Lecture 5: Application Layer
Overview and HTTP

COMP 332, Spring 2024
Victoria Manfredi

WESLEYAN UNIVERSITY

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved.

Today

Announcements

- homework 2 due Wed. by 11:59p

Network Measurement

- Wireshark: looking at real traffic
- Sources of delay
- Traceroute

Application layer

- Overview
- Web and HTTP

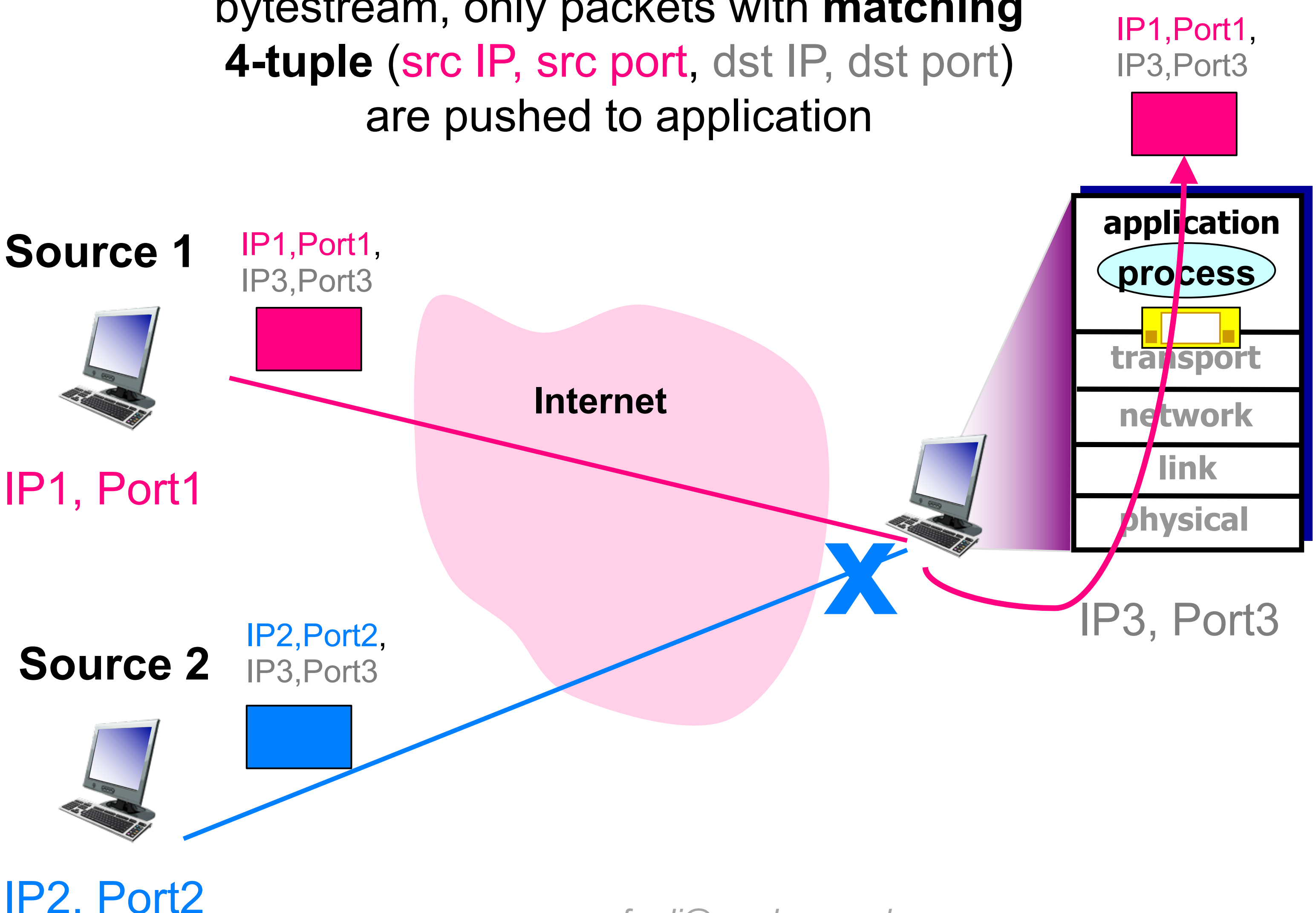
HTTP protocol

- Requests, responses, error codes

Last Time

TCP Socket

Establish connection, read/write bytestream, only packets with **matching 4-tuple** (src IP, src port, dst IP, dst port) are pushed to application



Network Measurement

WIRESHARK

How can I look at network traffic?

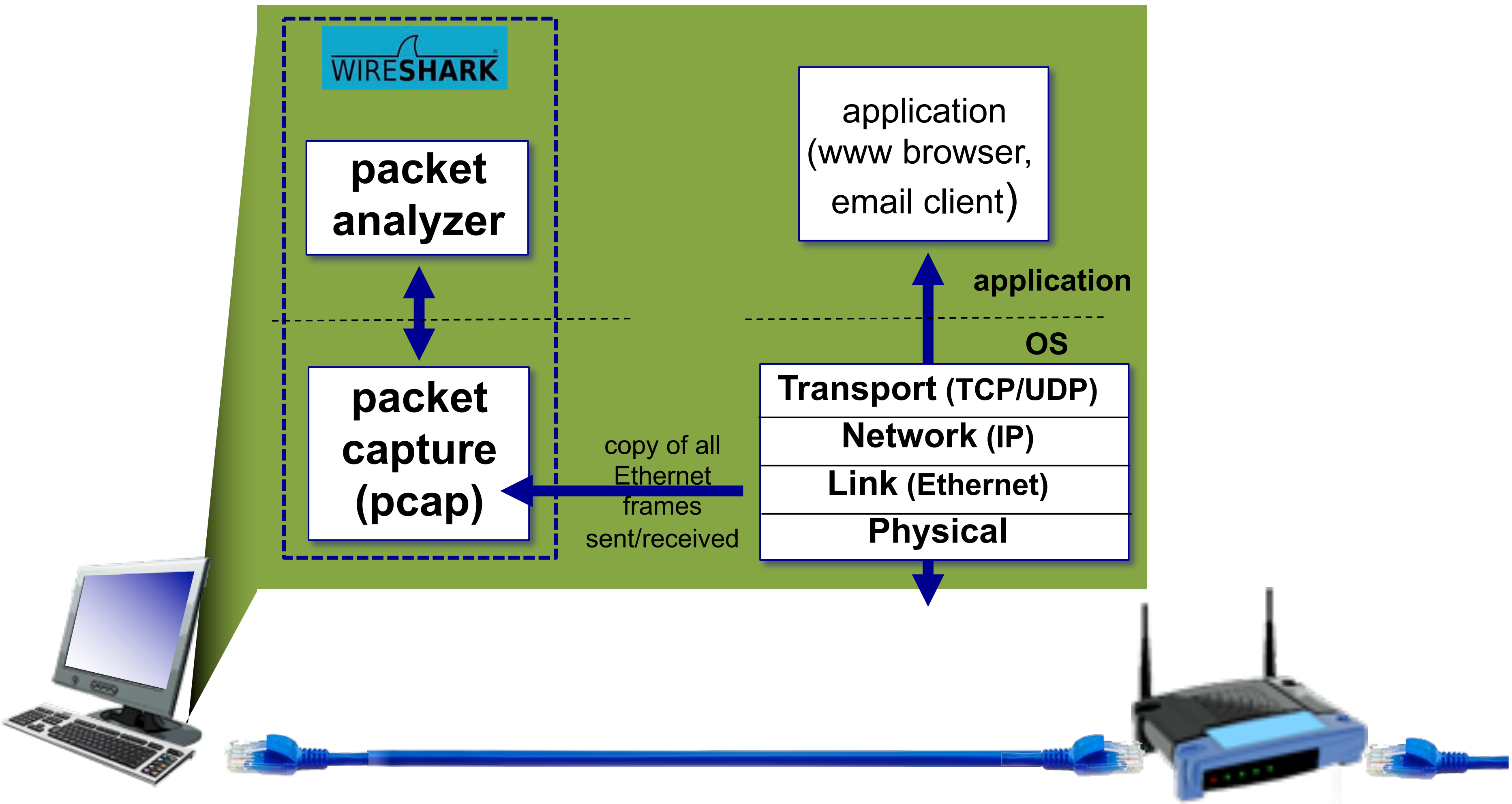
Packet sniffer

- passively observes messages transmitted and received on a particular network interface by processes running on your computer
- often requires root privileges to run

Popular packet sniffers

- Wireshark (also command-line version, tshark)
- tcpdump (Unix) and WinDump (Windows)
- use command line sniffers to analyze packet traces with bash script

Packet sniffer operation



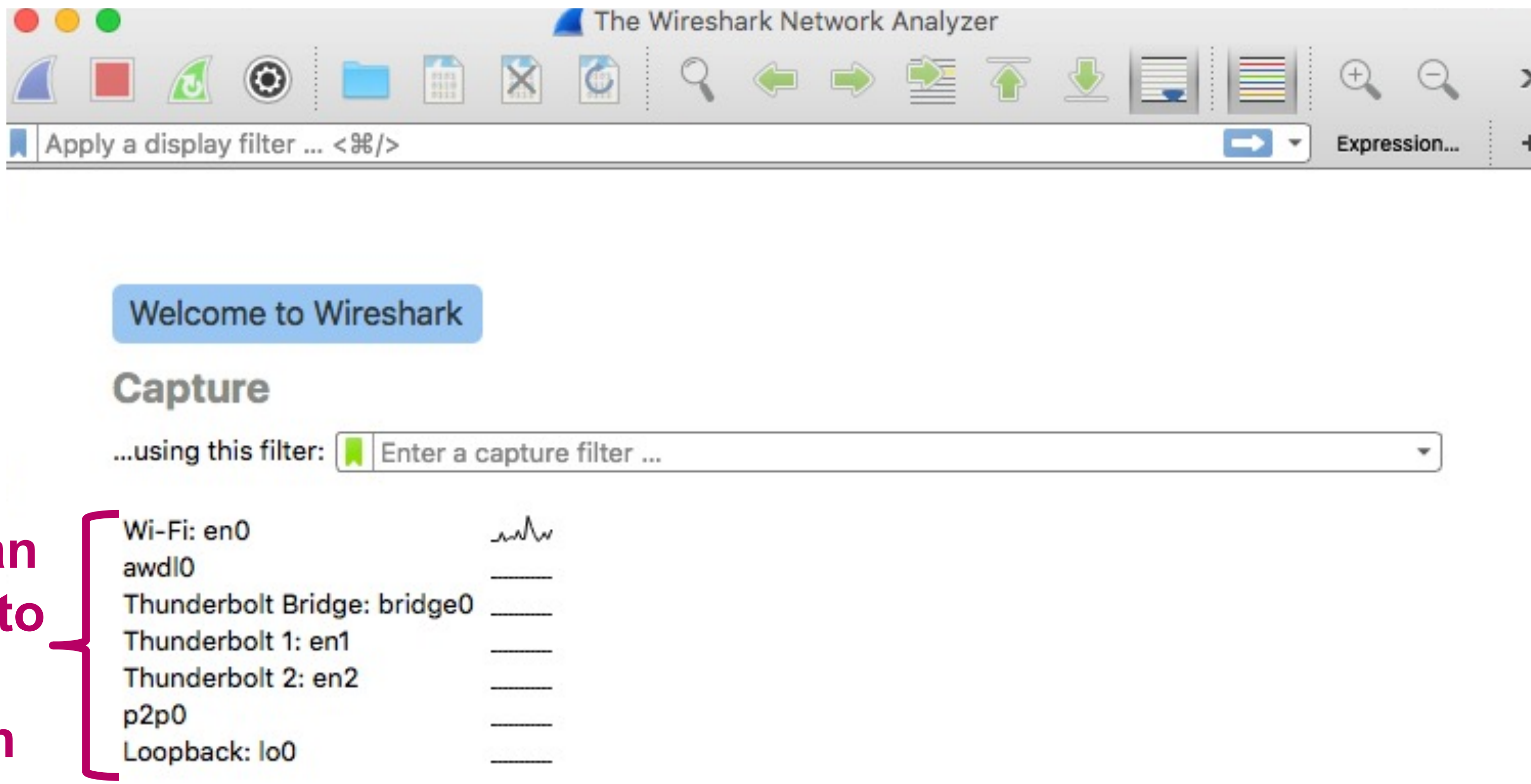
Wireshark

Install

– https://www.wireshark.org/download.html

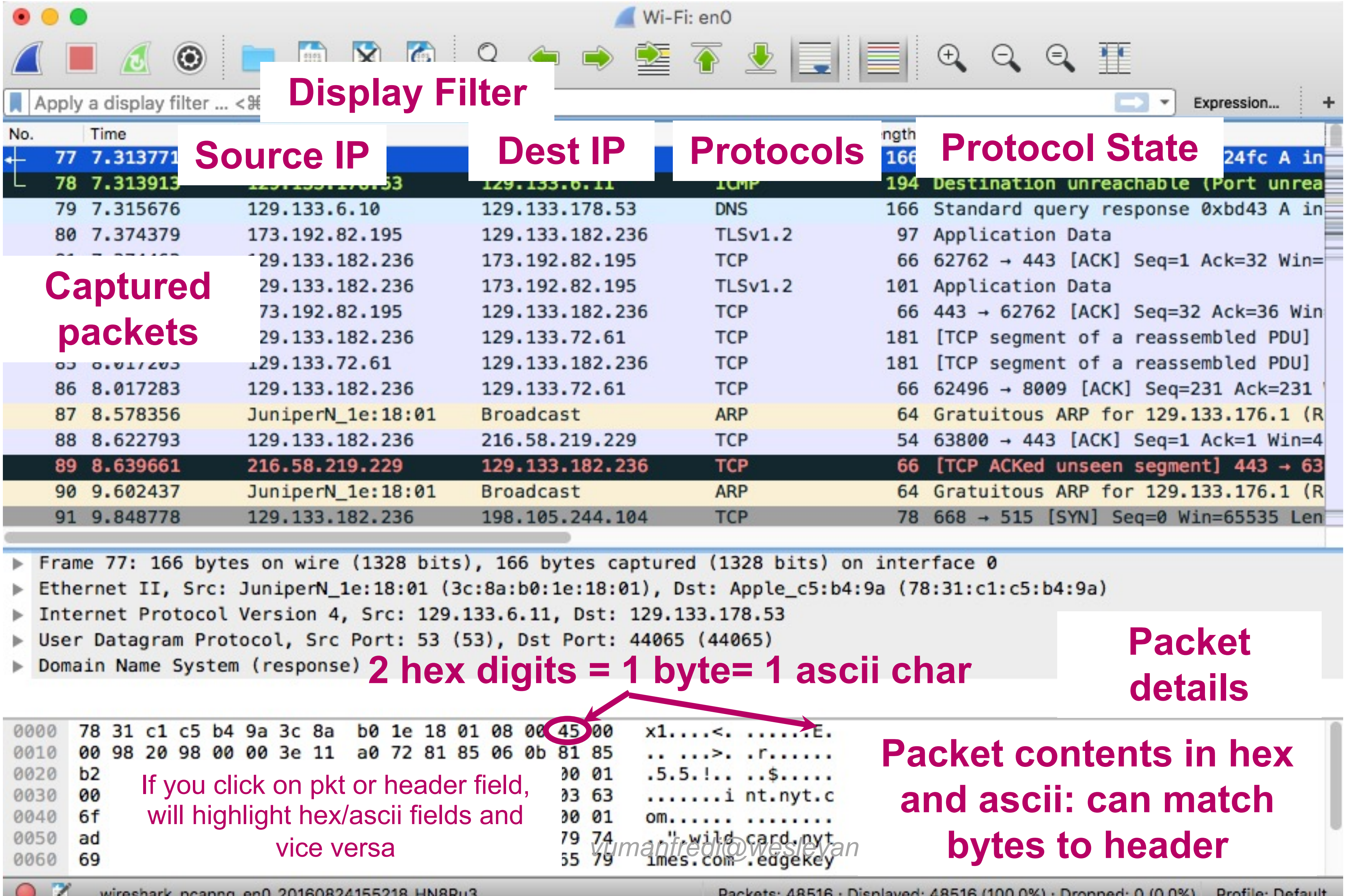
Run

– type Wireshark in terminal, or double-click icon
– Wireshark display may look different for Linux vs. Mac vs. Windows



Choose an interface to capture traffic on

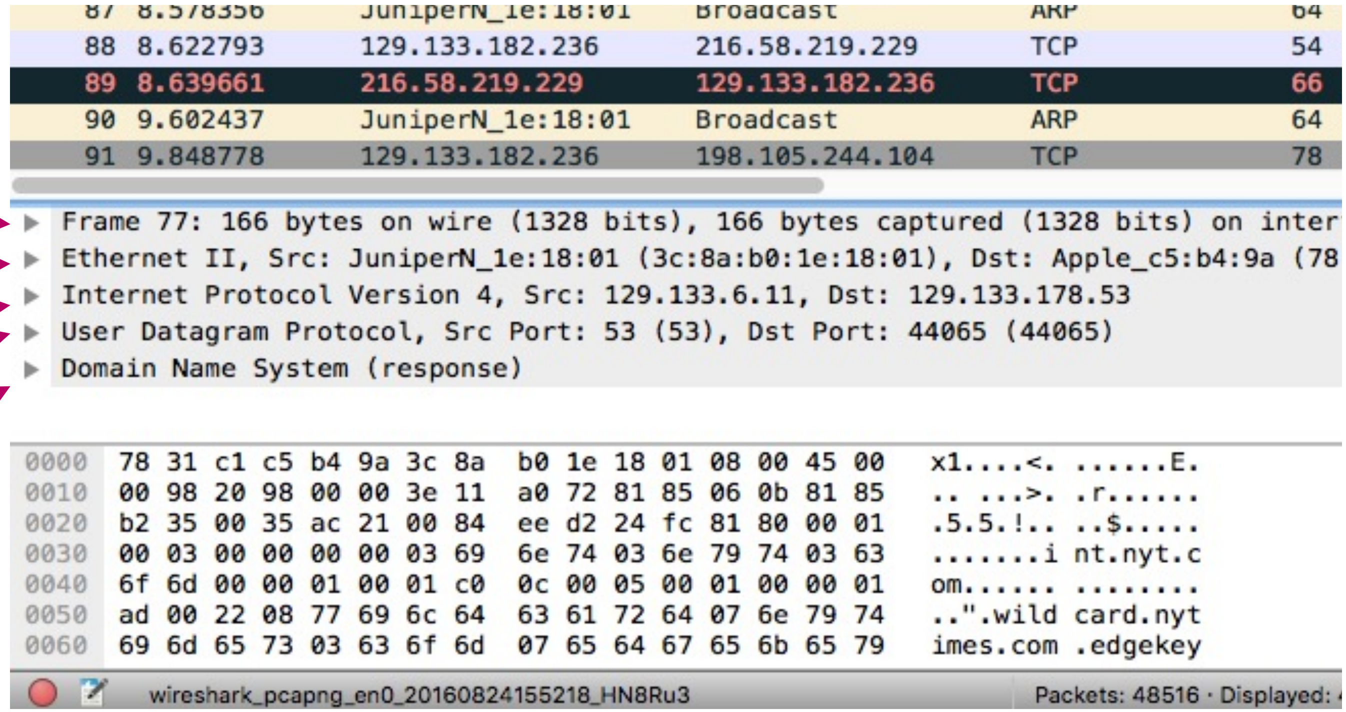
What do we see?



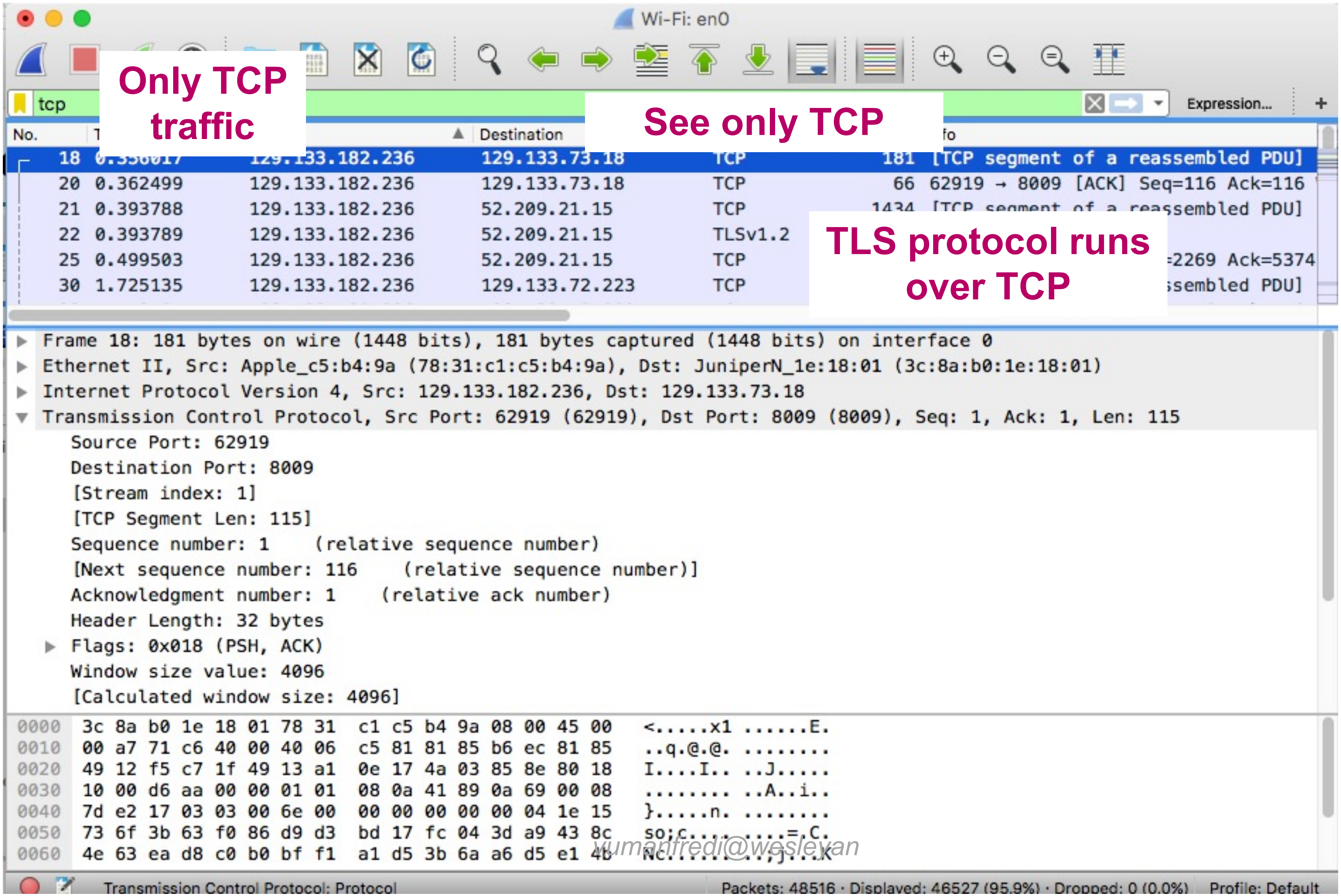
What do we see?

Layers

- Physical
- Link
- Network
- Transport
- Application



Add a filter


Wi-Fi: en0
tcp
Expression...
Only TCP traffic
See only TCP
TLS protocol runs over TCP
No.
Destination
18 0.356017 129.133.182.236 129.133.73.18 TCP 181 [TCP segment of a reassembled PDU]
20 0.362499 129.133.182.236 129.133.73.18 TCP 66 62919 → 8009 [ACK] Seq=116 Ack=116
21 0.393788 129.133.182.236 52.209.21.15 TCP 1434 [TCP segment of a reassembled PDU]
22 0.393789 129.133.182.236 52.209.21.15 TLSv1.2
25 0.499503 129.133.182.236 52.209.21.15 TCP =2269 Ack=5374
30 1.725135 129.133.182.236 129.133.72.223 TCP ssembled PDU]
Frame 18: 181 bytes on wire (1448 bits), 181 bytes captured (1448 bits) on interface 0
Ethernet II, Src: Apple_c5:b4:9a (78:31:c1:c5:b4:9a), Dst: JuniperN_1e:18:01 (3c:8a:b0:1e:18:01)
Internet Protocol Version 4, Src: 129.133.182.236, Dst: 129.133.73.18
Transmission Control Protocol, Src Port: 62919 (62919), Dst Port: 8009 (8009), Seq: 1, Ack: 1, Len: 115
Source Port: 62919
Destination Port: 8009
[Stream index: 1]
[TCP Segment Len: 115]
Sequence number: 1 (relative sequence number)
[Next sequence number: 116 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Header Length: 32 bytes
Flags: 0x018 (PSH, ACK)
Window size value: 4096
[Calculated window size: 4096]
0000 3c 8a b0 1e 18 01 78 31 c1 c5 b4 9a 08 00 45 00 <.....x1 ......E.
0010 00 a7 71 c6 40 00 40 06 c5 81 81 85 b6 ec 81 85 ..q.@.@. ....
0020 49 12 f5 c7 1f 49 13 a1 0e 17 4a 03 85 8e 80 18 I....I.. ..J.....
0030 10 00 d6 aa 00 00 01 01 08 0a 41 89 0a 69 00 08 ....A..i..
0040 7d e2 17 03 03 00 6e 00 00 00 00 00 00 04 1e 15 }.....n. ....
0050 73 6f 3b 63 f0 86 d9 d3 bd 17 fc 04 3d a9 43 8c so;c.... ....=.C.
0060 4e 63 ea d8 c0 b0 bf f1 a1 d5 3b 6a a6 d5 e1 4b
yumanfredi@wesleyan
Transmission Control Protocol: Protocol
Packets: 48516 · Displayed: 46527 (95.9%) · Dropped: 0 (0.0%) Profile: Default

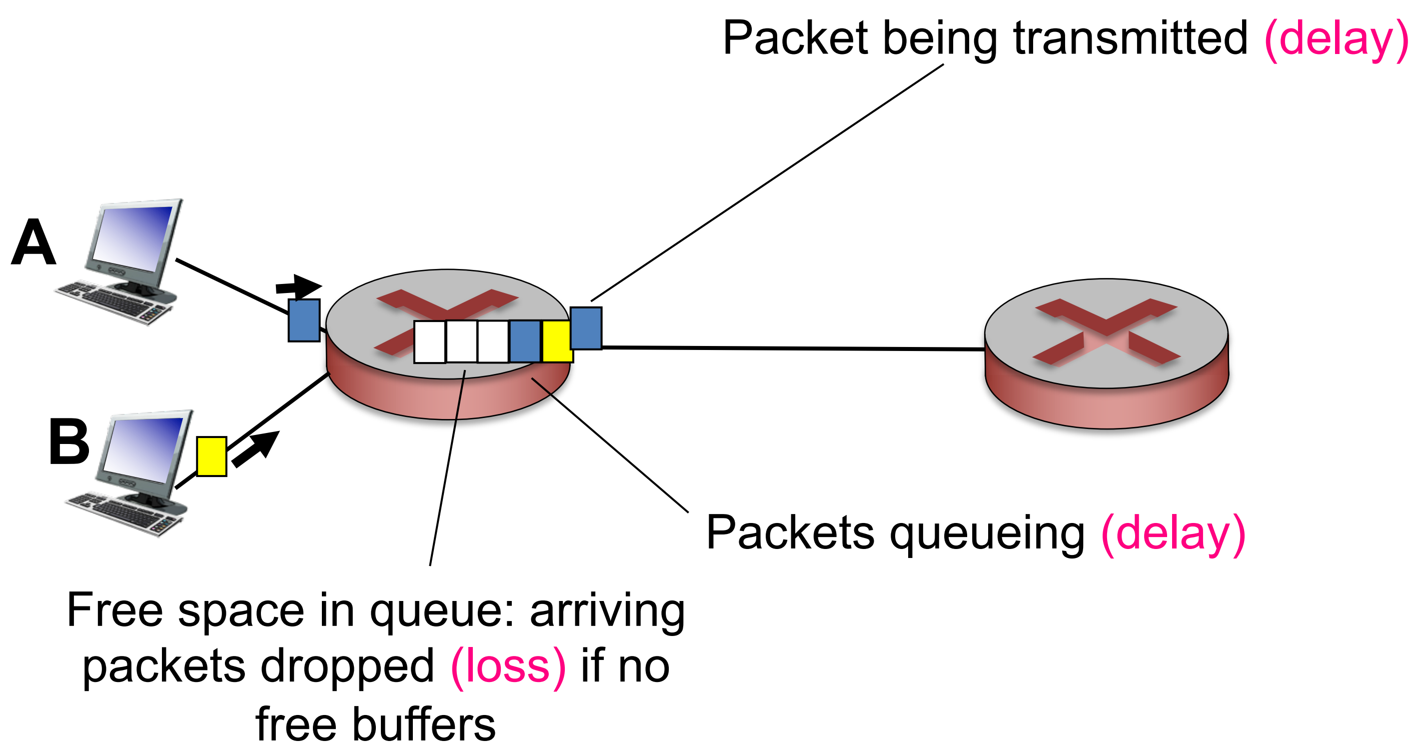
Network Measurement

SOURCES OF DELAY

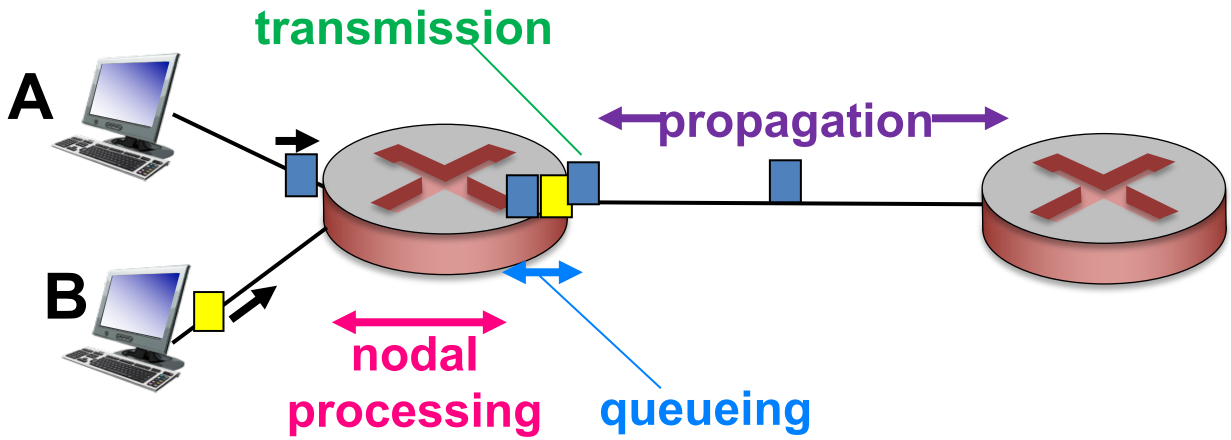
How do delay and loss occur?

If link arrival rate > transmission rate link for some time

- packets will queue, wait to be transmitted on link
- packets can be dropped (lost) if memory (buffer) fills up
- lost packet may be retransmitted by previous node, by source end system, or not at all



Four sources of packet delay

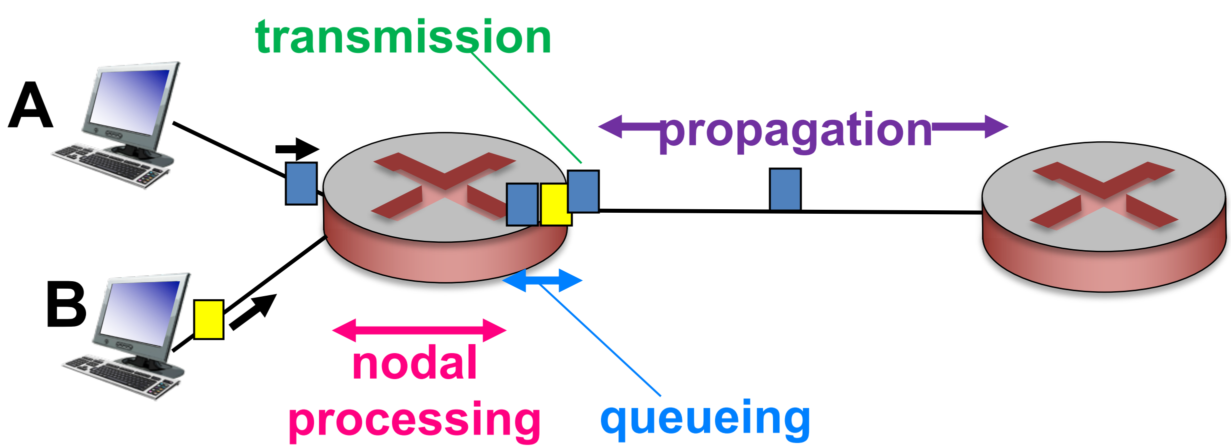


$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{proc}: processing delay
- check bit errors
- determine output link
- fast: typically < msec
- usually done in hardware not software

d_{queue}: queueing delay
- time waiting at output link for transmission
- depends on congestion level of router

Four sources of packet delay



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{trans}: transmission delay

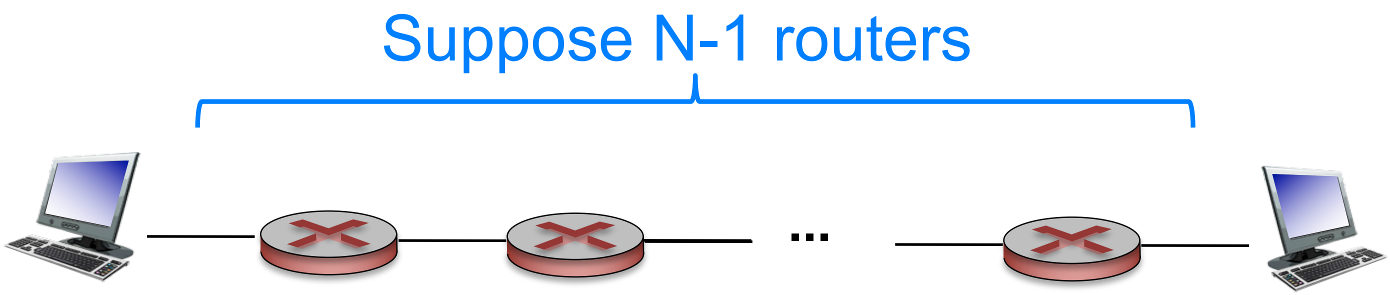
- depends on link bandwidth
- L: packet length (bits)
- R: link bandwidth (bps)
- $d_{trans} = L/R$

d_{trans} and d_{prop} ***very*** different

d_{prop}: propagation delay

- μs (within campus) to ms (satellite link)
- d: length of physical link
- s: propagation speed (~2×10^8 m/s)
- $d_{prop} = d/s$

End-to-end delay



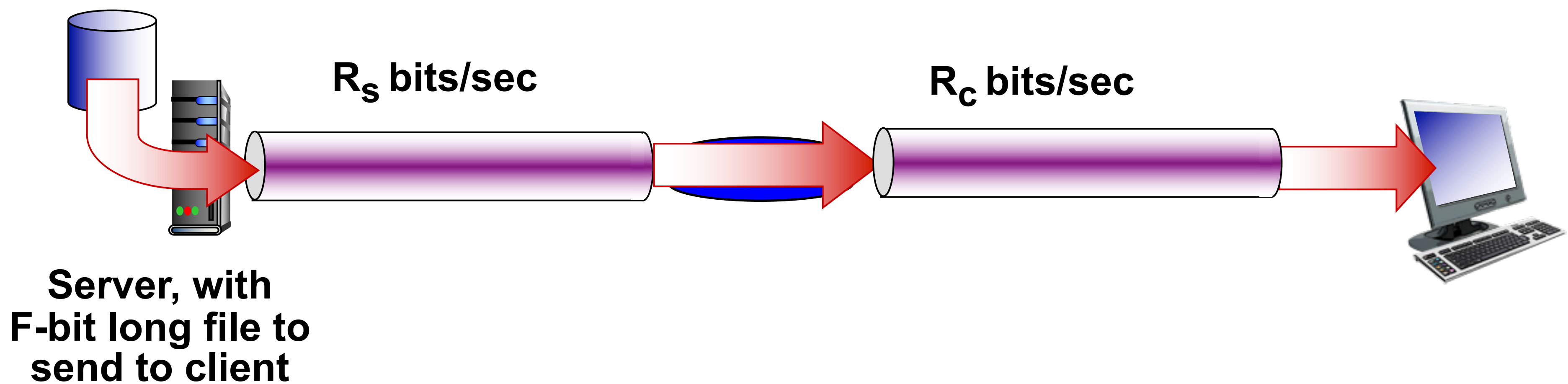
Q: what is end-end delay ignoring queuing delay?

End-end delay = $N * (d_{proc} + d_{trans} + d_{prop})$

Throughput

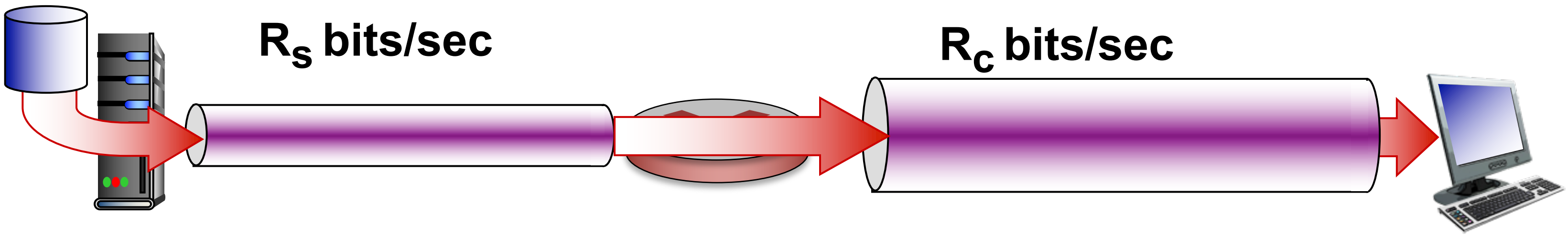
Rate at which bits transferred between sender/receiver

- measured in bits/time unit
- instantaneous: rate at given point in time
- average: rate over longer period of time

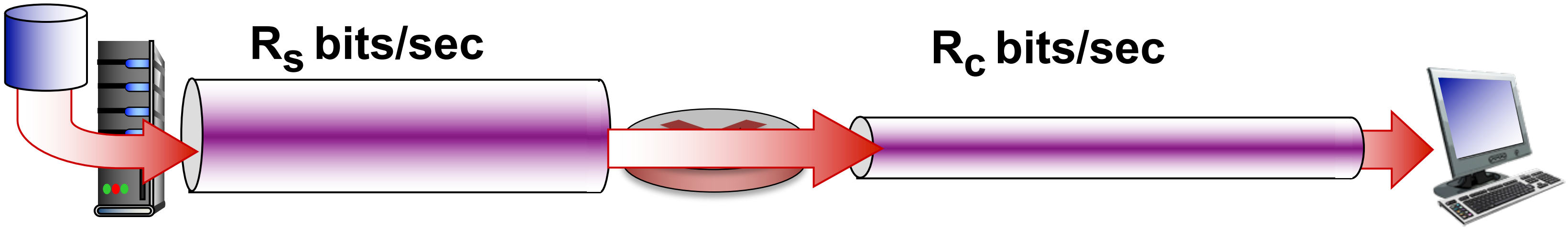


Throughput

$R_s < R_c$ What is average end-end throughput?



$R_s > R_c$ What is average end-end throughput?



bottleneck link

link on end-end path that constrains end-end throughput

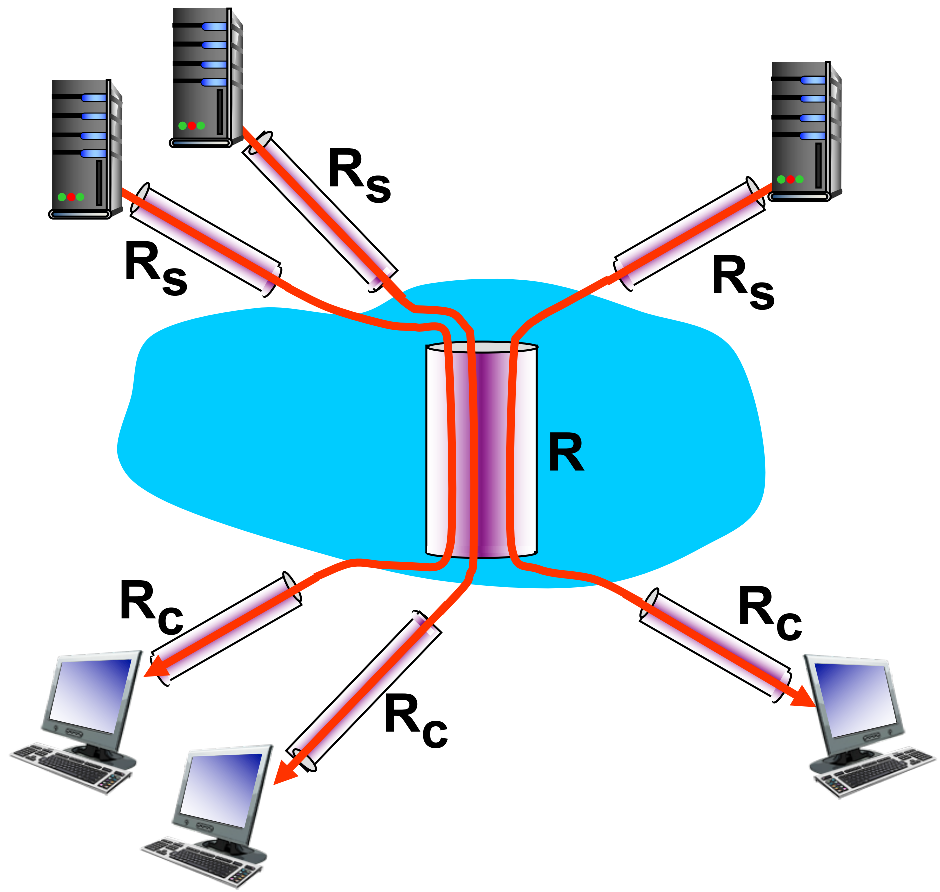
Internet scenario

Per-connection end-end throughput

- $min(R_c, R_s, R/10)$

In practice

- R_c or R_s is often bottleneck



10 connections (fairly) share *R* bits/sec backbone bottleneck link

Network Measurement

TRACEROUTE

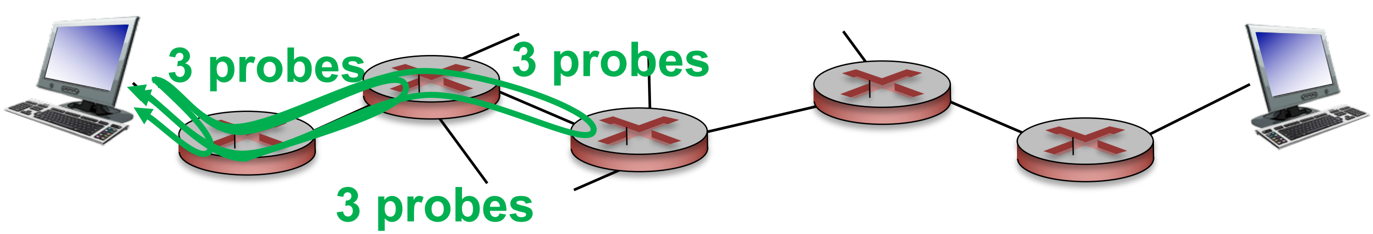
Real Internet delays and routes

Traceroute program

- provides delay measurement from source to router along end-end Internet path towards destination

How?

- for all i:
  - sends three packets that will reach router i on path towards destination
    - sets packet time-to-live (TTL) to i
  - router i will return packets to sender
  - sender times interval between transmission and reply for each packet
    - measures Round Trip Time (RTT) delay



Note

- different probe packets may take different paths, so delays can vary

Real Internet delays, routes

traceroute: from wesleyan network to cs.stanford.edu

```
> traceroute cs.stanford.edu
traceroute to cs.stanford.edu (171.64.64.64), 64 hops max, 52 byte packets
 1  129.133.176.1 (129.133.176.1)  6.138 ms  2.365 ms  3.913 ms
 2  172.16.100.1 (172.16.100.1)  3.857 ms  3.361 ms  5.545 ms
 3  129.133.2.5 (129.133.2.5)  1.958 ms  3.068 ms  1.906 ms
 4  129.133.4.11 (129.133.4.11)  3.865 ms  2.879 ms  3.850 ms
 5  72.10.111.129 (72.10.111.129)  1.984 ms  3.144 ms  3.761 ms
 6  64.251.60.122 (64.251.60.122)  5.928 ms  4.959 ms  4.910 ms
 7  enrt083h-9k-te0-3-0-7.net.cen.ct.gov (72.10.125.22)  7.003 ms  6.982 ms  5.593 ms
 8  enrt078h-9k-te-0-0-0-6-dwdm-1532-68.net.cen.ct.gov (67.218.83.185)  5.779 ms  6.966 ms  6.300 ms
 9  cen-re-nox300gw1.nox.org (192.5.89.202)  7.421 ms  5.172 ms  5.948 ms
10  nox300gw1-cen-re.nox.org (192.5.89.201)  8.196 ms  8.217 ms  8.240 ms
11  192.5.89.22 (192.5.89.22)  9.766 ms  7.964 ms  7.810 ms
12  i2-re-nox1sumgw1.nox.org (192.5.89.18)  12.955 ms  7.642 ms  8.033 ms
13  et-7-0-0.4079.sdn-sw.alba.net.internet2.edu (162.252.70.96)  11.953 ms  10.251 ms  12.146 ms
14  et-3-1-0.4079.rtsw.clev.net.internet2.edu (162.252.70.93)  21.406 ms  20.401 ms  21.959 ms
15  ae-1.4079.sdn-sw.eqch.net.internet2.edu (162.252.70.131)  29.059 ms  30.883 ms  29.264 ms
16  ae-2.4079.rtsw.chic.net.internet2.edu (162.252.70.132)  29.075 ms  30.298 ms  29.413 ms
17  ae-3.4079.rtsw.kans.net.internet2.edu (162.252.70.141)  40.831 ms  40.250 ms  41.068 ms
18  ae-5.4079.rtsw.salt.net.internet2.edu (162.252.70.145)  60.625 ms  61.459 ms  60.568 ms
19  ae-1.4079.rtsw.losa.net.internet2.edu (162.252.70.114)  72.171 ms  73.579 ms  74.209 ms
20  hpr-lax-hpr2--i2-r&e.cenic.net (137.164.26.200)  73.938 ms  73.487 ms  72.439 ms
21  hpr-svl-hpr3--lax-hpr3-100ge.cenic.net (137.164.25.74)  83.925 ms  84.645 ms  83.688 ms
22  hpr-stan-ge--svl-hpr2.cenic.net (137.164.27.162)  86.215 ms  86.925 ms  84.094 ms
23  csmx-west-rtr.sunet (171.64.255.214)  109.002 ms  144.984 ms  94.379 ms
24  cs.stanford.edu (171.64.64.64)  84.106 ms  84.984 ms  83.928 ms
```

3 delay measurements from cs.stanford.edu

cross-country links

* means no response (probe lost, router not replying)

Using wireshark

Run traceroute and see what traffic is generated

Application Layer
OVERVIEW

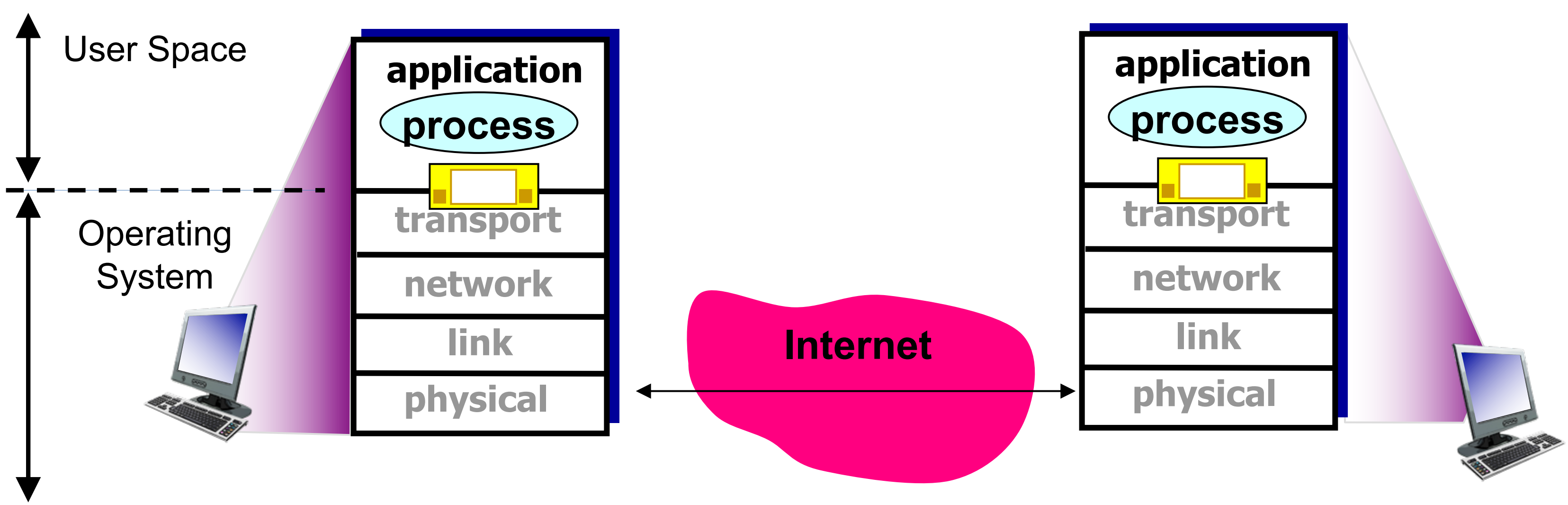
Application layer: where apps live

Application software

– processes running different hosts, communicate via messages

Application architecture

– client-server vs. peer-to-peer vs. hybrid
– overlaid on network architecture



Application layer protocols

Provide specific services to application

Define

- types of messages exchanged
  - e.g., request, response
- message syntax
  - fields in messages, how delineated
- message semantics
  - meaning of info in fields
- rules
  - for when and how processes send and respond to messages

Rely on transport layer

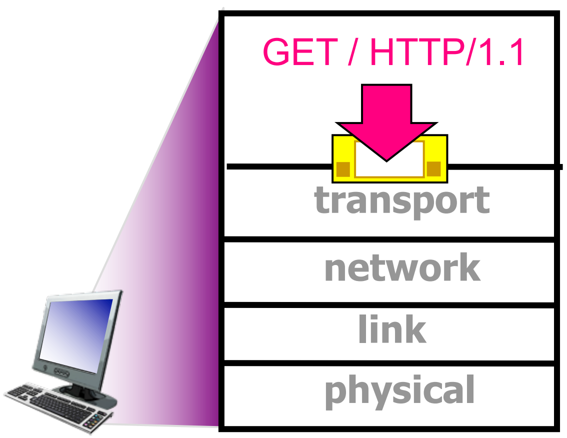
- to get messages from process on one host to process on another host

Open protocols

- defined in RFCs
- support interoperability
- e.g., HTTP, SMTP

Proprietary protocols

- e.g., Skype



Application requirements

Dictate what transport layer services application needs

TCP or UDP (or SSL/TCP or QUIC if you're Google)?

Service	App requirements
Reliable data transfer: does all data need to be received?	Loss-tolerant? E.g. video?
Throughput: does data need to be delivered quickly? Is app sending lots of data?	Bandwidth sensitive? E.g., video Elastic traffic? E.g., use as much/little bandwidth as available
Timing: does data need to be delivered at certain min rate?	Time-sensitive? E.g., voice, video need low delay
Security: does data need to be secured from eavesdroppers and modification?	Encryption? Data integrity? Endpoint authentication? Confidentiality?

Services provided by Internet transport protocols

TCP service

- connection-oriented
  - setup required between client and server processes
- reliable transport
  - messages delivered to destination process without error and in-order
- congestion control
  - sender reduces sending rate when network is overloaded
- flow control
  - sender reduces sending rate when destination is overloaded
- does not provide
  - timing, minimum throughput or delay guarantee, security

UDP service

- unreliable data transfer
  - best-effort service between sender and destination processes
- does not provide
  - reliability
  - flow control
  - congestion control
  - timing
  - throughput guarantee
  - security
  - connection setup

Q: why bother? Why is there a UDP?

Transport service requirements: common apps

Application	Data loss	Throughput	Time sensitive
File transfer	no loss	elastic	no
E-mail	no loss	elastic	no
Web documents	no loss	elastic	no
Real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
Stored audio/video	loss-tolerant	same as above	yes, few secs
Interactive games	loss-tolerant	few kbps up	yes, 100's msec
Text messaging	no loss	elastic	yes and no

Q: other apps you can think of?

Internet apps: application, transport protocols

Associated with each app is an app layer protocol: depending on app requirements, runs over specific transport protocols

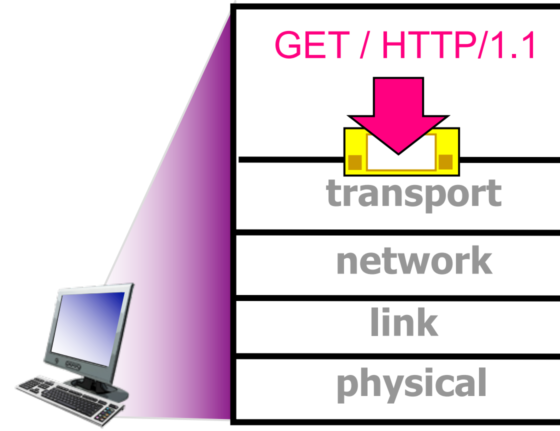
Application	Application layer protocol	Underlying transport protocol
E-mail	SMTP [RFC 2821]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Q: where does security come into play?

Securing TCP

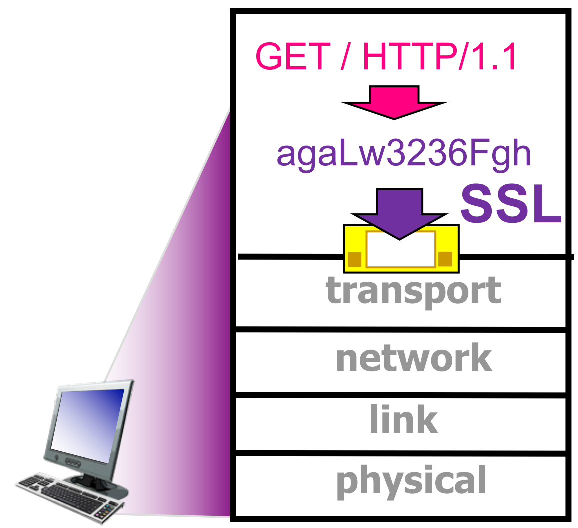
TCP & UDP

- no encryption: cleartext passwords sent into socket traverse Internet in cleartext



TLS/SSL

- at app layer
  - apps use SSL libraries, that "talk" to TCP
- provides encrypted TCP connection
  - data integrity
  - end-point authentication



TLS/SSL socket API

- cleartext passwords sent into socket traverse Internet encrypted

Q: Why does SSL run over TCP?
How is TLS/SSL related to OSI model?

Network Applications

WEB AND HTTP

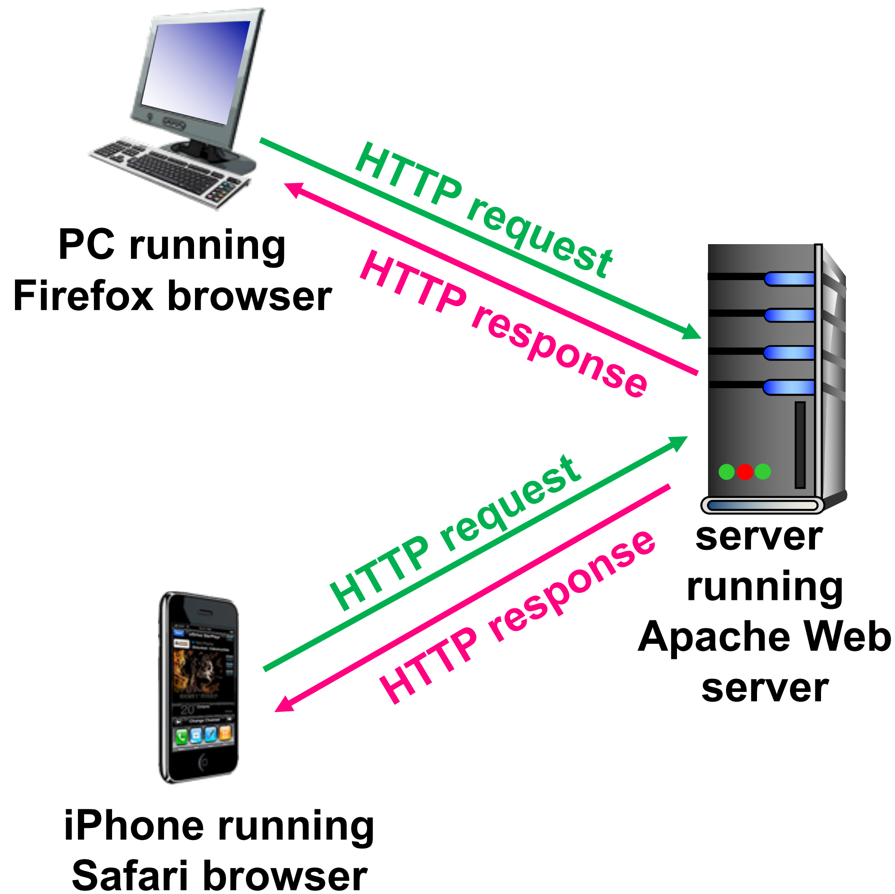
Web's application layer protocol

HTTP

- **H**yper**T**ext **T**ransfer **P**rotocol

Client/server model

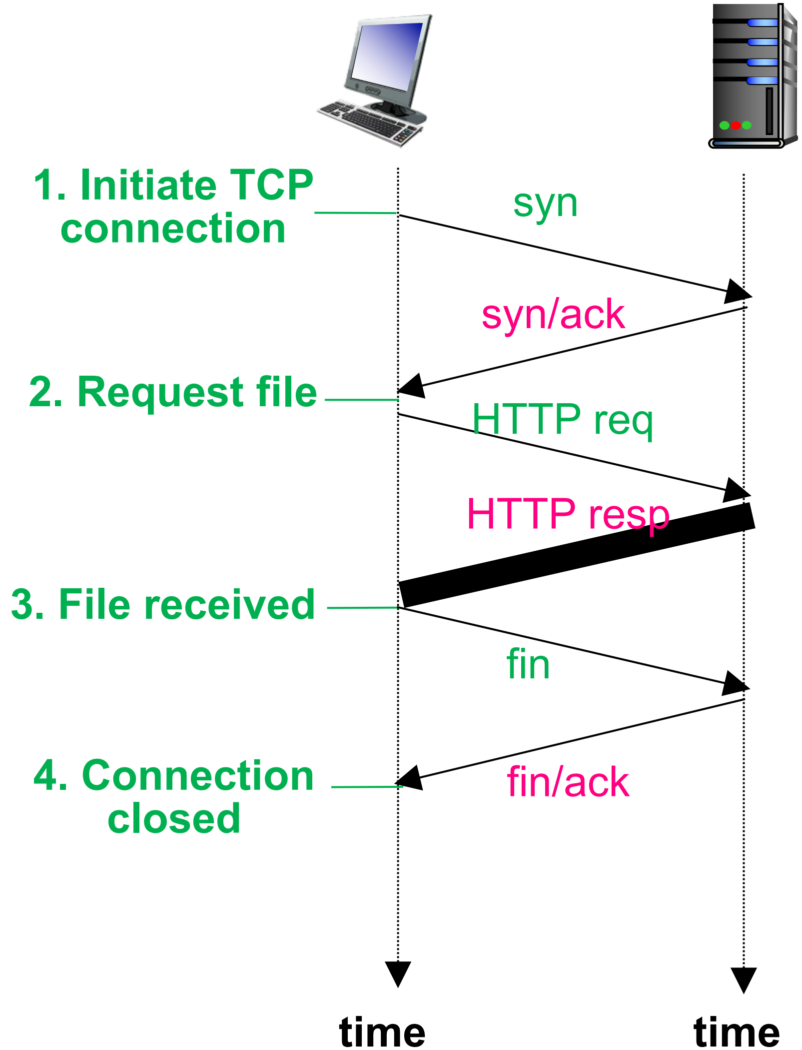
- client
  - browser that requests, receives, (using HTTP protocol) and "displays" Web objects
- server
  - Web server sends (using HTTP protocol) objects in response to requests



HTTP overview

When you click on a link

1. client initiates TCP connection
   – creates socket to server on port 80
2. server accepts TCP connection from client
3. HTTP messages exchanged between browser (HTTP client) and Web server (HTTP server)
4. TCP connection closed



Two types of HTTP messages

– request, response

HTTP is a stateless protocol

Stateless

- server maintains no information about past client requests

Why stateless?

- stateful protocols are complex
  - storage
    - state must be maintained for potentially many clients
  - server/client crashes
    - views of state may be inconsistent, must be reconciled
  - workaround: cookies

Format of a webpage

Web page consists of objects

- object can be HTML file, JPEG image, Java applet, audio file,…
- typically includes base HTML-file and several referenced objects

1. index.html

2. pic.jpg
3. HWK.pdf

All 3 objects must be requested from server in order to fully load webpage

Each object is addressable by URL, e.g.,

```
www.someschool.edu/someDept/pic.jpg
```

host name (www.someschool.edu) — **path** (someDept) — **object** (pic.jpg)

Q: How do we download multiple objects using HTTP?