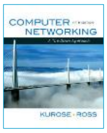
CSC358 *Intro. to Computer Networks*

Lecture 2: *layered architecture/models, application layer, Web and HTTP*

Amir H. Chinaei, Winter 2016

ahchinaei@cs.toronto.edu
http://www.cs.toronto.edu/~ahchinaei/



Many slides are (inspired/adapted) from the above source

Office Hours: T 17:00–18:00 R 9:00–10:00 BA4222

TA Office Hours: W 16:00-17:00 BA3201 R 10:00-11:00 BA7172
csc358ta@cdf.toronto.edu
http://www.cs.toronto.edu/~ahchinaei/teaching/2016jan/csc358/

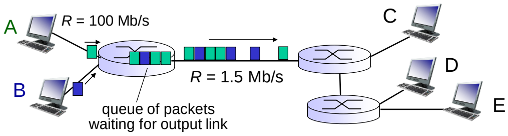
Key terms

- packet ~ chunk of data
- internet, protocol, network edge, access net, physical media, network core
- host ~ end system ~ (computing) device/machine/terminal ~ server (or client) ~ sender/transmitter ~ receiver
- router ~ (packet) switch ~ sender/transmitter ~ receiver
- packet/circuit switching
- (wired, wireless) link
- link capacity ~ link bandwidth ~ transmission rate
- propagation rate
- performance: loss, delay, throughput

Review

- Internet
  - "nuts and bolts" view
  - "service" view
- Protocol
  - e.g communication rules
- Transmission delay $\frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$
- Network core: routing, forwarding
- Circuit vs packet switching
  - dedicated vs sharing resources
  - e.g., traditional vs contemporary telephone networks

Packet switching: queueing delay, loss



queuing and loss:

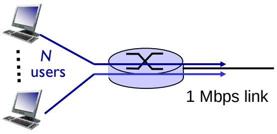
- If arrival rate (in bits) to link exceeds transmission rate of link for a period of time:
  - packets will queue, wait to be transmitted on link
  - packets can be dropped (lost) if memory (buffer) fills up

Packet switching versus circuit switching

packet switching allows more users to use network!

example:

- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time



- *circuit-switching:*
  - 10 users
- *packet-switching:*
  - more than 10
  - with 11 users, probability that all active at same time is 0.1^{11}
  - with e.g. 35 users, probability that 11 active at same time is less than 0.0004

Packet switching versus circuit switching

is packet switching a "slam dunk winner?"

- great for bursty data
  - resource sharing
  - simpler, no call setup
- excessive congestion possible: packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- *Q:* How to provide circuit-like behavior?
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem

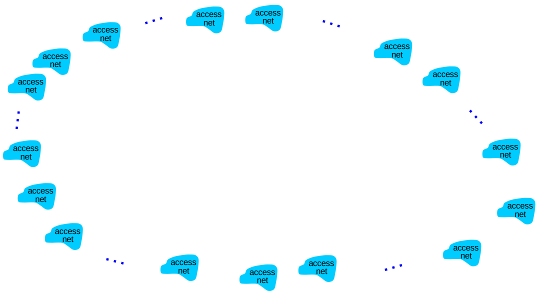
Q: human analogies of reserved resources (circuit-switching) versus on-demand allocation (packet-switching)?

Internet structure: network of networks

- End systems connect to Internet via access ISPs (Internet Service Providers)
  - Residential, company, and university ISPs
- Access ISPs in turn must be interconnected
  - So that any two hosts can send packets to each other
- Resulting network of networks is very complex
  - Evolution was driven by economics and national policies
- Let' s take a stepwise approach to describe current Internet structure

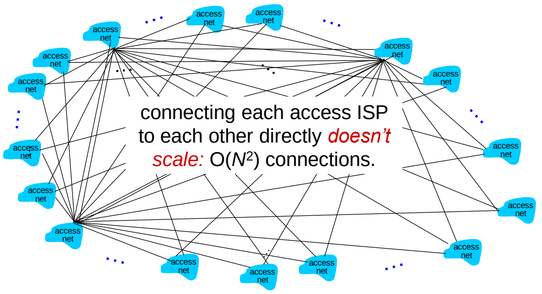
Internet structure: network of networks

Question: given *millions* of access ISPs, how to connect them together?



Internet structure: network of networks

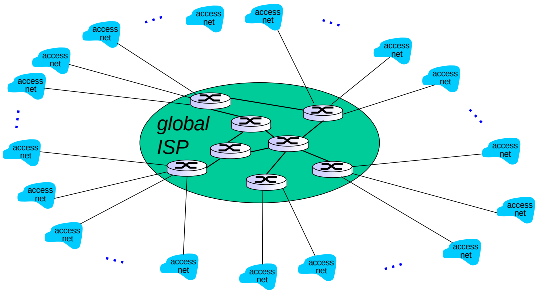
Option: connect each access ISP to every other access ISP?



connecting each access ISP to each other directly *doesn't scale:* O(*N*^{2}) connections.

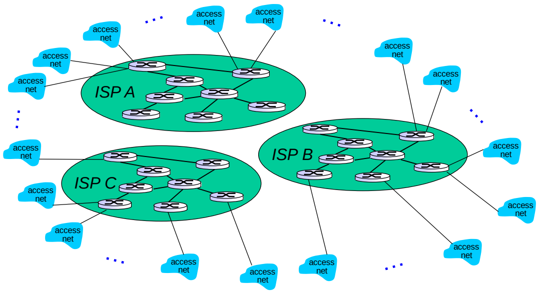
Internet structure: network of networks

Option: connect each access ISP to a global transit ISP? Customer and provider ISPs have economic agreement.



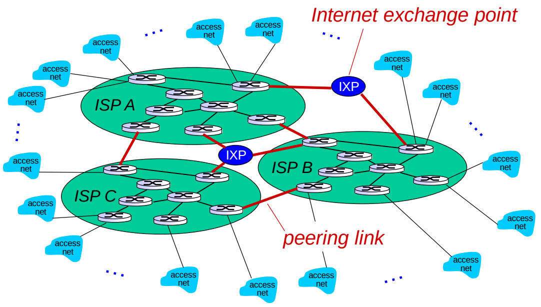
Internet structure: network of networks

But if one global ISP is viable business, there will be competitors ….



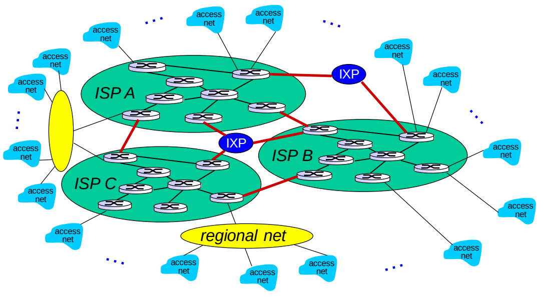
Internet structure: network of networks

But if one global ISP is viable business, there will be competitors …. which must be interconnected



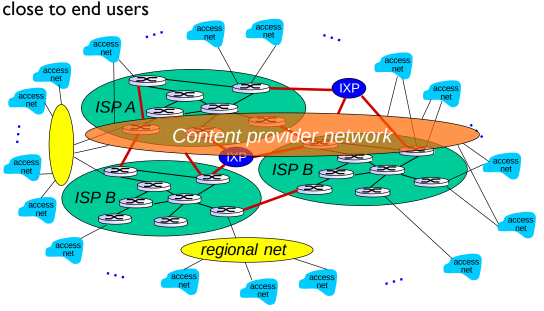
Internet structure: network of networks

… and regional networks may arise to connect access nets to ISPS

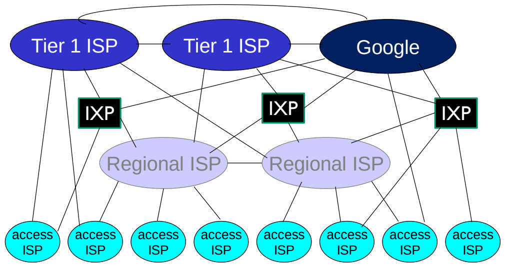


Internet structure: network of networks

… and content provider networks (e.g., Google, Microsoft, Akamai) may run their own network, to bring services, content close to end users



Internet structure: network of networks



- at center: small # of well-connected large networks
  - "tier-1" commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT), national & international coverage
  - content provider network (e.g, Google): private network that connects it data centers to Internet, often bypassing tier-1, regional ISPs

Continue on Chapter 1

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

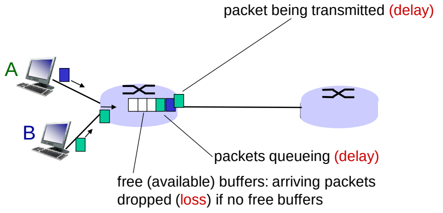
1.5 protocol layers, service models

1.6 networks under attack: security

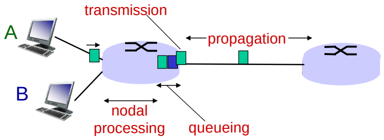
How do delay and loss occur?

packets *queue* in router buffers

- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn


packet being transmitted (delay)
A
B
packets queueing (delay)
free (available) buffers: arriving packets dropped (loss) if no free buffers

Four sources of packet delay



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

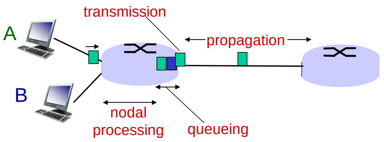
d_{proc}: nodal processing

- check bit errors
- routing: determine output link
- typically < μsec

d_{queue}: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router, μsec-msec

Four sources of packet delay

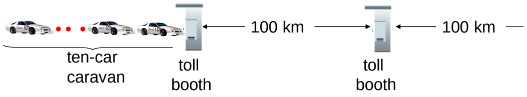


$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

d_{trans}: transmission delay:
- *forwarding*
- *L*: packet length (bits)
- *R*: link *bandwidth (bps)*
- $d_{trans} = L/R$
- µsec-msec

d_{prop}: propagation delay:
- *d*: length of physical link
- *s*: propagation speed in medium (~2-3x10^8 m/sec)
- $d_{prop} = d/s$
- in WANs: msec

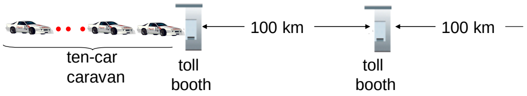
Caravan analogy



- cars "propagate" at 50 km/hr
- toll booth takes 15 sec to service car (bit transmission time).
- car~bit; caravan~packet; highway~medium, no queueing delay, no processing delay
- *Q*: How many minutes until caravan is lined up before 2nd tollbooth?

- time to "push" entire caravan through toll booth onto highway =
- time for caravan to propagate from 1st to 2nd tollbooth:
- *A: ?*

Caravan analogy (more)



- suppose cars now "propagate" at 1000 km/hr
- and suppose toll booth now takes one min to service a car
- *Q*: Will cars arrive to 2nd booth before all cars serviced at first booth?
  - *A: ?*

More on delays

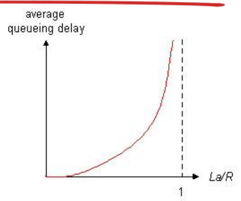
$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

- Which delay(s) is(are) negligible in some cases, e.g. in geographically smaller networks, or with having better hardware?

- Which delay(s) is(are) interesting? Why?
  - d_{queue}
  - different packets may be treated differently!
  - depends on statistical measures, such as average of delay, variance of delay, the probability that the delay exceeds a value

Queueing delay and traffic intensity

- *L*: packet length (bits)
- *R*: link bandwidth (bps)
- *a*: average packet arrival rate
- Traffic intensity=*La/R*



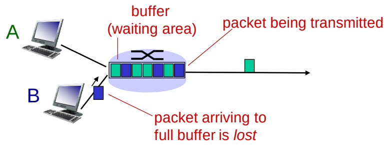
- *La/R*~0 : avg. queueing delay small
- *La/R*→1 : avg. queueing delay large
- *La/R*>1 : more "work" arriving than can be serviced, average delay infinite!
- Packets drop/loss

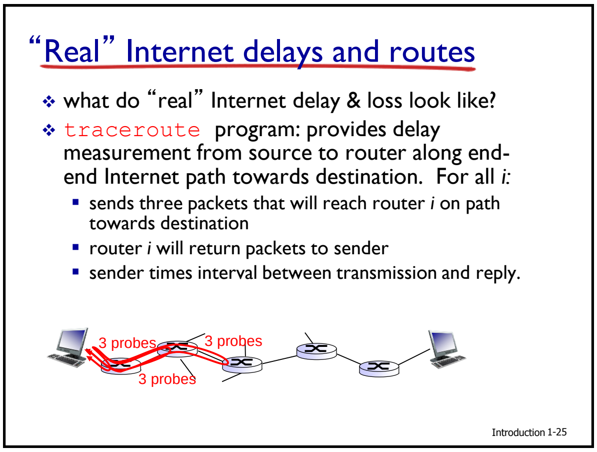
La/R ~ 0

La/R → 1

Packet loss

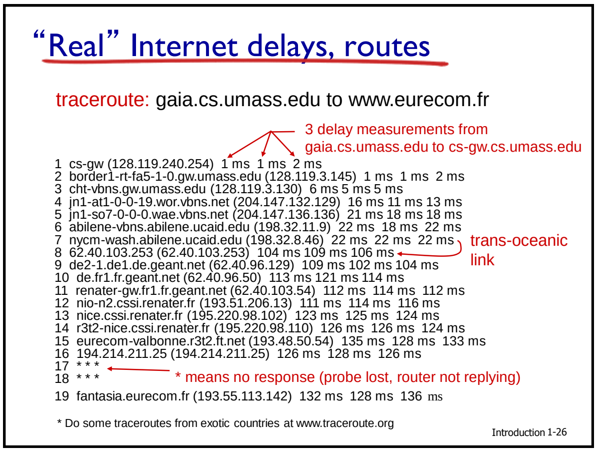
- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source host, or not at all





"Real" Internet delays and routes

- what do "real" Internet delay & loss look like?
- `traceroute` program: provides delay measurement from source to router along end-end Internet path towards destination. For all *i*:
  - sends three packets that will reach router *i* on path towards destination
  - router *i* will return packets to sender
  - sender times interval between transmission and reply.



"Real" Internet delays, routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

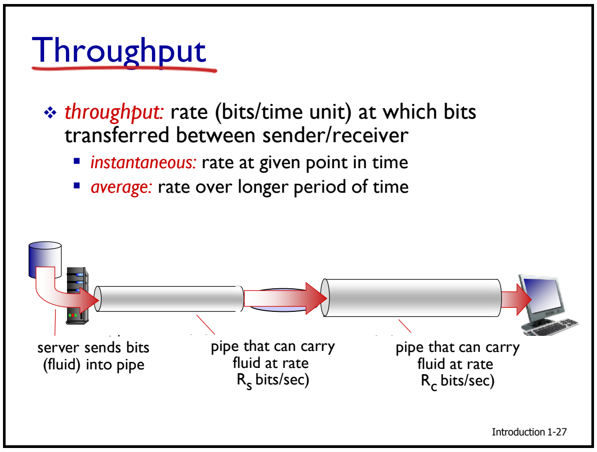
3 delay measurements from gaia.cs.umass.edu to cs-gw.cs.umass.edu

```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 ms 18 ms
6  abilene-vbns.abilene.ucaid.edu (198.32.11.9)  22 ms  18 ms  22 ms
7  nycm-wash.abilene.ucaid.edu (198.32.8.46)  22 ms  22 ms  22 ms
8  62.40.103.253 (62.40.103.253)  104 ms 109 ms 106 ms
9  de2-1.de1.de.geant.net (62.40.96.129)  109 ms 102 ms 104 ms
10  de.fr1.fr.geant.net (62.40.96.50)  113 ms 121 ms 114 ms
11  renater-gw.fr1.fr.geant.net (62.40.103.54)  112 ms  114 ms  112 ms
12  nio-n2.cssi.renater.fr (193.51.206.13)  111 ms  114 ms  116 ms
13  nice.cssi.renater.fr (195.220.98.102)  123 ms  125 ms  124 ms
14  r3t2-nice.cssi.renater.fr (195.220.98.110)  126 ms  126 ms  124 ms
15  eurecom-valbonne.r3t2.ft.net (193.48.50.54)  135 ms  128 ms  133 ms
16  194.214.211.25 (194.214.211.25)  126 ms  128 ms  126 ms
17  * * *
18  * * *
19  fantasia.eurecom.fr (193.55.113.142)  132 ms  128 ms  136 ms
```

trans-oceanic link

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

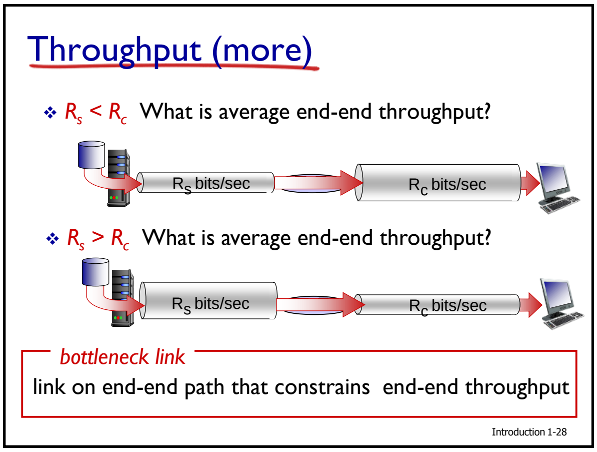
* Do some traceroutes from exotic countries at www.traceroute.org



Throughput

- *throughput:* rate (bits/time unit) at which bits transferred between sender/receiver
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time

server sends bits (fluid) into pipe
pipe that can carry fluid at rate R_s bits/sec)
pipe that can carry fluid at rate R_c bits/sec)



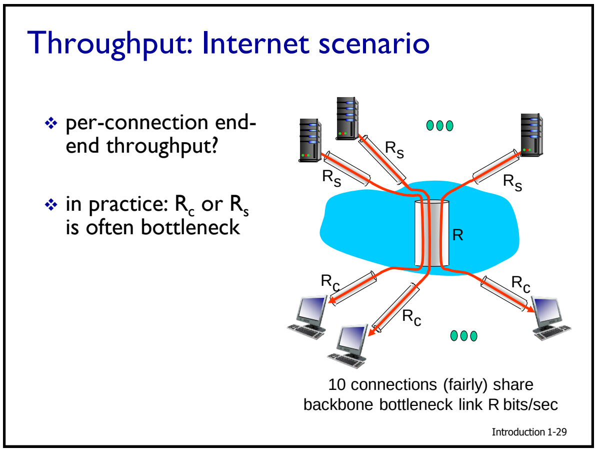
Throughput (more)

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

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

bottleneck link

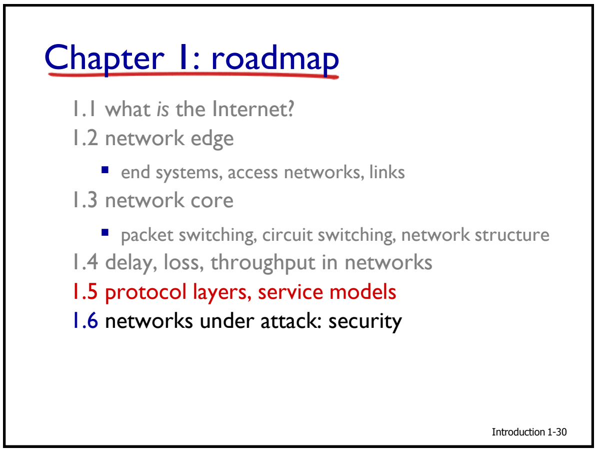
link on end-end path that constrains end-end throughput



Throughput: Internet scenario

- per-connection end-end throughput?
- in practice: R_c or R_s is often bottleneck

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



Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

1.6 networks under attack: security

Protocol "layers"

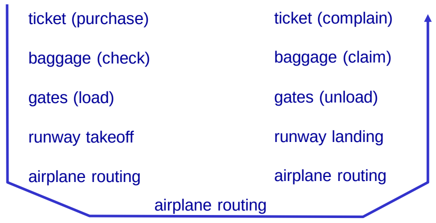
Networks are complex, with many "pieces":

- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:
is there any hope of *organizing* structure of network?

…. or at least our discussion of networks?

Organization of air travel



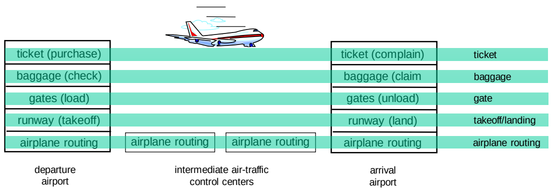
ticket (purchase)
baggage (check)
gates (load)
runway takeoff
airplane routing

airplane routing

ticket (complain)
baggage (claim)
gates (unload)
runway landing
airplane routing

- a series of steps

Layering of airline functionality



ticket (purchase)			ticket (complain)	ticket
baggage (check)			baggage (claim	baggage
gates (load)			gates (unload)	gate
runway (takeoff)			runway (land)	takeoff/landing
airplane routing	airplane routing	airplane routing	airplane routing	airplane routing
departure airport	intermediate air-traffic control centers		arrival airport	

layers: each layer implements a service

- via its own internal-layer actions
- relying on services provided by layer below

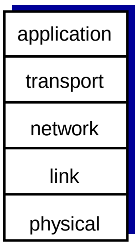
Why layering?

dealing with complex systems:

- explicit structure allows identification, relationship of complex system's pieces
  - layered *reference model* for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system
- layering considered harmful?

Internet protocol stack

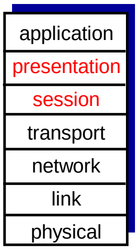
- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* bits "on the wire"



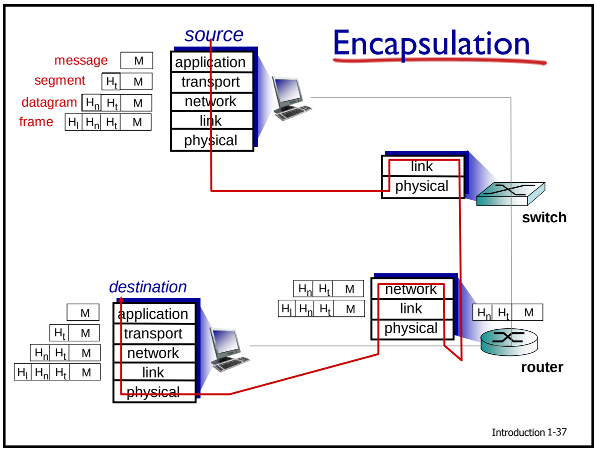
application
transport
network
link
physical

ISO/OSI reference model

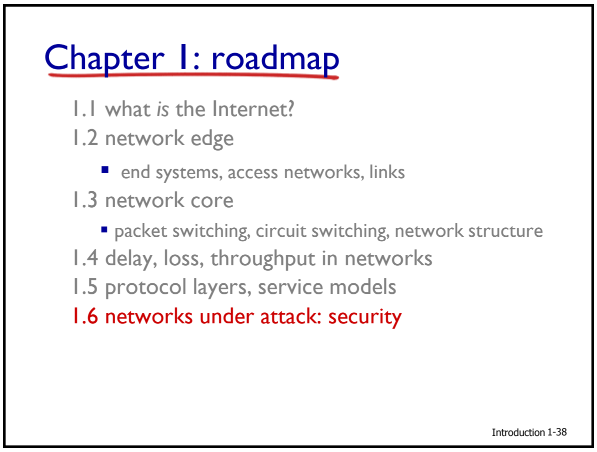
- *presentation:* allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application
  - needed?



application
presentation
session
transport
network
link
physical



Encapsulation



Chapter 1: roadmap

1.1 what *is* the Internet?

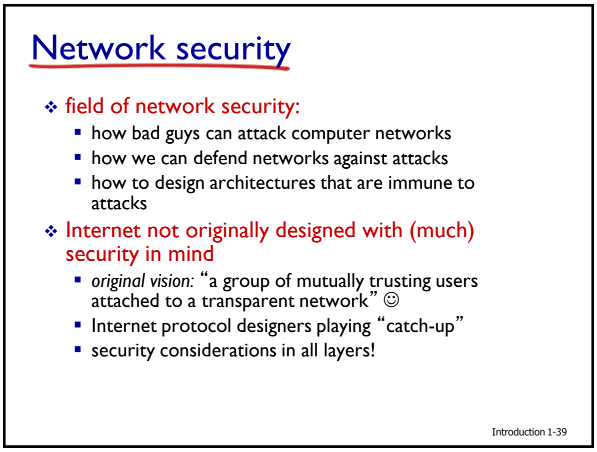
1.2 network edge
- end systems, access networks, links

1.3 network core
- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

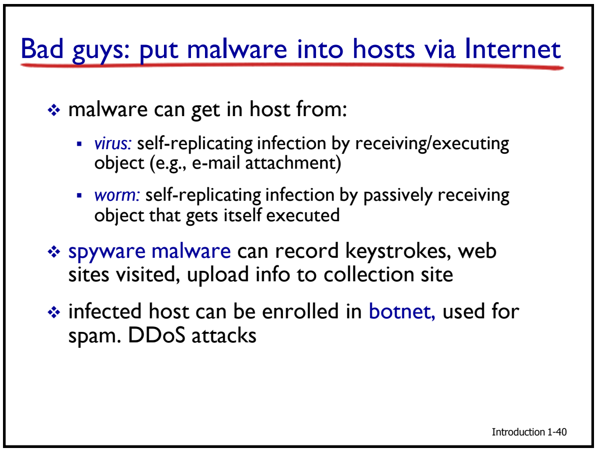
1.5 protocol layers, service models

1.6 networks under attack: security



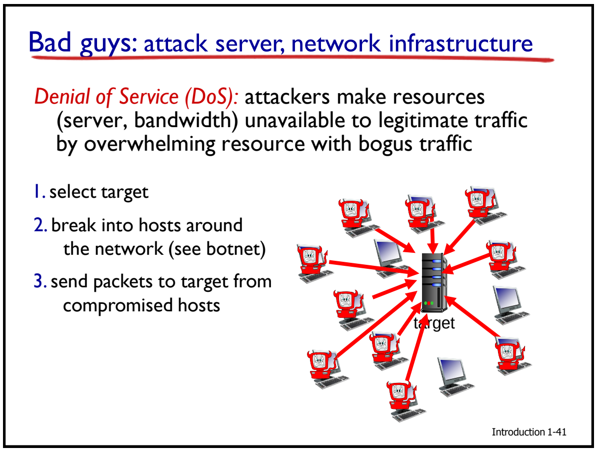
Network security

- field of network security:
  - how bad guys can attack computer networks
  - how we can defend networks against attacks
  - how to design architectures that are immune to attacks
- Internet not originally designed with (much) security in mind
  - *original vision:* "a group of mutually trusting users attached to a transparent network" ☺
  - Internet protocol designers playing "catch-up"
  - security considerations in all layers!



Bad guys: put malware into hosts via Internet

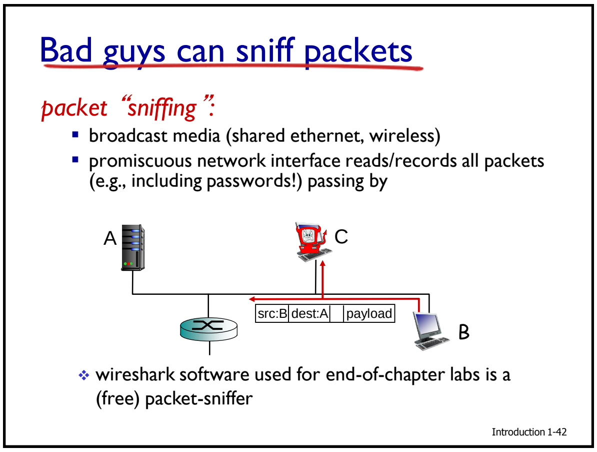
- malware can get in host from:
  - *virus:* self-replicating infection by receiving/executing object (e.g., e-mail attachment)
  - *worm:* self-replicating infection by passively receiving object that gets itself executed
- spyware malware can record keystrokes, web sites visited, upload info to collection site
- infected host can be enrolled in botnet, used for spam. DDoS attacks



Bad guys: attack server, network infrastructure

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts



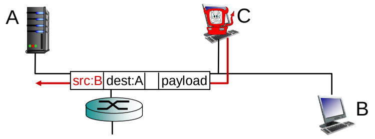
Bad guys can sniff packets

packet "sniffing":
- broadcast media (shared ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by

- wireshark software used for end-of-chapter labs is a (free) packet-sniffer

Bad guys can use fake addresses

IP spoofing: send packet with false source address



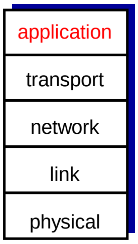
… lots more on security (throughout, Chapter 8)

Application layer

2.1 principles of network applications

- conceptual, implementation aspects of network application protocols
  - transport-layer service models
  - client-server paradigm
  - peer-to-peer paradigm

2.2 Web and HTTP



application

transport

network

link

physical

Some network apps

- e-mail
- web
- text messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video (YouTube, Hulu, Netflix)
- voice over IP (e.g., Skype)
- real-time video conferencing
- social networking
- search
- …
- …

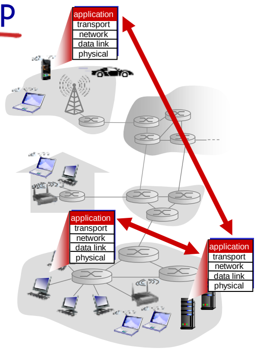
Creating a network app

write programs that:

- run on (different) *end systems*
- communicate over network
- e.g., web server software communicates with browser software

no need to write program for network-core devices

- network-core devices do not run user applications
- applications on end systems allows for rapid app development, propagation

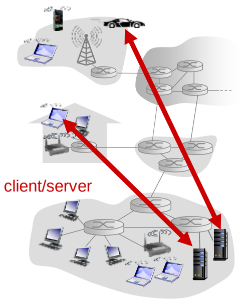


Application architectures

possible structure of applications:

- client-server
- peer-to-peer (P2P)

Client-server architecture



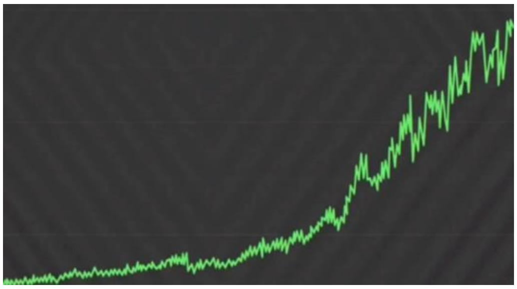
server:

- always-on host
- permanent IP address
- data centers for scaling

clients:

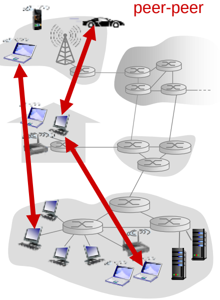
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

C/S: infrastructure intensive



P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
- complex management, not ISP friendly, security challenges, requires incentive design.



Processes communicating

process: program running within a host

- within same host, two processes communicate using inter-process communication (defined by OS)
- processes in different hosts communicate by exchanging messages

clients, servers

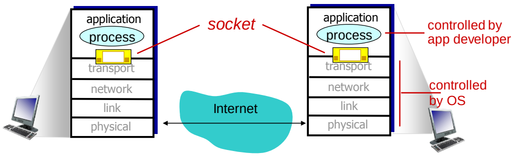
client process: process that initiates communication

server process: process that waits to be contacted

- applications with P2P architectures have client processes & server processes too

Socket: a software interface

- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process



Addressing processes

- to receive messages, process must have *identifier*
- host device has unique 32-bit IP address
- Q: does IP address of host on which process runs suffice for identifying the process?
  - A: ?
- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  - IP address: 128.119.245.12
  - port number: 80
- more shortly…

App-layer protocol defines

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

open protocols:

- defined in RFCs
- allows for interoperability
- e.g., HTTP, SMTP

proprietary protocols:

- e.g., Skype

What transport service does an app need?

reliable data transfer

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

throughput

- some apps (e.g., multimedia) require a minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

security

- encryption, data integrity, …

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

Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, or security
- *connection-oriented:* setup required between client and server processes

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup,

Q: why bother? Why is there a UDP?

Internet apps: application, 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

Securing TCP

TCP & UDP

- no encryption
- cleartext passwds sent into socket traverse Internet in cleartext

TLS

- provides encrypted TCP connection
- data integrity
- end-point authentication

TLS is at app layer

- Apps use TLS libraries, which "talk" to TCP

TLS socket API

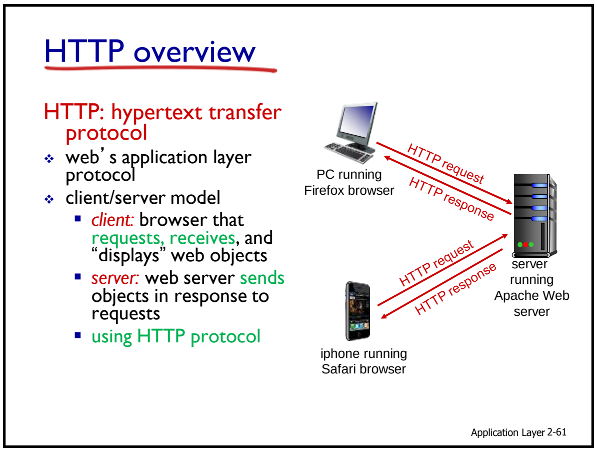
- cleartext passwds sent into socket traverse Internet encrypted
- Chapter 8

Web and HTTP

First, a review…

- *web page* consists of *objects*
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of *base HTML-file* which includes *several referenced objects*
- each object is addressable by a *URL,* e.g.,

$$\underbrace{\texttt{www.someschool.edu}}_{\text{host name}}\underbrace{\texttt{/someDept/pic.gif}}_{\text{path name}}$$



HTTP overview

HTTP: hypertext transfer protocol
- web' s application layer protocol
- client/server model
  - *client:* browser that requests, receives, and "displays" web objects
  - *server:* web server sends objects in response to requests
  - using HTTP protocol

PC running Firefox browser

HTTP request

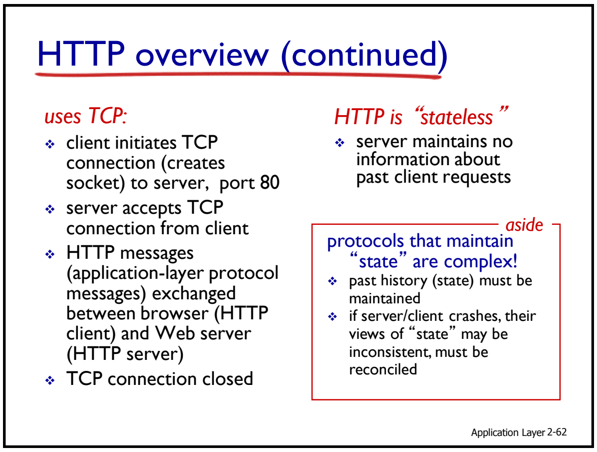
HTTP response

server running Apache Web server

HTTP request

HTTP response

iphone running Safari browser



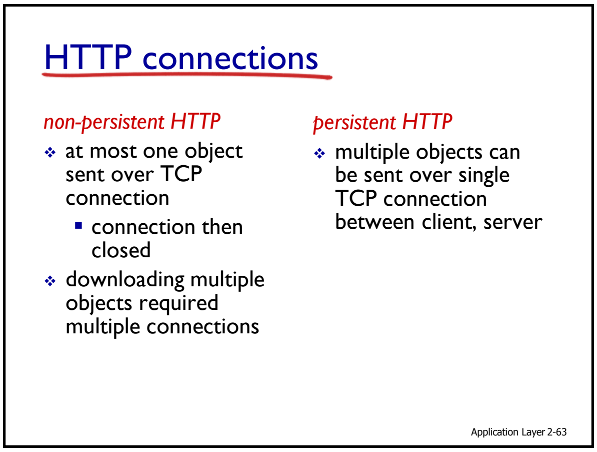
HTTP overview (continued)

uses TCP:
- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"
- server maintains no information about past client requests

aside

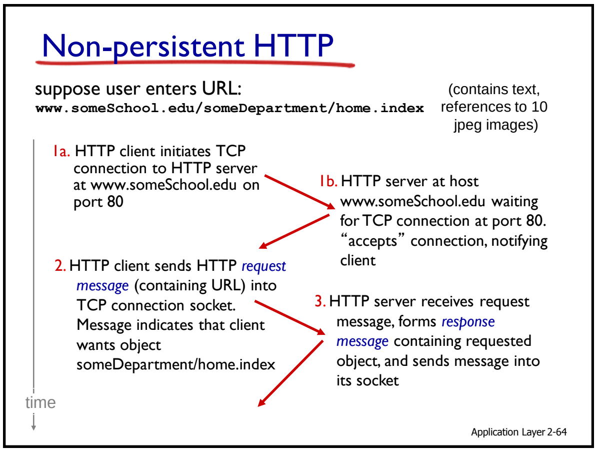
protocols that maintain "state" are complex!
- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled



HTTP connections

non-persistent HTTP
- at most one object sent over TCP connection
  - connection then closed
- downloading multiple objects required multiple connections

persistent HTTP
- multiple objects can be sent over single TCP connection between client, server



Non-persistent HTTP

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

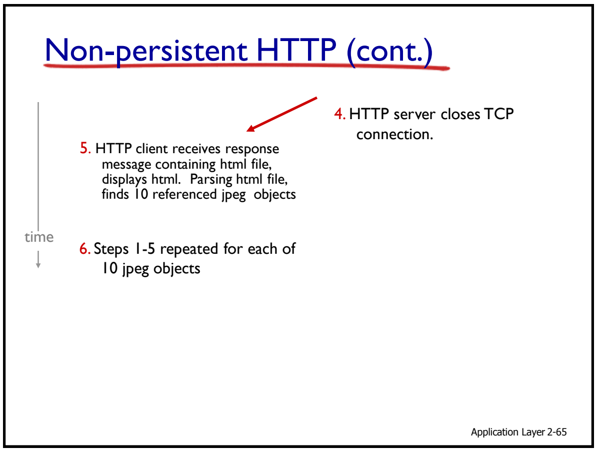
1a. HTTP client initiates TCP connection to HTTP server at www.someSchool.edu on port 80

1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

time



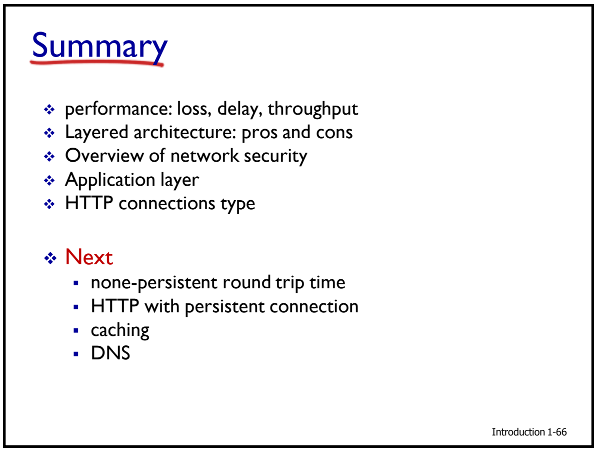
Non-persistent HTTP (cont.)

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

time



Summary

- performance: loss, delay, throughput
- Layered architecture: pros and cons
- Overview of network security
- Application layer
- HTTP connections type

- Next
  - none-persistent round trip time
  - HTTP with persistent connection
  - caching
  - DNS