CS2105 Finals Cheatsheet v1.0 (2020-11-26)
by Julius Putra Tanu Setiaji, page 1 of 2

1 Computer Networks and the Internet

1.1 What is the Internet

1.2 Network Edge

1.3 Network Core

1.3.1 Circuit switching

Dedicated circuit per call

- call setup required
- circuit-like (guaranteed) performance
- circuit segment idle if not used (no sharing)

1.3.2 Packet Switching

Data sent through the net in discrete chunks

- **Store-and-forward**: entire packet must arrive at a router before it can be transmitted to the next link.
- **Addressing**: each packet needs to carry source and destination information
- Users share network resources
- Resources are used on demand
- Excessive congestion is possible

1.4 Delay, Loss and Throughput in Networks

End-to-end packet delay consisting of 4 sources

1.4.1 4 Sources of Packet Delay

- Nodal Processing (d_{proc}): check bit errors, determine output link, typically < msec
- Queueing (d_{queue}): waiting in queue for transmission, depends on congestion level of router
- Transmission ($d_{trans} = \frac{L}{R}$): L = packet length (bits), R = link bandwidth (bps)
- Propagation ($d_{prop} = \frac{d}{s}$): d = length of physical link, s = propagation speed in medium ($2 \times 10^8 m/sec$)

1.4.2 Throughput

- How many bits can be transmitted per unit time
- Measured for end-to-end communication. Compare with link capacity (bandwidth) only for specific link

1.4.3 Units

- 1 byte = 8 bits
- (-) Prefixes: milli, micro, nano, pico, femto, atto, zepto, yocto
- (+) Prefixes: kilo, mega, giga, tera, peta, exa, zetta, yotta

1.5 Protocol Layers and Service Models

1.5.1 5 Layers

- **Application**: supporting network applications, e.g. FTP, SMTP, HTTP
- **Transport**: process-to-process data transfer, e.g. TCP, UDP
- **Network**: routing of datagrams from source to destination, e.g. IP, routing protocols
- **Link**: Data transfer between neighbouring network elements, e.g. Ethernet, 802.11, PPP
- **Physical**: "on the wire"

1.5.2 ISO/OSI Reference Model

Theoretical only, 2 additional layers between **Application** and **Transport**: **Presentation** (allow applications to interpret meaning of data, e.g. encryption, compression, machine-specific convention) and **Session** (synchronisation, checkpointing, recovery of data exchange)

2 Application Layer

2.1 Principles of Network Applications

2.1.1 Client-Server

- **Server**: waits for incoming requests, provides requested service to client, data centers for scaling
- **Client**: initiates contact with server, typically requests service from server, For web, client is usually implemented in browser

2.1.2 Peer-to-Peer (P2P)

- No always-on server
- Arbitrary end systems directly communicate.
- 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
- Peers are intermittently connected and change IP addresses (complex management)

2.1.3 Requirements of apps

- **Data integrity**: 100% reliable vs some data loss
- **Timing**: some apps require low delay to be "effective"
- **Throughput**
- **Security**

2.1.4 Definition of App-layer Protocols

- **Types of Messages exchanged**, e.g. request, response
- **Message syntax**, e.g. message fields and how they are delineated
- **Message semantics**: meaning of information in fields
- **Rules** for when and how application send and respond to messages

2.1.5 Transport-Layer Protocols

TCP	UDP
Reliable data transfer	**Unreliable** data transfer
Flow control: sender won't overwhelm receiver	**No flow control**
Congestion control: throttle sender when network is overloaded	**No congestion control**
Does not provide: timing, minimum throughput guarantee, security	**Does not provide**: timing, throughput guarantee, security

2.2 Web and HTTP

2.2.1 HTTP

- HyperText Transfer Protocol
- Client/server model
- RFC 1945 (HTTP 1.0), RFC 2616 (HTTP 1.1)
- Over TCP

2.2.2 Persistent HTTP

- Multiple objects can be sent over single TCP connection
- **Persistent with pipelining**: client may send requests as soon as it encounters a referenced object – as little as 1RTT for all referenced objects.

2.2.3 Non-Persistent HTTP

- At most 1 object sent over a TCP connection
- Requires 2 RTTs per object
- Response time = $2 \times RTT +$ file transmission time

Refer to slides for the rest of HTTP

2.3 DNS

- **Distributed, Hierarchical Database**
- **Root Server**: answers requests for records in the root zone by returning a list of the authoritative name servers for the appropriate TLD
- **DNS Caching**: based on TTL
- Runs over **UDP**

2.3.1 Resource Records (RR)

Stores mapping between hostnames and IP addresses, 4-tuple `(name, value, type, ttl)`

- type = A, `name` is hostname, `value` is IP address
- type = NS, `name` is domain, `value` is hostname of authoritative name server for the domain
- type = CNAME, `name` is alias for some canonical name, `value` is the canonical name
- type = MX, `value` is the name of mail server assoc with `name`

2.3.2 DNS Name Resolution

- **Iterative query**: Local DNS server makes DNS requests one by one in the hierarchy
- **Recursive query** (rarely used): each server in the hierarchy asks one server higher in the hierarchy

2.4 Socket Programming

- **IP address** is used to identify a host device
- **Process**: program running within a host, identified by `(IP address :: uint32, port number :: uint16)`
- **Socket**: the software interface between app processes and transport layer protocols

2.4.1 UDP

- No "connection" between client and server.
- Sender explicitly attaches destination IP address and port number to each packet
- Receiver extracts sender IP address and port number from the received packet

2.4.2 TCP

- When client creates socket, client TCP establishes a copnnection to server TCP.
- When contacted by client, server TCP creates a new socket for server process to communicate with that client
- Allows server to talk with multiple clients individually.
- Communicates as if there is a pipe between 2 processes, sending process doesn't need to attach a destination IP address and port number in each sending attempt.

3 Transport Layer

3.1 Transport-layer Services

- Sender: Breaks app messages into **segments**, passes them to network layer
- Receiver: Reassembles segments into message, passes it to app layer
- Packet switches in between: only check destination IP address to decide routing
- Each IP datagram contains source and dest IP addresses

3.2 Connectionless Transport: UDP

- UDP addes very little on top of IP
  - Multiplexing at sender
  - Demultiplexing at receiver
  - Checksum
- UDP transmission is unreliable, often used by (loss tolerant & rate sensitive apps)

3.2.1 Connectionless De-multiplexing

When UDP receiver receives a UDP segment:

- Check destination port number in segment, and direct that segment to the socket with that port number.

3.2.2 UDP Header

16 bits each for: source port number, dest port number, length, checksum

3.2.3 UDP Checksum

- Treat segment as sequence of 16-bit integers
- Apply binary addition, wraparound carry added to the result
- Compute 1's complement to get the checksum

3.3 Principles of Reliable Data Transport

Refer to slides for rdt example protocols

3.4 Connection-oriented Transport: TCP

- **Point-to-point**: 1 sender, 1 receiver
- **Connection-oriented**: handshake before sending app data
- **Full duplex service**: bi-directional data flow in the same connection
- **Reliable, in-order byte stream**: sequence numbers to label bytes

3.4.1 Connection-oriented de-mux

A TCP connection/socket is identified by 4-tuple `(srcIPAddr, srcPort, destIPAddr, destPort)`

3.4.2 TCP: buffers and Segments

- two buffers, send and receive, are created after handshaking at both sides
- **Max Segment Size (MSS)**: typically 1460 bytes, max app-layer data one TCP segment can carry.

3.4.3 TCP Header

```
|1              16             32|
| sourcePort#       destPort#    |
|       sequence number          |
|    acknowledgement number      |
|                                |
|  checksum     |                |
```

- **Sequence Number**: byte number of the first byte of data in a segment
- **ACK number**: sequence number of the next byte of data expected by the receiver
- **Cumulative ACK**: TCP ACKs up to the first missing byte in the tream

3.4.4 TCP ACK Generation, Timeout Value, Fast Retransmission

Refer to Lecture 4,5 Slide 66

4 Network Layer

4.1 DHCP (Dynamic Host Configuration Protocol)

4-step process (yiaddr = your internet address):

- Host broadcasts **DHCP discover** message (src 0.0.0.0:68, dest 255.255.255.255:67, yiaddr 0.0.0.0, txn ID)
- DHCP server responds with **DHCP offer** message (src DHCP.IP:67, dest 255.255.255.255:68, yiaddr, txn ID)
- Host requests IP address: **DHCP request** message (src 0.0.0.0:68, dest 255.255.255.255:67, yiaddr, txn ID)
- DHCP server sends address: **DHCP ACK** message (src DHCP.IP:67, dest 255.255.255.255:68, yiaddr, txn ID)

DHCP server at UDP port 67, client at UDP port 68

4.2 Special IP Addresses

Special Addresses	Present Use
0.0.0.0/8	Non-routable meta-address for special use
127.0.0.0/8	Loopback address. A datagram sent to an address within this block loops back inside the host. This is ordinarily implemented using only 127.0.0.1/32
10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16	Private addresses, can be used without any coordination with IANA or an Internet registry.
255.255.255.255/32	Broadcast address. All hosts on the same subnet receive a datagram with such a destination address.

4.3 CIDR

- Address format: **a.b.c.d/x** (x = no of bits in subnet prefix)
- number of IP addresses = $2^{(32-x)}$
- **Longest prefix match** for matching on forwarding table

4.4 Routing Algorithms

4.4.1 Distance Vector Algorithms

- Routers know neighbours & link costs to them
- Routers exchange local views with neighbours & update own local views iteratively to fixed pt.
- Use Bellman-Ford: $d_x(y) = min_v\{c(x,v) + d_v(y)\}$

4.4.2 RIP (Routing Information Protocol) – DV algo

- Hop count as cost metric (insensitive to network congestion)
- Entries in the routing table are aggregated subnet masks (routing to destination subnet)
- Exchange routing table every 30 seconds over UDP port 520.
- **Self-repair**: if no update from a neighbour router for 3 mins, assume neighbour has failed.

4.5 NAT (Network Address Translation)

4.5.1 Implementation

- Replace (source IP address, port #) of every **outgoing datagram** to (NAT IP address, new port #)
- Store in NAT translation table the mapping from (source IP address, port #) to (NAT IP addressj, new port #)
- Replace (NAT IP, new port #) in destination fields of every **incoming datagram** with (source IP, port #) from NAT translation table.

4.6 The Internet Protocol (IP): IPv4

- Header: 20 bytes (details in L6-7: 45)
- Different links may have different MTU (Max Transfer Unit): max amount of data a link-level frame can carry.
- Too large IP datagrams may be fragmented by routers, reassembled by destination host: same ID, different offsets, until fragmentation flag is 0
- Offset is expressed in unit of 8-bytes.

4.7 ICMP (Internet Control Message Protocol)

Carried by IP datagrams for error reporting + echo request/reply

Type	Code	Description
8	0	echo request (ping)
0	0	echo reply (ping)
3	1	dest host unreachable
3	3	dest port unreachable
11	0	TTL expired
12	0	bad IP header

5 Link Layer

Sends datagram between **adjacent nodes** over a single link.
Possible services:

- **Framing**: encapsulating into a frame adding header & trailer.
- **Link access control**: coordination when multiple nodes share a single link
- **Reliable delivery**: used often on error-prone links (e.g. wireless)
- **Error detection**: caused by signal attenuation/noise – receiver detects error & asks sender for retransmission or drops frame
- **Error correction**: corrects bit error(s) without retransmission

5.1 Error Detection and Correction

5.1.1 Parity Checks

- **Single-bit parity**: can detect single bit errors in data (d data bits with 1 parity bit)

- **2D bit parity**: can detect and correct single bit errors in data, can detect any 2-bit error in data (parity bits for each column, each row, and 1 parity bit for the columns and rows)

5.1.2 Cyclic Redundancy Check (CRC)

D: data bits, G: generator or $r+1$ bits, R: will generate CRC of r bits

- long-division by bit-wise XOR operation without carry or borrow

5.2 Multiple Access Links and Protocols

5.2.1 Channel Partitioning Protocols

- **TDMA** (time-division multiple access): each node gets a fixed length slot, unused slots go idle.
- **FDMA** (frequency-division multiple access): channel spectrum divided into frequency bands, each node assigned a fixed frequency band.

5.2.2 Taking-Turns Protocols

- **Polling**: master node invites slave nodes to transmit in turn (concerns: polling overhead, single point of failure – master node)
- **Token passing**: control token passed from a node to next sequentially (concerns: token overhead, single point of failure – token)

5.2.3 Random Access Protocols

- **Slotted ALOHA**
  - Assumptions: all frames are of equal size, time divided into slots of equal length, nodes transmit only at the beginning of a slot.
  - Listens to the channel while transmitting (channel detection)
  - On collision: node retransmits a frame in each subsequent slot with probability p until success
- **Pure ALOHA**: no slot, no synchronization (increased chance of collision)
- **Carrier Sense Multiple Access**
  - Sense the channel before transmission
  - Collisions can still occur due to propagation delay
- **CSMA/CD** (carrier sensing & deferral)
  - When collision is detected, transmission is aborted
  - Retransmit after a random amount of time
  - Minimum frame size to prevent undetected collisions
- **CSMA/CA** (collision avoidance): receiver needs to return ACK if a frame is received OK (due to hidden node problem)

5.3 Switched Local Area Networks

5.3.1 Link Layer Addressing

- Every adapter (NIC) has a MAC address, 48 bits long
- On receiving a frame, NIC checks if destination MAC address matches its own address. If yes, extracts datagram and passes to protocol stack. Otherwise, discards the frame.

5.3.2 ARP (Address Resolution Protocol)

- Each IP node has an ARP table: stores (IP address, MAC address, TTL) of other nodes in the same subnet
- Sending frame:
  - **In the same subnet**:
    * **Dest IP in ARP table**: create a frame with dest MAC address from ARP table.
    * **Otherwise**: broadcasts an ARP query packet containing dest IP addr (Dest MAC set to FF-FF-FF-FF-FF-FF), receive MAC addr, store to ARP table, then create a frame with dest MAC address.
  - **To another subnet**: Create a frame with router's MAC address and actual dest IP address. Router then move datagram to outgoing link, constructing a new frame with dest MAC address.

5.3.3 Ethernet

- Topology:
  - **Bus** (popular in mid 90s): all nodes can collide with each other
  - **Star** (prevails today): switch in the center, nodes don't collide with each other.
- Ethernet header:
  - Preamble (8 bytes): 7 bytes 10101010 and 1 byte 10101011
  - 6 bytes dest addr
  - 6 bytes src addr
  - 2 bytes type (indicating higher level protocol, mostly IP)
- Ethernet trailer: 4 bytes CRC-32
- Service:
  - **Connectionless**: no handshaking
  - **Unreliable**: no ACK or NAK
  - **Multiple Access**: CSMA/CD with binary exponential backoff
- Ethernet CSMA/CD:
  - On collision, abort, send jam signal
  - Then, binary backoff: after m-th collision, choose random K from $\{0, 1, 2, ..., 2^m - 1\}$
  - NIC waits $K \times 512$ bit times before retransmitting

5.4 Link-layer Switches

- Star topology: hosts have dedicated connection to switch.
- Switch buffer frames and is full-duplex
- **Switch Forwarding Table**: list of (MAC addr, interface, TTL)
- **Self-learning**: switch learns which hosts can be reached through which interfaces:
  - On receiving frame, note down (MAC addr, interface, TTL) of src in forwarding table
  - If dest is found on the table, forward frame to that link.
  - Otherwise, broadcast the frame to all outgoing links.
- Routers vs Switches:
  - Routers check **IP address**, Switches check **MAC address**
  - Both **store-and-forward**
  - Routers **compute routes** to destination, Switches **forward frame** to outgoing link or broadcast

6 Multimedia Networking

6.1 Streaming stored Video

- Client-side buffering and playout delay to compensate for network-added delay, delay jitter
- **Push-based streaming**: via UDP, transmission rate can be oblivious to congestion levels
- **Pull-based streaming**: via HTTP GET at max possible rate under TCP, fill rate fluctuates due to TCP congestion control, retransmissions

6.2 Voice-over-IP

Application sends segment into socket every 20 ms during talkspurt

6.2.1 Losses

- **Network loss**: IP datagram lost due to network congestion (router buffer overflow)
- **Delay loss**: datagram arrives too late for playout (max tolerable delay: 400 ms)
- **Loss tolerance**: depends on voice encoding, loss concealment, 1-10% can be tolerated

6.2.2 Dealing with Jitter

- **Fixed playout delay**
- **Adaptive playout delay**
  - Estimate network delay, adjust playout delay at beginning of each talk spurt.
  - Silent periods compressed and elongated
  - $d_i = (1-\alpha)d_{i-1} + \alpha(r_i - t_i)$ where α is a small constant, e.g. 0.1
  - Estimate avg deviation of delay $v_i = (1-\beta)v_{i-1} + \beta|r_i - t_i - d_i|$
  - For first packet in talk spurt, $\text{playout-time}_i = t_i + d_i + Kv_i$

6.2.3 Recovery from packet loss

Each ACK/NAk takes 1 RTT, alternative is FEC (Forward Error Correction):

- **Simple FEC**: for every group of n chunks, create redundant chunk by XOR-ing n original chunks (increase bandwidth by $\frac{1}{n}$, reconstruct at most 1 lost chunk from $n+1$ chunks)
- **Piggyback lower quality stream FEC**: send lower resolution audio stream as redundant information (non-consecutive loss: receiver can conceal loss)
- **Interleaving** to conceal loss, packet contains small units from differnet chunks.

6.3 Protocols for Real-Time Conversational Applications

6.3.1 RTP (Real-time Protocol)

- **RTP** for media flow (via UDP usually)
- RTCP (real-time control protocol) for out-of-band statistics and control information for an RTP session (via UDP usually)
- **RTSP** (real-time streaming protocol), e.g. Play, Pause (via TCP usually)
- Runs on UDP, details on headers on L10:43
- Advantage: Short end-to-end latency ($< 100 - 500$ms)
- Disadvantages:
  - Special-purpose server for media, keep state (complex)
  - Protocol use TCP and UDP transmissions (firewalls)
  - Difficult to cache data (no web caching)

6.3.2 SIP (Session Initiation Protocol)

- Mechanism for call setup:
  - For caller to let callee know the former wants to establish a call
  - Caller and calee can agree on media type, encoding
  - End call
- Determine current IP address of callee (similar to DNS)
- Call management (add new media streams, change encoding during call; invite others; transfer, hold calls)

6.4 DASH (Dynamic Adaptive Streaming over HTTP)

- Use HTTP protocol to stream media; divide media into small chunks.
- More info on L10:54-57
- Advantages:
  - Server is simple (no state)
  - No firewall problem
  - Standard web caching works
- Disadvantages:
  - Based on media segment transmissions, typically 2-10s long
  - By buffering on client side, DASH doesn't provide low latency for interactive, two-way app.

7 Network Security

Read lecture notes

8 Physical Layer

8.1 Digital Transmission

8.1.1 NRZ (Non Return to Zero)

- **NRZ-L**: absolute voltage level (high = 0, low = 1)
- **NRV-I**: inverts if bit 1 is encountered

8.1.2 RZ (Return to Zero)

3 voltage levels, return to zero halfway through a bit interval, (high = 1, low = 0)

8.1.3 Manchester

Inverts signal in the midle of the bit, ($- \rightarrow + = 1$, $+ \rightarrow - = 0$)

8.2 Analog transmission

- $A\sin(2\pi f t + \phi)$ where A = peak amplitude, f = frequency, ϕ = phase.
- **Channel bandwidth** = frequency range ($f_{hi} - f_{lo}$)
- **SNR (Signal to Noise Ratio)** = strength of signal over noise

8.2.1 Shannon Channel Capacity

- Theoretical max bit rate of a noisy channel
- $C = B \times \log_2(1 + SNR)$ where C = bit-rate, B = channel bandwidth

8.2.2 Analog encoding

Changing A, f, or ϕ.

- **ASK (Amplitude Shift Keying)**: change A to represent 0 and 1, susceptible to noise.
- **FSK (Frequency Shift Keying)**: change f to represent 0 and 1, limited by channel bandwidth.
- **PSK (Phase Shift Keying)**: change ϕ to represent 0 and 1
- **QPSK (Quadrature PSK)**: signal with 4 possible phases for 2 bits data for every signal
- **8-PSK**: 8 possible phases for 3 bits of data for every signal
- 2^k**-QAM (Quadrature Amplitude Modulation)**:
  - A **signal unit** is a combination of amplitude and phase that represents k bits.
  - **Baud rate** = no of signal units per second.
  - **Bit rate** = no of bits receiver receives / second
  - Combines ASK and PSK, check both amplitude and phase to determine data carried.

8.2.3 Examples

- Ethernet, RFID, NFC = Manchester
- USB = NRZ-I
- Digital RV uses DVB-T = QPSK, 16-QAM, 64-QAM
- Wi-Fi = PSK, QPSK, 16-QAM, 64-QAM