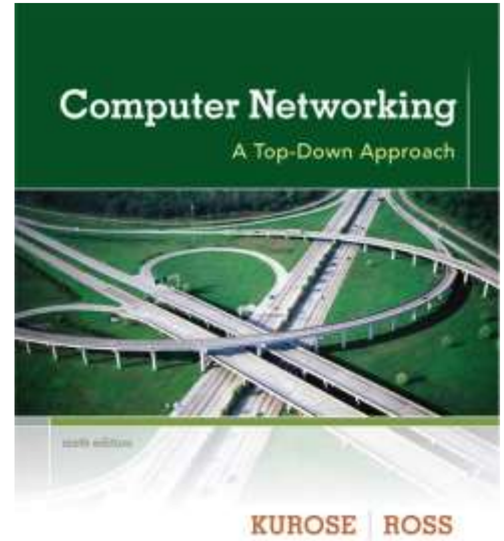
# Chapter 1
# Introduction

## Lecture 3




*Computer Networking: A Top Down Approach*
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

2021 __2020

# Chapter 1: roadmap

1.1 what *is* the Internet?
1.2 network edge
- end systems, access networks, links

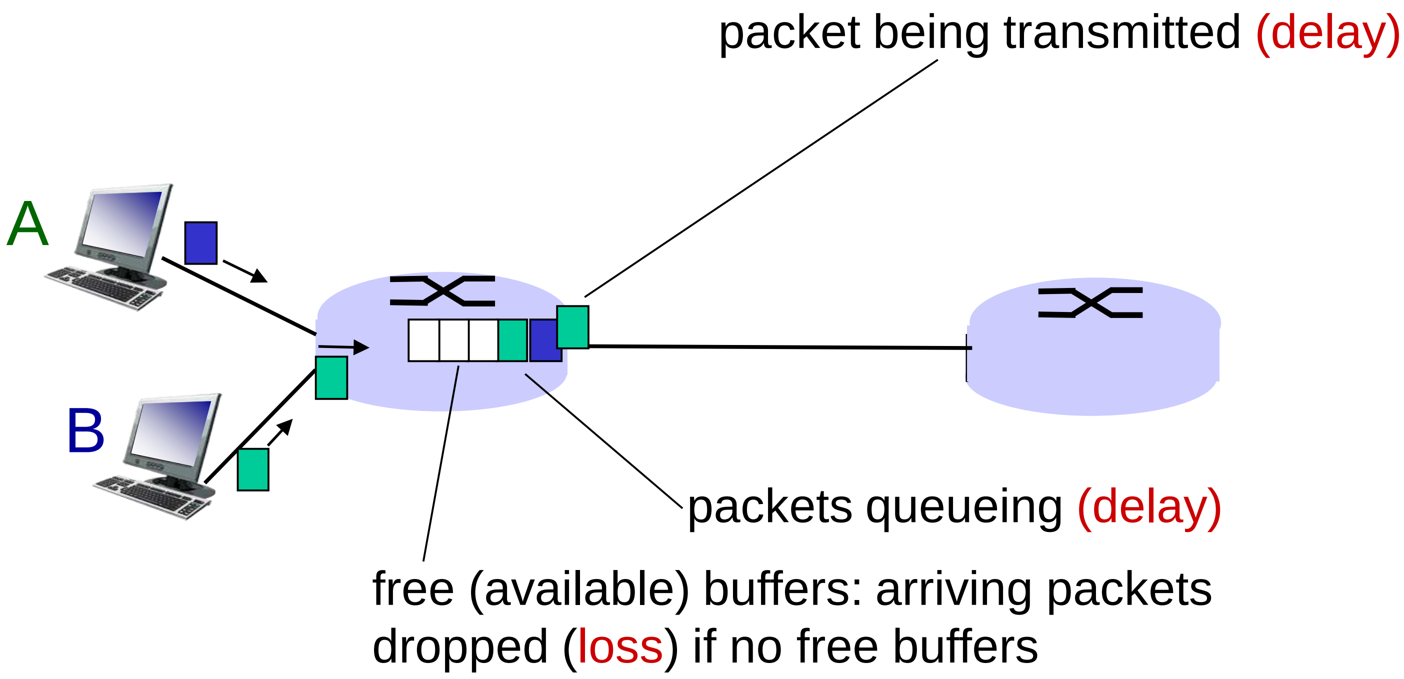
1.3 network core
- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks
1.5 protocol layers, service models
1.6 networks under attack: security
1.7 history

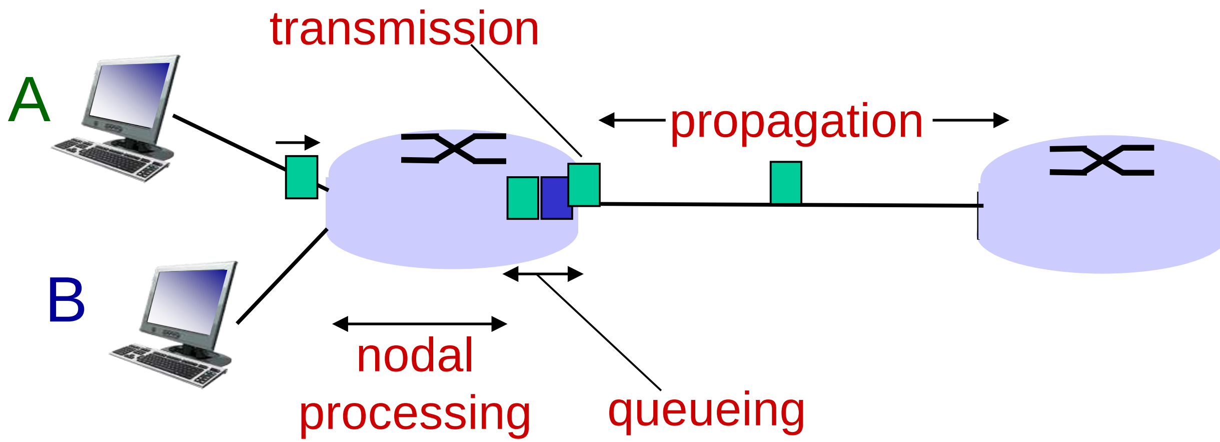
# How do loss and delay occur?

packets *queue* in router buffers

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



# Four sources of packet delay




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

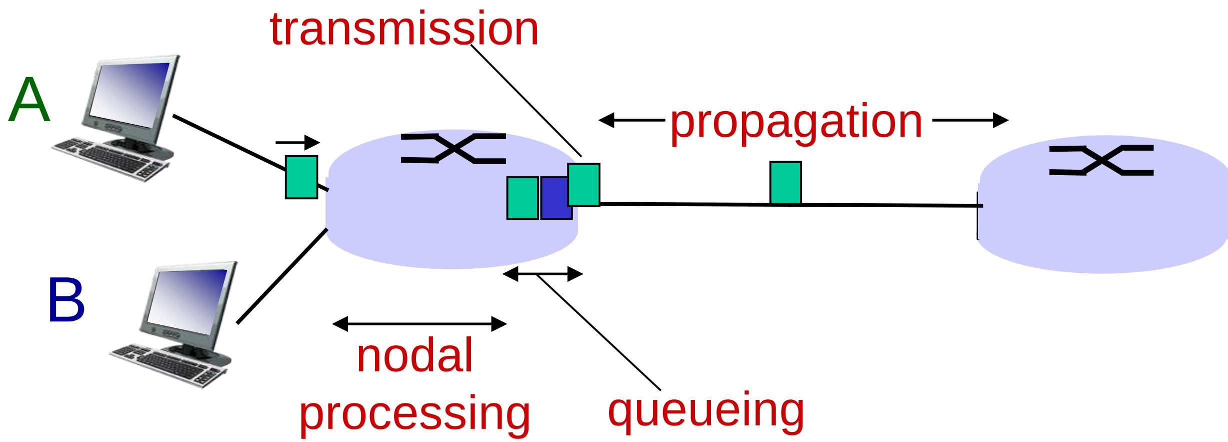
## $d_{proc}$: nodal processing

- check bit errors
- determine output link
- typically < msec

## $d_{queue}$: 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:

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

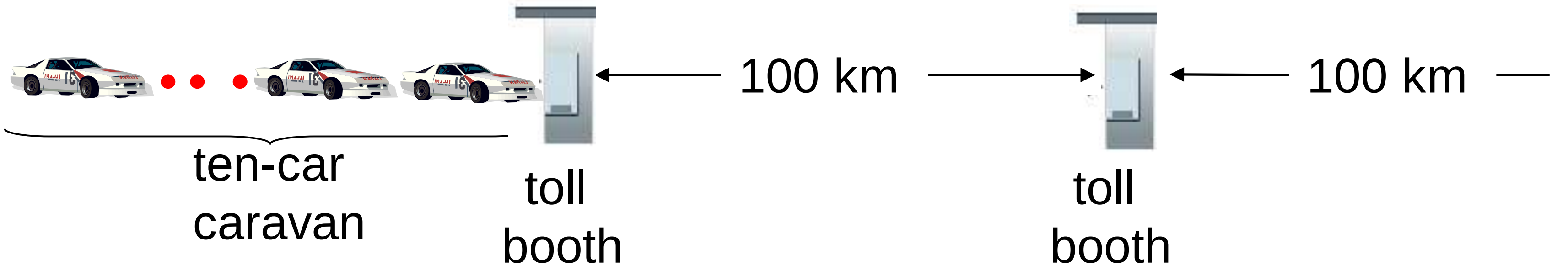
## $d_{prop}$: propagation delay:

- *d*: length of physical link
- *s*: propagation speed in medium (~$2 \times 10^8$ m/sec)
- $d_{prop} = d/s$

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

* Check out the Java applet for an interactive animation on trans vs. prop delay

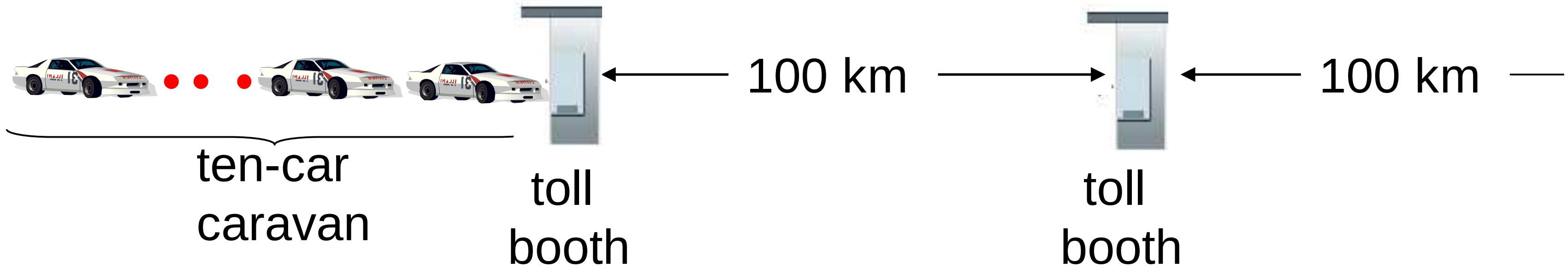
# Caravan analogy




- cars "propagate" at 100 km/hr
- toll booth takes 12 sec to service car (bit transmission time)
- car~bit; caravan ~ packet
- *Q:* How long until caravan is lined up before 2nd toll booth?

- time to "push" entire caravan through toll booth onto highway = 12*10 = 120 sec
- time for last car to propagate from 1st to 2nd toll both: 100km/(100km/hr)= 1 hr
- *A:* 62 minutes

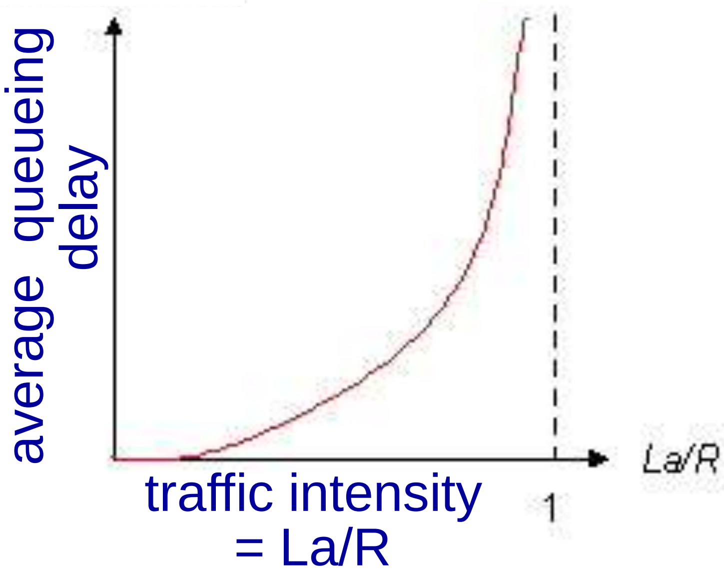
# Caravan analogy (more)




- 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: Yes!* after 7 min, 1st car arrives at second booth; three cars still at 1st booth.

# Queueing delay (revisited)

- *R:* link bandwidth (bps)
- *L:* packet length (bits)
- a: average packet arrival rate




- *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!

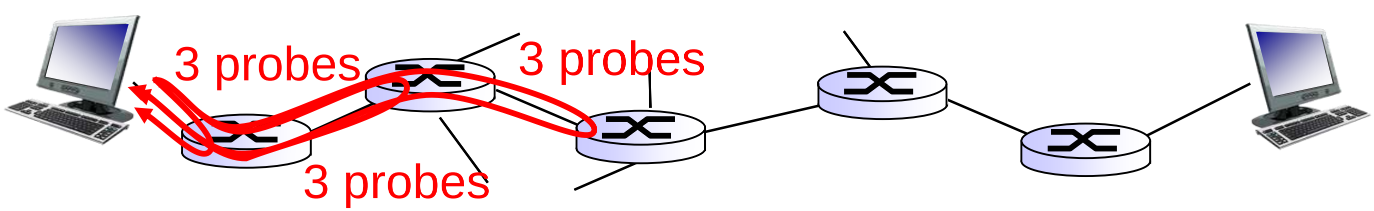
La/R ~ 0

La/R -> 1

* Check out the Java applet for an interactive animation on queuing and loss

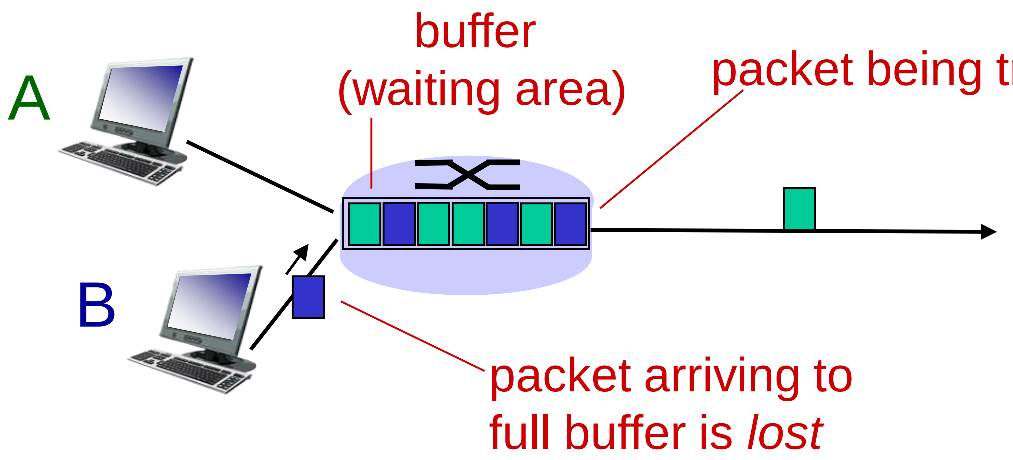
# "Real" Internet delays and routes

- 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.



# Packet loss

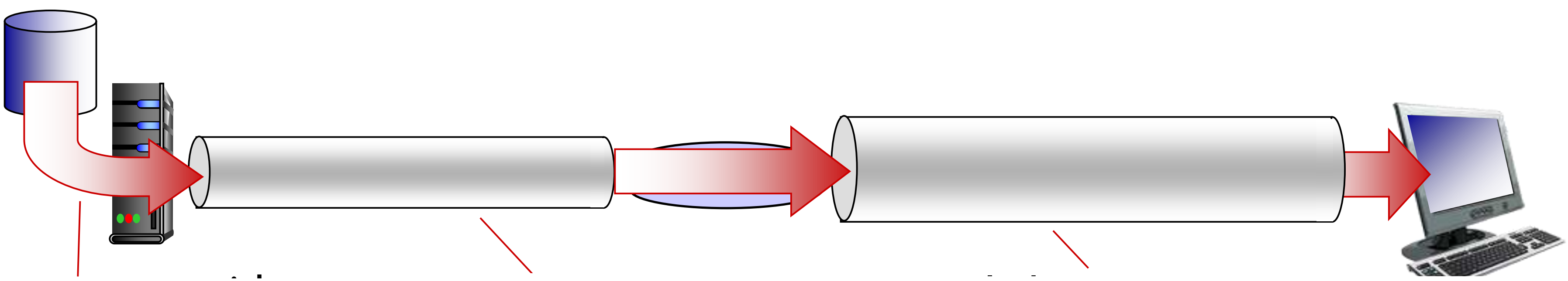
- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all




* Check out the Java applet for an interactive animation on queuing and loss

# 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



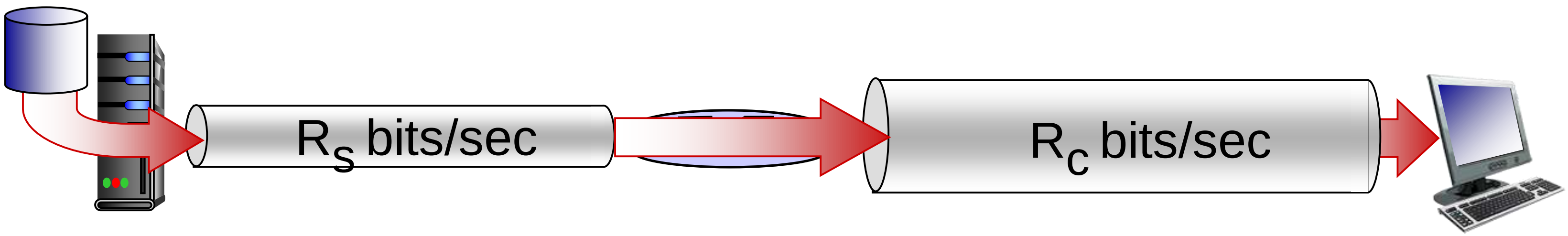
server sends bits (fluid) into pipe

pipe that can carry fluid at rate $R_s$ bits/sec)

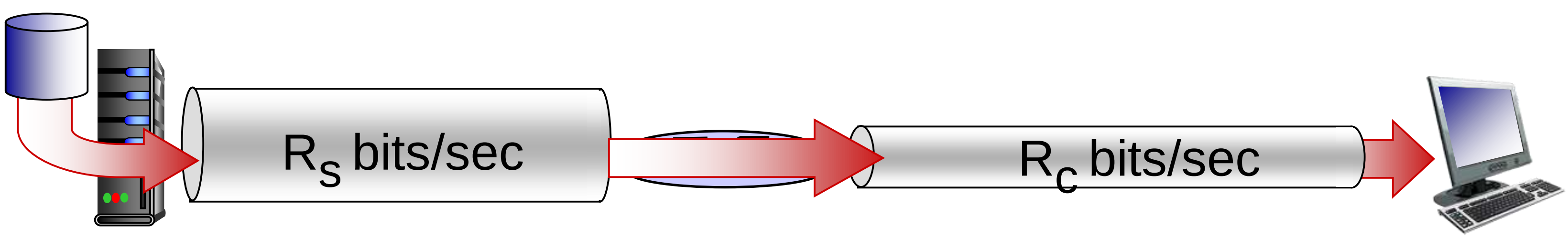
pipe that can carry fluid at rate $R_c$ bits/sec)

# Throughput (more)

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




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

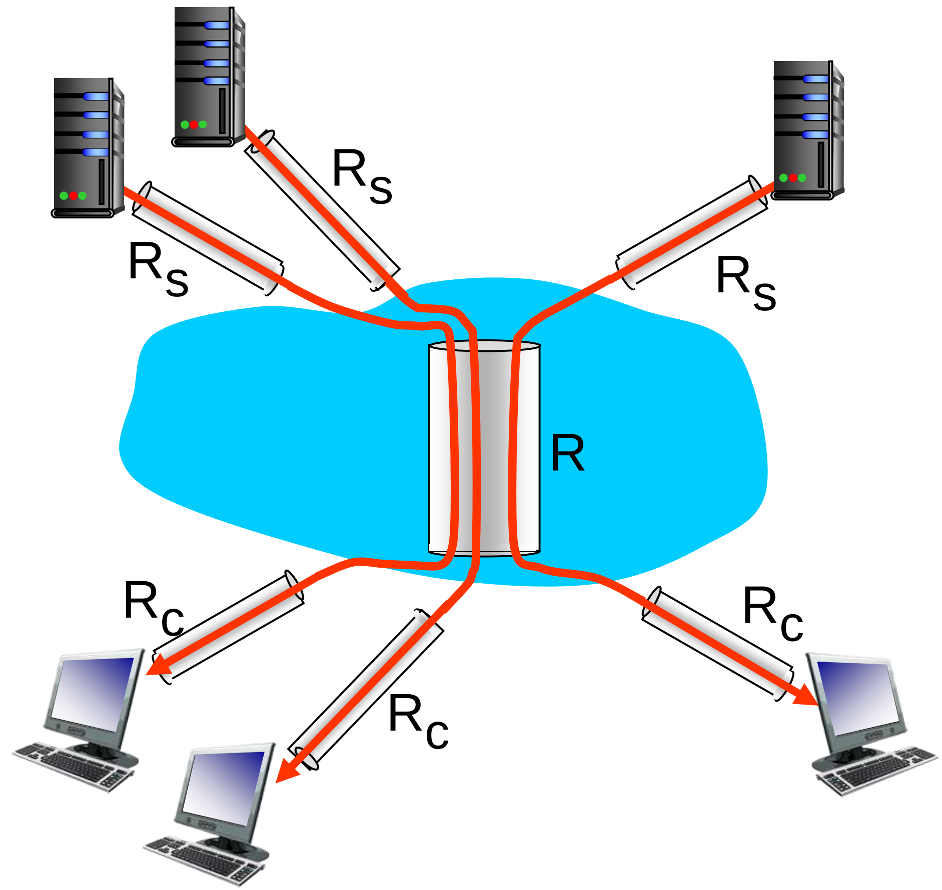



*bottleneck link*

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

# Throughput: Internet scenario

- per-connection end-end throughput: $\min(R_c, R_s, R/10)$
- 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"

From our discussion thus far, it is apparent that the Internet is an extremely complicated system.

*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?

•Before attempting to organize our thoughts on Internet architecture, let's look for a human analogy

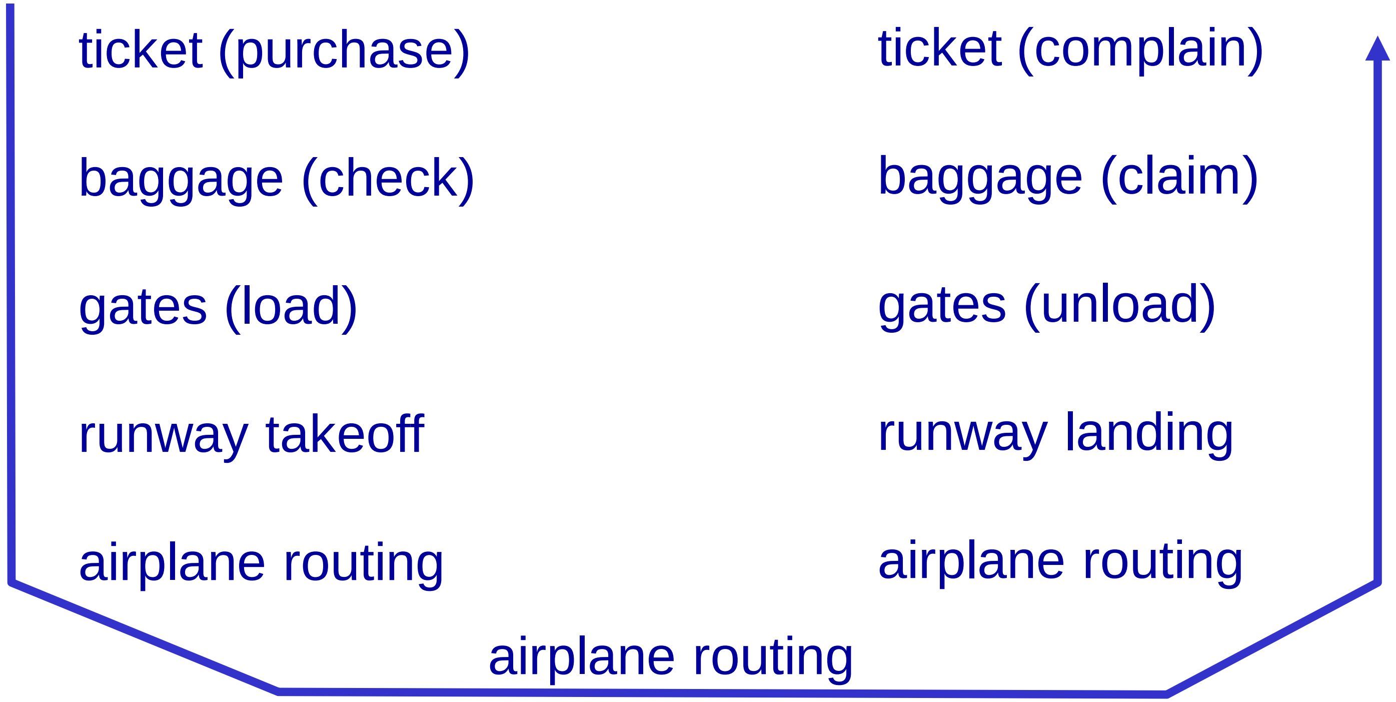
•Imagine if someone asked you to describe, for example, the airline system.

•How would you find the structure to describe this complex system that has ticketing agents, baggage checkers, gate personnel, pilots, airplanes, air traffic control, and a worldwide system for routing airplanes?

•

# Organization of air travel

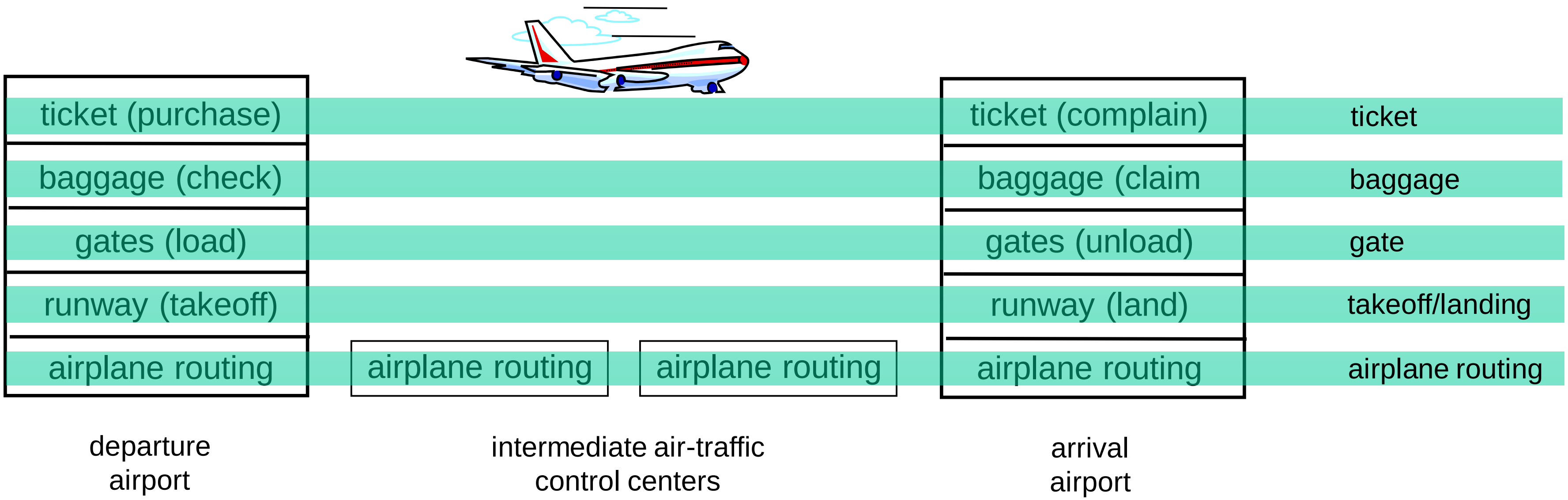
One way to describe this system might be to describe the series of actions you take (or others take for you) when you fly on an airline. This scenario is shown in this Figure:




- a series of steps

# Layering of airline functionality

we can look at the functionality in last Figure in a horizontal manner, as shown in this Figure:




*layers:* each layer implements a service

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

Ex. At the ticketing layer and below, airline-counter-to-airline-counter transfer of a person is accomplished .

Ex. At the baggage layer and below, baggage-check-to-baggage-claim transfer of a person and bags is accomplished.

# Why layering?

dealing with complex systems:

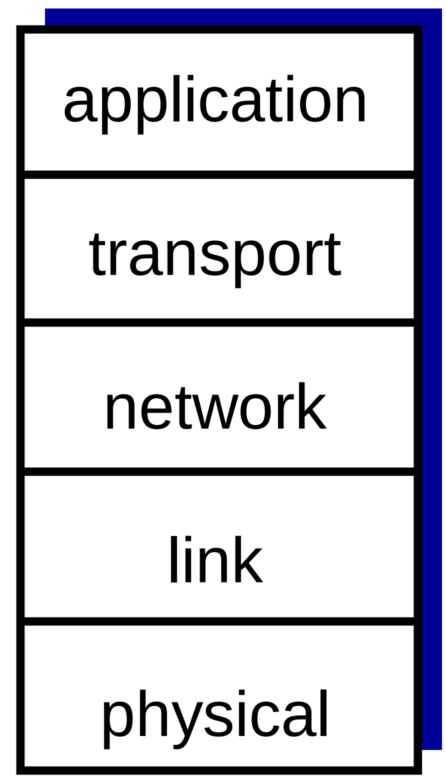
- explicit structure allows identification, relationship of complex system's pieces
  - layered *reference model* for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't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

To provide structure to the design of network protocols, network designers organize protocols—and the network hardware and software that implement the protocols in **layers**

# Internet protocol stack

- *application:* supporting network applications
  - FTP, SMTP, HTTP
  - We'll refer to this packet of information as a message
- *transport:* process-process data transfer
  - TCP, UDP
  - We'll refer to this packet of information as a segment
- *network:* routing of datagram's from source to destination
  - IP, routing protocols
  - We'll refer to this packet of information as a datagram
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.111 (WiFi), PPP
  - We'll refer to this packet of information as a frames
- *physical:* bits "on the wire"

  We'll refer to this packet of information as a frames



# Internet protocol stack

- each layer provides its service by (1) performing certain actions within that layer and by (2) using the services of the layer directly below it.

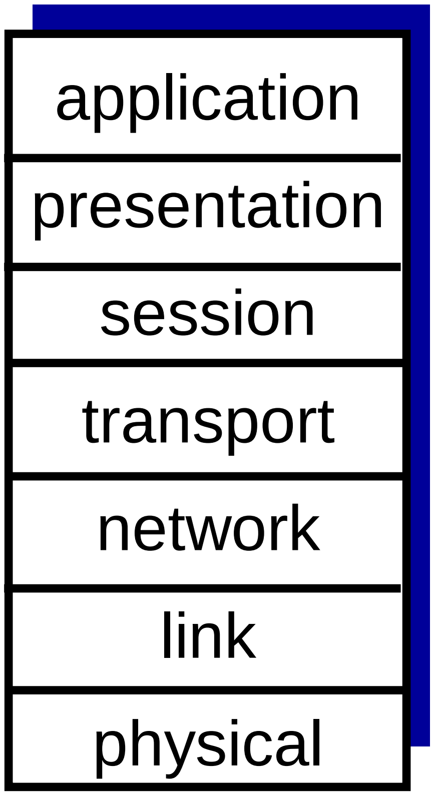
For example,
the services provided by layer *n* may include reliable delivery of messages from one edge of the network to the other. This might be implemented by using an unreliable edge-to-edge message delivery service of layer *n -1* , *and adding* layer *n* functionality to detect and retransmit lost messages.

# ISO/OSI reference model

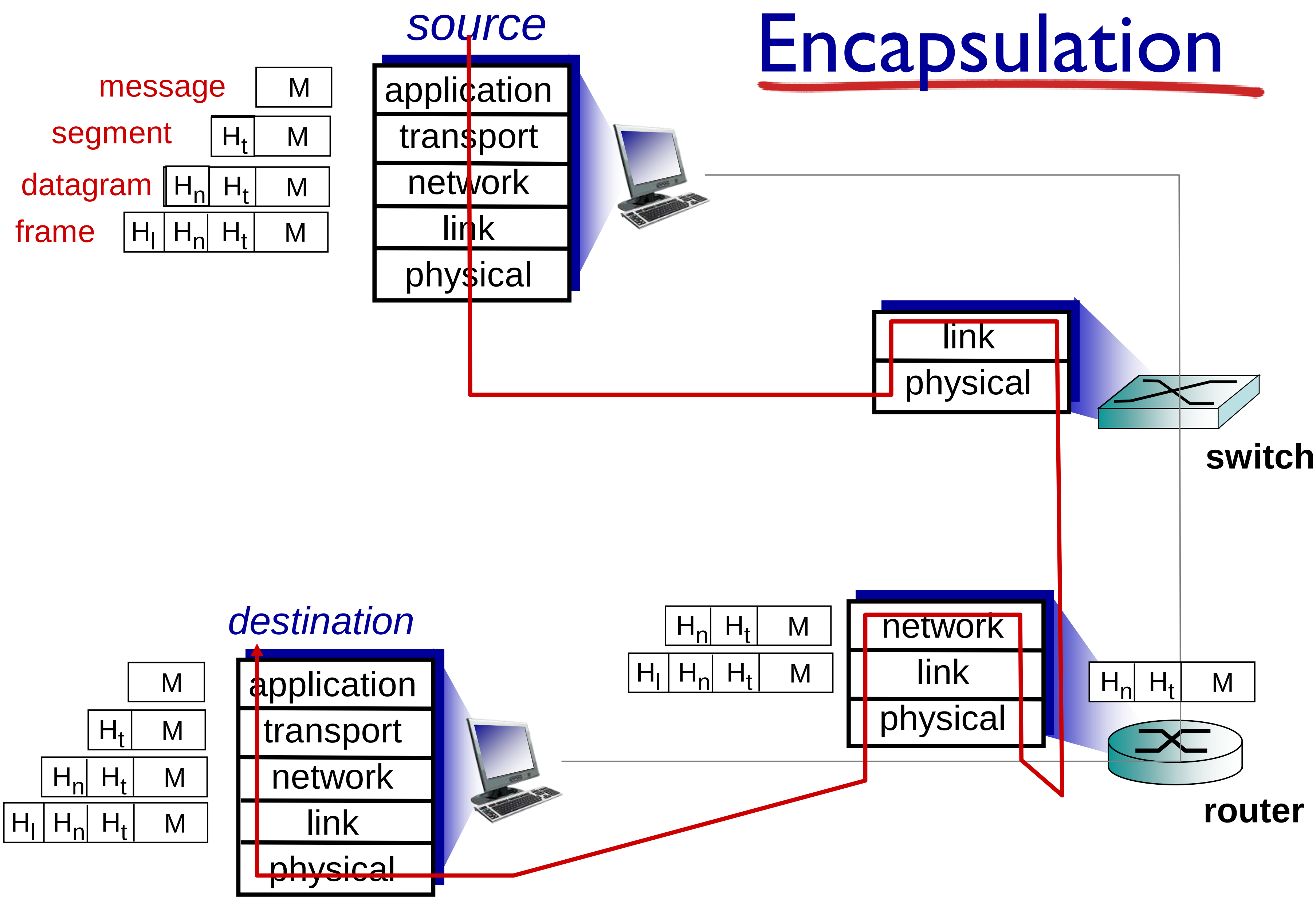
- the International Organization for Standardization (ISO) proposed that computer networks be organized around seven layers, called the Open Systems Interconnection (OSI)model

- The seven layers of the OSI reference model, shown in this Figure
- The functionality of five of these layers is roughly the same as their similarly named Internet counterparts

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



# Encapsulation



- Figure illustrates the important concept of **encapsulation. At the** sending host, an **application-layer message (M in Figure) is passed to the** transport layer.
- **the transport layer** takes the message and appends additional information (so-called transport-layer **header information, H*t***
- The transport layer then passes the segment to **the network layer**, which adds network-layer **header information H*n***
- at each layer, a packet has two types of fields: header fields and a **payload field. The payload is typically a packet from** the layer above.

# 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

1.7 history

# 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!

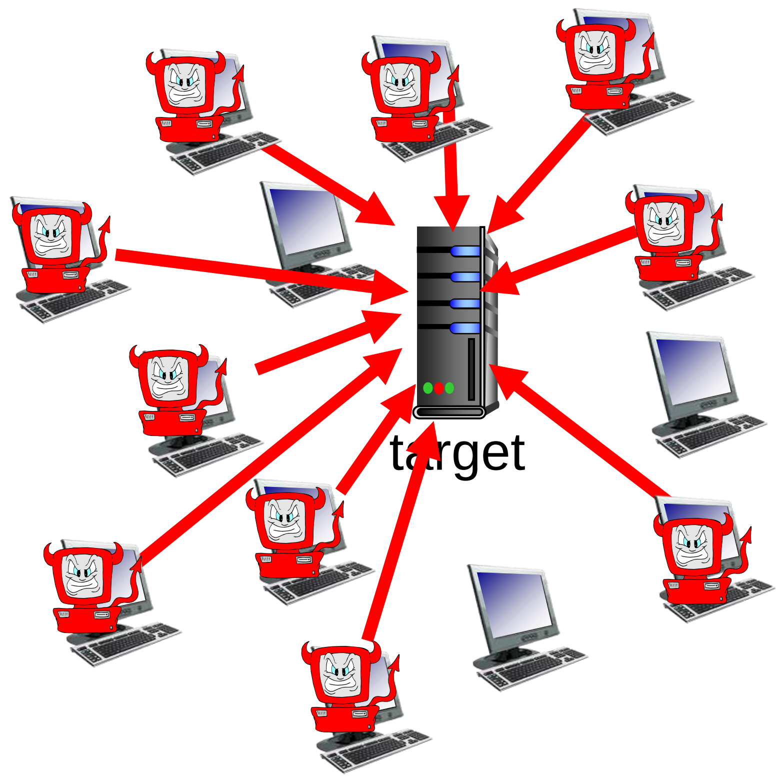
# Bad guys: put malware into hosts via Internet

- 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

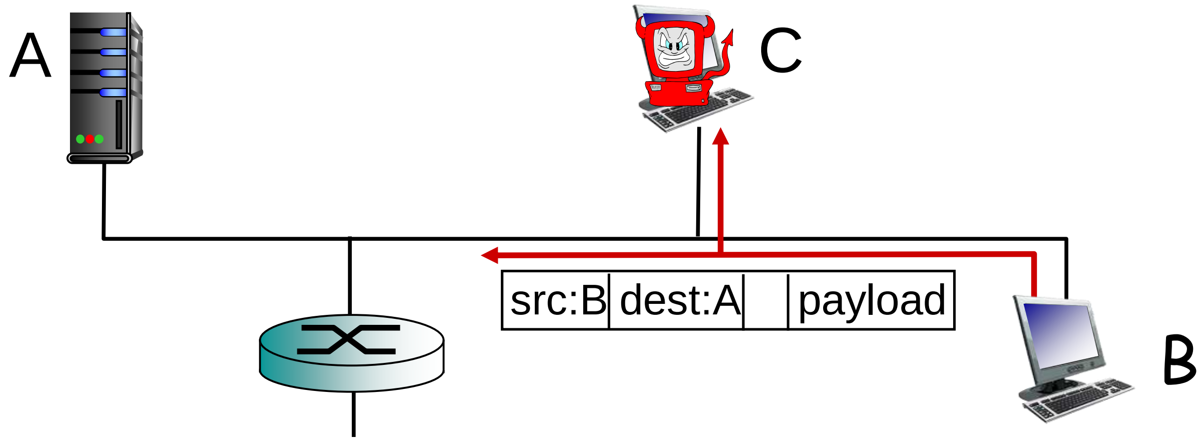
1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts



# Bad guys can sniff packets

*packet "sniffing":*

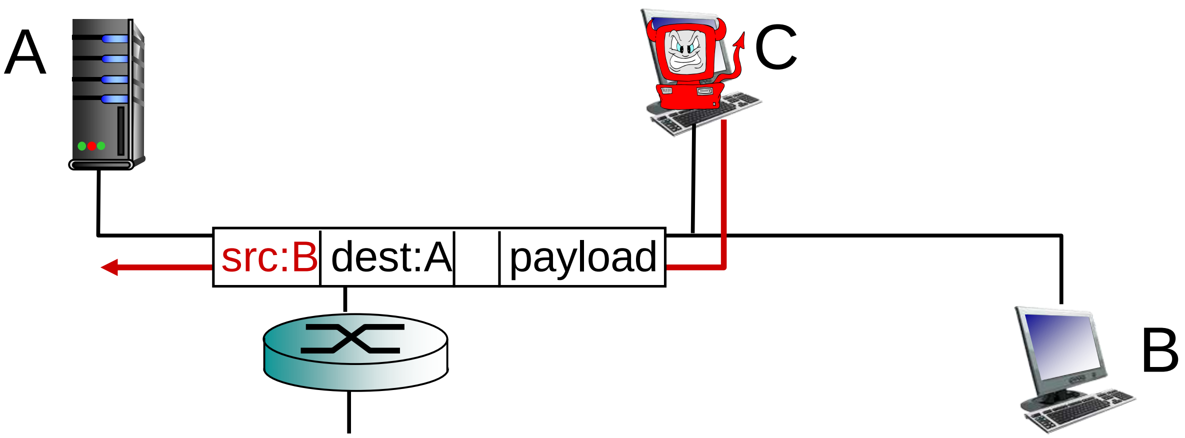
- broadcast media (shared ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by




- 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)*