CS2105 CheatSheet

by Zachary Chua

Network Edge (Access Network)
- Hosts access the Internet through **access network**
- eg Home/ Institute access networks

Network Core
A mesh of interconnected routers
Data transmitted by
1. Circuit Switching: dedicated circuit per call
2. Packet Switching: data sent through net in discrete "chunks"

Circuit Switching
End-to-end resources **allocated to and reserved for** "call" between source and dest
- call setup required
- circuit-like (**guaranteed**) performance
- circuit segment idle if not used by call
- used in traditional telephone networks
- limited capacity

Packet Switching
Resources are used on demand → no reservation and excessive congestion is possible
Performance not guaranteed
Host sending function
- breaks application message into smaller chunks, known as **packets** of length **L** bits
- transmits packets onto the link at **transmission rate R**
- link transmission rate is known as **link capacity** or **link bandwidth**

Packet Transmission Delay = $\frac{L}{R}$, assuming packet size L bits and link bandwidth R bits/sec

Store and Forward: entire packet must arrive at a router before it can be transmitted on the next link (check packet integrity; if corrupted, drop packet)

Delay and Loss
Loss: - Packets queue in router buffers - wait for turn to be sent out one by one
- if packet arrival rate exceeds departure rate → buffer full and drop packet

4 sources of delay

1. Nodal Processing - check bit error, determine output link
2. Queueing Delay - time waiting in queue for transmission
3. Transmission Delay - L/R, where L is packet length in bits, R is link bandwidth in bps
4. Propagation Delay - d/s, where d is length of link, s is propagation speed in medium

Throughput
How many bits can be transmitted per unit time
- Measured for end-to-end commmunication
- Link capacity (bandwidth) is meant for a specific link

Protocols
Defines format and order of messages exchanged and the actions taken after messages are sent or received

Internet Protocol
1. Application - supports network applications
2. Transport - process to process data transfer
3. Network - routing of datagrams from source to destination
4. Link - data transfer between neighbouring network elements
5. Physical - bits on wire

Application Layer

Structure of Network Application
- Client-Server
- Peer-to-peer

Transport Service an app needs
- Data Integrity
- Throughput - Some apps may need minimum bandwidth (streaming videos)
- Timing - might need low latency
- Security - excryption

Transport Layer Protocol
App-layer protocols ride on transport layer protocols

1. TCP
- Reliable data
- Flow Control: sender won't overwhelm receiver
- Congestion Control: throttle sender when network overloaded
- does not provide: timing, minimum throughput, security

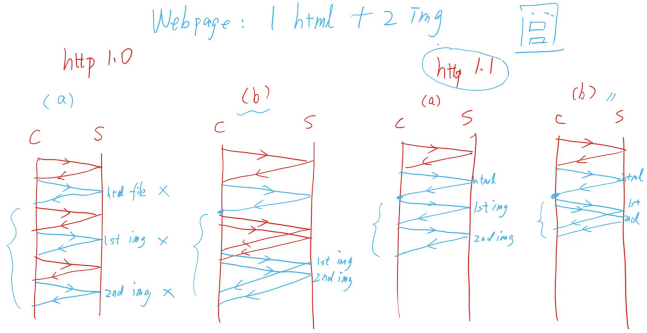
2. UDP
- Unreliable data
- no flow control
- no congestion control
- does not provide: timing, throughput, security

Web
Webpage typically consists of
- base HTML file
- several referenced objects
Each object is addressable by a URL

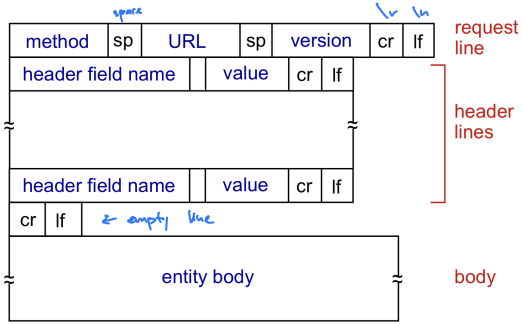
HTTP - Web app layer protocol
C/S model
- client in browser
- server on port 80 (default port number for web servers)
Uses TCP 1. Client initiates TCP connection to server
2. Server accepts TCP connection request
3. HTTP Messages are exchanged over TCP connection
4. TCP connection closed
1 and 2 are known as TCP handshaking
RTT: Round trup time, time for a packet to travel from client to server and go back.

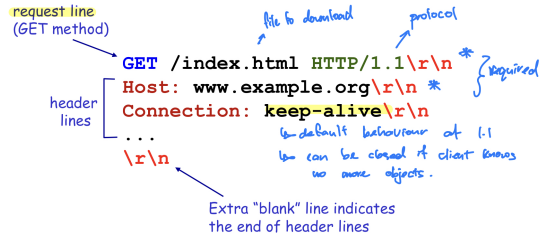
Non-Persistent HTTP
- At most one object sent over TCP connection; TCP connection closed after object sent
- downloading multiple objects requires multiple connections

Persistent HTTP
- multiple objects sent over single TCP connection between client and server
(a): Non-persistent
- $2 * RTT+$ file transmission time for each object (1 RTT for TCP, 1 RTT for object)
(b): Non-persistent with pipelining
- $2 * RTT+$ file transmission time for HTML page
- as little as $2 * RTT+$ file transmission time for other objects (concurrently)
(c): Persistent
- 1 RTT for TCP
- 1 RTT for each object
(d): Persistent with pipelining
- 1 RTT for TCP
- 1 RTT for HTML
- as little as 1 RTT for all referenced object



HTTP Request

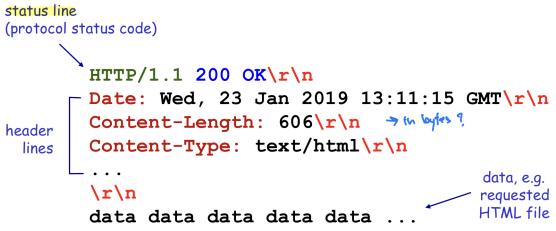




Request Method Types:
1.0: GET, POST (upload input to server), HEAD (leave out requested object)
1.1: GET, POST, HEAD, PUT (uploads file to path specified in URL field), DELETE (deletes file specified in URL field)

HTTP Response

Response Status Codes:
- 200: OK
- 301: Moved Permanently (new location specified later in message)
- 403: Fobidden
- 404: Not found
- 304: Not Modified



Cookies

HTTP designed to be stateless - server maintains no information about past client requests
Cookie: http messages carry state (forever, expiry, memory only)
1. cookie header field of http request/ response messages
2. cookie file kept on user's host, managed by user's browser
3. backend database at website

Conditional Get

- Goal: don't send object if client cache has up to date cached version
- 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; response is 304

Domain Name System

Identify host by: 1. Hostname: eg, www.example.org
2. IP address: eg, 93.184.216.34 (32-bit int)
DNS translates between the two
Client carries out DNS query to determine the IP address corresponding to server name prior to connection
NOTE: 1 hostname could map to many IP addresses (for load balancing purposes)

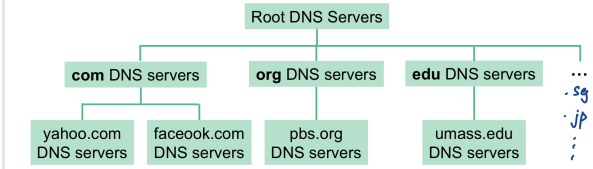
DNS Resource Record (RR)

Mappings are stored as RR
RR format: `(name, value, type, ttl)`

Types: 1. A - name is hostname; value is IP addr
2. NS (name server) -name is domain (eg. nus.edu.sg); value is hostname of authoritative name server for domain.
3. CNAME - name is alias name (eg. www.nus.edu.sg) for some canonical (real) name; value is canonical name (eg. mgnzsqc.x.incapdns.net)
4. MX (mail exchanger) - value is the name of mail server associated with name

Distributed Hierarchical Databases

DNS is stored in RR in distributed databases implemented in hierarchy of many name servers



A client wants IP address for www.facebook.com:
- client queries root server to find .com DNS server
- client queries .com DNS server to get facebook.com DNS server
- client queries facebook.com DNS server to get IP address for www.facebook.com

Root servers
- Answers request for records in the root zone by returning a list of the authoritative NS for the appropriate top level domain (TLD)

Top Level Domain
- responsible for com, org, net, edu, etc and all top-level country domains, eg. sg, jp, etc.

Authoritative Servers
- Organisation's own DNS server(s), providing authoritative hostname to IP mappings for organisation's named hosts (eg. web, mail)

Local DNS server

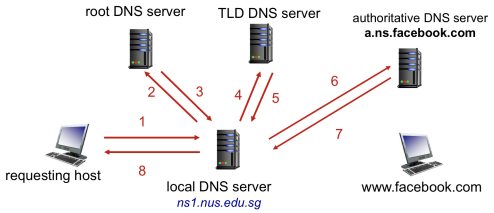
- Does not strictly belong to hierarchy
- Each ISP has one local DNS server (aka default name server)
- When host makes DNS query, query is sent to local DNS server
- Retrieve name-to-address translation from local cache
- Local DNS server acts as proxy and forwards query into hierarchy if answer not found locally

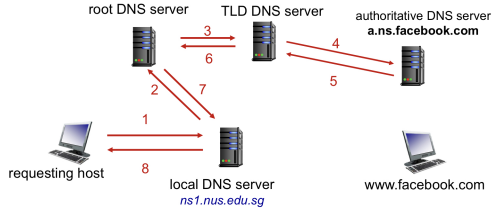
DNS Caching

Caches mapping once it learns of it for some time (TTL)
Runs over UDP

DNS name resolution

iterative and recursive





NOTE: Recursive rarely used as servers cannot respond to other queries until it recieves an answer, eg root DNS cannot answer other queries until TLD server replies

Socket Programming

Process: program running within a host
- Within host, two processes communicate using inter-process communication (defined by OS)
- Processes in different hosts communicate by exchanging messages (according to protocols)
- Identified by (IP address, port number) (port number is a 16 bit int, 1-1023 are reserved)

Sockets

Socket is the software interface between app processes and transport layer - Process send/ receives messages to/ from its socket
- Programming-wise: a set of APIs
Application send and receive messages through sockets (blackbox)

Socket Programming with UDP (datagram socket)

UDP: no connection between client and server
- Sender attaches destination IP addr and port number to each packet
1 socket listens to multiple clients.

Socket Programming with TCP (byte stream-oriented socket)

- Client TCP establishes a connection to server TCP
- When contacted by client, server TCP creates new socket for server process to communicate with that client
- Communicate as if there is a "pipe" between them.
- Just write to pipe, no need to attach dest IP, port to each packet. Pipe there until either process closes it. Pipe is reliable
Server can talk with multiple clients individually.

Transport Layer

Deliver messages between application processes running on different hosts
- TCP and UDP

What do they do
- Sender: break app message into segments, passes them to network (IP)

layer
- Receiver: reassembles segments into message, passes to app layer
- Routers in between: check dest IP to determine routing

Connectionless Transport: UDP
UDP adds the following services (very few)
- Multiplexing at sender: UDP gathers data from processes, forms packets and passes them to IP
- De-multiplexing at receiver: UDP receives packets from lower layer and sends them to right processes
- Checksum
NOTE: UDP transmission is **unreliable**
Reasons for UDP:
- No connection establishment (less delay)
- Simple: no connection state at sender, receiver
- small header size
- No congestion control: UDP can send as fast as desired

Connectionless Demultiplexing
When UDP receiver recieves UDP segment
- Checks dest port no of segment
- Directs UDP segment to the socket with that port number
- IP datagrams (from different soucres) with same dest port no. will be directed to the same UDP socket at dest

UDP Header
Comprises of (8 bytes / 64 bits)
- Source port no. (16)
- Dest port no. (16)
- length in bytes, including header(16)
- checksum (16)

UDP Checksum
To detect errors (flipped bits) in transmitted segment
Sender:
- Compute checksum value
- put checksum value in UDP checksum field
Receiver:
- Compute checksum of received segment
- Check if computed value = checksum
NOTE: Just because checksum = computed value does not mean that there is no error
Computation:
1. Treat segment as a sequence of 16-bit integers
2. Apply binary addition on every 16 bit integer (checksum field is currently 0)
3. Carry (if any) from the MSB will be added to the result
4. Compute 1s complement to get UDP checksum

Reliable Data Transfer
Network layer may:
- corrupt, drop, re-order (not considered in this mod) packets, and deliver packets after delay

rdt 2.0: Bit Errors
Solution: Use checksum to detect bit errors, receiver replies ACK/NAK
- ACK: send next packet
- NAK: resend current packet

Problem: ACK may be corrupted. Sender would resend packet, receiver does not know it is duplicate

rdt 2.1: Bit Errors
Solution: rdt 2.0 with packet sequence number
- ACK/NAK corrupted: Sender retransmits current packet
- Sender adds sequence number to each packet.
- Receiver discards duplicate packets.
rdt 2.2: no NAK, ACK with packet num

rdt 3.0: Errors and Loss
Channel may now:
- flip bits, lose packets, delay between packets
Solution: Sender waits some time for ACK, retransmits if no ACK before timeout
NOTE: If packet or ACK is just delayed not lost;
- Timeout triggers retransmission
- Retransmission generates duplicates, receiver uses seq no to detect
- Reciever must specify the packet seq no of the packet being ACKed
NOTE: Sender does not retransmit on ACK with wrong packet no. (just waits for timeout)

Performance of rdt 3.0 (bad)
- Example: packet size = 8000 bits, link rate = 1 Gbps:

$$d_{trans} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits/sec}} = 0.008 \text{ msec}$$

- If RTT = 30 msec, sender sends 8000 bits every 30.008 msec.

$$\text{throughput} = \frac{L}{RTT + d_{trans}} = \frac{8000}{30.008} = 267 \text{ kbps}$$

- U_{sender}: *utilization* – fraction of time sender is busy sending

$$U_{sender} = \frac{d_{trans}}{RTT + d_{trans}} = \frac{0.008}{30 + 0.008} = 0.00027$$

Pipelined Protocols
Allows multiple, yet-to-be-ACKed packets
Requires more seq no, and buffering at both sides
2 generic forms:
- Go-Back-N (GBN)
- Selective Repeat (SR)

GBN
GBN Sender:
- Can have up to N unACKed packets in pipeline
- use "sliding window" to keep track of unACKed packets
- keep a timer for the oldest unACKed packet
- on timeout, retransmit all packets in sender window

GBN Receiver:
- In order packets: ACK (keep track of the next expected seq no)
- Out of order packets: Discard and ACK the last in order packet
- Cumulative ACK: `ACK m` means all packets up to m are received

SR
SR Sender:
- maintains timer for each unACKed packet
- When timer expires, just retransmit only that unACKed packet

SR Receiver:
- Individually acknowledges all correctly received packets
- buffers out of order packets, for in-order delivery to upper layer

sender
data from above:
- if next available seq # in window, send pkt

timeout(n):
- resend pkt n, restart timer

ACK(n) in [sendbase, sendbase+N]
- mark pkt n as received
- if n is smallest unACKed pkt, advance window base to next unACKed seq. #

receiver
pkt n in [rcvbase, rcvbase+N-1]
- send ACK(n)
- out-of-order: buffer
- in-order: deliver (also deliver buffered, in-order pkts), advance window to next not-yet-received pkt

pkt n in [rcvbase-N, rcvbase-1]
- ACK(n)

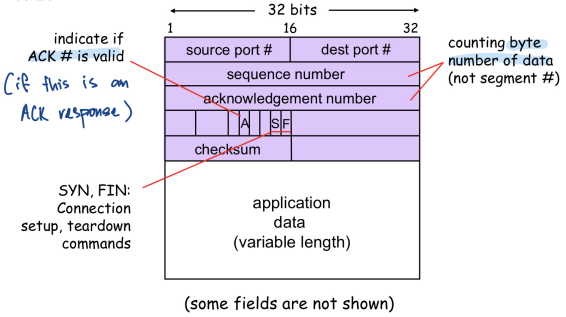
otherwise:
- ignore

TCP
Overview:
- Point-to-Point: 1 sender, 1 receiver
- Connection-oriented: handshaking
- Full duplex service: bi-directional data flow in same connection
- Reliable, in-order byte stream: use seq no. to label bytes

Connection Oriented De-mux:
A TCP connection socket is identified by the 4-tuple (srcIP, srcPort, destIP, destPort)

Segment Size:
A TCP segment typically has an MSS (maximum segment size) of 1460 bytes (app data, without header)

Header:



(some fields are not shown)

Sequence Number:
Byte number of the first byte of data in a segment

ACK (cumulative):
TCP ACKs next expected byte number)
Eg. If segment is [0-999], receiver would ACK 1000
NOTE: TCP does not specify how to handle out of order segments, it is up to implementer.

TCP Sender Events

TCP Sender Events (simplified)

```
loop (forever) {
    switch(event)

    event: data received from application above
        create TCP segment with sequence number NextSeqNum
        if (timer currently not running)
            start timer
        pass segment to IP
        NextSeqNum=NextSeqNum+length(data)
        break;

    event: timer timeout
        retransmit not-yet-acknowledged segment with
            smallest sequence number
        start timer
        break;

    event: ACK received, with ACK field value of y
        if (y > SendBase) {
            SendBase=y
            if (there are currently any not-yet-acknowledged segments)
                start timer
        }
        break;

} /* end of loop forever */
```

Sender keeps one timer only

Retransmit only oldest unACKed packet

first byte of data to be ACKed.

Cumulative ACK

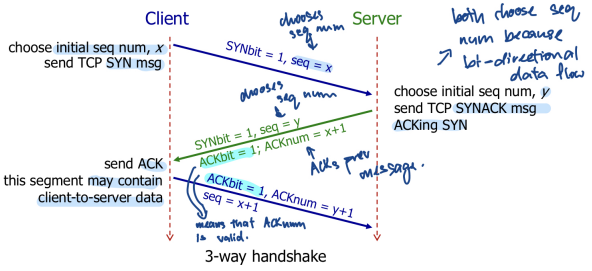
TCP Receiver Events

Event at TCP receiver	TCP receiver action	3 4 5
Arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	Delayed ACK: wait up to 500ms for next segment. If no next segment, send ACK	
Arrival of in-order segment with expected seq #. One other segment has ACK pending	Immediately send single cumulative ACK, ACKing both in-order segments	
Arrival of out-of-order segment higher-than-expect seq. # (gap detected)	Immediately send *duplicate ACK*, indicating seq. # of next expected byte	
Arrival of segment that partially or completely fills gap	Immediately send ACK, provided that segment starts at lower end of gap	

TCP Fast Retransmission

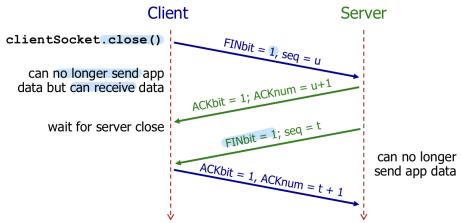
If sender receives 4 duplicate ACKs it supposes the segment is lost.
Immediately resend segment, even before timer expires

Establishing connection - 3-way Handshaking



Closing Connection

- Client, server each close their side of connection.
  - send TCP segment with FIN bit = 1



Network Layer

host to host

IP Address

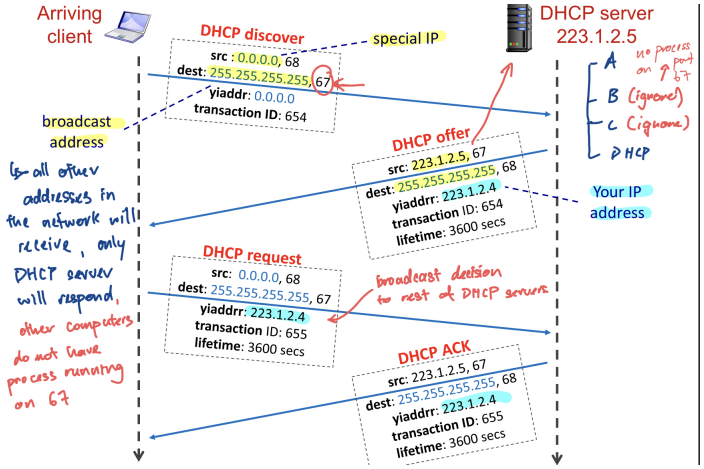
IP Address is used to identify a host or router
32-bit integer expressed in either binary or decimal
Hosts get IP addresses via:
1. Manual configuration by system administrator
2. Automatically assigned by Dynamic Host Configuration Protocol (DHCP) server (more scalable)

NOTE: IP addresses are associated with a network interface
- Eg laptop with ethernet port and wifi will have 2 IP addresses
- routers typically have many interfaces

DHCP

Allows a host to dynamically obtain its IP address from DHCP server when it joins network
- allows reuse of IP addresses
- supports mobile users joining network

4 step process:
1. Host broadcasts "DHCP discover"
2. DHCP server responds with "DHCP offer" (host may get > 1 offer)
3. Host requests IP addr with "DHCP request"
4. DHCP server sends address with "DHCP ACK"



NOTE: DHCP may also provide additional information
- IP address of first-hop router (default gateway)
- IP address of local DNS server
- Network mask

NOTE: DHCP runs over UDP, server on port 67, client port 68

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.

Subnets

IP addresses comprises of two parts:
1. network (subnet) prefix
2. host ID

Subnet is a network formed by a group of "directly" interconnected hosts
- hosts in the same subnet have the same network prefix
- hosts in the same subnet can physically reach each other without intervening router
- They connect to the outside world through a router

CIDR

- subnet prefix is of arbitrary length
- address format: **a.b.c.d/x**, where x is the number of bits in subnet prefix

Subnet Mask

Subnet Mask is used to determine which subnet an IP address belongs to
- Made by setting all subnet prefix bits to 1s and host ID bits to 0s and with IP address to get subnet prefix

Hierarchical Addressing

Allows efficient advertisement of routing information
- instead of having a routing table with 2^{32} entries, use block allocations

Can have smaller and more efficient routing tables

NOTE: These are not subnet masks/ prefixes!!

Routers use **longest prefix match** to determine where to route to

Routing Algorithms - Intra AS routing

- View network as a graph of nodes (routers) and edges (links)
- Each link has an associated cost eg. congestion, avail bandwidth etc.

Goal: find least cost path between two vertices in graph

"link state" algorithms

- All routers have complete knowledge of network topology and link cost; Routers periodically broadcast link costs to each other
- Use Djikstra's Algo to compute least path (using this global map)

"distance vector" algorithms

- Routers know physically-connected neighbours and link costs to neighbours

- Routers exchange local views with neighbours and update own local views based on neighbours'
- Iterative process: repeate above until no change
- Key Difference: Only broadcast link costs to immediate neighbours

Graph Notations:
- $c(x, y)$: cost of link between routers x, y, ∞ if not direct neighbours
- $d(x, y)$: cost of least-cost path between routers x, y

Bellman-Ford Equation

$$d_x(y) = min_v c(x, y) + d_v(y)$$

where min is taken over all direct neighbours v of x
Shortest path from x to y is minimum of least cost to each neighbour of x + least cost from neighbour to y

Algorithm Steps
1. Every router **x, y, z** sends its distance vectors to its neighbours.
2. If y sends a path to z that is cheaper than x currently knows, x will
   2.1 update its distance vector to z accordingly
   2.2 note to send all packets for z to y. Used to create x's forwarding table
3. After several rounds, all routers know least-cost paths to all other routers

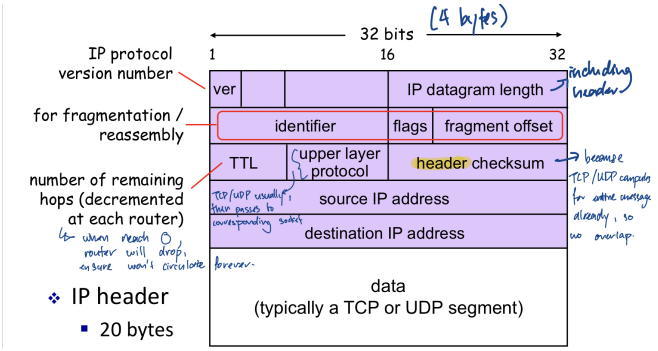
Routing Information Protocol (RIP)
- RIP implements DV algo, using hop count as cost metric (insensitive to network congestion)
- Exchange routing table every 30 seconds over UDP port 520
- "self-repair": if no update from a neighbour router for 3 mins, assume neighbour has failed, remove neighbour entry from table

Network Address Translation (NAT)
Public IP: unique, 2^{32} addresses, many are reserved
Private IP: used within a network, NOT globally unique, ie. can't be used outside network

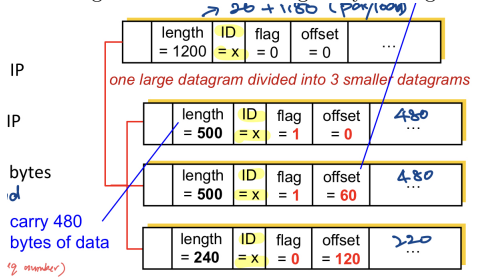
NAT router
NAT routers sit between a local network and wider internet
- Replace (src IP addr, port no.) of every outgoing datagram to (NAT IP addr, new port no.)
- Remember above mapping in NAT translation table
- Replace (NAT IP addr, new port no.) in dest fields of every incoming datagram with corresponding (src IP addr, port no.) in table
Effect: all datagrams leaving local network have the same src NAT IP addr (new port no. are not the same)

Benefits of using NAT
- Rent just one IP addr for NAT router
- All hosts use private IP addr. Can change address of hosts (within network) without notifying outside world
- Can change ISP without changing addresses of hosts in local network
- Hosts insde local network are not explicitly addressable and visible by outside world (security)

IPv4 Datagram Format
header size: 20 bytes



IP Fragmentation and Reassemby
- Different links may have different MTU (Max Transfer Unit): max amount of data a link layer frame can carry (max size of IP datagram)
- IP datagrams that are too large may be fragmented by router



- Flag = 1 if there is next fragment in same segment, 0 if last fragment
- Offset is expressed in units of 8-bytes; first byte num / 8 (relative ordering between fragments)
- ID remains the same for all fragments

Internet Control Message Protocol (ICMP)
Used by hosts and routers to communicate network-level info
- Error reporting eg. unreachable host/ network/ port/ protocol
- Echo request / reply (`ping`)
ICMP messages are carried in IP datagrams (ICMP header starts after IP header)
- ICMP header: Type + Code + Checksum + others.

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

Selected ICMP Type and subtype (Code)

eg. TTL reaches 0, packet discarded, ICMP error message sent to datagram's src addr

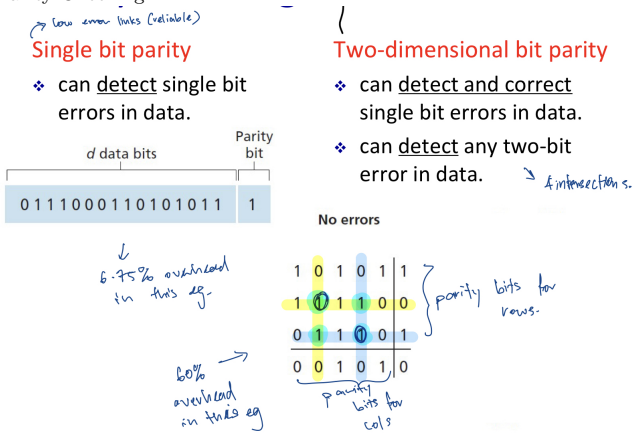
Link Layer
Sending datagrams between adjacent nodes (single link)

IP datagrams encapsulated in link layer frame for transmission
Implemented in adapter (NIC) or chip

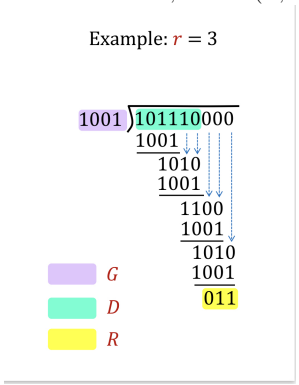
Link Layer Services (protocols may not implement all)
- Framing: encapsulate datagram in fram (adding header, trailer)
- Link access control: coordinate which node can send frames and when, when multiple nodes share common link
- Reliable delivery: seldom on low error (fibre) used on error-prone (wireless)
- Error detection
- Error correction: identifies and corrects bit errors without retransmission

Error Detection and Correction
Link Layer uses mainly Parity Checking and CRC (CRC good but slow in software → not used in above layers)
Error detection schemes are not 100% reliable

Parity Checking



Cyclic Redundancy Check (CRC)
Bitwise `xor` operation without carry or borrow
- D: data bits, viewed as a binary number
- G: generator of $r + 1$ bits, agreed by sender and receiver beforehand
- R: will generate CRC of r bits
Sender sends (D, R)
Receiver knows G, divides (D, R) by G → if non sero remainder, error!



Type of Network Links
1. Point-point link: sender and receiver connected by dedicated link

2. Broadcast link (shared medium): multiple notes connected to a shared broadcast channel
If two or more nodes transmit simultaneously, every node receives multiple frames at same time → frames collide and none correctly read

Multiple Access Protocols
- distributed algorithm that determines how nodes share channel, ie. when a node can transmit
- However, coordination about channel sharing must use channel itself (no out of band channel signalling)

Types of Multiple Access Protocols
Channel partitioning
- divide channel into fixed smaller "pieces" (time slots, frequency)
- allocate piece to node for exclusive use
- Cons: wastage, partitions set beforehand (hard to add new hosts)
- Pros: Good if all have equal amount of data to send

Taking turns
- nodes take turns to transmit

Random Access
- Channel is not divided, collisions possible

Channel Partitioning
Time Division Multiple Access (TDMA)
- Access to channel in "rounds"
- Each node gets fixed length (frame transmission time) slot in each round
- Unused slots go idle

Frequency Division Multiple Access (FDMA)
- Channel Spectrum is divided into frequency bands
- Each node is assigned a fixed frequency band
- Unused transmission time in frequency bands go idle

Taking Turns
Polling
- master node "invites" slave nodes to transmit in turn
- Concerns: polling overhead (not huge), single point of failure (master node)
- Need to be able to detect if master node fails, and choose new master

Token Passing
- Control token is passed from one node to next sequentially
- Concerns: token overhead, single point of failure (token)
- node with token may fail, need algo to generate exactly 1 token

Random Access
Protocols specify:
- how to detect collisions
- how to recover from collisions

Slotted ALOHA
Assumptions
- All frames are of equal size
- Time is divided into slots of equal length (time to transmit 1 frame)
- Nodes start to transmit only at the beginning of a slot

Operations
- Listens to channel while transmitting (collision detection)
- if collision: retransmit in each subsequent slot with prob p until success
- p is recalculated frequently taking things like congestion etc. into account

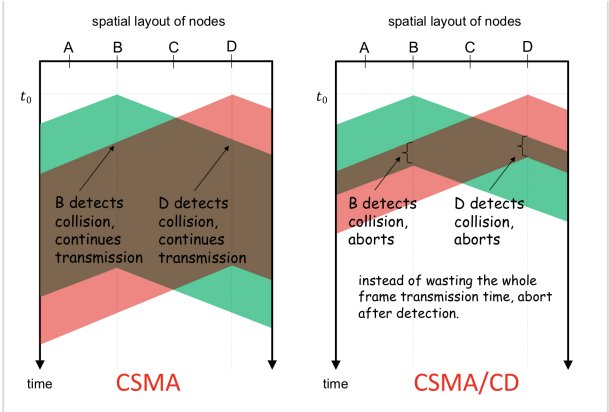
Pure ALOHA
No slots, no synchronisation
- When there is a fresh frame: transmit immediately
- Chance of collision ↑: frame at t_0 collides with frames sent in (t_0-1, t_0+1)

Node may start sending while another node is already sending

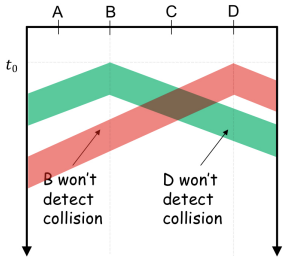
Carrier Sense Multiple Access (CSMA)
Sense the channel, if idle transmit, if busy, defer transmission
But collisions may still exist
- transmit at same time/ before detect another has started due to prop delay

After detect collision, node finishes transmission, then retransmit later

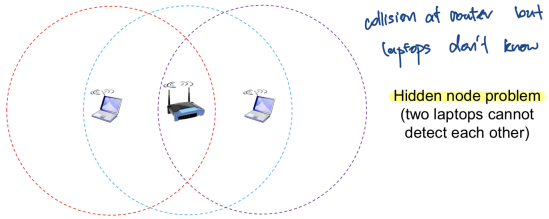
CSMA/CD (collision detection)
- Same as CSMA except that when collision is detected, transmission is aborted immediately (reduces channel wastage)
- Retransmit after random amount of time



Minimum Frame Size
if frame size too small, collision will still appear but may not be detected by sending nodes (no retransmission)
Eg. Ethernet requires min. frame size of 64 Bytes



CSMA/CA (Collision Avoidance)
Collision detection easy in wired LAN (10V vs 5V), but difficult in wireless
Hidden Node Problem:



In CSMA/CA, receiver returns ACK if frame received OK

NOTE: Switches are peer-transparent, just a way of connecting hosts, doesn't manipulate frames going through them

MAC Address
- Every adapter (NIC) has a MAC Address (Media Access Control)
- MAC address is used to send and receive link layer frames.
  - if MAC match: extract datagram and passes to protocol stack
  - if not: discard frame without interrupting host

Used because check is done very fast

- Every MAC Address is gloablly unique, typically hardcoded (burned) into NIC ROM, typically 48 bits

MAC Address vs IP Address

IP address	MAC address
32 bits in length	48 bits in length
network-layer address used to move datagrams from source to dest.	link-layer address used to move frames over every single link.
Dynamically assigned; ←hierarchical (to facilitate routing)	Permanent, to identify the hardware (adapter)
Analogy: postal address	Analogy: NRIC number

Address Resolution Protocol (ARP)
To get MAC address of receiver, knowing its IP
- Each node (host / router) has an ARP table
- ARP table stores the mappings of IP address and MAC address of other nodes in the same subnet
- Only stores nodes that it had communications with, not all nodes in subnet
- Each entry consists of IP addr, MAC addr, TTL

Sending a frame in the same subnet (A→B)
If A knows B's MAC addr (from ARP table)
1 Create a frame with B's MAC addr and send
2 Only B will process this frame
3 Other nodes may receive but ignore this frame

If A does not know B's MAC Addr
1 A broadcasts ARP query packet with B's IP addr
  1.1 Dest MAC addr = FF-FF-FF-FF-FF-FF
  1.2 All other nodes in subnet will receive but only B will reply
2 B replies to A with its MAC Addr
  2.1 Pass up to network layer to check IP addr
  2.2 Replay frame is sent to A's MAC address (included in query packet)
3 A adds B's IP-to-MAC mapping in ARP table (until TTL expires)

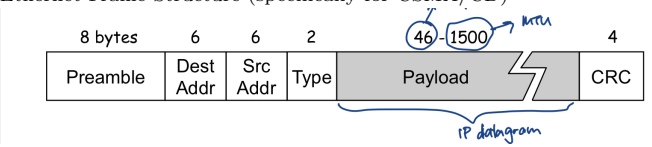
NOTE: Determine if B is in same subnet by checking network mask + prefix
NOTE: Get B IP addr via DNS

Sending fram to another subnet
1. A creates a link layer frame with R's MAC addr (R is default gateway), and B's IP addr as destination
2. R will move datagram to outgoing link and construct a new frame with the next node's MAC addr (not B unless B is next node!)
3. Repeat until reach B

Local Area Network (LAN)
- Computer network within a geographical area, like office or campus
- May be made up of multiple subnets
- Technologies: Token Ring, Ethernet, Wi-Fi, others

Ethernet
Ethernet Topology
- Bus Topology: popular last time when switches were expensive
  - all nodes connect to one ethernet wire
  - all nodes receive every packet → collisions
  - only the node with correct MAC address will respond
- Star Topology: popular now (switches cheaper)
  - switch in the center
  - all point-to-point
  - nodes do not collide with each other

Ethernet Frame Structure (specifically for CSMA/CD)



Preamble:
- 7 bytes with pattern AA_{16} followed by 1 byte of AB_{16}
- used to synchronise receiver and sender clock rates
- tells the receiver the width of a bit (how long each 0 or 1 is)
- important if there is a long string of bits of the same value

Source and Dest MAC addr

Type: Indicates higher layer protocol (usually IP)

CRC

Ethernet Data Delivery Service
- Connectionless: no handshaking
- Unreliable: receiving NIC doesn't send ACK or NAK to sending NIC (higher layers handle retransmission/ data recovery)
- Multiple Access Protocol: CSMA/CD with binary (exponential) backoff

NOTE: Collisions may happen in Ethernet of bus topology

Ethernet CSMA/CD Algorithm
1. NIC receives datagram from network layer, forms frame
2. If NIC senses channel idle, start transmission, if not, wait until channel idle then transmit
3. If NIC sends entire frame without detecting another frame, then done
4. Else, aborts and sends jam signal
5. After aborting NIC enters **binary backoff**

Exponential Backoff
- After 1st collision: choose K at random from 0, 1; wait K * 512 bit transmission times before retransmission
- After 2nd collision: choose k from 0, 1, ..., $2^2 - 1$
- After m^{th} collision, choose K from 0, 1, ..., $2^m - 1$

NOTE: maximum 16 times
- more collisions → heavier load
- longer back-off intervals when more collisions

Ethernet Switch
- Invisible to hosts; hosts think they are directly connected with each other (in star topology)
  - No IP address
- link-layer device used in LANs
  - to store and forward ethernet frames
  - Examine incoming frame MAC address, forward to one or more links
- Switches buffer frames and are full duplex
  - A and D can send frames to each other simultaneously
- No collisions

Switch Forwarding Table
- Tells switch where to send each packet
- Each switch has a table, each entry is (MAC addr of host, interface to reach host, TTL)
- Switches are self-learning, can just plug in and will create the table itself

Self-learning switches
- When receiving a frame from A, note down location of A in switch table
- if B is found in the switch forwarding table, forward frame to that link
- if B is unknown, broadcast to all outgoing links (finds out when B replies)

NOTE: Switches can be connected in hierarchy

Switches vs Routers
Switches (2 layer device - physical and link)
- checks MAC address
- store and forwards link layer frames
- forwards frame to outgoing link or broadcast based on switch forwarding table
- used in LAN to form star topology

Routers (3 layer device, physical, link, network)
- Check IP address
- stores and forwards IP datagrams
- Computes routes to destination
- used in network core

Multimedia Networking

Audio
- Analog audio signal sampled at a constant rate (aka sampling rate)
  - CD music: 44100 samples/sec
- Each sample is quantised
  - eg 8 bit value gives $2^8 = 256$ values
  - $2^{16} = 65,536$ for CD
- Nyquist-Shannon sampling theorem (minimum sampling frequency to be able to recover signal at recv end)

$$f_s \geq d \times 2$$

f_s: sampling frequency
d: highest signal frequency
- Bitrate: sampling rate * number of bits per value

NOTE: There is some quality reduction due to quantisation error
Eg. CD: 16 * 44100 * 2 (stereo has two channels) = 1.411Mbps
Compressing Audio: remove data that is not critical to perception of audio

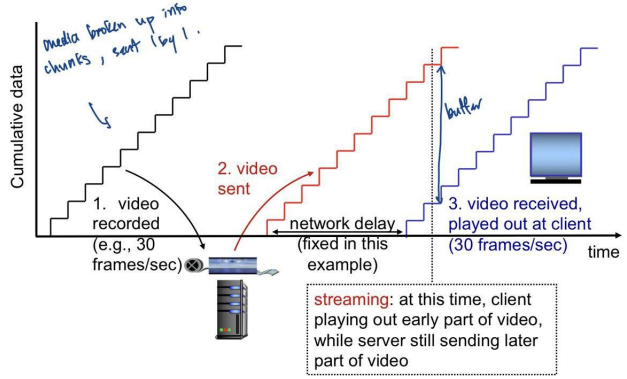
Video
Sequence of images (2D array of pixels) displayed at a constant rate.
Each pixel is represented by bits (RGB values; 8 bits per value)
Coding: use redundancy within and between images to decrease bits.
1. Spatial - adjacent bits have similar RGB value, send value , no of repeats
2. Temporal - send differences between frames

Motion Compensation - is bg moving or object moving? Eg if camera move 10px left, shift frame by 10px left then take diff

NOTE: Redundancy depends on content
CBR: constant bit rate (complex frames have lower quality)
VBR: variable bit rate (all frames have same quality)

Application Types
1. Streaming stored video
2. Conversational (2 way live)
3. Streaming live (1 way live)

Streaming stored video
Note that media is broken up into chunks (packets) and sent 1 by 1



Challenges
- Continuous playout constraint: Once client begins playout, playback must match original timing
  - but network conditions are variable
  - Solution: client side buffer to store packets before playout
  - Solution: Playout delay
- Client interactivity (pause, fast-forward, rewind, jump through video, etc)
- Packets may be lost, retransmitted

Client Side Buffer
- if empties: then stall playout (video buffers)
- if fill: then have to throw away data → stall playout

NOTE: playout rate is constant if CBR, variable if VBR

playout buffering: average fill rate ($\bar{x}$), playout rate (r):
- $\bar{x} < r$: buffer eventually empties (causing freezing of video playout until buffer again fills)
- $\bar{x} > r$: buffer will not empty, provided initial playout delay is large enough to absorb variability in x(t)
  - *initial playout delay tradeoff:* buffer starvation less likely with larger delay, but larger delay until user begins watching

Streaming with UDP/HTTP(TCP)
UDP

- Server sends at rate appropriate for client
  - send rate = encoding rate = constant rate (pushed based / server push)
  - transmission rate can be oblivious to congestion levels (may drop packets)
- short playout delay of (2-5 sec) to remove network jitter
- error recover at application level, if time permits
- UDP may not go through firewalls
- used in Real Time Protocol (RTP)

HTTP (TCP)
- file retrieved via HTTP GET
  - pull based streaming (client pull)
- send at max possible rate under TCP (due to TCP congestion control)
- fill rate fluctuates due to TCP congestion control, retransmission
- larger playout delay: smooth out TCP delivery rate
- HTTP/TCP passes more easily through firewalls
- used in DASH

Voice-over-IP (VoIP)
- Conversational applications
- end-end delay requirement: need to maintain conversational aspect
  - $<$ 150msec: good
  - $>$ 400msec: bad

Characteristics:
- Speaker's audio: alternating talk spurts, silent periods
  - 64kbps during talk spurts, packets only generated during talk spurts
  - 20 msec chucks at 64kbps: 160 bytes of data (uncompressed)
- Application layer header added to each chunk
- chunk+header encapsulated into UDP or TCP segment
- application sends segment to socket every 20msec during talk spurts

Packet losses and delays
1. Network loss - loss due to network congestion (router buffer overflow)
2. Delay loss - IP datagram arrives too late for playout (after 400msec)
   - considered as packet is lost as it cannot be played out anyway
3. Loss tolerance - depending on voice encoding, loss concealment, loss rates of 1-10% can be tolerated

Fixed Playout Delay
- Receiver tries to play each chunk exactly q msec after chunk was generated
  - chunk has timestamp t, then playout time is t + q
  - if chunk arrives after t + q, chunk is considered lost (delay loss)
- large q $\rightarrow$ less packet loss
- small q $\rightarrow$ better interactive experience

Adaptive Playout Delay
Goal: low playout delay, low delay loss rate
Approach: adaptive playout delay adjustment
  - estimate network delay, adjust playout delay at beginning of talk spurt
  - silent periods compressed and elongated
  - chunks still played out every 20msec during talk spurt
- Uses Estimated Weighted Moving Average (EWMA)

$$d_i = (1-\alpha)d_{i-1} + \alpha(r_i - t_i)$$

- d_i: delay estimate after ith packet
- α: small constant eg. 0.1
- r_i: time received
- t_i: time sent (timestamp)

Recovery from packet loss
Challenge: recover from packet loss within the small tolerable delay
- each ACK/NAK requires RTT: retransmission takes too long
- use Forward Error Correction (FEC) instead
  - send enough bits to allow recovery without retransmission

1. Simple FEC
- for every group of n chunks, create a redundant chunk by `XOR-ing` n original chunks
- send $n+1$ chunks, increasing bandwidth factor 1/n
- can reconstruct if at most one lost chunk from $n+1$ chunks, with playout delay

2. Piggybacking
- send lower resolution packet as redundant information by adding it as part of next packet
- Receiver can conceal non-consecutive loss

3. Interleaving to conceal loss
- audio chunks divided into smaller units (eg. 4 5msec chunks)
- packet contains small units from different chunks
- if packet loss, still have most of original packet (3 out of 4)
- no redundancy overhead, but increases playout delay
- not really recovery, does not cover hole (just spreads it / "hides" it)

Real Time Protocol (RTP)
- Specifies packet structure for packets carrying video and audio data
- Runs in sender and receivers (not routers)
- Encapsulated in UDP segments
- interoperability: if two VoIP apps run RTP, they may be able to work together

RTP suite
1. RTP: media data
2. RTCP: status info, transmission stats, QoS
3. RTSP: for commands (play, pause, etc), **runs over TCP**

RTP over UDP
Provides transport-layer interface extending UDP:
- port number, IP addresses
- payload type identification
- packet sequence numbering
- time stamping

Quality-of-Service
RTP does not provide any mechanism to ensure any QoS guarantees (reliable data delivery, etc)
- RTP not seen by routers (only end-systems) no special treatment for RTP packets

RTP Header
Payload type (7 bits)
- type of encoding being used (eg. PCM (uncompressed audio), H.261, MPEG2 video etc)

Sequence no. (16 bits)
- for detecting packet loss, restore packet sequence

Timestamp field (32 bit)
- sampling instant of first byte in RTP data packet

SSRC field (32 bits)
- identifies source of RTP stream (audio or video $\rightarrow$ can have both simultaneously)

Challenges/Advantages of RTP
Challenges
- Special-purpose server for media (complex)
- Protocols use TCP and UDP (firewalls)
- Difficult to cache data (no web caching)

Advantages
- Short end-to-end latency
- still used in WebRTC

Dynamic Adaptive Streaming over HTTP (DASH)
- used in streaming stored video / one way streaming
- Just using HTTP GET to get whole video file can be wasteful also needs large client buffer
- Main Idea: use HTTP to "stream" media, divide media into small chunks (streamlets)

Advantages / Disadvantages of DASH
Advantages
- Simple server (just a web server)
- No firewall problems (use port 80 for HTTP)
- Standard image web caching works
Disadvantages
- DASH is based on media segment transmission, usually 2-10 secs in length (too long for teleconferencing)
- By buffering segments at client side, DASH does not provide low latency for interactive applications

How DASH works
- Encode media into different streamlet files (low, medium, high, etc qualities)
- Server provides a playlist listing available qualities and streamlets for each quality
- Client downloads playlist (.mpd file) first
- Client executes adaptive bitrate algorithm (ABR) to determine which segment to download next
  - based on network quality, after every segment

Network Security
- Confidentiality: only sender, intended receiver understand message
- Authentication: confirm sender and receiver identities
- Message integrity: ensure that message not altered along the way (without detection)
- Access and availability: for services (eg. DDOS)

Bad Actors can:
- Eavesdrop
- Insert messages into connection
- Impersonate: fake src addr of packet
- Hijacking: Remove sender or receiver and insert himself
- Denial of service: prevent service from being used (eg. by overloading resources)

Breaking an encryption scheme
Cipher-only attack
1. Brute force
2. Statistical Analysis
   2.1 Known-plaintext attack: Trudy has corresponding plaintext
   2.2 Chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext
   Convince adversary to transmit your message to get ciphertext
   (use these to figure out encryption scheme)

Symmetric Key Cryptography
Bob and Alice share same (symmetric) key: K_s
- Encrypted message: $K_s(m)$
- To decrypt: $m = K_s(K_s(m))$

Simple scheme
- substitution cipher (swap one letter for another)

Better scheme
- n substitution ciphers $M_1, M_2, \ldots, M_n$
- cycling pattern: eg. 13432, 13432, ...

Public Key Cryptography
- Public key known to all
- Private key known only to receiver
- Eg. Alice send message to Bob
  - encrypt with Bob public key, K_B^+
  - Bob decrypt with his private key, K_B^-
  - $m = K_B^-(K_B^+(m)) = K_B^+(K_B^-(m))$

NOTE: given public key, must be impossible to compute private key

Modular Arithmetics
- [(a mod n) + (b mod n)] mod n = (a + b) mod n
- [(a mod n) - (b mod n)] mod n = (a - b) mod n
- [(a mod n) * (b mod n)] mod n = (a * b) mod n

Therefore, (a mod n)^d mod n = a^d mod n

RSA
- treats message as just a bit patter
- bit pattern can be uniquely represented by integer number
- encrypting message == encrypting number

Public and Private keys:
1. Choose two large (1024 bits each) prime numbers p and q
2. compute n = pq, z = (p-1)(q- 1)
3. Choose e (e < n) that has no common factors with z (e, z are "relatively" prime)
4. choose d such that ed-1 is exactly divisible by z (ed mod z = 1)
5. public key = (n,e), private key is (n,d)

Encryption/Decryption keys
1. Encrypt messsage m (< n), by computing c = m^e mod n
2. Decrypt by computing m = c^d mod n

m = (m^e mod n)^d mod n

NOTE: RSA is computationally intensive (Slow)
Use RSA to exchange session (symmetric) keys, then use symmetric key cryptography

Digital Signature
To prove that the sender is the one who sent the message
Must be verifiable and nonforgeable

How it works
1. get a from Bob message m, with signature ($K_B^-(m)$)
2. Verify is from Bob by applying Bob's public key

Message Digest
Expensive to public key encrypt long messages
Goal: fixed-length, easy-to-compute, digital fingerprint
- Apply hash function H to get fixed size message digest H(m)
- Properties: many-to-1, produces fixed-size, computationally infeasible to find m such that H(m) = digest

Public-key certification
Public keys must really be public, so that no one can change the key-pair (impersonate)

Certification Authorities
CA binds public key to a particular entity, E
Steps for E to register public key with CA:
1. E provides "proof of identity" to CA
2. CA creates certificate binding E to its public key encrypted with CA's private key (digital signature)
3. certificate containing E's public key digitally signed by CA

If Alice wants Bob's public key:
1. Alicee gets Bob's certificate (from Bob or elsewhere)
2. apply CA's public key to Bob's certificate, get Bob's public key

VPN
Motivation: create a private network for security
- Very costly to set up own routers links and DNS infrastructure

VPN: institution's inter-office traffic sent over public network
- encrypted first, separated from other traffic

NOTE: DHCP over VP gives you IP address that is part of organisation's network

Firewalls
Isolates organisation's internal net from larger internet, allowing some packets to pass, blocking others
Reasons for using firewall
- Prevent DDOS attacks: SYN flooding (attacker creates many fake TCP connections)
- Prevent illegal modification/access of internal data
- Allow only authorised access to inside network

Three types of firewalls
1. stateless packet filters
2. stateful packet filters
3. application gateways

Stateless packet filters
- internal net connected via router firewall
- router filters packet-by-packet, deciding based on
  - src and dest IP addr
  - TCP/UDP src and dest port no.
  - ICMP message type
  - TCP SYN and ACK bits

Eg. Block inbound TCP segments with ACK = 0
- blocks external hosts from making TCP connections with internal hosts, but allows internal hosts to make connections with outside hosts

Stateful packet filtering
Tracks status of every TCP connection
- track SYN, FIN: determine whether incoming, outgoing packets "make sense"
- timeout inactive connections at firewalls: no longer admits packets

Application Gateway
Filters packets on application data as well as on IP/TCP/UDP fields
Example:
1. require all telnet users to telnet through gateway
2. for authorised users, gateway sets up telnet connection to dest host. Gateway relays data between 2 connections
3. router filter blocks all telnet connections not originating from gateway

Limitations
- IP spoofing
- multiple apps need their own gateway
- client software must know how to contact gateway
- filters often use all or nothing policy for UDP