Programmable Networks

Jennifer Rexford

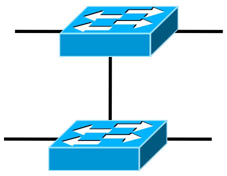
COS 461: Computer Networks

http://www.cs.princeton.edu/courses/archive/spr20/cos461/

The Internet: A Remarkable Story

- Tremendous success
  - From research experiment to global infrastructure
- Brilliance of under-specifying
  - Network: best-effort packet delivery
  - Hosts: arbitrary applications
- Enables innovation in applications
  - Web, P2P, VoIP, social networks, smart cars, …
- But, change is easy only at the edge… ☹

Inside the ‘Net: A Different Story…

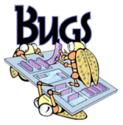


- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features

Impacts performance, security, reliability, cost…

Networks are Hard to Manage

- Operating a network is expensive
  - More than half the cost of a network
  - Yet, operator error causes most outages



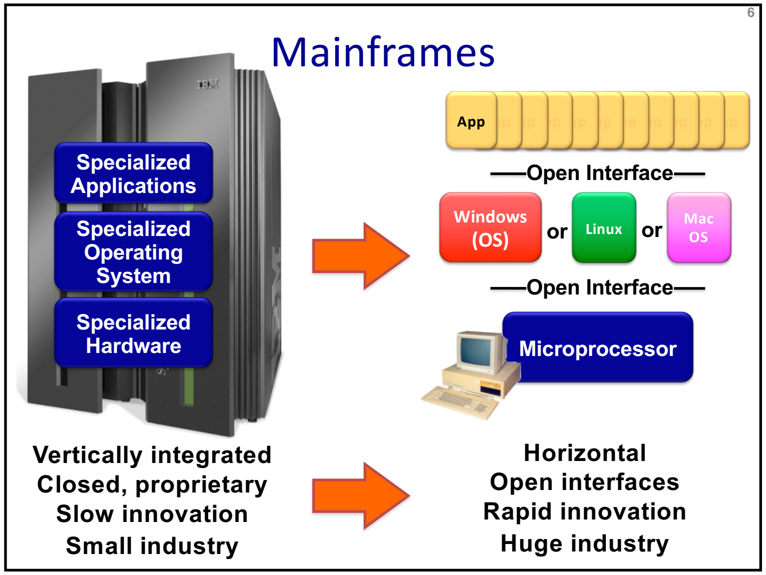
- Buggy software in the equipment
  - Routers with 20+ million lines of code
  - Cascading failures, vulnerabilities, etc.

- The network is “in the way”
  - Especially in data centers and the home

5

A Helpful Analogy

From Nick McKeown's talk "Making SDN Work" at the Open Networking Summit, April 2012



6

Mainframes

Specialized Applications

Specialized Operating System

Specialized Hardware

App

Open Interface

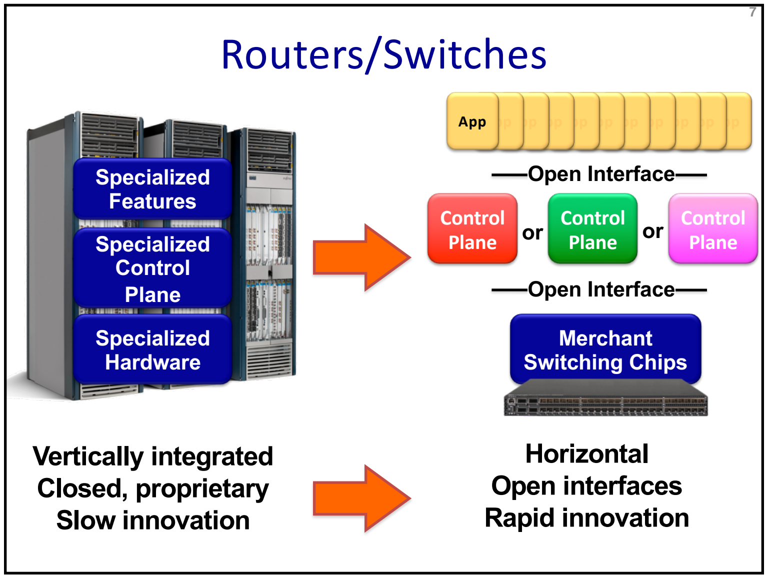
Windows (OS) or Linux or Mac OS

Open Interface

Microprocessor

Vertically integrated
Closed, proprietary
Slow innovation
Small industry

Horizontal
Open interfaces
Rapid innovation
Huge industry



7

Routers/Switches

Specialized Features

Specialized Control Plane

Specialized Hardware

App

Open Interface

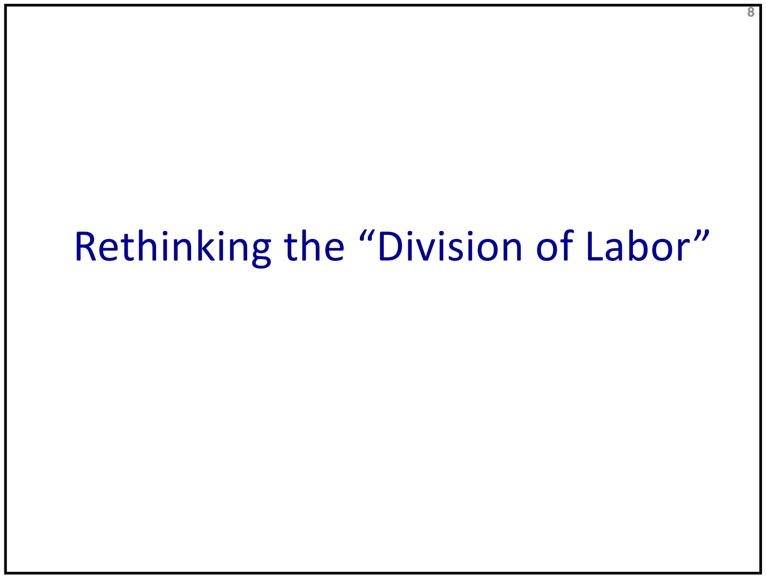
Control Plane or Control Plane or Control Plane

Open Interface

Merchant Switching Chips

Vertically integrated
Closed, proprietary
Slow innovation

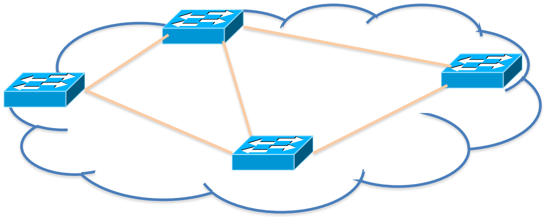
Horizontal
Open interfaces
Rapid innovation



8

Rethinking the "Division of Labor"

Traditional Computer Networks

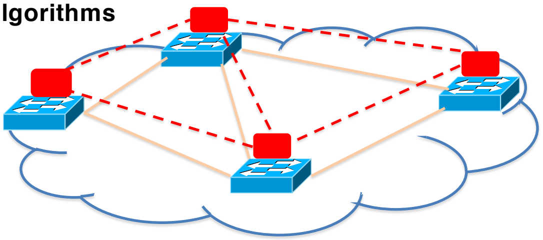


Data plane:
Packet streaming

Forward, filter, buffer, mark, rate-limit, and measure packets

Traditional Computer Networks

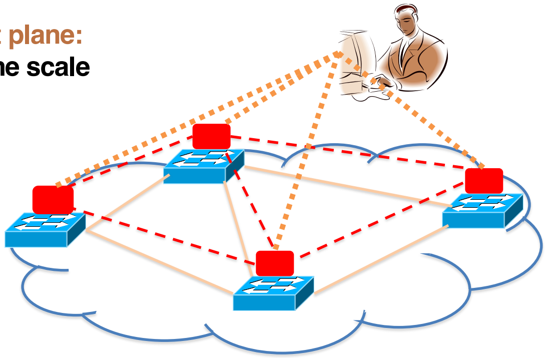
Control plane:
Distributed algorithms



Track topology changes, compute routes, install forwarding rules

Traditional Computer Networks

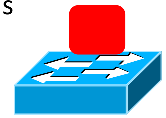
Management plane:
Human time scale



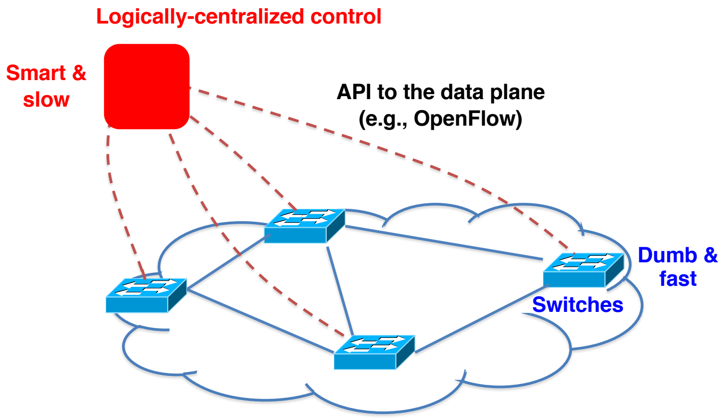
Collect measurements and configure the equipment

Death to the Control Plane!

- Simpler management
  - No need to “invert” control-plane operations
- Faster pace of innovation
  - Less dependence on vendors and standards
- Easier interoperability
  - Compatibility only in “wire” protocols
- Simpler, cheaper equipment
  - Minimal software



Software Defined Networking (SDN)


Logically-centralized control
Smart & slow
API to the data plane (e.g., OpenFlow)
Dumb & fast
Switches

OpenFlow Networks

http://ccr.sigcomm.org/online/files/p69-v38n2n-mckeown.pdf

Data-Plane: Simple Packet Handling

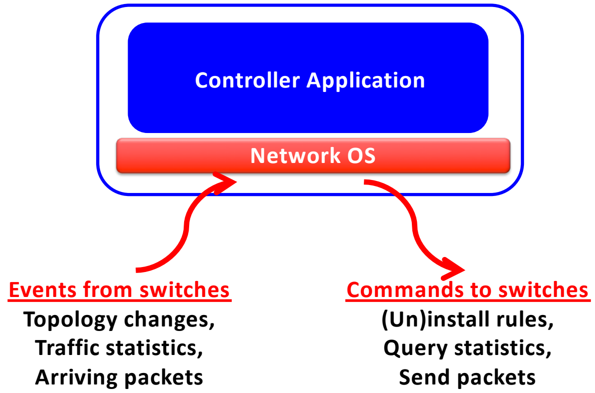
- Simple packet-handling rules
  - Pattern: match packet header bits
  - Actions: drop, forward, modify, send to controller
  - Priority: disambiguate overlapping patterns
  - Counters: #bytes and #packets

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

Unifies Different Kinds of Boxes

- Router
  - Match: longest destination IP prefix
  - Action: forward out a link
- Switch
  - Match: dest MAC address
  - Action: forward or flood
- Firewall
  - Match: IP addresses and TCP /UDP port numbers
  - Action: permit or deny
- NAT
  - Match: IP address and port
  - Action: rewrite addr and port

Controller: Programmability



Events from switches
Topology changes,
Traffic statistics,
Arriving packets

Commands to switches
(Un)install rules,
Query statistics,
Send packets

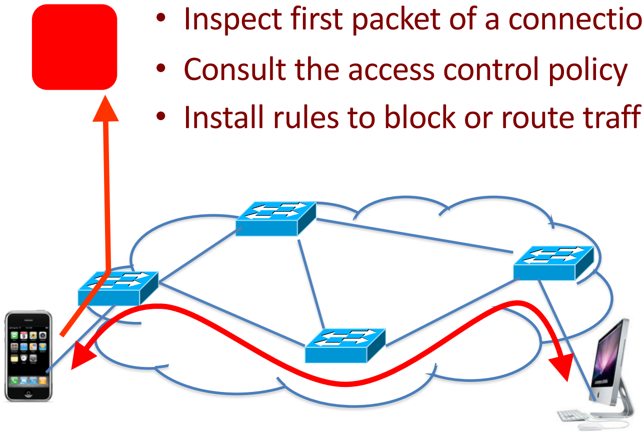
OpenFlow questions

- OpenFlow designed for
  - (A) Inter-domain management (between)
  - (B) Intra-domain management (within)
- OpenFlow API to switches open up the
  - (A) RIB (B) FIB
- OpenFlow FIB match based on
  - (A) Exact match (e.g., MAC addresses)
  - (B) Longest prefix (e.g., IP addresses)
  - (C) It's complicated

Example OpenFlow Applications

- **Dynamic access control**
- **Seamless mobility/migration**
- **Server load balancing**
- **Network virtualization**
- Using multiple wireless access points
- Energy-efficient networking
- Adaptive traffic monitoring
- Denial-of-Service attack detection

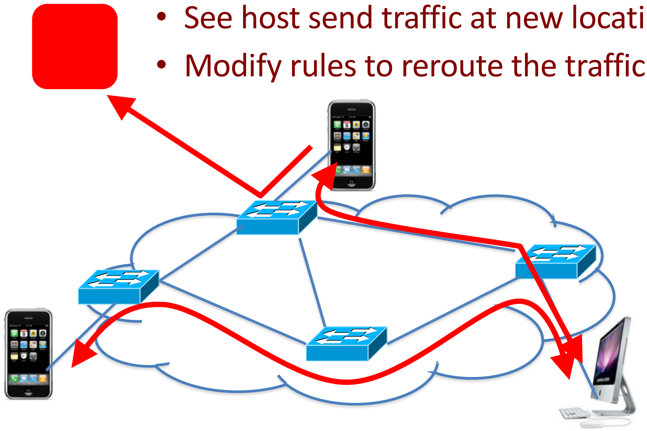
E.g.: Dynamic Access Control



- Inspect first packet of a connection
- Consult the access control policy
- Install rules to block or route traffic

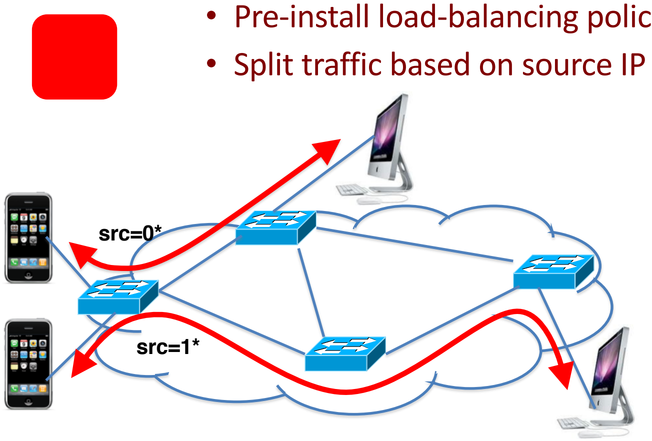
E.g.: Seamless Mobility/Migration

- See host send traffic at new location
- Modify rules to reroute the traffic

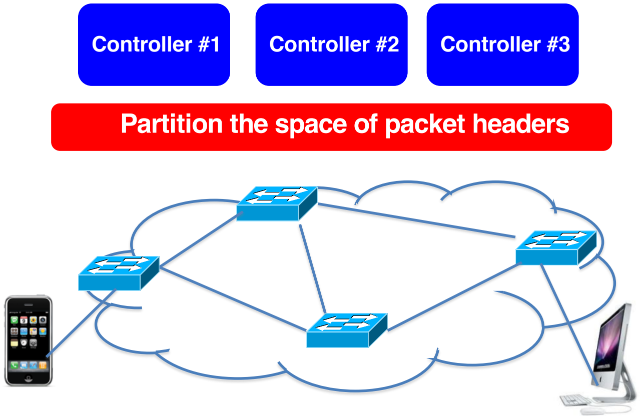


E.g.: Server Load Balancing

- Pre-install load-balancing policy
- Split traffic based on source IP



E.g.: Network Virtualization



Controller #1

Controller #2

Controller #3

Partition the space of packet headers

Controller and the FIB

- Forwarding rules should be added

(A) Proactively

(B) Reactively (e.g., with controller getting first packet)

(C) Depends on application

OpenFlow in the Wild

- Open Networking Foundation
  - Google, Facebook, Microsoft, Yahoo, Verizon, Deutsche Telekom, and many other companies
- Commercial OpenFlow switches
  - Intel, HP, NEC, Quanta, Dell, IBM, Juniper, …
- Network operating systems
  - NOX, Beacon, Floodlight, Nettle, ONIX, POX, Frenetic
- Network deployments
  - Data centers
  - Cloud provider backbones
  - Public backbones

Programmable Data Planes

https://www.sigcomm.org/sites/default/files/ccr/papers/2014/July/0000000-0000004.pdf

In the Beginning…

- OpenFlow was simple
- A single rule table
  - Priority, pattern, actions, counters, timeouts
- Matching on any of 12 fields, e.g.,
  - MAC addresses
  - IP addresses
  - Transport protocol
  - Transport port numbers

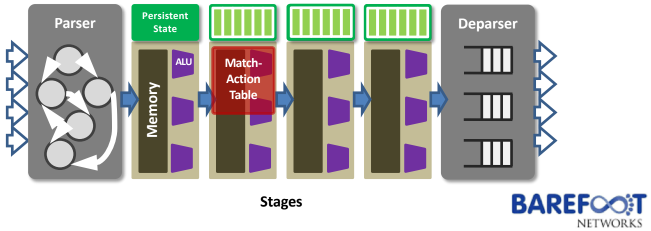
``Second System" Syndrome

- OpenFlow 1.0 limitations
  - One rule table
  - Limited headers and actions
  - Sending packets to the controller
- Later version of OpenFlow
  - More tables, headers, actions
  - But, still never enough
  - Where does it stop?!?

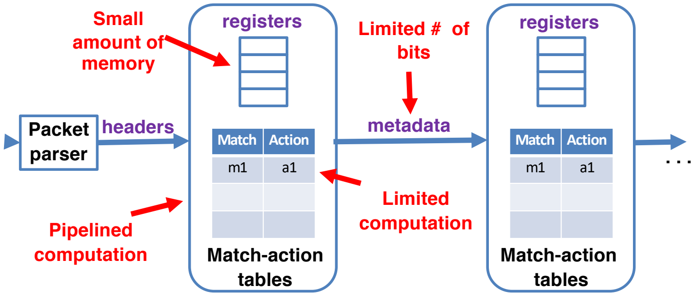
Version	Date	# Headers
OF 1.0	Dec '09	12
OF 1.1	Feb '11	15
OF 1.2	Dec '11	36
OF 1.3	Jun '12	40
OF 1.4	Oct '13	41

Programmable Data Planes

- Data plane designed for programmability
  - Programmable parsing
  - Typed match-action tables
  - Programmable actions
  - Storing and piggybacking metadata



Flexible, But With Constraints



Domain-specific processors: GPUs, TPUs, packet processors, …

P4 Language

(https://p4.org/)

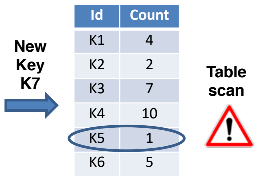
- Protocol independence
  - Configure a packet parser
  - Define typed match+action tables
- Target independence
  - Program without knowledge of switch details
  - Rely on compiler to configure the target switch
- Reconfigurability
  - Change parsing and processing in the field

Heavy-Hitter Detection (Junior IW Project)

Vibhaa Sivamaran ‘17

Heavy-Hitter Detection

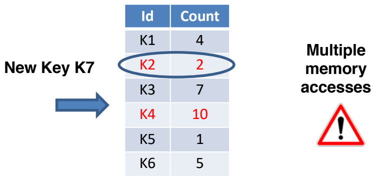
- Heavy hitters
  - The *k* largest trafic flows
  - Flows exceeding count threshold *T*
- Space-saving algorithm
  - Table of (key, value) pairs
  - Evict the key with the minimum value



Id	Count
K1	4
K2	2
K3	7
K4	10
K5	1
K6	5

Approximating the Approximation

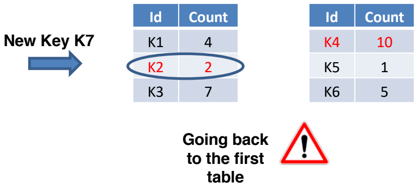
- Evict minimum of *d* entries
  - Rather than minimum of all entries
  - E.g., with *d = 2* hash functions



Id	Count
K1	4
K2	2
K3	7
K4	10
K5	1
K6	5

Approximating the Approximation

- Divide the table over *d* stages
  - One memory access per stage
  - Two different hash functions

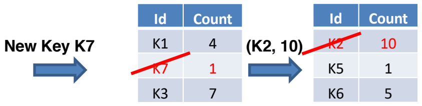


Id	Count
K1	4
K2	2
K3	7

Id	Count
K4	10
K5	1
K6	5

Approximating the Approximation

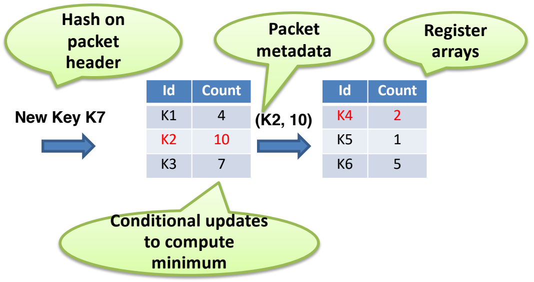
- Rolling minimum across stages
  - Avoid recirculating the packet
  - … by carrying the minimum along the pipeline



Id	Count
K1	4
~~K7~~	1
K3	7

Id	Count
~~K2~~	10
K5	1
K6	5

P4 Prototype and Evaluation



Id	Count
K1	4
K2	10
K3	7

Id	Count
K4	2
K5	1
K6	5

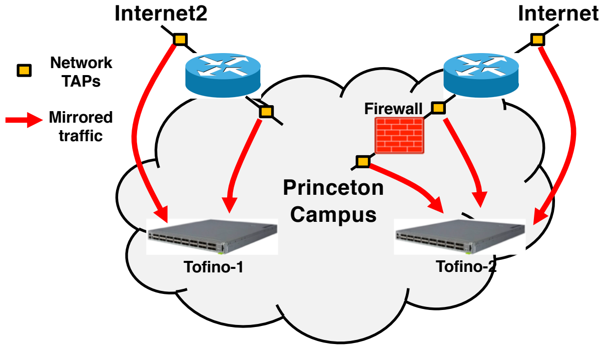
High accuracy with overhead proportional to # of heavy hitters

Undergraduate Student Projects

- OpenFlow
  - Hierarchical heavy hitters (Lavanya Jose ‘12)
  - Server load balancing (Dana Butnariu ‘13)
- P4
  - Heavy-hitter detection (Vibhaa Sivaraman ‘17)
  - Censorship circumvention (Blake Lawson ‘17)
  - Round-trip time measurement (Mack Lee ‘18)
  - Operating system fingerprinting (Sherry Bai ‘19)
  - Surveillance protection (Trisha Datta ‘19)
  - Heavy-hitters by domain name (Jason Kim ‘21)

Princeton Campus Deployment

(https://p4campus.cs.princeton.edu)



- Deployed: Microburst analysis, heavy hitter detection, trace anonymization
- In progress: surveillance protection, RTT, DNS heavy hitters, OS fingerprinting

Conclusion

- Rethinking networking
  - Open interfaces to the data plane
  - Separation of control and data
  - Deployment of new solutions
- Significant momentum
  - In industry and in academic research
- Next steps
  - Enterprises
  - Cellular (5G) networks