

# Computer Networks

Instructor: Andrei Gurtov

Notes derived from "*Computer Networking: A Top Down Approach*", by Jim Kurose and Keith Ross, Addison-Wesley.

The slides are adapted and modified based on slides from the book's companion Web site, as well as modified slides by Niklas Carlsson, Anirban Mahanti and Carey Williamson.

# Kick starting science ...



... well, cable into wall ...



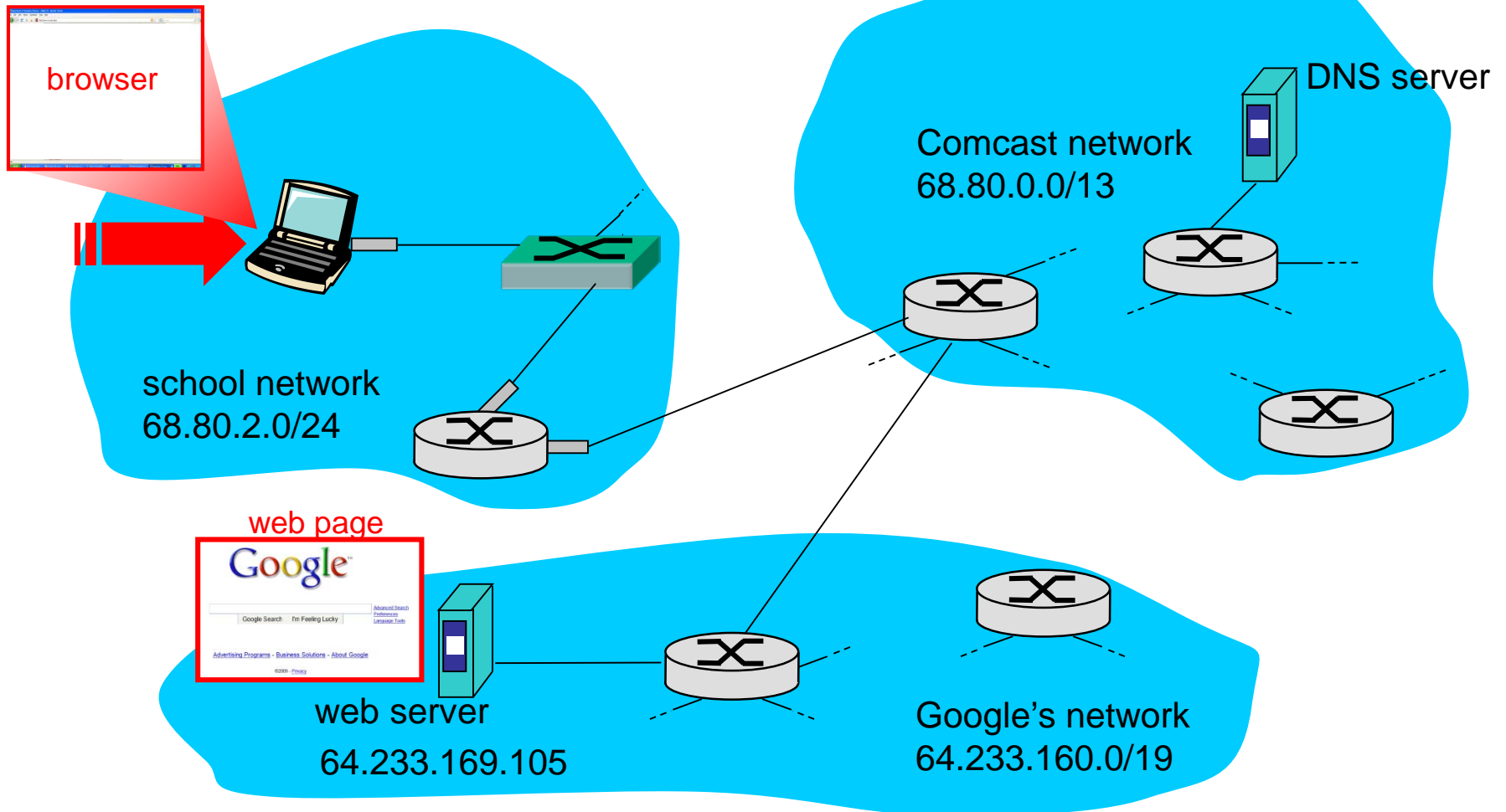
# What happens there?



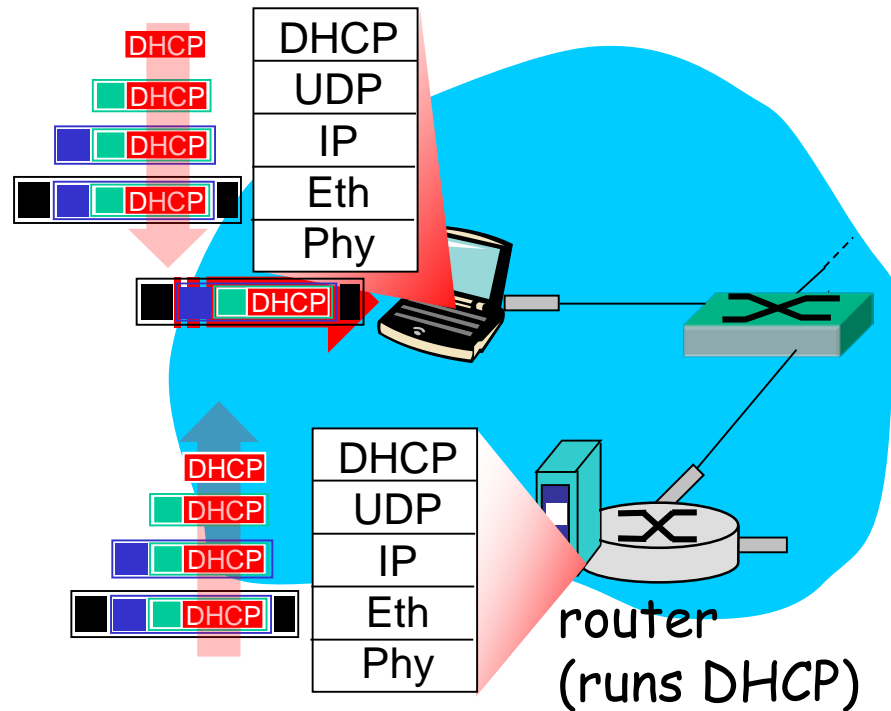
## Synthesis: a day in the life of a web request

- ❖ 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

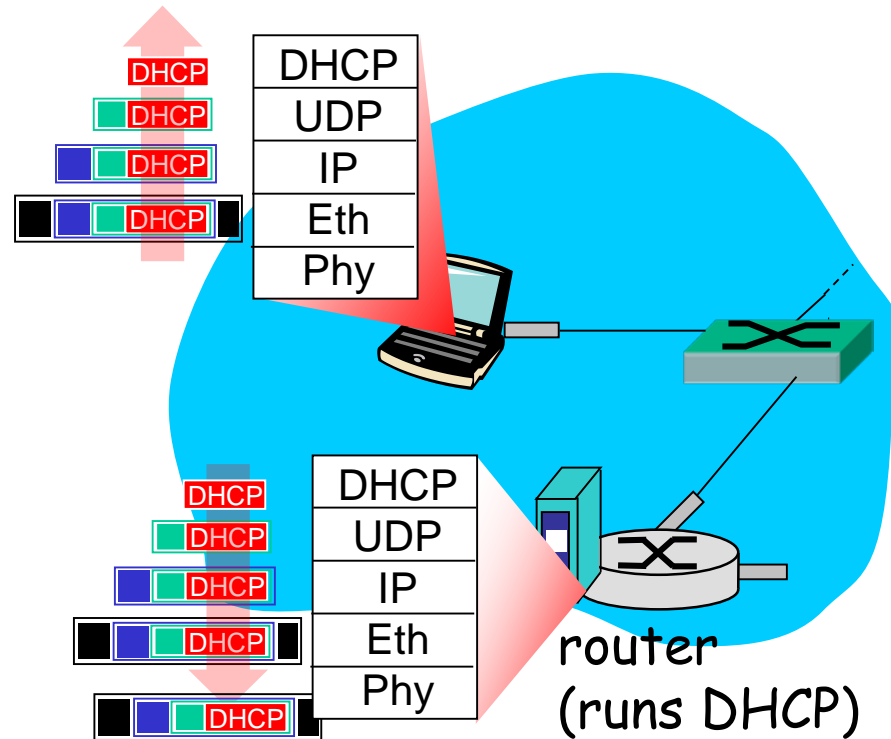


# A day in the life... connecting to the Internet



- ❖ 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 **802.3** Ethernet
- ❖ Ethernet frame **broadcast** (dest: FFFFFFFFFFFFFFFF) on LAN, received at router running **DHCP** server
- ❖ Ethernet **demuxed** to IP demuxed, UDP demuxed to DHCP

# A day in the life... connecting to the Internet

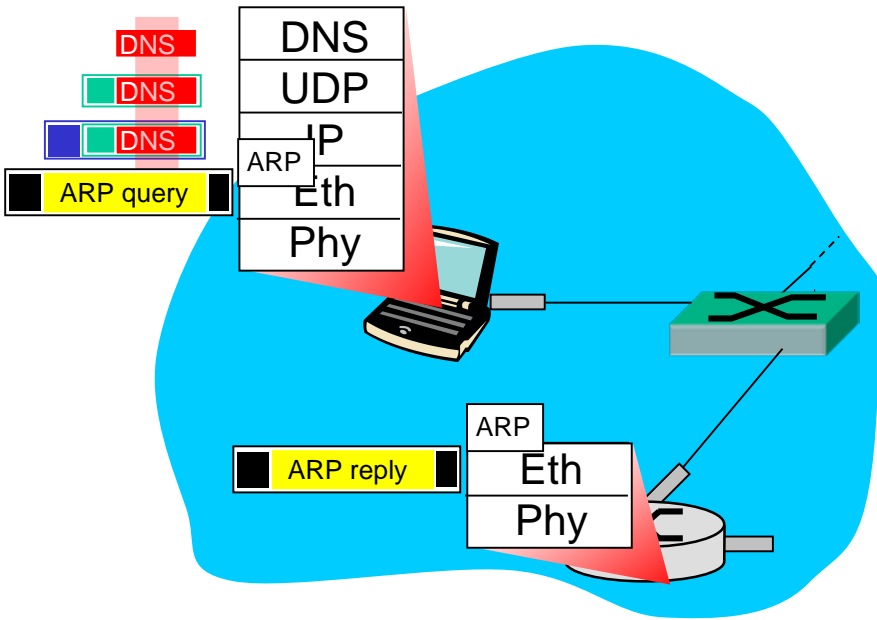


- ❖ DHCP server formulates **DHCP ACK** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- ❖ encapsulation at DHCP server, frame forwarded (**switch learning**) through LAN, demultiplexing at client
- ❖ DHCP client receives DHCP ACK reply

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

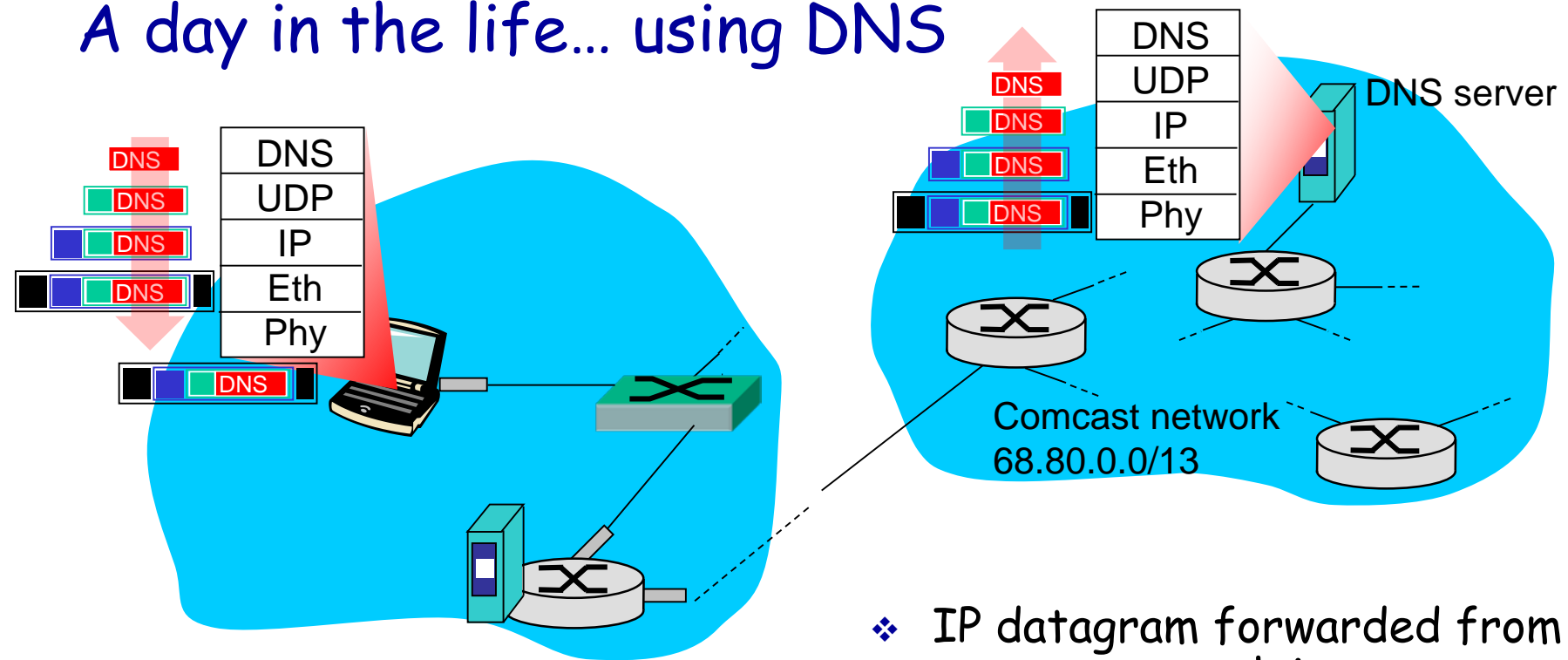


# A day in the life... ARP (before DNS, before HTTP)



- ❖ before sending **HTTP** request, need IP address of `www.google.com`: **DNS**
- ❖ DNS query created, encapsulated in UDP, encapsulated in IP, encapsulated in Eth. In order 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 first hop router, so can now send frame containing DNS query

# A day in the life... using DNS

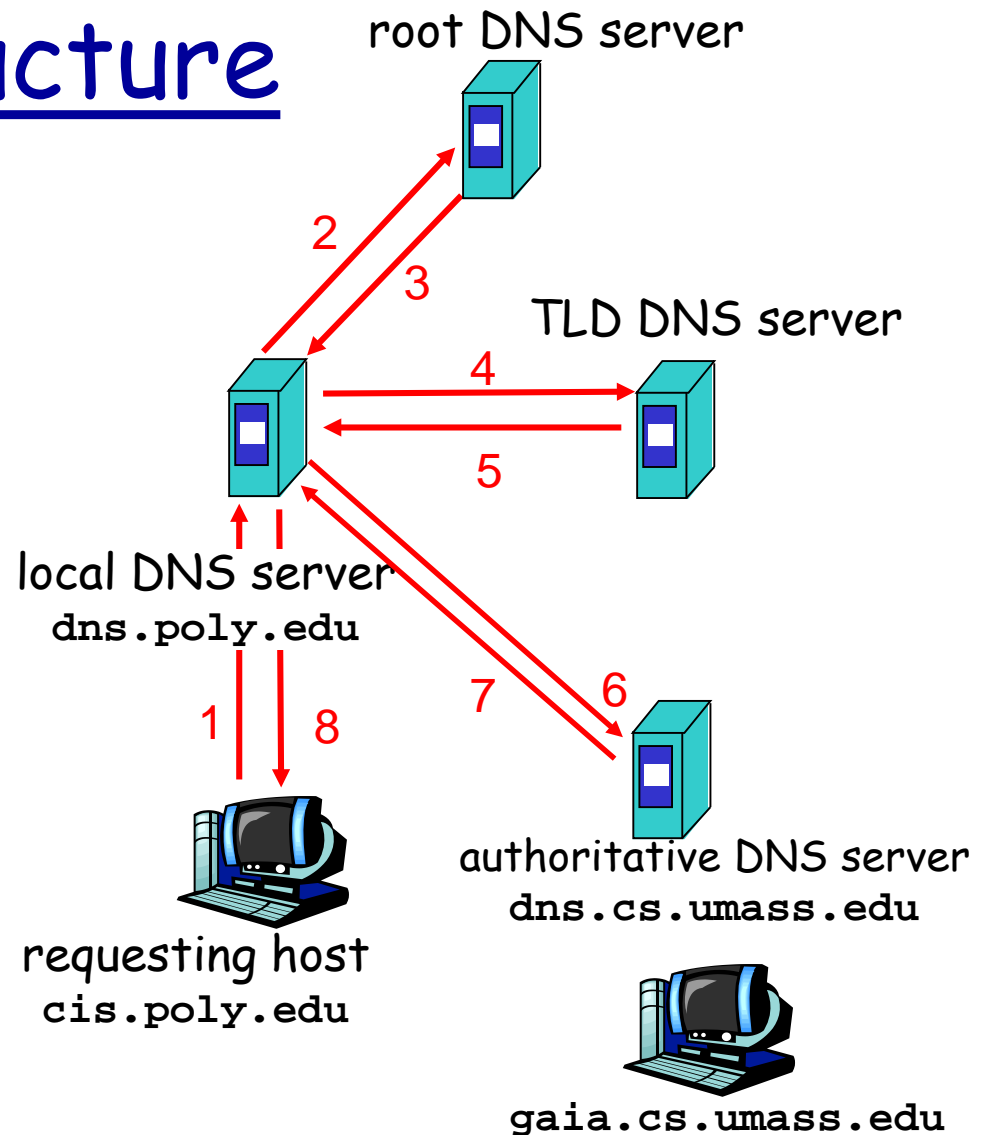


- ❖ IP datagram containing DNS query forwarded via LAN switch from client to 1<sup>st</sup> hop router

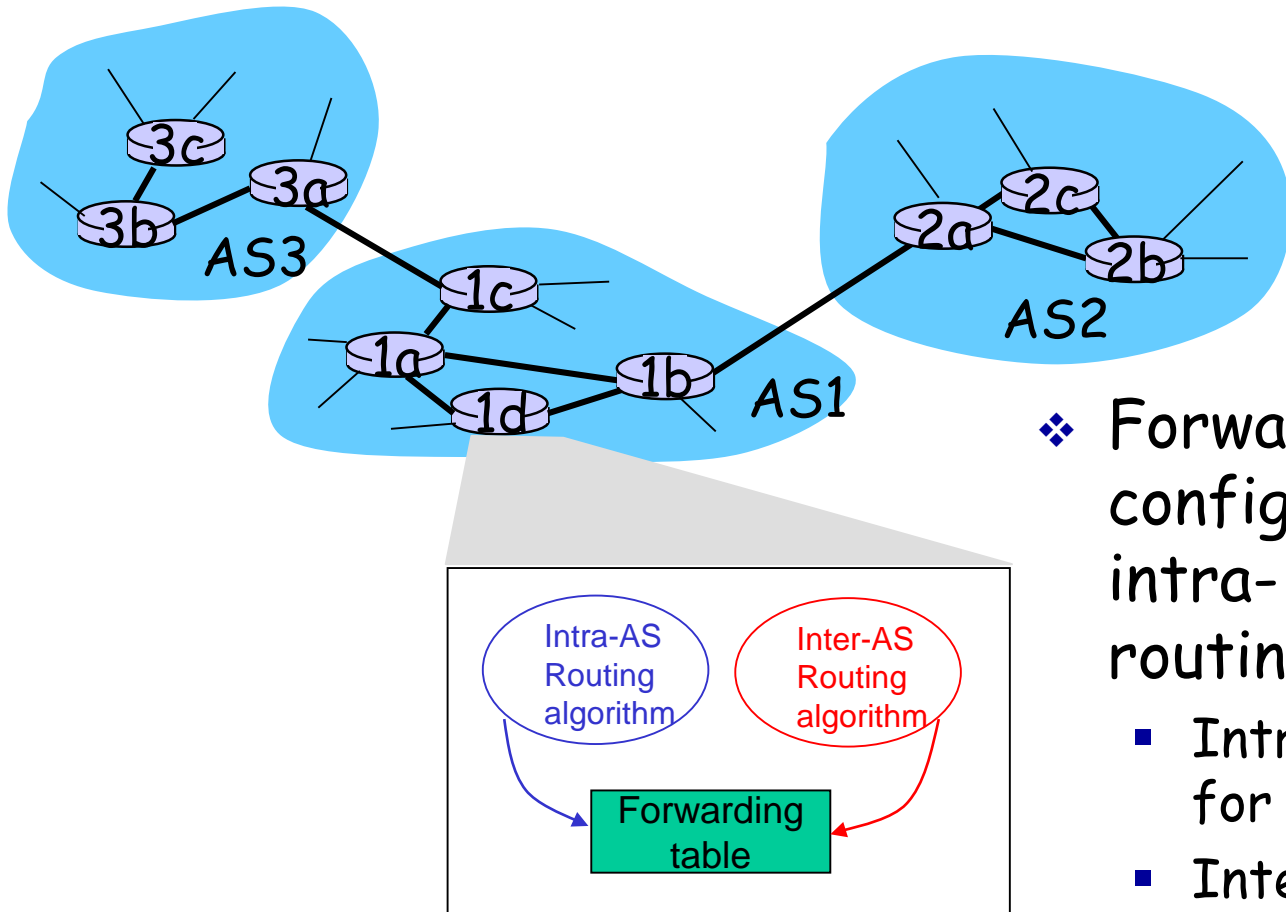
- ❖ IP datagram forwarded from campus network into comcast network, routed (tables created by **RIP**, **OSPF**, **IS-IS** and/or **BGP** routing protocols) to DNS server
- ❖ demuxed to DNS server
- ❖ DNS server replies to client with IP address of **www.google.com**

# DNS Infrastructure

- ❖ Host at cis.poly.edu wants IP address for gaia.cs.umass.edu
- ❖ Infrastructure:
  - Client resolver
  - Local DNS server
  - Authoritative DNS Server
  - Root DNS Server
  - Top-Level Domain DNS Server

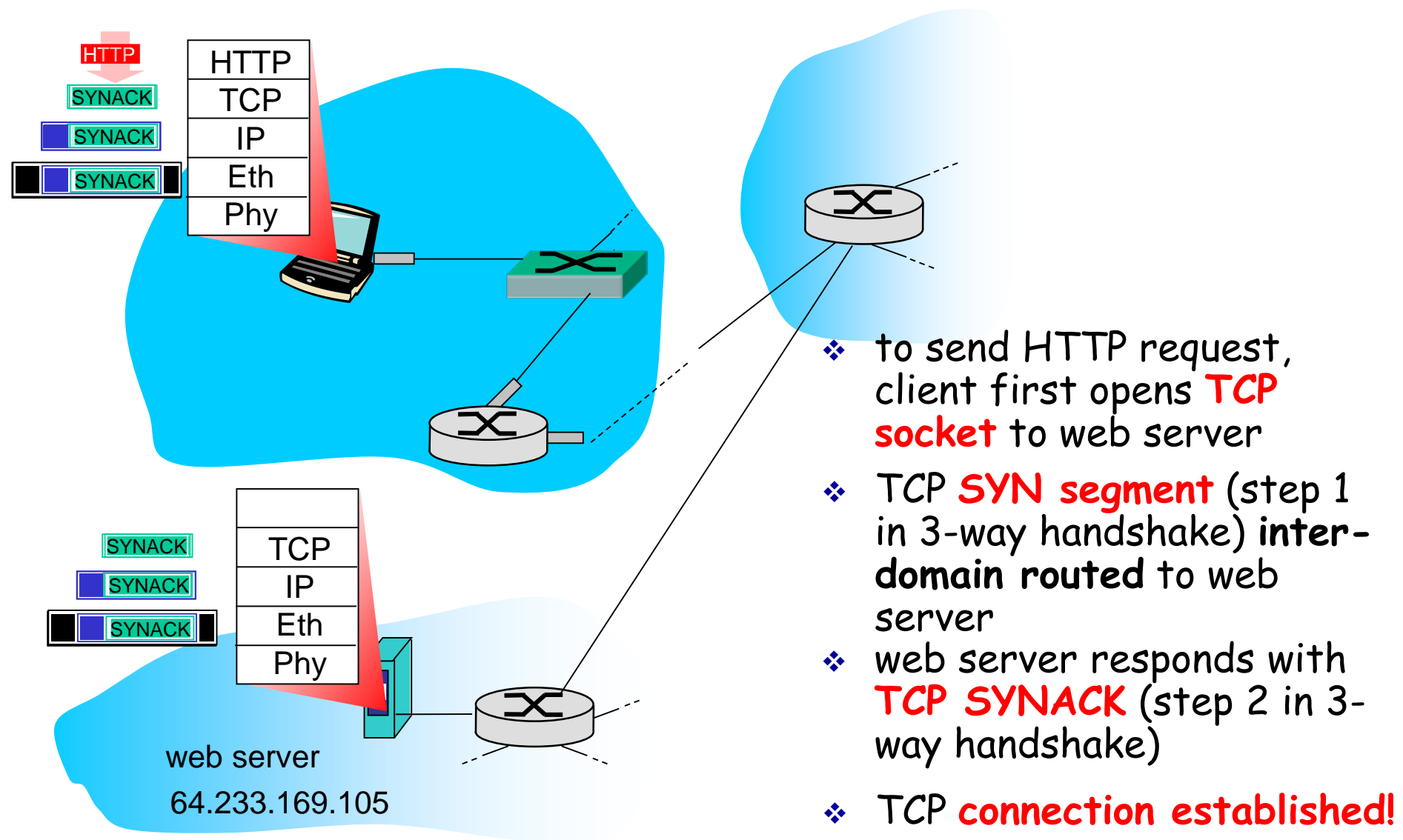


# Routing and Forwarding



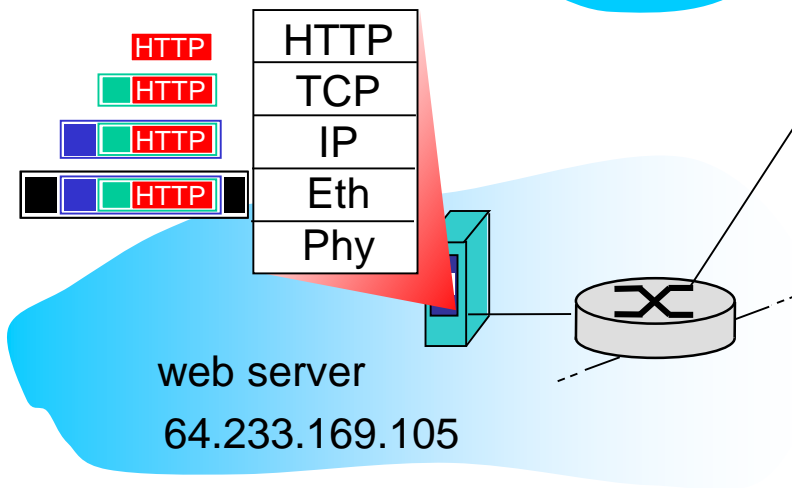
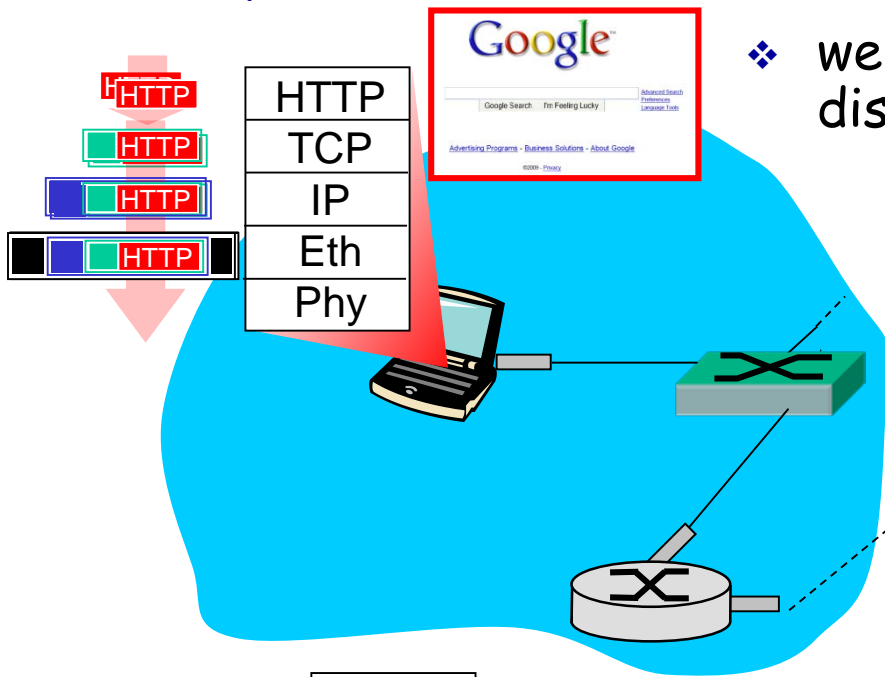
- ❖ Forwarding table is configured by both intra- and inter-AS routing algorithm
  - Intra-AS sets entries for internal dests
  - Inter-AS & Intra-As sets entries for external dests

# A day in the life... TCP connection carrying HTTP



# A day in the life... HTTP request/reply

❖ web page **finally (!!!)** displayed

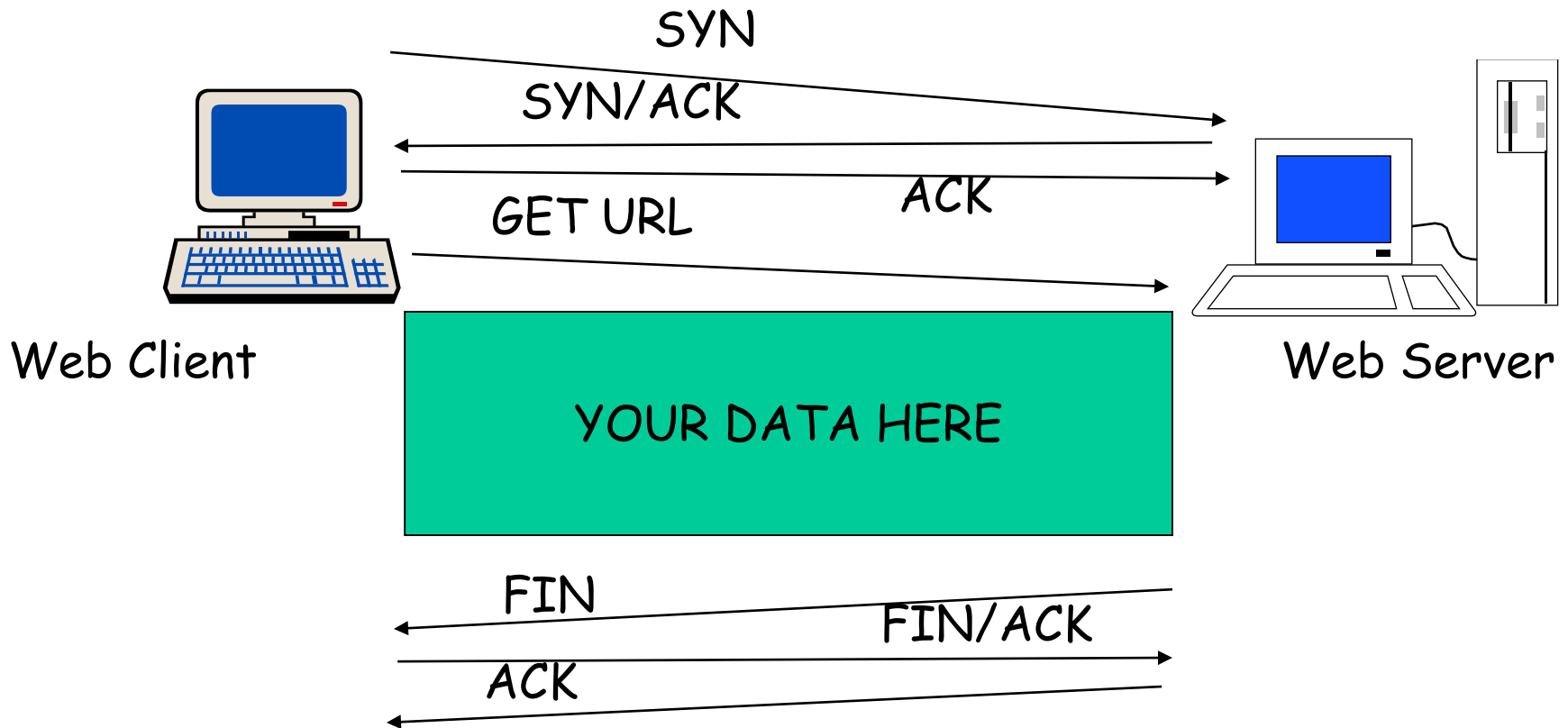


web server  
64.233.169.105

- ❖ **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

# Network View: HTTP and TCP

- ❖ TCP is a connection-oriented protocol



# How do we avoid sending too much for the receiver and network to handle?





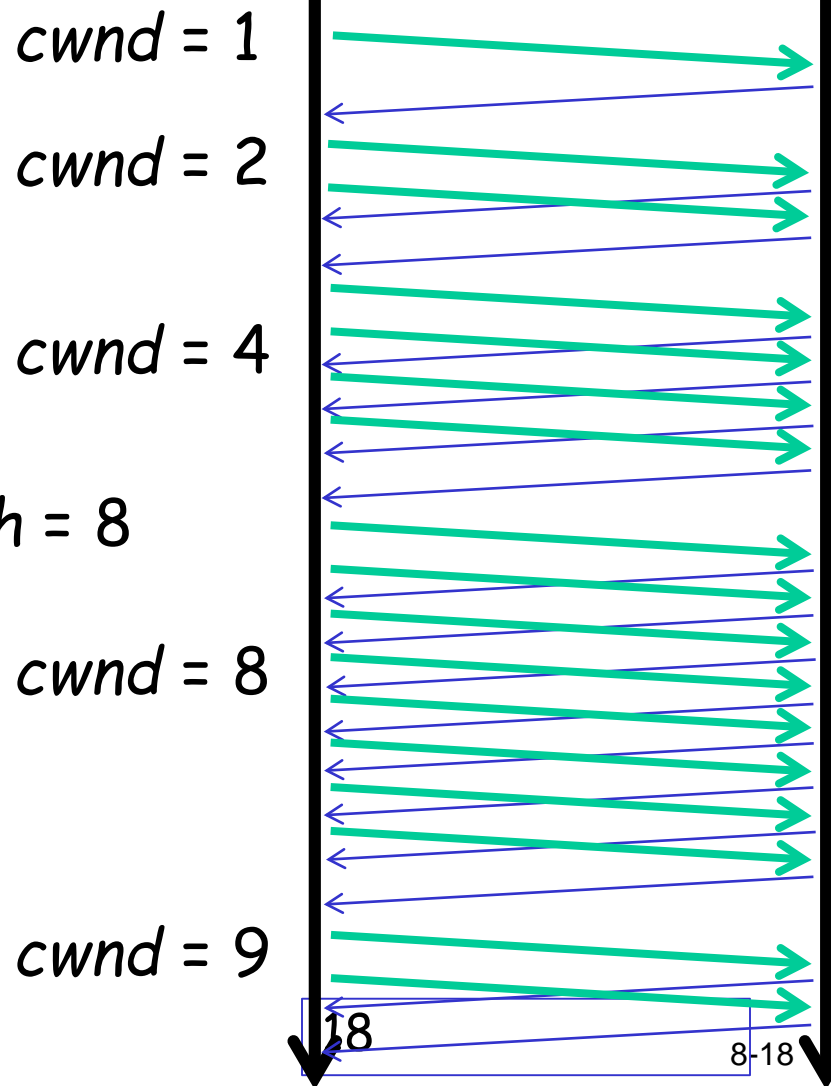
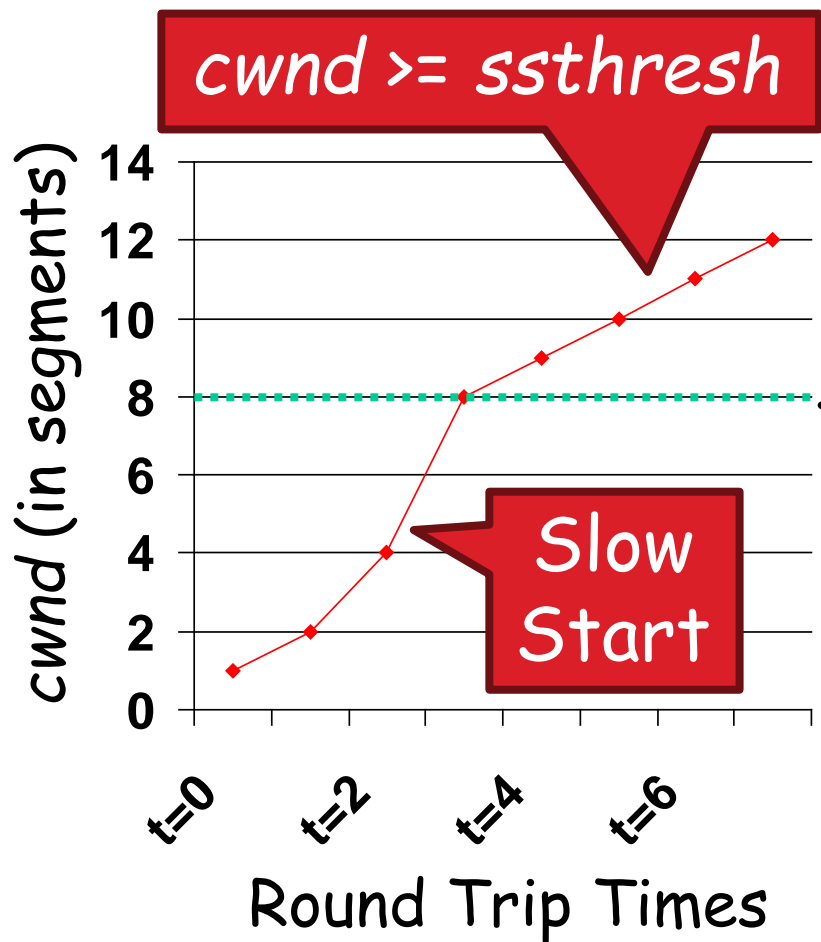
# TCP Tahoe: Summary

## ❖ Basic ideas

- Gently probe network for spare capacity
- Drastically reduce rate on congestion
- Windowing: self-clocking
- Other functions: round trip time estimation, error recovery

```
for every ACK {  
    if ( $W < ssthresh$ ) then  $W++$            (SS)  
    else  $W += 1/W$                           (CA)  
  
}  
for every loss {  
     $ssthresh = W/2$   
     $W = 1$   
}
```

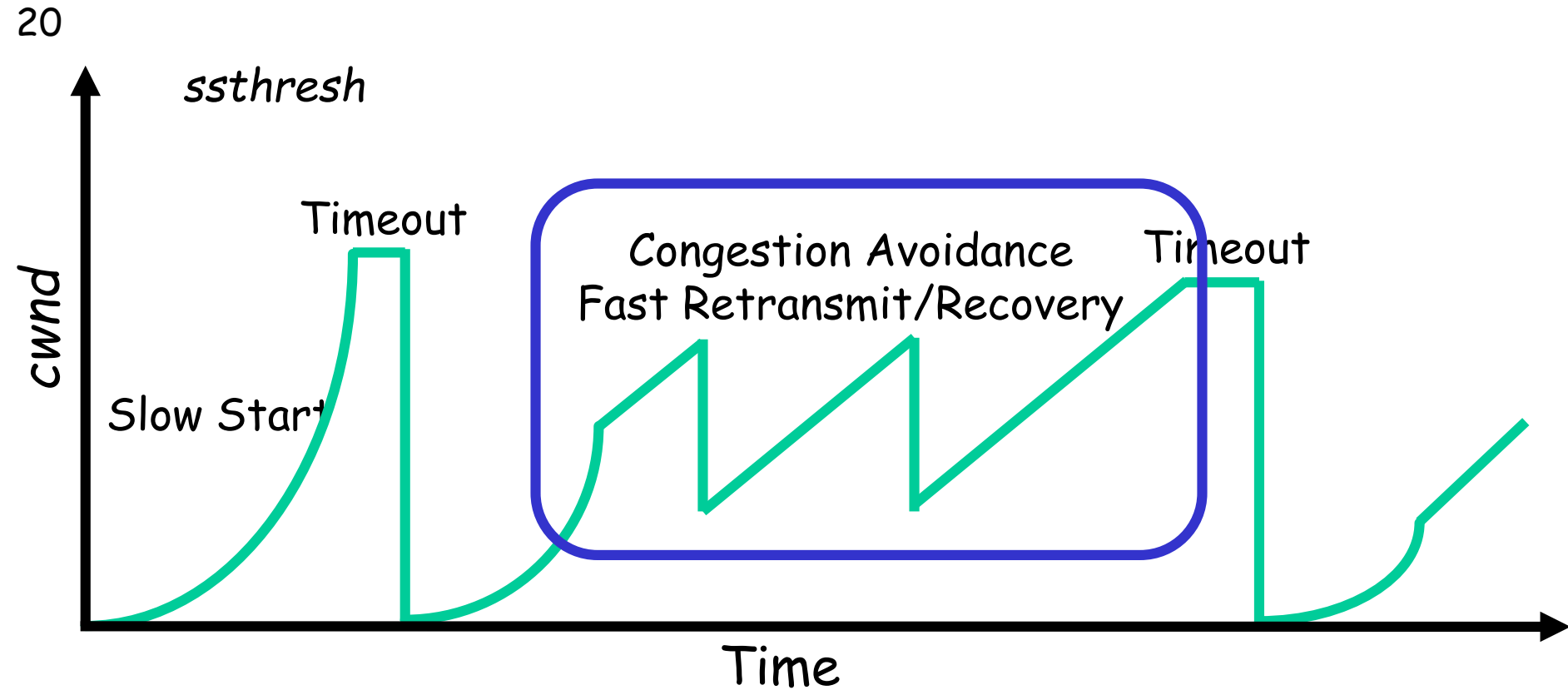
# Congestion Avoidance Example



# TCP Reno: Summary

- ❖ Fast Recovery along with Fast Retransmit used to avoid slow start
- ❖ On 3 duplicate ACKs
  - Fast retransmit and fast recovery
- ❖ On timeout
  - Retransmit last segment and slow start

# Fast Retransmit and Fast Recovery



- ❖ At steady state,  $cwnd$  oscillates around the optimal window size
- ❖ TCP always forces packet drops

# Example Web Page

## Harry Potter Movies

As you all know,  
the new HP book  
will be out in June  
and then there will  
be a new movie  
shortly after that...



hpface.jpg

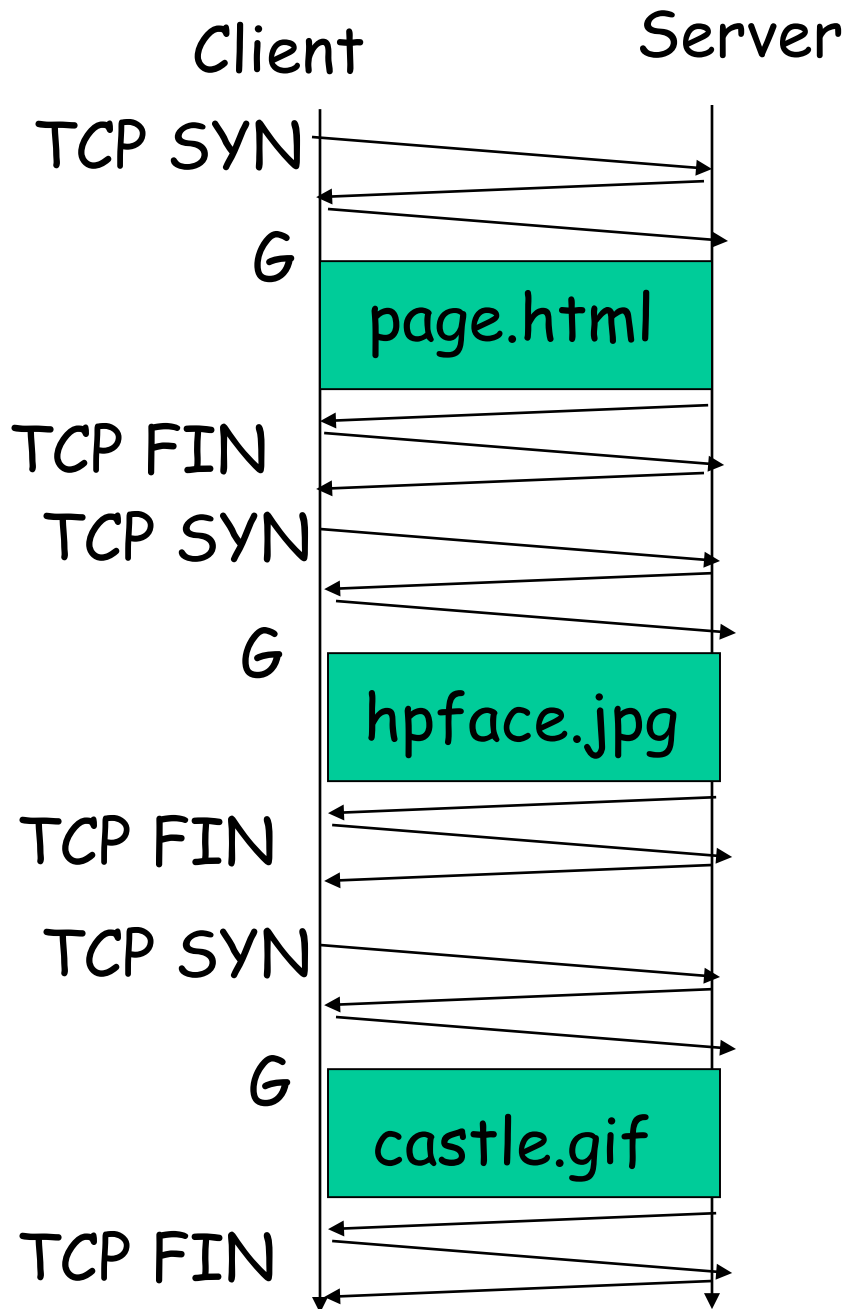
"Harry Potter and  
the Bathtub Ring"



castle.gif

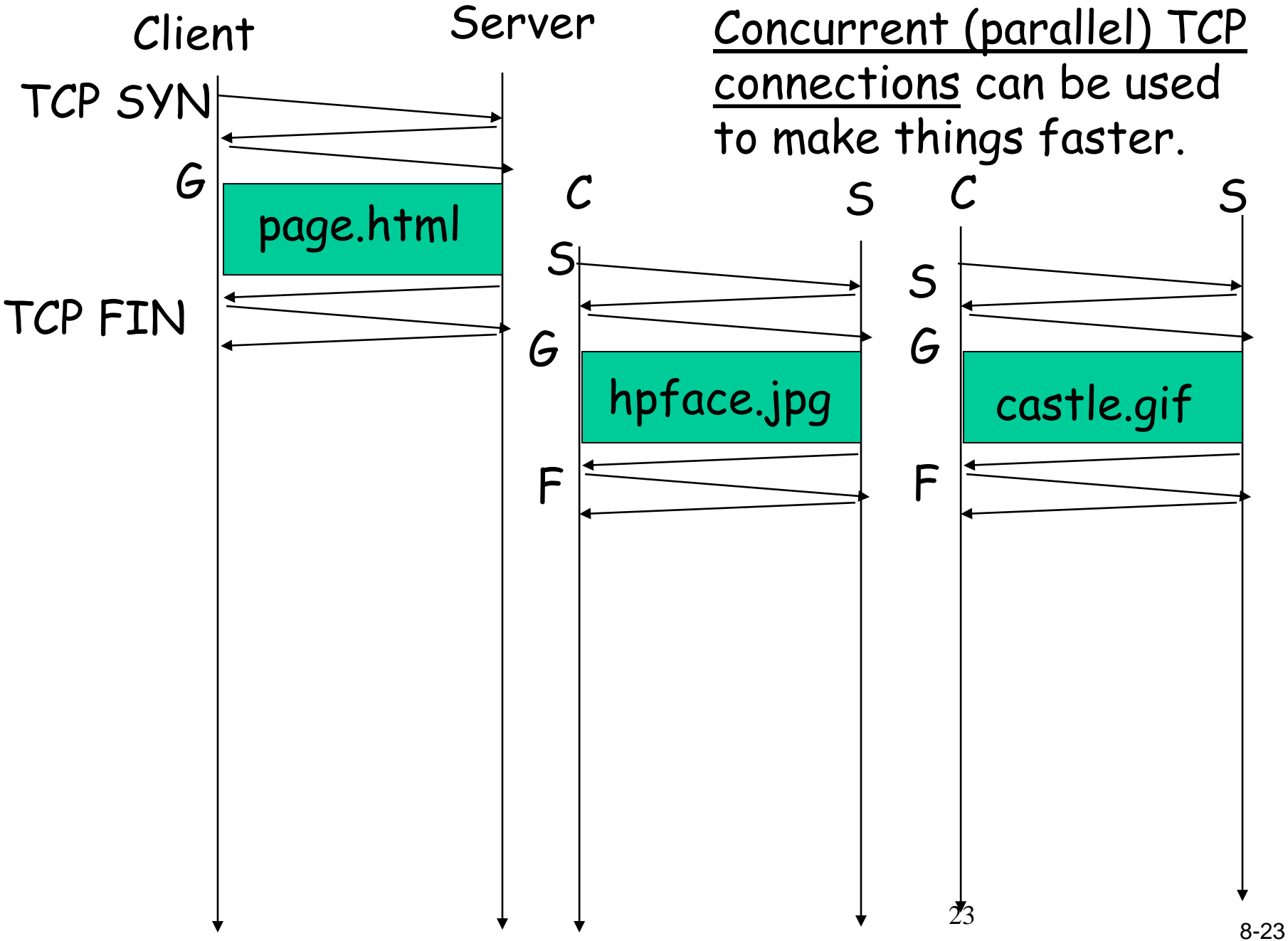
page.html

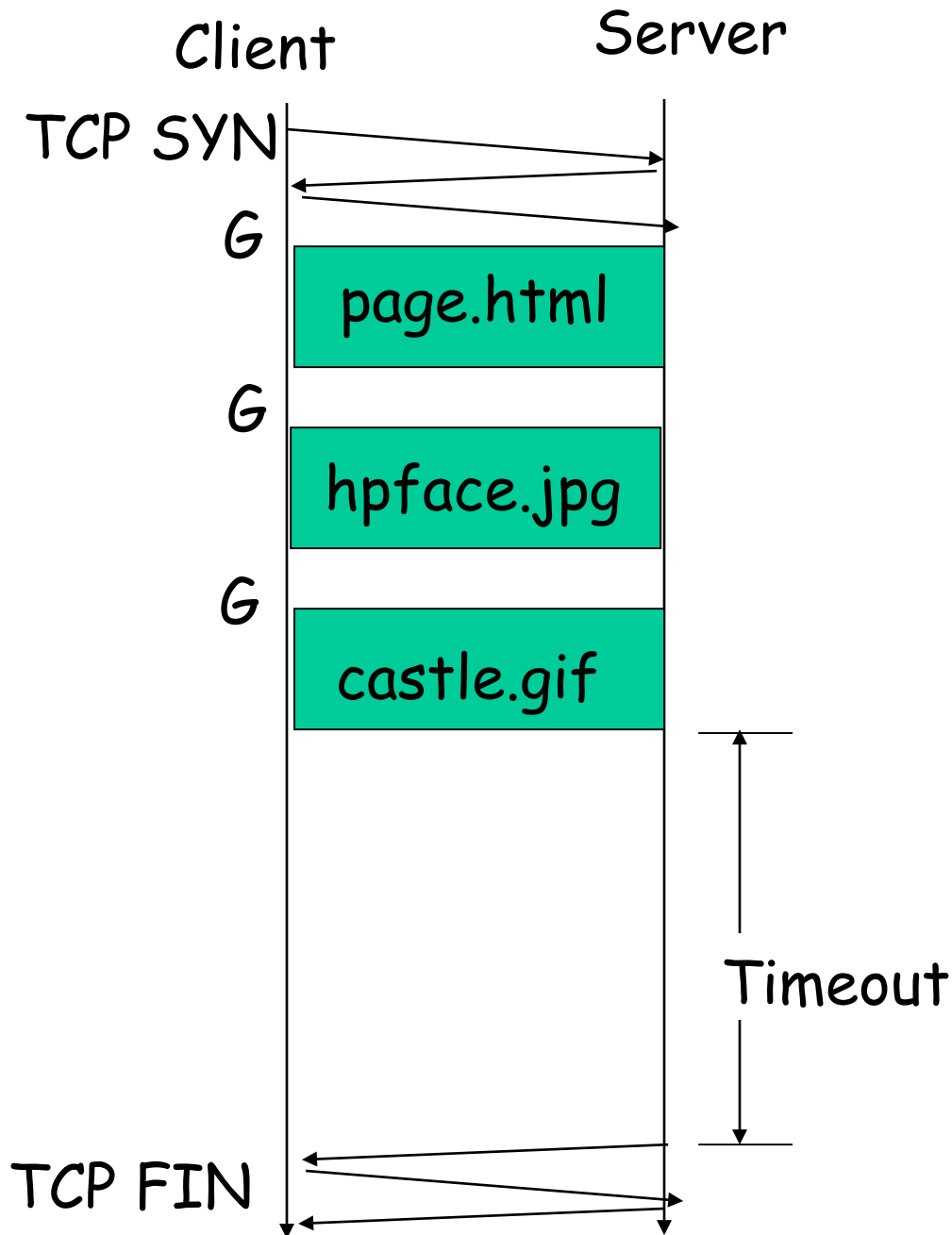




The "classic" approach in HTTP/1.0 is to use one HTTP request per TCP connection, serially.

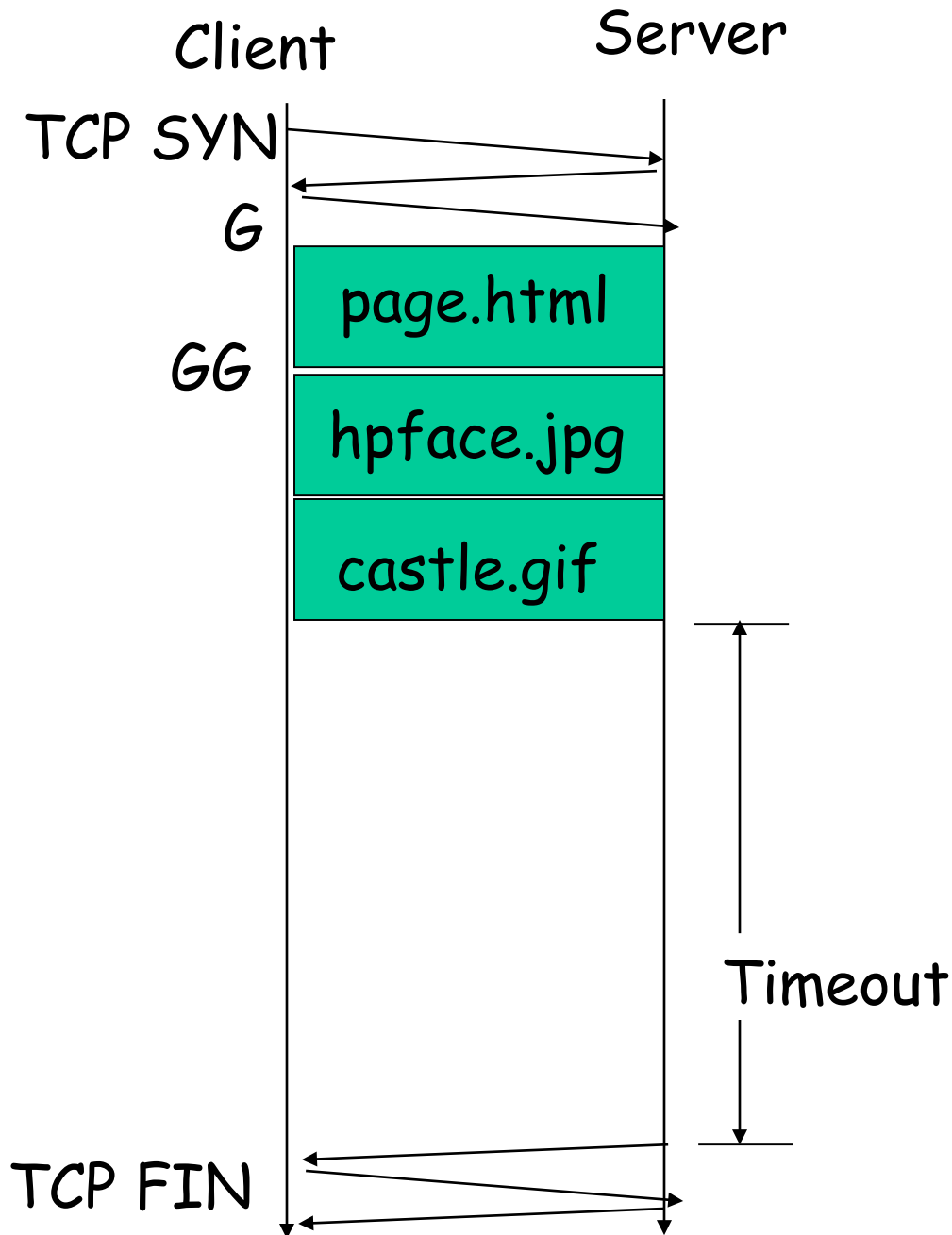
Concurrent (parallel) TCP connections can be used to make things faster.





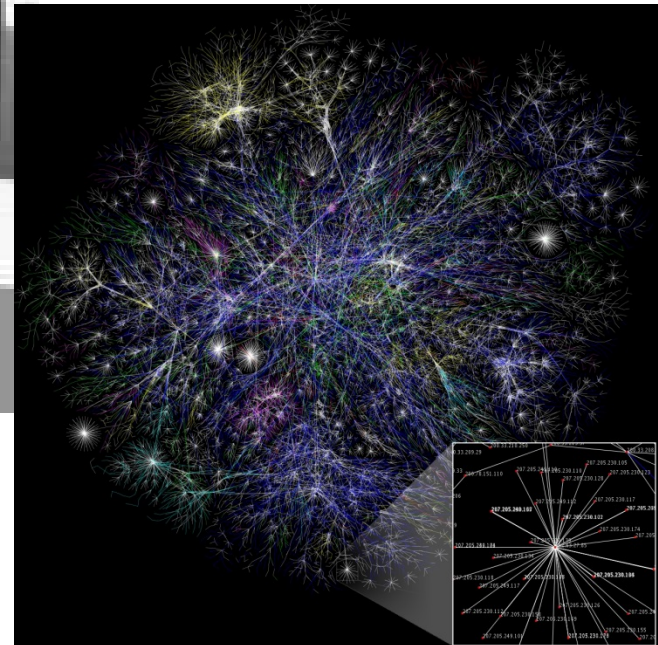
The "persistent HTTP" approach can re-use the same TCP connection for Multiple HTTP transfers, one after another, serially. Amortizes TCP overhead, but maintains TCP state longer at server.





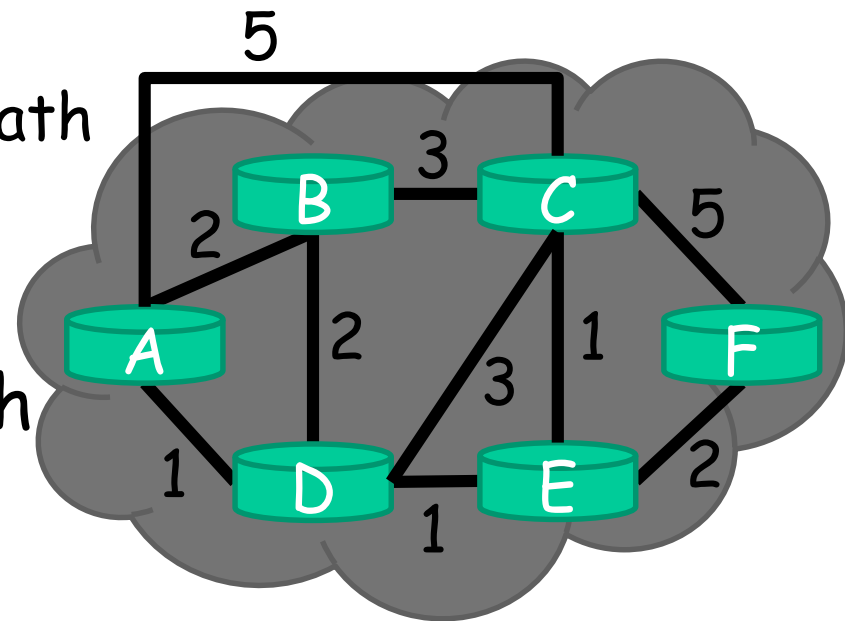
The "pipelining" feature in HTTP/1.1 allows requests to be issued asynchronously on a persistent connection. Requests must be processed in proper order. Can do clever packaging.

# How do we find a path?



# Routing on a Graph

- ❖ Goal: determine a “good” path through the network from source to destination
- ❖ What is a good path?
  - Usually means the shortest path
  - Load balanced
  - Lowest \$\$\$ cost
- ❖ Network modeled as a graph
  - Routers → nodes
  - Link → edges
    - Edge cost: delay, congestion level, etc.

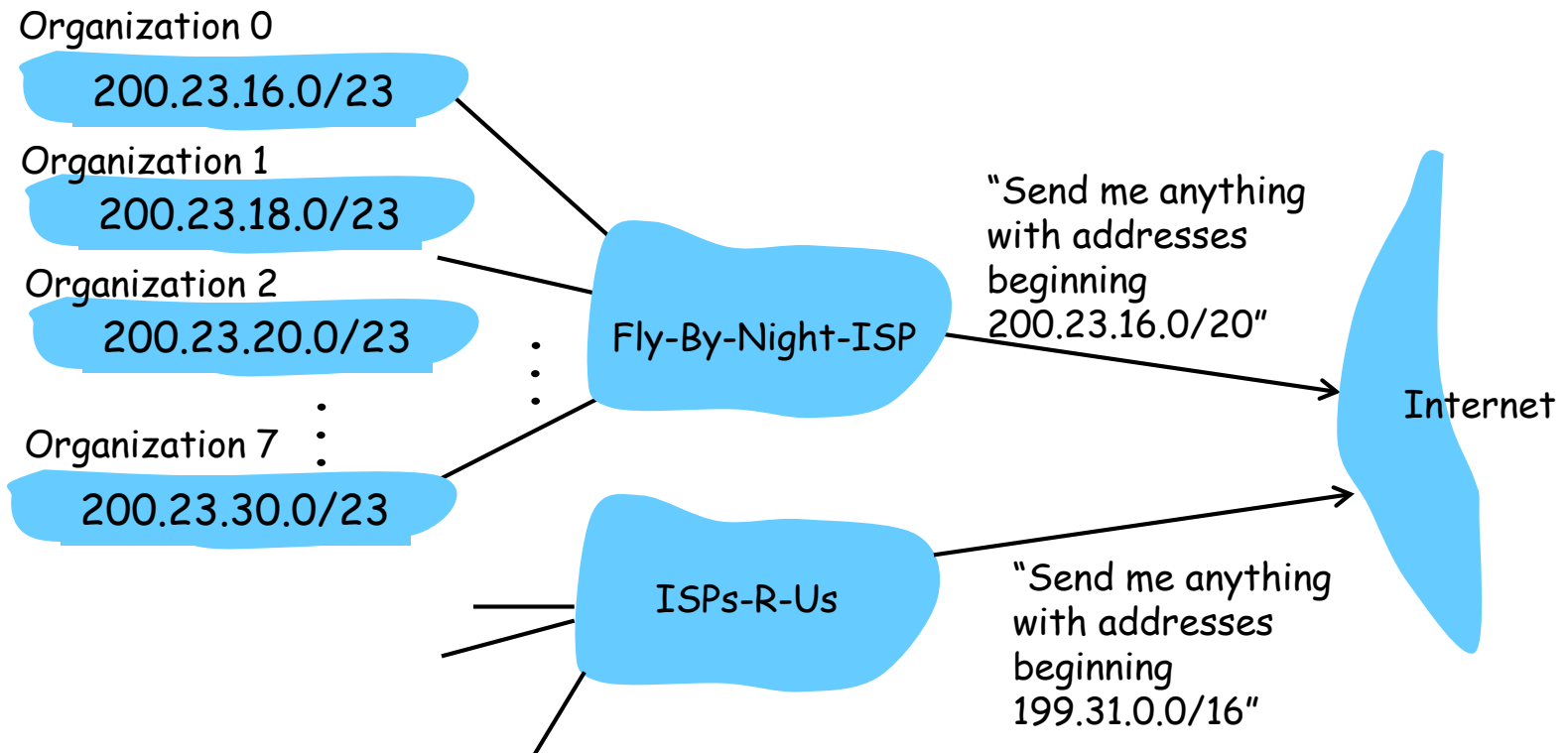


# Intra-domain Routing Protocols

- ❖ Distance vector
  - Routing Information Protocol (RIP), based on Bellman-Ford
  - Routers periodically exchange reachability info with neighbors
- ❖ Link state
  - Open Shortest Path First (OSPF), based on Dijkstra
  - Each network periodically **floods** neighbor information to all routers
  - Routers locally compute routes

# Hierarchical addressing: route aggregation

ISP has an address block; it can further divide this block into sub blocks and assign them to subscriber organizations.

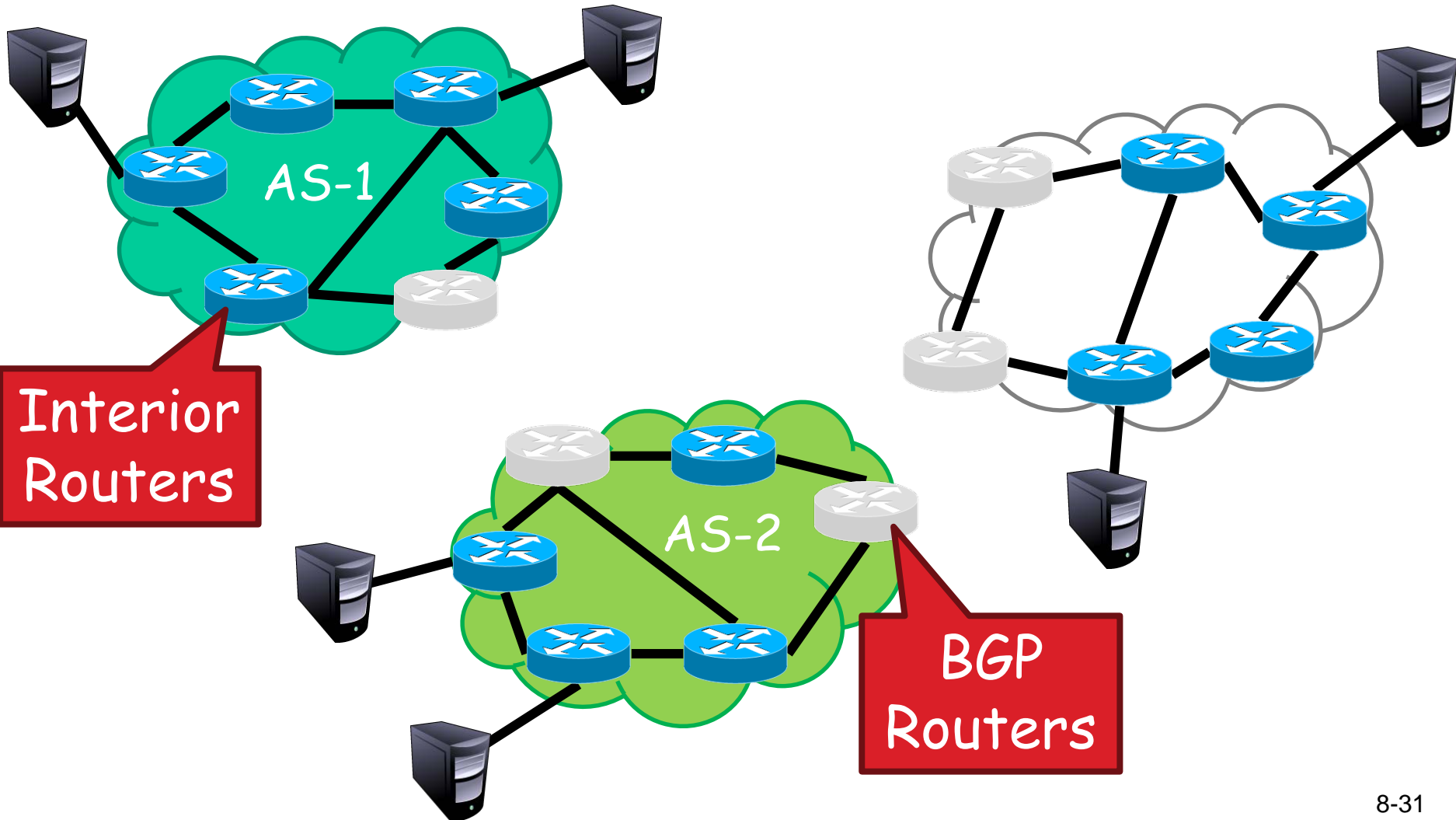


# Example CIDR Routing Table

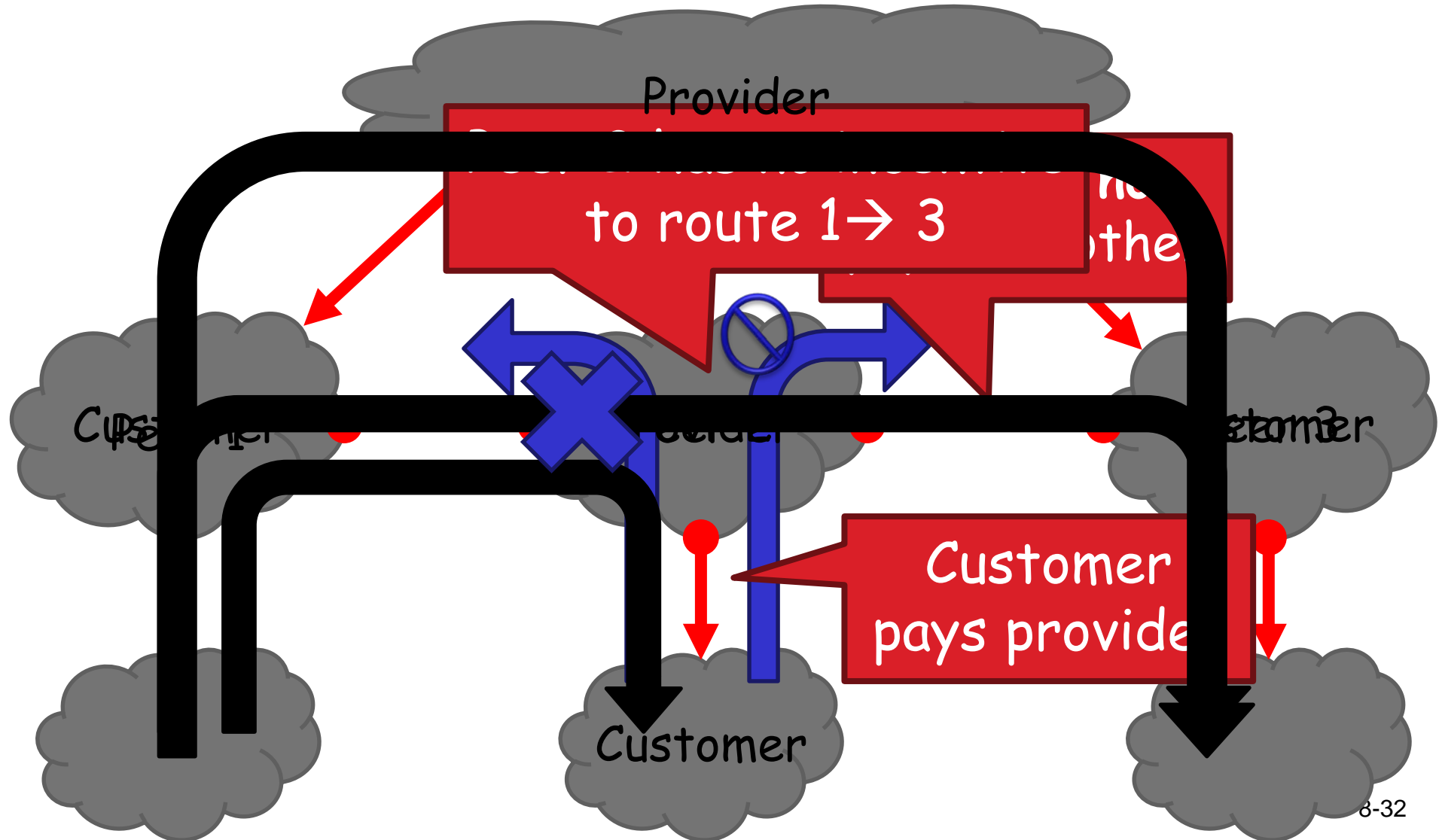
Address	Netmask	Third Byte	Byte Range
207.46.0.0	19	000xxxxx	0 - 31
207.46.32.0	19	001xxxxx	32 - 63
207.46.64.0	19	010xxxxx	64 - 95
207.46.128.0	18	10xxxxxx	128 - 191
207.46.192.0	18	11xxxxxx	192 - 255

Hole in the Routing Table: No coverage for 96 - 127  
207.46.96.0/19

# Network of networks: BGP and ASes

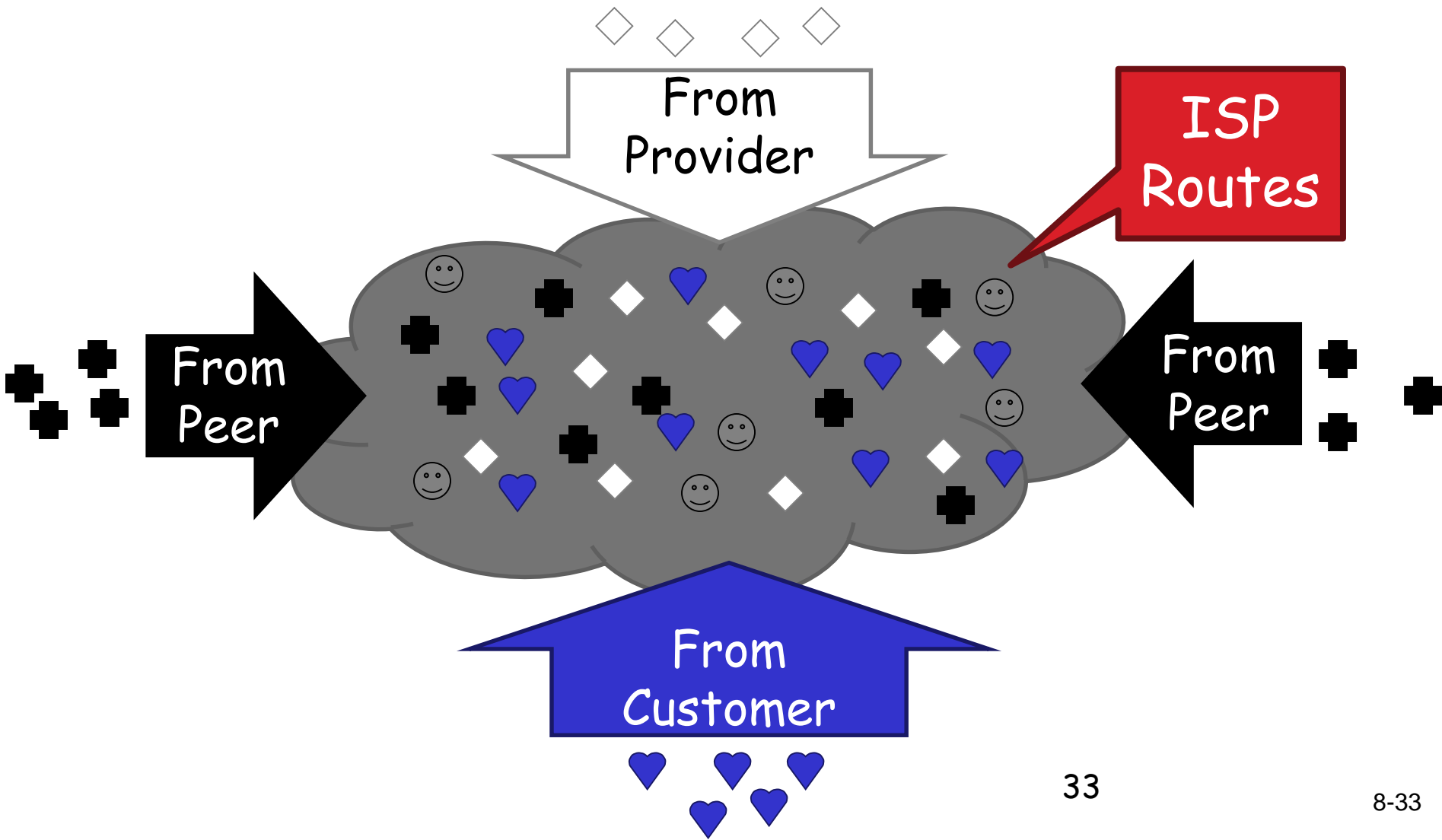


# BGP Relationships

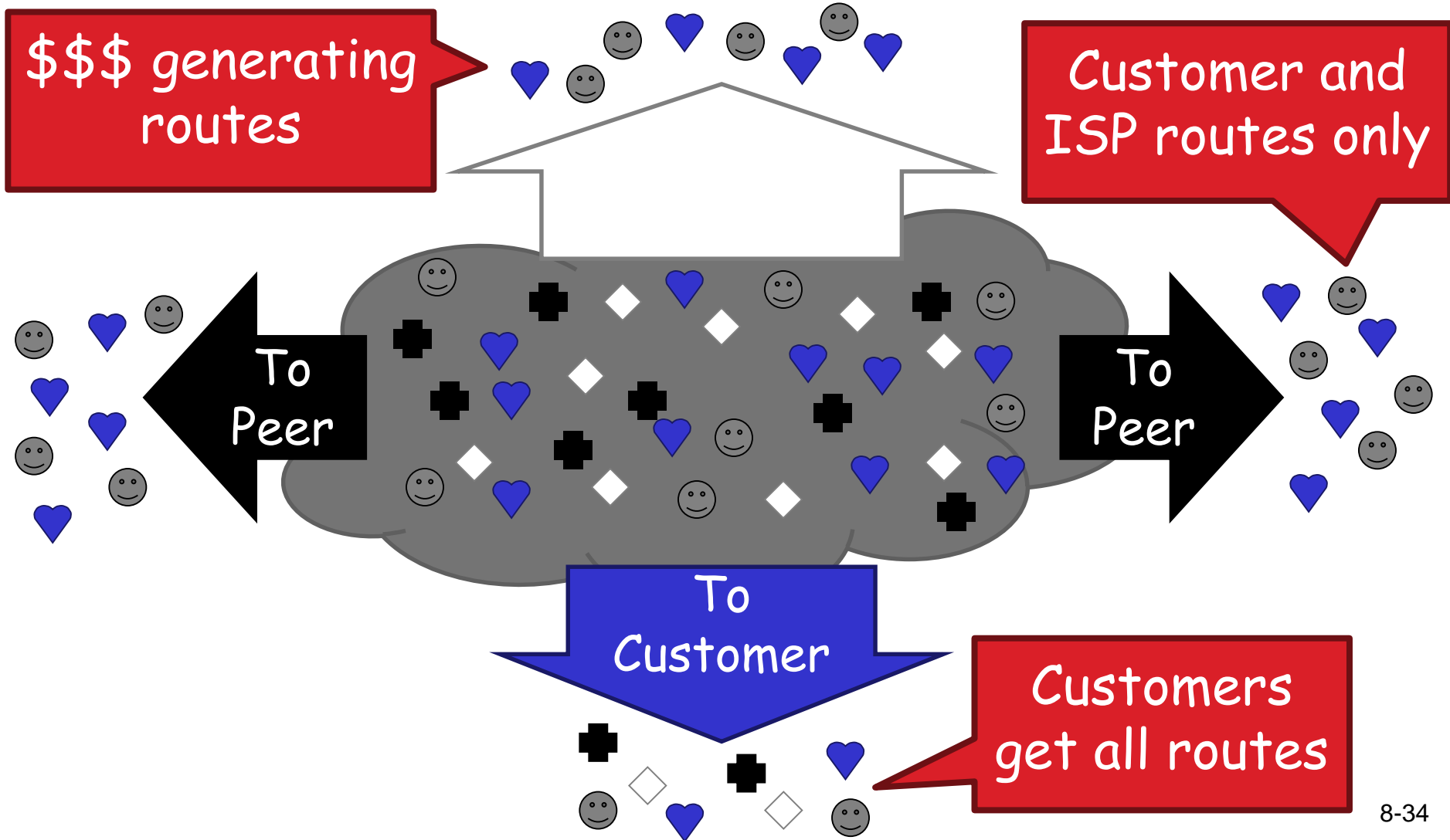




# Importing Routes



# Exporting Routes



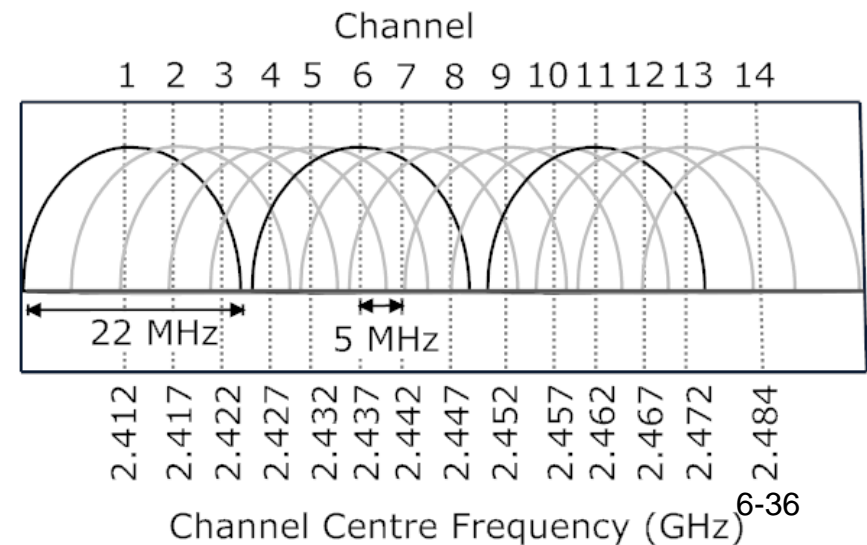
# Also cover many other topics ...

- ❖ For example ...
  - Wireless (Ch 7)
  - Multimedia networking (Ch 8)
  - Network security (Ch 9)

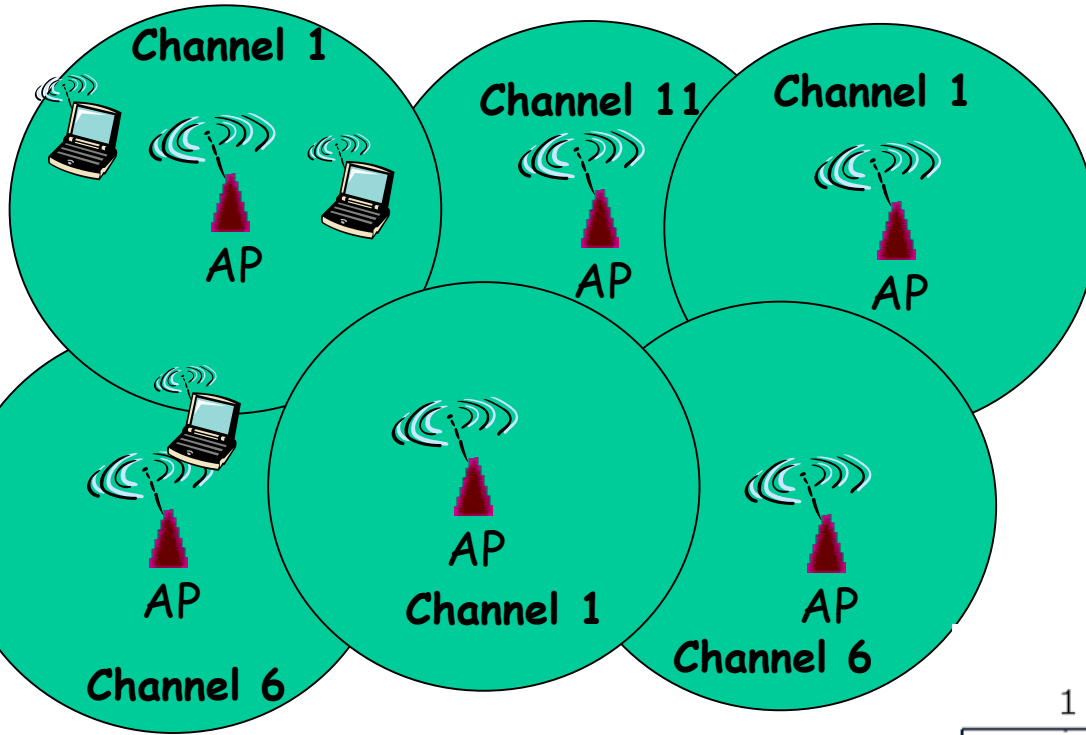
# Wireless Cells

- ❖ 802.11b has 11 channels
- ❖ Channels 1, 6, and 11 are non-overlapping

- ❖ Admin chooses frequency for AP

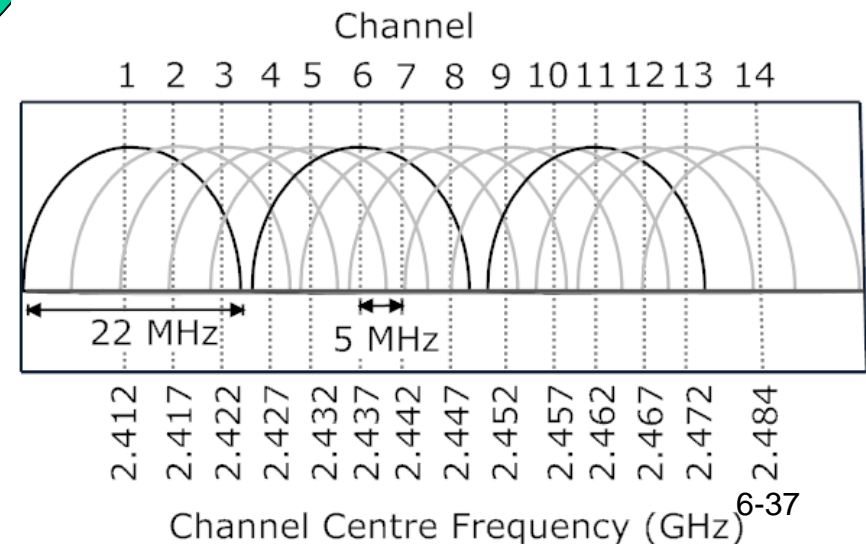


# Wireless Cells

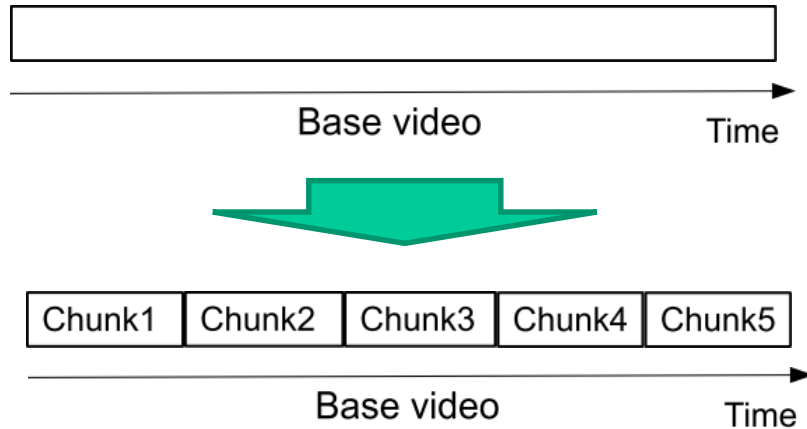


- ❖ 802.11b has 11 channels
- ❖ Channels 1, 6, and 11 are non-overlapping
- ❖ Each AP coverage area is called a "cell"
- ❖ Wireless nodes can roam between cells

- ❖ Admin chooses frequency for AP
- ❖ Interference possible: channel can be same as that chosen by neighboring AP!



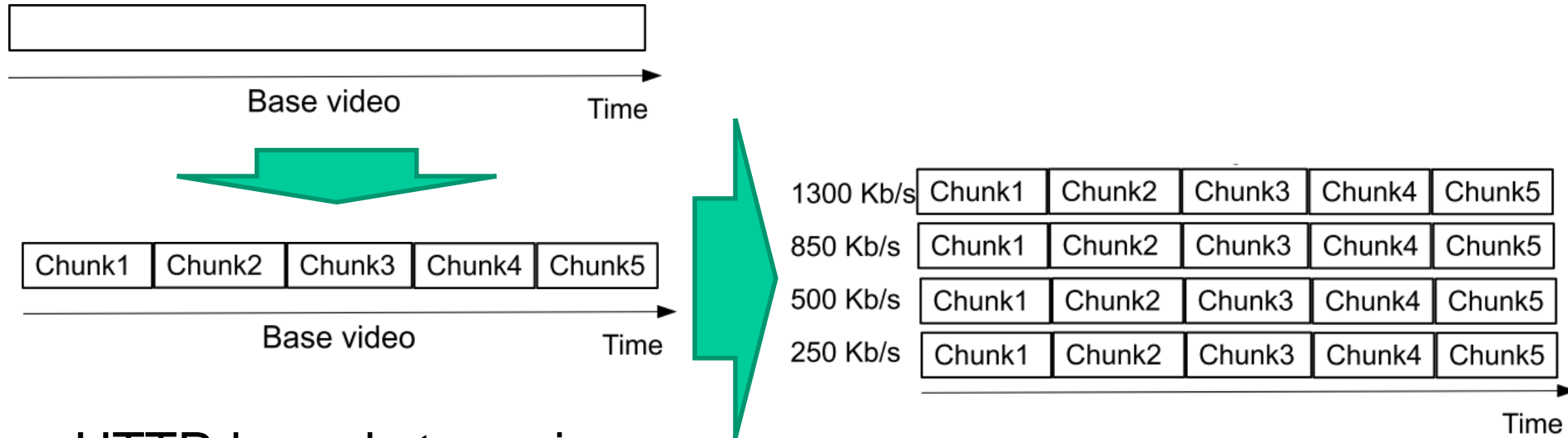
# HTTP-based Adaptive Streaming (HAS)



- ❖ HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD



# HTTP-based Adaptive Streaming (HAS)



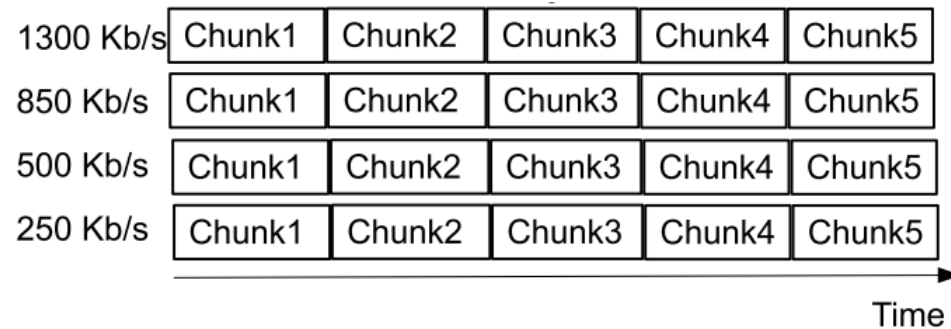
## ❖ HTTP-based streaming

- Video is split into chunks
- Easy firewall traversal and caching
- Easy support for interactive VoD

## ❖ HTTP-based **adaptive** streaming

- **Multiple encodings of each chunk (defined in manifest file)**
-

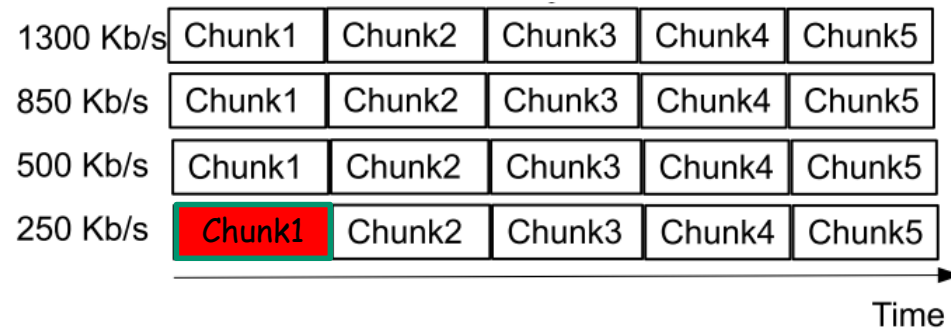
# HTTP-based Adaptive Streaming (HAS)



- ❖ HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD
- ❖ HTTP-based **adaptive** streaming
  - Multiple encodings of each chunk (defined in manifest file)
  -



# HTTP-based Adaptive Streaming (HAS)



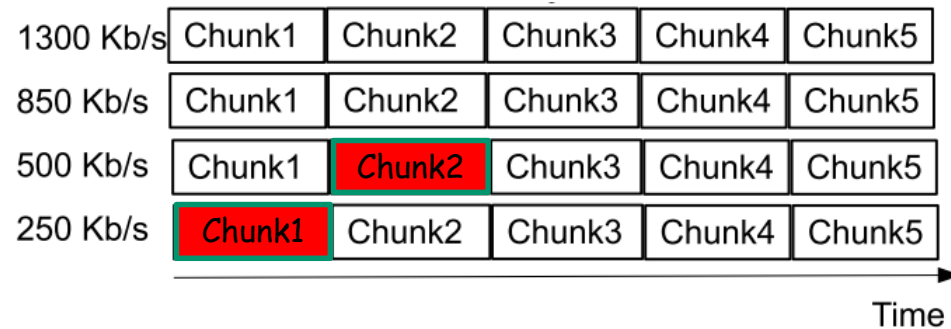
## ❖ HTTP-based streaming

- Video is split into chunks
- Easy firewall traversal and caching
- Easy support for interactive VoD

## ❖ HTTP-based **adaptive** streaming

- Multiple encodings of each chunk (defined in manifest file)
- **Clients adapt quality encoding based on buffer/network conditions**

# HTTP-based Adaptive Streaming (HAS)



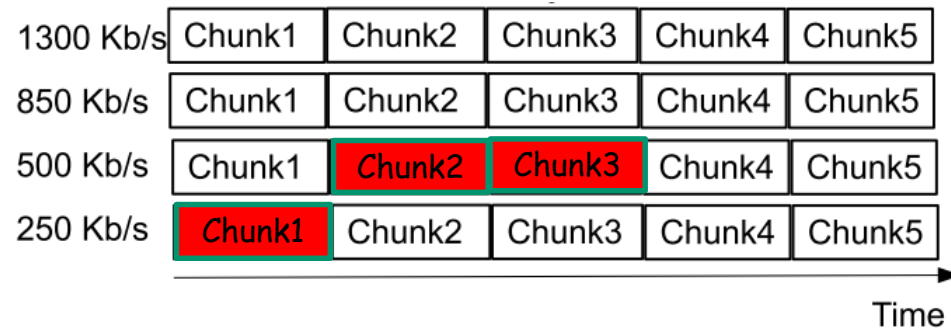
## ❖ HTTP-based streaming

- Video is split into chunks
- Easy firewall traversal and caching
- Easy support for interactive VoD

## ❖ HTTP-based **adaptive** streaming

- Multiple encodings of each chunk (defined in manifest file)
- **Clients adapt quality encoding based on buffer/network conditions**

# HTTP-based Adaptive Streaming (HAS)



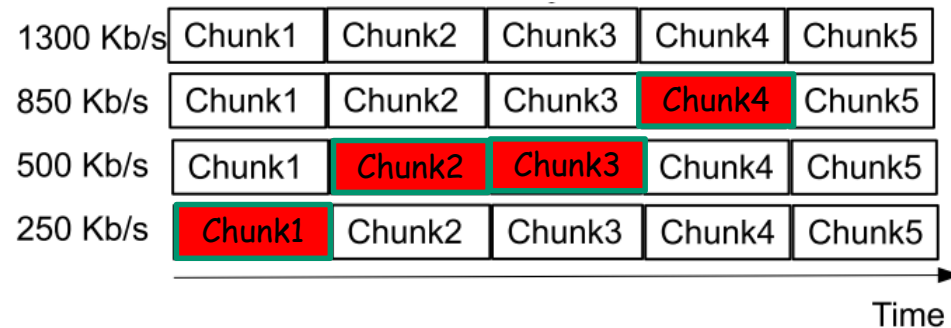
## ❖ HTTP-based streaming

- Video is split into chunks
- Easy firewall traversal and caching
- Easy support for interactive VoD

## ❖ HTTP-based **adaptive** streaming

- Multiple encodings of each chunk (defined in manifest file)
- **Clients adapt quality encoding based on buffer/network conditions**

# HTTP-based Adaptive Streaming (HAS)



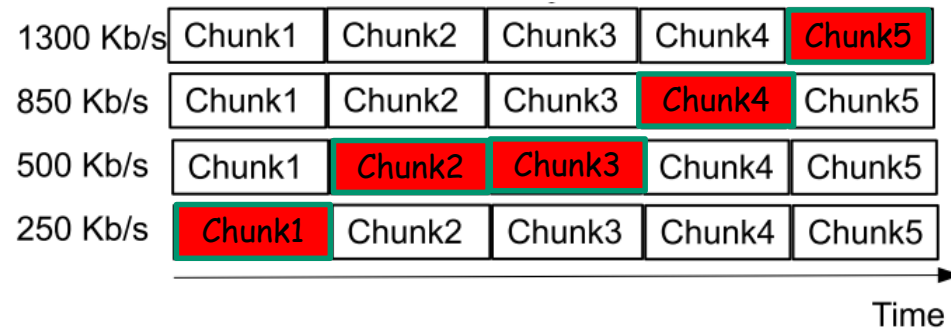
## ❖ HTTP-based streaming

- Video is split into chunks
- Easy firewall traversal and caching
- Easy support for interactive VoD

## ❖ HTTP-based **adaptive** streaming

- Multiple encodings of each chunk (defined in manifest file)
- **Clients adapt quality encoding based on buffer/network conditions**

# HTTP-based Adaptive Streaming (HAS)



## ❖ HTTP-based streaming

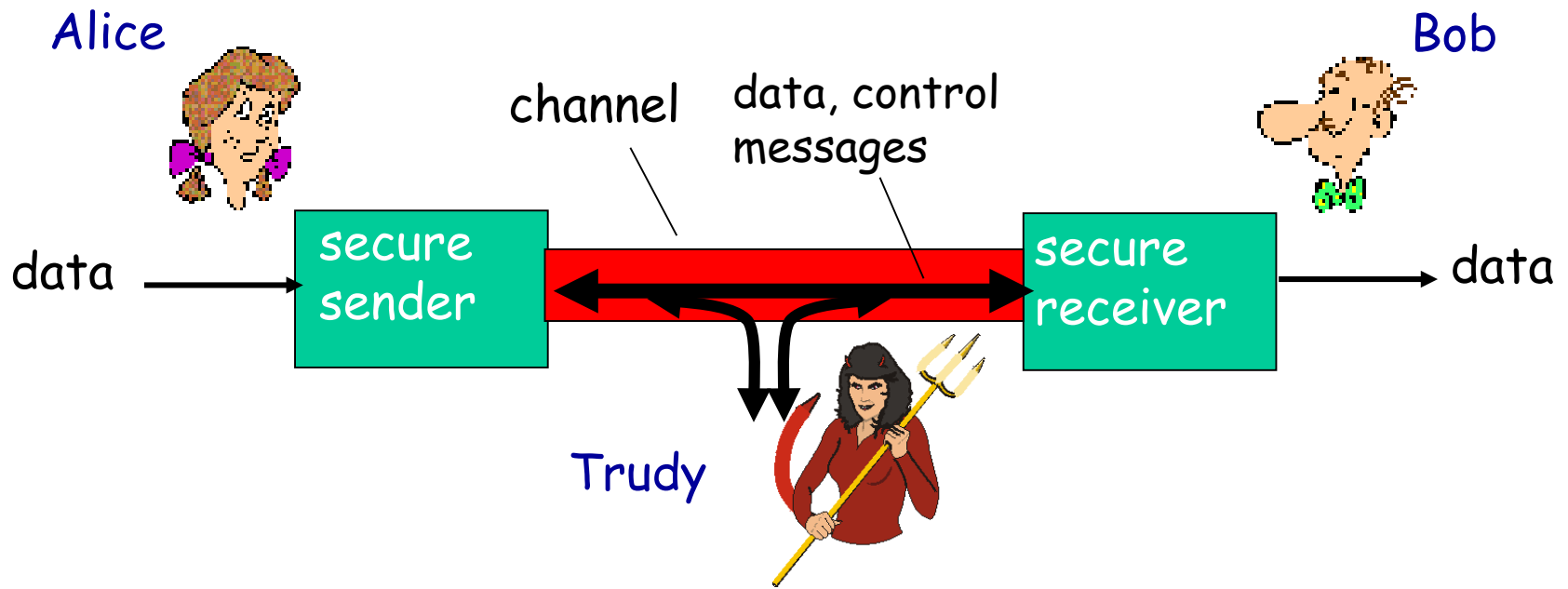
- Video is split into chunks
- Easy firewall traversal and caching
- Easy support for interactive VoD

## ❖ HTTP-based **adaptive** streaming

- Multiple encodings of each chunk (defined in manifest file)
- **Clients adapt quality encoding based on buffer/network conditions**

# Friends and enemies: Alice, Bob, Trudy

- ❖ well-known in network security world
- ❖ Bob, Alice (lovers!) want to communicate "securely"
- ❖ Trudy (intruder) may intercept, delete, add messages



# HTTPS (and TLS/SSL Cipher Suite)

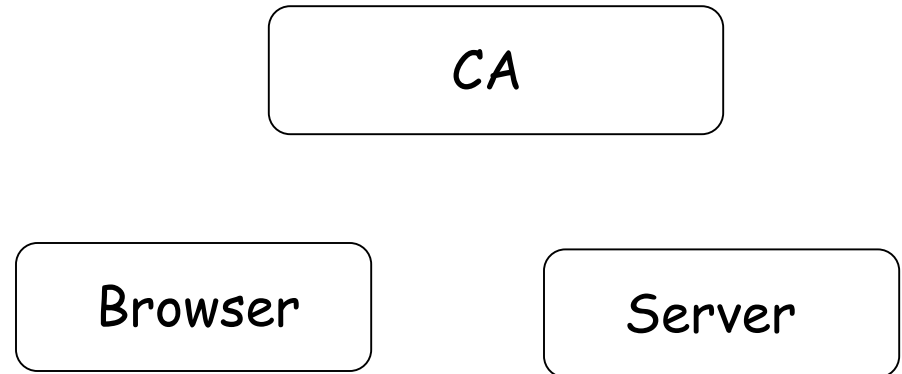
## ❖ cipher suite

- public-key algorithm
- symmetric encryption algorithm
- MAC algorithm

## ❖ TLS/SSL supports several cipher suites

## ❖ negotiation: client, server agree on cipher suite

- client offers choice
- server picks one







# Exam

- ❖ Read all instructions carefully
- ❖ Please explain how you derived your answers. Your final answers should be clearly stated.
- ❖ Write answers legibly; no marks will be given for answers that cannot be read easily.
- ❖ Where a discourse or discussion is called for, be concise and precise.
- ❖ No assistance: closed book, closed notes, and no electronics ...

# Exam (cnt.) ...

- ❖ If necessary, state any assumptions you made in answering a question. However, remember to read the instructions for each question carefully and answer the questions as precisely as possible. Solving the *wrong question* may result in deductions! It is better to solve the *right question incorrectly*, than the *wrong question correctly*.
- ❖ Please use English. (If needed, feel free to bring a dictionary from an official publisher. Hardcopy, not electronic!! Also, your dictionary is not allowed to contain any notes; only the printed text by the publisher.)

# Some example questions

- ❖ Note: Expect to see that you understand the material ... questions will therefore be given that test your understanding.
- ❖ Also, please see the old exams (that have been given by me). At a minimum, I expect that you by the exam understand the material well enough that you can solve these and similar questions ...
  - Including providing convincing arguments and explanatory figures



# Question: TCP

- ❖ Consider two machines A and B which are located 75ms apart. Assume that A is delivering a large file to B. Draw a figure that captures the send and receive events of the first 16 packets. You should consider three scenarios. (a) Assume no packet losses. (b) Assume that packets 8, 9, and 10 are lost. (c) Assume that packets 5, 6, and 7 are lost. You can assume that the TCP version is implementing fast retransmit and the timeout period is 250ms. For simplicity, you can also assume that the transmission rate is infinite.

# Question: Dijkstra's algorithm

- ❖ Consider the network in the graph below. Calculate the shortest path between A and D. (Please insert your own random graph topology here.)

# Question: Encapsulation

- ❖ Consider a HTTP packet sent from a client to a server. Assume that the client know all the MAC and IP addresses that it needs to send this packet, please show how the link-layer frame must look (including source and destination fields in the various headers).

# Question: BGP

- ❖ Given a set of customers, providers and peers and their traceroute data, calculate possible routes (in the order of preference) to a remote destination



# Question: SDN

- ❖ Draw a picture illustrating Software Defined Networking architecture showing a path of different packets in a flow