**Lecture 1**

* Internet
  + Hardware and software
  + Services
* Nuts and bolts view
* RFC – Request for Comments
  + interoperability guidelines
* IETF
  + Internet Engineering Task Force – standards, body for the internet
* Network edge
  + clients servers
* Access networks, physical media
  + wired, wireless communication links (cables, towers, home/local network routers, etc
* Network code
  + Routers
  + Network of networks
* End systems connect to routers via:
  + Various access networks (institutional, residential, mobile)
* FDM (frequency division multiplexing)
  + Multiplexing different channels transmitted in diff frequencies
* HFC (hybrid fiber coax)
  + Up to 40-1.2Gbps d/l and 30-100Mbps u/l
  + Multiple homes share cable to cable headend (cabel modem system) which then connects to ISP
  + Shared with TV but uses diff frequencies
* DSL (digital subscriber line)
  + Unlike HFC, has dedicated access to central office
  + 24-52 Mbps d/l, 3.5-16 Mbps u/l
  + Shared with phone but different frequencies used.
* Wireless access networks (WLAN)
  + WiFi
  + Building radius ish
  + 450Mbps rate
* Wide area cellular access networks:
  + Cellular operators (10s kms coverage)
  + 10s Mbps
  + 4G
* Data center networks
  + 10s to 100s Gbps
  + Hundreds to thousands servers
* Twisted pair
  + Twisted pair of copper wires
  + Up to 10 Gbps ethernet
* Coaxial cable
  + Concentric copper conductors
  + Bidirectional
  + Multiple channels on cable each 100s Mbps per channel
* Fiber optic cable
  + Glass carrying light pulses each being a bit
  + High speed 100s of Gbps
  + Low error
* Wireless radio:
  + Signal carried in bands in EMG spectrum
  + Half-duplex broadcast (one at a time like walkie talkie)
  + Affected by reflection, objects, interference / noise

**Lecture 2**

* Packet-switching
  + Good for bursty data
  + More scalable
  + Packet loss and delays possible due to congestion
  + Can pay ISPs for allocation of resources but costly
* Routing (global, data plane) vs forwarding aka switching (local, control plane)
* Store and forward
  + Entire packet must arrive at router before it is transmitted
* Queueing in packet switching
  + When arrival rate > transmission rate
  + Packets are queued until they are lost when buffer is full
* Circuit switching
  + End-to-end resources allocated for a session
  + Guaranteed performance (no loss no delays), no sharing of circuits
  + FDM (Freq. Div. Multiplexing) vs TDM
* Network of networks
  + Branches like trees(OlogN) instead of direct connections from every single host to every other host (ON^2)
  + Scalable
* Global ISP
  + Maintain regional ISPs
  + Ex. Verizon
  + Interconnected via IXP (internet exchange points)
* Regional ISPs
  + Connect access networks to global ISPs
* Content provider networks
  + Google, Microsoft, Akamai, etc. Own their own networks
* Traffic intensity
  + If > 1 traffic is ++++ with packet loss
* Traceroute programs
  + Program that captures real delays
  + Sends 3 probes per router on path to receiver, obtain a return and measure the time it took.
* Throughput bottleneck
  + In practice usually not on the ISP end but on the receiver to router or router to server end (or vice versa)

**Lecture 3**

* Advantages of layer architecture
  + Modularization facilitates maintenance
  + Isolation of problems and issues
  + Divide and conquer approach
* Protocol Data Unit (PDU)
  + Message (app layer)
  + Segment (Transport layer)
  + Datagram (Network layer)
  + Frame (link layer)

**Lecture 4 – Application Layer**

* Server in client-server paradigm
  + Always-on host running
  + Permanent IP address
  + In data centres usually for scaling and reliability
* P2P
  + No always on servers
  + Arbitrary pairing for direct communication (like pairing headphones to phone)
  + Peers request and provide services from and to one another like Bit Torrent
  + Self-scalable
  + Ips can change
* Addressing processes
  + Uses both IP and port # to differentiate between processes
  + IP address is 32 bits
  + HTTP port: 80
  + Mail server port: 25
* Application protocol defines
  + Types of messages (GET, POST)
  + Message syntax
  + Message semantics (meaning of msgs)
  + Rules (how to send respond to msgs)
* Open vs proprietary protocols
  + Proprietary use open protocols but customize them
* Quality of service (QoS)
  + Data integrity (loss tolerance)
  + Timing (time sensitivity
  + Throughput (requirements vary by app)
  + Security (ex. Banking)
* Transport Layer Security (TLS
  + Provide encrypted TCP connections
* Content Distribution Networks (CDNs)
  + Solution to scalability of massively growing streaming services
  + Solution to diversity in use capabilities (wired vs mobile networks, bandwidth variations etc.)
  + Reduce congestion over the core network keeping much traffic at the edge
  + 1 mega server not scalable, prone to congestions and bottlenecks, and single point of failure. Long path to clients too
  + CDN enter deep (servers deep in access networks) vs bring home (clusters near access networks)
* Video Coding
  + Compression of video data prior to transmission
  + Used to eliminate redundancy between and within images (spatial vs temporal)
* Bit rates (constant vs variable)
  + MPEG 1 MPEG 2 use CBR
  + MPEG 4 uses VBR
  + MPEG 5 new tech
* Continuous playout constraint
  + The requirement that video keeps playing despite fluctuating network conditions
  + Client side buffer required for that

**Lecture 5 – HTTP and Web Services**

* HTTP
  + Uses TCP
  + Stateless (does not store metadata on requests – some exceptions): simple, flexible, scalable but can result in redundant data transmissions
* Non-persistent HTTP
  + Extra TCP connections = extra overhead and CPU resources
  + Not scalable but more secure (therefore used in banking) and less network wastage
* Persistent HTTP
  + Connection left open after 1 request
  + Requests for objects sent as they are encountered
* HTTP messages
  + POST – data to store or process
  + GET – send data to server to retrieve something in return
  + HEAD – request a header without a file (ex. Size)
  + PUT – like post but to update or replace something
* Cookies
  + HTTP stateless – cookies store info on clients
  + Steps:
    - User sends request -> server includes cookie header line (including number) in response
    - Client includes that cookie header in next request
    - Cookie file kept on user host managed by browser
    - Data associated with cookie stored on server database
* Web cache
  + Browsers configured to send all HTTP reqs to cache.
  + If data not in cache, request forwarded to server
  + Objects from server retrieved by cache then forwarded to user
  + Use can be blocked by server with cache-control header
  + Benefits: Reduce service time, reduce traffic on access link
* Conditional GET
  + Used when cache has file
  + If not modified – response is 304 – Not Modified
* Head of Line (HoL)
  + Persistent HTTP sends multiple requests – arrive often out of order
  + Server treats requests in first-come first-served order
  + Small items sometimes wait for a long time when requests for large items arrive first
  + Loss recovery can also block small items
* HTTP 2
  + Reduces HoL
  + Server services requests based on client defined priority(ex. Html, css first)
  + Push unrequested objects to clients
  + Divide objects into frames, scheduling frames to mitigate HoL
  + HoL not fully resolved – present in cases of packet loss and retransmissions
* HTTP 3
  + Sends lower order items in parallel
  + Even if high order packet lost, the rest are still transmitting
  + Also introduces multiplexing – priority by file TYPE (html -> css -> js –> media)
  + Runs UDP not TCP
  + Html, css, js sent with application level reliability, media less so
* QUIC
  + Connection latency – 0-1RTT
  + Improved transmission
  + Forward error correction
  + Multiplexing without HoL blocking
  + Connection mitigation when switching between networks (like when on bus, from wifi to 4G etc.)
  + Reliable prioritized delivery
  + Encrypted delivery

**Lecture 6 – EMail + DNS**

* SMTP
  + Delivery / storage of emails between mail servers (not users)
  + Clients compose and send email using interface 🡪 email received and queued by client’s mail server 🡪 sends email using TCP to server (receiver of email) 🡪 E-mail placed into mailbox and retrieved using user’s preferred interface app.
  + Uses port 25
  + 3 phases: handshaking, transfer of messages, closure
  + Command (ASCII text)-response (code status) interaction
  + Persistent connection, multiple objects sent in multipart message
  + Syntax:
    - Headers
    - Blank line
    - To
    - From
    - Subject
    - Body message
* IMAP
  + Used to retrieve emails from servers when apps are used.
  + Also delete, update, folders
  + HTTP is used if a web browser is used instead.
* URL – used as web addresses by humans
* IP – used as web addresses by Network layer datagrams
* DNS = Domain Name System
  + Distributed database implemented in hierarchy of name servers
  + Implemented at application level aka edge of network
  + Functions:
    - Hostname to IP translation
    - Host aliasing
    - Mail server aliasing
    - Load distribution
    - Replicated web servers (many IPs corresponding to one host name)
  + Distributed vs centralized
    - Same as CDNs. Centralized :
      * Single point of failure
      * High traffic and bottlenecks
      * Distant central database
      * Difficult maintenance
      * Does not scale – trillions of queries a day per CDN provider now, imagine centralized
  + Many more reads than writes
  + Billions of records, each simple
  + Physically decentralized – millions of organizations are responsible for their records
  + “bulletproof” reliability / security
  + Hierarchy:
    - Root servers (retrieve .com DNS server address)
    - Top Level Domain servers (retrieve amazon.com DNS server address)
    - Authoritative DNS (retrieves IP address for amazon.com from amazon.com SND server)
* Root domain servers
  + ICANN – manages root DNS domain
  + 13 distributed around the world
  + Last resort by servers that cannot resolve name
  + Critical for the internet
  + DNSSEC – security measures
* Top Level Domain (TLD)
  + Responsible for .com, .org, .net, etc.
  + Authoritative registry for some of these like .edu, .com, .net
  + Authoritative DNS servers
    - Owned by organizations
    - Accurate and up to date on their domain
    - Maintained by organization or service provider.
* Local DNS name servers
  + Used efficiently for DNS queries within local network
  + Reduces response time
  + Forwards requests to authoritative DNS servers when necessary
  + Every ISP has one
* DNS name resolution
  + Iterative query
    - “I don’t know this name but you can ask THIS server”:
    - Host sends request to local DNS
    - Local DNS forwards to root DNS server – receives address of TLD DNS server
    - Local DNS forwards to TLD DNS server – receives address of authoritative level DNS server
    - Local DNS forwards request to authoritative DNS server – IP associated with URL is returned
    - More efficient than alternative with each query obtaining a part of the final address
  + Recursive query
    - Rare, harder to maintain, requires more messages as more data is sent per message, heavier load at higher levels of hierarchy
    - Requesting host 🡪 local DNS 🡪 Root DNS 🡪 TLD DNS 🡪 authoritative DNS 🡪 Returns IP address to TLD DNS 🡪 Returns to Root DNS 🡪 Returns to local DNS 🡪 host
* DNS caching
  + Occurs at local DNS server
  + When address obtained by cache it is cached and forwarded to user
  + Cache entries time out
  + Mostly caches TLD servers to avoid going to the root (but caches all anyway)
  + Entries can be out of date – if an address expires, the timeout (TTL) will ensure an updated address on next access.
* DNS records (name, value, type, ttl):
  + Type = A(classic) – host name and IP address – ex (amazon).com: 123.3.3.4, A)
  + Type = NS (name server) – domain name and hostname of authoritative name server – ex concordia.ca: dns1.concordia.ca, NS
  + Type = CNAME (canonical name)– alias name for a real (canonical) name – ex. Concordia.encs.ca: rely.west.concordia.encs.ca, CNAM
  + Type = MX – name of mailserver associated with name – ex. Concordia3.ca: mail.concordia.ca, MX
* DNS messages
  + 16 bit query ID (same num for request and response)
  + Flags Q or R (0 for query 1 for response), recursion desired, recursion available, reply is authoritative
  + #Qs
  + # As
  + # Authoritative
  + # additional
  + Answer follows same format but includes both the question(s) and the answer
* Steps for new entry into system
  + Register with DNS registrar: names, IP of authoritative server (primary and secondary)
  + Registrar inserts 2 entries into DNS: 1. Name, server name, NS 2. Server name, IP address, A.
  + Create authoritative server that stores type A record for new site and MX record for mail service
* DNS security
  + DDoS attacks – bombard DNS servers
  + Redirect attacks – intercepted queries, DNS poisoning (sending bogus replies which catch on)
  + Exploit DNS for DDoS – send queries with spoofed address : target IP.

**Lecture 7**

* P2P architecture
  + No always on server
  + Arbitrary end systems connect
  + Self-scalable
  + Intermittently connected and change IP addresses