Dr. Nick Feamster
Associate Professor

Software Defined Networking

In this course, you will learn about software defined networking and how it is changing the way communications networks are managed, maintained, and secured.

School of Computer Science

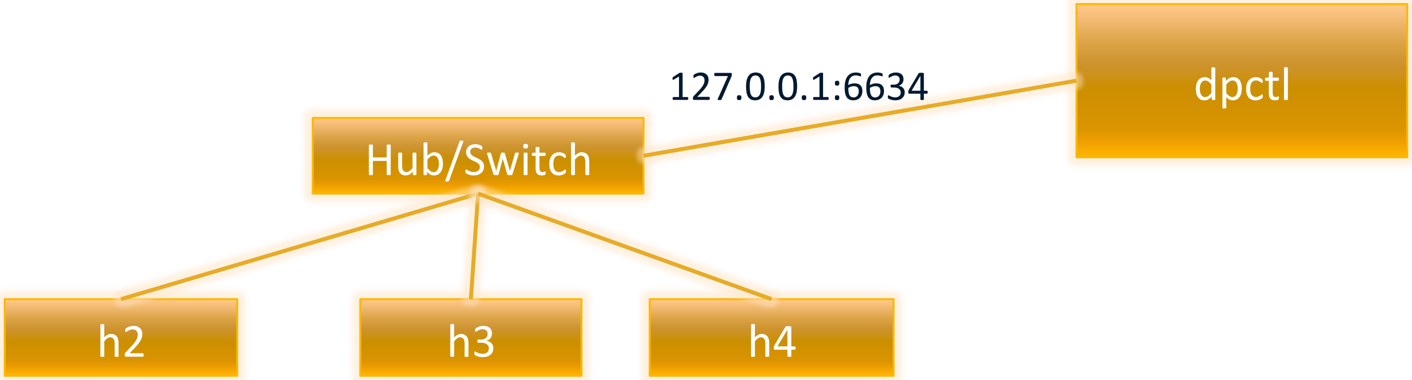
Module 4.3: The Control Plane

- Three Lessons
  - Control Plane Basics (OpenFlow 1.0 and Beyond)
  - SDN Controllers
  - **Using SDN Controllers to Customize Control**
- Programming Assignment (and Quiz)
- Quiz

This Lesson: Customizing Control

- Review of hub and switch
- POX Controller and simple Mininet topology
- Two types of control
  - Hub
  - Learning switch
- Looking at flow tables with dpctl
- Code walkthrough

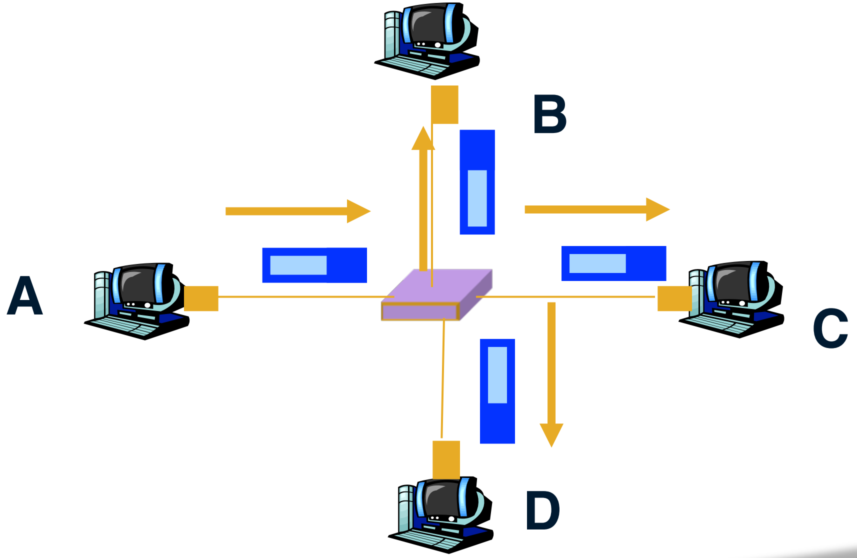
Example Topology



- `$ sudo mn --topo single,3 --mac --switch ovsk --controller remote`
- `dpctl` to communicate with switches
  - Switches listen on port 6634
  - Can inspect flow table entries, modify flows, etc.

Review: Hub

- No forwarding information stored at switch
- Every input packet is flooded out all ports



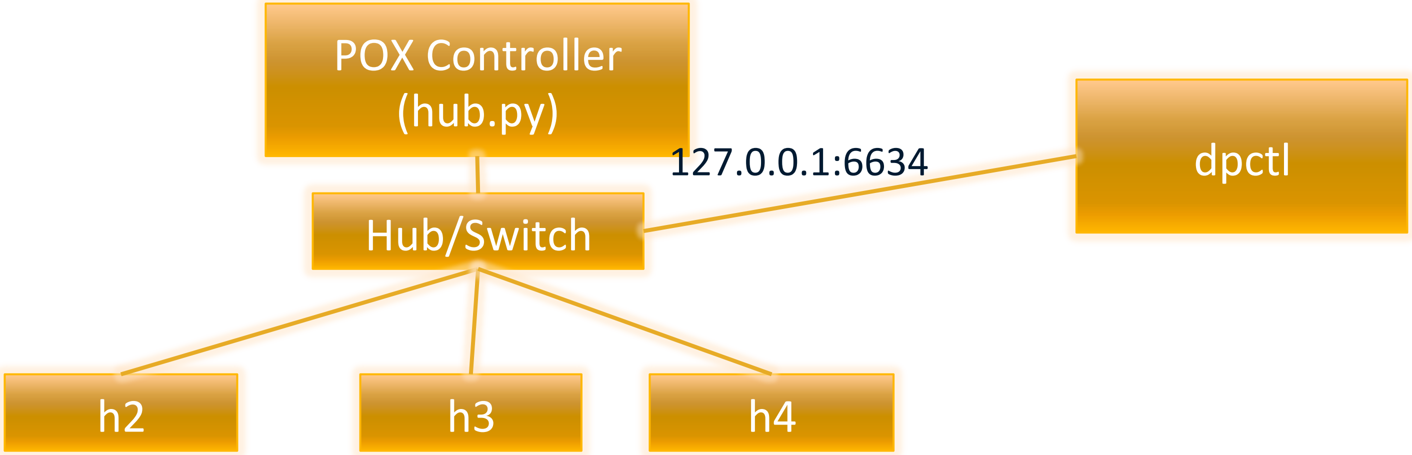
POX Hub

```
def _handle_ConnectionUp (event):
  msg = of.ofp_flow_mod()
  msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))
  event.connection.send(msg)
  log.info("Hubifying %s", dpidToStr(event.dpid))

def launch ():
  core.openflow.addListenerByName("ConnectionUp", _handle_ConnectionUp)
  log.info("Hub running.")
```

Code from hub.py POX distribution (James McCauley)

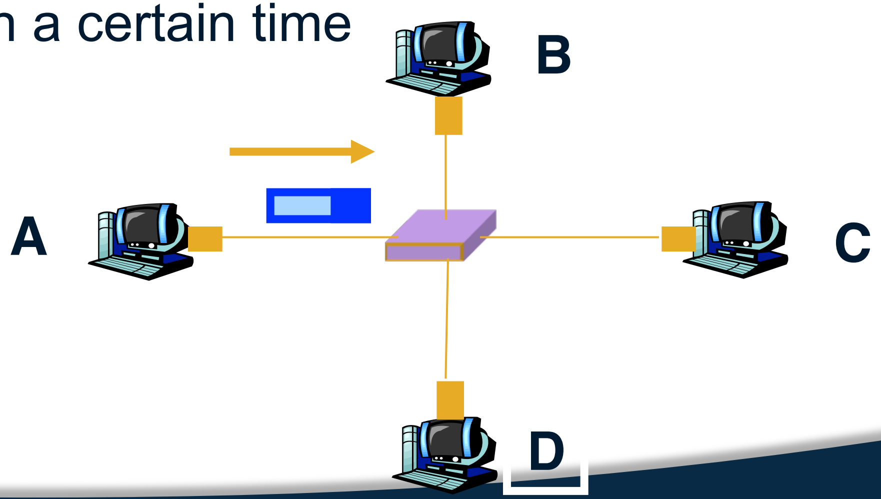
Example Topology



- `$ sudo mn --topo single,3 --mac --switch ovsk --controller remote`
- `dpctl` to communicate with switches
  - Switches listen on port 6634
  - Can inspect flow table entries, modify flows, etc.

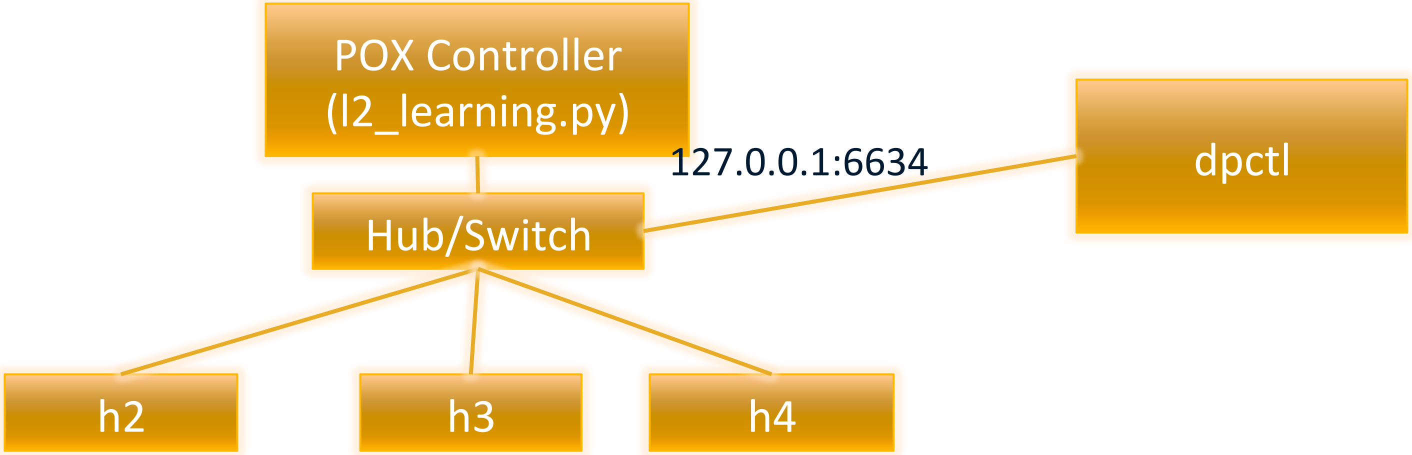
Review: Learning Switch

- Switch table is initially empty
- For each incoming frame, store
  - The incoming interface from which the frame arrived
  - The time at which that frame arrived
  - Delete the entry if no frames with a particular source address arrive within a certain time



Switch learns how to reach A.

Example Topology



- `$ sudo mn --topo single,3 --mac --switch ovsk --controller remote`
- `dpctl` to communicate with switches
  - Switches listen on port 6634
  - Can inspect flow table entries, modify flows, etc.

POX Learning Switch Algorithm

- Use source address and switch port to update address/port table
- Is transparent = False and either Ethertype is LLDP or the packet’s destination address is a Bridge Filtered address? If yes, DROP
- Is destination multicast? If so, FLOOD.
- Is port for destination address in our address/port table? If not, FLOOD.
- Is output port the same as input port? If yes, DROP
- Install flow table entry in the switch so that this flow goes out the appropriate port. Send the packet out appropriate port.

Important Concept: Listeners

- `connection.addListeners(self)` ensures that the controller will hear PacketIn messages

- `_handle_PacketIn` works all of the magic for handling a packet that arrives at the controller

Important Concept: Flow Mods

- Must define a **match** and **action**
- Must send the message to the switch
- Timeouts define how long a flow table entry remains in the table

```
msg = of.ofp_flow_mod()
msg.match = of.ofp_match.from_packet(packet, event.port)
msg.idle_timeout = 10
msg.hard_timeout = 30
msg.actions.append(of.ofp_action_output(port = port))
msg.data = event.ofp
self.connection.send(msg)
```

Summary

- Review of hub and switch
- POX Controller and simple Mininet topology
- Two types of control
  - Hub
  - Learning switch
- Looking at flow tables with dpctl
- Code walkthrough