CNT5008 Computer Networks

- Instructor: Dr. Ionut Cardei
- Textbook: “Computer Networks: A Systems Approach” , by Larry L. Peterson, Bruce S. Davie, 6th Edition
- Edition, Publisher: Morgan Kaufmann
- Available for free:
  - HTML Version of the Textbook (6th ed.)
  - 5th Edition is slightly outdated, but still OK. Also available for free online. Link to list of new material in the 6th edition compared to 5th.

CNT5008 Computer Networks

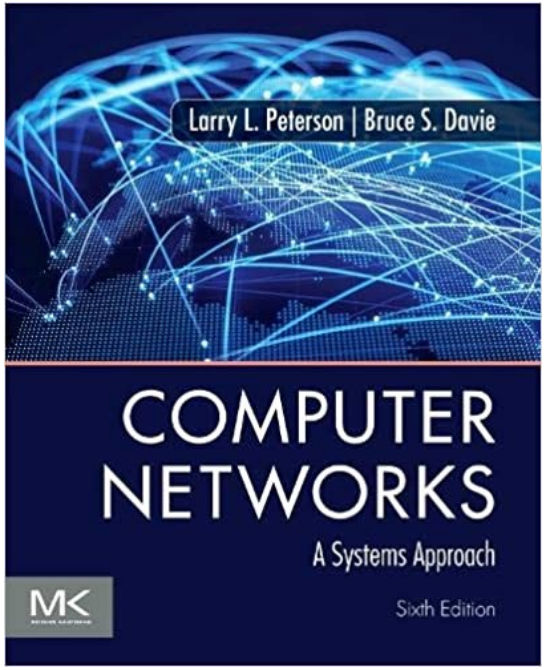
- Our lecture notes and lecture videos use the slides from textbook publisher
  - Additional material and examples not found in the textbook.
  - Mandatory to read in preparation for assignments and exams.
- Instructional materials:
  - Notes: required
  - Textbook chapters: required
  - Videos: recommended

CNT5008 Computer Networks

- Recipe for success:
  - Read textbook chapter and don't try to find answers with google or cheating websites
  - Pay attention to details
  - Ask for help:
    - Participate on the Homework Q&A Forum
    - Attend Office Hours
    - Email

Chapter 1

Foundation



Problems

- How to build a scalable network that will support different applications?
- What is a computer network?
- How is a computer network different from other types of networks?
- What is a computer network architecture?

Chapter Outline

- Applications
- Requirements
- Network Architecture
- Implementing Network Software
- Network Performance

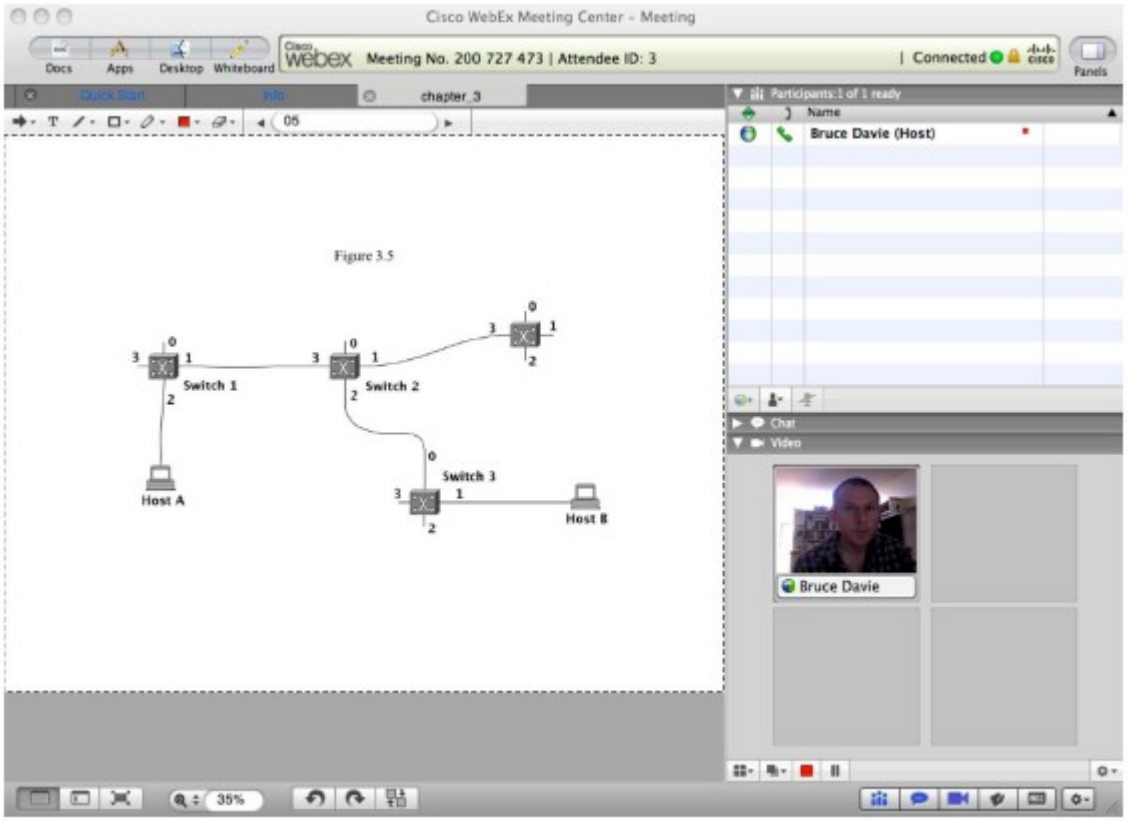
Chapter Goal

- Exploring the requirements that different applications and different communities place on the computer network
- Introducing the idea of network architecture
- Introducing some key elements in implementing Network Software
- Define key metrics that will be used to evaluate the performance of computer network

Applications

- Most people know about the Internet (a computer network) through applications
  - World Wide Web
  - Email
  - Online Social Network
  - Streaming Audio Video
  - File Sharing
  - Instant Messaging
  - …

Example of an application



A multimedia application including video-conferencing

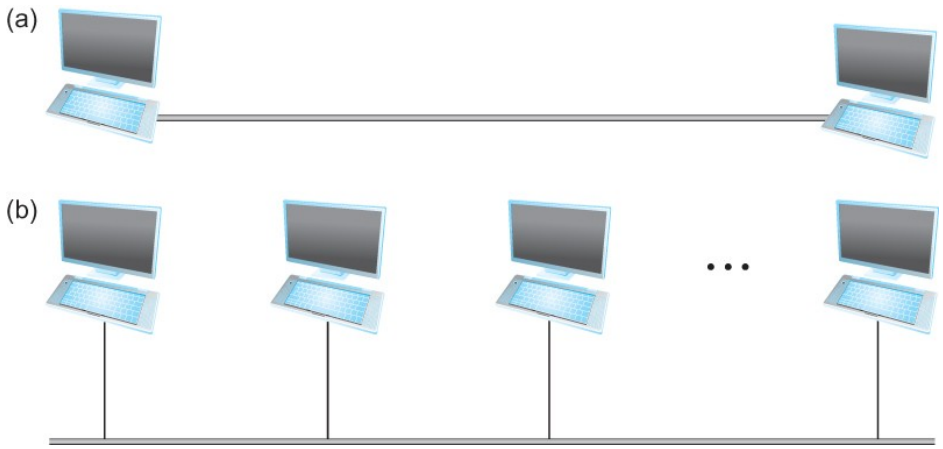
Application Protocol

- URL
  - Uniform resource locater
  - http://www.cs.princeton.edu/~llp/index.html
- HTTP
  - Hyper Text Transfer Protocol
- TCP
  - Transmission Control Protocol
- 17 messages for one URL request
  - 6 to find the IP (Internet Protocol) address
  - 3 for connection establishment of TCP
  - 4 for HTTP request and acknowledgement
    - Request: I got your request and I will send the data
    - Reply: Here is the data you requested; I got the data
  - 4 messages for tearing down TCP connection

Requirements

- Different stakeholders may list requirements differently:
- Application Programmer
  - List the services that his application needs: delay bounded delivery of data
- Network Designer
  - Design a cost-effective network with sharable resources
- Network Operator/Provider
  - List the characteristics of a system that is easy to manage

Scalable Connectivity

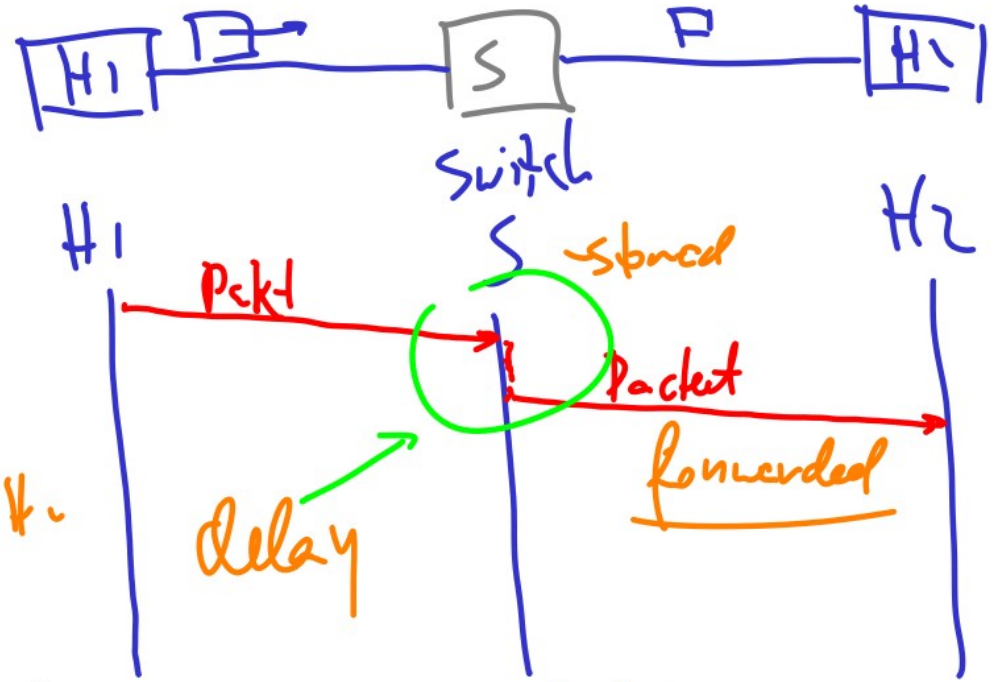


- Need to understand the following terminologies
  - **Scale: may grow to an arbitrary size**
  - Link
  - Nodes
  - Point-to-point
  - Multiple access
  - Switched Network
    - Circuit Switched
    - Packet Switched
  - Packet, message
  - Store-and-forward

(a) Point-to-point

(b) Multiple access

Store and Forward

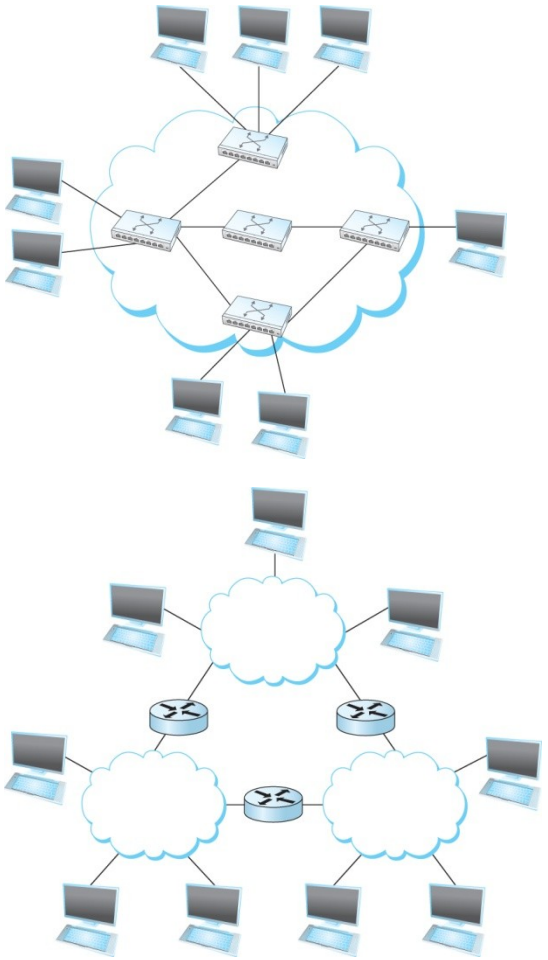


Routing:
- decide the path

pkt - received at S
- parsed / interpreted
- stored in memory
- transmitted by S to H2

time >0

Connectivity

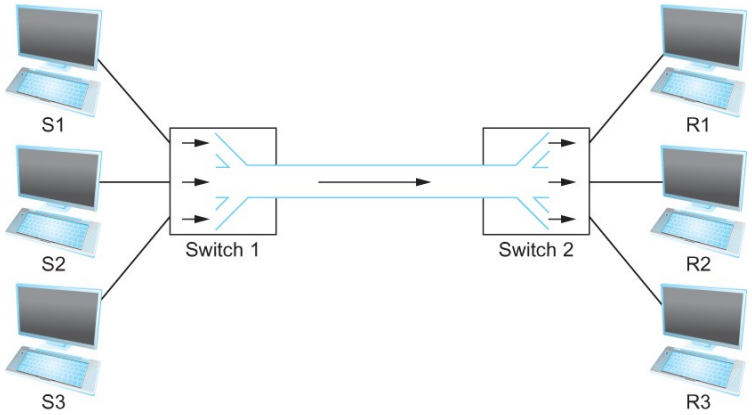


(a) A switched network

(b) Interconnection of networks

- Terminologies (contd.)
  - Cloud
  - Hosts
  - Switches
  - internetwork
  - Router/gateway
  - Host-to-host connectivity
  - Address
  - Routing
  - Unicast/broadcast/multicast

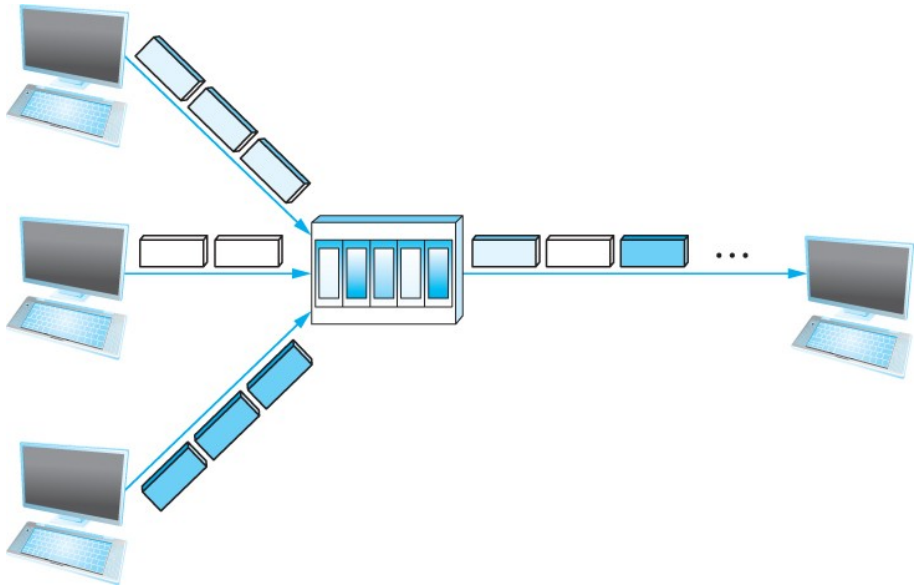
Cost-Effective Resource Sharing



Multiplexing multiple logical flows over a single physical link

- Resource: links and nodes
- How to share a link?
  - Multiplexing
  - De-multiplexing
  - Synchronous Time-division Multiplexing
    - Time slots/data transmitted in predetermined slots

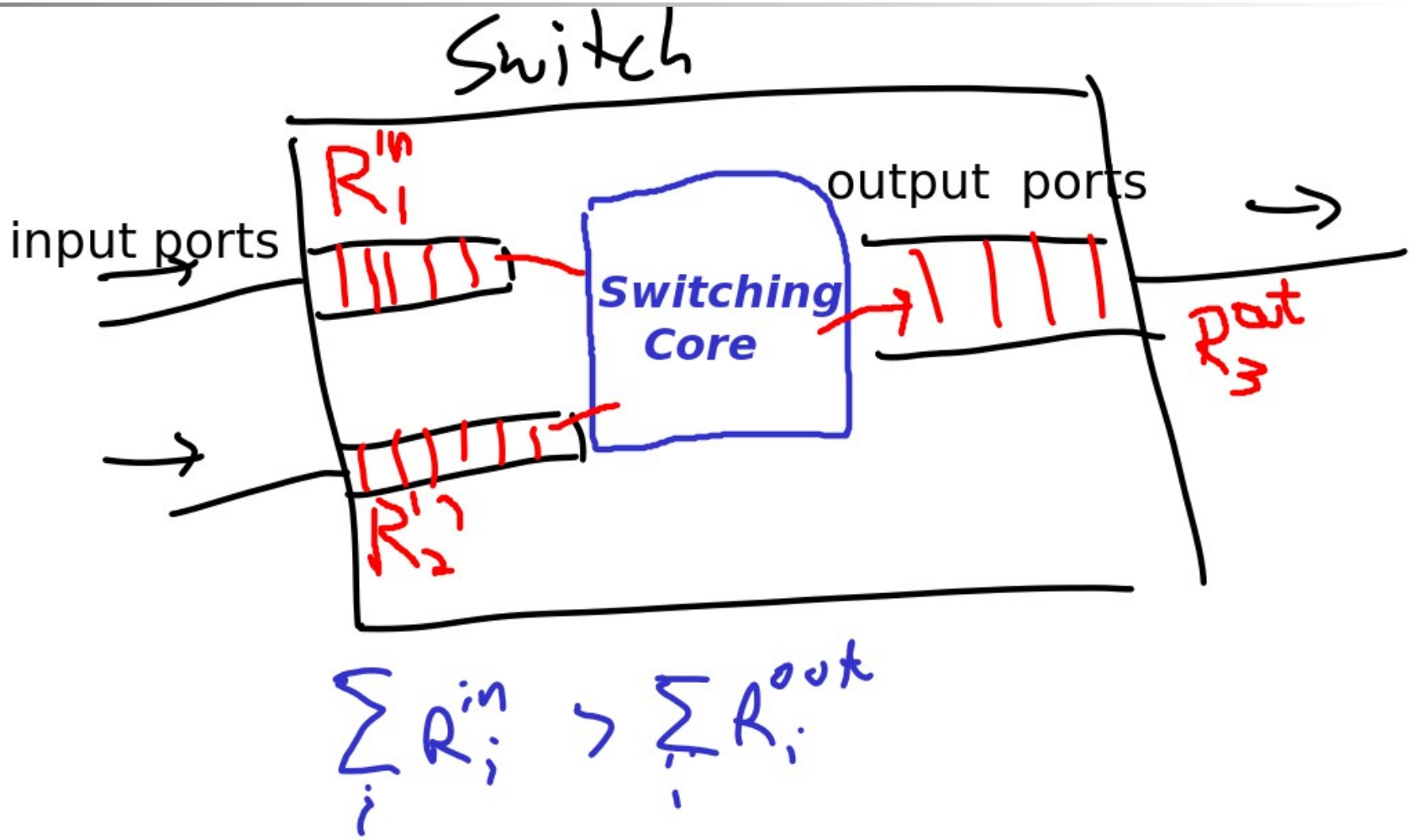
Cost-Effective Resource Sharing



A switch multiplexing packets from multiple sources onto one shared link

- FDM: Frequency Division Multiplexing
- Statistical Multiplexing
  - Data is transmitted based on demand of each flow.
  - What is a flow?
  - Packets vs. Messages
  - FIFO, Round-Robin, Priorities (Quality-of-Service (QoS))
  - Message fragmentation
  - Congestion
- LAN, MAN, WAN
- SAN (System Area Networks

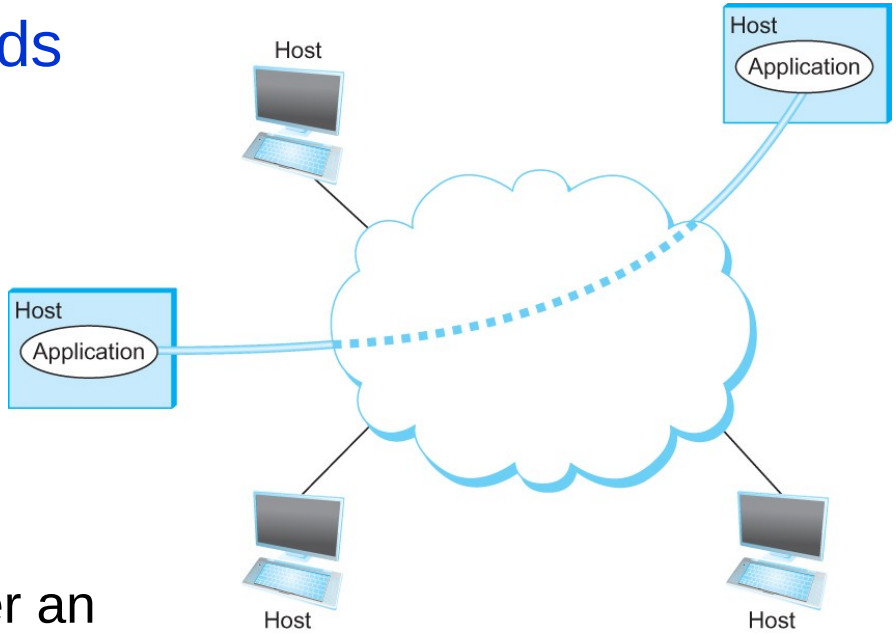
Congestion: at switches/routers



Condition for congestion:
total data rate in > total data rate out

Support for Common Services

- Logical Channel abstraction (i.e. model)
  - Application-to-Application communication path or a pipe
  - Look at application needs



Process communicating over an abstract channel

Common Communication Patterns

- Look at some typical applications:
  - File server
  - Digital library
  - Videoconferencing
- Client/Server
- Two types of communication channel
  - Request/Reply Channels
  - Message Stream Channels
- Find functionality and identify where is it provided:
  - End hosts (‘dumb core’) or in the network ?

Reliability

- Network should hide the errors
- Bits are lost
  - Bit errors (1 to a 0, and vice versa)
  - Burst errors – several consecutive errors
- Packets are lost (Congestion)
- Links and Node failures
- Messages are delayed
- Messages are delivered out-of-order
- Third parties eavesdrop

Key Takeaway for Defining *Channels*

- Defining useful channels involves
  - understanding the applications' requirements
  - recognizing the limitations of the underlying technology.
- The challenge is to fill in the gap between
  - application expectations
  - capabilities of underlying technologies
  - called the *semantic gap*

Manageability

- Managing a network involves:
  - upgrading equipment as the network grows to carry more traffic or reach more users
  - troubleshooting
  - adding new features in support of new applications
- Automation is required for scalability
- Fundamental tension between stability and feature velocity

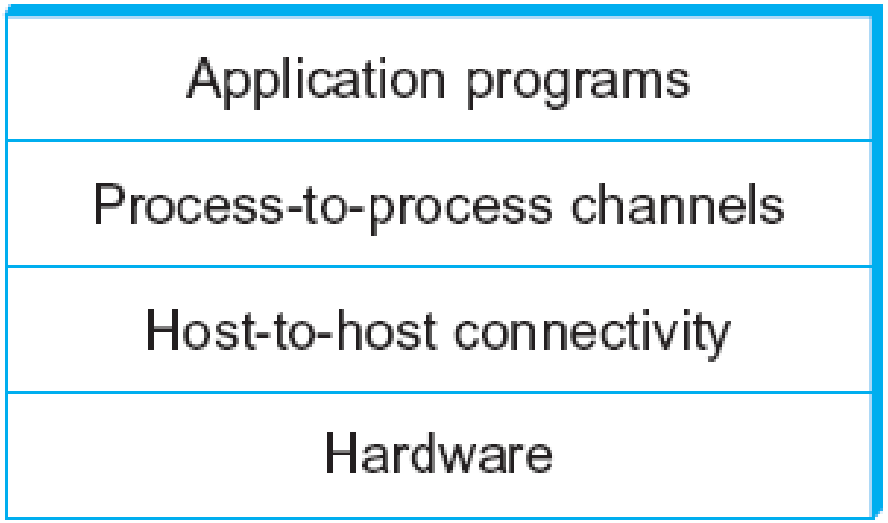
Network Architecture

- Network architecture is a general blueprint that guides the design and implementation of networks.
  - No need to redesign for each new application.
- We introduce general concepts and:
  - ISO – OSI 7 layer model
  - Internet architecture (TCP/IP based)

Layering - Abstraction

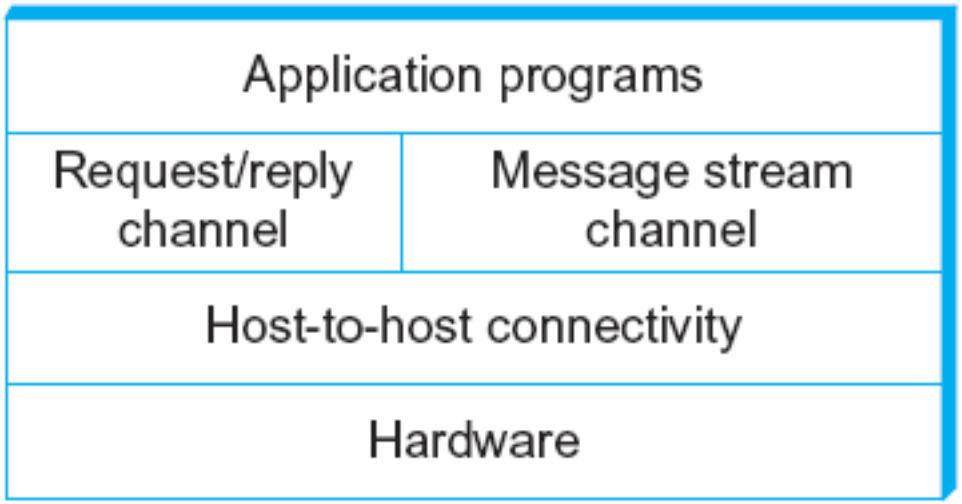
- Hiding of implementation details behind a well-defined interface
- Fundamental tool used by system designers to manage complexity
- Abstraction is to:
  - define a model capturing important aspects
  - encapsulate this model in an object that provides an interface that can be manipulated by other components of the system
  - hide the details of how the object is implemented from the users of the object.
- Examples in software development, science, engineering.

Network Architecture



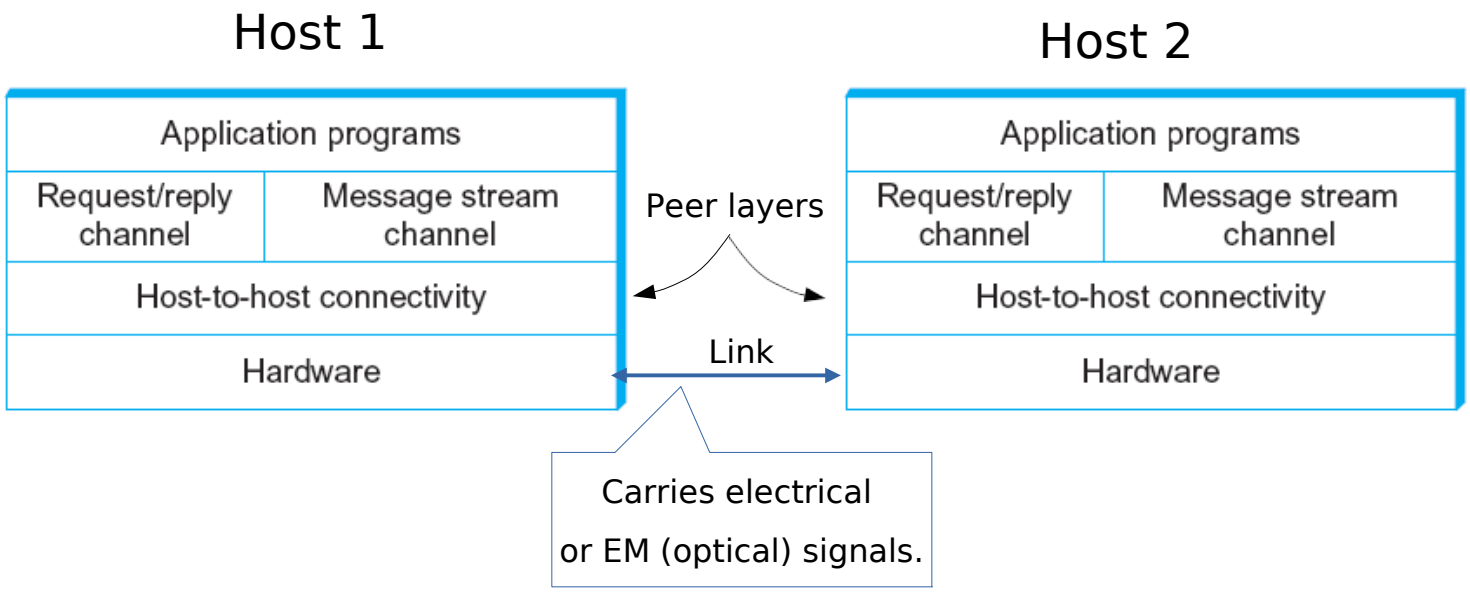
Example of a layered network system

Network Architecture



Layered system with alternative abstractions available at a given layer

Network Architecture



Between two nodes: physical link at the Hardware layer.

Application programs use the lower layers to communicate.

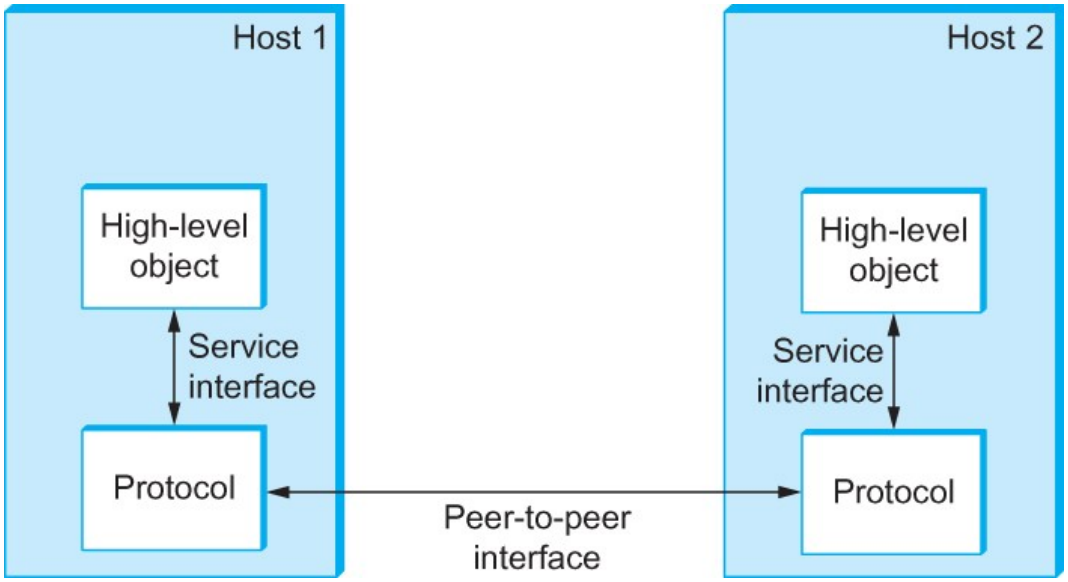
Advantages for Abstraction with Layers

- Decomposes the problem of building a network into manageable components. No need for a monolithic piece (of software/hardware).
- Provides modular design.
  - If you decide that you want to add some new service, you may only need to modify the functionality at one layer, reusing the functions provided at all the other layers.

Protocols

- Protocol defines the interfaces between the layers in the same system and with the layers of peer system
- Building blocks of a network architecture
- Each protocol object has two different interfaces
  - service interface: operations on this protocol
  - peer-to-peer interface: messages exchanged with peer
- Term “protocol” is overloaded
  - specification of peer-to-peer interface
  - module that implements this interface

Interfaces

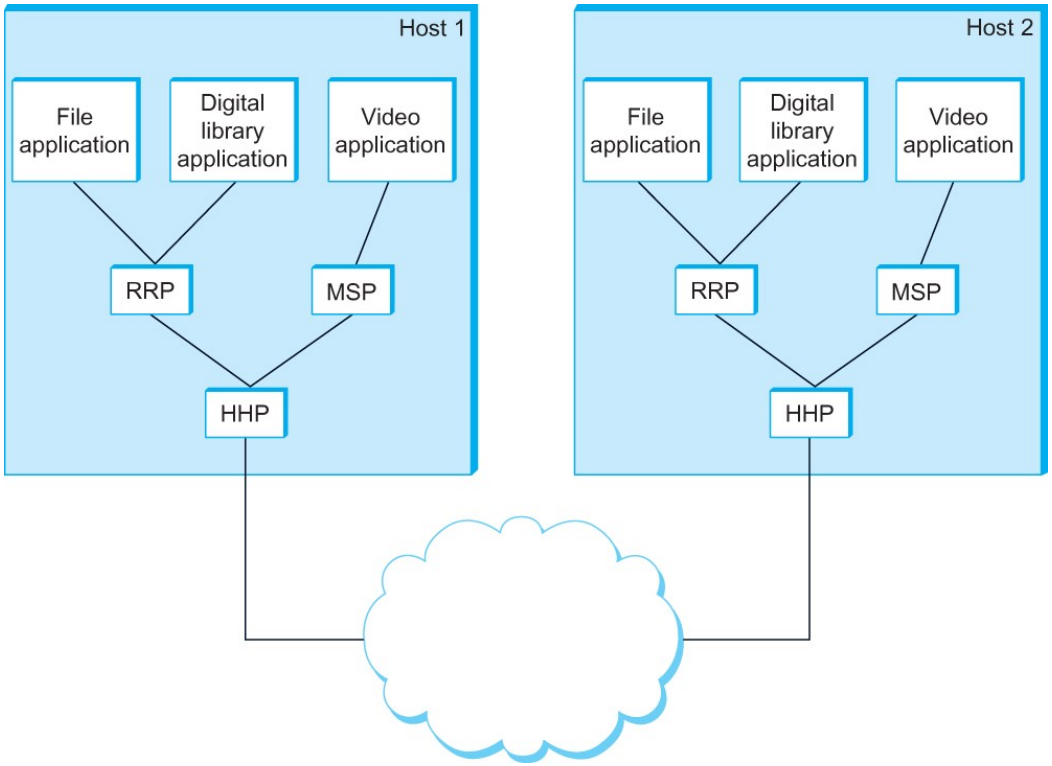


Service and Peer Interfaces

Protocols

- Protocol Specification: prose, pseudo-code, state transition diagram
- Interoperable: when two or more protocols that implement the specification accurately
- IETF: Internet Engineering Task Force

Protocol Graph



RRP=request/reply protocol

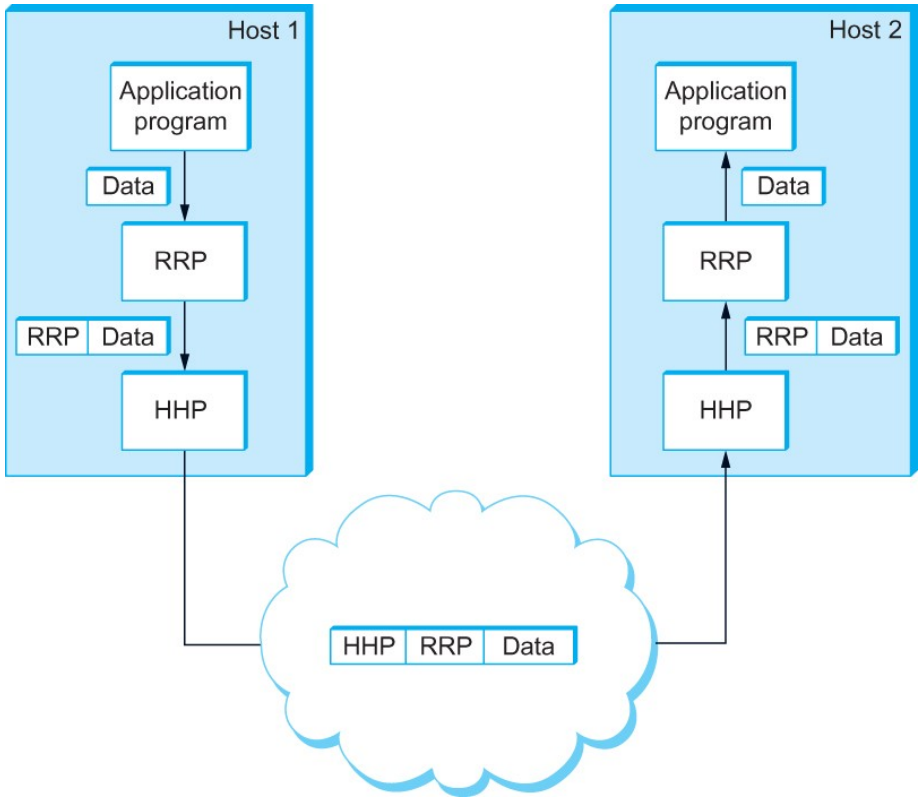
MSP=message streaming protocol

HHP=host to host protocol

Example of a protocol graph

nodes are the protocols and links the “depends-on” relation

Encapsulation

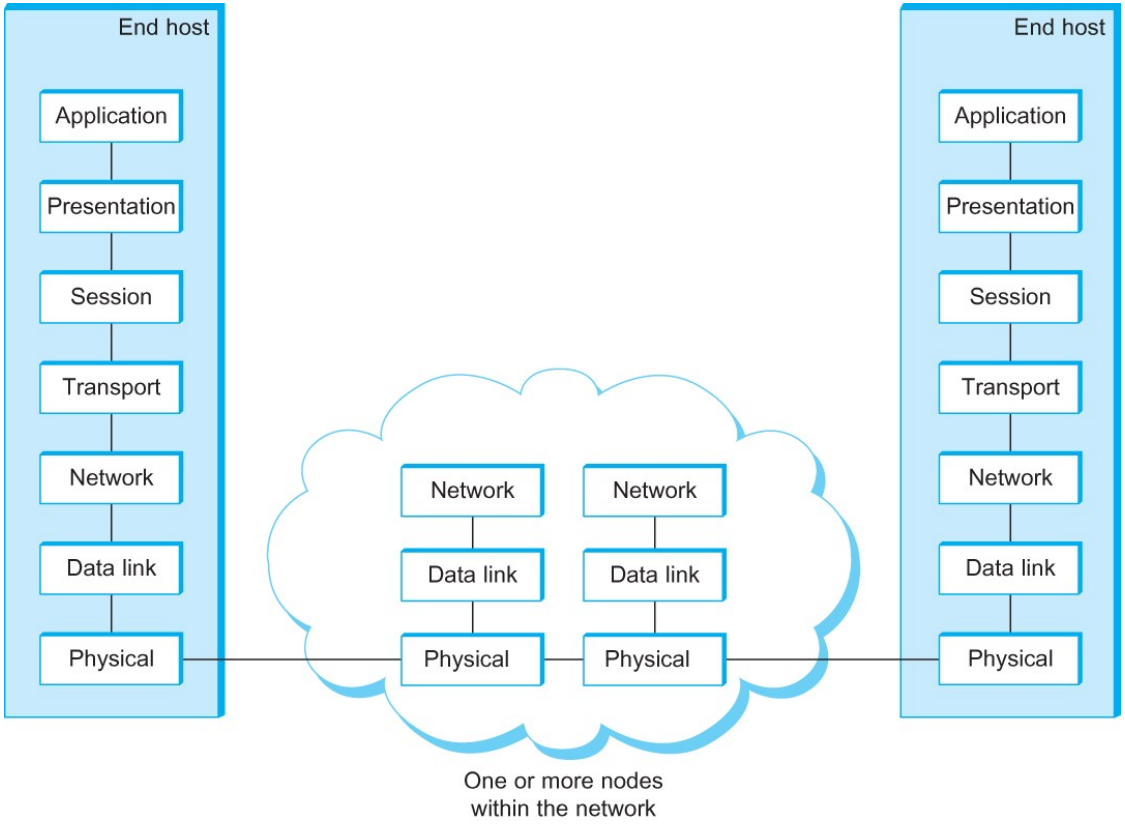


High-level messages are encapsulated inside of low-level messages

Multiplexing and Demultiplexing

- Fundamental idea of packet switching is to multiplex multiple flows of data over a single physical link.
- Works at **all** layers in a protocol graph.
- Packet header includes demultiplexing key.
- Demux key at the Hardware layer is the network adapter address (e.g. Ethernet MAC address: 48 bits).

ISO - OSI Architecture



The OSI 7-layer Model

ISO – International Standards Organization

OSI – Open Systems Interconnection

Description of Layers

- Physical Layer
  - Handles the transmission of raw bits over a communication link
- Data Link Layer
  - Collects a stream of bits into a larger aggregate called a *frame*
  - Network adaptor along with device driver in OS implement the protocol in this layer
  - Frames are actually delivered to hosts
- Network Layer
  - Handles routing among nodes within a packet-switched network
  - Unit of data exchanged between nodes in this layer is called a *packet*

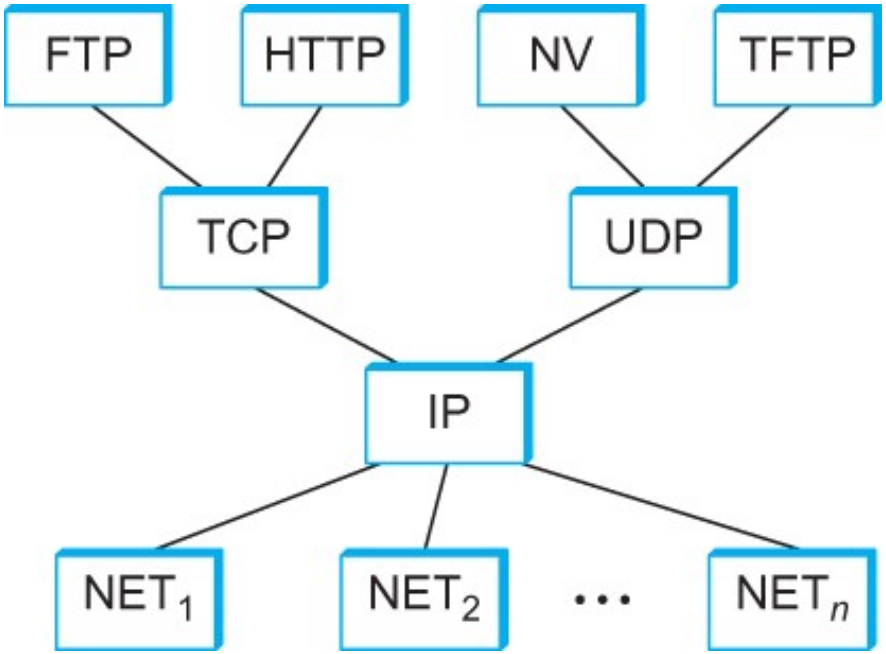
The lower three layers are implemented on all network nodes

Description of Layers

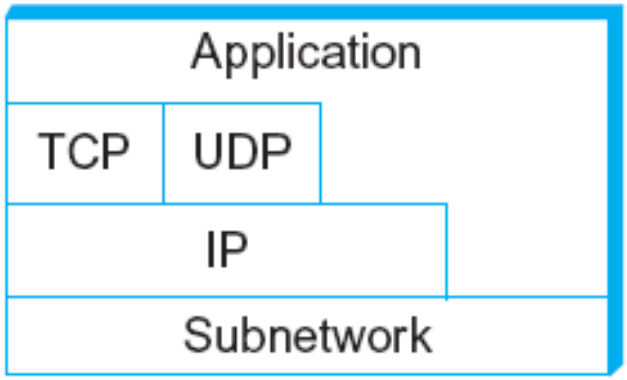
- Transport Layer
  - Implements a process-to-process channel
  - Unit of data exchanges in this layer is called a *message*
- Session Layer
  - Provides a name space that is used to tie together the potentially different transport streams that are part of a single application
- Presentation Layer
  - Concerned about the format of data exchanged between peers
- Application Layer
  - Standardize common type of exchanges

The transport layer and the higher layers typically run only on end-hosts and not on the intermediate switches and routers

Internet Architecture



Internet Protocol Graph



Alternative view of the Internet architecture. The "Network" layer shown here is sometimes referred to as the "sub-network" or "link" layer.

Internet Architecture

- Defined by IETF
- Three main features
  - Does not imply strict layering. The application is free to bypass the defined transport layers and to directly use IP or other underlying networks
  - An hour-glass shape – wide at the top, narrow in the middle and wide at the bottom. IP serves as the focal point for the architecture
  - In order for a new protocol to be officially included in the architecture, there needs to be both a protocol specification and at least one (and preferably two) representative implementations of the specification

Application Programming Interface

- Interface exported by the network
- Since most network protocols are implemented (those in the high protocol stack) in software and nearly all computer systems implement their network protocols as part of the operating system, when we refer to the interface "*exported by the network*", we are generally referring to the interface that the OS provides to its networking subsystem
- The interface is called the network Application Programming Interface (API)

Application Programming Interface (Sockets)

- Socket Interface was originally provided by the Berkeley distribution of Unix
  - Now supported in virtually all operating systems

- Each protocol provides a certain set of *services*, and the API provides a syntax by which those services can be invoked in this particular OS

Example Application: socket API with TCP

- Client – server structure
- Client:
  - reads text typed by user on the terminal
  - sends text to the server in a loop.
- Server:
  - listens for incoming connections
  - Accepts new TCP connection
  - Reads one line of text from client
  - Prints to the terminal.
  - Closes connection to client
  - Waits for connection from new client in a loop.

Socket

- What is a socket?
  - The point where a local application process attaches to the network
  - An interface between an application and the network
  - An application creates the socket
- The interface defines operations for
  - Creating a socket
  - Attaching a socket to the network
  - Sending and receiving messages through the socket
  - Closing the socket

Socket

- Socket Family
  - PF_INET denotes the Internet family
  - PF_UNIX denotes the Unix pipe facility
  - PF_PACKET denotes direct access to the network interface (i.e., it bypasses the TCP/IP protocol stack)
- Socket Type
  - SOCK_STREAM is used to denote a byte stream
  - SOCK_DGRAM is an alternative that denotes a message oriented service, such as that provided by UDP

Creating a Socket

```
int sockfd = socket(address_family, type, protocol);
```

- The socket number returned is the socket descriptor for the newly created socket

- `int sockfd = socket (PF_INET, SOCK_STREAM, 0);`
- `int sockfd = socket (PF_INET, SOCK_DGRAM, 0);`

  The combination of PF_INET and SOCK_STREAM implies TCP

Client-Serve Model with TCP

Server

- Passive open
- Prepares to accept connection, does not actually establish a connection

Server invokes

```
int bind (int socket, struct sockaddr *address,
                                    int addr_len)
int listen (int socket, int backlog)
int accept (int socket, struct sockaddr *address,
                                int *addr_len)
```

Client-Serve Model with TCP

Bind

- Binds the newly created socket to the specified address i.e. the network address of the local participant (the server)
- Address is a data structure which combines IP and port

Listen

- Defines how many connections can be pending on the specified socket

Client-Serve Model with TCP

Accept

- Carries out the passive open
- Blocking operation
  - Does not return until a remote participant has established a connection
  - When it does, it returns a new socket that corresponds to the new established connection and the address argument contains the remote participant's address

Client-Serve Model with TCP

Client

- Application performs active open
- It says who it wants to communicate with

Client invokes

```
int connect (int socket, struct sockaddr *address,
                                    int addr_len)
```

Connect

- Does not return until TCP has successfully established a connection at which application is free to begin sending data
- Address contains remote machine's address

Client-Serve Model with TCP

In practice

- The client usually specifies only remote participant's address and let's the system fill in the local information
- Whereas a server usually listens for messages on a well-known port
- A client does not care which port it uses for itself, the OS simply selects an unused one

Client-Serve Model with TCP

Once a connection is established, the application process invokes two operation

```
int send (int socket, char *msg, int msg_len,
                                        int flags)

int recv (int socket, char *buff, int buff_len,
                                          int flags)
```

Example Application: Client

```
 1 #include <stdio.h>
 2 #include <sys/types.h>
 3 #include <sys/socket.h>
 4 #include <netinet/in.h>
 5 #include <netdb.h>
 6
 7 #define SERVER_PORT 5432
 8 #define MAX_LINE 256
 9
10 int main(int argc, char * argv[])
11 {
12         FILE *fp;
13         struct hostent *hp;
14         struct sockaddr_in sin;
15         char *host;
16         char buf[MAX_LINE];
17         int s;
18         int len;
19         if (argc==2) {
20                 host = argv[1];
21         }
22         else {
23                 fprintf(stderr, "usage: simplex-talk host\n");
24         exit(1);
25         }
```

Example Application: Client

```
 1          /* translate host name into peer's IP address */
 2         hp = gethostbyname(host);
 3         if (!hp) {
 4                 fprintf(stderr, "simplex-talk: unknown host: %s\n", host);
 5                 exit(1);
 6         }
 7         /* build address data structure */
 8         bzero((char *)&sin, sizeof(sin));
 9         sin.sin_family = AF_INET;
10         bcopy(hp->h_addr, (char *)&sin.sin_addr, hp->h_length);
11         sin.sin_port = htons(SERVER_PORT);
12         /* active open */
13         if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
14                 perror("simplex-talk: socket");
15                 exit(1);
16         }
17         if (connect(s, (struct sockaddr *)&sin, sizeof(sin)) < 0) {
18                 perror("simplex-talk: connect");
19                 close(s);
20                 exit(1);
21         }
22         /* main loop: get and send lines of text */
23         while (fgets(buf, sizeof(buf), stdin)) {
24                 buf[MAX_LINE-1] = '\0';
25                 len = strlen(buf) + 1;
26                 send(s, buf, len, 0);
27         }
28 }
```

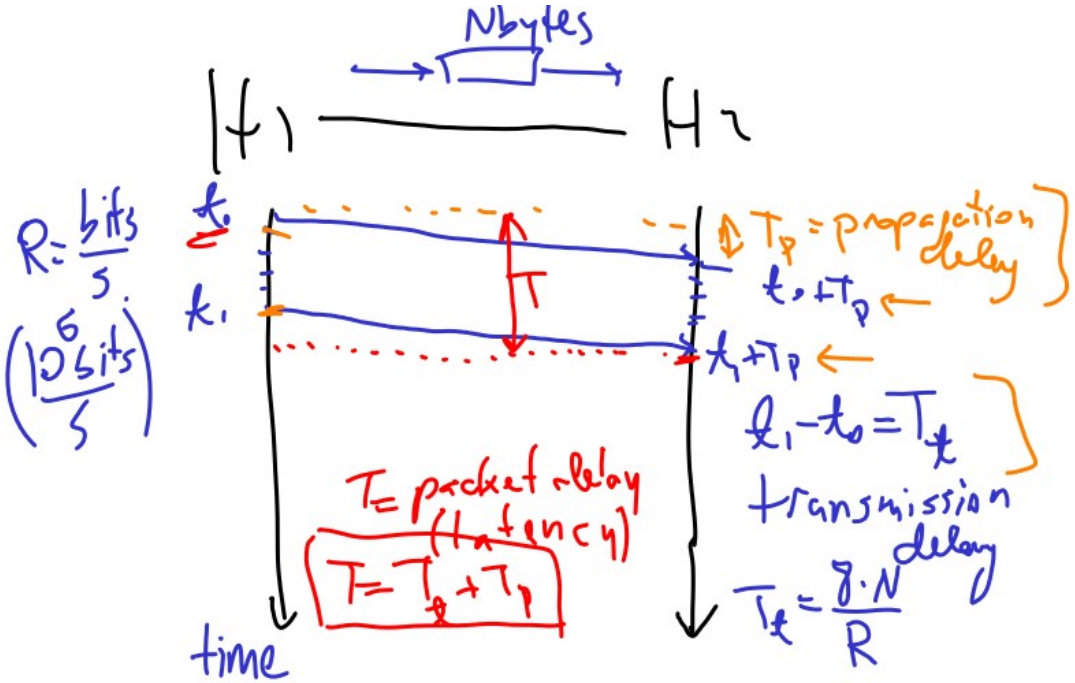
Example Application: Server

```
 1 #include <stdio.h>
 2 #include <sys/types.h>
 3 #include <sys/socket.h>
 4 #include <netinet/in.h>
 5 #include <netdb.h>
 6 #define SERVER_PORT 5432
 7 #define MAX_PENDING 5
 8 #define MAX_LINE 256
 9
10 int main()
11 {
12         struct sockaddr_in sin;
13         char buf[MAX_LINE];
14         int len;
15         int s, new_s;
16         /* build address data structure */
17         bzero((char *)&sin, sizeof(sin));
18         sin.sin_family = AF_INET;
19         sin.sin_addr.s_addr = INADDR_ANY;
20         sin.sin_port = htons(SERVER_PORT);
21
22         /* setup passive open */
23         if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
24                 perror("simplex-talk: socket");
25                 exit(1);
26         }
```

Example Application: Server

```
1         if ((bind(s, (struct sockaddr *)&sin, sizeof(sin))) < 0) {
2                 perror("simplex-talk: bind");
3                 exit(1);
4         }
5         listen(s, MAX_PENDING);
6         /* wait for connection, then receive and print text */
7         while(1) {
8                 if ((new_s = accept(s, (struct sockaddr *)&sin, &len)) < 0) {
9                       perror("simplex-talk: accept");
10                      exit(1);
11                }
12                while (len = recv(new_s, buf, sizeof(buf), 0))
13                      fputs(buf, stdout);
14                close(new_s);
15        }
16 }
```

Network Performance



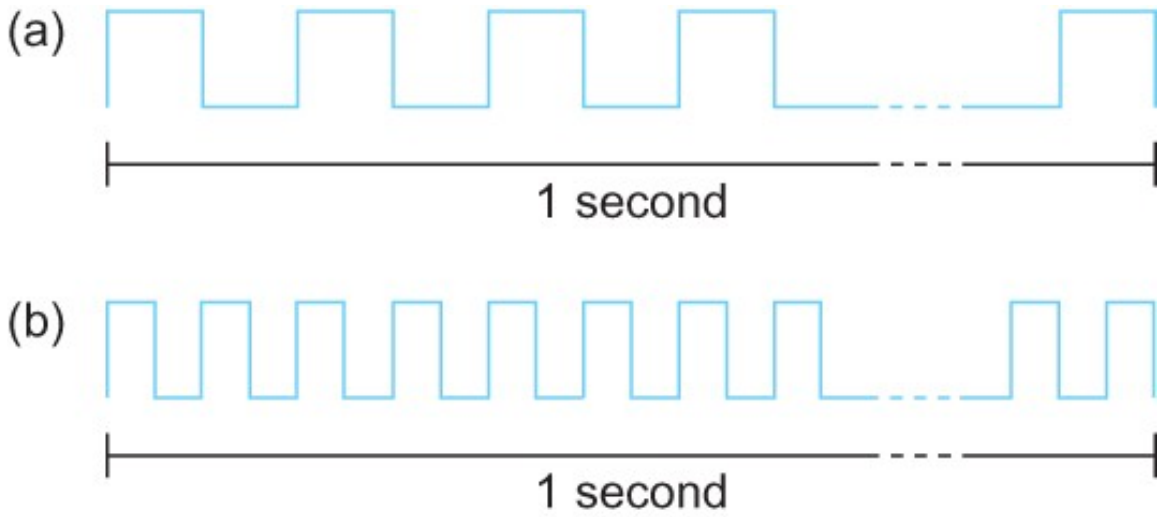
d = distance H1 → H2

V = signal propagation speed

$$T_p = \frac{d}{V}$$

$$T_t = \frac{8 \cdot N}{R}$$

Data Rate vs. Bit Duration



Bits transmitted at a particular data rate can be regarded as having some ‘width’ - the bit duration:

(a) bits transmitted at 1Mbps (each bit 1 μs wide);

(b) bits transmitted at 2Mbps (each bit 0.5 μs wide).

Network Performance

- Common term: Bandwidth – confusing, two meanings:
  - Width of the frequency band = EM spectrum used (e.g. 1 MHz)
  - Number of bits per second that can be transmitted over a communication link: also called data rate or throughput
- We prefer **Data Rate** (R)
- 1 Mbps: 1×10^6 bits/second
- 1×10^{-6} seconds to transmit each bit or imagine that a timeline, now each bit occupies 1 micro second space.
  - Called **Bit Duration** (T_b)
- On a 2 Mbps link the width is 0.5 micro second.
- Data Rate = 1 / Bit Duration

Performance

$$T = T_p + T_t + T_q$$

- Total packet latency = Propagation + transmit + queue
- Propagation = distance/ signal propagation speed: $T_p = \frac{d}{v}$
- Transmit = size/bandwidth : $T_t = 8\,x\,\frac{N}{R}$

(N=packet size in bytes, R=data rate in bps.)

- One bit transmission => propagation is important
- Large bytes transmission => bandwidth is important

Additional Relationships

- Data rate is the inverse of bit duration:

$$R = \frac{1}{T_b}$$

- Propagation distance is the product of the signal speed (v or c – for light) and the propagation delay:

$$d = v \times T_p$$

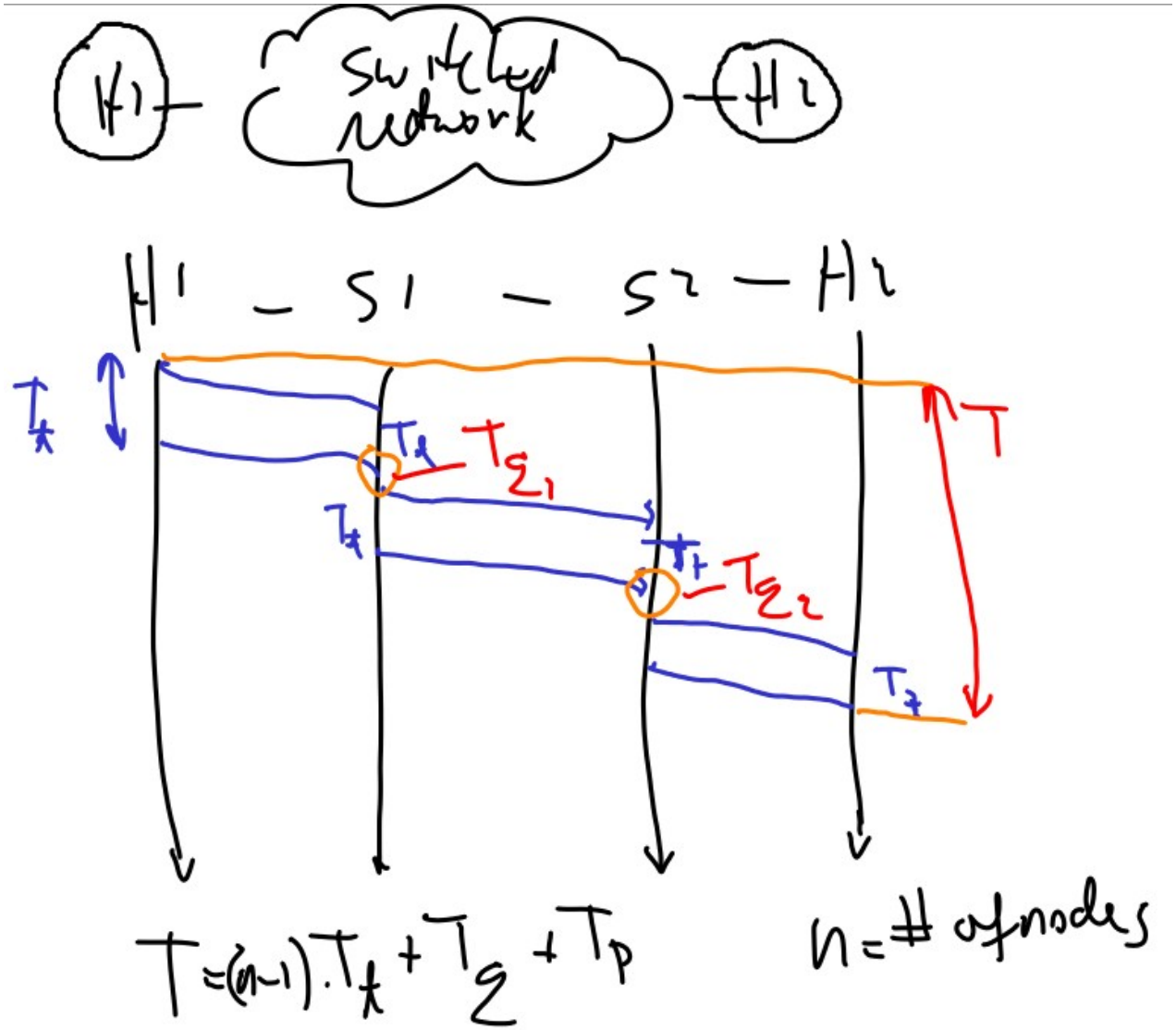
- Round trip time is twice the propagation delay:

$$RTT = 2T_p$$

Units for Data Rate vs. Capacity

- Easy confusion: kilobit vs. kilobit/s or kilo byte (KB) vs. kilobyte per second.
- Units for **capacity (memory)** use power of 2: kilo=2^{10}, mega=2^{20}, giga=2^{30}, etc.
- Units for **data rate** use power of 10: kilo=10^3, mega=10^6, giga=10^9, etc.
- Don’t forget conversion: 1 byte (1B)=8 bits.

Packet Delay In a Switched Network

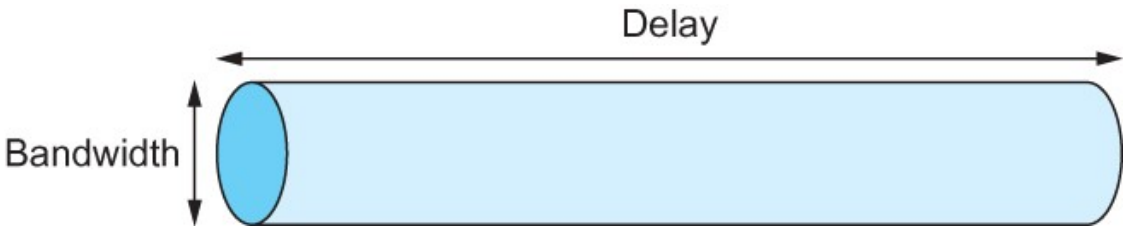


Delay X Bandwidth

- We think the channel between a pair of processes as a hollow pipe
- Latency (delay) length of the pipe and bandwidth the width of the pipe
- Delay of 50 ms and bandwidth of 45 Mbps

⇒ 50×10^{-3} seconds x 45×10^{6} bits/second

⇒ 2.25×10^{6} bits = 280 KB data.



Network as a pipe

Delay X Bandwidth

- Relative importance of bandwidth and latency depends on application
  - For large file transfer, bandwidth is critical
  - For small messages (HTTP, NFS, etc.), latency is critical
  - Variance in latency (jitter) can also affect some applications (*e.g.*, audio/video conferencing)

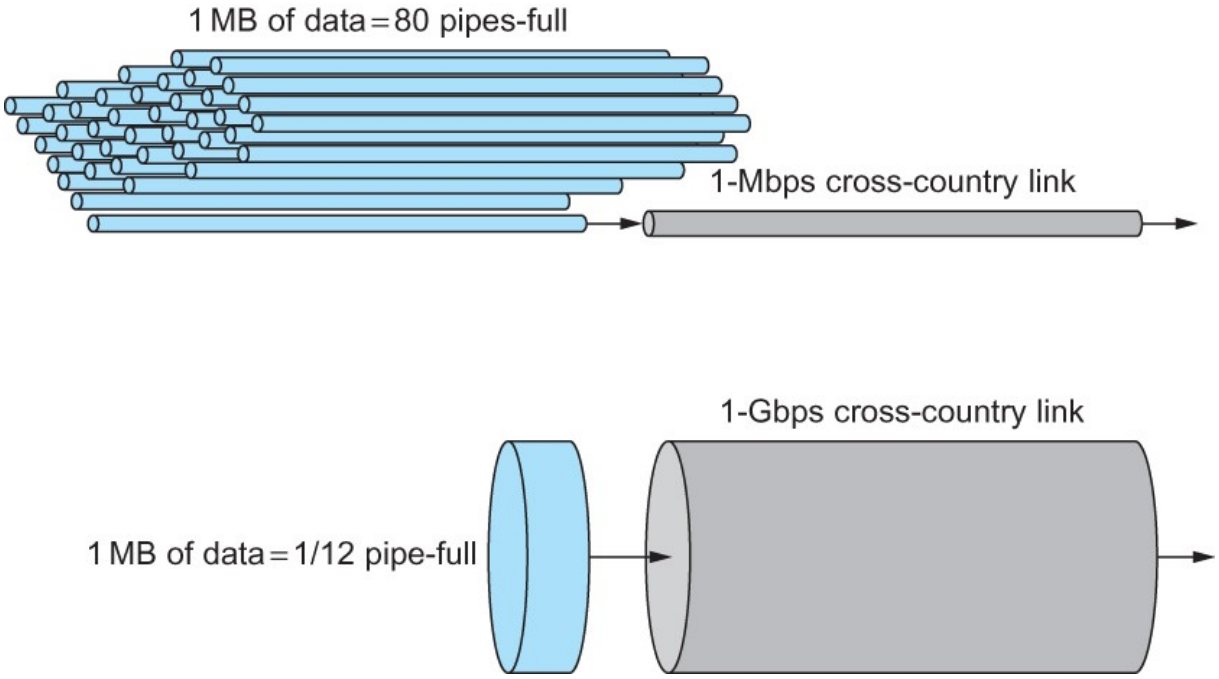
Delay X Bandwidth

- How many bits the sender must transmit before the first bit arrives at the receiver if the sender keeps the pipe full
- Takes another one-way latency to receive a response from the receiver
- If the sender does not fill the pipe—send a whole delay × bandwidth product's worth of data before it stops to wait for a signal—the sender will not fully utilize the network

Delay X Bandwidth

- Infinite bandwidth
  - RTT dominates
  - Throughput = TransferSize / TransferTime
  - TransferTime = RTT + 1/Bandwidth x TransferSize
- Its all relative
  - 1-MB file to 1-Gbps link looks like a 1-KB packet to 1-Mbps link

Relationship between bandwidth and latency



A 1-MB file would fill the 1-Mbps link 80 times, but only fill the 1-Gbps link 1/12 of one time

Solving Performance Problems

- Draw a diagram to clarify the problem.
- **ALWAYS** write symbolic equations and show the derivation of the solution. Write the symbolic solution:
  - e.g. If T_t and R are given, what is N ?
    - Answer: we know that T_t=8*N/R, so N=R T_t/ 8
  - WHY symbolic formulas ?? So that you:
    - Demonstrate that you understand the underlying mechanism, e.g. relationship between data rate, packet size, and transmission time. **You are graded for this!**
    - Get partial credit even in case of arithmetic mistakes.
  - Use MS Excel or some other spreadsheet to compute the numeric solutions. Avoid arithmetic mistakes.

Exercises - #1

Calculate the total time required to transfer a 1.5-MB file in the following cases, assuming an RTT of 80 ms, a packet size of 1 KB data, and an initial 2 × RTT of "handshaking" before data is sent:

(a) The bandwidth is 10 Mbps, and data packets can be sent continuously.

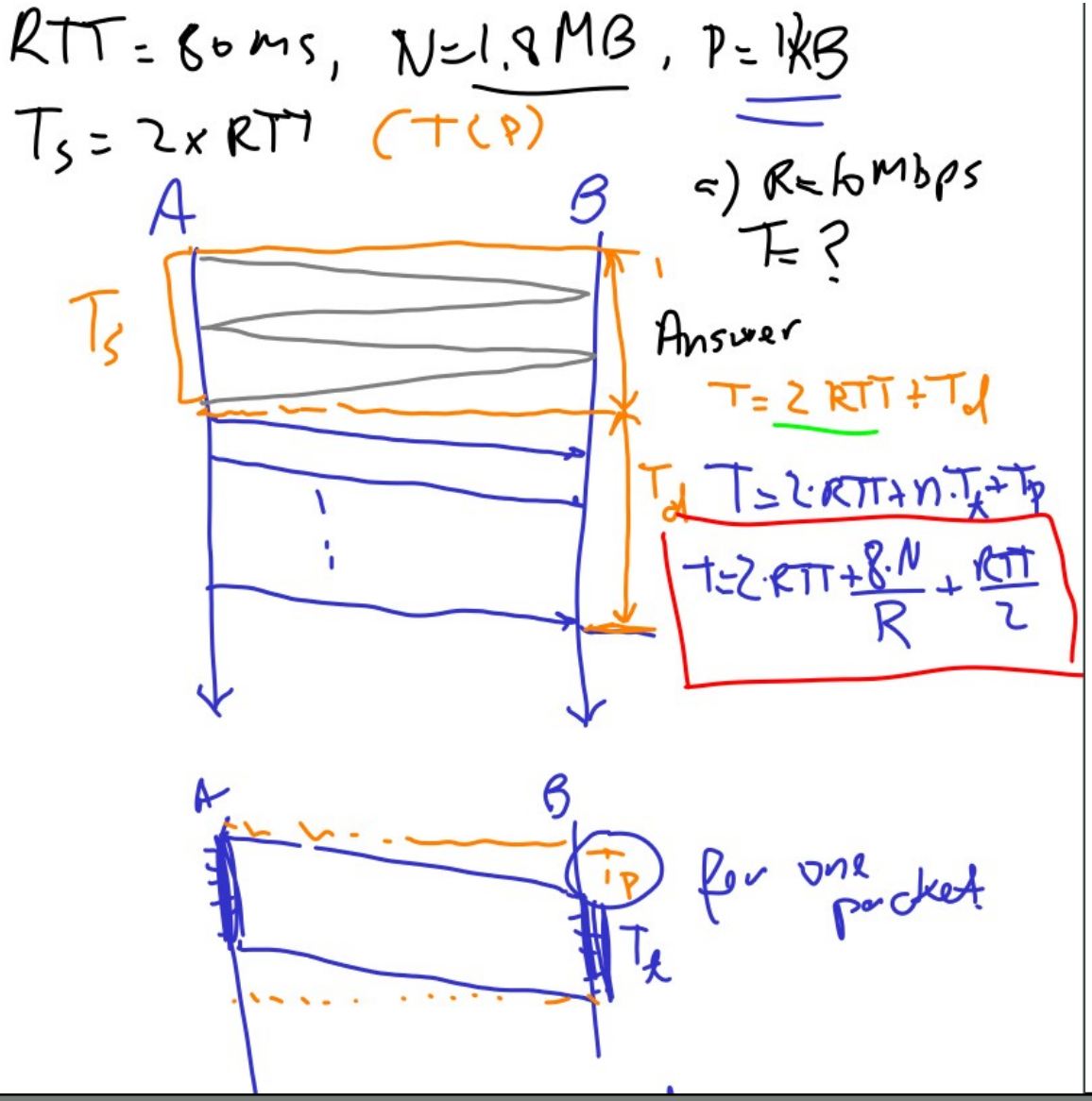
(b) The bandwidth is 10 Mbps, but after we finish sending each data packet we must wait one RTT before sending the next.

(c) The link allows infinitely fast transmit, but limits bandwidth such that only 20 packets can be sent per RTT.

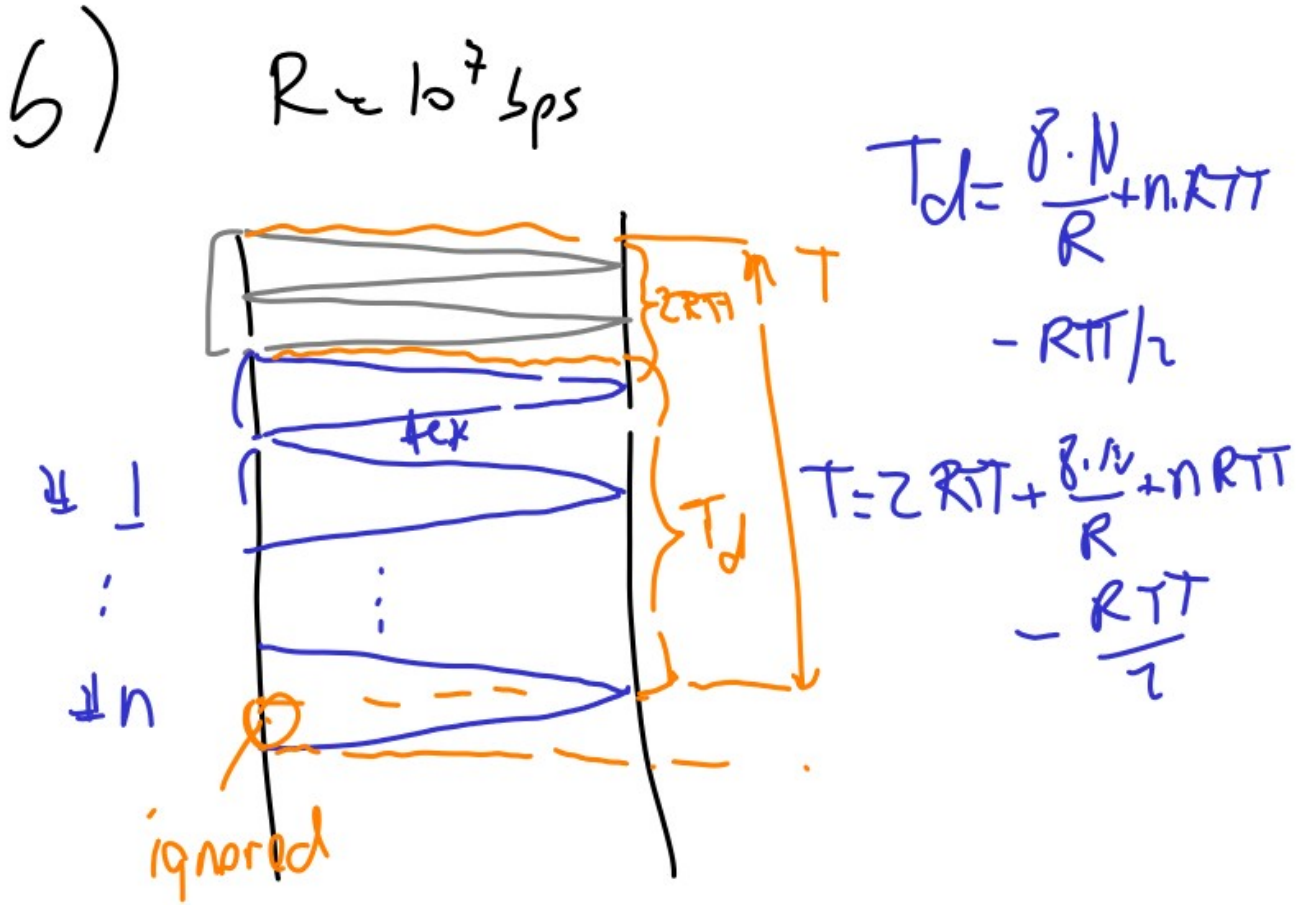
(d) Zero transmit time as in (c), but during the first RTT we can send one packet, during the second RTT we can send two packets, during the third we can send four (2^{3-1}), etc.

(A justification for such an exponential increase will be given in Chapter 6.)

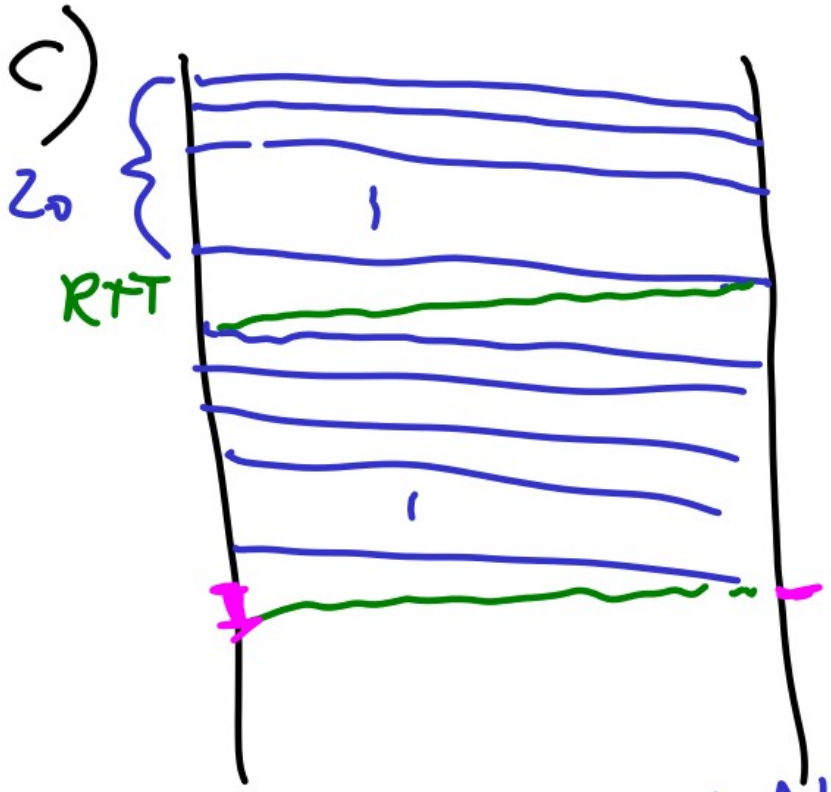
Exercises - #1 a) Solution



Exercises - #1 b) Solution



Exercises - #1 c) Solution



p = pkt size

$T_t = \frac{8N}{R} = 0 \quad (R = \infty)$

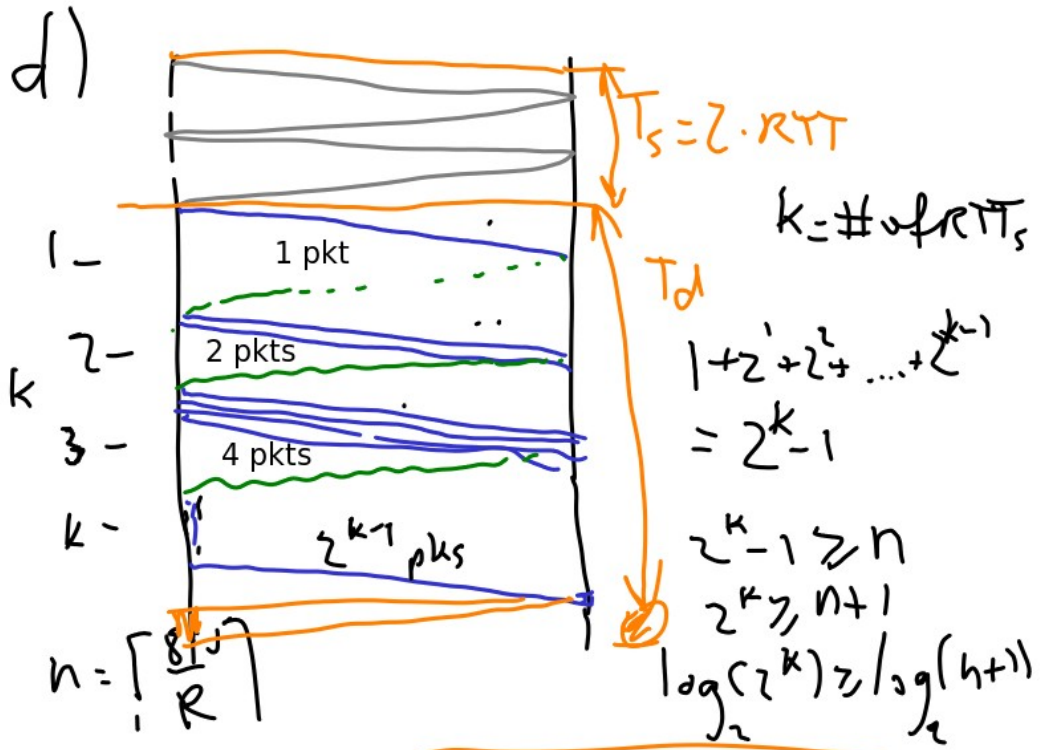
k = # of RTT to wait.

$n = \lceil \frac{N}{p} \rceil$ packets

$\lfloor \frac{N}{2_0 p} \rfloor$ = # of full RTT batches

$T = 2 \times RTT + \lfloor \frac{N}{2_0 p} \rfloor \times RTT + 0.5\,RTT.$

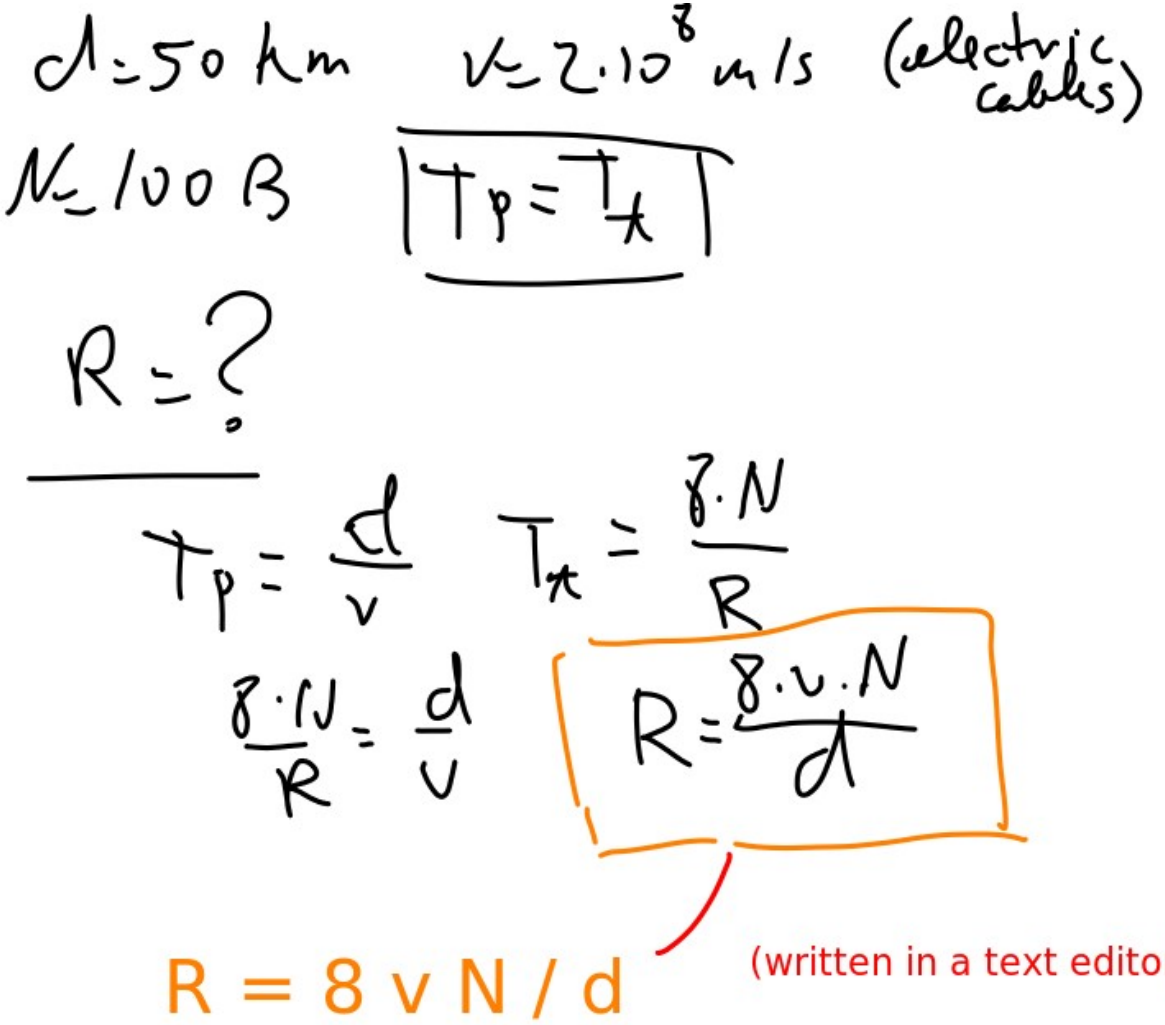
Exercises - #1 d) Solution



Exercises - #2

Consider a point-to-point link 50 km in length. At what bandwidth would propagation delay (at a speed of 2×10^8 m/s) equal transmit delay for 100-byte packets? What about 512-byte packets?

Exercise 2 Solution



Exercises - #3

How “wide” is a bit on a 10-Gbps link? How long is a bit in copper wire, where the speed of propagation is 2.3×10^8 m/s?

Exercise 3 Solution

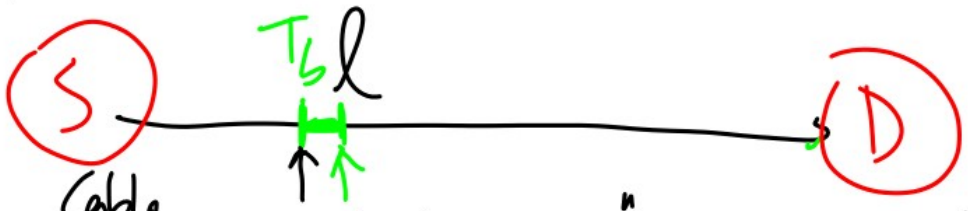
$R = 10\ Gbps \quad v = 2.3 \cdot 10^8\ m/s$

bit length / width = ?

↓

bit duration

$T_b = \frac{1}{R} = \frac{1}{10^7\ b/s} = 10^{-7}\ s/bit$ width?



$$l = v \cdot \frac{1}{R}$$

$l = v \cdot T_b$

Exercises - #4

Suppose a 128-kbps point-to-point link is set up between the Earth and a rover on Mars. The distance from the Earth to Mars (when they are closest together) is approximately 55 Gm, and data travels over the link at the speed of light—3×10^8 m/s.

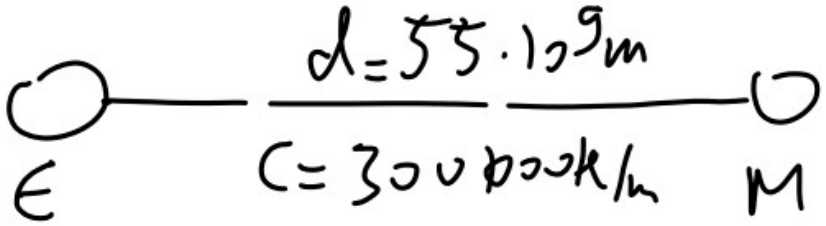
(a) Calculate the minimum RTT for the link.

(b) Calculate the delay × bandwidth product for the link.

(c) A camera on the rover takes pictures of its surroundings and sends these to Earth. How quickly after a picture is taken can it reach Mission Control on Earth? Assume that each image is

5 Mb in size.

Exercise 4 Solution



a) $RTT = 2 \cdot T_p = 2 \cdot \frac{d}{c}$

b) $D \times B = T_p \cdot R$

c) $N = 5MB$ ← B byte, 5 bit

$T = T_t + T_p$

$T = \frac{N}{R} + \frac{d}{c}$

Exercises - #5

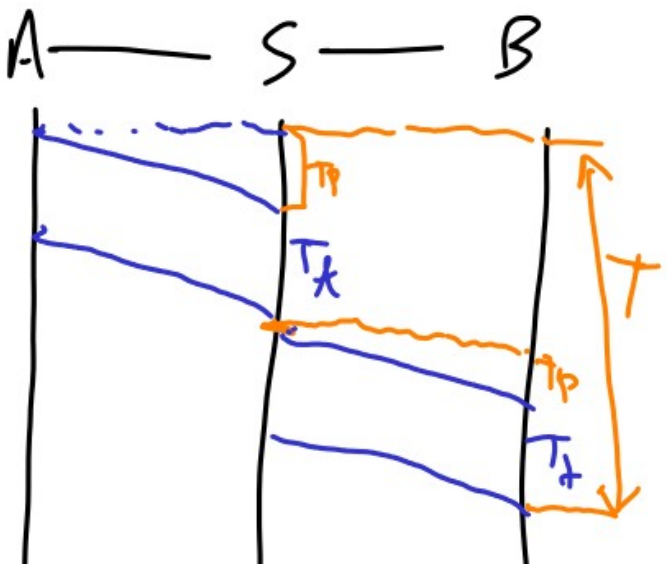
Calculate the latency (from first bit sent to last bit received) for:

(a) 1-Gbps Ethernet with a single store-and-forward switch in the path and a packet size of 5000 bits. Assume that each link introduces a propagation delay of 10 μs and that the switch begins retransmitting immediately after it has finished receiving the packet.

(b) Same as (a) but with three switches.

(c) Same as (b), but assume the switch implements “cut-through” switching; it is able to begin retransmitting the packet after the first 128 bits have been received.

Exercise 5 a) Solution



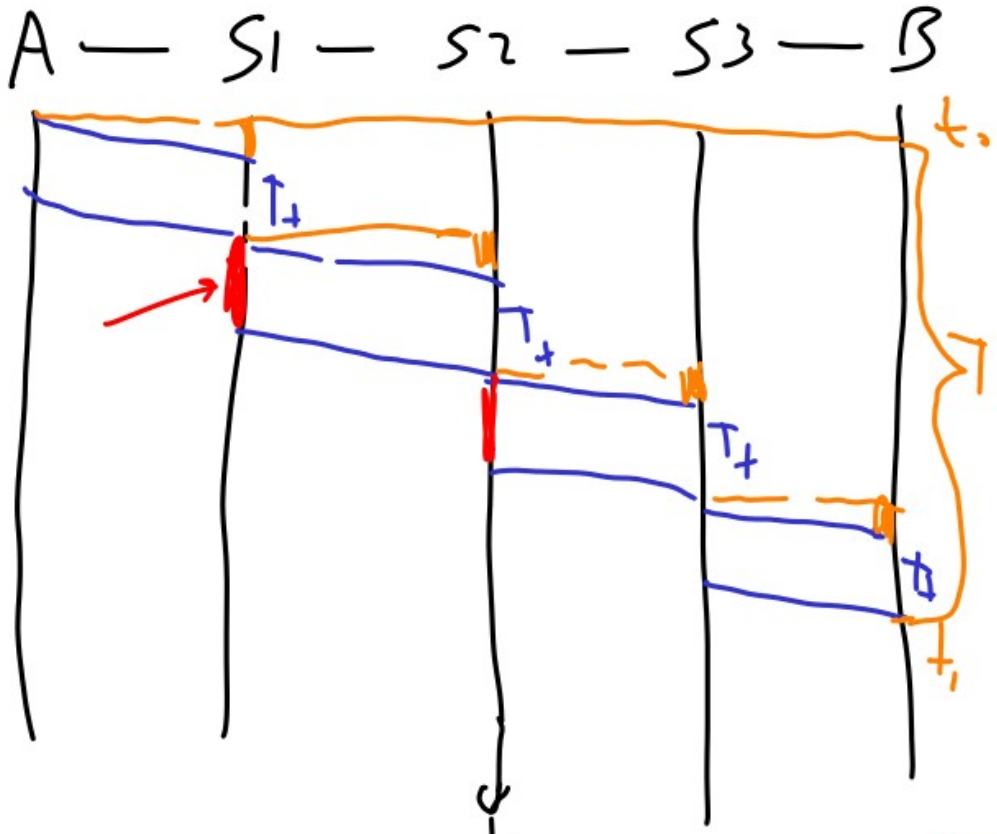
$R = 1\,Gbps$

$N = 5000\,bits$

$T_p = 10\,\mu s$

$T = 2 \cdot T_t + 2 \cdot T_p$

Exercise 5 b) Solution

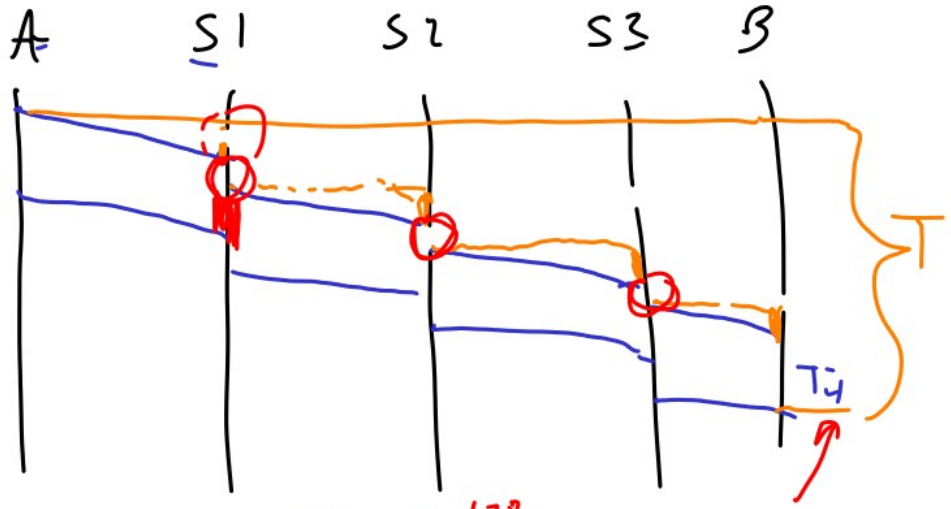


S = # of switches

$T = (S+1)T_t + (S+1)T_p$

$T_t = \frac{N}{R}$

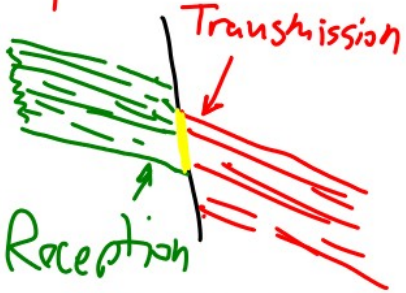
Exercise 5 c) Solution



$T = (S+1)T_p + T_t + S \cdot \frac{128}{R}$

Cut-through switch

S = number of switches



significant overlap of Reception and transmission

reduce overall delay from T at switches

Exercises - #6

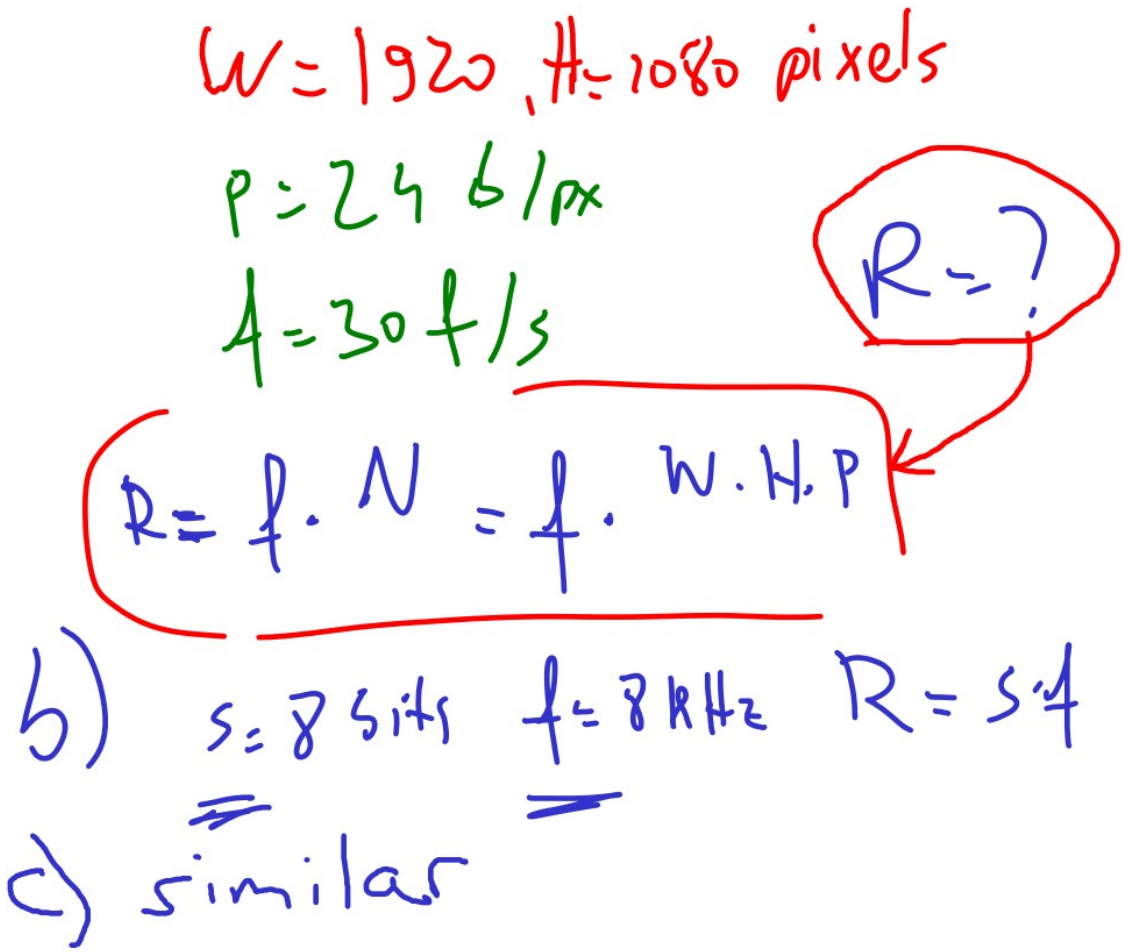
For the following, as in the previous problem, assume that no data compression is done. Calculate the bandwidth necessary for transmitting in real time:

(a) High-definition video at a resolution of 1920 × 1080, 24 bits/pixel, 30 frames/second.

(b) POTS (plain old telephone service) voice audio of 8-bit samples at 8 KHz.

(c) GSM mobile voice audio of 260-bit samples at 50 Hz.

Exercise 6 Solution



Exercises - #7

Suppose that a certain communications protocol involves a per-packet overhead of 50 bytes for headers and framing. We send 1 million bytes of data using this protocol; however, one data byte is corrupted and the entire packet containing it is thus lost. Give the total number of overhead + loss bytes for packet data sizes of 1000, 10,000, and 20,000 bytes. Which size is optimal?

Exercise 7 Solution

#7 H = overhead = 50 B

N = 10^6 Bytes

M = total # of bytes transmitted

p = 1000 B, 10^4 B ... $2 \cdot 10^3$ B

n = packet count

$n = \lceil \frac{N}{p} \rceil$ total overhead = h

$h = n \cdot H$ ceiling

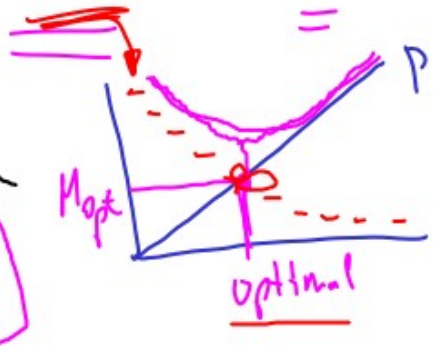
retransmitted pkt

$M = h + N + p = \lceil \frac{N}{p} \rceil H + N + p$

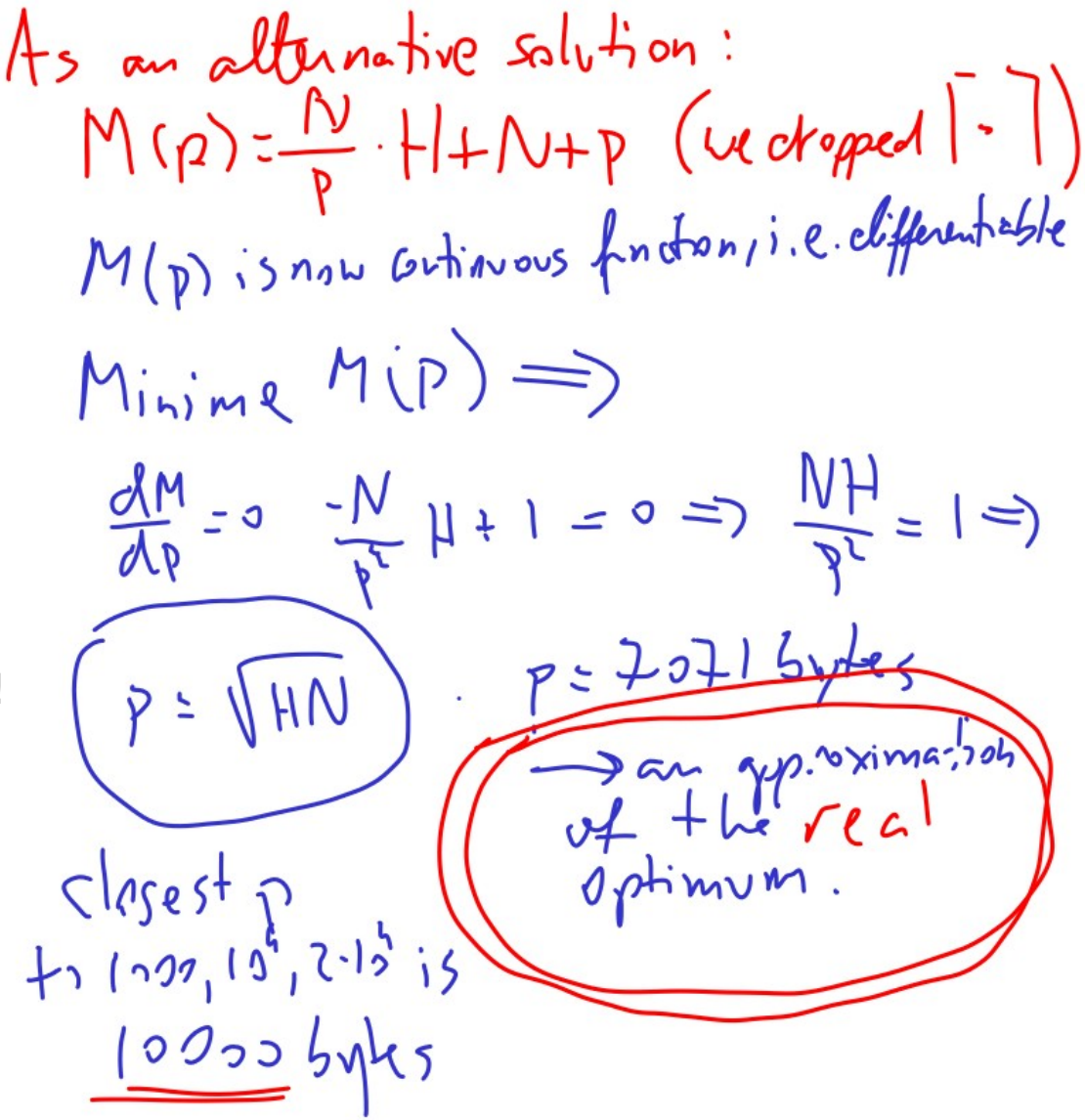
M = M(p)

Plug in numbers ...

p = 10000 wins!



Exercise 7 Solution



Geometric mean!

Summary

- We have identified what we expect from a computer network
- We have defined a layered architecture for computer network that will serve as a blueprint for our design
- We have discussed the socket interface which will be used by applications for invoking the services of the network subsystem
- We have discussed two performance metrics using which we can analyze the performance of computer networks