Revisiting Ethernet: Plug-and-play made scalable and efficient

(Invited Paper)

Changhoon Kim
Department of Computer Science
Princeton University
Princeton, NJ 08540-5233
Email: chkim@cs.princeton.edu

Jennifer Rexford
Department of Computer Science
Princeton University
Princeton, NJ 08540-5233
Email: jrex@cs.princeton.edu

***Abstract*— Because Ethernet bridging does not scale, most enterprise networks consist of small Ethernet-based subnets interconnected by IP routers. Although Ethernet's flat addressing and transparent bridging allow each subnet to run with minimal configuration, interconnecting subnets at the IP level introduces significant management overhead that increases with the size of the network. As an alternative, we propose a scalable and efficient zero-configuration enterprise (SEIZE) networking architecture. SEIZE provides *plug-and-play* capability via globally unique flat addressing, while ensuring *scalability* and *efficiency* through shortest-path routing and hash-based location resolution. Switches perform location resolution on demand and can cache the results to optimize routing paths and to reduce the number of location-resolution requests. We present a design overview of SEIZE and show that it attains the best of Ethernet and IP.**

I. Introduction

Ethernet has many appealing features, especially for enterprise network administrators. The end-host devices already have permanent, globally unique MAC-48 addresses, obviating the need to configure the hosts. In addition, Ethernet switches self-learn the locations of end hosts, keeping the configuration of networking nodes to a minimum as well. Moreover, permanent flat addresses simplify the handling of host mobility, network troubleshooting, and access-control policies. An "all Ethernet" network architecture would be extremely attractive, if it were not for the following serious scalability limitations:

- **Flooding-based delivery:** Ethernet bridging relies on flooding to deliver frames to unknown destinations. Flooding consumes excessive link bandwidth and leads to large forwarding tables in the switches.
- **Inefficient forwarding paths:** In its original form, Ethernet bridging cannot incorporate back-up paths because forwarding loops can arise. Although the Spanning Tree Protocol (STP) solves this problem, delivering packets along a single tree leads to unnecessarily long paths and inefficient use of network resources.
- **Broadcasting for basic service:** Ethernet relies on broadcasting to support essential discovery services, such as Address Resolution Protocol (ARP) and Dynamic Host Configuration Protocol (DHCP). This consumes excessive resources, and also introduces security vulnerabilities.

To overcome these limitations, most enterprises employ a hybrid architecture that interconnects small Ethernet subnets via IP routers. IP routing allows shortest-path forwarding over any topology, greatly improving efficiency and flexibility. Moreover, IP does not rely on flooding for packet delivery and employs a TTL (Time-To-Live) mechanism to discard packets stuck in loops, allowing routers to tolerate transient forwarding loops and change quickly to new paths after a failure. However, these benefits come at a price:

- **Configuration overhead:** IP uses hierarchical addressing and subnet-based routing. Although DHCP can automate end-host configuration, administrators must manage the subnet-specific configuration of router interfaces and routing protocols, and this configuration must be consistent with DHCP configuration. Correctly configuring multiple DHCP servers itself is difficult because there is no protocol for coordination among DHCP servers.
- **Addressing inefficiency:** Subnetting inevitably puts barriers between address blocks, resulting in inefficient address allocation. When the number of hosts per subnet varies widely (e.g., a wireless LAN), administrators must allocate IP addresses based on worst-case demand, which is often much worse than the average. In addition, hierarchical addressing makes mobility hard to support.

Moreover, these problems grow more serious in the face of change, such as deploying a new routing protocol, revising networking policy, adding or removing routers, or adding a new member to the network operations team unfamiliar with the current network configuration.

In this paper, we propose an alternative solution that combines Ethernet's plug-and-play capability with IP's scalability and performance, while avoiding any changes to the protocols and applications running on end-hosts. The outcome is a scalable and efficient, zero-configuration enterprise (SEIZE) architecture. To minimize configuration overhead, SEIZE employs flat addressing and automated host discovery at the network edge. To reduce the overhead of disseminating host-location information, switches *reactively* resolve a host's location using a robust hashing scheme. Switches can cache host-location information to optimize forwarding paths and prevent redundant resolution requests. For efficient resource usage, switches deliver packets along shortest paths. Scoping broadcast traffic in each broadcast domain is also possible via per-domain multicast trees. Finally, broadcast-based service-discovery protocols, such as ARP and DHCP, are handled via unicast-based resolution.

SEIZE is an alternative design of IEEE 802.1 Ethernet bridging—so called, Ethernet interconnection—protocols. Any Ethernet-based networks (e.g., 802.3 Ethernet, 802.11 Wireless LAN, 802.16 WiMAX) can incorporate SEIZE to interconnect themselves and can improve control-plane scalability and data-plane efficiency.

Our work is motivated by recent attempts to improve or redesign Ethernet bridging [1]–[5]. Among these, the most directly related proposals are Rbridges [1], [2] and Myers' architecture [5]. Rbridges improves efficiency via shortest-path forwarding and ensures robustness using a TTL mechanism; however, it does not address the *scalability* limitations of Ethernet bridging. In addition to advocating shortest-path forwarding, Myers' prohibits broadcasting and flooding to improve both efficiency and scalability. Nevertheless, the control-plane models proposed in Myers' architecture do not scale to a large network because of the overhead of disseminating each host's location information. SEIZE differs from Myers' work in that it provides *concrete and practical mechanisms* to solve the problems. We present more detailed review of the previous work in later sections.

II. Overcoming Ethernet's Limitations

In this section, we describe specific requirements that an Ethernet extension or replacement should meet for scalability and efficiency.

A. Avoiding flooding to unknown destinations

Upon receiving a frame from a source host to an unknown destination, Ethernet switches flood the frame, hoping that the actual destination eventually receives a copy. Meanwhile, every switch learns the source host's location, so that subsequent frames to the source (e.g., replies from the destination) do not require flooding. The self-learning mechanism makes Ethernet a "plug and play" technology. However, flooding introduces significant overhead in networks with many hosts and switches, such as campus-wide wireless LANs, multi-site wide-area VPNs (Virtual Private Networks) [6], or metropolitan-area Ethernet [7].

Network administrators typically manage this problem by reducing the size of broadcast domains (i.e., Ethernet segments) and interconnecting them via IP routing. However, this hybrid architecture introduces configuration overhead and inefficient use of IP addresses, as discussed in the previous section. Although VLAN (Virtual LAN) adds the flexibility to logically define each broadcast domain, each broadcast domain still corresponds to a separate IP subnet. Thus, the management complexity of IP still remains, in addition to VLAN's own configuration challenges (e.g., configuring VLAN trunks, and per-VLAN spanning trees). Moreover, use of VLANs does not scale well for switches that have to participate in multiple (or sometimes all) VLANs.

Instead, we argue that switches should always deliver unicast frames via unicasting. To accomplish this, we need a separate control-plane function to discover and disseminate host-location information.

B. Restraining broadcast for data-plane scalability

Both ARP and DHCP are integral bootstrapping protocols that utilize broadcasting to deliver a frame to a certain host (or a server) without knowing the host's (server's) layer-two address. We view the use of broadcasting for such a basic (and, hence, frequent) operation as a vestige of the shared-medium Ethernet where broadcasting had the same overhead as unicasting. Obviously, this design leads to unnecessary consumption of networking resources in the switched Ethernet architectures common today. Moreover, since ARP is required between each source and destination pair, its broadcasting overhead increases in proportion to the number of communicating host pairs.

We suggest that, in a large-scale Ethernet network, bootstrapping should not rely on network-wide broadcasting. Instead, ingress switches can work as intermediaries that transform bootstrap requests broadcasted by end-hosts into unicast queries to a directory service.

In addition to ARP and DHCP, other networking protocols and applications employ layer-two broadcasting and multicasting. Common examples include networked printing, P2P file sharing, and IP multicast. Thus, even when we employ the enhanced ARP and DHCP schemes, this broadcast/multicast traffic can impose significant load on the network. Again, network administrators typically cope with this problem by limiting broadcast domains physically or logically. Recently, some researchers have taken a clean-slate approach to redesign enterprise networks that preclude broadcasting under any circumstances [5], [8]. However, this approach requires application- or protocol-specific reverse engineering to design a substitute for broadcast for each application.

Our position is that general applications (i.e., not ARP/DHCP) should still be able to use broadcasting/multicasting for backwards-compatibility. Instead, we argue Ethernet itself should change, to handle broadcast traffic in a manner that is more scalable and easier to configure.

C. Keeping forwarding tables small

Each Ethernet switch maintains a forwarding table which is populated by the self-learning mechanism. Whenever a switch receives a frame with a new source address, the self-learning mechanism creates a corresponding entry (i.e., a tuple of the source address and an output port) for the new host. However, when combined with flooding and broadcasting, the self-learning mechanism creates forwarding-table entries in every switch in the broadcast domain. This problem becomes more serious as a network grows. Maintaining small forwarding tables is a critical issue especially for light-weight switches, like wireless access points, that do not have much memory. Even core switches can experience memory exhaustion when attacks, such as MAC flooding where the attacker sends many frames with randomly-spoofed source addresses, substantially increase the forwarding-table size.

Evicting inactive host entries reduces the table sizes, at the expense of more frequent flooding. Reducing the size of broadcast domains helps reduce table size, although the switches spanning multiple broadcast domains would still have very large tables.

Instead, we need a routing scheme that installs host entries only when and where the entries are actually needed. Traffic patterns in enterprise networks are well-suited to a *reactive* approach because most hosts communicate primarily with a small set of popular hosts, such as printers, department Web servers, and a gateway router to the Internet.

D. Ensuring efficient forwarding paths

Ethernet is especially vulnerable to forwarding loops. Even a transient loop, when combined with flooding and broadcasting, can lead to a broadcast storm (i.e., endless multiplication of frames). Also, forwarding loops make switches mis-learn the locations of sending hosts. Ethernet bridging prevents these problems by maintaining a loop-free topology (i.e., a tree) using the Spanning Tree Protocol (STP). However, using a spanning tree leads to longer forwarding paths, poor load balancing, and limited flexibility.

Supporting multiple spanning trees would make more efficient use of network resources. For example, MSTP (Multiple Spanning Tree Protocol) enables switches to maintain separate trees for different broadcast domains [9]. However, the forwarding paths in each domain remain sub-optimal. Simultaneously and adaptively utilizing multiple spanning trees in a single broadcast domain could offer futher improvements [4], at the expense of centralized monitoring and computation.

Fundamentally, allowing switches to send traffic over shortest paths, rather than a shared tree, would improve performance and make more efficient use of network resources. To accomplish this, the switches would need to run a routing (e.g., link-state or distance-vector) protocol. In this case, however, there must be an alternative way to prevent broadcast storms.

III. SEIZE Architecture

In this section, we describe the architectural components of SEIZE. We first introduce SEIZE's foundational structures: flat addressing (of end-hosts) and link-state routing (among switches). Then, we describe our mechanisms for managing host-location information and delivering frames along shortest paths. Next, we describe how to handle network changes, such as topology changes and mobile hosts, efficiently. Finally, we explain how to handle ARP, DHCP, and other broadcast traffic in a scalable fashion.

A. Flat addressing and shortest-path forwarding

SEIZE uses only end-hosts' flat MAC addresses for end-to-end delivery. Frames between two hosts are delivered along shortest paths, not along a spanning tree. To enable this, switches run link-state routing.

1) Flat addressing of end-hosts: SEIZE uses MAC-48 (a.k.a. IEEE EUI-48) addresses as unique identifiers of routable entities. Since Ethernet addresses are flat, unique, and hard-coded, no configuration (whether manual or automated) is required to use them. The use of Ethernet addressing also guarantees backwards-compatibility for end-hosts.

IP addresses are given to end-hosts only for external reachability and application-level compatibility, *not for routing*. Inside an enterprise, frames are delivered based only on their destination MAC-48 addresses. This practice liberates IP addresses from location, turning them to site-local identifiers assigned to end-hosts, not subnet-local ones assigned to network attachment points. This obviates the need to configure IP subnets or reconfigure hosts when they move. In addition, our approach makes address allocation more efficient and simplifies troubleshooting and access-control policies.

End hosts still use ARP to resolve the MAC address associated with an IP address. Although a host can keep its own IP address regardless of its location, network administrators can still utilize DHCP to save address space by assigning IP addresses to active hosts dynamically. Running DHCP this way is simple because there is no need to maintain consistency with subnet configuration, since IP subnets do not exist.

2) Link-state routing with no host information: To enable shortest-path forwarding, SEIZE switches run a link-state routing protocol and share a complete topology composed *only of the switches*. The configuration of the link-state routing, therefore, is similar to that of a typical backbone network, where IP prefixes are *not* injected into the IGP (Interior Gateway Protocol), rather than of a conventional enterprise network. Running a link-state routing protocol this way requires only minimal configuration. Using a simple peer-discovery protocol, each switch can easily discover neighbor switches and can advertise only switch-to-switch links as unnumbered interfaces, as well as its unique identifier (e.g., the smallest MAC addresses it possesses) as a router LSA (Link State Advertisement).

SEIZE switches utilize the topology information for two purposes: to compute shortest paths for unicast traffic between any two switches, and to construct a multicast tree for broadcast/multicast traffic in a given broadcast domain.

B. Scalable location management and optimal delivery

SEIZE employs a novel, hash-based location management scheme to avoid flooding frames to unknown destinations. Figure 1 illustrates the mechanisms introduced in this section; each step is annotated with a subsection number in which the step is described. Dashed lines denote control flows, and solid lines denote data flows.

1) Host location detection by adjacent switches: Each end-host's location is managed by its adjacent switch. The switch takes responsibility for detecting arrivals and departures of all end-hosts directly connected to it, in addition to delivering data to and from these hosts. A switch could detect a host's arrival and departure either explicitly or implicitly, depending on the underlying link technology. An example of *explicit*

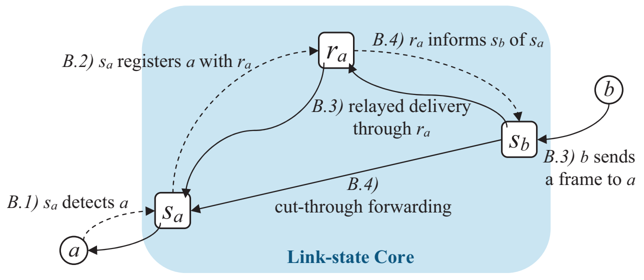
B.2) s_a registers a with r_a
r_a
B.4) r_a informs s_b of s_a
b
B.3) relayed delivery through r_a
s_b
B.3) b sends a frame to a
B.4) cut-through forwarding
B.1) s_a detects a
s_a
a
Link-state Core

Fig. 1. Hash-based location resolution and delivery

detection is an active registration initiated by a host, as in 802.11 association and authentication; explicit disassociation when a host leaves or moves (e.g., 802.11 hand-off) makes it easy to detect a departure. An example of *implicit* detection is the self-learning mechanism in Ethernet bridging. Since a host usually generates either a DHCP discovery or an ARP request as soon as it arrives at a network, the adjacent switch can detect arrival events in a timely fashion. A host's departure would be detected by periodic polling or repeated losses of link-layer acknowledgments.

2) Hash-based location registration: For each end-host there is a special switch (called a *relay*) that maintains the host's current location (i.e., the identifier of the host's adjacent switch). The mapping between a host and its relay is determined by a hash function F that all switches in a network jointly use. Upon detecting the arrival of host a with MAC-48 address mac_a, the adjacent switch s_a computes a relay switch $r_a = F(mac_a)$, and informs switch r_a of a's location.

3) Location resolution and relayed delivery: Suppose that a source host b connected to a different switch s_b wants to send a frame to a destination host a. Upon receiving the frame from b, the switch s_b needs to determine the location of a. Rather than flooding the frame, s_b sends the frame to the relay switch r_a, which in turn transmits the frame to switch s_a. Because r_a may be multiple hops away, s_b encapsulates the original frame in an outer frame destined to r_a; then, r_a decapsulates the frame and re-encapsulates in a frame destined to s_a. The traffic traverses the shortest path from s_b to r_a, followed by the shortest path from r_a to s_a.

4) Optimizing forwarding paths: To optimize the forwarding path, two approaches are possible: cut-through forwarding, and closest-relay selection. Under cut-through forwarding, when r_a receives the frame from s_b, r_a can notify s_b that a's current location is s_a. This enables s_b to send subsequent frames *directly* to s_a over a shortest path. Switch s_b can cache this location information and apply various cache-eviction policies (e.g., inactive timeout or least-recently-used eviction). To avoid cache poisoning attacks, however, these cached information entries for ongoing flows must not evict the information entries of the hosts that are directly connected to s_b or are registered with s_b for relaying. On the other hand, in the closest-relay selection scheme, each switch computes multiple independent hashes and selects the closest relay among the multiple candidates. Note that the link-state database provides a network topology on which switches can naturally compare distances, without employing a separate measurement framework.

C. Coping with changes with minimal overhead

We have described how SEIZE works in a stationary period. In this subsection, we introduce schemes to efficiently handle network dynamics—topology changes and host mobility.

1) Responding to network topology changes: To handle changes of network topology (e.g., arrival or departure of a switch) efficiently, SEIZE utilizes a Consistent Hash [10] for F. When using Consistent Hash, each switch maintains a sparse, ring-shaped hash space (e.g., a hash space of size 2^{128} using MD5) on which every switch in the network is placed. Each switch's position on the ring is determined by the hash value of the switch's unique identifier. When a switch needs to map a host to a switch, the switch first places the host onto the ring according to the hash value of the host's MAC-48 address. Then it maps the host to the closest switch on the ring. Thus, when a new switch appears, only some of the hosts who were previously assigned to the nearest switch of the arriving switch need to be remapped to the new switch. Similarly, when an existing switch disappears, only some of those hosts who were previously assigned to the departing switch should be remapped. Note that path changes alone do not trigger host remapping.

Switches handle this remapping process in a distributed fashion. That is, each switch individually responds to a network change by remapping some of the hosts that are directly connected to itself, and then by re-registering those remapped hosts with a new switch. Note that, since SEIZE utilizes a link-state protocol, any change in the core topology is quickly and reliably disseminated to every switch, allowing rapid remapping.

When a relay switch fails, or link failure partitions a relay from the rest of the network, an ingress switch might not be able to resolve a newly arriving flow's destination. To enhance availability, a SEIZE network can keep a host's information at more than one relay switch. For example, a host's location can be registered not only with the closest switch on a hash ring, but also with the second-closest switch.

2) Responding to changes in host location: When a host a moves from one location to another, the host's new adjacent switch s_a^{new} detects the arrival, hashes a's address, and registers the new location with a's relay switch r_a. Relay switch r_a notifies the old location s_a^{old} of the new location s_a^{new} before overwriting a's location information. However, since other ingress switches may have cached the s_a^{old} location, switch s_a^{old} may continue to receive frames destined to a. A cache-refresh mechanism (e.g., inactive timeout-based eviction) running at those ingress switches would ensure that the stale cache entries are eventually discarded or updated; alternatively, upon receiving a frame destined to a, the switch s_a^{old} could explicitly notify the ingress switch that the location has changed. Note that, since switches use encapsulation, s_a^{old} can determine the identifier of the ingress switch. In the meantime, for service continuity, switch s_a^{old} could forward these mis-delivered frames to r_a or s_a^{new}.

D. Scalable broadcasting and isolation via hashing

The hash-based on-demand resolution improves control-plane scalability, but not necessarily data-plane scalability because of broadcast and multicast traffic. To attain data-plane scalability, SEIZE handles ARP and DHCP, two prominent sources of broadcast traffic, via unicast-based resolution. For backwards-compatibility, SEIZE also supports regular broadcast/multicast with a VLAN-like scoping option.

1) Proxy resolution for reducing broadcast: SEIZE replaces ARP with a hash-based resolution scheme. To do this, switches run the same host registration and mapping mechanisms using the hash function F with *IP addresses*. When host a's IP address ip_a is first discovered (usually when a arrives at a network), its adjacent switch s_a registers both a's MAC and IP addresses with a corresponding *resolver* switch $F(ip_a)$. Later, when another host b issues an ARP request to determine the MAC address associated with ip_a, its adjacent switch s_b directs the request to switch $F(ip_a)$, instead of broadcasting the ARP request. We call this mode of ARP operation proxy resolution. Proxy resolution is different from proxy ARP in that an ARP reply contains the target's address (e.g., mac_a), not the proxy's one (e.g., mac_{s_b}).

Similarly, SEIZE also avoids broadcasting DHCP messages over the entire network by having ingress switches relay DHCP messages. Upon receiving a broadcast DHCP message from an end-host, the ingress switch encapsulates the message with a DHCP server's MAC address, and delivers it to the server along the shortest path. Network administrators can easily implement this mechanism using the DHCP relay agent standard [11]. A DHCP relay agent allows an end-host to communicate with a DHCP server located in a different broadcast domain by converting a broadcast DHCP message to a unicast IP packet destined to the DHCP server. In SEIZE, instead of using IP encapsulation, switches can utilize Ethernet encapsulation. To ensure minimal configuration, we can maintain a broadcast group that is, by default, composed of all switches and DHCP servers; a broadcast group is similar to a VLAN, as discussed in more detail in the next subsection. Using a simple broadcast discovery protocol in this special broadcast group, switches can find the nearest DHCP server without explicit configuration.

2) Group-based broadcasting for backwards-compatibility: To support regular (non-ARP, non-DHCP) broadcast traffic efficiently, SEIZE also allows network administrators to reduce broadcast domains by introducing the notion of *group*. A group is similar to a VLAN in Ethernet bridging in that broadcast traffic is confined within a group, but it is more flexible than VLAN in the following respects. First, a group does *not* correspond to an IP subnet; each host in a group uses its own site-local IP address regardless of its group membership. Second, a host can belong to multiple groups. Third, unlike VLAN, direct reachability (via unicast) between hosts in two different groups *may* or may not be permitted depending on the access-control policy between the two groups (or even hosts). In this subsection, we describe a group-based broadcasting mechanism; an inter-group access control mechanism is introduced in the next subsection.

Broadcasting in SEIZE works as follows. All broadcast frames within a group are delivered through a multicast tree sourced at a dedicated switch, namely a *broadcast server*, of the group. The mapping between a group and its broadcast server is again determined by the same hash function F using a group's identifier as an input. When a switch, for the first time, detects an end host that is a member of group g[1], the switch issues a join message that is carried up to the nearest graft point on the tree toward the root (i.e., g's broadcast server). Note that, by virtue of F, all the switches along the new branch taken by the join message can consistently map group g to its broadcast server. When a host departs, its adjacent switch also prunes a branch if necessary. As a result, each switch in a network maintains local knowledge about which of its interfaces belongs to which groups. Finally, when an end-host in g sends a broadcast frame, its adjacent switch encapsulates and forwards it to g's broadcast server. The frame is then decapsulated by the broadcast server and delivered along g's multicast tree. Note that, since multicasting with RPF (Reverse Path Forward) check ensures loop-free forwarding, our broadcasting mechanism prevents broadcast storms.

3) Hash-based inter-group access control: Network administrators can control inter-group reachability as follows. First, each end-host's group membership is determined by the host's adjacent switch and registered with corresponding switches along with other information (e.g., location or IP address). Enforcing access-control policy then takes place during the process of address/location resolution. The key idea is to permit resolution only when the access policy between a resolving host's group and a resolved host's group permits access. Thus, depending on access policy, one host in a group may not resolve the MAC address of another host in a different group. Moreover, even when the first host somehow obtains the second host's MAC address, delivering frames from the first to the second one may remain impossible because the relay switch for the second host can refuse it as per a corresponding access policy. When access is permitted, however, the path between two hosts in different groups remains optimal as if the hosts were in the same group.

IV. Scalability and Efficiency Benefits of SEIZE

When delivering every frame through a relay, SEIZE has a scalable control plane and light-weight support for host mobility. Moreover, SEIZE retains these properties even when cut-through forwarding is used to improve data-plane efficiency.

A. Ensuring a scalable control-plane via relaying

Delivering all frames through relay switches increases stretch (i.e., average path length), in exchange for reducing

[1]The way administrators associate hosts with corresponding groups and policies is beyond the scope of this paper. For Ethernet, a policy management framework that can automate this task (e.g., mapping an end-host or flow to a VLAN) is already available, so SEIZE can employ the same model.

both forwarding-table size and the number of control messages. Moreover, this practice enables SEIZE to support host mobility with minimal overhead.

1) Minimal overhead for disseminating host-location information: Each host's location information is advertised only to a small, constant number of switches—usually two (its adjacent switch and its relay); no other control messages are required. The amount of control overhead a switch has to handle depends on the number of hosts directly connected to it and the hash function F's distribution, both of which are predictable and tunable. Since link-state routing is used only to compute the paths between switches, a single routing protocol instance can serve an entire network.

2) Small forwarding tables: In SEIZE, each switch needs to keep only the following three types of location information: the hosts that are directly connected to the switch, the hosts for which the switch is a relay, and the other SEIZE switches. Assuming that the number of switches (S) is much smaller than the number of end-hosts (H), the first two terms remain as the major components of each forwarding table. Fortunately, in SEIZE, summing up the first two terms across all switches always results in a value that is linearly proportional to H, regardless of the physical connectivity and routing topology. This observation confirms the intuition that, in SEIZE, the number of host-information entries stored in a network is $O(H)$, which is much smaller than $O(SH)$ in other schemes.

3) Simple and robust mobility: Flat addressing and relay-based delivery simplify mobility management. Flat addressing liberates names from location, obviating the need to rename a host as it moves. Under relay-based delivery, switches do not need to purge stale host-location information from their caches; only the relay node and the host's adjacent switch need to know the host's location. Since updating the relay is essentially an atomic operation, SEIZE also does not suffer from transient forwarding loops resulting from host mobility.

B. Achieving data-plane efficiency with minimal cost

Path optimization via cut-through forwarding minimizes stretch because frames follow shortest paths from ingress to egress. Cut-through forwarding introduces the following costs: *i)* additional control messages for on-demand resolution (i.e., ingress switches receiving notifications from relays), *ii)* larger forwarding tables at ingress switches to cache the additional host information for ongoing flows, and *iii)* control overhead for updating cached location information when a host moves.

We do not expect these overheads to be significant in practice. Most end-hosts communicate with a small number of popular hosts, such as e-mail/file/Web servers, printers, a gateway router, and perhaps a few hosts for VoIP (Voice over IP) calls. Recent studies on end-host communication patterns confirm that most hosts have a small, nearly-static "community of interest" (COI) [12]. The COI for a mobile host is usually even smaller. Viewing a host's COI as analogous to the "working set" in a conventional caching system, we see that reactive host-location resolution and caching can ensure both scalability and efficiency.

V. Simple and Flexible Network Management

For management purposes, administrators can accurately control the way SEIZE operates. Additionally, SEIZE offers a number of unique benefits, including optimal load balancing and simple and robust access control.

A. Controlling SEIZE for management purposes

The use of hashing and caching in SEIZE offers wide opportunities to engineer a network for various operational needs.

For example, more powerful switches, with more memory and bandwidth, can serve as relay nodes for a disproportionate number of hosts. This can be achieved by introducing a supplementary pre-hash that creates a *set* of identifiers (IDs) for a switch. The size of a switch's ID set determines for how many hosts the switch provides relay service. Each switch learns all the other switches' ID sets (i.e., sets of pre-hash values) via link-state routing and then applies hash function F to the IDs. By choosing a pre-hash function with a large hash space (e.g., 128 bits hash values), we can minimize the probability of ID collision. Even when an ID collision occurs, however, switches can easily detect it and create a new ID because IDs are disseminated via link-state routing.

The trade-offs between relaying and cut-through forwarding can be made adaptively, based on observations of network conditions. For example, switches could disallow cut-through forwarding for inherently short flows (e.g., DNS), or traffic traveling to highly mobile hosts. An ingress switch could forward packets through a relay until a minimum number of packets have been sent or the receiving host remains in the same location for some minimum period of time.

Forwarding-table sizes are also predictable and controllable. As mentioned, pre-hashing can be used to adjust the number of hosts for which a switch provides relay service. The number of end-hosts which will be directly covered by a switch is also usually known ahead of time. Thus, network administrators can furnish a switch with the appropriate amount of resources.

B. Optimal load balancing

Delivering frames via relay switches serves as a form of load balancing, since flows between the same ingress and egress switches follow different paths. Recent work has shown that delivering traffic through multiple, uniformly-selected indirect paths, rather than direct shortest paths, can guarantee 100% throughput for any valid traffic matrix [13]. The study also proved that this model requires minimal capacity between each node and makes performance predictable. SEIZE's relayed delivery mode naturally corresponds to this optimal load balancing model. Moreover, indirect forwarding is especially well-suited to enterprise networks, where the extra latency from indirect delivery is typically small.

C. Simple and robust access control

Correctly enforcing access-control policies is very challenging in conventional networks because routing and access control are managed separately. When using the relayed delivery mode, the relay switches serve as consistent and deterministic *rendezvous points (RPs)* between routing and access control. Since every communication between hosts have to be relayed through a RP, implementing ACLs (Access Control Lists) at the RPs enables access control based only on identifiers. That is, unlike conventional networks, access controls do not need to be updated as routing changes or host locations change. Note, however, that access-control policies, not the implementations (i.e., ACLs) of the policies, can still be managed at a centralized platform. The centralized platform can automatically convert the policies into ACLs and then dispatch them to appropriate RPs using the hash function F.

By modifying the default behavior for an illegitimate relay/resolution attempts, SEIZE can implement more interesting approaches for handling unauthorized (often, malicious) traffic. For example, instead of discarding a non-relayable frame, a relay switch can redirect the frame to a middlebox, such as an IDS (Intrusion Detection System), that can detect or sanitize suspicious traffic. Note, however, that SEIZE itself offers stronger protection against resource exhaustion attacks (e.g., a large number of ARP requests for non-existing addresses, or high volume of data traffic to non-existing hosts) because it avoids flooding and broadcasting.

VI. Related work

The Rbridges [1] architecture is an Ethernet extension that improves path efficiency via shortest-path forwarding, and robustness against loops via TTL-based frame discard. Rbridges ensures optimal stretch by injecting each end-host's address into a link-state protocol, at the expense of poor control-plane scalability. In contrast to Rbridges, SEIZE enables end-to-end layer-two connectivity in an enterprise through scalable control- and data-plane mechanisms.

Myers' architecture [5] replaces flooding with unicast forwarding, and broadcasting with a separate control mechanism. To achieve optimal stretch, host information is injected into a link-state routing protocol, resulting in poor control-plane scalability. The paper also does not describe how the model can support existing broadcast/multicast applications using only unicast forwarding.

There have recently been a number of architectural proposals about networking on flat names. ROFL is one such effort that utilizes a DHT (Distributed Hash Table) to avoid excessive control overhead resulting from flat names [14]. SEIZE is motivated by ROFL but differs from it in the following respects. Most of all, SEIZE allows each switch to maintain a complete network map via link-state routing, whereas each router in ROFL possesses only partial knowledge about the network. Therefore, SEIZE employs shortest-path tunnels between the switches, as opposed to ROFL's source routing. This design removes the overhead to store source routes at each switch and in each packet. Moreover, maintaining a complete topology at each switch enables SEIZE to have smaller stretch because, even without caching of host-location information, paths in SEIZE are at most only two virtual-hops long (i.e., from an ingress to a relay, and then to an egress). In contrast, paths in ROFL are four to nine virtual-hops long when caching is not employed. To reduce path lengths, ROFL also employs caching. However, the cache size at each router increases exponentially as stretch improves.

In a parallel effort, Ray et al. also propose the use of hash-based location resolution for Ethernet networks [15]. This architecture, however, uses a conventional STP to establish paths between switches, and a proprietary identifier dissemination protocol. In contrast, SEIZE employs a link-state protocol, leading to shortest-path forwarding between switches.

VII. Conclusion

SEIZE is a plug-and-playable architecture that ensures scalability and efficiency through hash-based location management, reactive location resolution and caching, and shortest-path forwarding. Locality in traffic patterns ensures data-plane efficiency without sacrificing control-plane scalability.

A prototype implementation of SEIZE is running on Emulab. In our ongoing work, we are modeling and evaluating the strengths of the architecture with measurement traffic from enterprise networks.

Acknowledgments

We would like to thank Matthew Caesar, Sharon Goldberg, Rui Zhang-Shen, Yi Wang, KyoungSoo Park, Eugene Ng, and Kobus van der Merwe for their valuable comments.

References

[1] R. Perlman, "Rbridges: Transparent routing," in *Proc. IEEE INFOCOM*, March 2004.

[2] "IETF TRILL working group," www.ietf.org/html.charters/trill-charter.html.

[3] T. Rodeheffer, C. Thekkath, and D. Anderson, "Smartbridge: A scalable bridge architecture," in *Proc. ACM SIGCOMM*, August 2000.

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

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

[6] L. Andersson and E. Rosen, "Framework for Layer 2 Virtual Private Networks (L2VPNs)," Request for Comments 4664, September 2006.

[7] R. Santitoro, "Metro ethernet services - a technical overview," 2003, www.metroethernetforum.org/metro-ethernet-services.pdf.

[8] M. Casado, T. Garfinkel, A. Akella, M. Freedman, D. Boneh, N. McKeown, and S. Shenker, "Sane: A protection architecture for enterprise networks," in *Proc. Usenix Security*, August 2006.

[9] "IEEE Std 802.1Q - 2005, IEEE Standard for Local and Metropolitan Area Network, Virtual Bridged Local Area Networks," 2005, standards.ieee.org/getieee802/download/802.1Q-2005.pdf.

[10] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," in *Proc. of ACM Symposium on Theory of Computing*, 1997.

[11] R. Droms, "Dynamic Host Configuration Protocol," Request for Comments 2131, March 1997.

[12] W. Aiello, C. Kalmanek, P. McDaniel, S. Sen, O. Spatscheck, and J. van der Merwe, "Analysis of communities of interest in data networks," in *Proc. Passive and Active Measurement*, March 2005.

[13] R. Zhang-Shen and N. McKeown, "Designing a predictable Internet backbone network," in *Proc. HotNets*, November 2004.

[14] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: Routing on Flat Labels," in *Proc. ACM SIGCOMM*, September 2006.

[15] S. Ray, R. A. Guerin, and R. Sofia, "A distributed hash table based address resolution scheme for large-scale ethernet networks," in *Proc. International Conference on Communications*, 2007, to appear.