Computer networks
Theory

Classes

A ⇒ {1.0.0.0 → 127.255.255.255} – 2^7 networks /8

B ⇒ {128.0.0.0 → 191.255.255.255} = 2^{14} networks /16

C ⇒ {192.0.0.0 → 223.255.255.255} – 2^{21} networks /24

D ⇒ {224.0.0.0 → 239.255.255.255} ⇒ multicast range

E ⇒ {240.0.0.0 → 255.255.255.255} ⇒ never used

Longest routing table size = $2^7 + 2^{14} + 2^{21}$

Private networks:

A ⇒ 10.0.0.0/8 ⇒ {10.0.0.0 → 10.255.255.255}
B ⇒ 172.16.0.0/12 ⇒ {172.16.0.0 → 172.31.255.255}
C ⇒ 192.168.0.0/16 ⇒ {192.168.0.0 → 192.168.255.255}

OSI layers (Open Systems Interconnection): (top to bottom)

⑦ Application layer: * http, dns (example of protocols)
* provides services to user

⑥ Presentation layer: * encryption and decryption for secure data transmission
* jpeg, gif, ascii (example of protocols)

⑤ Session layer: * manages communication sessions between apps, it ensures synchronization
* NetBIOS, RPC, SMB

(4) Transport layer: * reliable data transfer between two systems, controls flow, error checking, retransmission
* UDP, TCP

(3) Network layer: * routing data from the source to destination across multiple networks, forwarding data packets
* IP, ICMP (pings, errors, signals), RIP

(2) Data link layer: * physical transmission of the data on the network, provides error detection and correction, ensures correctly formatted data, flow control
* Wi-Fi, Ethernet, PPP

(1) Physical layer: * actual transmission of raw data over the network
* cables (ethernet cables, fiber optics, radio freq.), interfaces (ex: switch, hub)

TCP/IP layers:
(4) Application layer: HTTP, DNS, ...
(3) Transport layer: TCP, UDP
(2) Internet layer: IP, ~~[illegible]~~ ICMP, ARP (resolves IP addr. to MAC addr.)
(1) Link layer: Ethernet, Wi-Fi, ARP

MAC (media access control) = unique network card 6 bytes address, in the data link layer, can be changed
↳ Broadcast Addr.: FF.FF.FF.FF.FF.FF

Localhost = 127.0.0.1, not a NA or BA, can be default gateway but not DNS

0.0.0.0 = valid mask

Bandwidth: property of transmission medium, represent the amount of data which we transmit over a quantity of time

IPV6 = 16 bytes
IPV4 = 4 bytes

(bitwise)
Operations on network * random IP in a network AND mask = NA
* NA AND mask = NA (for checking)
* BA = NA OR !mask

Supernetting: multiple routing entries become a single one
ex: /27 networks to obtain /24 network ⇒ 27-24=3 ⇒ $2^3 = 8$

Subnetting: borrow bits from mask ; 2^x subnets
ex: /24, borrow 1 bit ⇒ /25 ⇒ 2^1 subnets
⇒ $2^7 - 2$ hosts

Metric = nr. of routers that have to be passed in order to reach destination
⇒ 1 = no routers are passed

Proxy server: intermediary for requests from clients seeking resources from other clients

DHCP * dynamic host configuration protocol
* if router acts as a relay agent ⇒ DHCP server can relay IP addr. on another network
* we can have more DHCP servers in the same subnet if each has its own, distinct pool of addr.
* uses UDP at the transport layer

Cables: * crossover ⇒ switch → switch
switch → hub
hub → hub
router → router
router → pc
pc → pc

* Straight through ⇒ router → switch
switch → pc/server
hub → pc/server

Switch sends packet to destination only
Hub broadcasts ~~all~~ the message to all the network

Switch can transport UDP / IP / TCP packets
Hub does not understand MAC addr, switch does
~~Them~~ Firewall - 2 pc can't ping if firewall enabled on both

RIP = routing information protocol
RIPv1 doesn't support classless routing protocols, but has same timers as V2

IP datagram header: 20 bytes
* Version: 4 bits (ipv6, ipv4)
* Header length: 4 bits (how many 32-bits entities)
* Type of service: 8 bits
* Length: 16 bits (of the entire datagram, max ip datagram size = 64 KB)
* 16-bit identifier
* Flags: - DF = don't fragment, if set to 1 and packet not fit ⇒ not sent
  - MF = more fragments, set to 1 when packet is split
* 13-bit fragment offset ⇒ if a packet doesn't fit on a connection ⇒ fragmented, not reassembled until destination
* TTL - time to live: 8 bits = nr. routers before discarding datagram, decremented when passing through a router, if it reaches 0 ⇒ discarded and signal
* upper layer: 8 bits = which protocols are transported inside
* Header internet checksum: 16 bits = 16 bit one's complement of one's complement sum of all 16 bit words in the header ⇒ initial value 0
* Source IP: 32 bits
* Dest. IP: 32 bits

MTU = maximum transfer unit
ARP: address resolution protocol ⇒ determines the destination MAC address given its IP, uses broadcast

NAT = network address translation
* 64K simultaneous connections for TCP
* 64K simultaneous connections for UDP with a single LAN
* Outside see only 1 IP

UDP = user datagram protocol
* means to message communication
* header = 8 bytes (just for it + 20 bytes from IP + other from app layer)
  - source port + dest. port (16 bits each)
  - length = 16 bits, entire datagram
  - checksum = 16 bits, computed over header + UDP + IP
* datagram integrity only checked when reaching the (final) destination
* No congestion control, so it can overflow
* Datagram delivery not guaranteed

TCP = transmission control protocol
* ordered data transfer, retransmission of lost packets ⇒ error free
* flow control, so no overflow
* it writes to a stream of bytes, while UDP writes packets
* header = 20 bytes (+20 from IP + other from app layer)
  - source port + dest. port (16 bits each)
  - seq. number = 32 bits, counts the amount of bytes exch. over the connection
  - acknowledgement number = 32 bits, index of the next expected byte
  - header length = 4 bits, how many 32 bit entities (data offset)
  - flags = 6 bits
    ACK: 1 if acknowledgement number is ok
    SYN: synchronize when creating the connection
    FIN: final (when closing a connection)
  - window size = 16 bits, flow control (how much space left in buffer)
  - checksum = 16 bits, same as IP, also uses header + data from IP + TCP
  - urgent pointer = 16 bits, not been used

When initialised, a TCP connection needs a state, at the Kernel level for both sender /receiver

* starting sequence number (sent)
* received sequence number
* 2 buffers (queue) → one for sending one for receiving

Receiver window: a segment is retransmitted if a timer expires

Congestion: too much data is sent too fast for the network to handle
→ lost packets, long delays

* congestion window: starts from 1
  - → when below a threshold: grows exp. (slow-start phase)
  - → when above: grows linearly, imposed by the sender
  - → triple duplicate ACK occurs ⇒ threshold = $cw/2$, cw = threshold
  - → timeout occurs ⇒ threshold = $cw/2$, $cw = 1$

Mandatory calls	TCP	UDP
Client	connect, socket	socket
Server	accept, listen bind, socket	bind, socket