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

Changhoon Kim, and Jennifer Rexford
http://www.cs.princeton.edu/~chkim
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
    - Network troubleshooting
    - Access-control policies
But, Ethernet bridging does not scale

- Flooding-based delivery
  - Frames to unknown destinations are flooded

- Broadcasting for basic service
  - Bootstrapping relies on broadcasting
  - Vulnerable to resource exhaustion attacks

- Inefficient forwarding paths
  - Loops are fatal due to broadcast storms; use the STP
  - Forwarding along a single tree leads to inefficiency
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>Architectures</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>
<td>✅</td>
</tr>
<tr>
<td>Optimality in addressing</td>
<td></td>
<td>✅</td>
<td>🚫</td>
<td>✅</td>
</tr>
<tr>
<td>Mobility support</td>
<td></td>
<td>✅</td>
<td>🚫</td>
<td>✅</td>
</tr>
<tr>
<td>Path efficiency</td>
<td></td>
<td>🚫</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Load distribution</td>
<td></td>
<td>🚫</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Convergence speed</td>
<td></td>
<td>🚫</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Tolerance to loop</td>
<td></td>
<td>🚫</td>
<td>✅</td>
<td>✅</td>
</tr>
</tbody>
</table>

SEIZE (Scalable and Efficient Zero-config Enterprise)
Overview

- Objectives
- SEIZE architecture
- Evaluation
- Conclusions
Overview: Objectives

- Objectives
  - Avoiding flooding
  - Restraining broadcasting
  - Keeping forwarding tables small
  - Ensuring path efficiency

- SEIZE architecture
- Evaluation
- Conclusions
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
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
Overview: Architecture

- Objectives
- SEIZE architecture
  - Hash-based location management
  - Shortest-path forwarding
  - Responding to network dynamics
- Evaluation
- Conclusions
SEIZE in a Slide

- **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?

Host discovery or registration

Hash \( F(x) = B \)

Tunnel to egress node, \( A \)

Tunnel to relay switch, \( B \)

Store \( <x, A> \) at \( B \)

Entire enterprise (A large single IP subnet)

Optimized forwarding directly from \( D \) to \( A \)

Notifying \( <x, A> \) to \( D \)

Traffic to \( x \)

Hash \( F(x) = B \)

Switches
End-hosts
Control flow
Data flow
Ingress applies a cache eviction policy to this entry.
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.

Old Dst $\langle x, G \rangle$

New Dst $\langle x, G \rangle$

When cut-through forwarding is used.

Old Dst $\langle x, A \rangle$

New Dst $\langle x, A \rangle$

Relay (for $x$) $\langle x, G \rangle$
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
Overview: Evaluation

- Objectives
- SEIZE architecture
- Evaluation
  - Scalability and efficiency
  - Simple and flexible network management
- Conclusions
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
Prototype Implementation

- Link-state routing: eXtensible Open Router Platform [Handley et al., NSDI’05]
- Host information management and traffic forwarding:
  The Click modular router [Kohler et al., TOCS’00]
Evaluation: Set-up and Models

- Simulation; Emulation on Emulab
- Test Network Configuration

- Test Traffic
  - LBNL internal packet traces [Pang et al., IMC’05]
    - 17.8M packets from 5,128 hosts across 22 subnets
  - Real-time replay

- Models tested
  - Ethernet w/ STP, SEIZE w/o path opt., and SEIZE w/ path opt.
  - Inactive timeout-based eviction: 5 min $ltout$, 1 ~ 60 sec $rtout$
Simulations on AP: 315 rtrs, 50K hosts

![Graph showing the relationship between time-out values for ingress caching and fractions of packets requiring location resolution for different metrics: Table size (right axis), Location resolution prob. (left), Ctrl overhead (right).]
Simulations: SmallDC: 4 core + 21 agg
Simulations: Host mobility on campus

Packet loss rate (log)

| Eth  | SEA_CA | SEA_NOCA |

Mobility rate (num. of moving hosts per sec)

0.2, 1, 2, 10, 20, 100, 200
Prototype: Failover performance

Without backup pre-registration

With backup pre-registration

(i)

(ii)
Some Unique Benefits

- **Optimal load balancing via relayed delivery**
  - Flows sharing the same ingress and egress switches are spread over multiple indirect paths
  - For any valid traffic matrix, this practice guarantees 100% throughput with minimal link usage
    - 
    - [Zhang-Shen et al., HotNets’04/IWQoS’05]

- **Simple and robust access control**
  - Enforcing access-control policies at relays makes policy management simple and robust
  - Why? Because routing changes and host mobility do not change policy enforcement points
**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

- **Lessons**
  - Trading a little data-plane efficiency for huge control-plane scalability makes a qualitatively different system.
  - Traffic patterns (small static COIs, and short flow interarrival times) are our friends.
Future Work

- Enriching evaluation
  - Various topologies
  - Dynamic set-ups (topology changes, and host mobility)

- Applying reactive location resolution to other networks
  - There are some routing systems that need to be slimmer

- Generalization
  - How aggressively can we optimize control-plane without losing data-plane efficiency?
Thank you.

Full paper is available at
http://www.cs.princeton.edu/~chkim
Backup Slides
## Overall Comparison

<table>
<thead>
<tr>
<th></th>
<th>Eth-STP</th>
<th>SEIZE/no-opt</th>
<th>SEIZE/opt(10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stretch</td>
<td>100</td>
<td>102</td>
<td>82</td>
</tr>
<tr>
<td>Table Size</td>
<td>100</td>
<td>79</td>
<td>82</td>
</tr>
<tr>
<td># Ctrl Pkts</td>
<td>2</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Data-plane Efficiency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control-plane Scalability</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low Cost</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sensitivity to Cache Eviction Policy

Effect of Cache Entry Timeout

Ratio to Eth-STF

Counts

Timeout Values for Cached Entries (sec)

- stretch (left)
- # control pkts (right)
- table size (right)
Group-based Broadcasting

- SEIZE uses per-group multicast tree
Group-based Access Control

- Relay switches enforce inter-group access policies
- The idea
  - Allow resolution only when the access policy between a resolving host’s group and a resolved host’s group permits access
Simple and Flexible Management

- Using only a number of powerful switches as relays?
  - Yes, a pre-hash can generate a set of identifiers for a switch

- Applying cut-through forwarding selectively?
  - Yes, ingress switches can adaptively decide which policy to use (E.g., no cut-through forwarding for DNS look-ups)

- Controlling (or predicting) a switch’s table size?
  - Yes, pre-hashing can determine the number of hosts for which a switch provides relay service
  - The number of directly connected hosts to a switch is also usually known ahead of time

- Traffic engineering?
  - Yes, adjusting link weights works effectively
Control Overhead
Host Information Replication Factor

Max = NH

2H

RF 2.23

RF 1.76

RF 1.83
Path Efficiency

Number of Packets Forwarded

- Eth-STD: 22.2M
- SEIZE/no-opt: 22.9M
- SEIZE/bpt(10): 17.9M

Optimum: + 27%, + 29%, + 2%
Understanding Traffic Patterns

CDF of Flow Interarrival Time
(Flows identified by 5-tuple and inactive timeout = 1 sec)

- Red: entire net (about 5000 hosts)
- Light blue: 1000 ~ 2000 hosts
- Green: ~100 hosts

Inter-arrival time (sec) vs Probability
Understanding Traffic Patterns - cont’d

Destination Popularity Distribution

- red: entire net (about 5000 hosts)
- blue: 1000 ~ 2000 hosts
- green: ~100 hosts
Evaluation: Prototype Implementation

**XORP**
- Click FEA
- RIBD
- OSPF Daemon

**Click**
- IP Forwarding
- Ring Mgr
- HostInfo Mgr

SeizeSwitch

Link State Advertisements from other switches

Host info. registration and optimization msgs

Data Frames
Prototype: Inside a Click Process

```
FromDevice(em0)  FromDevice(em1)
     |             |
Classifier(...)  Classifier(...)  Classifier(...)
     |             |
Strip(14)        Strip(14)        Strip(14)
     |             |
CheckIPHeader(...)  ProcessIPMisc(...)  ProcessIPMisc(…)
     |             |
LookupIPRoute(...)  ARPQuerier(...)  ARPQuerier(...)
     |             |
ProcessIPMisc(...)  ARPQuerier(...)  Others
     |             |
To Device(em0)  To Device(em1)  To Device(eth0)
```

```
SeizeSwitch(…)
     |
IP Proto SEIZE
     |
Classifier(...)
     |
Strip(20)
     |
to upper layer
     |
To Device(eth0)  To Device(eth1)
```

- From `Device(em0)` and `Device(em1)`
- `Classifier(...)`: ARP or IP
- `Strip(14)`
- `CheckIPHeader(...)`
- `LookupIPRoute(...)`
- `ProcessIPMisc(...)`
- `ARPQuerier(...)`
- `SeizeSwitch(...)`
- `To Device(em0)` and `To Device(em1)`
Inside a SeizeSwitch Element

**EthFrame<srcmac, dstmac> arrives**

- c-hash srcmac, get a relay node rn
- store or update <srcmac, in-port> in host-table
- notify <srcmac, my-IP> to rn

**L2 Control**

- control message?
  - yes: apply to host table
  - no: *

**Layer 2**

- is dstmac me or broadcast?
  - no: *
  - yes: is dstIP me?
    - no: *
    - yes: look up routing table
      - send down to L2

**Layer 3**

- strip Ethernet header

**IP Forwarding**

- is dstIP me?
  - no: *
  - yes: send up to L4
    - send up to L4
      - strip IP header

**L2 Data Forwarding**

- c-hash dstmac, get a relay node rn
- encapsulate with <my-IP, rn>, set proto to SEIZE
- inform ingress of <dstmac, egress-IP>

**EthFrame<srcmac, dstmac> departs**

- is dstmac on host-table?
  - no: *
  - yes: is dstmac local?
    - no: *
    - yes: get egress-IP of dstmac, from host table
      - send out to interface

- encapsulate with <my-IP, egress-IP>, set proto to SEIZE
- send out to interface

- inform ingress of <dstmac, egress-IP>
- send up to L4
Control Plane: Single Hop DHT

1’s LOCAL

C, H

1 Forget L

1’s REMOTE_AUTH

D, F

2 Registers F

1’s LOCAL

E, K, L

3 Registers L
Temporal Traffic Locality

CDF of Flow Interarrival Time

- entire net (about 5000 hosts)
- 1000 ~ 2000 hosts
- ~100 hosts
Spatial Traffic Locality

Destination Popularity Distribution

- entire net (about 5000 hosts)
- 1000 ~ 2000 hosts
- ~100 hosts

frequency vs. destinations (sorted in decreasing order of popularity)
Failover Performance

Time/Sequence Graph

Sequence Num. [KB]

New ST built
SW up
SW down
New ST built

Time (s)

100,000
50,000
50,000
150
650

100,000
50,000
50,000
150

OSPF cnv &
host registration
Relay up
Relay
down
OSPF cnv &
host registration