centralized control — separating data- and control- planes

5590: software defined networking

anduo wang, Temple University

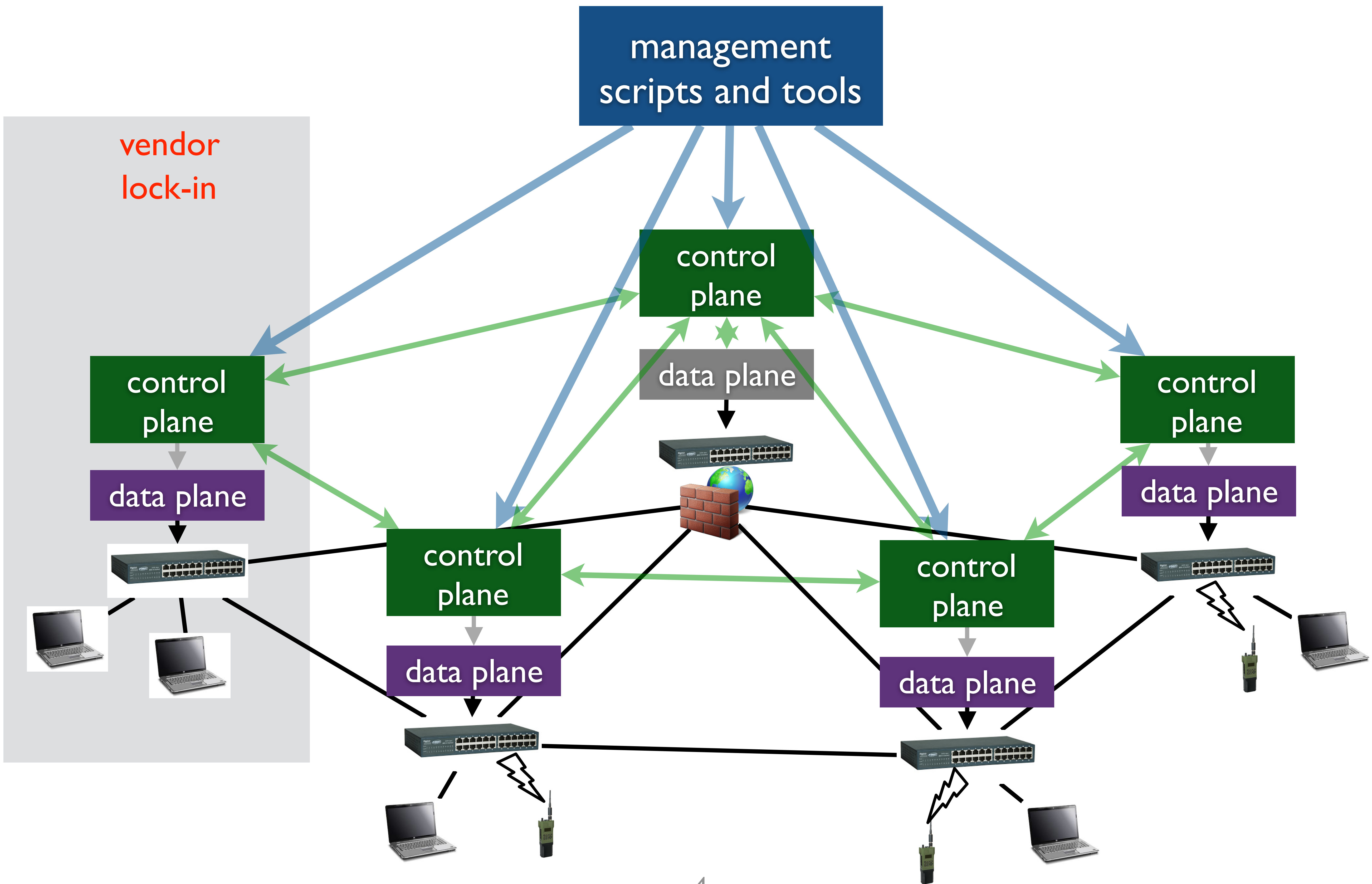
T 17:30-20:00

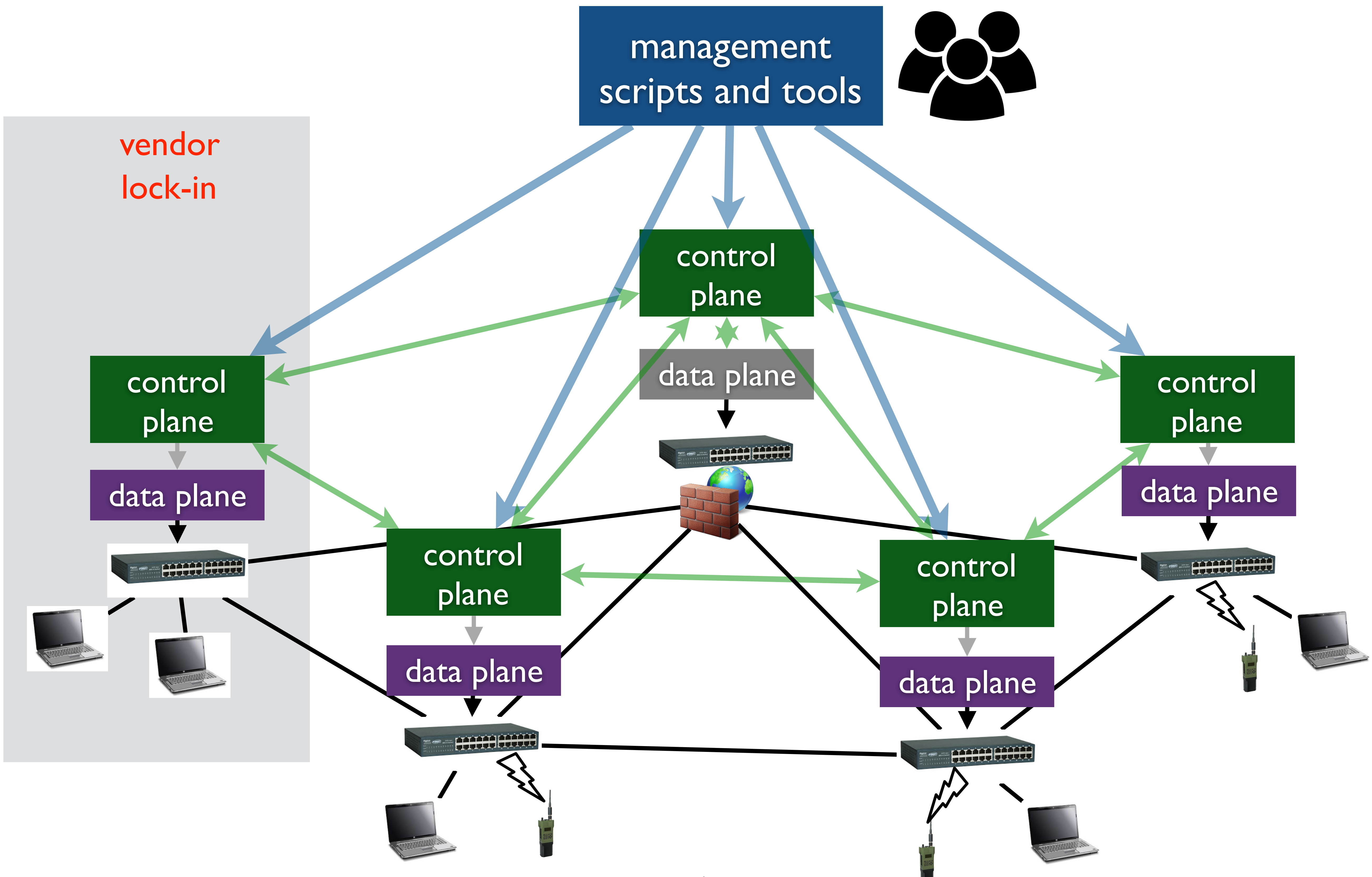
some materials in this slide are based on lectures by

Jennifer Rexford https://www.cs.princeton.edu/courses/archive/fall13/cos597E/

Nick Feamster http://noise.gatech.edu/classes/cs8803sdn/fall2014/

data, control, and management planes


management scripts and tools
vendor lock-in
control plane
data plane
control plane
data plane
control plane
data plane
control plane
data plane
control plane
data plane


management scripts and tools
vendor lock-in
control plane
data plane
control plane
data plane
control plane
data plane
control plane
data plane
control plane
data plane

management plane

defines network composition, control plane configuration, and monitoring schemes
example: CLI, scripts

control plane

generates forwarding tables and filters for the data plane
example: distributed routing protocols

data plane

handles packets
example: forwarding

management plane

defines network composition, control plane configuration, and monitoring schemes
example: CLI, scripts

control plane

generates forwarding tables and filters for the data plane
example: distributed routing protocols

data plane

handles packets
example: forwarding

management plane

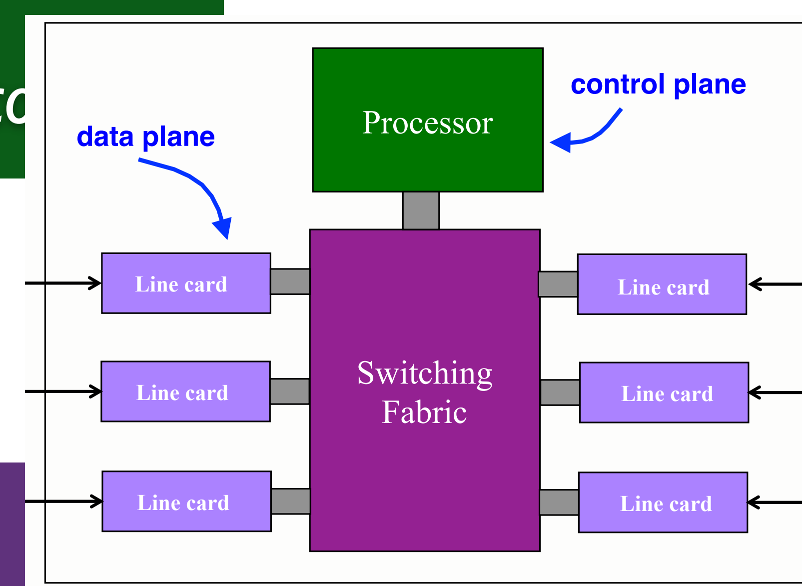
defines network composition, control plane configuration, and monitoring schemes
example: CLI, scripts

control plane

generates forwarding tables and filters for the data plane
example: distributed routing protoco

data plane

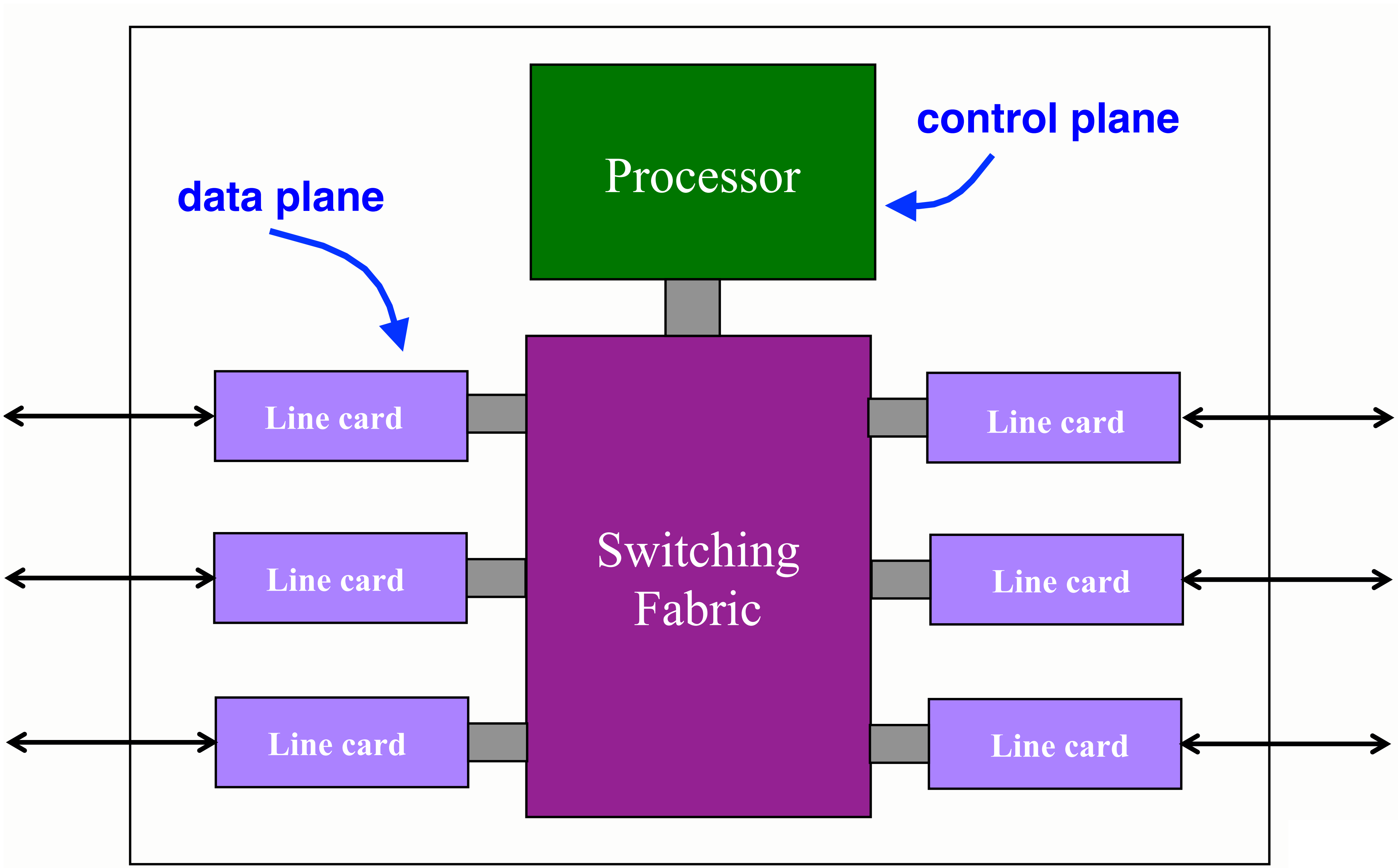
handles packets
example: forwarding



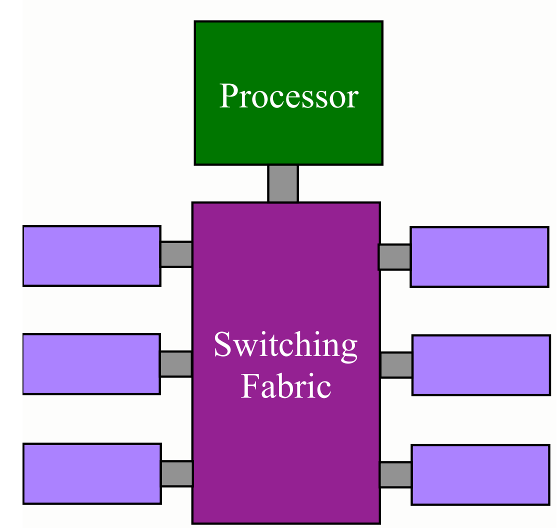
timescales

	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Tasks	Forwarding, buffering, filtering, scheduling	Routing, circuit set-up	Analysis, configuration
Location	Line-card hardware	Router software	Humans or scripts

data and control planes



data plane



streaming algorithms on packets

- matching on some bits
- perform some actions

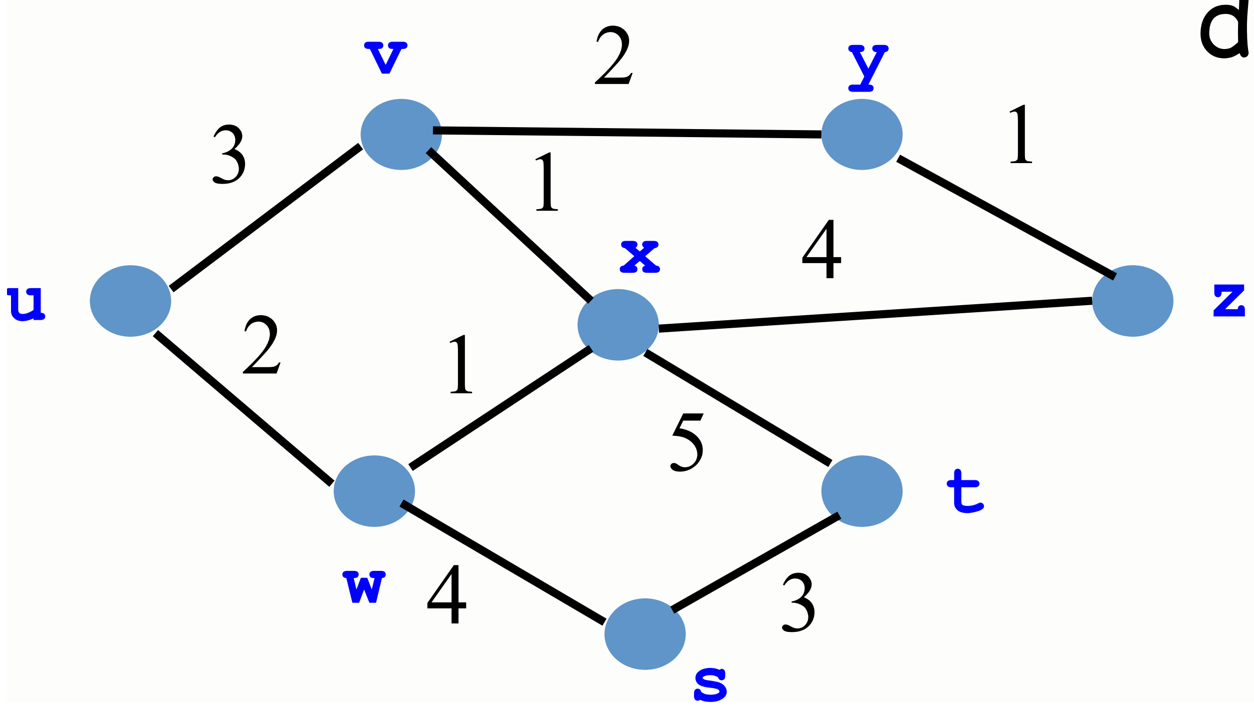
wide range of functionality

- forwarding
- access control
- traffic monitoring
- packet inspection

distributed control plane

example: distance-vector routing: RIP

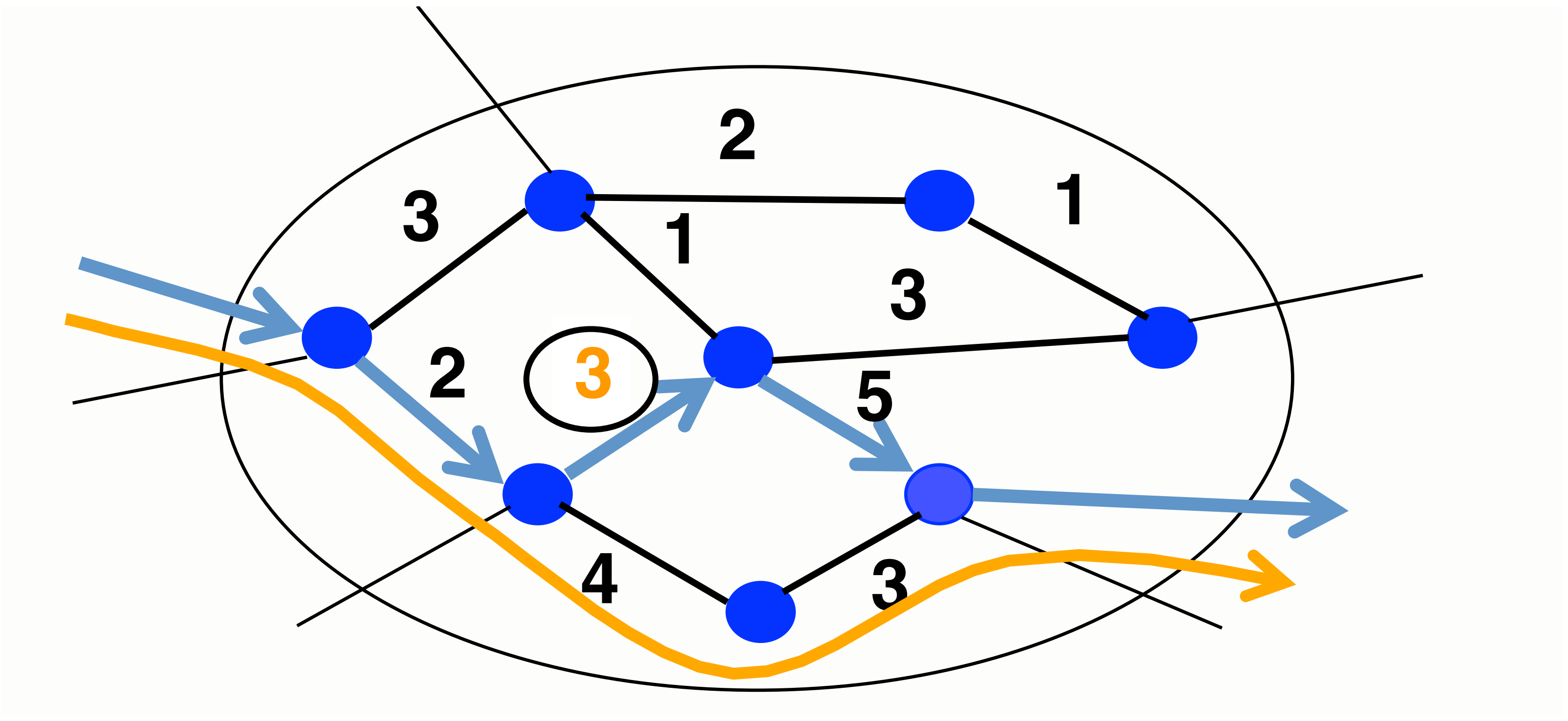
- each node computes path cost
  - …based on neighbor's path cost
  - Bellman-Ford algorithm



$$d_u(z) = \min\{c(u,v) + d_v(z),\; c(u,w) + d_w(z)\}$$

management plane

example: set weights for traffic engineering



management plane

Aaron Gember-Jacobson., et al. "Management Plane Analytics" IMC 2015

management plane

diverse management practice

- design practice
  - set physical network composition (heterogeneity), logical structure (spanning tree)
- operation practice
  - change network for diverse purposes (router, middle-box)
- *tedious, error-prone*

Aaron Gember-Jacobson., et al. "Management Plane Analytics" IMC 2015

management plane

diverse management practice

- design practice
  - set physical network composition (heterogeneity), logical structure (spanning tree)
- operation practice
  - change network for diverse purposes (router, middle-box)
- *tedious, error-prone*

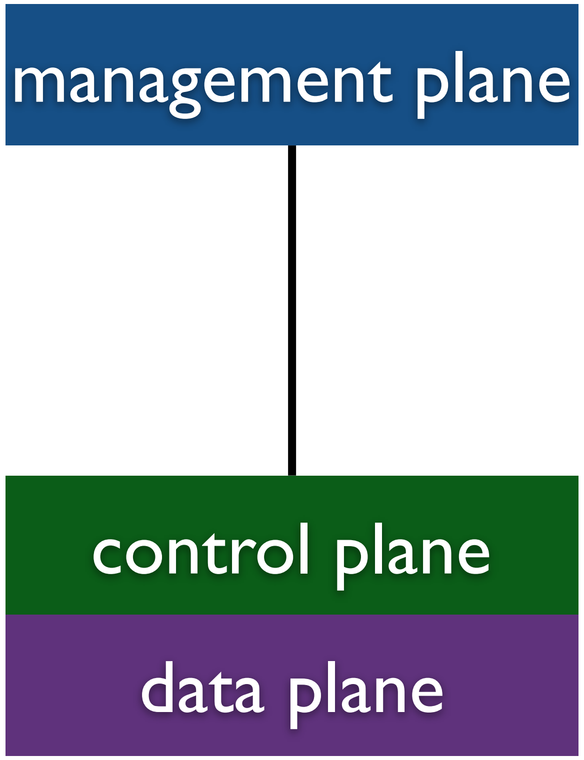
lacking principled understanding of management practice

- how practice impacts network health (performance, availability)?

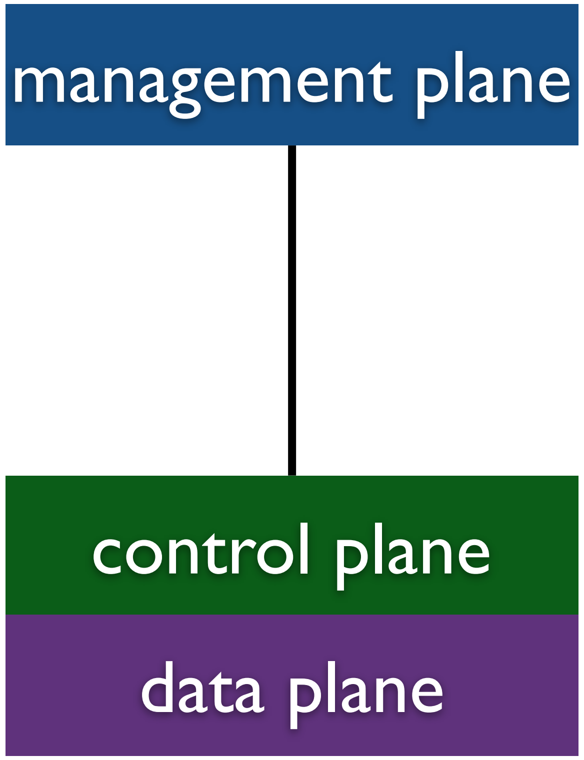
Aaron Gember-Jacobson., et al. "Management Plane Analytics" IMC 2015

network management today: mastering complexity

complexity



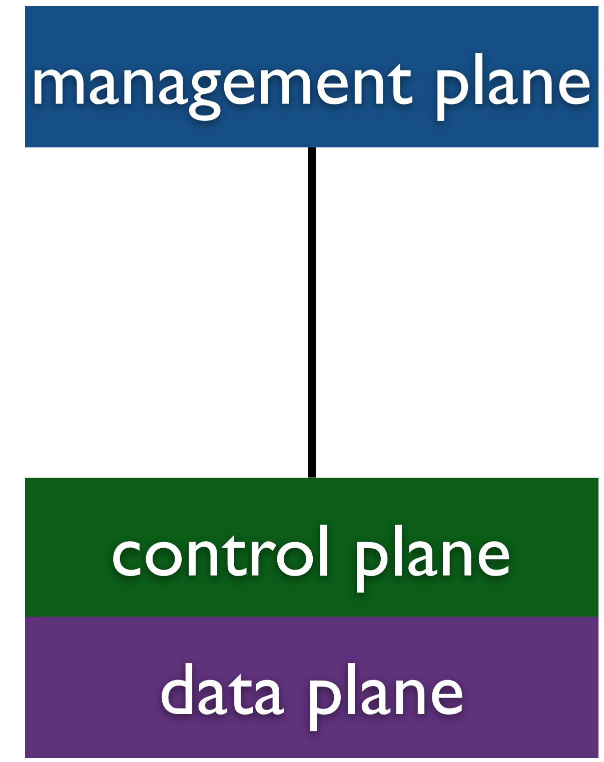
complexity



control logic and packet handling

- bundled in distributed switching element
- management objectives implicitly *embedded*

complexity



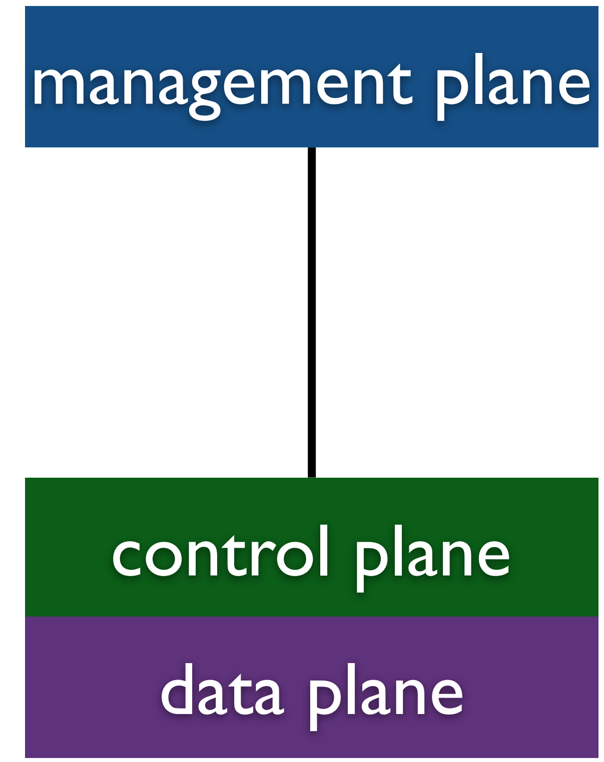
control logic and packet handling

- bundled in distributed switching element
- management objectives implicitly *embedded*

tension

- ever-evolving management requirement
- incremental point solutions to control plane, and complex management tools "coax" the control plane

complexity



control logic and packet handling

- bundled in distributed switching element
- management objectives implicitly *embedded*

tension

- ever-evolving management requirement
- incremental point solutions to control plane, and complex management tools "coax" the control plane

challenge

- indirect, coordinated control
- interacting protocols and mechanisms

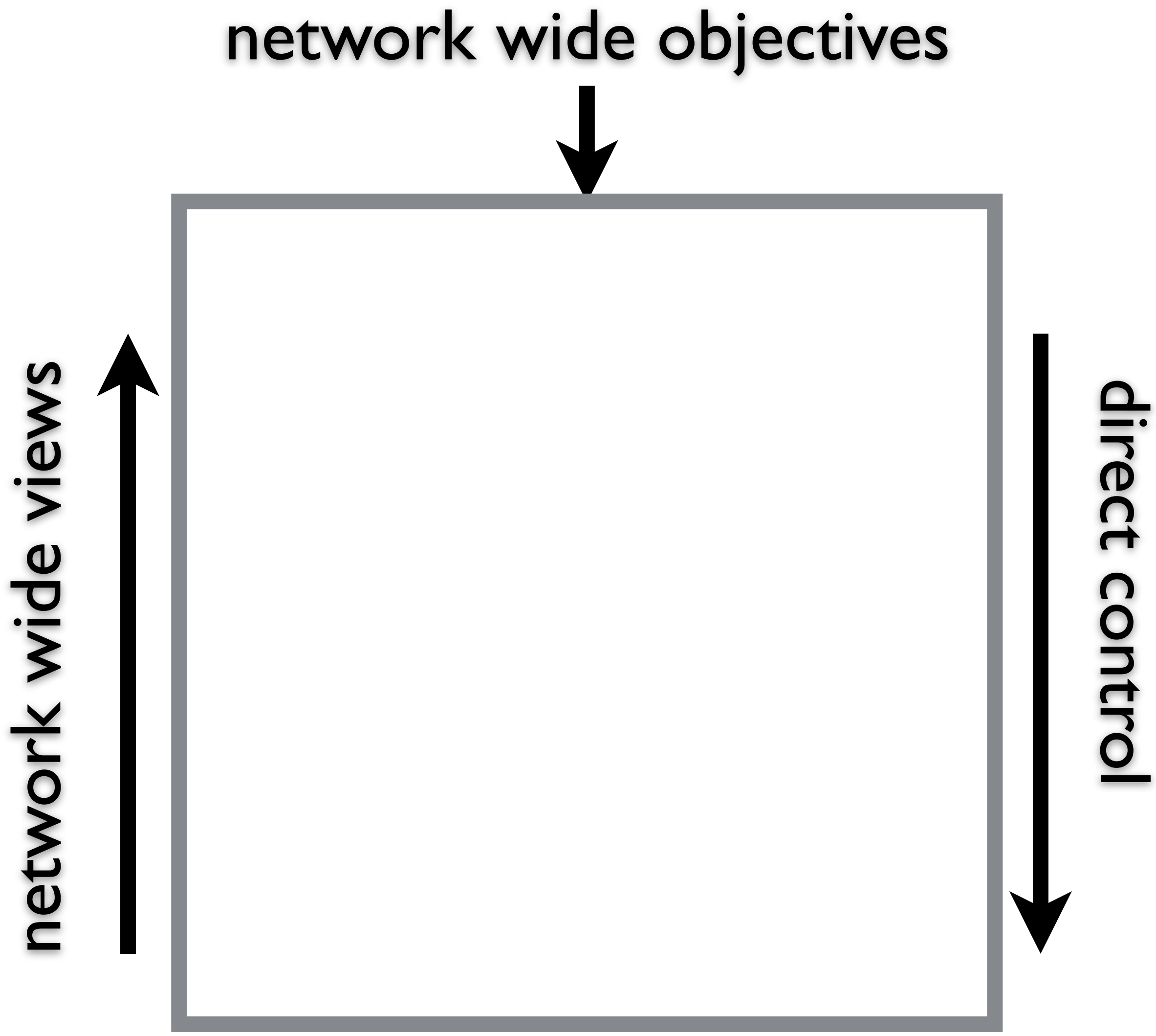
4D

further reading:

A clean slate 4D approach to network control and management

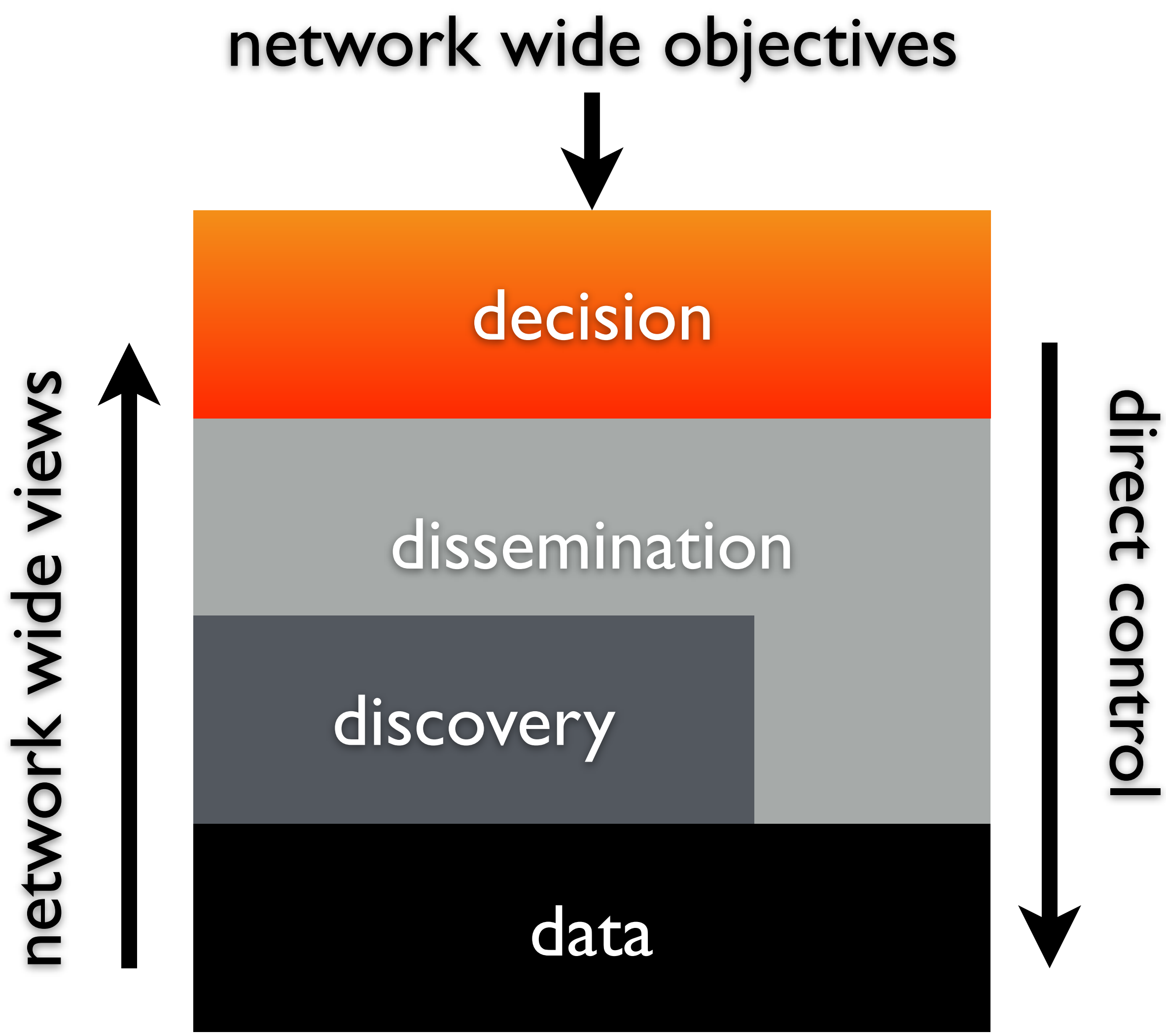
https://dl.acm.org/doi/10.1145/1096536.1096541

4D goals



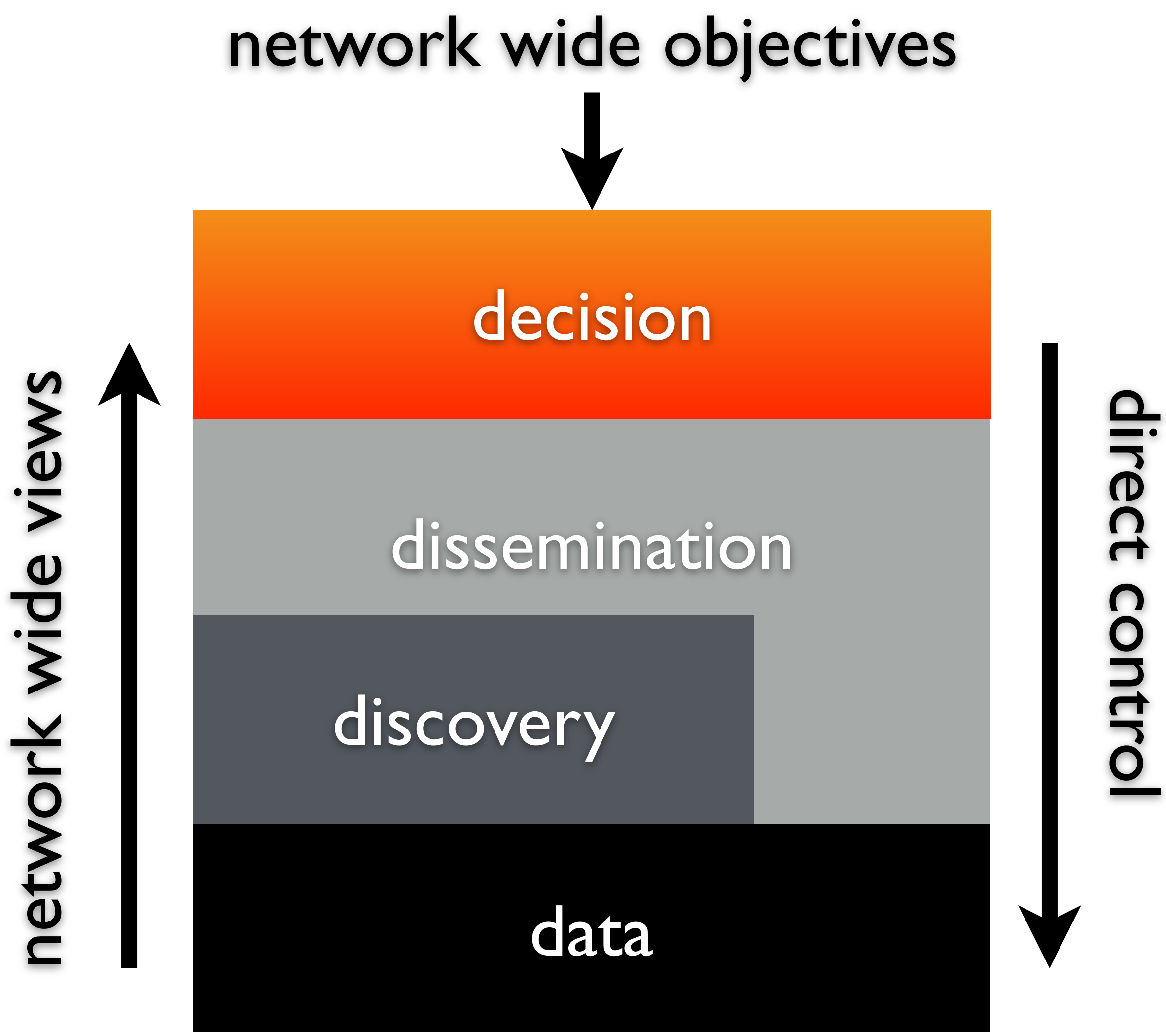
- network-wide objectives
  - observe and control
- network-wide views
  - complete visibility
- direct control
  - direct, sole control

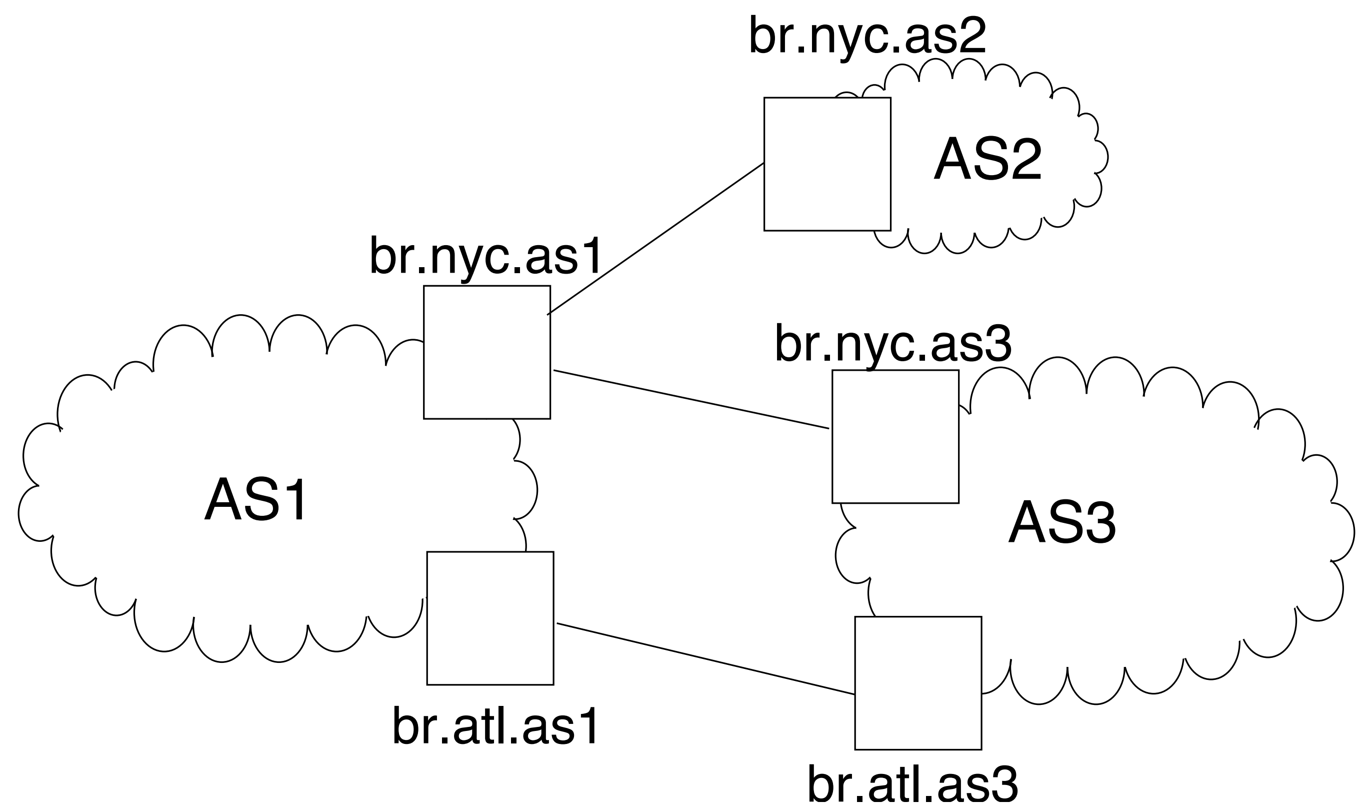
4D architecture



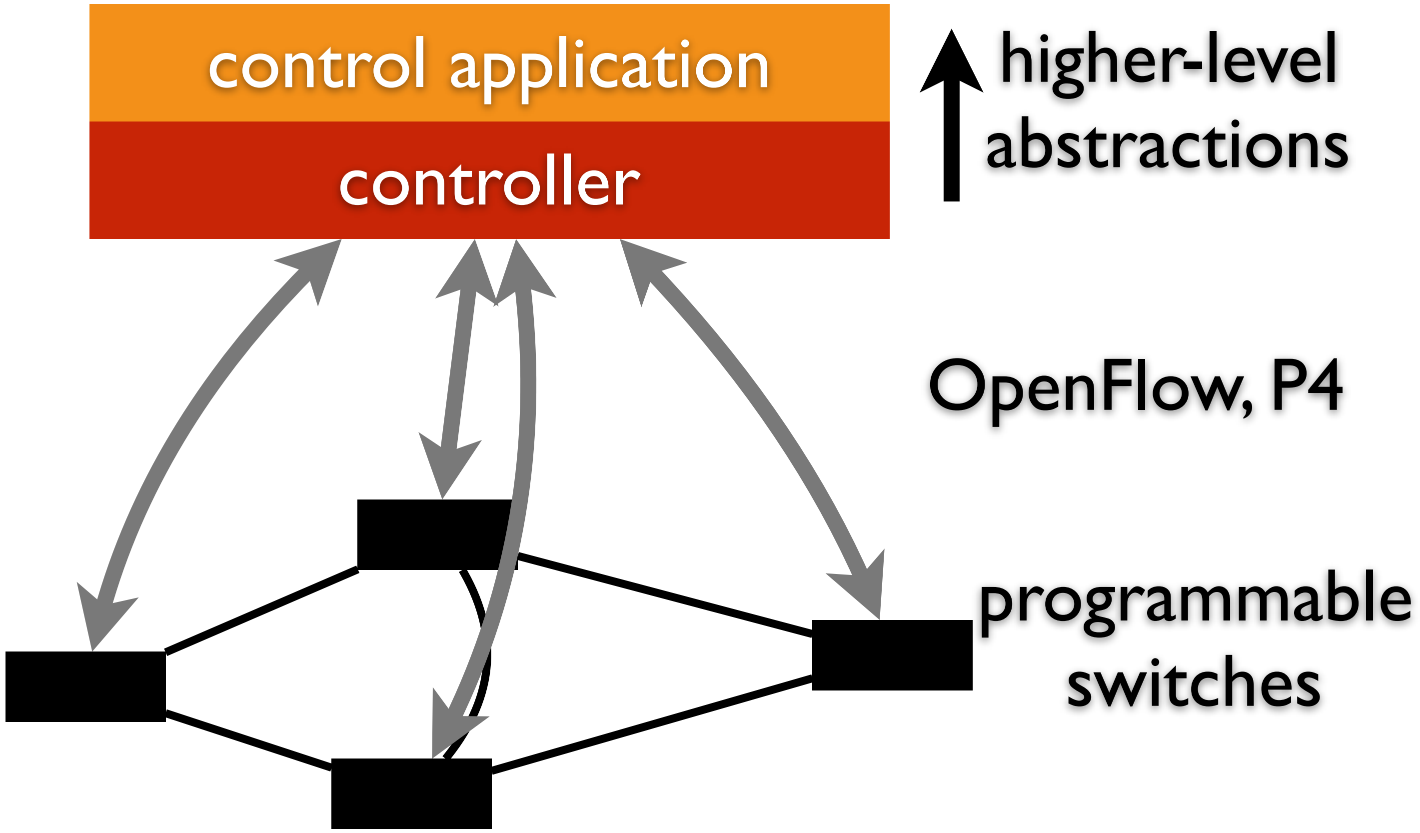
- refactoring network functionality
- extreme design point
  - decoupled, centralized control

4D by example





4D and SDN



2005

2020

Ethane:
a realization of 4D for secure enterprise network

further reading:
Ethane: Taking Control of the Enterprise
http://www.sigcomm.org/node/2620

Ethane goals

enterprise networks

- strict reliability and security constraints
- operated by non-experts

goals

- policy over principals
- direct path selection
- binding packets and its origin

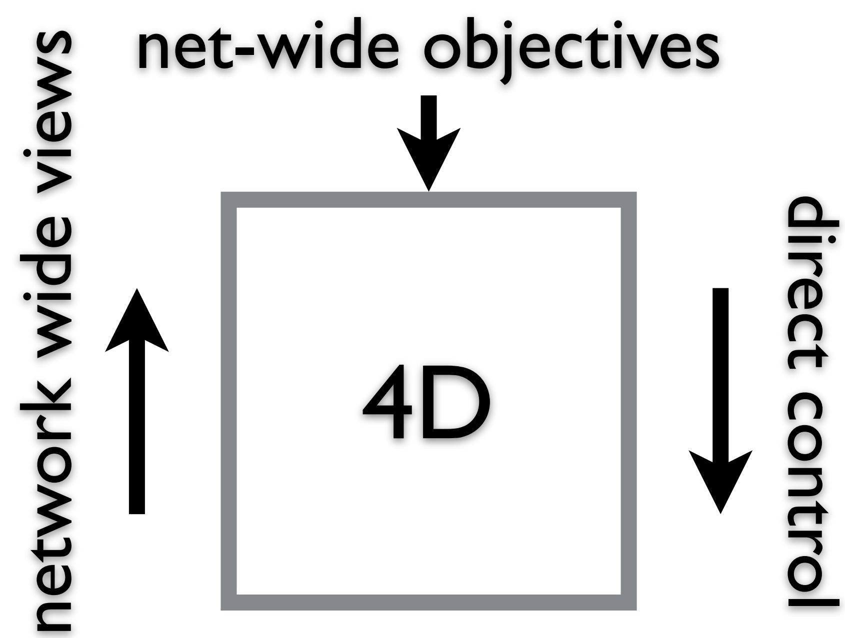
Ethane goals

enterprise networks

- strict reliability and security constraints
- operated by non-experts

goals

- policy over principals
- direct path selection
- binding packets and its origin



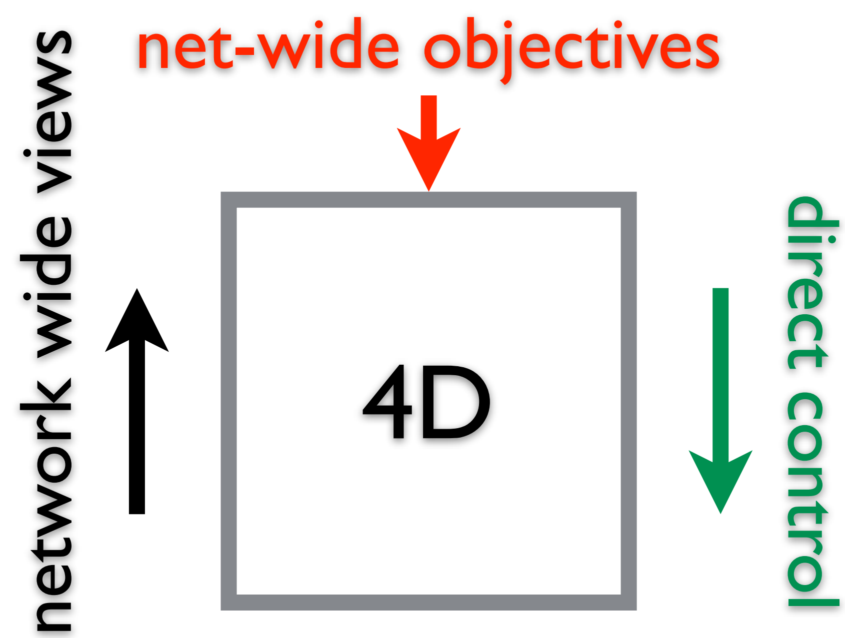
Ethane goals

enterprise networks

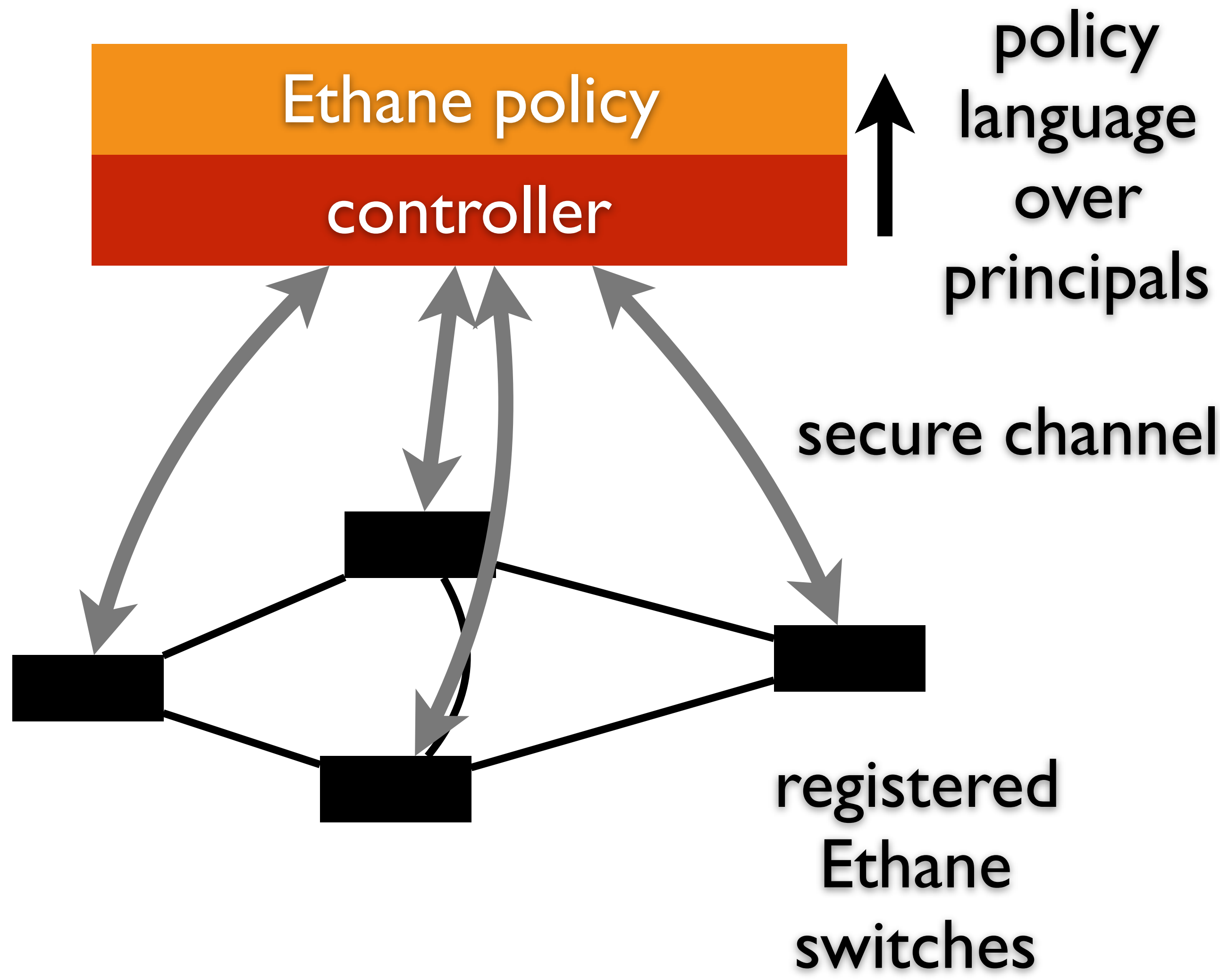
- strict reliability and security constraints
- operated by non-experts

goals

- policy over principals
- policy directs path
- binding packets and its origin



from 4D to Ethane



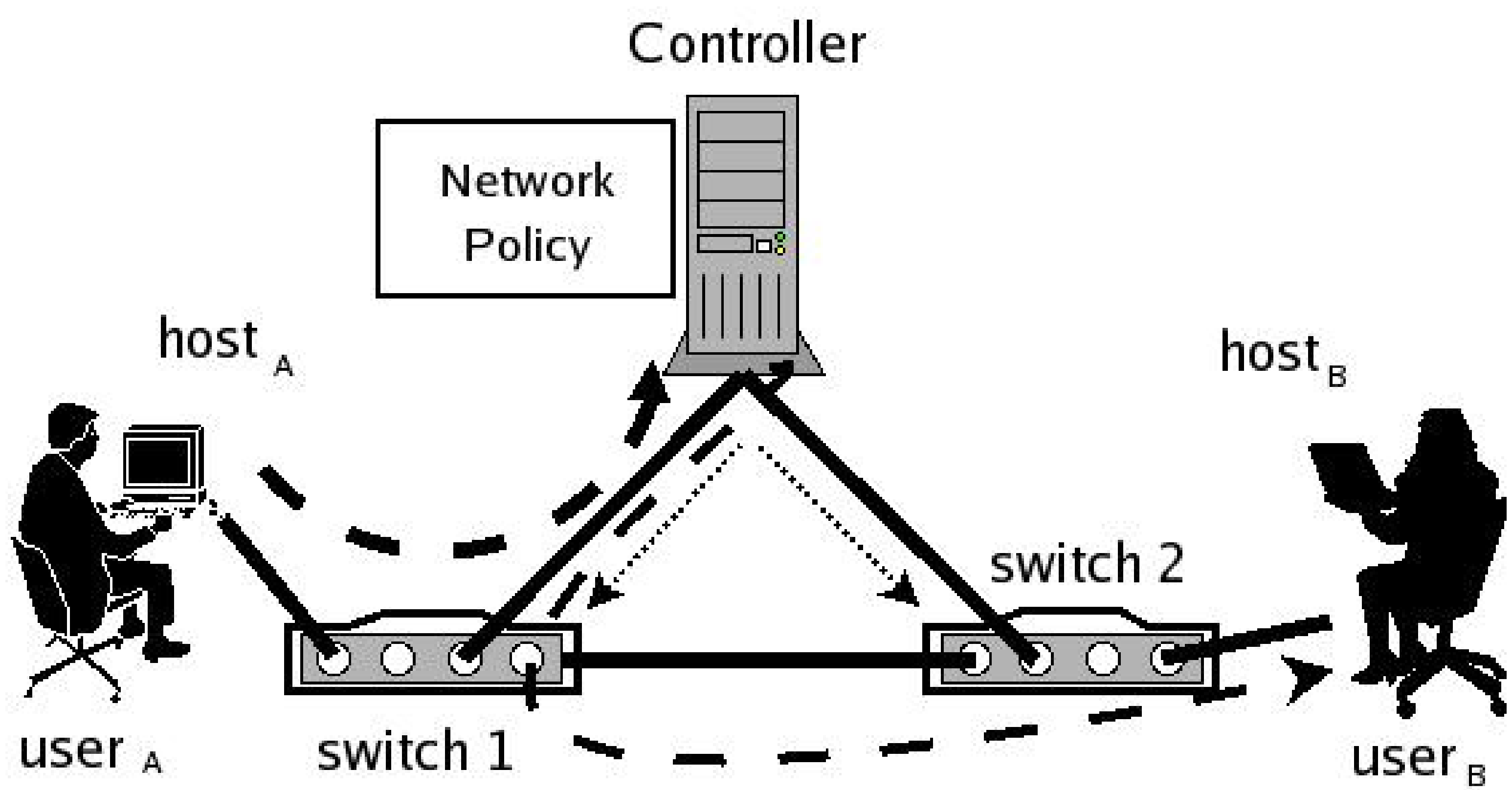
Ethane policy

```
# Groups —
desktops = ["griffin","roo"];
laptops = ["glaptop","rlaptop"];
phones = ["gphone","rphone"];
server = ["http_server","nfs_server"];
private = ["desktops","laptops"];
computers = ["private","server"];
students = ["bob","bill","pete"];
profs = ["plum"];
group = ["students","profs"];
waps = ["wap1","wap2"];
%%
# Rules —
[(hsrc=in("server")∧(hdst=in("private"))] : deny;
# Do not allow phones and private computers to communicate
[(hsrc=in("phones")∧(hdst=in("computers"))] : deny;
[(hsrc=in("computers")∧(hdst=in("phones"))] : deny;
# NAT-like protection for laptops
[(hsrc=in("laptops")] : outbound-only;
# No restrictions on desktops communicating with each other
[(hsrc=in("desktops")∧(hdst=in("desktops"))] : allow;
# For wireless, non-group members can use http through
# a proxy. Group members have unrestricted access.
[(apsrc=in("waps"))∧(user=in("group"))] :allow;
[(apsrc=in("waps"))∧(protocol="http)] : waypoints("http-proxy");
[(apsrc=in("waps"))] : deny;
[]: allow; # Default-on: by default allow flows
```

Ethane in action

three examples

- bootstrapping
- link failure
- replicating controller



deployment

Stanford CS department

- 100Mb/s Ethernet network: 300 hosts, several hundred users, 19 switches
- **policy**: looking at the use of VLANs, end-host firewall configurations, NATs, and router ACLs
- **controller**: standard Linux PC (1.6GHz, 512MB)

performance and scalability

how Ethane performs in the campus network

- controller performance as a function of flow-requests
- performance under (controller/link) failures
- flow table size

extrapolate for larger networks

- using measurement from two more data sets

performance

how Ethane performs in the campus network

- controller performance as a function of flow-requests
- performance under (controller/link) failures

performance

how Ethane performs in the campus network

- controller performance as a function of **flow-requests**
- performance under (controller/link) failures

performance — flow requests

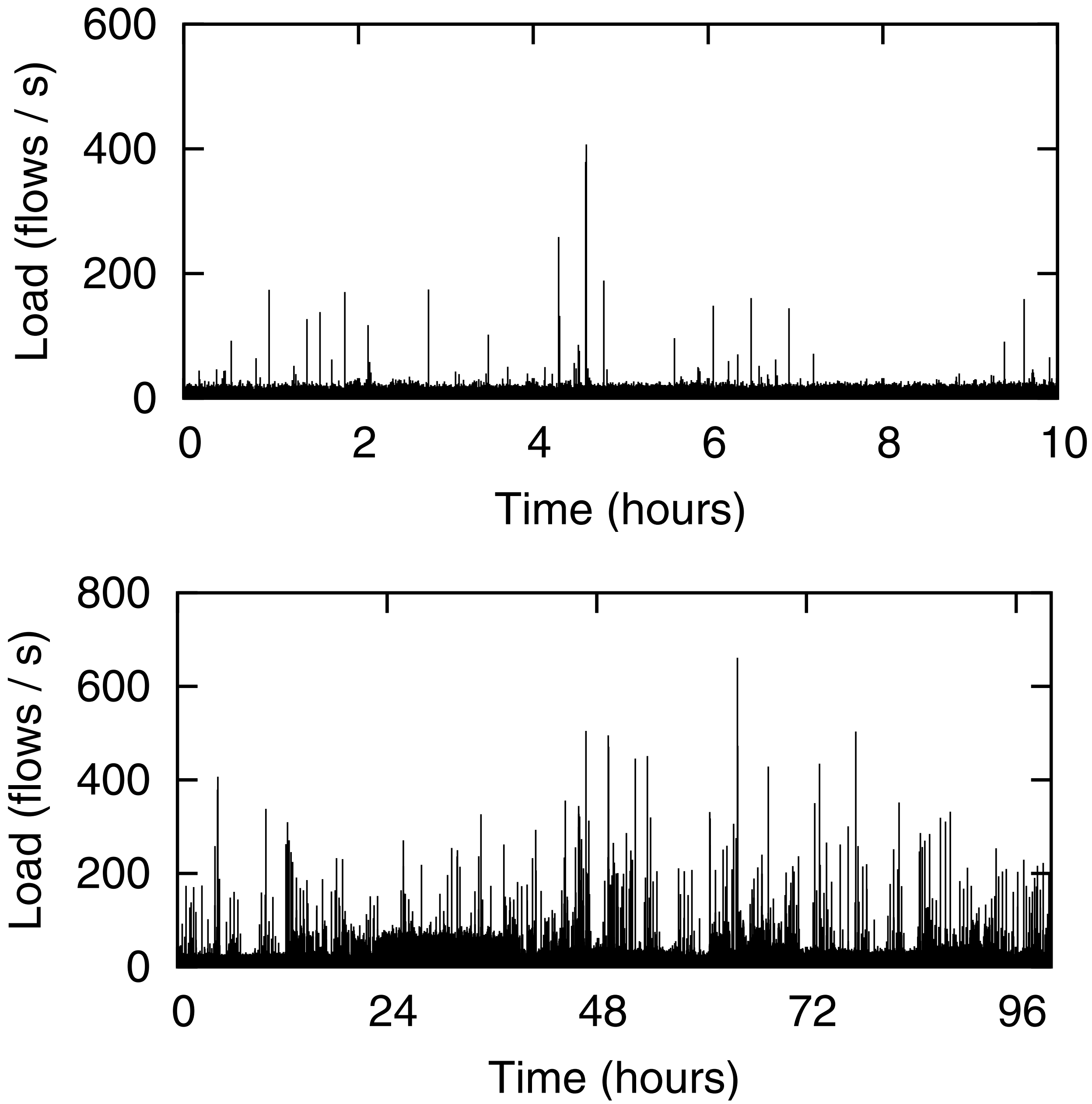
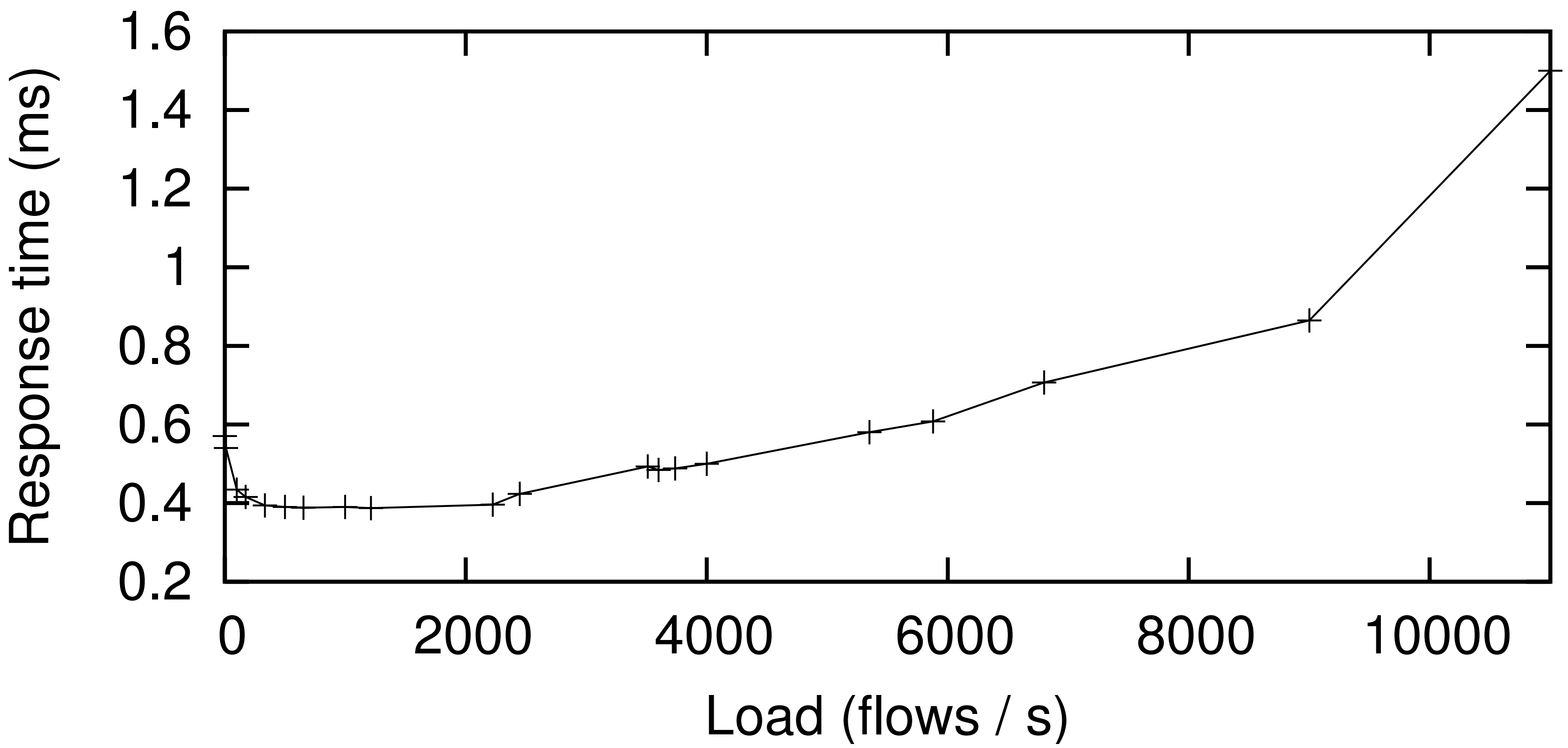


Figure 5: Frequency of flow-setup requests per second to Controller over a 10-hour period (top) and 4-day period (bottom).

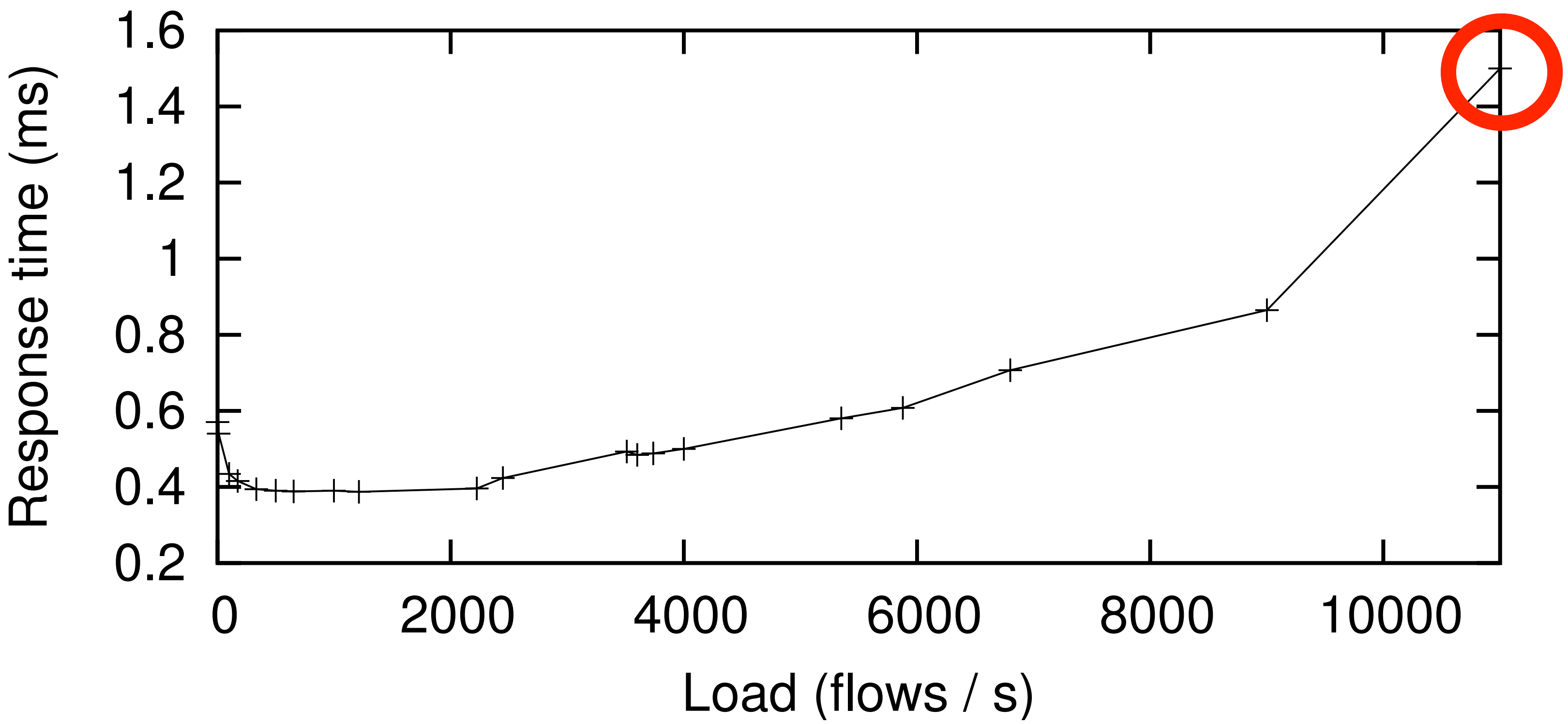
- 30-40 new flow requests per second
- peak: 750 requests

performance — controller setup time



flow-setup times as a function of controller load

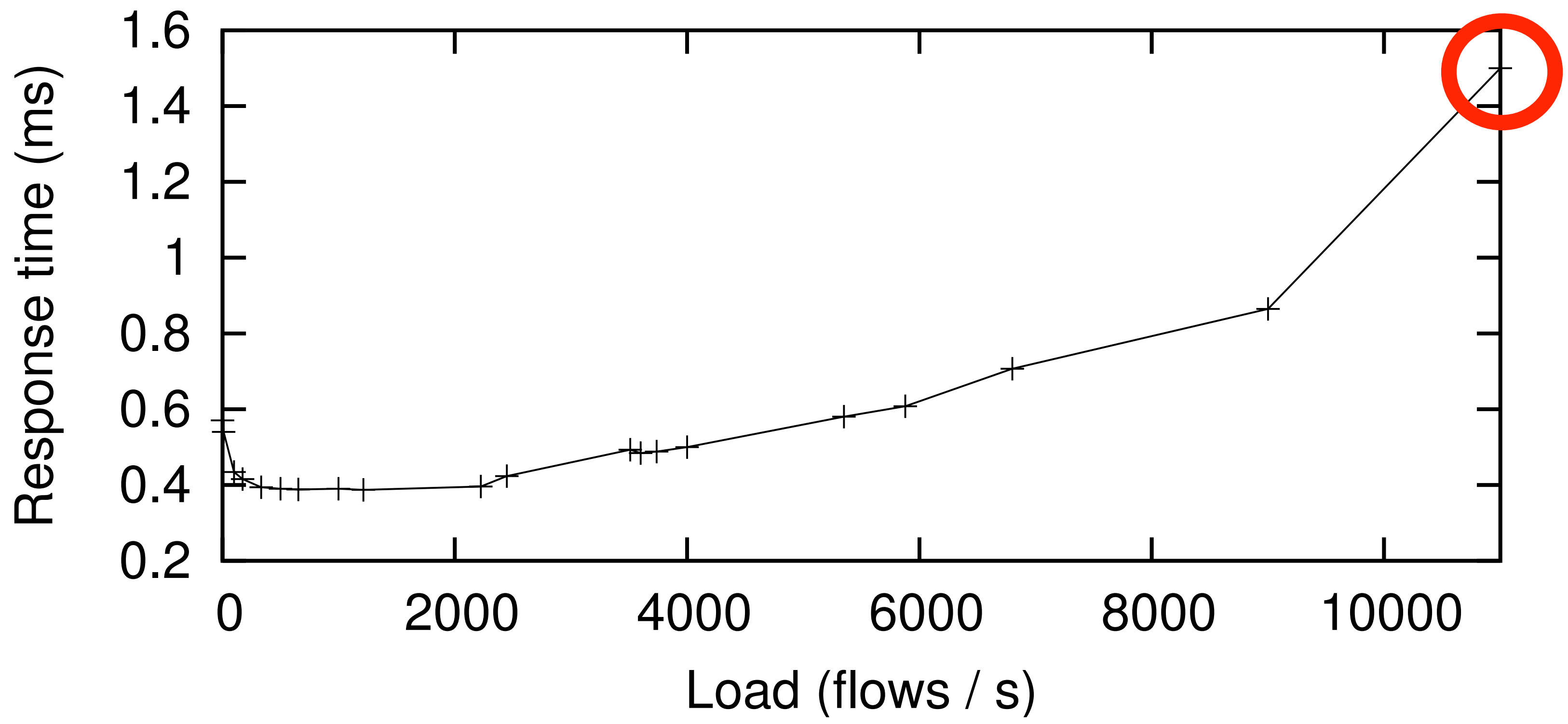
performance — controller setup time



- <1.5ms under worst load of 11,000 flows

flow-setup times as a function of controller load

performance — controller setup time



- <1.5ms under worst load of 11,000 flows

flow-setup times as a function of controller load

how about larger networks?

flow-requests on larger networks

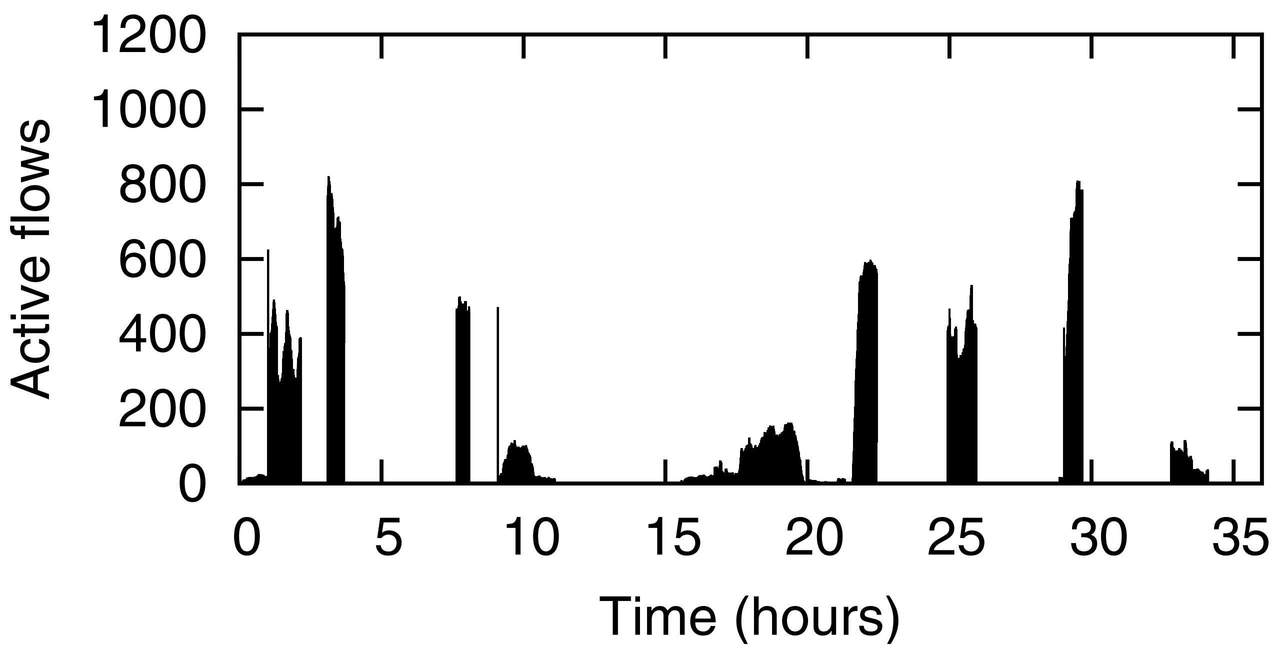


Figure 7: Active flows for LBL network [19].

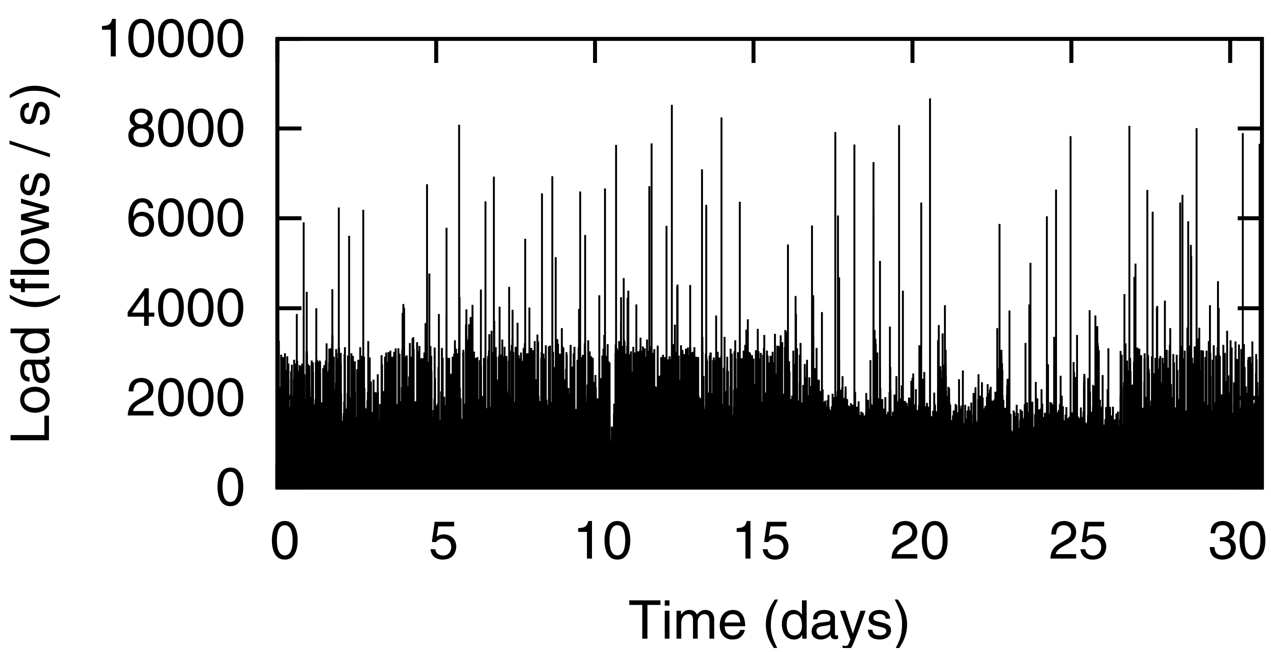


Figure 8: Flow-request rate for Standford network.

- LBL
  - 8,000 hosts
  - load <1200 flow
- Standford
  - 22,000 hosts
  - load < 9,000 new requests per second

flow-requests on larger networks

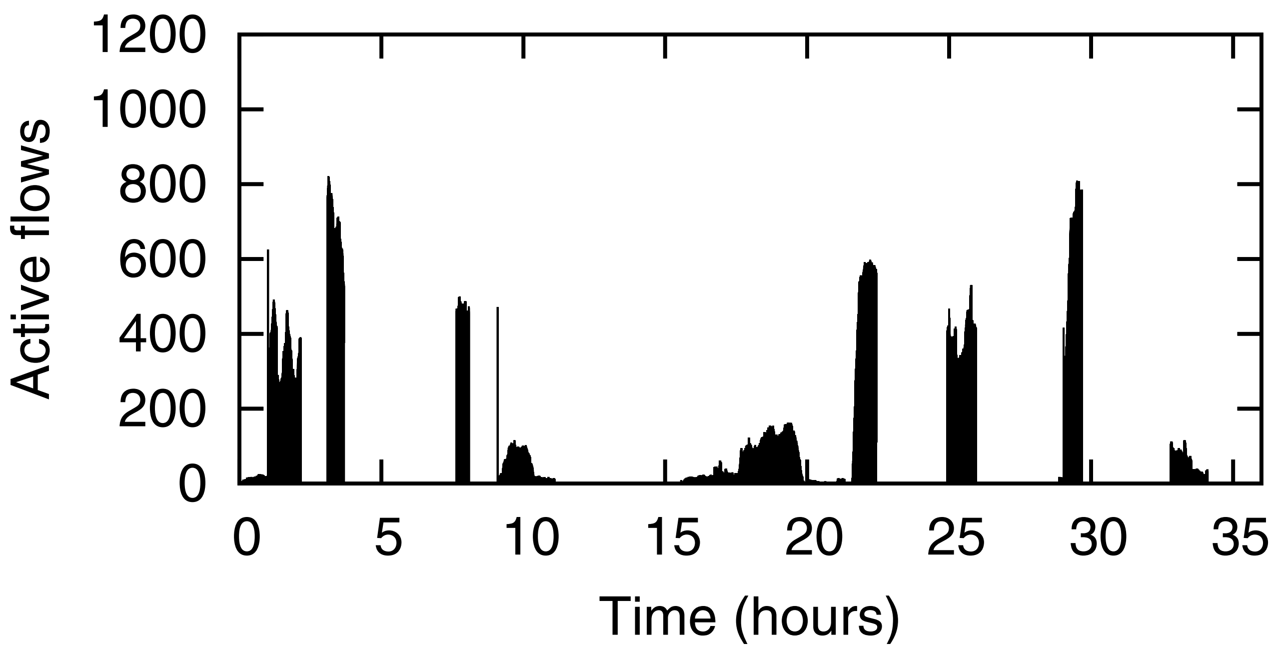
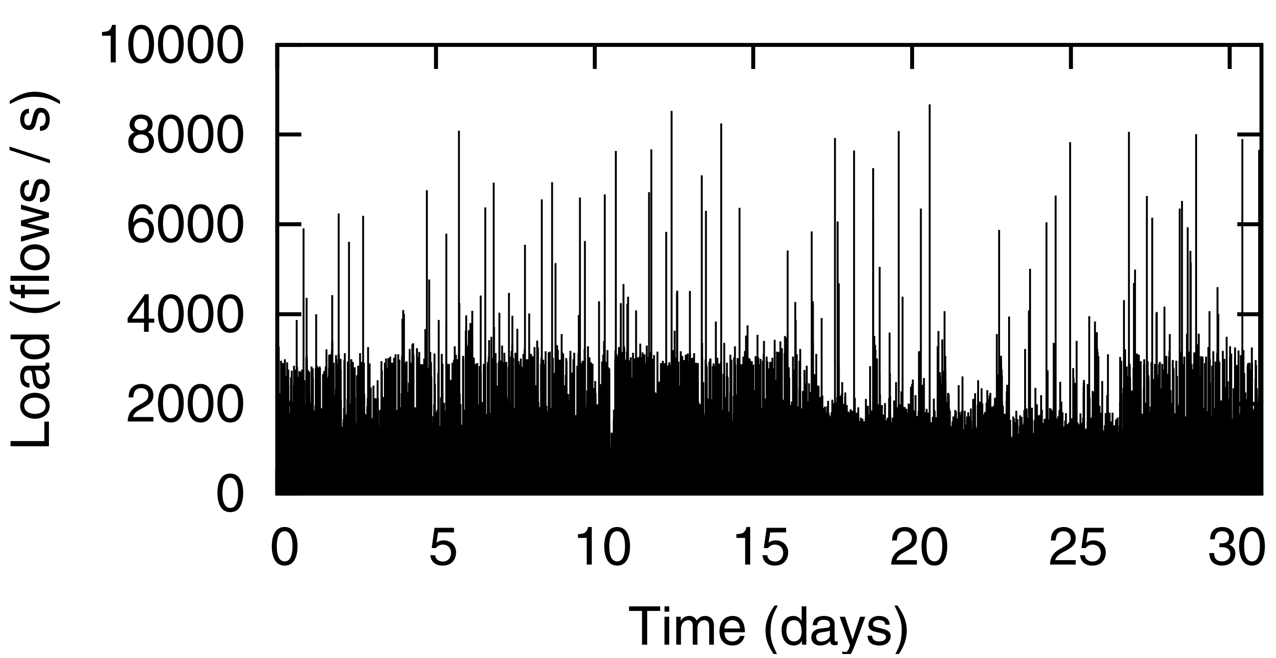


Figure 7: Active flows for LBL network [19].

- LBL
  - 8,000 hosts
  - load <1200 flow



- Standford
  - 22,000 hosts
  - load < 9,000 new requests per second

Ethane can comfortably handle

performance during failures

controller failure

link failure

performance during controller failure

controller failure

- Ethane implements cold-standby failure recovery (replica has no binding state)
- interruption of service for active flows and a delay with re-establishing

performance during controller failure

penalty for each failure

- 10% increase in overall completion time

Failures	0	1	2	3	4
Completion time	26.17s	27.44s	30.45s	36.00s	43.09s

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.

performance during link failure

require all outstanding flows re-contact the controller and re-establish the path

performance during link failure

require all outstanding flows re-contact the controller and re-establish the path

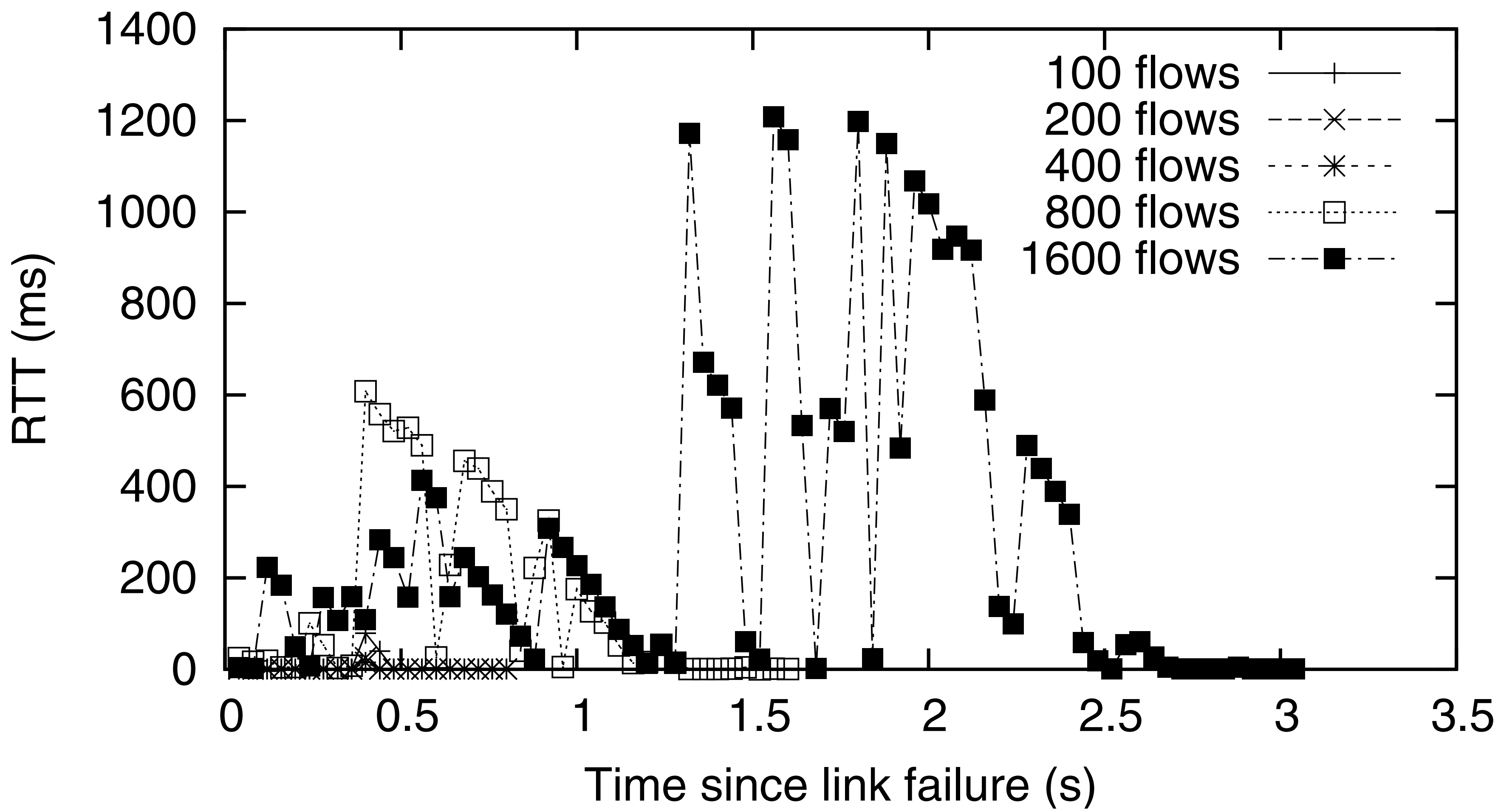


Figure 10: Round-trip latencies experienced by packets through a diamond topology during link failure.

flow table sizing

observation #1

- flow table size bound by # of active flows

flow table sizing

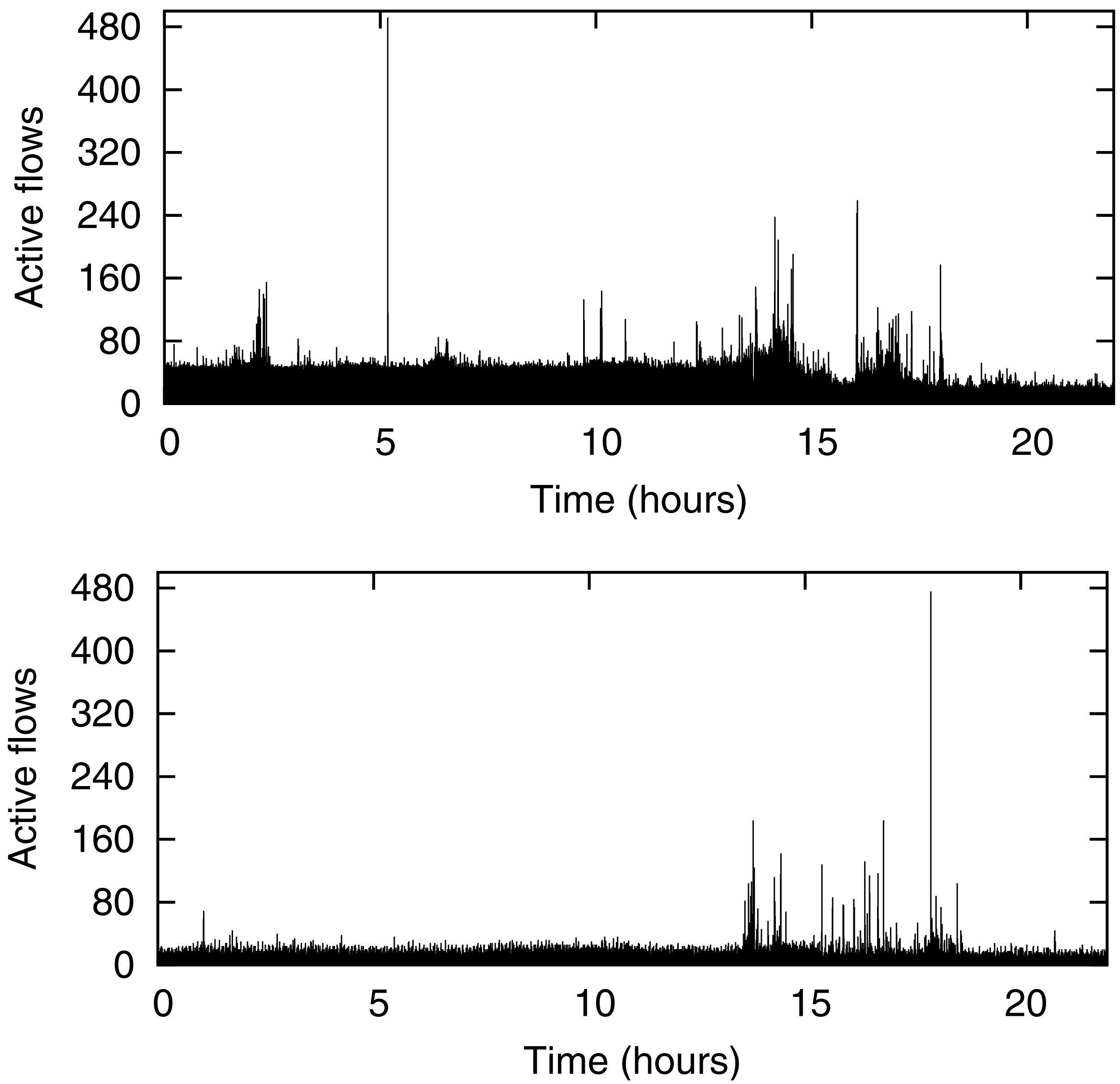


Figure 9: Active flows through two of our deployed switches

observation #1

- flow table size bound by # of active flows
- <500 active flows

flow table sizing

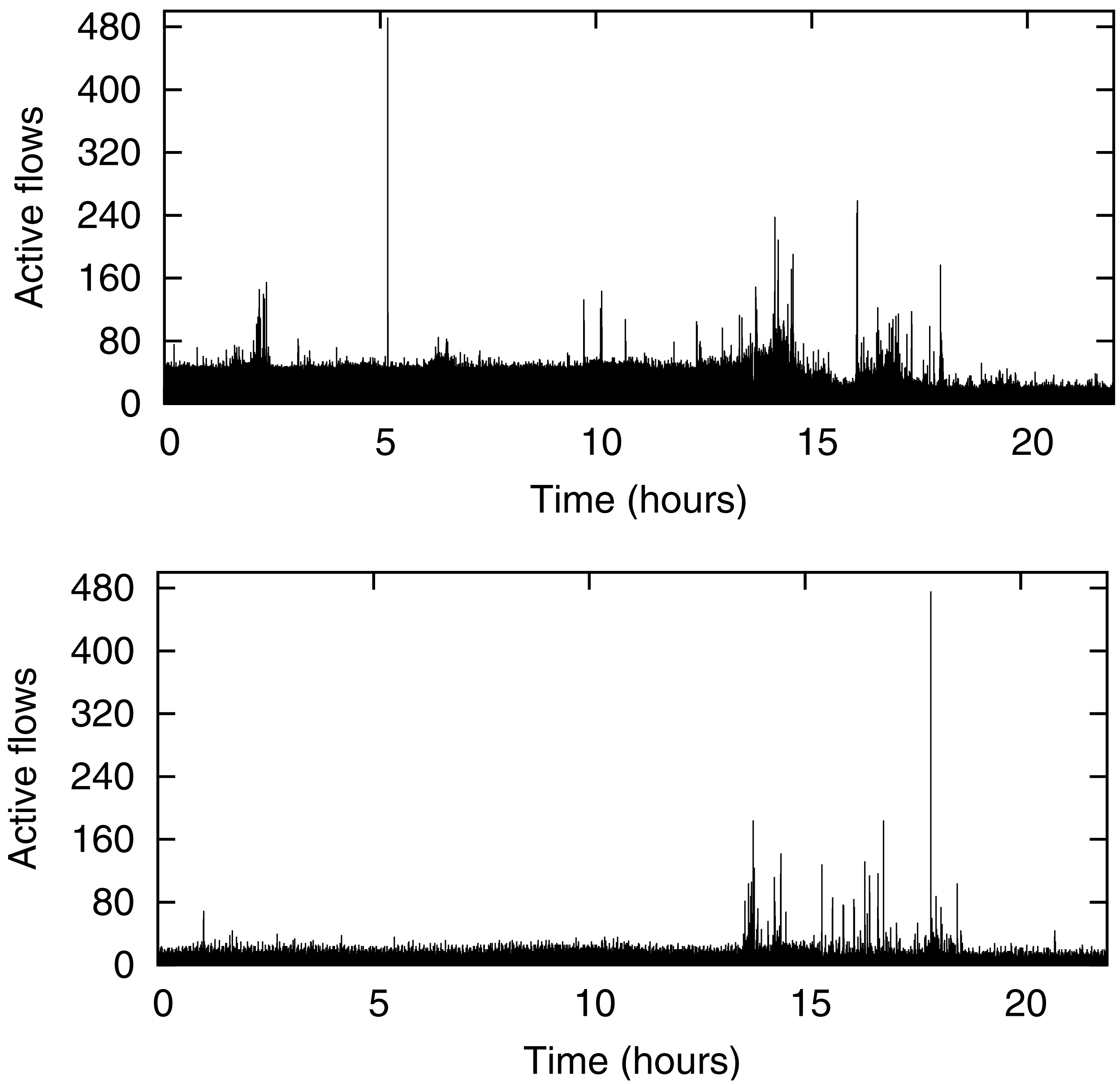


Figure 9: Active flows through two of our deployed switches

observation #1

- flow table size bound by # of active flows
- <500 active flows
- recall
  - LBL: < 1,200 flows for 8,000 hosts

flow table sizing

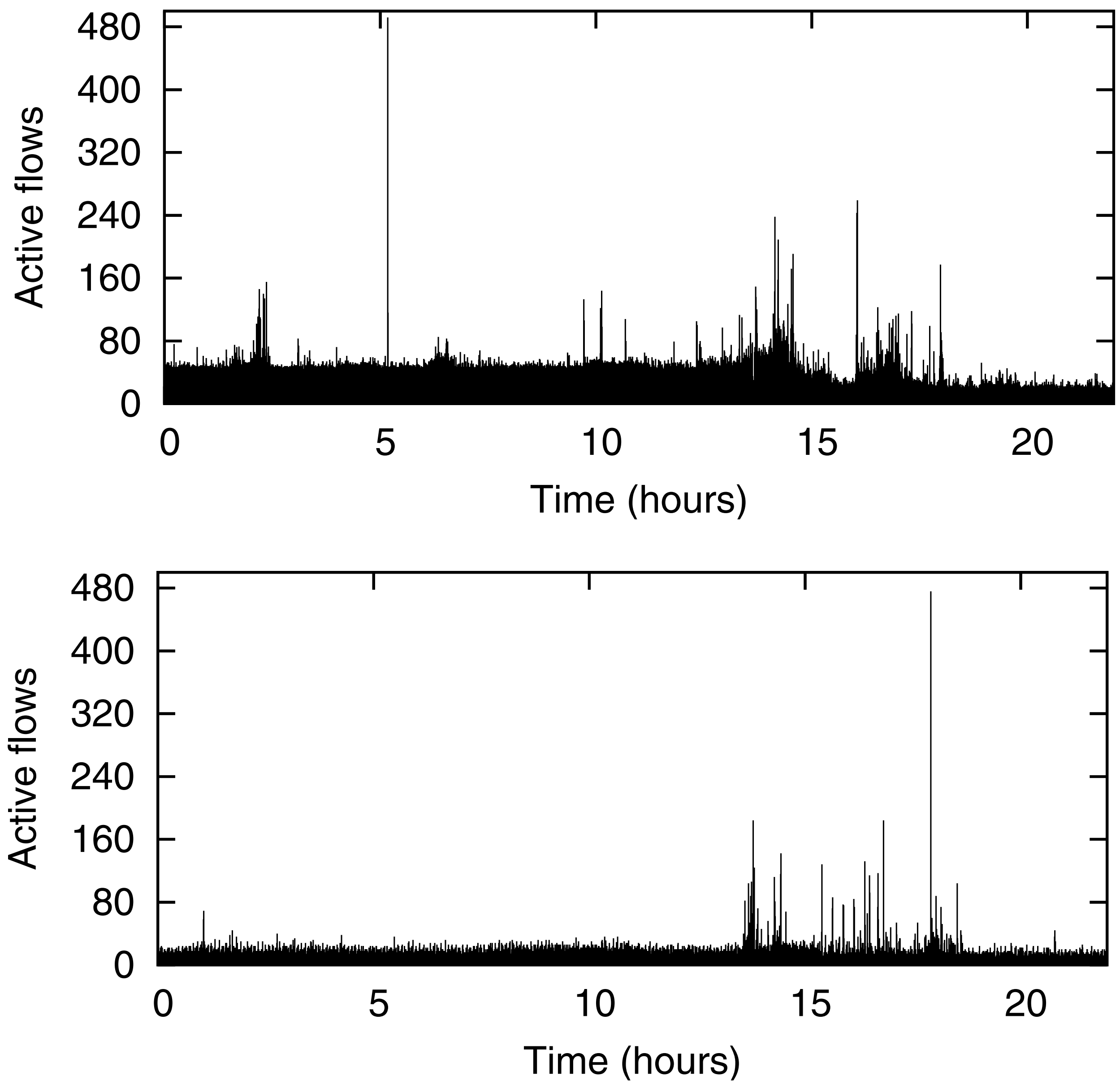


Figure 9: Active flows through two of our deployed switches

observation #1

- flow table size bound by # of active flows

flow table sizing

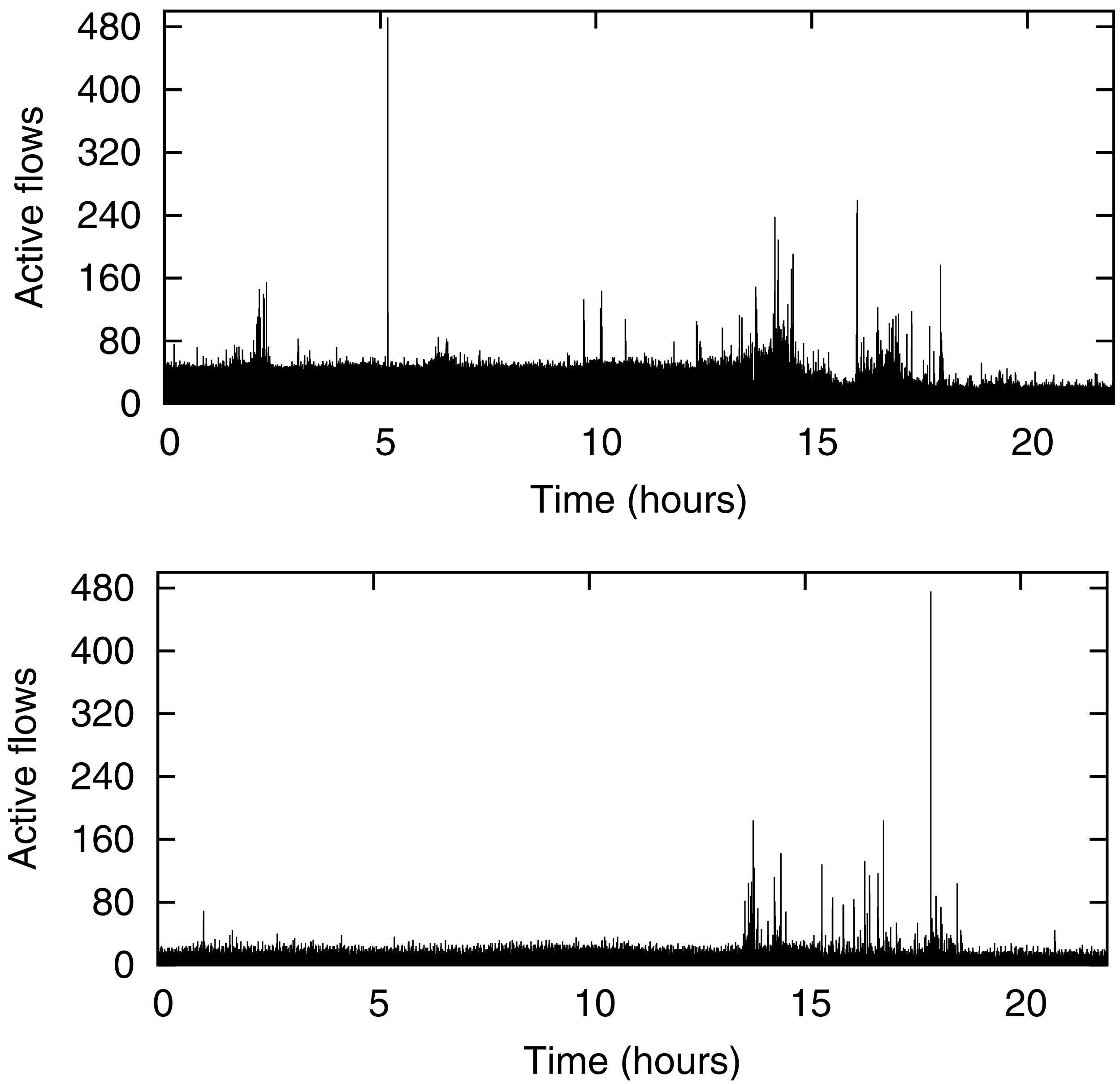


Figure 9: Active flows through two of our deployed switches

observation #1

- flow table size bound by # of active flows

observation #2

- # of active flows depend on switch location
  - edge: bound by connected hosts
  - core: more

flow table sizing

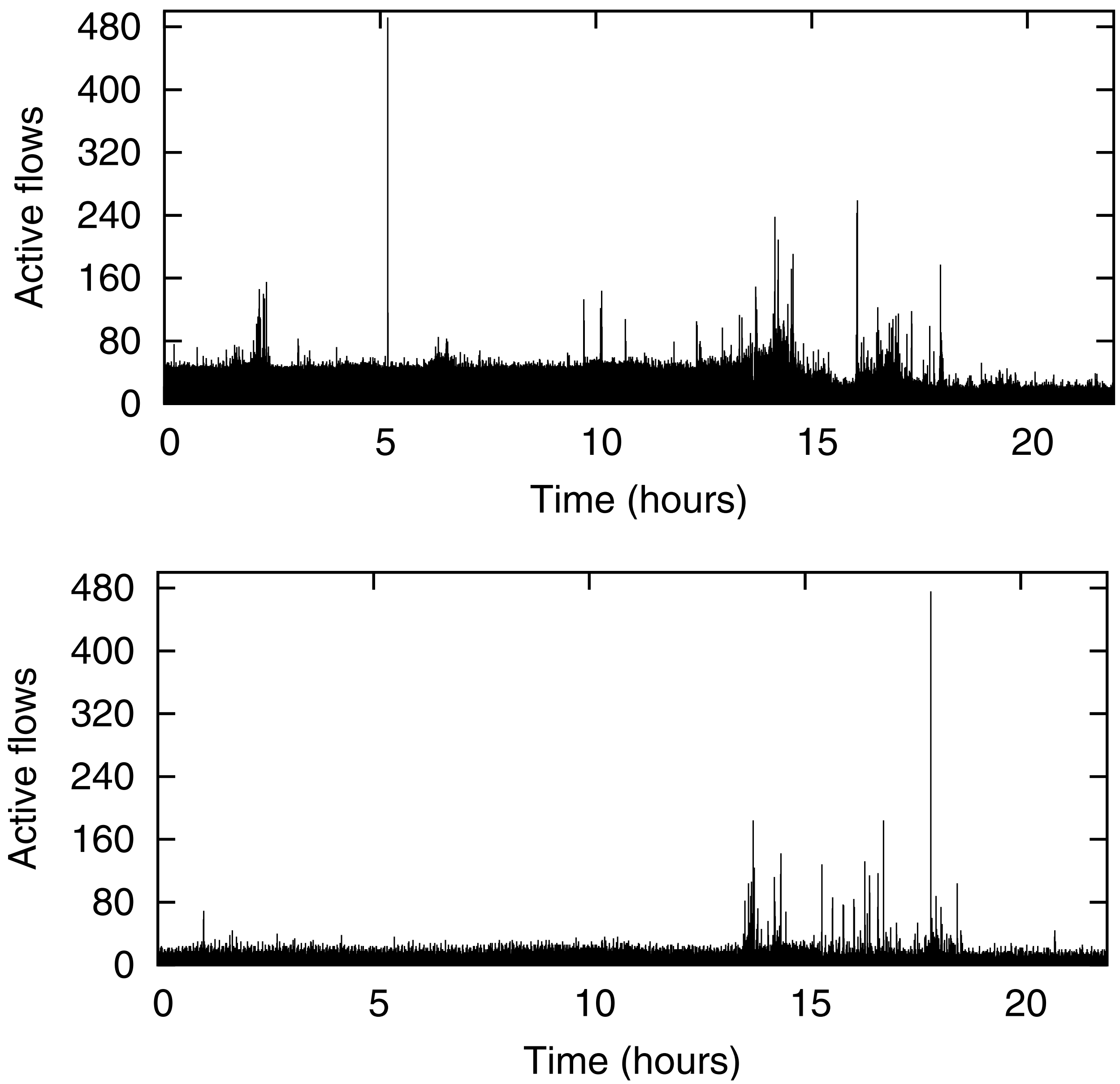


Figure 9: Active flows through two of our deployed switches

observation #1

- flow table size bound by # of active flows

observation #2

- # of active flows depend on switch location

observation #3

- Ethernet switch:
  - 1million Ethernet addresses
  - 1million IP addresses
  - thousands of ACLs

flow table sizing

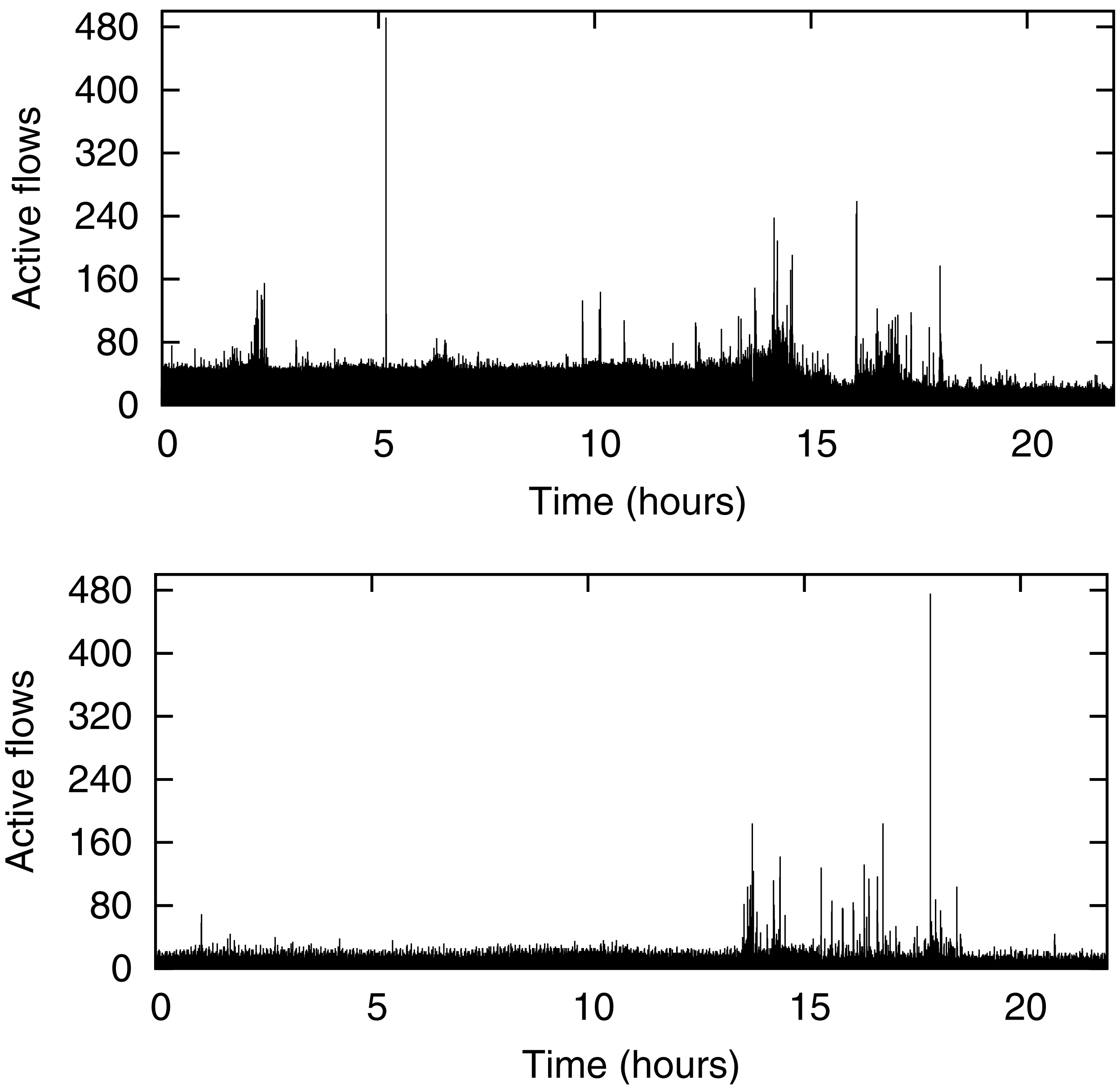


Figure 9: Active flows through two of our deployed switches

observation #1

- flow table size bound by # of active flows

observation #2

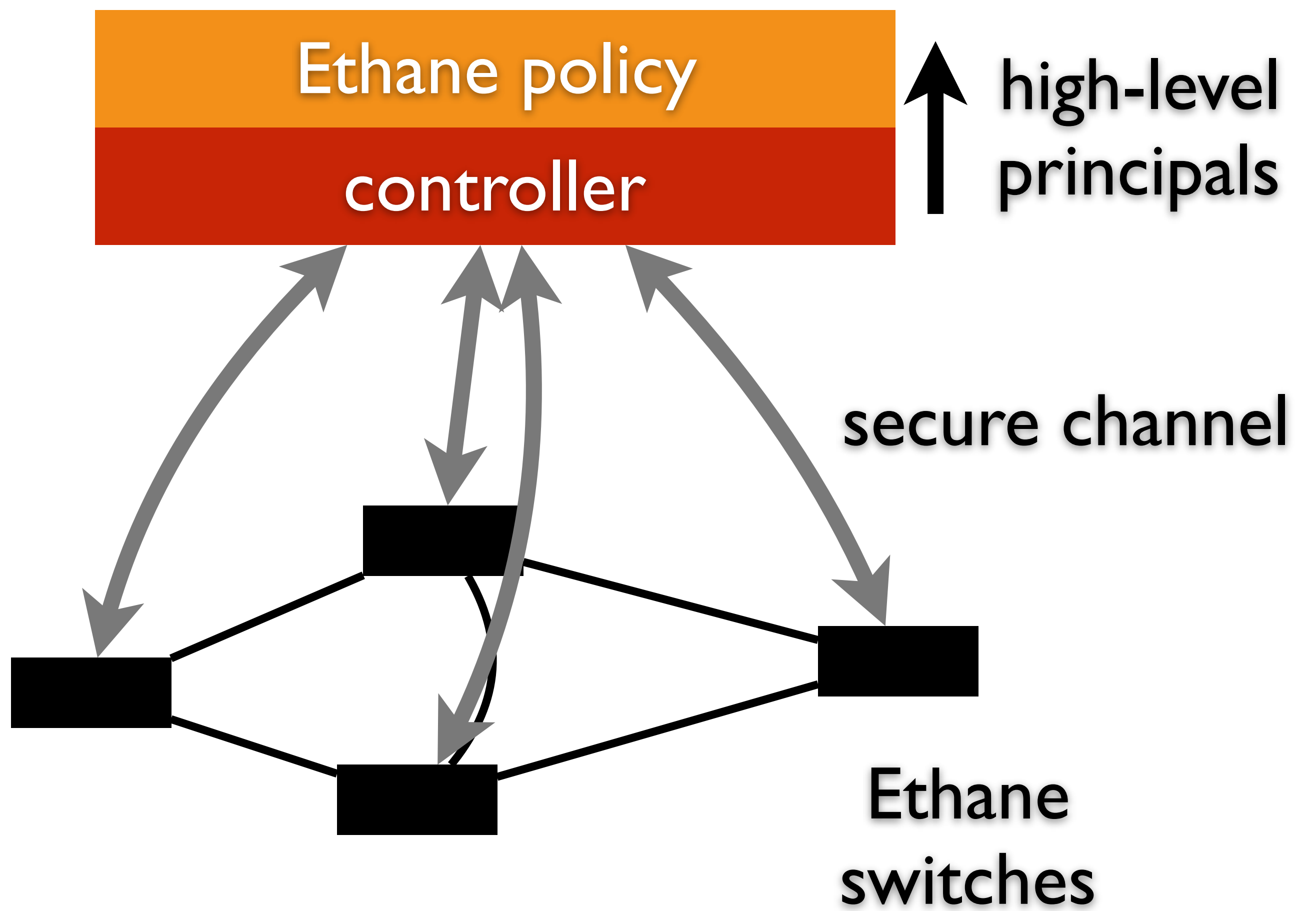
- # of active flows depend on switch location

observation #3

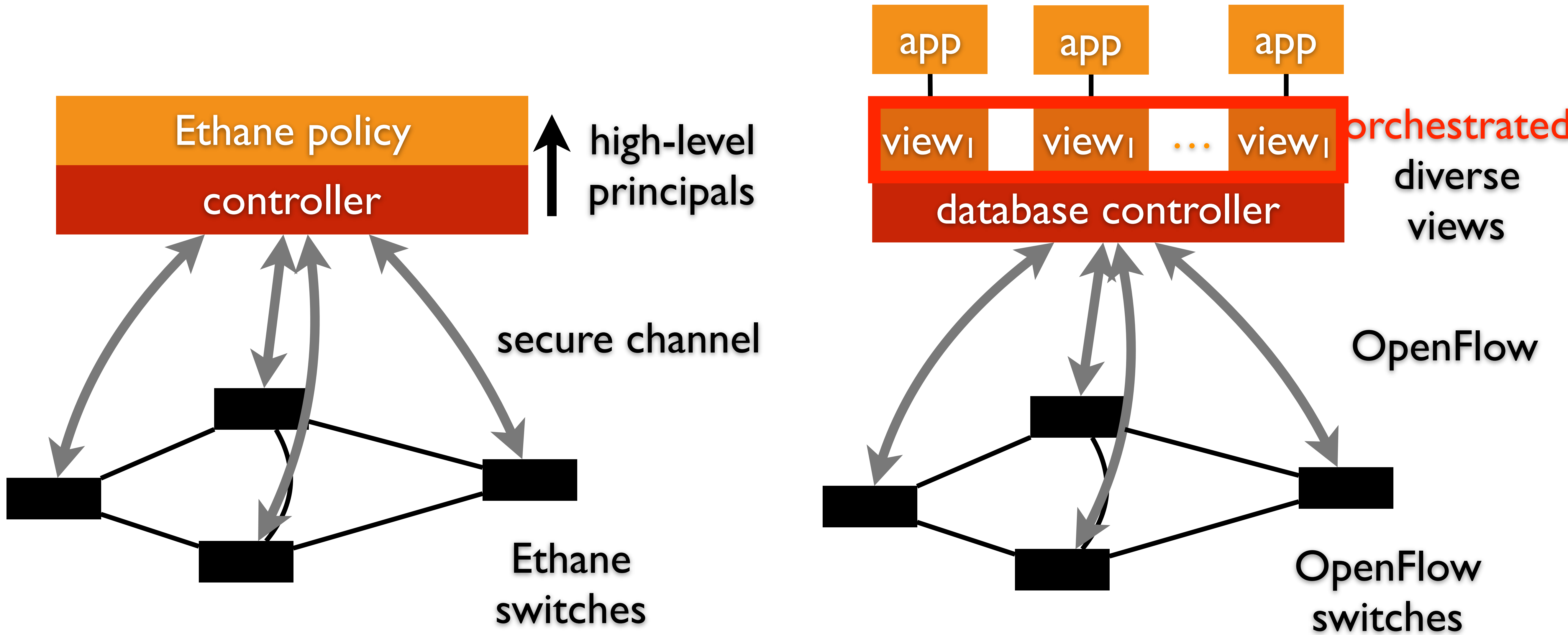
- Ethernet switch:
  - 1million Ethernet addresses
  - 1million IP addresses
  - thousands of ACLs

memory requirements on Ethane switch are modest

Ethane — recap



Ethane and Ravel



further reading:

Ravel: A Database-Defined Network

http://anduowang.github.io/docs/sosr16.pdf