lecture 07: programming SDN

5590: software defined networking

anduo wang, Temple University

TTLMAN 401B, R 17:30-20:00

OpenFlow

ossified network infrastructure

ossified network infrastructure

exceedingly high barrier to entry for new ideas

ossified network infrastructure

exceedingly high barrier to entry for new ideas

- installed base of equipments and protocols

ossified network infrastructure

exceedingly high barrier to entry for new ideas

- installed base of equipments and protocols
- lacking experiment with production traffic

ossified network infrastructure

exceedingly high barrier to entry for new ideas

- installed base of equipments and protocols
- lacking experiment with production traffic

programmable network?

ossified network infrastructure

exceedingly high barrier to entry for new ideas

- installed base of equipments and protocols
- lacking experiment with production traffic

programmable network?

- GENI

ossified network infrastructure

exceedingly high barrier to entry for new ideas

- installed base of equipments and protocols
- lacking experiment with production traffic

programmable network?

- GENI
  - nationwide facility are ambitious (and costly)

problems

commercial solutions

- too closed, inflexible

research solutions

- insufficient packet-processing performance, fanout (port-density)

OpenFlow approach

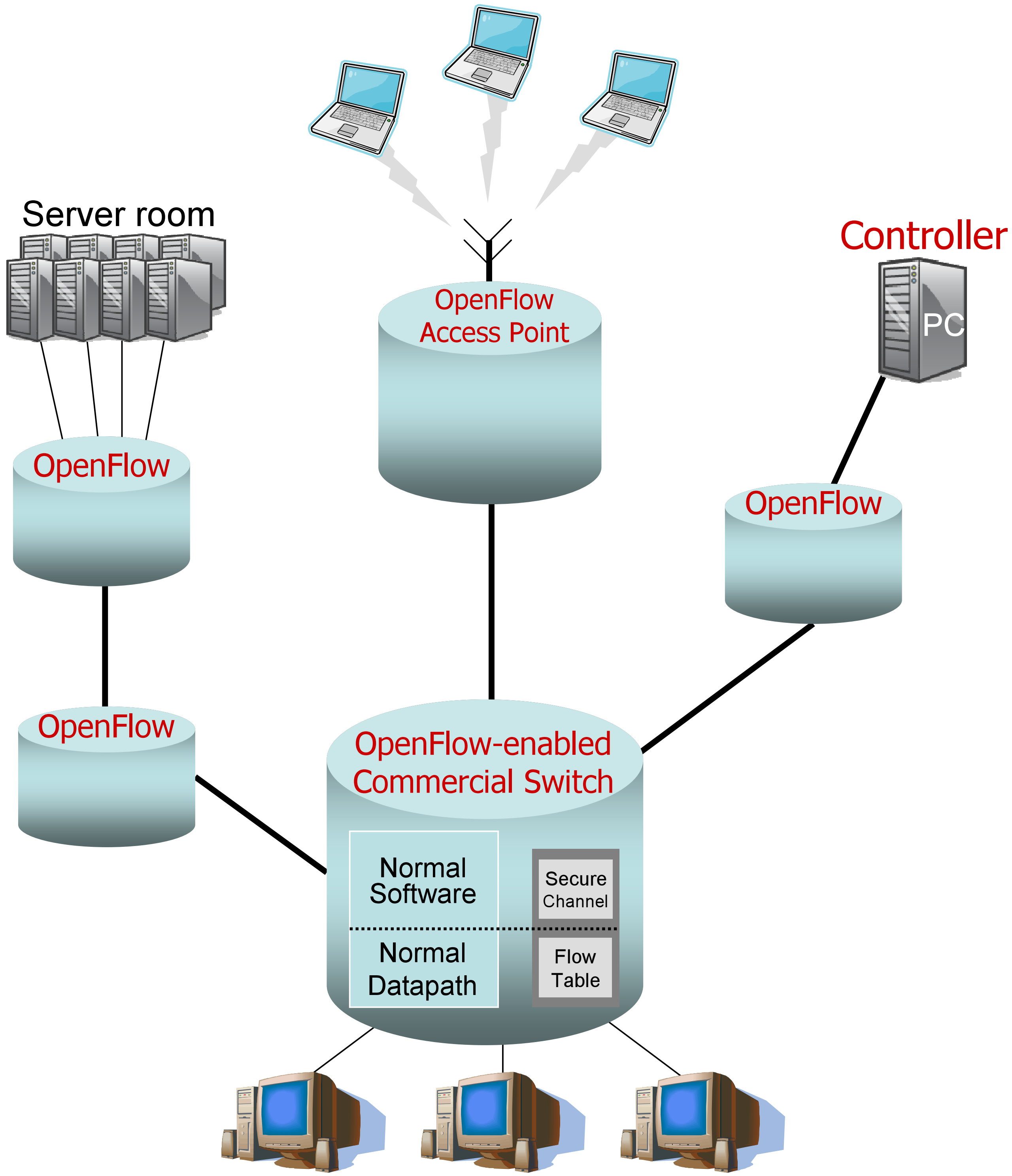
break vendor lock-in

- a pragmatic compromise
  - run experiments on heterogenous switches with unified interface
    - line rate, high port-density
  - vendors need not to expose internals of their switches

assure isolated experiments

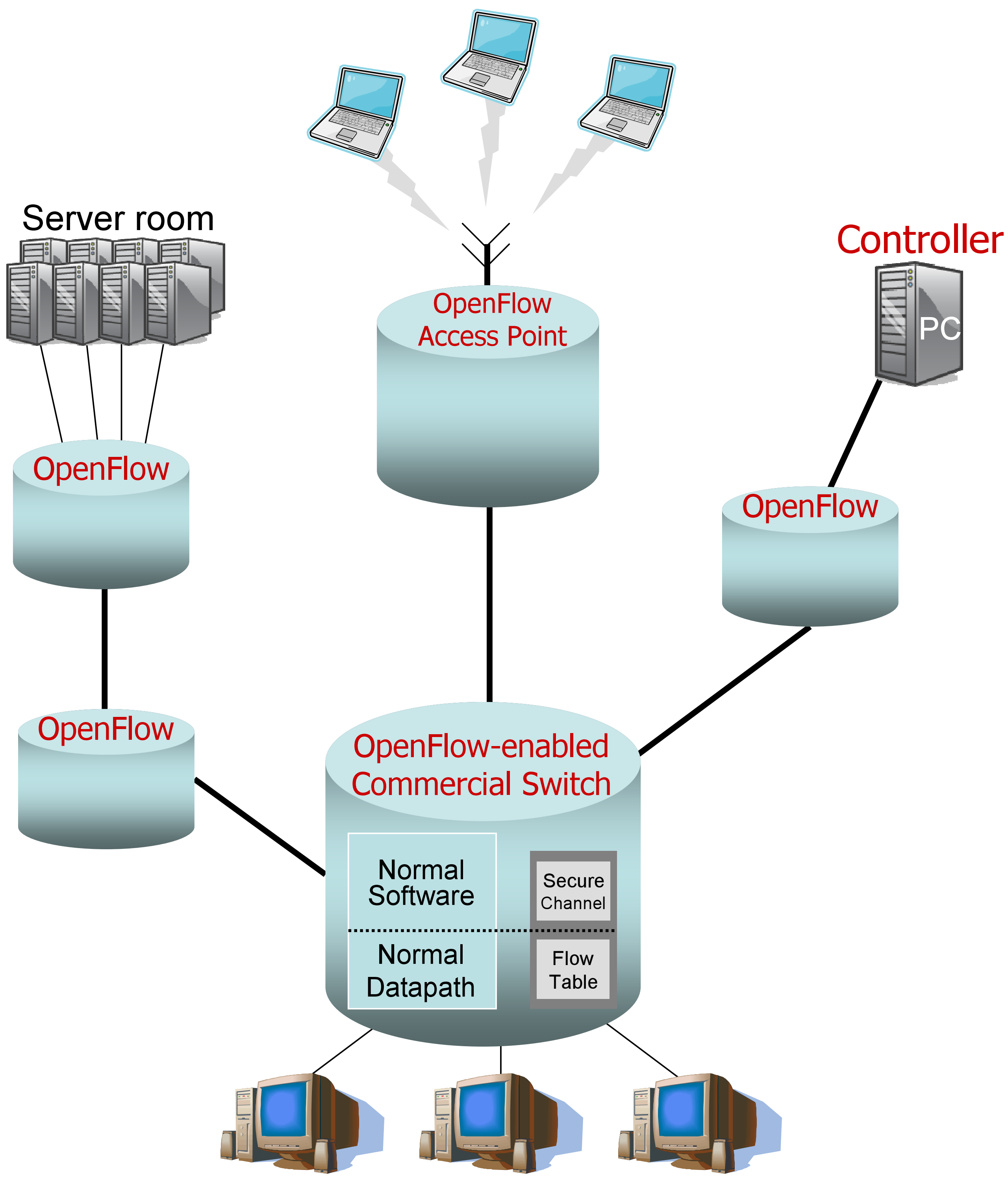
- pull out decision to a remote controller

OpenFlow overview

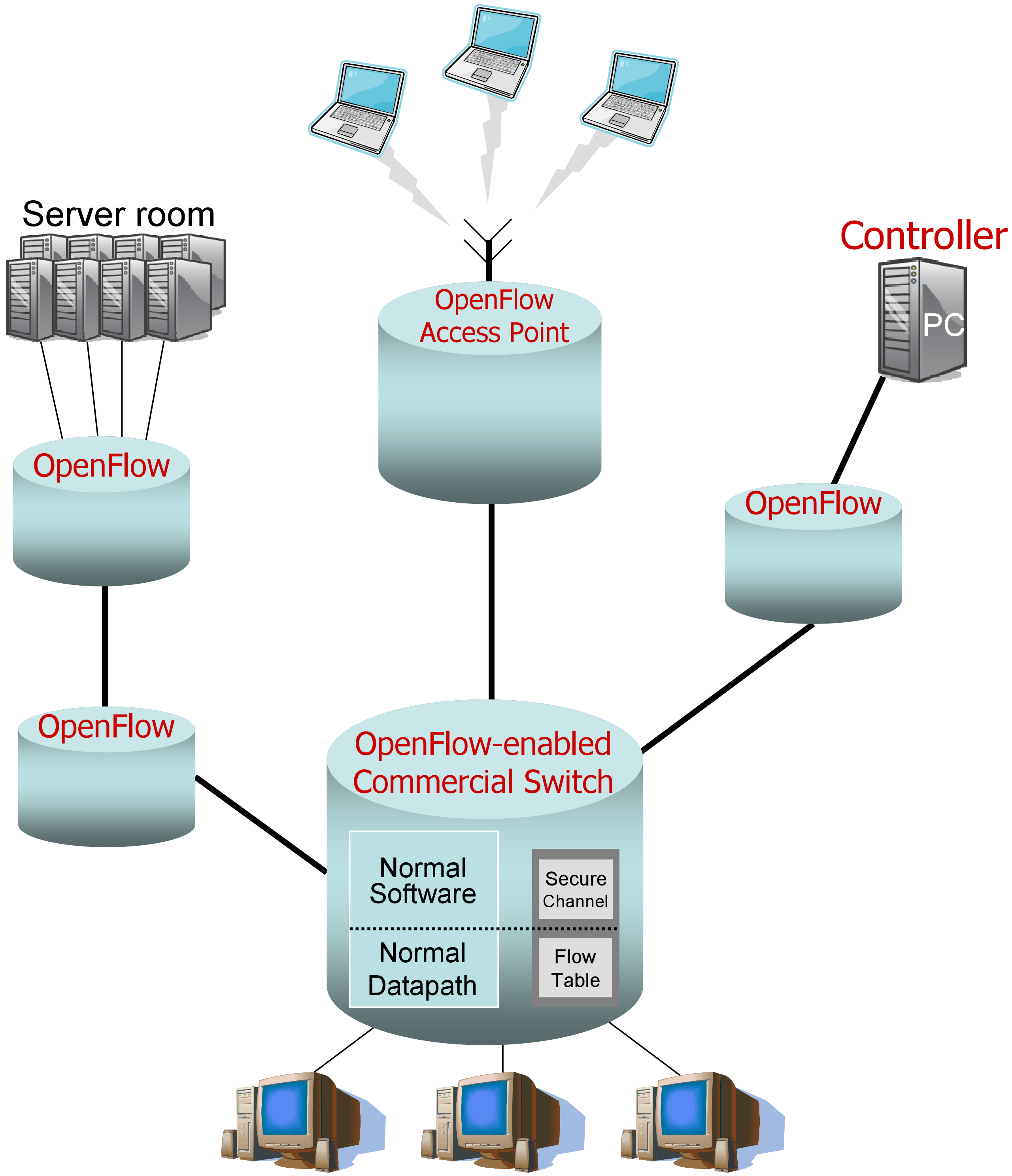


an open protocol to program different switches and routers

OpenFlow overview



OpenFlow overview



identify common functions

- flow-tables
  - implement FW/NAT/QoS, collect statistics
- secure channel to controller
- OpenFlow protocol
  - open, standard switch-controller communication

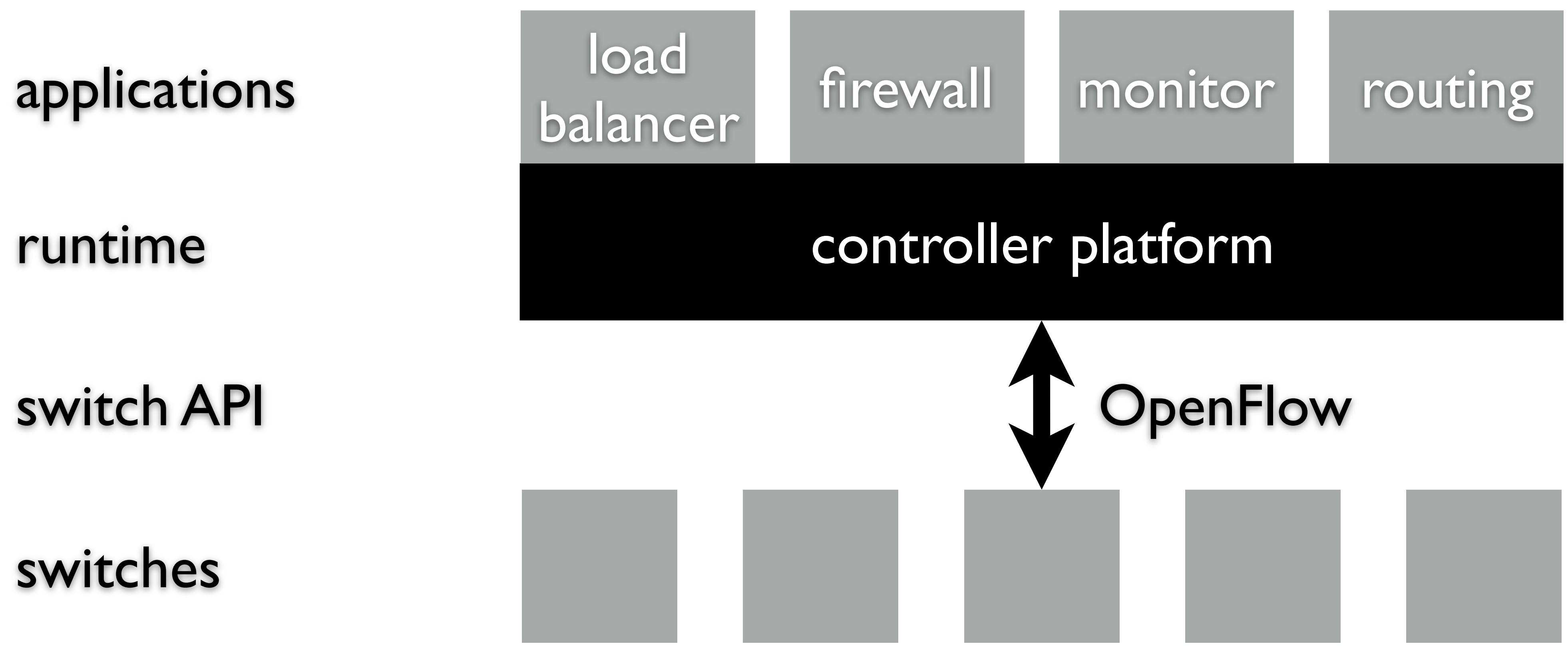
OpenFlow in action

goal: experiments in production network

- production traffic routed using some standard protocol
- Amy testing innovations on her isolated traffic

solution

- OpenFlow-enabled switch for production traffic
- controller assured to isolate Amy's traffic


applications
runtime
switch API
switches
load balancer
firewall
monitor
routing
controller platform
OpenFlow

but OpenFlow is hard to program

but OpenFlow is hard to program

low-level programming interface

- akin to assembly language: a thin “wrapper” around switch operations

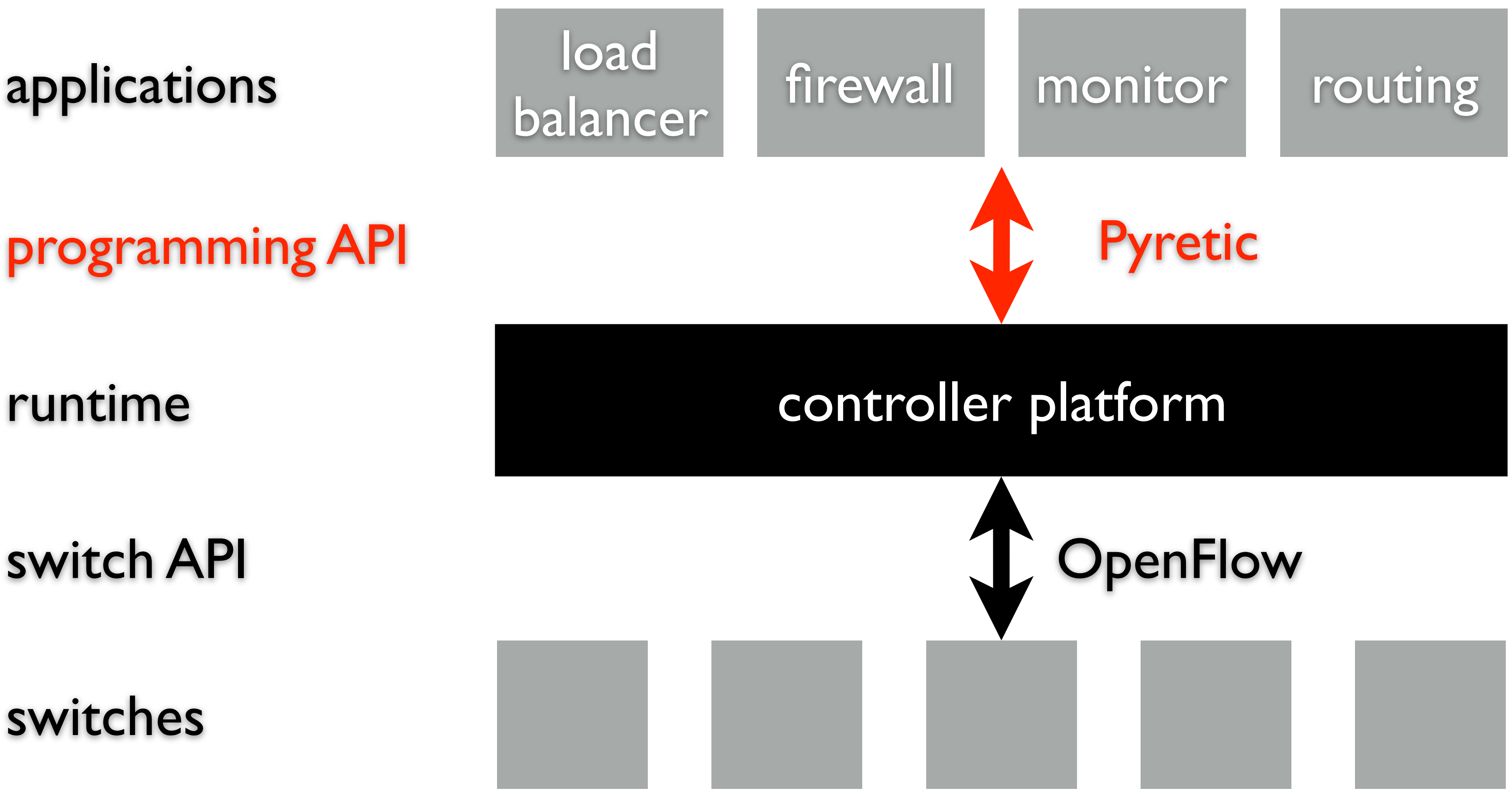
but OpenFlow is hard to program

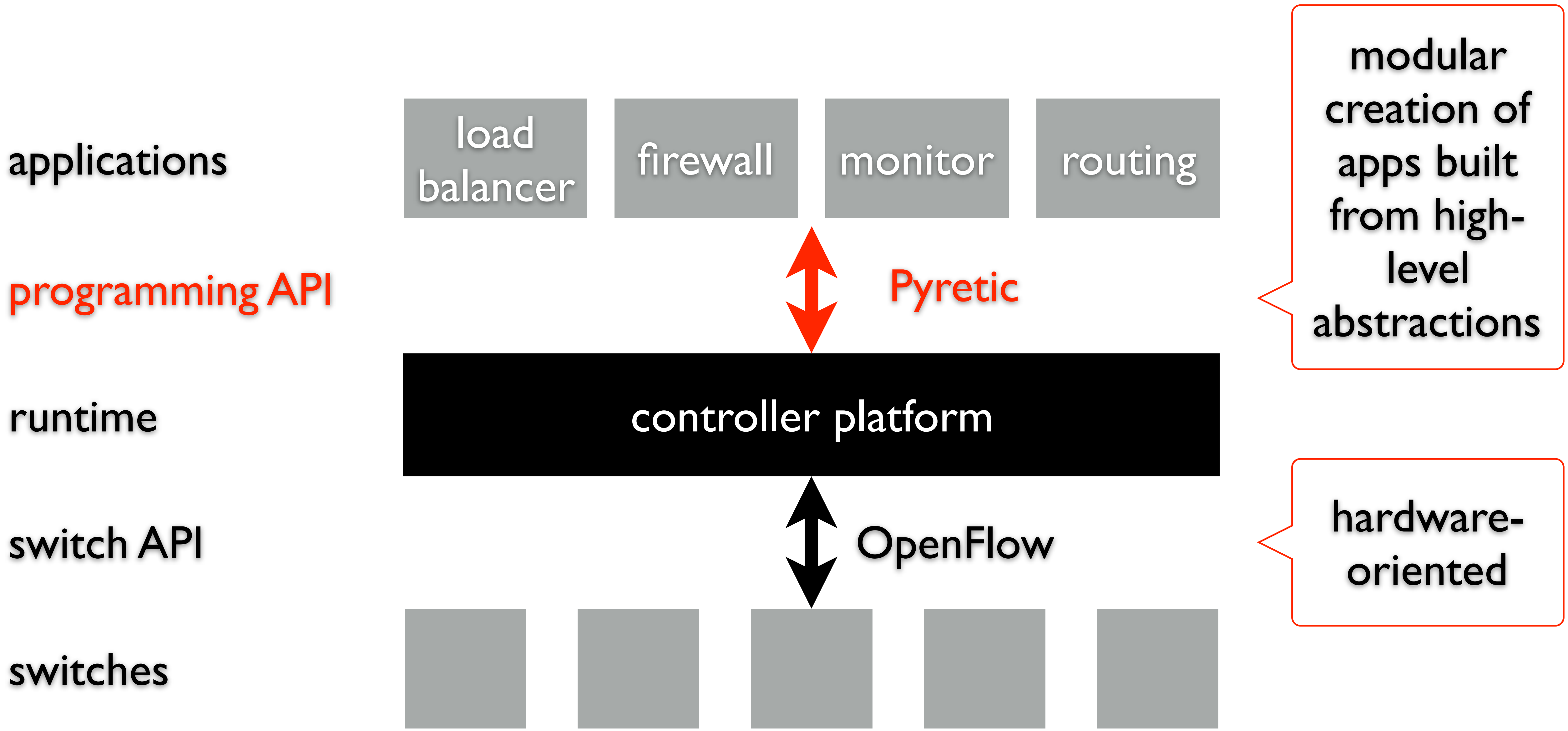
low-level programming interface

- akin to assembly language: a thin "wrapper" around switch operations

monolithic applications with intertwined logic

- handlers that respond to events
  - packet arrival
  - topology changes
  - traffic statistics


applications
load balancer
firewall
monitor
routing
programming API
Pyretic
runtime
controller platform
switch API
OpenFlow
switches


applications
load balancer
firewall
monitor
routing
modular creation of apps built from high-level abstractions
programming API
Pyretic
runtime
controller platform
switch API
OpenFlow
hardware-oriented
switches

Pyretic

Pyretic language and system

creating a single application out of multiple, independent, reusable network policies that affect the processing of the "same" traffic

Pyretic language and system

creating a single application out of multiple, independent, reusable network policies that affect the processing of the "same" traffic

Pyretic language and system

the enabling constructs and mechanisms

- high level abstraction
- composition
- abstract network topology

implementation

- an interpreter that handles each packet at the controller (POX)

Pyretic language and system

the enabling constructs and mechanisms

- high level abstraction
- composition
- abstract network topology

implementation

- an interpreter that handles each packet at the controller (POX)

from OF rules to functions

OF like rules at a switch *s*:

patten (field =value) ➡ action

↓

a function:

takes as input a packet on a particular port on *s*, outputs a multiset of zero or more packets on various outports of *s*

policy as functions

a function:

takes as input a packet on a particular port on s, outputs a multiset of zero or more packets on

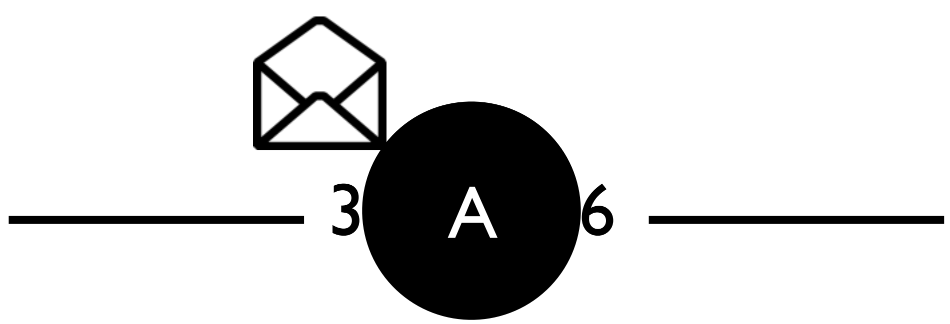
a network-wide policy function:

locate packets ➡ located packets

abstract packet model

the "located packet" model

- a packet is a
  {switch: A, inport: 3, ...}



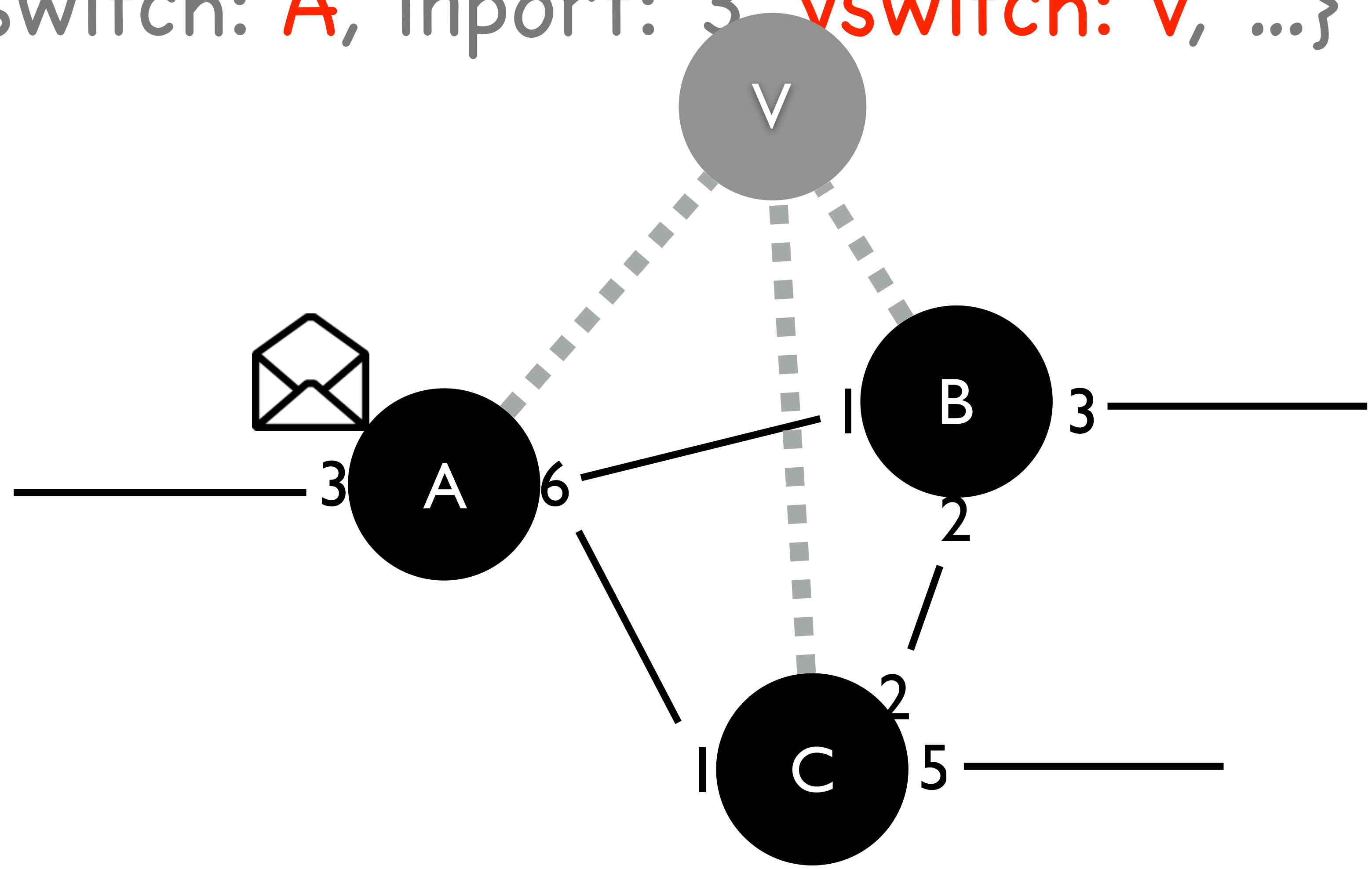
abstract packet model

the "located packet" model

- a packet is a

  {switch: [V,A], inport: 3, ...}

  {switch: A, inport: 3, vswitch: V, ...}



abstract packet model

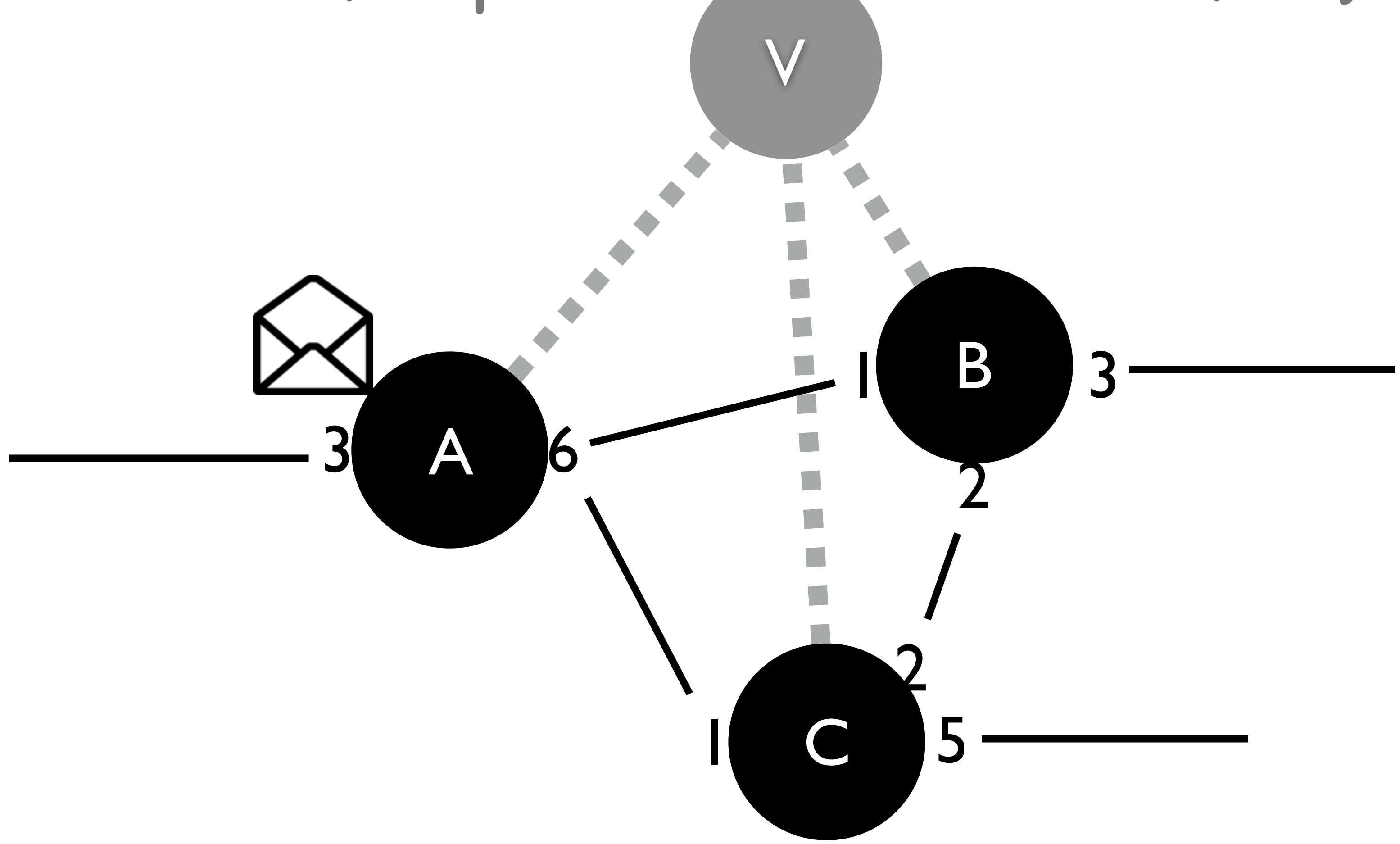
the "located packet" model

- a packet is a

  {switch: [V,A], inport: 3, ...}

  {switch: A, inport: 3, vswitch: V, ...}

location information



abstract packet model

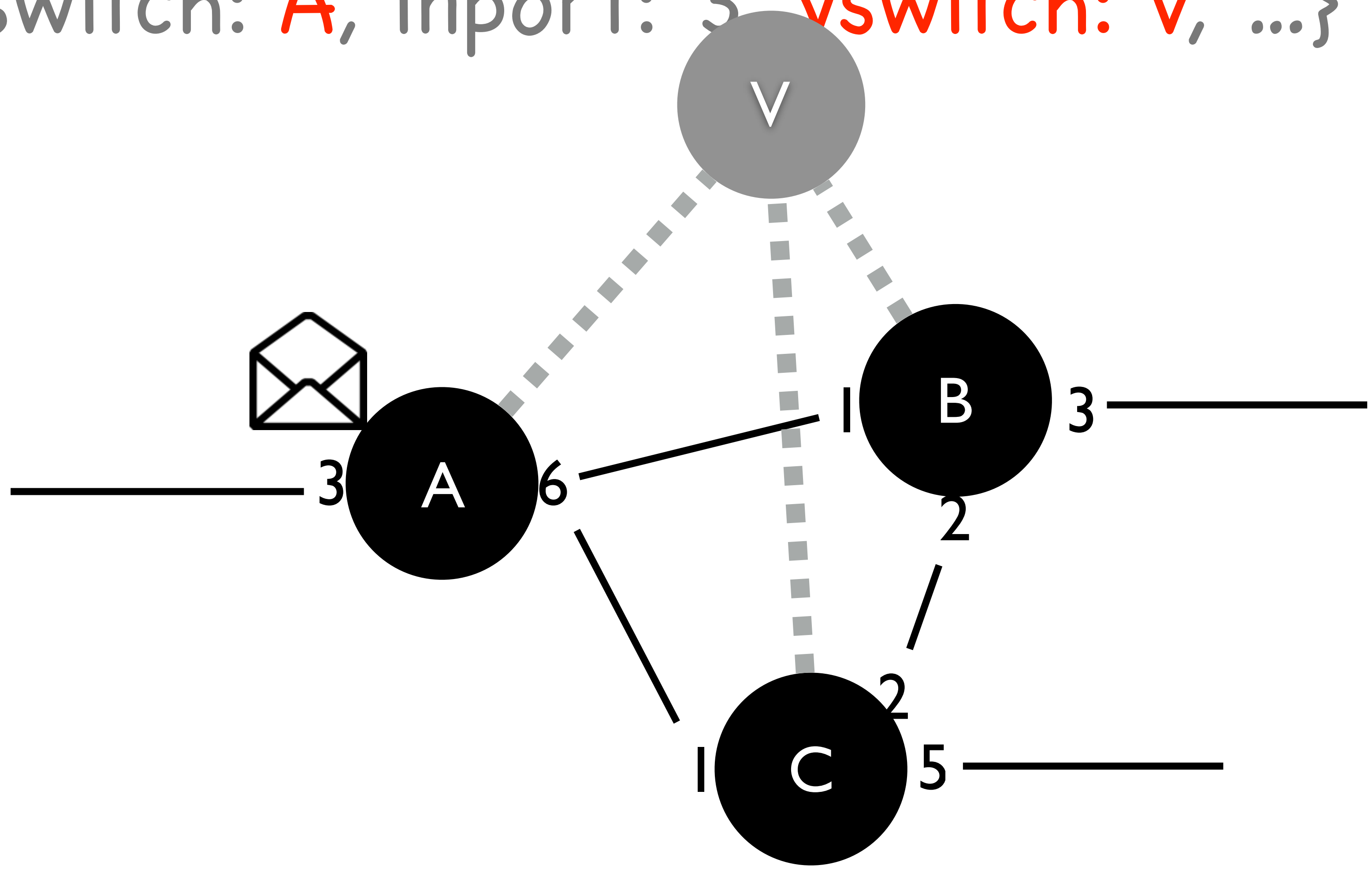
the “located packet” model

- a packet ✉ is a

  {switch: [V,A], inport: 3, ...}

  {switch: A, inport: 3, vswitch: V, ...}

location information



more: IP addresses, MAC addresses …

Pyretic policies

locate packets ➡ located packets

static policy

- a snapshot of a network's global forwarding behavior
- an abstract function

dynamic policy

- a series of static policies

static policy (simplified)

define policy

$C ::= A \mid P[C] \mid C|C \mid C\gg C$

static policy (simplified)

static policy (simplified)

define policy

static policy (simplified)

define policy

C ::= A | P**[**C**]** | C**|**C | C»C

static policy (simplified)

define policy

$C ::= A \mid P[C] \mid C|C \mid C » C$

define $C_1 » C_2$ as C_3

- $C_3(\text{packet}) =$
- $C_1(p_1) \cup \ldots \cup C_2(P_n)$ where $\{P_1, \ldots P_n\} = C_1(\text{packet})$

static policy (simplified)

define policy

$C ::= A \mid P[C] \mid C \| C \mid C \gg C$

define $C_1 \| C_2$ as C_3

- C_3(packet) = C_1(packet) U C_2(Packet)

query policy

define policy

C ::= A | P[C] | C|C | C»C | Q

Q ::= packets | count

packet, count buckets

- resulting located packets diverted to “buckets” in the controller
- application registers listeners with buckets
- buckets passes entire packets to the listeners

example sequential composition

Monitor
srcip=5.6.7.8 $\rightarrow$ count

Route
dstip=10.0.0.1 $\rightarrow$ fwd(1)
dstip=10.0.0.2 $\rightarrow$ fwd(2)

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

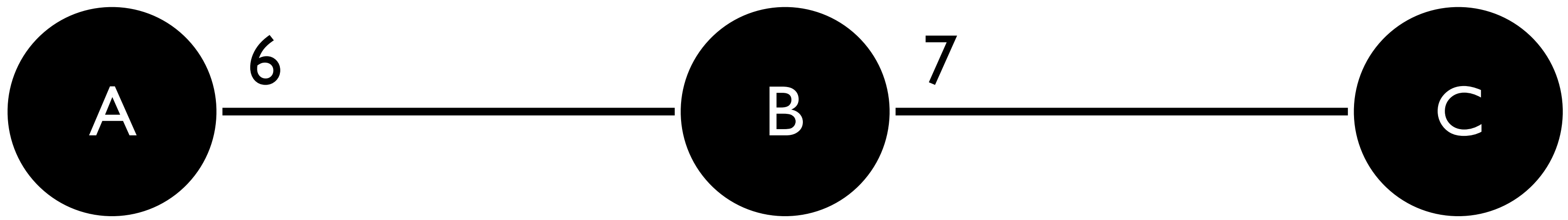
example parallel composition

Route
`dstip=10.0.0.1` $\rightarrow$ `fwd(1)`
`dstip=10.0.0.2` $\rightarrow$ `fwd(2)`

Load-balance
`srcip=0*,dstip=1.2.3.4` $\rightarrow$ `dstip=10.0.0.1`
`srcip=1*,dstip=1.2.3.4` $\rightarrow$ `dstip=10.0.0.2`

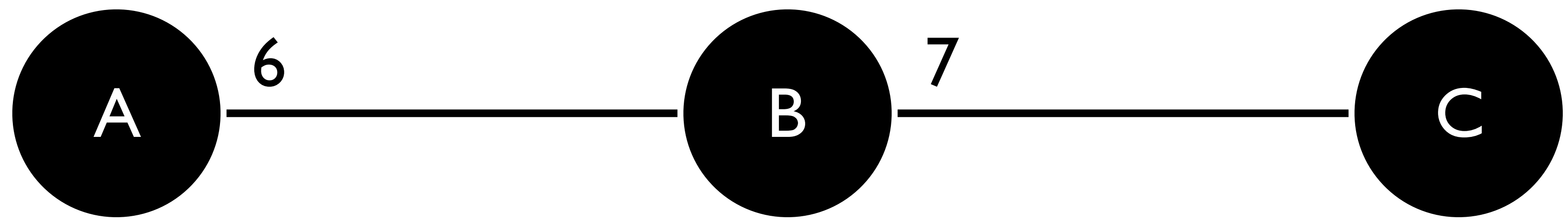
Compiled Prioritized Rule Set for "Load-balance >> Route"
`srcip=0*,dstip=1.2.3.4` $\rightarrow$ `dstip=10.0.0.1,fwd(1)`
`srcip=1*,dstip=1.2.3.4` $\rightarrow$ `dstip=10.0.0.2,fwd(2)`

limitation to Pyretic policies



match(switch=A) & match(dstip='C') » fwd(6) +
match(switch=B) & match(dstip='C') » fwd(7)

limitation to Pyretic policies



match(switch=A) & match(dstip='C') » fwd(6) +
match(switch=B) & match(dstip='C') » fwd(7)

programmers must specify policies in terms of the underlying physical topology

limitation to Pyretic policies

abstract network topology

allow a new derived topology to be built on top of an already existing existing underlying network

limitation to Pyretic policies

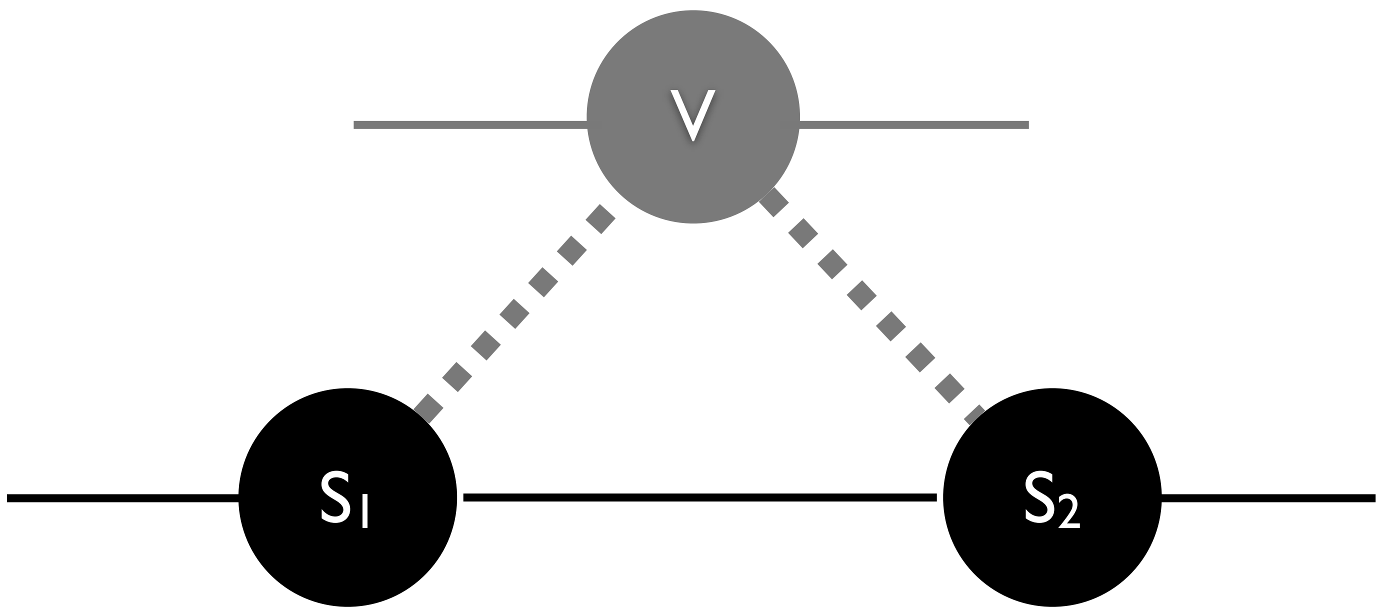
programmers must specify policies in terms of the underlying physical topology

↓

abstract network topology

allow a new derived topology to be built on top of an already existing existing underlying network

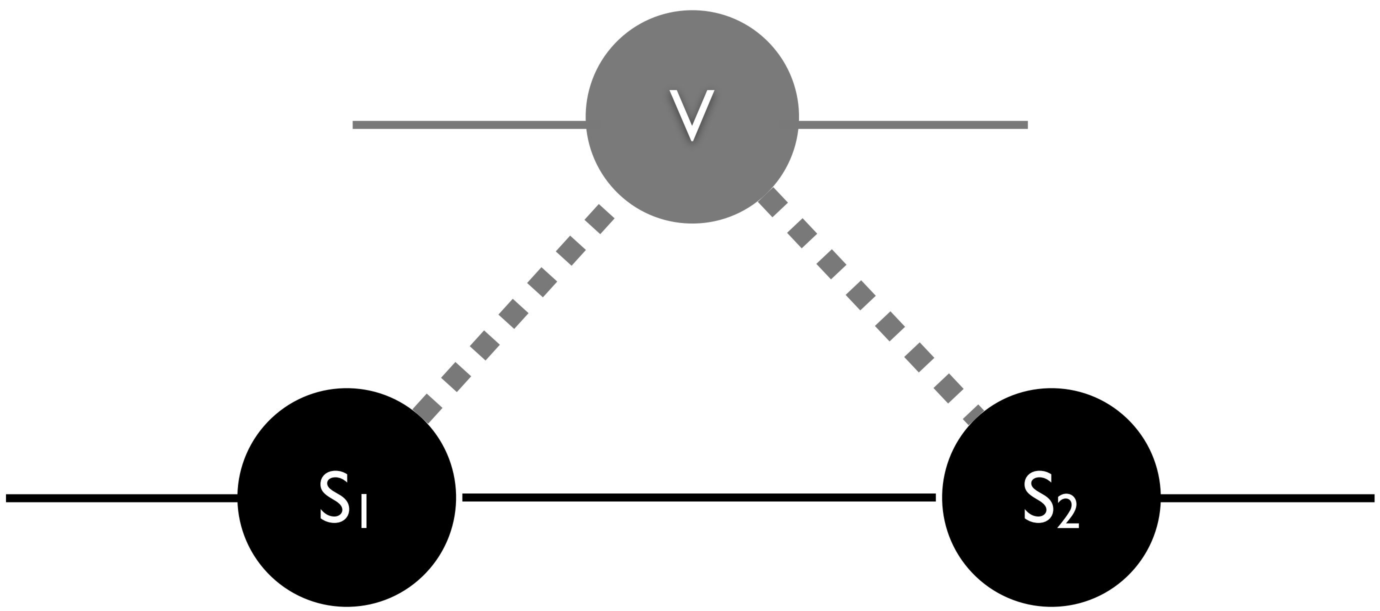
derived network



Pyretic network objects

- a topology
- a policy
- a mapping (for derived network)

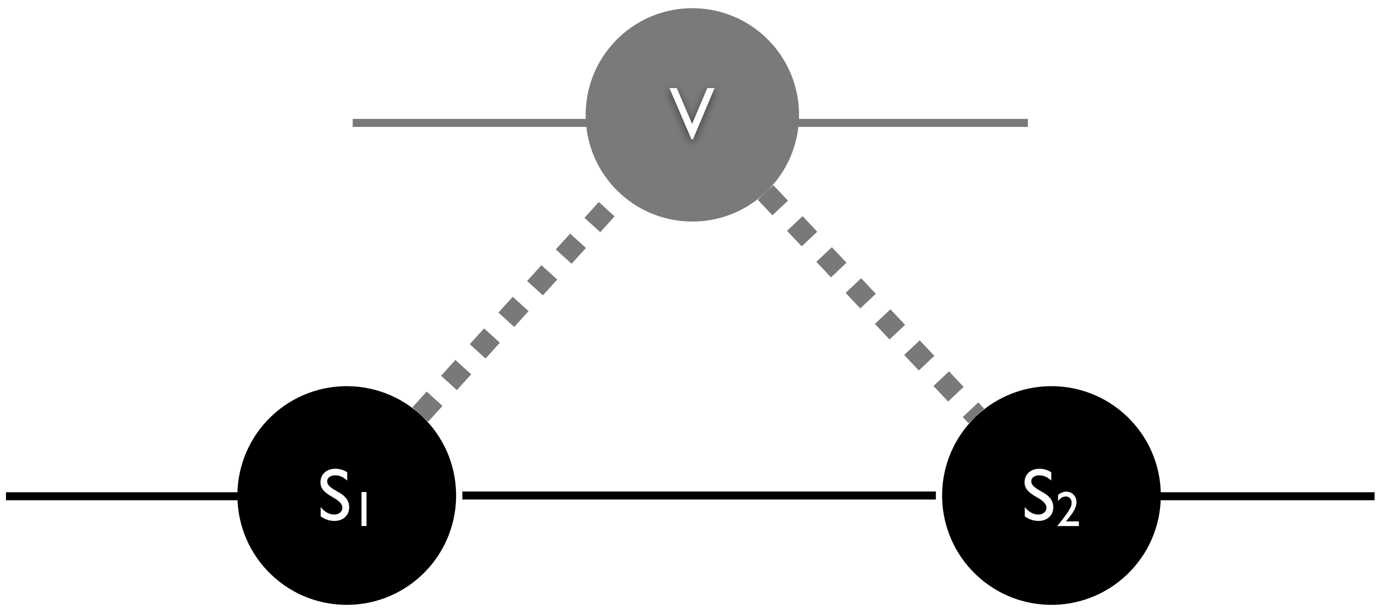
derived network



mapping

- a function to map changes to the underlying topology up to changes on the derived network
- a function to map policies against the derived topology down to equivalent policy expressed only in terms of the underlying topology

derived network



mapping — user inputs (program spec)

- mapping between elements of the topologies
- a function for calculating forwarding paths through the underlying topology

discussion

composition

- Pyretic composes the control logic that affects the handling of traffic on an entire network of OF switches

orchestration

- Maestro, allows programmers to write applications in terms of user-defined views of network state