How do we talk to the network?
(Thinking about the interface to the network)

Rodrigo Fonseca

Brown University

Oct 27th, 2016

(First: What should you know about me?)

Brown University

Berkeley

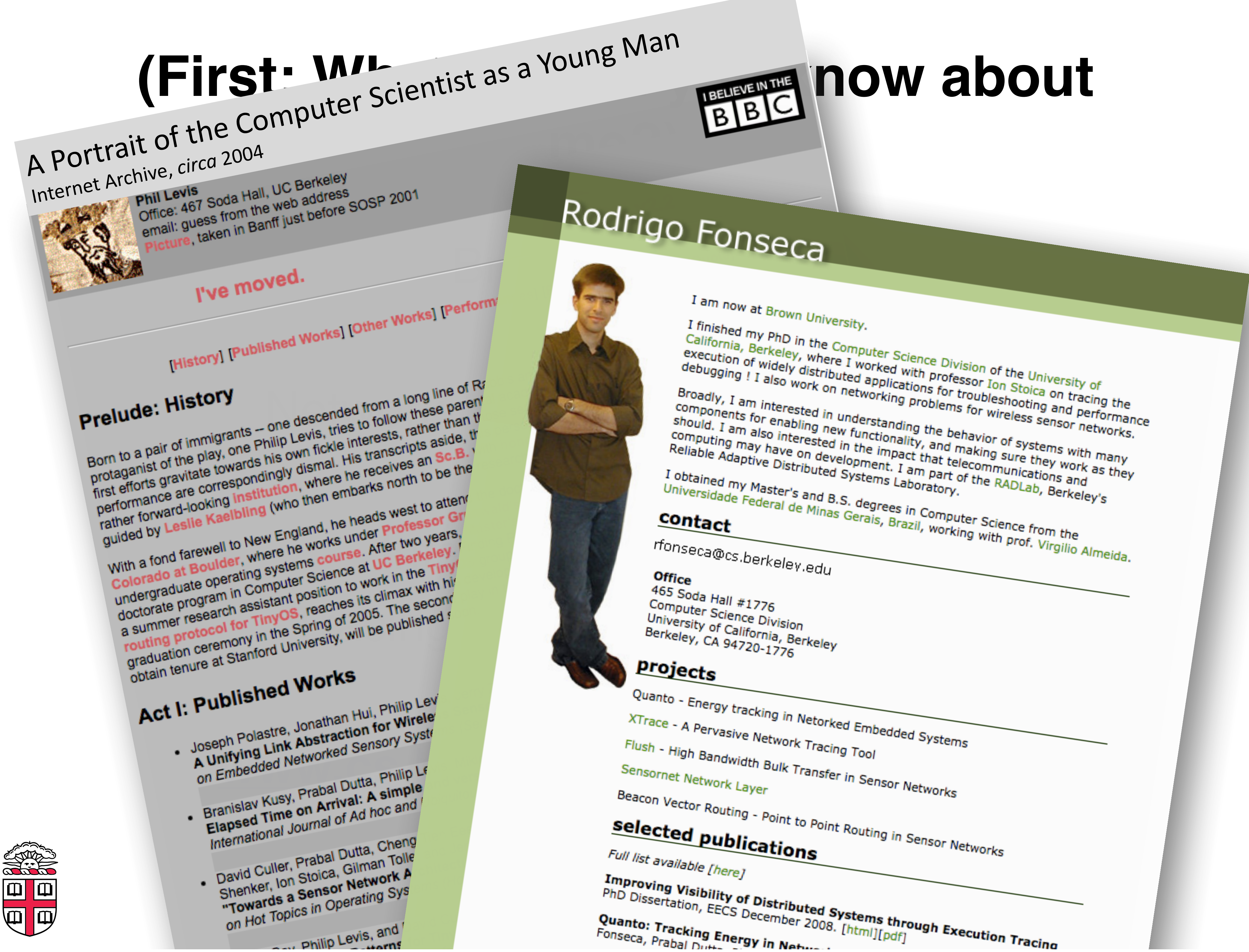
Networking & Distributed Systems

Interfaces, Abstractions

Tracing, Energy, Networking, …

www.cs.brown.edu/people/rfonseca

(First: Wh now about



A Portrait of the Computer Scientist as a Young Man
Internet Archive, *circa* 2004

Phil Levis
Office: 467 Soda Hall, UC Berkeley
email: guess from the web address
Picture, taken in Banff just before SOSP 2001

I've moved.

[History] [Published Works] [Other Works] [Perform

Prelude: History

Born to a pair of immigrants -- one descended from a long line of Ra
protaganist of the play, one Philip Levis, tries to follow these parent
first efforts gravitate towards his own fickle interests, rather than th
performance are correspondingly dismal. His transcripts aside, th
rather forward-looking institution, where he receives an Sc.B. v
guided by Leslie Kaelbling (who then embarks north to be the

With a fond farewell to New England, he heads west to atten
Colorado at Boulder, where he works under Professor Gr
undergraduate operating systems course. After two years,
doctorate program in Computer Science at UC Berkeley.
a summer research assistant position to work in the Tiny
routing protocol for TinyOS, reaches its climax with hi
graduation ceremony in the Spring of 2005. The secon
obtain tenure at Stanford University, will be published

Act I: Published Works

- Joseph Polastre, Jonathan Hui, Philip Lev
  A Unifying Link Abstraction for Wirele
  on Embedded Networked Sensory Syst
- Branislav Kusy, Prabal Dutta, Philip Le
  Elapsed Time on Arrival: A simple
  International Journal of Ad hoc and
- David Culler, Prabal Dutta, Cheng
  Shenker, Ion Stoica, Gilman Tolle
  "Towards a Sensor Network A
  on Hot Topics in Operating Sys
- Philip Levis, and
  Patterns

Rodrigo Fonseca

I am now at Brown University.

I finished my PhD in the Computer Science Division of the University of California, Berkeley, where I worked with professor Ion Stoica on tracing the execution of widely distributed applications for troubleshooting and performance debugging ! I also work on networking problems for wireless sensor networks.

Broadly, I am interested in understanding the behavior of systems with many components for enabling new functionality, and making sure they work as they should. I am also interested in the impact that telecommunications and computing may have on development. I am part of the RADLab, Berkeley's Reliable Adaptive Distributed Systems Laboratory.

I obtained my Master's and B.S. degrees in Computer Science from the Universidade Federal de Minas Gerais, Brazil, working with prof. Virgilio Almeida.

contact

rfonseca@cs.berkeley.edu

Office
465 Soda Hall #1776
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720-1776

projects

Quanto - Energy tracking in Netorked Embedded Systems

XTrace - A Pervasive Network Tracing Tool

Flush - High Bandwidth Bulk Transfer in Sensor Networks

Sensornet Network Layer

Beacon Vector Routing - Point to Point Routing in Sensor Networks

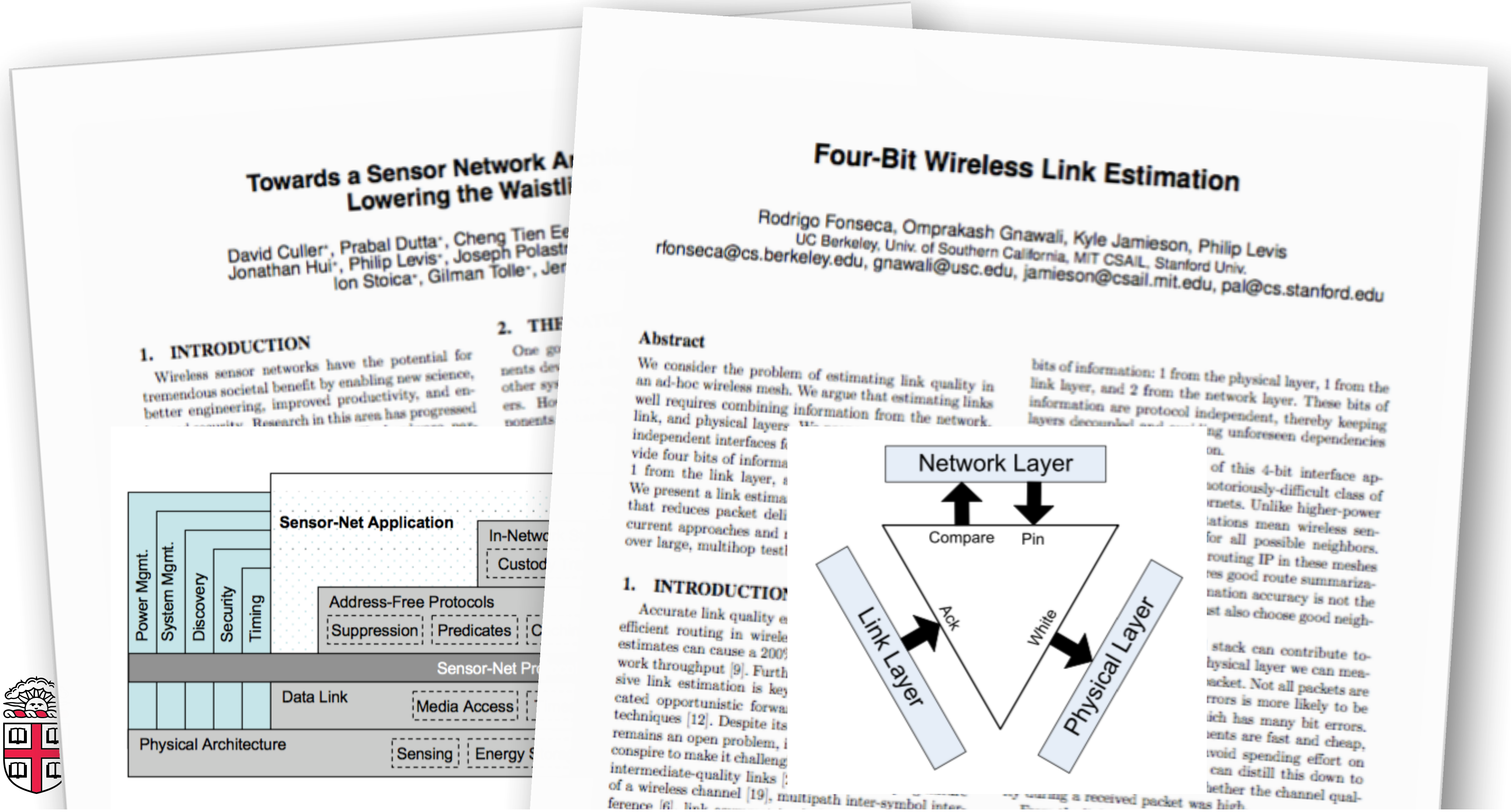
selected publications

Full list available [here]

Improving Visibility of Distributed Systems through Execution Tracing
PhD Dissertation, EECS December 2008. [html][pdf]

Quanto: Tracking Energy in Netw
Fonseca, Prabal Dutt

(First: What should you know about me?)



Towards a Sensor Network Ar
Lowering the Waistli

David Culler*, Prabal Dutta*, Cheng Tien Ee
Jonathan Hui*, Philip Levis*, Joseph Polastr
Ion Stoica*, Gilman Tolle*, Jer

1. INTRODUCTION

Wireless sensor networks have the potential for tremendous societal benefit by enabling new science, better engineering, improved productivity, and en- Research in this area has progressed

2. THE

One g nents dev other sy ers. Ho ponents

Four-Bit Wireless Link Estimation

Rodrigo Fonseca, Omprakash Gnawali, Kyle Jamieson, Philip Levis
UC Berkeley, Univ. of Southern California, MIT CSAIL, Stanford Univ.
rfonseca@cs.berkeley.edu, gnawali@usc.edu, jamieson@csail.mit.edu, pal@cs.stanford.edu

Abstract

We consider the problem of estimating link quality in an ad-hoc wireless mesh. We argue that estimating links well requires combining information from the network, link, and physical layers. independent interfaces f vide four bits of informa 1 from the link layer, We present a link estima that reduces packet deli current approaches and over large, multihop test

bits of information: 1 from the physical layer, 1 from the link layer, and 2 from the network layer. These bits of information are protocol independent, thereby keeping layers decoupled and ing unforeseen dependencies on. of this 4-bit interface ap notoriously-difficult class of rnets. Unlike higher-power ations mean wireless sen for all possible neighbors. routing IP in these meshes res good route summariza mation accuracy is not the st also choose good neigh-

1. INTRODUCTIO

Accurate link quality e efficient routing in wirele estimates can cause a 200 work throughput [9]. Furth sive link estimation is key cated opportunistic forwa techniques [12]. Despite its remains an open problem, i conspire to make it challeng intermediate-quality links [of a wireless channel [19], multipath inter-symbol inter- ference [6], link

stack can contribute to hysical layer we can mea acket. Not all packets are rrors is more likely to be ich has many bit errors. ents are fast and cheap, void spending effort on can distill this down to hether the channel qual- during a received packet was high

How do we talk to the network?

- **When we send packets over TCP/IP, what do we tell the network?**
  - Not too much!
- **TCP/IP (to a first approximation):**
  - Treats all packets the same
  - Treats all flows the same
    - Will be “fair” to flows
  - Lets everyone talk to everyone
- **This talk: different ways to talk to the network**

  “Participatory Networking”

Can/should applications change/ choose the behavior of the network?

IP

- **Lowest common denominator**
- **Best effort**
- **No differentiation**
  - (at initially, none with global scope)
- **Principles**
  - Design must scale
  - Keep it simple
  - Modularity is good
  - Don’t impose costs of features unneeded by some

https://tools.ietf.org/html/rfc1958 ‘Architectural Principles of the Internet’
Clark, D., “The Design Philosophy of the DARPA Internet Protocols”

Over the years, many other proposals

Question (for you): Why would you want end-users/applications to express their needs to the network?

IP's model not the only option

- **ATM (early 90's, competitor to IP)**
  - Supposed to unify data and traditional telecommunications
  - Virtual circuit-based
  - Constant/Variable/Available/Unspecified Bit Rate
- **Integrated Services**
  - Per-flow QoS guarantees across the Internet
  - Absolute guarantees
- **Differentiated Services**
  - Class of service (coarse)
  - Relative QoS

Active Networking (late 90’s)

- **End-user programmability of the network**
- **Radical change to the network API**
  - Packets would carry code (or pointer to code)
  - Users could choose which programs to run
- **Examples**
  - Multicast, application-specific QoS, information fusion, caching
- **Potential problems**
  - Protection among programs, exploitation of state in routers, global coordination (for non-local properties), misbehaving applications (e.g., forming loops)
  - No killer app

Many more proposals

- **E.g., congestion/rate control**
  - Great results if you know priorities, deadlines
  - PDQ, D3, D2CTCP, pFabric, QJump, …
  - Mostly extend the API in-band

Thorny questions

- **Do users really know what they want?**
- **What should an interface be like?**
- **On the Internet:**
  - Do users trust/care/know about each other?
  - What is the incentive to *not* say your traffic is important?
  - Business models: users really like flat rates
- **Easier (but not easy):**
  - Datacenters, single company, home network, …

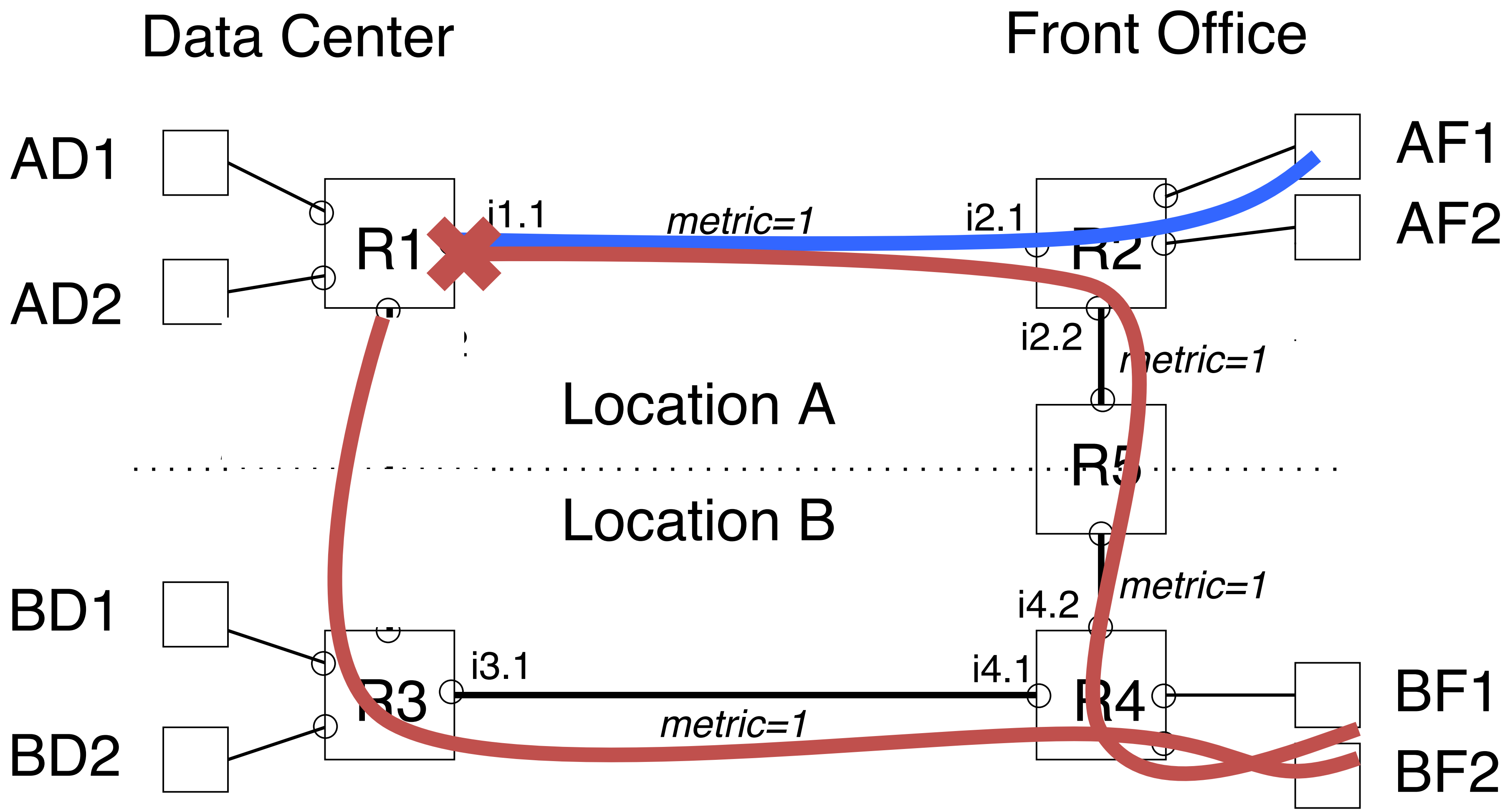
Hard to answer without doing,

hard to do as some mechanisms require consensus and changes to the network

Meanwhile…

- **Administrators were having a really hard time managing their networks**
  - Complex control plane protocols
  - Indirect ways to achieve policies
    - E.g., tweaking weights in routing protocols
  - Access control very hard to get right
  - With a pressure to scale AND become cheaper

Enterprise Network



(Too) Many Control Plane Mechanisms

- **Designed from scratch for specific goal**
- **Variety of goals, no modularity:**
  - **Routing:** distributed routing algorithms
  - **Isolation**: ACLs, VLANs, Firewalls,…
  - **Traffic engineering**: adjusting weights, MPLS,…
- **Variety of implementations**
  - **Globally distributed:** routing algorithms
  - **Manual/scripted configuration:** ACLs, VLANs
  - **Centralized computation:** Traffic engineering
- **No abstractions**
- **Network control plane is a complicated mess!**

Abstractions for the Control Plane

- **A number of projects in the early 2000's started talking about breaking the problem into simpler components**
  - Including Nick's group

How do you find abstractions?

- You first decompose the problem….

- …and define abstractions for each subproblem

- So what is the control plane problem?

Task: Compute forwarding state…

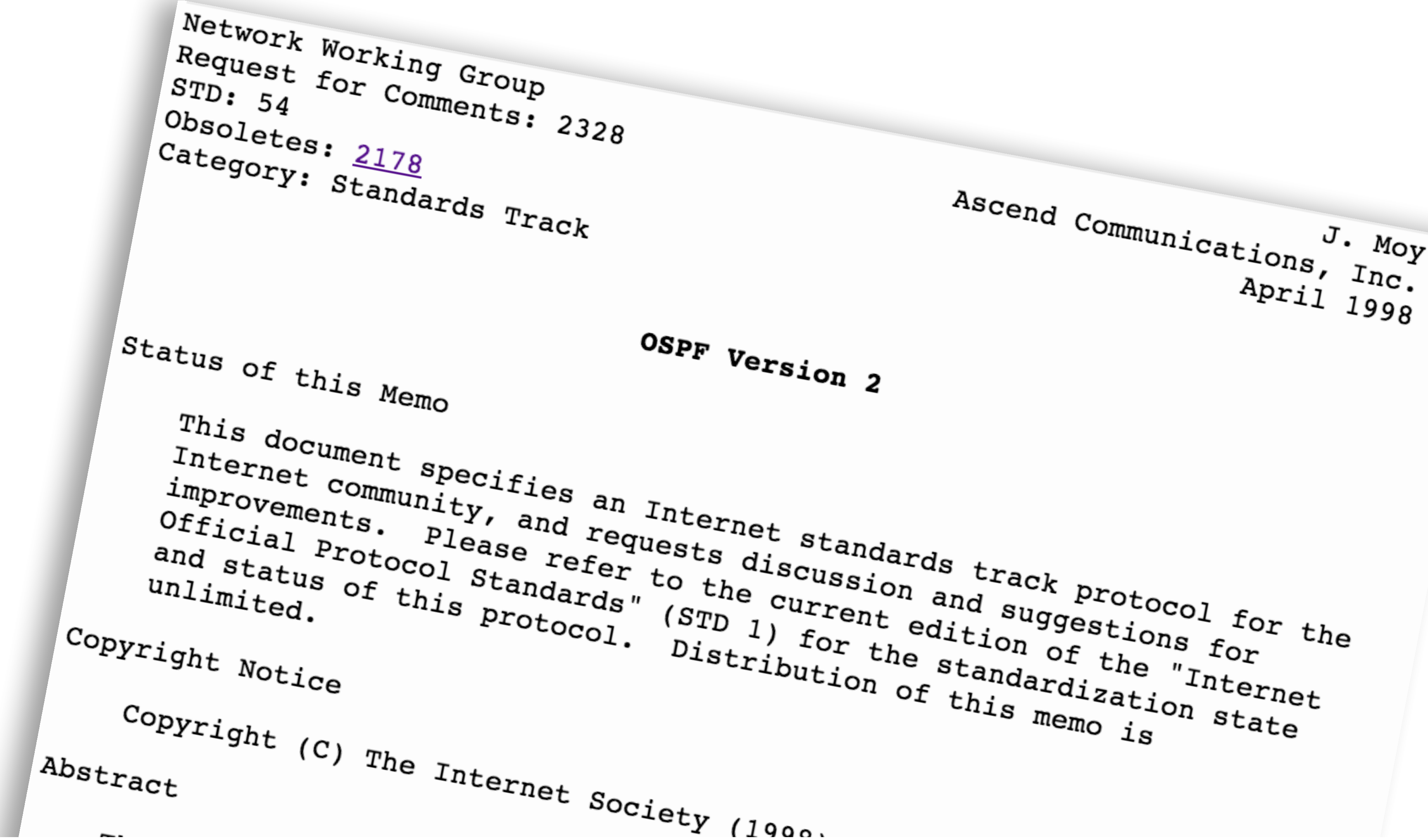
- **Consistent with low-level hardware/ software**
  - Which might depend on particular vendor
- **Based on entire network topology**
  - Because many control decisions depend on topology
- **For all routers/switches in network**
  - Every router/switch needs forwarding state

Previous approach

- **Design one-off mechanisms that solve all three**
  - A sign of how much we love complexity
- **No other field would deal with such a problem!**
- **They would define abstractions for each subtask**

Example

- **OSPF:**
  - 5% for Djikstra’s algorithm,
  - 95% to find and maintain the state of the network



Network Working Group
Request for Comments: 2328
STD: 54
Obsoletes: 2178
Category: Standards Track

J. Moy
Ascend Communications, Inc.
April 1998

OSPF Version 2

Status of this Memo

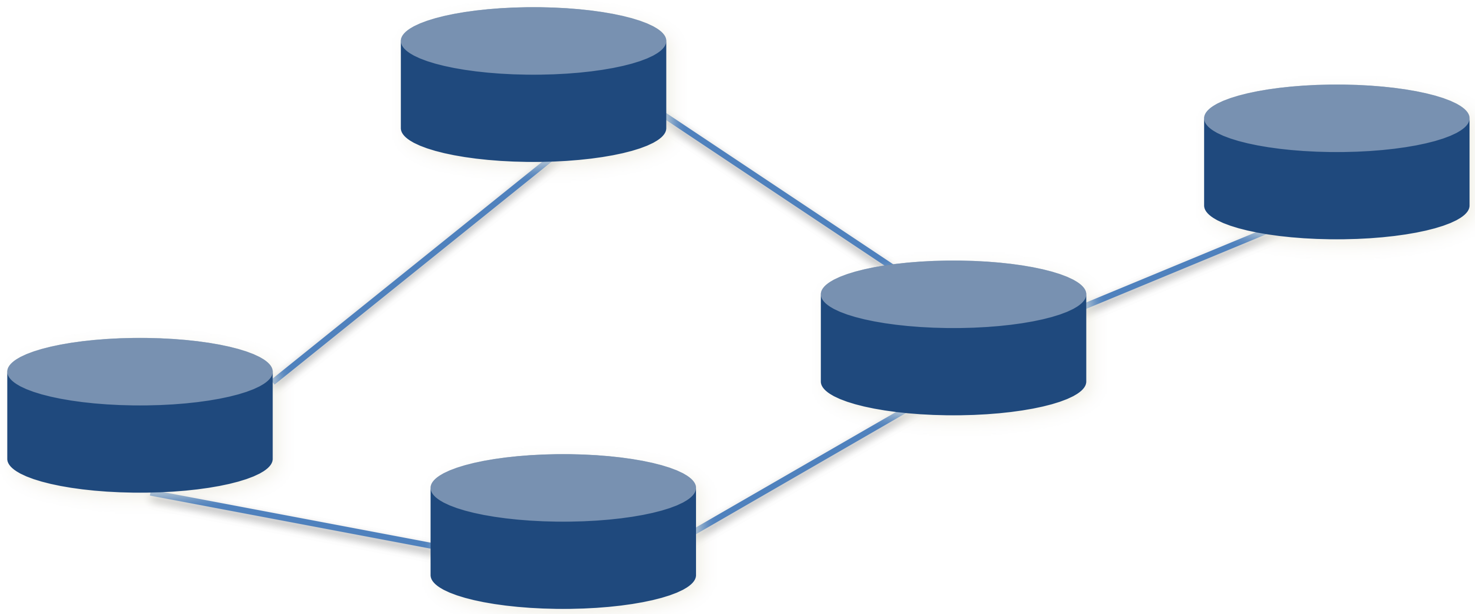
This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1998).

Abstract

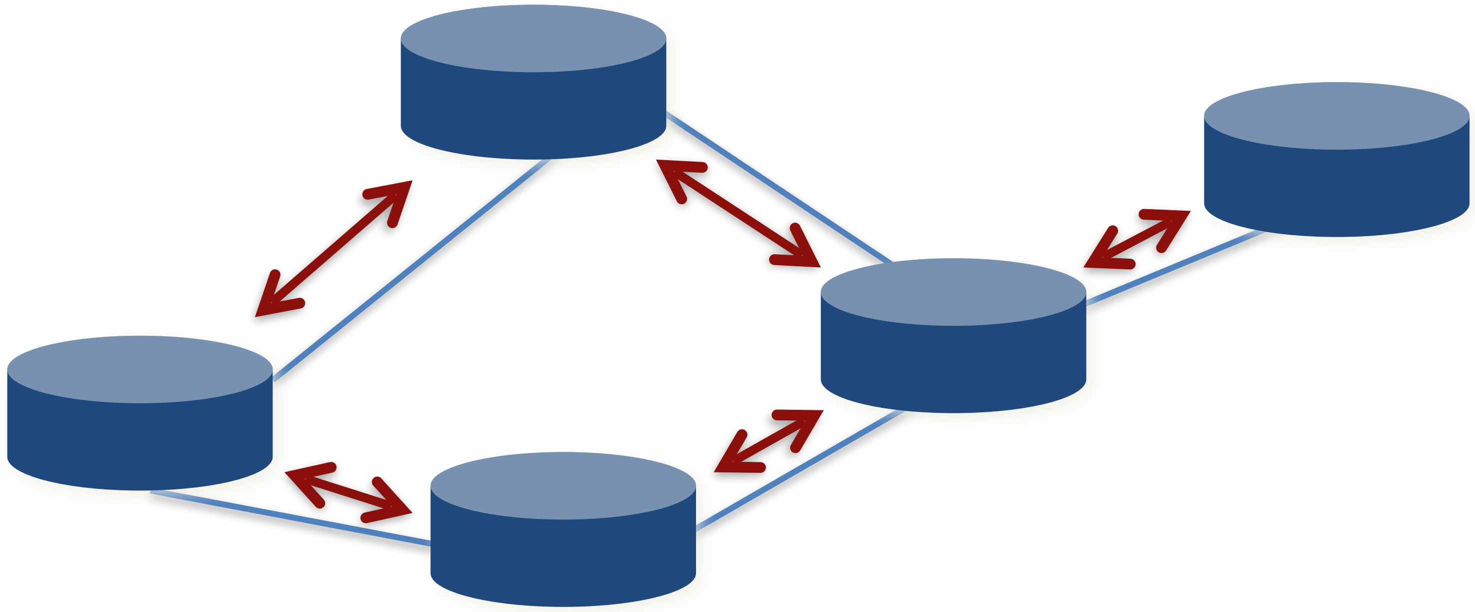
Network of Switches and/or Routers



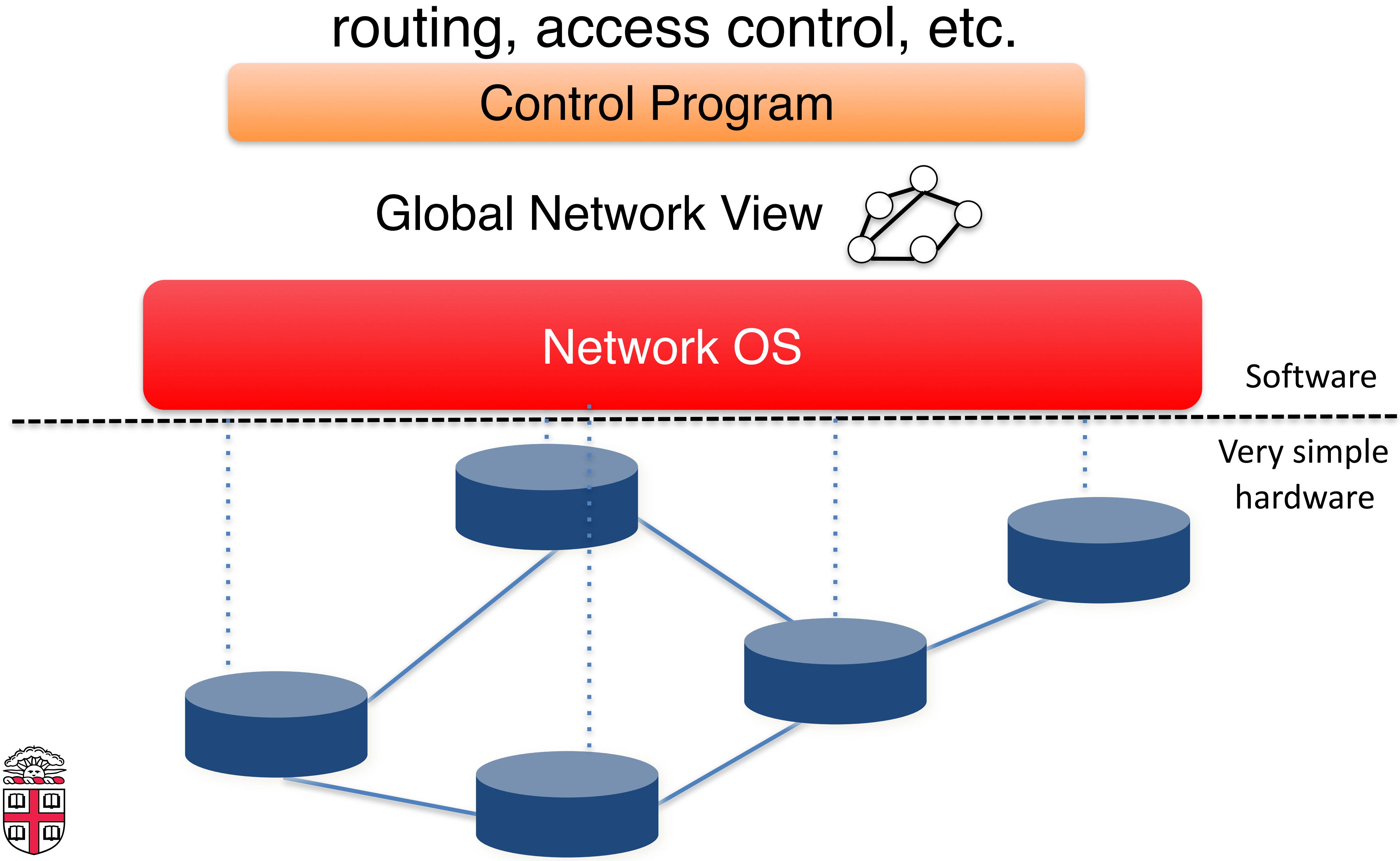
Traditional Control Mechanisms

Distributed algorithm running between neighbors

Complicated task-specific distributed algorithm



Software Defined Network (SDN)



routing, access control, etc.

Control Program

Global Network View

Network OS

Software

Very simple hardware

Major Change in Paradigm

- Control program:
  - Configuration = Function(view)
- Control mechanism now program using NOS API
- Not a distributed protocol, just a graph algorithm

Routing

- Look at graph of network
- Compute routes
- Give to SDN platform, which passes on to switches

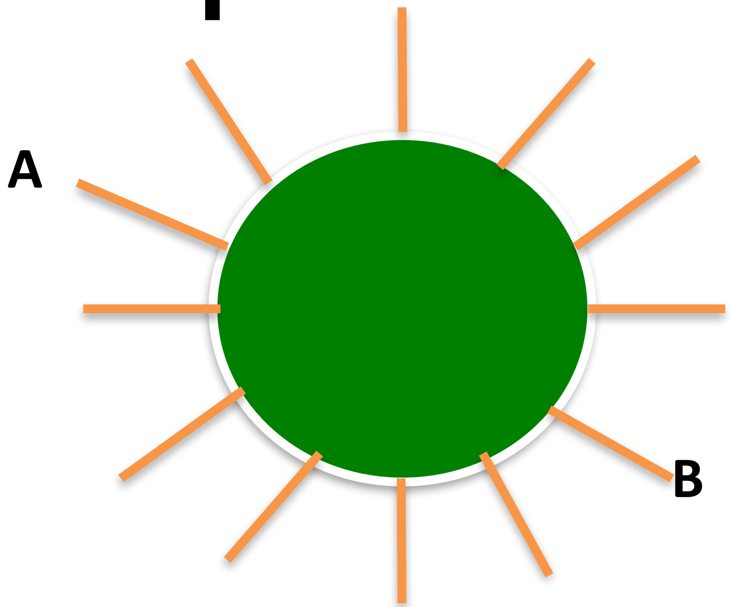
Access Control

- **Control program decides who can talk to whom**

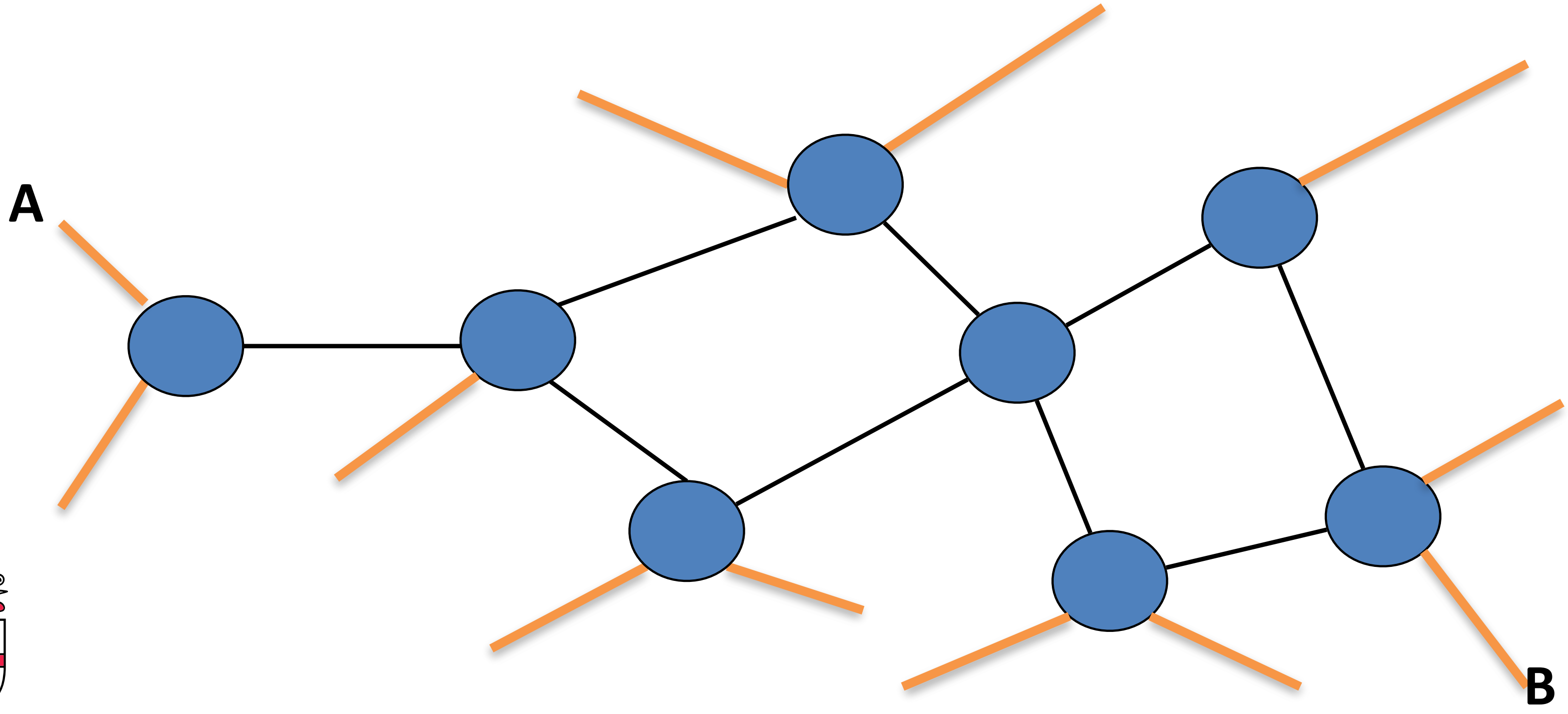
- **Pass this information to SDN platform**

- **Appropriate ACL flow entries are added to network**
  - In the right places (based on the topology)

Simple Example: Access Control

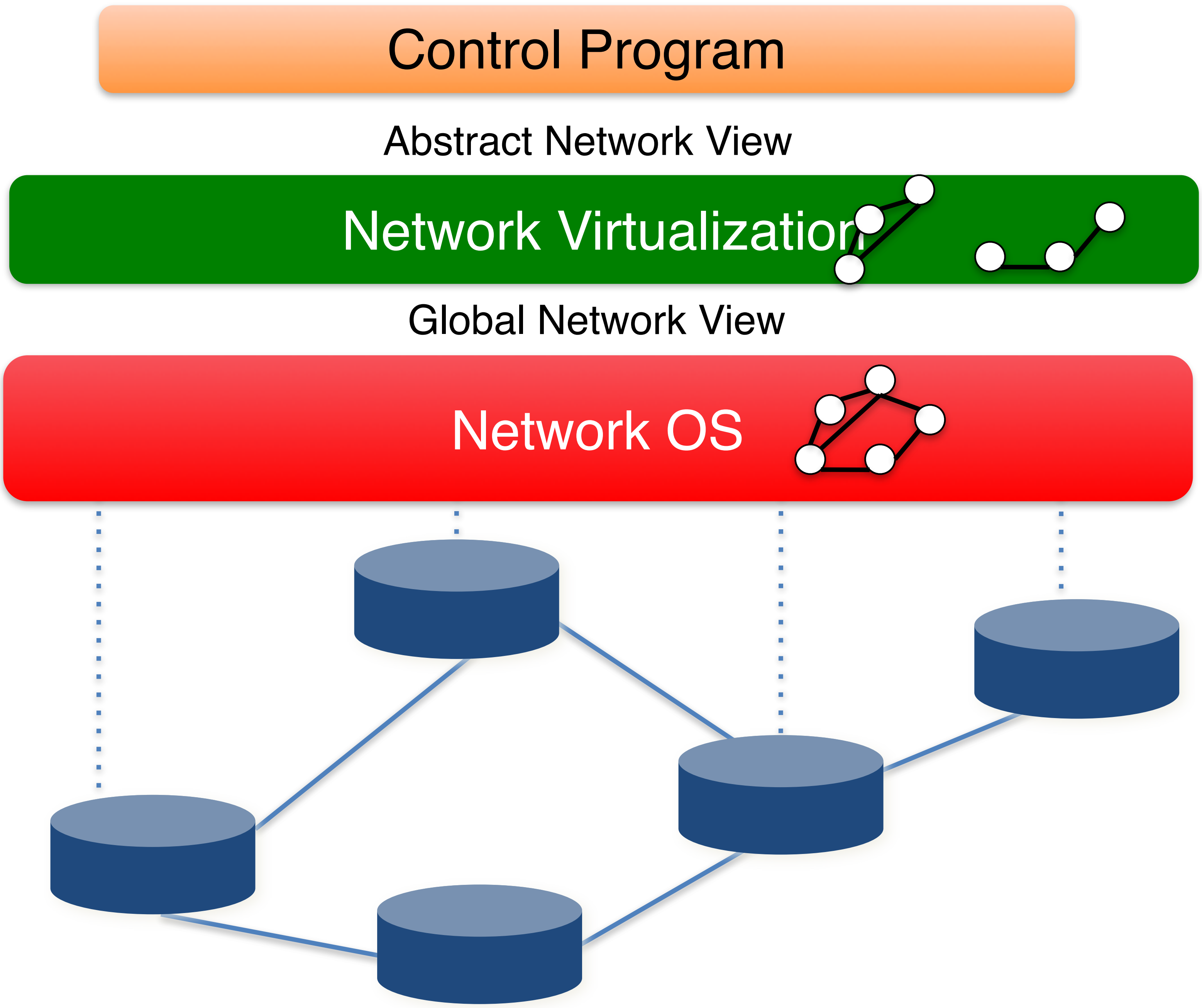


Abstract Network View



Global Network View

SDN: *Layers* for the Control Plane



Control Program

Abstract Network View

Network Virtualization

Global Network View

Network OS

Clean Separation of Concerns

- **Control program: express goals on abstract view**
  - Driven by **Operator Requirements**

- **Virtualization Layer: abstract view ←→ global view**
  - Driven by **Specification Abstraction** for particular task

- **NOS: global view ←→ physical switches**
  - API: driven by **Network State Abstraction**
  - Switch interface: driven by **Forwarding Abstraction**

Large Impact

- **Industry adoption**
- **Commoditization of switch hardware**
- **Independent innovation on each layer**
  - Evolution of programmable switches
  - Many controllers (Network OS)
  - Many applications
- **Network Virtualization, NFV, Google's and Microsoft's Wide Area Networks, SDX, …**
- **Great power to network administration!**

Thorny questions

- **Do users really know what they want?**
- **What would an interface be like?**
- **On the Internet:**
  - Do users trust/care/know about each other?
  - What is the incentive to *not* say your traffic is important?
  - Business models: users really like flat rates
- **Easier (but not easy)**
  - Datacenters, single company, home network, …

Hard to answer without doing,

hard to do as some mechanisms require consensus and changes to the network

Great power…

Can the users play too?

- Early OSs were single user, then came multiprogramming and time sharing
- Can we have the same for networks?

Participatory

An API for application control of SDNs

Andrew D. Ferguson, Arjun Guha, Chen Liang, Rodrigo Fonseca, and Shriram Krishnamurthi. Participatory Networking: An API for Application Control of SDNs. In Proc. ACM SIGCOMM 2013, August 2013.

1.

2. Ekiga

3.

4.

Motivation

1.

2. Ekiga

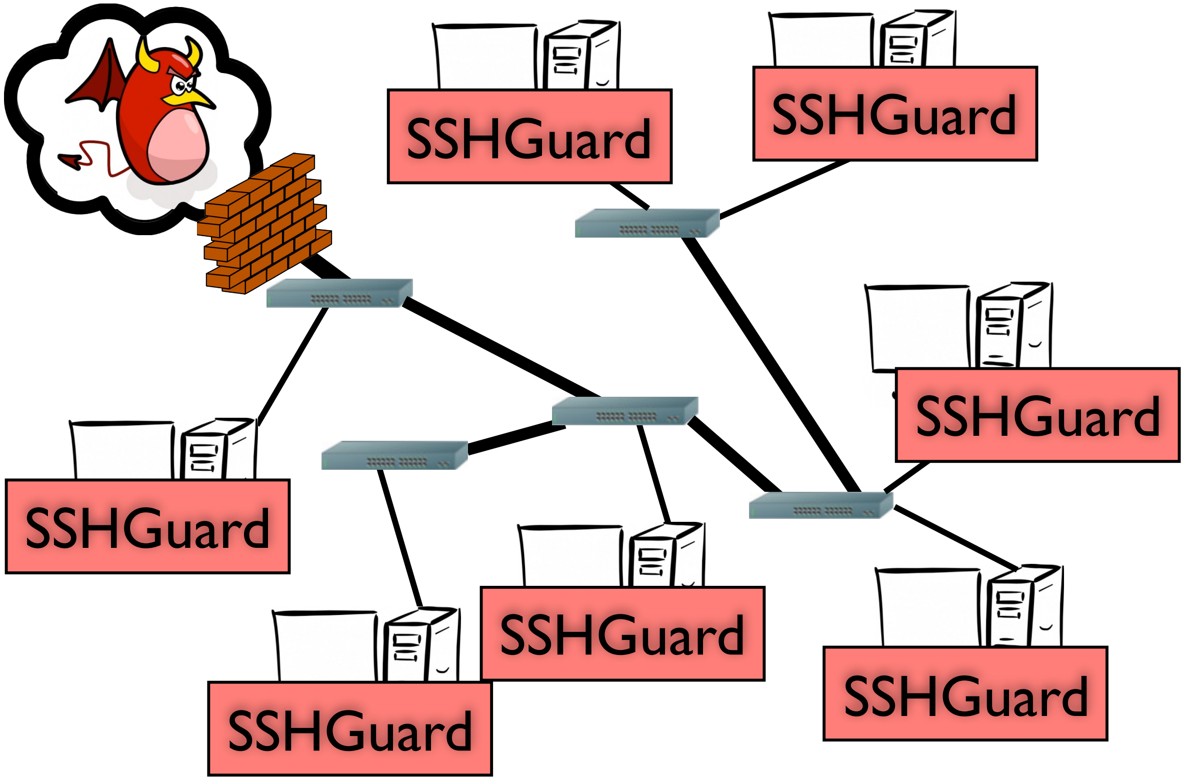
3.

4.

blocks hosts in response to login attempts

uses knowledge from host OS

prefers to deny traffic close to source



1.

2. Ekiga

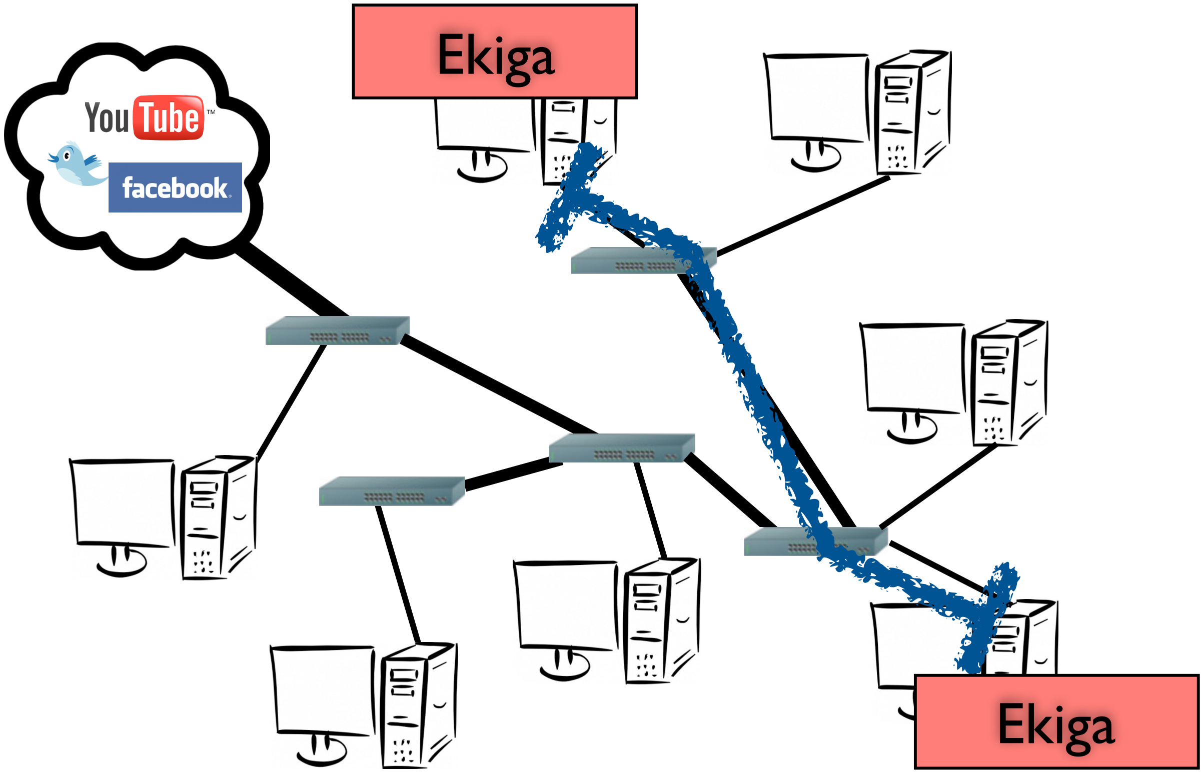
3.

4.

open source VOIP client

network needs dictated by end-user

prefers to reserve bandwidth



1.

2. Ekiga

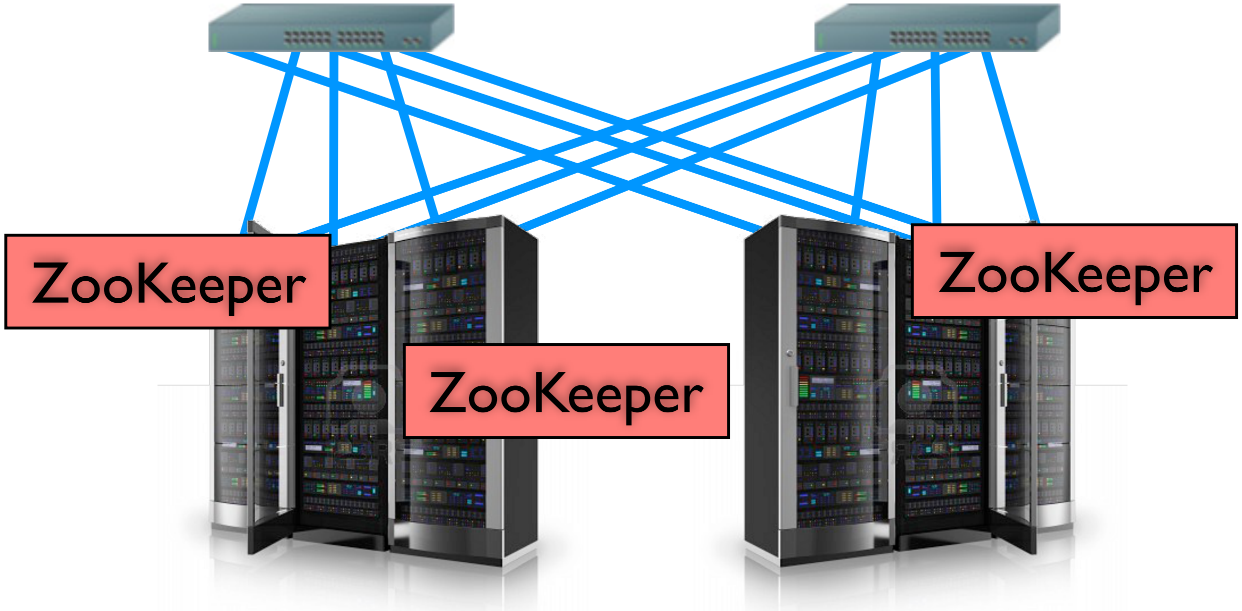
3.

4.

Paxos-like coordination service

network needs dictated by placement

prefers high-priority switch queues



1.

2. Ekiga

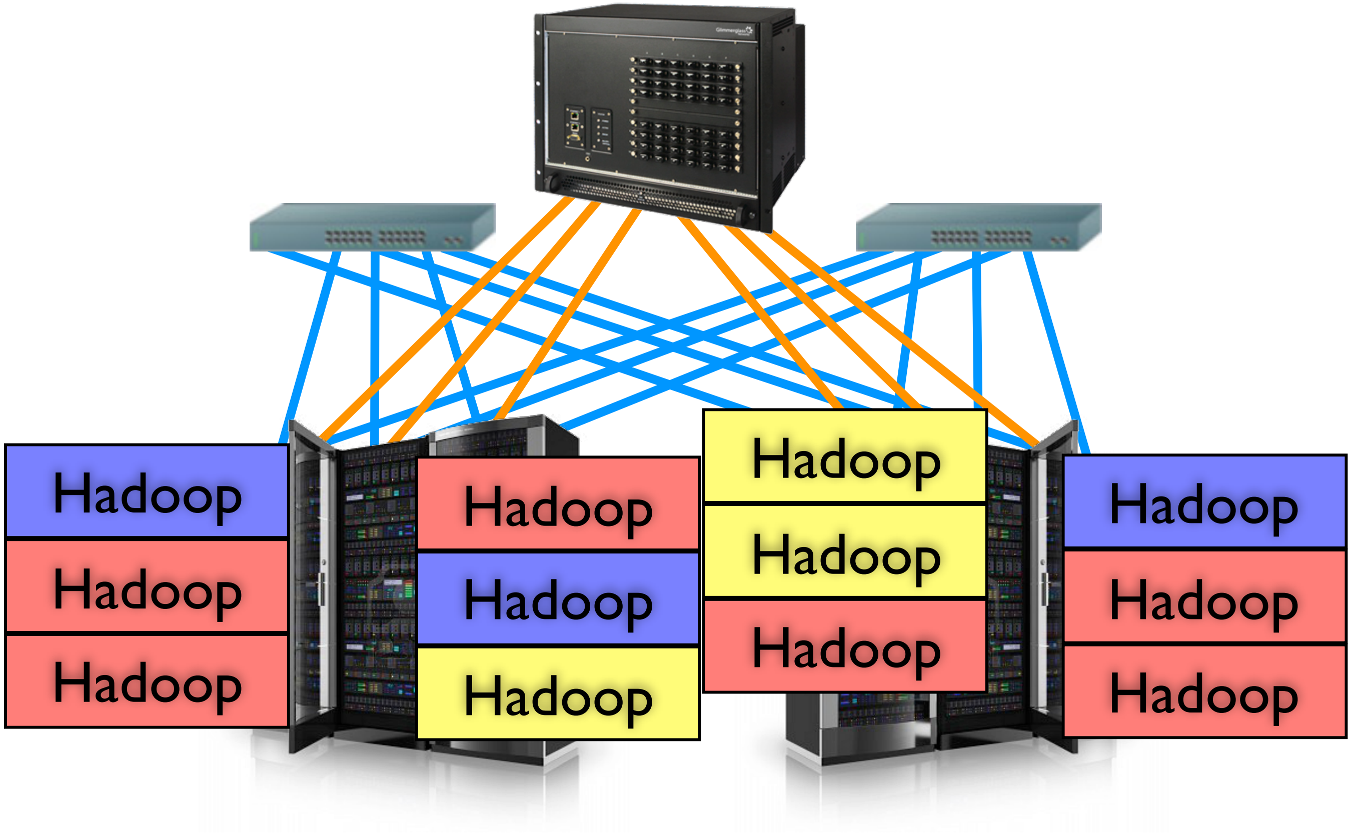
3.

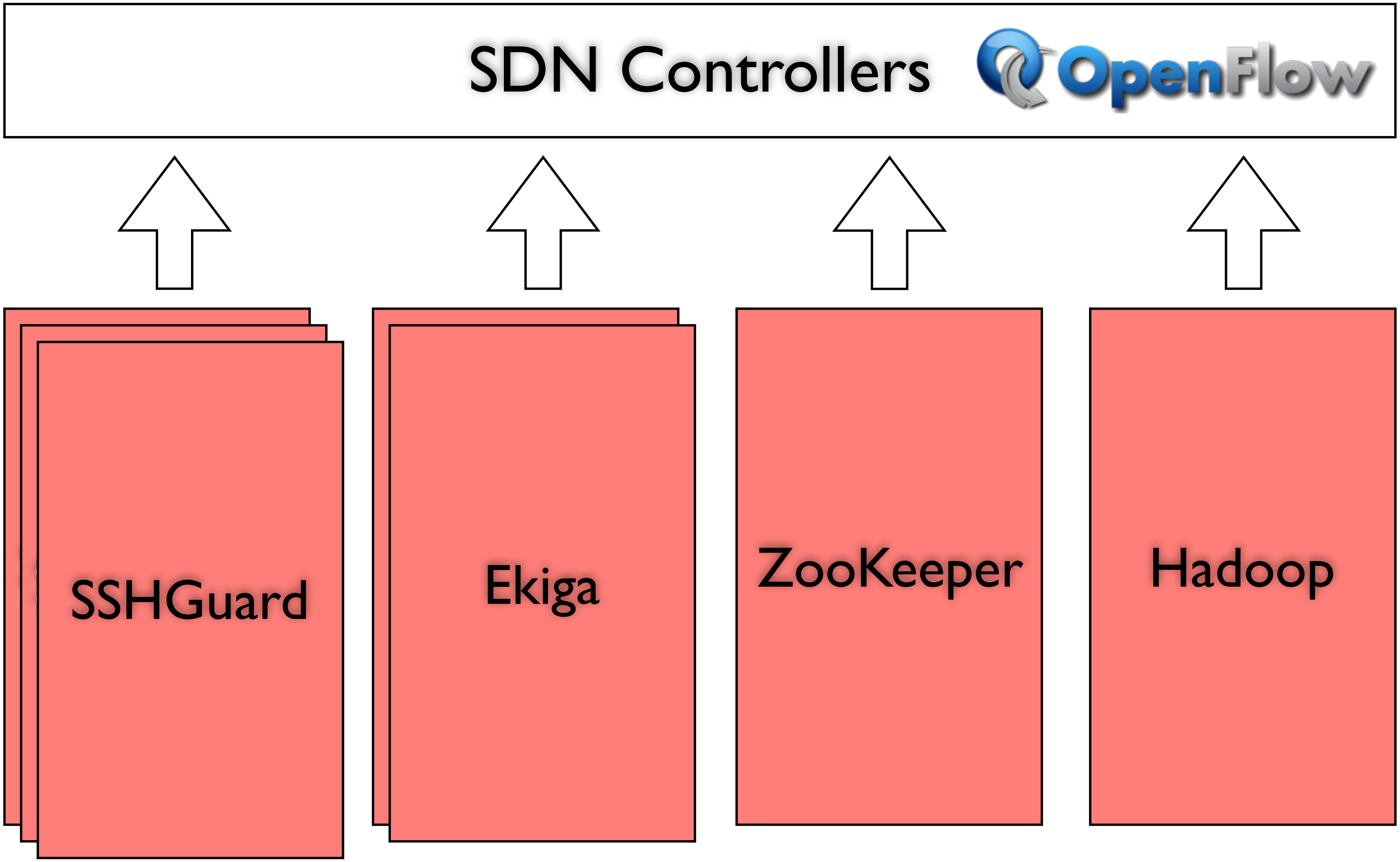
4.

open source data processing platform

network weights known by scheduler

prefers to reserve bandwidth





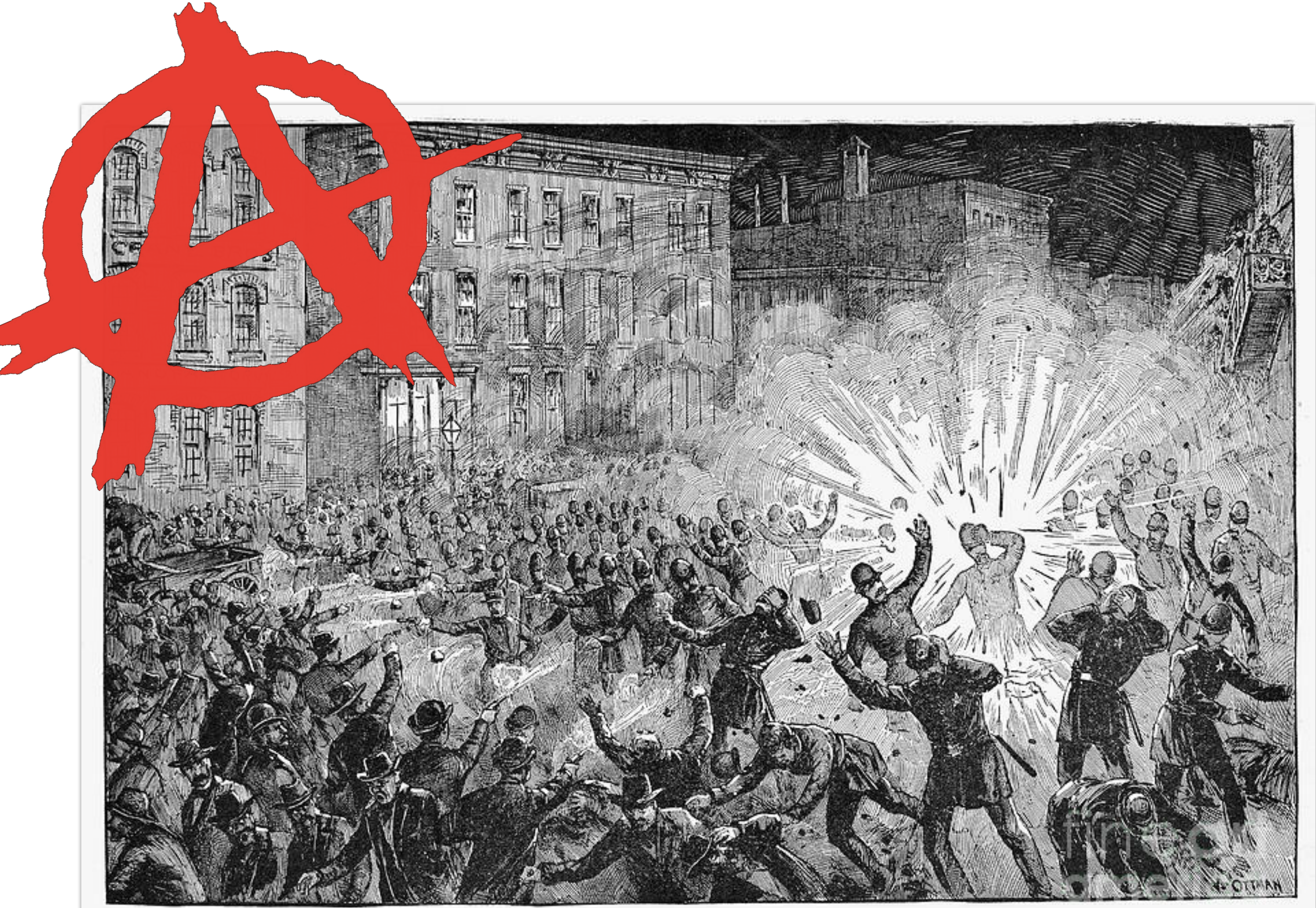
SDN Controllers OpenFlow

SSHGuard

Ekiga

ZooKeeper

Hadoop

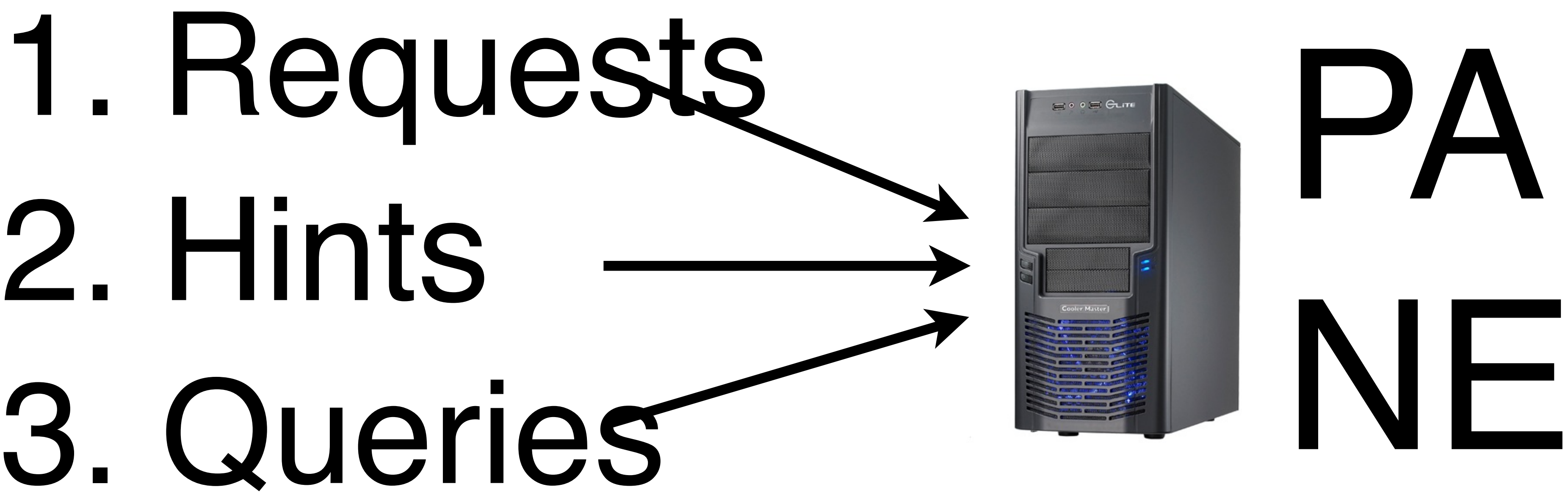


THE HAYMARKET RIOT. The Explosion and the Conflict.

1. decompose control and
2. resolve conflicts

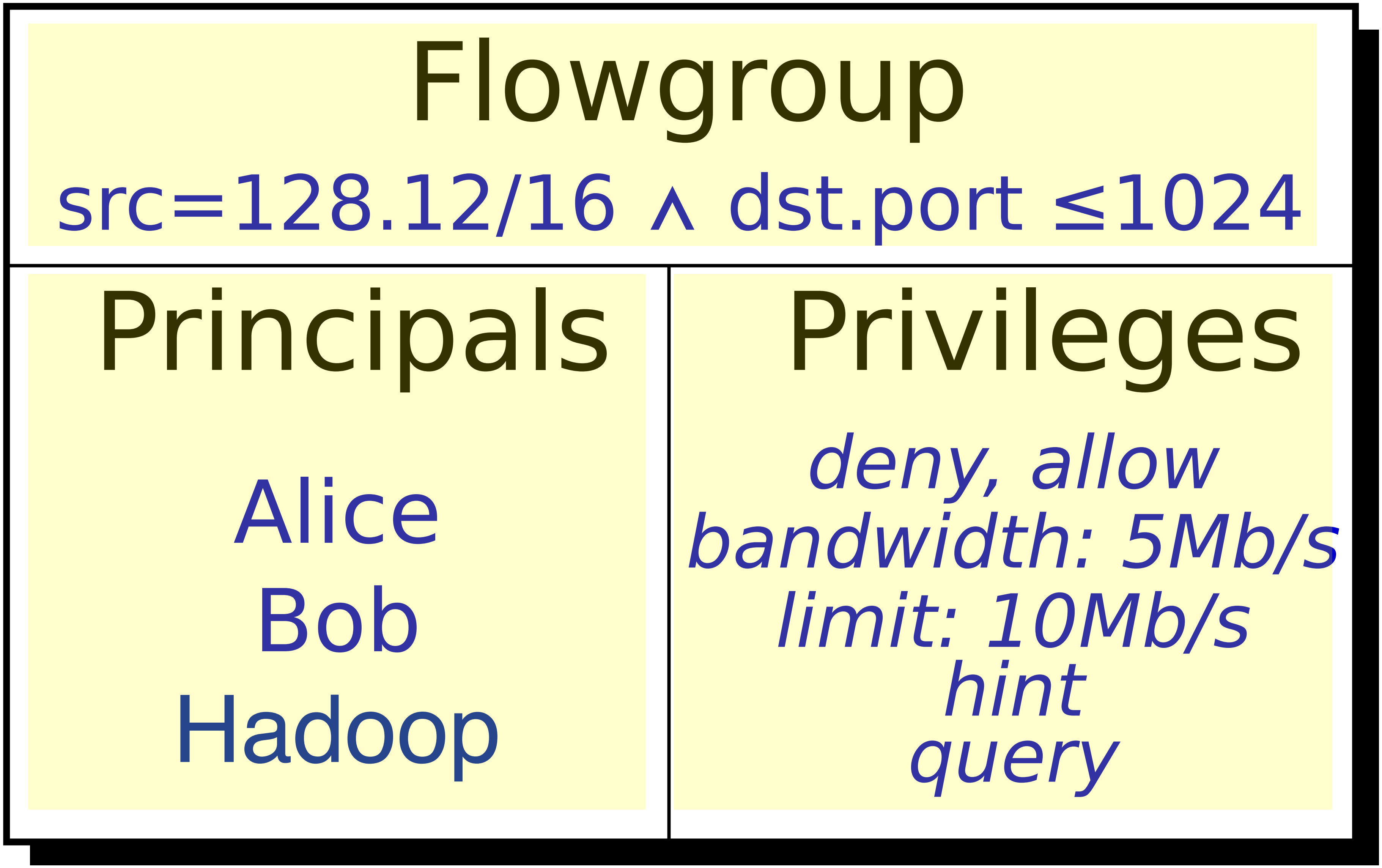
Challenges

Participatory Networking



1. Requests
2. Hints
3. Queries

PA
NE



Flowgroup

src=128.12/16 ∧ dst.port ≤1024

Principals

Alice

Bob

Hadoop

Privileges

deny, allow

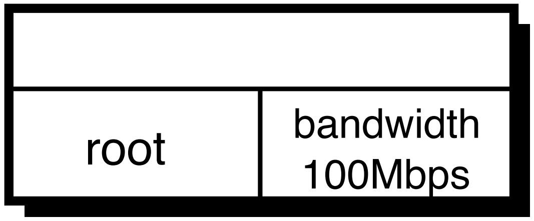
bandwidth: 5Mb/s

limit: 10Mb/s

hint

query

Shares

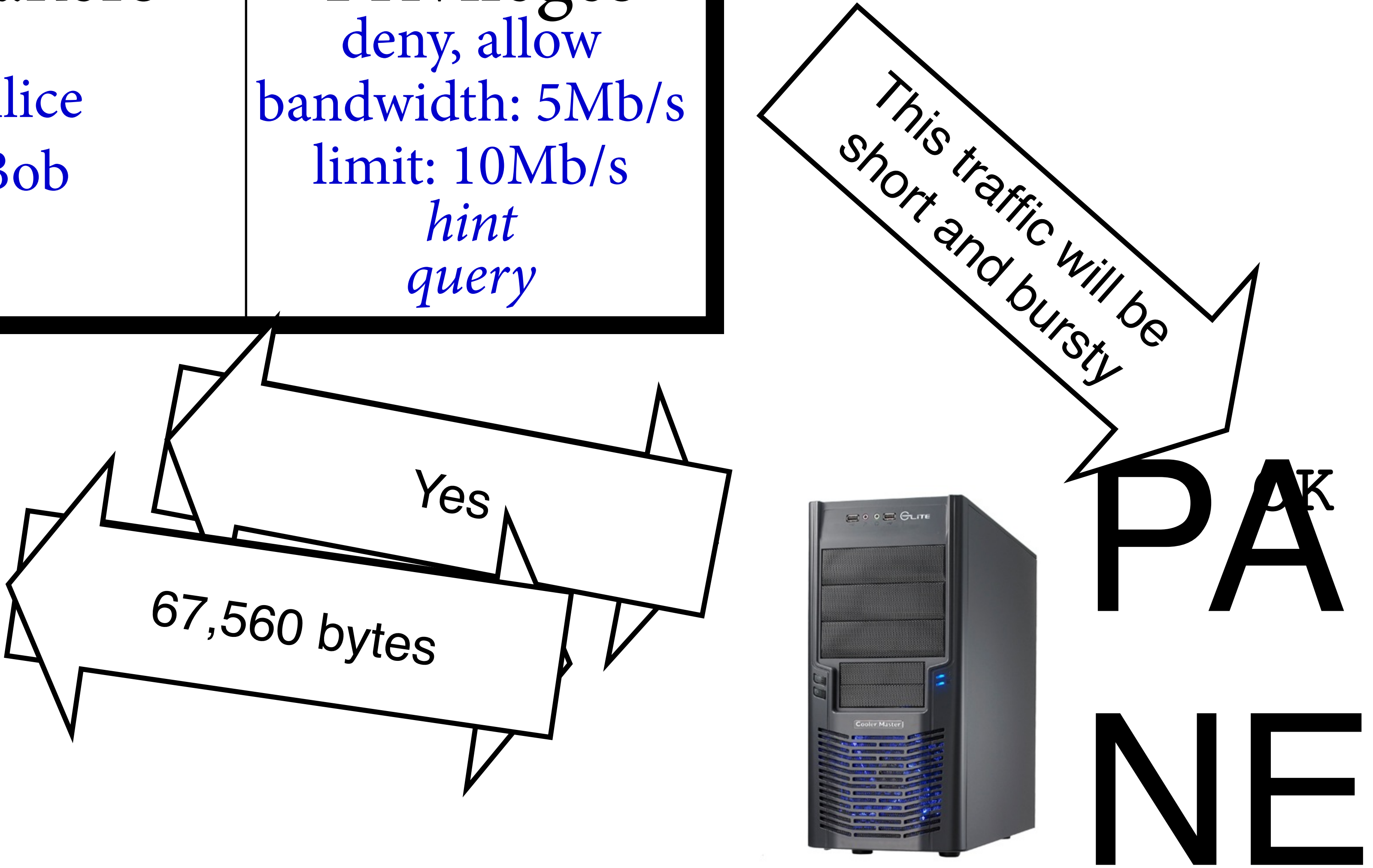


Share Tree

Flowgroup

src=128.12/16 ∧ dst.port ≤1024

Speakers	Privileges
Alice Bob	deny, allow bandwidth: 5Mb/s limit: 10Mb/s *hint* *query*



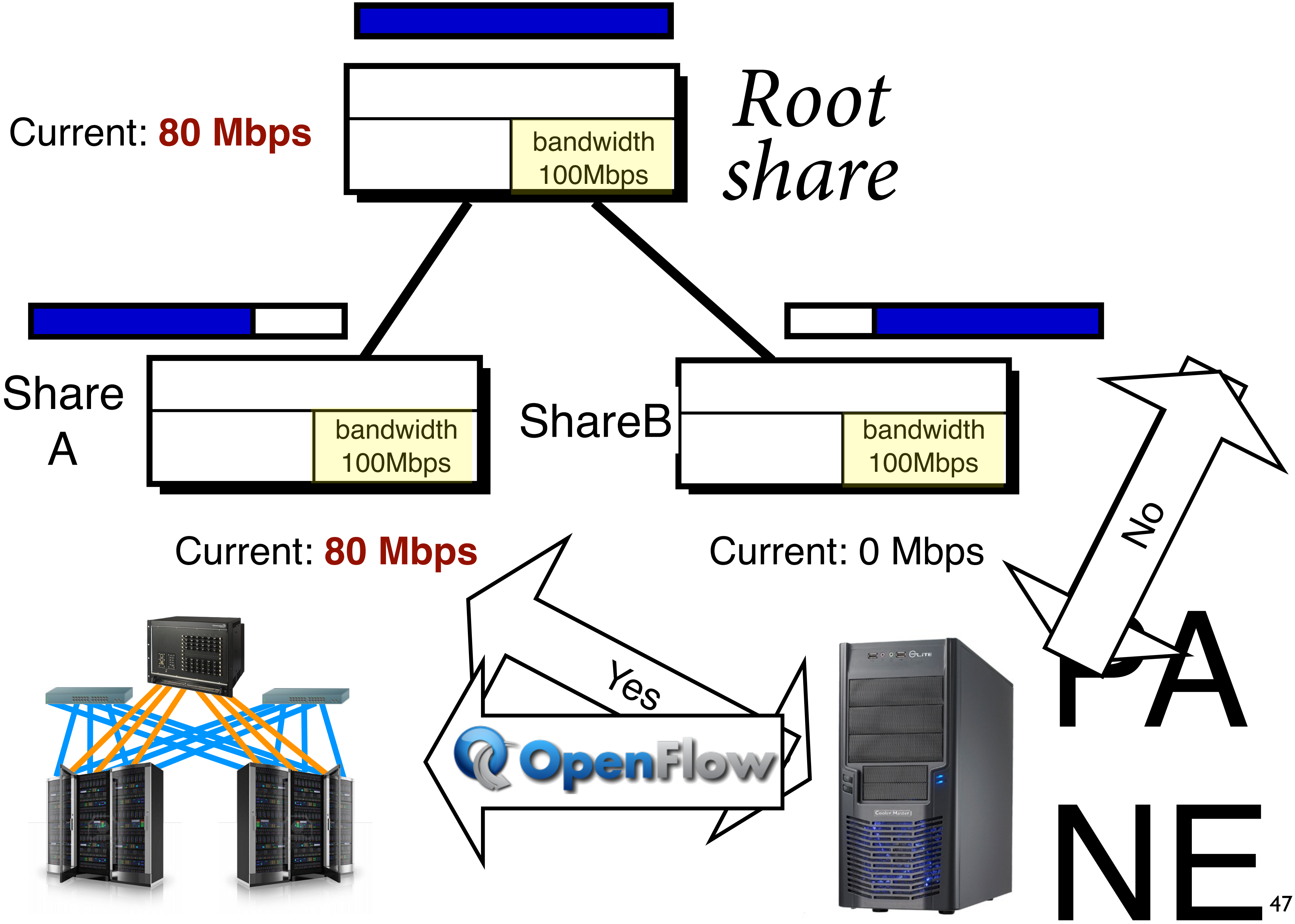
This traffic will be short and bursty

Yes

67,560 bytes

OK

PANE



Root share

Current: **80 Mbps**

bandwidth 100Mbps

Share A

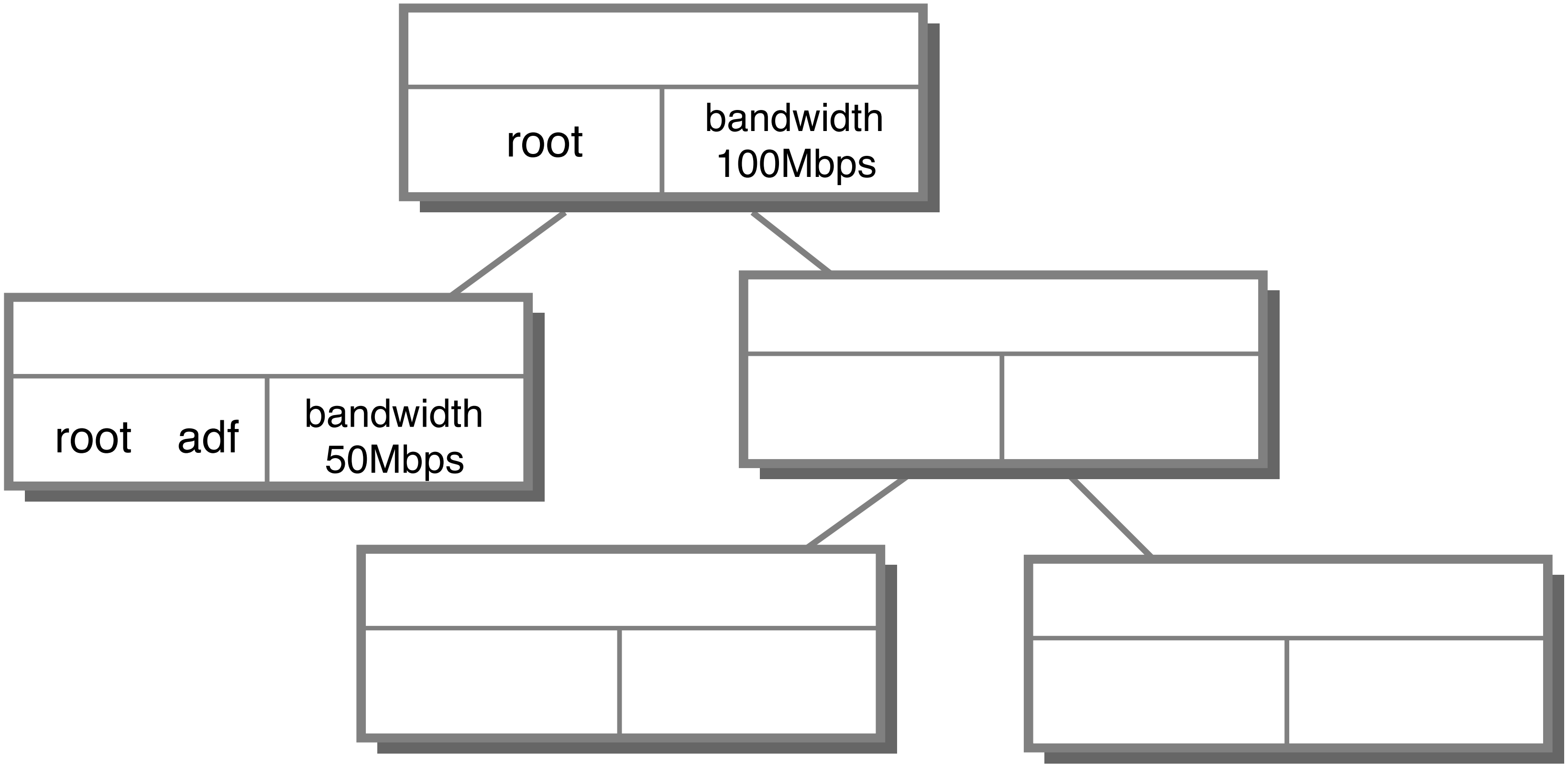
Current: **80 Mbps**

bandwidth 100Mbps

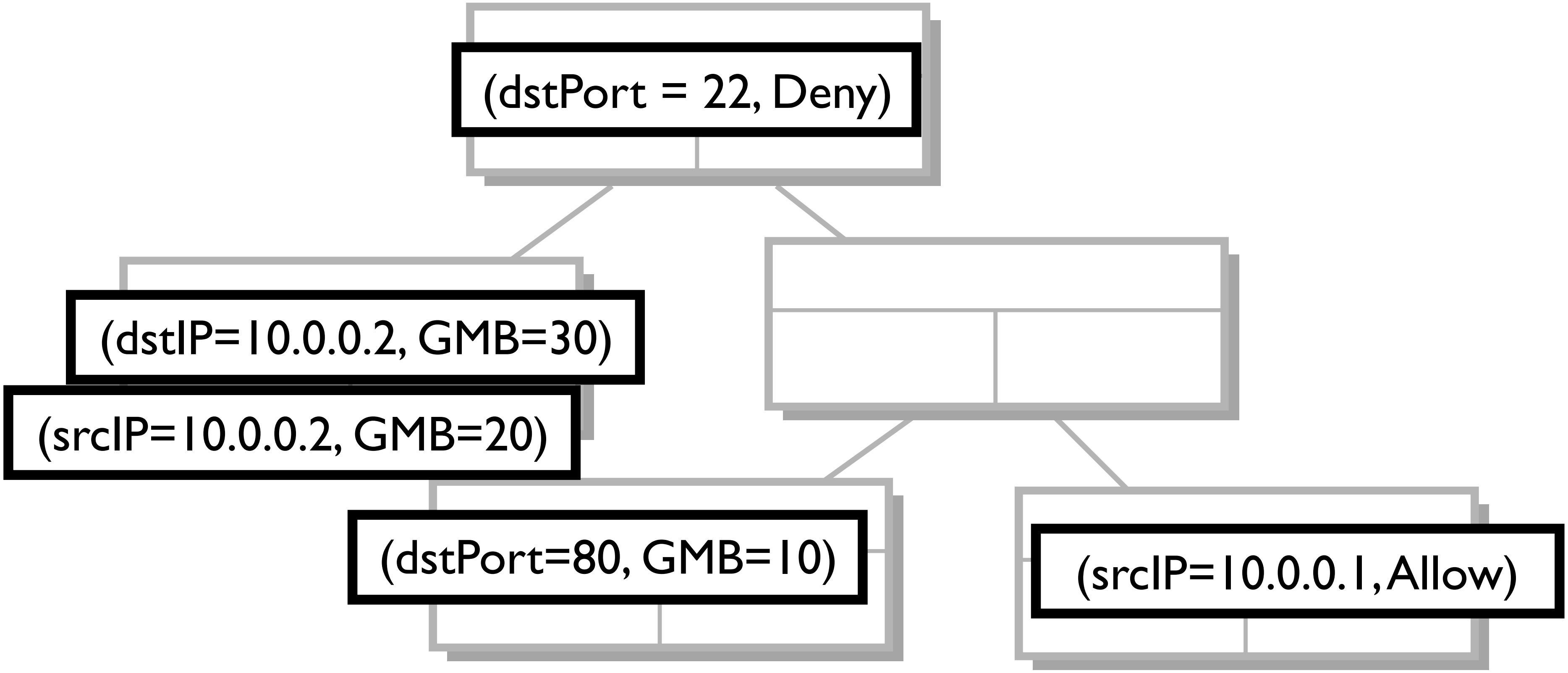
ShareB

Current: 0 Mbps

bandwidth 100Mbps



Share Tree


(dstPort = 22, Deny)

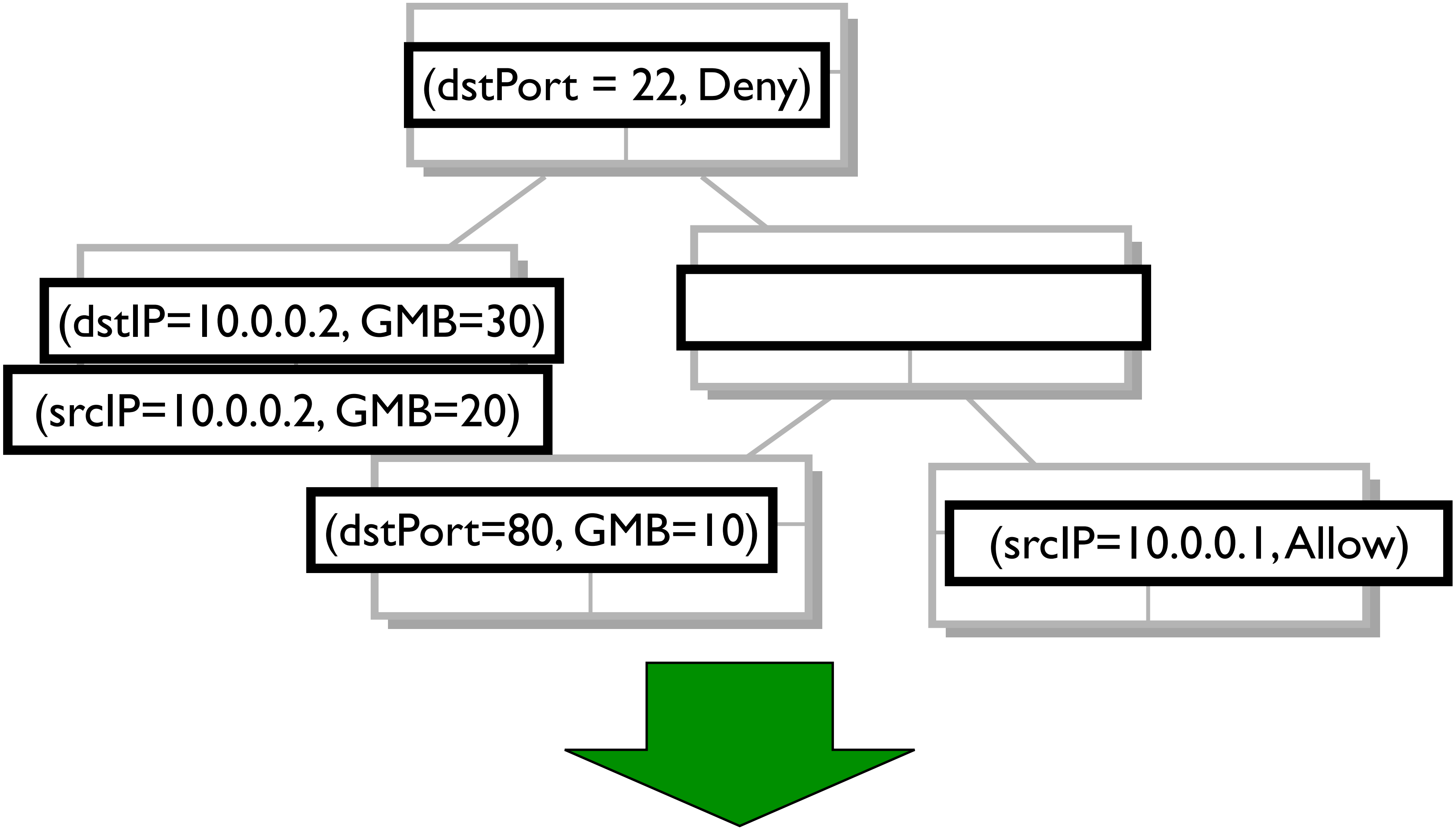
(dstIP=10.0.0.2, GMB=30)

(srcIP=10.0.0.2, GMB=20)

(dstPort=80, GMB=10)

(srcIP=10.0.0.1, Allow)

Policy Trees



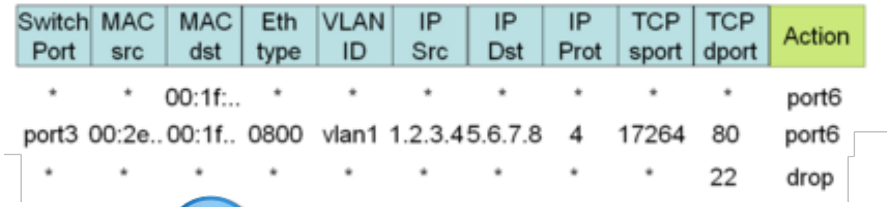
(dstPort = 22, Deny)

(dstIP=10.0.0.2, GMB=30)

(srcIP=10.0.0.2, GMB=20)

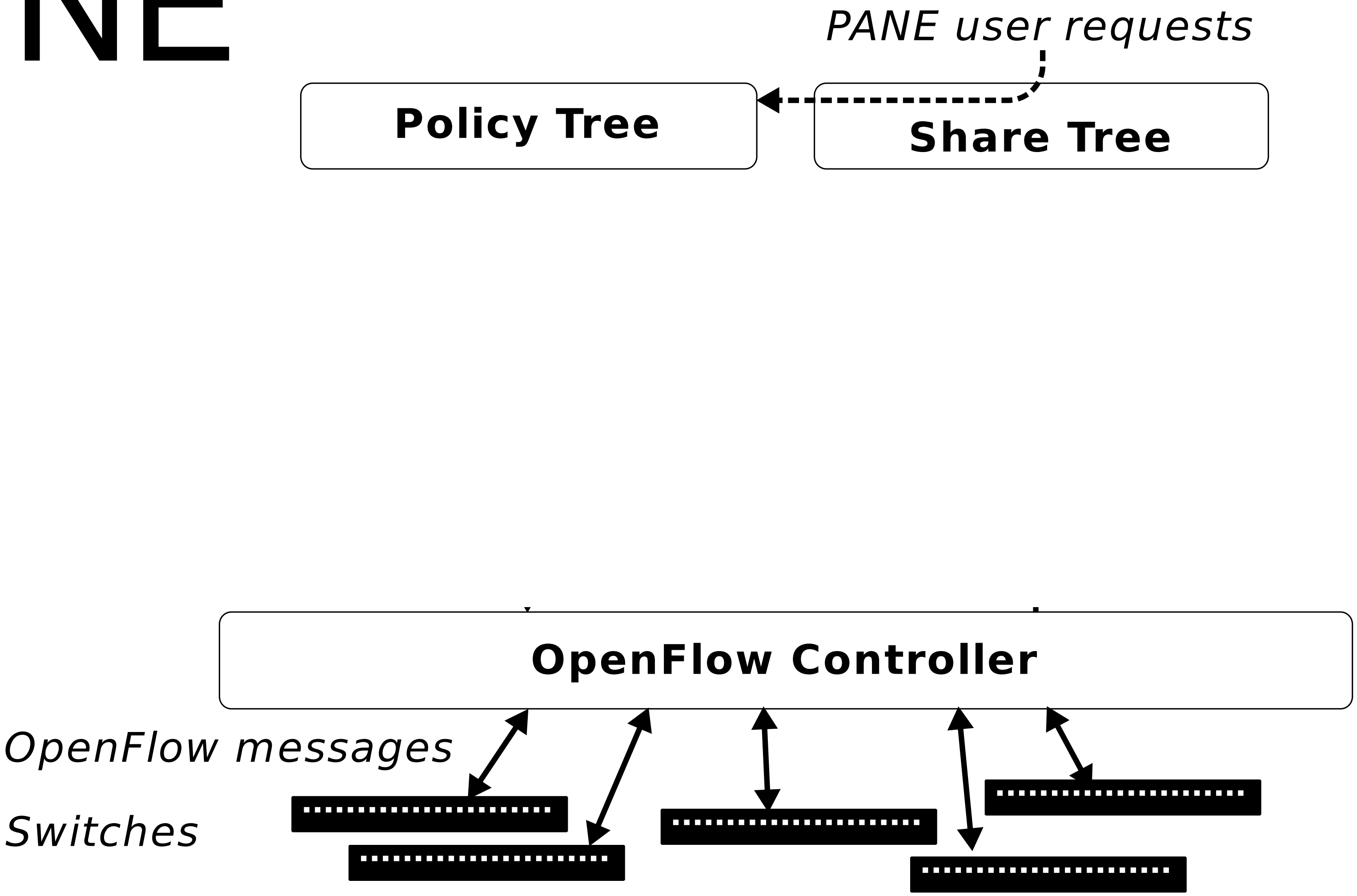
(dstPort=80, GMB=10)

(srcIP=10.0.0.1, Allow)

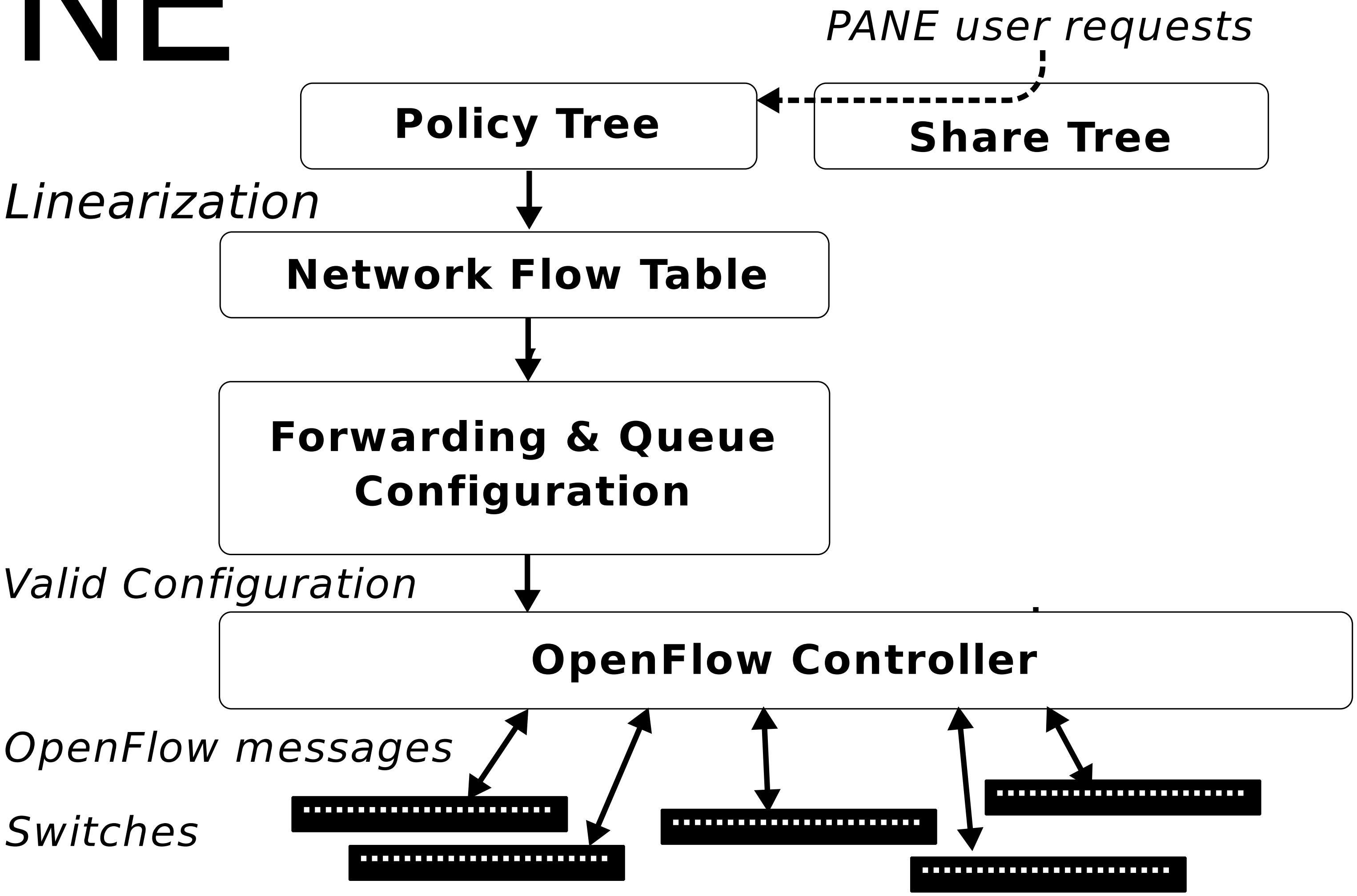


Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

PA NE


PANE user requests
Policy Tree
Share Tree
OpenFlow Controller
OpenFlow messages
Switches

PANE



PANE user requests

Policy Tree

Share Tree

Linearization

Network Flow Table

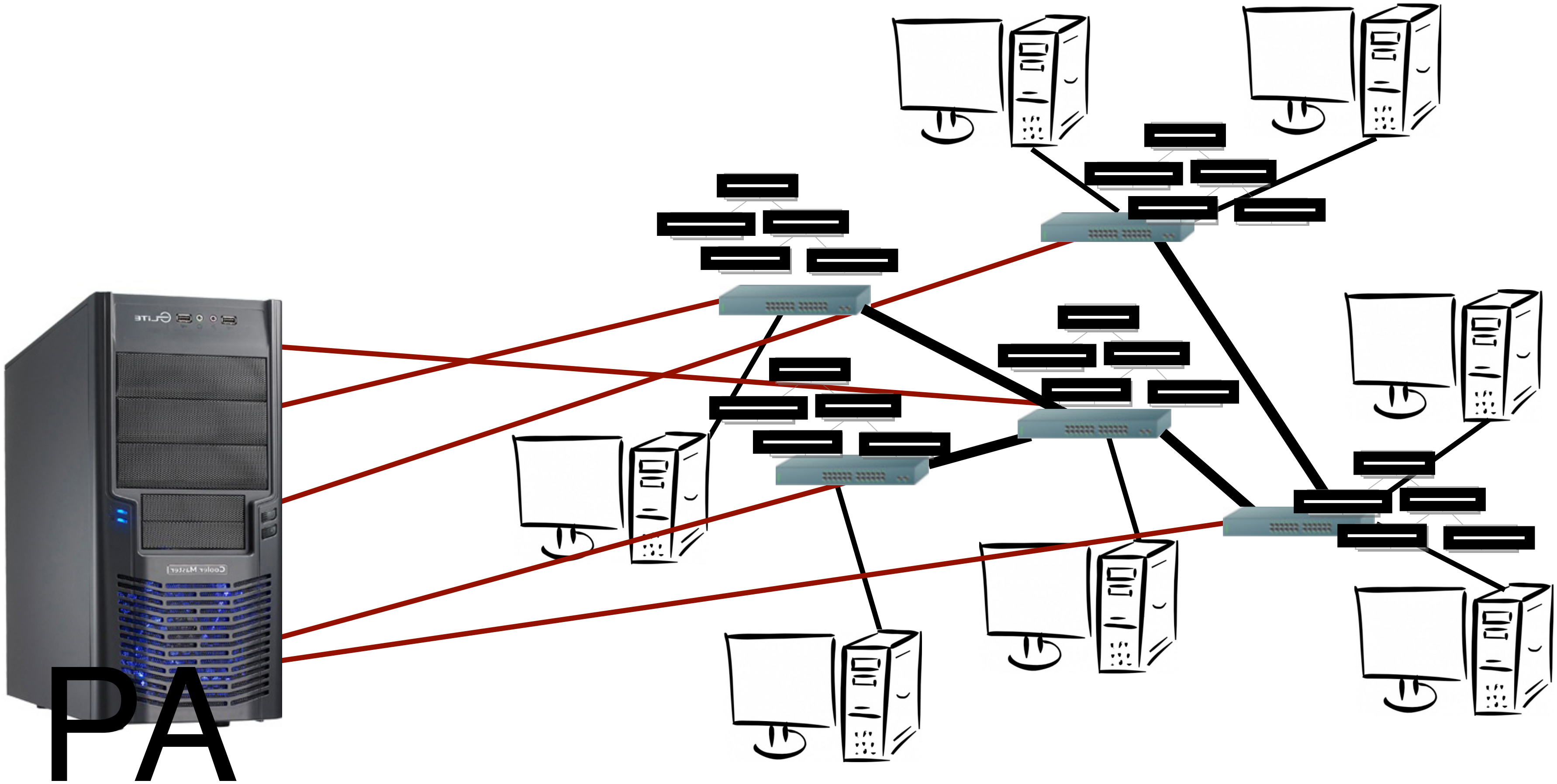
Forwarding & Queue Configuration

Valid Configuration

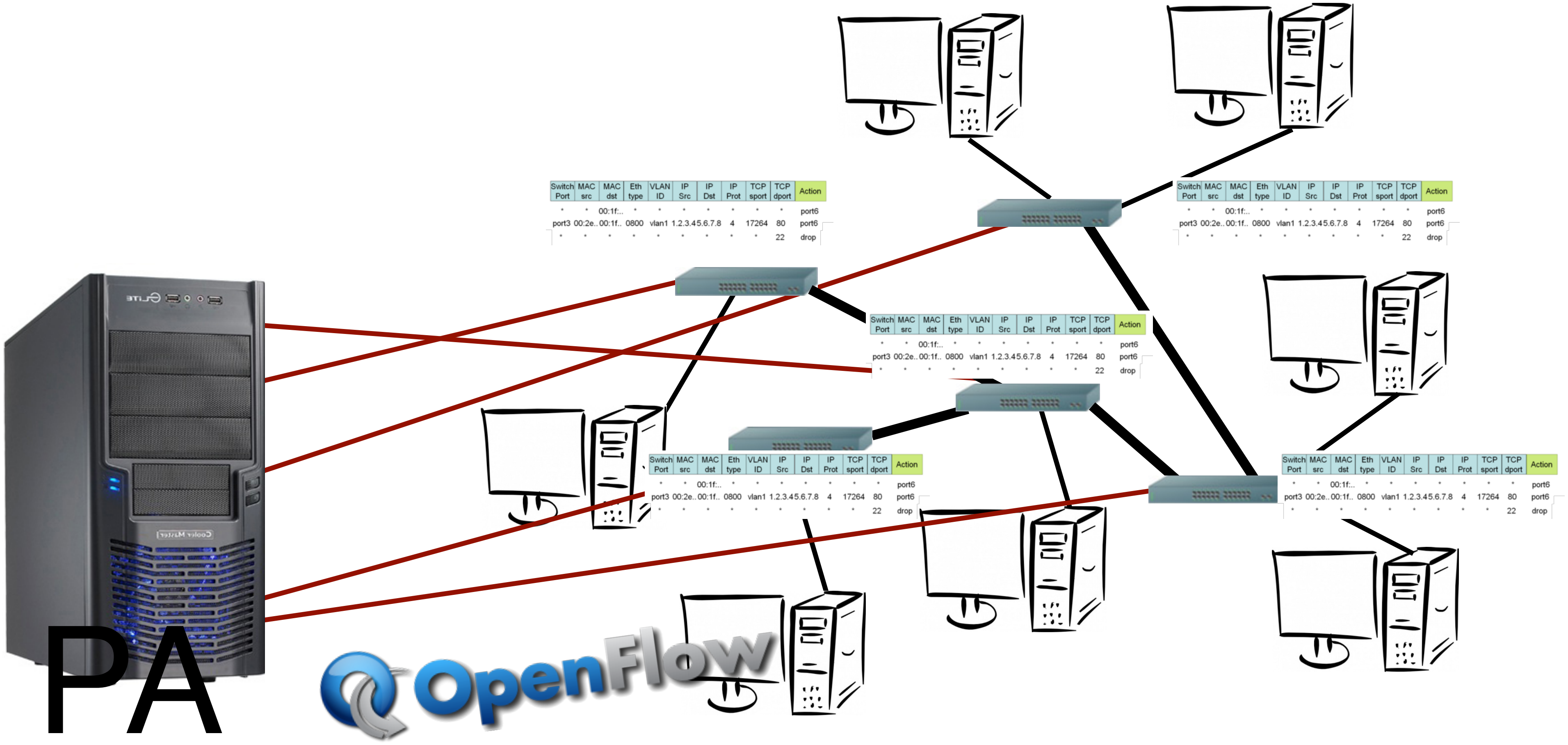
OpenFlow Controller

OpenFlow messages

Switches



PA
NE



Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

PA

NE

1.

2. Ekiga

3.

4.

access control

bandwidth

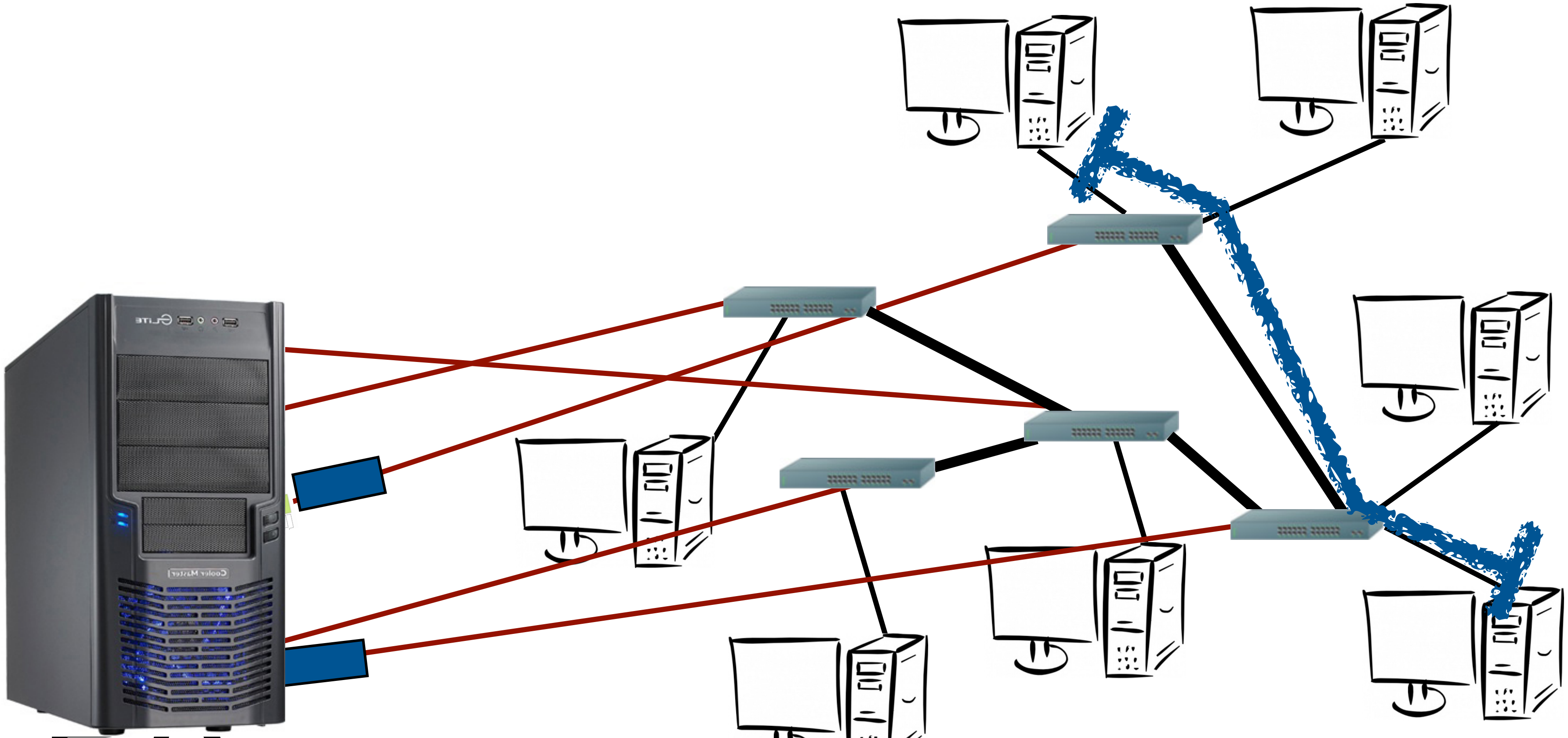
reservations

queues for low la

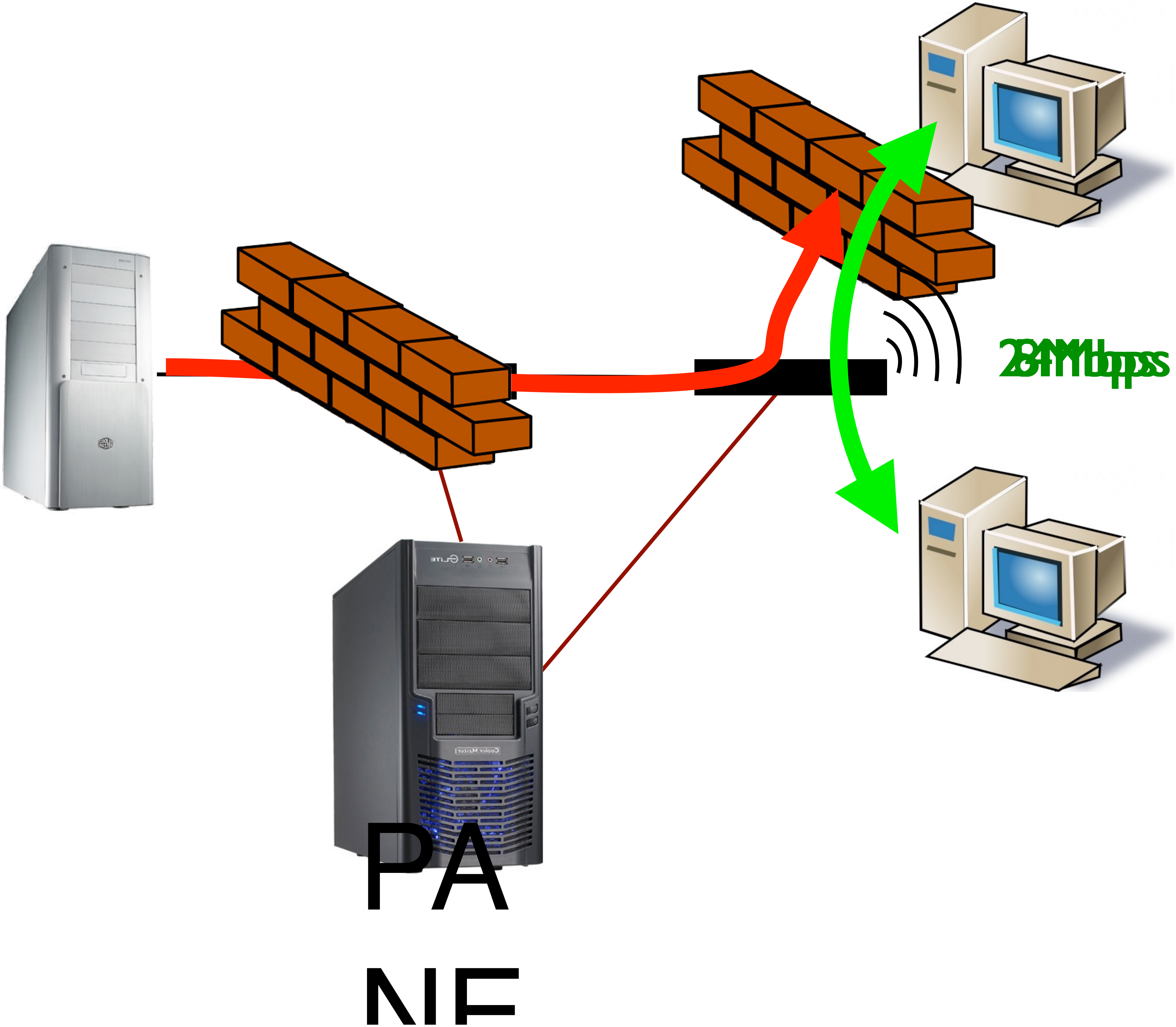
centralized traffic

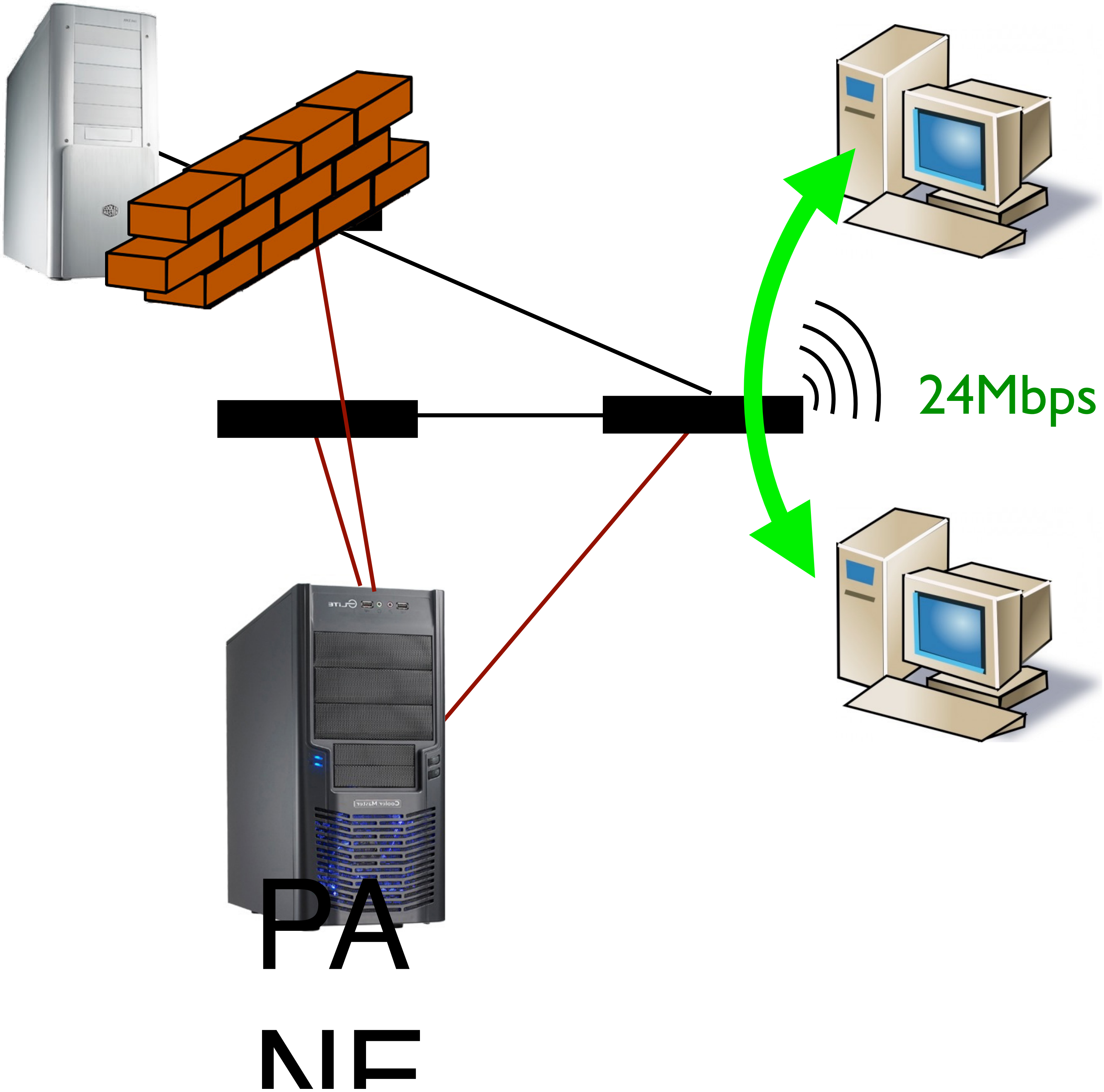
weights

Evaluation



NE

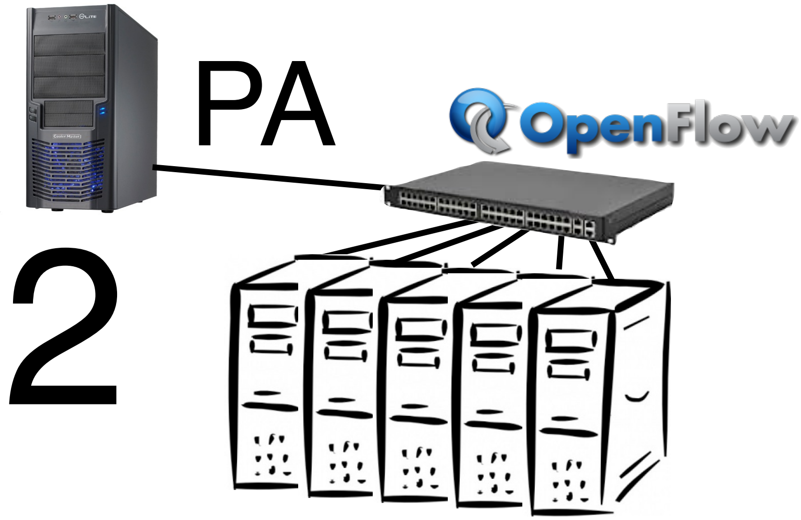

2Mbps
54Mbps
11Mbps
PA
NE

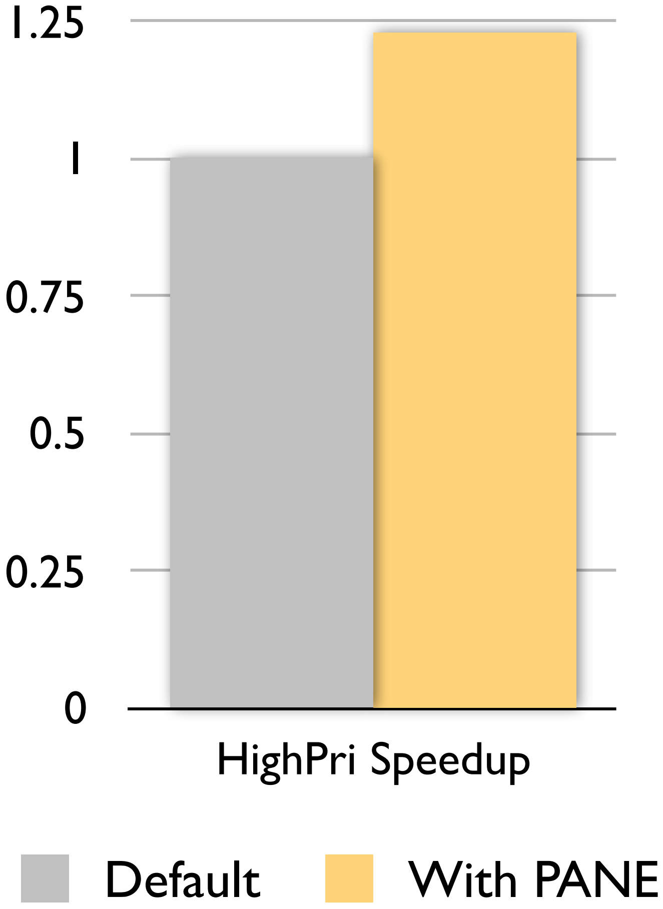

24Mbps
PA
NE

hadoop

Three equal-sized sort jobs:

- Two Low Priority with 25% weight
- One High Priority with 50% weight





Dynamically apply QoS to High Priority flows using PANE.

1. Allows applications to express what they want from the network (not only QoS)

2. Allows these applications to

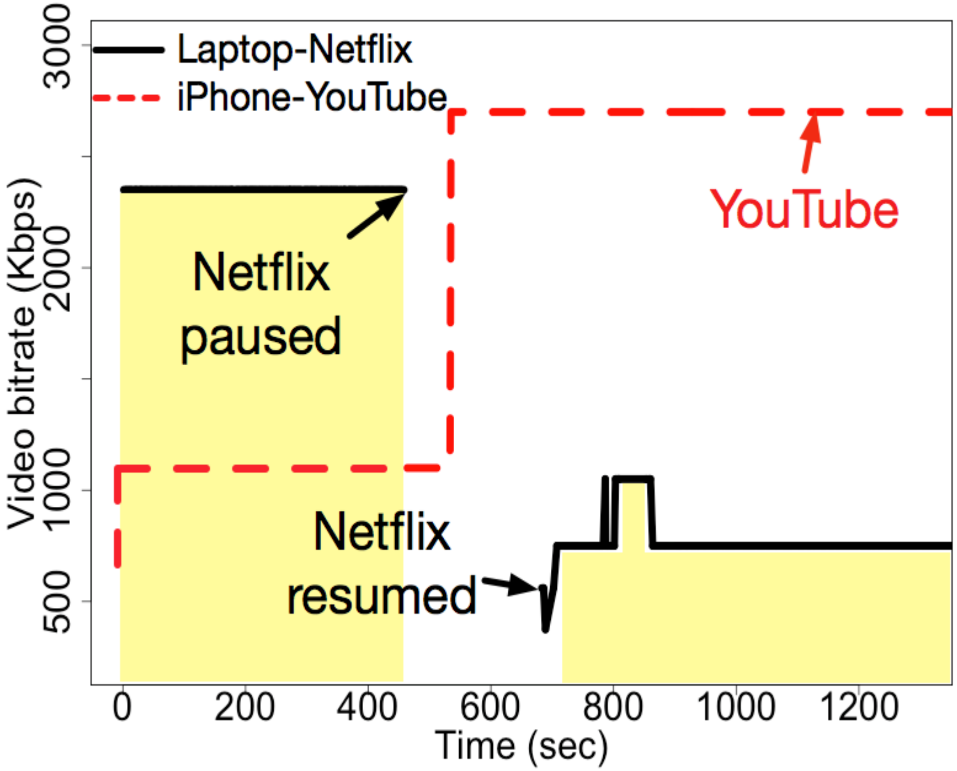
PANE

One last example

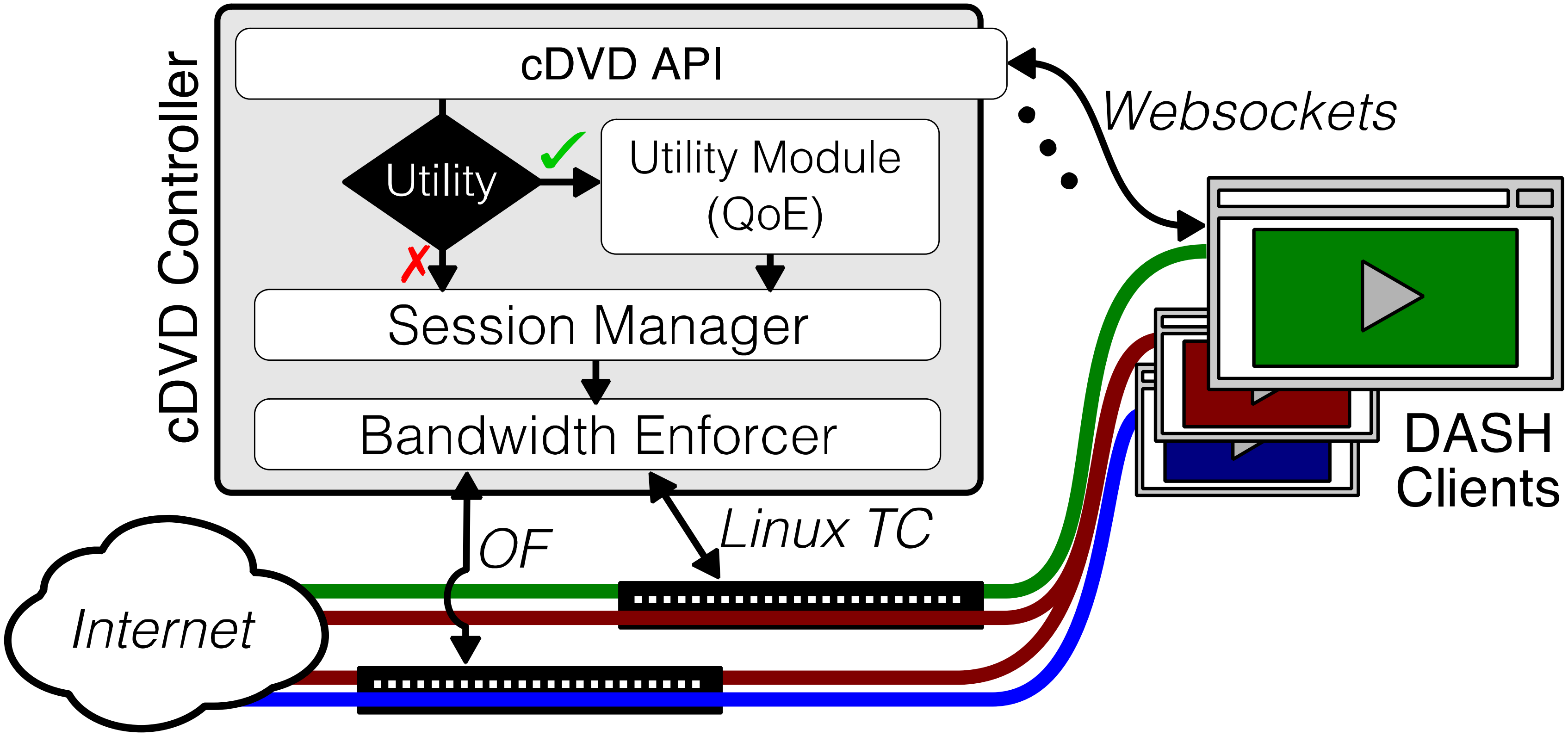
- Multiple video clients behind the same home router
- TCP equalizes the rate of flows
  - many flows per video, on-off behavior, and adaptive behavior
  - = unfairness

One last example

- Multiple video clients behind the same home router
- TCP equalizes the rate of flows
  - many flows per video, on-off behavior, and adaptive behavior
  - = unfairness

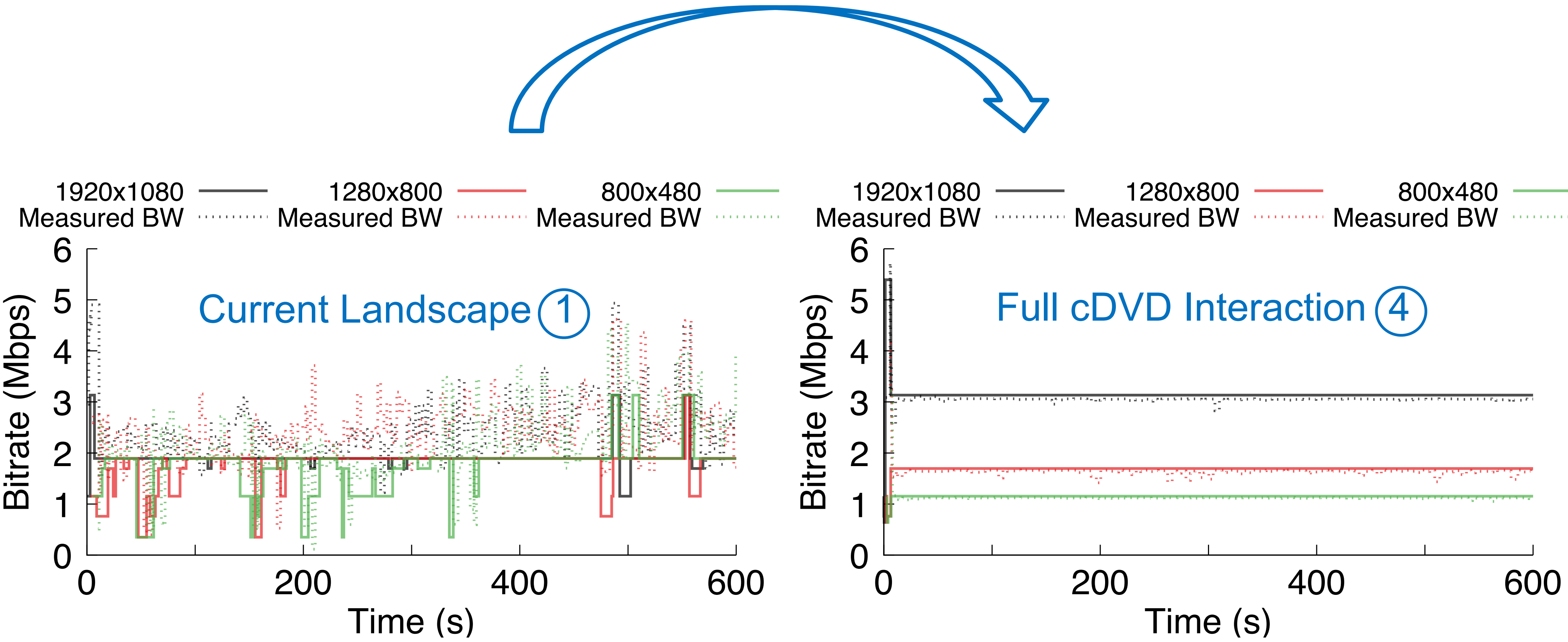


Modified Video Clients



Junyang Chen, Mostafa Ammar, Marwan Fayed, Rodrigo Fonseca. "Client-Driven Network-level QoE fairness for Encrypted ‘ DASH-S’”, InternetQoE 2016

Example Resulting Gains



- Measurement Details:
  - 6Mpbs bottleneck
  - modified `dash.js` client
  - BBC Testcard [4], with 13 video and 2 audio rates of encoding.

So, how do we talk to the network?

- **SDN gives us another way to change the network API**
  - Out-of-band, though flexible and fast control plane
  - Can address and configure many mechanisms
- **Contrast with in-band mechanisms**
  - Packet/flow tags, socket options
  - Increasingly programmable data path
- **A lot of research in mechanisms, still plenty to do in policies**

Questions?