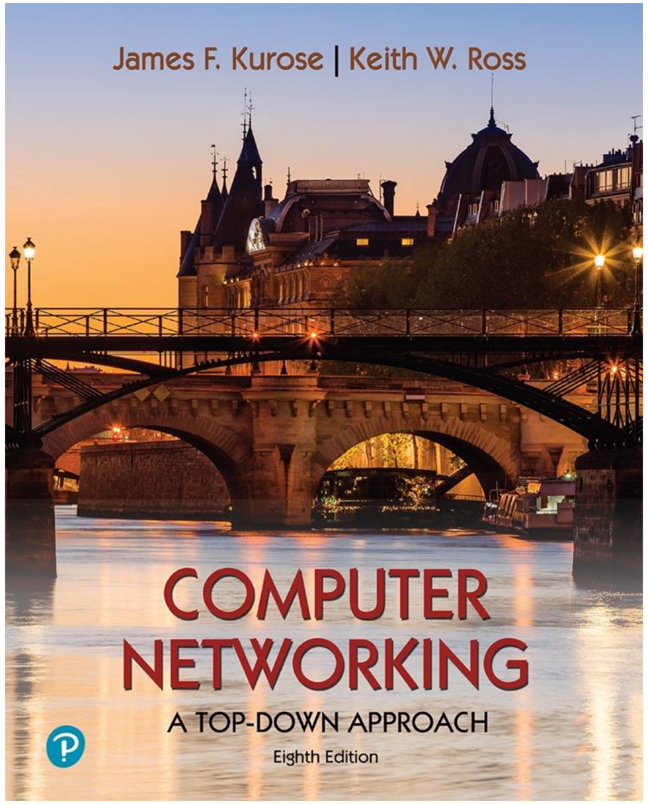
Mid-Term Overview

Yaxiong Xie

Department of Computer Science and Engineering
University at Buffalo, SUNY

Adapted from the slides of the book's authors



Computer Networking: A Top-Down Approach
8^{th} edition
Jim Kurose, Keith Ross
Pearson, 2020

Several Points

- I don’t have the mid-term exam question prepared yet
- Here, I will list all the ***topics*** that I think are important
  - If one topic I didn’t mention, then I won’t test it
  - It is about the topic, not the slides
    - If I didn’t mention one slides, but I do mention the topic, I probably will cover it
    - There are too many slides if I include every slides about that topic
- It will be fast, I won’t teach it again
- Ask questions, if you have any

Chapter 1: introduction

Chapter goal:

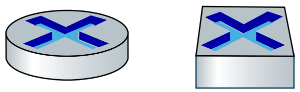
- Get "feel," "big picture," introduction to terminology
  - more depth, detail *later* in course

Overview/roadmap:

- What *is* the Internet? What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: packet/circuit switching, internet structure
- Performance: loss, delay, throughput
- Protocol layers, service models
- Security

The Internet: a “nuts and bolts” view

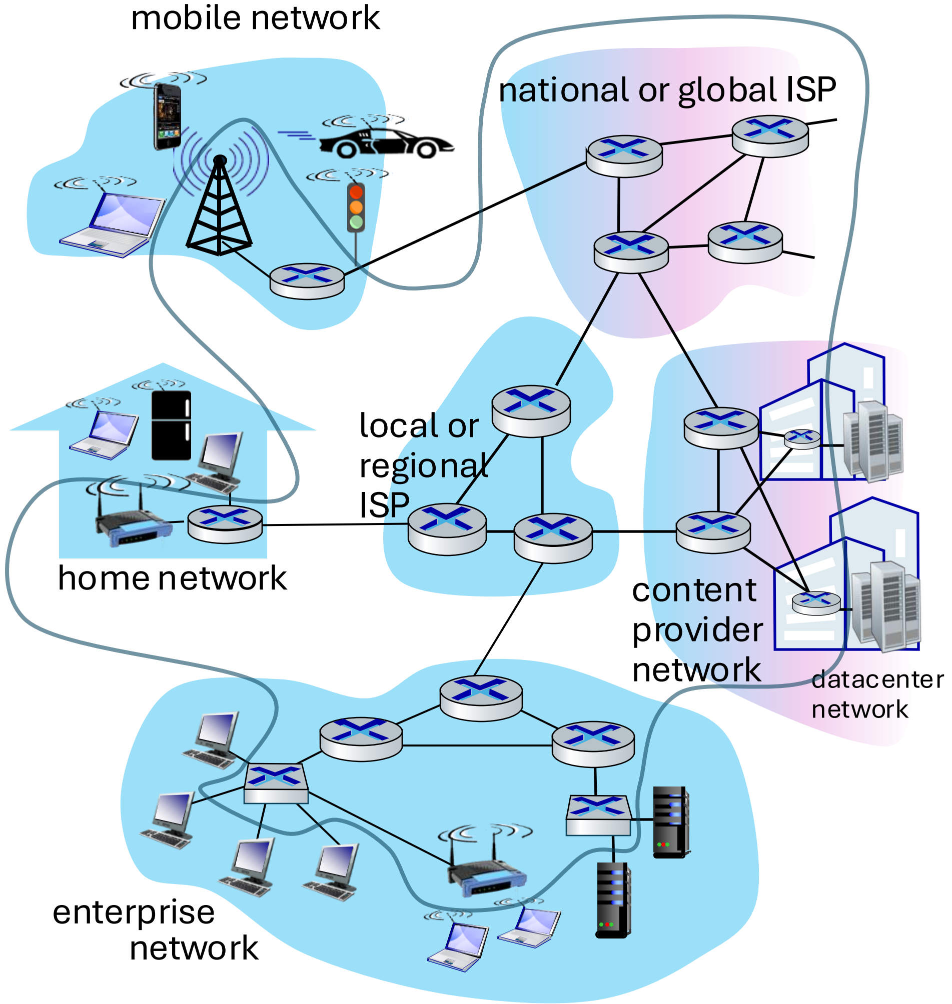
Billions of connected computing *devices*:
- *hosts* = *end systems*
- running *network apps* at Internet’s “edge”



Packet switches: forward packets (chunks of data)
- *routers*, *switches*

Communication links
- fiber, copper, radio, satellite
- transmission rate: *bandwidth*

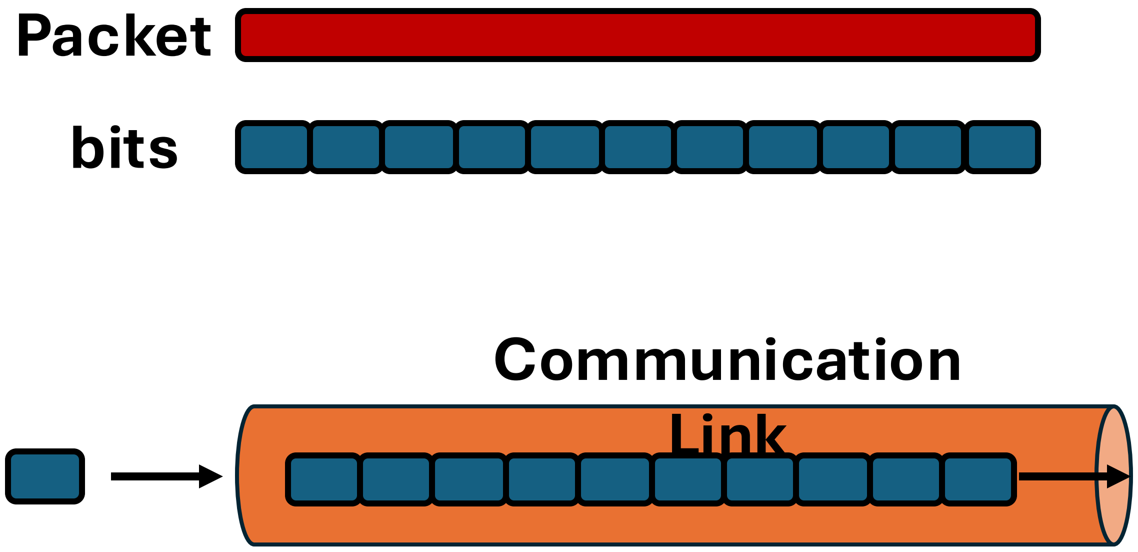
Networks
- collection of devices, routers, links: managed by an organization



Host: sends *packets* of data

host sending function:

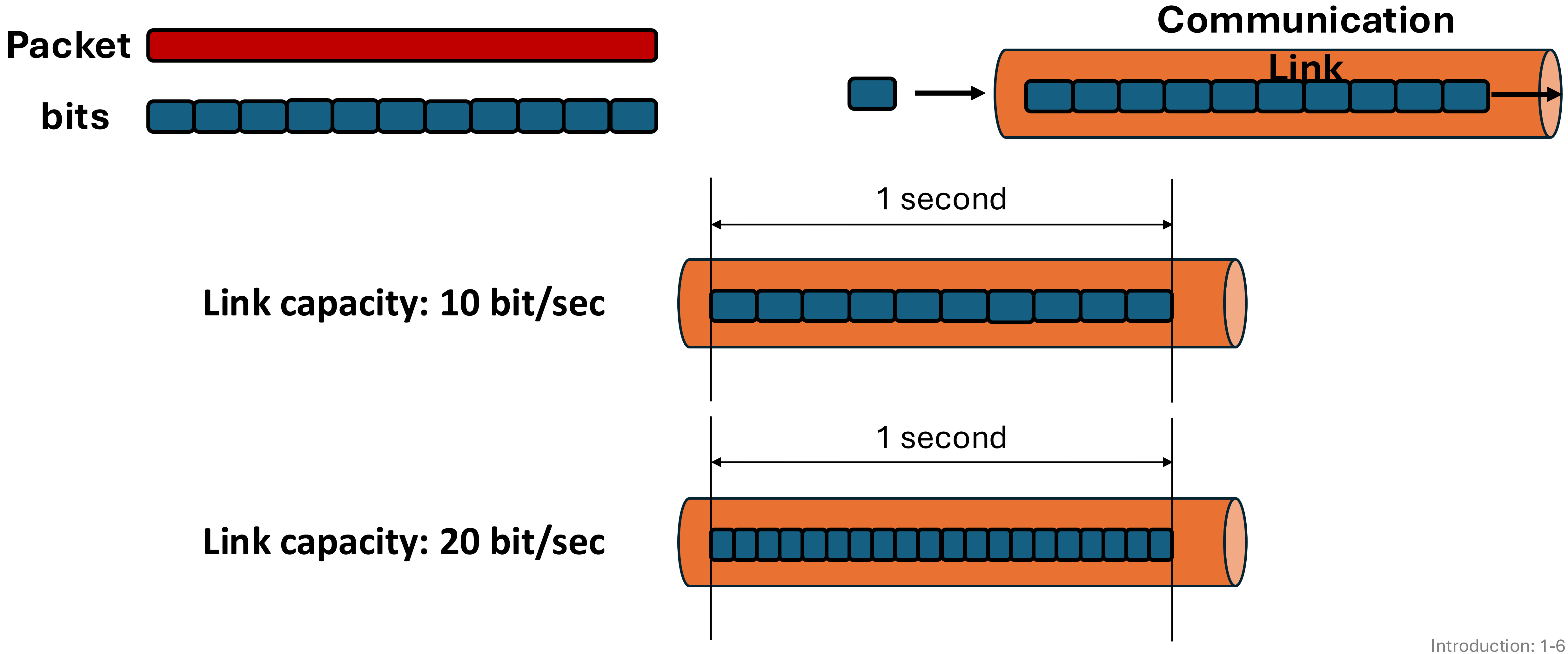
- takes application message
- breaks into smaller chunks, known as *packets*, of length *L* bits
- transmits packet into access network at *transmission rate R*
  - link transmission rate, aka link *capacity, aka link bandwidth*



What's the transmission rate R, link capacity or link bandwidth?

Host: sends *packets* of data

Link *transmission rate R, aka Link Capacity, aka link bandwidth*

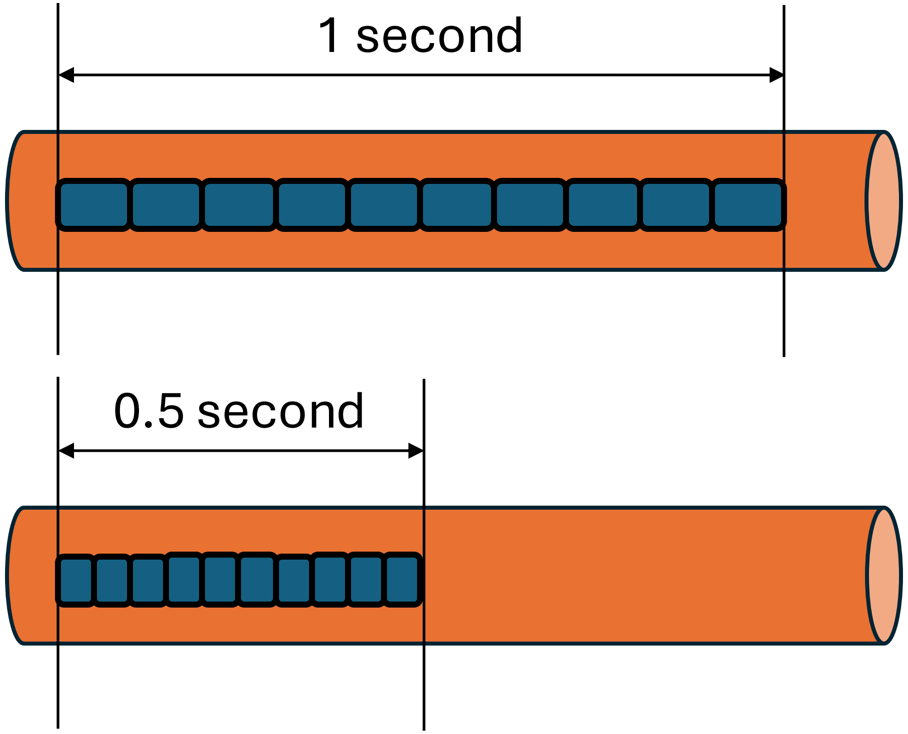


Host: sends *packets* of data

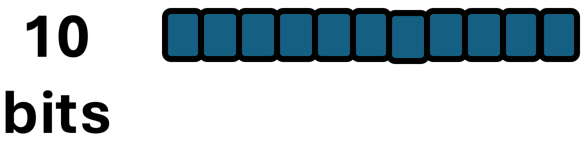
Link *transmission rate R*, *aka Link Capacity*, *aka link bandwidth*

Packet transmission delay

- How long it takes for transmitting all the bits into the network or communication link



A packet with 10 bits



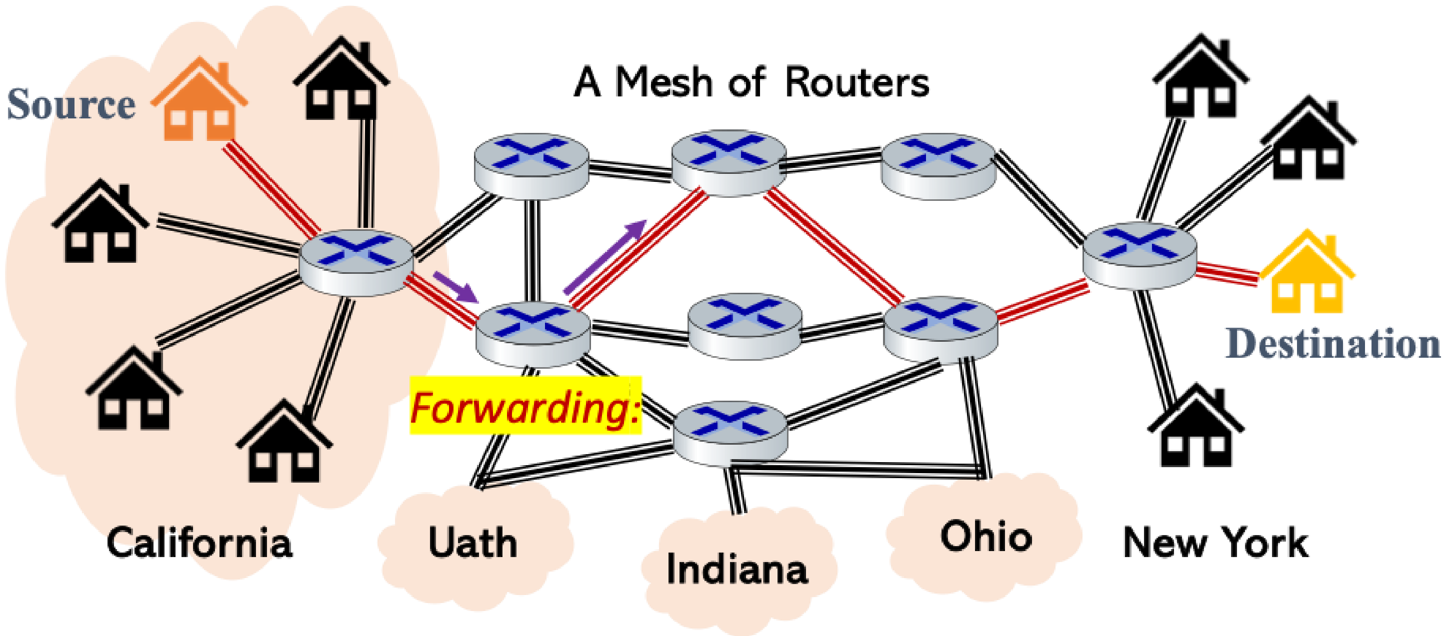
$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

Chapter 1: roadmap

- What *is* the Internet?
- What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: internet structure, routing and forwarding
- Performance: loss, delay, throughput
- Security
- Protocol layers, service models
- History

The network core – Routing

- Routing: Finding the Correct/Optimal path from source to destination



Forwarding:

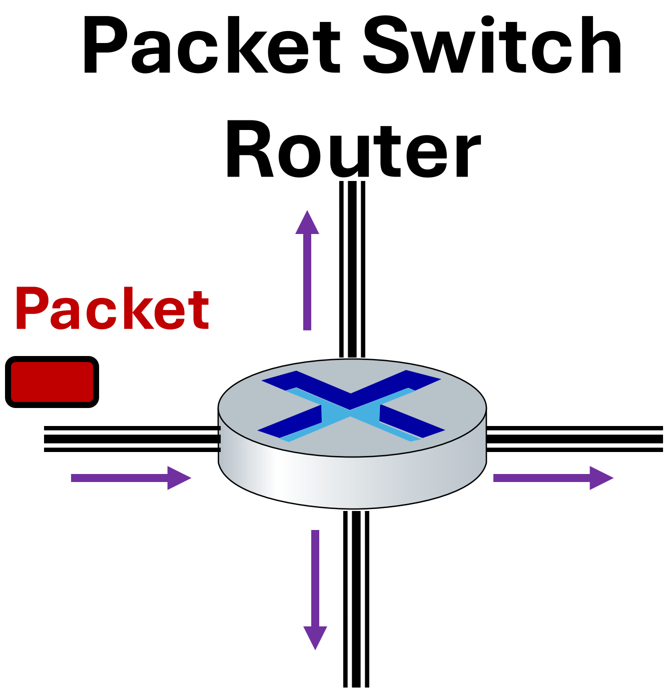
- *local* action: move arriving packets from router's input link to appropriate router output link

Routing:

- *global* action: determine source-destination paths taken by packets

Packet-switching: store-and-forward

- Forward is also called switching
  - *store and forward: entire* packet must arrive at router before it can be transmitted on next link



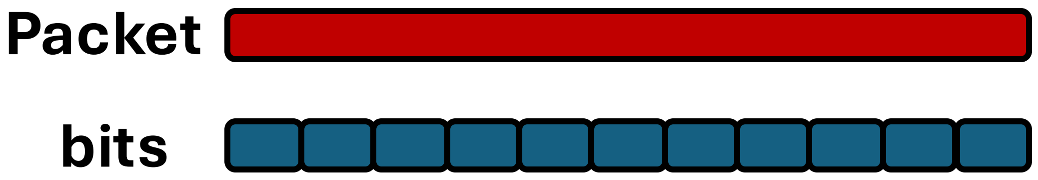
Forwarding:

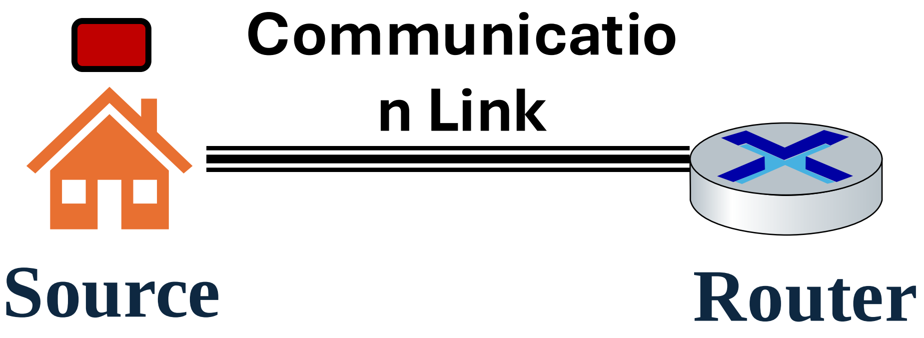
- aka "switching"
- *local* action: move arriving packets from router's input link to appropriate router output link

Chapter 1: roadmap

- What *is* the Internet?
- What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: internet structure, routing and forwarding
- Performance: loss, delay, throughput
- Security
- Protocol layers, service models
- History

How to send a packet via network

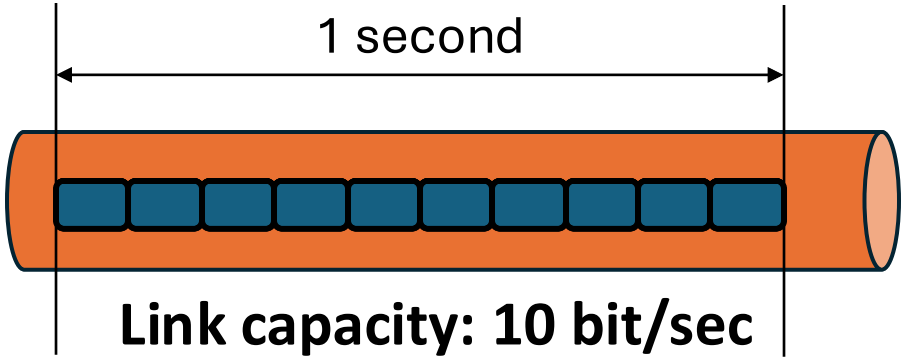




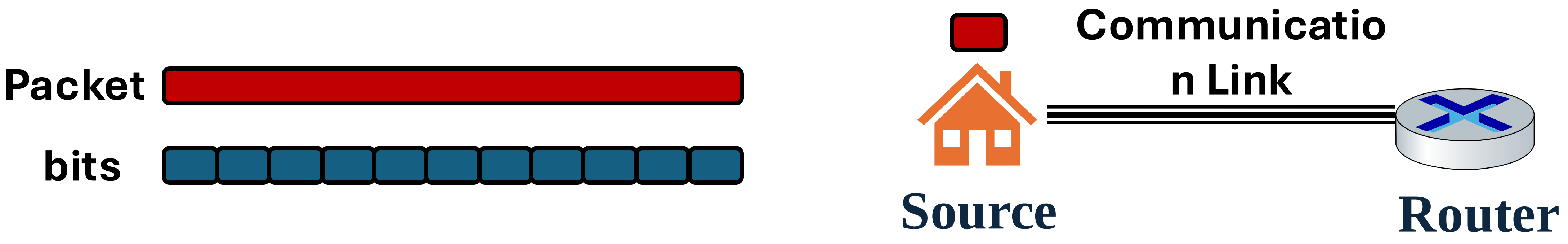
Step 1: Transmit the packets into the link

d_{trans}: transmission delay:

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



How to send a packet via network

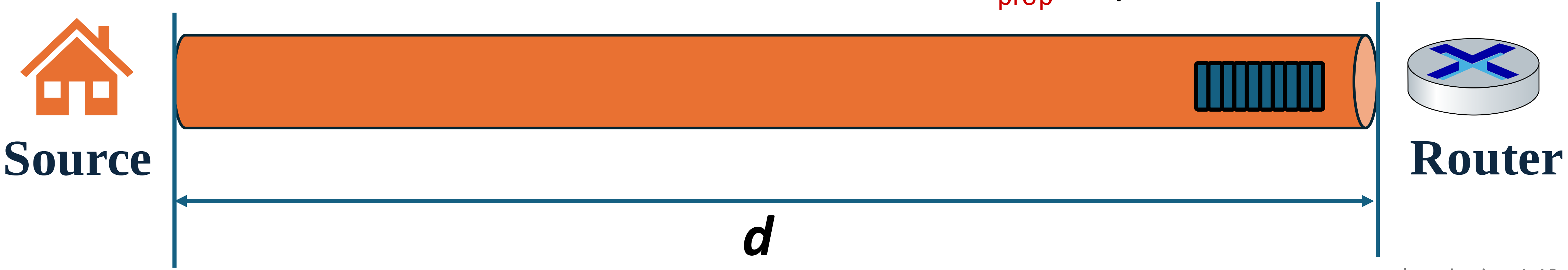


Step 1: Transmit the packets into the link

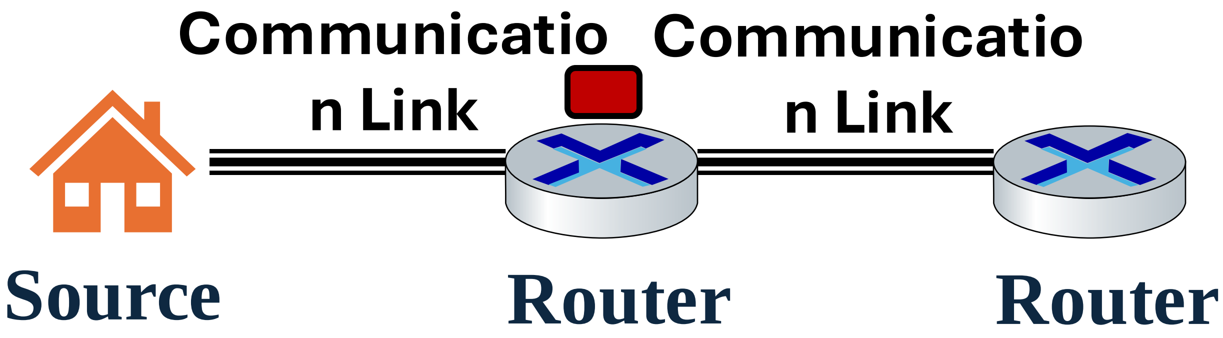
Step 2: The packet bits propagates to the router

d_{prop}: propagation delay:

- d: length of physical link
- s: propagation speed (~$2x10^8$ m/sec
- $d_{prop} = d/s$



How to send a packet via network

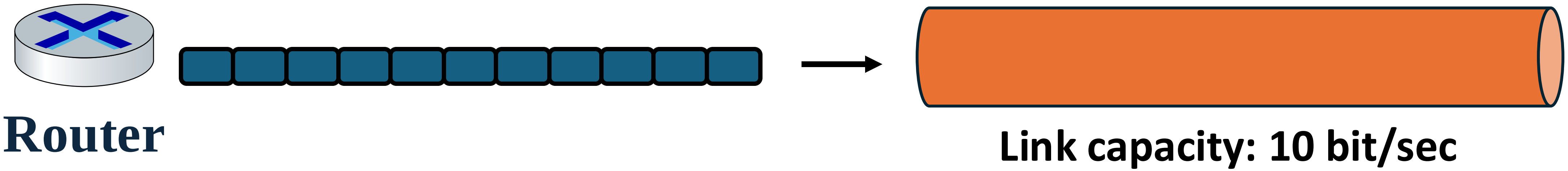


Step 1: Transmit the packets into the link

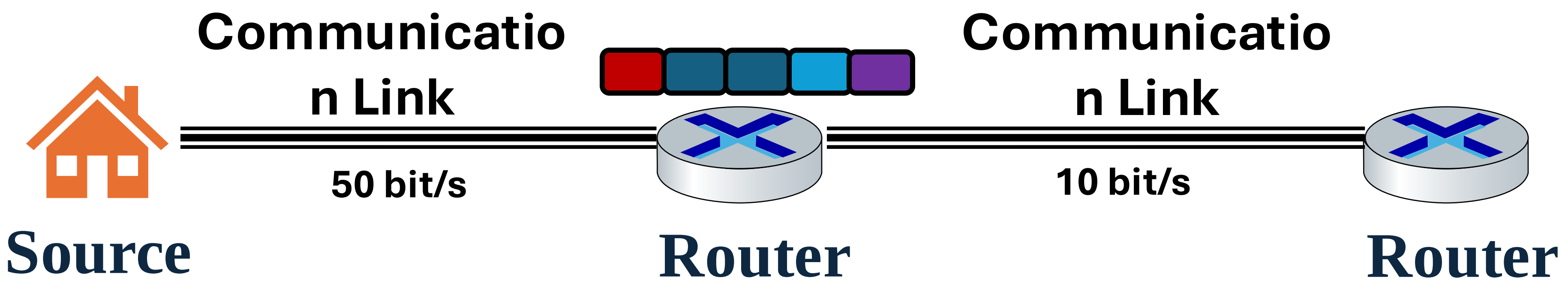
Step 2: The packet bits propagates to the router

d_{trans}: transmission delay:

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



How to send a packet via network



Key point:

- Router takes transmission delay to transmit a packet to the link
- The packet may arrive faster than the packets get out of the router
- The later arrived packets must wait at the router until all the packets arriving before it are transmitted into the link

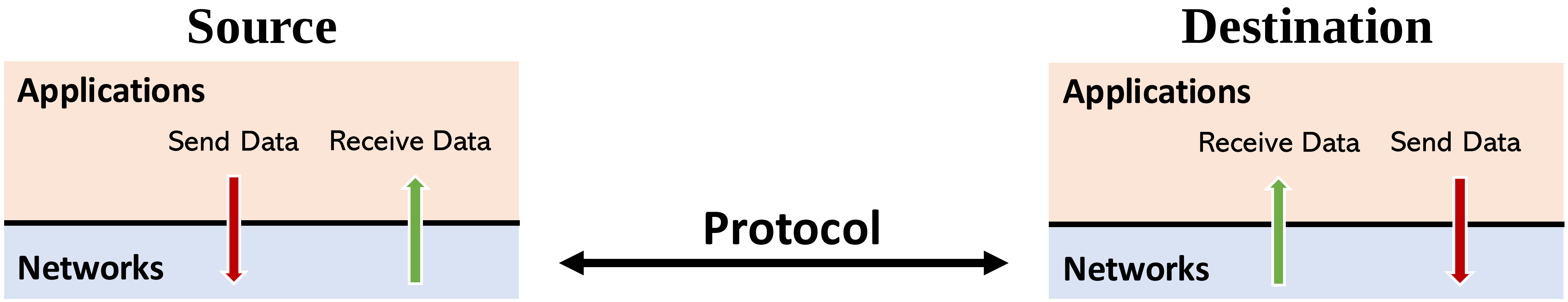
d_{queue}: queueing delay

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

Chapter 1: roadmap

- What *is* the Internet?
- What *is* a protocol?
- Network edge: hosts, access network, physical media
- Network core: packet/circuit switching, internet structure
- Performance: loss, delay, throughput
- Protocol layers, service models
- Security
- History

Structure of the layer design

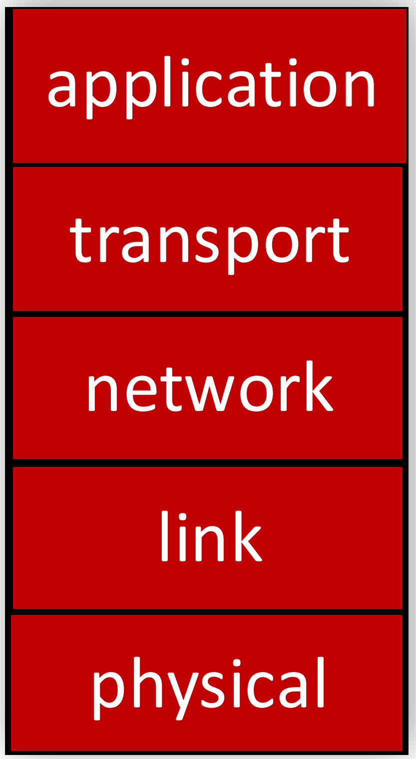


- **Service:** **What** a layer does
- **Service interface:** **How to access** the service
  - Interface for the layer **above**

- **Protocol interface:** **How peers communicate** to implement service
  - Set of rules and formats that govern the communication **between two Internet hosts**

Layered Internet protocol stack

- *application:* supporting network applications
  - HTTP, IMAP, SMTP, DNS
- *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, 802.11 (WiFi), PPP
- *physical:* bits “on the wire”

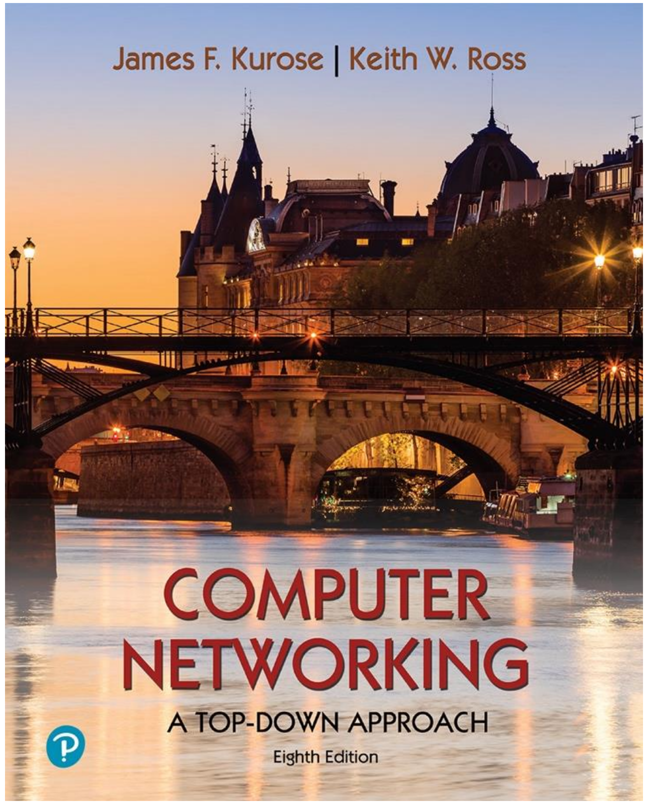


Chapter 2
Application Layer

Yaxiong Xie

Department of Computer Science and Engineering
University at Buffalo, SUNY

Adapted from the slides of the book's authors



Computer Networking: A Top-Down Approach
8th edition n
Jim Kurose, Keith Ross
Pearson, 2020

Client-server paradigm

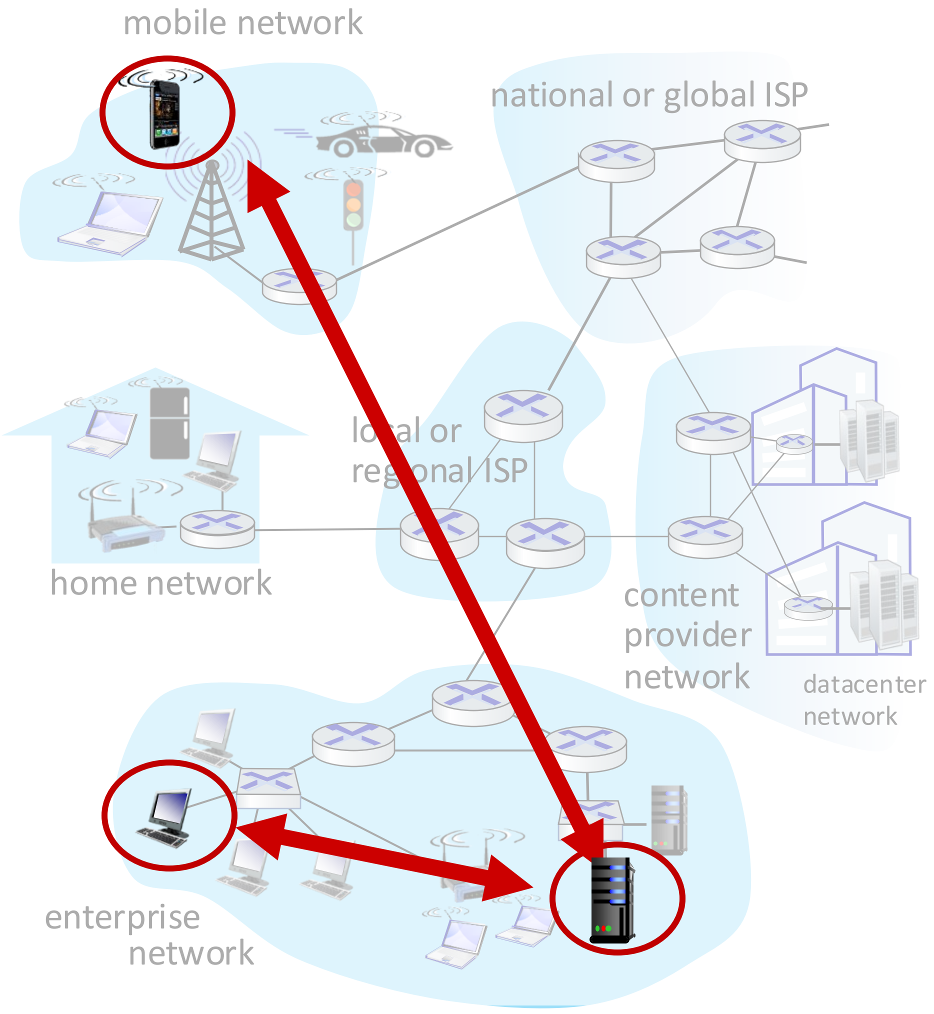
server:

- always-on host
- permanent IP address
- often in data centers, for scaling

clients:

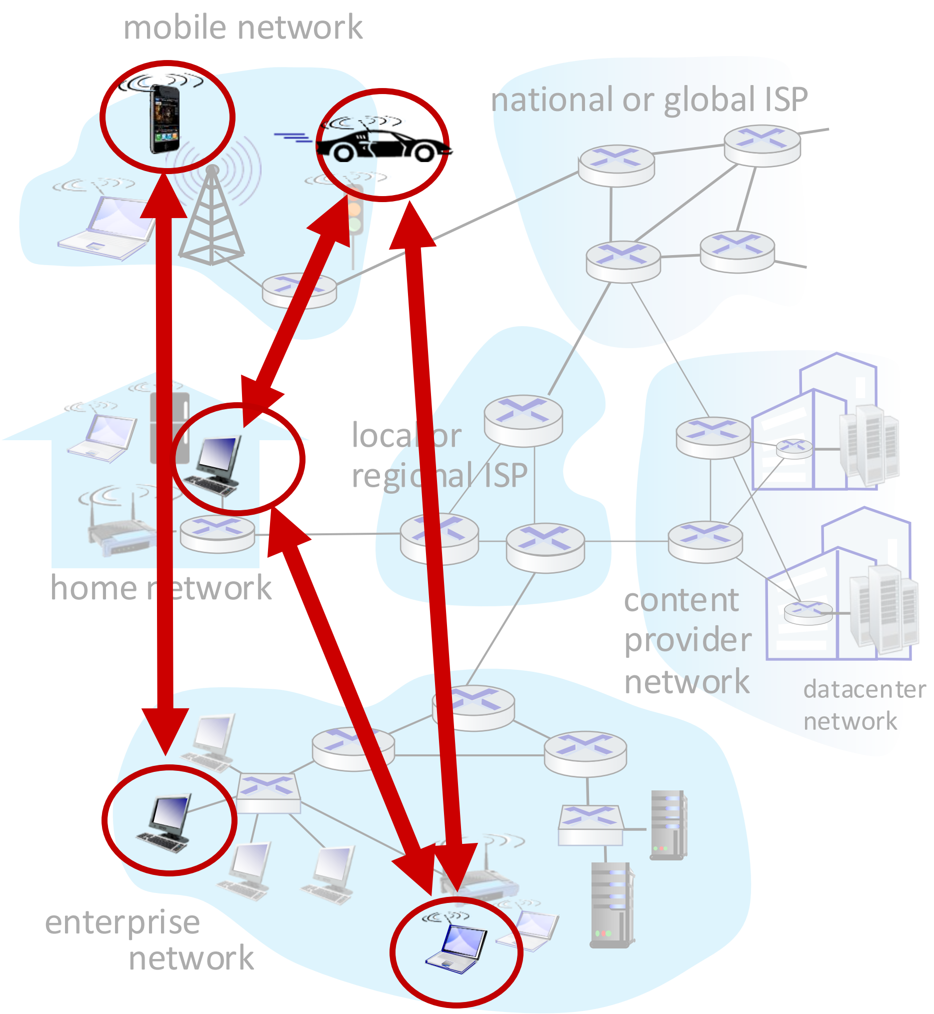
- contact, communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do *not* communicate directly with each other

- examples: HTTP, IMAP, FTP



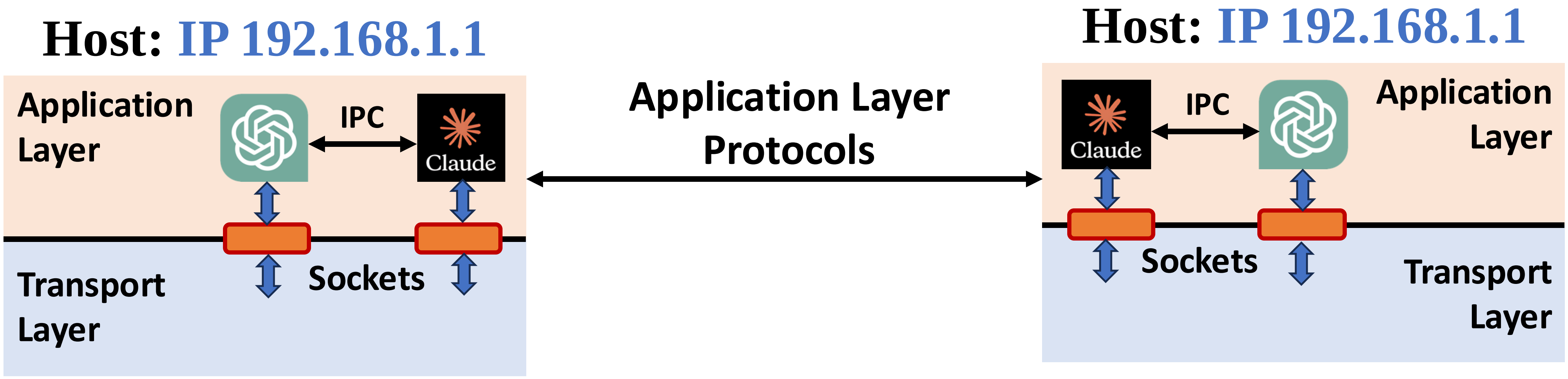
Peer-peer architecture

- *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
- example: P2P file sharing



Sockets (interface) and Protocols

- process sends/receives messages to/from its socket
- Process communicate with process on the other host via application layer protocols



Application layer: overview

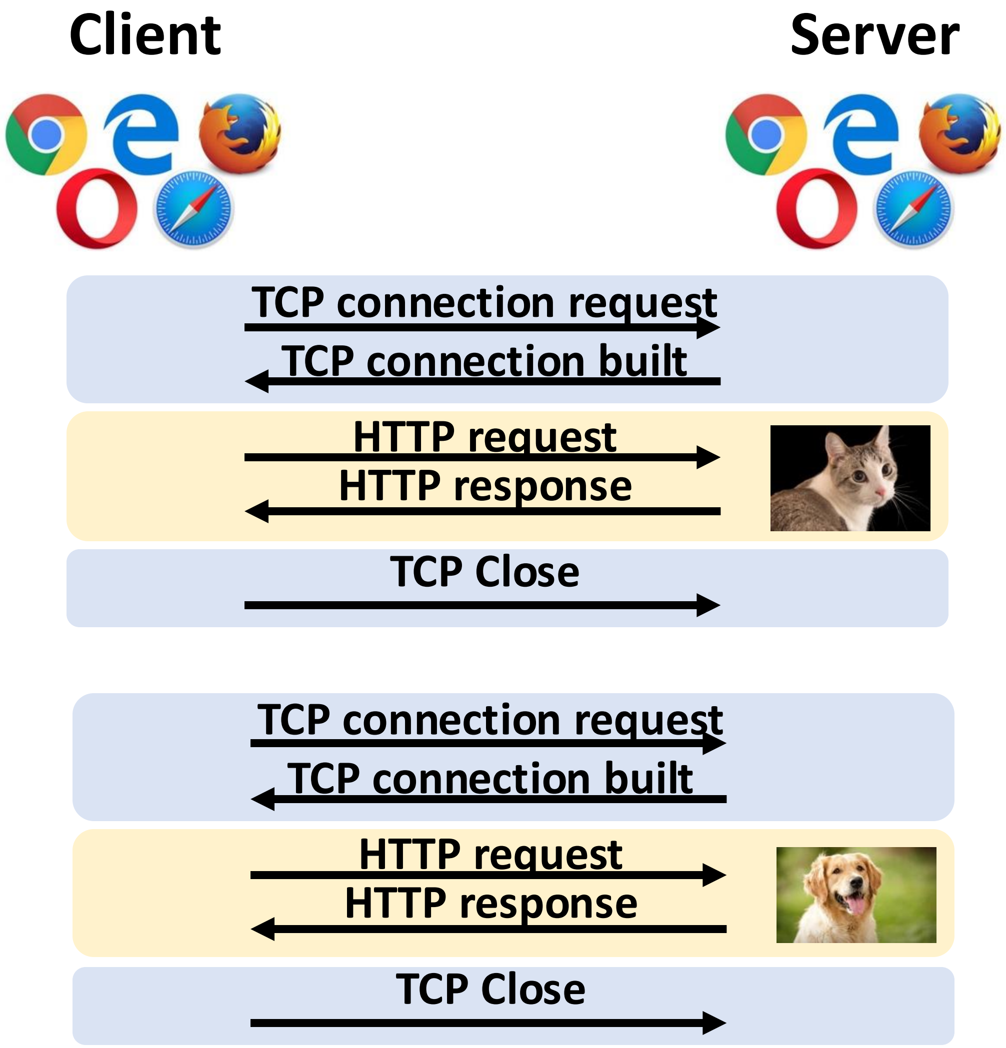
- Principles of network applications
- socket programming with UDP and TCP
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks

HTTP connections: two types

Non-persistent HTTP

1. TCP connection opened
2. at most one object sent over TCP connection
3. TCP connection closed

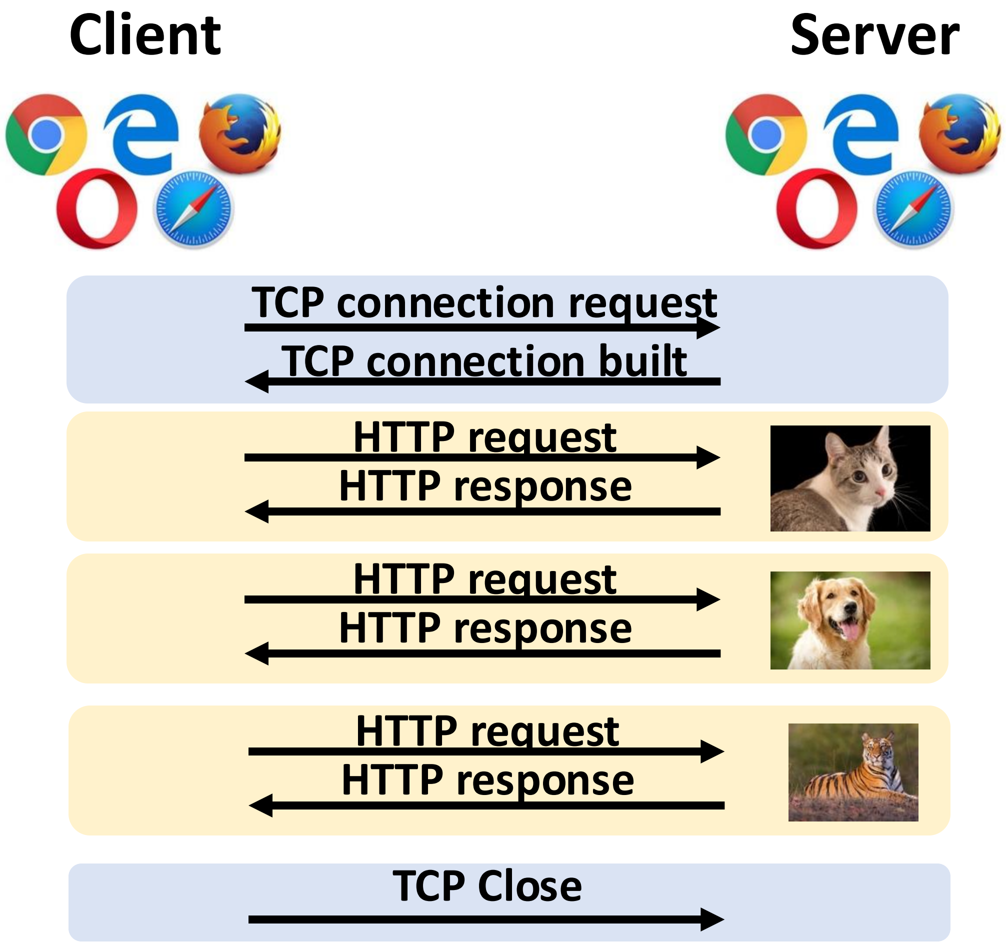
downloading multiple objects required multiple connections



HTTP connections: two types

Persistent HTTP

- TCP connection opened to a server
- multiple objects can be sent over *single* TCP connection between client, and that server
- TCP connection closed

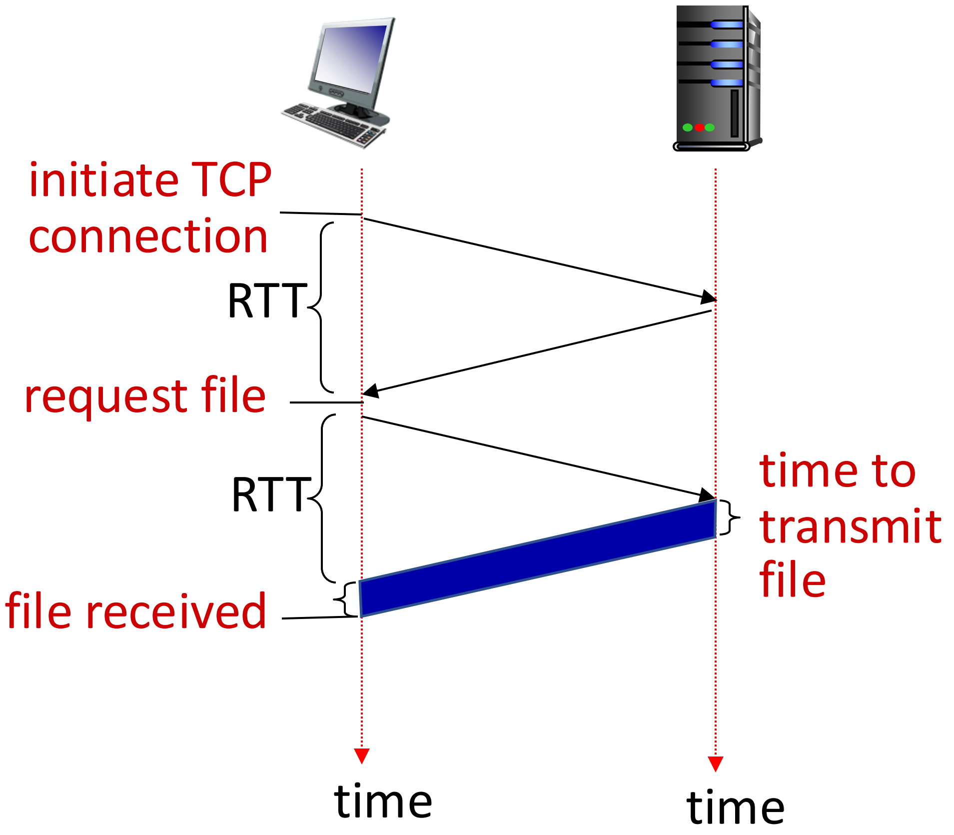


Non-persistent HTTP: response time

RTT (Round-Trip Time): time for a small packet to travel from client to server and back

HTTP response time (per object):

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- object/file transmission time

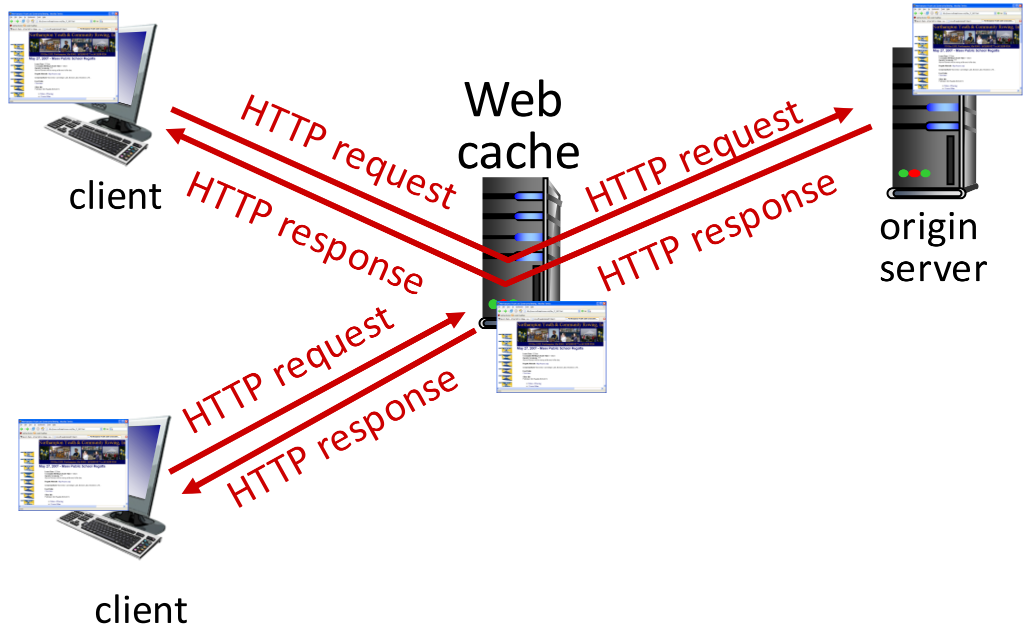


Non-persistent HTTP response time = 2RTT+ file transmission time

Web caches

Goal: satisfy client requests without involving origin server

- user configures browser to point to a (local) *Web cache*
- browser sends all HTTP requests to cache
  - *if* object in cache: cache returns object to client
  - *else* cache requests object from origin server, caches received object, then returns object to client



Caching example

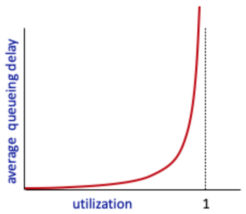
Scenario:

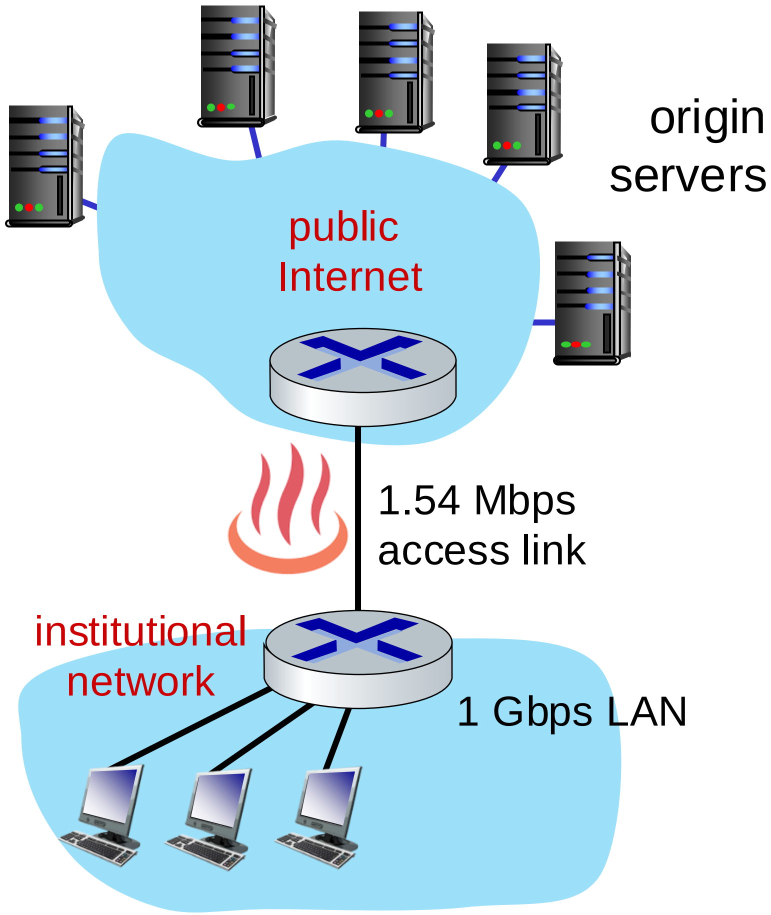
- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps

Performance:

- access link utilization = .97
- LAN utilization: .0015
- end-end delay = Internet delay + access link delay + LAN delay
  = 2 sec + minutes + usecs

problem: large queueing delays at high utilization!



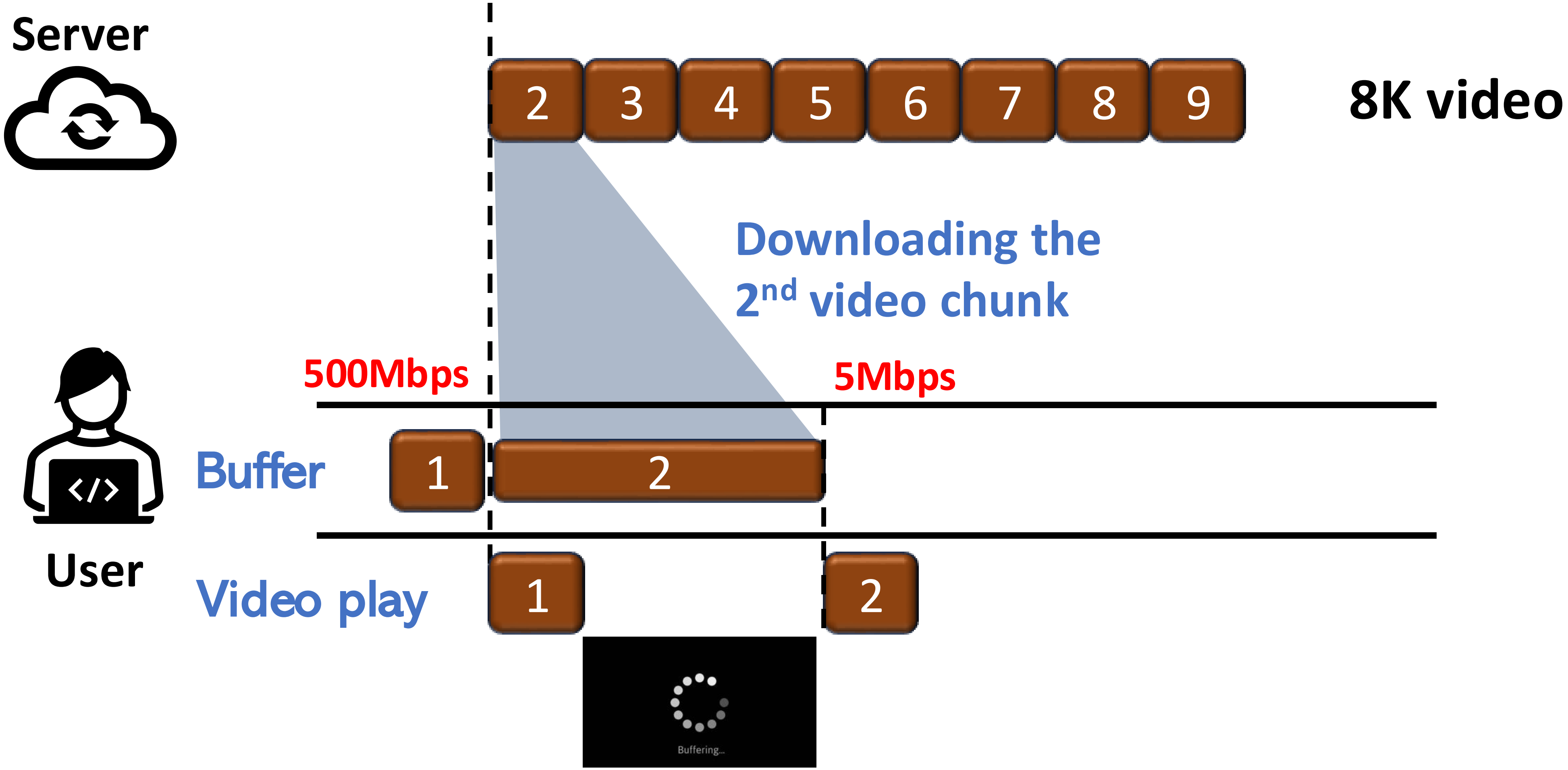


Application layer: overview

- Principles of network applications
- socket programming with UDP and TCP
- Web and HTTP
- E-mail, SMTP, IMAP
- The Domain Name System DNS
- P2P applications
- video streaming and content distribution networks

Video Streaming Applications

- **<u>Challenge:</u>** video quality and channel capacity



Transport layer: overview

Our goal:

- understand principles behind transport layer services:
  - multiplexing, demultiplexing
  - reliable data transfer
  - flow control
  - congestion control
- learn about Internet transport layer protocols:
  - UDP: connectionless transport
  - TCP: connection-oriented reliable transport
  - TCP congestion control

Chapter 3: roadmap

- Transport-layer services
- Multiplexing and demultiplexing
- Connectionless transport: UDP
- Principles of reliable data transfer
- Connection-oriented transport: TCP
- Principles of congestion control
- TCP congestion control
- Evolution of transport-layer functionality