Wireless and Mobile Networks

Slides originally from Carey Williamson

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

Slides are adapted from the book's companion Web site, with changes by Anirban Mahanti and Carey Williamson.

Outline

- Introduction
- Standards and Link Characteristics
- IEEE 802.11 Wireless LANS
- Mobility
- Wireless/Mobility Performance Issues
- Summary

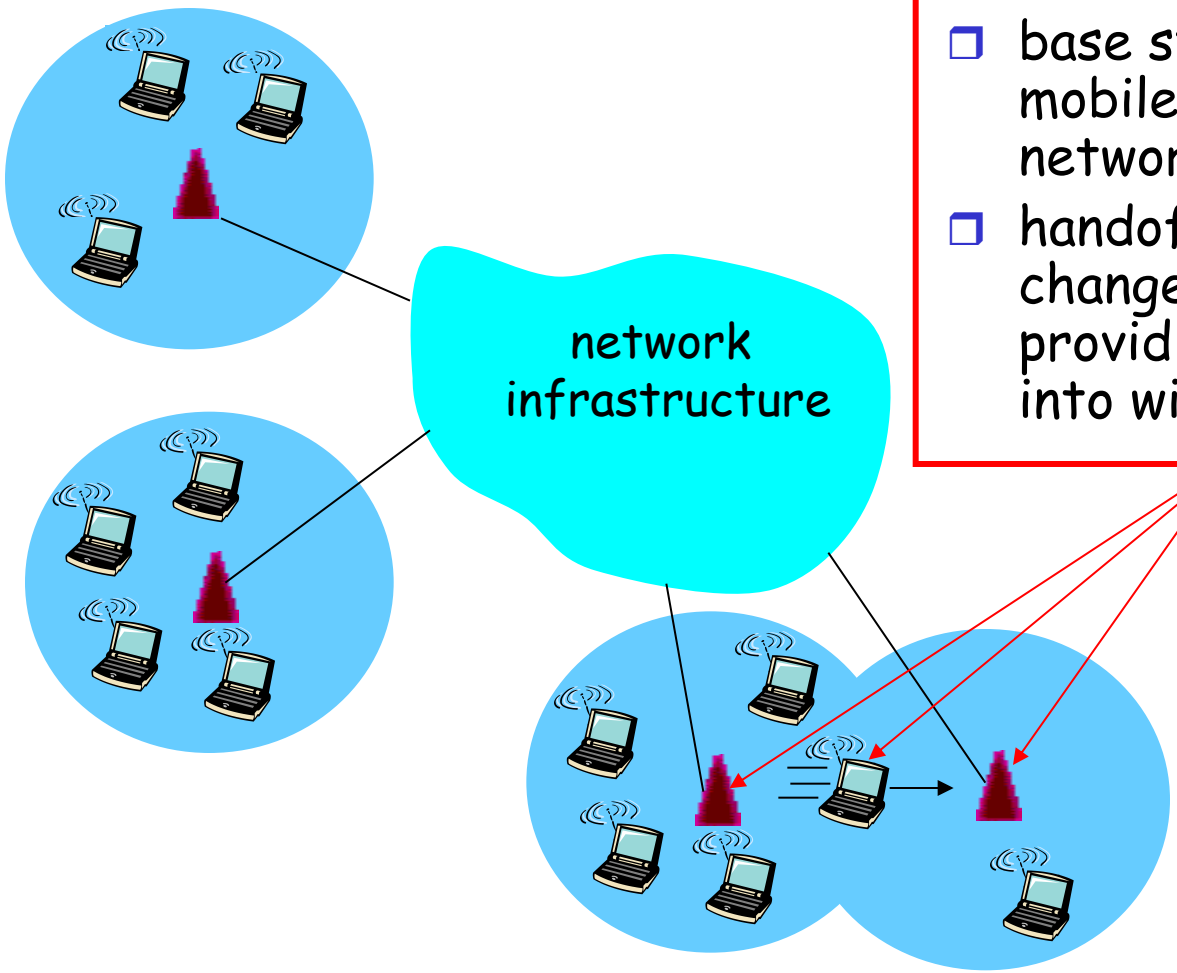
What is Wireless Networking?

- The use of infra-red (IR) or radio frequency (RF) signals to share information and resources between devices
- Promises *anytime*, *anywhere* connectivity
  - laptops, palmtops, PDAs, Internet-enabled phone promise anytime *untethered* Internet access
- Two important (but different) challenges
  - communication over wireless link
  - handling mobile user who changes point of attachment to network
- Lots of media buzzwords!
  - Mobile Internet, Pervasive Computing, Nomadic Computing, M-Commerce, Ubiquitous Computing …

Wireless Networking Technologies

- Mobile devices – laptop, PDA, cellular phone, wearable computer, …
- Operating modes
  - Infrastructure mode (uses Access Point (AP))
  - Ad hoc mode
- Access technology
  - Bluetooth (1 Mbps, up to 3 meters)
  - IEEE 802.11 (up to 54 Mbps, 20 – 100 meters)

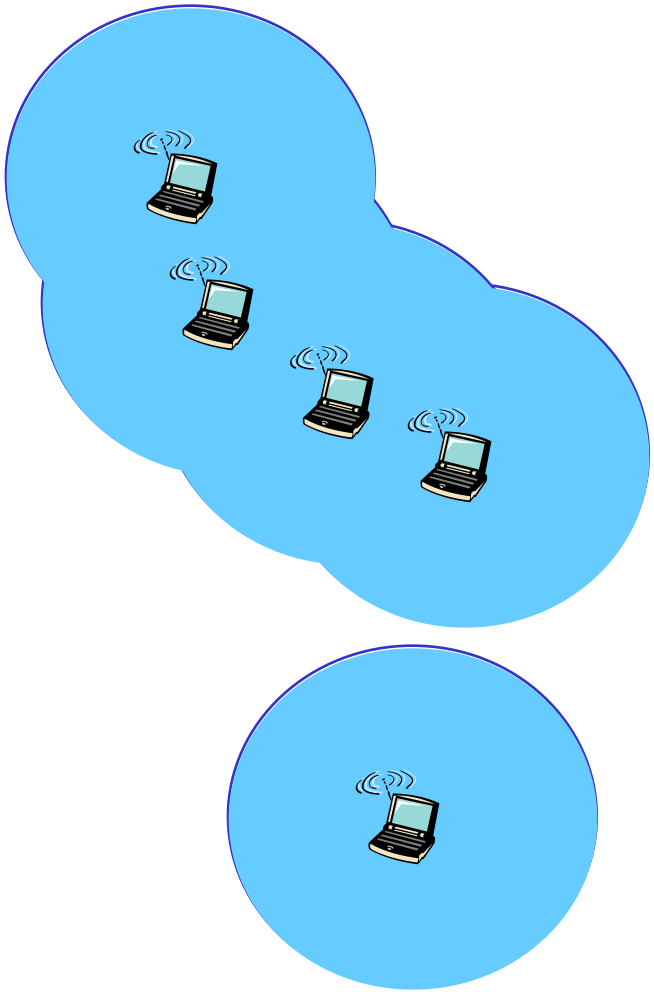
Infrastructure Mode



infrastructure mode

- base station connects mobiles into wired network
- handoff: mobile changes base station providing connection into wired network

Ad hoc Mode



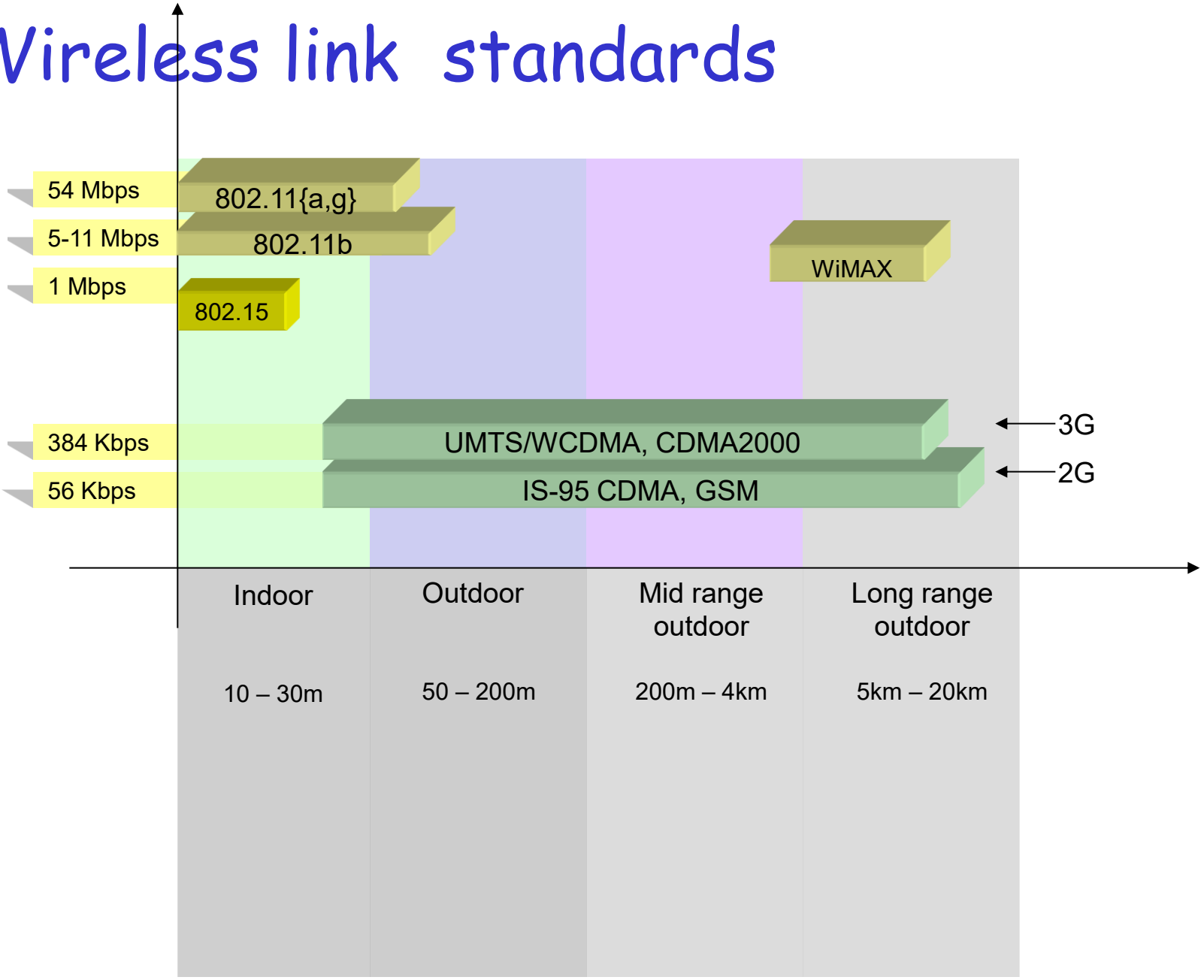
Ad hoc mode

- no base stations
- nodes can only transmit to other nodes within link coverage
- nodes organize themselves into a network: route among themselves

Outline

- Introduction
- Standards and Link Characteristics
- IEEE 802.11 Wireless LANS
- Mobility
- Wireless/Mobility Performance Issues
- Summary

Wireless link standards



Two Popular 2.4 GHz Standards:

- IEEE 802.11
  - Fast (11 Mbps)
  - High Power
  - Long range
  - Single-purpose
  - Ethernet replacement
  - Easily Available
    - Apple Airport, iBook
    - Cisco Aironet 350
- Bluetooth
  - Slow (1 Mbps)
  - Low Power
  - Short range
  - Flexible
  - Cable replacement

Wireless Link Characteristics

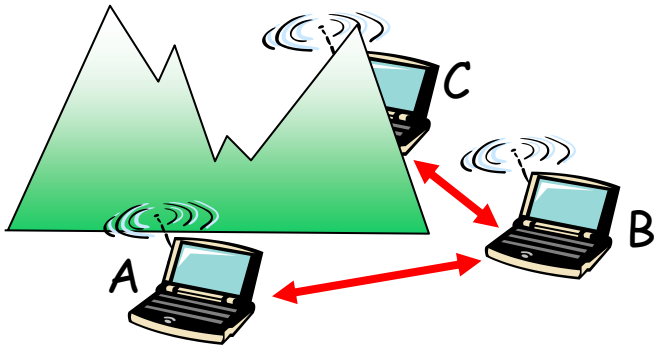
Differences from wired link ….

- Decreasing signal strength: radio signal attenuates as it propagates through matter (path loss)
- Interference from other sources: standardized wireless network frequencies (e.g., 2.4 GHz) shared by other devices (e.g., phone); devices (motors) interfere as well
- Multi-path propagation: radio signal reflects off objects ground, arriving at destination at slightly different times

…. make communication across (even a point to point) wireless link much more "difficult"

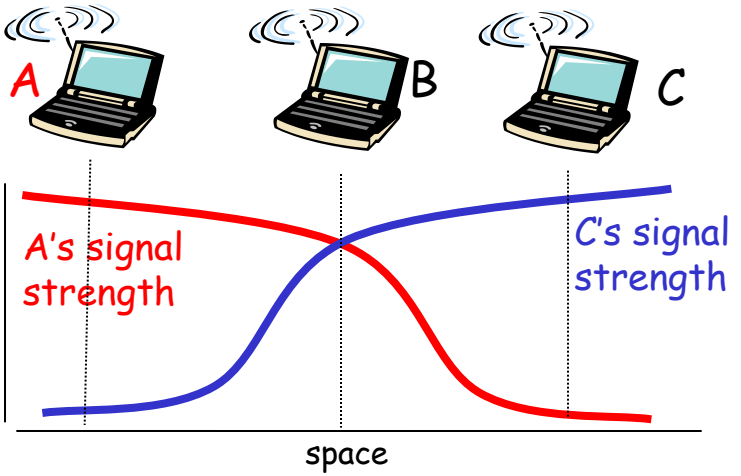
Wireless Network Characteristics

Multiple wireless senders and receivers create additional problems (beyond multiple access):



Hidden terminal problem

- A and B can hear each other
- B and C can hear each other
- A and C can't hear each other
- thus A and C are unaware of their interference at B



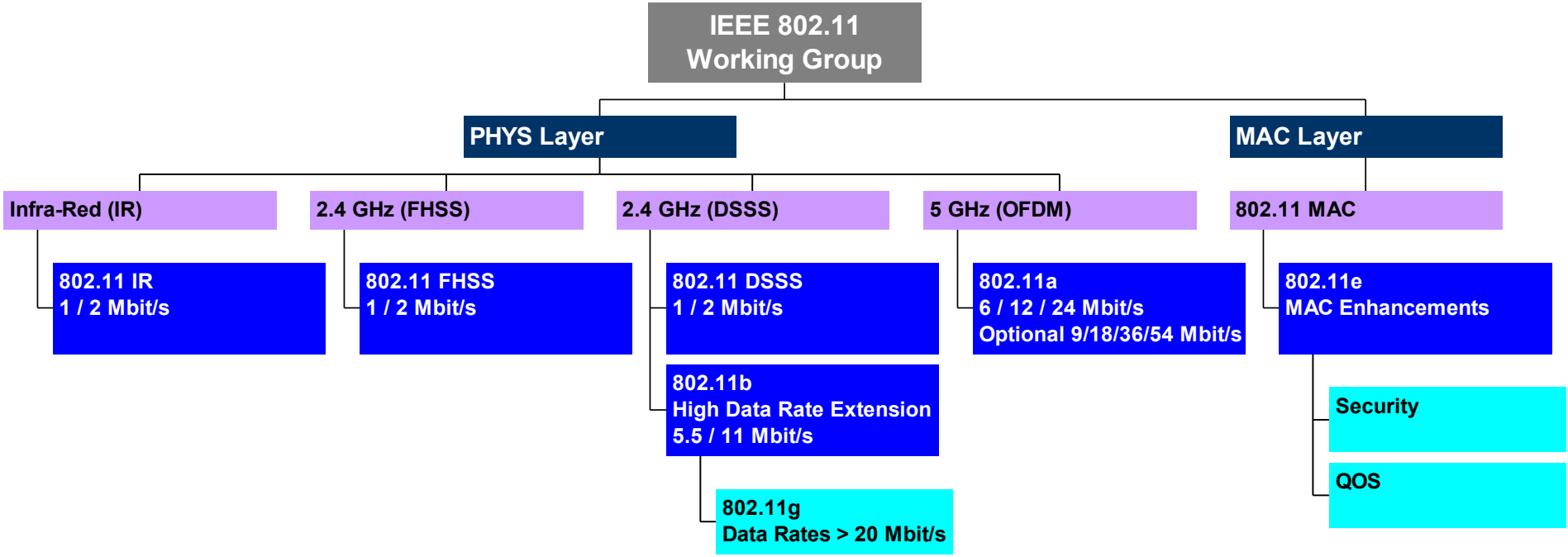
Signal fading:

- A and B hear each other
- B and C hear each other
- A and C can't hear each other interfering at B

Outline

- Introduction
- Standards and Link Characteristics
- IEEE 802.11 Wireless LANS
- Mobility
- Wireless/Mobility Performance Issues
- Summary

IEEE 802.11 Organization Tree:

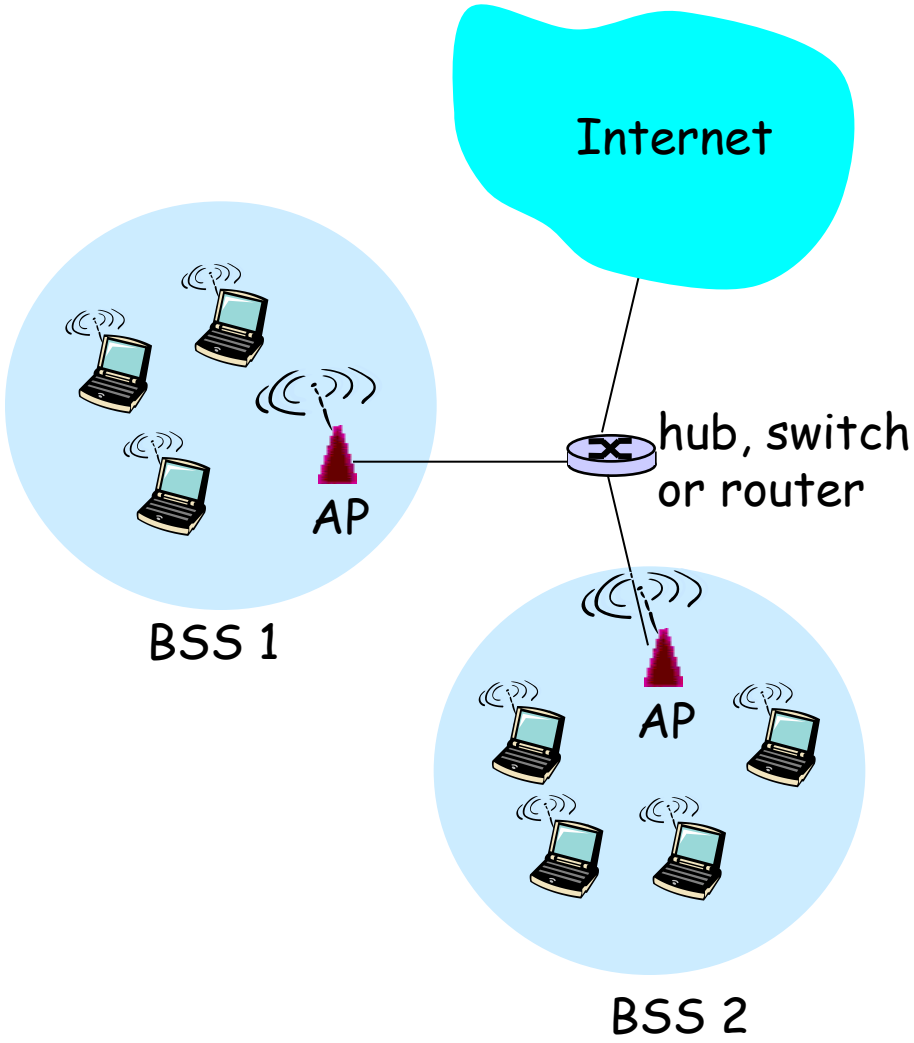


PHYS: physical
FHSS: Frequency-hopping spread spectrum
DSSS: Direct-sequence spread specturm
OFDM: Orthogonal frequency-division multiplexing

IEEE 802.11 Wireless LAN

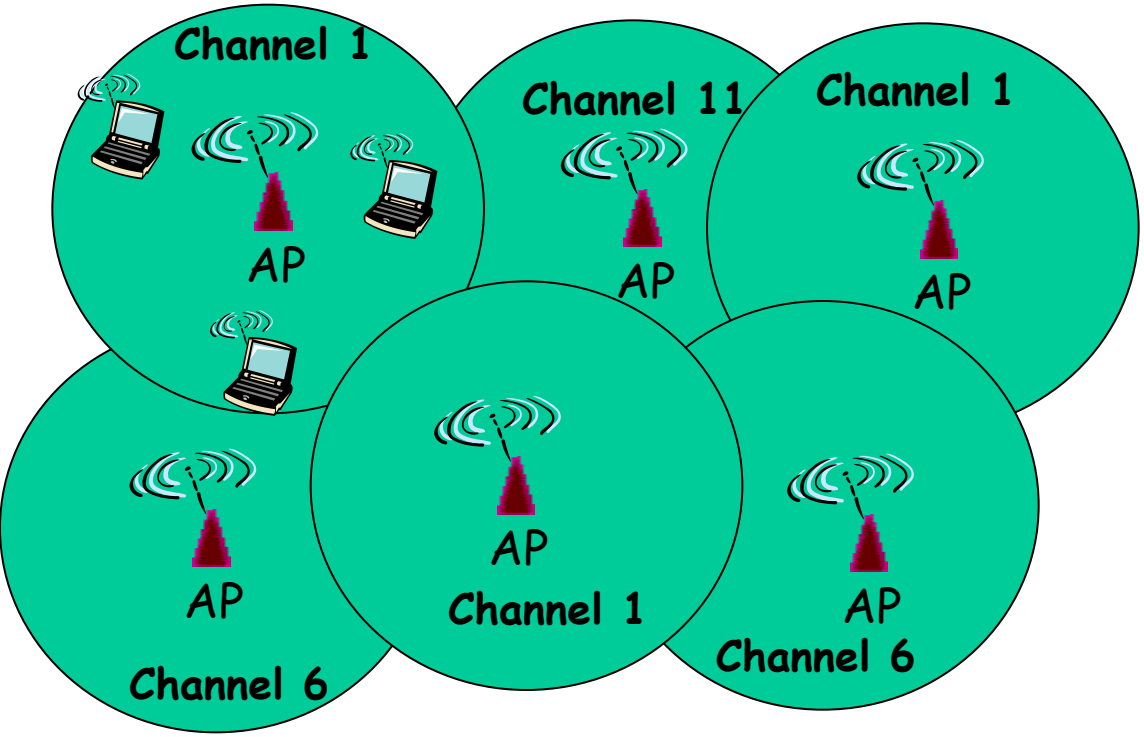
- 802.11b
  - 2.4 GHz unlicensed radio spectrum
  - up to 11 Mbps
  - direct sequence spread spectrum (DSSS) in physical layer
    - all hosts use same chipping code
  - widely deployed, using base stations
- 802.11a
  - 5 GHz range
  - up to 54 Mbps
- 802.11g
  - 2.4-5 GHz range
  - up to 54 Mbps
- All use CSMA/CA for multiple access
- All have infrastructure and ad hoc modes

802.11 LAN architecture



- wireless host communicates with base station
  - base station = access point (AP)
- Basic Service Set (BSS) (aka "cell") in infrastructure mode contains:
  - wireless hosts
  - access point (AP)
  - ad hoc mode: hosts only

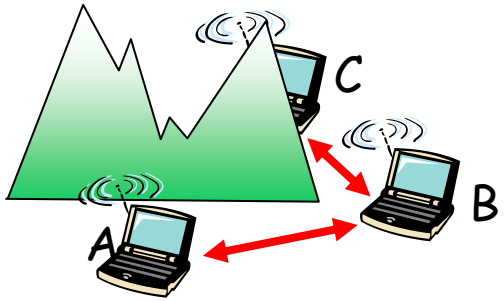
Wireless Cells

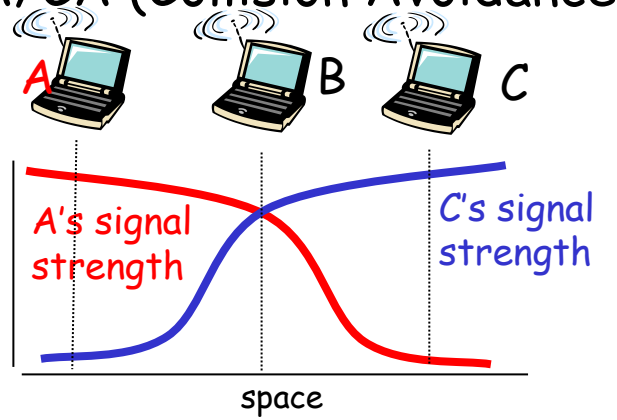


- 802.11 has 11 channels
- Channels 1, 6, and 11 are non-overlapping
- Each AP coverage area is called a "cell"
- Wireless nodes can roam between cells

IEEE 802.11: multiple access

- avoid collisions: > 1 nodes transmitting at same time
- 802.11: *CSMA* – (Carrier Sense Multiple Access / Collision Detection) sense before transmitting
  - don't collide with ongoing transmission by other node
- 802.11: *no* collision detection!
  - half-duplex: antenna can't receive (sense collisions) when transmitting due to weak received signals (fading)
  - can't sense all collisions in any case: hidden terminal, fading
  - goal: *avoid collisions:* CSMA/CA (Collision Avoidance)





IEEE 802.11 MAC Protocol: CSMA/CA

If the medium is continuously idle for DCF Interframe Space (DIFS) duration then only it is allowed to transmit a frame.

DIFS = SIFS + (2 * Slot time)

802.11 sender

1 if sense channel idle for **DIFS** then

transmit entire frame

2 if sense channel busy then

start random backoff time

timer counts down while channel idle

transmit when timer expires

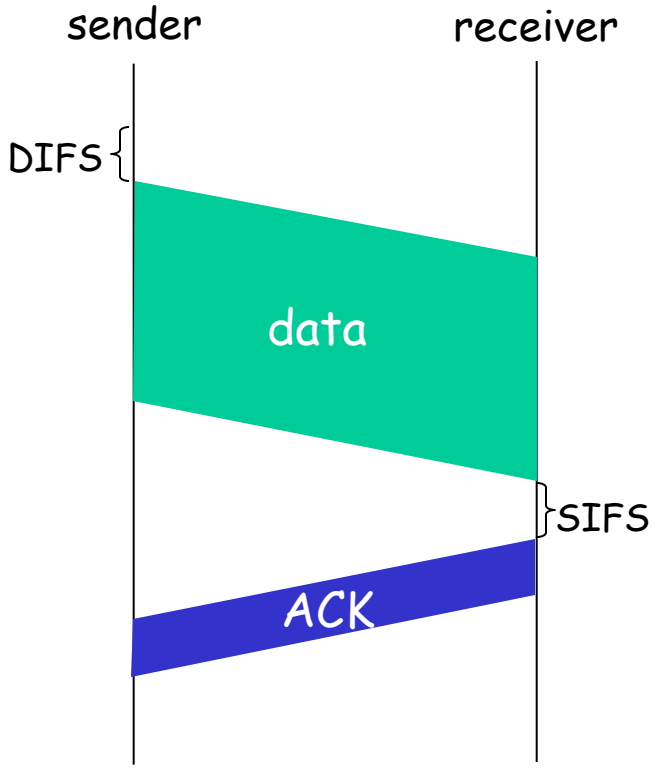
3 if no ACK then increase random backoff interval, repeat step 2

802.11 receiver

- if frame received OK

return ACK after **SIFS**

(service model is connectionless, acked)



Short Interframe Space (SIFS), is the small time interval between the data frame and its acknowledgment. SIFS are found in IEEE 802.11 networks.

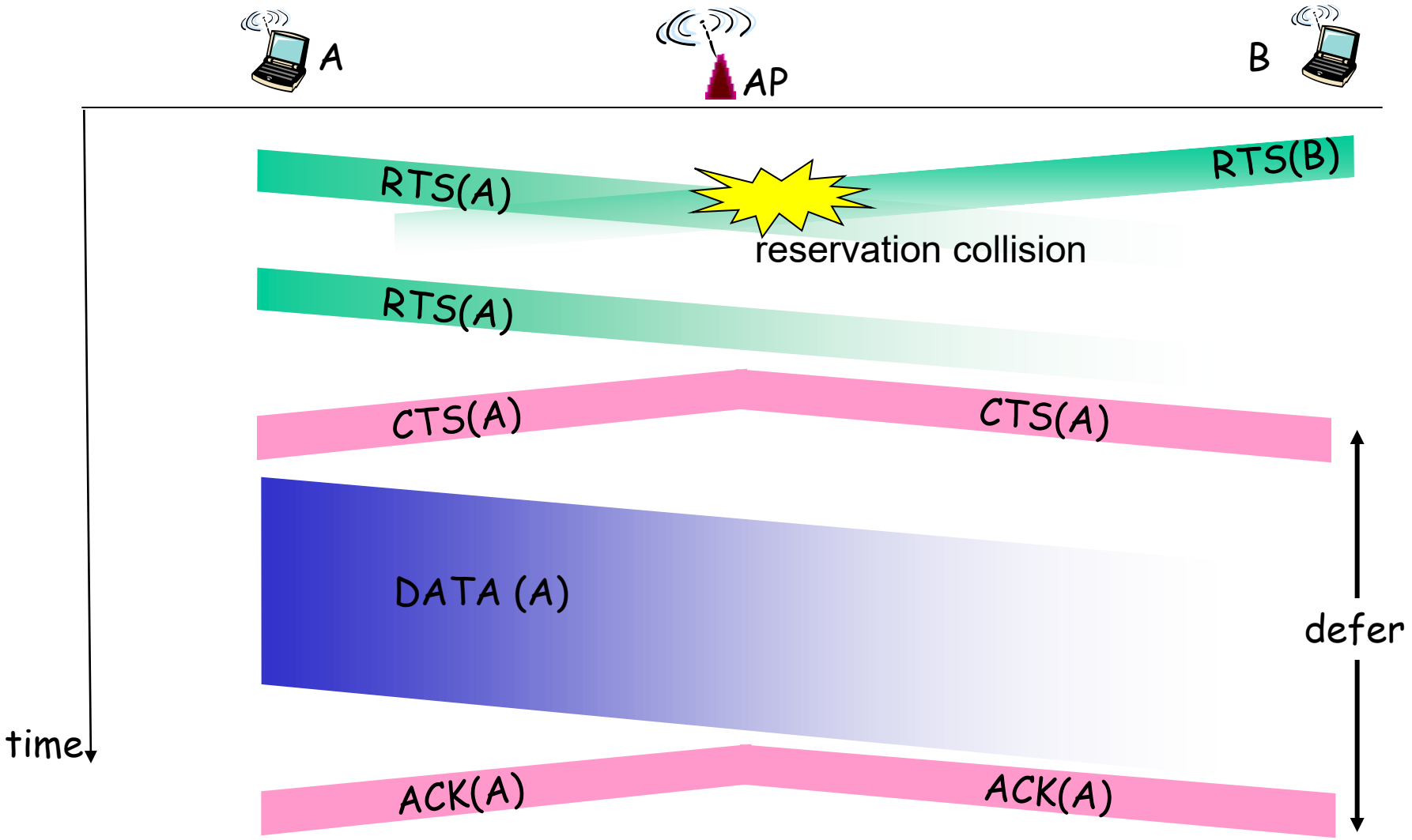
Avoiding collisions (more)

idea: allow sender to "reserve" channel rather than random access of data frames: avoid collisions of long data frames

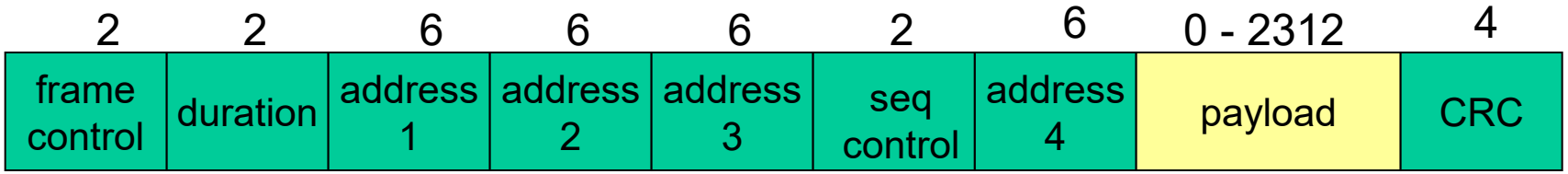
- sender first transmits *small* request-to-send (RTS) packets to base station using CSMA
  - RTS may still collide with each other (but they're short)
- BS broadcasts clear-to-send CTS to host in response to RTS
- RTS heard by all nodes because of broadcast property
  - sender transmits (large) data frame
  - other stations defer transmissions until it is done

Avoid data frame collisions completely using small reservation packets!

Collision Avoidance: RTS-CTS exchange



802.11 frame: addressing



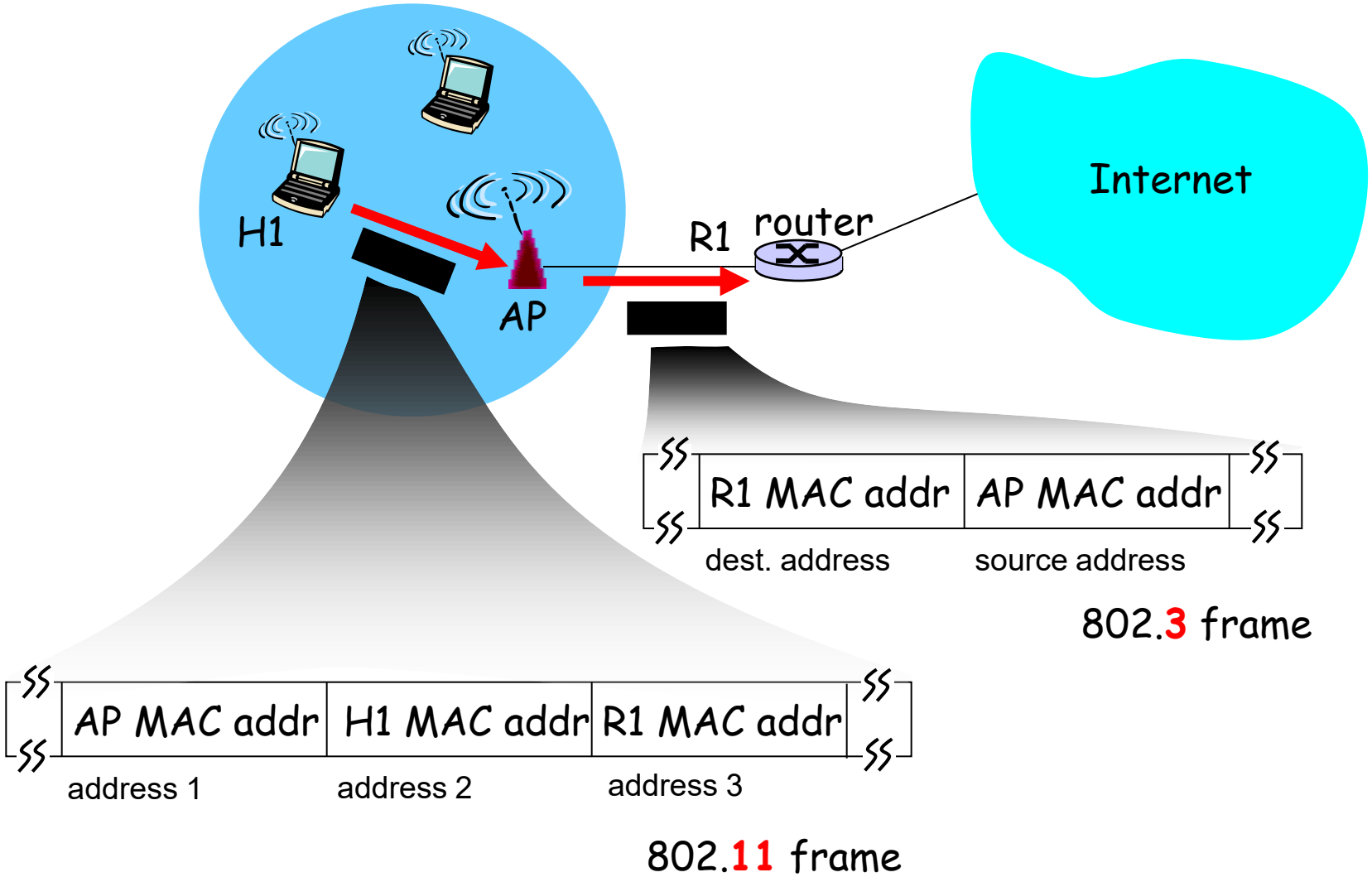
Address 1: MAC address of wireless host or AP to receive this frame

Address 2: MAC address of wireless host or AP transmitting this frame

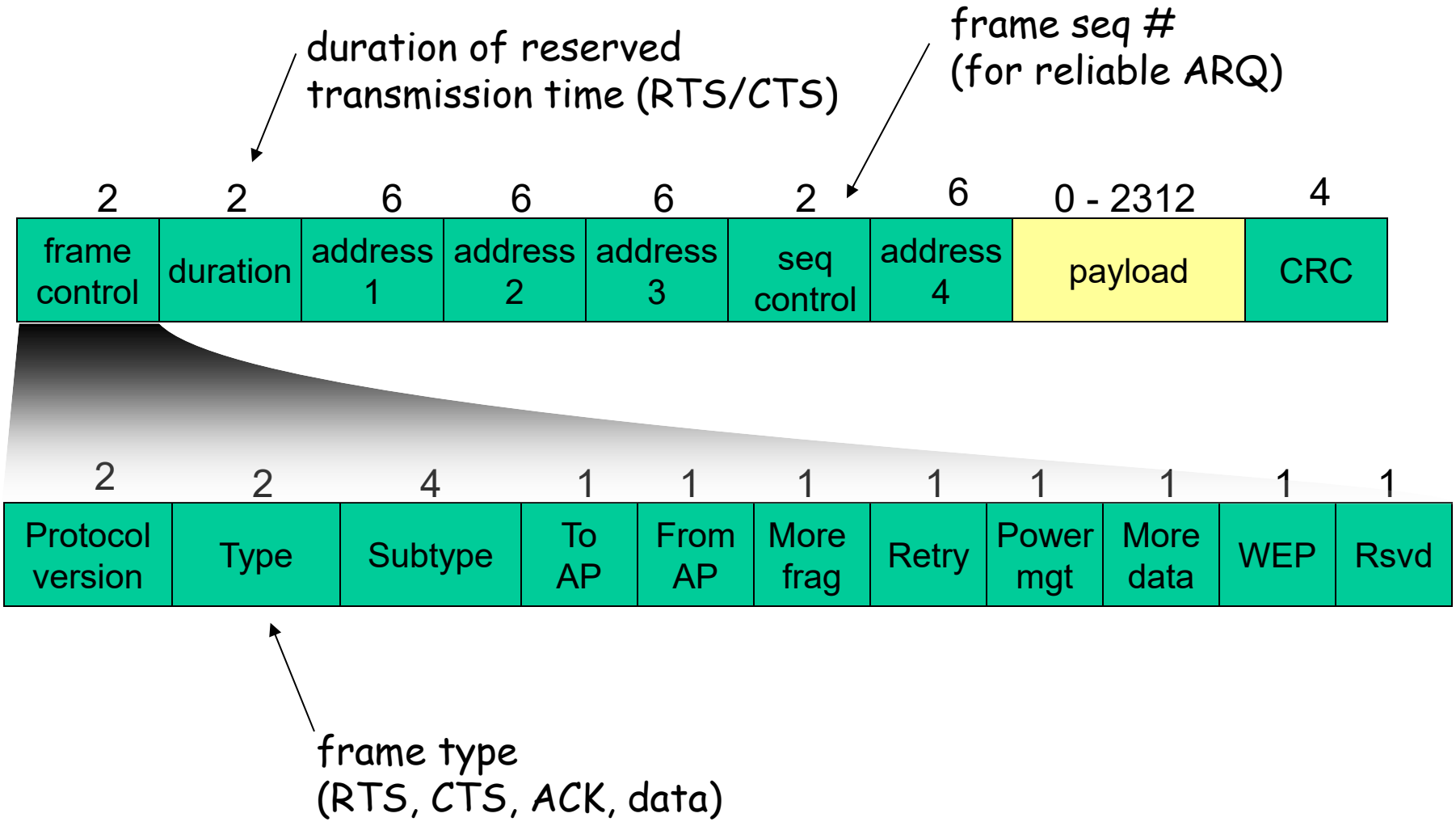
Address 3: MAC address of router interface to which AP is attached

Address 3: used only in ad hoc mode

802.11 frame: addressing



802.11 frame: more

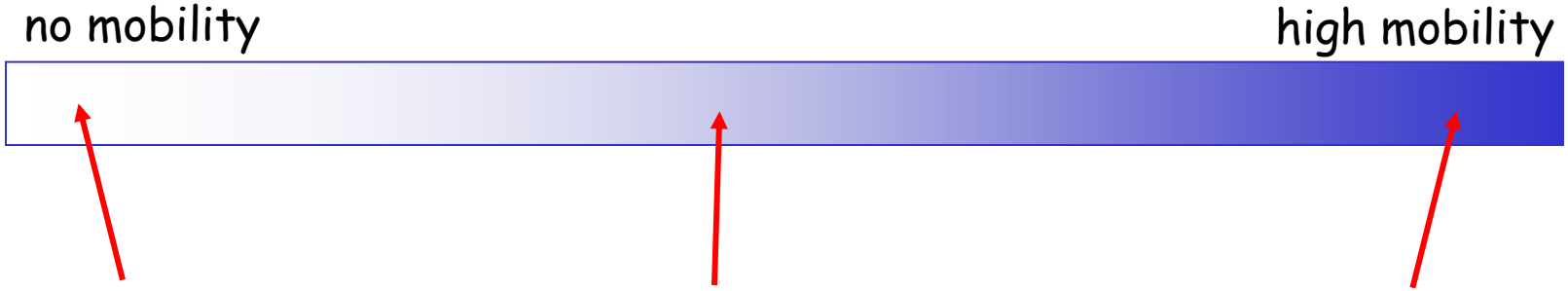


Outline

- Introduction
- Standards and Link Characteristics
- IEEE 802.11 Wireless LANS
- Mobility
- Wireless/Mobility Performance Issues
- Summary

What is mobility?

- spectrum of mobility, from the *network* perspective:



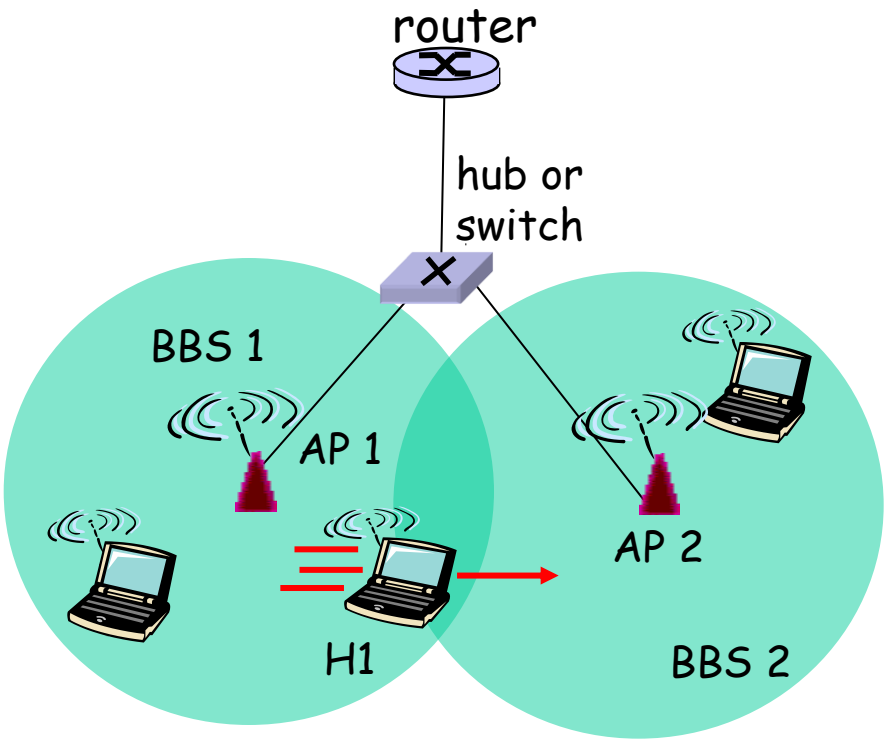
mobile wireless user, using same access point

mobile user, connecting/ disconnecting from network using DHCP.

mobile user, passing through multiple access point while maintaining ongoing connections (like cell phone)

802.11: mobility within same subnet

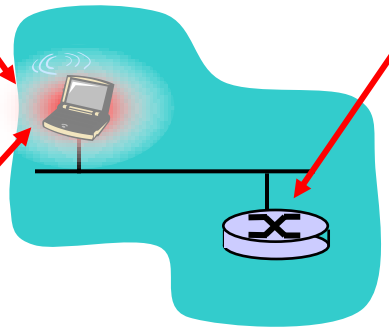
- H1 remains in same IP subnet: IP address can remain same
- switch: which AP is associated with H1?
  - self-learning: switch will see frame from H1 and "remember" which switch port can be used to reach H1

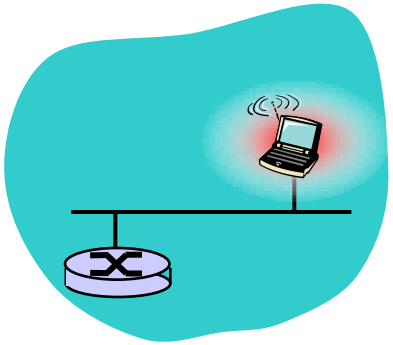


Mobility: Vocabulary

home network: permanent "home" of mobile (e.g., 128.119.40/24)

home agent: entity that will perform mobility functions on behalf of mobile, when mobile is remote





wide area network

permanent address: address in home network, *can always* be used to reach mobile e.g., 128.119.40.186

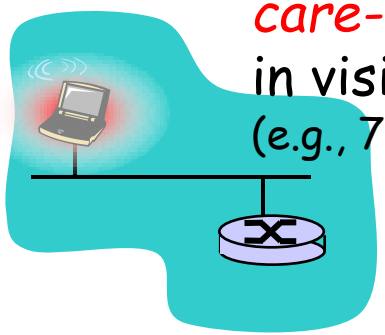
correspondent

Mobility: more vocabulary

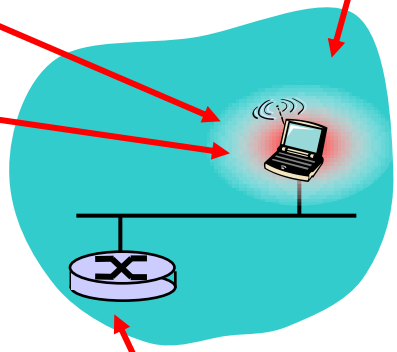
permanent address: remains constant (e.g., 128.119.40.186)

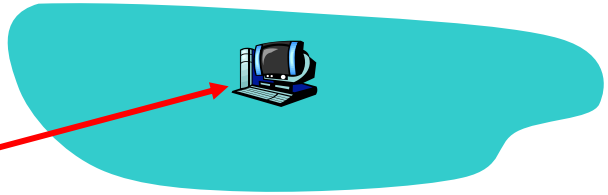
visited network: network in which mobile currently resides (e.g., 79.129.13/24)

care-of-address: address in visited network. (e.g., 79,129.13.2)



wide area network





correspondent: wants to communicate with mobile

home agent: entity in visited network that performs mobility functions on behalf of mobile.

How do *you* contact a mobile friend:

Consider friend frequently changing addresses, how do you find her?

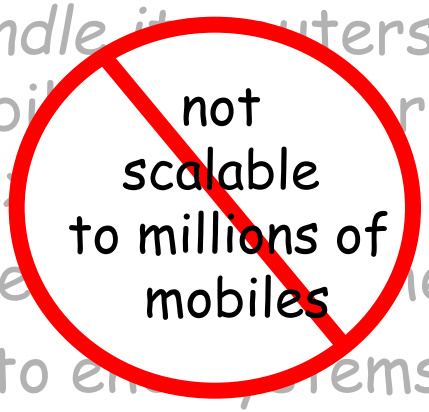
- search all phone books?
- call her parents?
- expect her to let you know where he/she is?

Mobility: approaches

- *Let routing handle it:* routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems
- *Let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

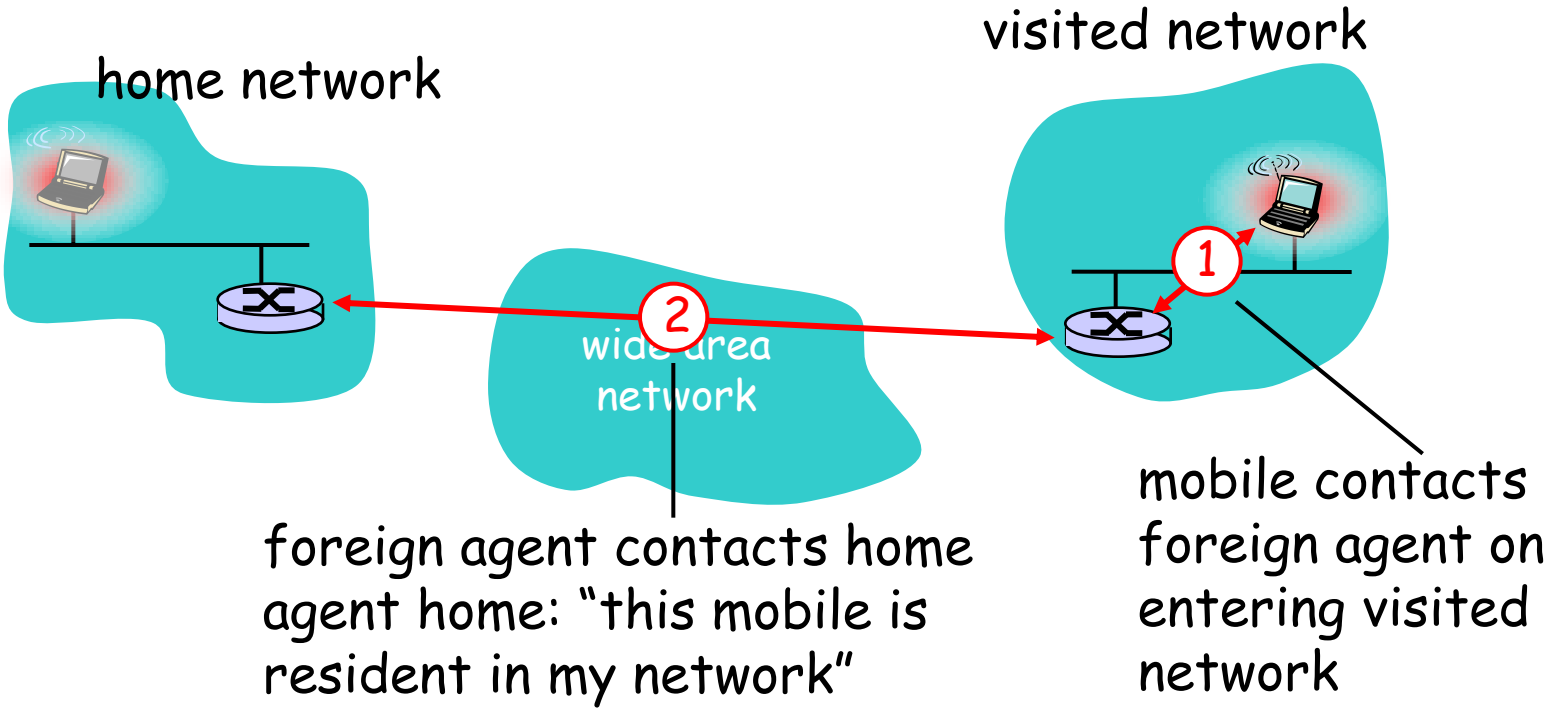
Mobility: approaches

- *Let routing handle it:* routers advertise permanent address of mobile-nodes-in-residence via usual routing table exchange.
  - routing tables indicate where each mobile located
  - no changes to end-systems



- *let end-systems handle it:*
  - *indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

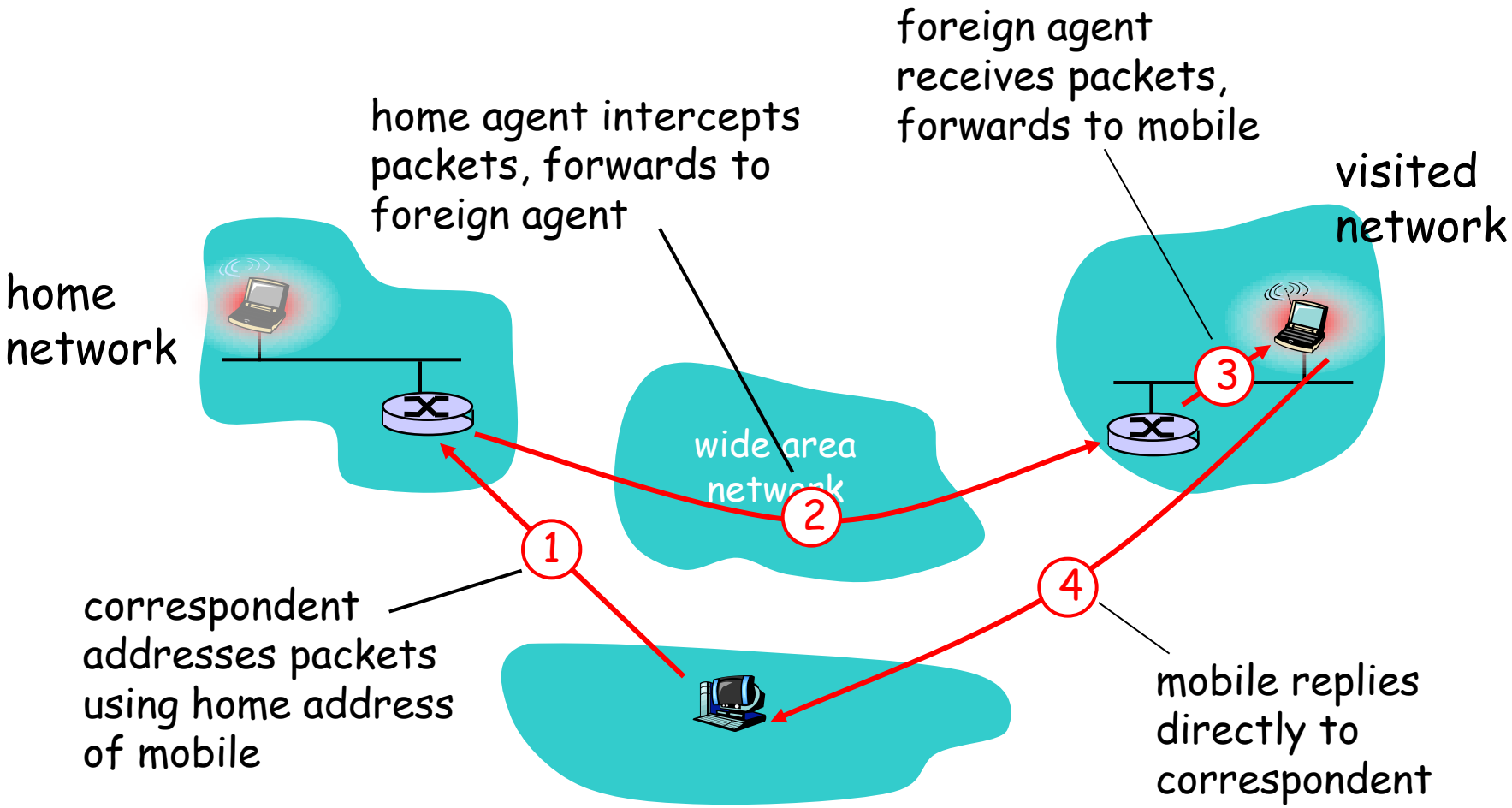
Mobility: registration



End result:

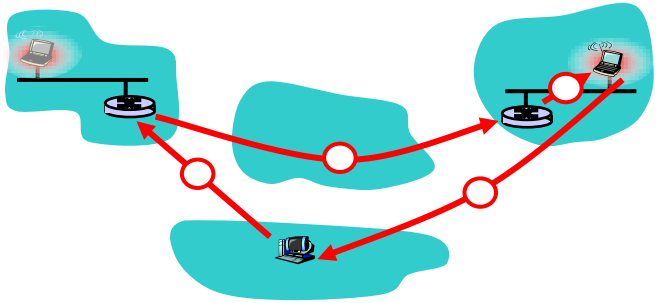
- Foreign Agent (FA) knows about mobile
- Home Agent (HA) knows location of mobile

Mobility via Indirect Routing



Indirect Routing: comments

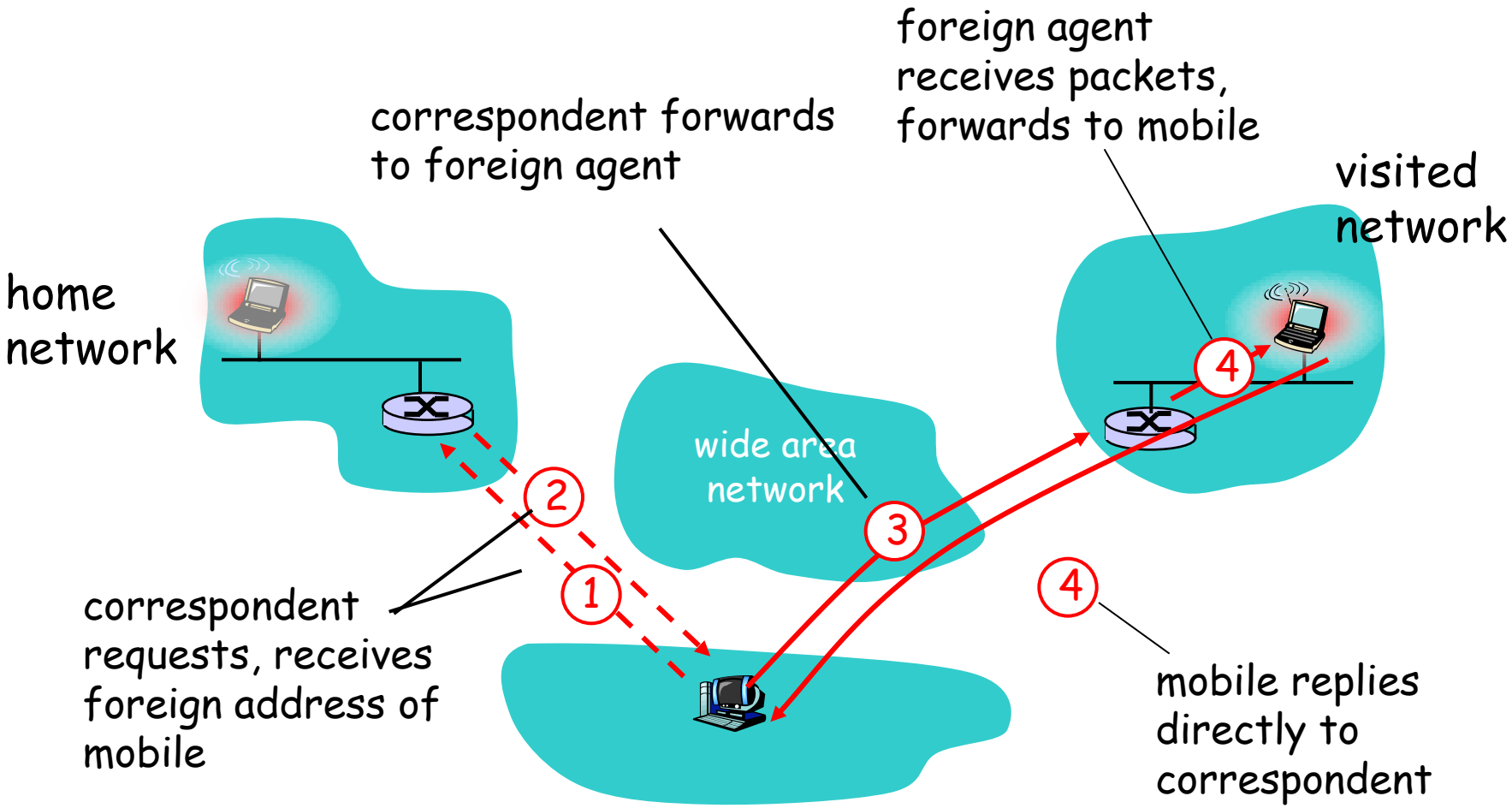
- Mobile uses two addresses:
  - permanent address: used by correspondent (hence mobile location is *transparent* to correspondent)
  - care-of-address: used by home agent to forward datagrams to mobile
- foreign agent functions may be done by mobile itself
- triangle routing: correspondent-home-network-mobile
  - inefficient when correspondent, mobile are in same network



Indirect Routing: moving between networks

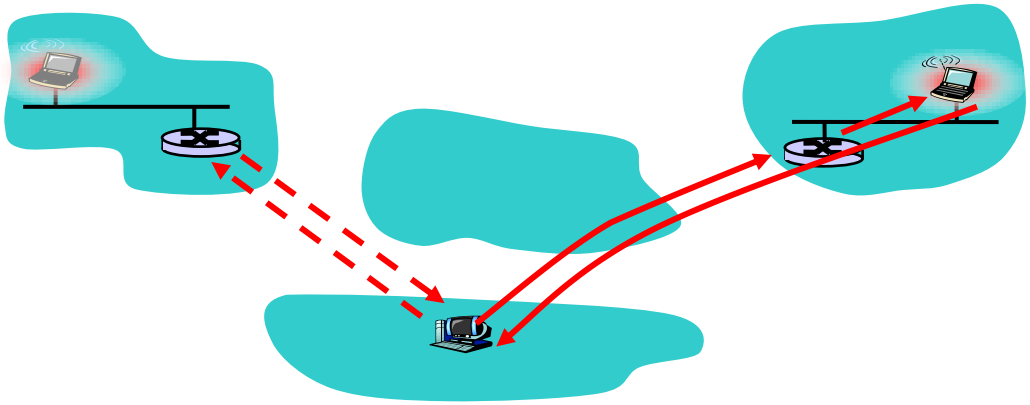
- suppose mobile user moves to another network
  - registers with new foreign agent
  - new foreign agent registers with home agent
  - home agent update care-of-address for mobile
  - packets continue to be forwarded to mobile (but with new care-of-address)
- mobility, changing foreign networks transparent: *ongoing connections can be maintained!*

Mobility via Direct Routing



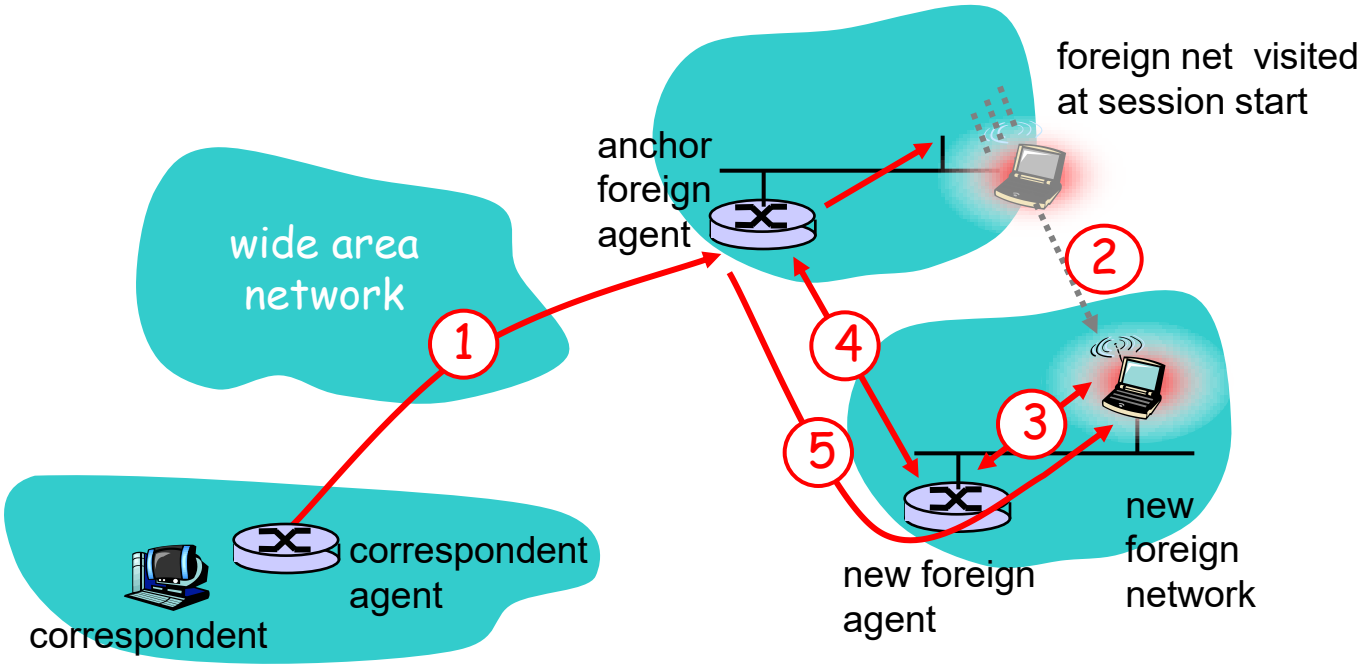
Mobility via Direct Routing: comments

- overcome triangle routing problem
- non-transparent to correspondent: correspondent must get care-of-address from home agent
  - what if mobile changes visited network?



Accommodating mobility with direct routing

- anchor foreign agent: FA in first visited network
- data always routed first to anchor FA
- when mobile moves: new FA arranges to have data forwarded from old FA (chaining)



Mobile IP

- RFC 3220
- has many features we've seen:
  - home agents, foreign agents, foreign-agent registration, care-of-addresses, encapsulation (packet-within-a-packet)
- three components to standard:
  - indirect routing of datagrams
  - agent discovery
  - registration with home agent

Outline

- Introduction
- Standards and Link Characteristics
- IEEE 802.11 Wireless LANS
- Mobility
- Wireless/Mobility Performance Issues
- Summary

Wireless, mobility: impact on higher layer protocols

- logically, impact *should* be minimal …
  - best effort service model remains unchanged
  - TCP and UDP can (and do) run over wireless, mobile
- … but performance-wise:
  - packet loss/delay due to bit-errors (discarded packets, delays for link-layer retransmissions), and handoffs from mobility and transient connectivity
  - TCP interprets loss as congestion, will decrease congestion window un-necessarily
  - delay impairments for real-time traffic
  - limited bandwidth of wireless links

Summary

Wireless

- wireless links:
  - capacity, distance
  - channel impairments
  - CDMA
- IEEE 802.11 ("WiFi")
  - CSMA/CA reflects wireless channel characteristics

Mobility

- principles: addressing, routing to mobile users
  - home, visited networks
  - direct, indirect routing
  - care-of-addresses
- Mobile IP
- impact on higher-layer protocols

Wireless Network Overview

- Slides originally from Carey Williamson and David Schwab
- Notes derived from "*Computer Networking: A Top Down Approach*", by Jim Kurose and Keith Ross, Addison-Wesley.
- **Slides are adapted from the book's companion Web site, with changes by Anirban Mahanti and Carey Williamson.**

What Is Wireless Networking?

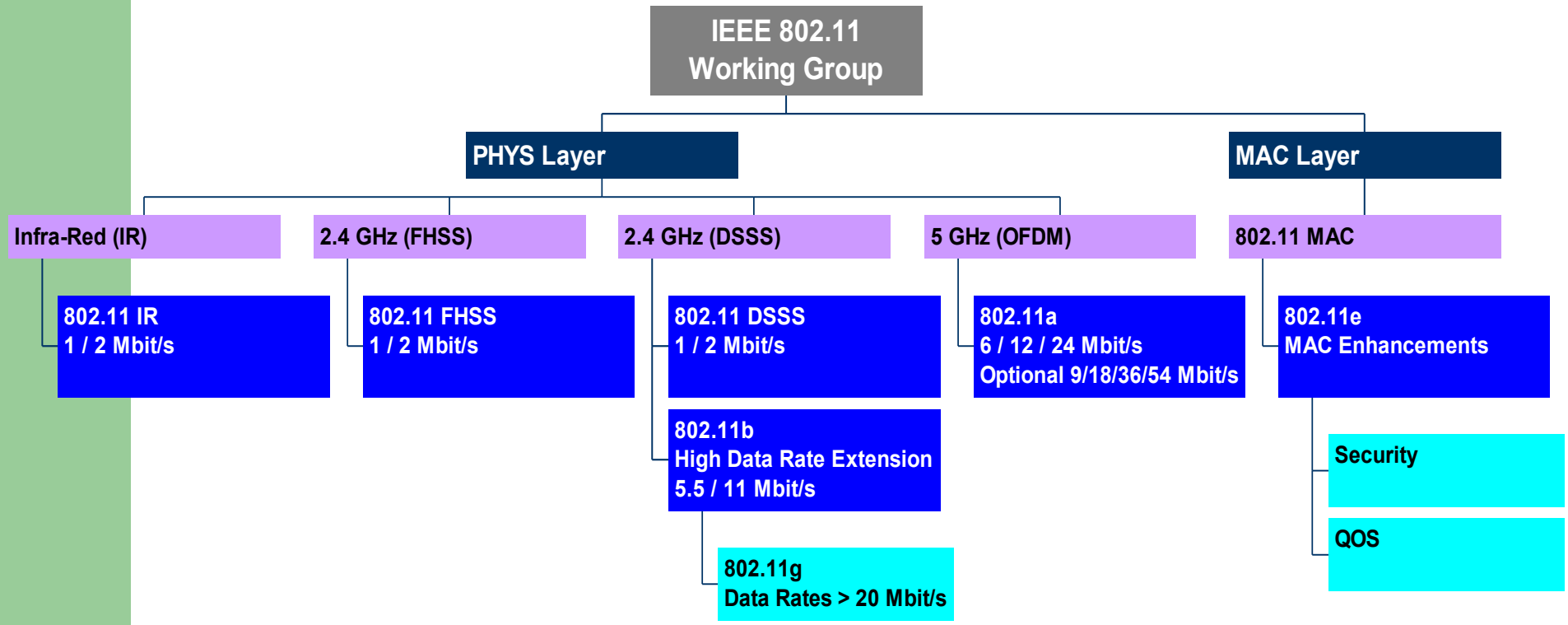
- The use of infra-red or radio frequency signals to share information and resources between devices
- A hot computer industry buzzword:
  - Lots of advertising by companies and media
  - Wireless Broadband, 3G wireless, WAP, iMode, Bluetooth
- Mobile Internet, Pervasive Computing, M-Commerce
  - Ubiquitous?
  - Global?
  - Revolutionary?

Two Popular 2.4GHz Standards:

- IEEE 802.11
  - Fast (11B)
  - High Power
  - Long range
  - Single-purpose
  - Ethernet replacement
  - Easily Available
    - Apple Airport, iBook, G4
    - Cisco Aironet 350

- Bluetooth
  - Slow
  - Low Power
  - Short range
  - Flexible
  - Cable replacement
  - “Vapourware”
    - Anoto, Test cards, phone

IEEE 802.11 Organization Tree:



Pros and Cons of 802.11:

- Pro:
  - High bandwidth (up to 11 Mbps)
  - Two modes of operation: infrastructure vs. ad hoc
- Con:
  - Incompatibility between old and new cards
  - Signal blocked by reinforced concrete or tinted glass
  - High channel BER can degrade performance (lots!)
  - No standard for hand-off between base stations
  - Some channel numbers overlap spectrum
  - High power consumption in laptops

Multi-Hop Ad Hoc Wireless Networking

- Routing protocols used to improve wireless connections
- Infrastructure-free, dynamic
- True Peer-to-Peer routing
- Fault tolerant

- Examples: DSDV, TORA, DSR, ...

Bluetooth

- Think USB, not Ethernet
- Created by Ericsson
- PAN - Personal Area Network
  - 1-2 Mbps connections
  - 1600 hops per second FHSS
  - Includes synchronous, asynchronous, voice connections
  - Piconet routing
- Small, low-power, short-range, cheap, versatile radios
- Used as Internet connection, phone, or headset

More Bluetooth

- SIG: Special Interest Group
  - http://www.bluetooth.com
  - 2164 member companies
  - Including 3Com, Ericsson, IBM, Intel, Lucent, Microsoft, Motorola, Nokia, Toshiba, etc.
  - Bluetooth Specification (v1.1)
- Trouble:
  - Not an Ethernet replacement
  - Incomplete specification
  - Incompatible implementations
  - Delayed products, low demand
- Predictions still optimistic

Security

- Wireless sniffers
- IEEE 802.11:
  - ESSID – Extended Services Set ID
  - WEP – Wired Equivalent Privacy
    - 40 bit RC4 (RSA) encryption
- Bluetooth Security
  - Rapid hop sequence
  - Short range
  - Encrypted transmissions

Guerrilla.net

- An underground alternative to the wired Internet
- A grassroots movement established in 1996
  - 802.11 Wireless LAN cards
  - Roof mounted antennae
  - Free software (FreeBSD)
- Multi-hop routing, Internet connectivity
- About $800 per node
- Other networks popping up in SF, Seattle, London

Future of Wireless

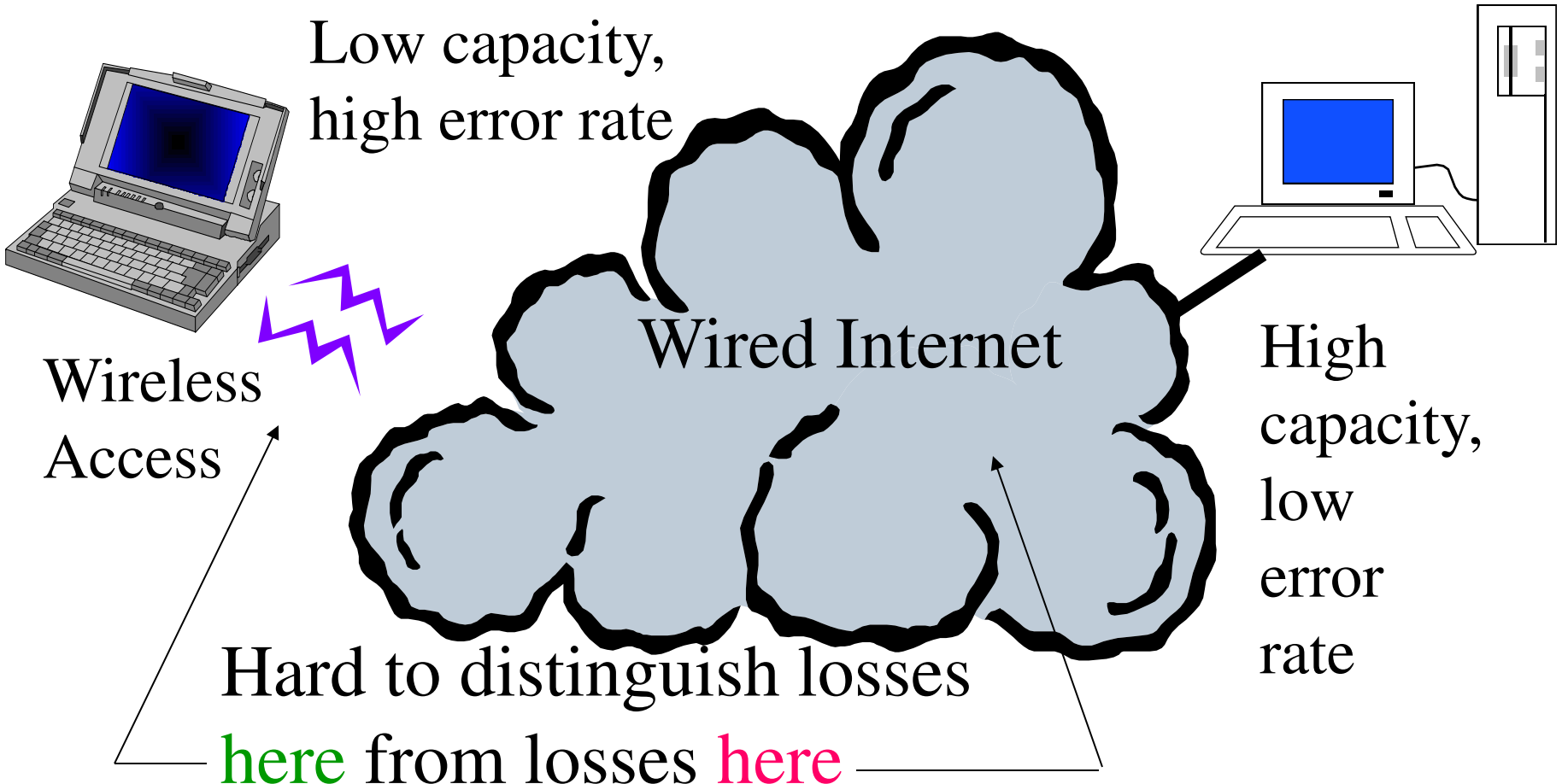
- Better simulations + movement models
- Better security
- Wider selection
- Lower prices
- Less configuration required
- More end-user focus
- Better software
- Less visible
- More popular

Wireless TCP Performance Issues

- Slides originally from Carey Williamson
- Notes derived from "*Computer Networking: A Top Down Approach*", by Jim Kurose and Keith Ross, Addison-Wesley.
- **Slides are adapted from the book's companion Web site, with changes by Anirban Mahanti and Carey Williamson.**

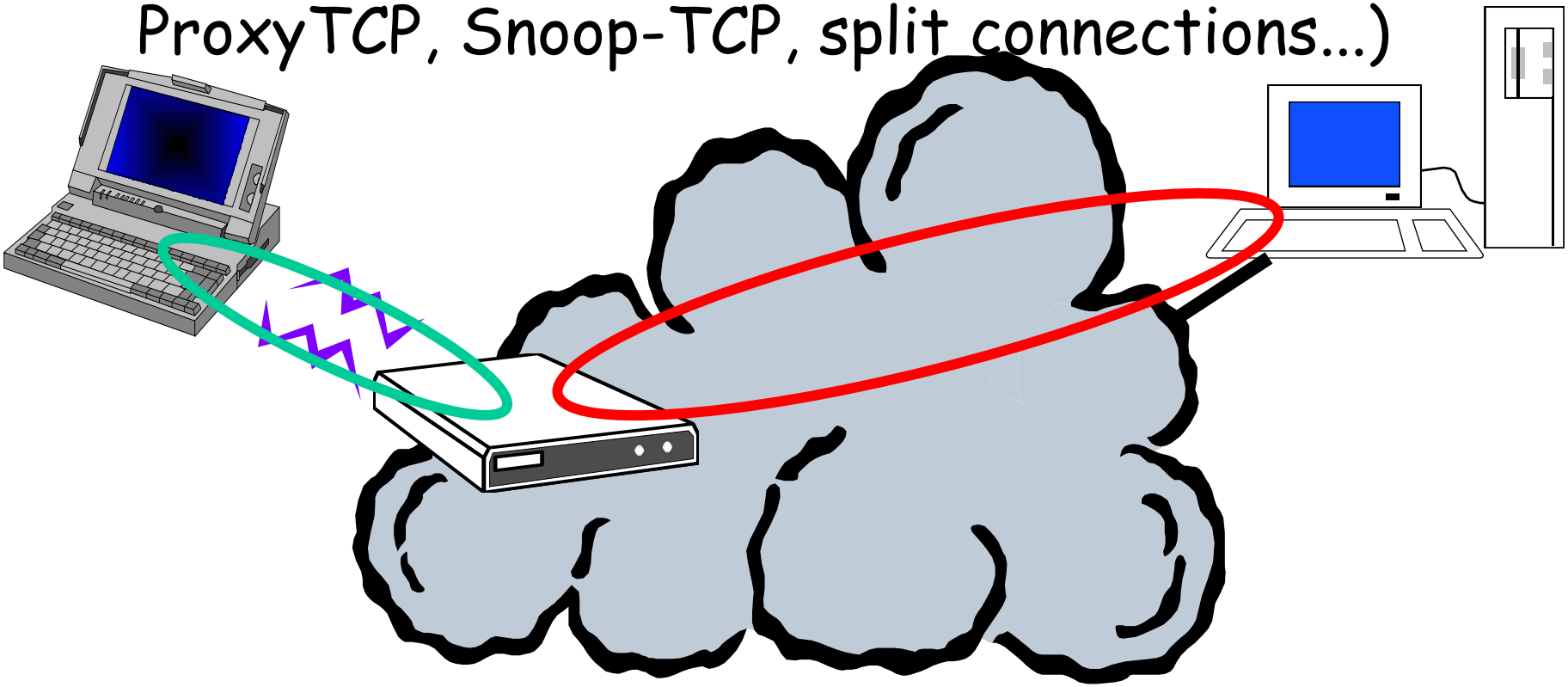
Example #1

- Wireless TCP Performance Problems



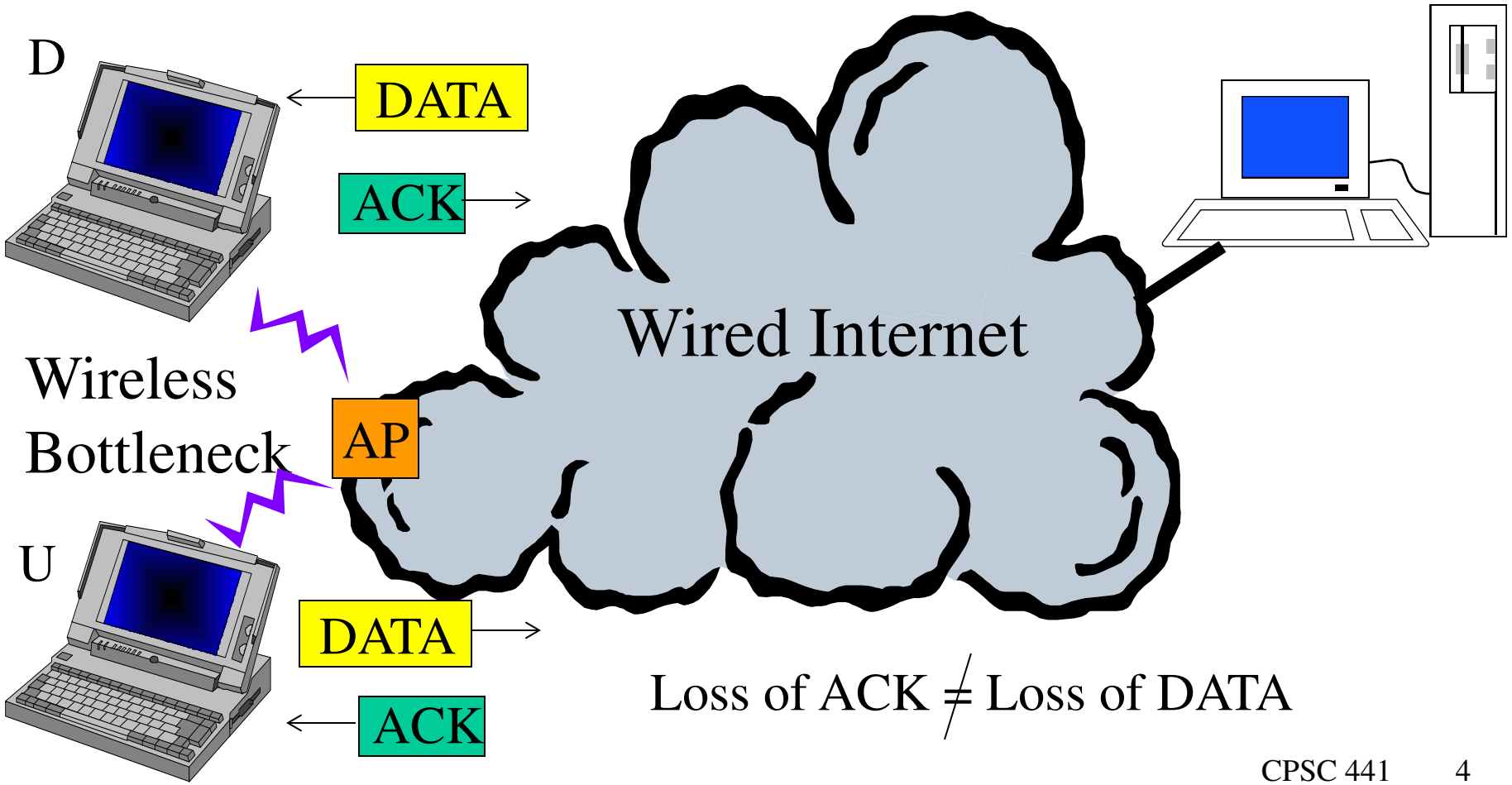
Example #1 (Cont'd)

- Solution: "wireless-aware TCP" (I-TCP, ProxyTCP, Snoop-TCP, split connections...)



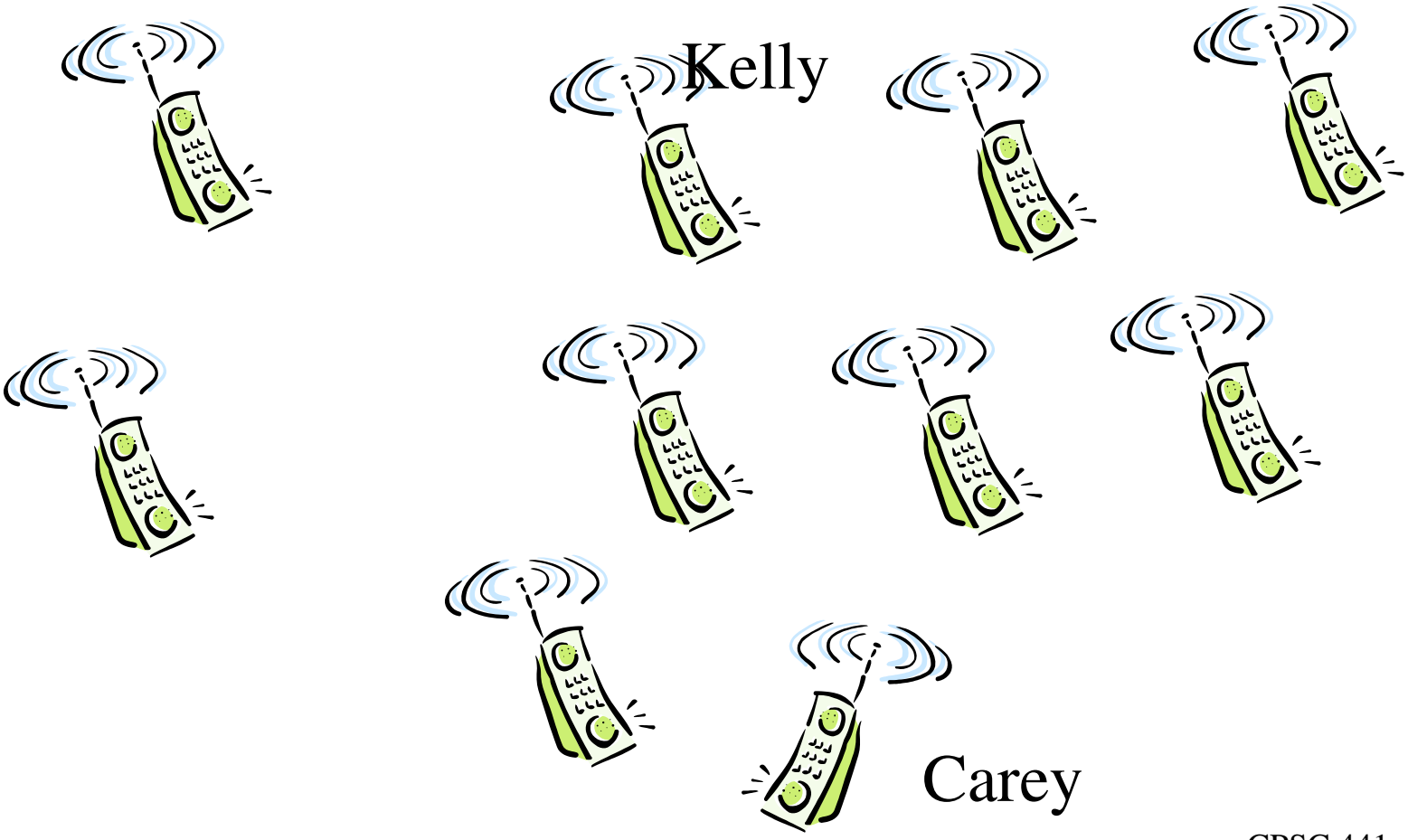
Example #2

- Wireless TCP Fairness Problems



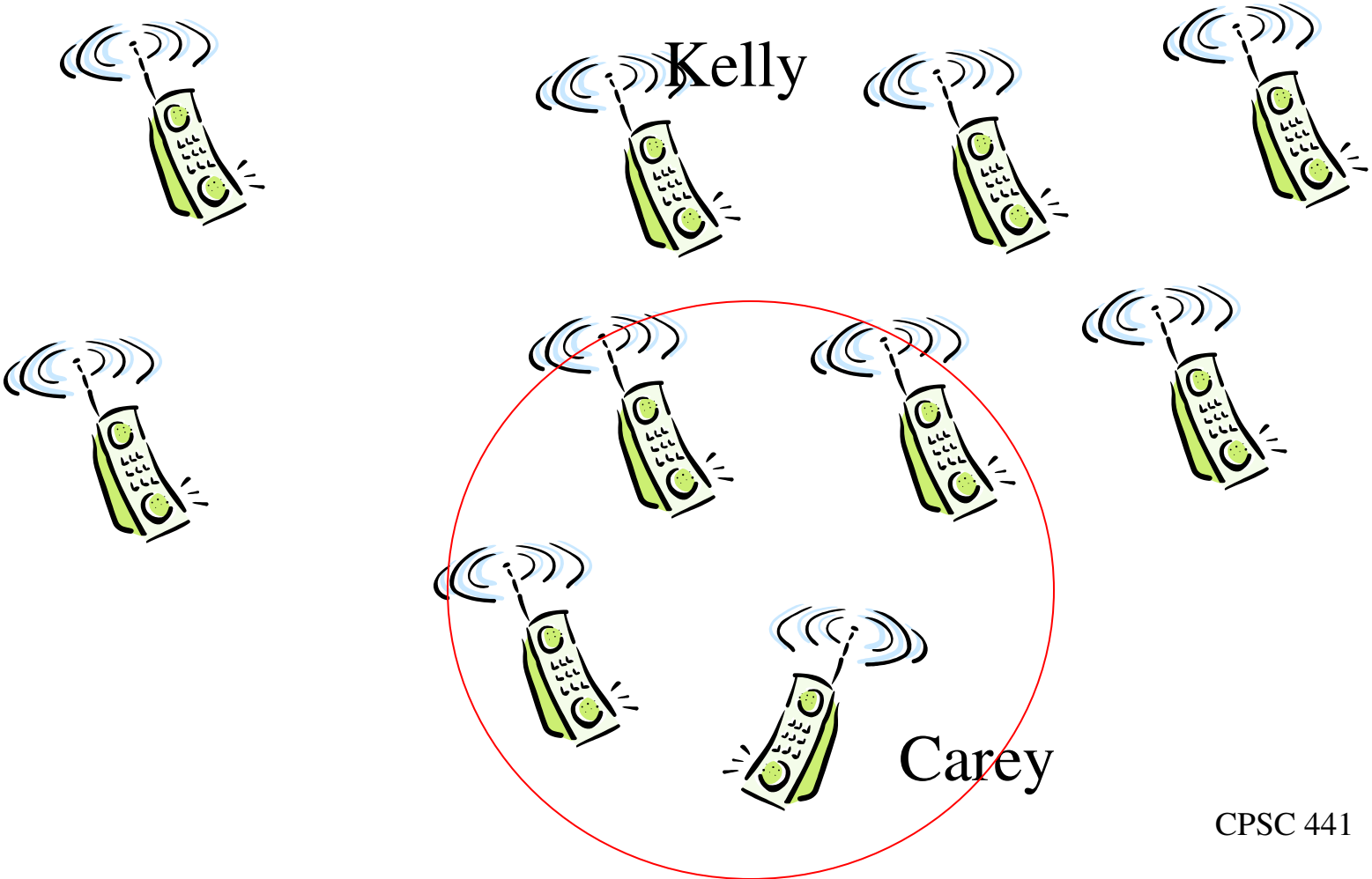
Example #3

- Multi-hop "ad hoc" networking



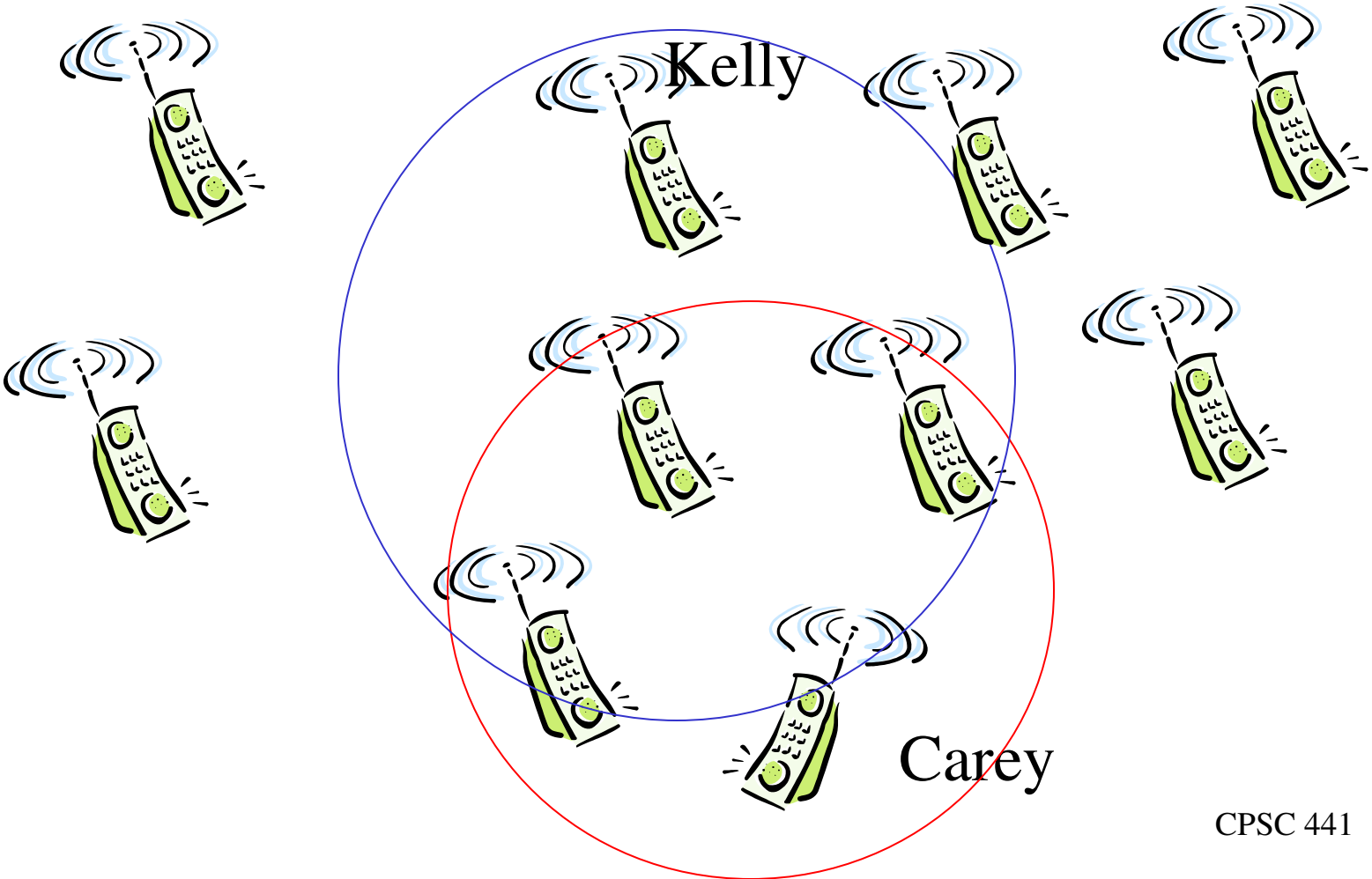
Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



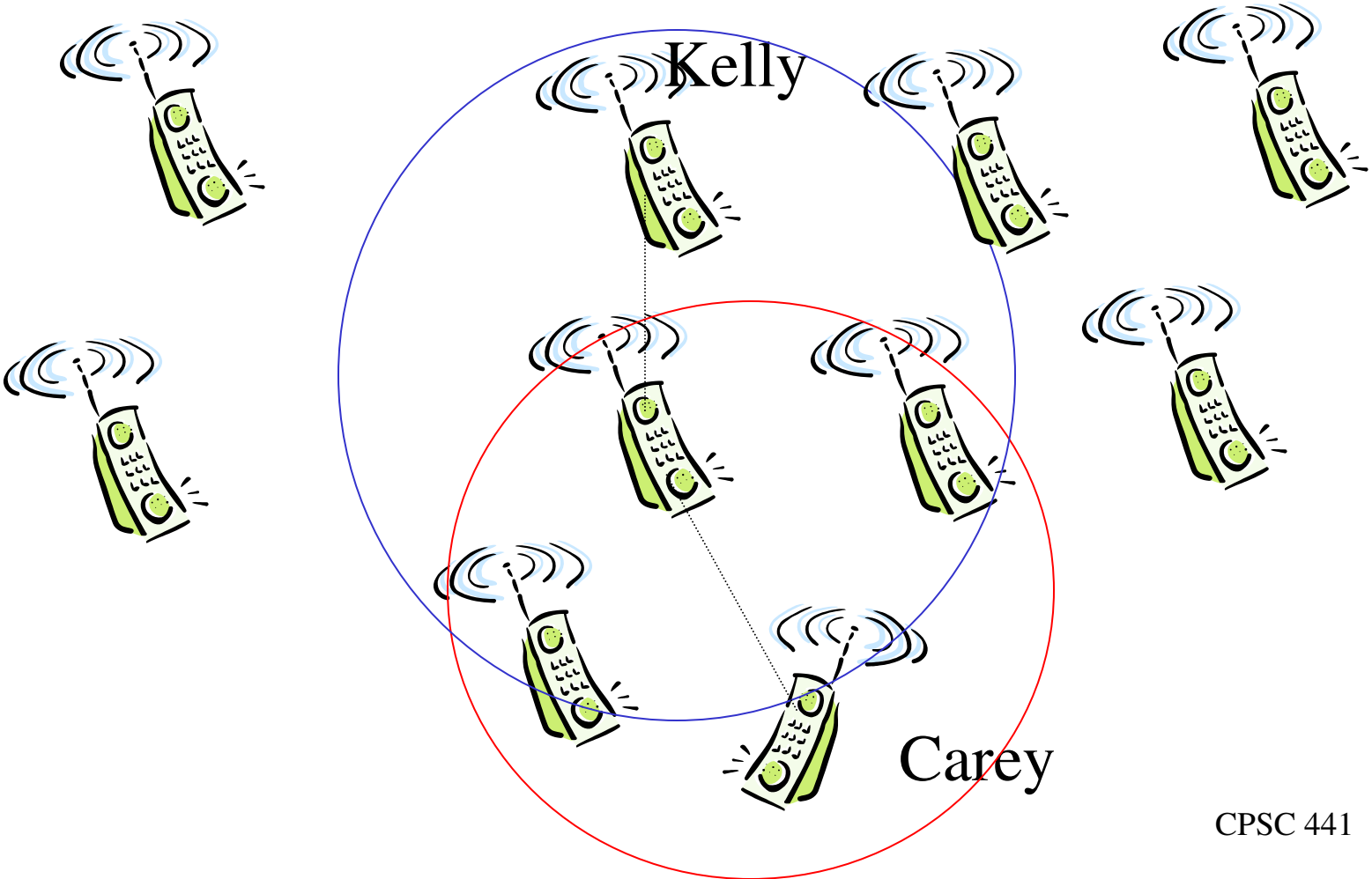
Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



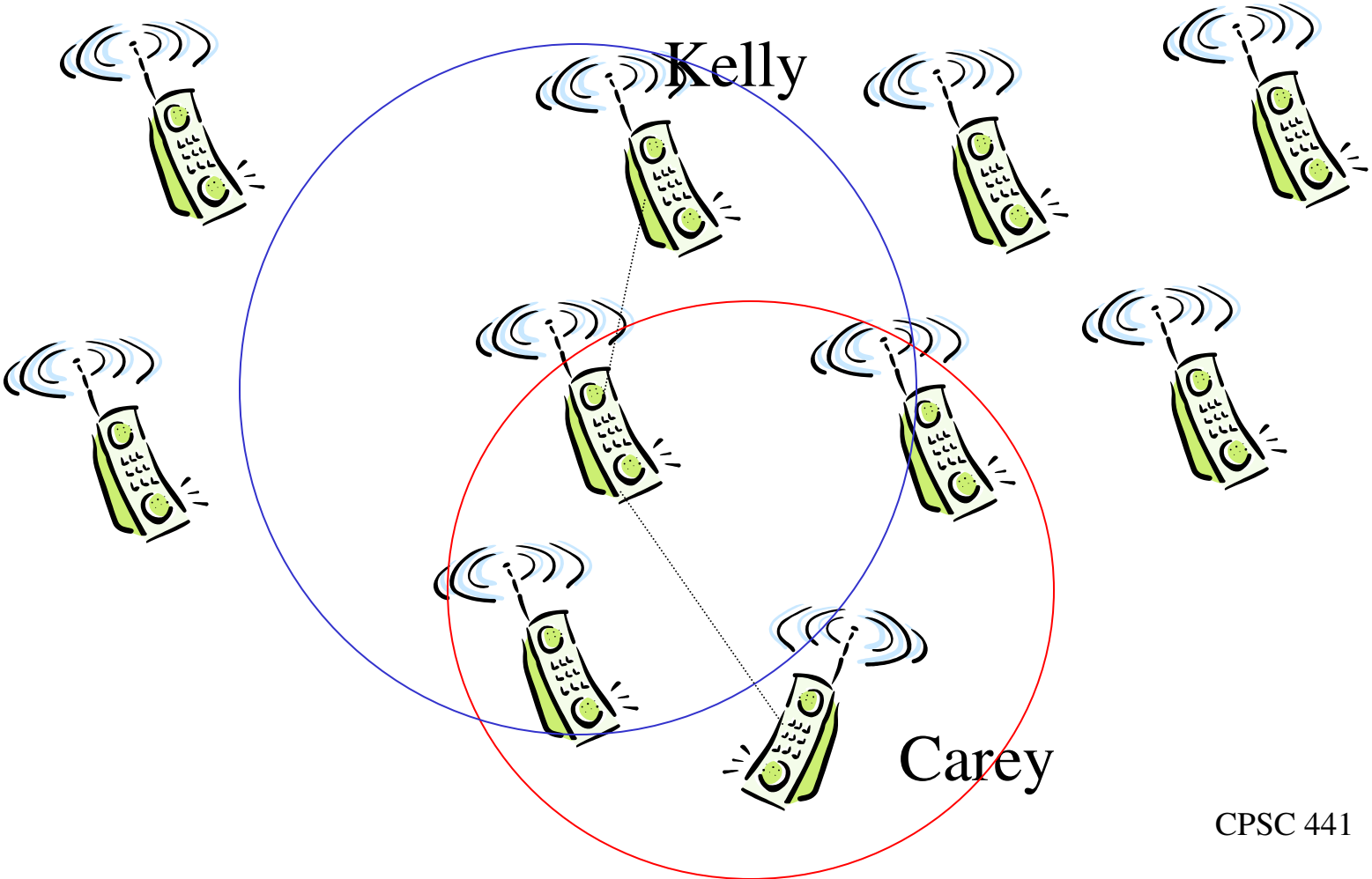
Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



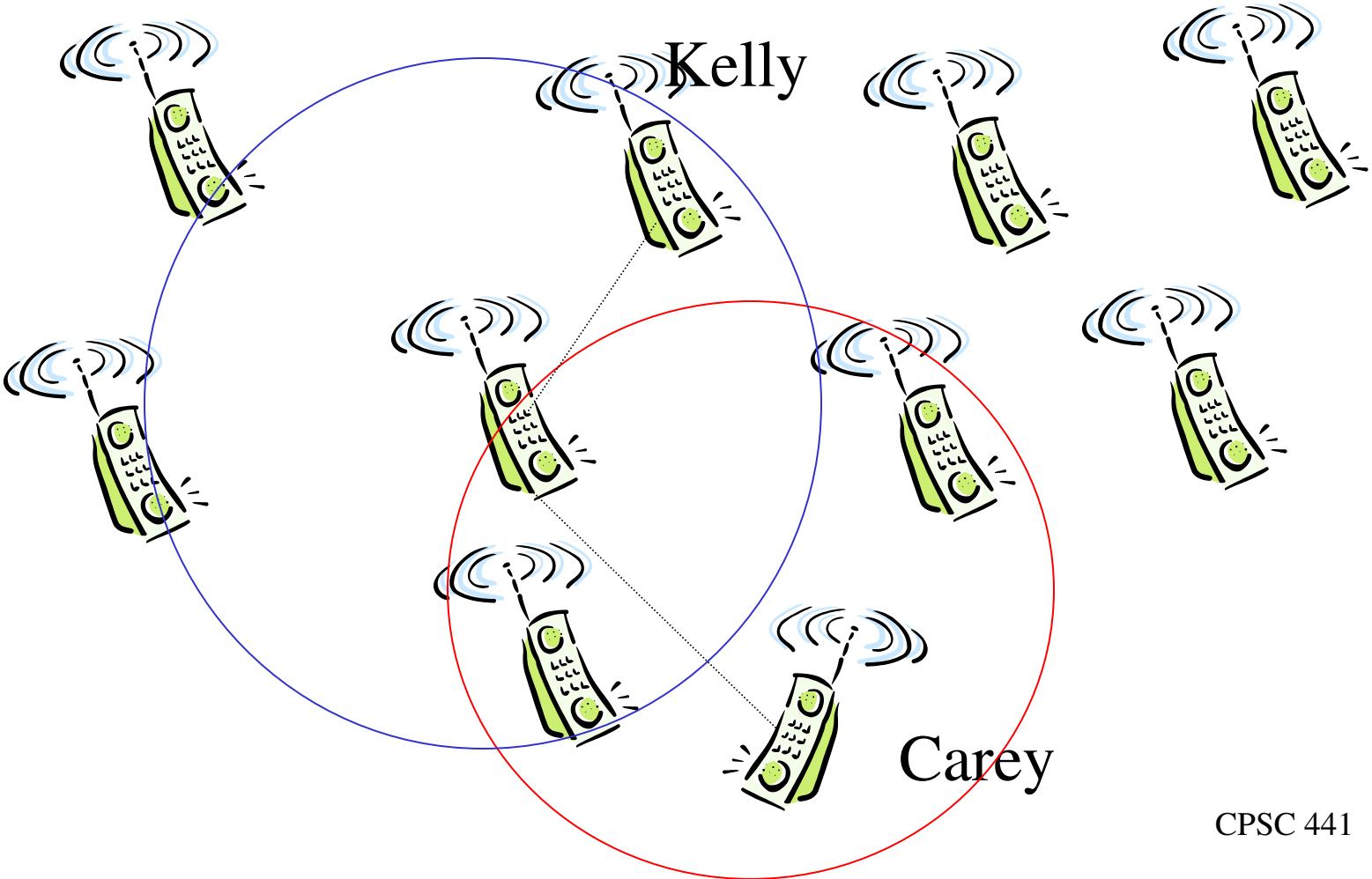
Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



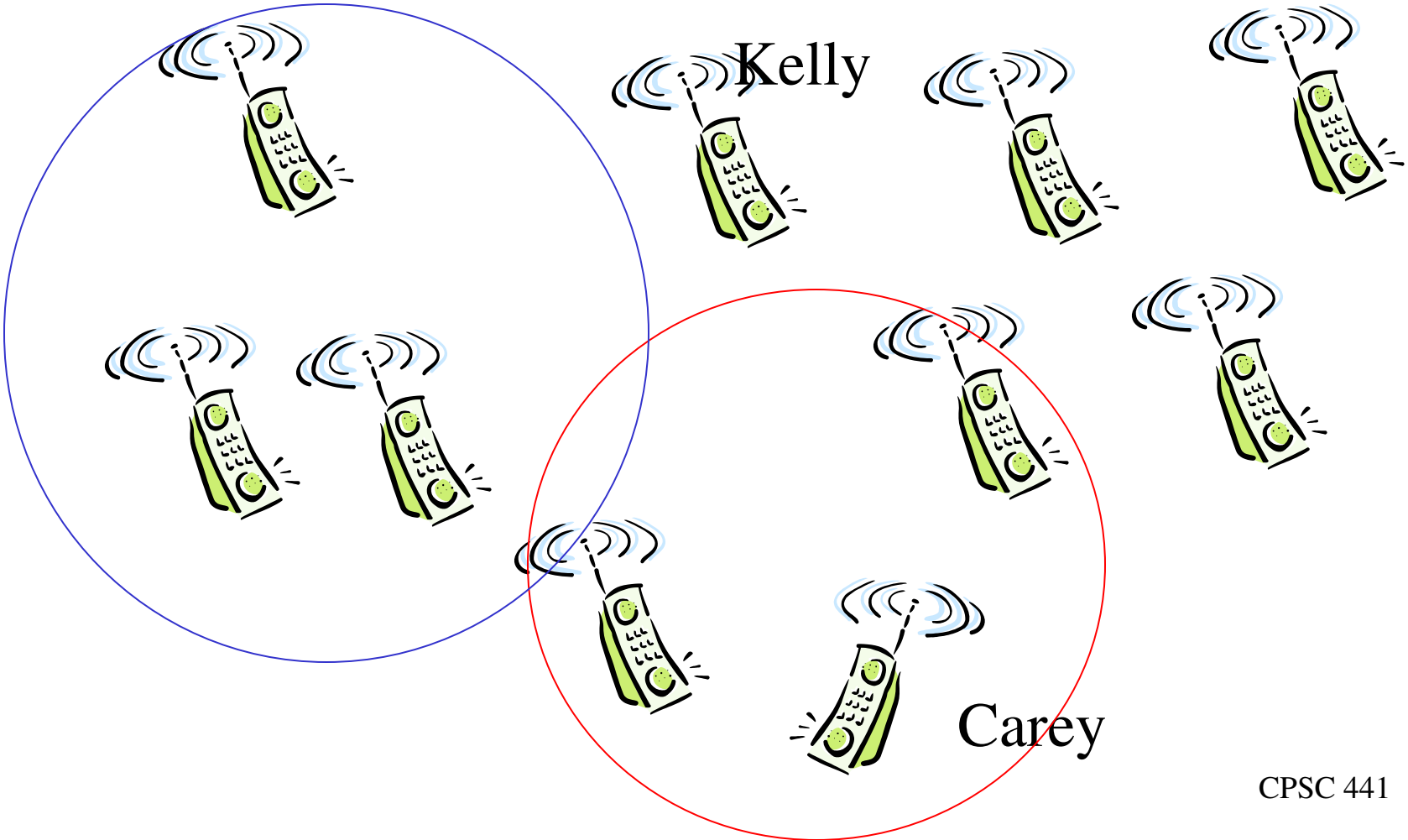
Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



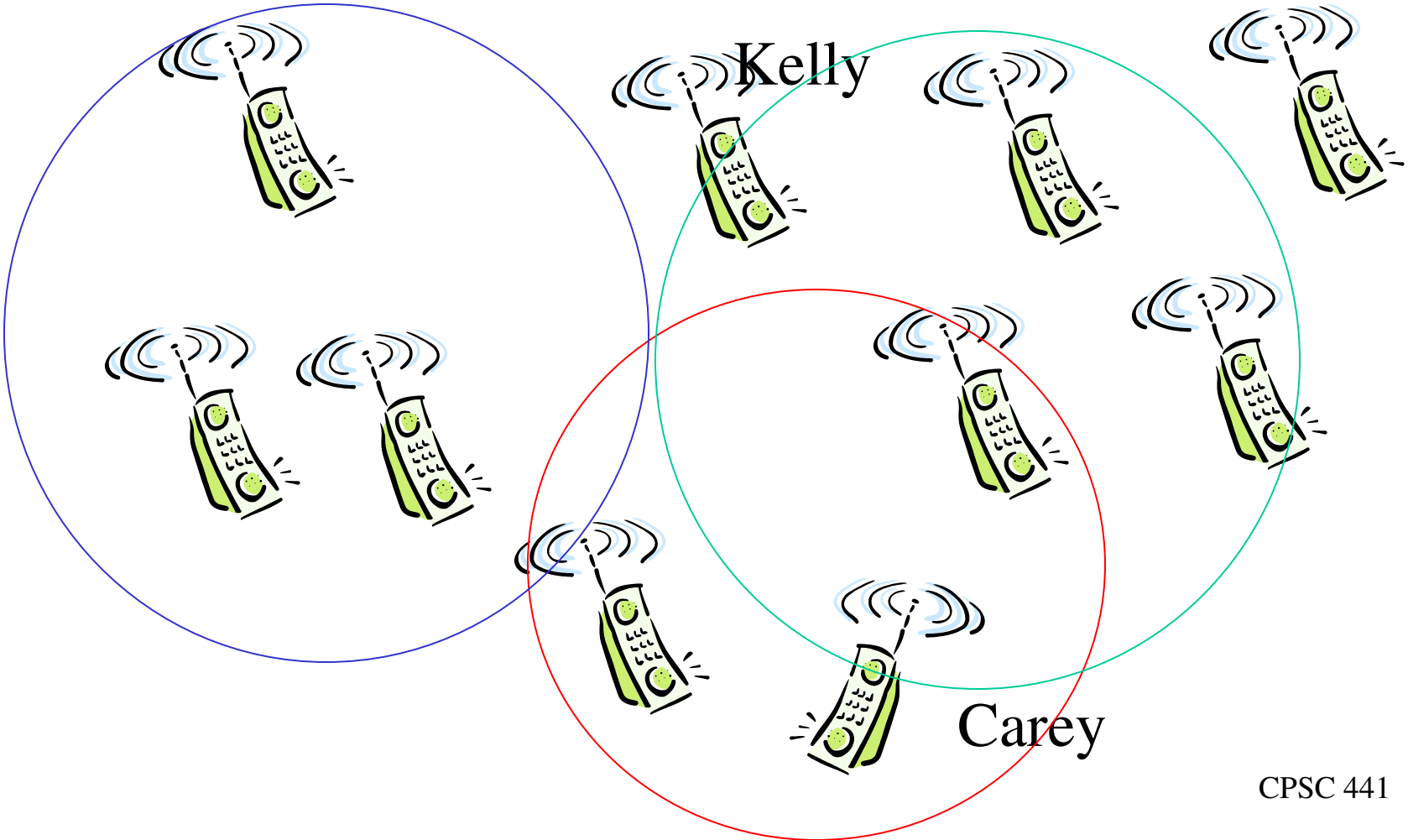
Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



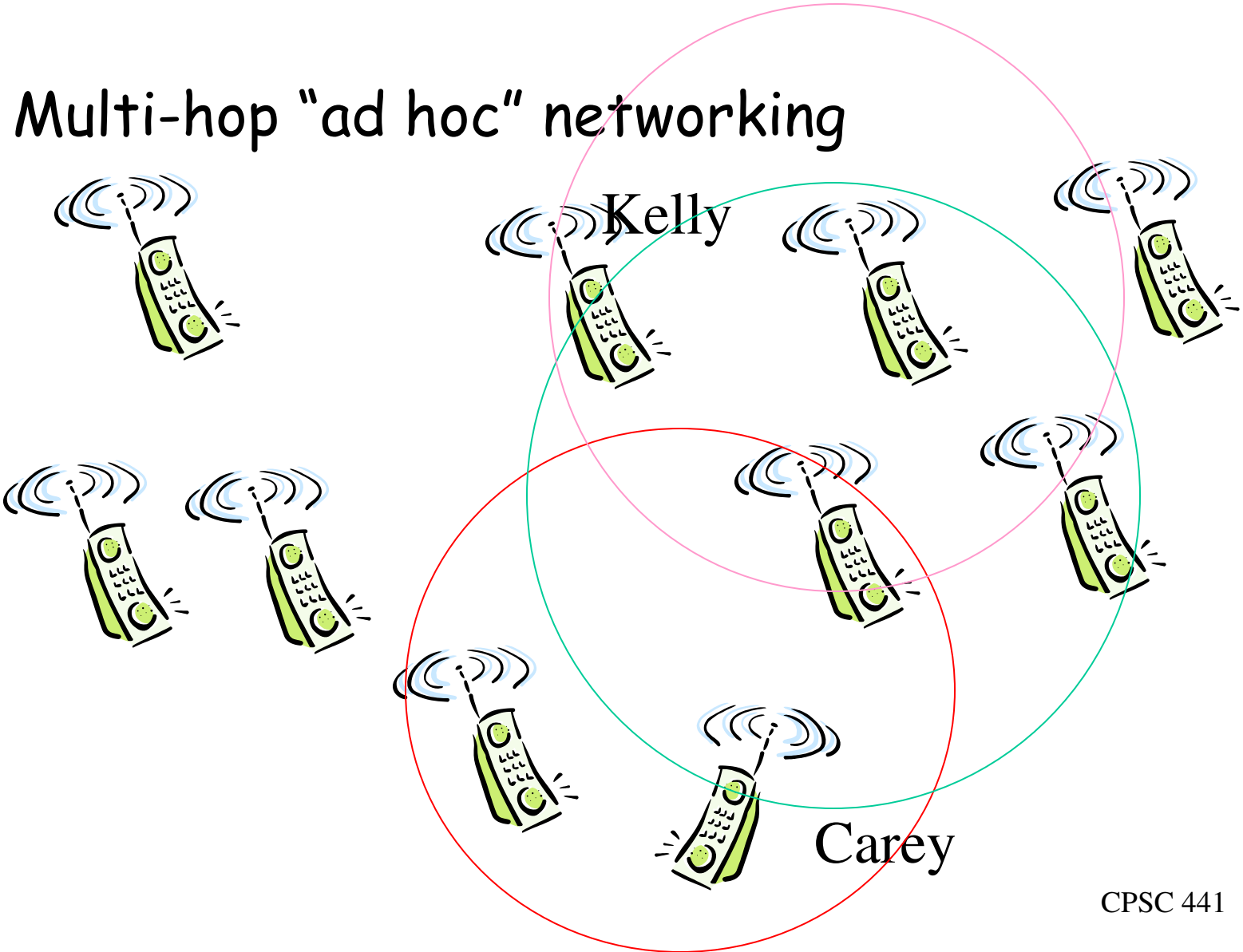
Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



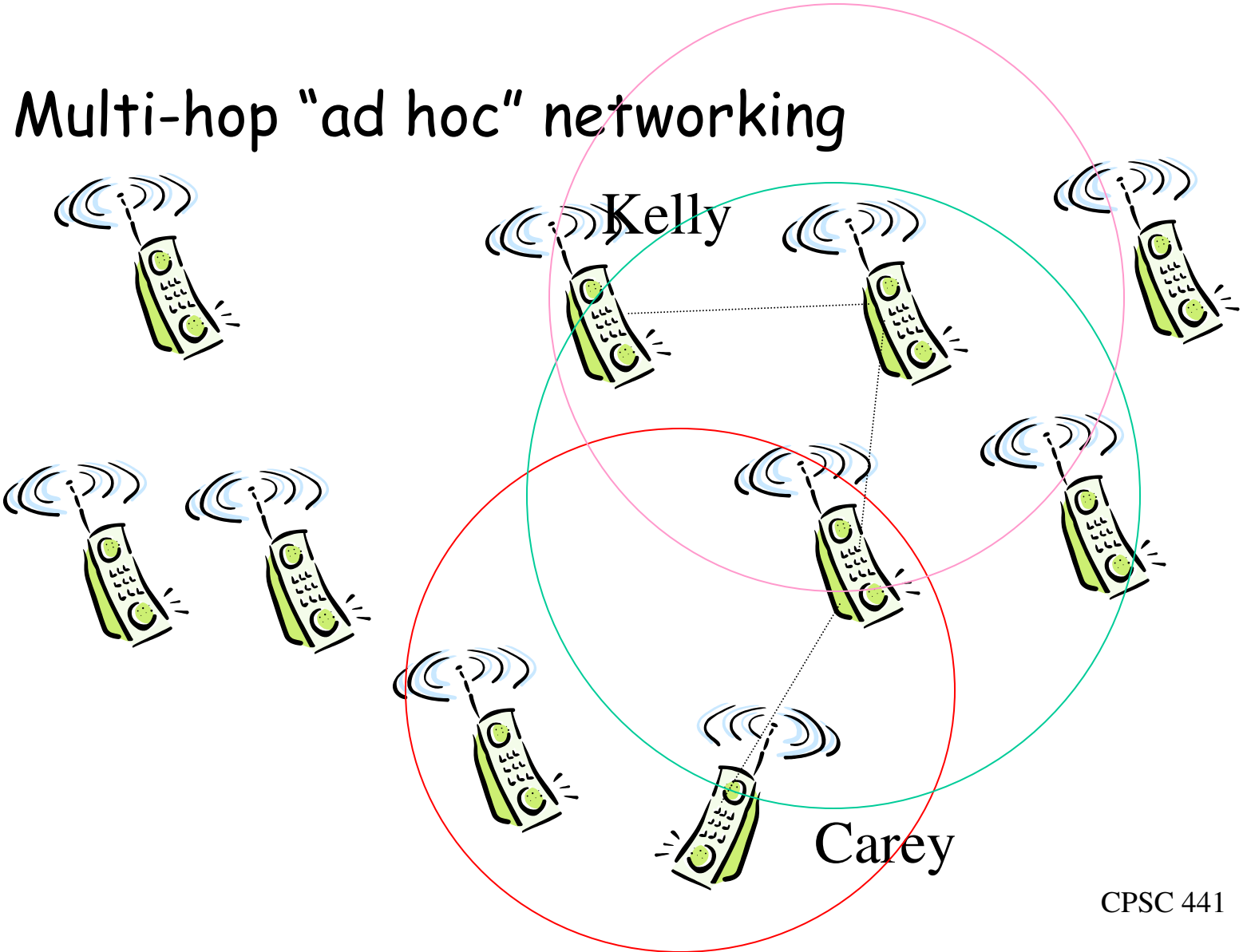
Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



Example #3 (Cont'd)

- Multi-hop "ad hoc" networking



Example #3 (Cont'd)

- Two interesting subproblems:
  - Dynamic ad hoc routing: node movement can disrupt the IP routing path at any time, disrupting TCP connection; yet another way to lose packets!!!; possible solution: Explicit Loss Notification (ELN)? Handoff? Route prediction?
  - TCP flow control: the bursty nature of TCP packet transmissions can create contention for the shared wireless channel among forwarding nodes; collisions between DATA and ACKs possible solution: rate-based flow control? Burst mode? Spatial reuse of channels?

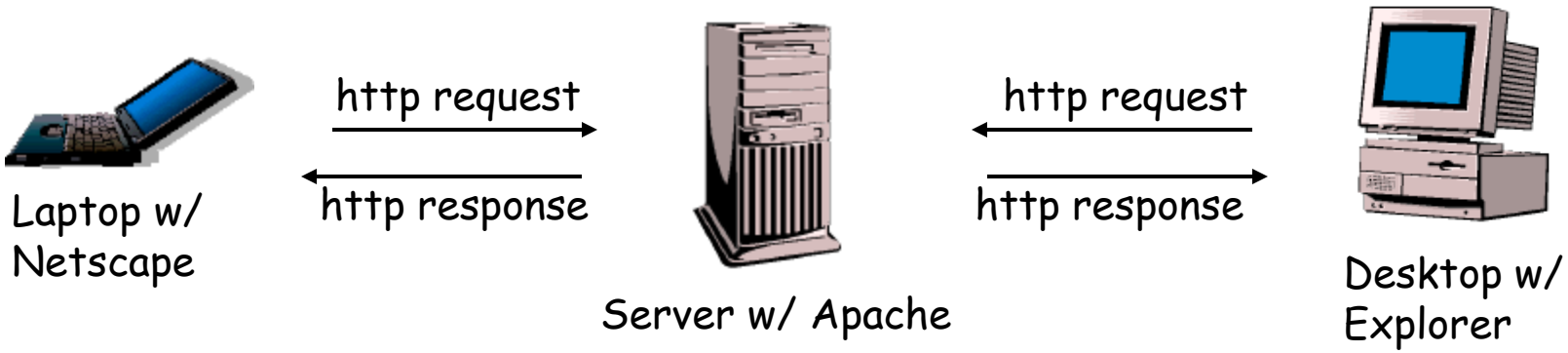
Summary of Wireless TCP

- TCP is the "four wheel drive" of TP's
- Wireless is a newly emerging technology with rapidly growing deployment popularity
- "TCP" and "Wireless" don't fit together all that well
- Making TCP smarter about wireless helps!

HTTP Protocol

- Slides originally from Williamson at Calgary
- Minor modifications are made

Introduction to HTTP



- HTTP: HyperText Transfer Protocol
  - Communication protocol between clients and servers
  - Application layer protocol for WWW
- Client/Server model:
  - Client: browser that requests, receives, displays object
  - Server: receives requests and responds to them
- Protocol consists of various operations
  - Few for HTTP 1.0 (RFC 1945, 1996)
  - Many more in HTTP 1.1 (RFC 2616, 1999)

Request Generation

- User clicks on something
- Uniform Resource Locator (URL):
  - `http://www.cnn.com`
  - `http://www.cpsc.ucalgary.ca`
  - `https://www.paymybills.com`
  - `ftp://ftp.kernel.org`
- Different URL schemes map to different services
- Hostname is converted from a name to a 32-bit IP address (DNS lookup, if needed)
- Connection is established to server (TCP)

What Happens Next?

- Client downloads HTML document
  - Sometimes called "container page"
  - Typically in text format (ASCII)
  - Contains instructions for rendering
    (e.g., background color, frames)
  - Links to other pages

- Many have embedded objects:
  - Images: GIF, JPG (logos, banner ads)
  - Usually automatically retrieved
    - I.e., without user involvement
    - can control sometimes
      (e.g. browser options, junkbusters)

```
<html>
<head>
<meta
name="Author"
content="Erich Nahum">
<title> Linux Web
Server Performance
</title>
</head>
<body text="#00000">
<img width=31
height=11
src="ibmlogo.gif">
<img
src="images/new.gif>
<h1>Hi There!</h1>
Here's lots of cool
linux stuff!
<a href="more.html">
Click here</a>
for more!
</body>
</html>
```

sample html file

Web Server Role

- Respond to client requests, typically a browser
  - Can be a proxy, which aggregates client requests (e.g., AOL)
  - Could be search engine spider or robot (e.g., Keynote)
- May have work to do on client's behalf:
  - Is the client's cached copy still good?
  - Is client authorized to get this document?
- Hundreds or thousands of simultaneous clients
- Hard to predict how many will show up on some day (e.g., "flash crowds", diurnal cycle, global presence)
- Many requests are in progress concurrently

HTTP Request Format

```
GET /images/penguin.gif HTTP/1.0
User-Agent: Mozilla/0.9.4 (Linux 2.2.19)
Host: www.kernel.org
Accept: text/html, image/gif, image/jpeg
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: B=xh203jfsf; Y=3sdkfjej
<cr><lf>
```

- Messages are in ASCII (human-readable)
- Carriage-return and line-feed indicate end of headers
- Headers may communicate private information
  (browser, OS, cookie information, etc.)

Request Types

Called Methods:

- GET: retrieve a file (95% of requests)
- HEAD: just get meta-data (e.g., mod time)
- POST: submitting a form to a server
- PUT: store enclosed document as URI
- DELETE: removed named resource
- LINK/UNLINK: in 1.0, gone in 1.1
- TRACE: http "echo" for debugging (added in 1.1)
- CONNECT: used by proxies for tunneling (1.1)
- OPTIONS: request for server/proxy options (1.1)

Response Format

```
HTTP/1.0 200 OK
Server: Tux 2.0
Content-Type: image/gif
Content-Length: 43
Last-Modified: Fri, 15 Apr 1994 02:36:21 GMT
Expires: Wed, 20 Feb 2002 18:54:46 GMT
Date: Mon, 12 Nov 2001 14:29:48 GMT
Cache-Control: no-cache
Pragma: no-cache
Connection: close
Set-Cookie: PA=wefj2we0-jfjf
<cr><lf>
<data follows…>
```

- Similar format to requests (i.e., ASCII)

Response Types

- 1XX: Informational (def'd in 1.0, used in 1.1)

  `100 Continue, 101 Switching Protocols`

- 2XX: Success

  `200 OK, 206 Partial Content`

- 3XX: Redirection

  `301 Moved Permanently, 304 Not Modified`

- 4XX: Client error

  `400 Bad Request, 403 Forbidden, 404 Not Found`

- 5XX: Server error

  `500 Internal Server Error, 503 Service Unavailable, 505 HTTP Version Not Supported`

Outline of an HTTP Transaction

- This section describes the basics of servicing an HTTP GET request from user space
- Assume a single process running in user space, similar to Apache 1.3
- We'll mention relevant socket operations along the way

```
initialize;
forever do {
  get request;
  process;
  send response;
  log request;
}
```

server in a nutshell

Readying a Server

```
s = socket(); /* allocate listen socket */
bind(s, 80);  /* bind to TCP port 80    */
listen(s);    /* indicate willingness to accept */
while (1) {
    newconn = accept(s); /* accept new connection */b
```

- First thing a server does is notify the OS it is interested in WWW server requests; these are typically on TCP port 80. Other services use different ports (e.g., SSL is on 443)
- Allocate a socket and bind()'s it to the address (port 80)
- Server calls listen() on the socket to indicate willingness to receive requests
- Calls accept() to wait for a request to come in (and blocks)
- When the accept() returns, we have a new socket which represents a new connection to a client

Processing a Request

```
remoteIP = getsockname(newconn);
remoteHost = gethostbyname(remoteIP);
gettimeofday(currentTime);
read(newconn, reqBuffer, sizeof(reqBuffer));
reqInfo = serverParse(reqBuffer);
```

- getsockname() called to get the remote host name
  - for logging purposes (optional, but done by most)
- gethostbyname() called to get name of other end
  - again for logging purposes
- gettimeofday() is called to get time of request
  - both for Date header and for logging
- read() is called on new socket to retrieve request
- request is determined by parsing the data
  - "GET /images/jul4/flag.gif"

Processing a Request (cont)

```
fileName = parseOutFileName(requestBuffer);
fileAttr = stat(fileName);
serverCheckFileStuff(fileName, fileAttr);
open(fileName);
```

- stat() called to test file path
  - to see if file exists/is accessible
  - may not be there, may only be available to certain people
  - "/microsoft/top-secret/plans-for-world-domination.html"
- stat() also used for file meta-data
  - e.g., size of file, last modified time
  - "Has file changed since last time I checked?"
- might have to stat() multiple files and directories
- assuming all is OK, open() called to open the file

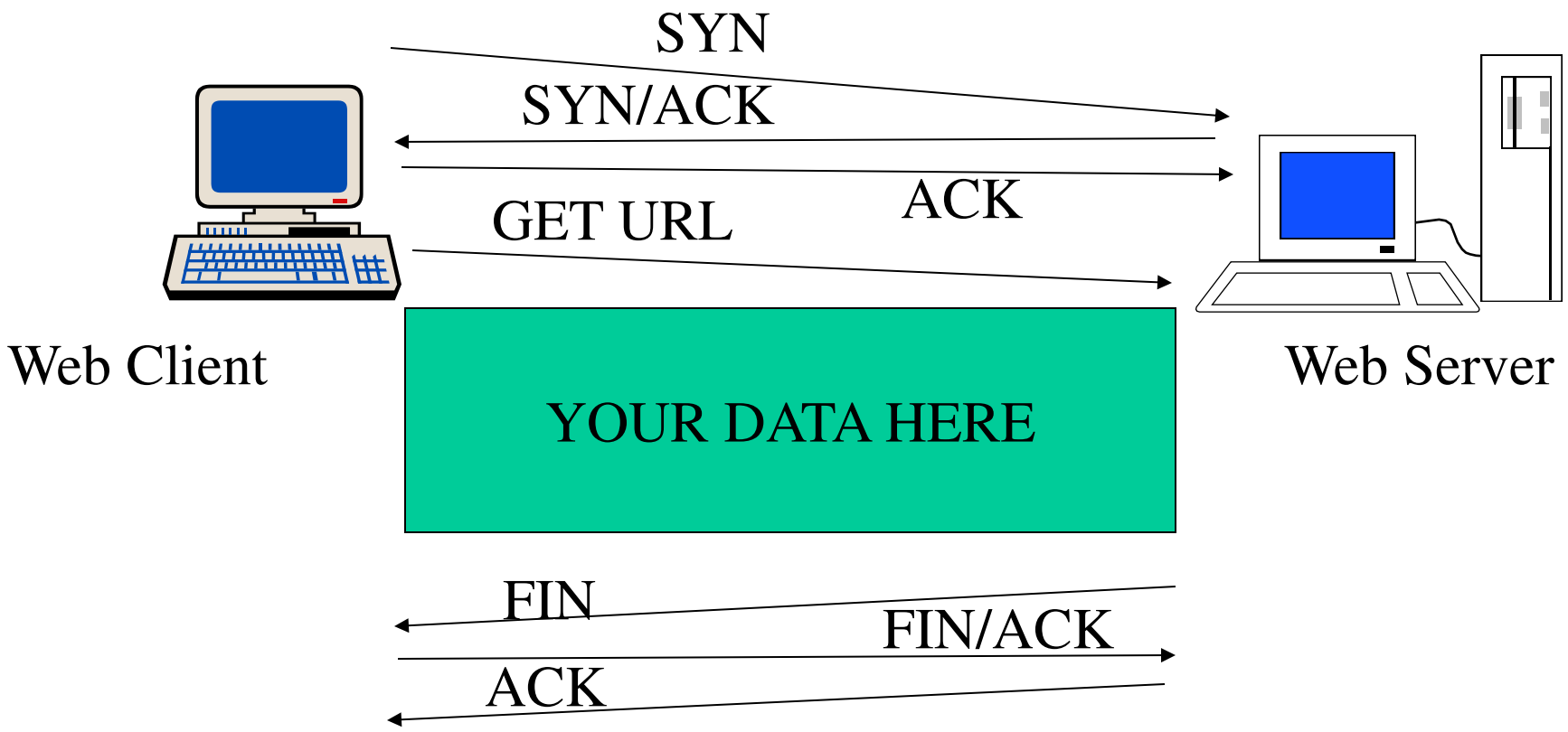
Responding to a Request

```
read(fileName, fileBuffer);
headerBuffer = serverFigureHeaders(fileName, reqInfo);
write(newSock, headerBuffer);
write(newSock, fileBuffer);
close(newSock);
close(fileName);
write(logFile, requestInfo);
```

- read() called to read the file into user space
- write() is called to send HTTP headers on socket
  (early servers called write() for *each header!*)
- write() is called to write the file on the socket
- close() is called to close the socket
- close() is called to close the open file descriptor
- write() is called on the log file

Network View: HTTP and TCP

- TCP is a connection-oriented protocol



Example Web Page

page.html

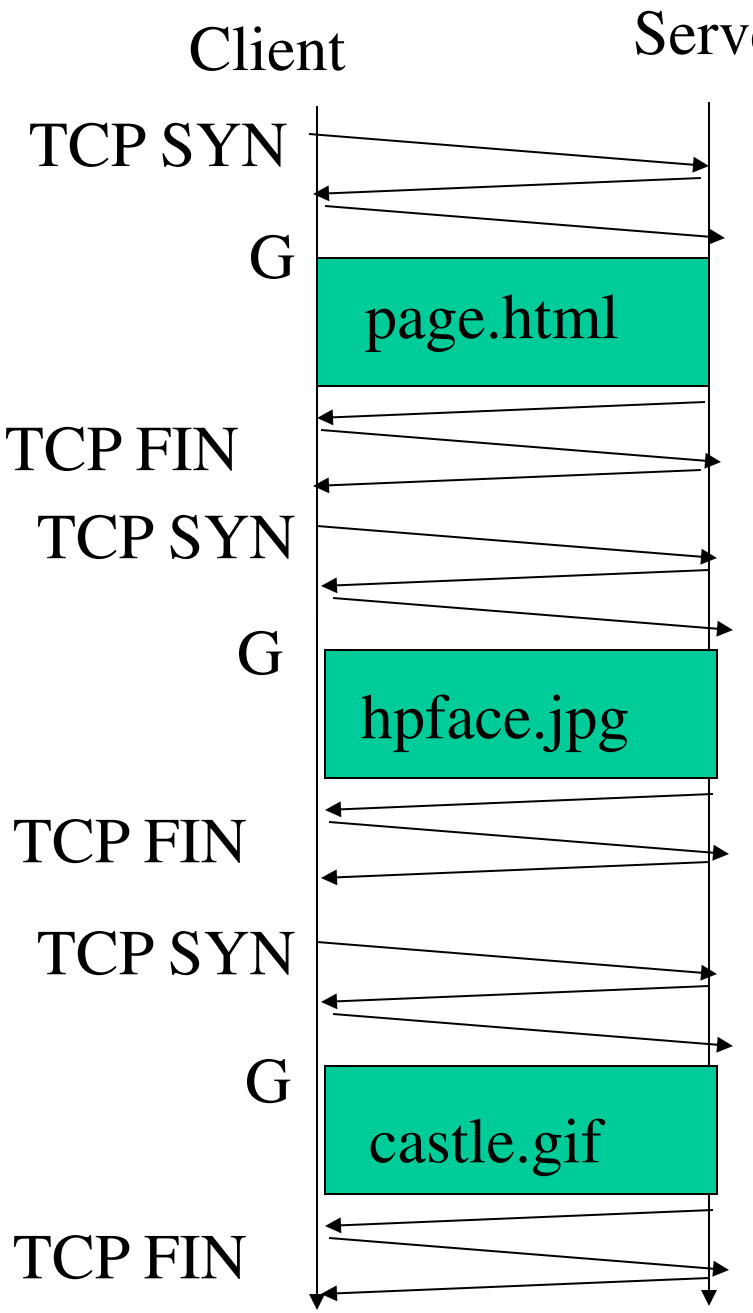
Harry Potter Movies

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

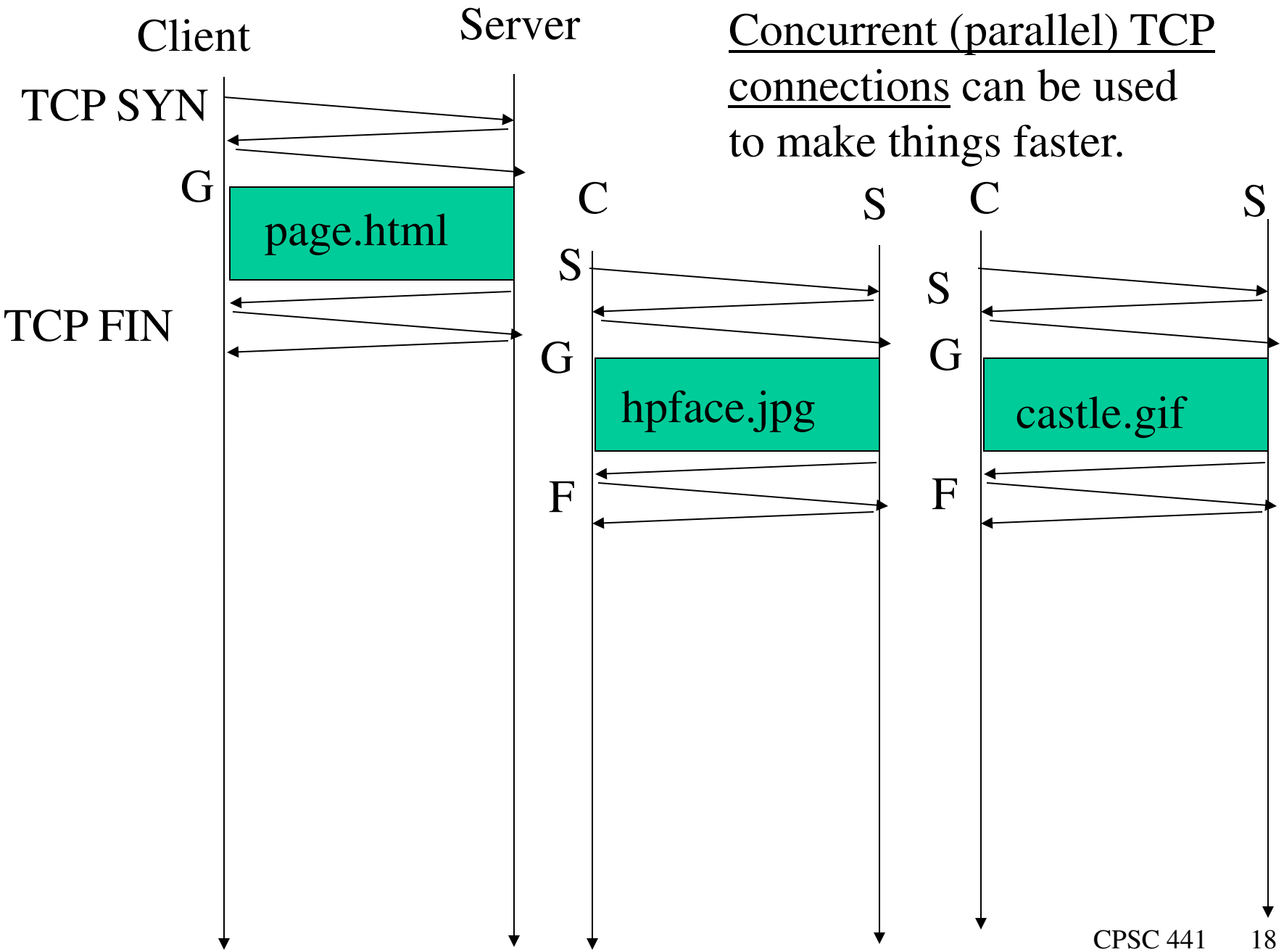
hpface.jpg

"Harry Potter and the Bathtub Ring"

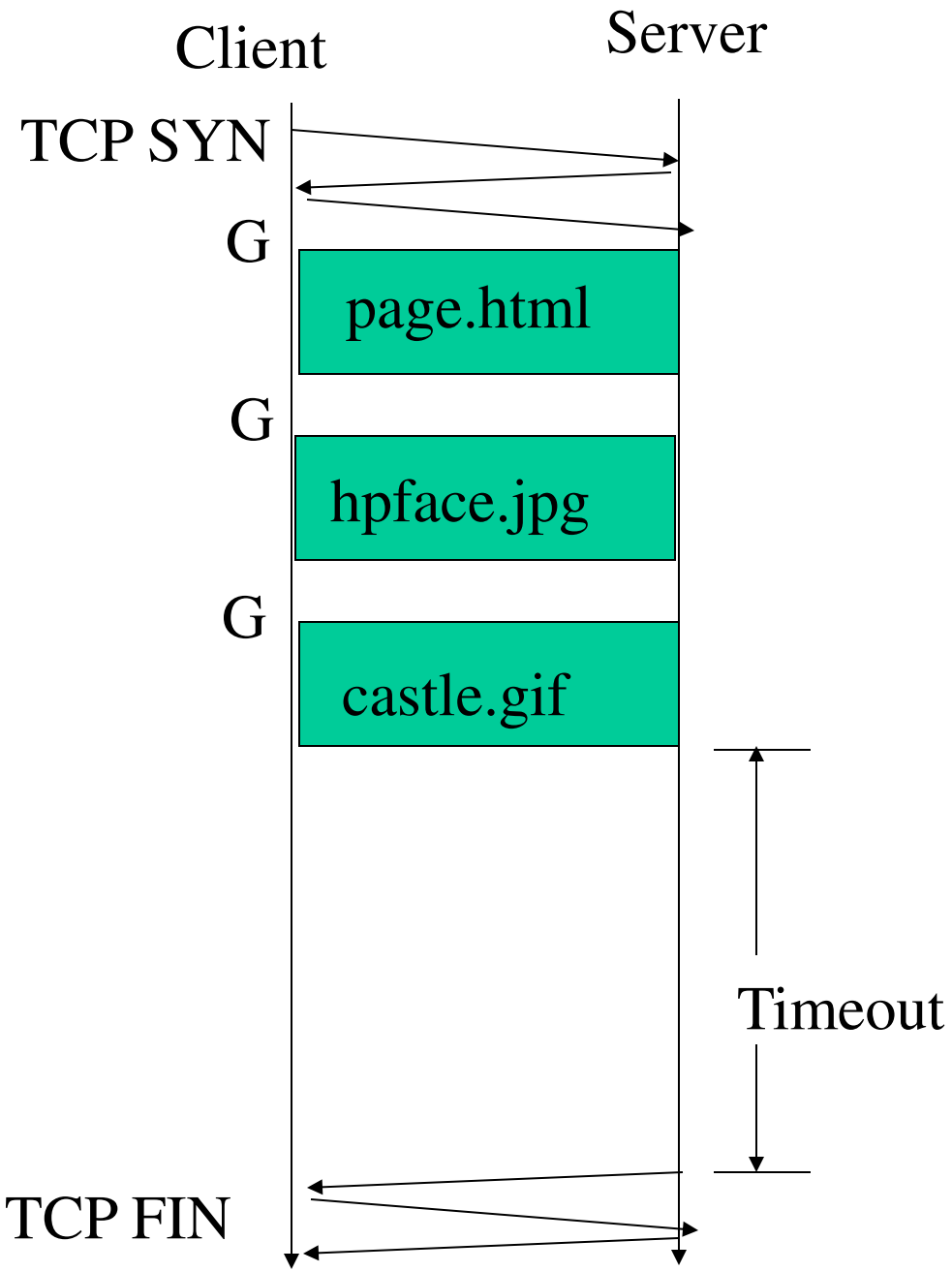
castle.gif



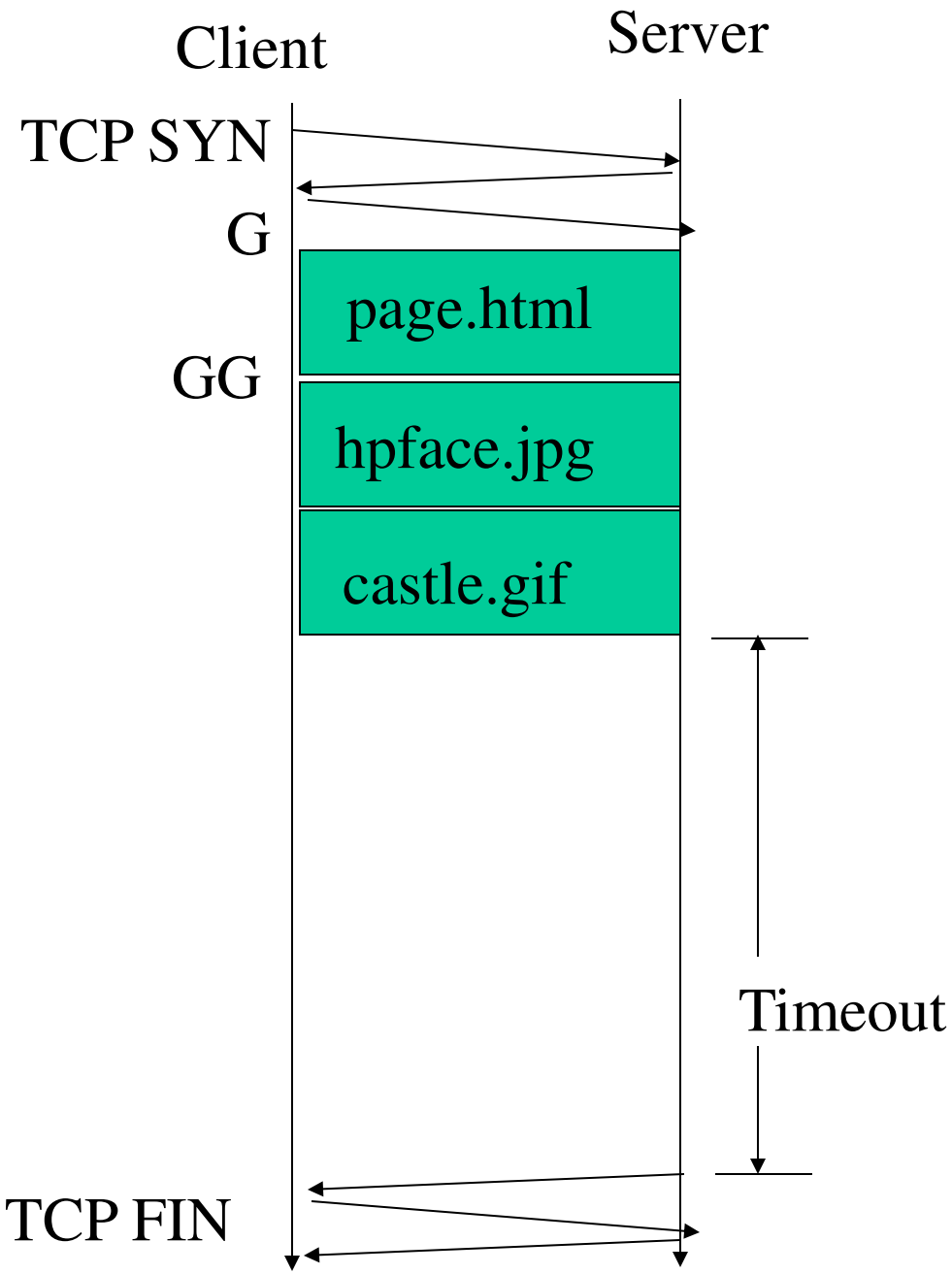
The "classic" approach in HTTP/1.0 is to use one HTTP request per TCP connection, serially.



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.

Summary of Web and HTTP

- The major application on the Internet
  - Majority of traffic is HTTP (or HTTP-related)
- Client/server model:
  - Clients make requests, servers respond to them
  - Done mostly in ASCII text (helps debugging!)
- Various headers and commands
  - Too many to go into detail here
  - Many web books/tutorials exist (e.g., Krishnamurthy & Rexford 2001)

HTML5

Gaurav Jaiswal

Singsys Pte. Ltd.

singsys

What is HTML5?

What is HTML5?

HTML5 is the new standard for HTML.

The previous version of HTML was – HTML 4.01, came in 1999.

HTML5 is designed to deliver almost everything you want to do online without requiring additional plugins. It does everything from animation to apps, music to movies, and can also be used to build complicated applications that run in your browser.

HTML5 is also cross-platform (it does not care whether you are using a tablet or a smartphone, a notebook, notebook or a Smart TV).

Differences Between HTML4 and HTML5

Differences Between HTML4 & HTML5

1. HTML5 is a work in progress
2. Simplified Syntax
3. The New <canvas> Element for 2D drawings
4. New content-specific elements, like <article>, <header>, <footer>, <nav>, <section>
5. New <menu> and <figure> Elements
6. New <audio> and <video> Elements
7. New form controls, like calendar, date, time, email, url, search
8. No More <frame>, , <big>, and <b>, <font>
9. Support for local storage

Browser Support for HTML5

Browser Support for HTML5

HTML5 is not yet an official standard, and no browsers have full HTML5 support.

But all major browsers (Safari, Chrome, Firefox, Opera, Internet Explorer) continue to add new HTML5 features to their latest versions.

HTML5 Document

The HTML5 <!DOCTYPE>

In HTML5 there is only one **<!doctype>** declaration, and it is very simple:

```
<!DOCTYPE html>
```

Minimum HTML5 Document

Below is a simple HTML5 document, with the minimum of required tags:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document.....
</body>

</html>
```

HTML5 New Elements

The New <canvas> Element

The <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).

New Media Elements

Tag	Description
<audio>	Defines sound content
<video>	Defines a video or movie
<source>	Defines multiple media resources for <video> and <audio>
<embed>	Defines a container for an external application or interactive content (a plug-in)
<track>	Defines text tracks for <video> and <audio>

New Form Elements

Tag	Description
<datalist>	Specifies a list of pre-defined option for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

New Semantic/Structural Elements

Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<bdi>	Isolates a part of text that might be formatted in a different direction from other text outside it
<command>	Defines a command button that a user can invoke
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window
<summary>	Defines a visible heading for a <details> element

New Semantic/Structural Elements

Tag	Description
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<figcaption>	Defines a caption for a <figure> element
<footer>	Defines a footer for a document or section
<header>	Defines a header for a document or section
<mark>	Defines marked/highlighted text
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links

New Semantic/Structural Elements

Tag	Description
<progress>	Represents the progress of a task
<ruby>	Defines a ruby annotation (for East Asian typography)
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<rp>	Defines what to show in browsers that do not support ruby annotations
<section>	Defines a section in a document
<time>	Defines a date/time
<wbr>	Defines a possible line-break

Removed Elements

The following HTML 4.01 elements are removed from HTML5:

- ✓ <acronym>
- ✓ <applet>
- ✓ <basefont>
- ✓ <big>
- ✓
- ✓ <dir>
- ✓ <font>
- ✓ <frame>
- ✓ <frameset>
- ✓ <noframes>
- ✓ <strike>
- ✓ <tt>

HTML5 Canvas

HTML5 Canvas

- ✓ The HTML5 <canvas> element is used to draw graphics, on the fly, via scripting (usually JavaScript).
- ✓ The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics.
- ✓ Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

HTML5 Inline SVG

HTML5 Inline SVG

- ✓SVG stands for Scalable Vector Graphics
- ✓SVG is used to define vector-based graphics for the Web
- ✓SVG defines the graphics in XML format
- ✓SVG graphics do NOT lose any quality if they are zoomed or resized
- ✓Every element and every attribute in SVG files can be animated
- ✓SVG is a W3C recommendation

SVG Advantages

Advantages of using SVG over other image formats (like JPEG and GIF) are:

- ✓SVG images can be created and edited with any text editor
- ✓SVG images can be searched, indexed, scripted, and compressed
- ✓SVG images are scalable
- ✓SVG images can be printed with high quality at any resolution
- ✓SVG images are zoomable (and the image can be zoomed without degradation)

Difference Between SVG & Canvas

Canvas	SVG
Resolution dependent	Resolution independent
No support for event handlers	Support for event handlers
Poor text rendering capabilities	Best suited for applications with large rendering areas (Google Maps)
You can save the resulting image as .png or .jpg	Slow rendering if complex (anything that uses the DOM a lot will be slow)
Well suited for graphic-intensive games	Not suited for game applications

HTML5 Geolocation

HTML5 Geolocation

- ✓ The HTML5 Geolocation API is used to get the geographical position of a user.
- ✓ Since this can compromise user privacy, the position is not available unless the user approves it.

Information you get from Geolocation API

Property	Description
coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitudeAccuracy	The altitude accuracy of position
coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the response

HTML5 Video

HTML5 Video

Many modern websites show videos. HTML5 provides a standard for showing them.

Video Formats and Browser Support

Browser	MP4	WebM	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	NO **Update:** Firefox 21 running on Windows 7, Windows 8, Windows Vista, and Android now supports MP4	YES	YES
Safari	YES	NO	NO
Opera	NO	YES	YES

singsys

HTML5 Audio

HTML5 Audio

HTML5 provides a standard for playing audio files.

Audio Formats and Browser Support

Browser	MP3	Wav	Ogg
Internet Explorer	YES	NO	NO
Chrome	YES	YES	YES
Firefox	NO **Update:** Firefox 21 running on Windows 7, Windows 8, Windows Vista, and Android now supports MP3	YES	YES
Safari	YES	YES	NO
Opera	NO	YES	YES

HTML5 Input Types

HTML5 Input Types

HTML5 has several new input types for forms. These new features allow better input control and validation.

- ✓ color
- ✓ Date
- ✓ datetime
- ✓ datetime-local
- ✓ email
- ✓ month
- ✓ number
- ✓ range
- ✓ search
- ✓ tel
- ✓ time
- ✓ url
- ✓ week

HTML5 Form Elements

HTML5 Form Elements

HTML5 has the following new form elements:

- ✓ <datalist>
- ✓ <keygen>
- ✓ <output>

HTML5 <datalist> Element

The <datalist> element specifies a list of pre-defined options for an <input> element.

The <datalist> element is used to provide an "autocomplete" feature on <input> elements. Users will see a drop-down list of pre-defined options as they input data.

Use the <input> element's list attribute to bind it together with a <datalist> element.

HTML5 <keygen> Element

- ✓The purpose of the <keygen> element is to provide a secure way to authenticate users.
- ✓The <keygen> tag specifies a key-pair generator field in a form.
- ✓When the form is submitted, two keys are generated, one private and one public.
- ✓The private key is stored locally, and the public key is sent to the server. The public key could be used to generate a client certificate to authenticate the user in the future.

HTML5 <output> Element

HTML5 <output> Element

The <output> element represents the result of a calculation (like one performed by a script).

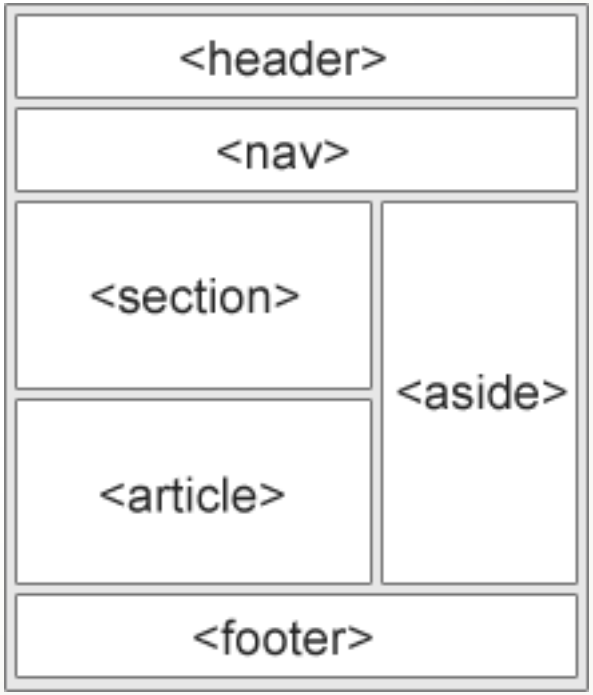
HTML5 Semantic Elements

HTML5 Semantic Elements

- ✓ A semantic element clearly describes its meaning to both the browser and the developer.
- ✓ Examples of **non-semantic** elements: and - Tells nothing about its content.
- ✓ Examples of **semantic** elements: <form>, <table>, and <img> - Clearly defines its content.

HTML5 Semantic Elements

- ✓ HTML5 offers new semantic elements to clearly define different parts of a web page:
- ✓ <header>
- ✓ <nav>
- ✓ <section>
- ✓ <article>
- ✓ <aside>
- ✓ <figcaption>
- ✓ <figure>
- ✓ <footer>



HTML5 Web Storage

HTML5 Web Storage

- ✓With HTML5, web pages can store data locally within the user's browser.
- ✓Earlier, this was done with cookies. However, Web Storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for. It is also possible to store large amounts of data, without affecting the website's performance.
- ✓The data is stored in key/value pairs, and a web page can only access data stored by itself.

HTML5 Web Storage

- There are two new objects for storing data on the client:
  - localStorage - stores data with no expiration date
  - sessionStorage - stores data for one session
- The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the browser window.

HTML5 Application Cache

HTML5 Application Cache

HTML5 introduces application cache, which means that a web application is cached, and accessible without an internet connection.

- ✓ Application cache gives an application three advantages:
- ✓ Offline browsing - users can use the application when they're offline
- ✓ Speed - cached resources load faster
- ✓ Reduced server load - the browser will only download updated/changed resources from the server

HTML5 Cache Manifest Example

The example below shows an HTML document with a cache manifest (for offline browsing):

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">

<body>
The content of the document.....
</body>

</html>
```

Cache Manifest Basics

To enable application cache, include the manifest attribute in the document's <html> tag.

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

Every page with the manifest attribute specified will be cached when the user visits it. If the manifest attribute is not specified, the page will not be cached (unless the page is specified directly in the manifest file).

The recommended file extension for manifest files is: ".appcache”

The Manifest File

The manifest file is a simple text file, which tells the browser what to cache (and what to never cache).

The manifest file has three sections:

- ✓ **CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time
- ✓ **NETWORK** - Files listed under this header require a connection to the server, and will never be cached
- ✓ **FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

CACHE MANIFEST

The first line, CACHE MANIFEST, is required:

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

The manifest file above lists three resources: a CSS file, a GIF image, and a JavaScript file. When the manifest file is loaded, the browser will download the three files from the root directory of the web site. Then, whenever the user is not connected to the internet, the resources will still be available.

NETWORK

The NETWORK section below specifies that the file "login.asp" should never be cached, and will not be available offline.

```
NETWORK:
login.asp
```

An asterisk can be used to indicate that all other resources/files require an internet connection:

```
NETWORK:
*
```

FALLBACK

The FALLBACK section below specifies that "offline.html" will be served in place of all files in the /html/ catalog, in case an internet connection cannot be established:

FALLBACK:
/html/ /offline.html

Updating the Cache

Once an application is cached, it remains cached until one of the following happens:

- ✓The user clears the browser's cache
- ✓The manifest file is modified (see tip below)
- ✓The application cache is programmatically updated

Simple Sender / Receiver Interaction with PK

- Sender (Alice)
- Message to send = m
- Compute
  - h = Hash(m)
  - d = Alice(pri key) (h)
- Sender sends (m, d)To the receiver
- (m, d) goes through public network

- Coming out from public network …
- Receiver (Bob) receives (m2, d2)
- Compute
  - h2 = Hash(m2)
  - Alice(pub key) (d2) = x
- If x == h2 => Good: Nothing change and the message received is good
- If x <> h2 ==> Bad: Message is altered

Process of Getting Certificate

- CA – trusted authority (Certificate Authority)
- Alice
  - Input last name, first name, email, subject attr (x.500)
  - Generate a pri / pub key pair
  - Pri key stays with Alice
  - Pub key can be distributed
  - Package up last/first/email/attr/pub(alice) ←- request is called CSR (Certificate Signing Request)
  - Generate a request with the above package and send to the CA for approval
  - Request -> CA (CA pri, CA pub)
    - Verify that the requester is actually Alice
    - CA will do the approval process - Interview / phone / in person / pay
    - Package (last/first/email/attr/pub(alice))
    - Sign (last/first/email/attr/pub(alice) with CA pri => x.509 public key certificate

TCP Protocol

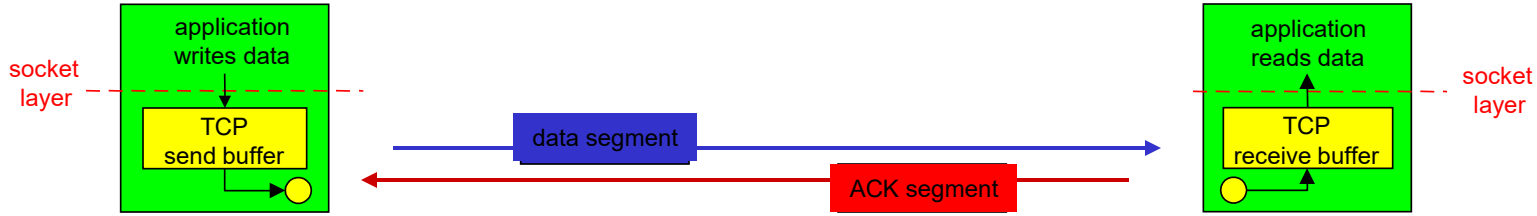
- Slides originally from Williamson at Calgary
- Minor modifications are made

The TCP Protocol

- Connection-oriented, point-to-point protocol:
  - Connection establishment and teardown phases
  - 'Phone-like' circuit abstraction (application-layer view)
  - One sender, one receiver
  - Called a "reliable byte stream" protocol
  - General purpose (for any network environment)

- Originally optimized for certain kinds of transfer:
  - Telnet (interactive remote login)
  - FTP (long, slow transfers)
  - Web is like neither of these!

TCP Protocol (cont)

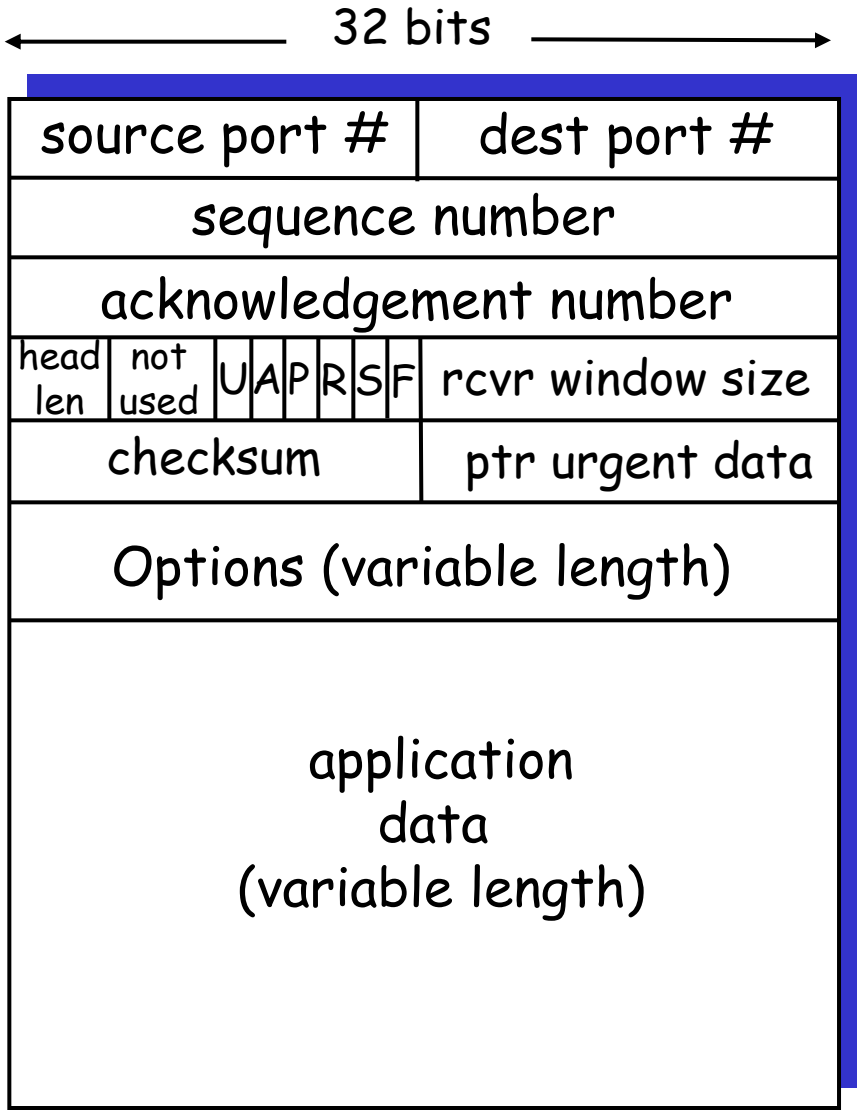


- Provides a reliable, in-order, byte stream abstraction:
  - Recover lost packets and detect/drop duplicates
  - Detect and drop corrupted packets
  - Preserve order in byte stream, no "message boundaries"
  - Full-duplex: bi-directional data flow in same connection
- Flow and congestion control:
  - Flow control: sender will not overwhelm receiver
  - Congestion control: sender will not overwhelm the network
  - Sliding window flow control
  - Send and receive buffers
  - Congestion control done via adaptive flow control window size

The TCP Header

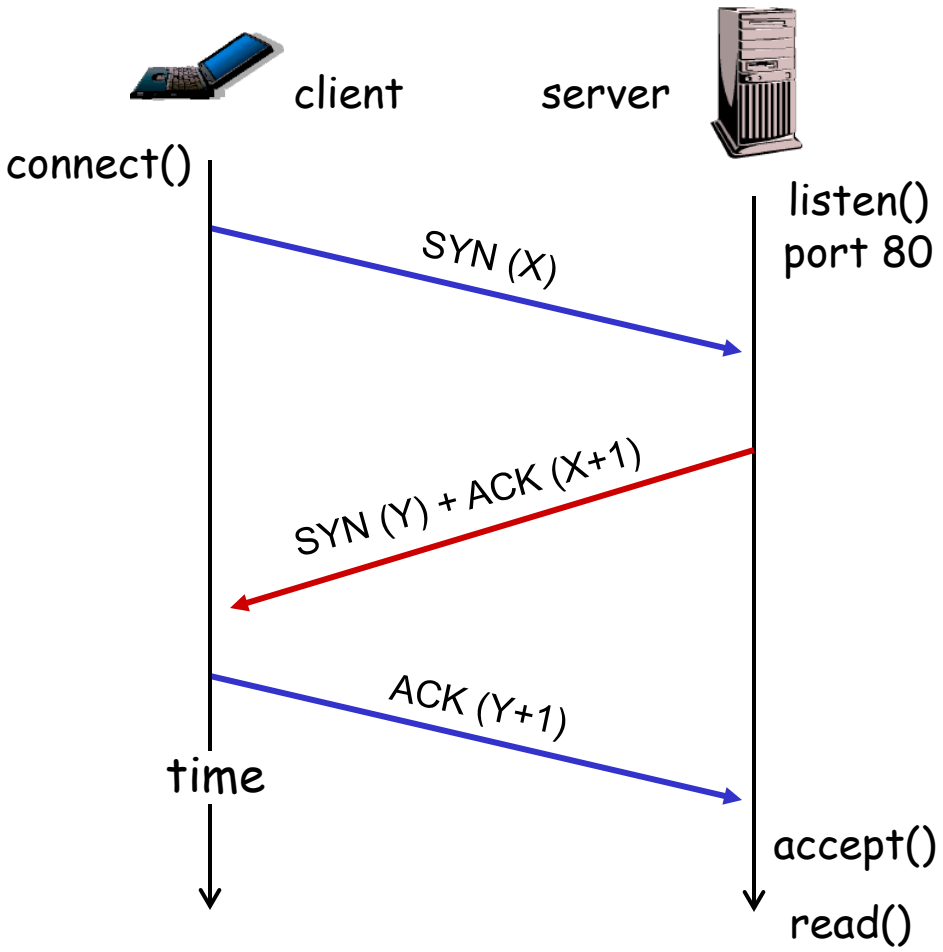
Fields enable the following:

- Uniquely identifying a connection
  (4-tuple of client/server IP address and port numbers)
- Identifying a byte range within that connection
- Checksum value to detect corruption
- Flags to identify protocol state transitions (SYN, FIN, RST)
- Informing other side of your state (ACK)

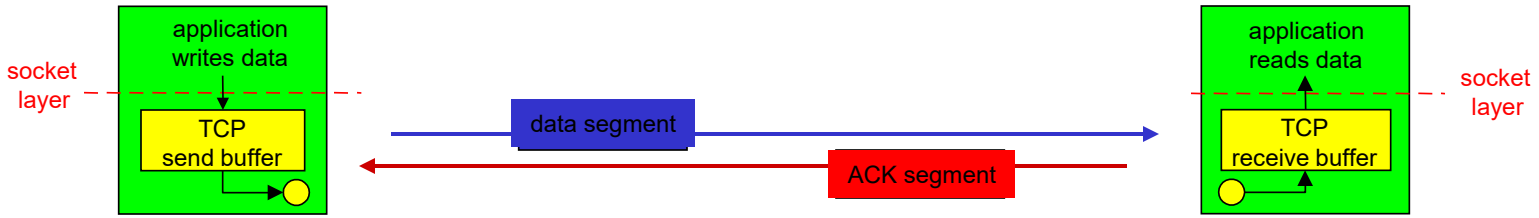


Establishing a TCP Connection

- Client sends SYN with initial sequence number (ISN = X)
- Server responds with its own SYN w/seq number Y and ACK of client ISN with X+1 (next expected byte)
- Client ACKs server's ISN with Y+1
- The '3-way handshake'
- X, Y randomly chosen
- All modulo 32-bit arithmetic



Sending Data



- Sender TCP passes segments to IP to transmit:
  - Keeps a copy in buffer at send side in case of loss
  - Called a "reliable byte stream" protocol
  - Sender must obey receiver advertised window
- Receiver sends acknowledgments (ACKs)
  - ACKs can be piggybacked on data going the other way
  - Protocol allows receiver to ACK every other packet in attempt to reduce ACK traffic (delayed ACKs)
  - Delay should not be more than 500 ms. (typically 200 ms)
  - We'll see how this causes problems later

Preventing Congestion

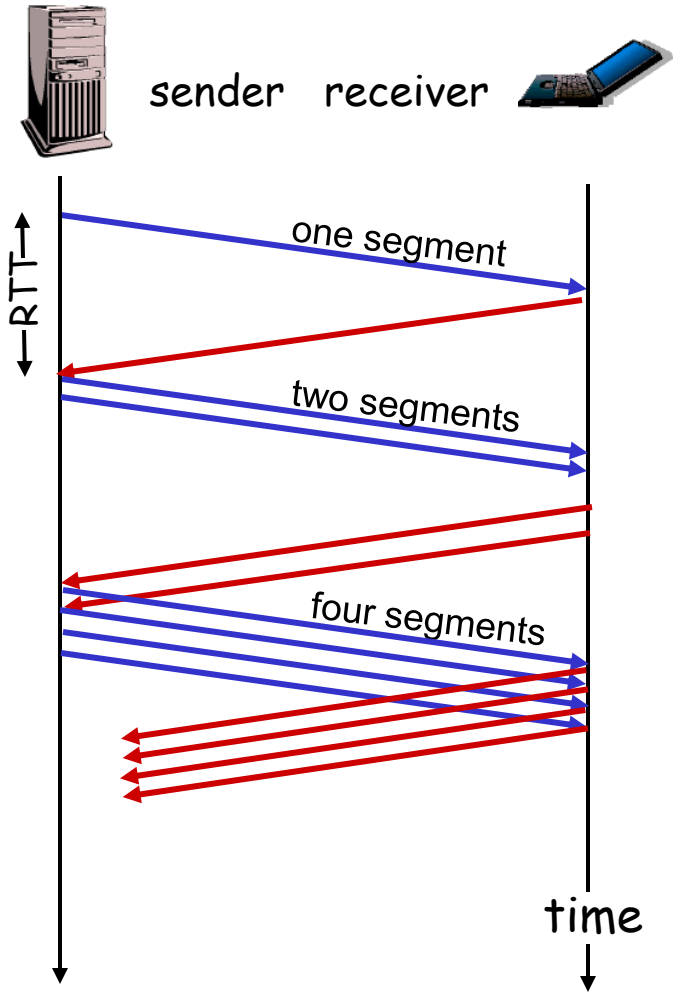
- Sender may not only overrun receiver, but may also overrun intermediate routers:
  - No way to explicitly know router buffer occupancy, so we need to infer it from packet losses
  - Assumption is that losses stem from congestion, namely, that intermediate routers have no available buffers
- Sender maintains a congestion window:
  - Never have more than CW of un-acknowledged data outstanding (or RWIN data; min of the two)
  - Successive ACKs from receiver cause CW to grow.
- How CW grows based on which of 2 phases:
  - Slow-start: initial state.
  - Congestion avoidance: steady-state.
  - Switch between the two when CW > slow-start threshold

Congestion Control Principles

- Lack of congestion control would lead to congestion collapse (Jacobson 88).
- Idea is to be a "good network citizen".
- Would like to transmit as fast as possible without loss.
- Probe network to find available bandwidth.
- In steady-state: linear increase in CW per RTT.
- After loss event: CW is halved.
- This is called additive increase /multiplicative decrease (AIMD).
- Various papers on why AIMD leads to network stability.

Slow Start

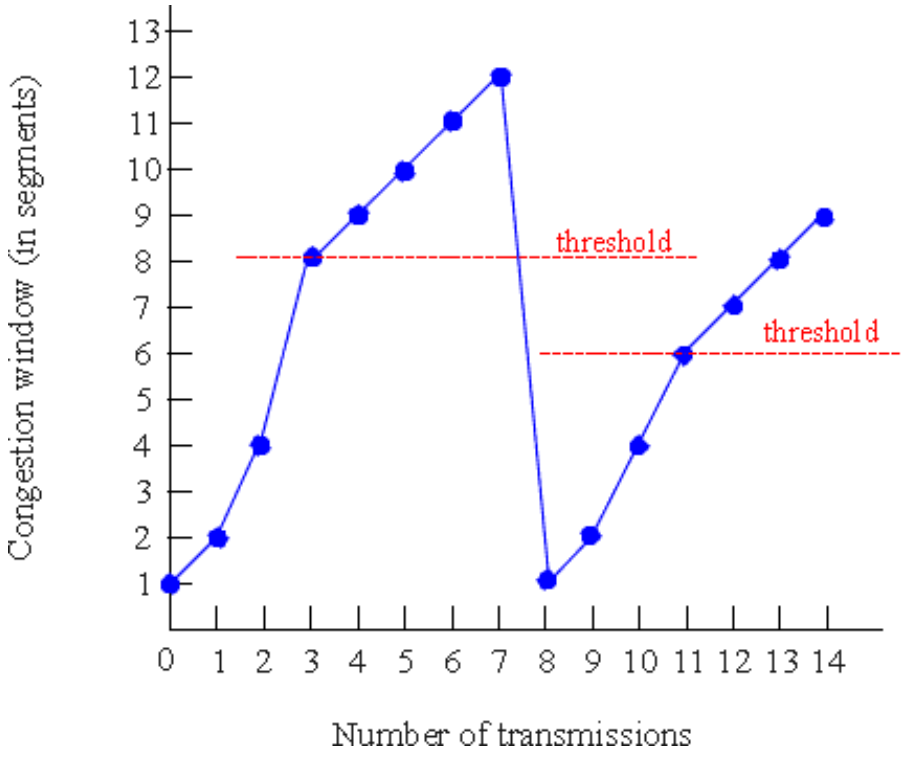
- Initial CW = 1.
- After each ACK, CW += 1;
- Continue until:
  - Loss occurs OR
  - CW > slow start threshold
- Then switch to congestion avoidance
- If we detect loss, cut CW in half
- Exponential increase in window size per RTT



Congestion Avoidance

```
Until (loss) {
 after CW packets ACKed:
   CW += 1;
}
ssthresh = CW/2;
Depending on loss type:
  SACK/Fast Retransmit:
    CW/= 2; continue;
  Course grained timeout:
    CW = 1; go to slow start.
```

(This is for TCP Reno/SACK: TCP Tahoe always sets CW=1 after a loss)



How are losses recovered?

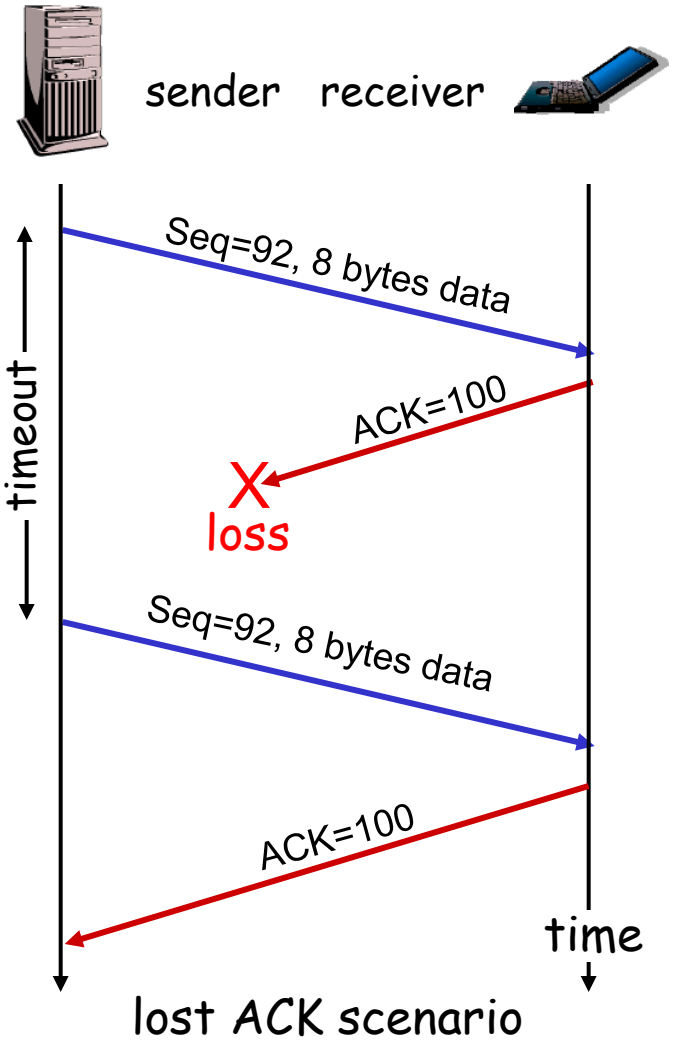
Say packet is lost (data or ACK!)

- Coarse-grained Timeout:
  - Sender does not receive ACK after some period of time
  - Event is called a retransmission time-out (RTO)
  - RTO value is based on estimated round-trip time (RTT)
  - RTT is adjusted over time using exponential weighted moving average:

    RTT = (1-x)*RTT + (x)*sample

    (x is typically 0.1)

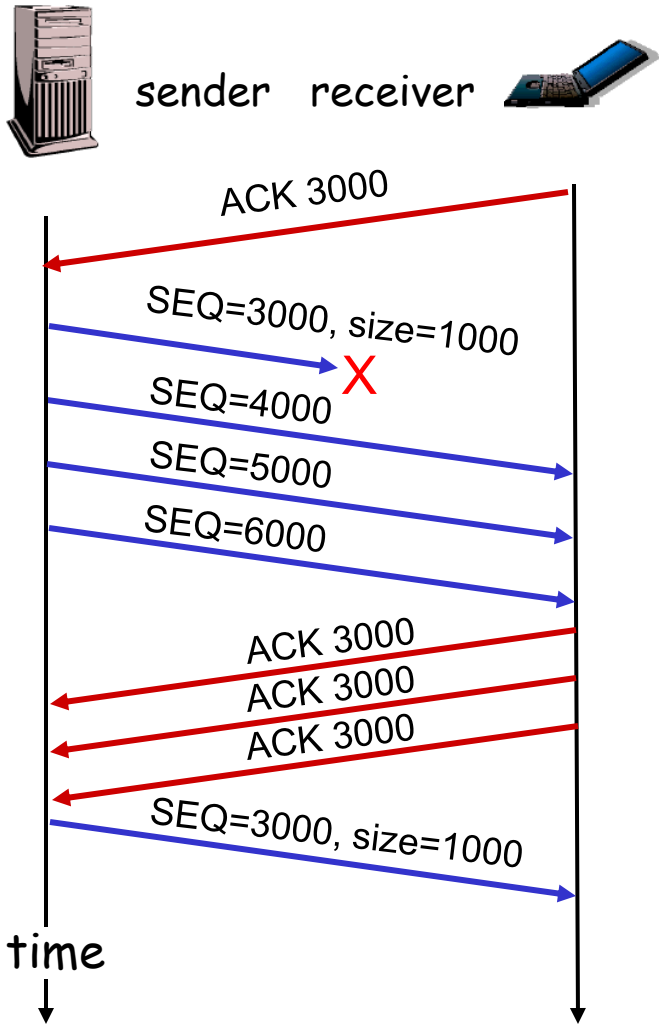
First done in TCP Tahoe



Fast Retransmit

- Receiver expects N, gets N+1:
  - Immediately sends ACK(N)
  - This is called a duplicate ACK
  - Does NOT delay ACKs here!
  - Continue sending dup ACKs for each subsequent packet (not N)
- Sender gets 3 duplicate ACKs:
  - Infers N is lost and resends
  - 3 chosen so out-of-order packets don't trigger Fast Retransmit accidentally
  - Called "fast" since we don't need to wait for a full RTT

Introduced in TCP Reno

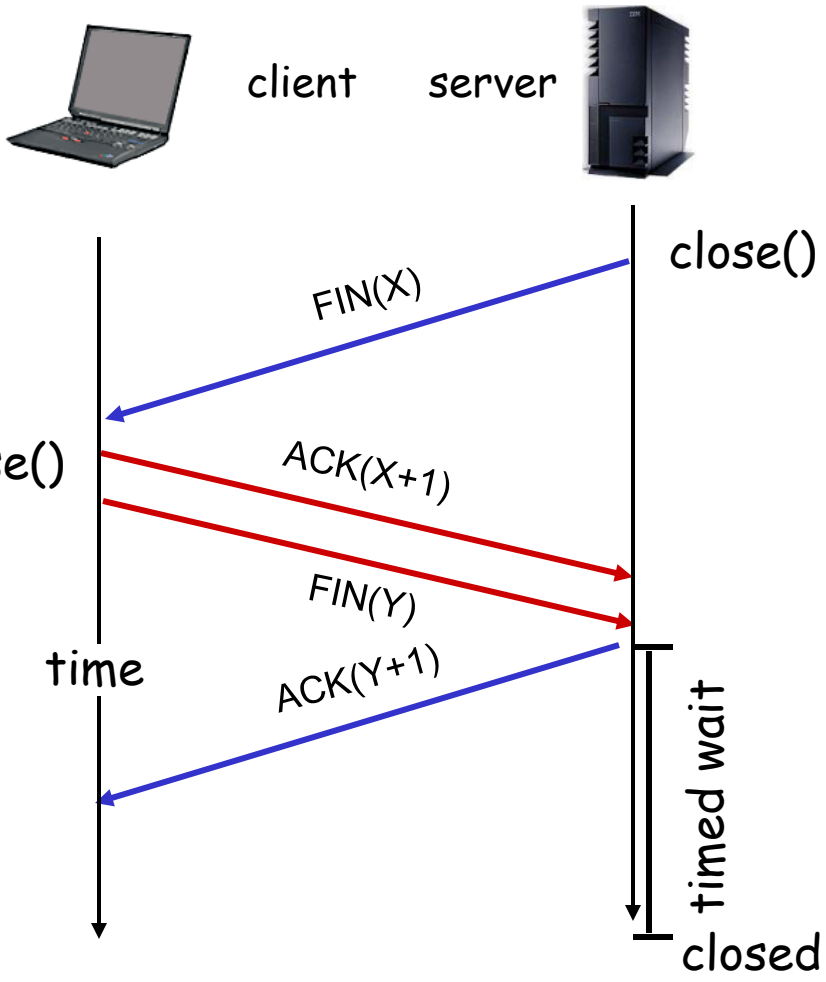


Other loss recovery methods

- Selective Acknowledgements (SACK):
  - Returned ACKs contain option w/SACK block
  - Block says, "got up N-1 AND got N+1 through N+3"
  - A single ACK can generate a retransmission
- New Reno partial ACKs:
  - New ACK during fast retransmit may not ACK all outstanding data. Ex:
    - Have ACK of 1, waiting for 2-6, get 3 dup acks of 1
    - Retransmit 2, get ACK of 3, can now infer 4 lost as well
- Other schemes exist (e.g., Vegas)
- Reno has been prevalent; SACK now catching on

How about Connection Teardown?

- Either side may terminate a connection. (In fact, connection can stay half-closed.) Let's say the server closes (typical in WWW)
- Server sends FIN with seq Number (SN+1) (i.e., FIN is a byte in sequence)
- Client ACK's the FIN with SN+2 ("next expected")
- Client sends it's own FIN when ready
- Server ACK's client FIN as well with SN+1.

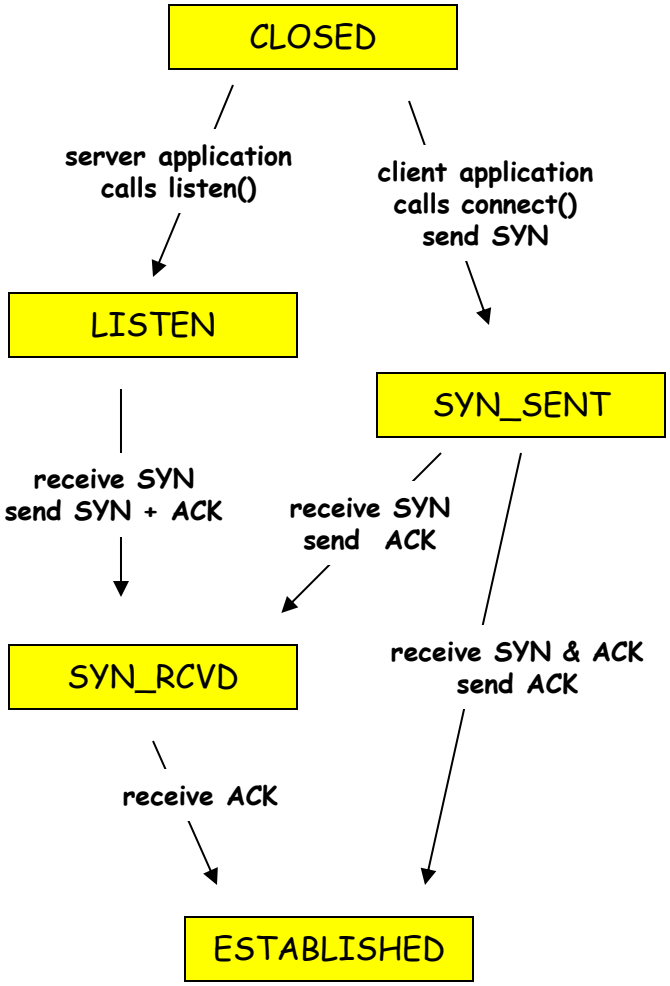


The TCP State Machine

- TCP uses a Finite State Machine, kept by each side of a connection, to keep track of what state a connection is in.
- State transitions reflect inherent races that can happen in the network, e.g., two FIN's passing each other in the network.
- Certain things can go wrong along the way, i.e., packets can be dropped or corrupted. In fact, machine is not perfect; certain problems can arise not anticipated in the original RFC.
- This is where timers will come in, which we will discuss more later.

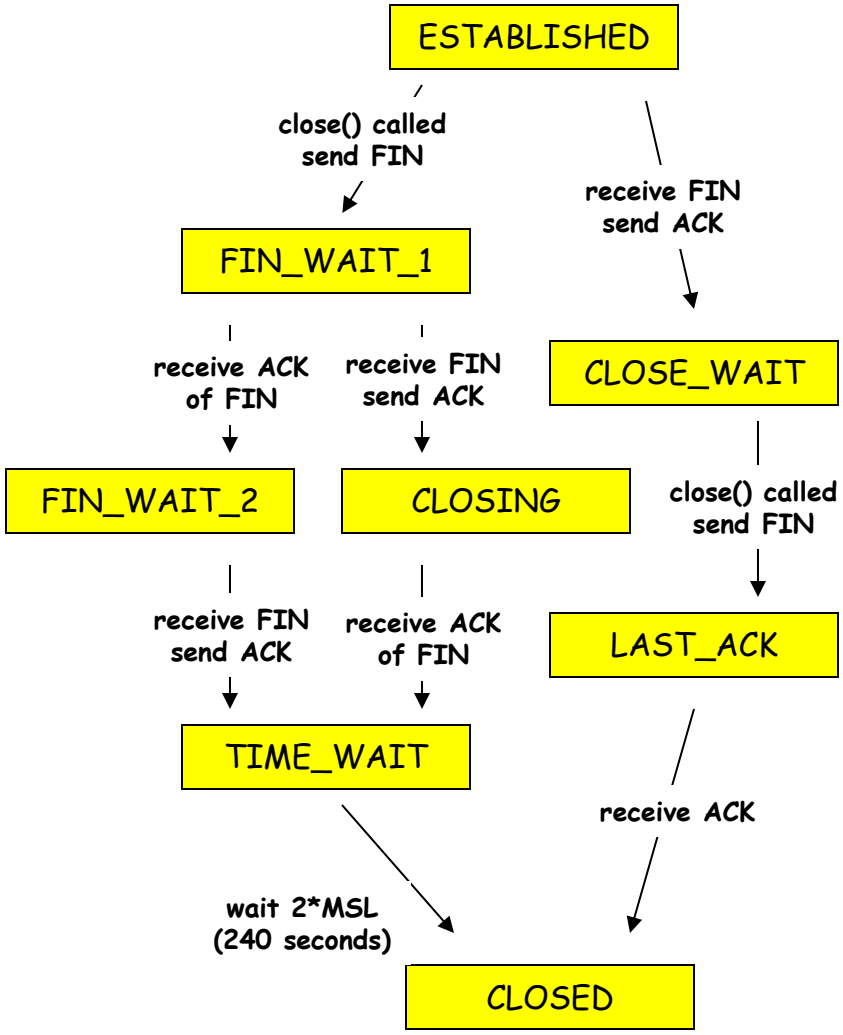
TCP State Machine: Connection Establishment

- CLOSED: more implied than actual, i.e., no connection
- LISTEN: willing to receive connections (accept call)
- SYN-SENT: sent a SYN, waiting for SYN-ACK
- SYN-RECEIVED: received a SYN, waiting for an ACK of our SYN
- ESTABLISHED: connection ready for data transfer



TCP State Machine: Connection Teardown

- FIN-WAIT-1: we closed first, waiting for ACK of our FIN (active close)
- FIN-WAIT-2: we closed first, other side has ACKED our FIN, but not yet FIN'ed
- CLOSING: other side closed before it received our FIN
- TIME-WAIT: we closed, other side closed, got ACK of our FIN
- CLOSE-WAIT: other side sent FIN first, not us (passive close)
- LAST-ACK: other side sent FIN, then we did, now waiting for ACK



Summary: TCP Protocol

- Protocol provides reliability in face of complex network behavior
- Tries to trade off efficiency with being "good network citizen"
- Vast majority of bytes transferred on Internet today are TCP-based:
  - Web
  - Mail
  - News
  - Peer-to-peer (Napster, Gnutella, FreeNet, KaZaa)

TCP Plots

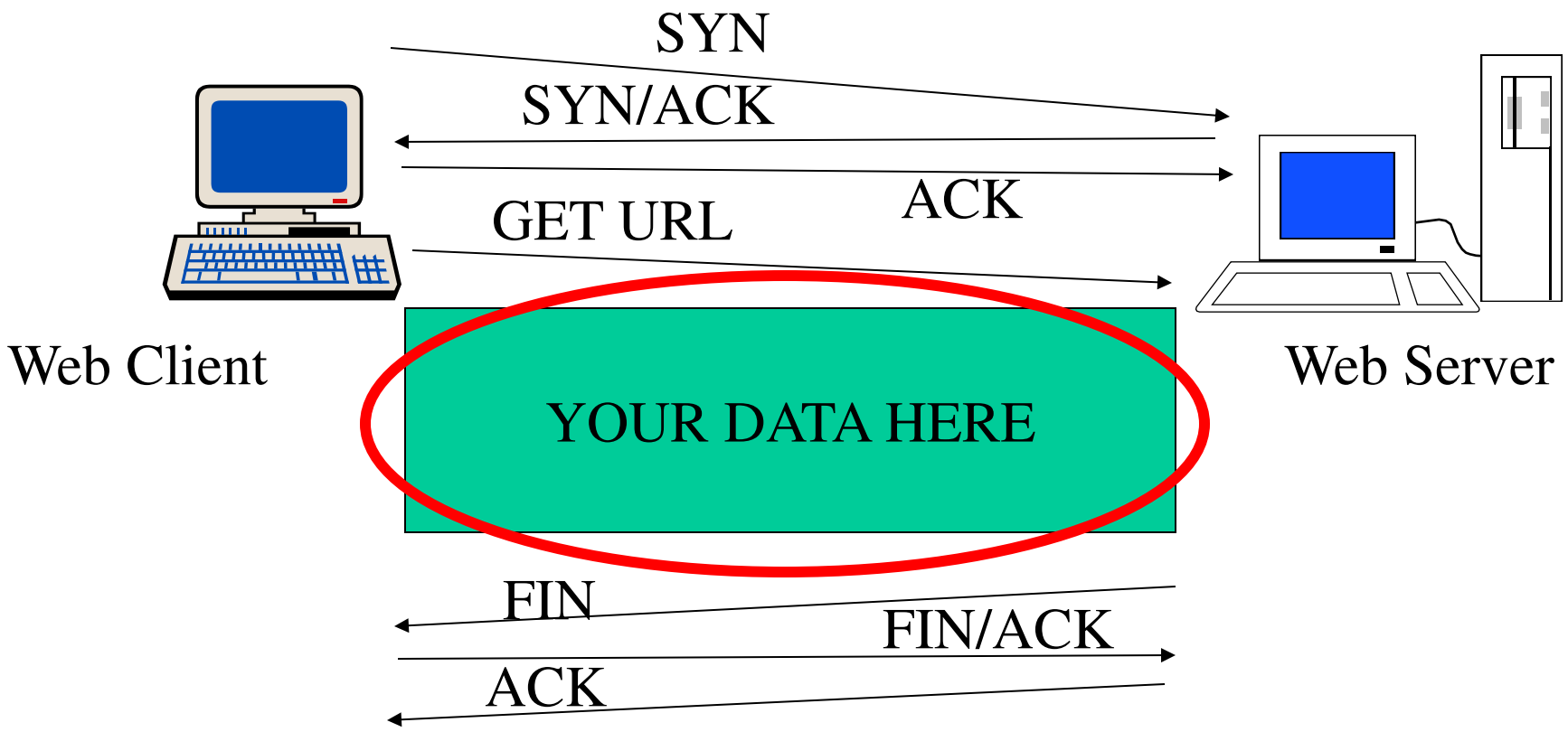
- Slides originally from Williamson at Calgary
- Minor modifications are made

Tutorial: TCP 101

- The Transmission Control Protocol (TCP) is the protocol that sends your data reliably
- Used for email, Web, ftp, telnet, p2p,…
- Makes sure that data is received correctly: right data, right order, exactly once
- Detects and recovers from any problems that occur at the IP network layer
- Mechanisms for reliable data transfer: sequence numbers, acknowledgements, timers, retransmissions, flow control…

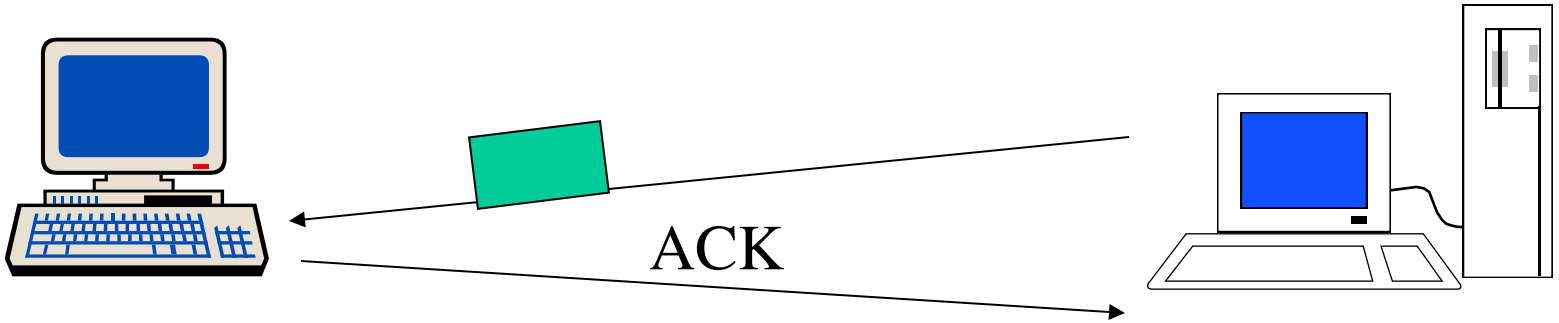
TCP 101 (Cont'd)

- TCP is a connection-oriented protocol



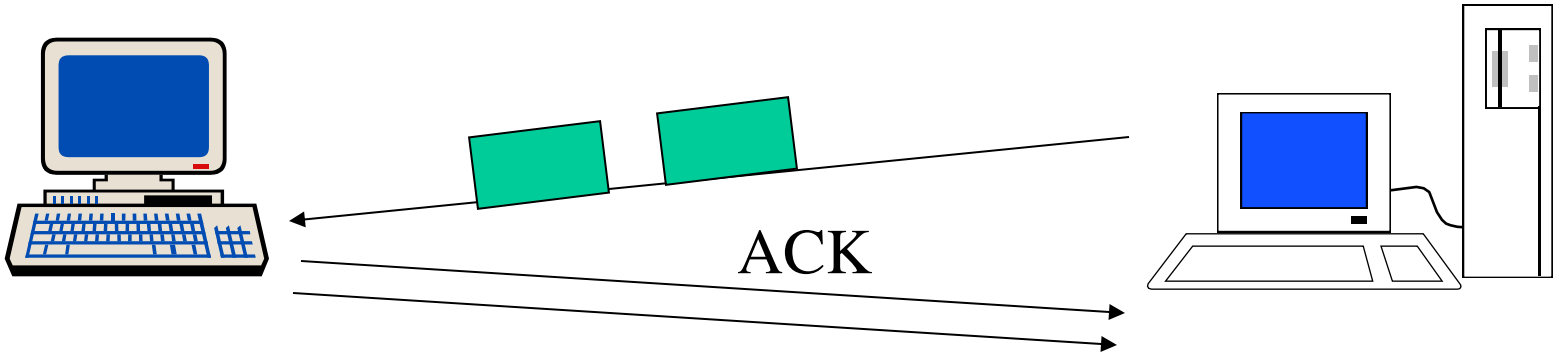
TCP 101 (Cont'd)

- TCP slow-start and congestion avoidance



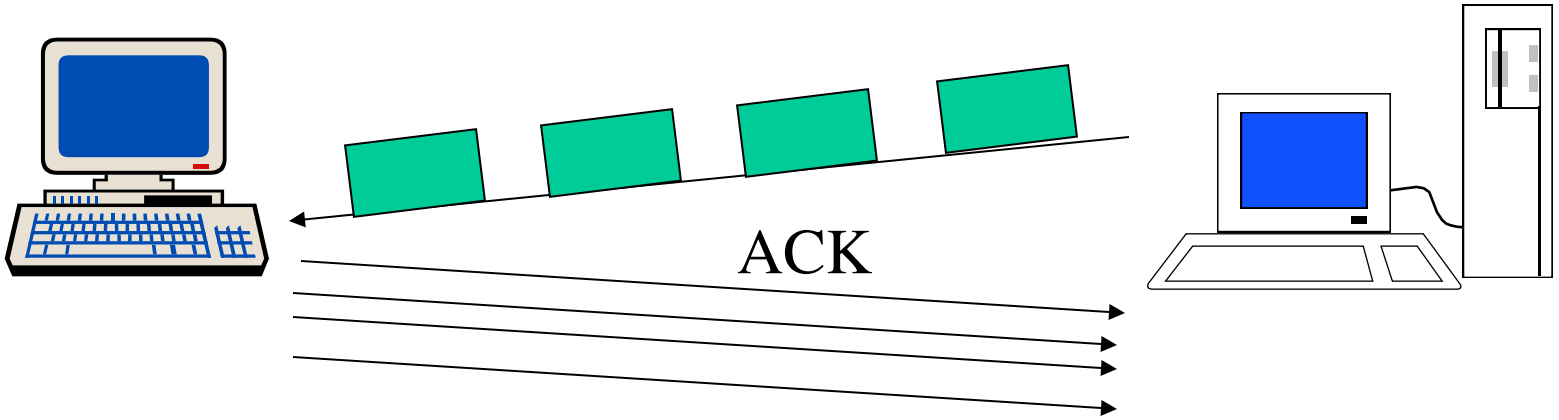
TCP 101 (Cont'd)

- TCP slow-start and congestion avoidance



TCP 101 (Cont'd)

- TCP slow-start and congestion avoidance

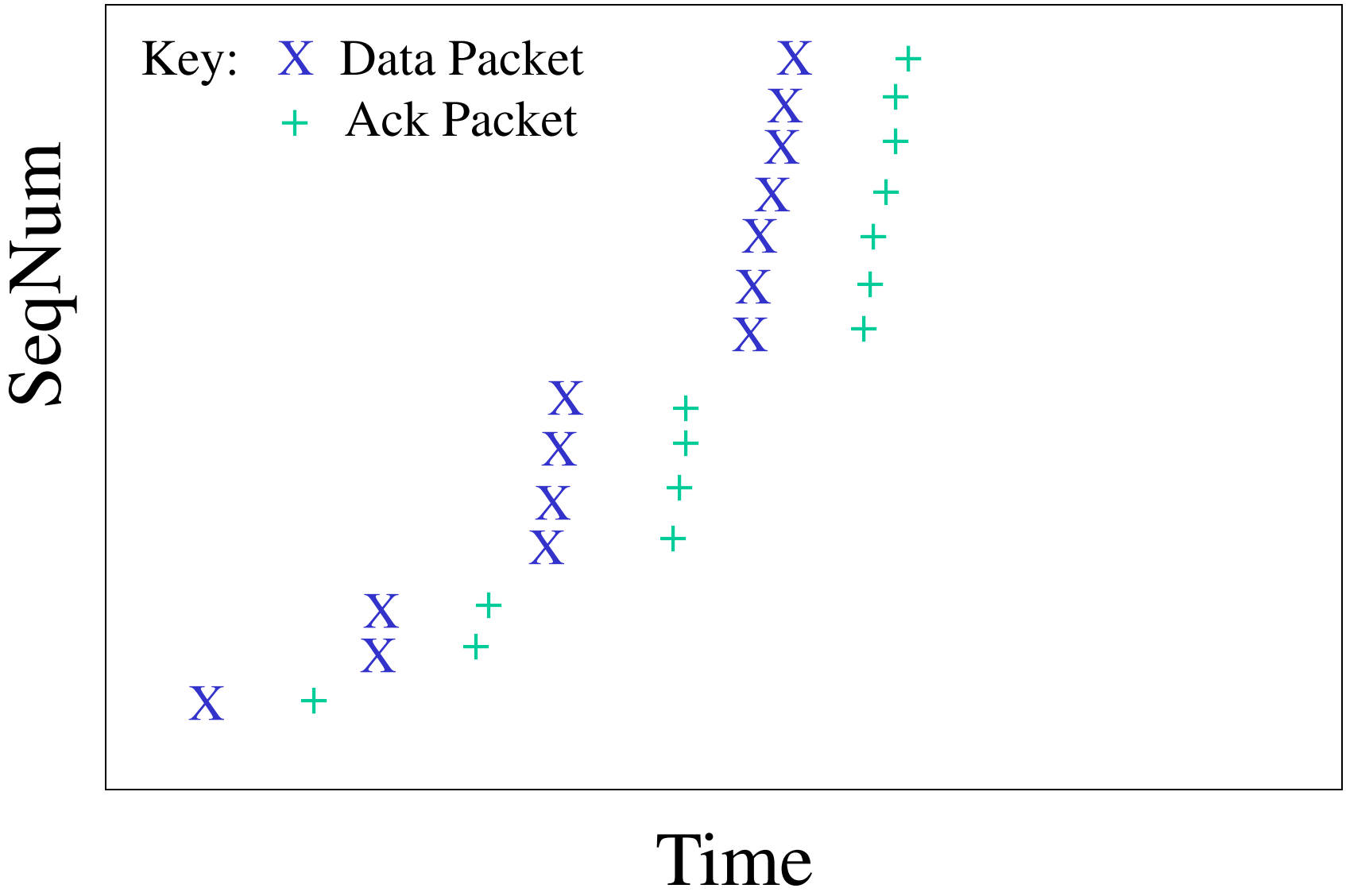


TCP 101 (Cont'd)

- This (exponential growth) "slow start" process continues until either:
  - packet loss: after a brief recovery phase, you enter a (linear growth) "congestion avoidance" phase based on slow-start threshold found
  - limit reached: slow-start threshold, or maximum advertised receive window size
  - all done: terminate connection and go home

Tutorial: TCP 201

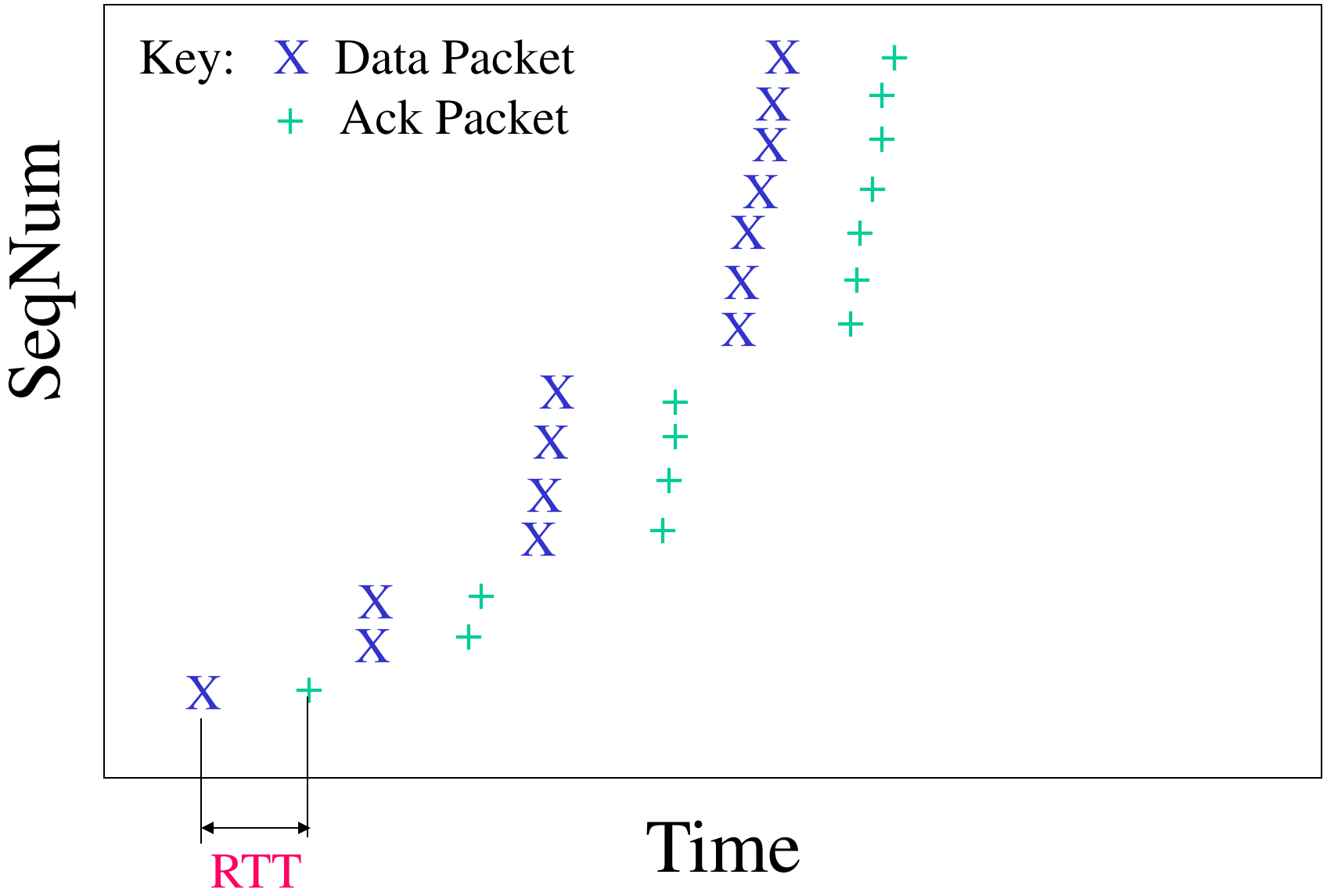
- There is a beautiful way to plot and visualize the dynamics of TCP behaviour
- Called a "TCP Sequence Number Plot"
- Plot packet events (data and acks) as points in 2-D space, with time on the horizontal axis, and sequence number on the vertical axis
- Example: Consider a 14-packet transfer

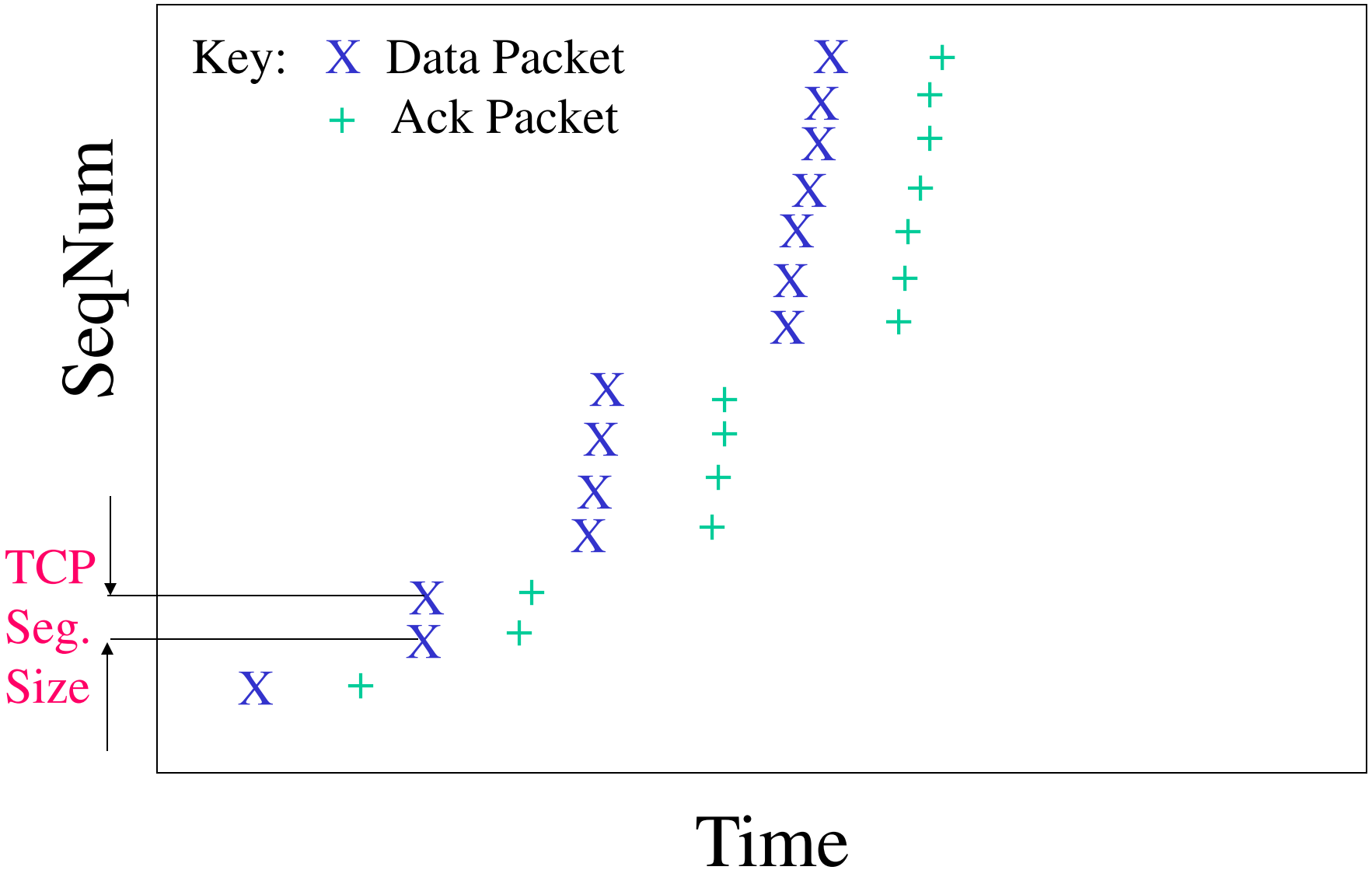

Key: X Data Packet
+ Ack Packet
SeqNum
Time

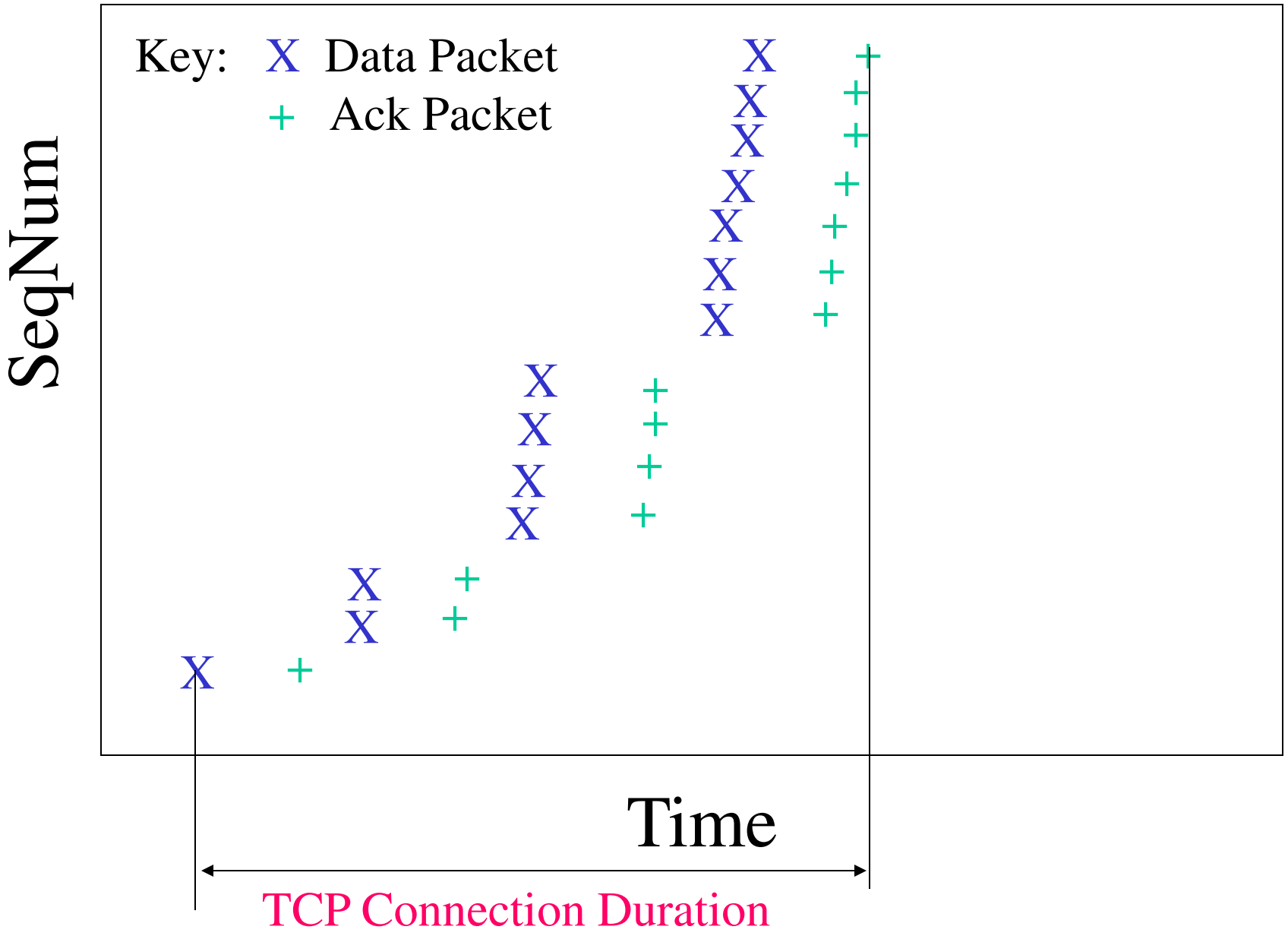
So What?

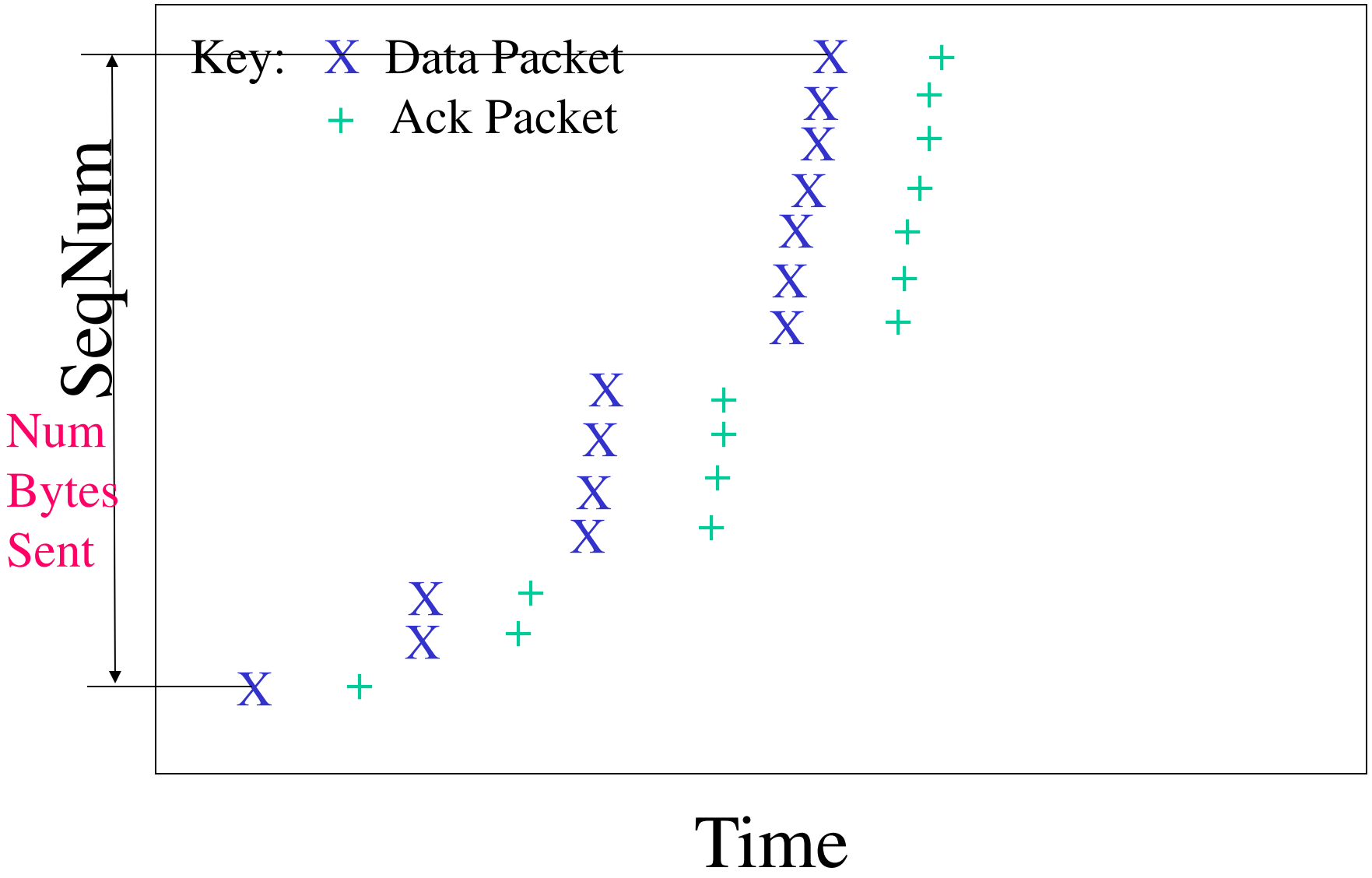
- What can it tell you?

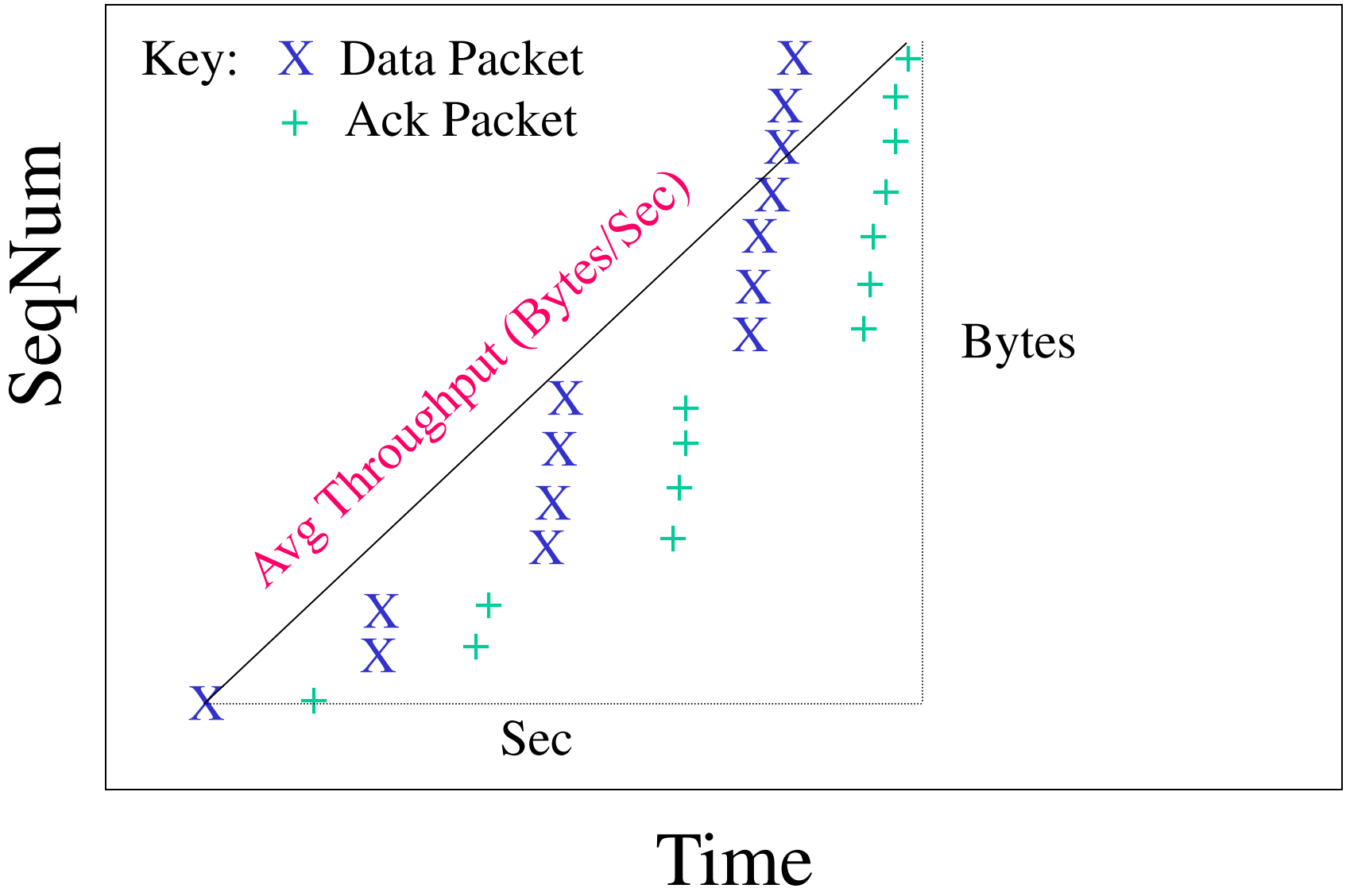
- Everything!!!

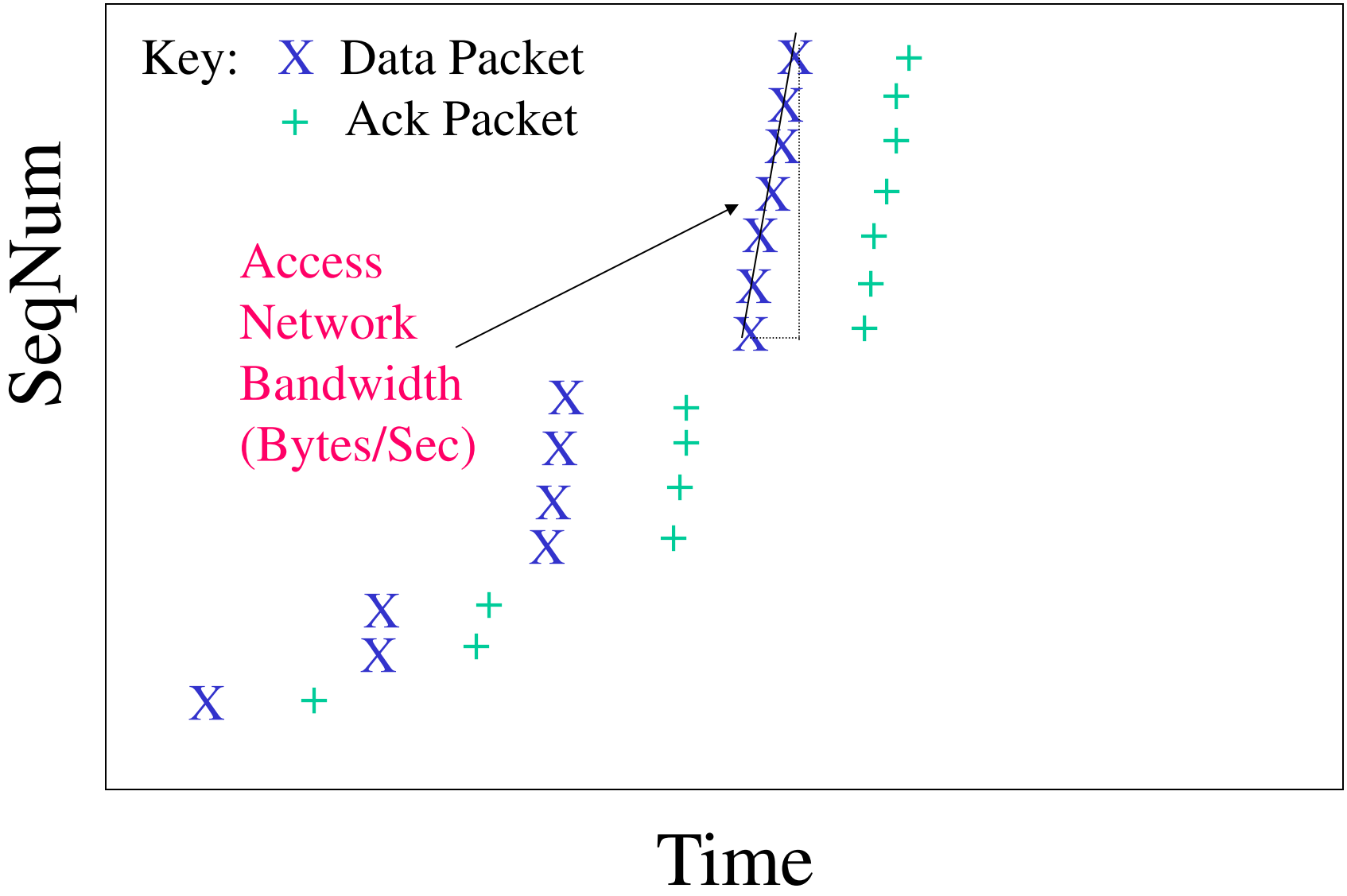

Key: X Data Packet
+ Ack Packet
SeqNum
Time
RTT

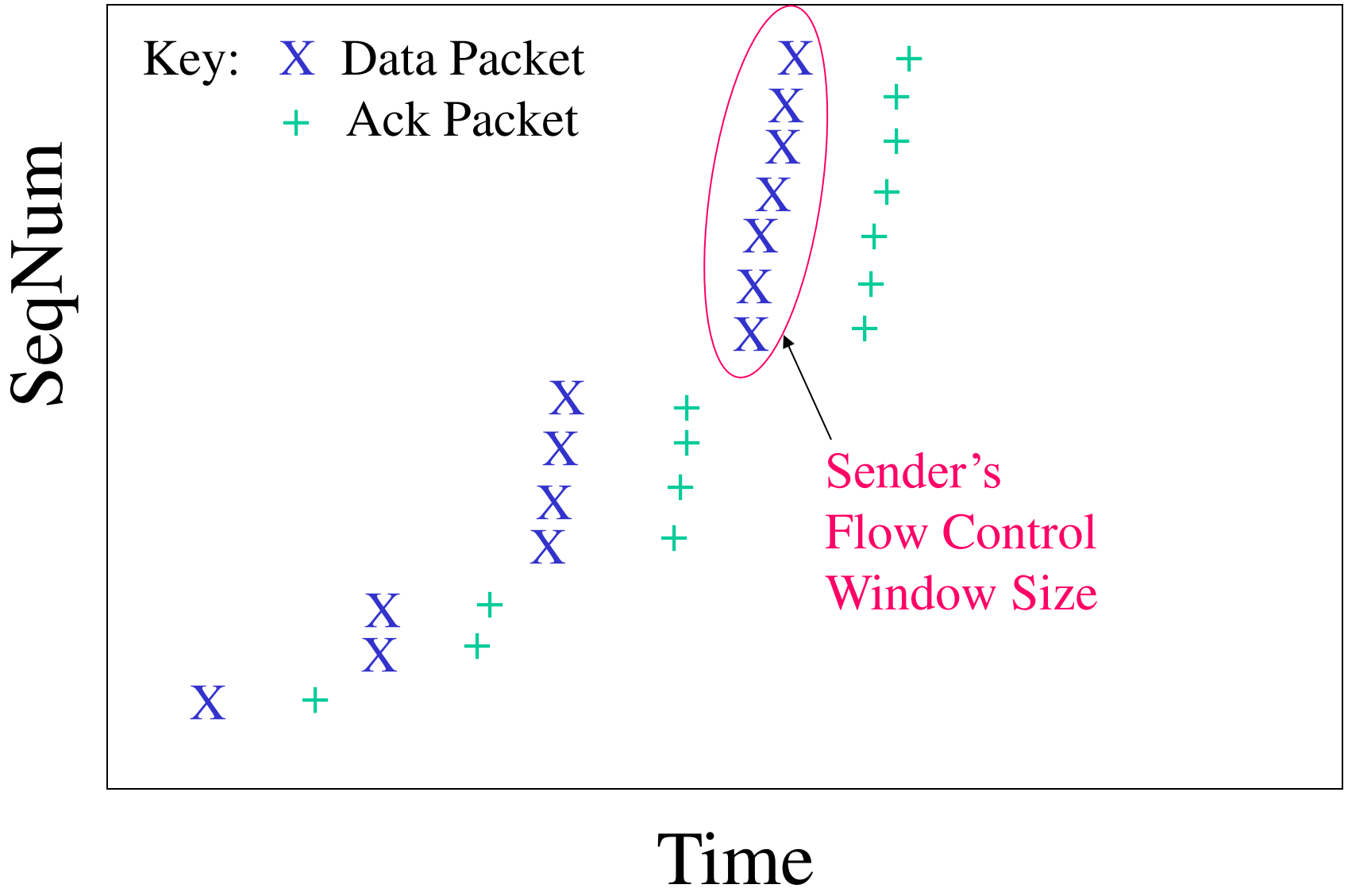

Key: X Data Packet
+ Ack Packet
SeqNum
TCP
Seg.
Size
Time

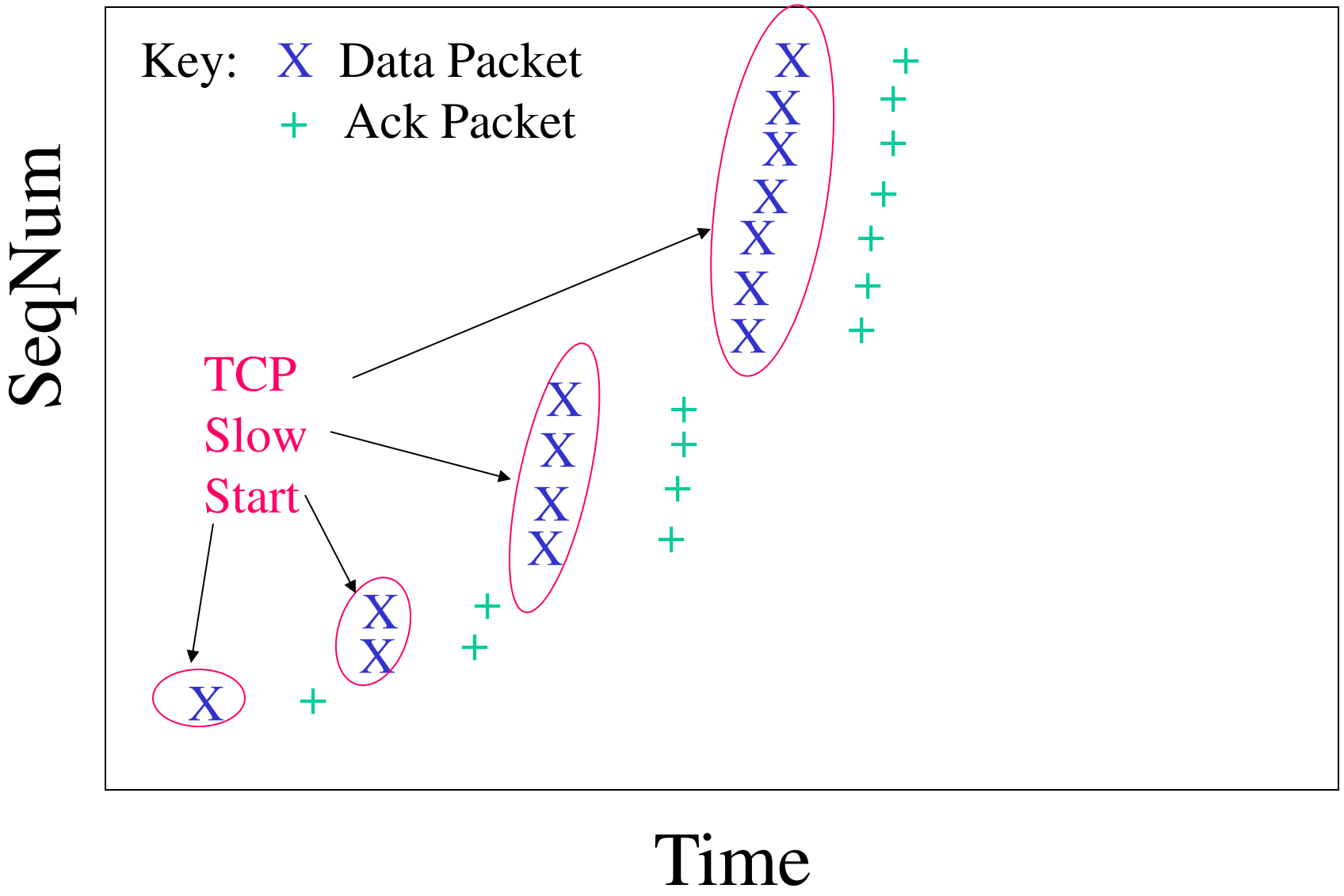

Key: X Data Packet
+ Ack Packet
SeqNum
Time
TCP Connection Duration

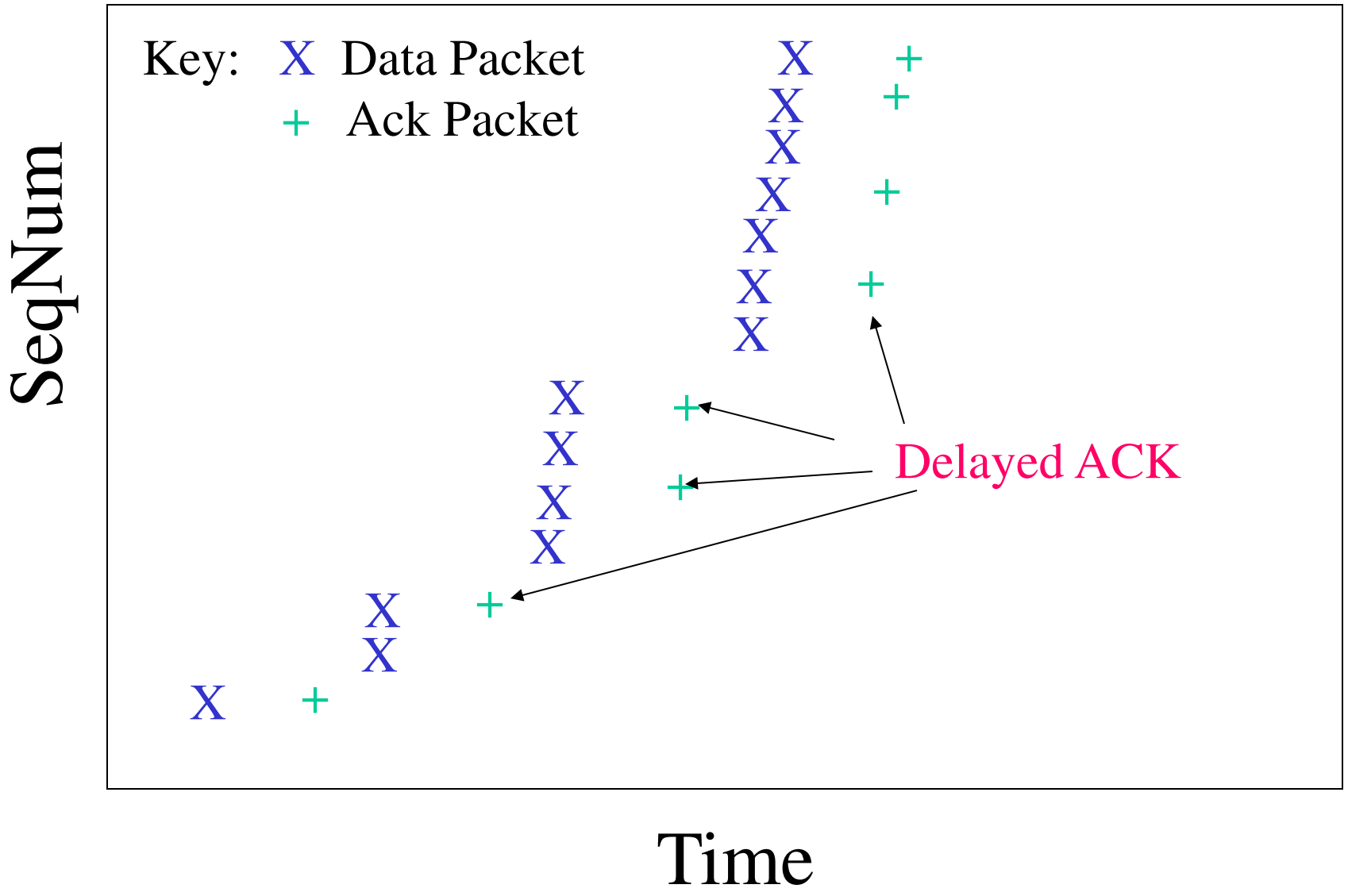

Key: X Data Packet
+ Ack Packet
SeqNum
Num
Bytes
Sent
Time

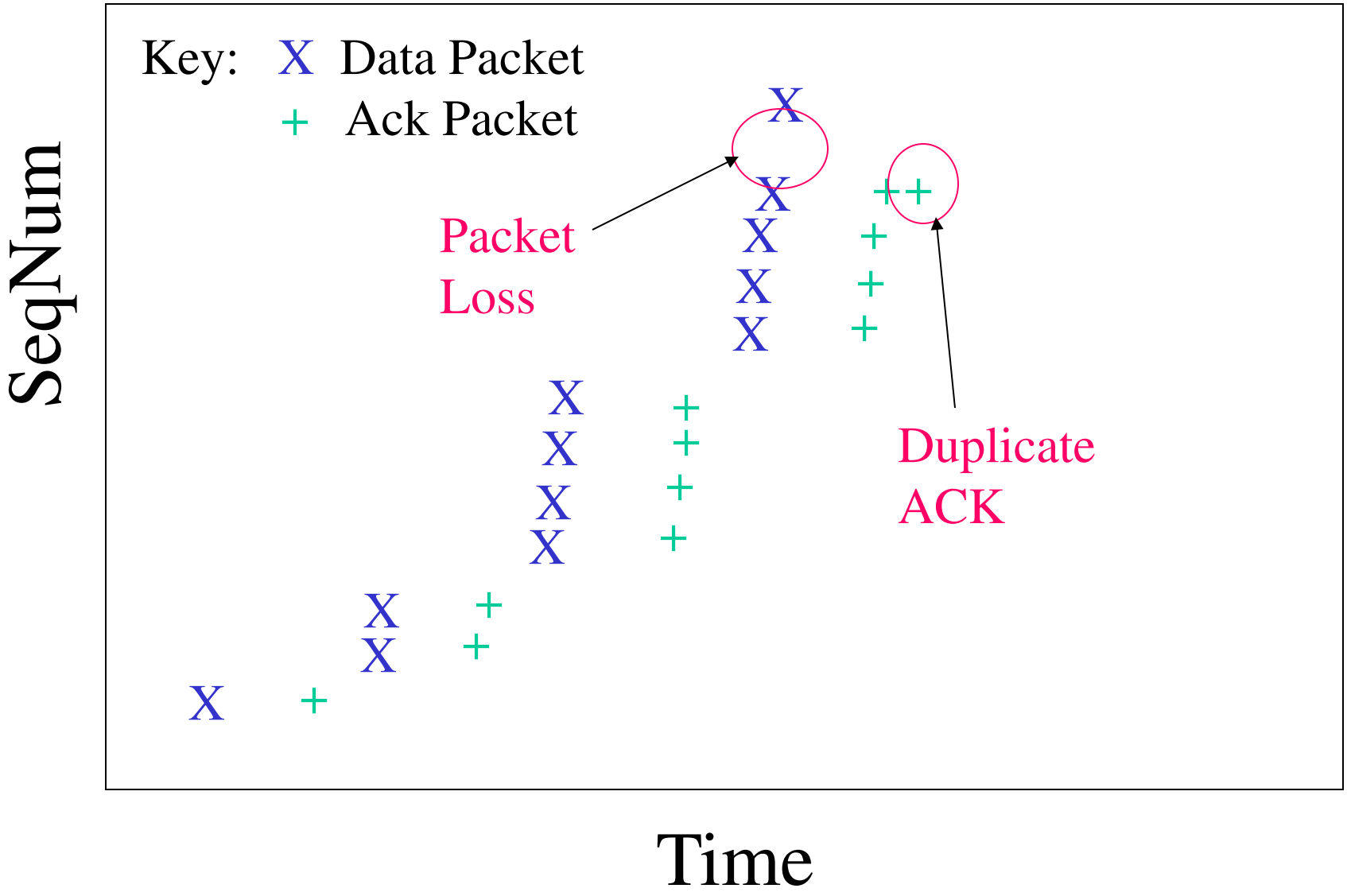

Key: X Data Packet
+ Ack Packet
Avg Throughput (Bytes/Sec)
SeqNum
Bytes
Sec
Time

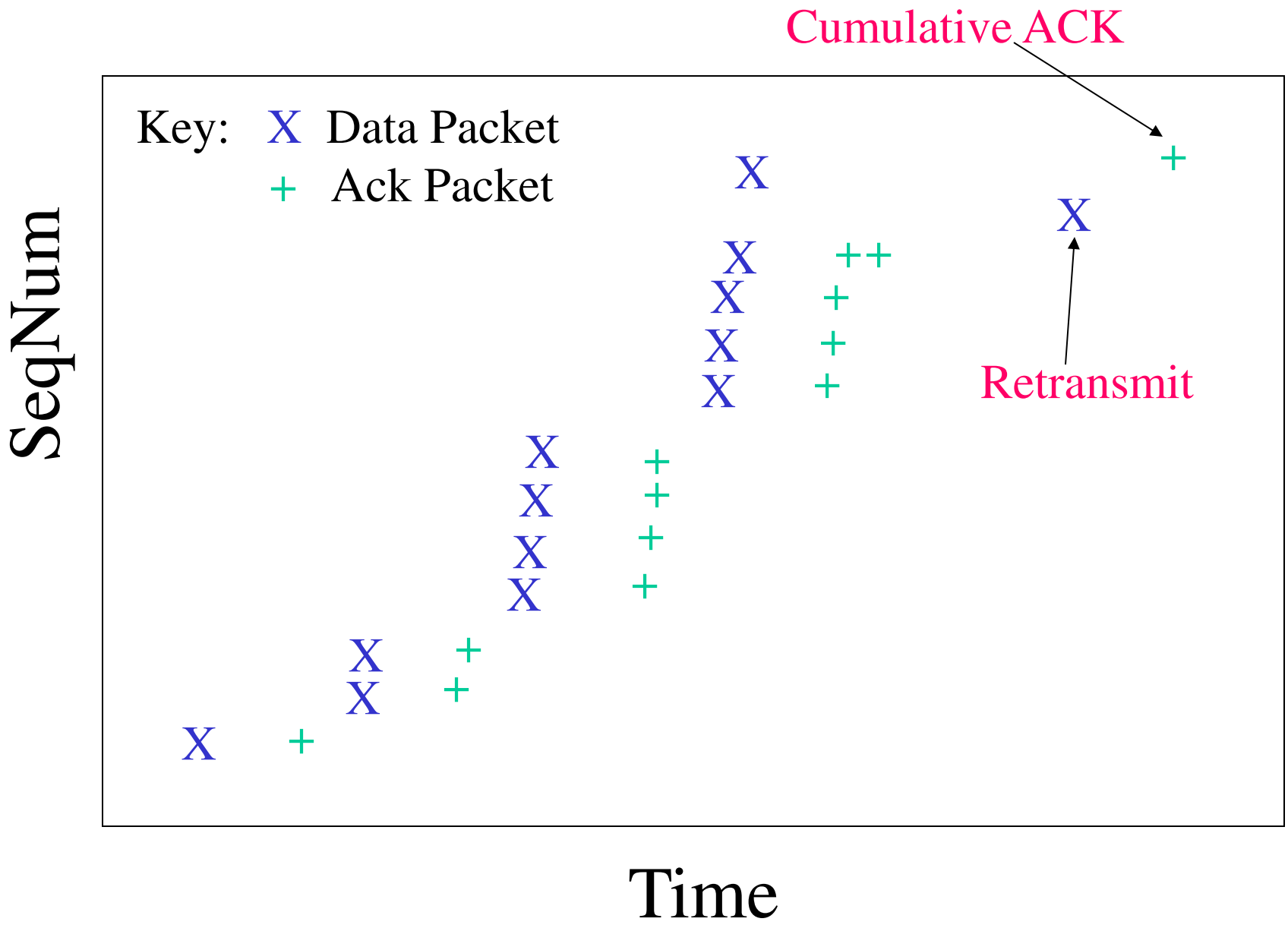

Key: X Data Packet
+ Ack Packet
Access
Network
Bandwidth
(Bytes/Sec)
SeqNum
Time

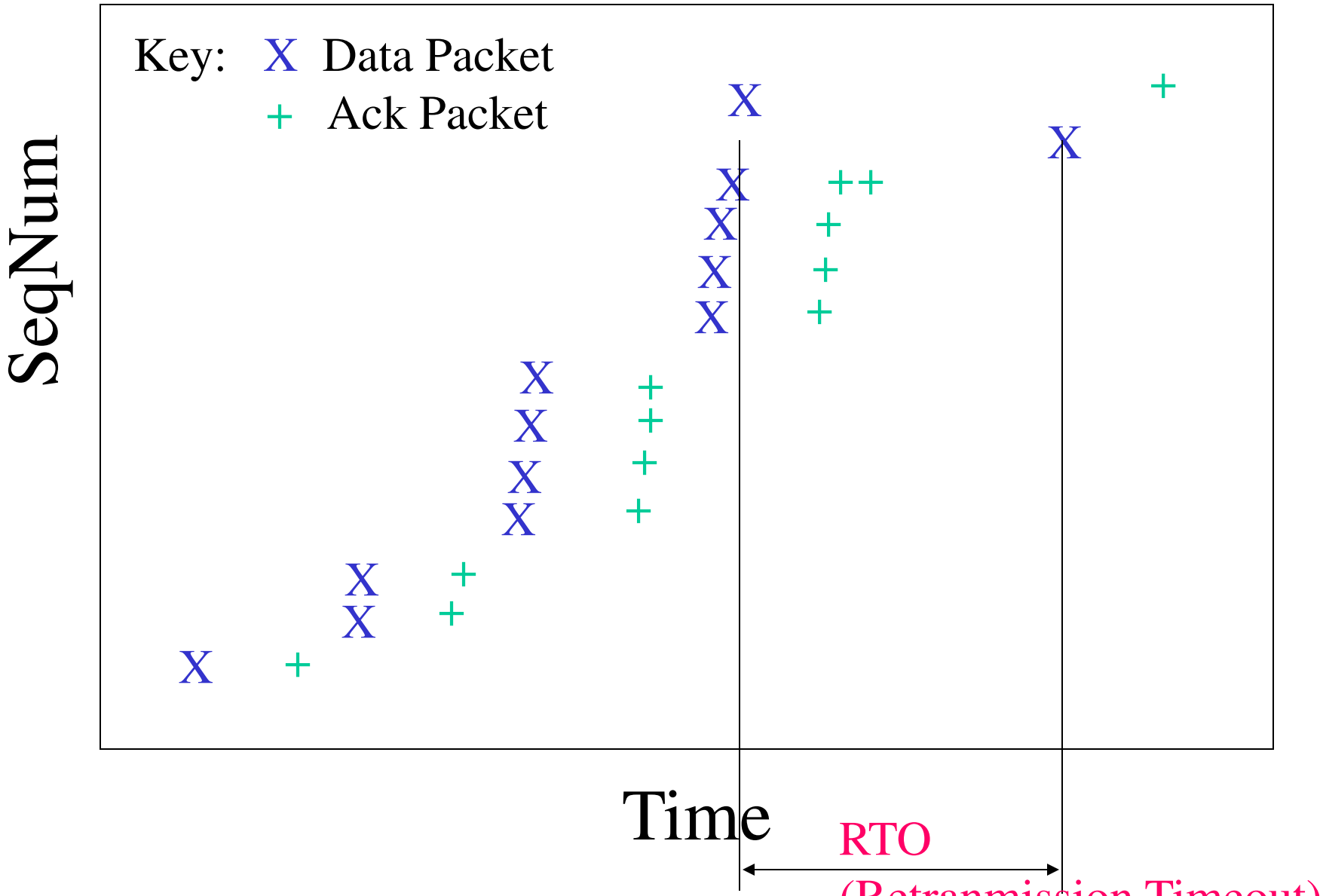

Key: X Data Packet
+ Ack Packet
SeqNum
Time
Sender's
Flow Control
Window Size


SeqNum
Key: X Data Packet
+ Ack Packet
TCP
Slow
Start
Time


Key: X Data Packet
+ Ack Packet
SeqNum
Time
Delayed ACK


Key: X Data Packet
+ Ack Packet
Packet Loss
Duplicate ACK
SeqNum
Time

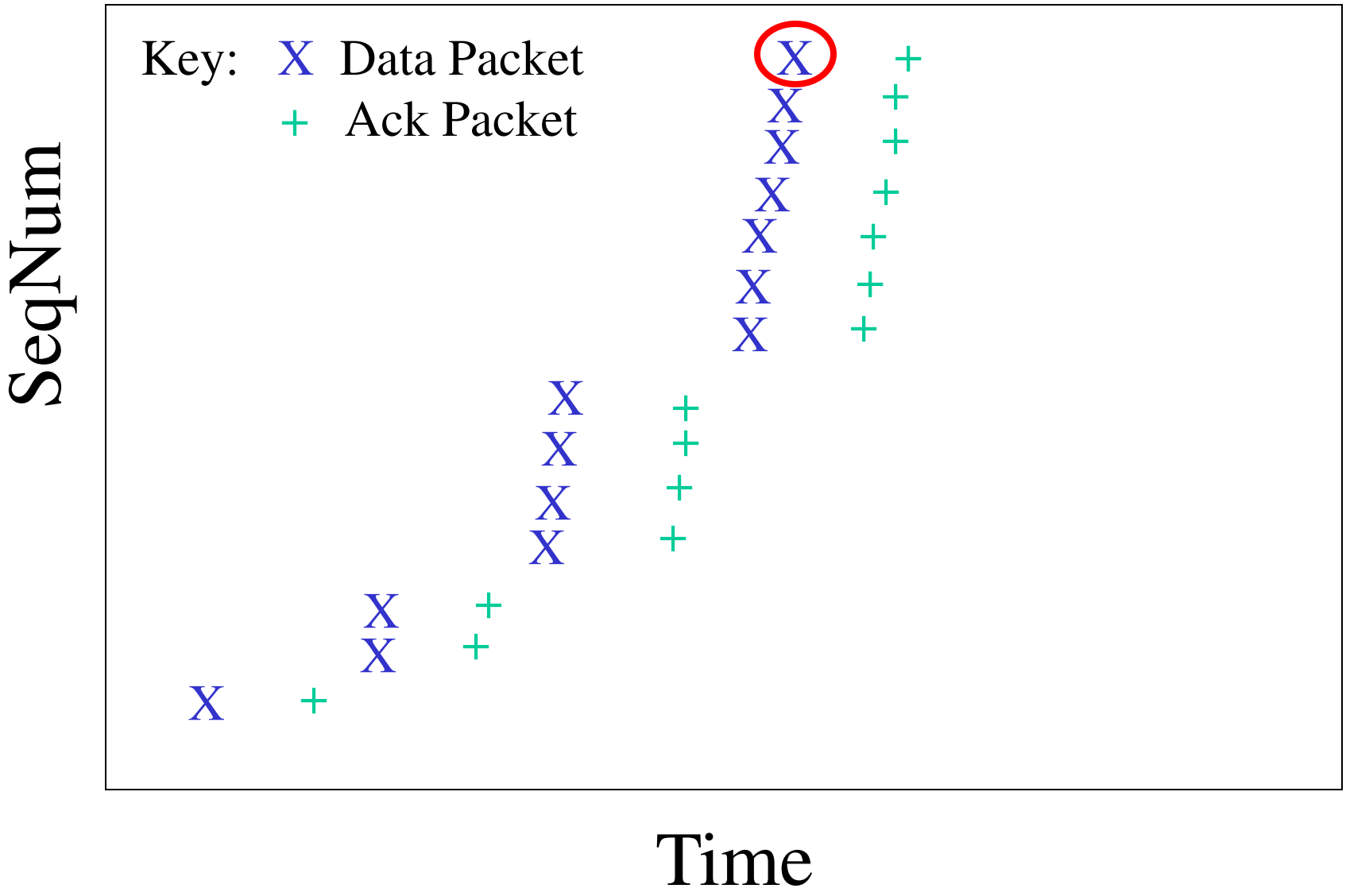

Cumulative ACK
Key: X Data Packet
+ Ack Packet
SeqNum
Retransmit
Time

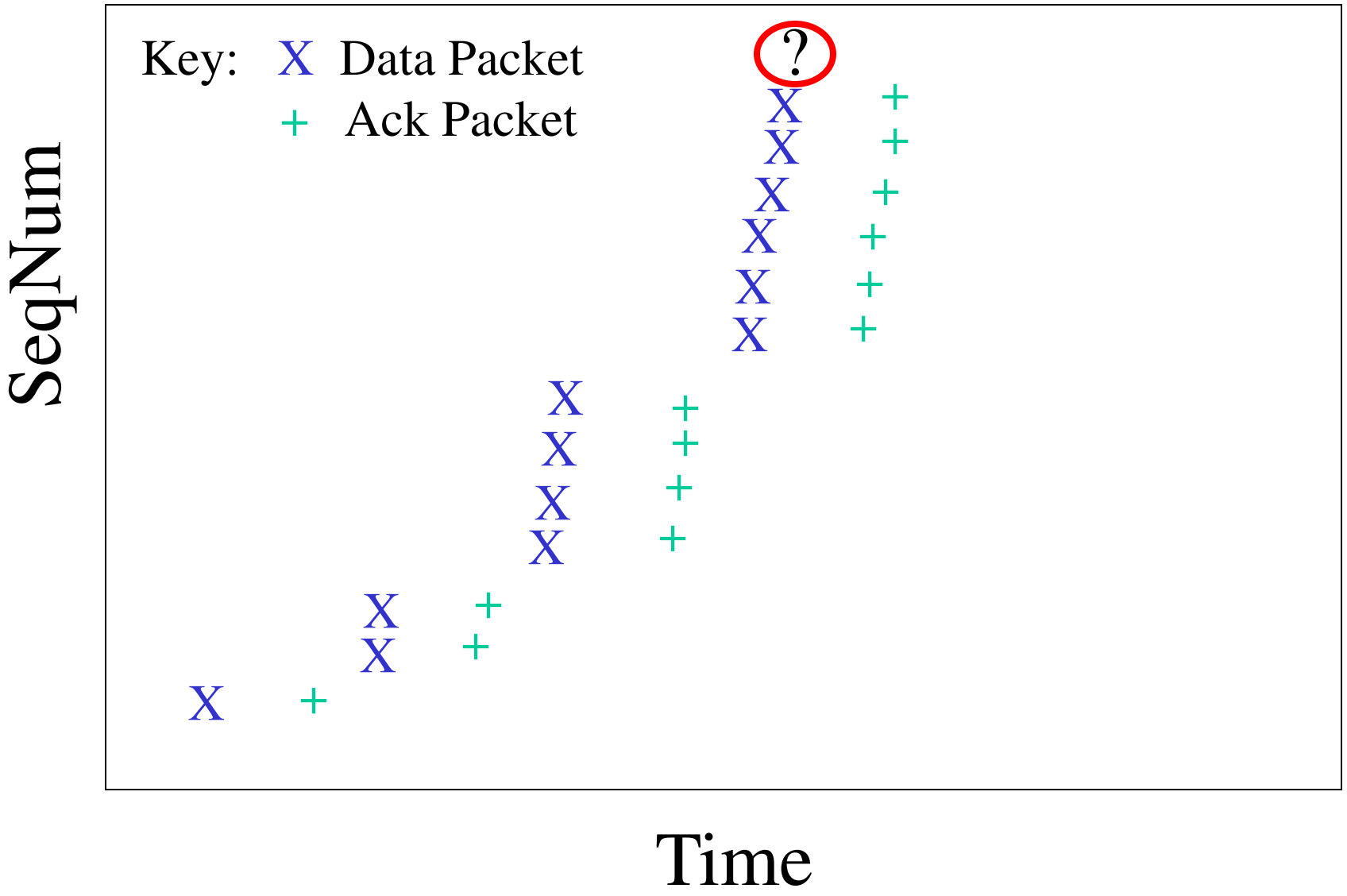

Key: X Data Packet
+ Ack Packet
SeqNum
Time
RTO
(Retranmission Timeout)

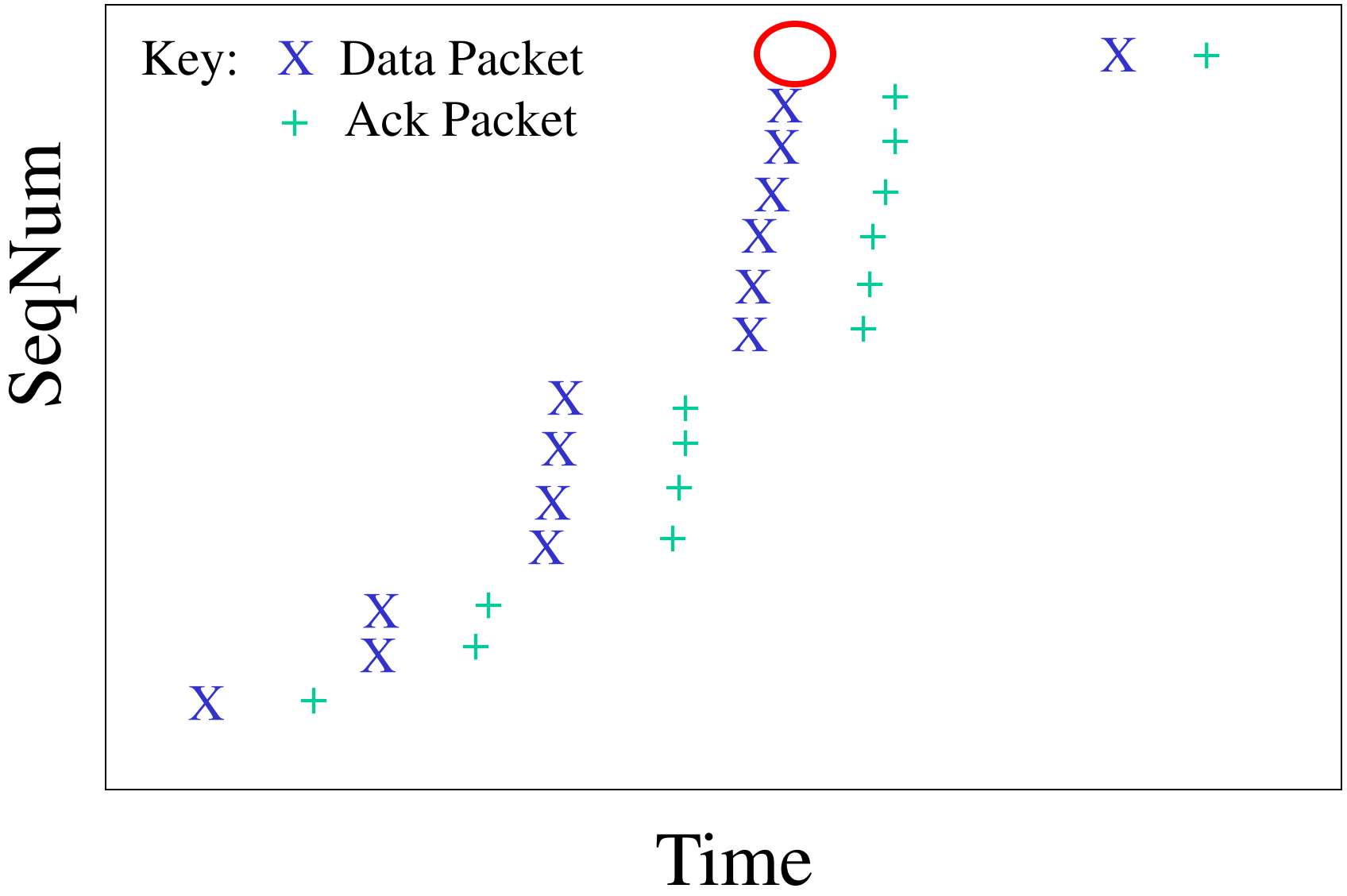
TCP 201 (Cont'd)

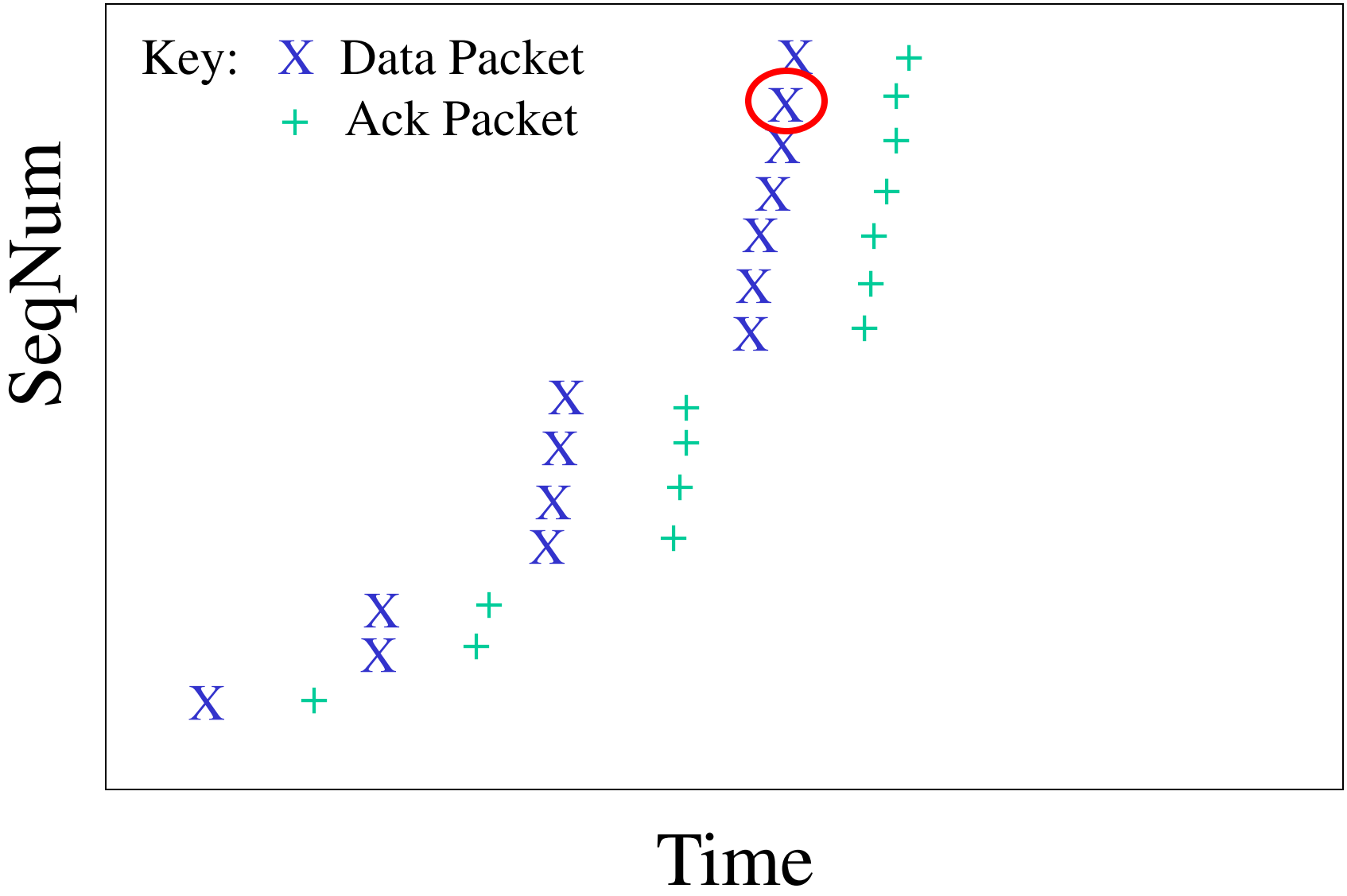
- What happens when a packet loss occurs?

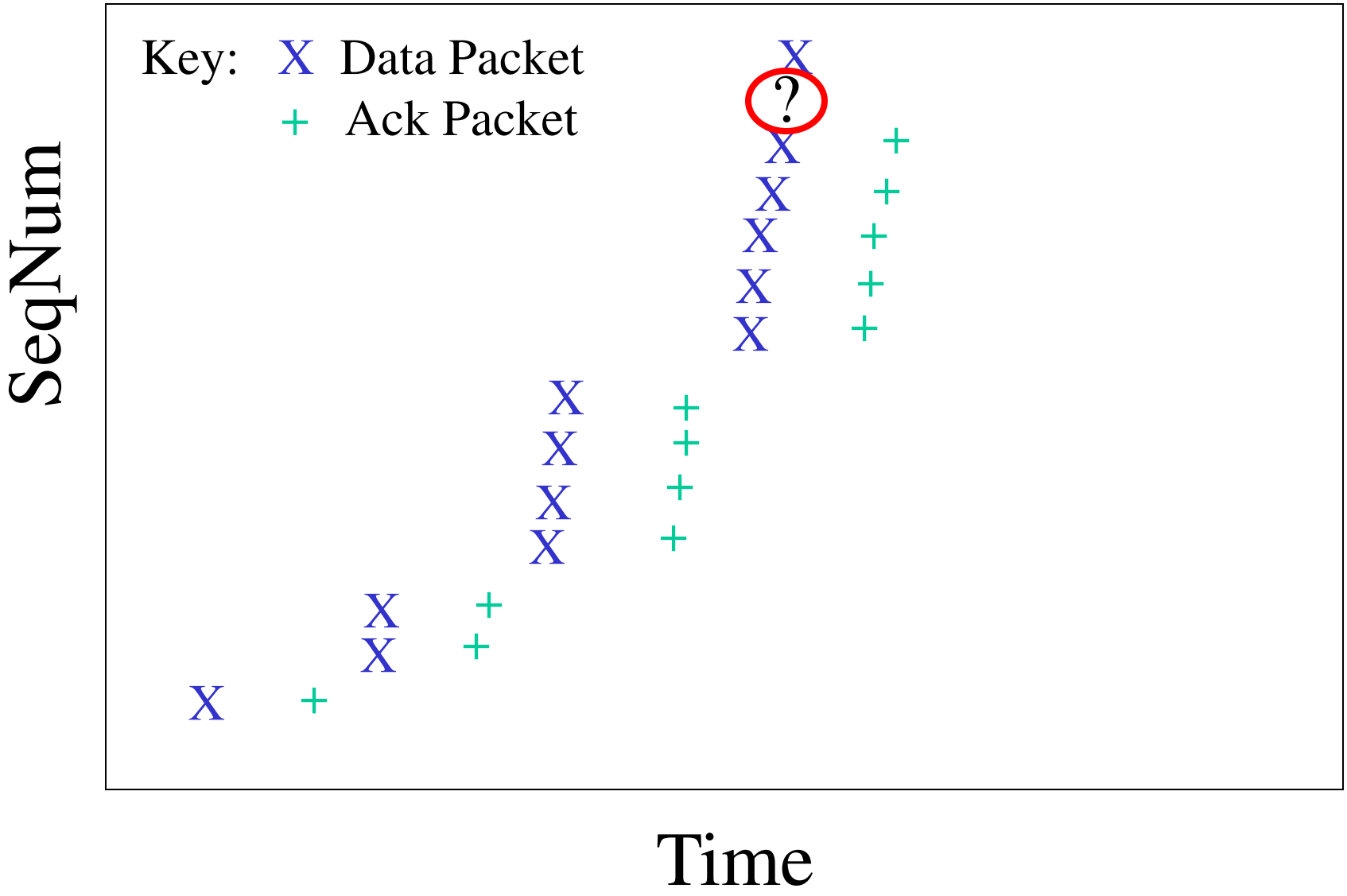
- Quiz Time...
  - Consider a 14-packet Web document
  - For simplicity, consider only a single packet loss

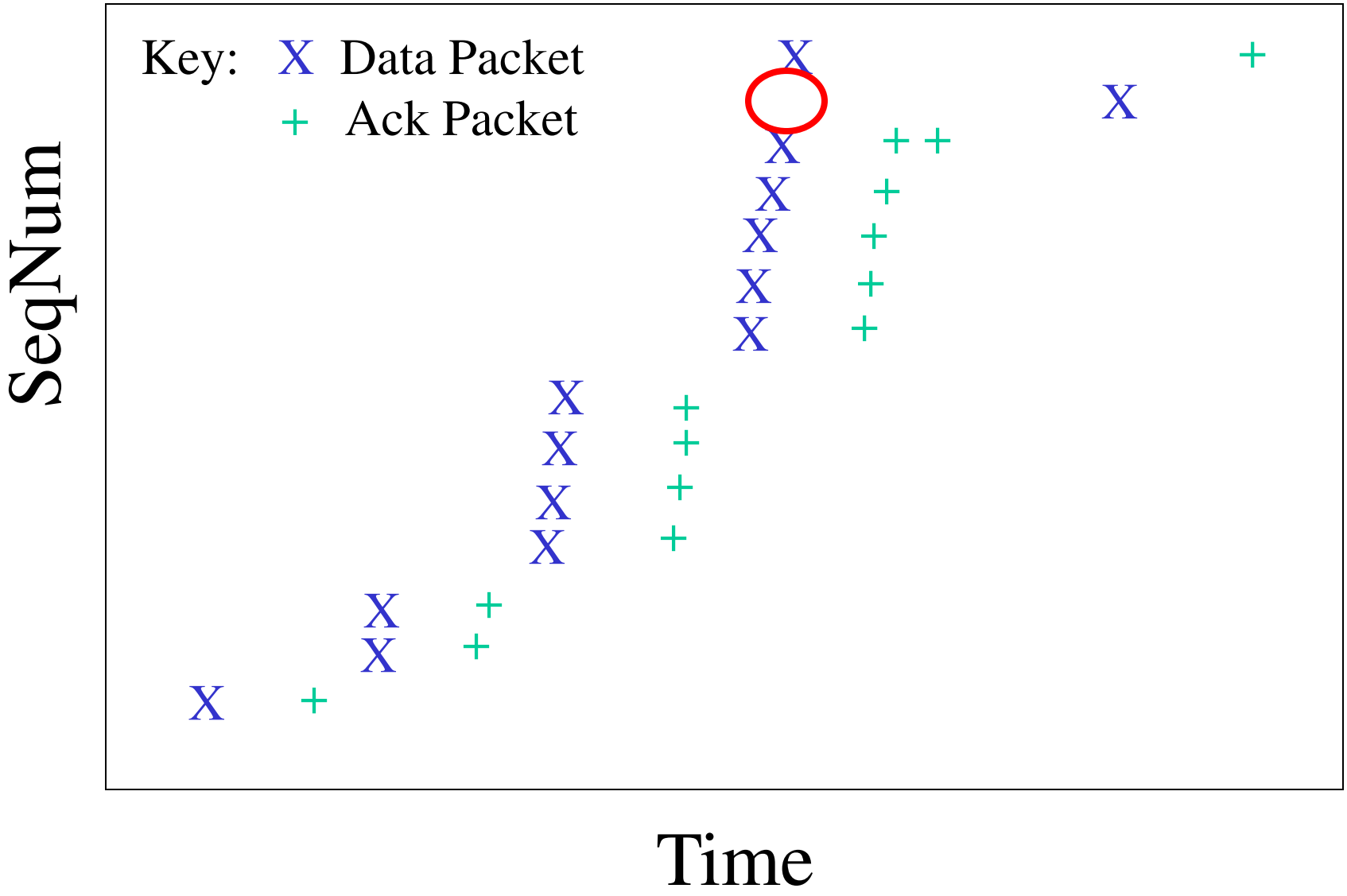

Key: X Data Packet
+ Ack Packet
SeqNum
Time

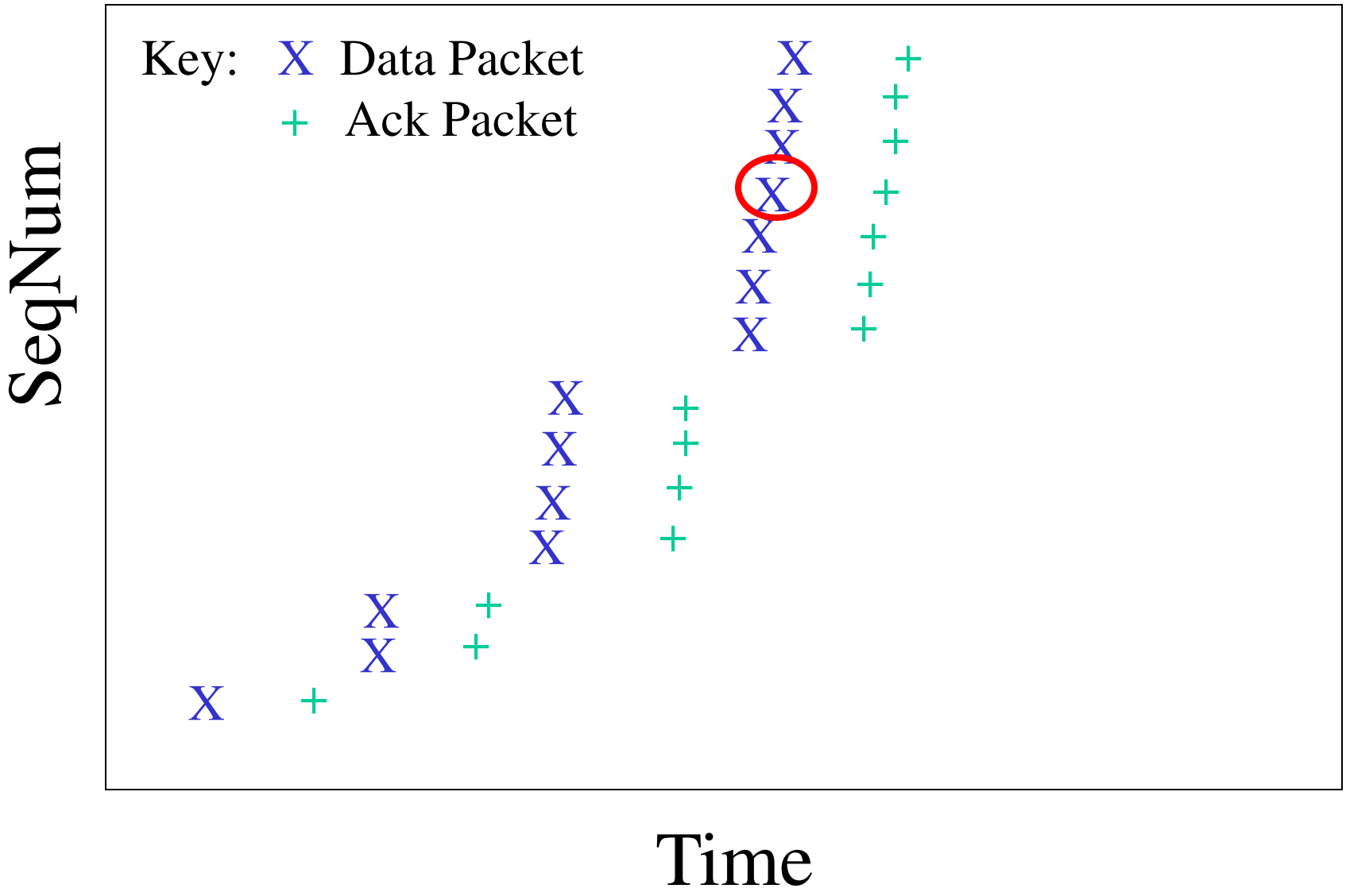

Key: X Data Packet
+ Ack Packet
?
SeqNum
Time

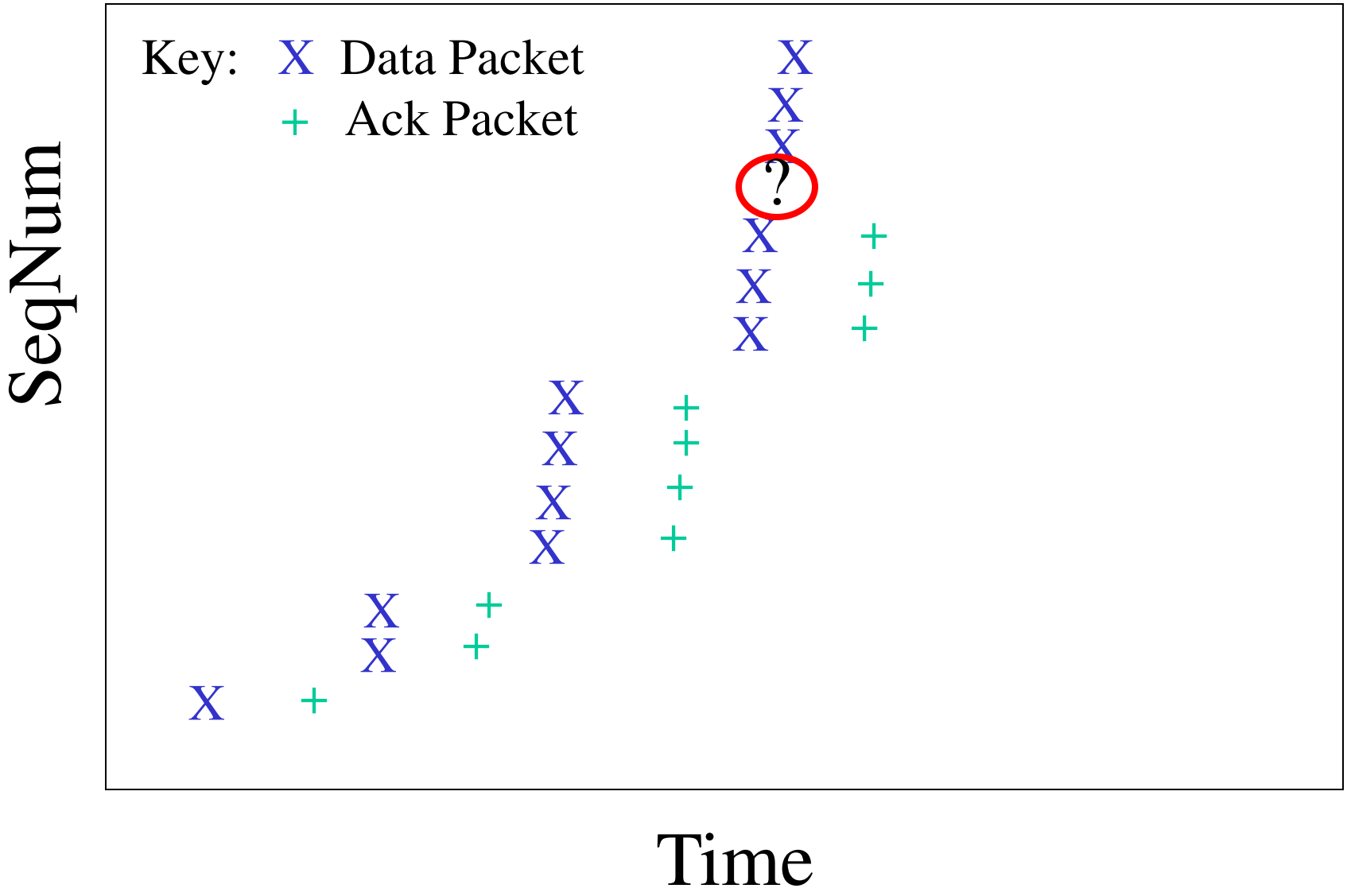

Key: X Data Packet
+ Ack Packet
SeqNum
Time

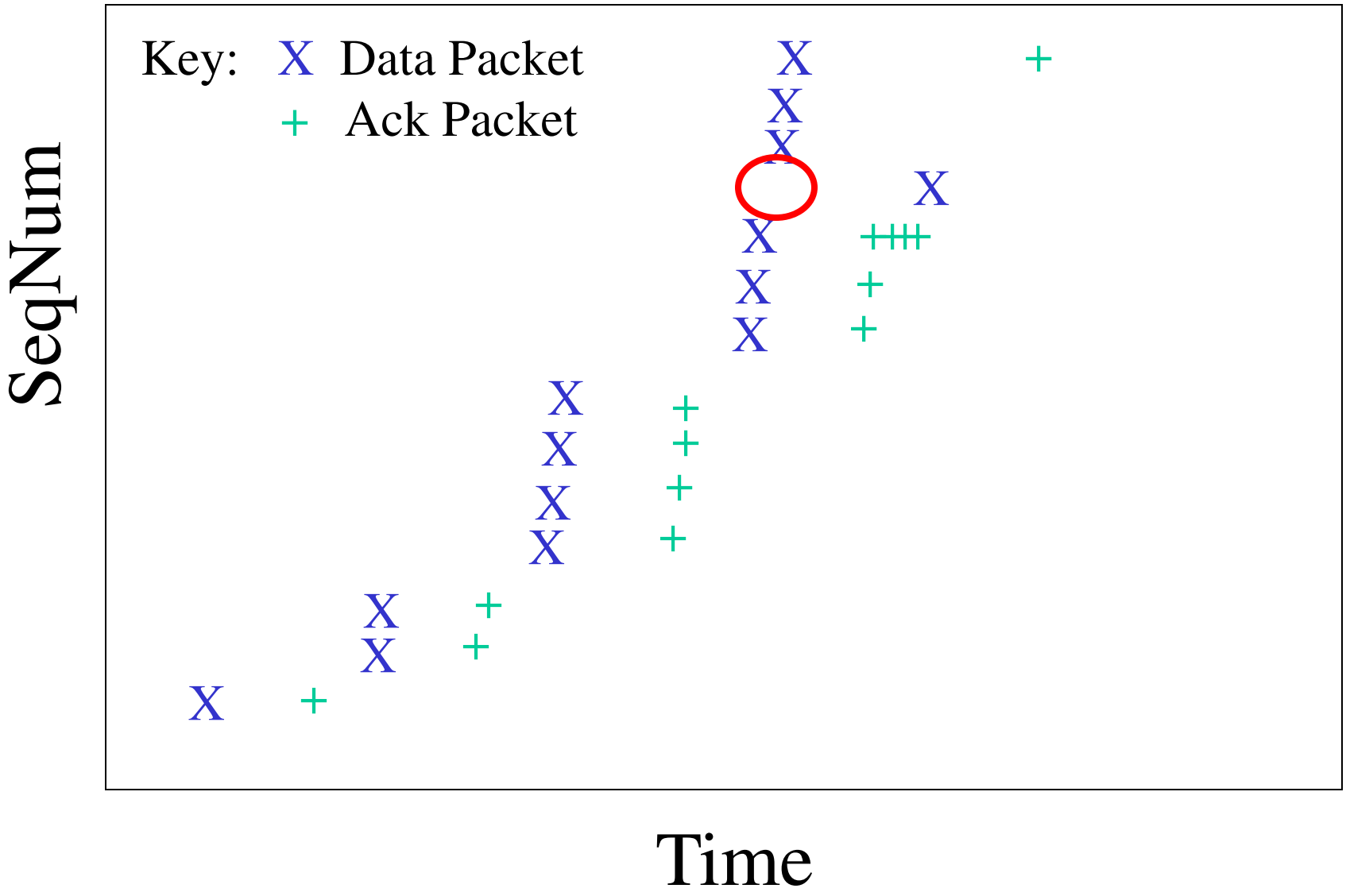

Key: X Data Packet
+ Ack Packet
SeqNum
Time

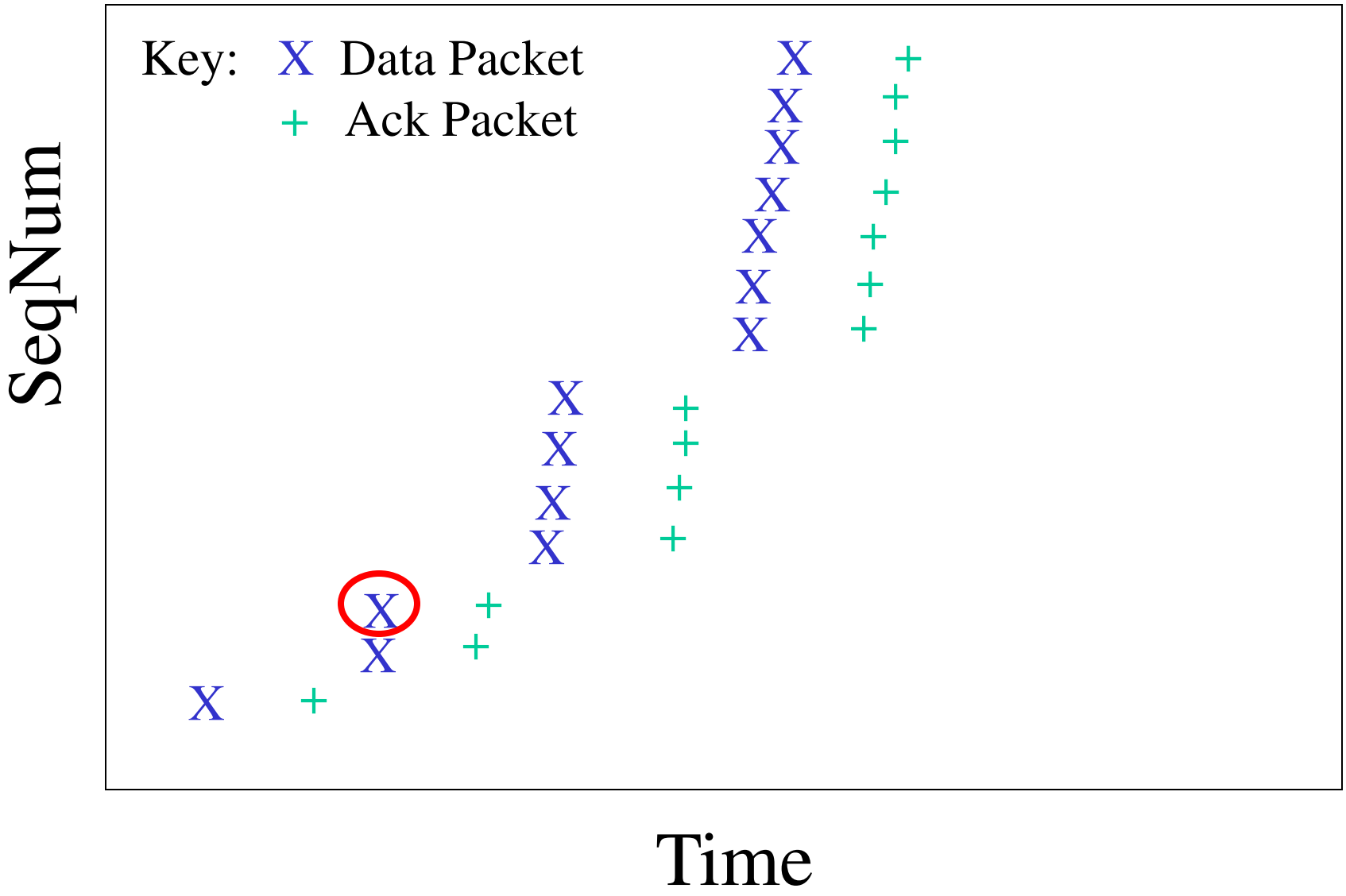

Key: X Data Packet
+ Ack Packet
?
SeqNum
Time

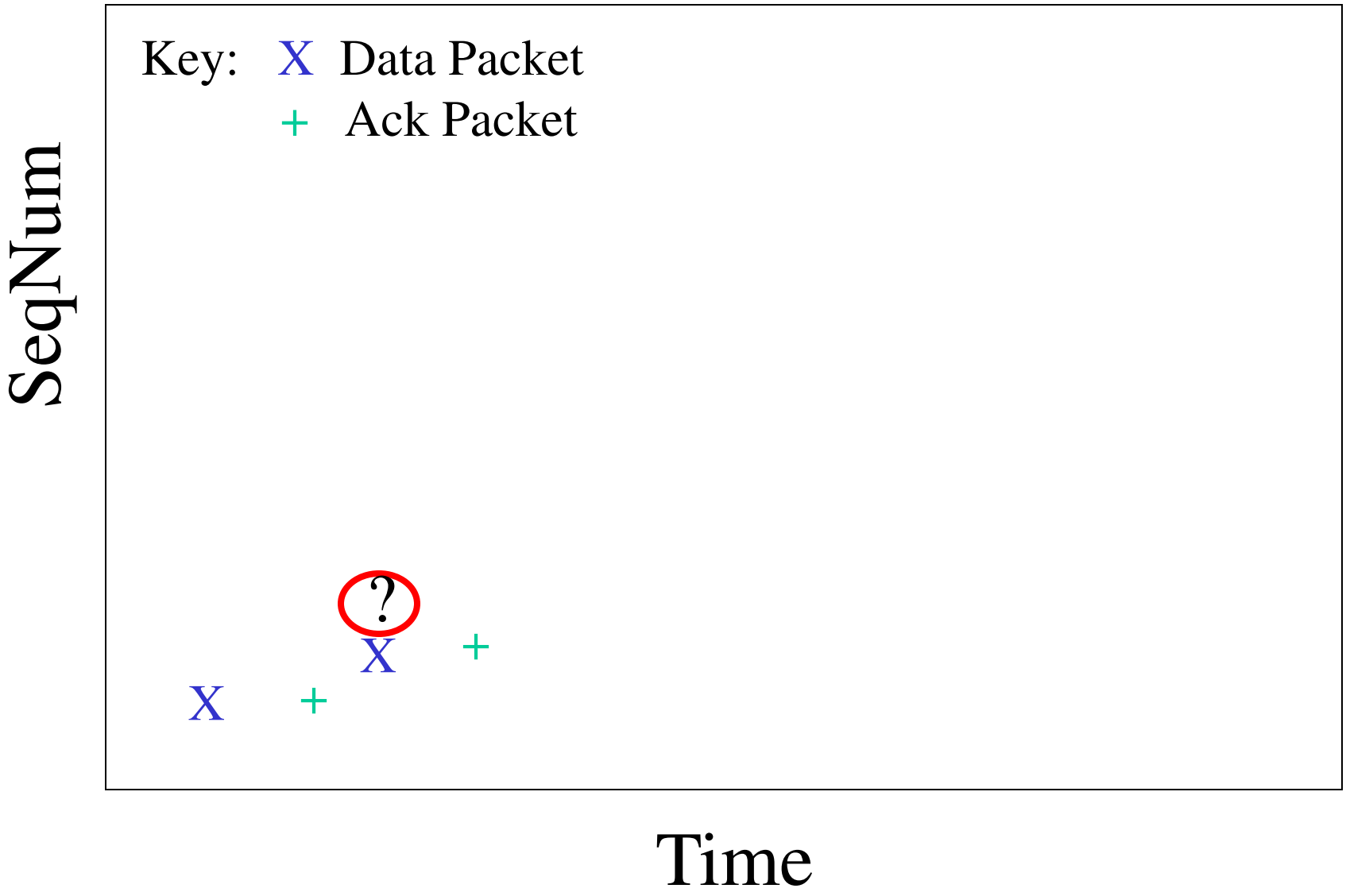

Key: X Data Packet
+ Ack Packet
SeqNum
Time

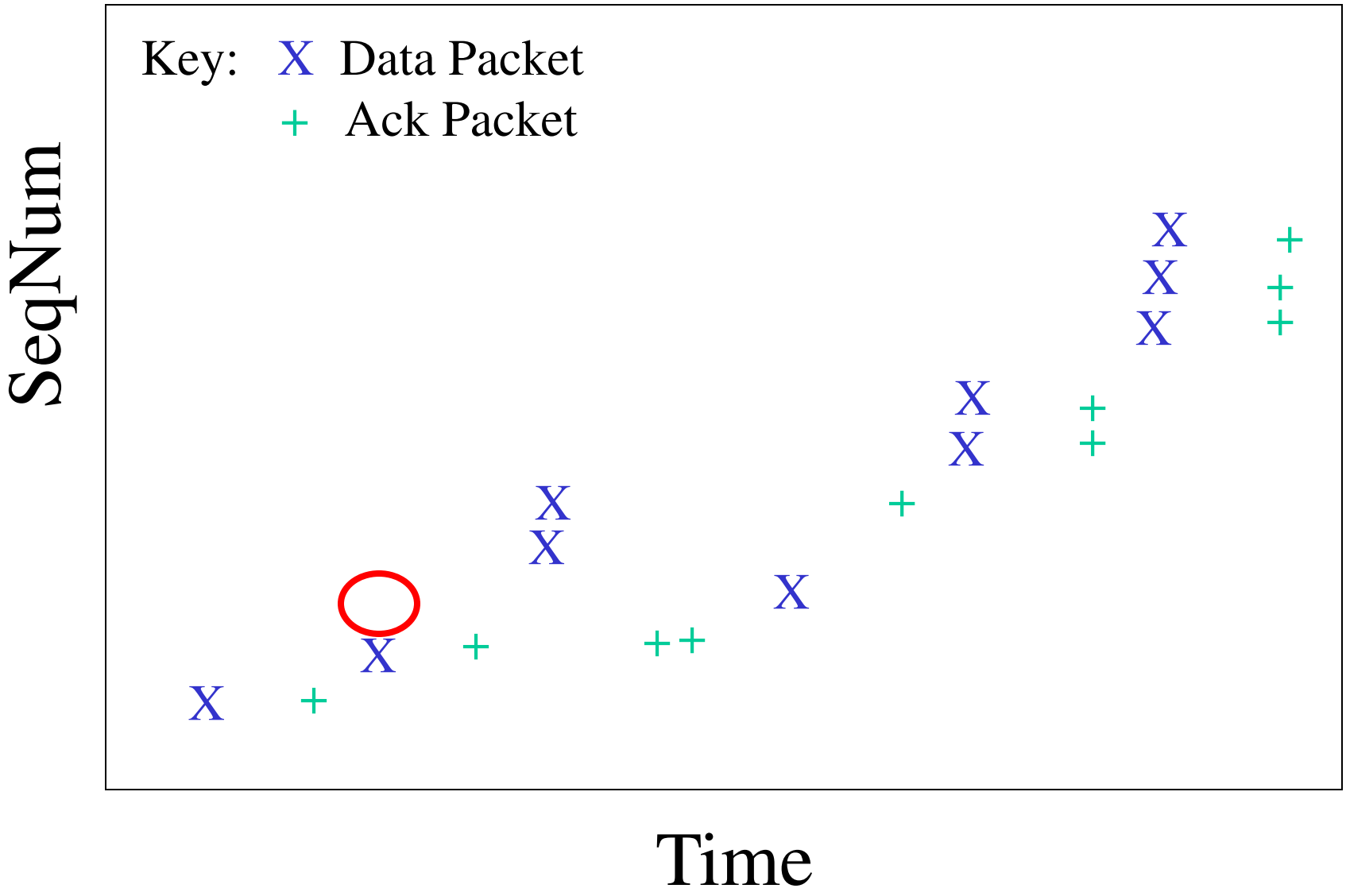

Key: X Data Packet
+ Ack Packet
SeqNum
Time


Key: X Data Packet
+ Ack Packet
?
SeqNum
Time


Key: X Data Packet
+ Ack Packet
SeqNum
Time


Key: X Data Packet
+ Ack Packet
SeqNum
Time


Key: X Data Packet
+ Ack Packet
SeqNum
?
X
+
X
+
Time


Key: X Data Packet
+ Ack Packet
SeqNum
Time

TCP 201 (Cont'd)

- Main observation:
  - "Not all packet losses are created equal"

- Losses early in the transfer have a huge adverse impact on the transfer latency
- Losses near the end of the transfer always cost at least a retransmit timeout
- Losses in the middle may or may not hurt, depending on congestion window size at the time of the loss

Congratulations!

- You are now a TCP expert!
- Or you are so confused now that you will finally take my advice and buy the TCP book and read through some chapters