Computer Networks

Software Defined Networks

Material based on courses at Princeton, MIT

The Networking "Planes"

Data plane: processing and delivery of packets with local forwarding state

- Forwarding state + packet header → forwarding decision
- Filtering, buffering, scheduling

Control plane: computing the forwarding state in routers

- Determines how and where packets are forwarded
- Routing, traffic engineering, failure detection/recovery, …

Management plane: configuring and tuning the network

- ACL config, device provisioning, …

Timescales

	Data	Control	Management
Time-scale	Packet (nsec)	Event (10 msec to sec)	Human (min to hours)
Location	Linecard hardware	Router software	Humans or scripts

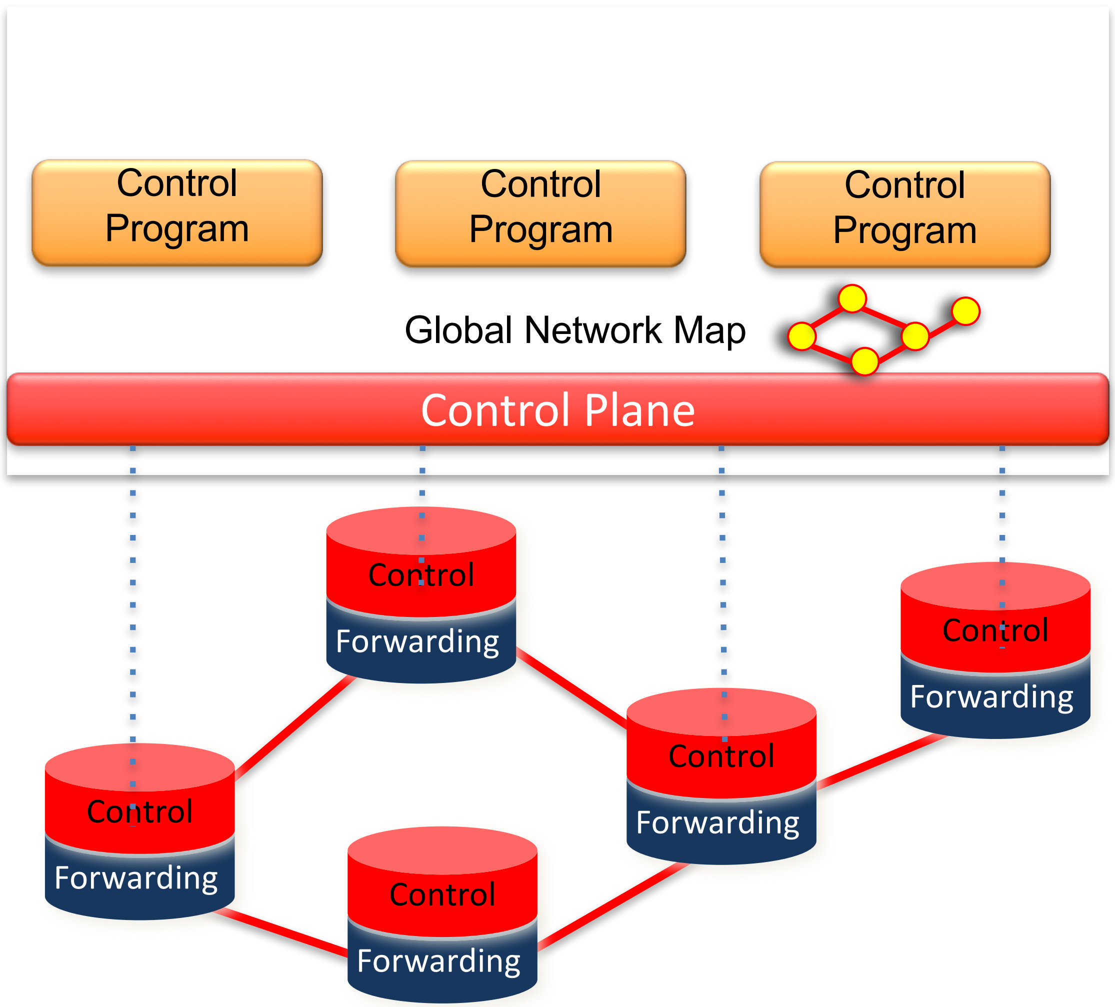
Software Defined Network

A network in which the control plane is physically separate from the data plane.

and

A single (logically centralized) control plane controls several forwarding devices.

Software Defined Network (SDN)

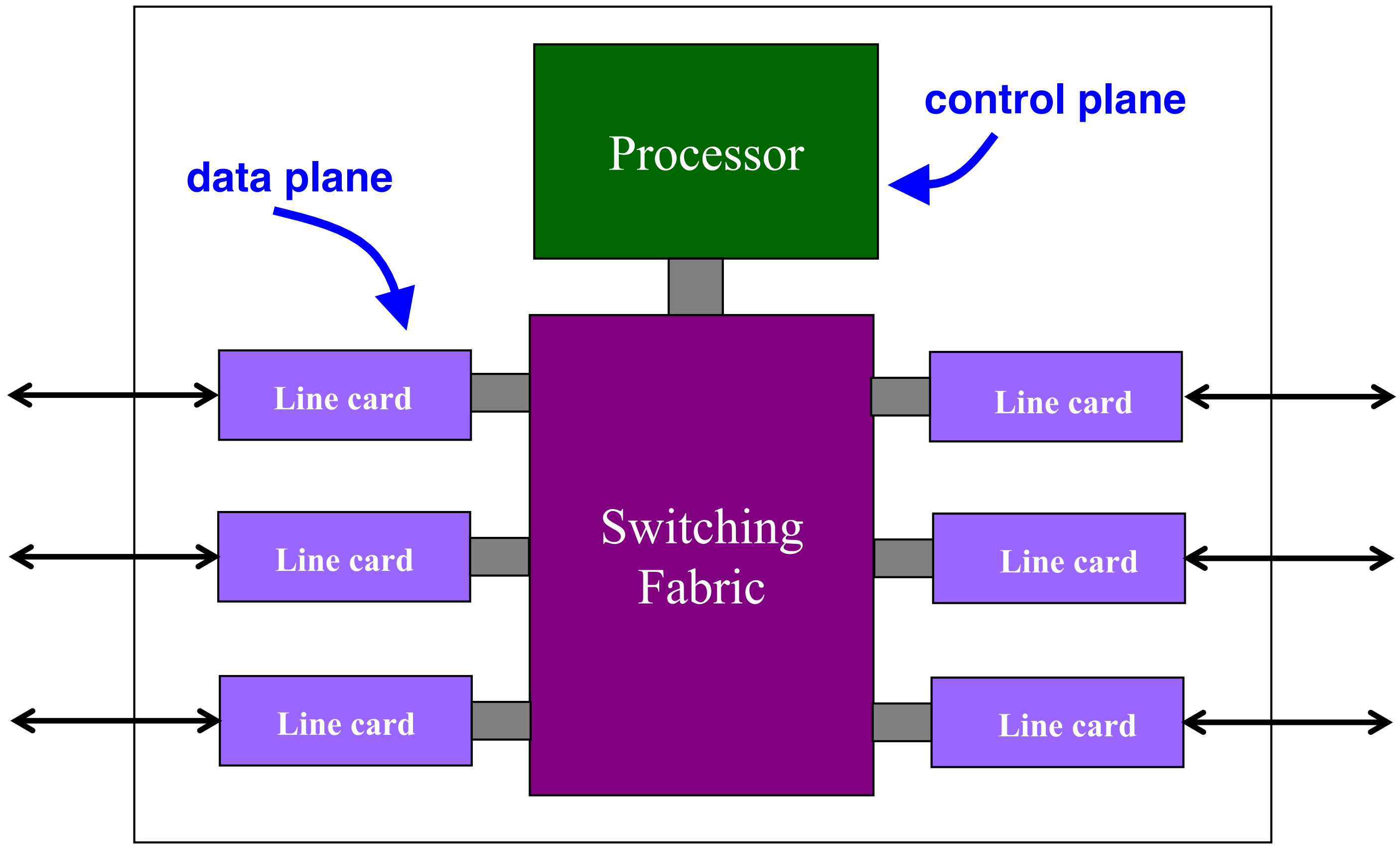


Questions

What are the pros and cons of these two approaches:

- traditional control and data planes
- SDN approach of separating the two planes

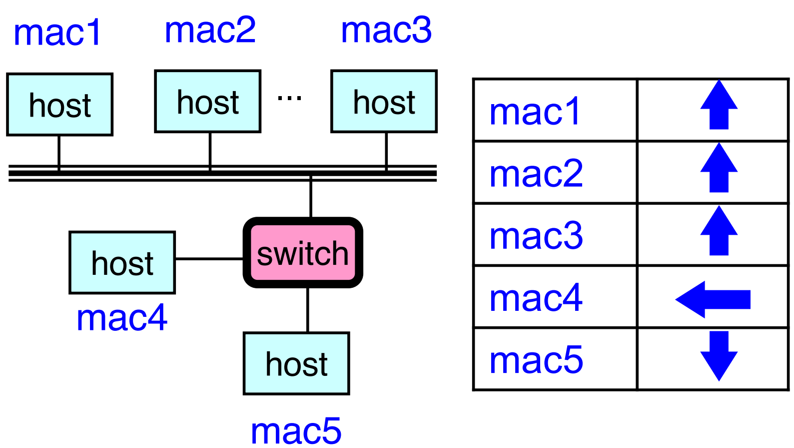
Data and Control Planes



Switch: Match on Destination MAC

MAC addresses are location independent

- Assigned by the vendor of the interface card
- Cannot be aggregated across hosts in the LAN

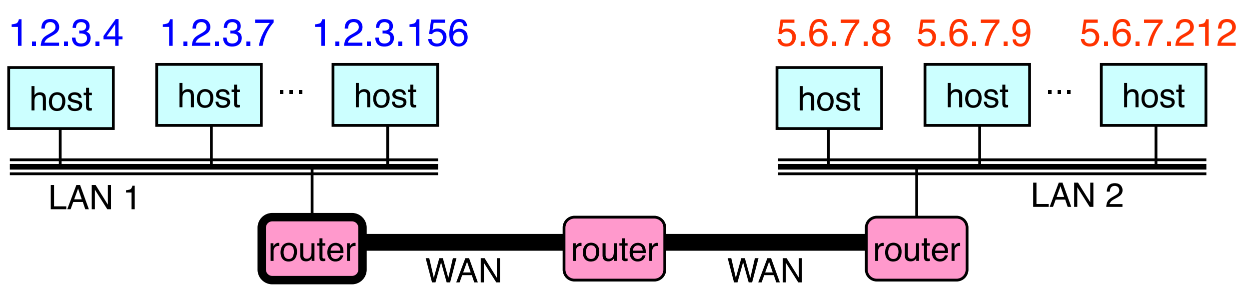


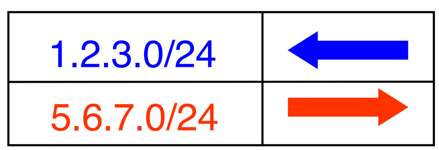
Implemented using a hash table or a content addressable memory.

IP Routers: Match on IP Prefix

IP addresses grouped into common subnets

- Allocated by ICANN, regional registries, ISPs, and within individual organizations
- Variable-length prefix identified by a mask length

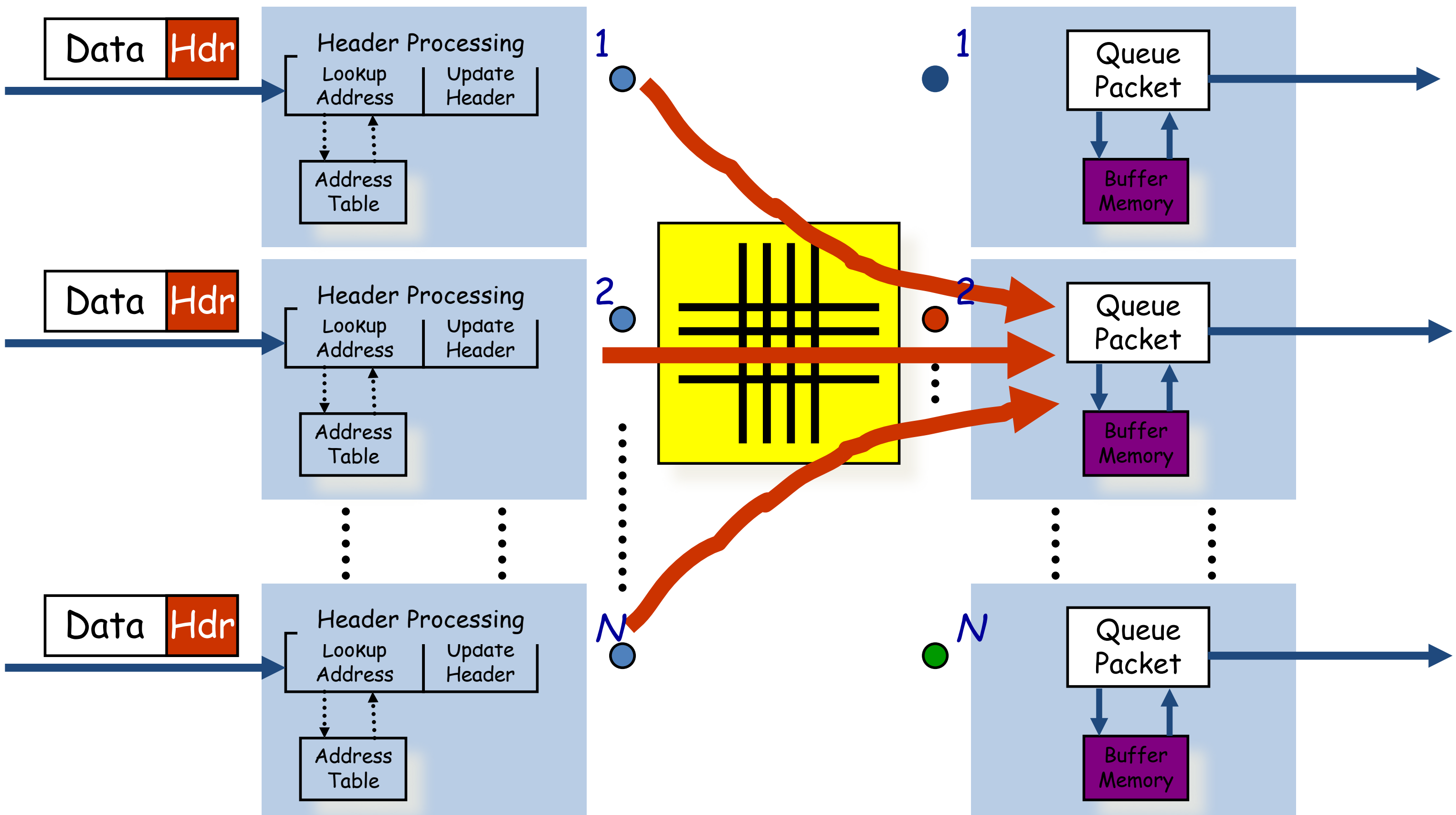




forwarding table

Prefixes may be nested. Routers identify the *longest matching* prefix.

Switch Fabric: From Input to Output



Access Control

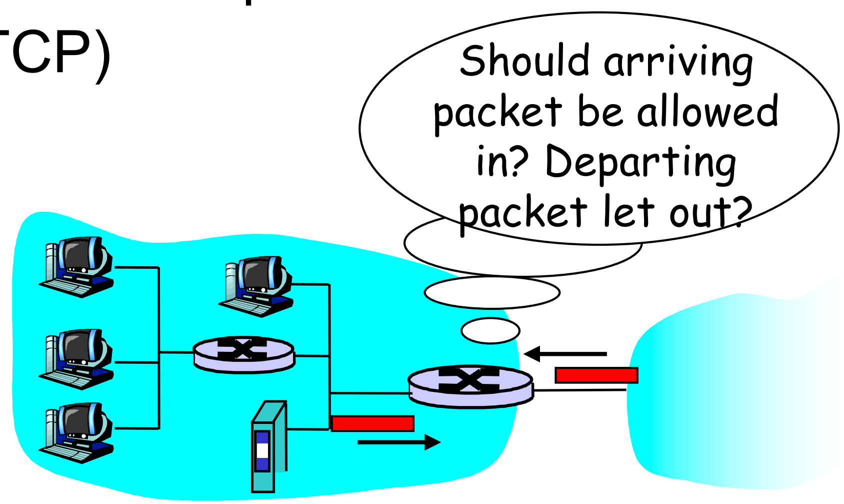
Access Control: Packet Filtering

"Five tuple" for access control lists (ACLs)

- Source and destination IP addresses
- TCP/UDP source and destination ports
- Protocol (e.g., UDP vs. TCP)

Can be more sophisticated

- E.g., block all TCP SYN packets from outside
- E.g., stateful



Applying Access Control Lists

Ordered list of "accept/deny" clauses

- A clause can have wild cards
- Clauses can overlap
- … so order matters

Packet classification

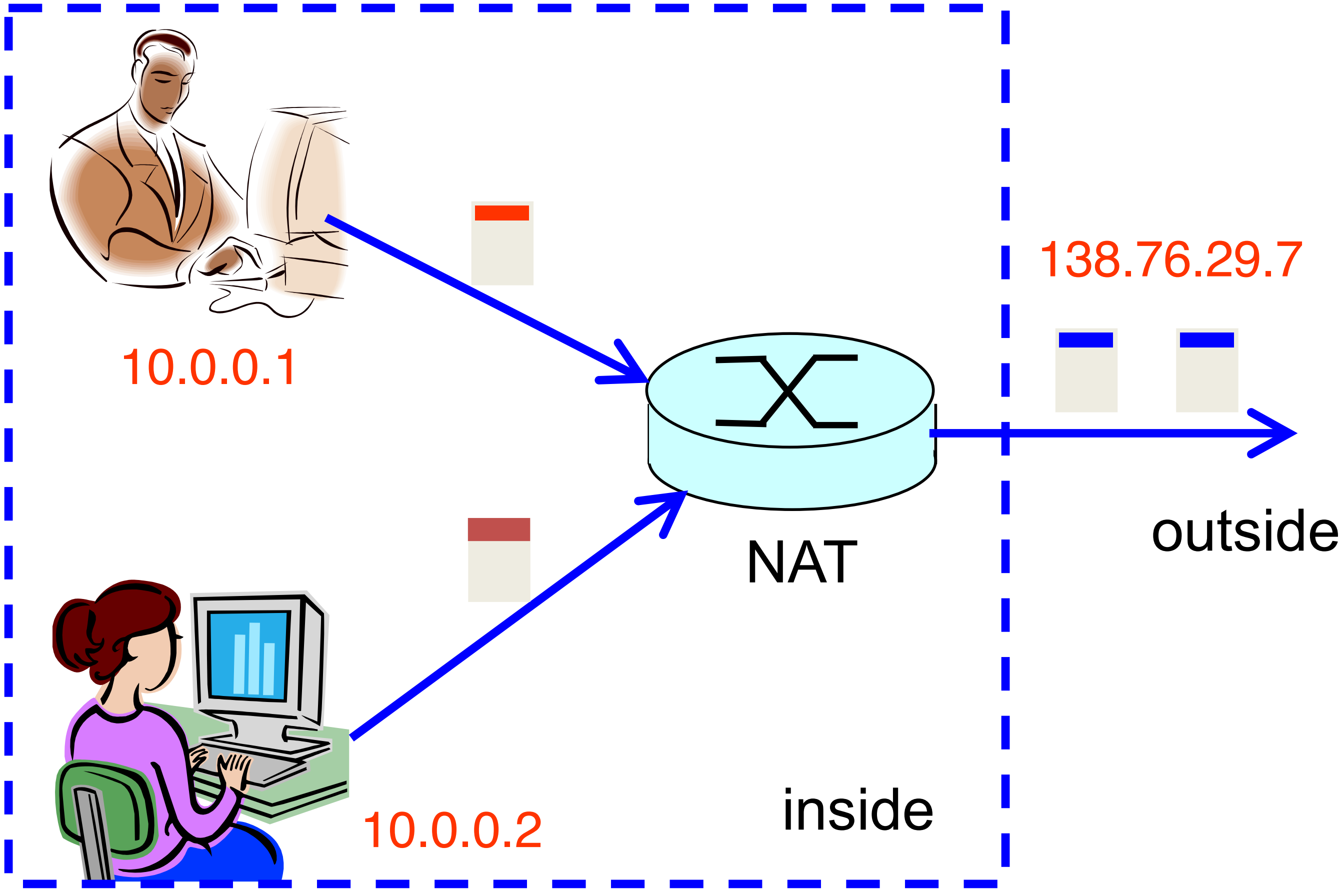
- Given all of the fields
- … identify the match with the highest priority

Src=1.2.3.4, Dest=5.6.7.8	Deny
Dest=1.2.3.*	Allow
Dest=1.2.3.8, Dport!=53	Deny
Src=1.2.3.7, Dport=100	Allow
Dport=100	Deny

Questions

Is this model powerful enough? What are the performance implications?

Network Address Translation (NAT)



Mapping Addresses and Ports

Remap IP addresses and TCP/UDP port numbers

- Addresses: between end-host and NAT addresses
- Port numbers: to ensure each connection is unique

Create table entries as packets arrive

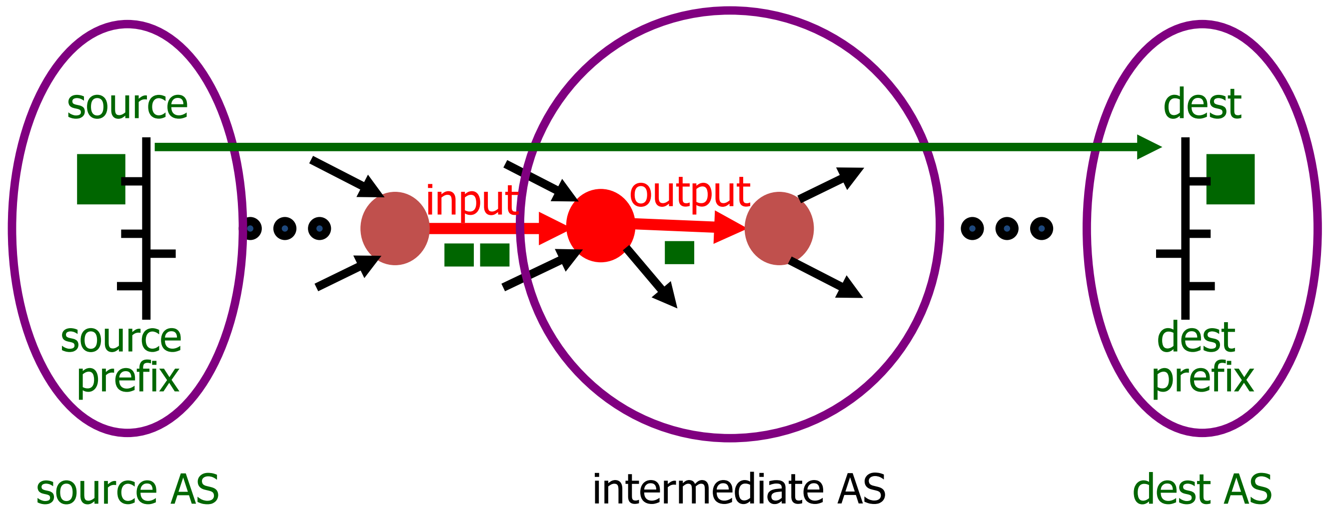
- Src 10.0.0.1, Sport 1024, Dest 1.2.3.4, Dport 80
  - Map to Src 138.76.29.7, Sport 1024, Dest 1.2.3.4, Dport 80
- Src 10.0.0.2, Sport 1024, Dest 1.2.3.4, Dport 80
  - Map to Src 138.76.29.7, Sport 1025, Dest 1.2.3.4, Dport 80

Questions

What are the difficulties in using NATs?

Traffic Monitoring

Observing Traffic Passing Through



- Applications of traffic measurement
  - Usage-based billing
  - Network engineering
  - Detecting anomalous traffic

Passive Traffic Monitoring

- Counting the traffic
  - Match based on fields in the packet header
  - … and update a counter of #bytes and #packets
- Examples
  - Link
  - IP prefixes
  - TCP/UDP ports
  - Individual "flows"

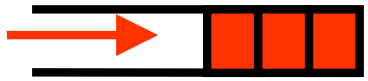
Dest Prefix	#Packets	#Bytes
1.2.3.0/24	3	1500
7.8.0.0/16	10	13000
8.0.0.0/8	100	85020
7.7.6.0/23	1	40

- Challenges
  - Identify traffic aggregates in advance vs. reactively
  - Summarizing other information (e.g., time, TCP flags)

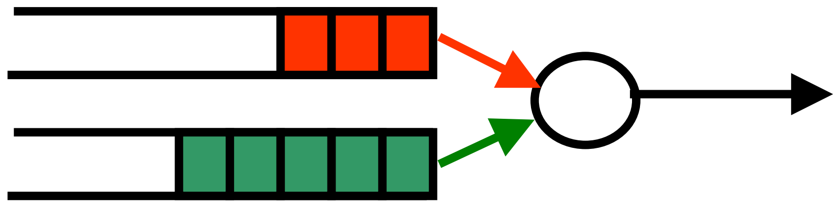
Resource Allocation: Buffering, Scheduling, Shaping, and Marking

Buffering

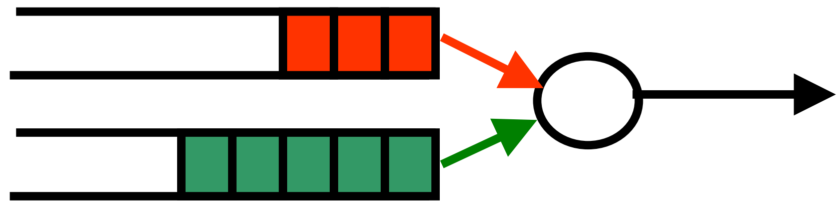
- Drop-tail FIFO queue
  - Packets served in the order they arrive
  - … and dropped if queue is full



- Random Early Detection (RED)
  - When the buffer is nearly full
  - … drop or mark some packets to signal congestion
- Multiple classes of traffic
  - Separate FIFO queue for each flow or traffic class
  - … with a link scheduler to arbitrate between them



Link Scheduling

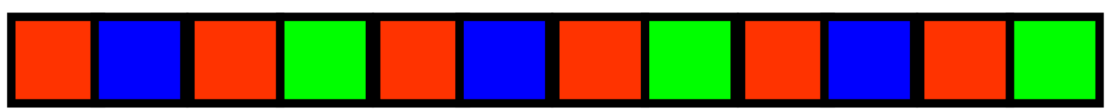


Strict priority

- Assign an explicit rank to the queues
- … and serve the highest-priority backlogged queue

Weighted fair scheduling

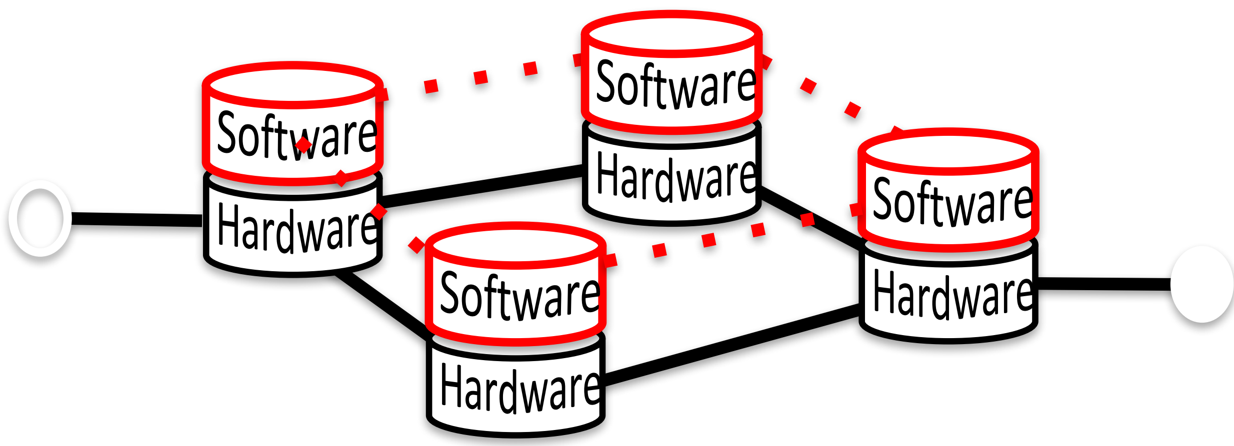
- Interleave packets from different queues
- …in proportion to weights



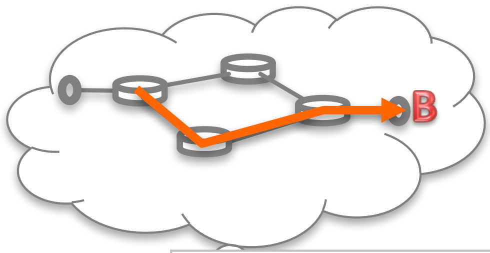
50% red, 25% blue, 25% green

Control Plane

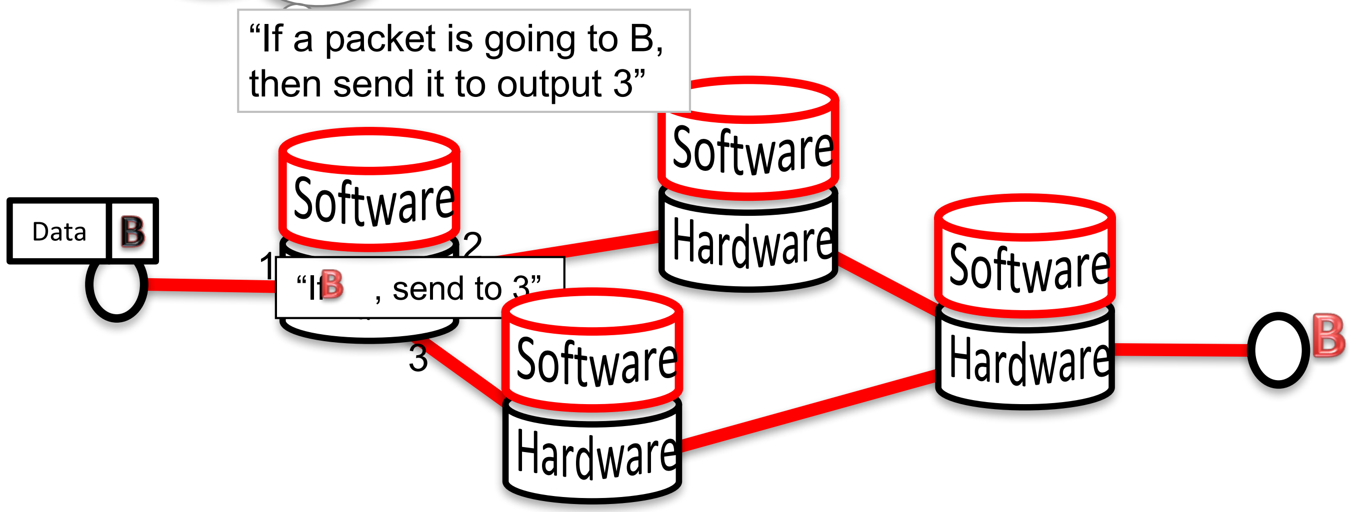
Control Plane



Example: Link-state routing (OSPF, IS-IS)



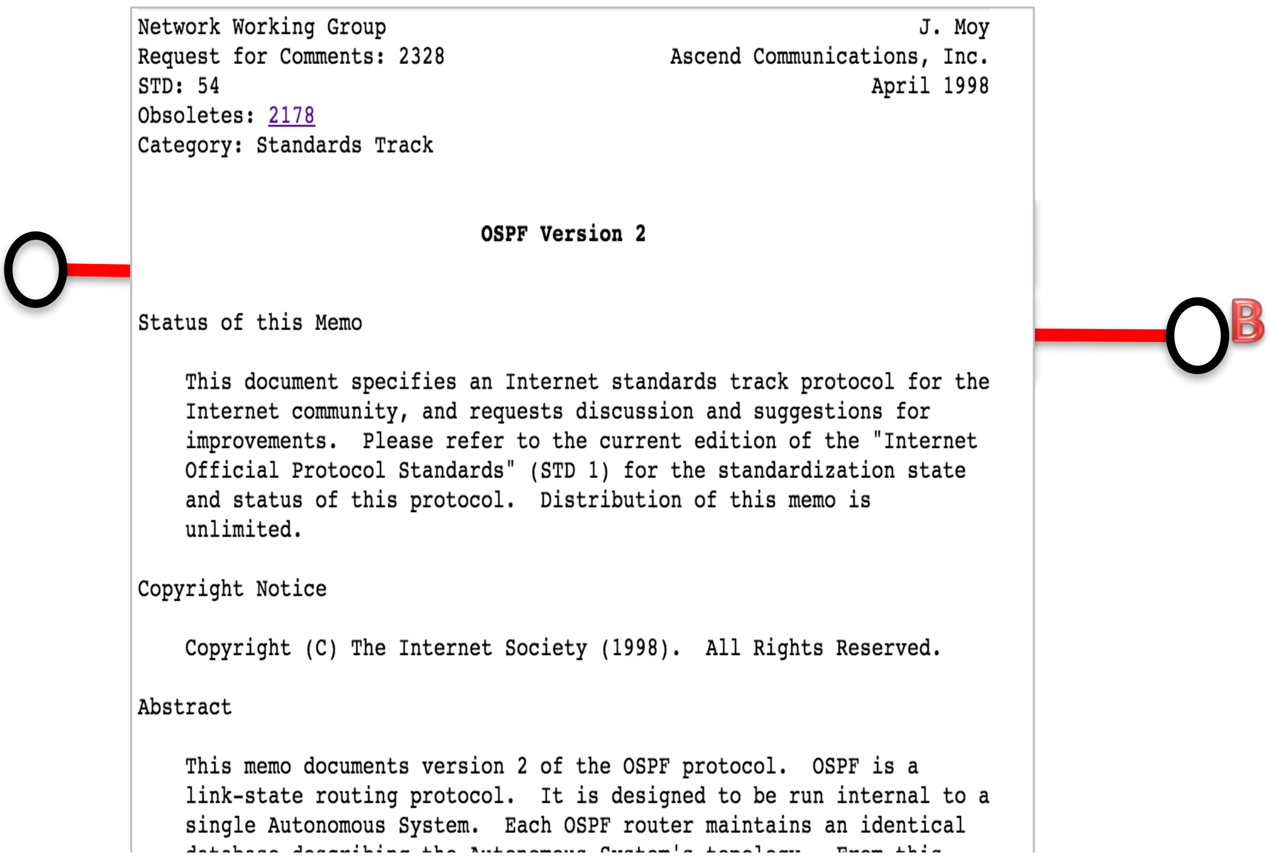
1. Figure out which routers and links are present.
2. Run Dijkstra’s algorithm to find shortest paths.



95%

1. Figure out which routers and links are present.
2. Run Dijkstra's algorithm to find shortest paths.

5%



Network Working Group J. Moy
Request for Comments: 2328 Ascend Communications, Inc.
STD: 54 April 1998
Obsoletes: 2178
Category: Standards Track

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). All Rights Reserved.

Abstract

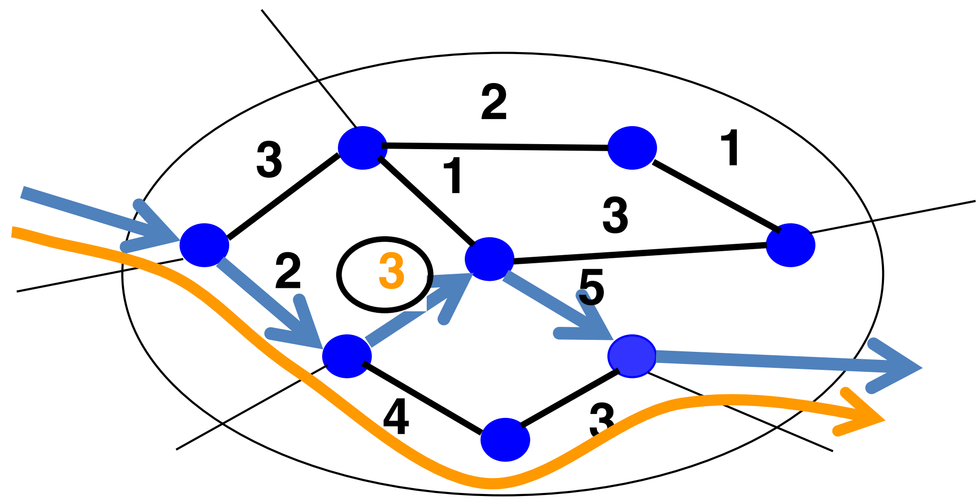
This memo documents version 2 of the OSPF protocol. OSPF is a link-state routing protocol. It is designed to be run internal to a single Autonomous System. Each OSPF router maintains an identical

Example: Traffic Engineering

Which paths to use to deliver traffic?

How to control paths?

- Set link weights used by routing protocol



Outline

The networking “planes”

➡ Traditional network challenges

How SDN changes the network?

Why is SDN happening now? (A brief history)

Traditional Network Challenges

(Too) many task-specific control mechanisms

- Routing, addressing, access control, QoS
- N

Indirect

- M … what
  y
- E

Uncoordinated control

- Cannot control which router updates first

The network is

- Hard to reason about
- Hard to evolve
- Expensive

Example 1: Inter-domain Routing

Today's inter-domain routing protocol, BGP, artificially constrains routes

- Routing only on **destination IP address blocks**
- Can only influence **immediate neighbors**

Application-specific peering

– Route video traffic one way, and non-video another

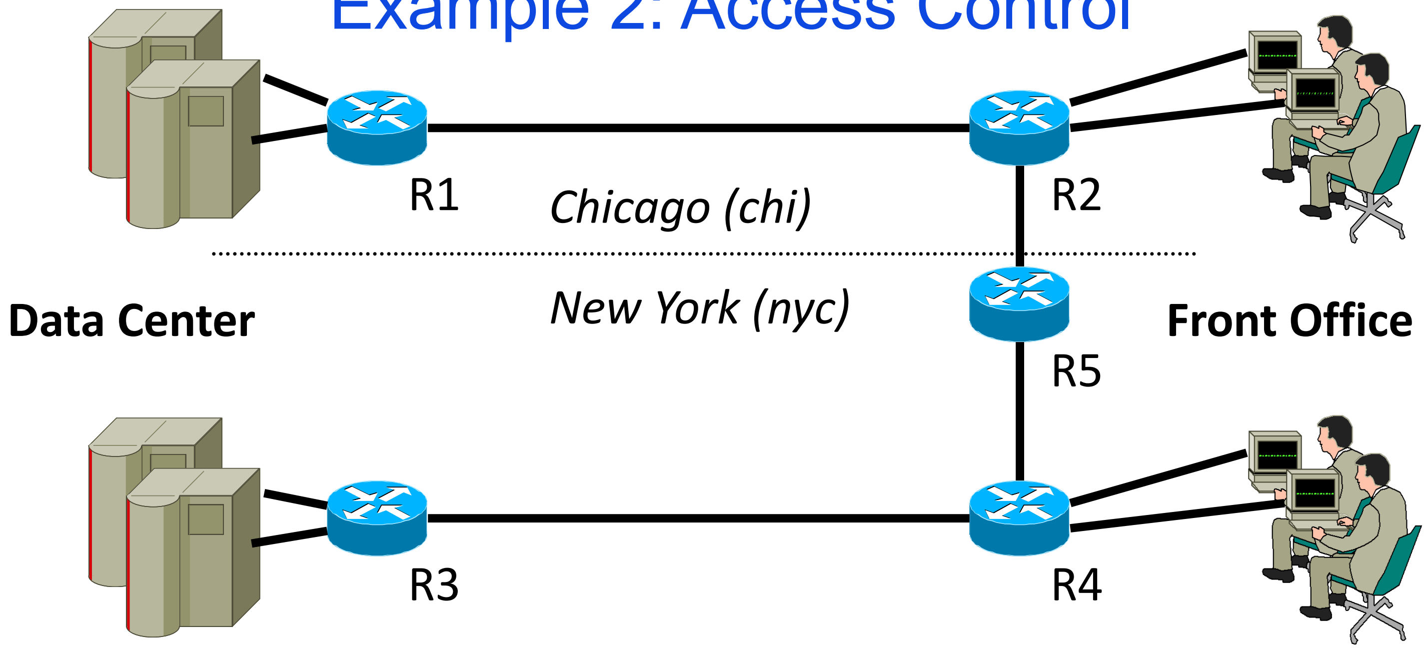
Blocking denial-of-service traffic

– Dropping unwanted traffic further upstream

Inbound traffic engineering

– Splitting incoming traffic over multiple peering links

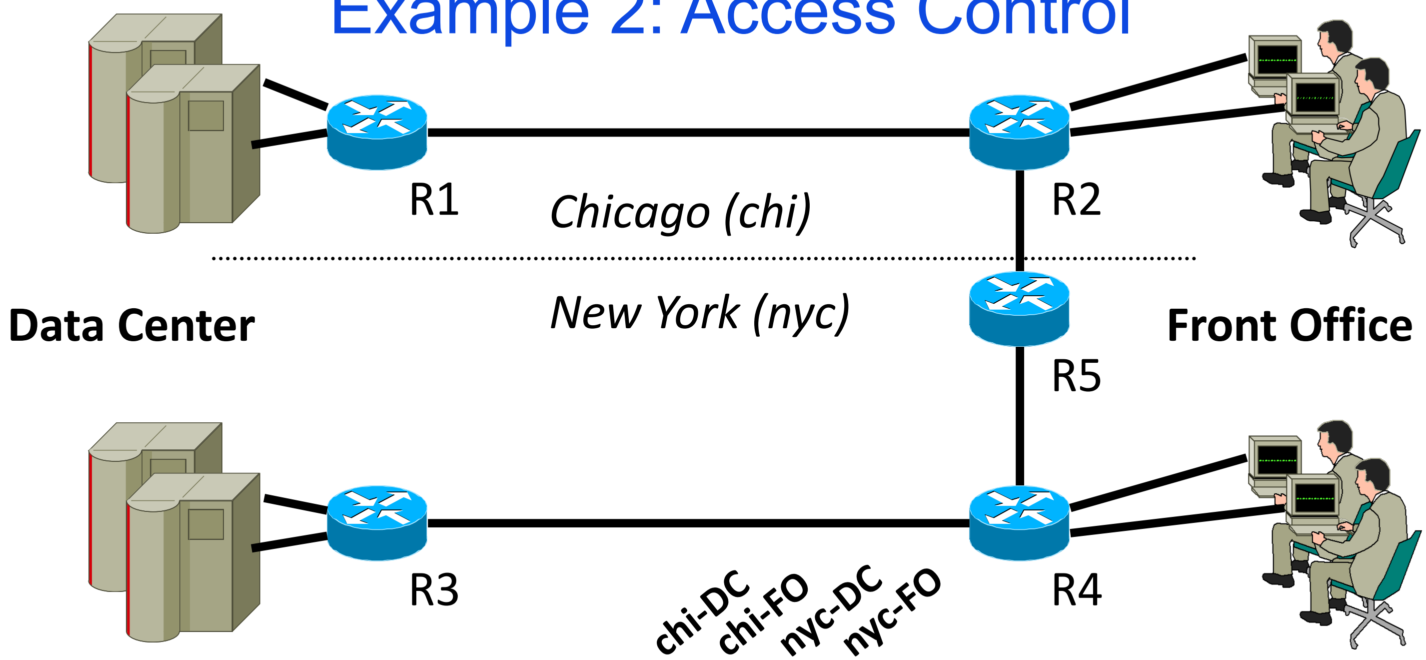
Example 2: Access Control



Two locations, each with data center & front office

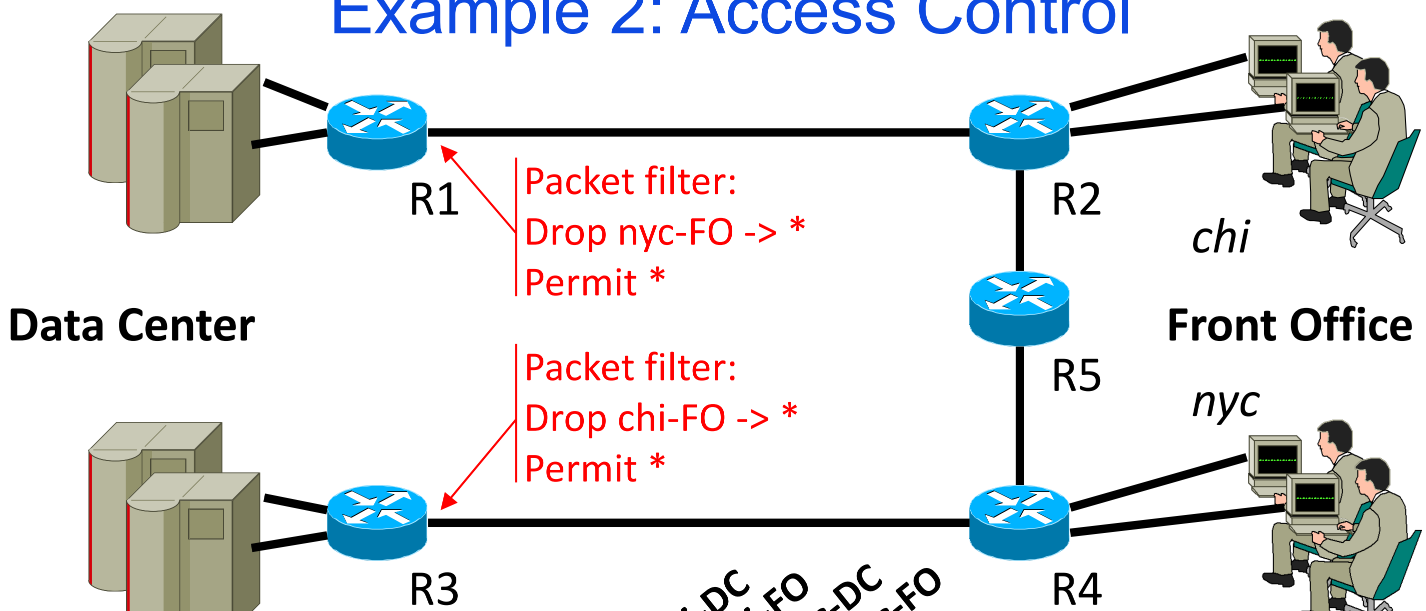
All routers exchange routes over all links

Example 2: Access Control



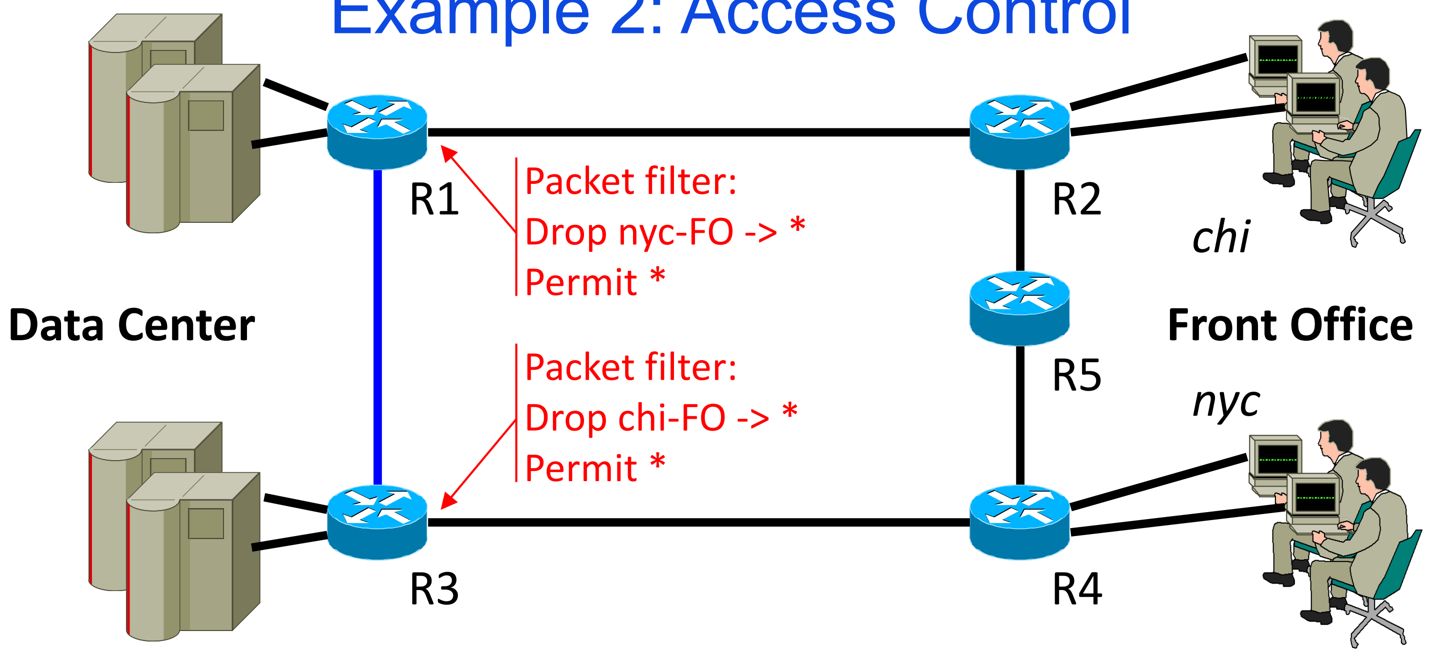
	chi-DC	chi-FO	nyc-DC	nyc-FO
chi-DC		●	●	⊘
chi-FO	●		⊘	●
nyc-DC	●	⊘		●
nyc-FO	⊘	●	●	

Example 2: Access Control



	chi-DC	chi-FO	nyc-DC	nyc-FO
chi-DC		●	●	⊘
chi-FO	●		⊘	●
nyc-DC	●	⊘		●
nyc-FO	⊘	●	●	

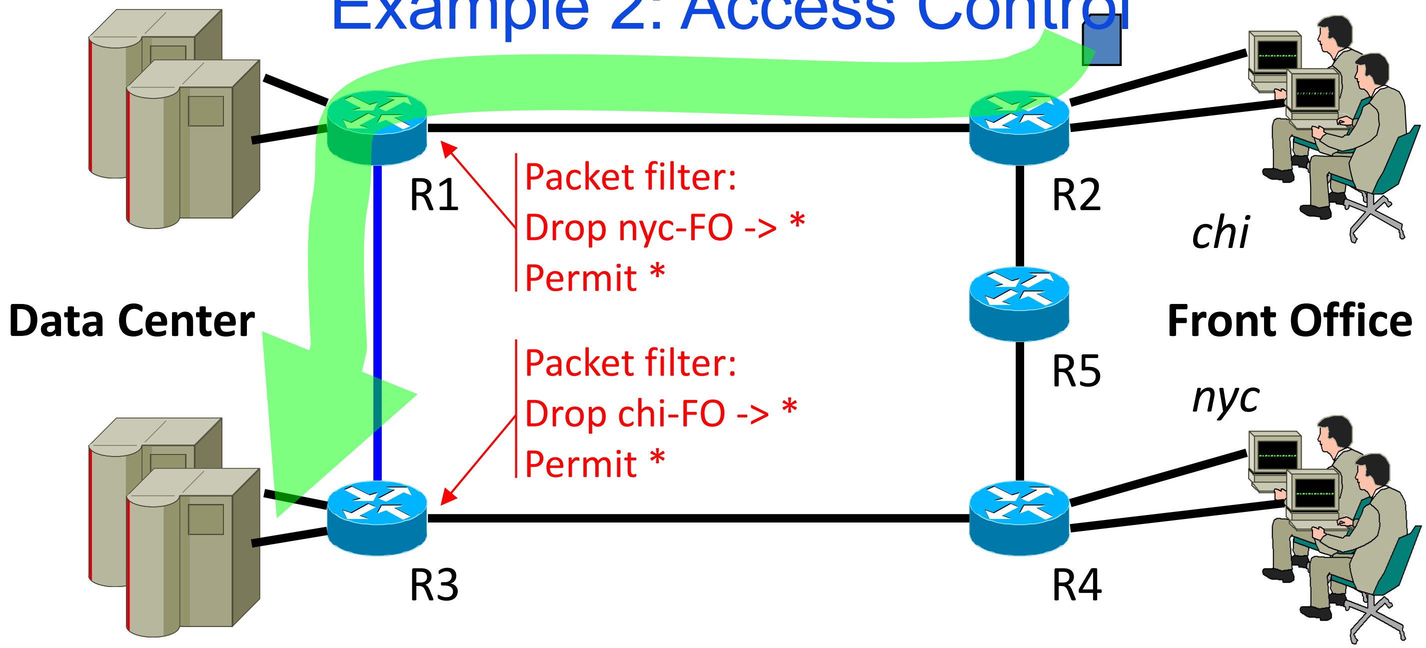
Example 2: Access Control



A new short-cut link added between data centers

Intended for backup traffic between centers

Example 2: Access Control



Oops – new link lets packets violate ***access control policy***!

Routing changed, but

Packet filters don't update automatically

Outline

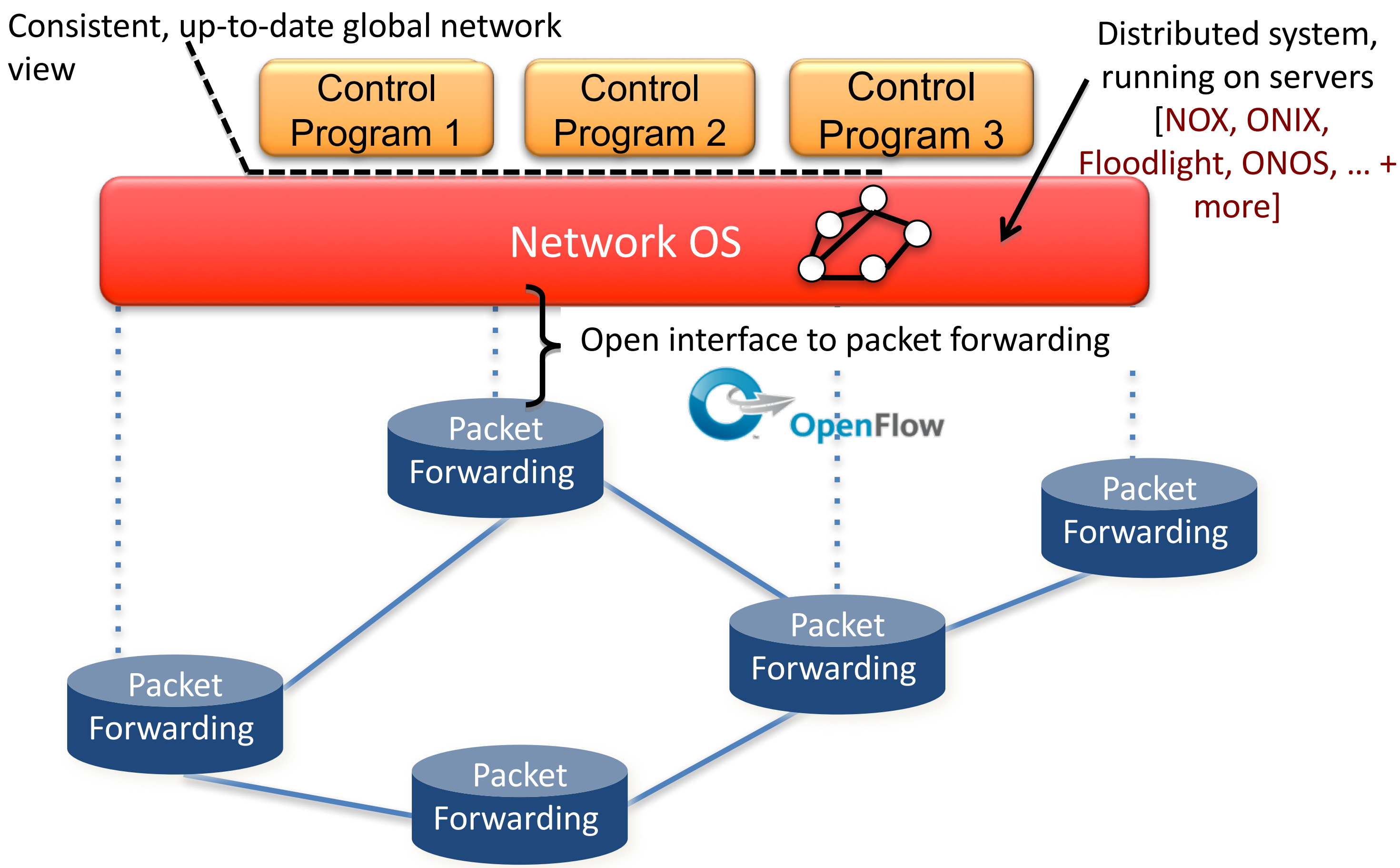
The networking "planes"

Traditional network challenges

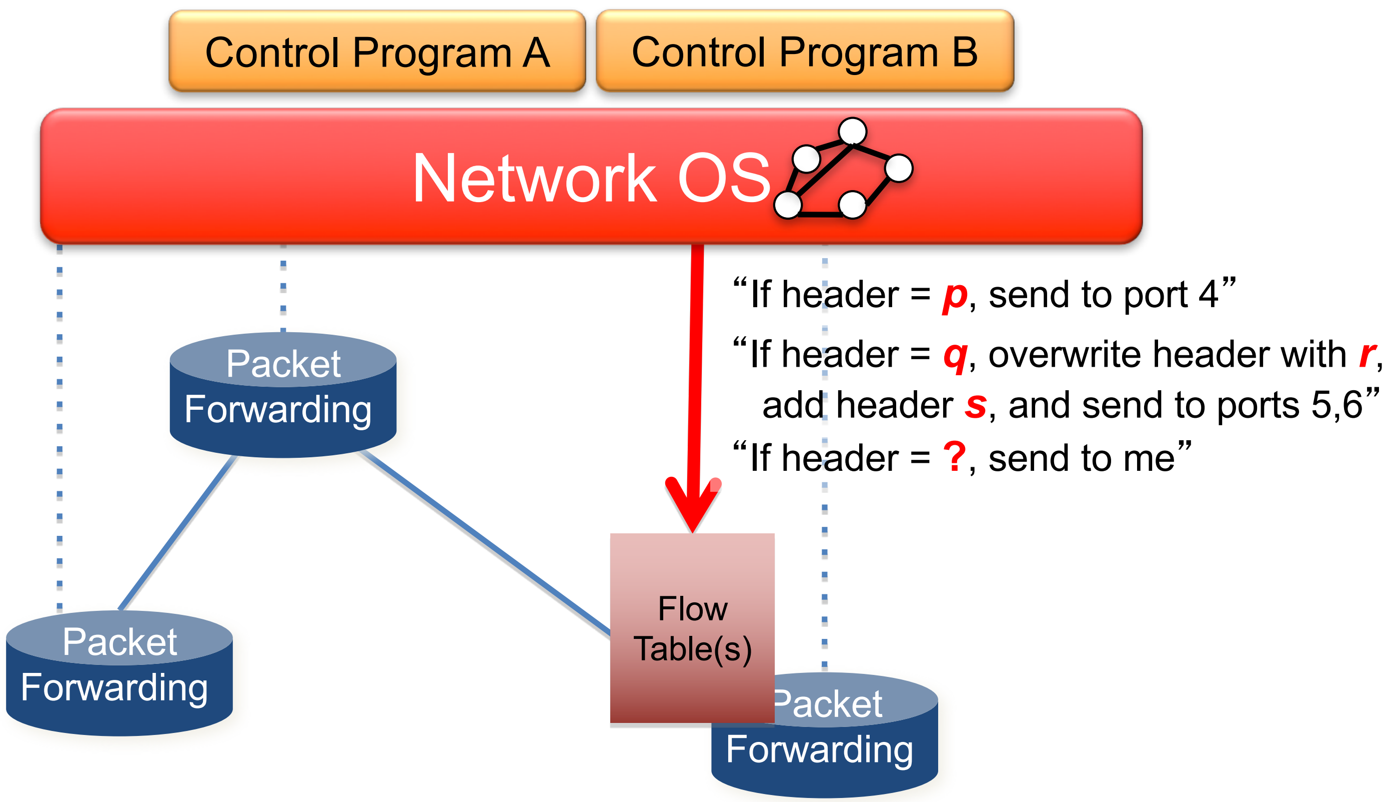
➡ How SDN changes the network?

Why is SDN happening now? (A brief history)

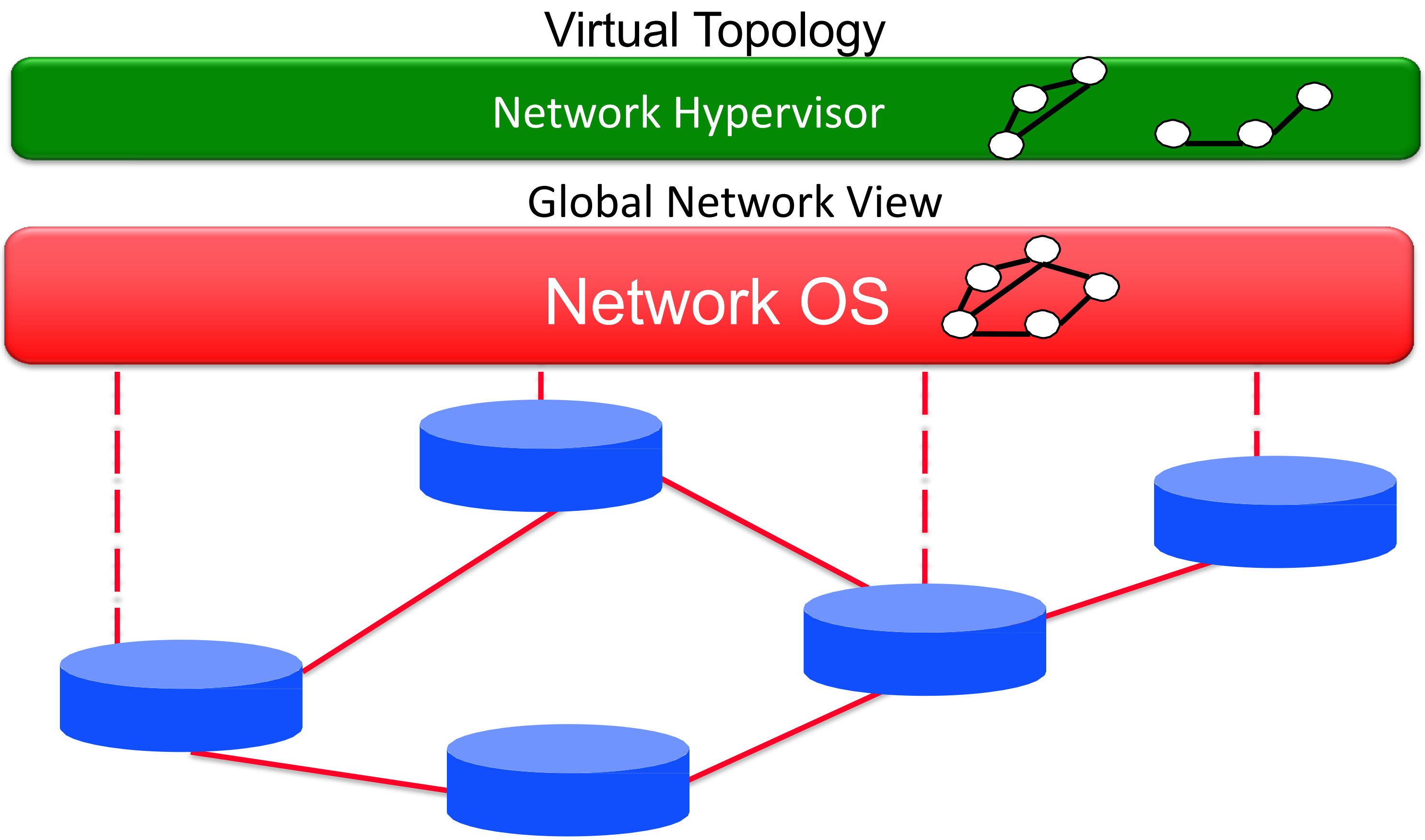
Software Defined Network (SDN)



OpenFlow


Control Program A
Control Program B
Network OS
"If header = p, send to port 4"
"If header = q, overwrite header with r, add header s, and send to ports 5,6"
"If header = ?, send to me"
Packet Forwarding
Flow Table(s)
Packet Forwarding
Packet Forwarding

Network Hypervisor



Does SDN Simplify the Network?

Abstraction doesn't eliminate complexity

- NOS, Hypervisor are still complicated pieces of code

SDN main achievements

- Simplifies interface for control program (user-specific)
- Pushes complexity into reusable code (SDN platform)

Just like OS & compilers….

Outline

The networking "planes"

Traditional network challenges

How SDN changes the network?

➡ Why is SDN happening now? (A brief history)

The Road to SDN

Active Networking: 1990s

- First attempt make networks programmable
- Demultiplexing packets to software programs, network virtualization, …

Control/Dataplane Separation: 2003-2007

- ForCes [IETF],
  RCP, 4D [Princeton, CMU],
  SANE/Ethane [Stanford/Berkeley]
- Open interfaces between data and control plane, logically centralized control

OpenFlow API & Network Oses: 2008

- OpenFlow switch interface [Stanford]
- NOX Network OS [Nicira]

N. Feamster et al., "The Road to SDN: An Intellectual History of Programmable Networks", ACM SIGCOMM CCR 2014.

SDN Drivers

Rise of merchant switching silicon

- Democratized switching
- Vendors eager to unseat incumbents

Cloud / Data centers

- Operators face real network management problems
- Extremely cost conscious; desire a lot of control

The right balance between vision & pragmatism

- OpenFlow compatible with existing hardware

A “killer app”: Network virtualization

Virtualization is Killer App for SDN

Consider a multi-tenant datacenter

- Want to allow each tenant to specify virtual topology
- This defines their individual policies and requirements

Datacenter's network hypervisor compiles these virtual topologies into set of switch configurations

- Takes 1000s of individual tenant virtual topologies
- Computes configurations to implement all simultaneously

Programmable switches

Slides courtesy of Patrick Bosshart, Nick McKeown, and Mihai Budiu

The consequences of SDN

- Move control plane out of the switch onto a server.
- Well-defined API to data plane (OpenFlow)
  - Match on fixed headers, carry out fixed actions.
  - Which headers: Lowest common denominator (TCP, UDP, IP, etc.)
- Write your own control program.
  - Traffic Engineering
  - Access Control Policies

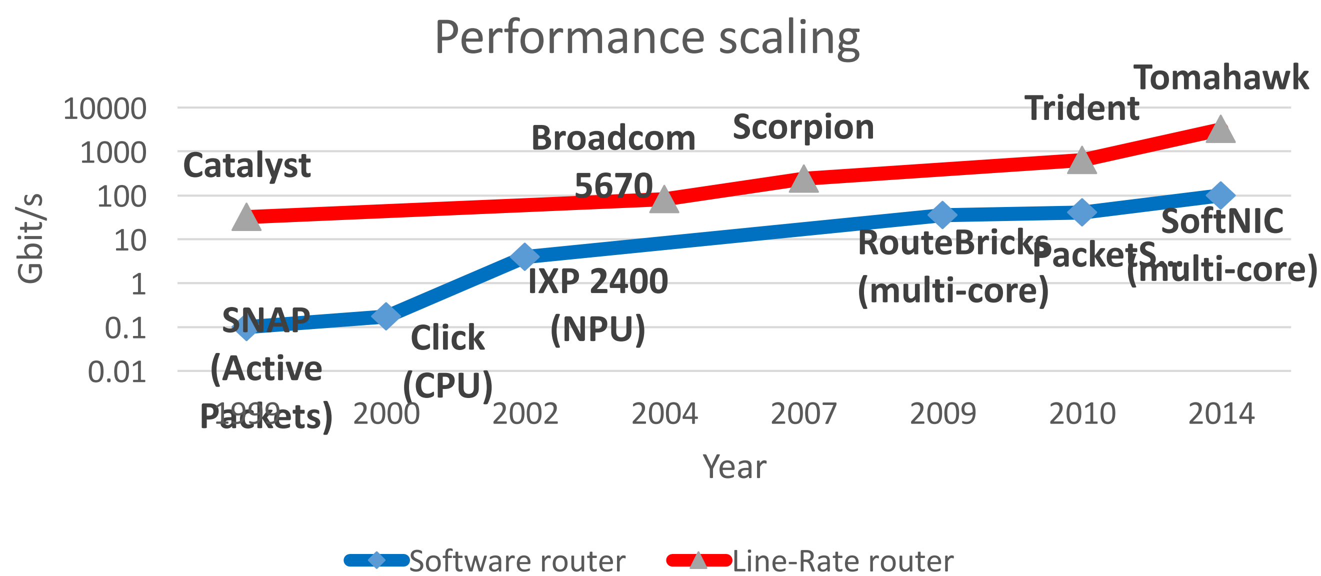
The network isn’t truly software-defined

- What else might you want to change in the network?
- RED, WFQ, PIE, XCP, RCP, DCTCP, …
- Lot of performance left on the table.
- What about new protocols like IPv6?

The solution: a programmable switch

- Change switch however you like.
- Each user ”programs” their own algorithm.
- Much like we program desktops, smartphones, etc.

Early attempts at programmable routers



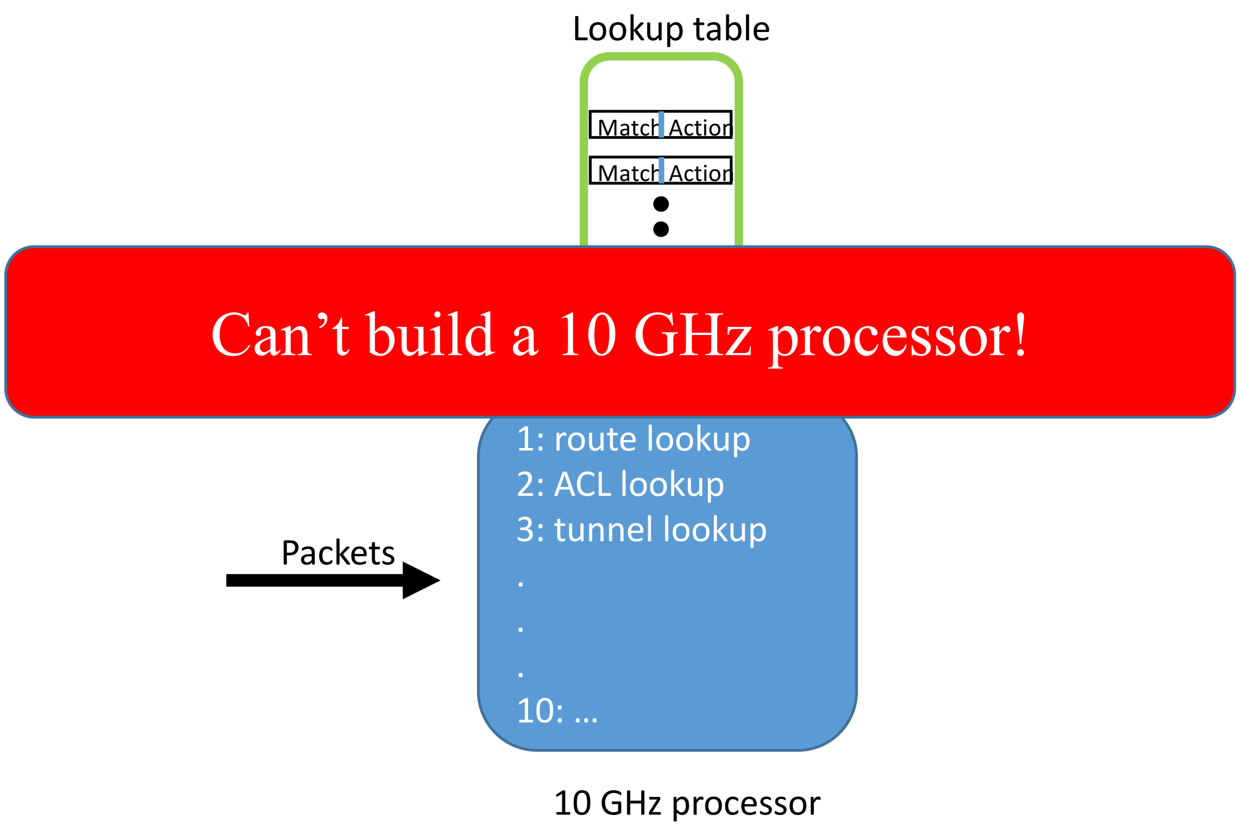
- 10—100 x loss in performance relative to line-rate, fixed-function routers
- Unpredictable performance (e.g., cache contention)

Performance requirements at line-rate

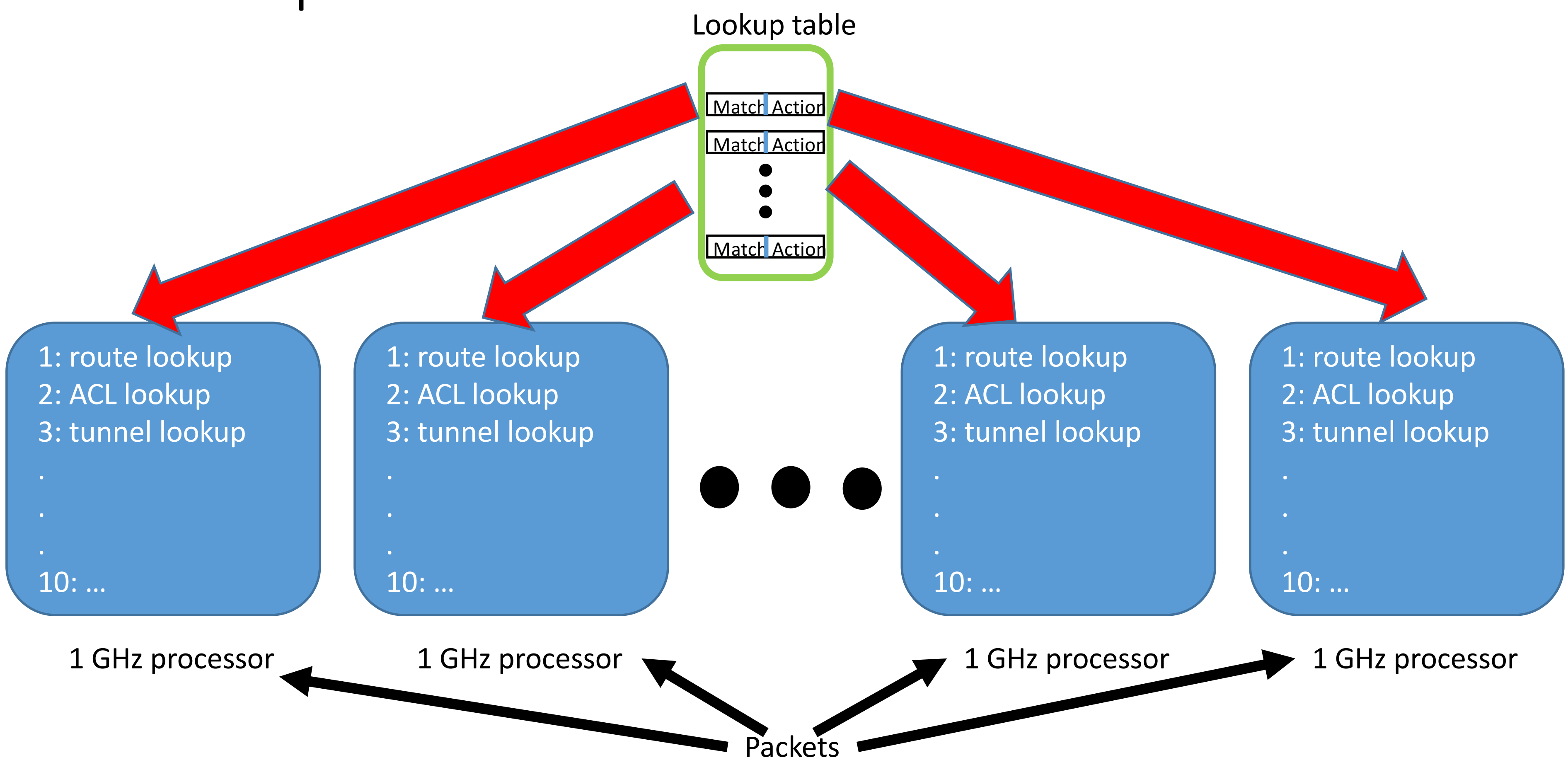
- Aggregate capacity ~ 1 Tbit/s
- Packet size ~ 1000 bits
- ~10 operations per packet (e.g., routing, ACL, tunnels)

Need to process 1 billion packets per second, 10 ops per packet

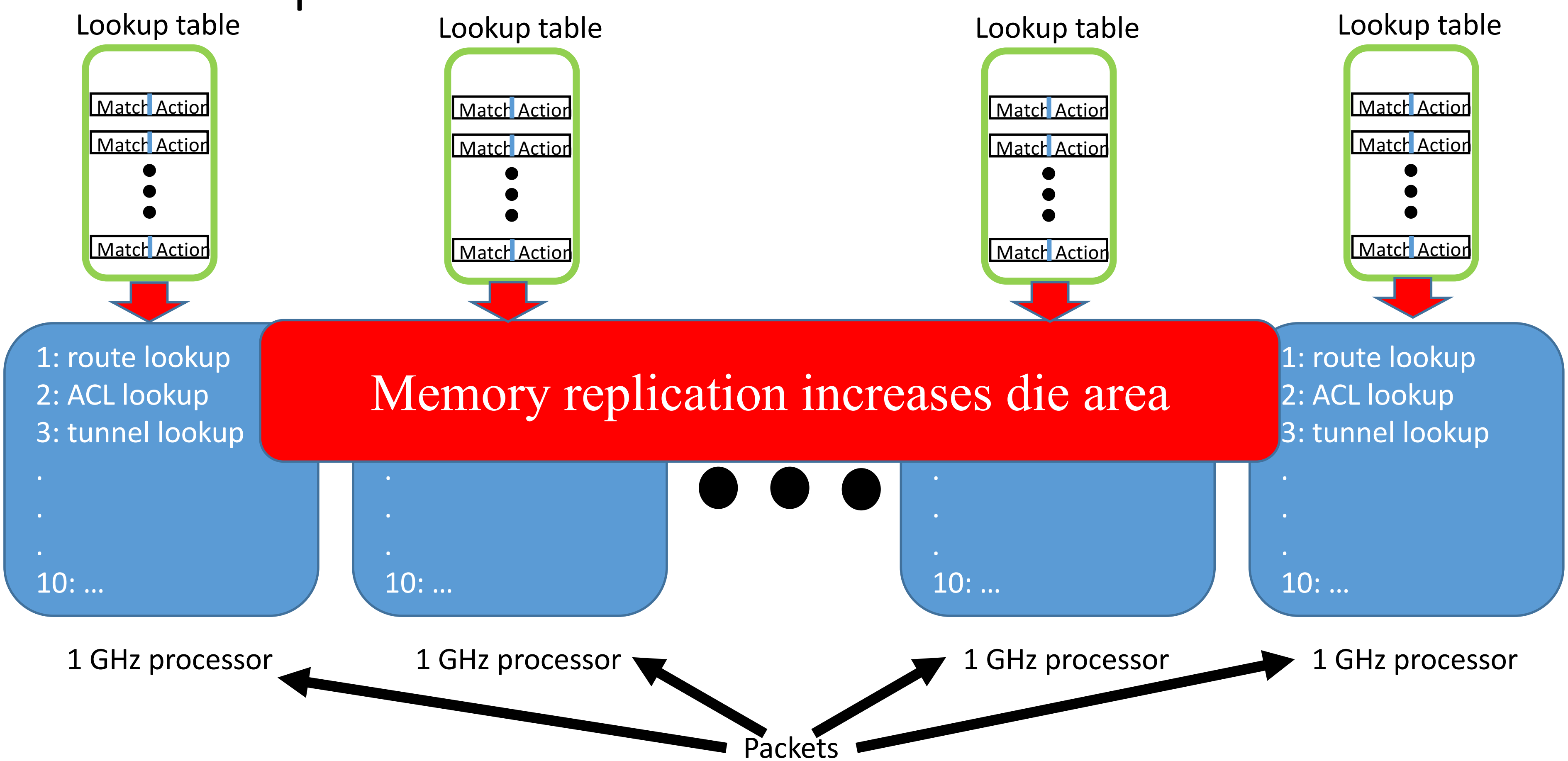
Single processor architecture



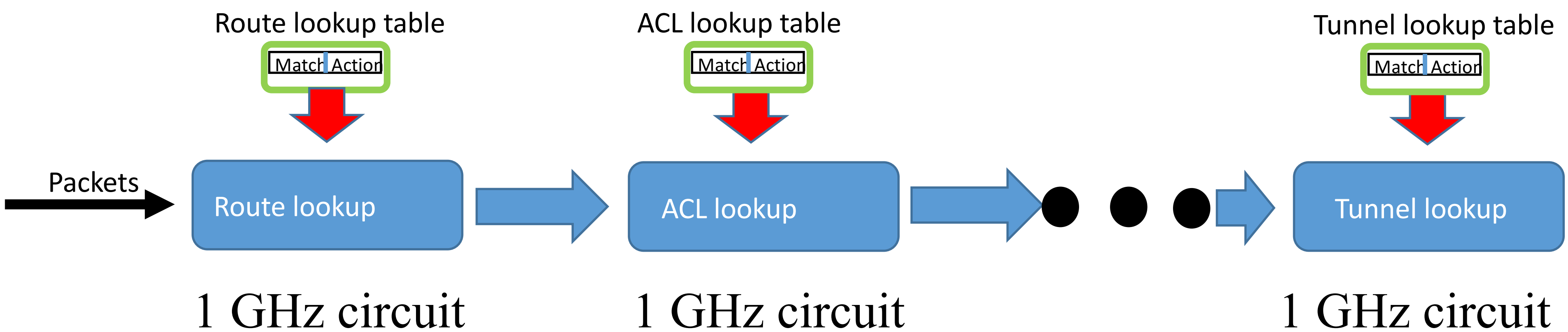
Packet-parallel architecture



Packet-parallel architecture

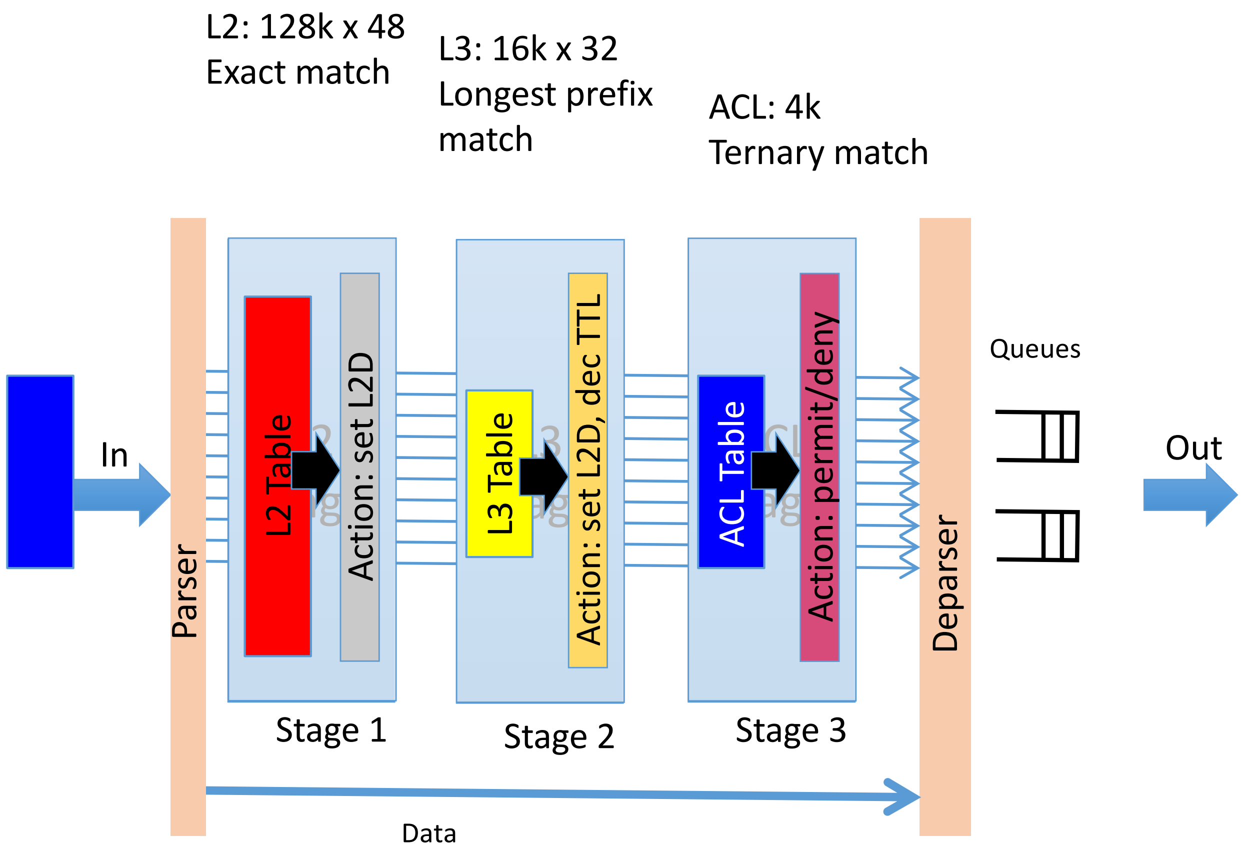


Function-parallel or pipelined architecture



- Factors out global state into per-stage local state
- Replaces full-blown processor with a circuit
- But, needs careful circuit design to run at 1 GHz

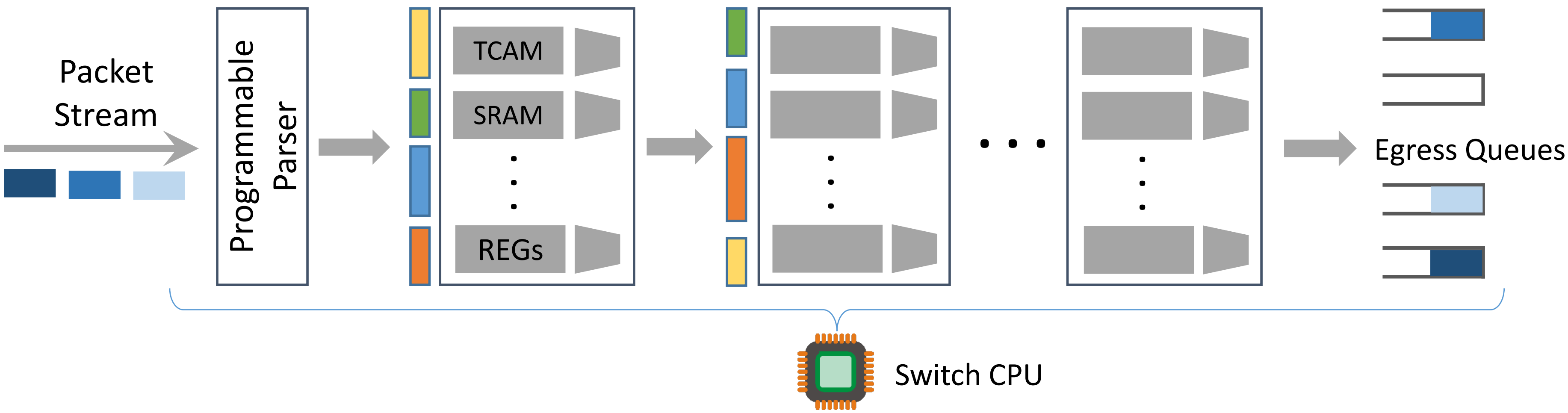
Fixed function switch

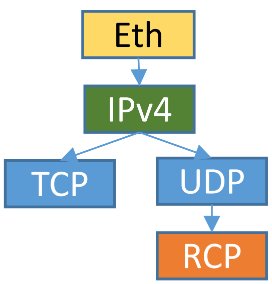


Adding flexibility to a fixed-function switch

- Flexibility to:
  - Trade one memory dimension for another:
    - A narrower ACL table with more rules
    - A wider MAC address table with fewer rules.
  - Add a new table
    - Tunneling
  - Add a new header field
    - VXLAN
  - Add a different action
    - Compute RTT sums for RCP.
- But, can't do everything: regex, state machines, payload manipulation

Features of Flexible Switches





- TCAM for arbitrary wildcard matches
- SRAM for exact/LPM lookups
- Stateful memory for counter and meters
- ALUs for modifying headers and registers

Match (TCAM, SRAM, stateful memory)

Action (ALUs)

1. p = lookup(eth.dst_mac)
2. pkt.egress_port = p
3. counter [ipv4.src_ip] ++

Flexible Switches are not all-powerful

Processing primitives are limited

- Cannot perform arbitrary operations

Available stateful memory is constrained

- Cannot maintain significant per-flow state

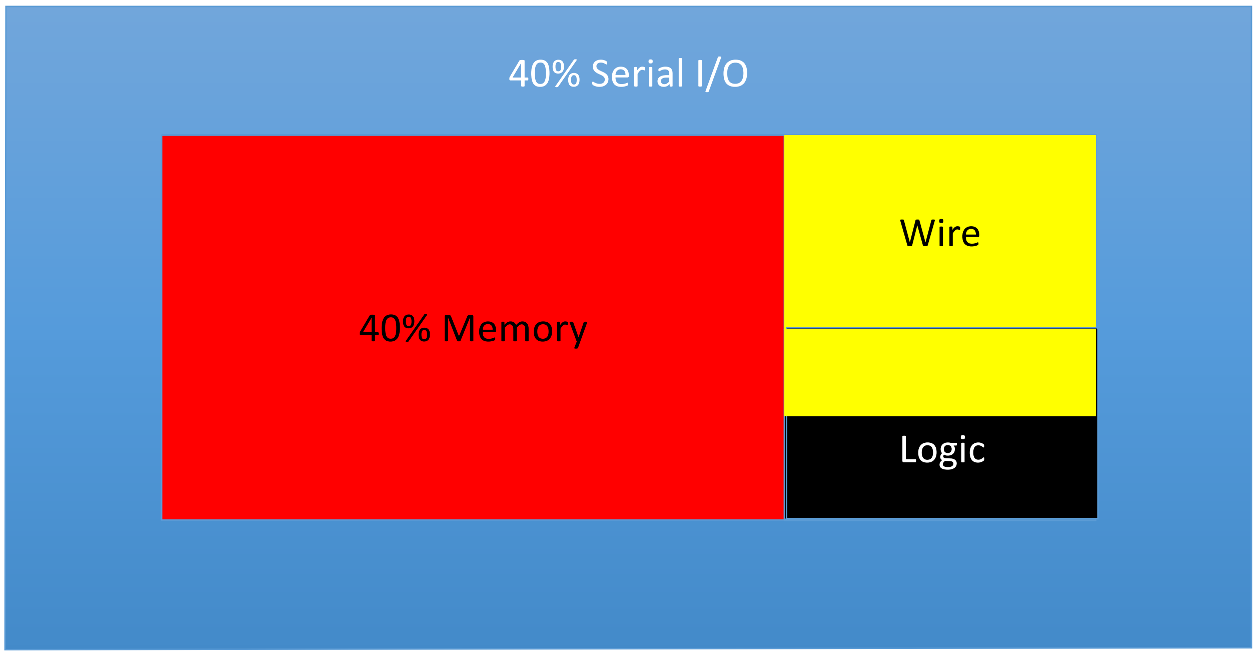
Limited number of stages and limited communication across stages

- Imposes a limit on computation performed per-packet

What can we do with these switches?

- Custom routing and tunneling protocols such as VxLAN or MPLS
- Most are packet-level transformations involving static table lookups

Switch chip area



Programmability mostly affects logic, which is decreasing in area.

Programming RMT: P4

- RMT provides flexibility, but programming it is akin to x86 assembly
- Concurrently, other programmable chips being developed: Intel FlexPipe, Cavium Xpliant, CORSA, …
- Portable language to program these chips