Foundations

COS 460 & COS 540

1

Foundations

- Applications
- Requirements
- Architecture
- Performance
- Writing Code

2

Applications

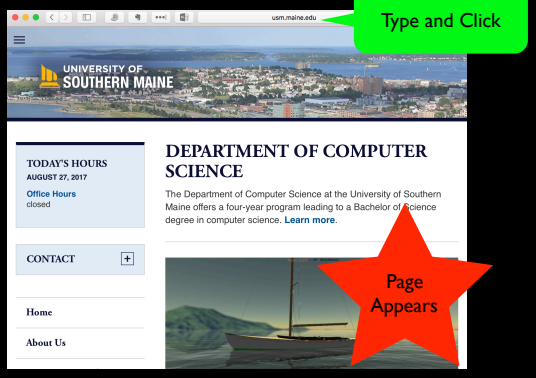
...and these are just the web "social media" ones

3

Example: HTTP

- HyperText Transfer Protocol
- HTTPS is "secure" HTTP
- Request for Comment (RFC) 2616, 2068, and 1954
- Utilizes TCP/IP (Internet Architecture)

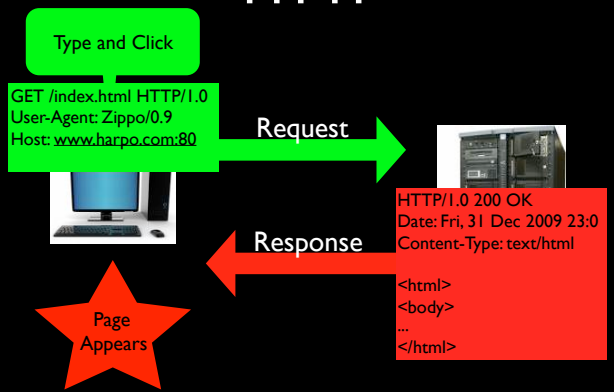
4



How does this work?

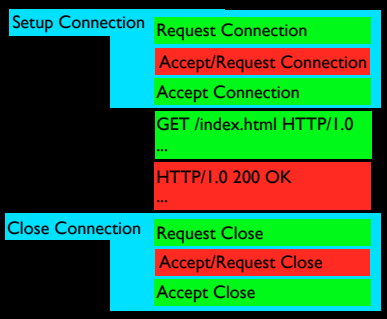
5

HTTP



6

HTTP, TCP



7

HTTP

- HTTP is essentially **file transfer**
- Every bit is important
- Similar to Email, Printing, File Servers...

8

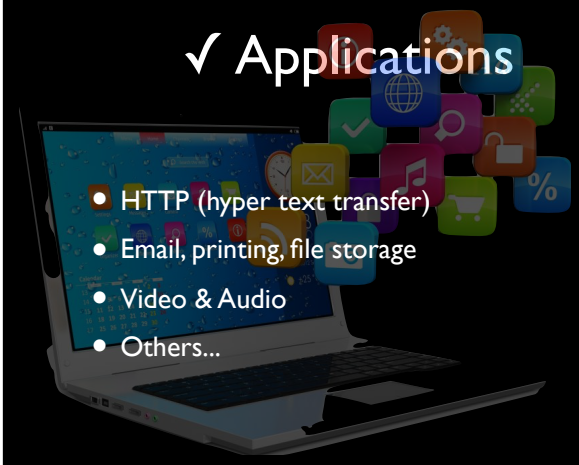
Audio & Video

Every bit is not **important** but their **timing** is

9

10

✓ Applications



- HTTP (hyper text transfer)
- Email, printing, file storage
- Video & Audio
- Others...

11

Foundations

- ✓ Applications
- Requirements
- Architecture
- Performance
- Writing Code

12

Requirements

- Connectivity
- Cost-effective
- Supports (common) Services

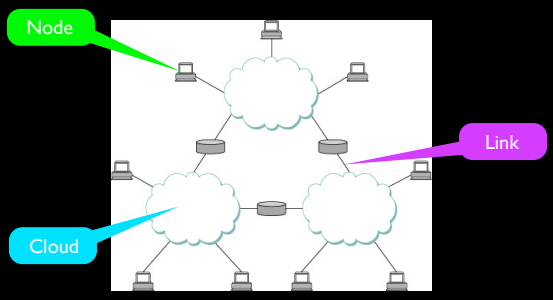
13

Who's Requirements?

- Application Programmer
- Network Provider (ISP)
- Network Designer

14

Connectivity



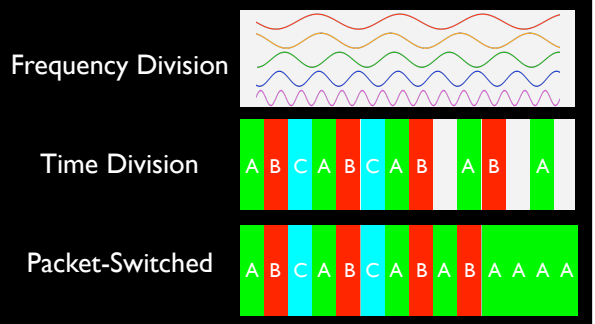
15

Cost-Effective

How do we get as **much information** as possible, from **multiple** sources to multiple sources across the "wire"?

...one bit at a time is slow.

Multiplexed



16

Imagine A has 2x B's data needs and B has 2x C's
FDM – Unused allocation for D and E as well as B and C not using all
TDM – Imagine A has more data, unused time slots
PSwitched – Quality of Service & Congestion problems

Common Services

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

17

Common Services

- File transfer vs. Streaming
- Reliable vs. Unreliable
- Bandwidth vs. Latency
- Large vs. Small

18

19

✓ Requirements

- **Connectivity**
  Links, Nodes, Clouds
- **Cost-Effective**
  Packet Switched
- **Support (common) Services**
  File transfer, streaming, bandwidth...

20

Foundations

✓ Applications
✓ Requirements
- Architecture
- Performance
- Writing Code

21

Architecture

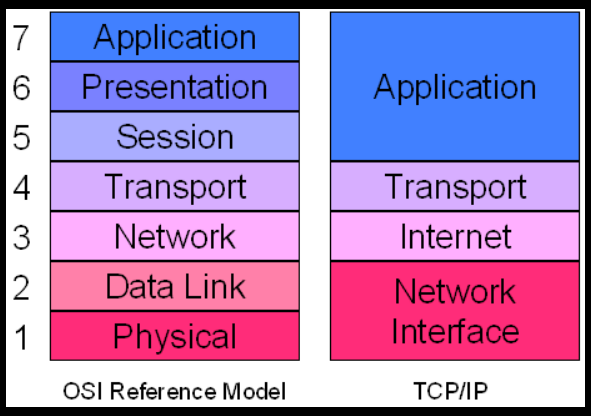
How do we build a Network so it is...

- understandable
- programmable
- flexible
- useful

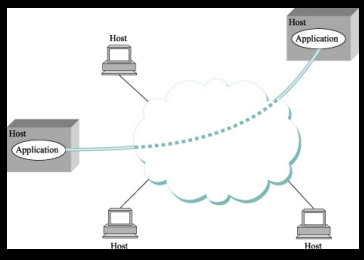
22

Layers make everything better

23

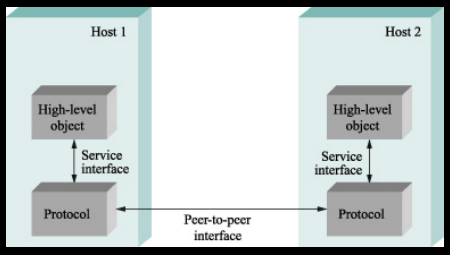


24



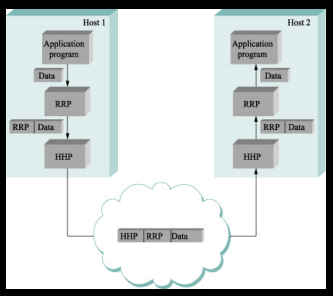
Application View

25



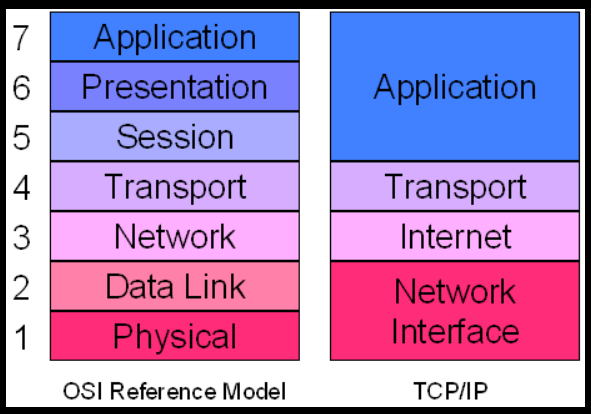
Service/Peer Interfaces

26



Encapsulation

27



28

✓ Architecture

Network Layers

- Subdivide the problem
- Address one at a time
- Do a good job at each layer

29

Foundations

- ✓ Applications
- ✓ Requirements
- ✓ Architecture
- Performance
- Writing Code

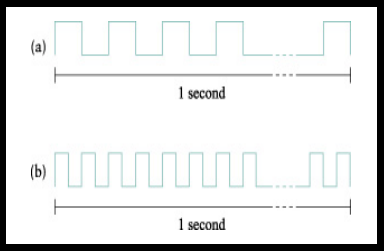
30

Performance

- Bandwidth
- Latency
- Delay X Bandwidth
- High-Speed Networks

31

Bandwidth



Bits transmitted per unit of time

32

Bandwidth

- 10 Mbps
  - 10 million bits per second
  - 0.1µS for each bit
- 20 Mbps; 0.5µS for each bit
- 100 Mbps; 0.01µS for each bit

33

Latency

How long does it take to get there?

34

Latency

- = Propagation + Transmit + Q-Delay
  - Propagation Delay (distance / c)
  - Transmit Time (size / bandwidth)
  - Queueing Delays

35

Latency

- Speed of Light (c)
  - copper = 2.3×10^8 m/s
  - vacuum = 3.0×10^8 m/s
  - fiber = 2.0×10^8 m/s

36

Latency

- One Way Latency
- Round Trip Time (RTT)

37

Bandwidth vs Latency

For...

- HTTP?
- File Transfer?
- Audio?
- Video?

38

Bandwidth vs Latency

Consider a 1-byte message and a 1-byte response over a 10Mbps link with a 10ms RTT.

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

…When will bandwidth dominate?

0.8us + 10ms = 10.0008ms — Latency dominates
Even at 1Gbps — 0.008us + 10ms = 10.000008 Latency dominates

@10Mbps; 10,000bits = 1ms; 100,000bits = 10ms; 12,500 bytes or ~12kB

39

Delay x Bandwidth

Amount of data "in-flight"

10 Mbps x 50 mS

(10×10^6) bits/sec x (50×10^{-3}) sec

$500 \times 10^3 = 500Kb$

62.5KB

40

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.

41

✓ Performance

- Bandwidth
- Latency
- Delay x Bandwidth
- High-Speed networks

42

Foundations

✓Applications
✓Requirements
✓Architecture
✓Performance
- Writing Code

Code

43

It Takes Two to Tango

- Server (passive)
- Client (active)

Server

44

- **Binds** to a "port"
  - Passively **listens** for connection request
    - **Accepts** connections
      - reads and writes data
        - **Closes** connection when done

Client

45

- Actively **connects** to server's port
  - reads and writes on connection
    - **Closes** connection when done

46

Java Server

```
// passive bind and listen
ServerSocket srv = new ServerSocket(port);
Socket s = srv.accept();
    s.getInputStream() & read()
    s.getOutputStream() & write()
 s.close();
```

47

Java Client

```
// active connect
Socket s = new Socket("server", port);
    s.getOutputStream() & write()
    s.getInputStream() & read()
s.close()
```

48

✓ Code

- Client (active) - Server (passive)
- Peer to Peer networks?
  - Who's the server?
- Distributed networks
  - Multiple servers and clients

49

Foundations

- ✓ Applications
- ✓ Requirements
- ✓ Architecture
- ✓ Performance
- ✓ Writing Code

50

Foundations

End