EEE 4121F-B

Routing and Network Management

Routers

What a Router Chassis Looks like

Cisco CRS-1

50cm

1.8m

Capacity: 1.2Tb/s
Power: 10.4kW
Weight: 0.5 Ton
Cost: $100k

60 cm

Juniper M320

42cm

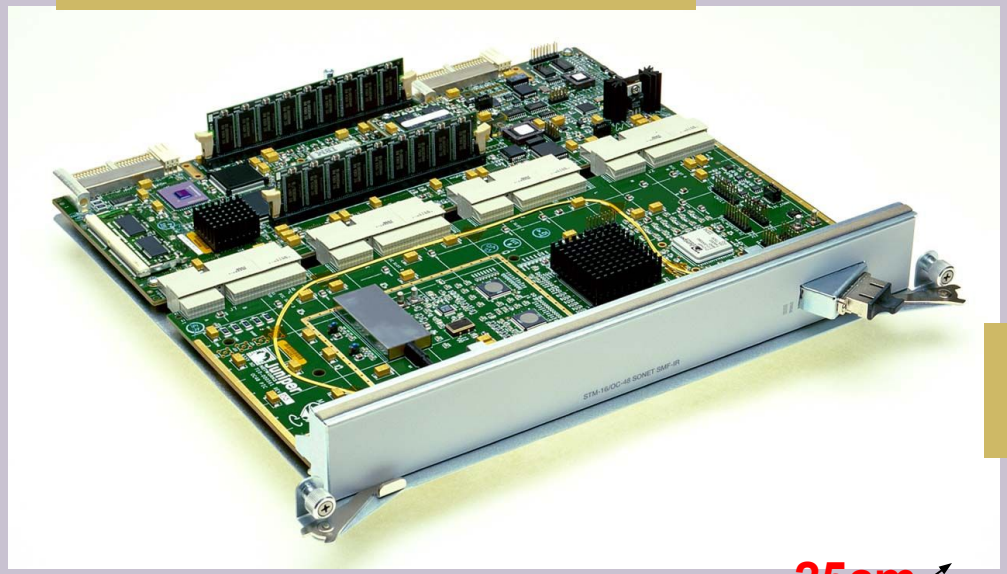
1m

60cm

Capacity: 320 Gb/s
Power: 3.1kW

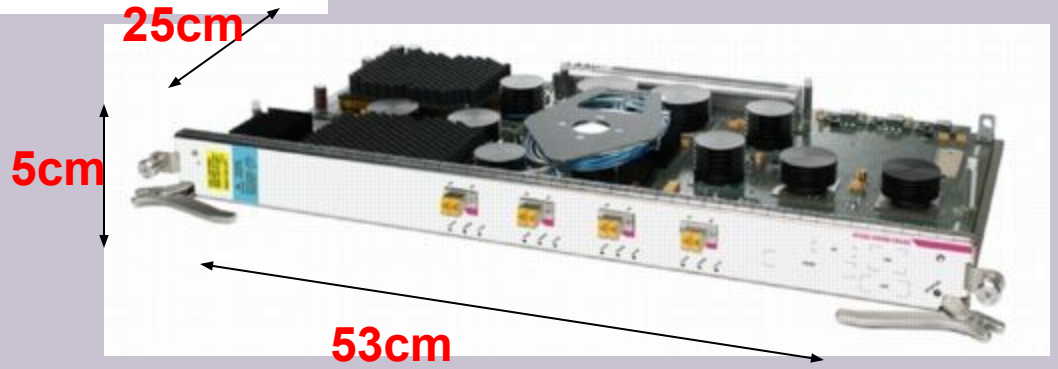
What a Router Line Card Looks Like

1-Port OC48 (2.5 Gb/s)
(for Juniper M40)



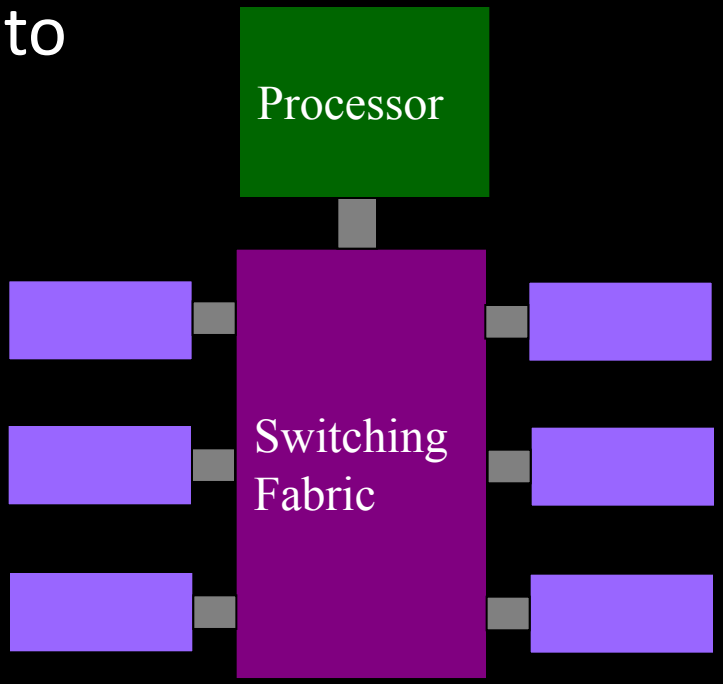
4-Port 10 GigE
(for Cisco CRS-1)

Power: about 150 Watts



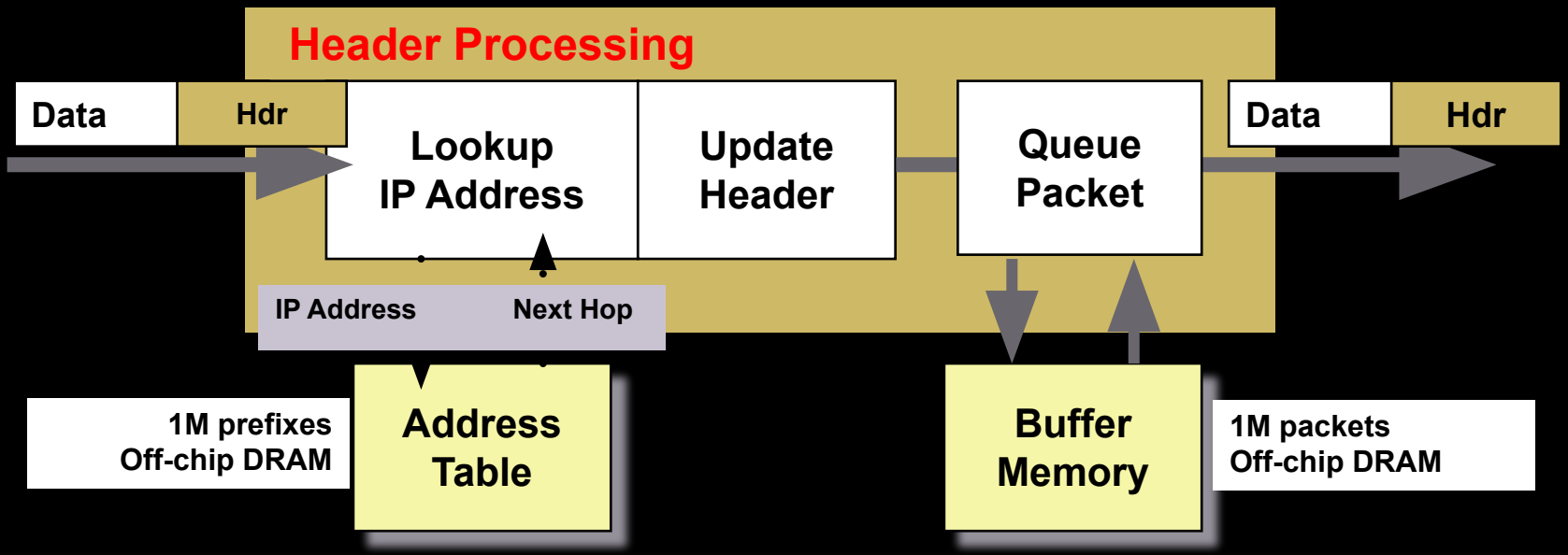
What Routers Do

- **Control plane:** computes a forwarding table
  - Maps destination address(es) to an output link
- **Data plane:** handles an incoming packet
  - Match: destination address
  - Action: direct the packet to the chosen output link
- Switching fabric
  - Directs packet from input link to output link



Life of a Packer at a Router

- Router gets packet
- Looks at packet header for destination
- Looks up forwarding table for output interface
- Modifies header (TTL, IP header checksum)
- Passes packet to appropriate output interface



Routing vs. Forwarding

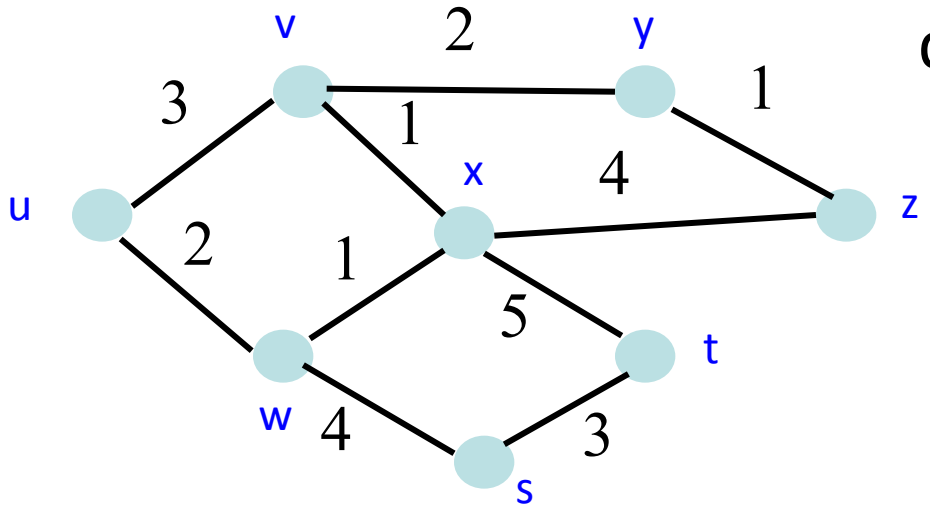
- **Routing:** control plane
  - Computing paths the packets will follow
  - Routers talking amongst themselves
  - Creating the forwarding tables
- **Forwarding:** data plane
  - Directing a data packet to an outgoing link
  - Using the forwarding tables

Routing Protocols

- What does the protocol compute?
  - E.g., shortest paths
- What algorithm does the protocol run?
  - E.g., link-state routing
- How do routers learn end-host locations?
  - E.g., injecting into the routing protocol

Distance Vector: Bellman-Ford

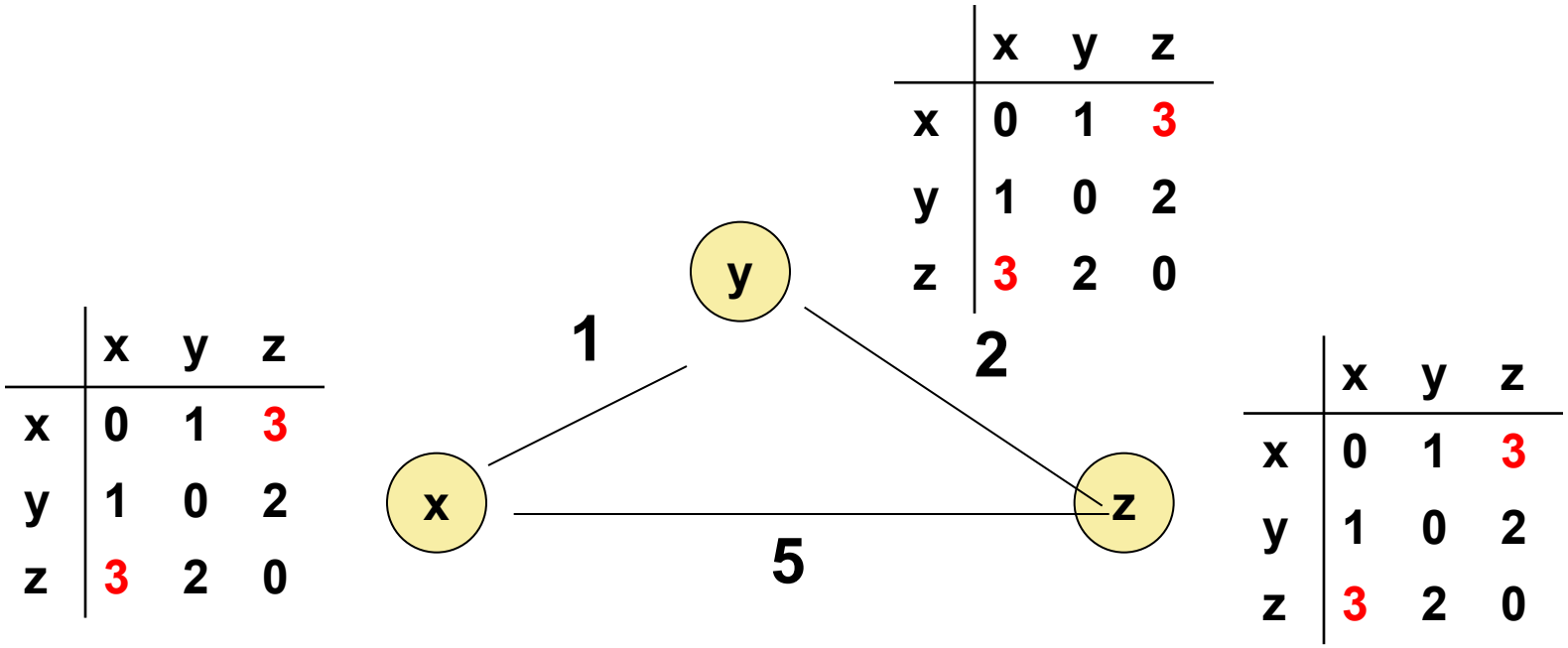
- Define distances at each node x
  - $d_x(y)$ = cost of least-cost path from x to y
- Update distances based on neighbors
  - $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors v



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

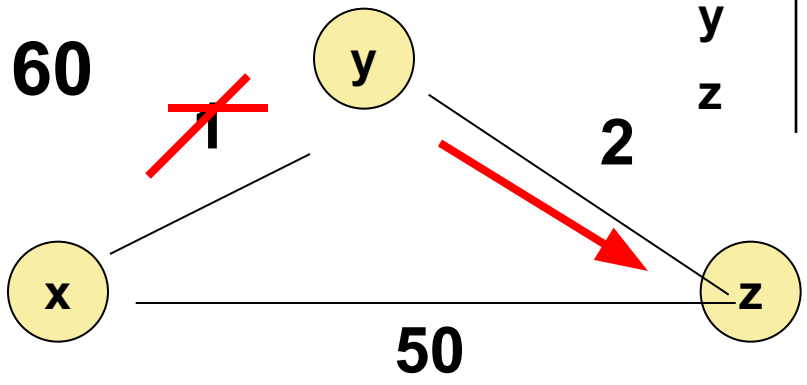
Used in RIP

Good News Travels Quickly



- When costs decrease, network converges quickly

Bad News Travels Slowly



	x	y	z
x	0	60	50
y	5	0	2
z	3	2	0

	x	y	z
x	0	60	50
y	5	0	2
z	7	2	0

Note also that there is a forwarding loop between *y* and *z*.

Link State: Dijkstra's Algorithm

- Flood the topology information to all nodes
- Each node computes shortest paths to other nodes

Initialization

```
S = {u}
for all nodes v
   if (v is adjacent to u)
      D(v) = c(u,v)
   else D(v) = ∞
```

Loop

```
add w with smallest D(w) to S
update D(v) for all adjacent v:
    D(v) = min{D(v), D(w) + c(w,v)}
until all nodes are in S
```

Used in OSPF and IS-IS

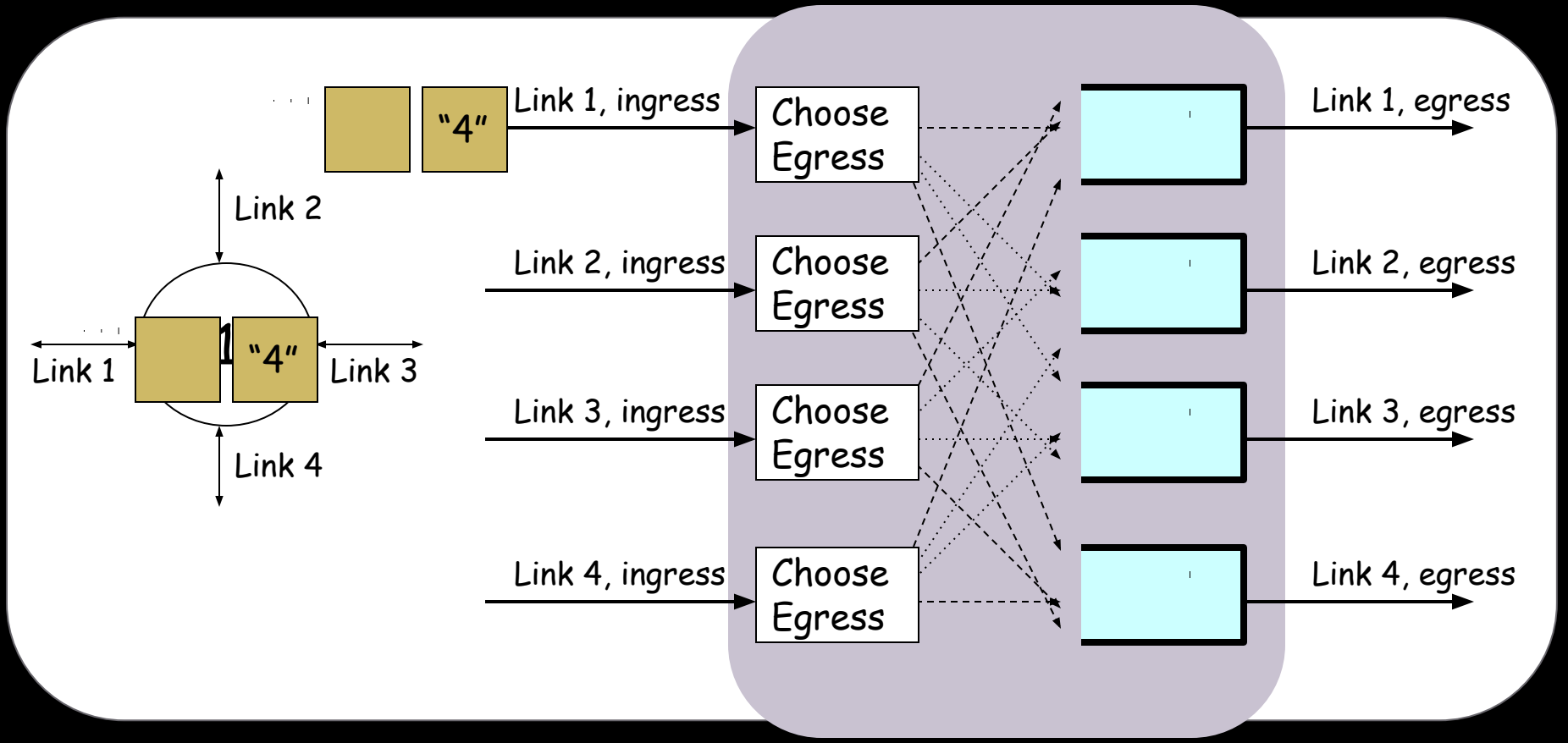
Link-State vs. Distance-Vector

- **Convergence**
  - DV has count-to-infinity
  - DV often converges slowly (minutes)
- **Robustness**
  - Route calculations a bit more robust under link-state.
  - DV algorithms can advertise incorrect least-cost paths
- **Bandwidth Consumption** for Messages
- **Computation**

Routers

Queue Management

Packet Switching and Forwarding

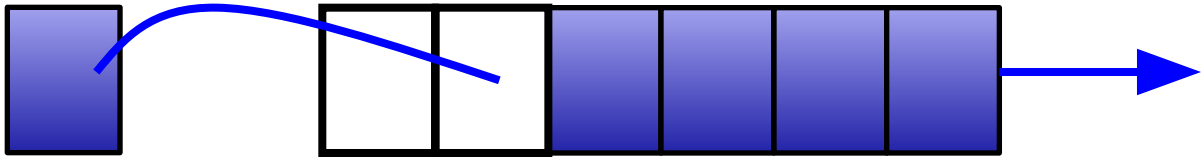


Queue Management Issues

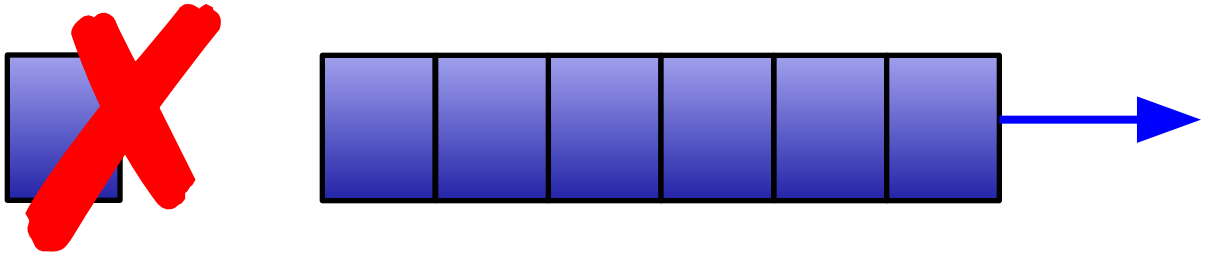
- Scheduling discipline
  - Which packet to send?
  - Some notion of fairness? Priority?
- Drop policy
  - When should you discard a packet?
  - Which packet to discard?
- Goal: balance throughput and delay
  - Huge buffers minimize drops, but add to queuing delay (thus higher RTT, longer slow start, …)

FIFO Scheduling and Drop-Tail

- Access to the bandwidth: first-in first-out queue
  - Packets only differentiated when they arrive

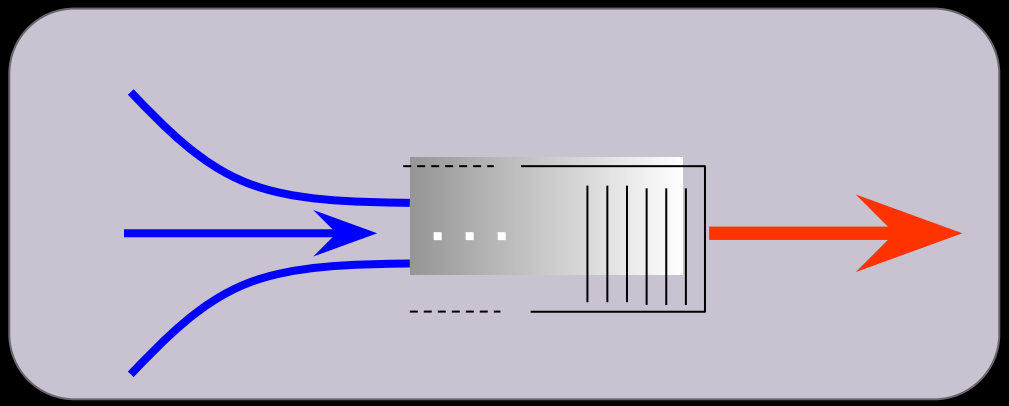


- Access to the buffer space: drop-tail queuing
  - If the queue is full, drop the incoming packet



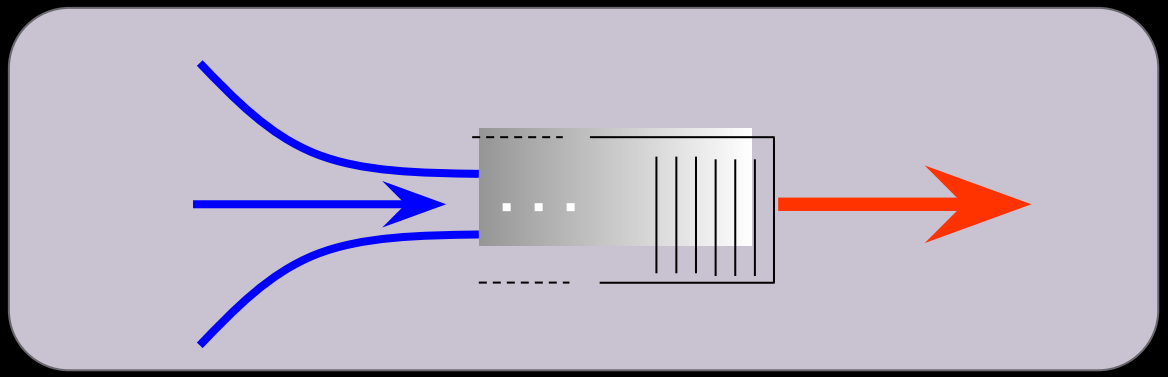
Bursty Loss From Drop-Tail Queuing

- TCP depends on packet loss
  - Packet loss is indication of congestion
  - TCP additive increase drives network into loss
- Drop-tail leads to *bursty* loss
  - *Congested link*: many packets encounter full queue
  - *Synchronization*: many connections lose packets at once



Slow Feedback from Drop Tail

- Feedback comes when buffer is completely full
  - … even though the buffer has been filling for a while
- Plus, the filling buffer is increasing RTT
  - … making detection even slower
- Better to give early feedback
  - Get 1-2 connections to slow down before it's too late!

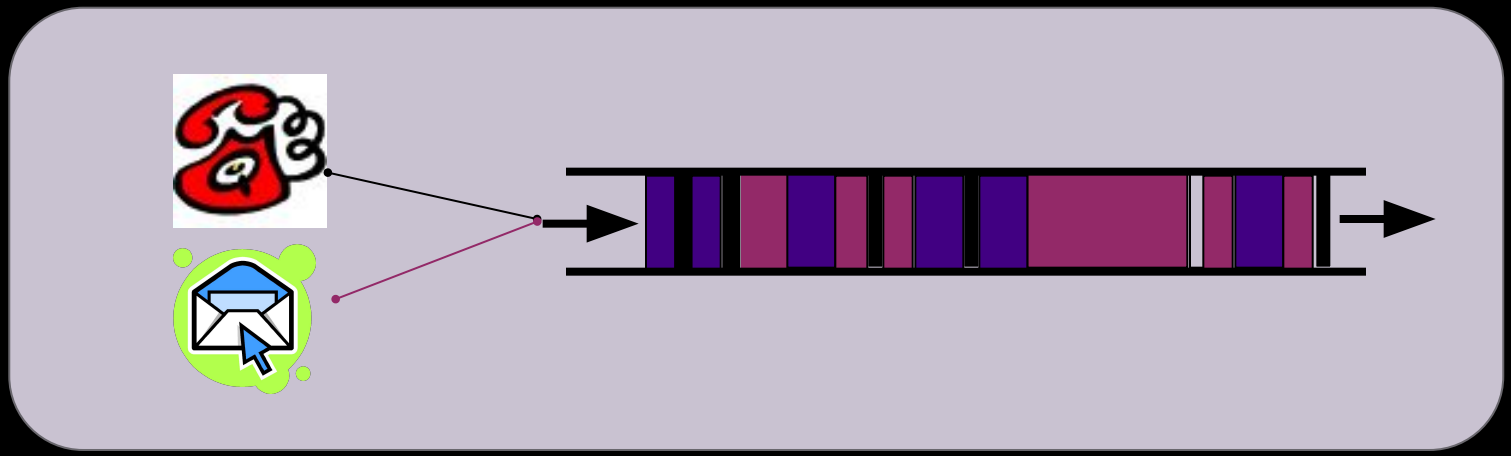


Routers

Link Scheduling

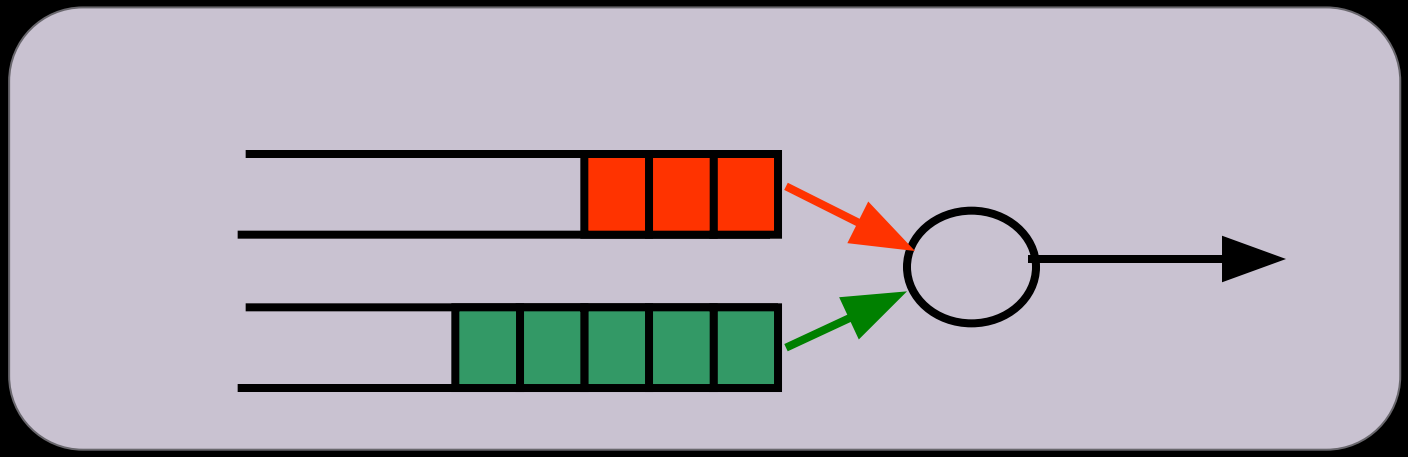
First-In First-Out Scheduling

- First-in first-out scheduling
  - Simple, but restrictive
- Example: two kinds of traffic
  - Voice over IP needs low delay
  - E-mail is not that sensitive about delay
- Voice traffic waits behind e-mail



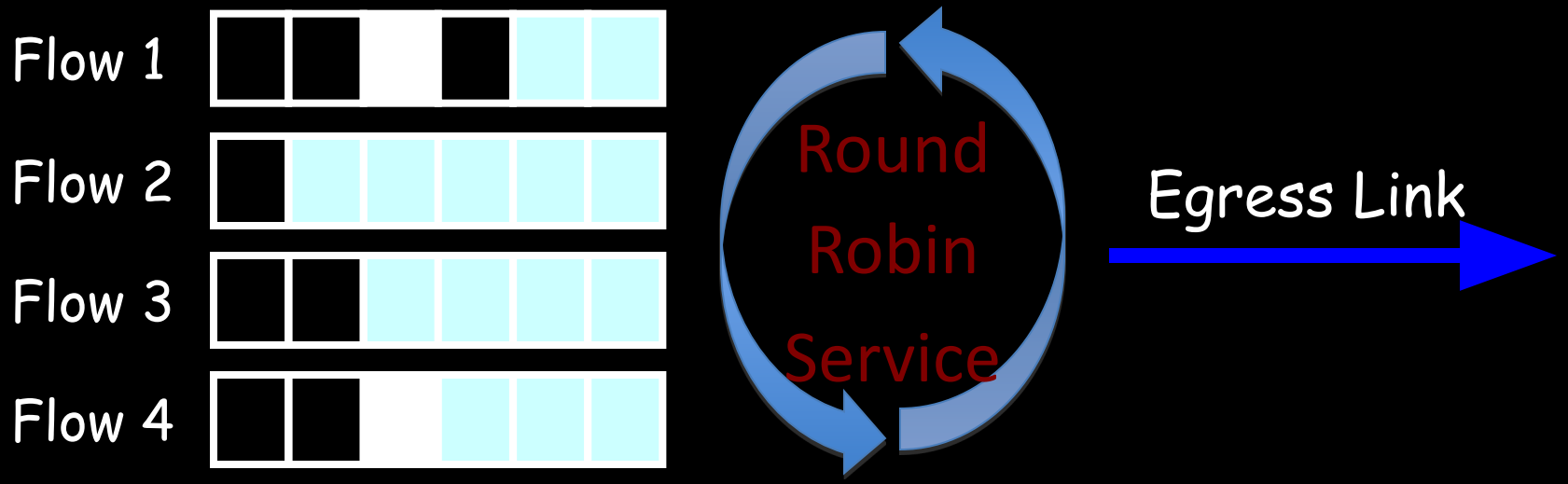
Strict Priority

- Multiple levels of priority
  - Always transmit high-priority traffic, when present
- Isolation for the high-priority traffic
  - Almost like it has a dedicated link
  - Except for (small) delay for packet transmission
- But, lower priority traffic may starve ☹



Fair Queuing (FQ)

- Maintains separate queue per flow
- Ensures no flow consumes more than its 1/n share
  - If all packets were same length, would be easy
- If ***non-work-conserving*** (resources can go idle), also would be easy, yet lower utilization



Fair Queuing (FQ)

- Fair Queuing
  - To understand the algorithm for approximating bit-by-bit round robin, consider the behavior of a single flow
  - For this flow, let
    - P_i denote the length of packet i
    - S_i the time when the router starts to transmit packet i
    - F_i the time when router finishes transmitting packet i
    - Clearly $F_i = S_i + P_i$

Fair Queuing (FQ)

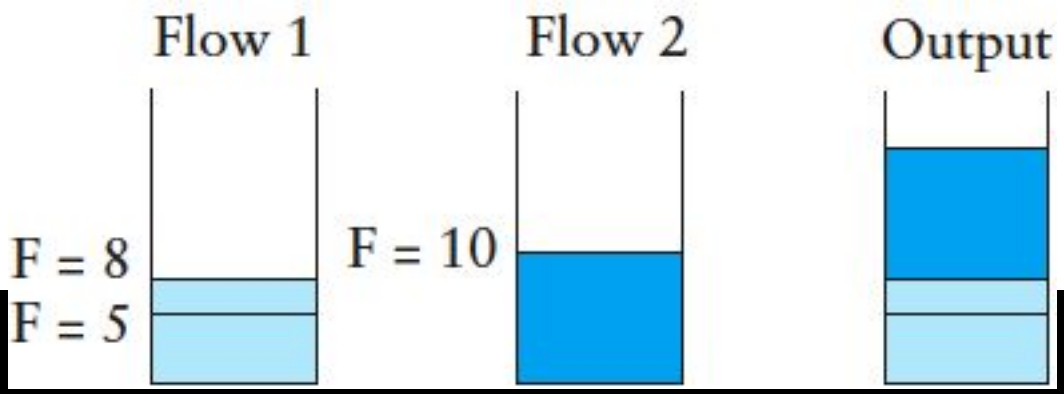
- Fair Queuing
  - When do we start transmitting packet i?
    - Depends on whether packet i arrived before or after the router finishes transmitting packet i-1 for the flow
  - Let A_i denote the time that packet i arrives at the router
  - Then $S_i = max(F_{i-1}, A_i)$
  - $F_i = S_i + P_i$

FQ Algorithm

- Imagine clock ticks per bit, then tx time ~ length

Finish time $F_i = max(F_{i-1}, A_i) + P_i$

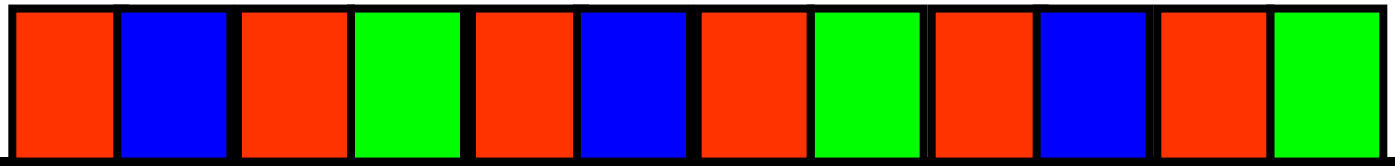
- Calculate estimated F_i for all queued packets
- Transmit packet with lowest F_i next



Weighted Fair Queueing (WFQ)

- Weighted fair scheduling
  - Assign each queue a fraction of the link bandwidth
  - Rotate across queues on a small time scale
  - In general, if a flow has a weight w then the effective packet size is $\frac{P_i}{w}$
  - Hence the final time-stamps are calculated as

$$F_i = max(F_{i-1}, A_i) + \frac{P_i}{w}$$



50% red 25% blue 25% green

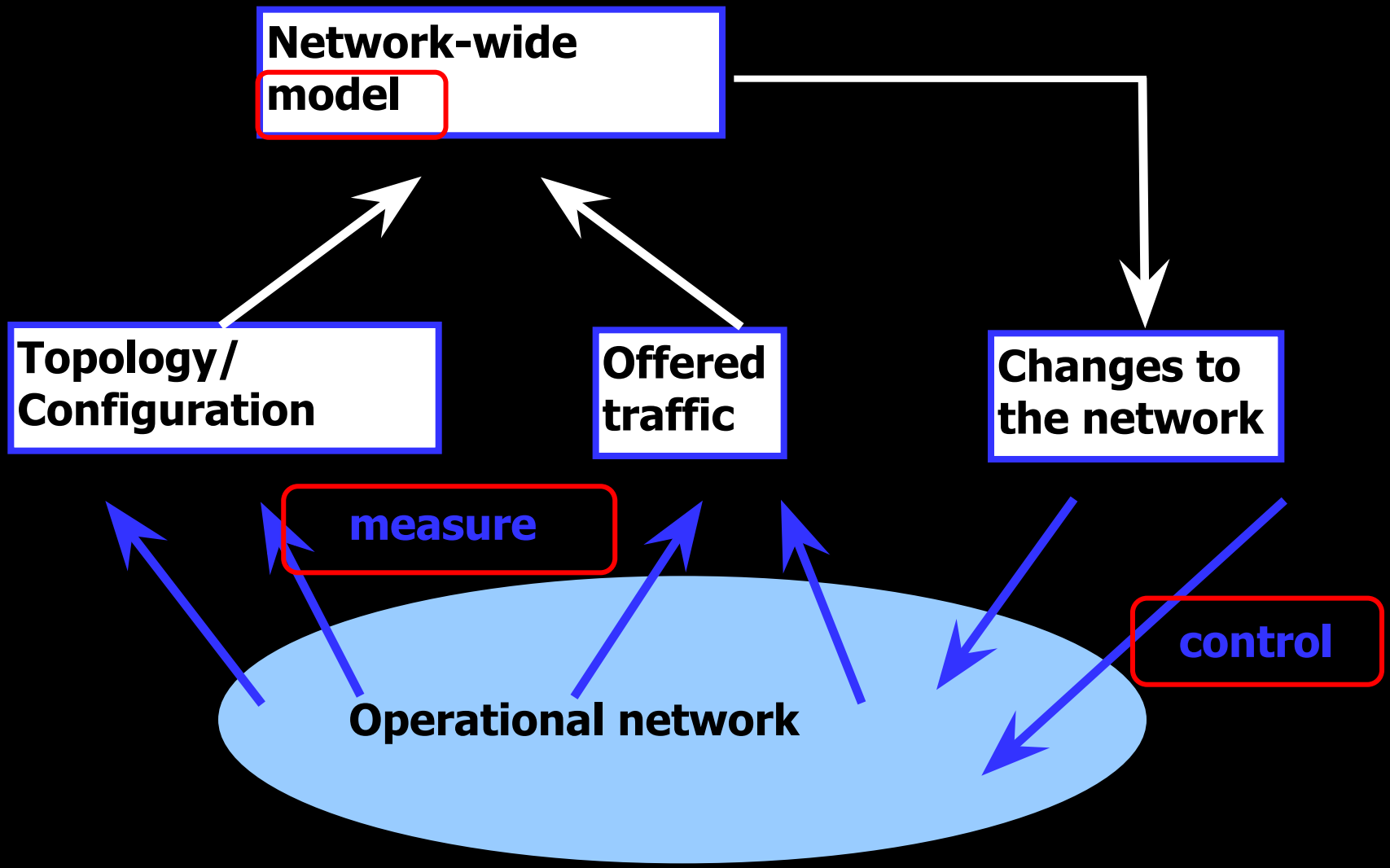
Implementation Trade-Offs

- FIFO
  - One queue, trivial scheduler
- Strict priority
  - One queue per priority level, simple scheduler
- Weighted fair scheduling
  - One queue per class, and more complex scheduler

Network Management

Network Measurement

Measure, Model, and Control



Network Management Key Ingredients

1. Measurement
   - Topology: monitoring of the routing protocols
   - Traffic matrix: passive traffic measurement
2. Network-wide models
   - Representations of topology and traffic
   - models of shortest-path routing
3. Network optimization
   - Efficient algorithms to find good configurations
   - Operational experience to identify constraints

Types of Measurement Data

Active

- traceroute
- ping
- UDP probes
- TCP probes
- Application-level "probes"
  - Web downloads
  - DNS queries

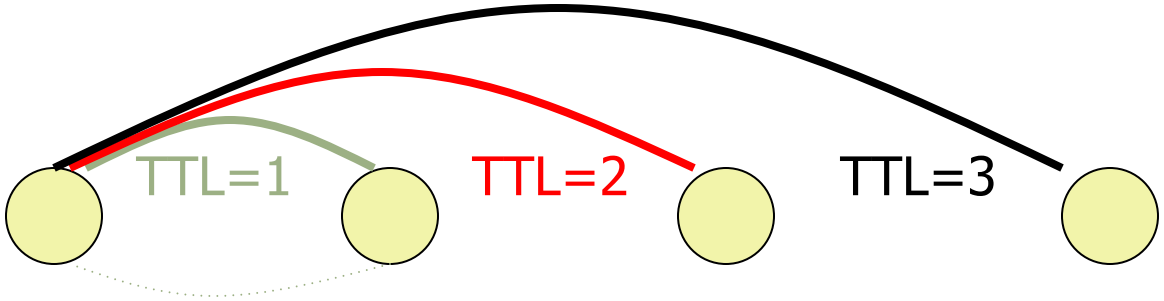
Passive

- Packet traces
  - Complete
  - Headers only
  - Specific protocols
- Flow records
- Specific data
  - Syslogs …
  - HTTP server traces
  - DHCP logs
  - Wireless association logs
  - DNSBL lookups
  - …
- Routing data
  - BGP updates / tables, ISIS, etc.

Active Measurement

How Traceroute Works

- Send packets with increasing TTL values

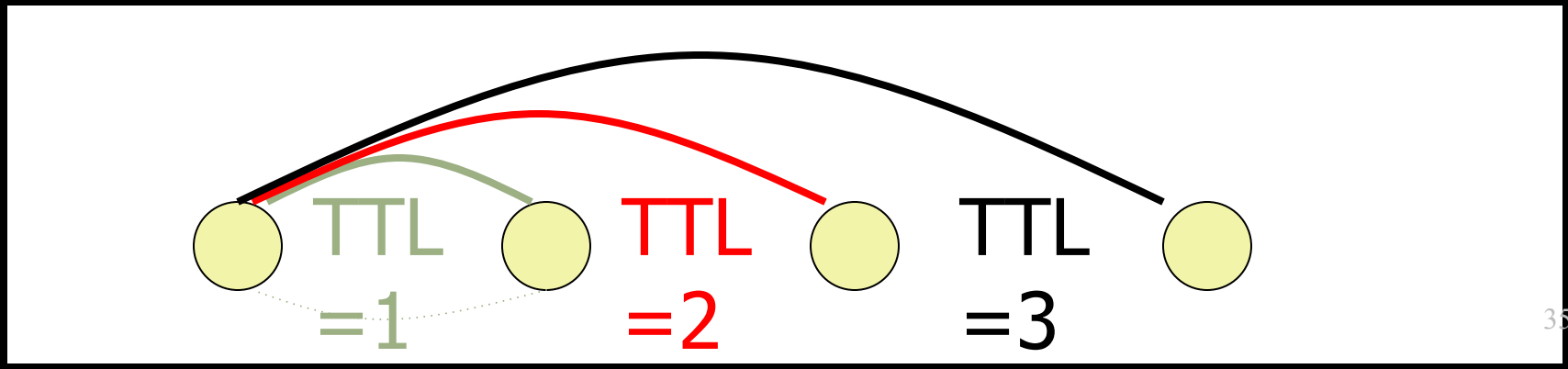


ICMP "time exceeded

- Nodes along IP layer path decrement TTL

Problems with Traceroute

- Can't unambiguously identify one-way outages
  - Failure to reach host : failure of *reverse* path?
- ICMP messages may be filtered or rate-limited
- IP address of "time exceeded" packet may be the *outgoing* interface of the *return* packet



Traceroute Pitfall

- **How would traceroute help with network measurements?**
- **Strawman approach**
  - Run traceroute to destination
  - Collect IP addresses
  - Use “whois” to map IP addresses to AS numbers

- Thought Questions
  - What IP address is used to send “time exceeded” messages from routers?
  - How are interfaces numbered?
  - How accurate is whois data?

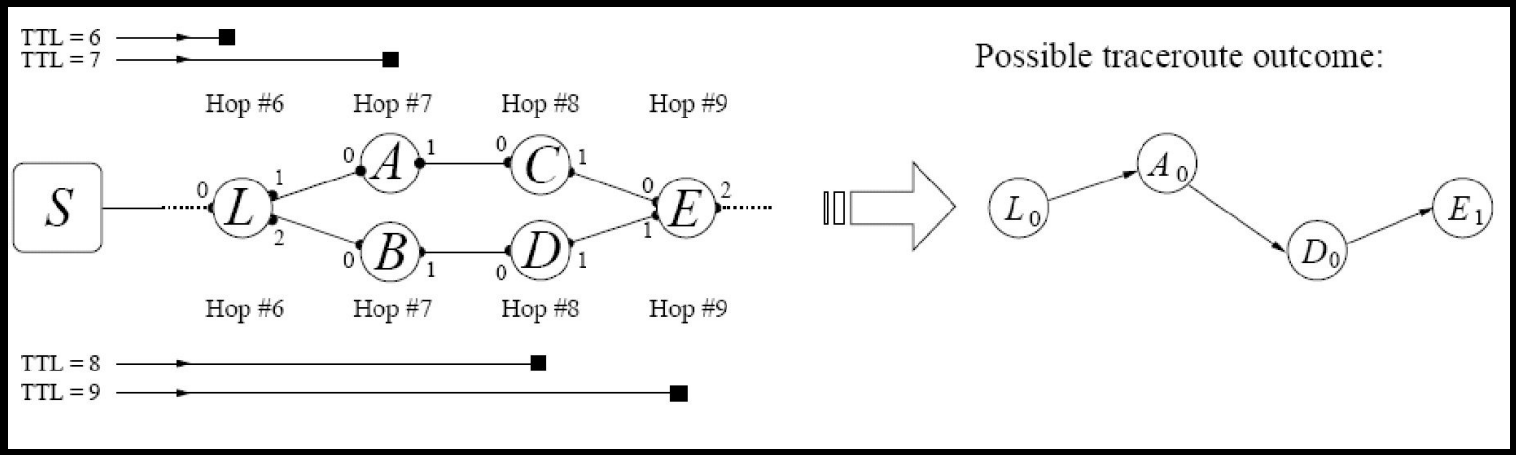
Topology Measurement with Traceroute

- Routers have multiple interfaces
- Measured topology is a function of vantage points

- **Example:** Node degree
  - Must "alias" all interfaces to a single node
  - Is topology a function of vantage point?
    - Each vantage point forms a tree, so clearly each measurement location is going to yield a different view of "the Internet" that does not contain all nodes and edges

How Load Balancing Can Cause Inaccurate Traceroutes

- Suppose the path has a load balancer
  - Load balancers send probes along different paths
    - Per flow load balancing
- Host sends out a sequence of packets
  - Each has a different destination port



Soule *et al.*, "Avoiding Traceroute Anomalies with Paris Traceroute", *IMC 2006*

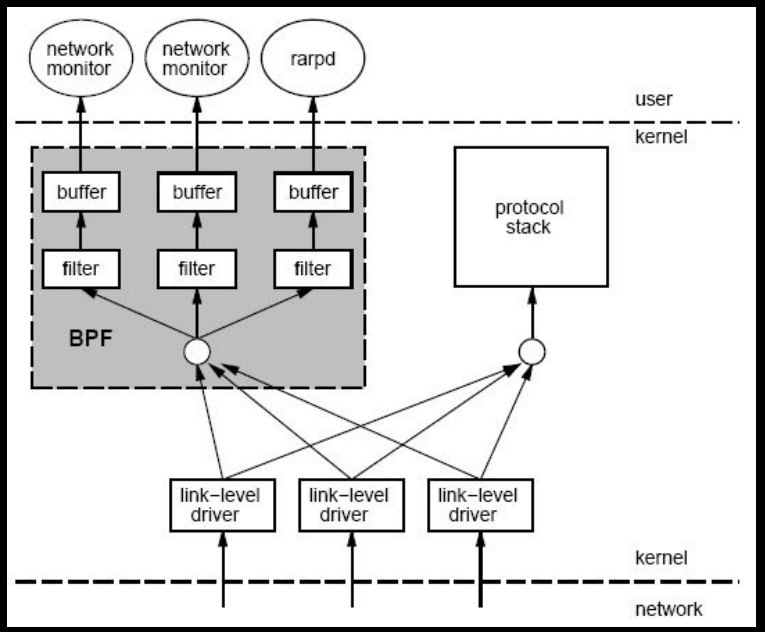
Passive Measurement (Traffic)

Two Main Approaches

- Packet-level Monitoring
  - Keep packet-level statistics
  - Examine (and potentially, log) variety of packet-level statistics. Essentially, anything in the packet.
  - **Timing**
- Flow-level Monitoring
  - Monitor packet-by-packet (though sometimes sampled)
  - Keep aggregate statistics on a flow

Packet Capture: tcpdump/bpf

- Put interface in promiscuous mode
- Use bpf to extract packets of interest



Accuracy Issues

- Packets may be dropped by filter
  - Failure of tcpdump to keep up with filter
  - Failure of filter to keep up with dump speeds

Traffic Flow Statistics

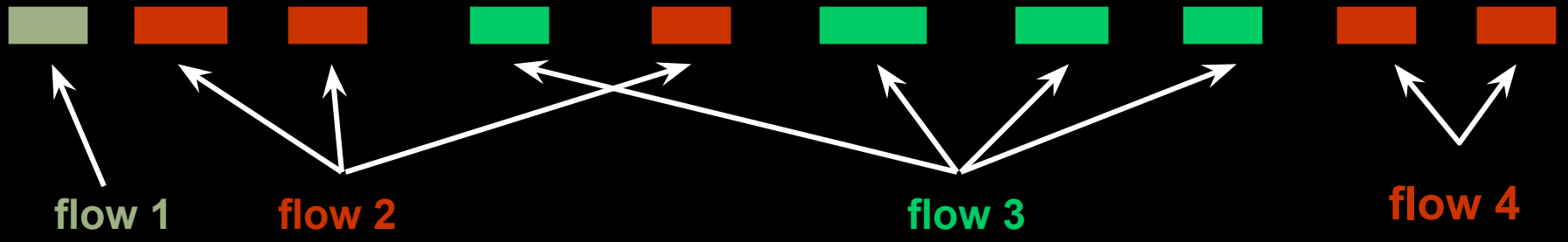
- *Flow monitoring* (*e.g.*, Cisco Netflow)
  - Statistics about groups of related packets (*e.g.,* same IP/TCP headers and close in time)
  - Recording header information, counts, and time

- More detail than SNMP, less overhead than packet capture
  - Typically implemented directly on line card

What is a flow?

- Source IP address
- Destination IP address
- Source port
- Destination port
- Layer 3 protocol type

Aggregating Packets into Flows



- **Criteria 1**: Set of packets that "belong together"
  - Source/destination IP addresses and port numbers
  - Same protocol, ToS bits, …
  - Same input/output interfaces at a router (if known)

- **Criteria 2**: Packets that are "close" together in time
  - Maximum inter-packet spacing (e.g., 15 sec, 30 sec)
  - **Example:** flows 2 and 4 are different flows due to time

Reducing Measurement Overhead

- **Filtering:** on interface
  - destination prefix for a customer
  - port number for an application (e.g., 80 for Web)
- **Sampling:** before insertion into flow cache
  - Random, deterministic, or hash-based sampling
  - 1-out-of-n or stratified based on packet/flow size
  - *Two types:* packet-level and flow-level
- **Aggregation:** after cache eviction
  - packets/flows with same next-hop AS
  - packets/flows destined to a particular service

Problems with Packet Sampling

- Determining size of original flows is tricky
  - For a flow originally of size *n*, the size of the *sampled* flow follows a binomial distribution
  - Extrapolation can result in big errors
  - Much research in reducing such errors
- Flow records can be lost
- Small flows may be eradicated entirely

Network Management

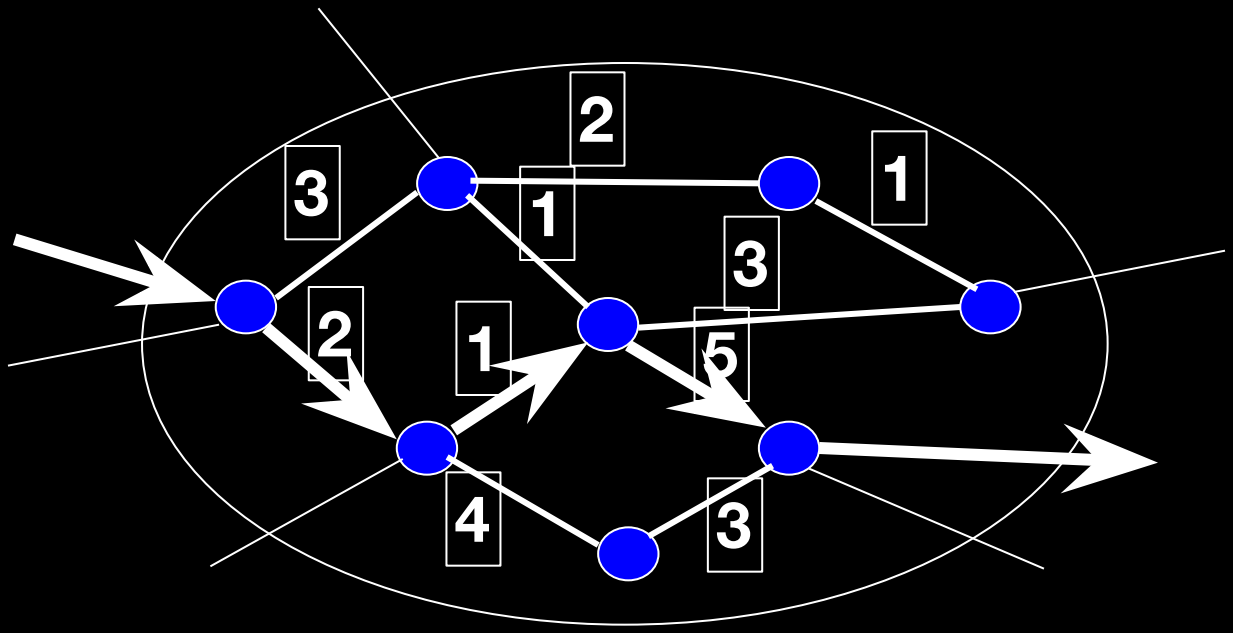
Traffic Engineering

Do IP Networks Manage Themselves?

- In some sense, yes:
  - TCP senders send less traffic during congestion
  - Routing protocols adapt to topology changes
- But, does the network run efficiently?
  - Congested link when idle paths exist
  - High-delay path when a low-delay path exists
- How should routing adapt to the traffic?
  - Avoiding congested links in the network
  - Satisfying application requirements (e.g., delay)
- … essential questions of traffic engineering

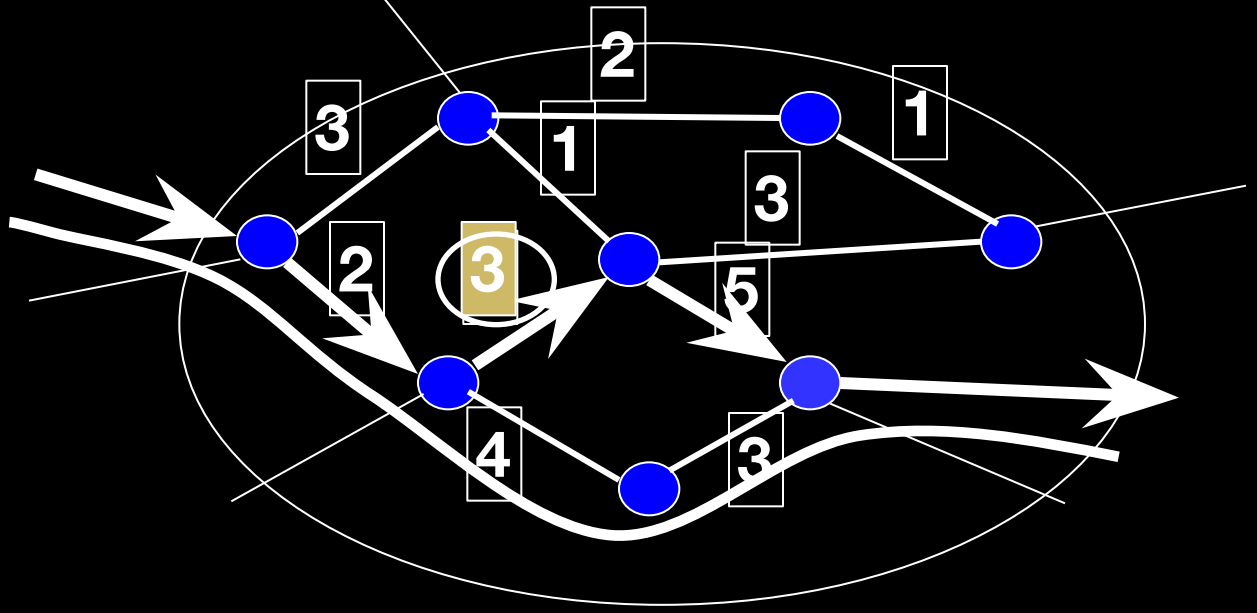
Routing With "Static" Link Weights

- Routers flood information to learn topology
  - Determine "next hop" to reach other routers…
  - Compute shortest paths based on link weights
- Link weights configured by network operator

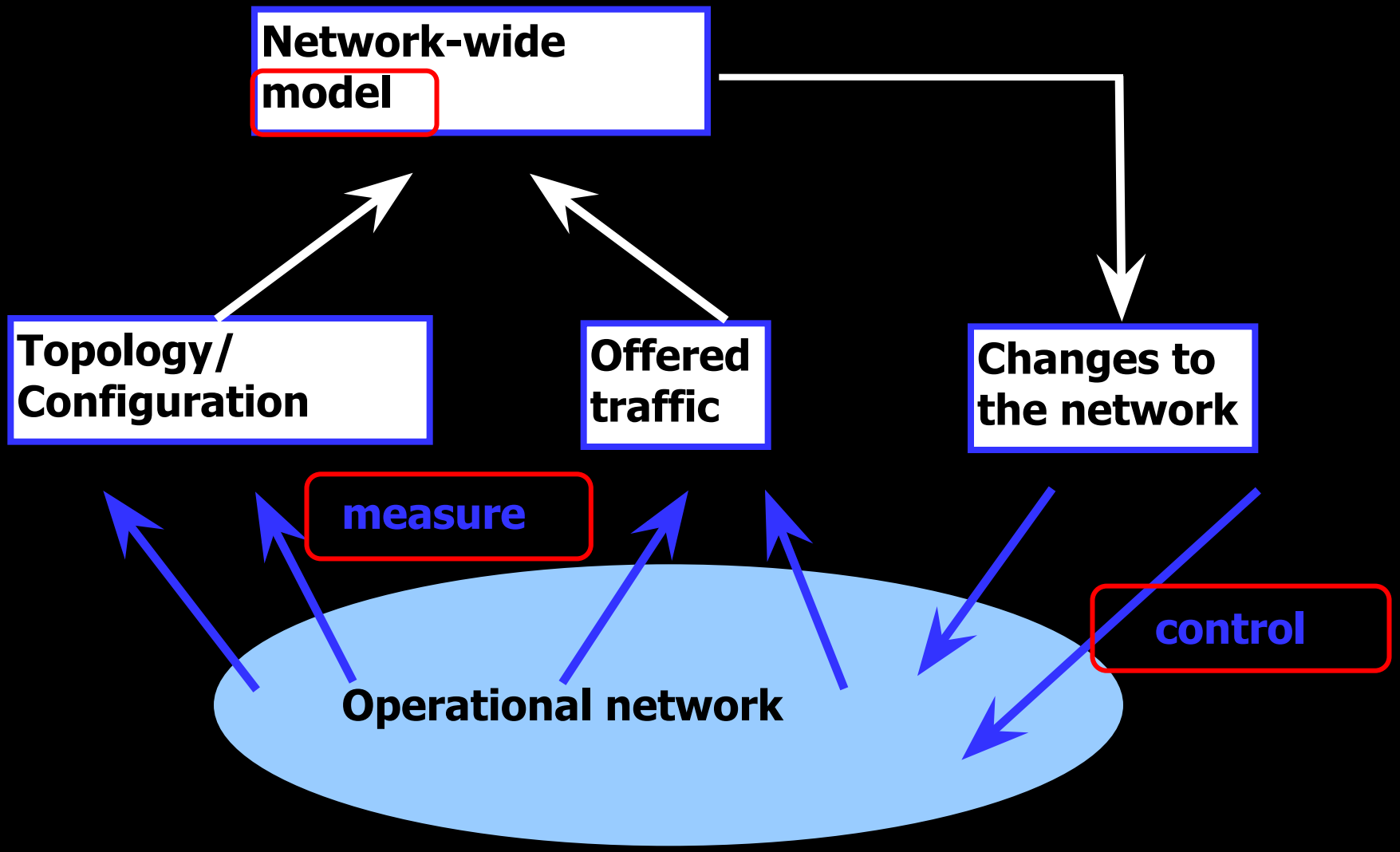


Setting the Link Weights

- How to set the weights
  - Inversely proportional to link capacity?
  - Proportional to propagation delay?
  - Network-wide optimization based on traffic?



Measure, Model, and Control

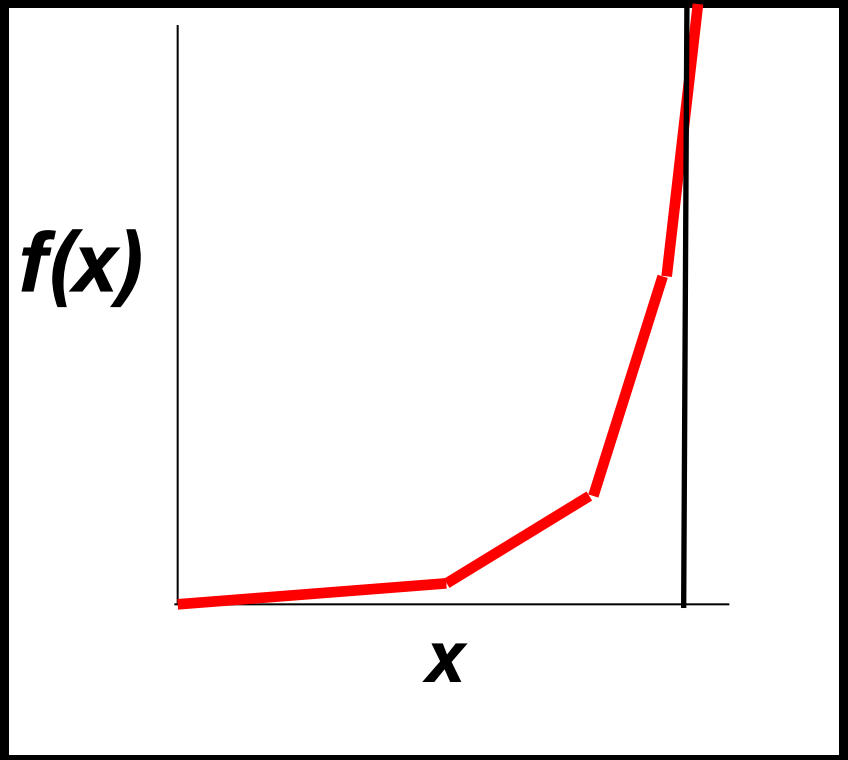


Optimization Problem

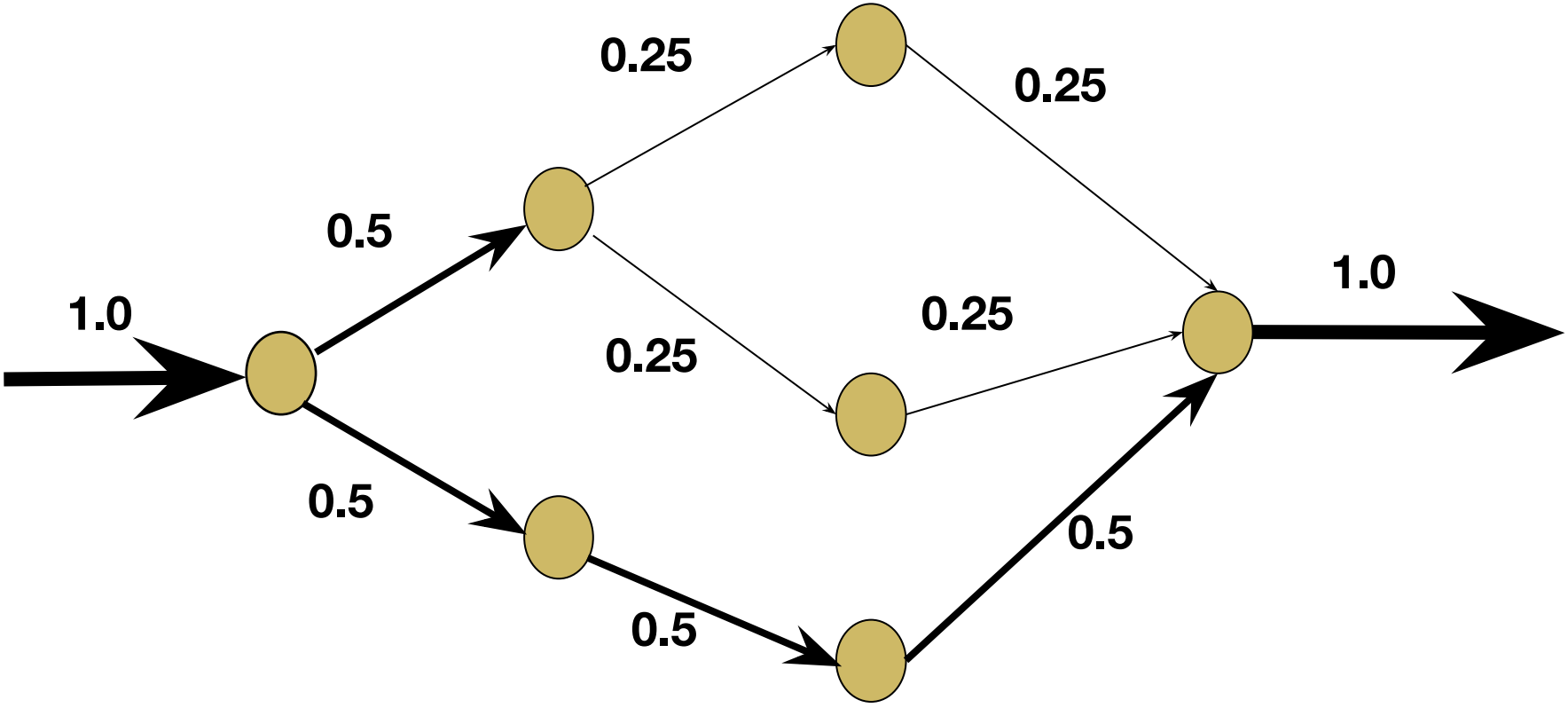
- Input: graph G(R,L)
  - R is the set of routers
  - L is the set of unidirectional links
  - c_l is the capacity of link l
- Input: traffic matrix
  - $M_{i,j}$ is traffic load from router i to j
- Output: setting of the link weights
  - w_l is weight on unidirectional link l
  - $P_{i,j,l}$ is fraction of traffic from i to j traversing link l

Objective Function

- Computing the link utilization
  - Utilization: u_l/c_l
- Objective functions
  - $\min(\max_l(u_l/c_l))$
  - $\min(\sum_l f(u_l/c_l))$



Equal-Cost Multipath (ECMP)



Values of $P_{i,j,l}$

Complexity of Optimization Problem

- NP-complete optimization problem
  - No efficient algorithm to find the link weights
  - Even for simple objective functions
- What are the implications?
  - Have to resort to searching through weight settings
- Clearly suboptimal, but effective in practice
  - Fast computation of the link weights
  - Good performance, compared to “optimal” solution

Challenges

- Protocol dynamics
  - Stability: avoid overreacting to congestion
  - Convergence time: avoid under-reacting to congestion
  - Analysis using control or optimization theory!
- Protocol overhead
  - Bandwidth overhead of disseminating link metrics
  - Computation overhead of recomputing traffic splits
  - Implementing non-equal traffic splitting
  - Hash-based splitting to prevent packet reordering
- Applying the approach in an interdomain setting

Conclusion: Main Issues

- How are the paths expressed?
  - Shortest-path routing with (changing) link weights
  - End-to-end path (or paths) through the network
- Timescale of routing decisions?
  - Packet, flow, larger aggregates, longer timescale, …
- Role of path diversity?
  - Single-path routing where the one path can be changed
  - Multi-path routing where splitting over paths can change
- Who adapts the routes?
  - Routers: through adaptive routing protocols
  - Management system: through central (re)optimization

What types of problems do bad configuration cause?

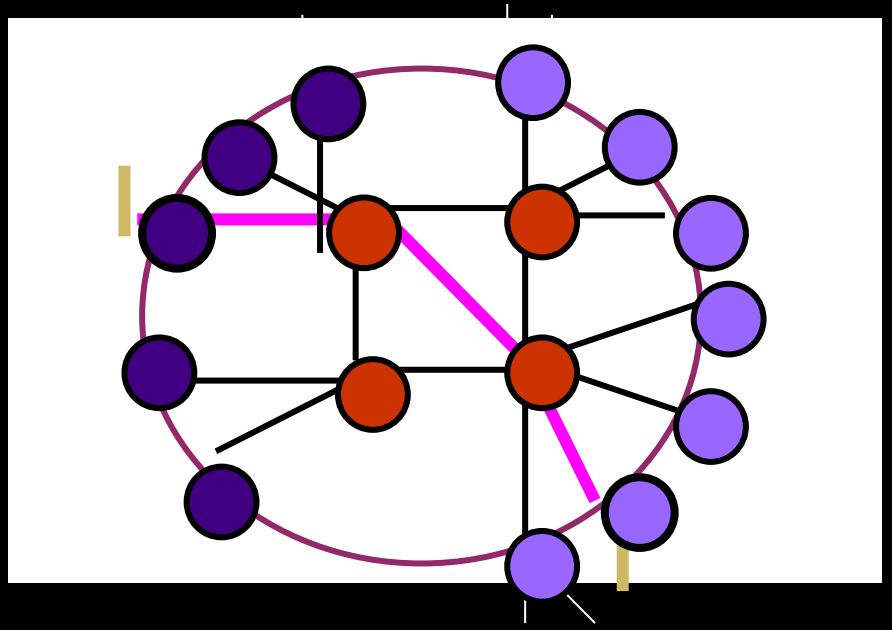
- Persistent oscillation
  - Link weights changing to often
- Forwarding loops
  - Forwarding that runs into an unending loop
- Partitions
  - Destinations that are not reachable from some parts of the network
- “Blackholes”
  - Destinations that are reachable but not known from some parts of the network
- …

Why is configuration hard to get right?

- **Defining correctness is hard**
- **Interactions cause unintended consequences**
  - Each network independently configured
  - Unintended policy interactions
- **Operators make mistakes**
  - Configuration is difficult
  - Complex policies, distributed configuration

What Network Operators Need

- Network-wide views
  - Network topology (e.g., routers, links)
  - Mapping to lower-level equipment
  - Traffic matrix
- Network-level objectives
  - Load balancing
  - Survivability
  - Reachability
  - Security
- Direct control
  - Explicit configuration of data-plane mechanisms



What Should Routers Do?

- Forward packets: yes
  - Must be done at high speed
  - … in line-card hardware on fast routers
  - So, needs to be done on the routers
- Collect measurement data: yes
  - Traffic statistics
  - Topology information
- Compute routes
  - Distributed computation of forwarding tables
  - Doesn't inherently need to run on the routers

SOFTWARE-DEFINED NETWORKING: OVERVIEW AND BRIEF HISTORY

Roots of SDN: Make Network Management Easier by "Removing Routing From Routers"

- Routing is hard to do in a distributed fashion
  - Beyond single-path and/or shortest-path routing
- Difficult to make load-sensitive routing stable
  - Over-reacting to out-of-date information
- Poor visibility to drive good decisions
  - Incomplete local views of topology and load
- Not flexible enough for end users
  - Cannot easily select customized routes
- Difficult to extend over time
  - Hard-coded into the underlying routers

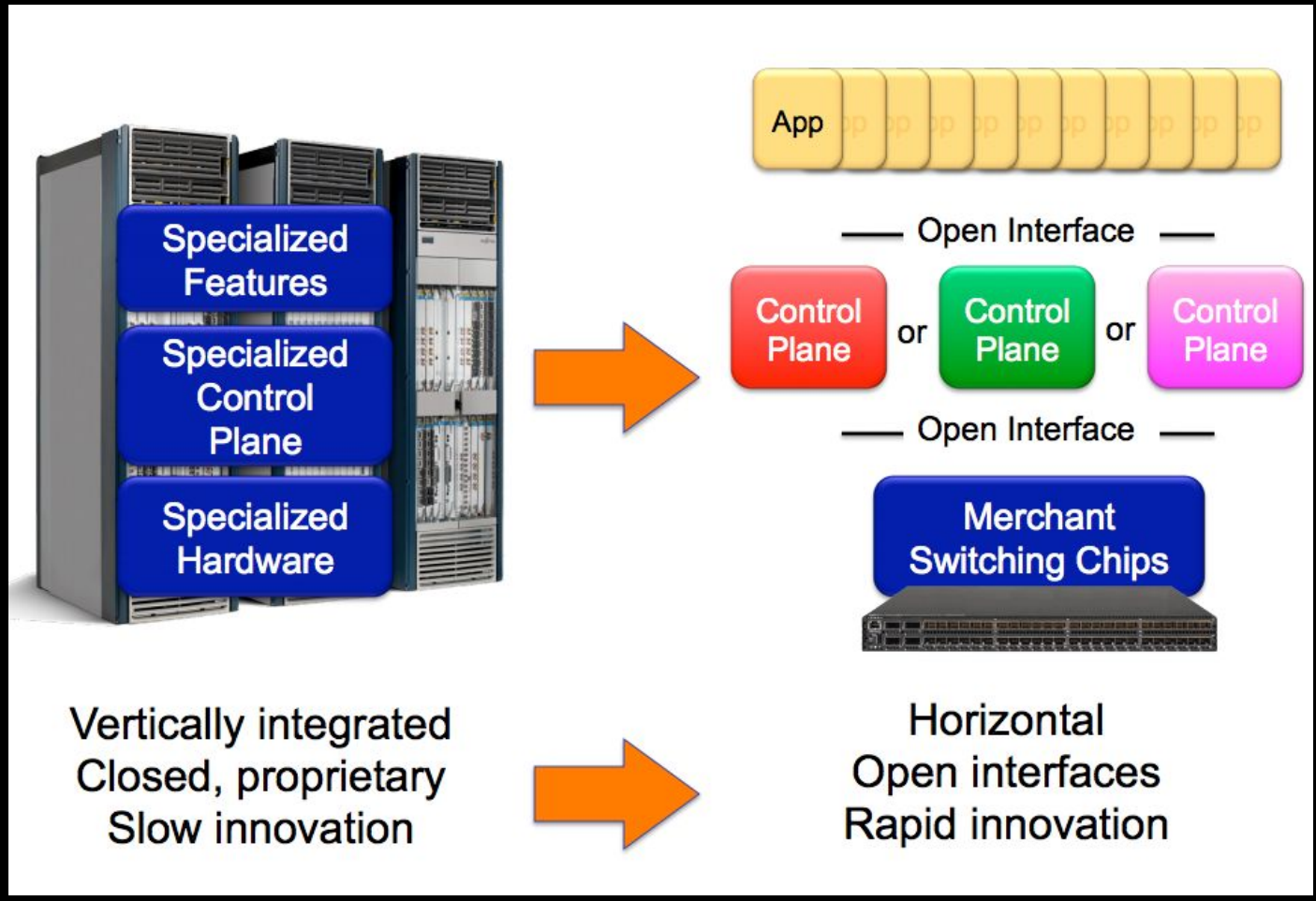
What is a Software-Defined Network?

- Separate data plane and control plane
- Control network behavior from single high-level control program
- Two parts to the infrastructure
  - Data plane: programmable switches
  - Control plane: controllers, apps
- Deployments to solve network management problems in real networks

SDN Infrastructure: Two Parts

- Software Control Plane
  - The network's "brain"
  - Can be run separately from devices
  - Computes logic of how traffic will be forwarded
- Programmable Data Plane
  - Typically programmable hardware
  - Controlled by the control plane

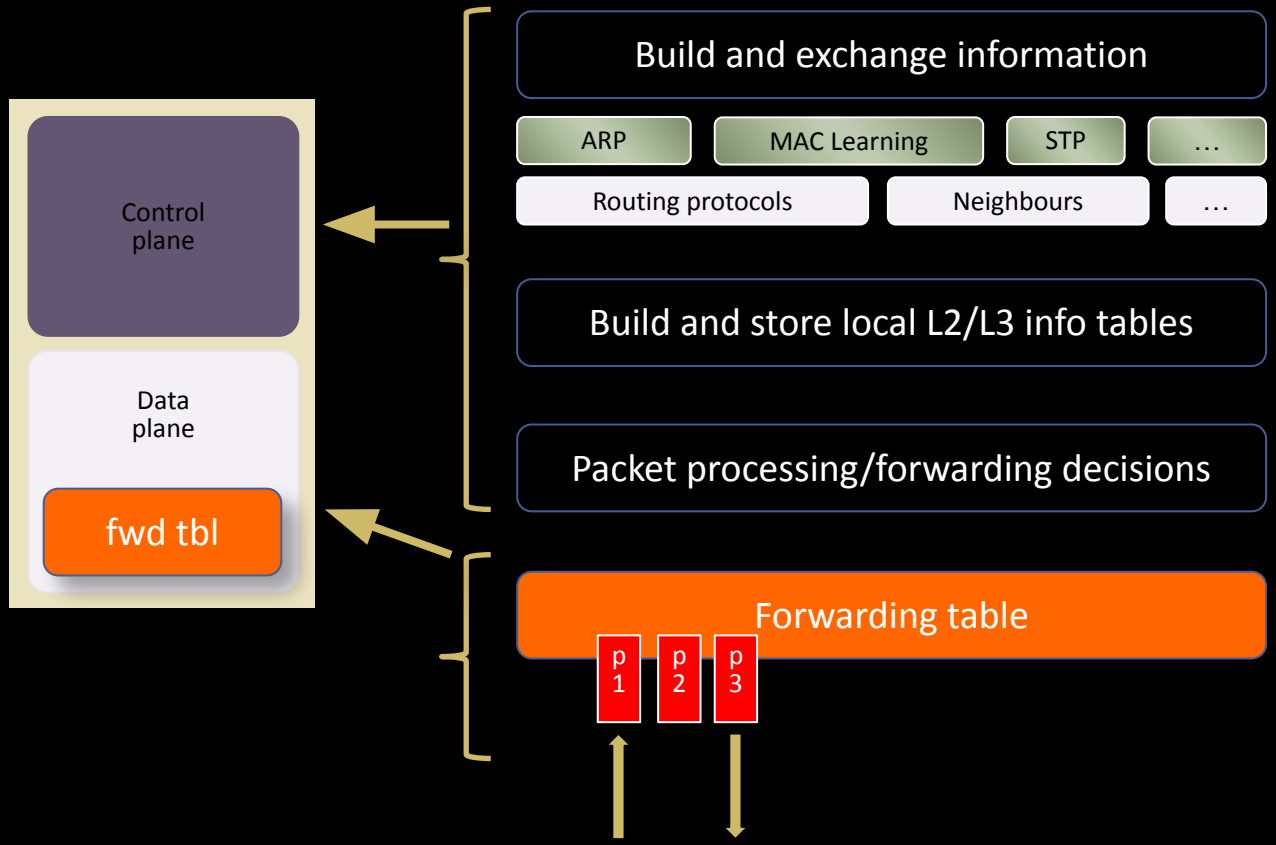
Software Defined Networking



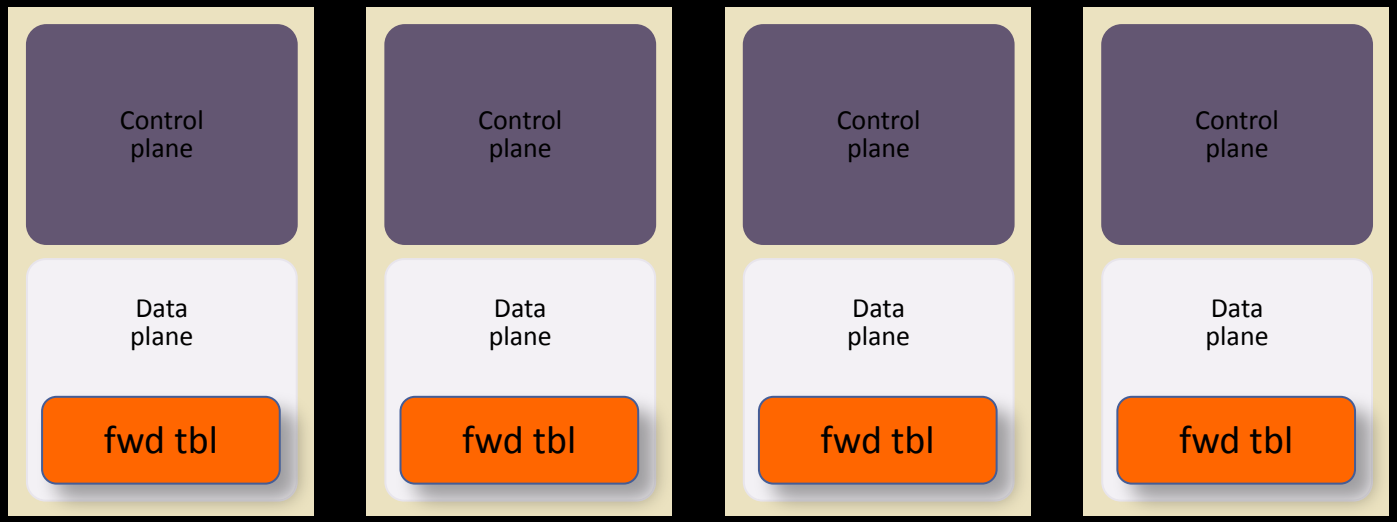
Why Separate the Control and Data Planes?

- Independent evolution and development
  - The software control of the network can evolve independently of the hardware.
- Control from high-level software program
  - Control behavior using higher-order programs
  - Debug/check behavior more easily

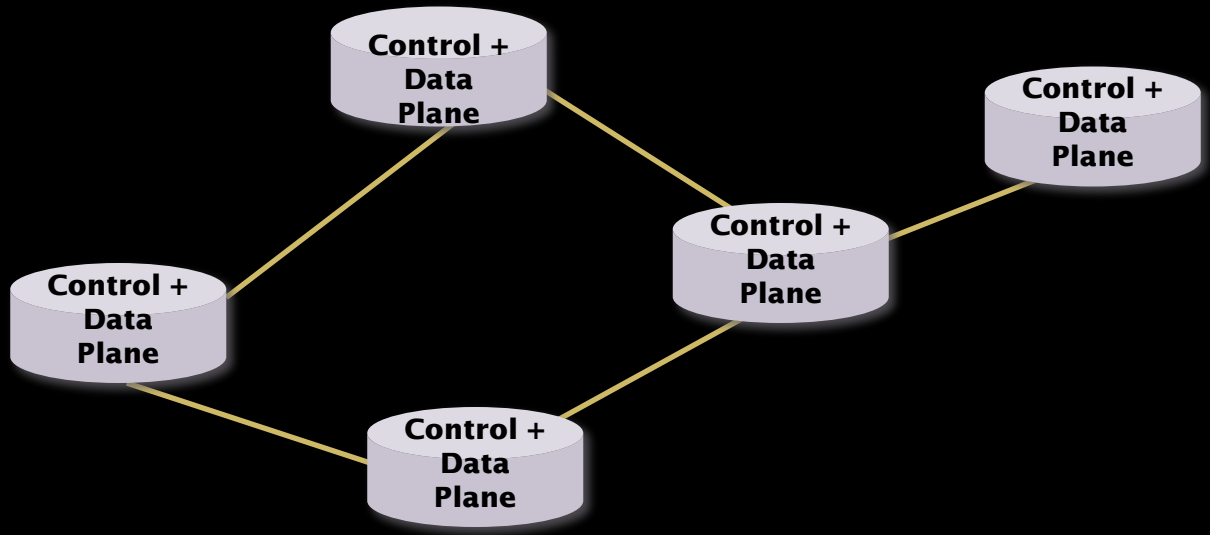
Network Devices and Planes



Network Devices and Planes

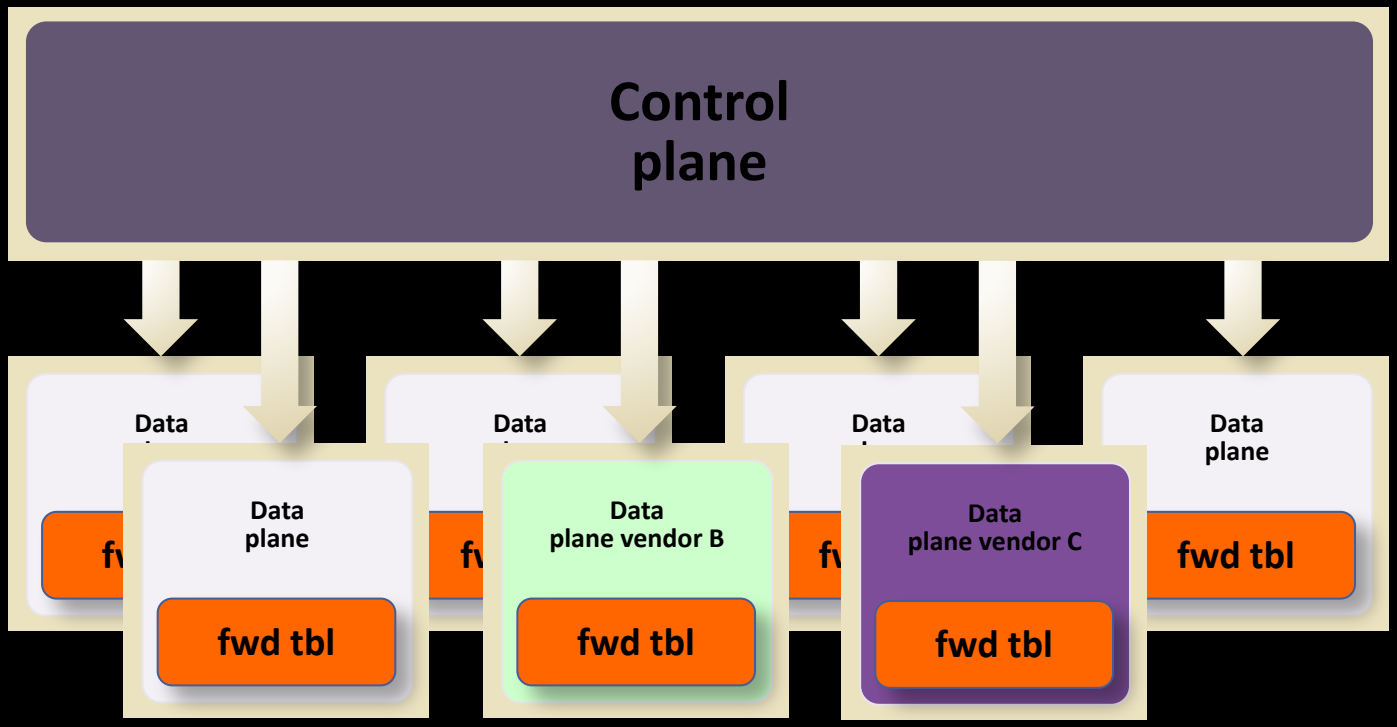


Network Devices and Planes

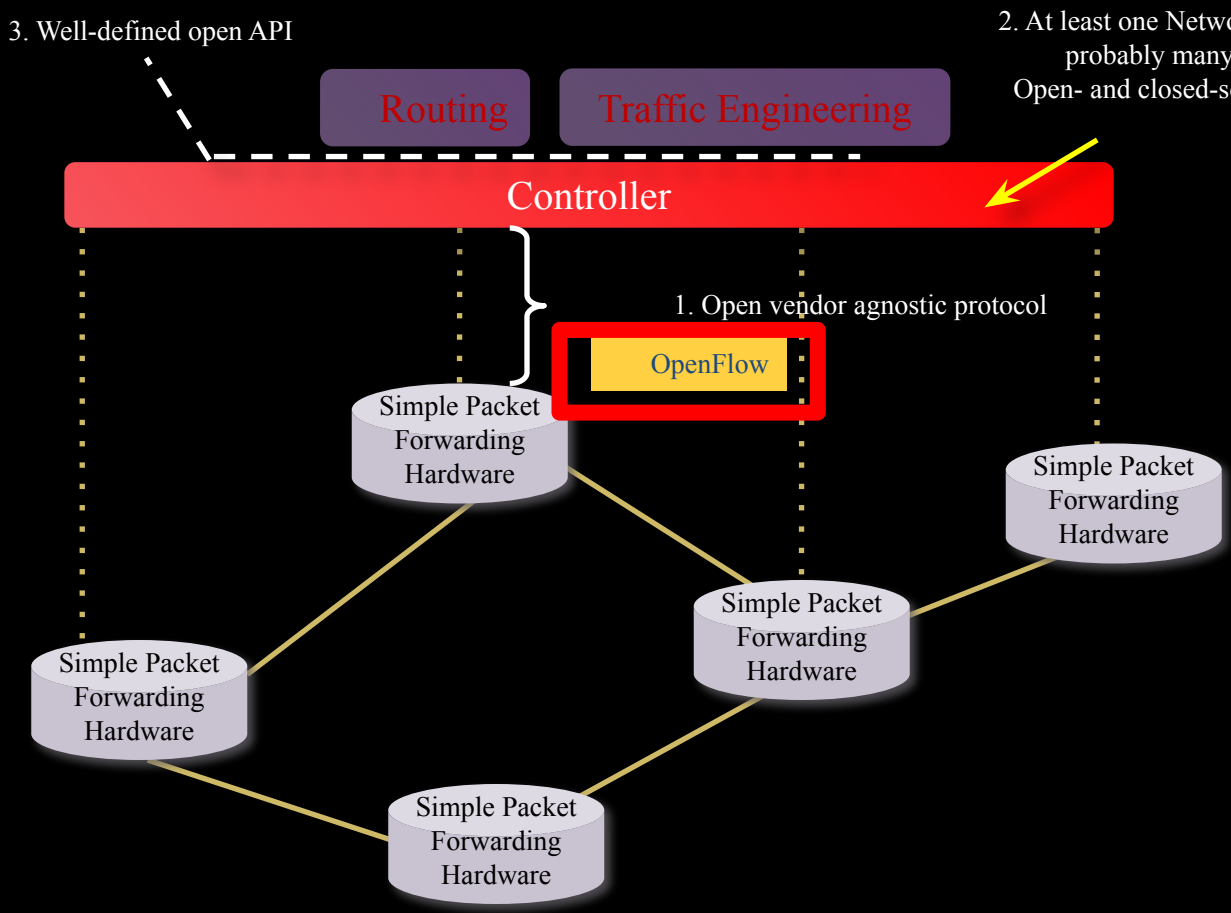


SDN Network Devices and Planes

SDN Network Devices and Planes



OpenFlow/Software-Defined Network



Components of OpenFlow based SDN

- The SDN switch (e.g. OpenFlow)
- The SDN controller
- The interfaces
  1. OpenFlow (the southbound interface)
  2. Network application interface (the northbound interface)
- The Network Applications

Conclusions

- Assignment 1 is based on Congestion, Routing and Network Management
- Next week we will cover more on Software Defined-Networking and Lab 1 will be released

References

1. http://web.stanford.edu/class/cs244/
2. https://www.cs.princeton.edu/courses/archive/spring19/cos461/
3. Computer Networking - A top down approach
4. Computer Networks - A systems approach