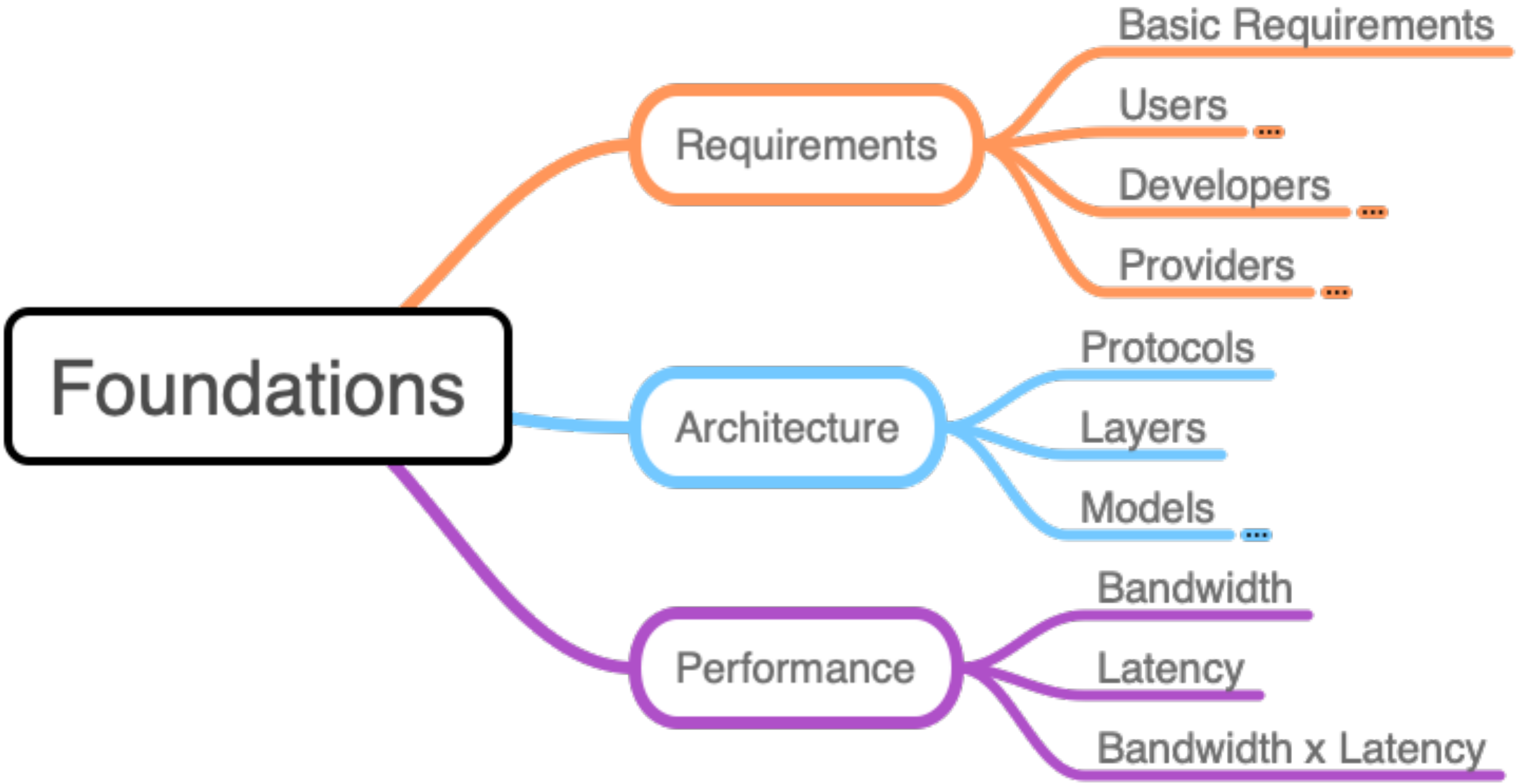
Foundations

COS 460 - Computer Networks
University of Southern Maine
Fall 2019


Foundations
Requirements
Basic Requirements
Users
Developers
Providers
Architecture
Protocols
Layers
Models
Performance
Bandwidth
Latency
Bandwidth x Latency

Goals

Do that thing Computer Scientists do…

1. **Reflect on and collect requirements**
   (the basis of our "network problem")

   Requirements

2. **Introduce the idea of Architecture, Layers, and Protocols**
   (apply structured problem solving)

   Architecture

3. **Define performance metrics for network**
   (analyze our results)

   Performance

“How did I get here?”

–Talking Heads

Applications

The Web

Based on Hypertext Transfer Protocol (HTTP)

Designed for marked up text, Hypertext Markup Language (HTML)

Includes linking, descriptive formatting, style, images, video, …

Incredibly flexible, often used by higher-level applications

Applications

The Web

Email

The first *killer app* for the Internet

- Simple Mail Transport Protocol (SMTP)
- Post Office Protocol (POP)
- Internet Message Access Protocol (IMAP)

Now more common to access email via web (HTTP).

Applications

The Web

Email

File Storage

Even before *the cloud* there was network storage

- Network File System (NFS)
- Common Internet File System (CIFS)
- IP Small Computer System Interconnect (iSCSI)

…and of course, over HTTP these days

Applications

The Web

Email

File Storage

Multimedia

Often hidden behind web browser or media player

Motion Picture Experts Group (MPEG) is most common

- Audio - Layers 1, 2, 3 (mp3)
- Images - JPEG
- Video - MPEG-2, …, MPEG-4

Applications

The Web

Email

File Storage

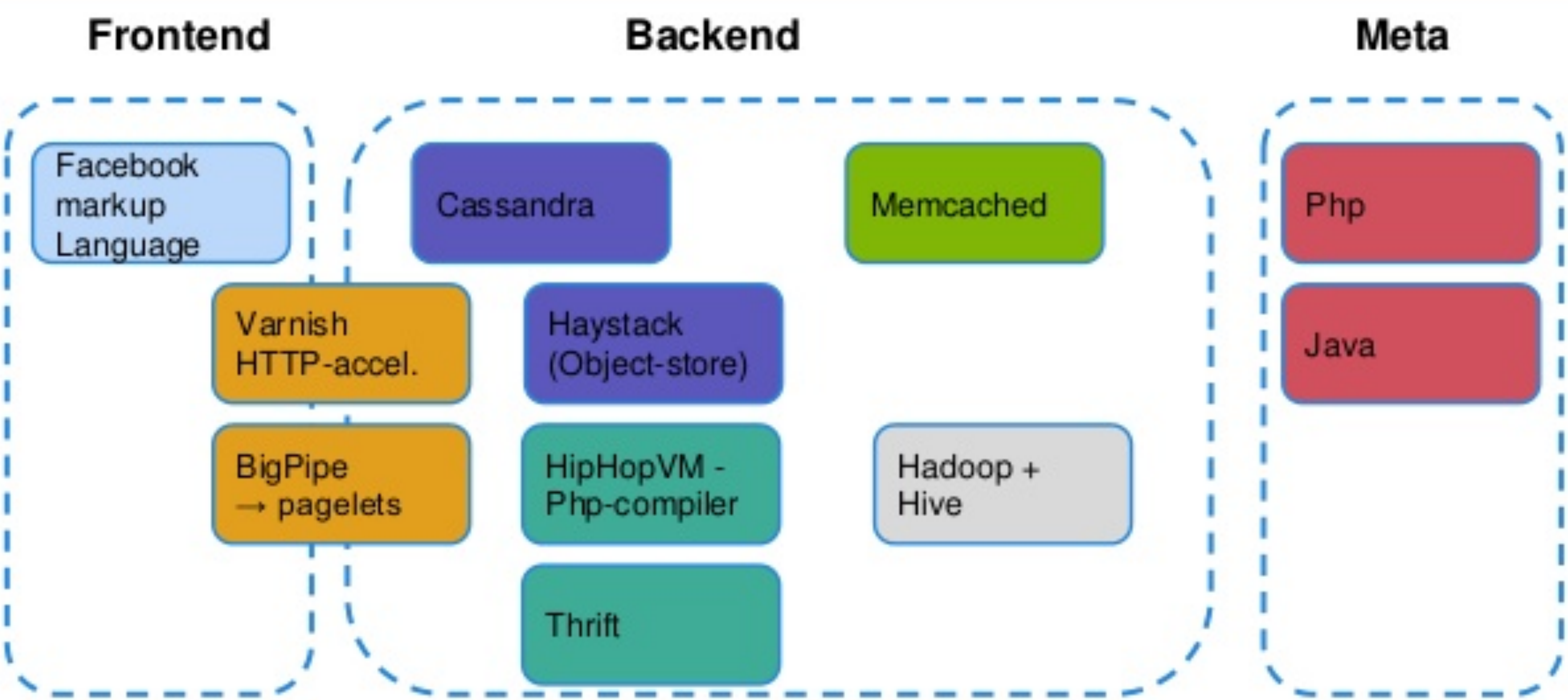
Multimedia

More…

There are too many other applications to enumerate

- Social Media (facebook, twitter, instagram)
- News (reddit, feedly)
- Document and Spreadsheet editing (Google, Microsoft, …)
- Slide presentations (Google, Microsoft, …)

Facebook



Source: Gerald Maduabuchi(Quora), Phillip Webber

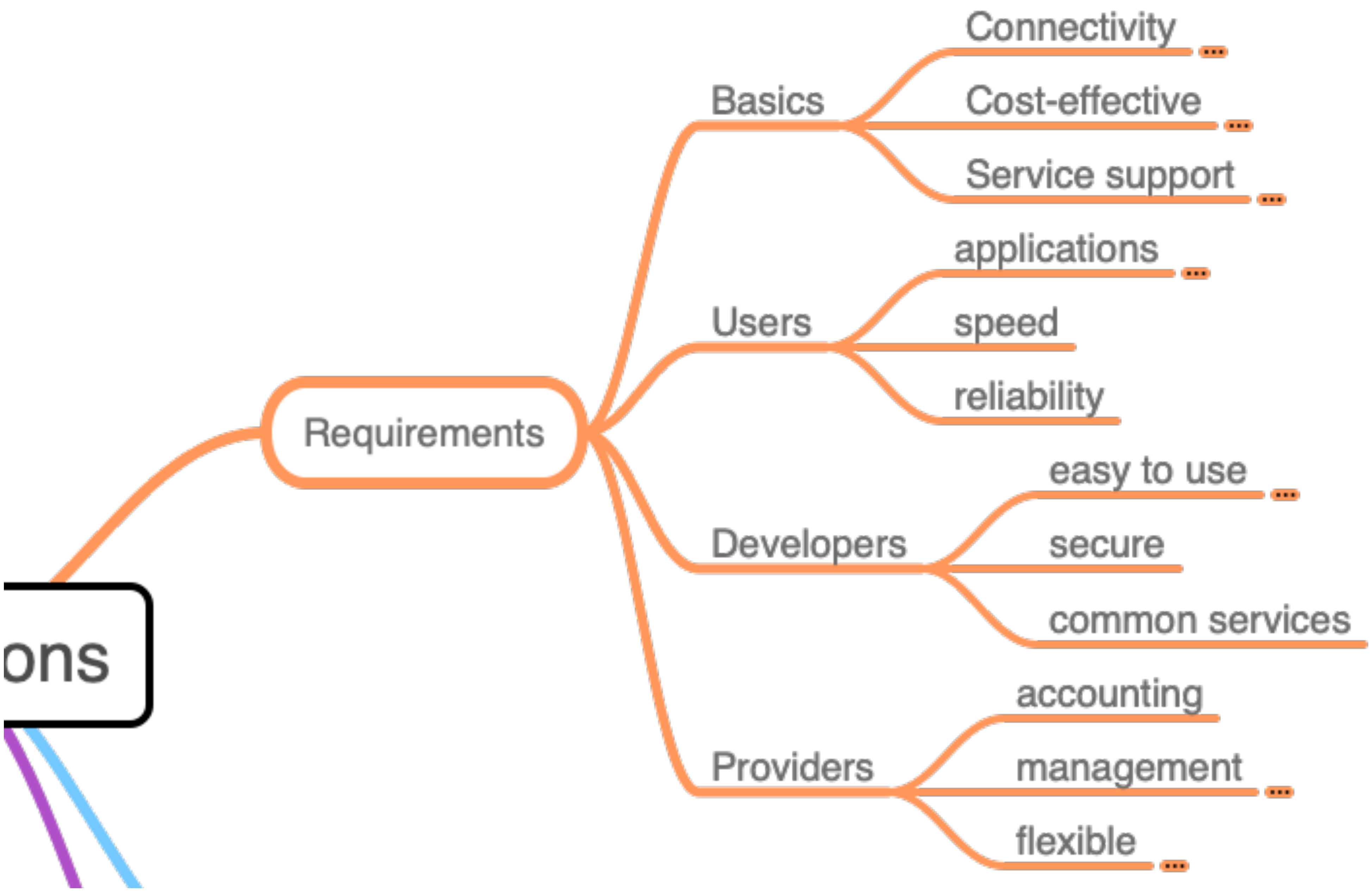
Tech Stacks

In groups of 2-3
Drill down into the “tech stack” of a web application

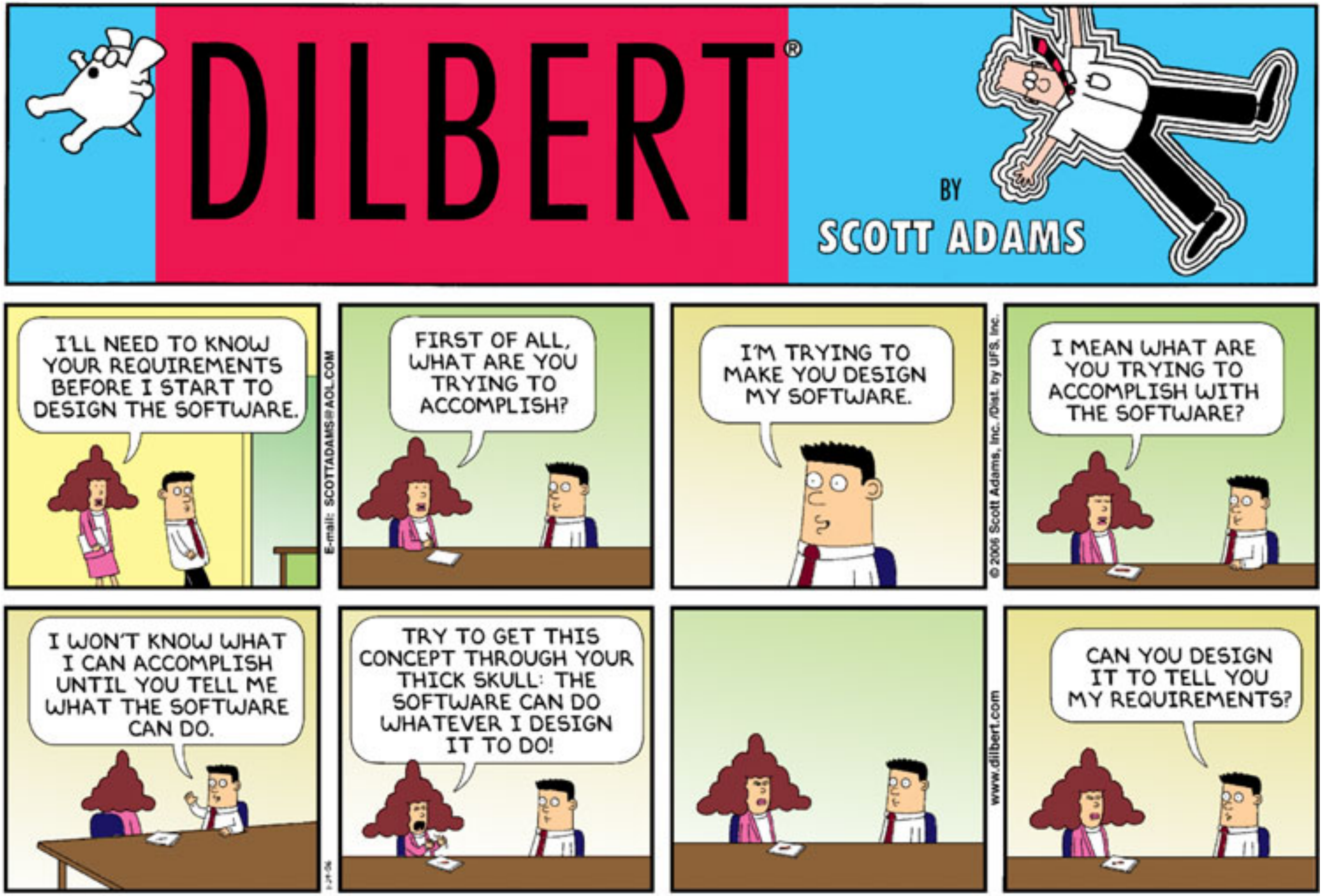
What technologies do they use and what for?

Which are ‘network’ technologies?

5:00



Requirements


DILBERT®
BY SCOTT ADAMS
I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
E-mail: SCOTTADAMS@AOL.COM
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
www.dilbert.com
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

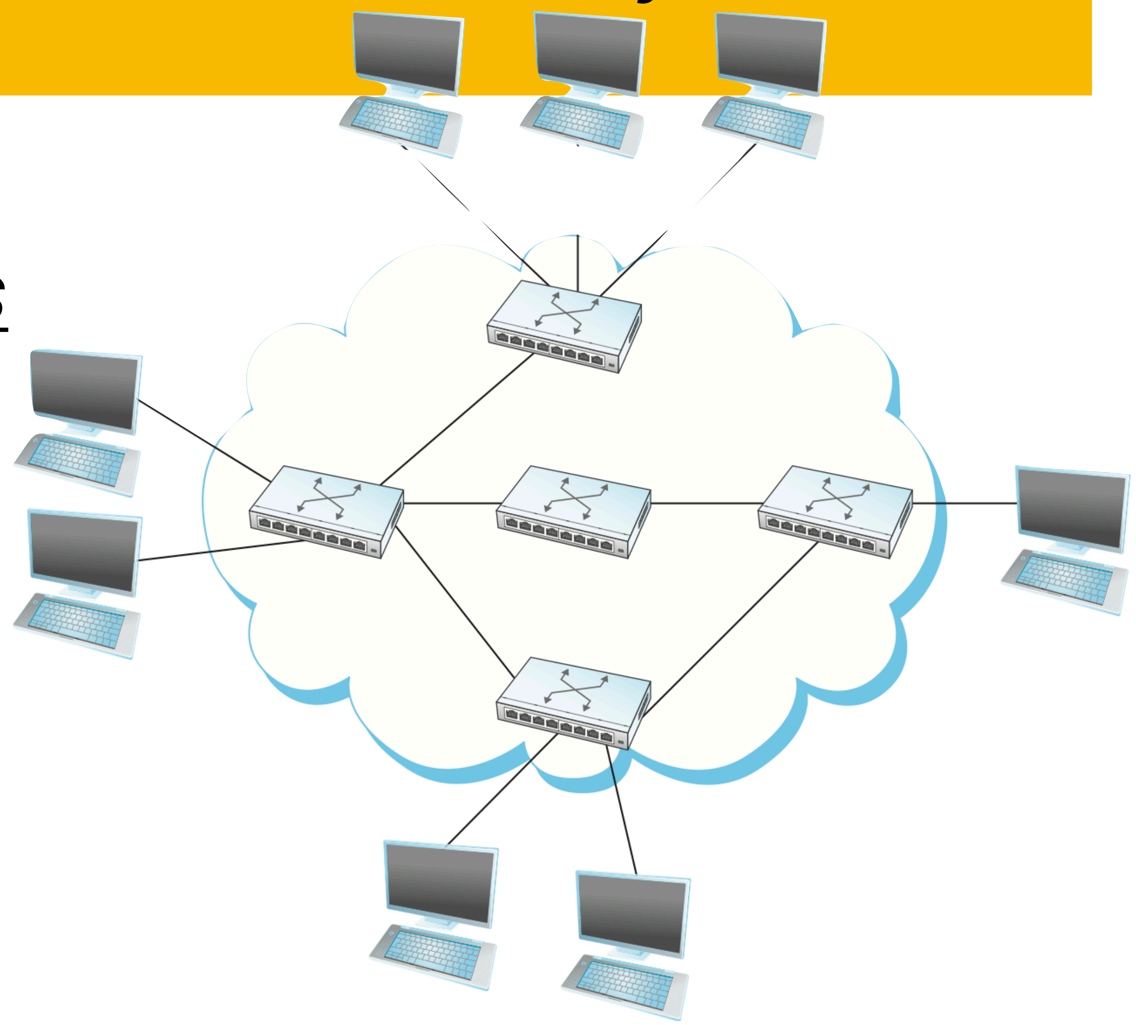
Requirements

Scalable Connectivity

<u>Definition of **Network**</u>

"Two or more nodes connected by a physical link, or two or more networks connected by a node."

- <u>*Nodes*</u> and <u>*Links*</u>
  <u>*Hosts*</u> and <u>*Switches*</u>
- Data in <u>*packets*</u>
- Packet Switched
  Store and Forward

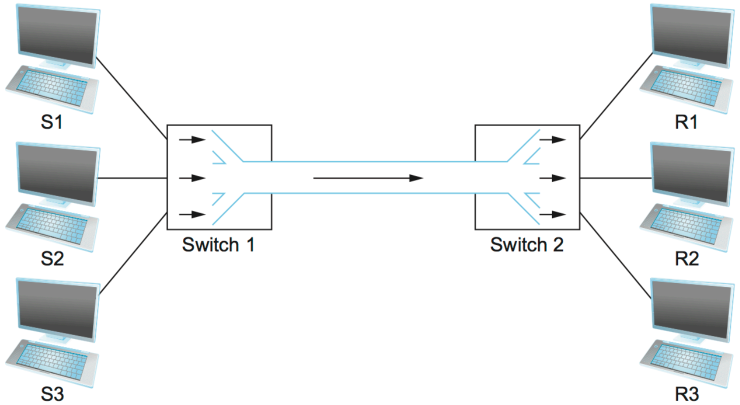


Requirements

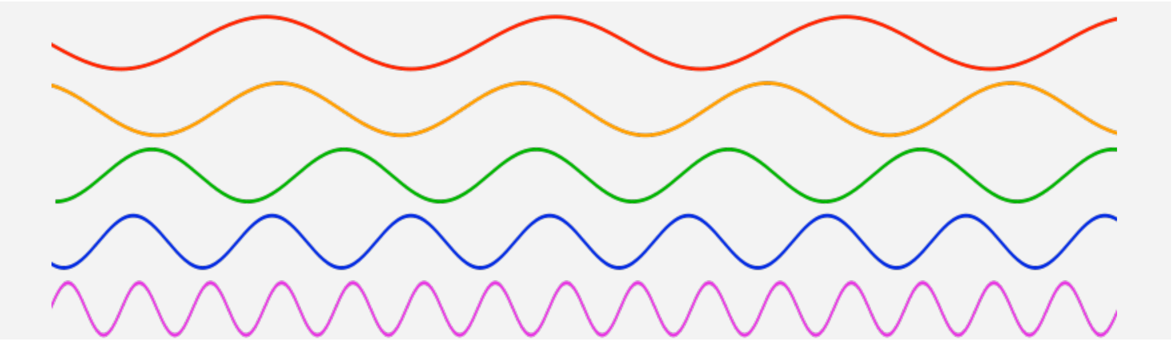
Scalable Connectivity

Cost Effective

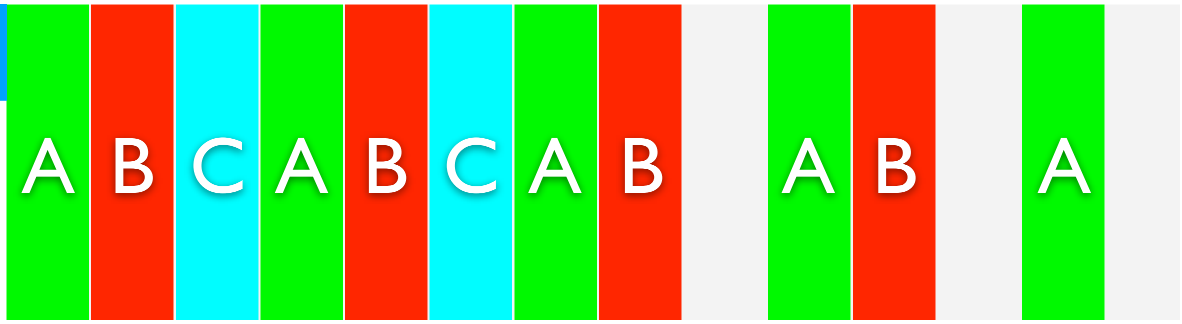
Multiplexing data



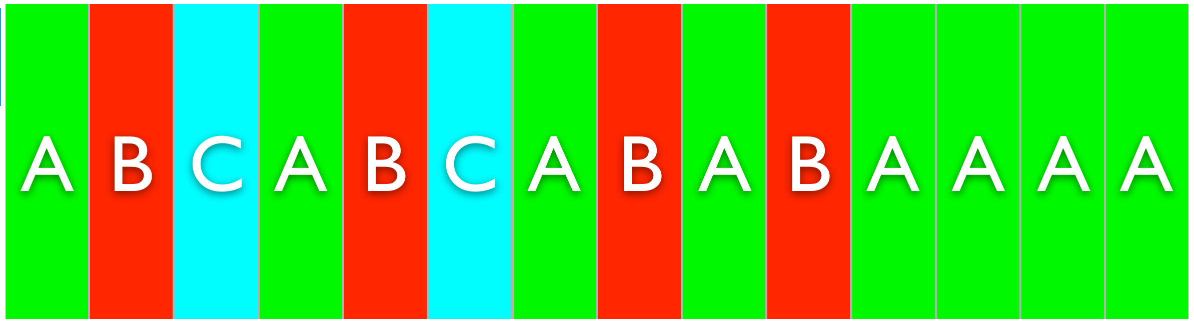
Frequency



Time Division



Packet-Switched

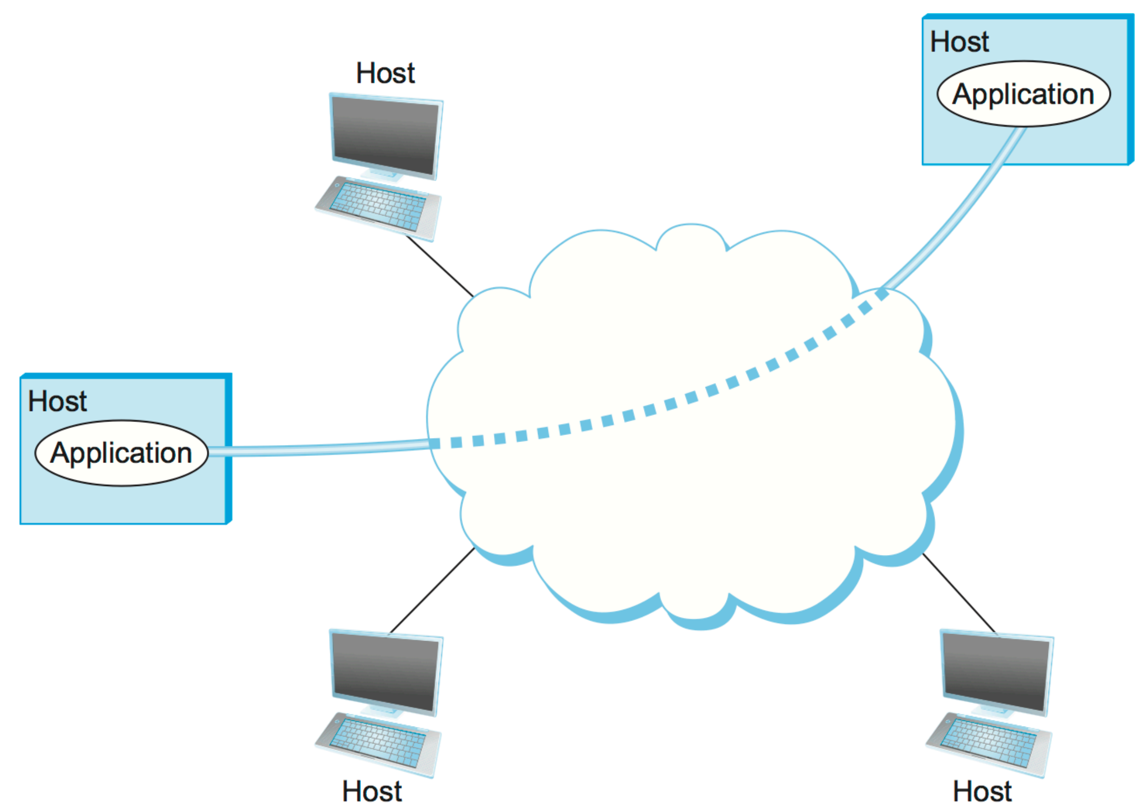


Requirements

Scalable Connectivity

Cost Effective

Support Common Services



Define **useful channels** that understands the **application needs** and the **networks ability**.

file transfer — streaming — web browsing

Requirements

Scalable Connectivity

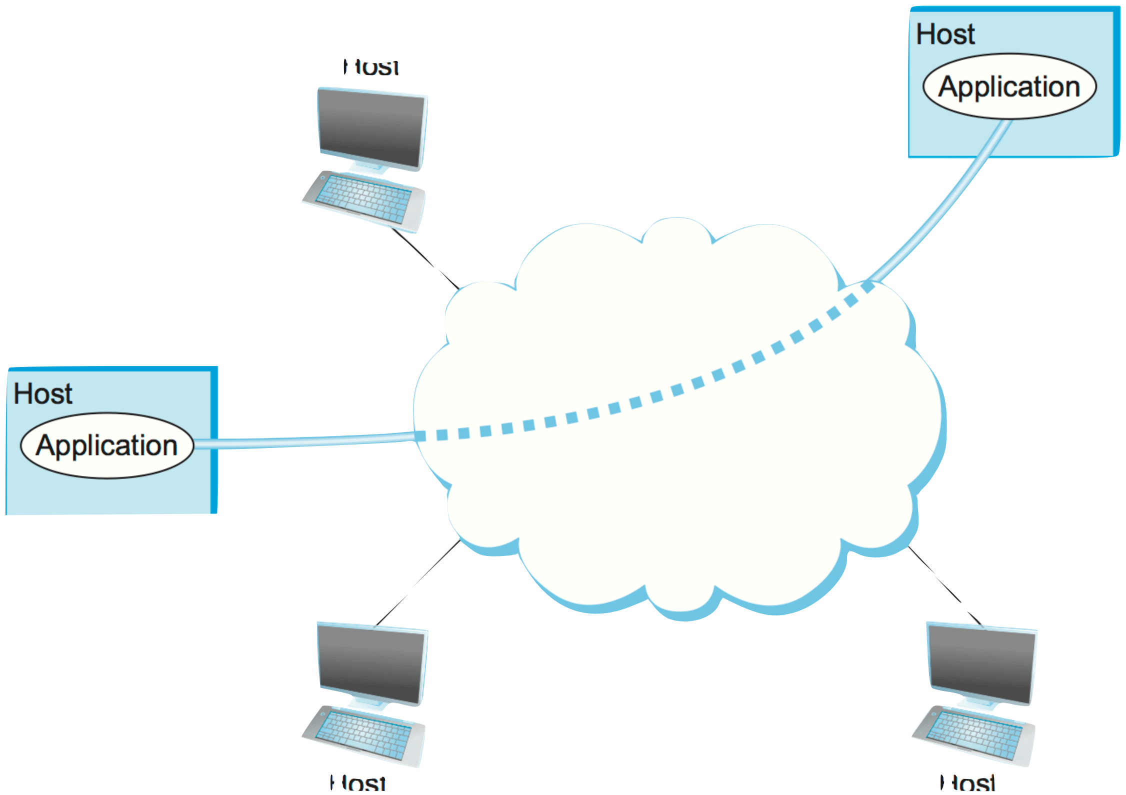
Cost Effective

Support Common Services

Reliable & Manageable

<u>Reliable</u>

Fill in the gap between what the application expects and what the underlying technology can provide.



<u>Manageable</u>

Upgrades, billing, support new applications

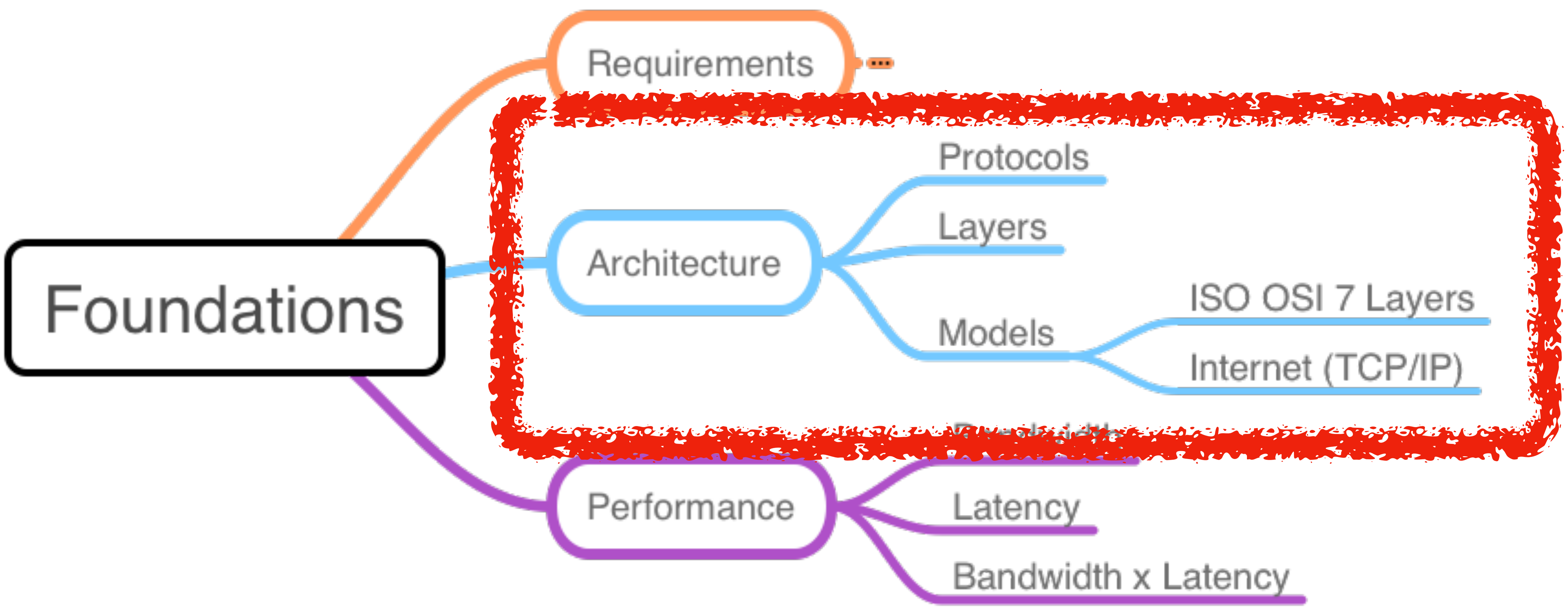
Requirements

Take a couple of minutes to think about and write down

Who’s requirements are we missing?

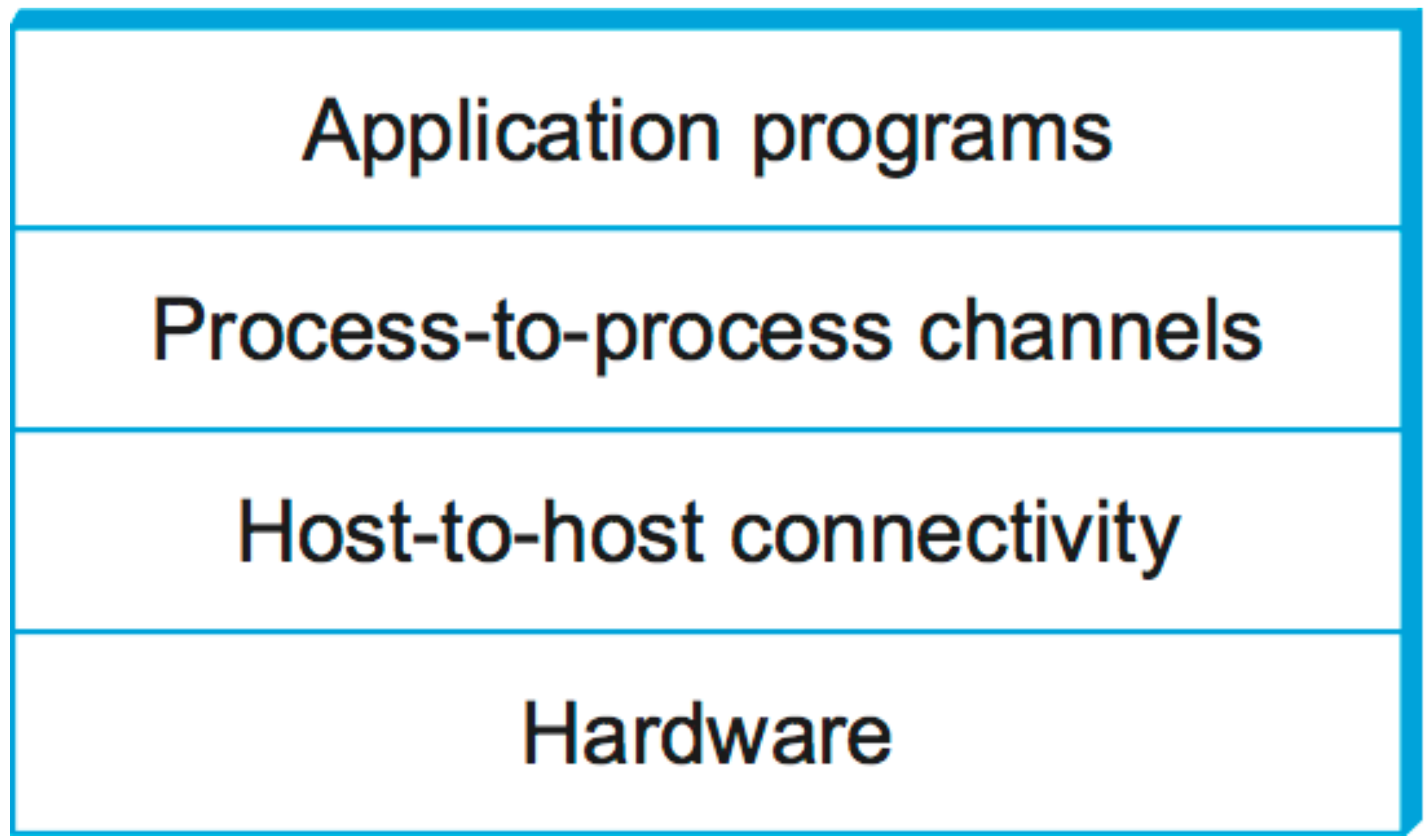
What are they?

Architecture



Architecture

Layers & Protocols

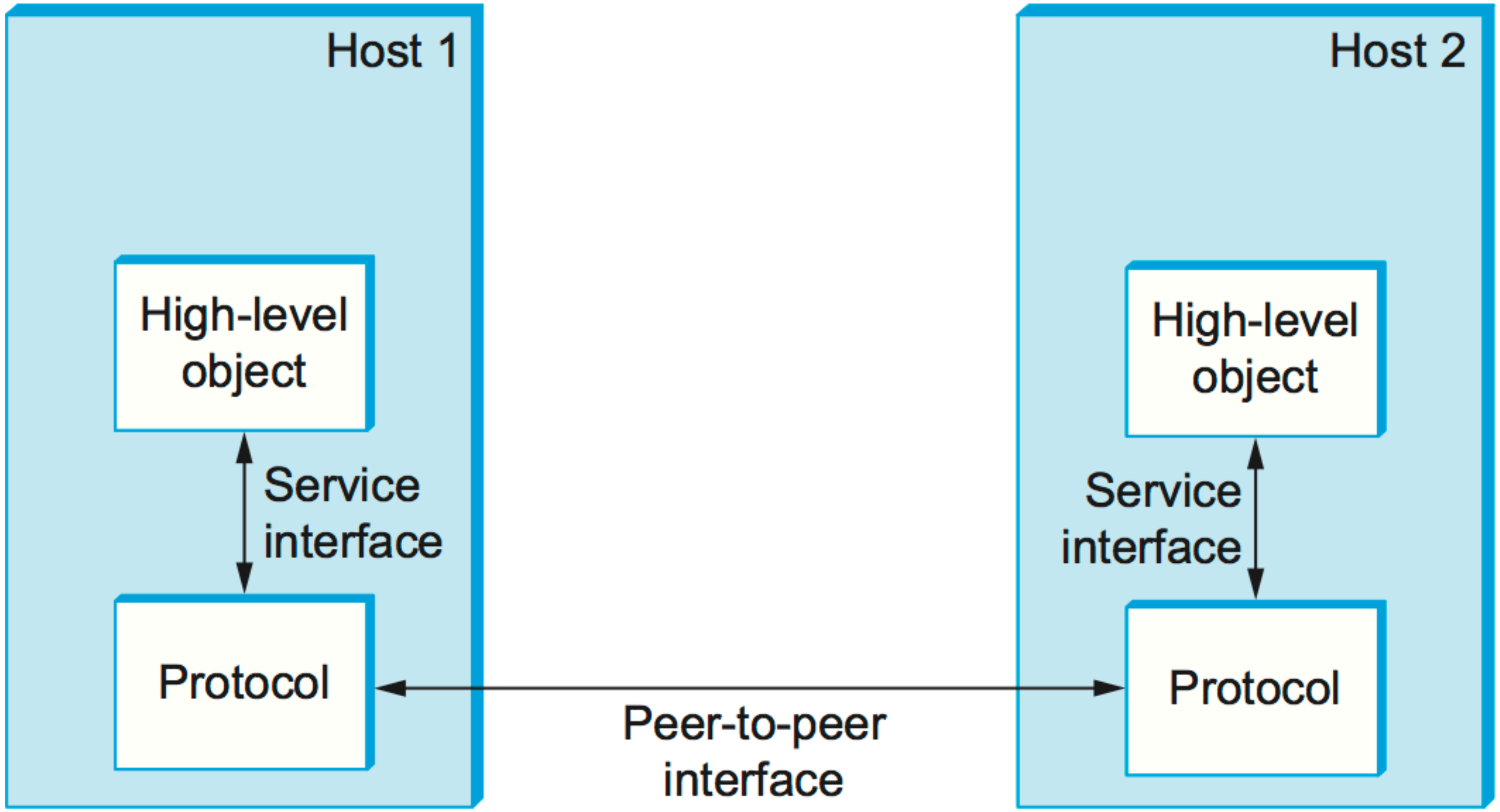


Simplified Layers

Architecture

Layers & Protocols

Service and Peer Interfaces (Protocols)

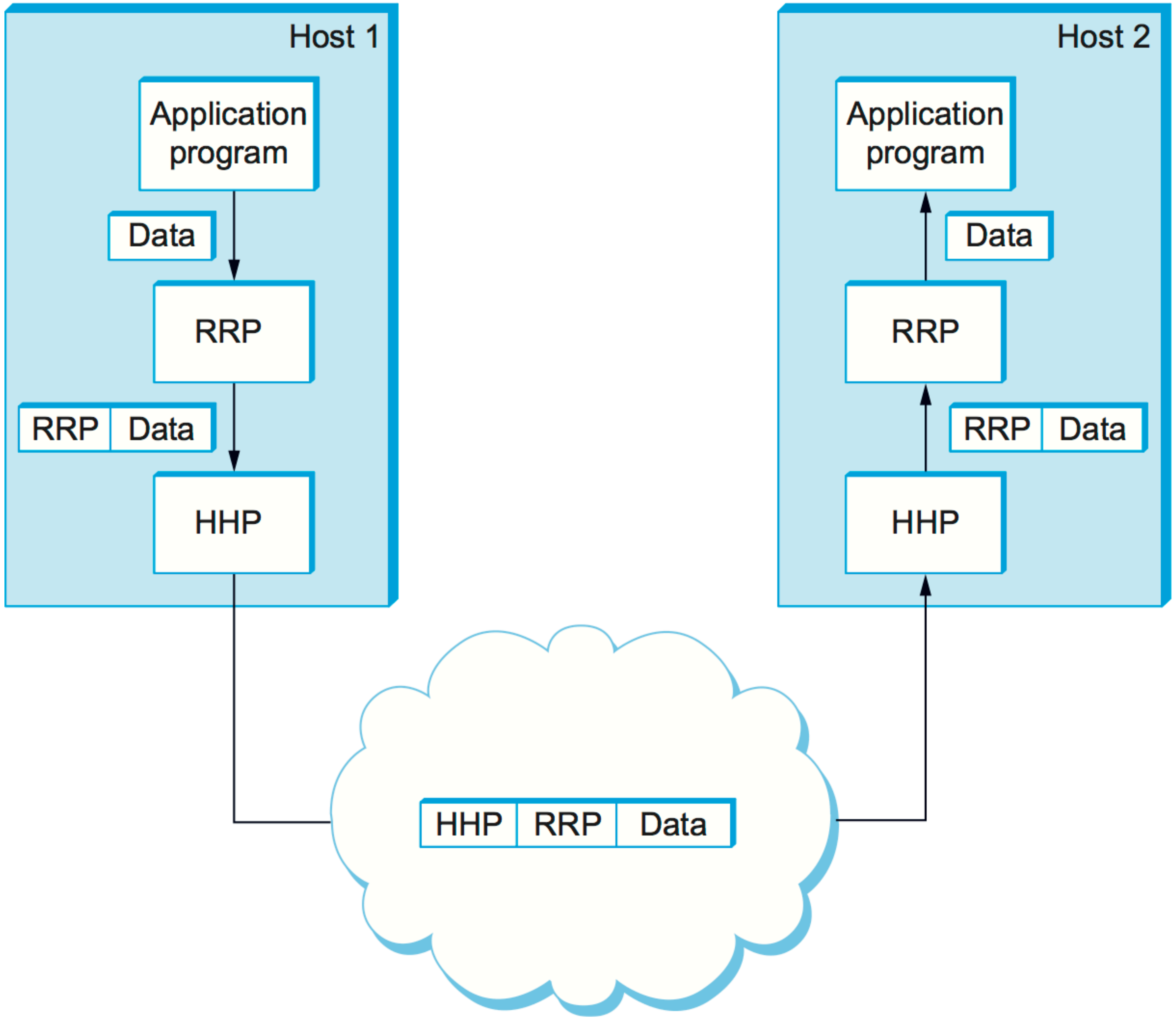


Architecture

Layers & Protocols

Encapsulation

High-level encapsulated in low-level

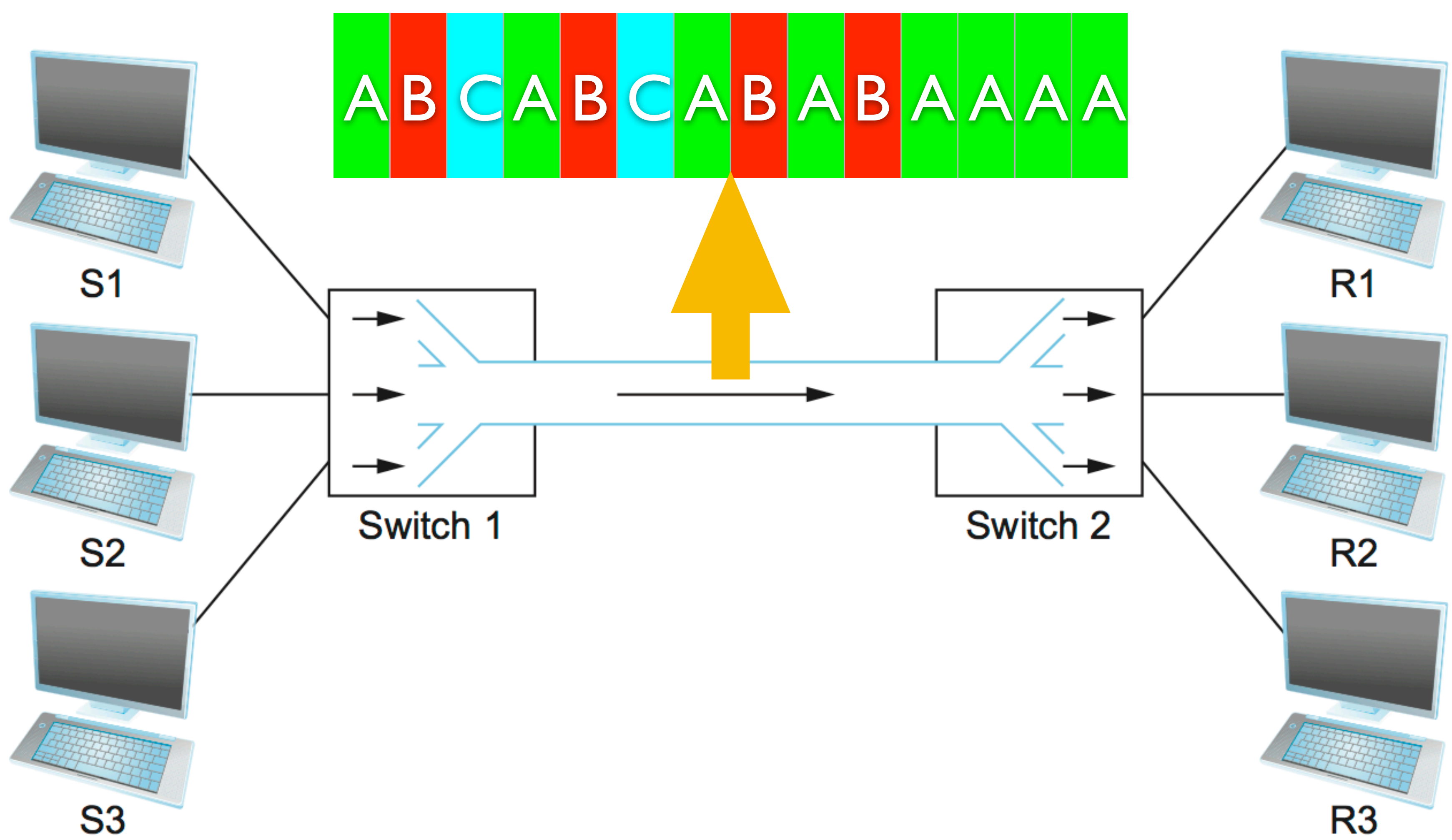


Architecture

Layers & Protocols

Encapsulation

Mux/Demux



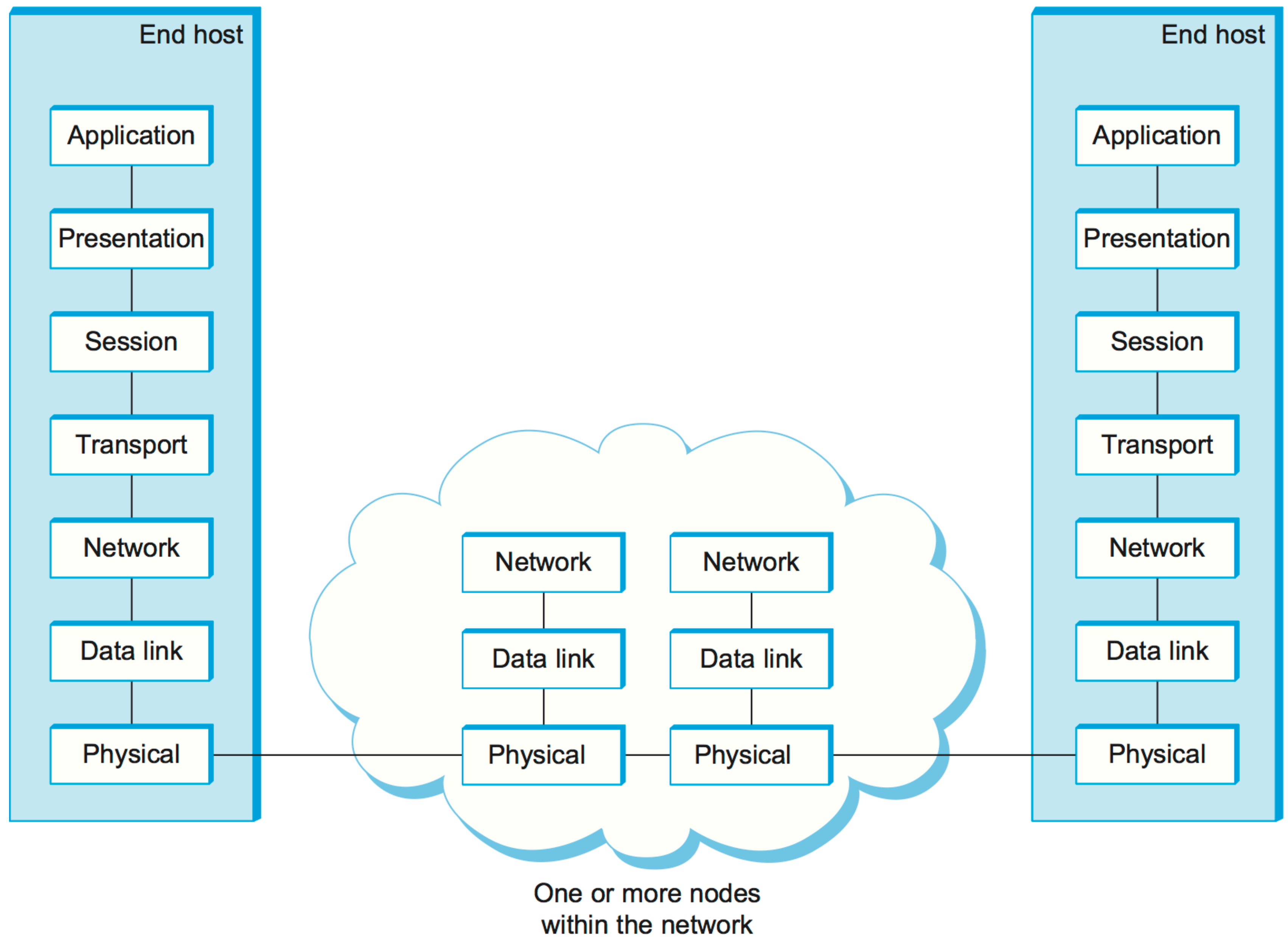
Architecture

Layers & Protocols

Encapsulation

Mux/Demux

OSI 7-Layer



Architecture

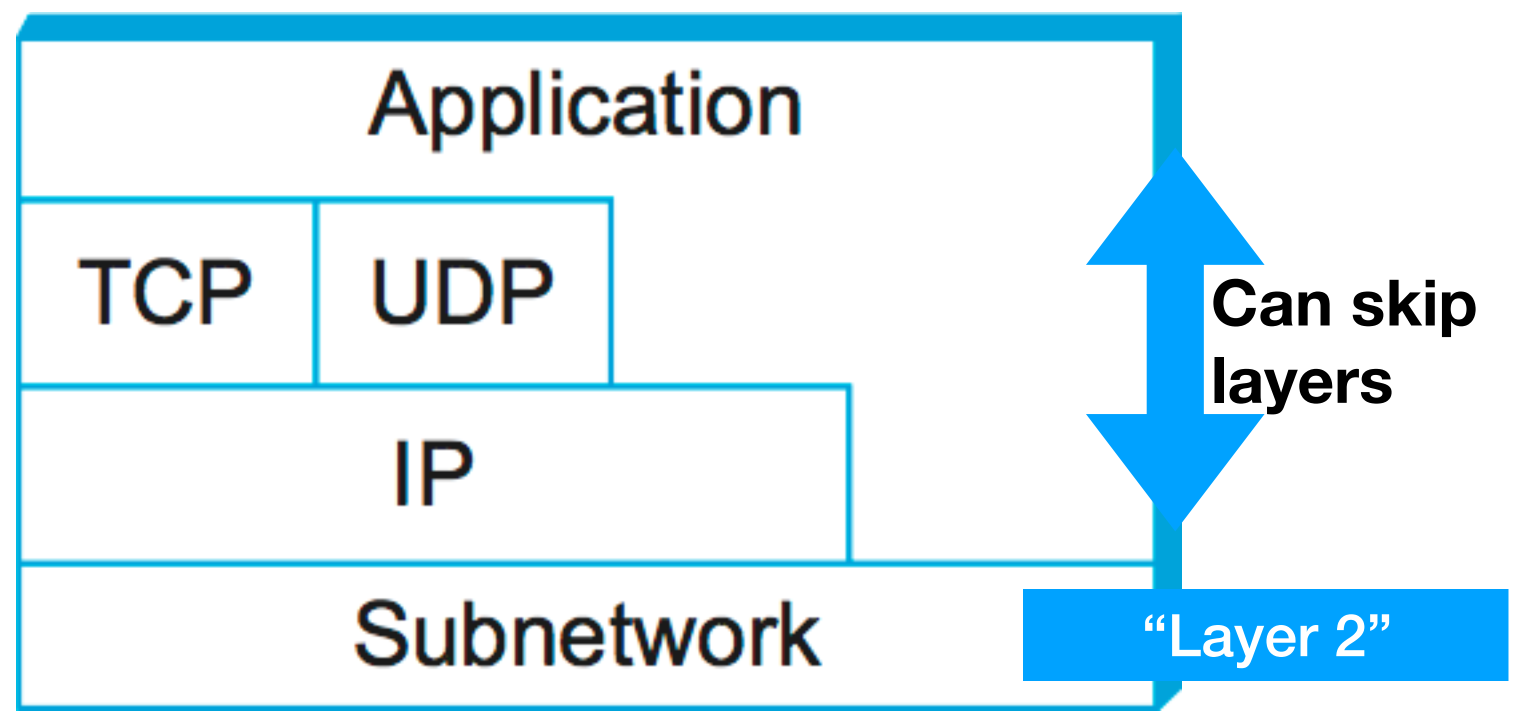
Layers & Protocols

Encapsulation

Mux/Demux

OSI 7-Layer

Internet



Architecture

Layers & Protocols

Encapsulation

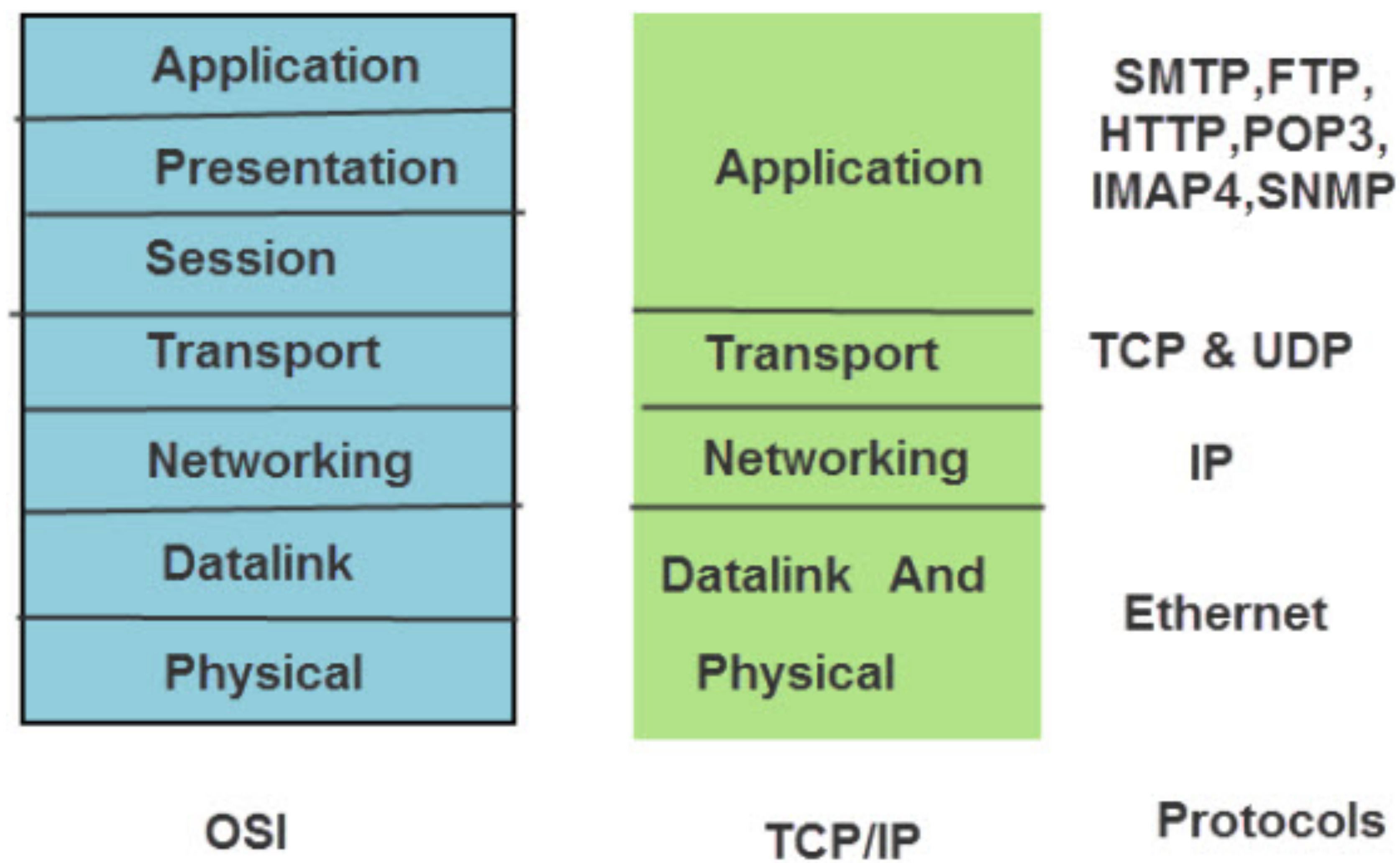
Mux/Demux

OSI 7-Layer

Internet

Comparison

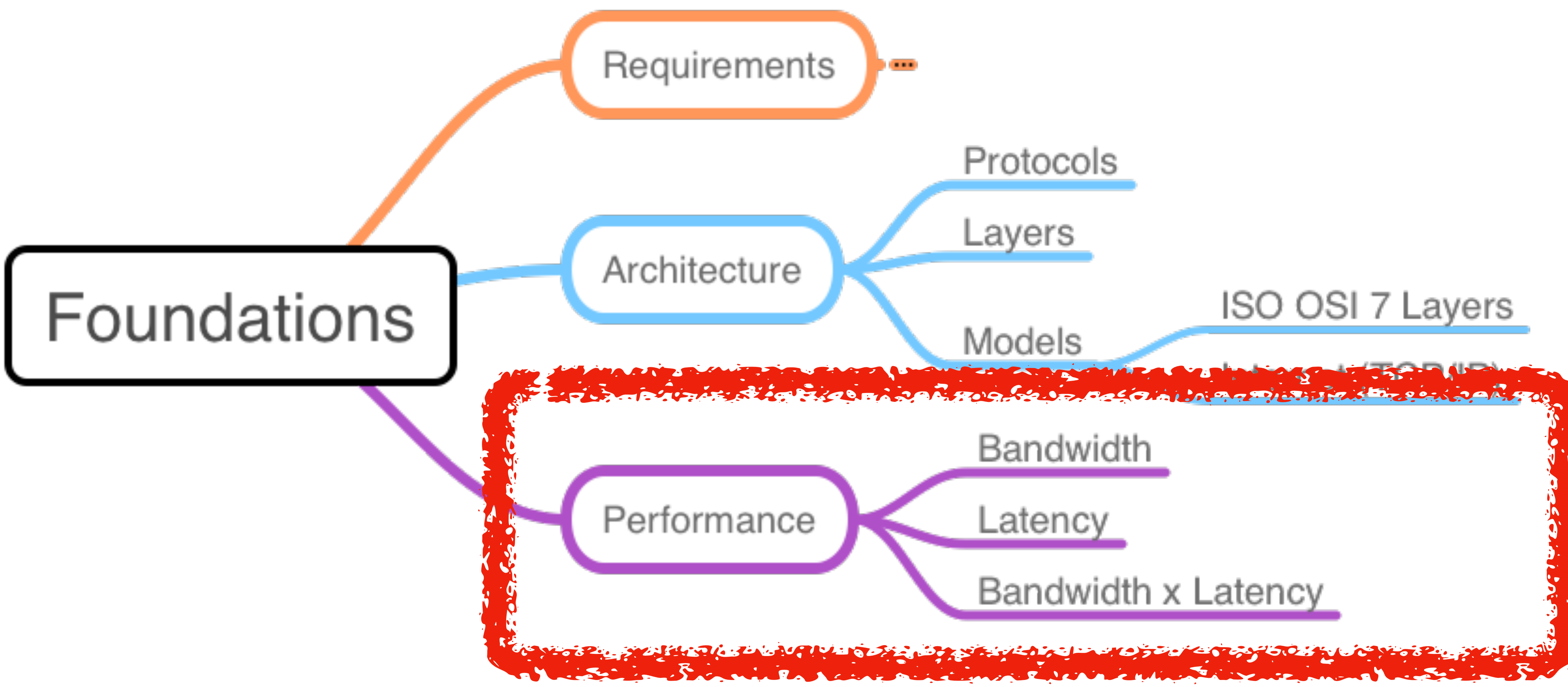
OSI & TCP/IP Protocol-Stacks and Protocols



— No direct mapping —

Source: Steve's Internet Guide

Performance

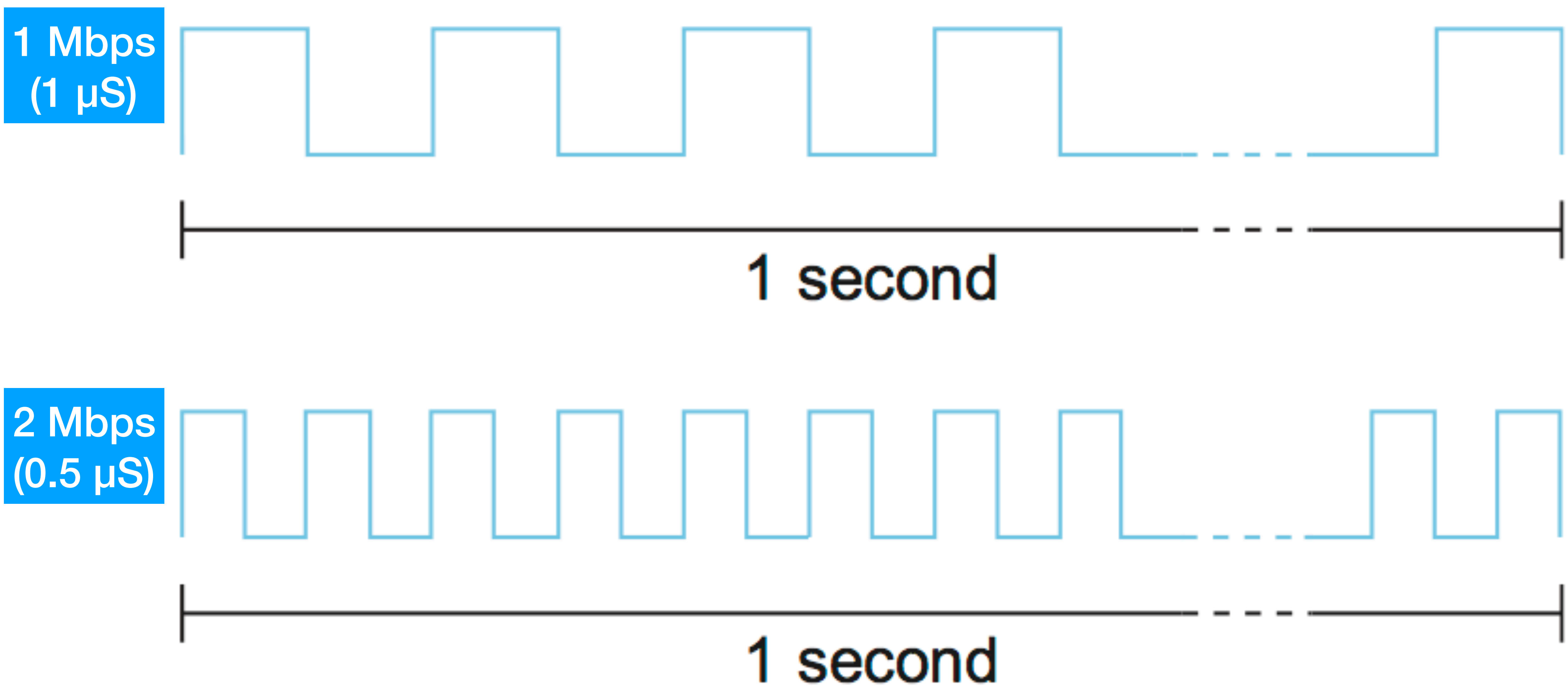


Bandwidth

Used to denote the <u>number of bits that can be transmitted over the network in a certain period of time</u>, also called the *'data rate'*

Bandwidth is literally a measure of the width of a frequency band

Bandwidth



Bandwidth

10 Mbps

- 10 million bits per second
- 0.1µS for each bit

20 Mbps - 0.05 µS for each bit

100 Mbps - 0.01 µS for each bit

Latency

How long it takes a message to travel from one node to another.

`Latency = Propagation + Transmit + Queue`

Where:

```
Propagation = Distance / Speed of Light

Transmit = Size / Bandwidth
```

Speed of Light

Light travels across different mediums at different speeds

Medium	Speed of Light
Vacuum	3 x 108 m/s
Copper	2.3 x 108 m/s
Fiber	2.0 x 108 m/s

Bandwidth vs Latency

Consider a 1 Byte message and a 1 Byte response over a 10 Mbps link with a 10 ms Round Trip Time (RTT)

Does **bandwidth** or **latency** dominate the transmission time?

When will **bandwidth** dominate?

Delay x Bandwidth

Amount of data "in-flight"

Consider a 10 Mbps link with 50 mS latency

How much data is "on the wire"?

10 Mbps x 50 mS

(10×10^6) bits/sec x (50×10^{-3}) seconds

500×10^3 = 500 Kb

= 62.5 KB

High-speed Networks

- Very **large bandwidth**
- Latency does **not** change
- Delay x Bandwidth goes **up**
- More data in-flight during RTT
- Latency begins to dominate transfer times

Bandwidth & Latency

Importance depends on application

Application	Important
File Transfer	Bandwidth
Small Messages (HTTP, NFS)	Latency
Audio / Video Conference	Variation in Latency (jitter)

4. 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

Problem 4 from chapter 1 of textbook

Exercises

1.5 MB file, 80 mS RTT, 1KB packets, handshake 2xRTT, 10 Mbps network, continuous sending.

data size	$= (1.5 \times 2^{20}) \times 8$	= 12,582,912 bits
bandwidth	$= 10 \times 10^{6}$	= 10,000,000
RTT		= 0.080 seconds
packet size	$= 2^{10} \times 8$	= 8,192 bits
handshake	= 2 x RTT	= 0.16 seconds
total	= propagation + transmit + queue(ignore)	

total	= handshake + propagation + transmit	
propagation	= RTT / 2	= 0.04 seconds
transmit	= size / bandwidth	= 1.258 seconds
total	= 0.16 + 0.04 + 1.2583	= **1.458 seconds**

Exercises

1.5 MB file, 80 mS RTT, 1KB packets, handshake 2xRTT,
10 Mbps network, 1 RTT delay after each packet is sent.

data size	= $(1.5 \times 2^{20}) \times 8$	= 12,582,912 bits
bandwidth	= 10×10^{6}	= 10,000,000
RTT		= 0.080 seconds
packet size	= $2^{10} \times 8$	= 8,192 bits
handshake	= 2 x RTT	= 0.16 seconds
total	= propagation + transmit + queue(ignore)	

propagation	= RTT / 2	= 0.04 seconds
transmit	= size / bandwidth	= 1.258 seconds
packets	= size / packet size	= 1,536 packets
packet delay	= (packets - 1) * RTT	= 122.8 seconds
total	= 0.16 + 0.04 + (1.258 + 122.8)	= **124.258 s**

Exercises

1.5 MB file, 80 mS RTT, 1KB packets, handshake 2xRTT,
infinite transmit network, can send 20 packets per RTT

data size = $(1.5 \times 2^{20}) \times 8$ = 12,582,912 bits
RTT = 0.080 seconds
packet size = $2^{10} \times 8$ = 8,192 bits
handshake = 2 x RTT = 0.16 seconds
total = propagation + transmit + queue(ignore)

total = handshake + (chunks x RTT)
packets = size / packet size = 1,536 packets
chunks = ceil(packets / 20) = 77 chunks
total = 0.16 + (77 x 0.080) = **6.32 seconds**

Don't need to wait for last response (- prop delay)

Exercises

1.5 MB file, 80 mS RTT, 1KB packets, handshake 2xRTT,
infinite transmit network, send $2^{(n-1)}$ packets each RTT.
(1 packet 1st RTT, 2 second RTT, 4 third RTT, …)

data size = $(1.5 \times 2^{20}) \times 8$ = 12,582,912 bits
RTT = 0.080 seconds
packet size = $2^{10} \times 8$ = 8,192 bits
handshake = 2 x RTT = 0.16 seconds
total = propagation + transmit + queue(ignore)

total = handshake + (chunks x RTT) - prop_delay
packets = size / packet size = 1,536 packets
chunks = ceil($\log_2$(packets)) = 11 chunks

3. Calculate the total time required to transfer a 1000-KB file in the following cases, assuming an RTT of 50 ms, a packet size of 1 KB data, and an initial $2 \times$ RTT of "handshaking" before data is sent:

(a) The bandwidth is 1.5 Mbps, and data packets can be sent continuously.

(b) The bandwidth is 1.5 Mbps, but after we finish sending each data packet we must wait one RTT before sending the next.

(c) The bandwidth is "infinite," meaning that we take transmit time to be zero, and up to 20 packets can be sent per RTT.

(d) The bandwidth is infinite, and during the first RTT we can send one packet (2^{1-1}), during the second RTT we can send two packets (2^{2-1}), during the third we can send four (2^{3-1}), and so on. (A justification for such an exponential increase will be given in Chapter 6.)

Exercise

Problem 3 from chapter 1 of textbook

```
# Exercise 1.3
1,000KB file,
50ms RTT,
1KB packet size,
2 RTT handshake at start

   size = (1,000 * 2^10) * 8 => 8,192,000
   rtt = 0.050 => 0.05
   packet_size = 2^10 * 8 => 8,192
   handshake = 2 * rtt => 0.1
```

Exercise 1.3

1,000KB file, RTT 50ms, 2 RTT handshake at start

```
size = (1,000 * 2^10) * 8 => 8,192,000
rtt = 0.050 => 0.05
packet_size = 2^10 * 8 => 8,192
handshake = 2 * rtt => 0.1
```

a) 1.5Mbps, continuious

```
bandwidth = 1.5 * 10^6 => 1,500,000
propogation_delay = rtt / 2 => 0.025
transmit_time = size / bandwidth => 5.4613

handshake + propogation_delay + transmit_time
                                            => 5.5863
```

Exercise 1.3

1,000KB file, RTT 50ms, 2 RTT handshake at start

```
    size = (1,000 * 2^10) * 8 => 8,192,000
    rtt = 0.050 => 0.05
    packet_size = 2^10 * 8 => 8,192
    handshake = 2 * rtt => 0.1
```

b) 1.5Mbps, one RTT after each packet

```
    bandwidth = 1.5 * 10^6 => 1,500,000
    propogation_delay = rtt / 2 => 0.025
    transmit_time = size / bandwidth => 5.4613

    packet_count = size / packet_size => 1,000
    t_time = transmit_time + ((packet_count - 1) * rtt)
                                             => 55.4113

    handshake + propogation_delay + t_time => 55.5363
```

Exercise 1.3
1,000KB file, RTT 50ms, 2 RTT handshake at start

```
size = (1,000 * 2^10) * 8 => 8,192,000
rtt = 0.050 => 0.05
packet_size = 2^10 * 8 => 8,192
handshake = 2 * rtt => 0.1
```

c) infinite bandwidth, 20 packets per RTT

```
packet_count = size / packet_size => 1,000
packet_chunks = ceil(packet_count / 20) => 50

handshake + (packet_chunks * rtt) => 2.6
handshake + (packet_chunks * rtt)
          - propogation_delay => 2.575
```

Exercise 1.3

1,000KB file, RTT 50ms, 2 RTT handshake at start

```
size = (1,000 * 2^10) * 8 => 8,192,000
rtt = 0.050 => 0.05
packet_size = 2^10 * 8 => 8,192
handshake = 2 * rtt => 0.1
```

d) exponential packet counts, 1, 2, 4, 8, ...

```
packet_count = size / packet_size => 1,000
packet_chunks = ceil(log(packet_count, 2)) => 10
```

Don't need to wait for response on last packet to calculate arrival times.

```
handshake + (packet_chunks * rtt)
          - propogation_delay => 0.575
```

fin

Foundations