Quality of Service

Lecture 25

http://www.cs.rutgers.edu/~sn624/352-S22

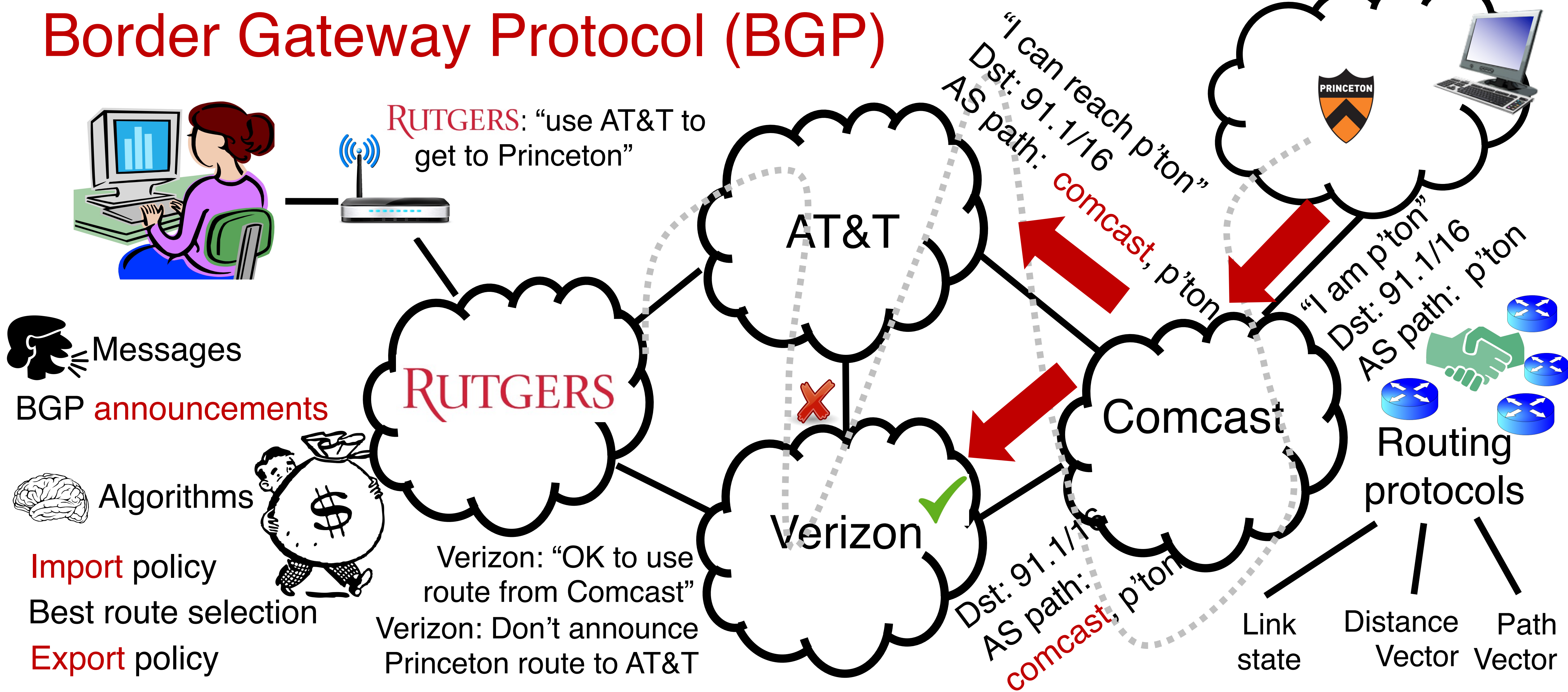
Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

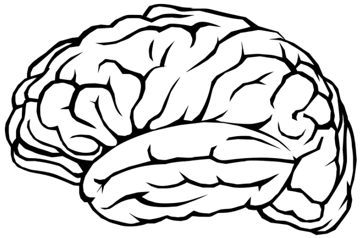
The network layer is all about reachability.

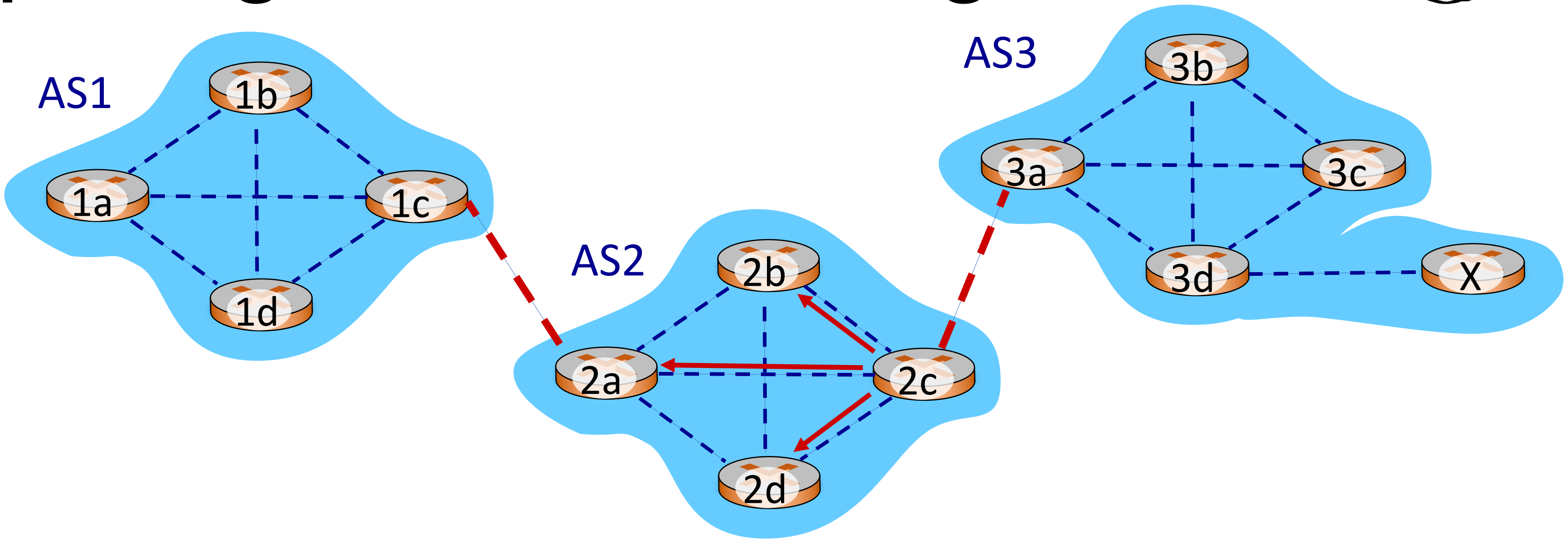
Internet is federated! Incomplete visibility

Border Gateway Protocol (BGP)



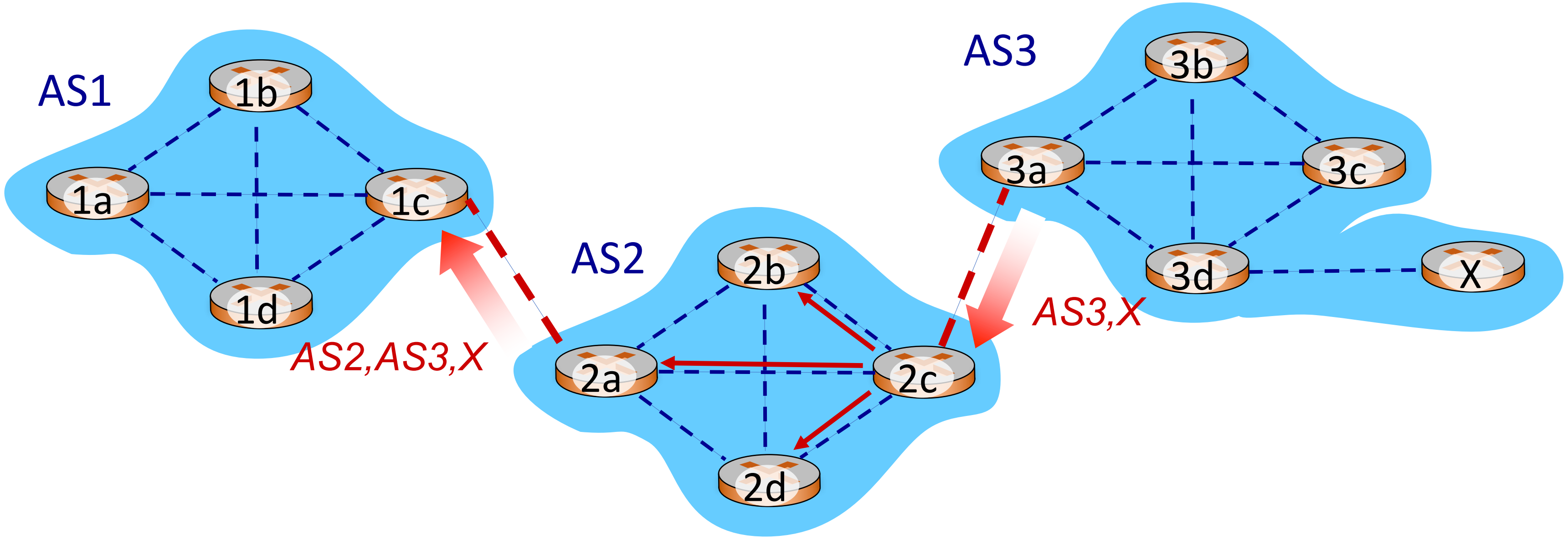
Computing the forwarding table





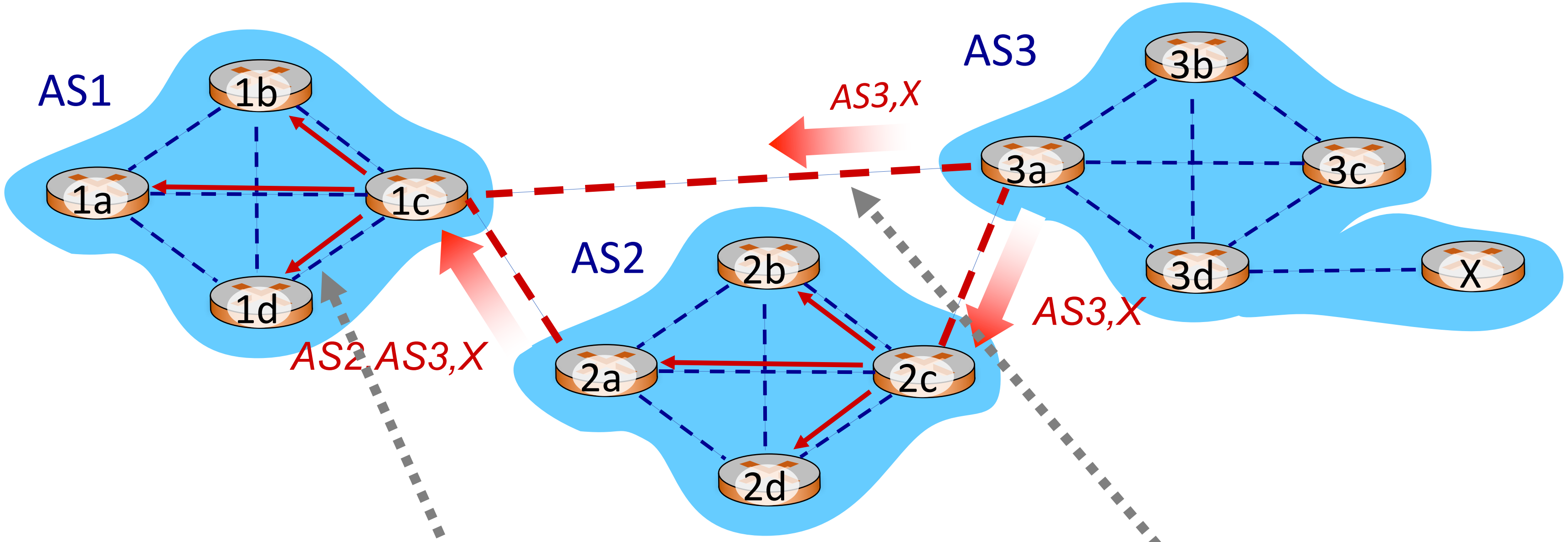
- Suppose a router in AS1 wants to forward a packet destined to external prefix X.
- How is the forwarding table entry for X at 1d computed?
- How is the forwarding table entry for X at 1c computed?

eBGP and iBGP announcements



- AS2 router 2c receives path announcement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 import policy, AS2 router 2c imports and selects path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 export policy, AS2 router 2a announces (via eBGP) path AS2, AS3, X to AS1 router 1c

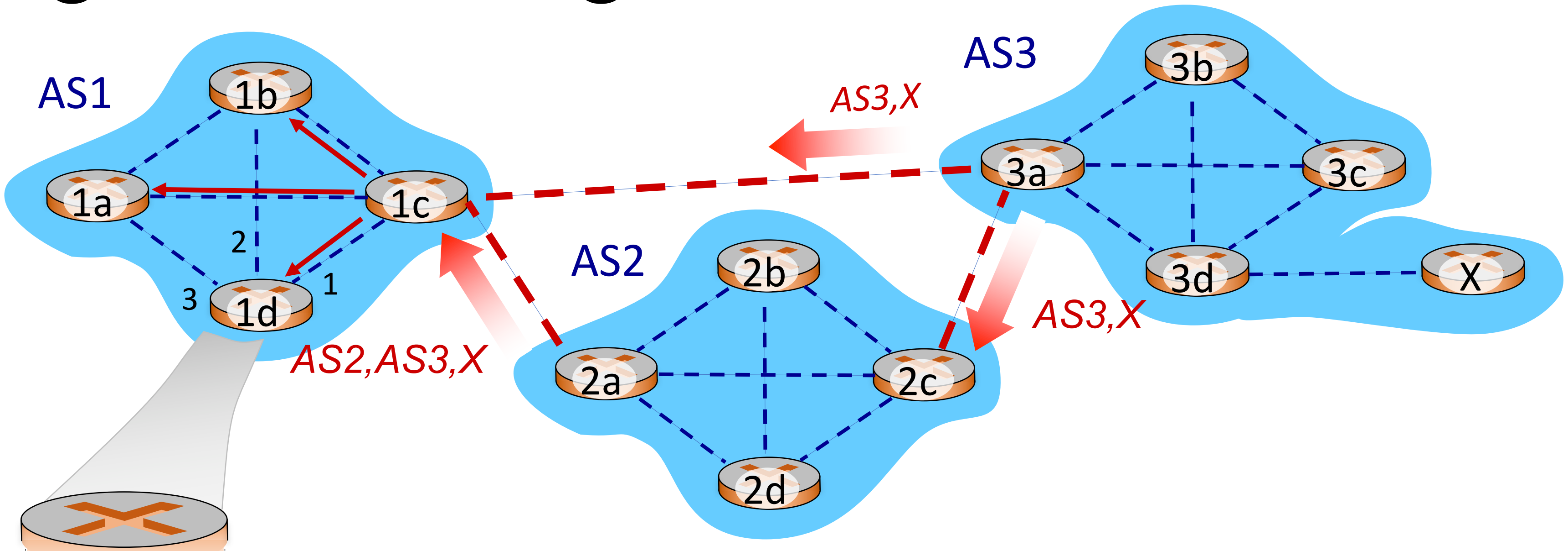
eBGP and iBGP announcements



A given router may learn about multiple paths to destination:

- AS1 gateway router 1c learns path AS2,AS3,X from 2a (next hop 2a)
- AS1 gateway router 1c learns path AS3,X from 3a (next hop 3a)
- Through BGP route selection process, AS1 gateway router 1c chooses path AS3,X, and announces path within AS1 via iBGP (next hop 1c)

Setting forwarding table entries



dest	Next-Hop
…	…
X	1c
…	…

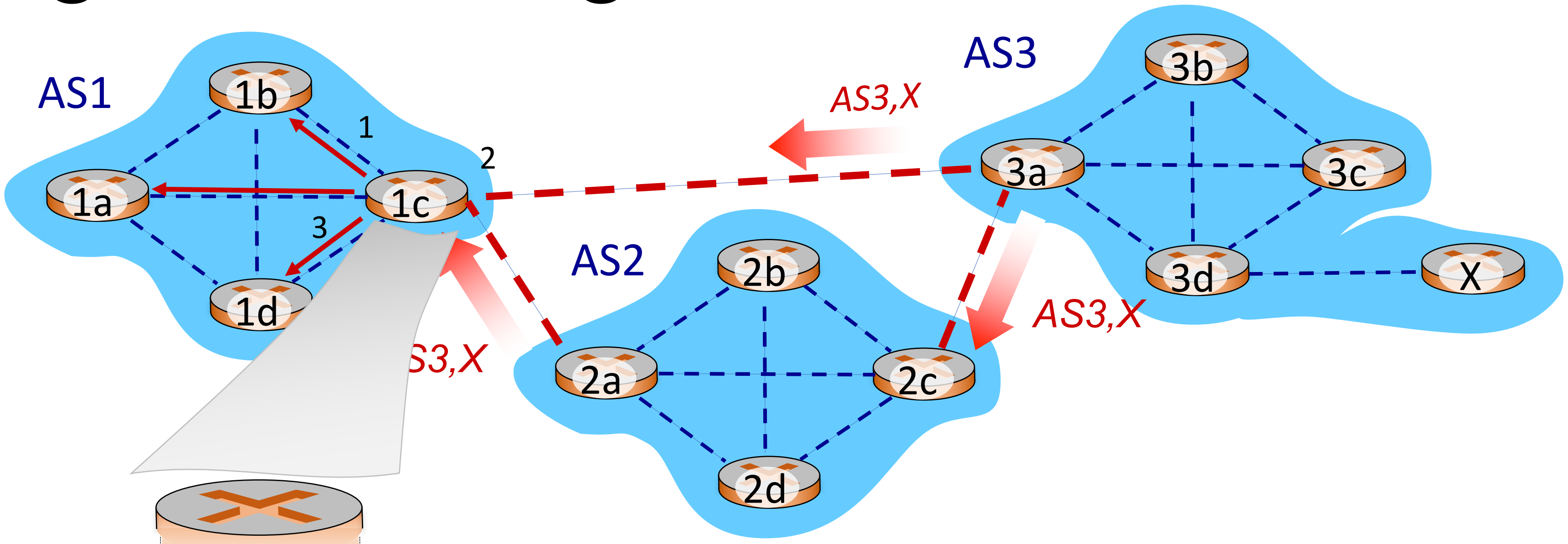
NH	interface
…	…
1c	1
…	…

IP Dst	Out port
X	1

Forwarding table

- recall: 1a, 1b, 1d learn about dest X via iBGP from next-hop 1c: "path to X goes through 1c"
- 1d: intra-domain routing: to get to 1c, forward over outgoing local interface 1

Setting forwarding table entries



dest	Next-Hop
…	…
X	3a
…	…

NH	interface
…	…
3a	2
…	…

IP Dst	Out port
X	2

Forwarding table

- recall: 1c learns about dest X via eBGP from next-hop 3a: "path to X goes through 3a"
- 1c: to get to link-local neighbor 3a, forward out interface 2

Summary: Inter-domain routing

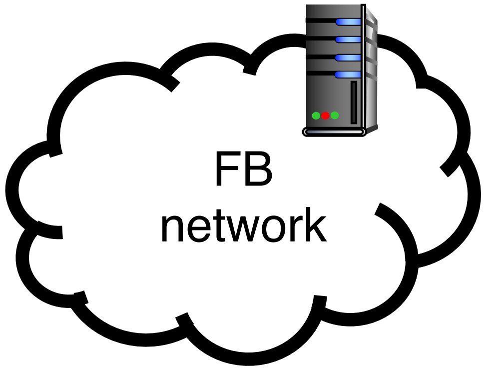
- Federation and scale introduce new requirements for routing on the Internet

- BGP is *the* protocol that handles Internet routing

- Path vector: exchange paths to a destination with attributes

- Policy-based import of routes, route selection, and export

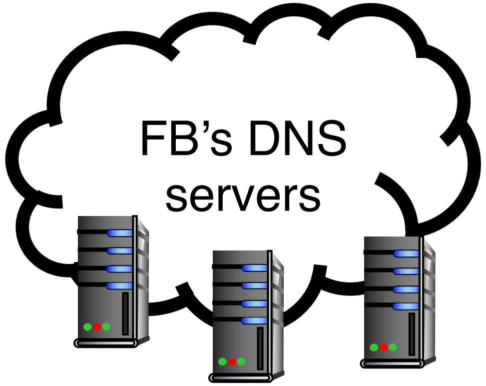
BGP's impact: October '21 FB++ outage

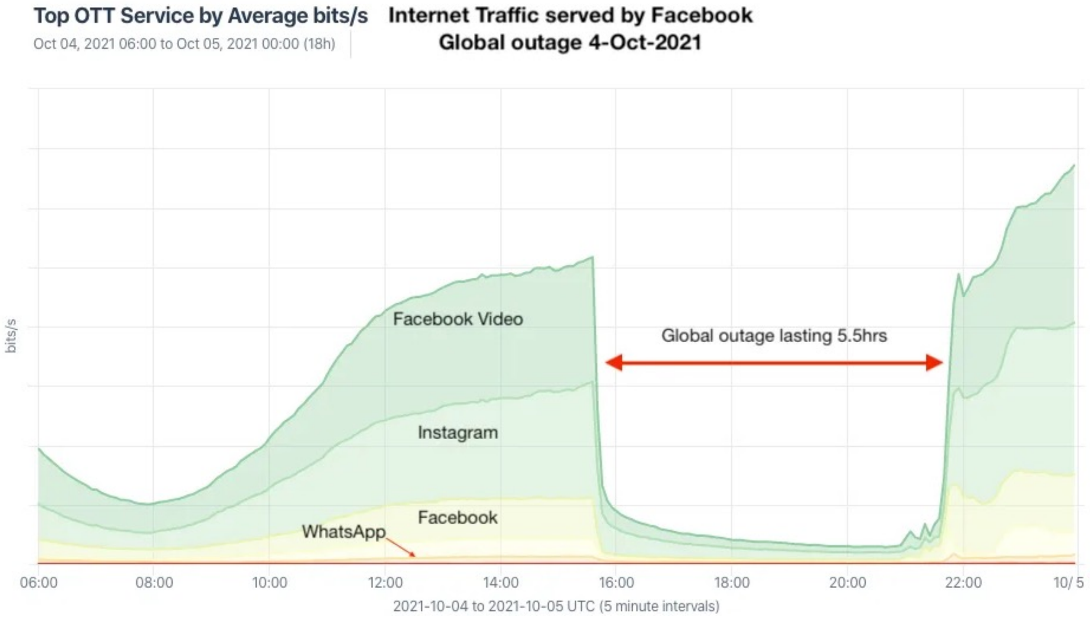


BGP route
withdrawal:
"I can't reach FB
anymore"

Rest of the
Internet

BGP route
withdrawal: don't
use me to get to FB





No remote access (no more reachability due to BGP withdrawal of DC and DNS servers)

Restricted physical access (prox can't verify, can't access prox server)

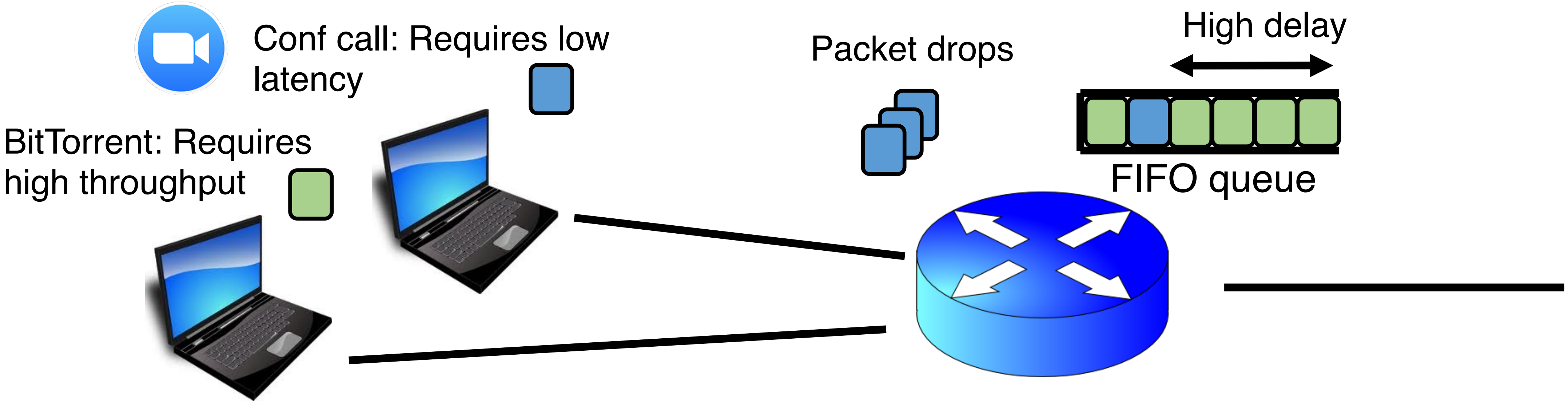
https://engineering.fb.com/2021/10/05/networking-traffic/outage-details/

Beyond best effort networking

Network support for applications

- A best effort Internet architecture does not offer any guarantees on delay, bandwidth, and loss
  - Network may drop, reorder, corrupt packets
  - Network may treat traffic randomly regardless of their “importance”
- However, many apps require special treatment & guarantees
  - E.g., voice over IP (phone calls) require strict delay guarantees
  - E.g., HD video requires a reasonable minimum bandwidth
  - E.g., remote surgery with 3D-vision requires strict sync & latency
- Q: How to provide quality of service (QoS) for apps?

Why best effort isn't enough: Contention



- Resource contention occurs in the core of the network
- Congestion control will react, but may be too little & too late:
  - Congestion control can't prevent packet drops "now"
  - Congestion control won't prevent high-sending-rate flows from inflicting large delays or recurring drops

Can networks help improve the quality of service for applications?

Yes, but networks must become better than best-effort.

Approach 1: Provision more capacity

- If you're an ISP (e.g., AT&T), you might deploy enough capacity so that contention doesn't occur any more
  - Low complexity: can use current "best effort" network
- However, this approach incurs high costs (e.g., bandwidth)
- A key challenge: estimating how much bandwidth is enough
  - Need to estimate demand over time
  - Network operators can do this quite well usually
  - But there are exceptional circumstances: pandemics, Superbowl, etc.

Approach 2: Classes of service

- Have the network treat different traffic differently
  - Also called traffic differentiation

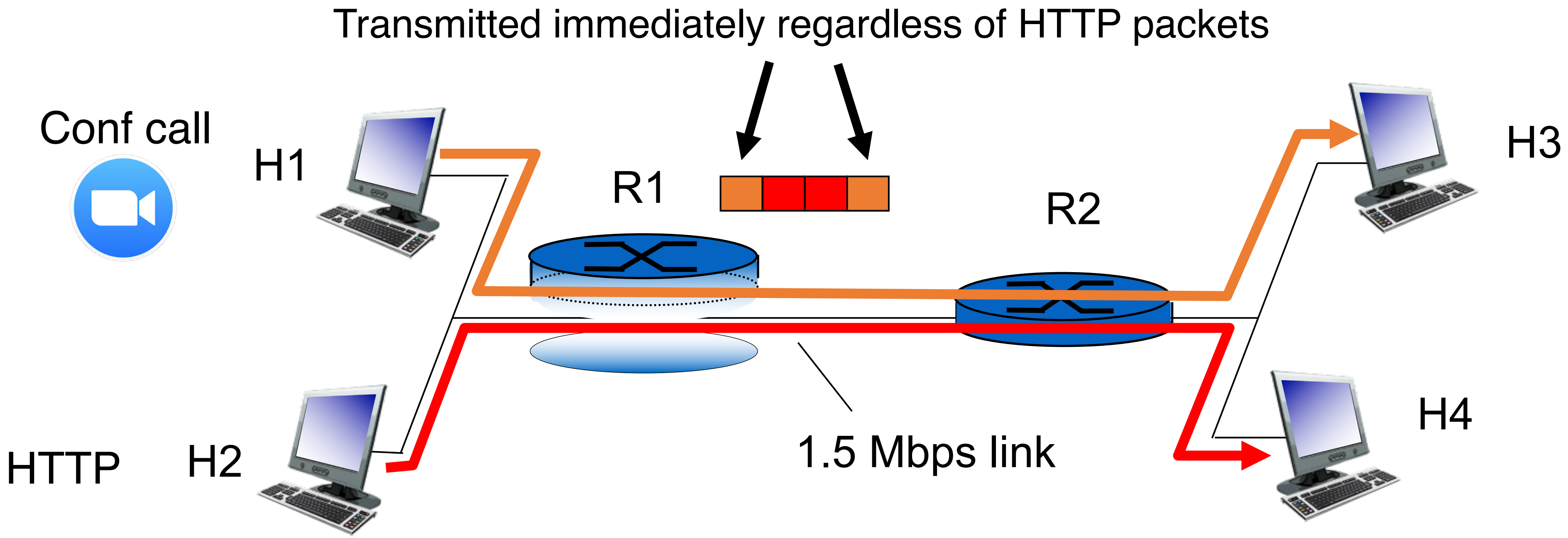
- Analogy: lines at an airport (e.g., first class vs. economy)

- Partition traffic into classes and offer service guarantees per class and across classes
  - Classes may be indicated using the IP type of service header bits
  - Classes may be inferred from IP & transport headers (e.g., src/dst/ports)

- Packet classification: assigning packets to classes
  - (Not in scope: we won't discuss packet classification)

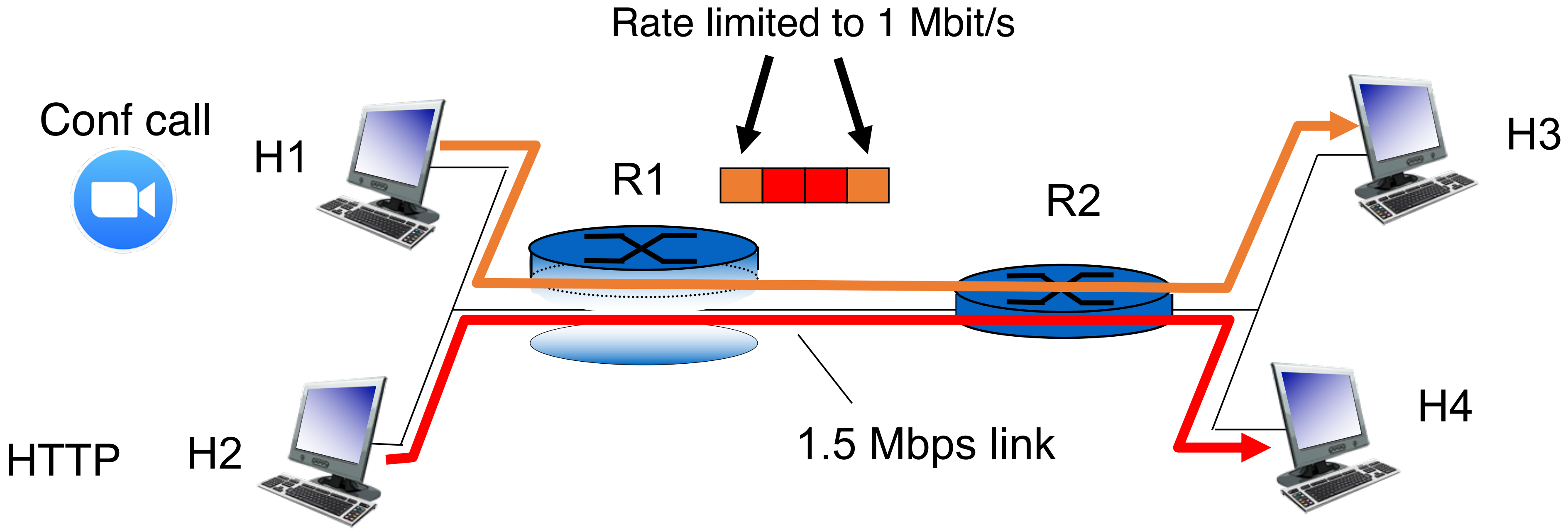
Kinds of Service Guarantees

(1) Strict prioritization



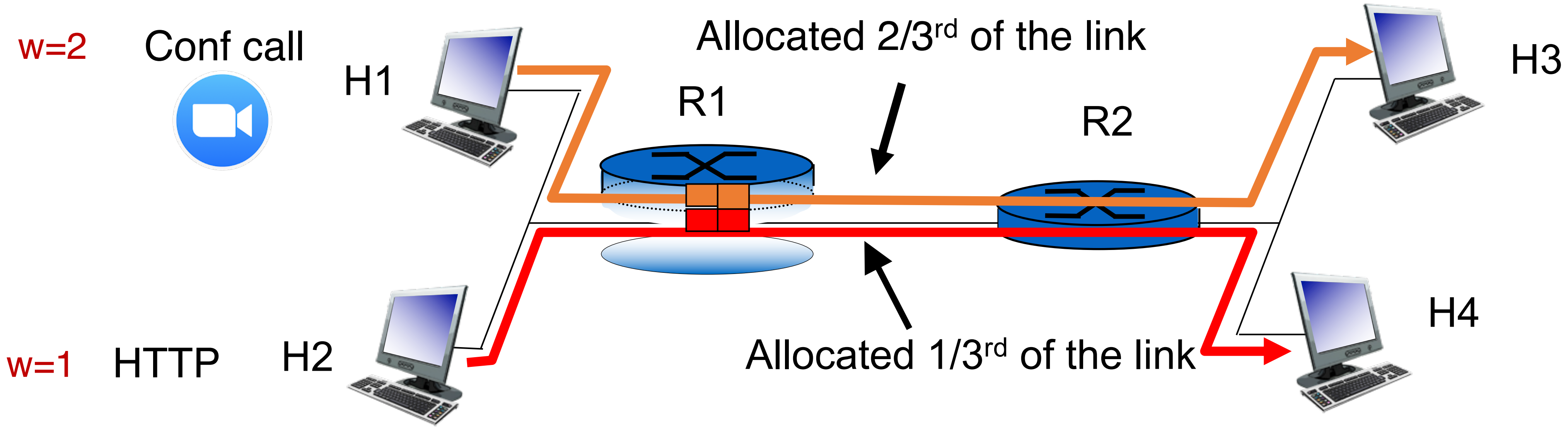
- Suppose a 1Mbps interactive flow and an HTTP connection share a 1.5 Mbps link.
- A network operator (e.g., Rutgers admin) might choose to prioritize the interactive app strictly over the HTTP flow.

(2) Rate limiting



- What if a flow doesn't respect its allocation?
  - Example: Say, conf call flow goes beyond 1 Mbit/s
  - Don't want to starve HTTP flow!
- An operator might want to limit a flow to a certain max rate
- Isolation: HTTP should not be impacted by the conf call

(3) Weighted fair sharing

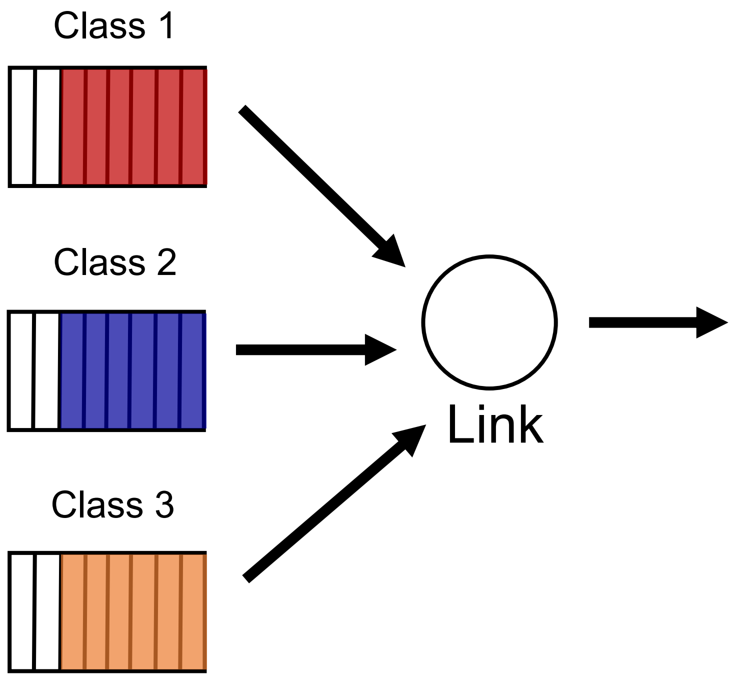


- An operator might want to partition the link's rate C into separate allocations for each class
  - Partitions may have weights w (example: 2, 1)
- Usually, class i gets the illusion of traversing a logical link of rate

$$w_i * C / \sum_j w_j$$

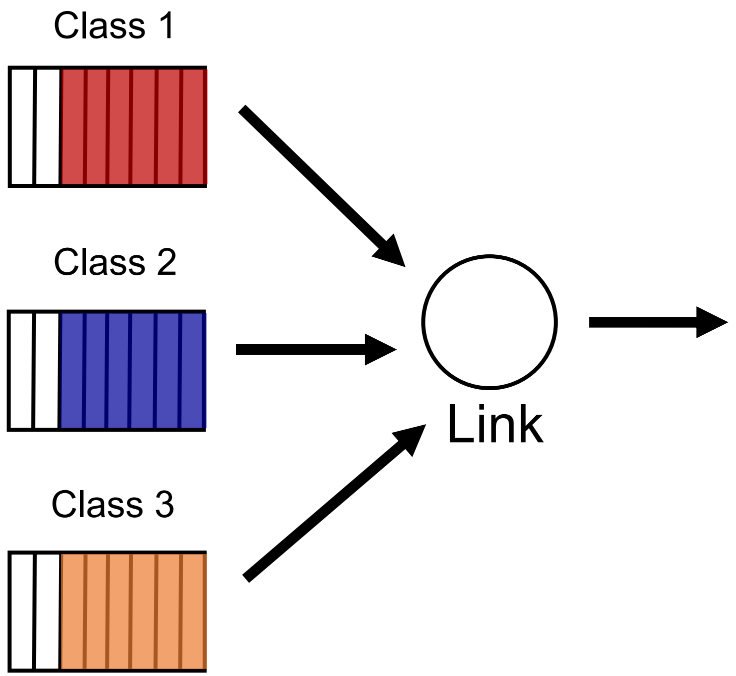
(3) Weighted fair sharing

- Customary to think of different classes as belonging to different queues
- For this reason, weighted fair sharing is also called weighted fair queueing (WFQ)
- Each queue is first-in-first-out (FIFO)
- The link multiplexes among these queues
- Intuitively, packets of one queue should not influence the behavior of other queues
- Hence, fair queueing is also a form of isolation across traffic classes



(3) Weighted fair sharing

- But what if one class doesn't use its share?
  - Can other classes use the spare capacity?
- Yes! WFQ is work-conserving: a router implementing WFQ will allow other classes to use the unused capacity
- Work conservation makes WFQ different from rate limits applied separately to each class
  - Class i's usage can exceed $w_i * C / \sum_j w_j$
  - (only if spare capacity is available, of course.)



Q: Where are guarantees enforced?

- We’ve seen three kinds of service guarantees: prioritization, rate limiting, and fair sharing
- Common goal: allocate the bottleneck link capacity across packets from traffic classes
- This allocation occurs in the packet scheduler in the bottleneck router
  - Recall: scheduling is the task of choosing the packet (among buffered packets) which is transmitted over the output link
- A router is said to implement packet scheduling policies

Why care about service guarantees?

- Influences how packets are treated at contentious resources in the core of the network
  - Regardless of the endpoint transport

Next lecture: a deeper look at mechanisms for one kind of service guarantee

- Service guarantees: prioritization, rate limiting, fair sharing

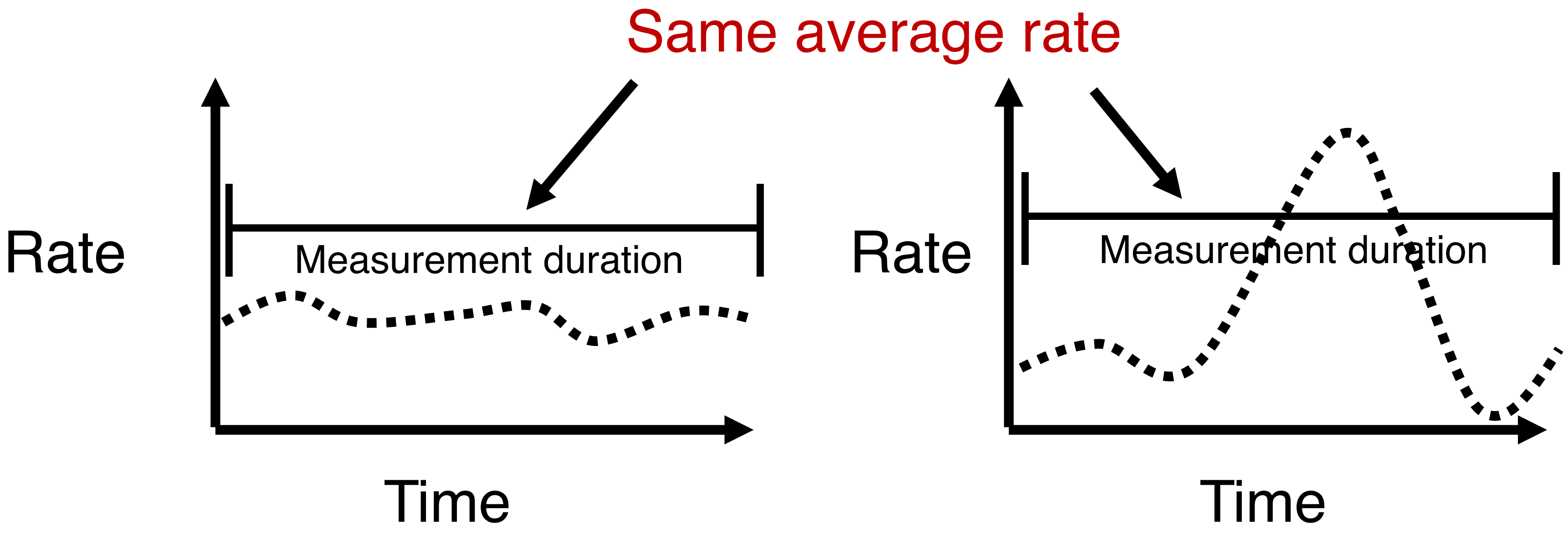
- Implementations of scheduling (QoS) within large networks have implications for debates on network neutrality

- Scheduling is a fundamental problem in computer networks

Rate Limiting

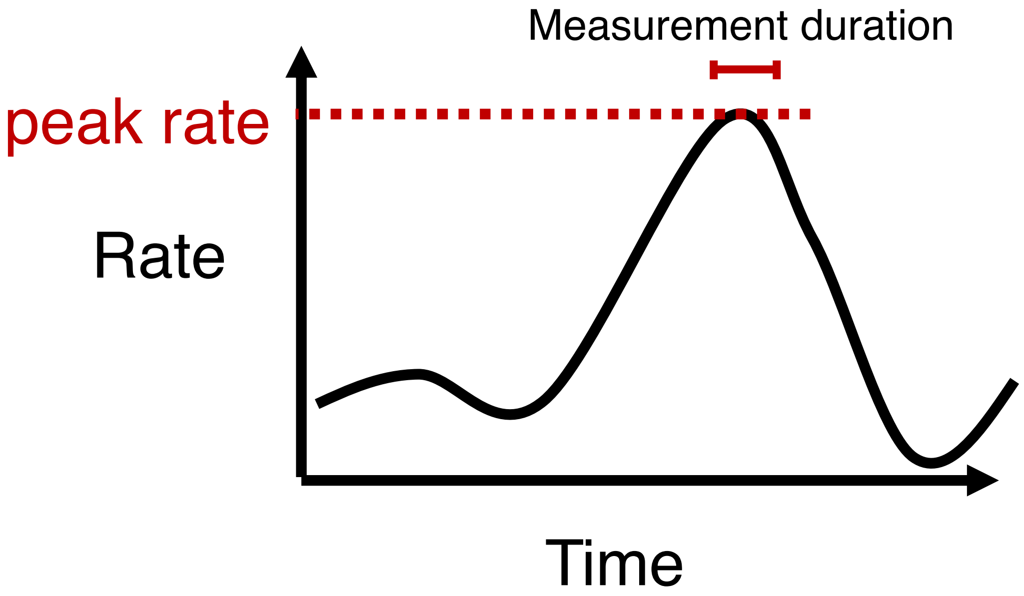
Measures of transmission rate

- Long-term/average rate: data rate transmitted per unit time, over a long period
  - Crucial question: what is the time interval over which rate is measured?
- Average and instantaneous behaviors can be very different



Measures of transmission rate

- Peak rate: largest instantaneous rate that is transmitted
  - Measurement duration is typically very small
- Burst size: maximum amount of data sent consecutively without any intervening idle periods



Rate enforcement

- There are two kinds of rate enforcement policies:
  - shaping and policing

- Two specific mechanisms to implement those:
  - leaky buckets and token buckets

Shaping vs. Policing

Shaping

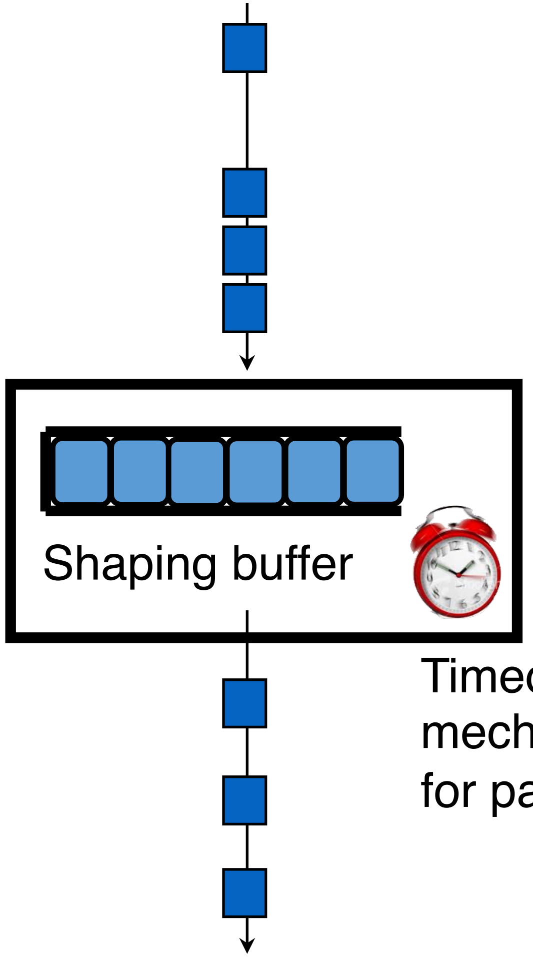
- Enforces rate by queueing excess packets in a buffer
  - Drop only if buffer is full
- Requires memory to buffer packets
- Can inflate round-trip time (queueing in shaping buffer)

Policing

- Enforces rate by dropping excess packets immediately
  - Can result in high loss rates
- Does not require a memory buffer
- No additional inflation in round-trip times

Leaky bucket shaper

Intuition: release packets at steady rate



Leaky Bucket Shaper

- Packets may enter in a bursty manner
- However, once they pass through the leaky bucket, they are evenly spaced
- The shaping buffer holds packets up to a certain point
  - If the buffer is full, packets are dropped
- Setting the rate is a policy concern
  - Assume an admin provides us the rate
- Shapers may be used in the core of a network to limit bandwidth use, or at the edge to pace packets entering the network in the first place

Leaky Bucket Shaper

- For a leaky bucket shaper, assume average rate == peak rate

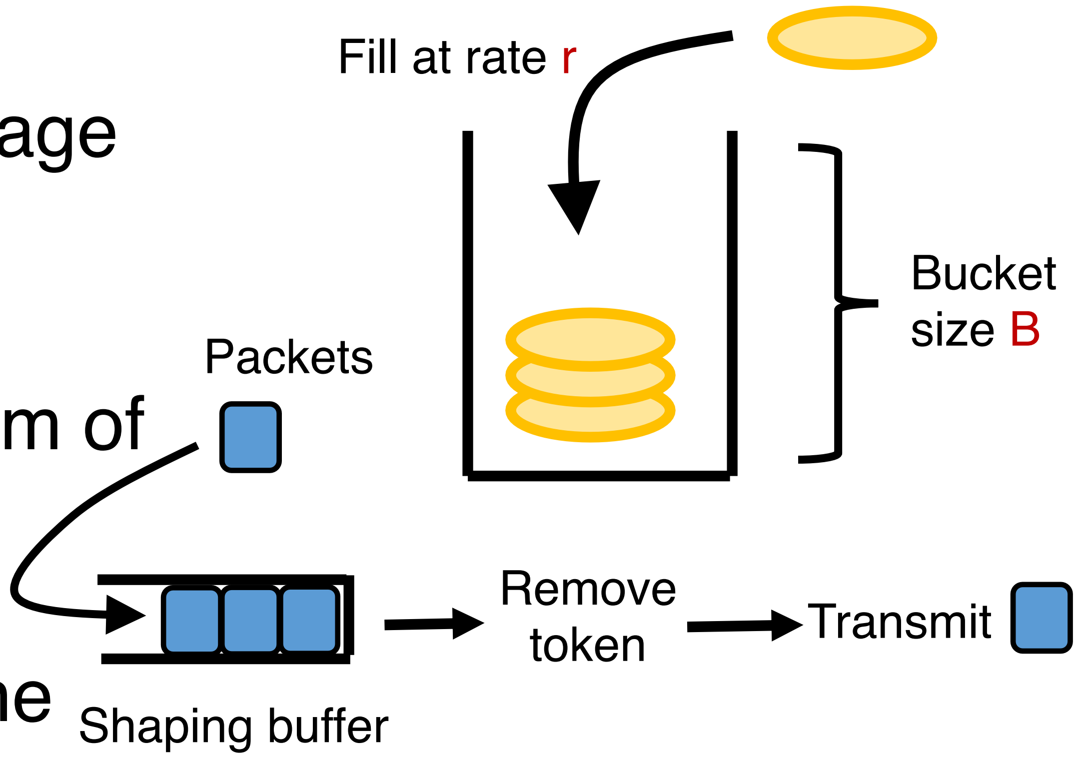
- However, many Internet transfers just have a few packets
  - For example, web requests and responses
  - Enforcing rate limit for those can significantly delay completion

- We often wish to have peak rate higher than avg rate
  - Especially at the beginning of a connection
  - If so, use a token bucket: burst-tolerant version of a leaky bucket

Token bucket shaper

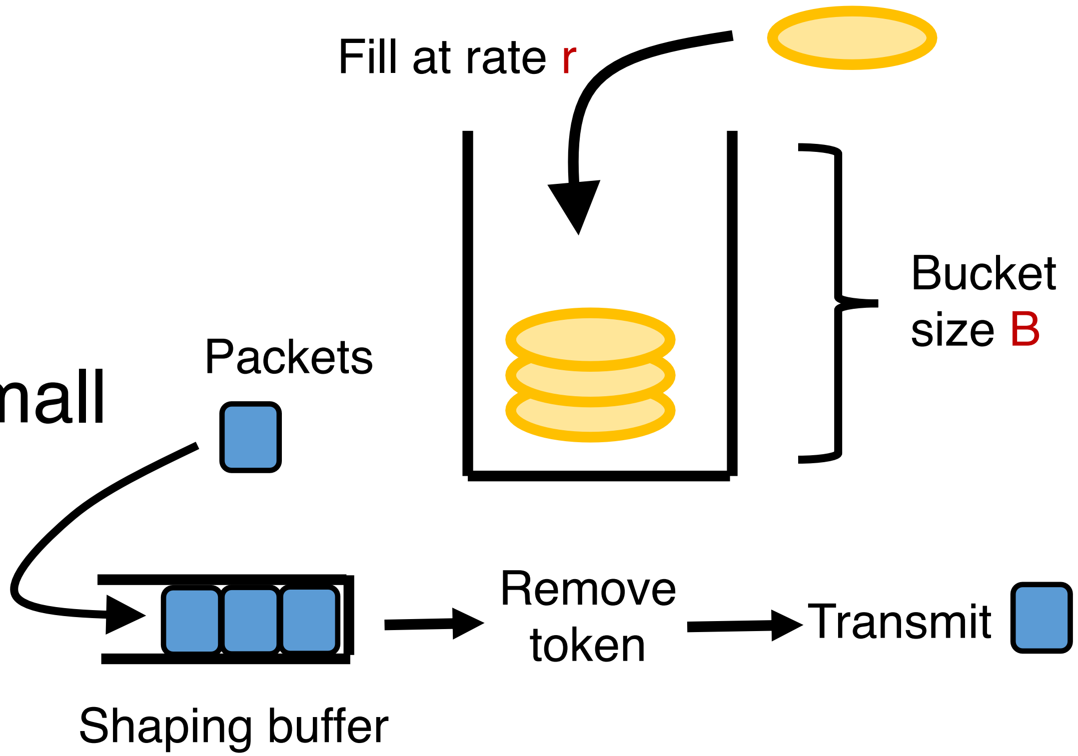
Token bucket shaper

- Limits traffic class to a specified average rate r and burst size B
- Tokens are filled in at rate r
- The token bucket can hold a maximum of B tokens. Further tokens dropped
  - Note: distinct from shaping buffer size
- Suppose a packet is at the head of the shaping buffer
- If a token exists in the bucket, remove token, and transmit the packet
  - If not, wait.



Token bucket shaper

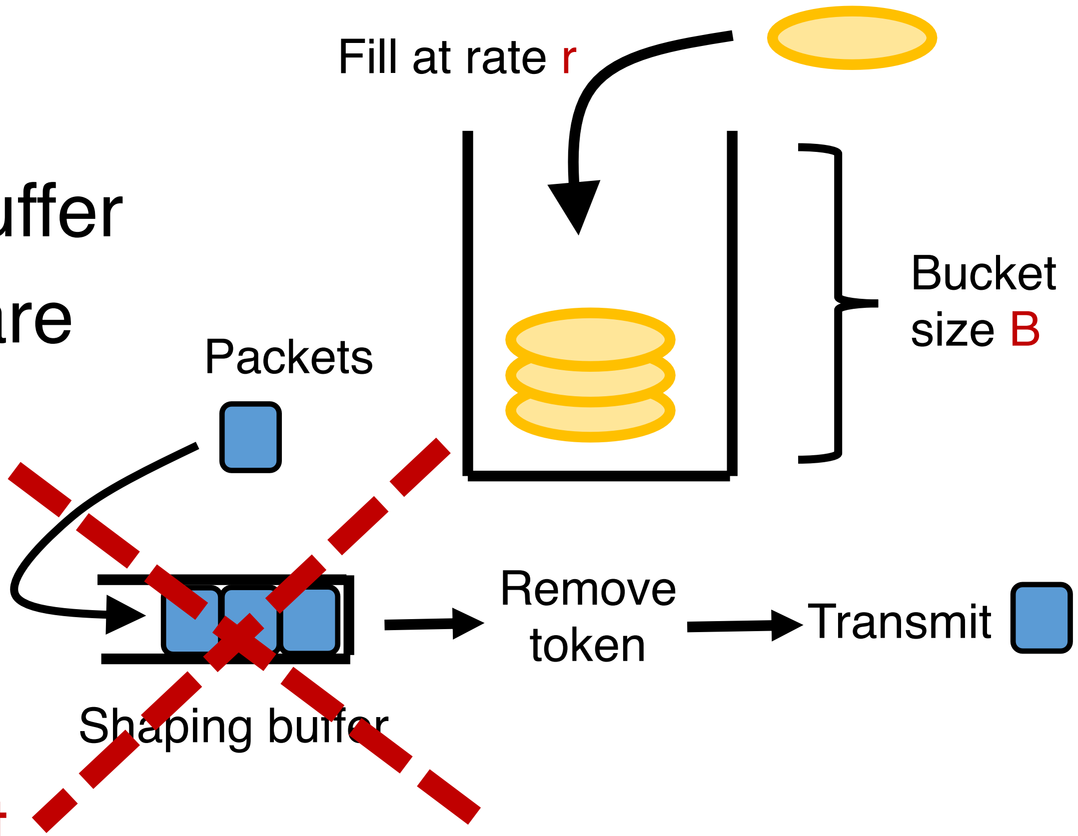
- In time t, the maximum number of packets that depart the shaper is (r * t) + B
- A full bucket of tokens would allow small flows to go through unaffected
  - A maximum burst of B packets
- Longer flows have average rate r
  - Bucket emptied initially, the rest of the flow must respect the token fill rate
  - As t → ∞, the average rate approaches r
  - That is, (1/t) * (r*t + B) → r



Token bucket policers

Token bucket policer

- A token bucket policer is just a token bucket shaper without the shaping buffer
- No place for packets to wait if there are no tokens
- If token exists, packet transmitted.
- If not, packet dropped
- Simple and efficient to implement.
- The internet has tons of token bucket policers



Google study from 2016

Region	Policed segments (among lossy)	Policed segments (overall)	Loss rate (policed)	Loss rate (non-pol.)
India	6.8%	1.4%	28.2%	3.9%
Africa	6.2%	1.3%	27.5%	4.1%
Asia (w/o India)	6.5%	1.2%	22.8%	2.3%
South America	4.1%	0.7%	22.8%	2.3%
Europe	5.0%	0.7%	20.4%	1.3%
Australia	2.0%	0.4%	21.0%	1.8%
North America	2.6%	0.2%	22.5%	1.0%

Table 2: % segments policed among lossy segments (≥ 15 losses, the threshold to trigger the policing detector), and overall. Avg. loss rates for policed and unpoliced segments.

Small but non-trivial fraction of policed links

Significant impact on packet loss rate

Flach et al., An Internet-Wide Analysis of Traffic Policing, SIGCOMM 2016

Impact on TCP

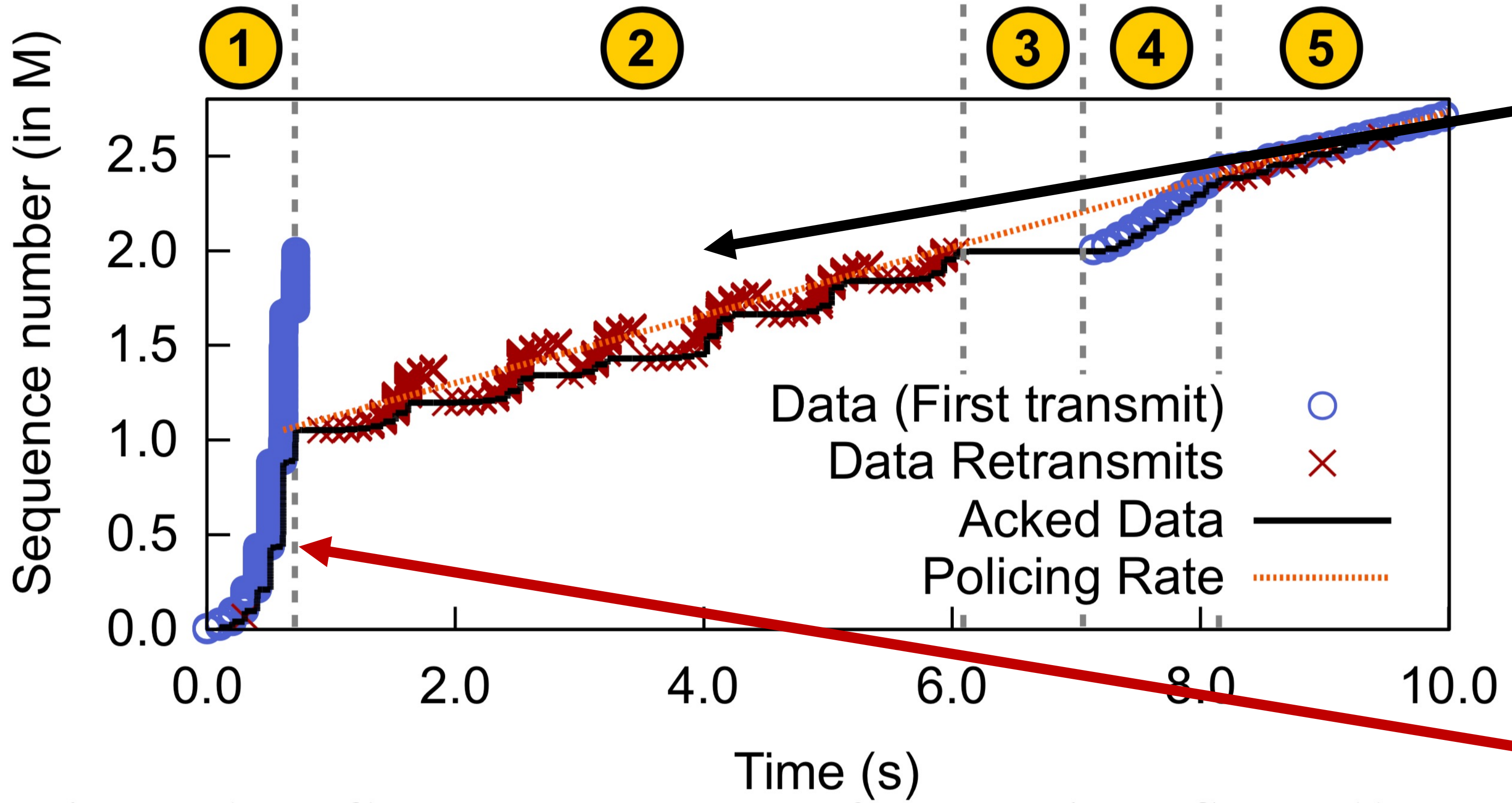


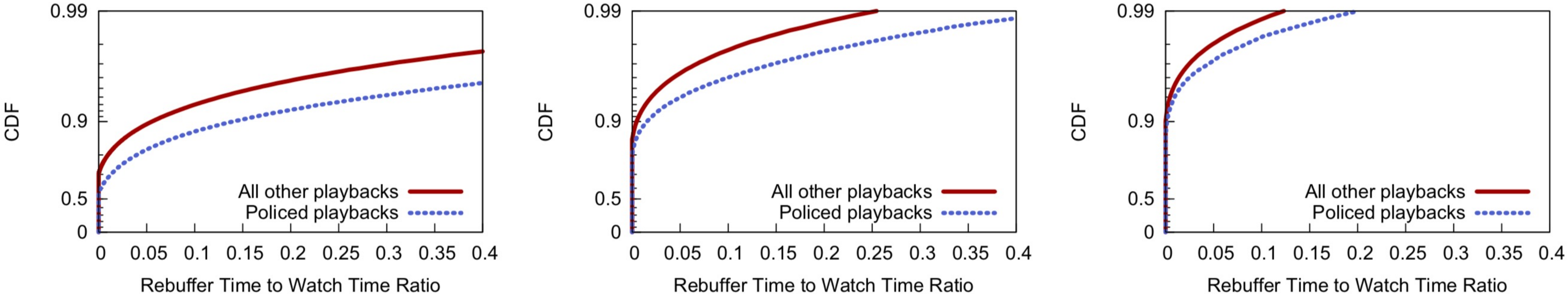
Figure 1: TCP sequence graph for a policed flow: (1 and 4) high throughput until token bucket empties, (2 and 5) multiple rounds of retransmissions to adjust to the policing rate, (3) idle period between chunks pushed by the application.

Policers drop multiple packets in a burst: causing RTOs and retransmissions after emptying of token bucket

Slow start period: burst allowed with a full bucket of tokens

Effect on actual apps: YouTube

- Video rebuffer rate: rebuffer time / overall watch time



(a) 300 kbps (b) 1.5 Mbps (c) 3 Mbps

Figure 9: Rebuffer to watch time ratios for video playbacks. Each had at least one chunk with a goodput of 300 kbps, 1.5 Mbps, or 3 Mbps ($\pm 15\%$).

Summary of rate limiting

- Rate limiting is a useful mechanism to isolate traffic classes from each other
- Two strategies: policing and shaping
  - Leaky bucket and token bucket
- The Internet has a lot of token bucket policers, causing real impact on TCP connections and app performance
- Understand how ISPs treat consumer Internet traffic

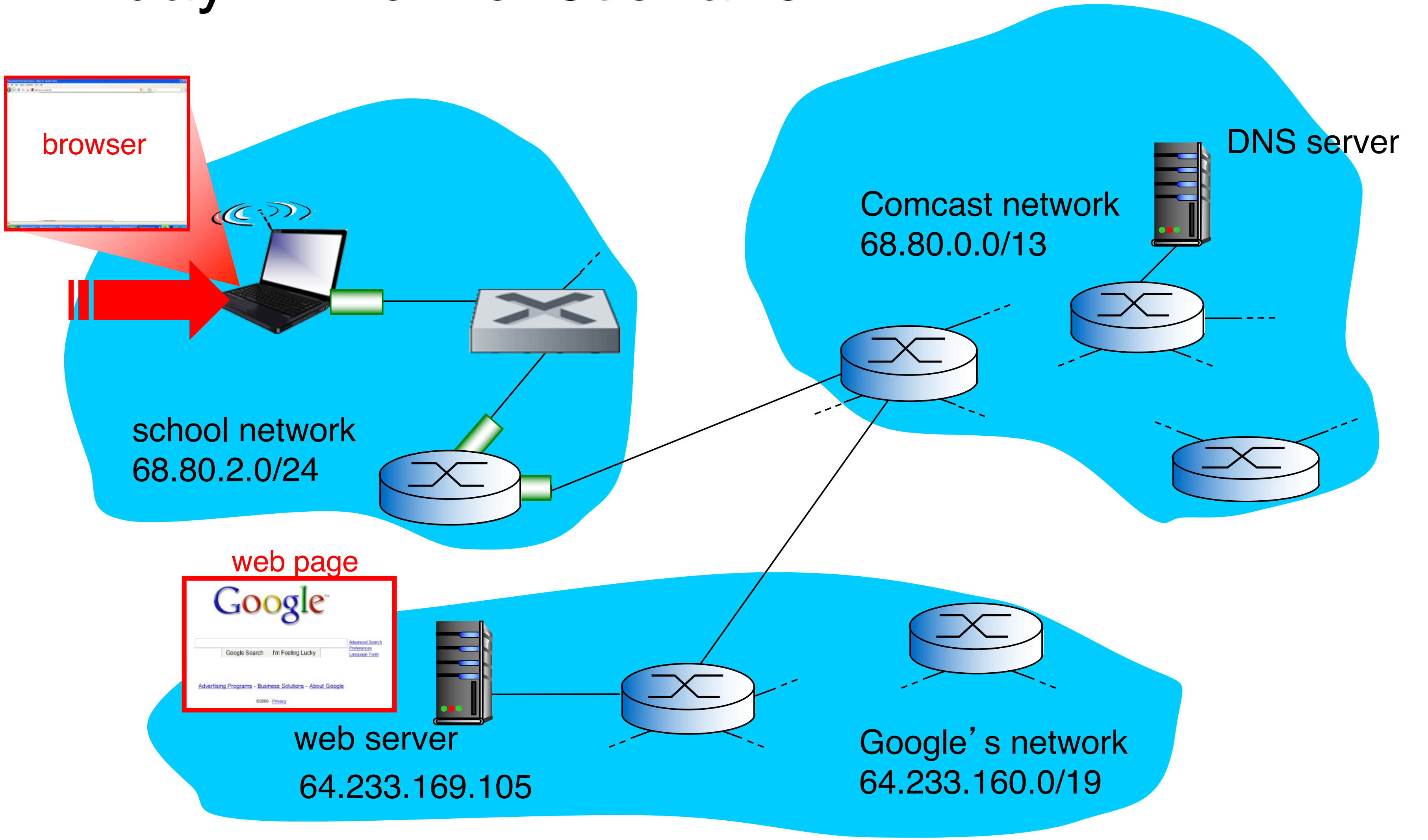
Synthesis of protocols

Synthesis: a day in the life of a web request

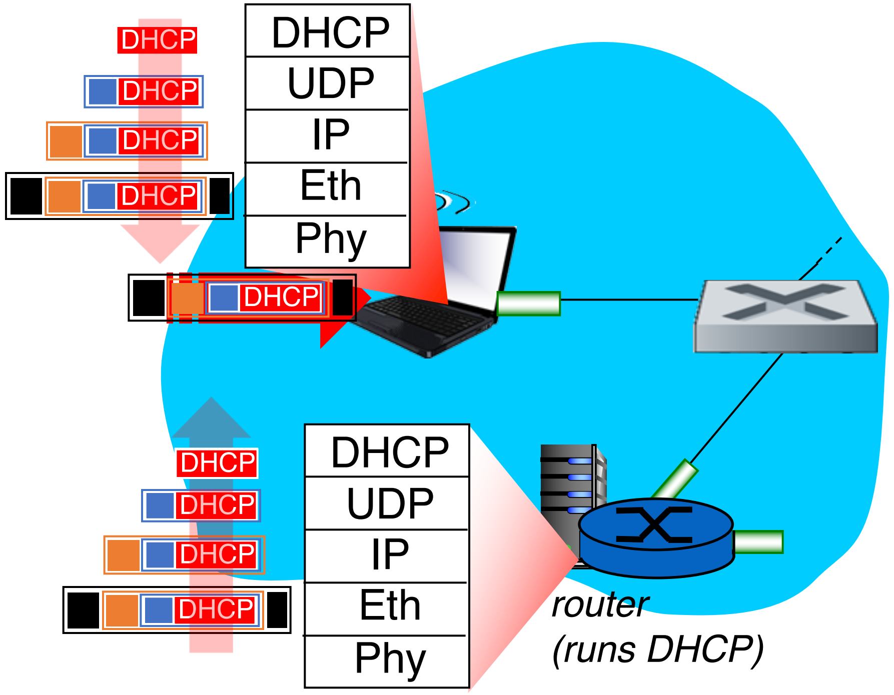
- Our journey down protocol stack complete!
  - application, transport, network, link

- putting-it-all-together: synthesis!
  - Goal: identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - Scenario: student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario



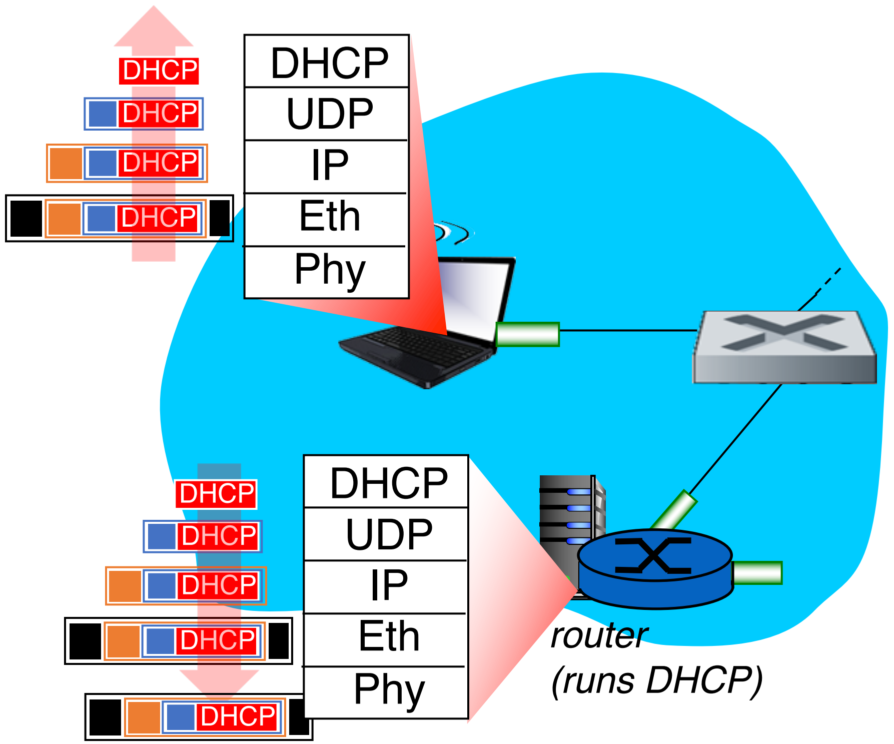
A day in the life… connecting to the Internet



- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use *DHCP*

- DHCP request encapsulated in UDP, encapsulated in IP, encapsulated in link layer Ethernet
- Packet broadcast (dest: FFFFFFFFFFFF) on the local network, received at a router running DHCP server
- Ethernet decapsulated to IP decapsulated to UDP decapsulated to DHCP

A day in the life… connecting to the Internet

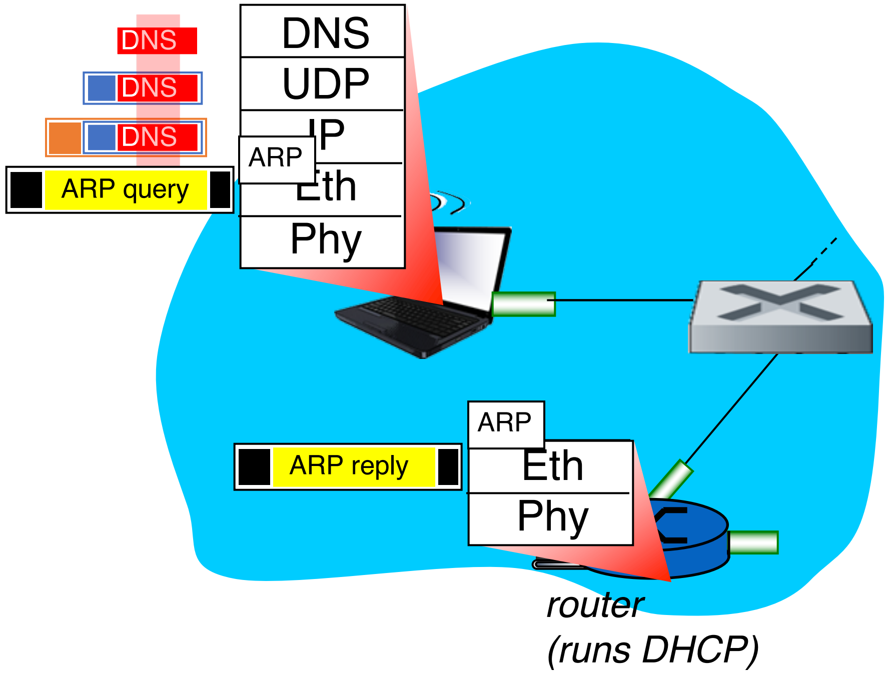


- DHCP server formulates *DHCP ACK* containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

- DHCP client receives DHCP ACK reply

Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

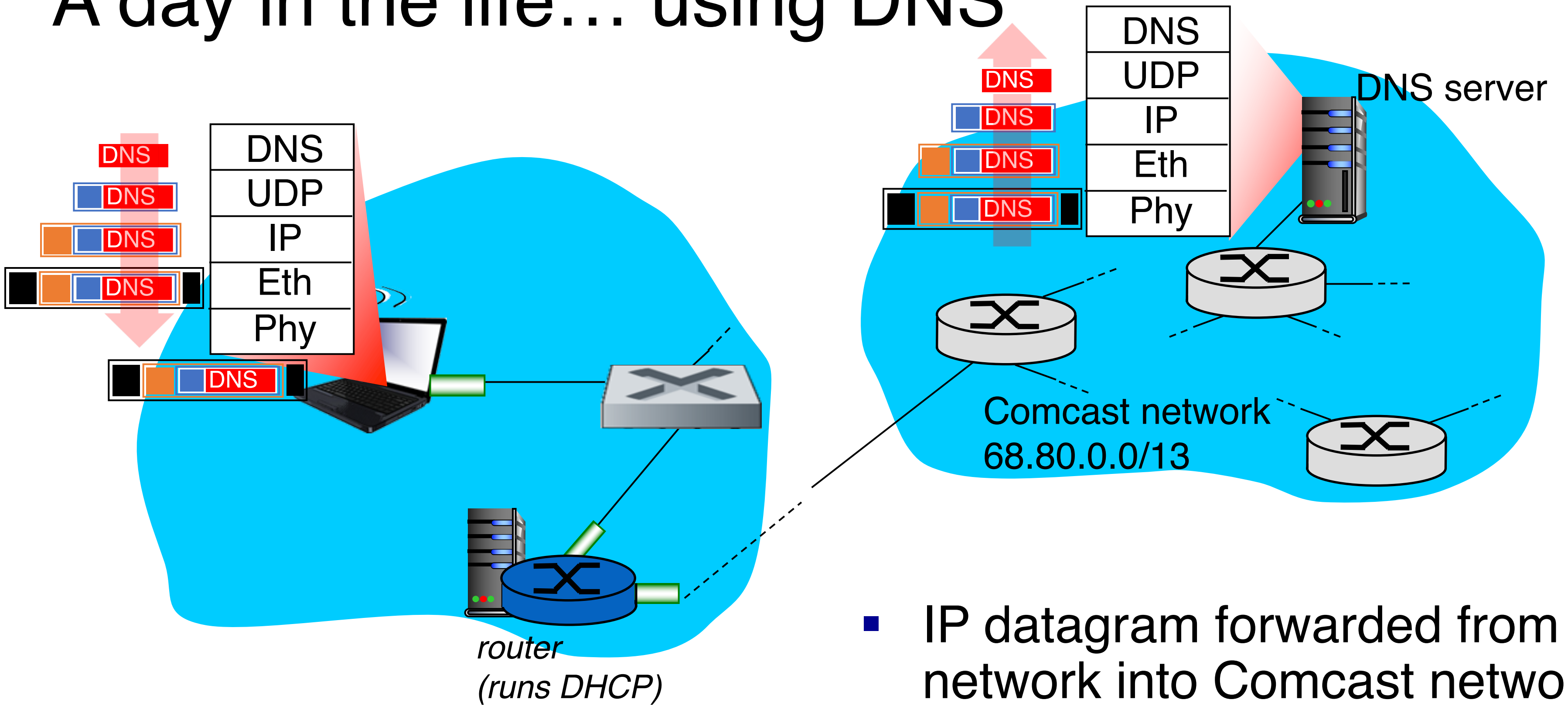
A day in the life… ARP (before DNS, before HTTP)



- before sending *HTTP* request, need IP address of www.google.com: *DNS*

- DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. To send frame to router, need MAC address of router interface: ARP
- ARP query broadcast, received by router, which replies with ARP reply giving MAC address of router interface
- client now knows MAC address of gateway router, so can now send packet containing DNS query

A day in the life… using DNS

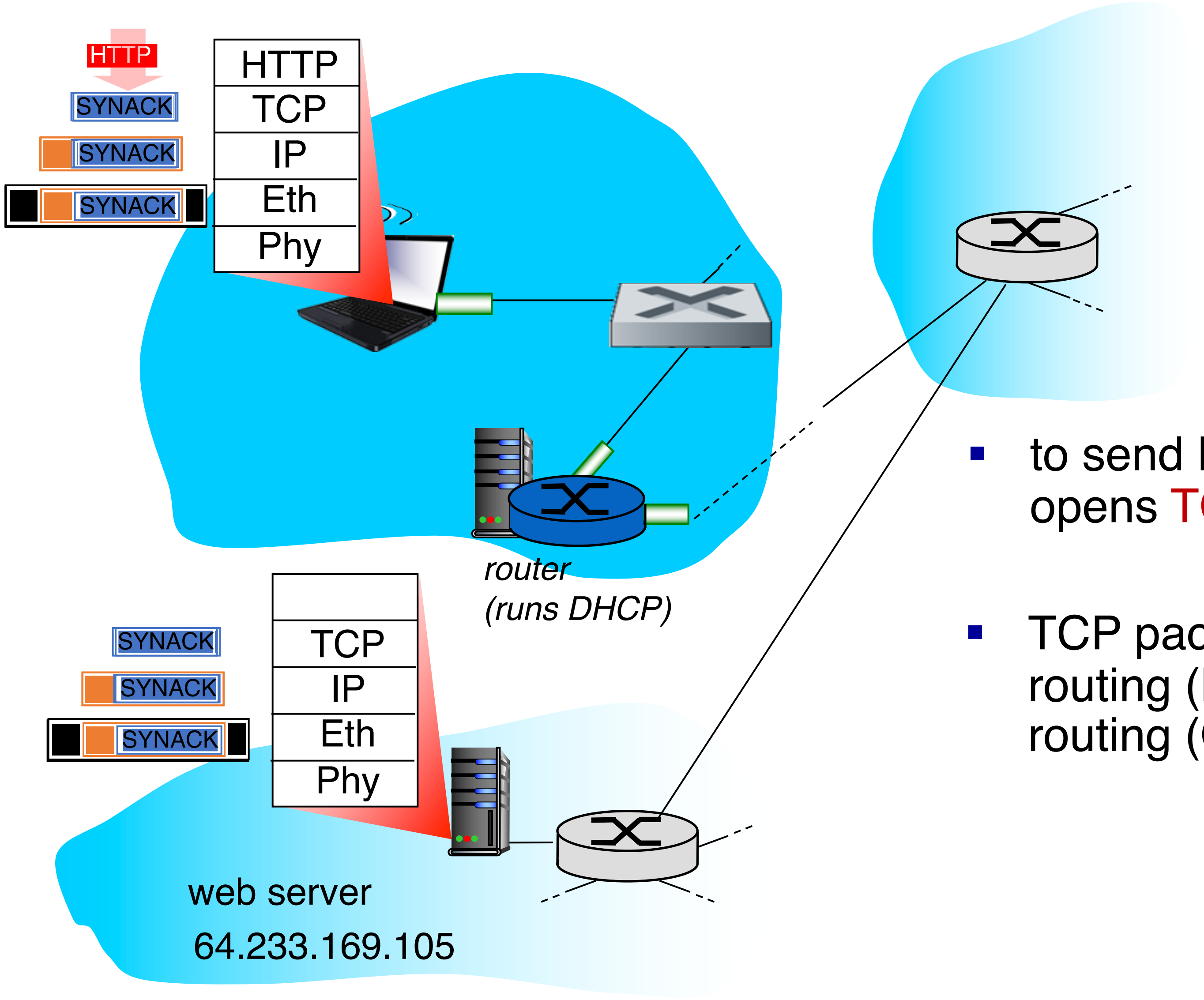


- IP datagram containing DNS query from client to gateway router

- IP datagram forwarded from campus network into Comcast network, routed (tables created by EIGRP, OSPF, and/or BGP routing protocols) to DNS server

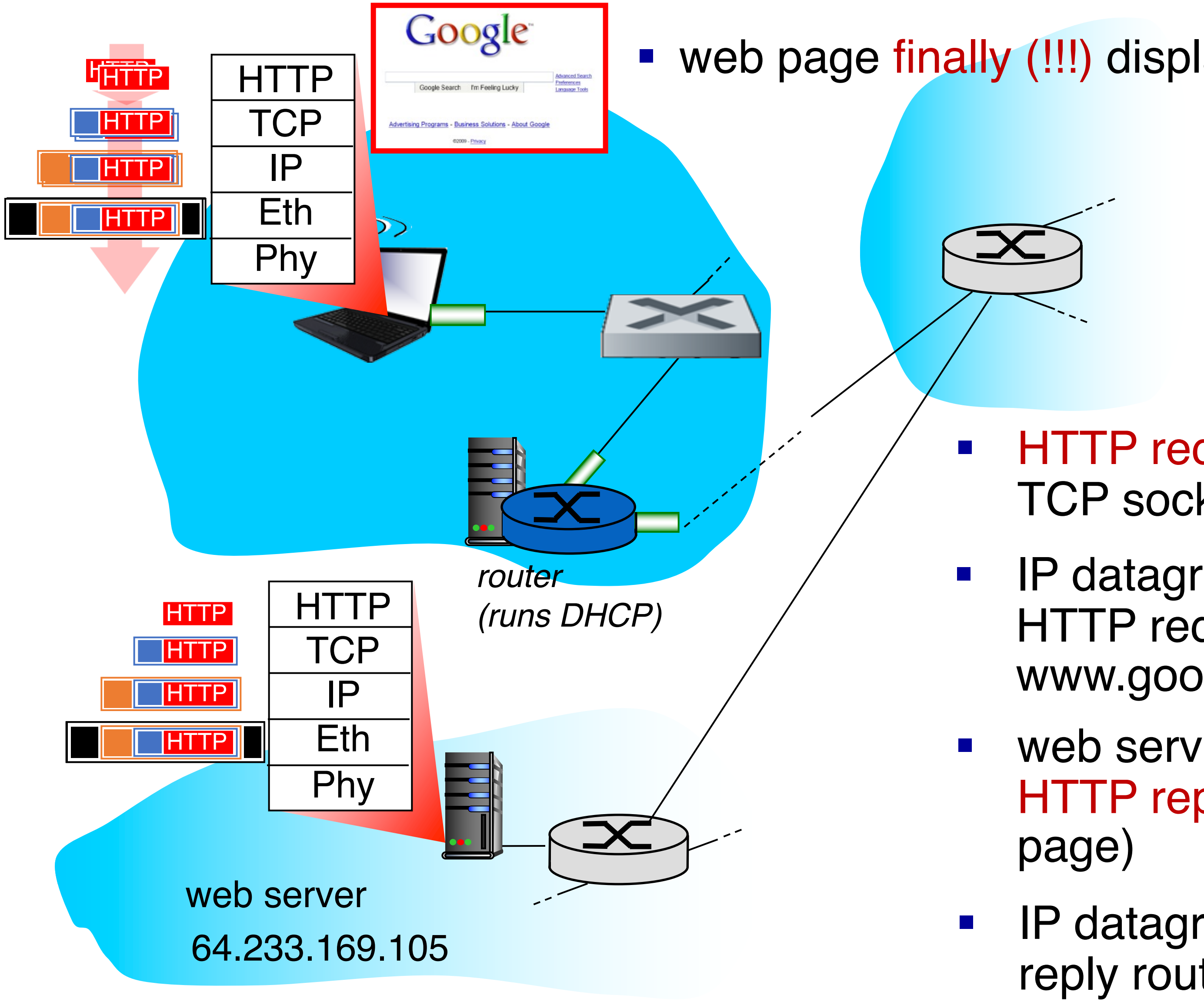
- decapsulated to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life…TCP connection carrying HTTP



- to send HTTP request, client first opens TCP socket to web server
- TCP packet routed using inter-domain routing (BGP) and intra-domain routing (OSPF, EIGRP) to web server

A day in the life… HTTP request/reply



- web page finally (!!!) displayed

- HTTP request sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with HTTP reply (containing web page)
- IP datagram containing HTTP reply routed back to client