Ethane: taking control of the enterprise

Martin Casado et al

Giang Nguyen
Motivation

• Enterprise networks are large, and complex, and management is distributed.

• Requires substantial manual configuration. Kerravala (Yankee Group 2002):
  – 62% of network downtime in multi-vendor networks comes from human-error.
  – 80% of IT budget on maintenance and operations.
Motivation (cont)

• Current approaches:
  – Insert middleboxes at network choke points:
    • Problem: traffic might accidentally or is maliciously diverted around the middleboxes
  – Introduce tools/additional protocols/layers:
    • Hide the issue instead of fixing it.
    • Additional complexity (e.g., managing the mgmt tools)
Motivation (cont)

• “How could we change the enterprise network architecture to make it more manageable?”

1. “The network should be governed by policies declared over high-level names.”

2. “Policy should determine the path that packets follow.”

3. “The network should enforce a strong binding between a packet and its origin.”
Ethane design overview

1. Central controller
   - Has a global network policy and topology view.
   - From configured rules, decides whether each flow is allowed and how it is routed.

2. Ethane switches
   - Contains simple flow tables.
   - All packets not from known flows are forwarded to controller for decision on “action.”
   - If allowed, then added to flow table and subsequent packets from same flow are forwarded without consulting controller.

3. Names and policy language
   - All users, hosts, switches, protocols etc have names, that are used when writing rules for the controller.
Example deployment
5 basic activities in an Ethane network

1. Registration:
   - All switches, hosts, and users register with the controller.

2. Bootstrapping:
   - Switches maintain secure channels with controller.
   - Minimum spanning tree (MST) rooted at controller.

3. Authentication:
   - A host joining the network is redirected by switch to the controller for authentication (by MAC) when it does DHCP. Controller records bindings host->IP, IP->MAC, MAC->switch port.
   - User is authenticated (e.g. password) via browser. Controller records binding user->host.
5 basic activities in an Ethane network (cont)

1. Flow setup:
   - UserA initiates connection to userB.
   - Switch1 has no matching entry in flow table -> forwards to controller.
   - If controller accepts, computes path and updates all switches along path.

2. Forwarding:
   - Controller sends packet back to switch1, which forwards it and adds new entry in table to allow subsequent packets from this flow without asking the controller.
Ethane switches

• Simpler than Ethernet switches
  – Doesn’t need to learn addresses, support VLANs, run routing protocols, etc…
  – Flow table orders of magnitude smaller because only contains active flows.
  – Flow (header) matching is exact, not longest prefix.
• 2 common types of flow table entries:
  – Per-flow: allow action.
  – Per-(misbehaving-)host: drop action.
• Other possible actions/services:
  – Multiple queues, controller tells in which to place flow.
  – NAT: by replacing packet headers.
Ethane controller

• Registration:
  – Hosts, users, Switches, protocols, access points (Switch, port pairs) must be registered. Directly, or queried from LDAP etc.

• Authentication:
  – Hosts, users, and Switches must authenticate, (e.g., MAC, password, SSL certs).

• Tracking of bindings:
  – Bindings between users, addresses, and access points are logged.

• Enforcing resource limits:
  – Can direct Switches to rate-limit flows.
  – Can limit number of authentication requests per host per access point.
  – More possibilities.
Ethane controller (cont)

• Fault tolerance:
  – Cold standby: secondary controllers participate in same global MST.
    • After primary controller goes down, will take over when MST converges.
    • Simple, but slow recovery: hosts/users have to re-authenticate.
  – Warm standby: a separate MST for each secondary controller.
    • Controllers monitor one another’s liveness.
    • Bindings are replicated across controllers.
    • Complex, but faster recovery.

• Fault tolerance and scalability:
  – Multiple active controllers:
    • Switches need to authenticate with only one controller.
    • Spread flow decision queries across multiple controllers.
    • Complex consistency issues etc.
Multicast and broadcast traffic

• In theory:
  – Switch: keeps for each flow a bitmap of ports to forward.
  – Controller: from computed broad/multicast tree, assigns appropriate bits during path setup.
  – Broadcast are mostly discovery protocols, e.g. ARP, which the controller can reply without creating a new flow or broadcasting.

• In practice:
  – ARP causes a significant load on the controller.
  – Might setup a dedicated ARP server, and controller directs ARP traffic there.
  – But what about other disc protocols? Tradeoff: controller implements common protocols, and broadcasts unknown ones with rate-limit.
  – Doesn’t scale well, but expecting discovery protocols to go away if Ethane is used widely.
Rules

• Network policy is a set of rules:
  – \([<condition(s)>]:action;\)
  – Conditions: conjunction of predicates.
  – Actions: allow, deny, waypoints (list of entities to route the flow through), and outbound-only.
  – Example: \([(usrc=“bob”)\land(protocol=“http”)\land(hdst=“websrv”)]:allow;\)
  – Means if the user initiating the flow is bob and the flow protocol is http and the destination is host “websrv”, then allow the flow.
  – Rules are independent. First rule that matches is used.

• Rule lookups have to be fast.
  – Can’t simply compile because of huge namespace of users, hosts, etc
  – So use compilation plus just-in-time creation of search functions.
Prototype

• Switches:
  – Wireless access points using WRTSL54GS.
  – 4-port gigabit switches using FPGA.
  – 4-port gigabit switches using desktop PCs.

• Controller:
  – Standard desktop PC.
Deployment

- 100Mb/s network
- 11 wired and 8 wireless Switches.
- ~300 hosts
- Create a network policy that matches existing firewall configs, NATs, router ACLs etc.

- Hosts connected to an Ethane switch port does not require user authentication.
Evaluation: controller scalability

Figure 6: Flow-setup times as a function of Controller load. Packet sizes were 64B, 128B and 256B, evenly distributed.

• A 22,000-host network observed max 9,000 flow requests per second, suggesting that a single controller can handle 20,000 hosts with flow request setup time under 1.5ms.
Evaluation: effect of failures

<table>
<thead>
<tr>
<th>Failures</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completion time</td>
<td>26.17s</td>
<td>27.44s</td>
<td>30.45s</td>
<td>36.00s</td>
<td>43.09s</td>
</tr>
</tbody>
</table>

Table 1: Completion time for HTTP GETs of 275 files during which the primary Controller fails zero or more times. Results are averaged over 5 runs.

Figure 10: Round-trip latencies experienced by packets through a diamond topology during link failure.
Shortcomings

- Broadcast and discovery protocols.
- Application-layer routing: hostA not allowed to talk to hostC, so hostB can relay hostA’s messages.
- Tunneling other protocols in http.
- Spoofing Ethernet MACs.
  - Physically allow only one host per switch port.
  - Or use 802.1X plus link-level encryption such as 802.1AE.