Axon: A Flexible Substrate for Source-routed Ethernet

Jeffrey Shafer
University of the Pacific
Stockton, CA
jshafer@pacific.edu

Brent Stephens, Michael Foss,
Scott Rixner, Alan L. Cox
Rice University
Houston, TX
{brents, mikefoss, rixner, alc}@rice.edu

ABSTRACT

This paper introduces the *Axon*, an Ethernet-compatible device for creating large-scale datacenter networks. Axons are inexpensive, practical devices that are demonstrated using prototype hardware. Functionally, Axons replace Ethernet switches and maintain full compatibility with existing Ethernet hosts. Between themselves, however, Axons transparently use *source-routed Ethernet*. This unlocks many benefits, such as improved network scalability, performance, and flexibility.

In an Axon network, all state required to route a host's packets is placed in the *local* Axon—the Axon to which the host is directly connected. Therefore, regardless of the scale of the network, the route computation and storage needs of a single Axon device only need to scale with the demands of its locally-connected hosts. This is in stark contrast to conventional switched Ethernet, which requires routing resources proportional to the traffic that flows through the device. Scalability is also increased by eliminating the use of packet flooding for automatic location and address discovery. Further, source-routed Ethernet increases network flexibility by supporting different route selection strategies. For example, shortest-path routing could be employed, or longer paths selected to minimize congestion by balancing traffic across redundant links.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Packet Switching Networks

General Terms

Design, Performance

Keywords

Ethernet, Axon, Datacenter, Source Routing

ANCS '10, October 25-26, 2010, La Jolla, CA, USA

1. INTRODUCTION

Datacenters play a key role in supporting modern Internet services such as search engines, e-commerce, and social networking sites. In these datacenters, storage and computing resources are assembled on a massive scale. Commodity technologies are used whenever possible to provide the greatest cost efficiency. The performance and cost of the network is especially critical. As such, there has recently been a surge of interest in datacenter network architectures.

Ethernet is a ubiquitous technology that provides high bandwidth at a low cost per link. However, prior work has identified a number of shortcomings of Ethernet at the datalink layer, particularly when large networks are desired [8, 13, 14, 16, 23]. The scalability of an Ethernet network is constrained by a fundamental reliance on broadcasts and packet flooding as a location discovery mechanism. Further, Ethernet suffers from performance bottlenecks caused by its requirement of a loop-free forwarding topology.

Many solutions to these challenges have been proposed, such as using IP routers to connect many smaller Ethernet switched networks, using VLANs to overlay multiple spanning trees over a single physical topology [13], re-writing Ethernet addresses to impose a hierarchy [16, 23], and providing new network services to determine network location instead of broadcasting [8]. All of these proposed solutions, however, are limited by a reliance on the traditional Ethernet frame format that has been unchanged since its inception 30 years ago.

This paper explores the benefits to large-scale, local-area datacenter networks of modifying the traditional Ethernet frame format while keeping the underlying link layer. To this end, we have created the *Axon*, an inexpensive, practical device that replaces an Ethernet switch while maintaining full compatibility. 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. Hosts can use the DHCP and ARP protocols to obtain a network address, and locate other hosts on the network, respectively. However, among themselves, Axons use *source-routed Ethernet*, a new datalink layer protocol, and transparently rewrite traditional Ethernet packets to follow this new protocol.

Source-routed Ethernet has several advantages over traditional switched Ethernet. First, it provides flexibility when designing the physical network topology. Redundant paths can be included and used to actively carry data, as opposed to functioning purely as backups in traditional Ethernet using a spanning tree protocol.

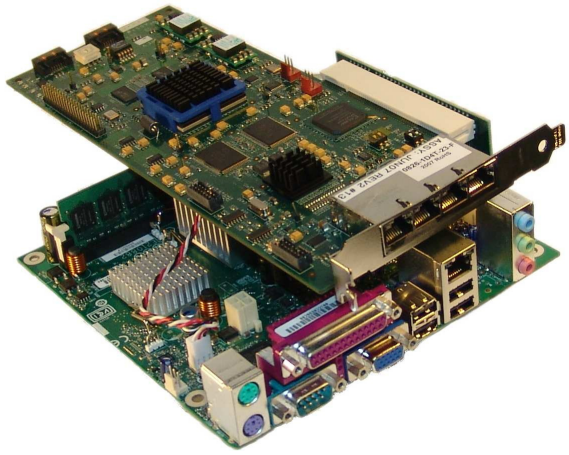


Figure 1: Axon Prototype

Second, source-routed Ethernet enables flexible routing algorithms, as packets can take any path desired. For example, shortest-path routing could be used. Or, a congestion-aware routing algorithm could send packets via alternate paths that are longer but less congested. Neither approach is supported in traditional Ethernet.

Third, source-routed Ethernet enables a flat address space for the entire local area network. This is beneficial for migrating services and virtual machines, because they can maintain the same address and open network connections. Hierarchical addressing schemes that encode location information in the network address (such as IP) do not have the same flexibility.

Finally, transparent source-routed Ethernet improves network scalability. All network and routing state needed by a host is stored 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. 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 device.

To evaluate the Axon architecture and demonstrate the practicality of source-routed Ethernet, a hardware prototype was created that is shown in Figure 1. The prototype Axon demonstrates both improved bandwidth and reduced latencies of less than 1us per hop. This compares favorably to latencies of 7–28us per hop in switched Ethernet and directly impacts the performance of latency-sensitive network applications. For example, Axon devices improve the performance of PostMark, a file server benchmark, by 20–77% for clients accessing an NFS file server over the network.

In addition to a hardware prototype, a software Axon simulator was created. The simulator demonstrates the scalability of an Axon network by showing that the average control overhead per link is less than or equal to 0.25% of the total link bandwidth, even across multiple topologies with as many as 50,000 hosts.

The remainder of this paper proceeds as follows. Section 2 describes the benefits and limitations of modern switched Ethernet. Section 3 then introduces the Axon network architecture, while Section 4 details the Axon hardware and software components. Next, Section 5 evaluates the Axon architecture using a hardware prototype and simulator. Finally, Section 6 discusses related work and Section 7 concludes the paper.

2. BACKGROUND

Switched Ethernet is deployed in a variety of environments, including home, office, campus, and datacenter 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. Interface 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. 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, 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.

Switched Ethernet does permit the existence of redundant links, but only to heal the network after a failure. Redundant links 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. 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 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.

The Ethernet packet format 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 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. 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.

Ethernet networks rely on broadcast protocols for key functions. The Dynamic Host Configuration Protocol (DHCP) allows hosts to join a network without any manual configuration, and the Address Resolution Protocol (ARP) allows a sending IP host to discover the receiving IP host's Ethernet interface address (MAC address) dynamically. Both protocols function by broadcasting a request to

all hosts on the Ethernet network, which can either ignore the request or respond with a unicast message containing the desired network configuration (for DHCP) or target MAC address (for ARP).

3. AXON NETWORK OVERVIEW

The Axon device is a direct replacement for Ethernet switches that can be used to create a highly-efficient local-area network. Hosts[1] can be directly connected to an Axon without modification. Axons transparently use a novel source-routed Ethernet protocol 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 Axon Packet Format

Axon packets are transmitted over conventional Ethernet cables using standard Ethernet media access control and physical transceivers. However, Axon packets have their own header format, as shown in Figure 2. The Axon header includes a type, length, and source-routing information. Although Axon packets do not have an Ethernet header at the beginning of the packet, standard Ethernet physical transceivers (operating at the lower link layer) 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. In order to ensure that subsequent fields start on an even byte boundary, 4 bits of padding are added after the reverse hop count that can be used for future expansion. 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. In the prototype design, each hop in the path is a byte-sized value that indicates an output port number. A hop with the value 0xFF indicates the packet should be forwarded to the control plane, rather than an Ethernet output port. Production Axons could be designed with larger numbers of ports by increasing the width of the hop field beyond 8 bits. Further, different encoding schemes could be employed to support variable-length hop fields for Axons of different sizes, or to provide extra checksumming support (beyond the standard Ethernet CRC) to detect data corruption in transit and drop or repair the packet as desired.

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 *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. Further, Ethernet MACs support *promiscuous mode*, which disables checking of the destination MAC address, thereby allowing the first 6 bytes to be used by the Axon header.

From the perspective of the Ethernet MACs and PHYs, an Axon packet is an oversized Ethernet frame. As such, 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.

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. This overhead was analytically calculated for source-routes containing 1, 10, and 100 hops. The overhead of source-routing is negligible for short routes with 1–10 hops. For longer 100 hop routes, the maximum bandwidth available to the application only decreases by 3% at the largest packet size.

3.2 Axon Packet Routing

An Axon can determine the output port for an Axon packet after receiving the seventh byte, which contains the next forward hop in the source-route. At this point, the paths in the Axon header can be updated for the subsequent hop at the same time that the packet is forwarded to the appropriate output port.

Axons perform cut-through routing at the packet level. This is necessary due to the use of Ethernet as the link layer. 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. Unlike other interconnection networks, Ethernet networks do not guarantee packet delivery. Higher level network protocols are used to throttle the transmission rate and retransmit. Thus, when buffers are full, the Axon can safely drop packets. However, flow control can be applied at the packet level to reduce/eliminate packet loss.

3.3 Compatibility

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 3.2. 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 and responds after coordination with a network controller described later. 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

[1]In this paper, the term *host* means any non-Axon Ethernet device, including a standard Ethernet switch, an IP router, or a computer.

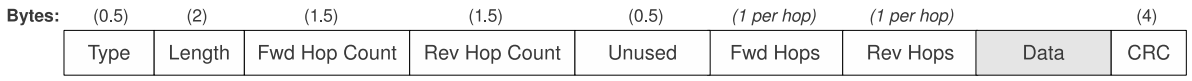


Figure 2: Axon Packet Format

connected to the target host in order to set up routes to allow communication in both directions between the hosts.

A content addressable memory (CAM) is provided in every Axon to store a mapping between MAC address and source-route. This CAM can be similar in design to those used in conventional Ethernet switches. A larger, more complicated ternary CAM as used in IP routers to do partial address matches is not necessary. When a host sends an Ethernet packet to a target MAC address, the Axon will transparently use the target MAC address to lookup the appropriate source-route in its local route memory. The Axon will then encapsulate the Ethernet packet inside of an Axon packet using that source-route, and transmit it to the next Axon along the route.

When the next forward hop of an Axon packet indicates a port which is connected to 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.

Axons can be incrementally deployed into an existing Ethernet network. An Axon that is connected to a switch simply behaves as if it were connected to a host. Because the Axon prototype maintains routes on a per-port basis, all hosts connected to the switch (which is attached to a single Axon port) use the same set of routes, retrieved by target MAC address. A production Axon could extend this design for security or performance reasons, however, and retrieve source-routes based on a target/source MAC address tuple. This way, each host hidden behind a switch could still have its own route to a given target.

3.4 Axon Network Controller

The route selection problem for arbitrary network topologies has been solved using techniques such as a distributed hash table [8, 20] or a central controller with full topology knowledge [4, 25]. While a distributed hash table is more scalable and resilient to failure, a central controller can have a simpler design. Further, previous work has shown that the controller can both scale to support tens of thousands of hosts and be replicated across multiple controllers to eliminate a single point of failure for the network.

The prototype Axon network uses a central controller because of its simplicity and the extra flexibility it allows in route selection, but this is not fundamental to an Axon network. The controller must be able to learn the network topology, and this is accomplished through a discovery protocol described in Section 4.2.1.

4. AXON DEVICE ARCHITECTURE

Each Axon device includes both a hardware switching fabric—the *data plane*—and a processing element to perform control operations for locally connected hosts—the *control plane*. 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 provides an overview of the data and control plane architecture.

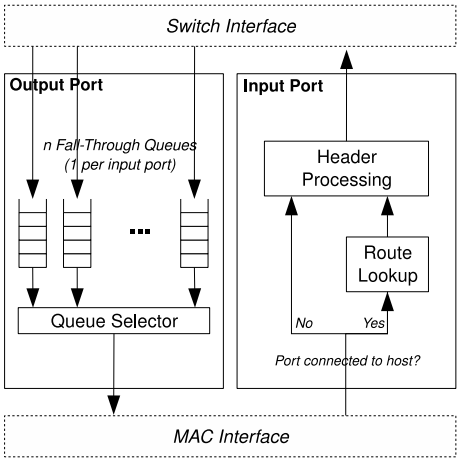


Figure 3: Axon Ethernet Port in Data Plane

4.1 Axon Data Plane

The Axon hardware data plane contains input ports and output ports (1 per physical Ethernet port) that are interconnected by switch capable of cut-through packet forwarding. A subset of the full Axon design (showing a single Ethernet port) is shown in Figure 3. Of these components, only the input port provides new functionality compared to traditional designs, and is described in more detail here.

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 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.

4.1.1 Input Port

When a packet is received over an Ethernet link, it is first processed by the corresponding input port. As Figure 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.

Host connected to an Axon always send Ethernet frames. These frames cannot be routed through the Axon network until they are first encapsulated in an Axon packet. The *route lookup* module searches a CAM using the destination MAC address of the received packet. If a match is found, an index is returned to a specific route stored in that port's route memory.

Each Ethernet port has its own associated 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. By providing this memory on a per-Ethernet-port basis, network security policies can be supported that provide isolation between hosts.

The route lookup module uses the destination MAC address of the received packet to retrieve an Axon header from the route memory. If the MAC address matches an existing route, then the retrieved header is prepended to the Ethernet frame, creating an Axon packet. If the destination MAC address does not match an existing route, then a default Axon header is prepended to the frame with a single forward hop that targets the local software control plane. A lookup failure which causes a packet to be forwarded to the control plane is not necessarily an error, as broadcast traffic is sent to the control plane by this same mechanism.

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 transformed by the route lookup module are next processed by the header processing module. 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. Otherwise, the bare Ethernet frame can be immediately sent over the switch to the appropriate output port.

4.2 Axon Control Plane

The primary responsibility of the Axon control plane is to setup routes for the data plane. The control plane will intercept and process Ethernet broadcast packets transmitted by local hosts, such as DHCP and ARP. For flexibility, the control plane is implemented in software running on a low-power embedded processor. The control plane has a port into the switching fabric of the data plane. This allows the control plane to receive packets from and inject packets into the data plane seamlessly. The data plane forwards all Ethernet packets from a host with a destination address that cannot be matched to a valid source-route, including all broadcast traffic.

4.2.1 Discovery

In order to install routes, each Axon needs to find the controller and the controller needs to learn the network topology. This is accomplished through the fully automated *Axon-Discovery* protocol, which requires no manual intervention. Three different types of packets are used: *Axon Hello*, *Heartbeat*, and *Controller Hello*.

Both Axons and the controller periodically send Axon Hello requests out of all uninitialized ports. Axon Hellos contain the unique ID for the Axon and are encrypted with a pre-shared key. If an Axon authenticates the request, it sends back a reply, which completes the handshake.

After two Axons or an Axon and the controller exchange Hellos, they periodically send Heartbeat messages to each other to monitor link state and find the controller. Heartbeat messages contain the sender's shortest verified path to the controller. If a Heartbeat is not received by an Axon after twice the send interval, the link is assumed to be down, and the port is set to an uninitialized state.

Controller Hellos are sent by Axons any time a port changes state or a Heartbeat with a shorter path is received. These messages contain the full port state information for the sending Axon. To ensure that the controller receives the new topology information, Controller Hellos are sent out periodically until a valid reply is received from the controller.

4.2.2 ARP

Hosts use the ARP protocol to determine the location of other hosts on the Ethernet network. As with all broadcast Ethernet traffic, ARP requests are forwarded to the control plane. The control plane then uses the *Axon-ARP* protocol to satisfy the request. The Axon-ARP protocol involves the controller and 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 with the controller in order to setup routes in both directions between the source and target hosts. When routes become invalid due to failure, the Axon local to the failure notifies the controller, which reroutes all affected flows.

Upon receiving an ARP request, the source Axon first reserves a CAM entry and space in the input port route memory corresponding to the source host. This space will be used to hold an Axon header containing a route from the source host to the target host. However, the source Axon does not yet have an Axon header for the flow nor the target Ethernet addresses to place in the CAM, so it cannot yet update the route memory and mark it valid for use. To obtain the missing information, the source Axon sends an *Axon-ARP request* to the controller. This request includes the original ARP request and the Axon port to which the source host is connected.

When the controller receives the Axon-ARP request, it computes a path from the source Axon to the target host, the target Axon to the source host, and a path from the controller to the target Axon. In order to find these paths, the controller must know the topology of the network, which is the job of the Axon-Discovery protocol, and the mapping of hosts to Axons, which the controller learns from previous ARP and DHCP requests. Hosts with static IPs that do

not send DHCP requests are required to send a gratuitous ARP to initialize, as is commonly done when migrating virtual machines to a different port on a conventional switched Ethernet network. If any of the route computations fail, the request is dropped or the source host's port is blocked if the controller decides the source host is misbehaving. If the path computation succeeds, the controller sends the Axon-ARP request to the target Axon with the routes between the Axons attached.

When the target Axon receives an Axon-ARP request, it also reserves a CAM entry and space in the input port route memory corresponding to the target host. This memory will hold an Axon header containing the route back from the target host to the source host. The target Axon then sends the ARP request in the Axon-ARP request to the target host. When the target host sends back an ARP reply, the data plane will forward it to the control plane, since the reserved route memory is not yet valid.

When the target Axon receives the ARP reply from the target host, it can then install a route to the source host. The Axon header will include the route back to the source host, which is included in the Axon-ARP request, and a CAM entry will be installed. After installing a valid route, the target Axon can respond to the control plane of the source Axon with an *Axon-ARP reply*.

When the source Axon receives the Axon-ARP reply, it can complete the route setup for the source. 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. 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, it will arrive correctly at the source. (This is unlikely, however, as it is the source host that is initiating communication.) If the source host then responds before the source Axon has validated the route, the hardware will simply forward that packet to the control plane. The control plane can either queue the packet until the route is available, or simply drop it, as this should affect only a small number of packets, and higher level network protocols should retransmit if needed.

The Axon-ARP protocol is only necessary for new ARP requests. When an ARP request arrives at an Axon for an existing flow, the Axon can send a cached copy of the ARP reply directly to the source host. To avoid stale flows, each Axon sends a periodic gratuitous ARP to every host connected to it. If an Axon does not receive a timely ARP reply, it destroys all flows for the host and notifies the controller. The controller is responsible for notifying all of the target Axons involved in prior communication with this host, so that they can also delete the now-invalid routes in their local route memories.

4.2.3 DHCP

DHCP enables a host to dynamically discover its IP address. Axons intercept DHCP messages to provide transparent compatibility with existing hosts. Here, the initial DHCP discovery message is broadcast by the host. As with all Ethernet broadcast traffic, DHCP discovery messages will be forwarded to the Axon control plane. The control plane forwards the DHCP traffic to the controller, which updates the topology information and acts as a conventional DHCP server. 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 controller 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.

5. EVALUATION

To evaluate the Axon design and the benefits of source-routed Ethernet, a variety of experiments were conducted using both a hardware prototype and a custom simulator.

Figure 1 shows the prototype Axon device used for evaluation. In the prototype, the hardware data plane is implemented on Stanford's NetFPGA platform [11] 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 4 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 production Axon implementation would have more Ethernet ports, the NetFPGA effectively limits the prototype to 4, which is still sufficient to demonstrate the viability of the Axon device.

The control plane is implemented as a user-level application running on a standard x86 Linux kernel. The bandwidth between the control and data planes is limited by the 32-bit PCI bus in the prototype. In practice, there is more than enough bandwidth for the address and location discovery tasks currently performed by the control plane.

Creating an experimental prototype yielded several benefits. First, it provided a strong understanding of the low hardware complexity of source-routed Ethernet. The Axon was implemented on an old FPGA family with limited logic resources, and was still able to operate at full gigabit wire speeds. Second, a prototype allowed hardware performance metrics such as forwarding latency to be experimentally measured, instead of estimated. Third, a prototype allowed the design to be validated, particularly with regards to compatibility. With the prototype, full compatibility was demonstrated with a variety of unmodified hardware and software platforms, including computers running Windows, Mac OS X, FreeBSD, and Linux, as well as a Cisco IP router. The hosts were able to communicate at full wire speed over the Axon network with transparent source-routing.

In addition to the Axon prototype, an Axon simulator was also implemented. It contains three main components: the Axon control plane software, a switchboard process, and the host software. The control plane software is identical to the software used on the prototype Axon but interfaces with the switchboard process instead of the physical device. One

instance of the control plane is created for each simulated Axon, but only one switchboard is created. The switchboard simulates all of the links and Axon hardware in the network and is responsible for forwarding Axon packets by their source routes, attaching headers to host packets, and keeping statistics. The host software can either be a virtual machine or an ARP generator. Arbitrary traffic can be forwarded through the simulator between virtual machines, while the ARP generator is used as a lower overhead means to simulate host discovery. Control plane software for PortLand [16] was also implemented in the Axon simulator to provide a reference.

The simulated network topology is read from a static configuration file. Five different topologies were simulated: torus, fat tree, *flattened butterfly* (FBFLY) [10], random-low, and random-high. Torus is dimensioned by the ring size and the number of rings. In torus, each Axon is connected to exactly four other Axons, two in each dimension. Fat tree and FBFLY are built as described in [16] and [10], respectively, and support non-blocking communication. The random topologies are generated by visiting each Axon and creating n links from the visited Axon to other random Axons, for an average degree of *2n*. Random-low uses an *average* degree of 4 to match the torus topology and random-high uses a degree to match the FBLY topology for a given size network. Each Axon in torus and random-low are connected to 10 hosts. The topologies use as few Axons as required for the simulated number of hosts, and the hosts are evenly distributed across the Axons.

The ARP generator uses two different ARP patterns. The first ARP pattern is derived from packet traces collected from the Lawrence Berkeley National Lab (LBNL) [17], after scanning traffic has been filtered out. The scanning traffic was omitted because the Axon controller has a global view of ARPs and could detect and disable ports engaged in scanning. Further, the scanning traffic in the trace is anonymized separately from other traffic, so a host engaged in scanning appears in the traces as two separate hosts. The LBNL traces cover at most 17,000 hosts, so simulations with more hosts use a random ARP pattern. Each host sends a set number of ARPs to random hosts. For each LBNL topology, 5 different random mappings of hosts to Axons were simulated and the results were averaged across the runs.

The simulator has no knowledge of time, but it can be used to derive bandwidth results by normalizing total byte overhead to seconds. For example, if the total byte overhead of each host sending 5 ARPs is known from the simulator, then it is trivial to calculate the total byte overhead per second of each host sending 5 ARPs per second. The control traffic bandwidth is obtained from the simulator by normalizing the ARP overhead to a constant rate and configuring the Axons to send a set number of Heartbeats after the network has been allowed to initialize.

Creating a simulator provided several benefits. It allowed networks to be evaluated with a large number of hosts and Axons in order to determine scalability. This was not possible using real Axons, because only a limited number of prototype devices and network hosts were available. Further, it allowed a variety of network topologies to be evaluated.

In this section, several facets of the Axon architecture are evaluated, including its ability to increase bandwidth due to redundant routes and the lower latency provided by source-routed Ethernet. Further, the necessary size for an Axon route memory is quantified. Finally, the scalability of the Axon control architecture is evaluated, along with the flexibility of the Axon substrate to accommodate new routing algorithms.

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

Table 1: Individual and Aggregate Bandwidth for Line and Ring Topologies, Measured in Mb/s

5.1 Higher Bandwidth

One major advantage of source-routed Ethernet over conventional Ethernet using a spanning tree protocol is that source-routing allows for arbitrary topologies that include cycles. The addition of these redundant links can significantly increase available bandwidth. To demonstrate this, three prototype Axons were assembled in *line* and *ring* topologies. The line topology represents the simplest Ethernet network that is restricted by a spanning tree, and the ring topology is the simplest network with a cycle and shows how the Axon design can exploit the increased bandwidth. In both configurations, two hosts were connected to each Axon, for a total of 6 hosts per network. Each host transmits a single flow to another host attached to a different Axon. All 6 flows are equally dispersed across the network.

Table 1 shows the bandwidths of the six flows between hosts for each topology for the UDP and TCP protocols. These application-level bandwidths were measured using the *netperf* microbenchmark. In the line topology, there is network contention over the links connecting the three Axons because each of the six hosts produces a 1 Gb/s flow, but there are only 4Gb/s of cross-Axon bandwidth available in the network (accounting for the bidirectional rate). In the ring topology, the achieved bandwidth improves significantly. 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 bandwidth is more pronounced because there is traffic in exactly one direction per flow. TCP requires acknowledgement packets to be sent in the reverse path for a given flow, causing occasional packet loss and subsequent TCP bandwidth throttling. As shown here, a network can significantly benefit from using the redundant links that would otherwise be disabled by the spanning tree protocol.

5.2 Lower Latency

In addition to increasing available bandwidth, the Axon design also achieves lower forwarding latency than traditional Ethernet switches for two key reasons. First, the Axon design incorporates cut-through forwarding. Although Eth-

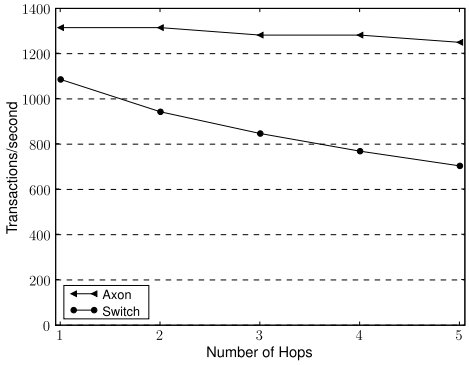


Figure 4: PostMark Performance

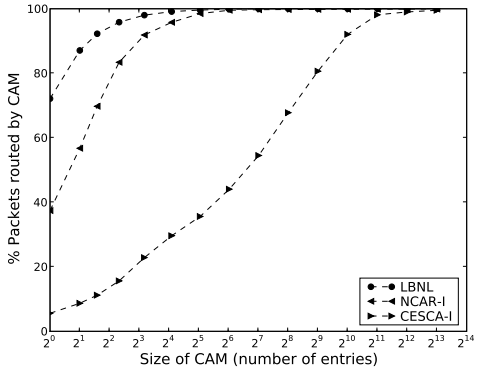


Figure 5: CAM Effectiveness At Varying Sizes

ernet switches could also employ this technique, most implement a store-and-forward design. Second, source-routing incurs fewer time-consuming routing lookups than traditional switched Ethernet, as only the first Axon along the path needs to retrieve and append the route header. Subsequent Axons simply read the output port directly from the packet header as it is received from the network link.

Forwarding latency for the Axon prototype was measured directly from the gate-level design. Each Axon adds 520ns of latency to transit a packet from an Axon-port to an Axon-port, or 720ns from a host-port to an Axon-port (due to the header lookup and packet rewriting). This latency on the prototype can be considered an upper bound. Production devices would be faster for 3 reasons: the prototype uses an older Virtex-II Pro FPGA, prototype CAM and FIFO structures are implemented in (slower) FPGA logic, and the prototype includes an Ethernet media access controller that adds 224ns of latency but provides no useful functionality for the Axon design. For comparison, a store-and-forward gigabit Ethernet switch typically adds latency of 7–28us per hop, depending on packet size. (Emerging 10Gbps Ethernet switches have latencies measured in hundreds of nanoseconds, but these devices cannot be fairly compared against the Axon prototype, which was restricted to a 1Gbps line speed and FPGA technology dating from 2004.)

The latency benefits of the Axon translate into performance improvements for latency-sensitive applications such as the PostMark file system benchmark. PostMark approximates a large Internet e-mail server that maintains a large pool of continually changing small files [7]. Figure 4 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 write accesses on a random set of files of different sizes. The graph shows that Axons outperform Ethernet switches for 1–5 devices (as limited by the number of available prototype Axons), but the trend is clear as the number of network hops 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 an Axon network.

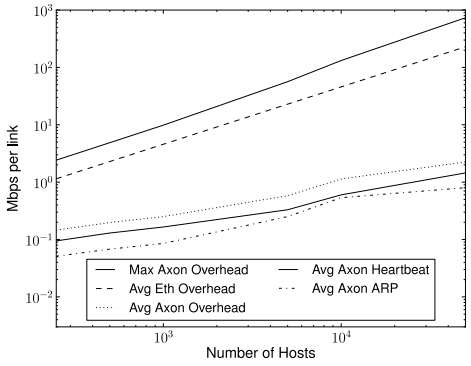
5.3 Route Memory

To determine the size of the per-port CAM needed for an Axon, a variety of gigabit router traces from NLANR were evaluated for two different configurations: traffic entering/exiting the network, and traffic internal to the network. 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 interconnect 22 subnets at Lawrence Berkeley National Labs, and is the closest approximation to a single large datacenter network.

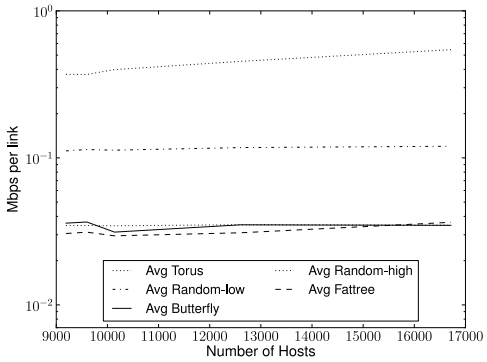
Each trace was analyzed to find the re-use distance (in packets) between messages to the same destination IP address. The distance corresponds to the number of entries in an Axon's CAM that would be needed to support all active flows during the trace. The effectiveness (*e.g.,* hit rate) of the CAM at a range of sizes is shown in Figure 5. Here, a CAM of 4000 entries is sufficient for CESCA trace, which is analogous to an Axon at the network edge that is exposed to all flows entering and exiting the network. Smaller CAMs are sufficient for other traces that monitor internal network traffic. For comparison, even low-end managed Ethernet switches used in datacenters typically contain CAMs with 8k or more entries. Further, in an Axon network, the CAM size is only relevant for Axons directly attached to hosts. Axons in the network core do not need to perform route lookups, a stark contrast to a traditional switched Ethernet network, where core switches need forwarding entries for every network host whose packets transit that network hop.

5.4 Lower Control Overhead

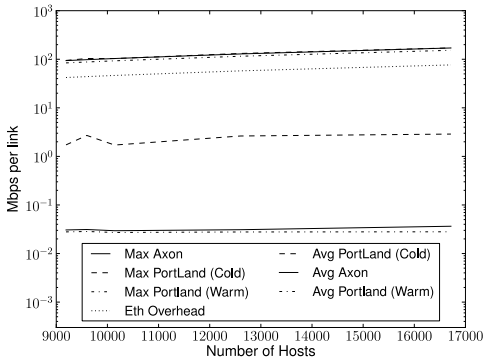
The control overhead of a network is one factor limiting scalability. To determine the control overhead on an Axon network, the simulator was used to evaluate the torus, fat tree, FBFLY, random-low, and random-high topologies on networks with up to 50,000 hosts and 5,000 Axons. The control overhead on an Axon network is incurred from the Axon-ARP protocol and the Axon-Discovery protocol. This is in contrast to a switched Ethernet network, where the control overhead is dominated by ARP traffic.



(a) Torus Topology



(b) Average LBNL Overhead



(c) PortLand vs. Axon Overhead

Figure 6: Per-Link Control Overhead (Given 10 ARP/sec/host and 10 Hellos/sec/Axon)

The overhead of network discovery by the central controller during initialization is minimal. On the largest topology with 5,000 Axons and 10,000 links between Axons, the average link carried 40Kbit of traffic and the worse-case link (attached to the central controller) carried 108Mbit of traffic. The overhead until convergence on a PortLand network is also minimal, but is between 50–100% larger than on an Axon network.

The overhead of an Axon network during normal network operation is presented in Figure 6(a) for a torus configuration. This figure shows the control overheads in Mbps for both Axon and Ethernet networks as the number of attached hosts (and thus the network size) increases.

The simulator is configured to provide a constant rate of 10 new ARP requests per host per second and a Heartbeat rate of 10 per second. The ARP rate was chosen based on data provided in [4], which observed a peak new ARP per host per second rate of 0.5 on a 22,000 host enterprise network. A datacenter network might have a higher ARP rate, so an average rate of 10 ARPs per second was chosen. This is a conservative choice, especially considering that it only represents ARPs caused by new communication flows that require a new route generated by the controller. In contrast, a traditional Ethernet network also experiences extra broadcast ARP traffic caused by timeouts of old entries in the ARP cache. In an Axon network, however, such ARPs are intercepted by the local Axon and answered using cached data. Thus, the rate of 10 ARPs per second represents only completely new communication flows in the network, not the maintenance of existing flows.

In Figure 6(a), *Avg Axon Heartbeat* represents the topology discovery messages periodically sent by Axons to the central controller, and *Avg Axon ARP* represents the communication between Axon and controller to generate a new route. *Avg Axon Overhead* is simply the sum of these two components, and represents the average overhead incurred by any link in the network. Max overhead represents the same sum of heartbeat and ARP overhead, but instead of an average across all links, it shows only the most congested link in the network, which is attached to the controller. Finally, Ethernet overhead is also provided for comparison. Because ARPs are broadcast in traditional Ethernet, the total ARP traffic from all the hosts is a lower bound on the overhead of each link.

The controller receives and sends a packet for every ARP request, and each packet includes route information. Thus, it is expected that the maximum link overhead on an Axon network would be more than twice that of an Ethernet network, and this is confirmed by the results. However, because ARPs are not broadcast, the average Axon link experiences a much lighter load than the average Ethernet link.

At the same rate, the overhead of ARP traffic is greater than that of discovery traffic on every topology other than torus. The Heartbeat overhead grows linearly with the number of links and average distance from the controller, which were both large on torus. The ARP overhead is typically greater because each Heartbeat traverses only one link, but each Axon-ARP has to reach the central controller, which is on average a longer distance.

A comparison of overheads on different topologies with the LBNL ARP traces is presented in Figure 6(b). Torus has the highest average distance from the controller, and thus has the highest average ARP and Heartbeat overheads because the packets are larger to accommodate a longer source-route. Random-low has a shorter average distance to the controller, and this is reflected in the overheads.

A comparison of the total overheads of both an Axon network and a PortLand network on the same topologies is presented in Figure 6(c). In PortLand, the controller broadcasts ARPs that it cannot resolve, unlike an Axon network. *PortLand (Cold)* represents simulations where the none of the hosts are initialized, whereas *PortLand (Warm)*, represents a system where all of the hosts are initialized before any ARPs are sent. The Portland (Warm) overheads are slightly

Topology	Torus		Random	
	SP	SPAIN	SP	SPAIN
Flows Per Host	5	5	5	5
Avg Route Length	28.469	28.677	6.366	6.43
Avg Flows/Link	1511	1524	324	330
Std Dev Flows/Link	1310	1216	112	45
Max Flows/Link	4300	2836	1026	494
Avg Mb/Link Failure	938	948	145	148
Max Mb/Link Failure	2670	1765	460	222

Table 2: Flow Distribution for Shortest Path (SP) and SPAIN Routing

lower than the Axon overheads because the Axon packets include source-routes, but the average Portland (Cold) overhead is over an order of magnitude larger than the average Axon overhead, showing the overhead and scalability benefits of never broadcasting packets.

5.5 Flexible Route Selection

In the previous results, the controller calculated routes using a shortest-path algorithm. However, source-routed Ethernet is flexible and can accommodate routes generated by other algorithms. As an example, the controller was modified to imitate the proposed SPAIN network architecture [13]. Here, a greedy weighted shortest path algorithm is used to calculate routes. The weight of a link is the number of flows on the link multiplied by a large number, which will distribute the flows more evenly across the links, even at the expense of taking a slightly longer path. A comparison of SPAIN-style routing versus shortest path uniform link weight routing demonstrates the effectiveness of the Axon as a substrate for intelligent and arbitrary route selection with no modifications to the network hardware.

The controller is not limited to SPAIN-style routing. SPAIN is used as an example, but the controller can arbitrarily choose routes and could also implement either of the flow scheduling algorithms presented in Hedera [1].

Table 2 shows that the benefits of improved route distribution with SPAIN-style routing for torus and random-low with 10,000 hosts. Although the average number of flows per link only changed slightly with SPAIN, the network-wide balance improved significantly, as evidenced by the standard deviation of the number of flows per link. This was most prominent on random-low, as shortest-path routing was already reasonably effective on a torus network due to its uniform structure.

The overhead per link failure in Table 2 is obtained directly from the number of flows per link. In the case of link failure, one packet must be sent from the controller to the source Axon for every flow that traverses the failed link. A routing algorithm like SPAIN that balances flows across all network links reduces the number of links that must be re-routed if any one link fails.

6. RELATED WORK

It is well known that switched Ethernet does not scale well to large numbers of hosts [14]. 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 campus or datacenter 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.

Others have proposed architectural modifications to Ethernet switches to enable large-scale networks [6, 8, 9, 13, 14, 18, 19, 20, 21, 22, 23, 24]. Techniques used include overlaying multiple spanning trees to exploit redundant links, using a distributed hash table, and rewriting MAC addresses to impose a hierarchy on the address space. In general, these approaches 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 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.

In contrast to these previous approaches, source-routed Ethernet moves the burden of route determination to the source of the traffic. An Axon only need know about routes to destinations with which its locally connected hosts are communicating. An Axon does not need to have any routing or topology information for traffic that it is forwarding.

Identity-based routing can also be used to improve the scalability of local area networks [2, 3, 5]. These prior techniques, however, do not route directly to the destination. Instead, they determine paths using a hash of the destination identifier. The Axon network is not constrained to use any particular path for a traffic flow. When necessary, shortest-path routing can be used. In other instances, different routes could be chosen; in order to take advantage of redundant links, for instance.

Others have recognized that there are many redundant links that Ethernet switches never use. Many have proposed new ways to manipulate the spanning tree [18, 24, 13]. Axons, in contrast, exploit redundant links by eliminating broadcast and using source-routing. For example, source-routes can trivially be created for different traffic flows across the redundant links.

OpenFlow switches can be programmed to identify flows, process packets, and forward packets in a flexible manner [12]. 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. In an Axon network, however, network state for a host's traffic flows is only needed at the local Axon, and not at all Axons in the network.

Myricom's 10Gbps network devices, like Axons, also use commodity ethernet physical interfaces [15]. 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 Eth-

ernet 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.

Of all the related work, the Axon is evaluated on the largest topology, with a total of 50,000 hosts and 5,000 Axons. The next largest simulation is done in SPAIN [23], which simulates a network with 27,000 hosts and 2,880 switches. Evaluation on such a large topology effectively demonstrates that an Axon network is at least as scalable as alternate networking schemes.

7. CONCLUSION

The Axon is an inexpensive, practical device that replaces an Ethernet switch. Axons are compatible with existing Ethernet hosts, allowing them to benefit from the performance advantages of source-routing while maintaining the plug-and-play simplicity of traditional Ethernet. By employing source-routed Ethernet, a new datalink layer protocol, Axon networks provide greater scalability than conventional switched Ethernet networks by pushing state to the edge of the network and eliminating broadcast messages. The control overhead on an Axon network with 50,000 hosts is only 0.25% of the total link bandwidth. By increasing Ethernet's scalability, large local-area networks can be created with a flat address space, facilitating virtual machine migration in the datacenter.

An Axon network is more flexible than a traditional Ethernet network and supports both arbitrary topologies and arbitrary routes. Source routing enables the use of redundant paths that can increase available network bandwidth compared with traditional Ethernet. Further, this datalink layer protocol provides a substrate that is compatible with many recent innovations in large-scale network design. For example, the SPAIN routing algorithm can be implemented on an Axon network without using the VLAN tricks required with traditional Ethernet, and PortLand and Hedera can be implemented without limiting topology requirements.

The Axon hardware prototype demonstrates full compatibility with unmodified hosts and improved performance. Axons can saturate 1Gbps Ethernet links and 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.

8. REFERENCES

[1] M. Al-fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *Proc. of Networked Systems Design and Implementation (NSDI) Symposium*, 2010.

[2] 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*, 2006.

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

[4] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker. Ethane: taking control of the enterprise. In *Proceedings of ACM SIGCOMM*, 2007.

[5] B. Ford. Unmanaged Internet protocol: Taming the edge network management crisis. In *HotNets*, Nov. 2003.

[6] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *Proceedings of ACM SIGCOMM*, 2009.

[7] J. Katcher. PostMark: A new file system benchmark. Technical Report TR3022, Network Appliance, 1999.

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

[9] C. Kim and J. Rexford. Revisiting Ethernet: plug-and-play made scalable and efficient. In *IEEE LANMAN*, 2007.

[10] J. Kim and W. J. Dally. Flattened butterfly: A cost-efficient topology for high-radix networks. In *in Proc. of the Intl. Symp. on Computer Architecture*, 2007.

[11] 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*, 2007.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, 2008.

[13] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul. SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. In *Proceedings of Networked Systems Design and Implementation*, 2010.

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

[15] Myricom. Myri-10G NICs and software, Aug. 2008. Product brief.

[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. In *Proceedings of ACM SIGCOMM*, 2009.

[17] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A first look at modern enterprise traffic. In *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 2–2, Berkeley, CA, USA, 2005. USENIX Association.

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

[19] R. Perlman. Rbridges: Transparent routing. In *Proceedings of IEEE Infocom*, Mar. 2004.

[20] 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*, 2007.

[21] 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*, 2004.

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

[23] M. Scott, A. Moore, and J. Crowcroft. Addressing the scalability of Ethernet with MOOSE. In *Proceedings of DC CAVES Workshop*, 2009.

[24] 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*, 2004.

[25] 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 Network Systems Design and Implementation*, 2007.