 → Host 2	481	952	566	752
Host 2 → Host 3	483	952	598	792
Host 3 → Host 1	476	930	243	815
Host 4 → Host 6	481	952	244	524
Host 6 → Host 5	476	952	397	493
Host 5 → Host 4	509	952	377	575
Aggregate	**2906**	**5690**	**2425**	**3951**

Table 4.1 : Bandwidths seen on different topologies, measured in Mb/s.

one direction per flow. TCP requires acknowledgement packets to be sent in the reverse path for a given flow, and these may cause some packet loss, which will throttle back the TCP bandwidth. These data show that a network can significantly benefit from using the redundant links that would otherwise be disabled by the spanning tree protocol.

4.4 Latency

4.4.1 Data Plane

By using source-routing and cut-through routing, the Axon is able to achieve low switching latencies. The prototype proves this to be possible. Table 4.2 shows a

Action	**From Axon**		**From Host**	
	Cycles	**Time (ns)**	**Cycles**	**Time (ns)**
MAC Receive	11	88	11	88
Cross clock domain	6	48	6	48
Pre-processing	6	48	2	16
Header lookup & retrieval	0	0	35	280
Header processing	9	72	9	72
Cross the switch and buffer at output	11	88	11	88
Cross clock domain	5	40	5	40
MAC Transmit	17	136	17	136
Total	**65**	**520**	**96**	**768**

Table 4.2 : Forwarding latency of a packet through the Axon device

breakdown of the 520 ns uncongested forwarding latency of an Axon packet (from Axon port to Axon port). This breakdown was determined using ModelSim to simulate the prototype design. These latencies reflect the use of a Virtex-II Pro FPGA and commodity soft-core Ethernet MAC units. Asynchronous FIFOs are used to bridge clock domains between the MACs and the internal Axon logic. While all run at the same clock frequency, each Ethernet link has an independent clock and is not guaranteed to be in phase with the rest of the system.

Almost half of the forwarding latency, 224 ns, is spent in the Ethernet MAC units. Much of this latency is unnecessary in the Axon. For example, the latency of the MAC receive unit is necessary only so that the MAC unit can receive enough of the Ethernet header to examine the destination MAC address and Ethernet type. In the Axon, this is unnecessary, as the header processing unit performs these functions for Axon packets. The remainder of the forwarding latency could be lowered using an ASIC or a faster FPGA. Regardless, the prototype's overall forwarding latency in the uncongested case is quite low.

Ethernet packets arriving on host ports have additional forwarding latency, as they must also be processed by the route lookup module. The latency of the route lookup module is 35 cycles (280 ns) plus an additional cycle for each 32-bit word (equivalent to 8 hops) in the Axon header that needs to be retrieved from the route memory. The smallest route header (a single hop and two Ethernet addresses) that can be stored in the route memory is 18 bytes. So, the minimum, uncongested forwarding latency of a packet from a host port is 768 ns.

4.4.2 Control Plane

In this section, we examine whether an Axon network is capable of handling the demands of hosts since an Axon network introduces a delay due to intercepting ARP requests and setting up routes between hosts.

Table 4.3 shows the amount of time spent in two Axons participating in the Axon-ARP protocol described in section 3.2.3. In this case, a source host wishes to communicate with a target host over an Axon network, and the route must be

Phase	**Activity**	**Latency (us) for this many hops**					
		1	2	3	4	5	AVG
I (Axon A)	Route Calculation	38	38	41	42	41	40
	Route Allocation	3	4	3	3	9	4
	Pseudo-ARP Request	116	122	100	110	138	117
	Total	157	164	144	155	188	161
II (Axon B)	Route Determination / Allocation	160	106	130	156	122	135
	ARP Request	90	58	102	66	90	81
	Total	250	164	232	222	212	216
III (Axon B)	Route Storage	33	35	50	66	27	42
	Pseudo-ARP Reply	139	135	102	130	122	126
	Total	172	170	152	196	149	168
IV (Axon A)	Route Storage	163	91	78	85	117	107
	ARP Reply	61	130	93	70	99	91
	Total	224	221	171	155	216	198

Table 4.3 : Control plane latency breakdown for an ARP request

created. The source host is connected to Axon A, and the target host is connected to Axon B. The following sequence occurs while the ARP request is being processed by the Axon network.

Phase I Axon A receives an ARP request. The route calculation step is how long it takes the Axon to determine the route to Axon B along a line. This is done purely in software. The route allocation step begins after the last step ends. It merely allocates the memory for the route in software, which is why it occurs so quickly. The last step constructs the pseudo-ARP request and sends it to the data plane to be sent to the desired Axon B. Axon A is busy for an average of 117us during this phase.

Phase II Axon B receives the pseudo-ARP request. It first examines the reverse route and allocates memory for this route to be stored. In the next step, Axon B constructs the ARP request to be sent to the target host. Axon B is busy for an average of 216us during this phase.

Phase III Axon B receives the ARP reply from the target host. The first step is where the route is actually written to the hardware (route storage). At this point, the route to the source host from the target host is set up such that the data plane can handle this on its own from this point forward. The second step is the amount of time that it takes for the software to construct and send the pseudo-ARP reply back to Axon A. Axon B is busy for an average of 168us during this phase.

Phase IV Axon A receives the pseudo-ARP reply from Axon B. The first step is to store this route to the data plane. Finally, Axon A constructs the ARP reply and sends it to the source host. Axon A is busy for an average of 168us during this phase.

Phases I and IV occur on Axon A, and phases II and III occur on Axon B. In this case, Axon A takes 359 us to handle each ARP request, and Axon B takes 384 us for each ARP request. If we take the greater of these two times, we find that each Axon can set up approximately 2600 routes per second for locally connected hosts. Since this rate only affects locally connected hosts, it does not necessarily limit network scalability.

4.4.3 Cut-through vs. Store-and-Forward

Comparison to Ethernet Switch

The latencies given in section 4.4.1 underscore the performance benefits of cut-through routing. An Ethernet switch would have all of the latencies shown in Table 4.2, except for the header processing. However, Ethernet switches are generally store-and-forward, incurring an additional 512–12112 ns delay (depending on packet size) simply storing the entire packet. There would also be some additional delay to lookup the destination MAC address in the forwarding table, although this could be done in parallel with storing the remainder of the packet.

Figure 4.4 shows the latencies of Axons and store-and-forward Ethernet switches, measured using probe packets (described in Section 3.2.2). A single Axon transmits a probe packet into the network and then receives that same packet after it has traversed the Axons or switches. The timestamps were then compared to calculate network latency. As the figure shows, the latency of a single Axon device is approximately 1 *us*, regardless of the packet size, and the latency of a single Ethernet switch is approximately 7 *us* for a minimum-sized Ethernet frame and is over 28 *us* for a maximum-sized frame. These latencies scale roughly linearly, leading to a latency of

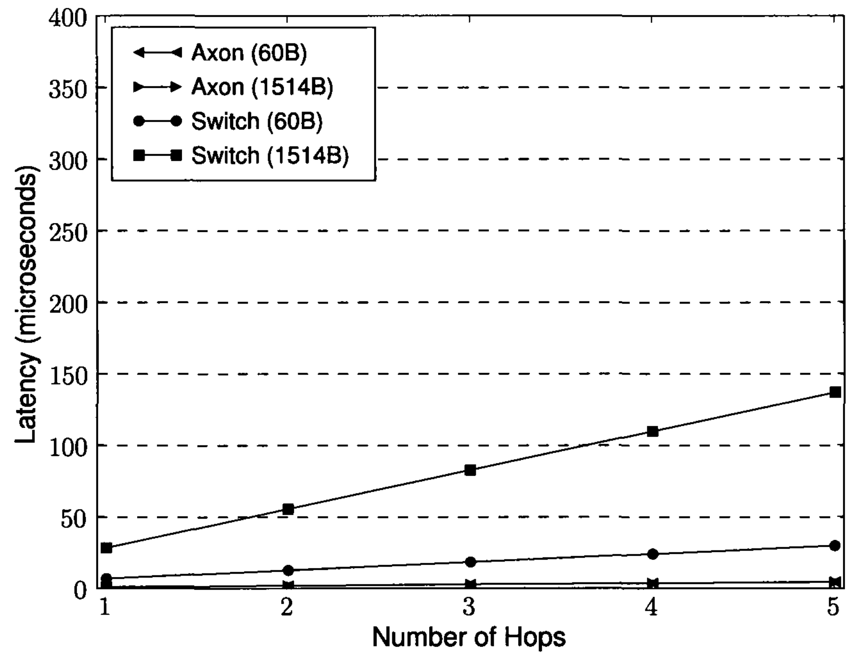


Figure 4.4 : Network latency of small and large probe packets

4.5 *us* for 5 Axon hops and 137 *us* for 5 switch hops.

This latency difference is large enough that it is clearly noticeable to the host system. Figure 4.5 shows the latency of Axons and Ethernet switches as measured by a host system. This graph shows the round-trip latency of an ICMP ping packet through 1–5 Axons or switches. Interrupt coalescing in the network interface is turned off, as interrupt coalescing would delay the received ping given that no other network traffic is being received by the host. The delays incurred by the host itself are so high that the network latency of even 5 Axons is barely noticeable in comparison. This is clearly not the case for Ethernet switches. The ping latency is doubled when the

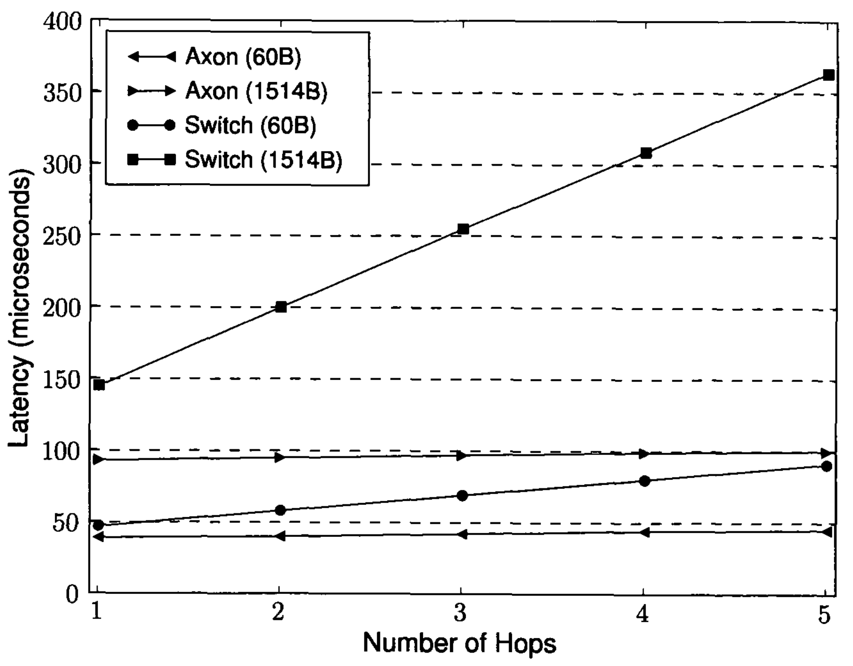


Figure 4.5 : Network latency of small and large ping packets

ping traverses 5 Ethernet switches when compared to traversing only 1. Clearly, a larger local-area network can be created out of Axon devices than Ethernet switches before host systems will begin to experience latency-related network problems.

Cut-throuth with Congestion

In order to fairly compare the advantages of cut-through routing versus store-and-forward routing when the network is congested, we implemented a software-programmable register in the Axon that allows the control plane to have the data plane switch between both modes of operation. This way, the only difference between experiments

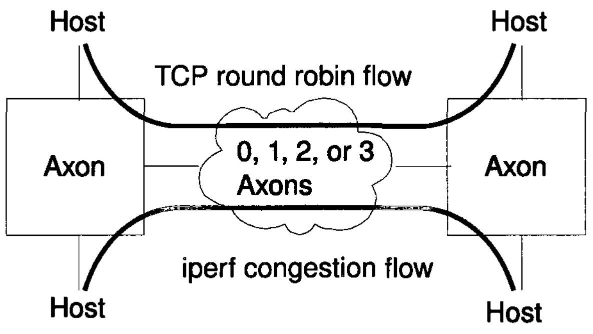
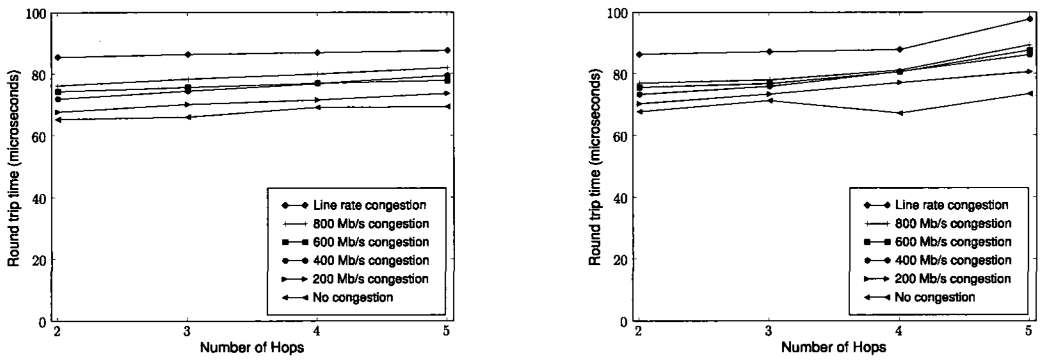


Figure 4.6 : Experimental Setup for measuring impact of cut-through with congestion

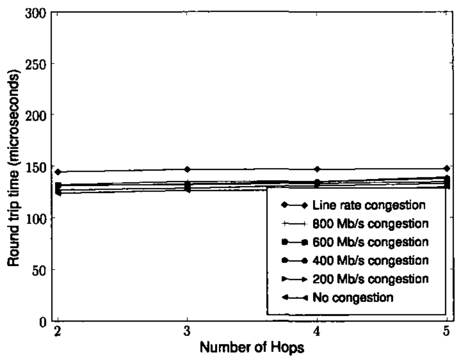


(a) Cut-through Latency (b) Store-and-forward Latency

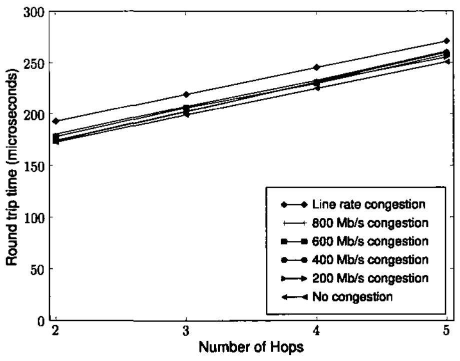
Figure 4.7 : Average round trip time of small packets (64 bytes) under different congestion loads

would be whether the same device implements cut-through or store-and-forward. All other implementation latencies and functions are identical.

In the following experiment, two hosts are using netperf's TCP round-robin test. In this test, the source host sends a TCP packet to a target host, and waits to receive an ACK from the target. Once the source receives the ACK, it immediately sends out another packet to the target. Figure 4.6 shows the experimental setup for this



(a) Cut-through Latency



(b) Store-and-forward Latency

Figure 4.8 : Average round trip time of large packets (1514 bytes) under different congestion loads

test. Figures 4.7 and 4.8 compare the average round-trip-time (RTT) given by netperf under varying congestion loads on the network. Separate hosts connected to same Axons create the congestion traffic by each using iperf to send UDP packets across the network at a configurable rate.

Comparing figure 4.7a to figure 4.7b, it is clear that for 64B packets the benefit of cut-through is inconsequential because the amount of time taken to store a small packet is minimal. This remains true whether the network is congested or not. However, large packets do see significant latency benefits from cut-through as shown in Figure 4.8.

In all of these cases, congestion affects primarily the first hop of both the outgoing and incoming paths. In the rest of the hops to the destination, both flows no longer compete for any output ports since the flows have been merged together.

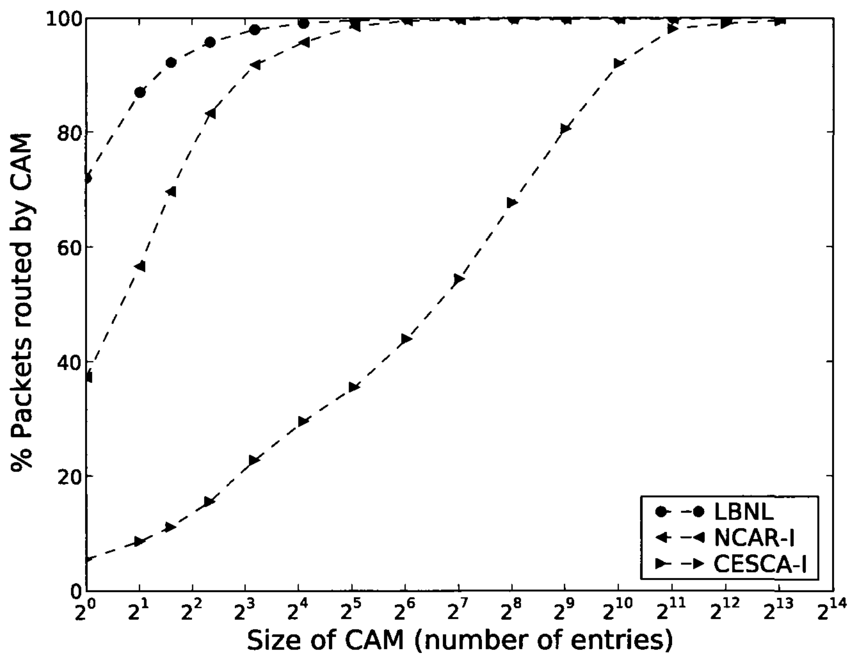


Figure 4.9 : Size of CAM needed to route packets from different traces

In practice, a cut-through device will not hurt flows in a network, but the benefit will vary depending on the number of congested and uncongested links along a path, the size of the packet, and the mechanism that handles fairness when multiple packets compete for one port.

4.5 Route Memory

One concern about the Axon architecture is whether it can support enough entries in each port's CAM. NLANR provides several long traces from gigabit routers that can help address this concern. CESCA-I is a 3-hour trace that covers a gigabit link between an Internet-facing router and the scientific ring in Europe. NCAR-I is a 1-hour gigabit trace covering traffic seen from the Internet to a router in the National Center for Atmospheric Research. LBNL is a trace from two routers that route between 22 subnets at Lawrence Berkely National Labs. This trace most closely resembles that which might be seen in a data center.

Each trace was analyzed to find the number of packets in between two packets whose destination IP address repeats. The number of packets in between (plus 1) represents the number of entries in an Axon's CAM that would be needed in order for an Axon's CAM to support all routes during this trace. Figure 4.9 shows how many entries in our CAM would be needed to handle these gigabit traces. This shows that each port of an Axon would need about 4000 entries. To compare, modern switches are able to support CAMs whose size is even much larger than 4000.

4.6 Application Benefit

The latency benefits of the Axon can translate into performance improvements for latency-sensitive applications. PostMark is one such representative benchmark. PostMark is a file system benchmark that approximates a large Internet e-mail server [19]. PostMark creates a large pool of continually changing small files. Figure 4.10 shows the performance of PostMark when a client accesses an NFS server via a network of Axons or Ethernet switches. The NFS file server was configured to use a RAM disk to eliminate disk latency, and PostMark was used as a client to perform read and

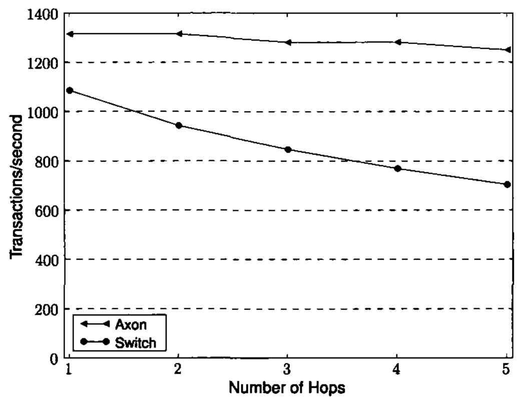


Figure 4.10 : PostMark Performance

write accesses on a random set of files of different sizes.

The graph shows that Axons outperform Ethernet switches for 1–5 devices. This range of devices is a reasonable approximation of the number of network devices likely to be on the path between a client and server in a campus-sized network. Furthermore, the trend is clear as the number of devices increases.

The latency of an Ethernet switch, with its store-and-forward design, clearly degrades the file system performance as the number of switches increases. When using an Axon network, however, the additional cut-through latency added by each Axon is minimal, and thus the file system performance remains nearly constant as the network size increases. Thus, the performance penalty of using centralized network storage is significantly reduced with the Axon network device.

Chapter 5

Conclusions

The Axon network device is an inexpensive, practical device that replaces an Ethernet switch in the data center. By employing source-routed Ethernet, a new datalink layer protocol, Axon networks provide lower latency and greater scalability than conventional switched Ethernet networks. The use of transparent source routing is necessary to enable these improvements. Connected hosts may enjoy increased bandwidth and scalability of the Axon network without modification. Axons only need to store routing state for locally connected hosts, and not for the entire network. In addition, route lookups are only performed at the initial Axon along a path, as opposed to every switch or router along a path. Dynamic address and location discovery services are provided by the local Axon, instead of requiring broadcast packets and packet flooding across the entire network.

The Axon prototype, which consists of an Atom processor and a NetFPGA PCI card, demonstrates the viability and strengths of the design. Axons can saturate 1Gbps Ethernet links and fairly allocate bandwidth among competing traffic flows when the network is congested. The Axon prototype can forward packets in less than 1 us per hop by using cut-through routing, in contrast to 7–28 us per hop with switched Ethernet. The bandwidth increase introduced by using redundant links in the network has shown an improvement of 96% for UDP flows and 63% for TCP flows.

We believe that the Axon will prove to be an attractive network substrate for the datacenter. The network can easily be managed, as the complexity of routers has been removed. It is inexpensive because no special netowrking devices are required of the hosts. The network can easily be partitioned, as virtual machines are free to move about the network and access control among virtual networks can easily be managed directly within the Axon. These properties will enable an Axon network to efficiently meet the demands of a large-scale, high-performance data center.

5.1 Future Work

Multicast and prioritization are two features of switched Ethernet that are not addressed by this thesis. Arguably, supporting packet priorities may be simpler within the Axon than it is within an Ethernet switch. Specifically, with the Axon's lower latency, it may suffice to account for priority when deciding which packet to drop on a buffer overflow. Currently, multicast packets are passed to the Axon's control plane, just like broadcast packets. However, the bandwidth limitations of the 32-bit, 33MHz PCI bus connecting the prototype's data and control planes make it unsuitable for exploring the performance of data dissemination applications and different approaches to multicast.

The Axon device does require a higher-level management system such as Tesseract in order to operate in a production environment. We believe that the Axon and the source route primitive in particular, provides a suitable substrate for data center networking. Future work includes adapting an existing network management system to control a network of Axons.

One functionality that we have not addressed in this work is when hosts physically move on the network. We expect that this will not happen often in a data center, but timeout mechanisms should be introduced into the design in order to ensure that the Axon network recovers properly. In general, a mechanism for eliminating source routes should be developed.

Bibliography

[1] A. Myers, T. S. E. Ng, and H. Zhang, "Rethinking the service model: Scaling Ethernet to a million nodes," in *HotNets*, November 2004.

[2] C. Kim, M. Caesar, and J. Rexford, "Floodless in SEATTLE: a scalable Ethernet architecture for large enterprises," in *Proceedings of ACM SIGCOMM*, August 2008.

[3] C. Kim and J. Rexford, "Revisiting Ethernet: plug-and-play made scalable and efficient," in *IEEE LANMAN*, June 2007.

[4] F. D. Pellegrini, D. Starobinski, M. G. Karpovsky, and L. . Levitin, "Scalable cycle-breaking algorithms for gigabit Ethernet backbones," in *Proceedings of IEEE Infocom*, March 2004.

[5] R. Perlman, "Rbridges: Transparent routing," in *Proceedings of IEEE Infocom*, March 2004.

[6] S. Ray, R. A. Guerin, and R. Sofia, "A distributed hash table based address resolution scheme for large-scale Ethernet networks," in *International Conference on Communications*, June 2007.

[7] J. Rexford, A. Greenberg, G. Hjalmtysso, D. Maltz, A. Myers, G. Xie, J. Zhan, and H. Zhang, "Network-wide decision making: Toward a wafer-thin control plane," in *HotNets*, November 2004.

[8] T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson, "SmartBridge: a scalable bridge architecture," in *Proceedings of ACM SIGCOMM*, August 2000.

[9] S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh, "Viking: A multi-spanning-tree Ethernet architecture for metropolitan area and cluster networks," in *Proceedings of IEEE Infocom*, March 2004.

[10] "IEEE standard for local and metropolitan area networks - link aggregation (IEEE std. 802.1ax)," November 2008.

[11] M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, and A. Rowstron, "Virtual ring routing: Network routing inspired by DHTs," in *Proceedings of ACM SIGCOMM*, September 2006.

[12] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, I. Stoica, and S. Shenker, "ROFL: Routing on flat labels," in *Proceedings of ACM SIGCOMM*, September 2006.

[13] B. Ford, "Unmanaged Internet protocol: Taming the edge network management crisis," in *HotNets*, November 2003.

[14] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rex-

ford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[15] H. Yan, D. A. Maltz, T. S. E. Ng, H. Gogineni, H. Zhang, and Z. Cai, "Tesseract: A 4D network control plane," in *Proceedings of the Symposium on Network Systems Design and Implementation*, April 2007.

[16] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, 2009.

[17] Myricom, "Myri-10G NICs and software," August 2008. Product brief.

[18] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo, "NetFPGA - an open platform for gigabit-rate network switching and routing," in *IEEE International Conference on Microelectronic Systems Education (MSE'2007*, June 2007.

[19] J. Katcher, "PostMark: A new file system benchmark," Tech. Rep. TR3022, Network Appliance, 1997.