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

Changhoon Kim and Jennifer Rexford
Princeton University
An “All Ethernet” Enterprise Network?

- “All Ethernet” makes network management easier
  - Zero-configuration of end-hosts and network due to
    - Flat addressing
    - Self-learning
  - Location independent and permanent addresses also simplify
    - Host mobility
    - Troubleshooting
    - Access control

- But, Ethernet has problems
  - Poor scalability
  - Poor efficiency
Today: Hybrid Architecture For Scalability

Enterprise networks comprised of **Ethernet-based IP subnets** interconnected by routers.

**Ethernet Bridging**
- Flat addressing
- Self-learning
- Flooding
- Forwarding along a tree

**IP Routing**
- Hierarchical addressing
- Subnet configuration
- Host configuration
- Forwarding along shortest paths
Motivation

Neither bridging nor routing is satisfactory.
Can’t we take only the best of each?

<table>
<thead>
<tr>
<th>Features</th>
<th>Ethernet Bridging</th>
<th>IP Rout</th>
<th>SEIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of configuration</td>
<td>🌿</td>
<td>🟥</td>
<td>🌿</td>
</tr>
<tr>
<td>Optimality in addressing</td>
<td>🌿</td>
<td>🟥</td>
<td>🌿</td>
</tr>
<tr>
<td>Mobility support</td>
<td>🌿</td>
<td>🟥</td>
<td>🌿</td>
</tr>
<tr>
<td>Path efficiency</td>
<td>🟥</td>
<td>🌿</td>
<td>🌿</td>
</tr>
<tr>
<td>Load distribution</td>
<td>🟥</td>
<td>🌿</td>
<td>🌿</td>
</tr>
<tr>
<td>Convergence speed</td>
<td>🟥</td>
<td>🌿</td>
<td>🌿</td>
</tr>
<tr>
<td>Tolerance to loop</td>
<td>🟥</td>
<td>🌿</td>
<td>🌿</td>
</tr>
</tbody>
</table>

SEIZE (Scalable and Efficient Zero-config Enterprise)
Avoiding Flooding

- Bridging uses flooding as a routing scheme
  - Unicast frames to unknown destinations are flooded

- Does not scale to a large network

- Objective #1: **Unicast unicast traffic**
  - Need a control-plane mechanism to discover and disseminate hosts’ location information
Restraining Broadcasting

- Liberal use of broadcasting for bootstrapping (DHCP and ARP)
  - Broadcasting is a vestige of shared-medium Ethernet
  - Very serious overhead in switched networks

- Objective #2: Support unicast-based bootstrapping
  - Need a directory service

- Sub-objective #2.1: Support general broadcast
  - However, handling broadcast should be more scalable
Keeping Forwarding Tables Small

- Flooding and self-learning lead to unnecessarily large forwarding tables
  - Large tables are not only inefficient, but also dangerous

- Objective #3: Install hosts’ location information only when and where it is needed
  - Need a reactive resolution scheme
  - Enterprise traffic patterns are better-suited to reactive resolution
Ensuring Optimal Forwarding Paths

- Spanning tree avoids broadcast storms. But, forwarding along a single tree is inefficient.
  - Poor load balancing and longer paths
  - Multiple spanning trees are insufficient and expensive

- Objective #4: Utilize shortest paths
  - Need a routing protocol

- Sub-objective #4.1: Prevent broadcast storms
  - Need an alternative measure to prevent broadcast storms
Backwards Compatibility

- **Objective #5:** Do not modify end-hosts
  - From end-hosts’ view, network must work the same way
    - End hosts should
      - Use the same protocol stacks and applications
      - Not be forced to run an additional protocol
Flat addressing of end-hosts
- Switches use hosts’ MAC addresses for routing
- Ensures zero-configuration and backwards-compatibility (Obj # 5)

Automated host discovery at the edge
- Switches detect the arrival/departure of hosts
- Obviates flooding and ensures scalability (Obj #1, 5)

Hash-based on-demand resolution
- Hash deterministically maps a host to a switch
- Switches resolve end-hosts’ location and address via hashing
- Ensures scalability (Obj #1, 2, 3)

Shortest-path forwarding between switches
- Switches run link-state routing with only their own connectivity info
- Ensures data-plane efficiency (Obj #4)
How does it work?

1. **Host discovery or registration**
   - Hash function: $F(x) = B$
   - Notifying $<x, A>$ to $D$

2. **Traffic to $x$**
   - Tunnel to egress node, $A$
   - Tunnel to relay switch, $B$

3. **Entire enterprise (A large single IP subnet)**
   - Optimized forwarding directly from $D$ to $A$

4. **End-hosts**
   - Switches
   - Control flow
   - Data flow

5. **Switching**
   - Deliver to $x$
   - Store $<x, A>$ at $B$
   - Tunnel to relay switch, $B$
   - LS core
   - Traffic to $y$
Ingress applies a cache eviction policy to this entry.

Cut-through forwarding.
Responding to Topology Changes

- Consistent Hash [Karger et al., STOC’97] minimizes re-registration.
Single Hop Look-up

Every switch on a ring is logically one hop away

$y$ sends traffic to $x$
Responding to Host Mobility

When cut-through forwarding is used, the new Dst replaces the old Dst.

Old Dst
Old Device

New Dst
New Device

Relay (for x)

Src
Source

D
Destination

x, G
New address x, new value G

x, A
Old address x, old value A

x, A
New address x, new value A

x, G
Old address x, new value G
Unicast-based Bootstrapping

- **ARP**
  - Ethernet: Broadcast requests
  - **SEIZE**: Hash-based on-demand address resolution
    - Exactly the same mechanism as location resolution
    - Proxy resolution by ingress switches via unicasting

- **DHCP**
  - Ethernet: Broadcast requests and replies
  - **SEIZE**: Utilize DHCP relay agent (RFC 2131)
    - Proxy resolution by ingress switches via unicasting
Control-Plane Scalability When Using Relays

- Minimal overhead for disseminating host-location information
  - Each host’s location is advertised to only two switches

- Small forwarding tables
  - The number of host information entries over all switches leads to $O(H)$, not $O(SH)$

- Simple and robust mobility support
  - When a host moves, updating only its relay suffices
  - No forwarding loop created since update is atomic
Data-Plane Efficiency w/o Compromise

- **Price for path optimization**
  - Additional control messages for on-demand resolution
  - Larger forwarding tables
  - Control overhead for updating stale info of mobile hosts

- **The gain is much bigger than the cost**
  - Because most hosts maintain a small, static communities of interest (COIs) [Aiello et al., PAM’05]
  - Classical analogy: COI ↔ Working Set (WS);
    Caching is effective when a WS is small and static
Conclusions

- SEIZE is a plug-and-playable enterprise architecture ensuring both scalability and efficiency

- Enabling design choices
  - Hash-based location management
  - Reactive location resolution and caching
  - Shortest-path forwarding

- Ongoing work
  - Analysis of enterprise traffic measurements
  - Evaluation of a SEIZE prototype in Emulab
  - Exploring ways to incrementally deploy SEIZE