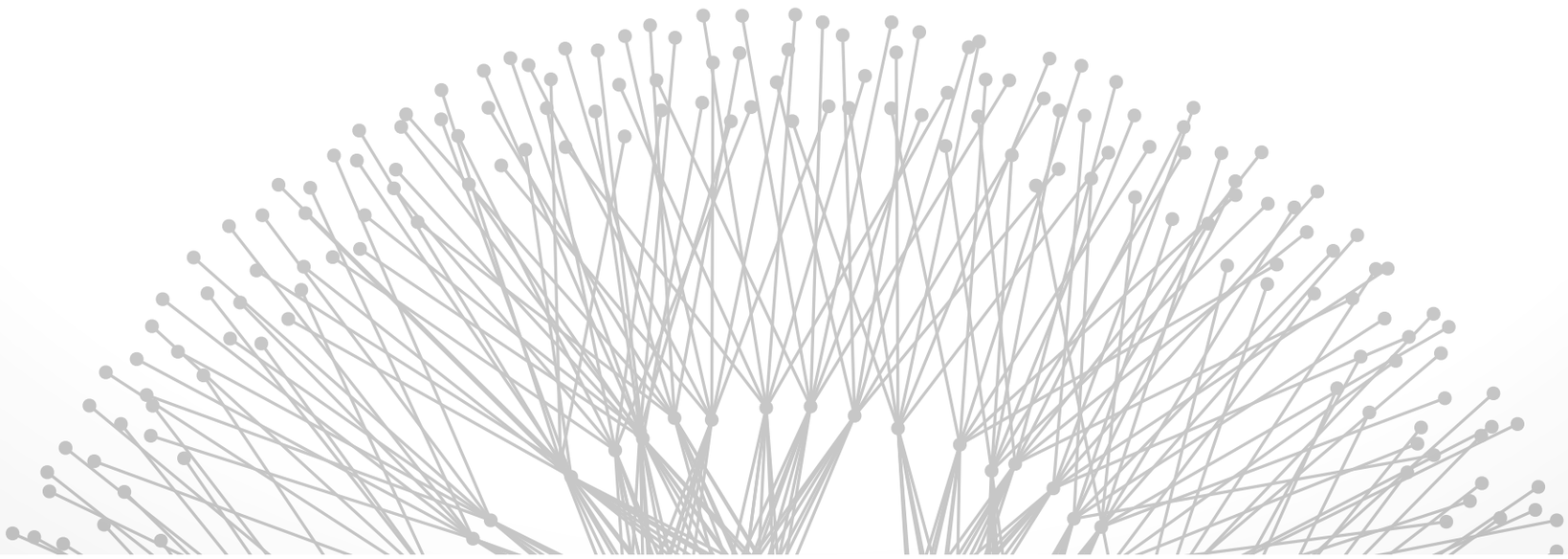
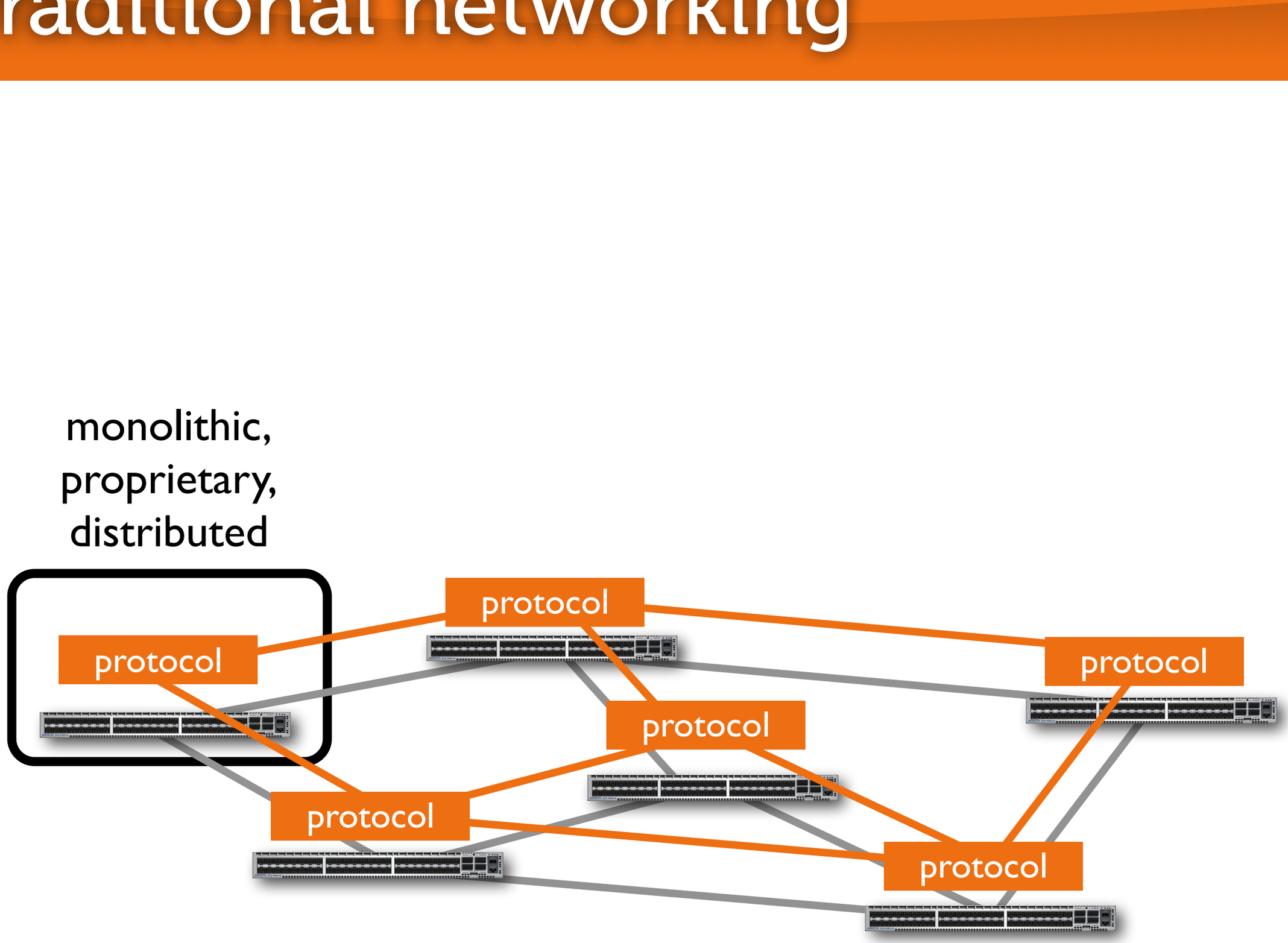
SDN Control Frameworks

Brighten Godfrey
CS 538 December 5 2013



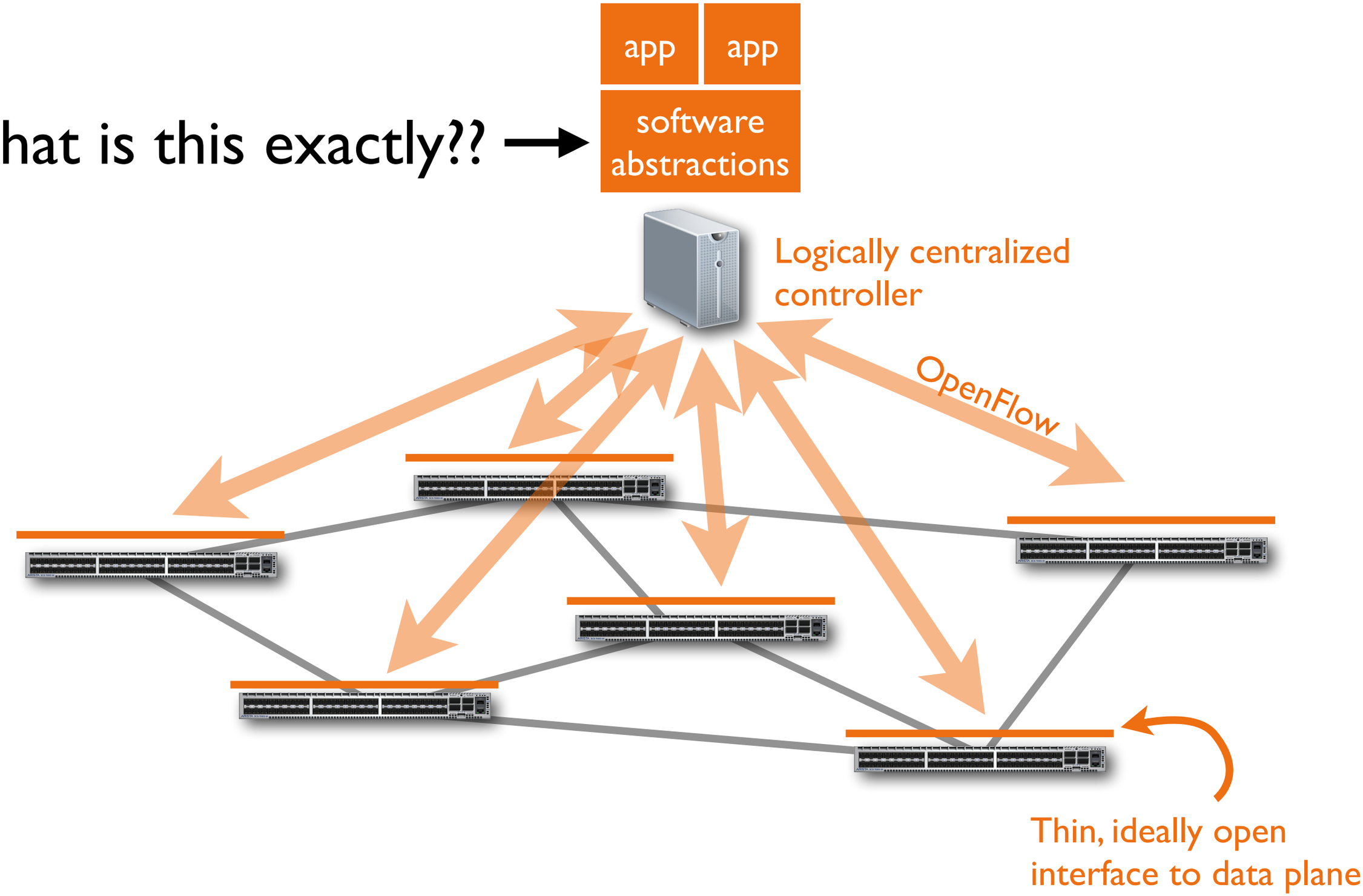
Traditional networking

monolithic,
proprietary,
distributed



Software Defined Networking

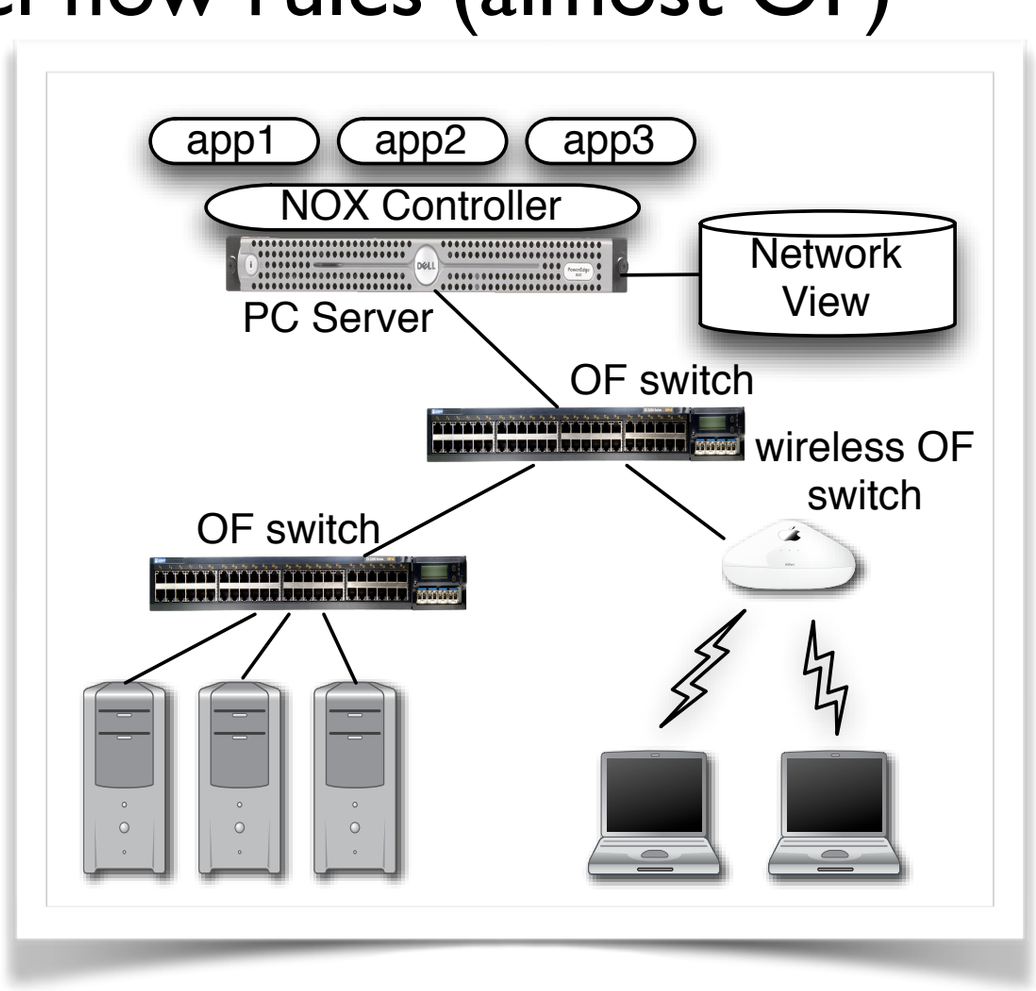
What is this exactly?? →



Early controllers

NOX [Gude, Koponen, Pettit, Pfaff, Casado, McKeown, Shenker, CCR 2008]

- First OF controller: centralized network view provided to multiple control apps as a database
- Behind the scenes, handles state collection & distribution
- Control "language" is low-level flow rules (almost OF)



NOX code example (from paper)

```
# On user authentication, statically setup VLAN tagging
# rules at the user's first hop switch
def setup_user_vlan(dp, user, port, host):
    vlanid = user_to_vlan_function(user)
    # For packets from the user, add a VLAN tag
    attr_out[IN_PORT] = port
    attr_out[DL_SRC] = nox.reverse_resolve(host).mac
    action_out = [(nox.OUTPUT, (0, nox.FLOOD)),
        (nox.ADD_VLAN, (vlanid))]
    install_datapath_flow(dp, attr_out, action_out)
    # For packets to the user with the VLAN tag, remove it
    attr_in[DL_DST] =      nox.reverse_resolve(host).mac
    attr_in[DL_VLAN] = vlanid
    action_in = [(nox.OUTPUT, (0, nox.FLOOD)), (nox.DEL_VLAN)]
    install_datapath_flow(dp, attr_in, action_in)
nox.register_for_user_authentication(setup_user_vlan)
```

Match specific set of packets

NOX code example (from paper)

```
# On user authentication, statically setup VLAN tagging
# rules at the user's first hop switch
def setup_user_vlan(dp, user, port, host):
    vlanid = user_to_vlan_function(user)
    # For packets from the user, add a VLAN tag
    attr_out[IN_PORT] = port
    attr_out[DL_SRC] = nox.reverse_resolve(host).mac
    action_out = [(nox.OUTPUT, (0, nox.FLOOD)),
        (nox.ADD_VLAN, (vlanid))]
    install_datapath_flow(dp, attr_out, action_out)
    # For packets to the user with the VLAN tag, remove it
    attr_in[DL_DST] =      nox.reverse_resolve(host).mac
    attr_in[DL_VLAN] = vlanid
    action_in = [(nox.OUTPUT, (0, nox.FLOOD)), (nox.DEL_VLAN)]
    install_datapath_flow(dp, attr_in, action_in)
nox.register_for_user_authentication(setup_user_vlan)
```

Match specific set of packets

Construct action

NOX code example (from paper)

```
# On user authentication, statically setup VLAN tagging
# rules at the user's first hop switch
def setup_user_vlan(dp, user, port, host):
    vlanid = user_to_vlan_function(user)
    # For packets from the user, add a VLAN tag
    attr_out[IN_PORT] = port
    attr_out[DL_SRC] = nox.reverse_resolve(host).mac
    action_out = [(nox.OUTPUT, (0, nox.FLOOD)),
        (nox.ADD_VLAN, (vlanid))]
    install_datapath_flow(dp, attr_out, action_out)
    # For packets to the user with the VLAN tag, remove it
    attr_in[DL_DST] =     nox.reverse_resolve(host).mac
    attr_in[DL_VLAN] = vlanid
    action_in = [(nox.OUTPUT, (0, nox.FLOOD)), (nox.DEL_VLAN)]
    install_datapath_flow(dp, attr_in, action_in)
nox.register_for_user_authentication(setup_user_vlan)
```

Match specific set of packets

Construct action

Apply flow entry to specific switch

Pyretic: higher-level abstractions

Composing SDNs [Monsanto, Reich, Foster, Rexford, Walker, NSDI 2013]

Key idea: modularize software

- Write specific functionality in each module without worrying about others
- Describe high-level composition of modules
- Pyretic takes care of the details

Pyretic abstractions

Sequential composition (x >> y)

Parallel composition (x | y)

Monitor
srcip=5.6.7.8 → count

|

Route
dstip=10.0.0.1 → fwd(1)
dstip=10.0.0.2 → fwd(2)

=

Compiled Prioritized Rule Set for "Monitor | Route"
srcip=5.6.7.8,dstip=10.0.0.1 → count,fwd(1)
srcip=5.6.7.8,dstip=10.0.0.2 → count,fwd(2)
srcip=5.6.7.8 → count
dstip=10.0.0.1 → fwd(1)
dstip=10.0.0.2 → fwd(2)

Key points: (1) modularize functionality;
(2) programmer avoids combinatorial explosion

Pyretic abstractions

Sequential composition (x >> y)

Parallel composition (x | y)

Topology virtualization

- e.g., network could be “one big switch”
- or a single switch could be virtualized to many

Virtual packet headers

- Modules can annotate packets
- Pass information between modules, carry in packets
- Example of use?

Pyretic discussion from reviews

Performance and scalability

- What challenges would Pyretic face?

What happens when the network changes?

- Need to adjust policies automatically
- Consistent updates

Language features

Discussion

What do you want the controller to do for you?

Verification

Announcements

Last class Tuesday — please do come!

Paper due next Thu Dec 12

- See syllabus for guidelines
- Submit by email to Brighten by 11:59 pm
- PDF format only

Poster Session Dec 17, 1:30 - 4:30