# Contents

**Contributors:**
- Daniel Fitz (Sanchez)

## 1 Sem Outline

| Week (dates) | Lecture |
|---|---|
| 1 | Computer Networks and the Internet |
| 2 | Principles of Nw Apps: HTTP, SMTP, DNS |
| 3 | Application Layer: P2P, CDN, Sockets |
| 4 | Networking at UQ |
| 5 | Transport Layer: UDP |
| 6 | Transport Layer: TCP |
| 7 | Network Layer: Data Plane |
| 8 | Network Layer: Control Place |
| 9 | Link Layer |
| 11 | Wireless and Mobile |
| 12 | Security |
| 13 | Multimedia |

Table 1: Week Outline

# 2 Lecture 1

- billions of connected computing devices
- transmission rate: **bandwidth**
- **Packet Switches:** Forward packets
  - **routers** and **switches**
- **Internet: "network of networks"** (Interconnected ISPs)
- **Protocols** control sending, receiving (e.g. TCP, IP, HTTP, Skype, 802.11)
- **Internet standards**
  **RFC:** Request for comments
  **IETF:** Internet Engineering Task Force

## 2.1 Network Structure

- **Network Edge**
  - hosts: clients and servers
  - servers often in data centers
- **Access networks, physical media:** wired, wireless communication links
- **network core:**
  - interconnected routers
  - network of networks

## 2.2 Access Network

### 2.2.1 Digital Subscriber Line (DSL)

- use **existing** telephone line to central office DSLAM
  - data over DSL phone line goes to Internet
  - voice over DSL phone line goes to telephone net
- < 2.5 Mbps upstream transmission rate (typically < 1 Mbps)
- < 24 Mbps downstream transmission rate (typically < 10 Mbps)

### 2.2.2 Cable Network

> **frequency division multiplexing:** different channels transmitted in different frequency bands

- **HFC: hybrid fiber coax**
  - asymmetric: up to 30Mbps downstream transmission rate, 2 Mbps upstream transmission rate
- **network** of cable, fiber attaches homes to ISP router
  - homes **share access network** to cable head-end
  - unlike DSL, which has dedicated access to central office

**wireless LANS:**

- within building (30 meters)
- 802.11b/g/n (WiFi): 11,54,450 Mbps transmission rate

**wide-area wireless access:**

- provided by telco (cellular) operator, 10's km
- between 1 and 10 Mbps
- 3G, 4G, LTE

## 2.3 Sending

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

**Note 1: Packet Transmission Delay**

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

## 2.4 Physical Media

- **bit:** propagates between transmitter/receiver pairs
- **physical link:** what lies between transmitter and receiver
- **guided media:** signals propagate in solid media (copper, fiber, coax)
- **unguided media:** signals propagate freely, e.g. radio
- **twisted pair (TP):** two insulated copper wires
  - Category 5: 100 Mbps, 1 Gbps Ethernet
  - Category 6: 10 Gbps

### 2.4.1 Coax

- two concentric copper conductors
- bidirectional
- broadband: multiple channels on cable, HFC

### 2.4.2 Fiber Optic Cable

- glass fiber carrying light pulses, each pulse a bit
- high-speed operation: high-speed point-to-point transmission (e.g. 10's - 100's Gbps transmission rate)
- low error rate
  - repeaters spaced far apart
  - immune to electromagnetic noise

### 2.4.3 Radio

- signal carried in electromagnetic spectrum
- no physical "wire"
- bidirectional
- propagation environment effects:
  - reflection
  - obstruction by objects
  - interference

**Radio Link Types:**

- **terrestrial microwave:** up to 45 Mbps channels
- **LAN** (e.g. WiFi) 54 Mbps
- **wide-area** (e.g. cellular) 4G cellular: ˜10 Mbps
- **satellite**
  - Kbps to 45 Mbps channel (or multiple smaller channels)
  - 270 msec end-end delay
  - geosynchronous versus low altitude

## 2.5 Packet-switching

### 2.5.1 Store-and-forward

> $L$ bits per packet
> Source to destination: $R$ bps

- takes $\frac{L}{R}$ seconds to transmit (push out) $L$-bit packet into link at $R$ bps
- **store and forward:** entire packet must arrive at router before it can be transmitted on next link

**Note 2: End-End delay**

$$\text{delay} = 2\frac{L}{R}$$

(assuming zero propagation delay)

### 2.5.2 Packet switching versus circuit switching

Is packet switching a "slam dunk winner?"

- great for bursty data (resource sharing, simpler, no call setup)
- excessive congestion possible: packet delay and loss (protocols needed for reliable data transfer, congestion control)

## 2.6 Packet Loss

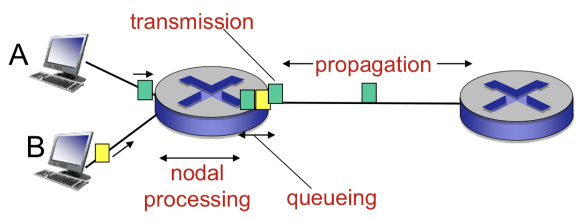

Figure 1: Packet Delay Algorithm Explanation

**Note 3: Packet Delay Algorithm**

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

### 2.6.1 Nodal Processing

$$d_{\text{proc}}$$

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

### 2.6.2 Queuing Delay

$$d_{\text{queue}}$$

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

### 2.6.3 Transmission Delay

$$d_{\text{trans}}$$

- $L$: packet length (bits)
- $R$: link bandwidth(bps)
- $d_{\text{trans}} = \frac{L}{R}$

### 2.6.4 Propagation Delay

$$d_{\text{prop}}$$

- $d$: length of physical link
- $s$: propagation speed ($\approx 2 \times 10^8$ m/sec)
- $d_{\text{prop}} = \frac{d}{s}$

## 2.7 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

**Note 4: Bottleneck Link**

Link on end-end path that constrains end-end throughput

## 2.8 Layering

### 2.8.1 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 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
- layering considered harmful?

### 2.8.2 Internet Protocol Stack

**Application:** supporting network applications (FTP, SMTP, HTTP)
**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, 802.111 (WiFi), PPP)
**Physical:** bits "on the wire"

### 2.8.3 ISO/OSI Reference Model

Internet stack "missing" these layers. These services, if needed, must be implemented in application.
**Application:**
**Presentation:** allow applications to interpret meaning of data, e.g. encryption, compression, machine-specific conventions
**Session:** synchronization, check-pointing, recovery of data exchange
**Transport:**
**Network:**
**Link:**
**Physical:**

## 2.9 Security

- 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

### 2.9.1 DoS: Denial of Service

**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 (botnet)
3. send packets to target from compromised hosts

### 2.9.2 Sniffing

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

### 2.9.3 IP Spoofing

Send packet with false source address

# 3 Lecture 2

## 3.1 Application Architectures

### 3.1.1 Client-Server

**Server:** Always-on host, Permanent IP address
**Clients:** Do not communicate directly with each other, May have dynamic IP addresses

### 3.1.2 Peer-to-Peer (P2P)

- No always-on server
- 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

- Pers are intermittently connected and change IP addresses

**Note 5: App-layer protocol defines**

- **type of messages exchanged** – e.g. request, response
- **message syntax** – what fields in messages and how fields are delineated
- **message semantics** – meaning of information in fields
- **rules** for when and how processes send and respond to messages
- **open protocols** – defined in RFCs, allows for interoperability (e.g. HTTP, SMTP)
- **proprietary protocols** – e.g. Skype

## 3.2 Transport Service is needed

**Data Integrity:** Some programs need 100% reliable data transfer (e.g. file transfer, web transactions), others can tolerate loss (e.g. audio)

**Timing:** Some programs require low delay to be "effective" (e.g. online games)

**Throughput:** Some programs require minimum amount of throughput to be "effective" (e.g. multimedia), some use whatever they have available ("elastic apps")

**Security:** Encryption, Data Integrity

## 3.3 Transport Protocol Services

### 3.3.1 TCP

**Reliable Transport** between sending and receiving process

**Flow Control:** sender won't overwhelm receiver

**Congestion Control:** throttle sender when network overloaded

**Connection-Oriented:** setup required between client and server processes

**Does Not Provide:** timing, minimum throughput guarantee, security

### 3.3.2 UDP

**Unreliable Data Transfer** between sending and receiving process

**Does Not Provide:** reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

### 3.3.3 Securing TCP

**TCP and UDP**

- no encryption
- cleartext passwords sent into socket traverse Internet in cleartext

**SSL**

- provides encrypted TCP connection
- data integrity
- end-point authentication

**SSL is at app layer**

- app use SSL libraries, that "talk" to TCP

**SSL socket API**

- cleartext passwords sent into socket traverse Internet encrypted

## 3.4 HTTP: Hypertext Transfer Protocol

- Web's application layer protocol
- client/server model. Client request website and server serves HTTP object in response
- Uses TCP
- HTTP is stateless. Server maintains no information about past client requests
- **non-persistent HTTP:** one object sent over one TCP connection, downloading multiple object required multiple connections
- **persistent HTTP:** multiple object sent over single TCP connection

Non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

### 3.4.1 Method Types

**HTTP/1.0:** GET, POST, HEAD (asks server to leave requested object out of response)

**HTTP/1.1:** GET, POST, HEAD, PUT (uploads file in entity body to path specified in URL field), DELETE (deletes file specified in the URL field)

#### 3.4.2 Response Codes

**200 OK:** request succeeded, requested object later in this msg

**301 Moved Permanently:** requested object moved, new location specified later in this msg

**400 Bad Request:** request msg not understood by server

**404 Not Found:** requested document not found on this server

**505 HTTP Version Not Supported**

## 3.5 Cookies

Uses: authorization, shopping carts, recommendations, user session state (Web, email)

## 3.6 Web Caches (proxy server)

> **Goal:** satify client request without involving origin server

- Browsers requests object from cache, if in cache the object is sent back otherwise cache requests object from origin
- Cache acts as both client and server
- Reduce response time for client request
- Reduce traffic

### 3.6.1 Conditional GET

> **Goal:** don't send object if cache has up-to-date cached version (lower link usage)

- **Cache:** specify date of cached copy in HTTP request `If-modified-since: <date>`
- **Server:** response contains no object if cached copy is up-to-date: `HTTP/1.0 Not Modified`

## 3.7 Electronic Mail: SMTP

*RFC 2821*

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer: handshaking, transfer of messages, closure
- command/response interaction
- messages must be in 7-bit ASCII
- uses persistent connections
- requires message to be in 7-bit ASCII
- uses `CRLF.CRLF` to determine end of message

Difference to HTTP being, HTTP is server sending data, SMTP is client connection sending data

**SMTP:** protocol for exchanging email messages

**RFC 822:** standard for text message format (To, From, Subject, Body)

### 3.7.1 Mail Access Protocols

> **SMTP:** delivery/storage to receiver's server

**POP:** Post Office Protocol *(RFC 1939)*: authorization, download
- POP3 is stateless across sessions
- Two main modes; download and delete, download and keep (allows multiple clients to read the same email)

**IMAP:** Internet Mail Access Protocol *(RFC 1730)*: more features, including manipulation of stored message on server
- All messages stored on server
- Supports folders
- Keeps user state across sessions: names of folders and mappings between message IDs and folder name

**HTTP:** gmail, Hotmail, Yahoo, etc

## 3.8 DNS: Domain Name System

- Lookup between names (e.g. google.com) and IP addresses
- **Distributed Database** implemented in hierarchy of many **name servers**
- **Application-layer protocol:** hosts, name servers communicate to **resolve** names (address/name translation)

Why not centralize DNS? Single point of failure, traffic volume, doesn't scale

### 3.8.1 DNS Services

- hostname to IP address translation
- host aliasing (canonical, alias names)
- mail server aliasing
- load distribution (many IP addresses correspond to one name)

### 3.8.2 TLD, authoritative servers

**top-level domain (TLD) servers:**
- responsible for com, org, net, edu, aero, jos, io
- and top-level country domains au, uk, ca
- Network Solutions maintains servers for .com TLD
- Educause for .edu TLD

**Authoritative DNS servers:**
- organization's own DNS server(s), providing authorative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

### 3.8.3 Local DNS name server

- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one (also called "default name server")
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

### 3.8.4 DNS Name Resolution

**Iterated query:** contacted server replies with name of server to contact. So root dns sends the ip of the next dns server to contact

**Recursive query:** puts burden of name resolution on contacted name server. So root dns server contacts the next levels down which contacts next level down.

### 3.8.5 Caching

Once (any) name server learns mapping, it **caches** mapping. Cache entries timeout (disappear) after some time (TTL). If name host changes IP address, the name servers might not update until TTLs expire.

> update/notify mechanisms proposed IETF standard RFC 2136

### 3.8.6 DNS Records

**Note 6: RR Format**

`(name, value, type, ttl)`

**type=A** `name` is hostname, `value` is IP address

**type=NS** `name` is domain (e.g. google.com), `value` is hostname of authoritative name server for this domain

**type=CNAME** `name` is alias name for some "canonical" (the real) name (`www.ibm.com` is really `servereast.backup2.ibm.com`), `value` is canonical name

**type=MX** `value` is name of mailserver associated with `name`

### 3.8.7 Protocol

Query and reply messages both follow same format

Table 2: Protocol Layout

| 2 bytes | 2 bytes |
|---|---|
| identification | flags |
| # questions | # answer RRs |
| # authority RRs | # additional RRs |
| questions (variable # of questions) | |
| answers (variable # of RRs) | |
| authority (variable # of RRs) | |
| additional info (variable # of RRs) | |

### 3.8.8 Attacking DNS

**DDoS attacks**
- bombard root servers with traffic. Not successful to date, traffic filtering, local DNS servers cache protecting root DNS
- bombard TLD server. Potentially more dangerous

**Redirect Attacks**
- man-in-middle (Intercept queries)
- DNS Poisoning (Send bogus relies to DNS server, which caches)

**Exploit DNS for DDoS**
- send queries with spoofed source address: target IP
- requires amplification

# 4 Lecture 3

## 4.1 File Distribution Time

### 4.1.1 Client-server

**Server Tranmission:** must sequentially send (upload) $N$ file copies. Time to send one copy: $\frac{F}{u_s}$. Time to send $N$ copies: $\frac{NF}{u_s}$

**Client:** each client must download file copy. $d_{\text{min}} =$ min client download rate. min client download time $\frac{F}{d_{\text{min}}}$

**Note 7: Client-server File Distribution**

time to distribute $F$ to $N$ clients using client-server approach

$$D_{c-s} \geq \max\{\frac{NF}{u_s}, \frac{F}{d_{\text{min}}}\}$$

### 4.1.2 P2P

**Server Tranmission:** must upload at least one copy. Time to send one copy: $\frac{F}{u_s}$

**Client:** each client must download file copy. Min client download time: $\frac{F}{d_{\text{min}}}$

**Clients:** as aggregate must download $NF$ bits. Max upload rate (limiting max download rate) is $u_s + \sum u_i$

**Note 8: P2P File Distribution**

time to distribute $F$ to $N$ clients using P2P approach

$$D_{\text{P2P}} \geq \max\{\frac{F}{u_s}, \frac{F}{d_{\text{min}}}, \frac{NF}{u_s + \sum u_i}\}$$

### 4.1.3 BitTorrent

File divided into 256Kb chunks
**Tracker:** tracks peers participating in torrent
**Torrent:** group of peers exchanging chunks of a file

## 4.2 Multimedia

### 4.2.1 Video

Coding: used redundancy **within** and **between** images to decrease # bits used to encode image
**Spatial:** within image
**Temporal:** from one image to next
**CBR** (constant bit rate): video encoding rate fixed
**VBR** (variable bit rate): video encoding rate changes as amount of spatial, temporal coding changes

### 4.2.2 DASH

DASH: Dynamic, Adaptive Streaming over HTTP
**Server:** Divides video file into multiple chunks. Each chunk stored, encoded at different rates. **Manifest file:** provides URLs for different chunks
**Client:** Periodically measures server-to-client bandwidth. Consulting manifest, requests one chunk at a time. Chooses maximum coding rate sustainable given current bandwidth. Can choose different coding rates at different points in time (depending on available bandwidth at time)

"intelligence" at client: client determines

- **when** to request chunk (so that buffer starvation, or overflow does not occur)
- **what encoding rate** to request (higher quality when more bandwidth available)
- **where** to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

### 4.2.3 Content Distribution Networks (CDNs)

CDN stores copies of content at CDN nodes. Subscriber requests content from CDN, directed to nearby copy, retrieves content, may choose different copy if network path congested.

## 5 Acronyms

**IP:** Internet Protocol  
**TCP:**  
**UDP:**  
**HTTP:** Hypertext Transfer Protocol  
**SMTP:** Simple Mail Transfer Protocol  
**RDP:** Remote Desktop Protocol  
**VOIP:** Voice over IP  
**RTT:**  
**POP:** Post Office Protocol  
**IMAP:** Internet Mail Access Procotol  
**DNS:** Domain Name System  
**SSN:**  
**TLD:** Top-level Domain  
**TTL:** Time To Live  
**RR:** Resource Records  
**DDoS:**  
**CBR:** Constant bit rate  
**VBR:** Variable bit rate  
**DASH:** Dynamic, Adaptive Streaming over HTTP  
**CDN:** Content Distribution Networks