

Rethinking the Service Model: Scaling Ethernet to a Million Nodes

Andy Myers[†], T. S. Eugene Ng[‡], Hui Zhang[†]
[†]Carnegie Mellon University [‡]Rice University

Abstract—Ethernet has been a cornerstone networking technology for over 30 years. During this time, Ethernet has been extended from a shared-channel broadcast network to include support for sophisticated packet switching. Its plug-and-play setup, easy management, and self-configuration capability are the keys that make it compelling for enterprise applications. Looking ahead at enterprise networking requirements in the coming years, we examine the relevance and feasibility of scaling Ethernet to one million end systems. Unfortunately, Ethernet technologies today have neither the scalability nor the reliability needed to achieve this goal. We take the position that the fundamental problem lies in Ethernet’s outdated service model that it inherited from the original broadcast network design. This paper presents arguments to support our position and proposes changing Ethernet’s service model by replacing broadcast and station location learning with a new layer 2 directory service.

I. INTRODUCTION

Today, very large enterprise networks are often built using layer 3, i.e., IP, technologies. However, Ethernet being a layer 2 technology has several advantages that make it a highly attractive alternative. First of all, Ethernet is truly plug-and-play and requires minimal management. In contrast, IP networking requires subnets to be created, routers to be configured, and address assignment to be managed – no easy tasks in a large enterprise. Secondly, many layer 3 enterprise network services such as IPX and AppleTalk persist in the enterprise, coexisting with IP. An enterprise-wide Ethernet would greatly simplify the operation of these layer 3 services. Thirdly, Ethernet equipment is extremely cost effective. For example, a recent price quote revealed that Cisco’s 10 Gbps Ethernet line card sells for one third as much as Cisco’s 2.5 Gbps Resilient Packet Ring (RPR) line card and contains twice as many ports. Finally, Ethernets are already ubiquitous in the enterprise environment. Growing existing Ethernets into a multi-site enterprise network is a more natural and less complex evolutionary path than alternatives such as building a layer 3 IP VPN. Already many service providers [1] [3] are offering metropolitan-area and wide-area layer 2 Ethernet VPN connectivity to support this emerging business need. Looking ahead at enterprise networking requirements in the coming years, we ask whether Ethernet can be scaled to one million end systems.

Ethernet’s non-hierarchical layer 2 MAC addressing is often blamed as its scalability bottleneck because its flat addressing

scheme makes aggregation in forwarding tables essentially impossible. While this may have been the case in the past, improvements in silicon technologies have removed flat addressing as the main obstacle. Bridges that can handle more than 500,000 entries already ship today [2].

We argue that the fundamental problem limiting Ethernet’s scale is in fact its outdated service model. To make our position clear, it is helpful to briefly review the history of Ethernet. Ethernet as invented in 1973 was a shared-channel broadcast network technology. The service model was therefore extremely simple: hosts could be attached and re-attached at any location on the network; no manual configuration was required; and any host could reach all other hosts on the network with a single broadcast message. Over the years, as Ethernet has been almost completely transformed, this service model has remained remarkably unchanged. It is the need to support broadcast as a first-class service in today’s switched environment that plagues Ethernet’s scalability and reliability.

The broadcast service is *essential* in the Ethernet service model. Since end system locations are not explicitly known in this service model, in normal communication, packets addressed to a destination system that has not spoken *must* be sent via broadcast or flooded throughout the network in order to reach the destination system. This is the normal behavior of a shared-channel network but is extremely dangerous in a switched network. Any loop in the network can create an exponentially increasing number of duplicate packets from a single broadcast packet.

Thus, the primary concern in a switched Ethernet is to support a loop-free packet broadcast service. The existing protocol that handles this task is the Rapid Spanning Tree Protocol (RSTP) [11], which computes a spanning tree forwarding topology to ensure loop freedom. Unfortunately, based on our analysis (see Section II), RSTP is not scalable and cannot recover from bridge failure quickly. Note that ensuring loop-freedom has also been a primary concern in much research aimed at improving Ethernet’s scalability [14] [15] [18]. Ultimately, ensuring that a network always remains loop-free is a hard problem.

The manner in which the broadcast service is being used by higher layer protocols and applications makes the problem even worse. Today, many protocols such as ARP [17] and DHCP [6] liberally use the broadcast service as a discovery or bootstrapping mechanism. For instance, in ARP, to map an IP address onto an Ethernet MAC address, a query message is broadcast throughout the network in order to reach the end system with the IP address of interest. While this approach is simple and highly convenient, flooding the entire network when the net-

This research was sponsored by the NSF under ITR Awards ANI-0085920 and ANI-0331653 and by the Texas Advanced Research Program under grant No.003604-0078-2003. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF, the state of Texas, or the U.S. government.

work has one million end systems is clearly unscalable.

In summary, the need to support broadcast as a first-class service plagues Ethernet’s scalability and reliability. Moreover, giving end systems the capability to actively flood the entire network invites unscalable protocol designs and is highly questionable from a security perspective. To completely address these problems, we believe the right solution is to replace the broadcast service with a general directory service that enables scalable and robust end system discovery and protocol bootstrapping. Because the MAC addresses of end systems are stored in the directory, we can employ link-state-based unicast routing for fault-resilient communications, multicast, and better link utilization than spanning tree-based routing.

In the next section, we expose the scalability and reliability problems of today’s Ethernet. In Section III, we propose a general directory service to replace the outdated broadcast service in the Ethernet service model. We discuss the related work in Section IV, and conclude in Section V.

II. SUPPORTING BROADCAST MAKES ETHERNET FRAGILE AND UNSCALABLE

In this section, we present evidence that Ethernet today is neither scalable enough to support one million end systems nor fault-resilient enough for mission-critical applications. The problems we discuss here all arise as a result of the broadcast service model supported by Ethernet. To the best of our knowledge, this is also the first study to evaluate the behavior and performance of RSTP. Our results strongly contradict the popular beliefs about RSTP’s benefits.

A. Poor RSTP Convergence

In order to safely support the broadcast service in a switched Ethernet, a loop-free spanning tree forwarding topology is computed by a distributed protocol and all data packets are forwarded along this topology. The speed at which a new forwarding topology can be computed after a network component failure determines the availability of the network. Rapid Spanning Tree Protocol (RSTP) [11] is a change to the original Ethernet Spanning Tree Protocol (STP) [10] introduced to decrease the amount of time required to react to a link or bridge failure. Where STP would take 30 to 50 seconds to repair a topology, RSTP is expected to take roughly three times the worst case delay across the network [19].

We now provide a simplified description of RSTP which suffices for the purpose of our discussion. RSTP computes a spanning tree using distance vector-style advertisements of cost to the root bridge of the tree. Each bridge sends a BPDU (bridge protocol data unit) packet containing a *priority vector* to its neighbors containing the root bridge’s identifier and the path cost to the root bridge. Each bridge then looks at all the priority vectors it has received from neighbors and chooses the neighbor with the best priority vector as its path to the root. One priority vector is superior to another if it has a smaller root identifier, or, if the two root identifiers are equal, if it has a smaller root path cost.

The port on a bridge that is on the path toward the root bridge is called the *root port*. A port on a bridge that is connected to a bridge that is further from the root is called a *designated port*. Note that each non-root bridge has just one root port but can

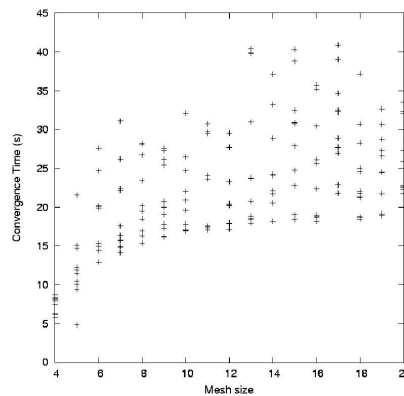


Fig. 1. Time for a full mesh topology to stabilize after the root bridge crashes.

have any number of designated ports; the root bridge has no root port. To eliminate loops from the topology, if a bridge has multiple root port candidates, it drops data traffic on all candidates but the root port. Ports that drop data traffic are said to be in state *blocking*.

We have built a simulator for RSTP and have evaluated its behavior on ring and mesh topologies varying in size from 4 to 20 nodes. We have found that, contrary to expected behavior, in some circumstances, RSTP requires *multiple seconds* to converge on a new spanning tree. Slow convergence happens most often when a root bridge fails, but it can also be triggered when a link fails. In this section, we explore two significant causes of delayed convergence: count to infinity and port role negotiation problems.

1) *Count to Infinity*: Figure 1 shows the convergence time for a fully connected mesh when the root bridge crashes. We define the convergence time to be the time from the crash until all bridges agree on a new spanning tree topology. Even the quickest convergence time (5 seconds) is far longer than the expected convergence time on such a topology (less than 1 ms). The problem is that RSTP frequently exhibits count to infinity behavior if the root bridge should crash and the remaining topology has a cycle.

When the root bridge crashes and the remaining topology includes a cycle, old BPDUs for the crashed bridge can persist in the network, racing around the cycle. During this period, the spanning tree topology also includes the cycle, so data traffic can persist in the network, traversing the cycle continuously. The loop terminates when the old root’s BPDU’s MessageAge reaches MaxAge, which happens after the BPDU has traversed MaxAge hops in the network.

Note that the wide variation in convergence time for a given mesh size is indicative of RSTP’s sensitivity to the synchronization between different bridges’ internal clocks. In our simulations, we varied the offset of each bridge’s clock, which leads to the wide range of values for each topology.

Figure 2 shows a typical occurrence of count to infinity in a four bridge topology. The topology is fully connected, and each bridge’s priority has been set up so that bridge 1 is the first choice as the root bridge and bridge 2 is the second choice. At time t_1 , bridge 1 crashes. Bridge 2 will then elect itself root because it knows of no other bridge with superior priority. Simultaneously, bridges 3 and 4 both have cached information

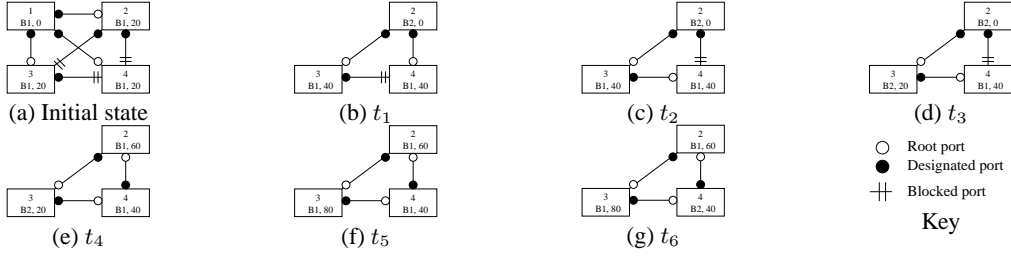


Fig. 2. When bridge 1 crashes, the remaining bridges count to infinity.

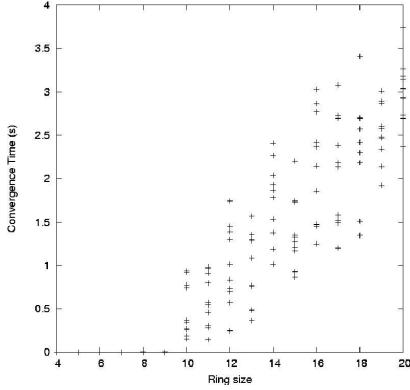


Fig. 3. Time for a ring topology to stabilize after one of the root bridge’s links fails.

saying that bridge 2 has a path to the root with cost 20, so both adopt their links to bridge 2 as their root ports. Note that bridges 3 and 4 both still believe that bridge 1 is root, and each sends BPDUs announcing that it has a cost 40 path to B1. At t_2 , bridge 4 sees bridge 2’s BPDU announcing bridge 2 is the root. Bridge 4 switches its root port to its link to bridge 3, which it still believes has a path to bridge 1. Through the rest of the time steps, BPDUs for bridges 1 and 2 chase each other around the cycle in the topology.

RSTP attempts to maintain least cost paths to the root bridge, but unfortunately, that mechanism breaks down when a bridge crashes. The result is neither quick convergence nor a loop-free topology.

2) *Hop by Hop Negotiation*: Figure 3 shows RSTP’s convergence times on a ring topology when a link between the root bridge and another bridge fails. Convergence times are below 1 second until the 10 bridge ring, when enough nodes are present that protocol negotiation overhead begins to impact delay, eventually driving the convergence time above 3 seconds.

Because RSTP seeks to avoid loops at all costs, it negotiates each port role transition. Initially, in a ring topology, each of the two links to the root bridge is used by half the nodes in the ring to reach the root bridge. When one of those links fails, half the links need to be “turned around.” In other words, the two bridge ports connected to a link trade roles, converting a port that’s considered further from the root bridge into a port that’s considered closer to the root.

Because the swap operation can form a loop, the transition has to be explicitly acknowledged by the bridge connected to that port. Usually, the bridge on the other side of the port responds almost immediately, since it can either accept the tran-

sition or suggest a different port role based entirely on its own internal state. But two problems can significantly slow the transition. First, if two neighboring bridges propose the opposite transitions to each other simultaneously, they will deadlock in their negotiation and both will wait 6 seconds for their negotiation state to expire. (This situation was not observed in the simulations above.)

Second, RSTP’s per-port limit on the rate of BPDU transmissions can add seconds to convergence. The rate limit takes effect during periods of convergence, when several bridges issue BPDUs updating their root path costs in a short time. When the rate limit goes into effect, it restricts a bridge to sending only one BPDU per port per second.

RSTP’s port role negotiation mechanism, which it uses to maintain a loop free topology, can in some circumstances cause convergence to be delayed by seconds.

B. MAC Learning Leads to Traffic Floods

As a direct consequence of the Ethernet service model, Ethernet bridges need to dynamically learn station locations by noticing which port traffic sent from each host arrives on. If a bridge has not yet learned a host’s location, it will flood traffic destined for that host along the spanning tree. When the spanning tree topology changes (e.g. when a link fails), a bridge clears its cached station location information because a topology change could lead to a change in the spanning tree, and packets for a given source may arrive on a different port on the bridge. As a result, during periods of network convergence, network capacity drops significantly as the bridges fall back to flooding. The ensuing chaos converts a local event (e.g. a link crash) into an event with global impact.

C. ARP Does Not Scale

ARP (Address Resolution Protocol) [17] is a protocol that liberally uses the Ethernet broadcast service for discovering a host’s MAC address from its IP address. For host H to find the MAC address of a host on the same subnetwork with IP address, D_{IP} , H broadcasts an ARP query packet containing D_{IP} as well as its own IP address (H_{IP}) on its Ethernet interface. All hosts attached to the LAN receive the packet. Host D , whose IP address is D_{IP} , replies (via unicast) to inform H of its MAC address. D will also record the mapping between H_{IP} and H_{MAC} . Clearly the broadcast traffic presents a significant burden on a large network. Every host needs to process *all* ARP messages that circulate in the network.

To limit the amount of broadcast traffic, each host caches the IP to MAC address mappings it is aware of. For Microsoft

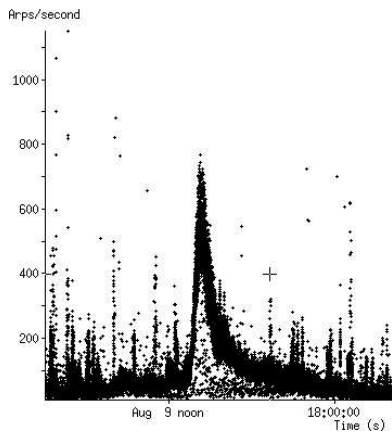


Fig. 4. ARPs received per second over time on a LAN of 2456 hosts.

Windows (versions XP and server 2003), the default ARP cache policy is to discard entries that have not been used in at least two minutes, and for cache entries that are in use, to retransmit an ARP request every 10 minutes [4].

Figure 4 shows the number of ARP queries received at a workstation on CMU’s School of Computer Science LAN over a 12 hour period on August 9, 2004. At peak, the host received 1150 ARPs per second, and on average, the host received 89 ARPs per second, which corresponds to 45 kbps of traffic. During the data collection, 2,456 hosts were observed sending ARP queries. We expect that the amount of ARP traffic will scale linearly with the number of hosts on the LAN. For 1 million hosts, we would expect 468,240 ARPs per second or 239 Mbps of ARP traffic to arrive at each host at peak, which is more than enough to overwhelm a standard 100 Mbps LAN connection. Ignoring the link capacity, forcing hosts to handle an extra half million packets per second to inspect each ARP packet would impose a prohibitive computational burden.

Note that ARP helps illustrate the more general problem of how giving end systems the ability to flood the entire network opens the door to unscalable protocol designs.

III. REPLACING BROADCAST SERVICE WITH DIRECTORY SERVICE

Our position is to eliminate Ethernet’s broadcast service entirely such that no more network flooding is allowed. Without the capability to broadcast, however, hosts on the network can no longer be passively discovered. Thus, we need to enable active host discovery and find alternative ways to support applications that currently use the broadcast service. We propose to replace the broadcast service with a directory service that can scale to one million hosts. Not having to deal with the complication of broadcast frees the network to employ a more robust link-state routing protocol. The directory service supports explicit end system location registration, replacing Ethernet’s MAC address learning. This enables the use of link-state unicast and multicast routing. The directory service also acts as a rendezvous to facilitate applications that currently use broadcast as a rendezvous mechanism.

A. Overview

We can classify the use of the broadcast service in Ethernet into two categories. The first category is usage to reach an unknown

system. For example, an ARP client uses broadcast to reach an unknown system with a particular IP address. The second category is usage of broadcast to facilitate point-to-multipoint communication in applications such as video conferencing. Although only a multicast to a subset of the systems is usually needed, Ethernet implements multicast by broadcasting and packet filtering at end systems.

To support these application categories, we propose a new directory service and the use of link-state routing to replace the broadcast service. In this design, the directory is replicated at all bridges. Each end system registers with the instance of the directory service running at its local bridge. Since bridges can discover the MAC addresses of the end systems attached to them through this directory registration, robust link-state-based unicast routing is made possible. This eliminates data packet flooding in normal communications, and allows better link utilization and load balancing than the current spanning tree-based forwarding approach. End systems that offer services can also register their attributes with the directory. Bootstrapping problems in ARP and DHCP can be solved by querying the directory service at the local bridge, using the directory as a rendezvous. Finally, multicast using source-specific trees or core-based trees can be built on top of link state-based unicast routing. The directory service eliminates the scalability and fault-resiliency problems with today’s Ethernet’s broadcast service while retaining the plug-and-play benefits of Ethernet.

B. Service Primitives

The **Register** message, which includes a sequence number, is sent from an end system to its locally connected bridge to register the end system’s MAC address and any additional attributes (e.g. the end system’s IP address). If there are multiple bridges on the segment, only the bridge with the smallest bridge id will listen to register messages. All end systems must register when they first connect to the network, and periodically re-register while still connected. When a bridge receives a register message, it will return an ACK, store the MAC address and its attributes locally, and will begin distributing the information to neighboring bridges. This mechanism completely replaces the ad hoc MAC address learning in today’s Ethernet.

State messages are sent from one bridge to another to circulate MAC/attribute data and link state. All State messages are uniquely identified by the sending bridge’s MAC address and a sequence number. If a bridge receives a previously unseen State message, it stores the message contents locally and floods the message to its neighbors. The directory is therefore replicated at all bridges. The end system MAC addresses associated with every bridge combined with link-state topology information enables unicast forwarding. The choice of a link-state protocol ensures quick route convergence when network components fail. Temporary routing loops are still possible, but since broadcast is not supported, a data packet can at worst persist in a network loop until the loop is resolved.

The **Query** message is sent by an end system to its local bridge in order to perform service discovery. The query contains an attribute/value tuple (e.g. attribute IP_Address and value "1.2.3.4"). The bridge will reply either with one or more MAC addresses of end systems that have matching attributes,

or, if no matches are found, the bridge will return a negative acknowledgment.

A host can specify two attributes. The first is an IP address, which is used to eliminate the need for ARP. The second is a service specifier, which is used to advertise hosts which provide a service, e.g. DHCP. Adding new attributes is a straightforward way to support layering additional functionality on top of our system.

C. Usage Examples

1) *ARP*: Instead of sending a broadcast query, a host in our system sends a query directly to its local bridge asking for the MAC address, H_{MAC} , associated with an IP address, H_{IP} . If the host is on the network, the query will be answered immediately with a reply containing H_{MAC} , otherwise, the bridge will return a negative acknowledgment.

2) *DHCP*: In the DHCP [6] protocol, when a host attaches to a network, it broadcasts a DHCPDISCOVER message. Any DHCP server on the network can send back a DHCPOFFER message in reply to offer the host configuration parameters. In our system, a host queries its local bridge for the MAC address of a host with its attribute containing "DHCP". The bridge may return one (or more) MAC addresses. The client then unicasts its DHCPDISCOVER message to one of the MAC addresses. The rest of the protocol proceeds normally.

Our system can also support redundant servers. Assume there is a server monitoring protocol that a pool of servers use to elect an "active" server and to promote a standby server to active status in the event of a server failure. In the case of a DHCP service, after a server failure, the newly active server re-registers with its local bridge, with its attribute set to "DHCP". The state dissemination protocol will immediately spread this information to all bridges.

3) *Host Mobility*: When a host moves from one port to another (whether on the same bridge or a different bridge), the host sends a new registration packet. If the host has moved to a new port on the same bridge, the bridge just updates its internal forwarding tables. If the host has moved to a new bridge, that bridge will disseminate the host's new registration information. This new information immediately overwrites the old information stored in bridges. The registration message's sequence number enables bridges to tell that the newer state information should overwrite the previous state information since the new registration's sequence number is larger. To deal with the possibility of a crash or reboot, a host can either keep its sequence number in stable storage, or choose the sequence number based on the current time, or in the worst case, wait several minutes for its old state to timeout.

D. Scaling to One Million Nodes

At the center of our proposal is a replicated directory and a link-state routing protocol. Based on our estimates, we believe maintaining the directory and link-state database in a one million end system network has acceptable overhead relative to today's technologies. There are three aspects of scalability to consider: end system overhead, scalability of the routing protocol, and scalability of the directory of end systems.

From the end host's perspective, the additional burden of having to register and refresh state with a local bridge is neg-

ligible in comparison to the reduction in the amount of broadcast traffic it receives. Extrapolating based on our results from Section II, on a million end system network, ARP traffic alone could account for a peak of 239 Mbps of traffic and an average of 18.55 Mbps of traffic to each host.

It would require at least 1000 bridges to build a network capable of supporting one million end systems. First, we consider the routing message processing overhead at each bridge. Work by Alaettinoglu, Jacobson, and Yu [5] has demonstrated that a router's link state protocol processing can be fast enough to scale up to networks with thousands of nodes if incremental shortest path algorithms are employed instead of Dijkstra's algorithm. As for link state update traffic, if we assume 1000 bridges with 50 links per bridge sending updates every 30 minutes, this would translate into an average traffic load of only 3 kbps. Further, recent work [16] proposes that flooding be decreased or even temporarily stopped when a topology is stable.

We now consider the burden that the distributed directory places on bridges. Host directory information will be refreshed every minute. For a network with 1 million end systems, at an average of 14 bytes of state per end system (6 bytes for the MAC address, 4 bytes for the IP address, and 4 bytes for the sequence number), 14 MB of storage is required at each bridge, and an average of 1.87 Mbps of bandwidth is required per bridge-to-bridge link. Again using adaptive flooding when the directory is stable, we can lower this even further. Even at 1.87 Mbps, this is a factor of 10 less than the average amount of traffic that ARP would impose on a network of this size. Further, the directory flooding traffic is only sent over bridge-to-bridge links, which are typically an order of magnitude or more faster than the host-to-bridge links that ARP traffic is sent over.

In terms of handling dynamic directory changes, because our target network is an enterprise, we assume that the hosts on the network are relatively stable and that, on average, a host only changes state (being switched on, or changing the location at which they're attached to the LAN) once every 24 hours. With this model, we should expect an average of 12 state changes per second. Let us assume the peak rate is two orders of magnitude higher, or 1200 changes per second. To support this amount of on-demand update only requires 134 kbps per bridge-to-bridge link.

In addition to a bridge's central processor, which will store and track the complete directory of MAC locations, a bridge also has forwarding tables on each line card. The line cards need not hold the entire database, but only those entries currently being used. This makes updating the line cards a much less expensive operation since each change in a route to a bridge would otherwise require updating 1000 MAC addresses in each line card. Instead, when a line card receives traffic for a destination for which it has no forwarding entry, the line card will query the central processor for and then install the proper entry. Entries that have not been used recently will be flushed from the line card's forwarding table, just as they are in today's bridges.

IV. RELATED WORK

Several researchers, having observed that spanning trees provide poor network throughput, have proposed ways to augment

spanning tree so that data traffic can use additional, off-tree network links. Viking [20] is a system that uses Ethernet's built-in VLANs to deliver data over multiple spanning trees. Pellegrini et al's Tree-based Turn-Prohibition [14] is reverse-compatible with RSTP bridges, while STAR [12] is reverse-compatible with STP bridges. Finally, SmartBridge [18] creates multiple source-specific spanning trees that are always promised to be loop-free, even during convergence events. All these proposals aim to maintain Ethernet's original service model, where flooding is used to discover end systems and ad hoc MAC address learning is employed.

LSOM [7] and Rbridges [15] are both systems that propose to replace spanning tree with a link state protocol, but without removing broadcast. LSOM includes no mechanism to damp traffic that might be generated during a routing loop that could occur during convergence. Rbridges, on the other hand, encapsulate layer 2 traffic with an additional header that contains a TTL value. While our proposal uses link state routing, we address the fundamental cause for fearing that a loop will occur: we change the service model by removing broadcast and learning and adding a directory service to support end system discovery and protocol rendezvous.

ARP's scalability problem is fairly well known. ISO's ES-IS [9] and CLNP [8] protocols specify a host registration protocol and a LAN-wide database of station locations instead of ARP. CLNP and ES-IS are layer 3 protocols that provide some of the desirable features of layer 2 (e.g. host mobility). In contrast, our approach is to enhance a layer 2 protocol by adding desirable features from layer 3 (e.g. scalability). At the mechanism level, our directory service generalizes the end-system-router interaction in ES-IS by creating a rendezvous mechanism for service location in addition to host location. Finally, McDonald and Znati [13] have compared ARP's performance to that of ES-IS [9].

V. CONCLUSION

In this paper, we argue that there are strong incentives to scale Ethernet up to support 1 million hosts. However, Ethernet's service model, which embraces broadcast, limits its scalability and fault-resiliency. We propose changing Ethernet's service model, removing its broadcast and learning facilities and adding a directory service feature. This and link state routing together enable robust service discovery, eliminate the scalability and security problems inherent to broadcast, support multicast, and make routing converge more quickly in case of failure.

Ethernet's traditional use has been in enterprise networks, but today it is being deployed in other settings including residential broadband access, metropolitan area networks, and storage area networks. As we argue in this paper, even the enterprise network has become a new setting since it will contain orders of magnitude more hosts. That Ethernet can even be envisioned to work in these new settings is a testament to its architecture, but there are still numerous challenges to overcome. Most of Ethernet's design decisions were made long before it was being deployed in any of these new settings. Therefore, it makes sense to revisit Ethernet's design with an eye toward determining how its architecture should continue to evolve.

REFERENCES

- [1] Bellsouth metro ethernet. <http://www.bellsouthlargebusiness.com>.
- [2] The Force10 EtherScale architecture - overview. <http://www.force10networks.com/products/pdf/EtherScaleArchwp2.0.pdf>.
- [3] Yipes. <http://www.yipes.com>.
- [4] Microsoft Windows Server 2003 TCP/IP implementation details. <http://www.microsoft.com/technet/prodtechnol/windowsserver2003/technologies/networking/tcpip03.mspx>, June 2003.
- [5] C. Alaettinoglu, V. Jacobson, and H. Yu. Towards milli-second IGP convergence. IETF draft-alaettinoglu-ISIS-convergence-00, November 2000. Available at <http://www.packetdesign.com/news/industry-publications/drafts/convergence00.pdf>.
- [6] R. Droms. Dynamic host configuration protocol. RFC 2131, March 1997.
- [7] R. Garcia, J. Duato, and F. Silla. LSOM: A link state protocol over mac addresses for metropolitan backbones using optical ethernet switches. In *Proceedings of the Second IEEE International Symposium on Network Computing and Applications (NCA '03)*, April 2003.
- [8] ISO. ISO 8473 Protocol for Providing the OSI Connectionless-Mode Network Service.
- [9] ISO. ISO 9542 End System to Intermediate System Routing Information Exchange Protocol for Use in Conjunction with the Protocol for Providing the Connectionless-Mode Network Service.
- [10] LAN/MAN Standards Committee of the IEEE Computer Society. *IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Common specifications Part 3: Media Access Control (MAC) Bridges*, 1998.
- [11] LAN/MAN Standards Committee of the IEEE Computer Society. *IEEE Standard for Local and metropolitan area networks-Common Specifications Part 3: Media Access Control (MAC) Bridges-Amendment 2: Rapid Reconfiguration*, June 2001.
- [12] K.-S. Lui, W. C. Lee, and K. Nahrstedt. STAR: A transparent spanning tree bridge protocol with alternate routing. In *ACM SIGCOMM Computer Communications Review*, volume 32, July 2002.
- [13] B. McDonald and T. Znati. Comparative analysis of neighbor greeting protocols ARP versus ES-IS. In *Proceedings of IEEE 29th Annual Simulation Symposium*, April 1996.
- [14] F. De Pellegrini, D. Starobinski, M. G. Karpovsky, and L. B. Levitin. Scalable cycle-breaking algorithms for gigabit Ethernet backbones. In *Proceedings of IEEE Infocom 2004*, March 2004.
- [15] R. Perlman. Rbridges: Transparent routing. In *Proceedings of IEEE Infocom 2004*, March 2004.
- [16] P. Pillay-Esnault. OSPF refresh and flooding reduction in stable topologies. IETF draft-pillay-esnault-ospf-flooding-07.txt, June 2003.
- [17] D. C. Plummer. An Ethernet address resolution protocol. RFC 826, November 1982.
- [18] T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson. SmartBridge: A scalable bridge architecture. In *Proceedings of ACM SIGCOMM 2000*, August 2000.
- [19] M. Seaman. Loop cutting in the original and rapid spanning tree algorithms. http://www.ieee802.org/1/files/public/docs99/loop_cutting08.pdf, November 1999.
- [20] 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*, March 2004.