Announcements
- Assign #5 released -- due Thursday, May 7 at 11pm

Outline
- Recap
- SDN Stack
- Layer 2 Learning Switch
- Control Application Design Considerations
- Challenges and Ongoing Research

Recap: SDN Motivation
- **What were the key limitations of traditional networks?**
  - Distributed routing decisions -> difficult to debug; maybe suboptimal decisions
  - Distributed policy -> prone to inconsistencies
  - Policies defined over subnets -> cannot customize forwarding per application
  - Closed software stack -> limited flexibility
- **How does SDN address this issues?**
  - Centralized control
    - Global view to make optimal decisions
    - Theoretically, avoid the complexities of distributed algorithms
  - Operator-provided control applications
    - Can control forwarding however you want -- e.g., implement IETF April Fool’s RFC for “green routing”
  - Flexible forwarding decisions
    - Match on 10 packet header fields (arbitrary byte matching in newest OpenFlow)
    - Forward, drop, modify, or send packets to the controller (for arbitrary processing)
- **Problems with SDN?**
  - Central controller can become bottleneck, and is single point of failure

OpenFlow Flow Table
- Traditional route table

<table>
<thead>
<tr>
<th>Subnet</th>
<th>Mask</th>
<th>Gateway</th>
<th>Interface</th>
</tr>
</thead>
</table>

- Flow table

<table>
<thead>
<tr>
<th>Iface</th>
<th>Src MAC</th>
<th>Dst MAC</th>
<th>VLAN</th>
<th>Ether Type</th>
<th>Src IP</th>
<th>Dst IP</th>
<th>Proto</th>
<th>Src Port</th>
<th>Dst Port</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Forward to port(s), drop, rewrite header field(s), send to controller</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Rules can have timeouts</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Soft timeout -- switch removes rule if no packet matches it for some period of time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Hard timeout -- switch removes rule after some period has elapsed since it was added</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Rules can have priorities</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Switch also maintains statistics (packet and byte counters) for each rule</td>
</tr>
</tbody>
</table>
- **What rule should we install if we want to…**
  - Forward all HTTP traffic out iface 3?
  - Forward SSH traffic from a host with IP 10.0.0.5 out iface 4?
  - Do network address translation (NAT)?
  - Forward HTTP GET requests out iface 1?
- Packet sent to controller if no forwarding entry matches
  - Controller passes packet to control app(s)
  - Control app decides whether to:
    - Add a new rule(s) in one or more switches
    - Modify or drop packet
    - Have switch send packet out specific iface
- Switch also sends messages to controller when:
  - Switch starts
  - Switch port goes up/down
  - Receives a request for statistics

**SDN Stack**

```
Application (Routing, Firewall
Load balancing, etc.)

Optional
Language syntax
Languages: Frenetic, Pyretic, FlowLog, etc.

Controller (Floodlight, POX, OpenDaylight, Ryu, etc.)

Optional
OpenFlow protocol
Network slicing: FlowVisor

Switch: Open vSwitch, Cisco, HP, Brocade, etc.

Flow Table
```

Controller API
Layer 2 Forwarding
- Hello world of SDN apps
- **What options do we have?**
  - Flood
  - Learn about locations of hosts
  - Look at network topology and find the path
    - A traditional L2 switch cannot do this, but we can do this with SDN
- **What is the pseudo-code for flooding?**
  - When get packet from switch, send it back to the switch and tell it to flood
  - Or, install a rule for each input port to flood any packets
- **What is the pseudo-code for learning?**
  - if (source mac address is new)
    - record the source mac and input port mapping
  - if (destination mac address is known)
    - install a flow table rule
    - forward the packet to the destination
  - else
    - FLOOD the packet
- Look at network topology and find the path -- this is what you will do for assign #5

Motivating Applications
- Dynamic access control
  - Inspect first packet of a connection
  - Consult the access control policy
  - Install rules to block or route traffic
- Seamless mobility
  - See host send traffic at new location
  - Modify rules to reroute the traffic
Server load balancing
- Pre-install load-balancing policy
- Split traffic based on source IP

Google uses SDN in their WAN

Control App Design Considerations
- Per-flow vs. coarser rules
  - Per-flow (flow=TCP connection) rules allow fine-grained control of network traffic
    - E.g., network address translation for each flow
    - E.g., round robin load balancing of flows across links and/or servers
  - Requires as many flow entries as flows!
    - Border of college of engineering network has 300 new flows per second ⇒ flow table of 1500 entries (capacity of first generation OpenFlow switches) is exhausted in about 5 seconds
    - Combat flow table size limits with very short timeouts
  - Coarser rules require less resources at switches, but are less flexible
    - Per src/dst host pair, per dst host, per dst subnet, per dst TCP port, etc.
Proactive vs. reactive rule installation

- Proactive installs rules ahead of time
  - Usually based on observing the topology
  - E.g., install paths between all switches when hosts start

- Reactive installs rules when packets arrive
  - Usually based on the first packet of every flow (flow = TCP connection)
  - E.g., install paths for specific flow between hosts when flow starts
Challenges and Ongoing Research

Controller Delay and Overhead
- Controller is much slower than the switch
- Processing packets leads to delay and overhead
- Need to keep most packets in the “fast path”

Testing and Debugging
- OpenFlow makes programming possible
  - Network-wide view at controller
  - Direct control over data plane
- Plenty of room for bugs
  - Still a complex, distributed system
- Need for testing techniques
  - Controller applications
  - Controller and switches
  - Rules installed in the switches

Programming Abstractions
- Controller APIs are low-level
  - Thin veneer on the underlying hardware
- Need better languages
  - Composition of modules
  - Managing concurrency
  - Querying network state
  - Network-wide abstractions
- Ongoing at Princeton
  - http://www.frenetic-lang.org/