Computer Networks Lecture Notes

▼ Quick Links

Computer Network/LAN

Definition

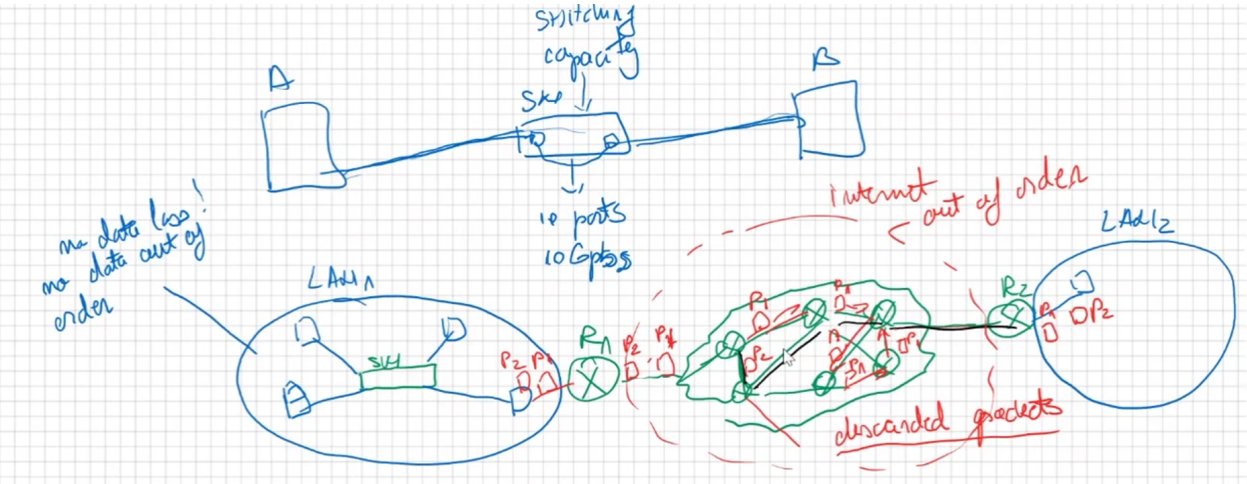
- a collection of computers (PCs, Workstations)
- Lan: components with the same ip prefix

Components

- hosts (computer)
- links (coaxial cable, twisted pair - ethernet, optical fiber, radio - wifi, sattelite)
- switch
  - communicate a to b, for the moment the ports are connected
  - always knows where ports are by learning their MAC address
  - can work as a hub through broadcast
- router
  - interconnecting networks
- hubs
  - interconects all the ports
  - collision span largen then a switch → more lost data
  - dumber then switch
  - mirror the signal on all the ports
- access points

Communication

- stations communicate by sending messages called Frames - electrical signal
- frames may pass through multiple switches; each switch read the frame and passes it on
- signal on the wire is never lost



Type of links

-
  - access links
    - connect stations to the first switch
    - usually copper wire
  - trunk links
    - connect switches
    - often optical fiber
- Direct Links
  - point to point communication
- Bus type links
  - multiple access

Types of transmission

- switched networks
  - circuit switched networks: public telephone network
    - set up a connection path betwenn the source and the destination (permanent for the lifetime of the connection
    - network resources (e.g. bandwidth) divided into pieces → allocated to calles
    - resource piece idle if not used by owing call
    - all bytes follow the same dedicated paht
    - while A talks to C, B cannot talk to D on the same line
  - packed swittched networks: internet (collection of networks)
    - each end-end data stream divided into packeds
    - allows more users then circuit switching
    - user A,B packets share network resources
    - eack packet uses full link bandwitdth
    - resources used as needed
    - resource contention
      - aggregate resource demand can exceed amount available
      - congestion: packets queue, wait for link use
      - store and forward: packets move one hop at a time
        - transmit over link
        - wait turn at next link
      - statistical multiplexing
        - nobobdy reserves a lane on the freeway
        - sequence A&B packets does not have fixed pattern
      - message segmenting
- frame relay

- asynchronous transfer mode

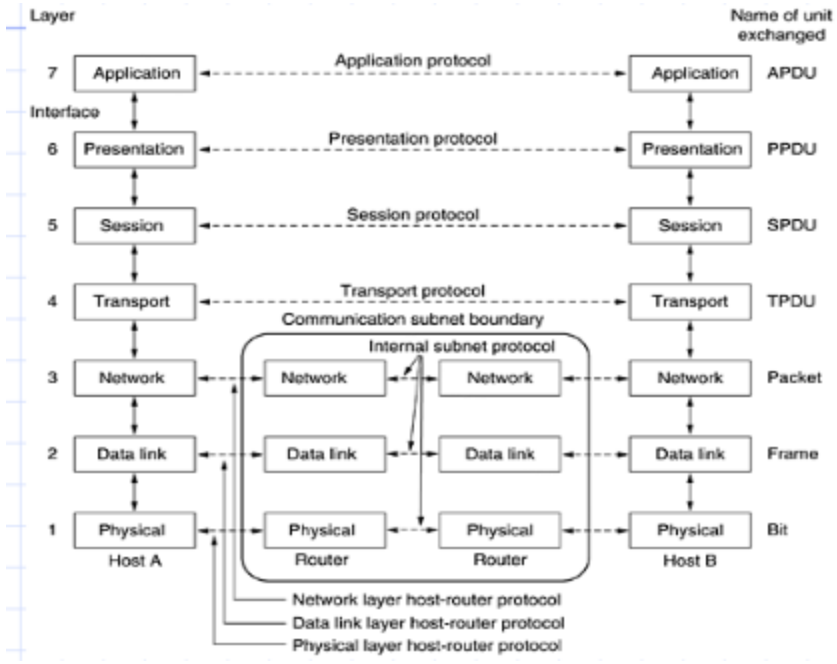
Communication protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Dialpad)	typically UDP

- protocol = agreement about communication
- specifies
  - format of the messages
  - meaning of the messages
  - rules of exchanges: who and when
  - procedures for handling problems (errors)
- need for protocols
  - hardware is low-level
  - problems that can occur
    - bits corrupted or destroyed
    - entire packet lost
    - packet is duplicated
    - flow control

- protocol hierarchies
  - networks organized as stack of layers
    - reduce complexity
    - each layer offers services to higher layers
  - equivalent to data abstaction
  - network arhitecture = a set of layers and protocols

The Open System Interconnection (OSI) Reference Model



Principles

- a layer should be created where a different abstraction is needed
- each layer should perform a well-defined function
- the function of each layer should be chosen with an eye toward defining internationally standardized protocols

- the layer boundaries should be chosen to minimize the information flow across the interface
- the number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldly

The Physical Layer

- unit of transmision = signal
- raw bits over a communicaition channel
- data representation
  - 1 - how many volts ?
  - 0 - how many volts?
- 1 bit - how many nanoseconds?
- electrical, mechanical, timing interfaces

Data Link Layer

- unit of transmission = frames → thousands of bytes
- Turn the raw transmission into an error free communication line
- handles traffic regulation (flow control)
- access to the medium in broadcast shared communication lines

The Network Layer

- unit of transmission = packet or datagram
- control the operation of a subnet
- how packets are routed from source to destination
- quality of service → data priority
- congestion control
- fragmentation and inter-network problems

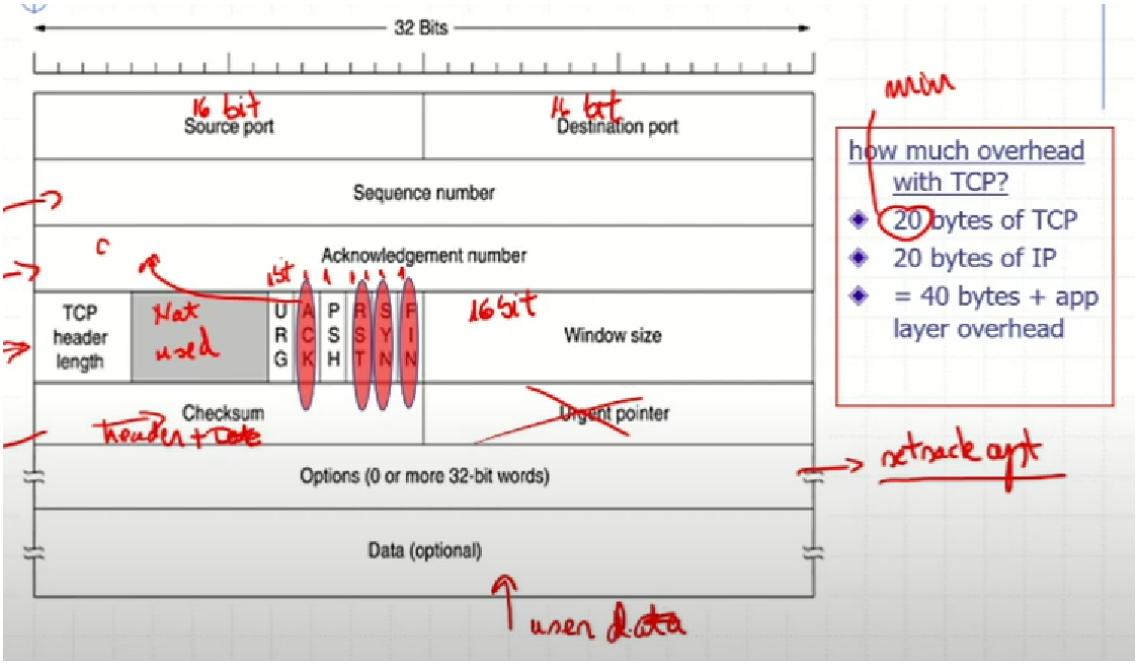
Transport Layer

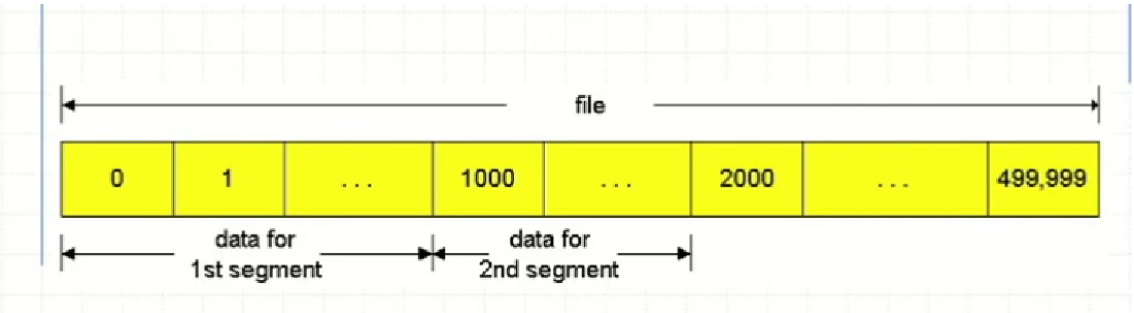
- unit of transmission = packets or segments (sessions)
- accept data stream from upper layers and splits it into packets (small units)
- ensure that packets arrive correctly to the other end → reliability
- type of service: error free PtoP, preserve order or not, guarantees delivery or not, broadcast
- true end-to-end layer
- timing
- bandwidth

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
		same as above	yes, few secs
stored audio/video	loss-tolerant	few kbps up	yes, 100's msec
interactive games		elastic	yes and no
instant messaging	loss-tolerant		

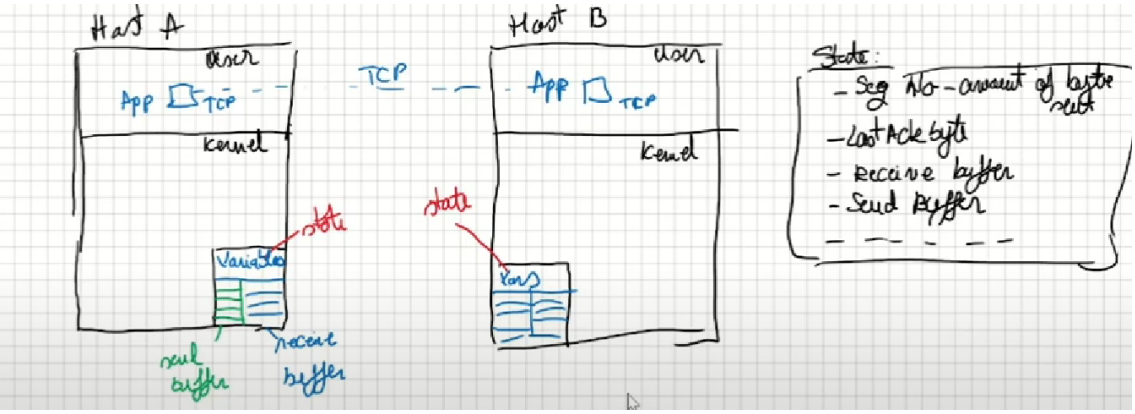
- TCP
  - connection oriented: setup required between client and server processes
  - advantages
    - reliable transport between sending and receiving process
    - flow control: sender wont overwhelm receiver
    - congestion control: throttle sender when network overloaded
  - doesnt not provide: timing, minim bandwidth guarantees

- ordered data transfer - the destination host rearranges data according to the sequence numbeer
- retransmission of lost packets - any cumulative stream not acknowledged is retransmitted
- error-free data transfer - consequence of the above
- flow control (Window based) - limits the rate a sender transfers data to guarantee reliable delivery. The receiver continually hints the sender on how much data can be received (controlled by the sliding window). When the receiving host's buffer fills, the next acknowledgement contains a 0 in the window size, to stop transfer and allow the data in the buffer to be processed
- handles
  - data loss → each sent segment needs confirmation if not re-transmission
  - out of order → data buffering on reception
  - data flow → infer network and peer status
  - network congestion → infer network status
  - all of the above → keep status on each side and initialize it at beginning
- TCP Segment

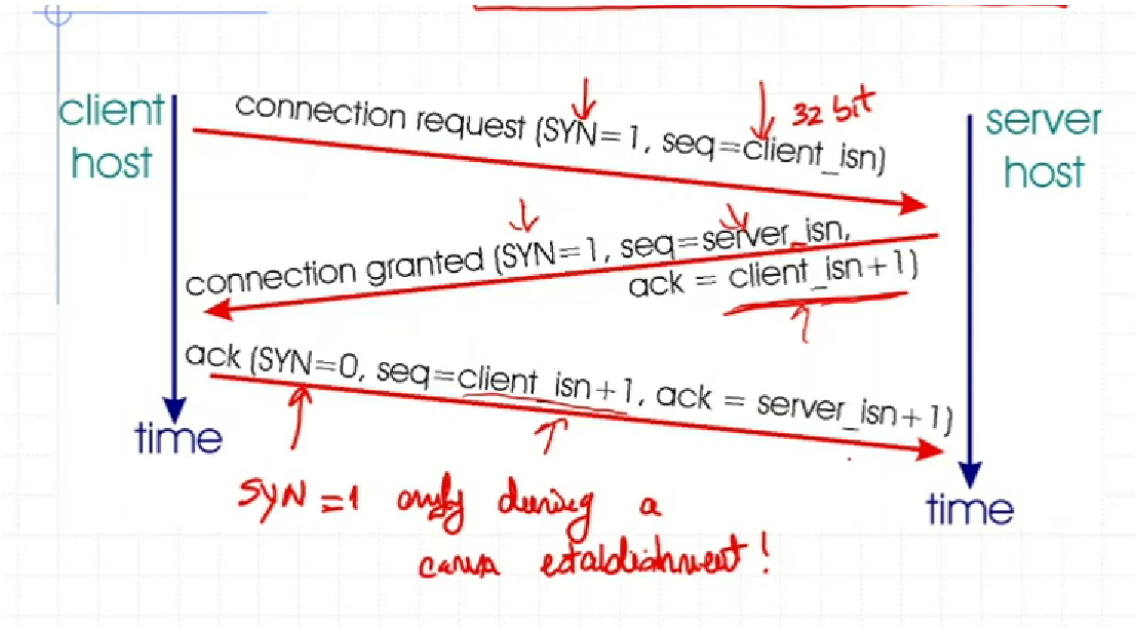




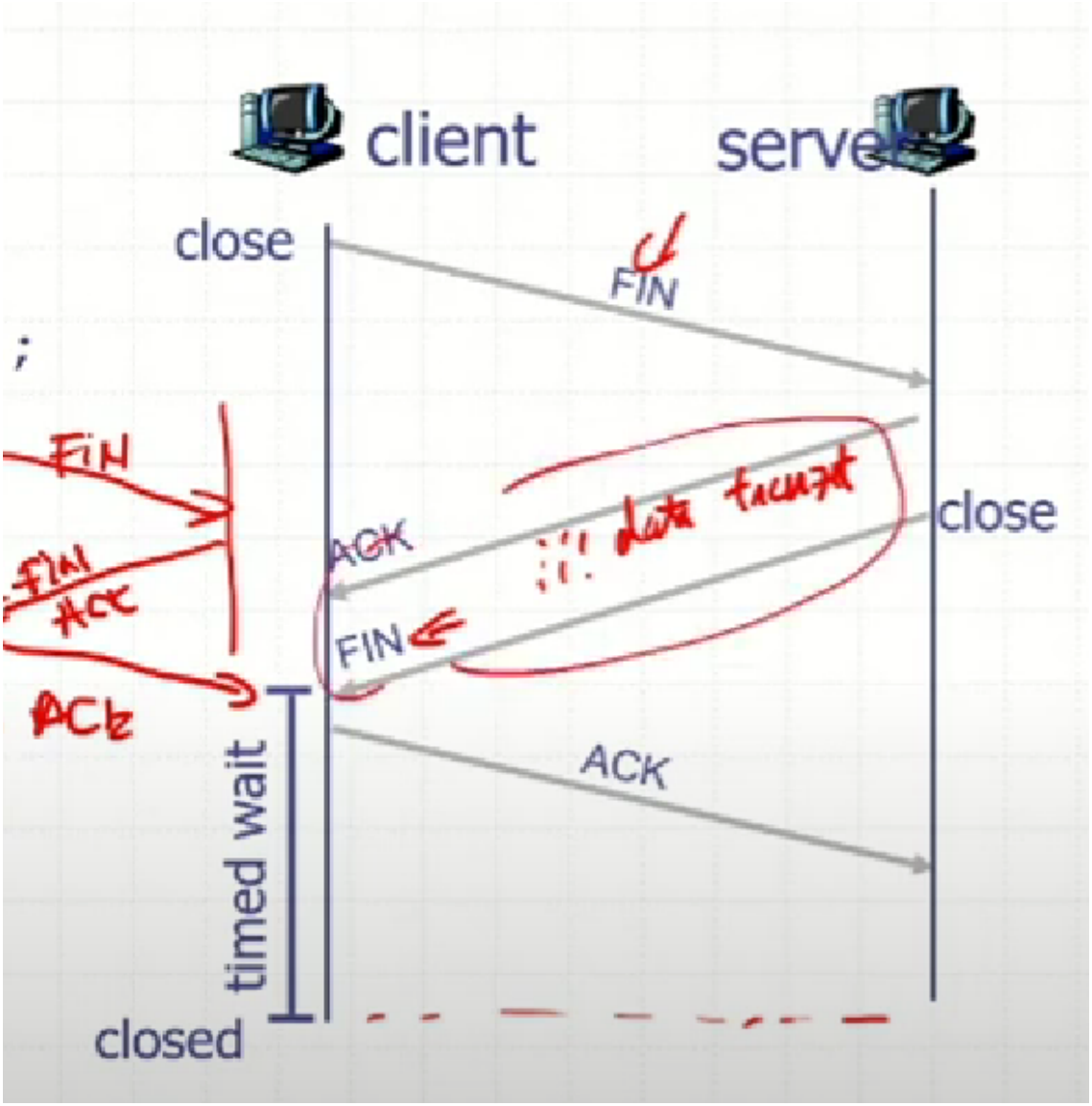
- data is a contiguous stream split into segments
- each segment is carried into one (or multiple) IP datagrams
- each segment needs an acknowledgment from receiver
- send/receive operations act on the stream they are not paired to each-other (i.e. one read for multiple senders



- TCP - open 3-way handshake a.k.a connect call



- TCP Connection Teardown
  - closing a connection ↔ clientSocket.close();

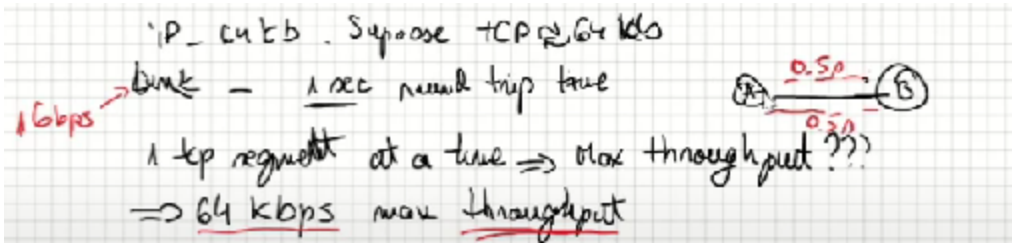


1. client end system sends TCP FIN control segment to server
2. server receives FIN, replies with ACK,
3. server sends the remaning data then FIN to client
4. client receives the remaing data and FIN, replies with ACK, waits some time for all data to arrive

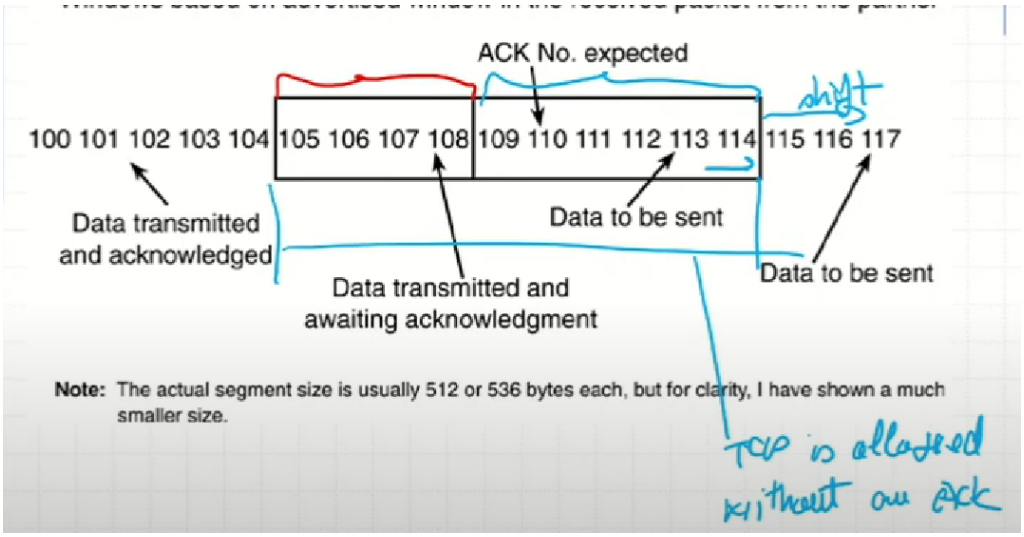
- Seq numbers and Acks
  - sequence numbers are used to reassemble data in the order in which it was sent

- sequence numbers incremented based on the number of bytes in the TCP data field → byte sequencing protocol
- each segment transmitted must be acknowledged
  - multiple segments can be acknowledged at a time
- the ack field indicates the next byte(sequence) number the receiver expects to receive
- the sender, no matter how many transmitted segments, expects to receive an ack that is one more than the number of the last transmitted byte

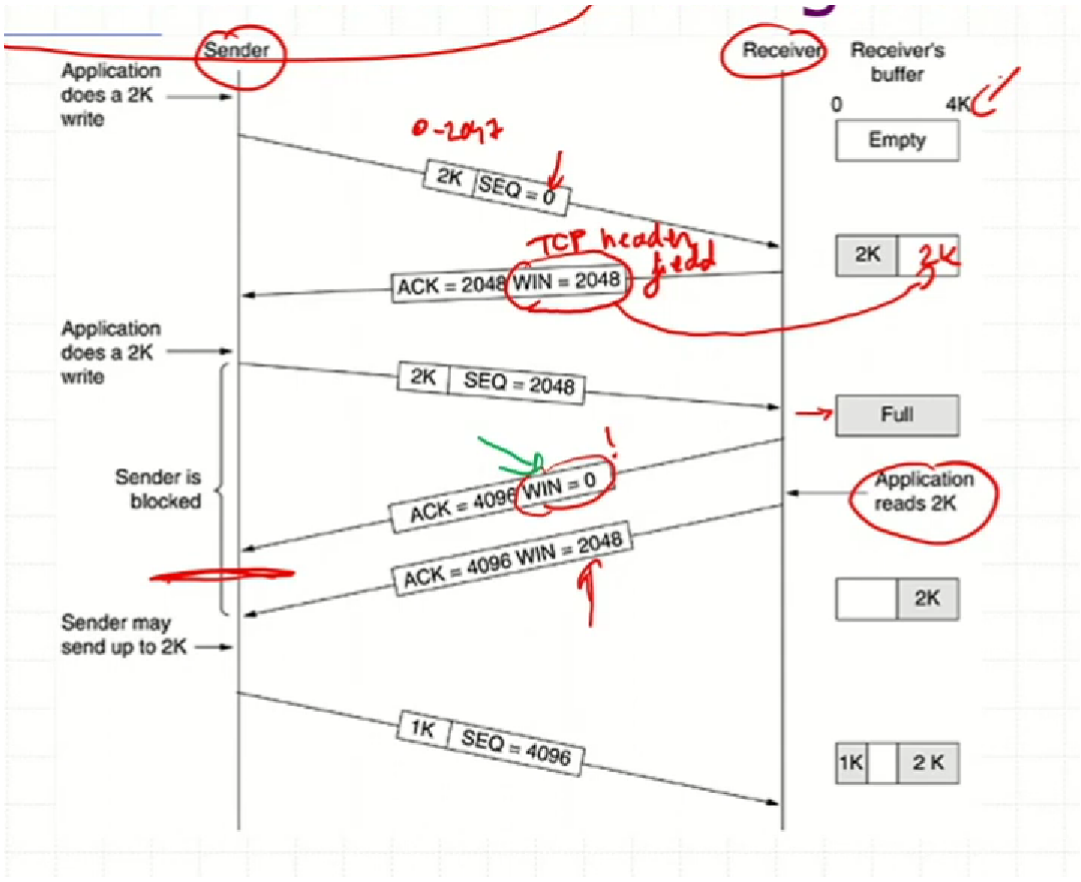
- Flow Control
  - TCP (Send) Sliding Window
    - problem if we send data one at a time



    - solution: sliding window



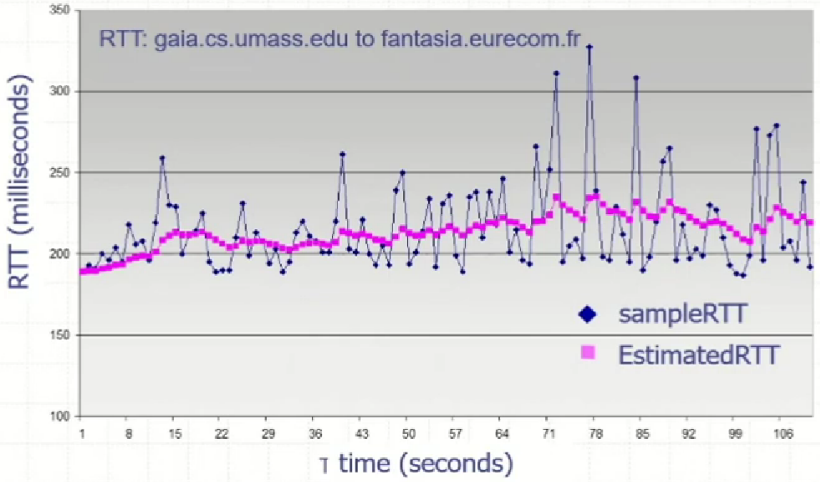
  - TCP (Receive) Window Managment



- TCP Retransmission
  - TCP will retransmit a segment upon expiration of an adaptive transmission timer
  - the timer is variable
  - when TCP transmits a segment, it records the time of the transmission and the sequence number of the segment
  - when TCP receives an acknowledgment, it records the time
  - this allows tcp to build a sample round-trip delay time → RTT
  - TCP will build an average delay time for a packet to be sent and received
  - the timer slowly changed to allow for the varying differences in the internet
  - estimatedRTT +4DevRTT = Timeout Interval

EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT

- exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value: α = 0.125



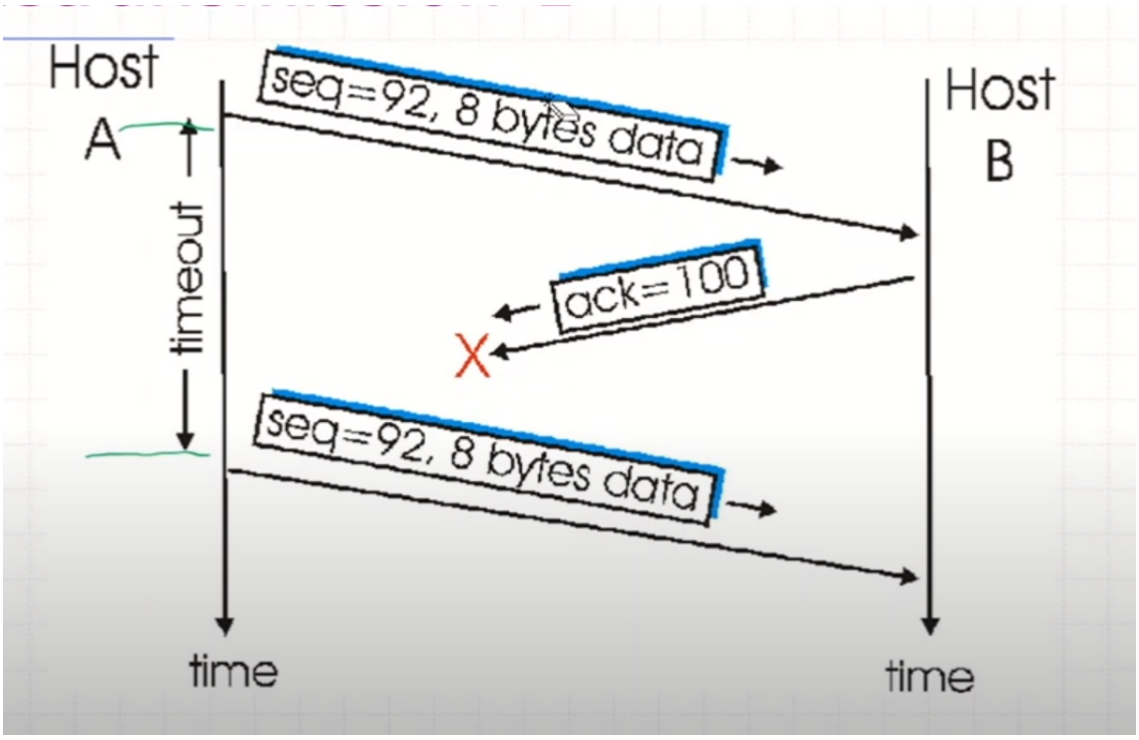
3-32

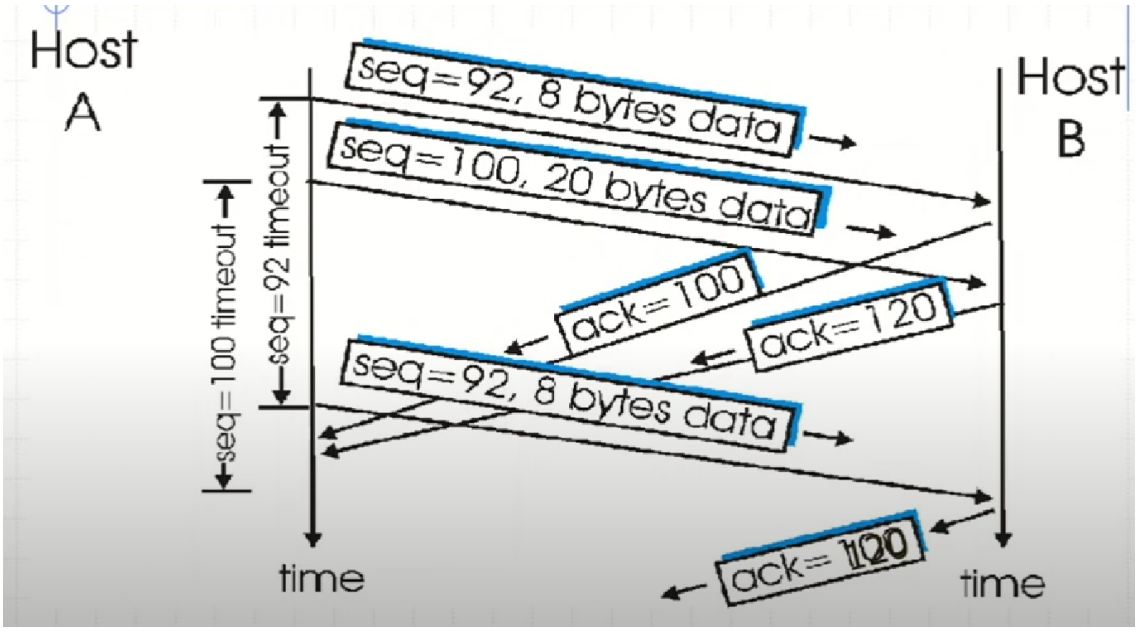
$$EstimatedRTT = (1-\alpha)EstimatedRTT + \alpha\, SampleRTT$$

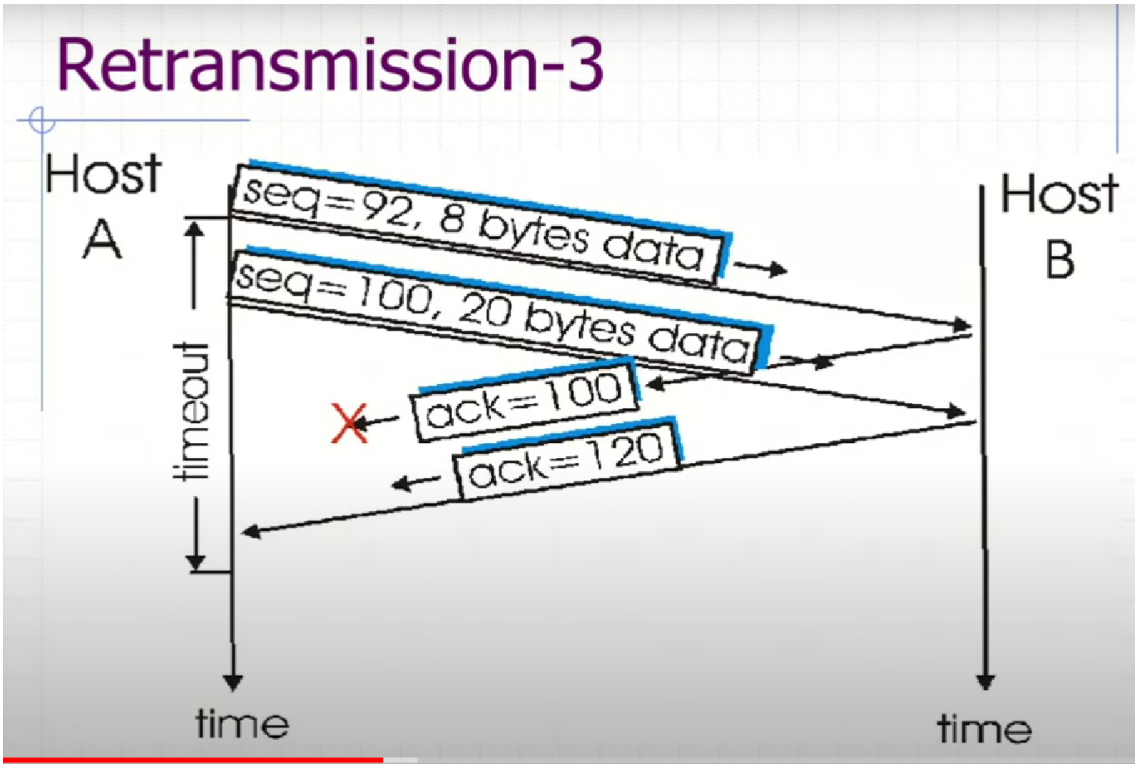
$$DevRTT = (1-\beta)DevRTT + \beta \mid SampleRTT - EstimatedRTT \mid$$

TimeoutInterval = EstimatedRTT + 4 DevRTT

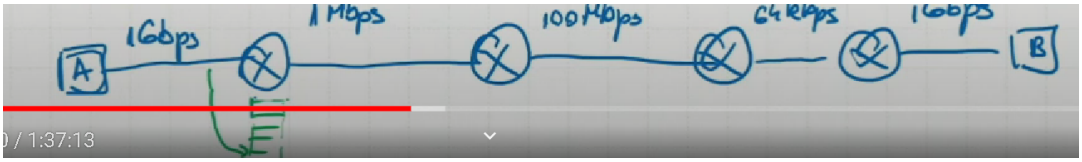
- Examples


Host
A
Host
B
seq=92, 8 bytes data
ack=100
X
timeout
seq=92, 8 bytes data
time
time


Host
A
Host
B
seq=92, 8 bytes data
seq=100, 20 bytes data
ack=100
ack=120
seq=92, 8 bytes data
seq=100 timeout
seq=92 timeout
time
ack=120
time



- Congestion Control
  - Congestion
    - informally: too many sources sending too much data too fast for network to handle
    - different to flow control
    - manifestation:
      - lost packets (buffer overflow at routers)
      - long delays (queueing in router buffers)



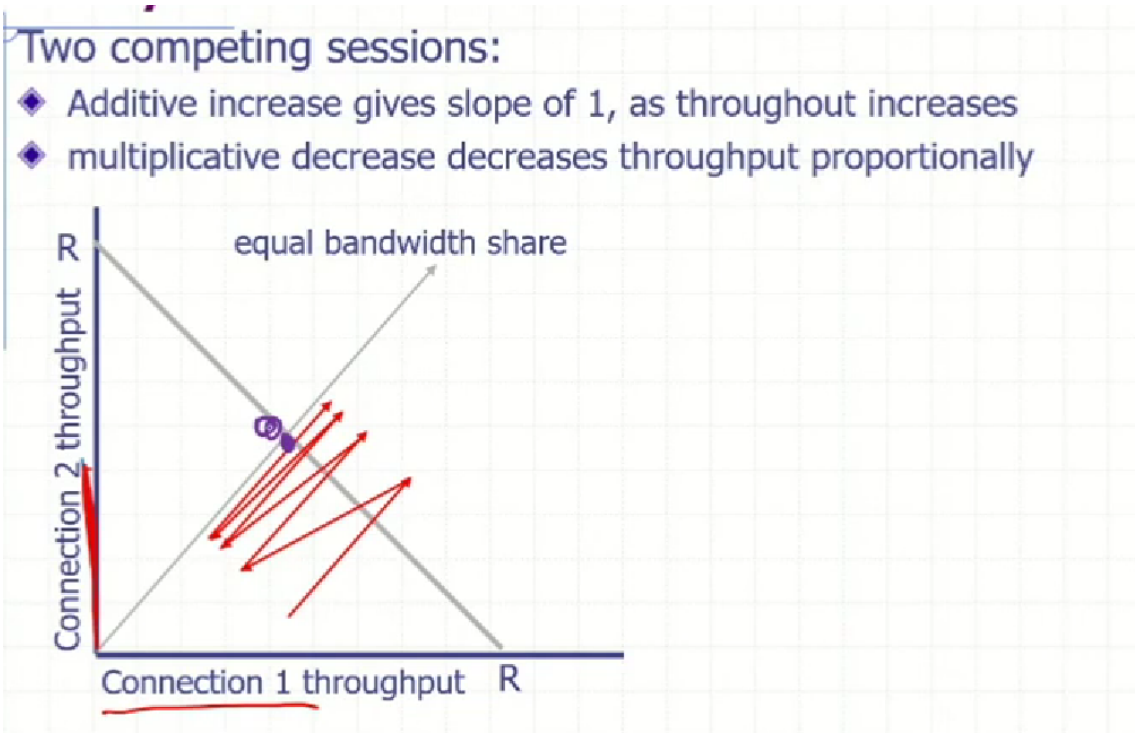
  - solution: end-end congestion control
    - no explicit feedback from network
    - congestion inferred from end-system observed loss, delay

- apporach taken by TCP
- congestion window
    - it is not negotiated, it is assumed
    - it starts with one segment

—> TCP sends min(sliding window(fixed), advertised window(fixed), congestion window)

- when connection begins, increase rate exponentially until first loss even:
-
    - double congestion window every RTT
    - done by incrementing cong win for every ack received
- loss even
    - timeout
        - congwin is set to 1 MSS
        - grows exp to a threshold=congwin/2, then linear
    - 3 duplicate acks
        - is cut in half,
        - then window grows linearly
- when congwin is below threshold, sender is in slow-start phase, window grows exponentially
- when congwin is above threshold, sender is in congestion-avoidance phase, window grows linearly

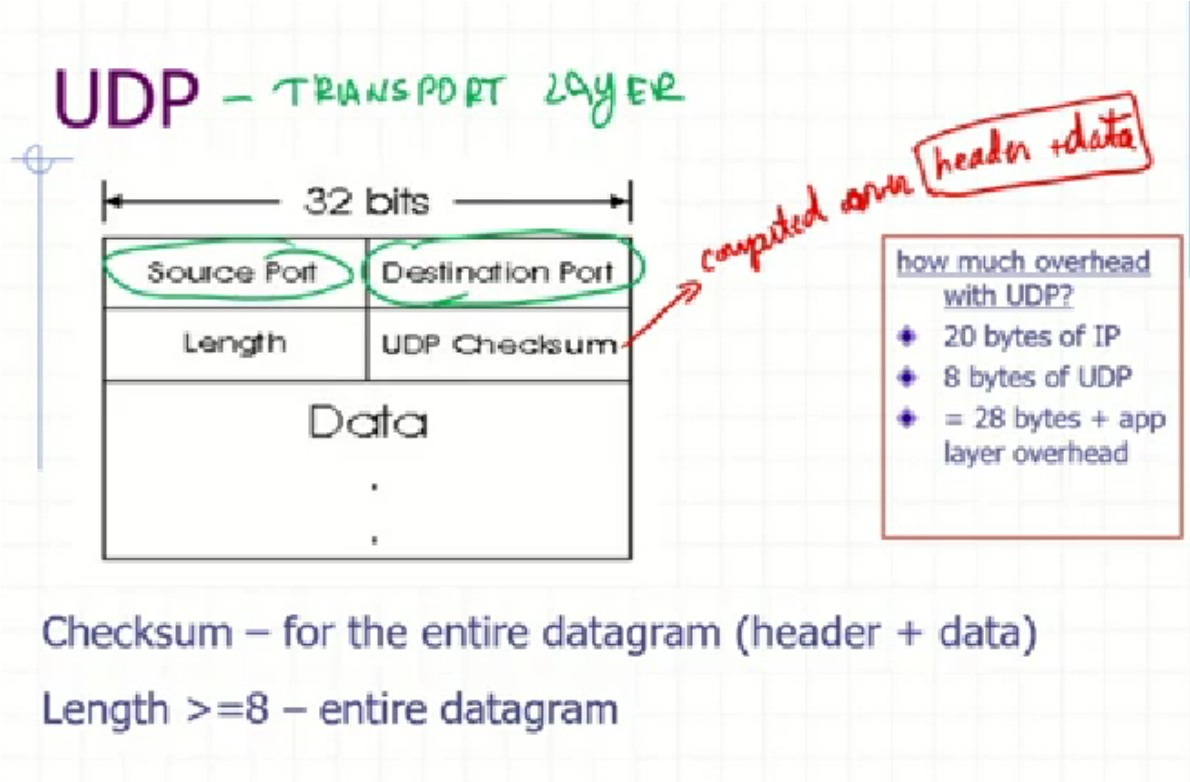
- TCP Fairness
    - goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have averate rate of R/K

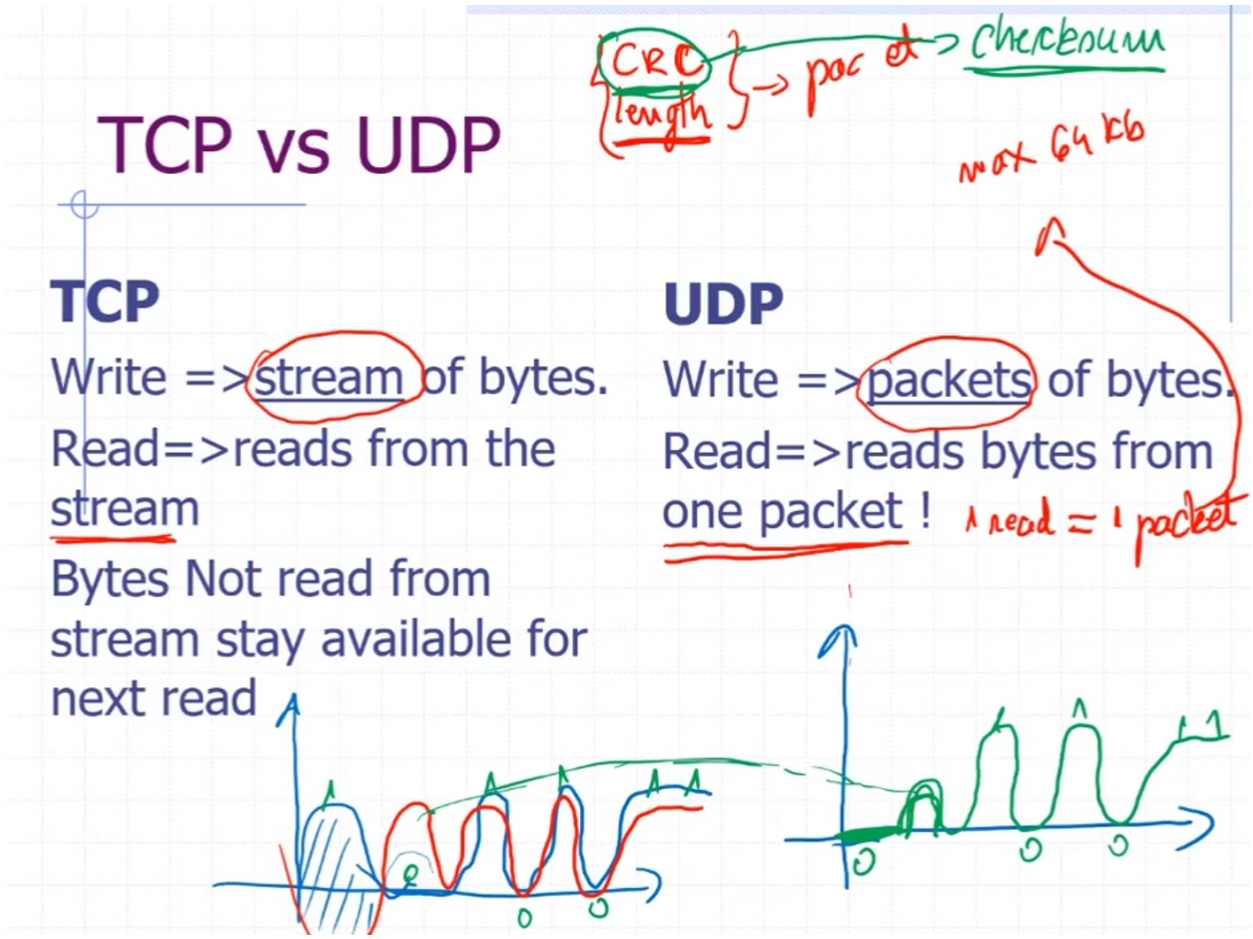


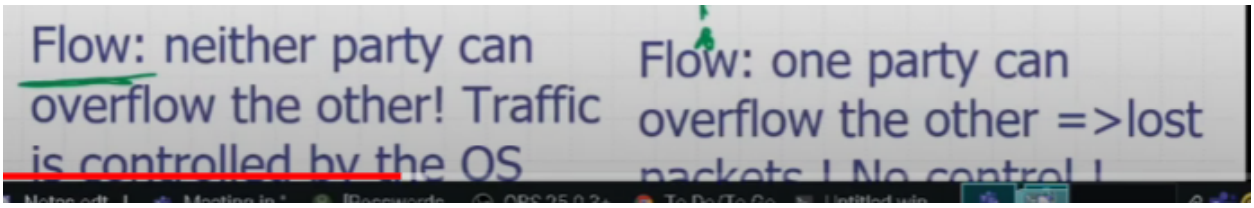
- UDP
  - connection-less - datagram oriented
  - advantages: less latency, higher bandwidth, broadcast
  - does not provide: connection setup, flow control, congestion control, timing, bandwidth guarantee
  - unreliable - when a message is sent, it cannot be known if it will reach its destination; it could get lost along the way. There is no concept of acknowledgment, retransmission, or timeout
  - not ordered - if two messages are sent to the same recipient, the order in which they arrive cannot be predicted
  - lightweight - there is no ordering of messages, no tracking connections, etc. It is a small transport layer designed on top of IP
  - Datagrams - packets are sent individually and are checked for integrity only if they arrive. Packets have definite boundaries which are honored upon receipt,

meaning a read operation at the receiver socket will yield an entire message as it was originally sent

- no congestion control - UDP itself does not avoid congestion, and its possible for high bandwidth applications to trigger congestion collapse, unless they implement congestion control measures at the application level







The Session Layer

- unit of transmission = control messages
- virtual layer
- allows for establishing sessions
- session
  - dialog control
  - token management

- synchronization

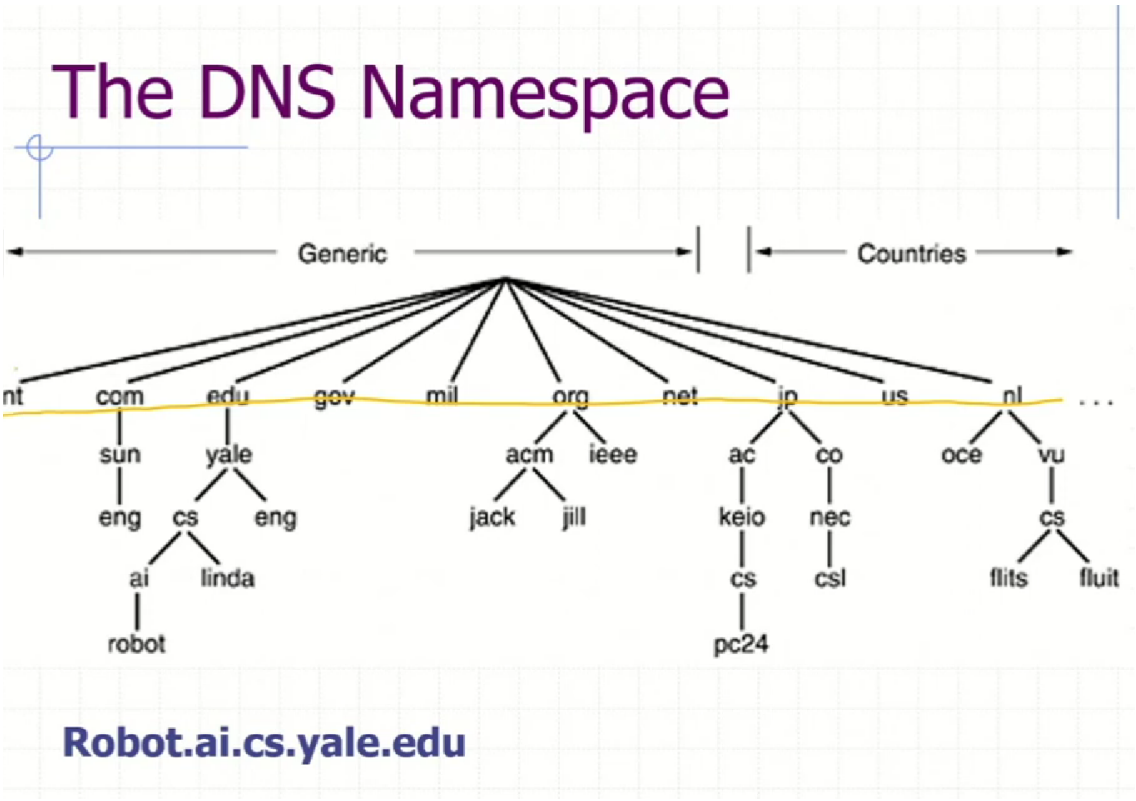
The Presentation Layer

- unit of transmission = record type
- syntax and semantics of data
- abstract data definitions / encoding for information exchange between heterogeneous systems
- standard encoding “on the wire”

The Application layer

- unit of transmission = message ↔ user data
- defines:
  - types of messages: request & response message
  - syntax of message types: what fields in messages & how fields are delineated
  - semantic of fields
  - rules for when and how processes send and respond to messages
- public-domain protocols
  - dns - domain name system, udp port 53=query, tcp port 53=master to slave
    - in the TCP/IP world each machine has its own unique IP address
    - numbers are hard to remember, names are easier → each machine is assigned a name in a tree-like structure
    - domain names (fqdn) or URLs are used by users - www.google.com
    - ip address needed by programs
    - the dns service provides IP name resolution
    - DNS is a distributed database of Domain Names and IP addresses
    - a hierachical naming system used to give each server on the internet a unique name

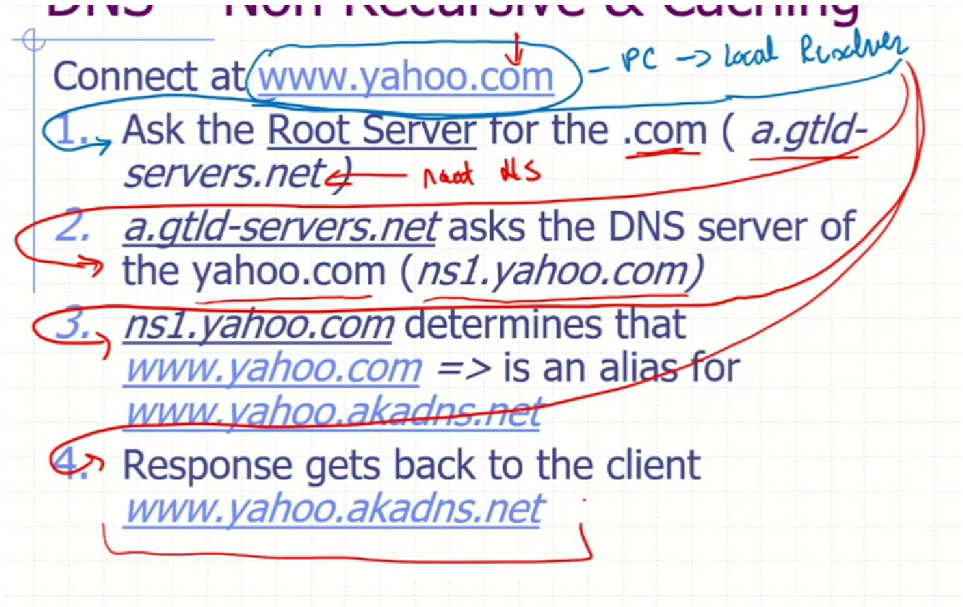
- hostname.domain.tld(top level domain) (host + domain = fullt qualified domain name ↔ FQDN)



- DNS software: resolver + name server (ussualy coupled)
  - resolver
    - build into client tcp/ip software
    - ask designated name server for ip address when client enters fqdn
  - name server
    - dns server (availabel with most os's)
    - retreives ip addresses for clients
    - supplies ip address to other name servers
    - provided by the internet, ISP, or at the client
- dns system

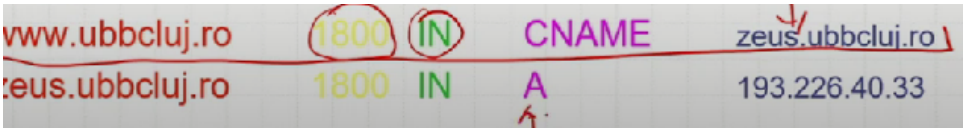
- originally one single central huge table (hosts file) /etc/hosts → nowadays it has only local hosts
- hierachial structure
  - root dns servers (serving .com, .org, .net,…)
  - dns servers - serve domain queries
- dns servers
  - primary/master - authoritative on a zone (ubbcluj.ro), can update db
  - secondary/slaves - temporarily authorative, read only copies of db
  - forwarders/caching dns - no local database, resolver only
- types of queries
  - recursive queries

  - non-recursive (iterative) queries



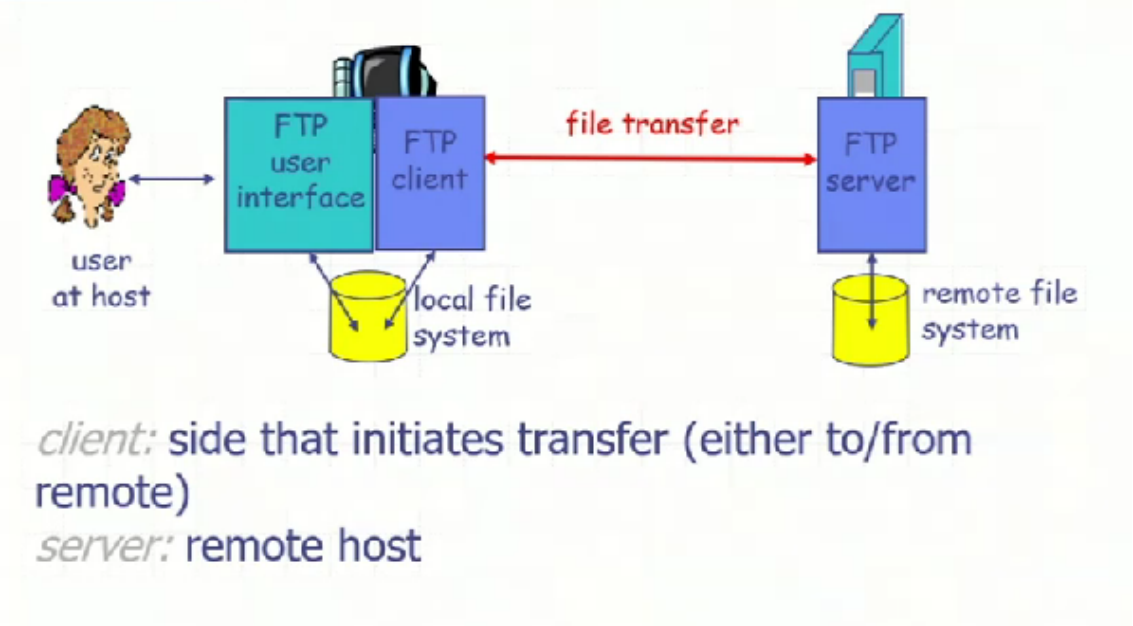
- resource records
  - RR = (domain_name, time_to_live, class, type, value)
  - types

Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

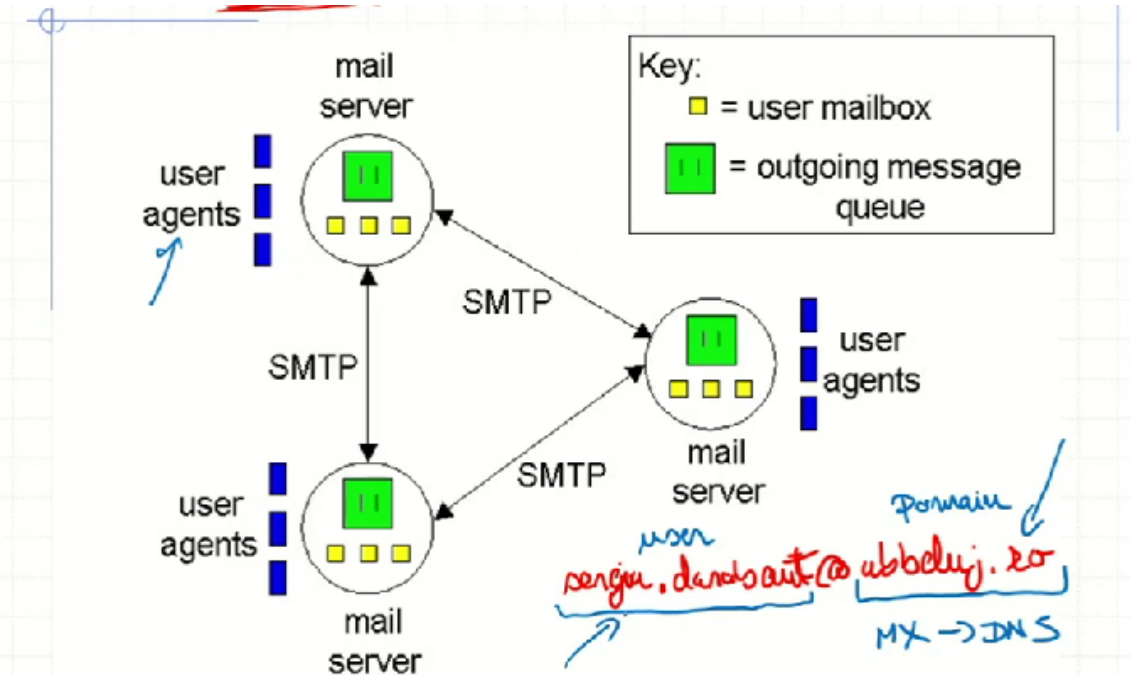


- http/https - www, browsing, http - port 80, https - port 443
  - allows exchange of html and web data
- ftp - tcp port 21
  - allows file exchange between 2 machines

- text protocol
- it is designed to cope with different arhitectures
- its not encrypted

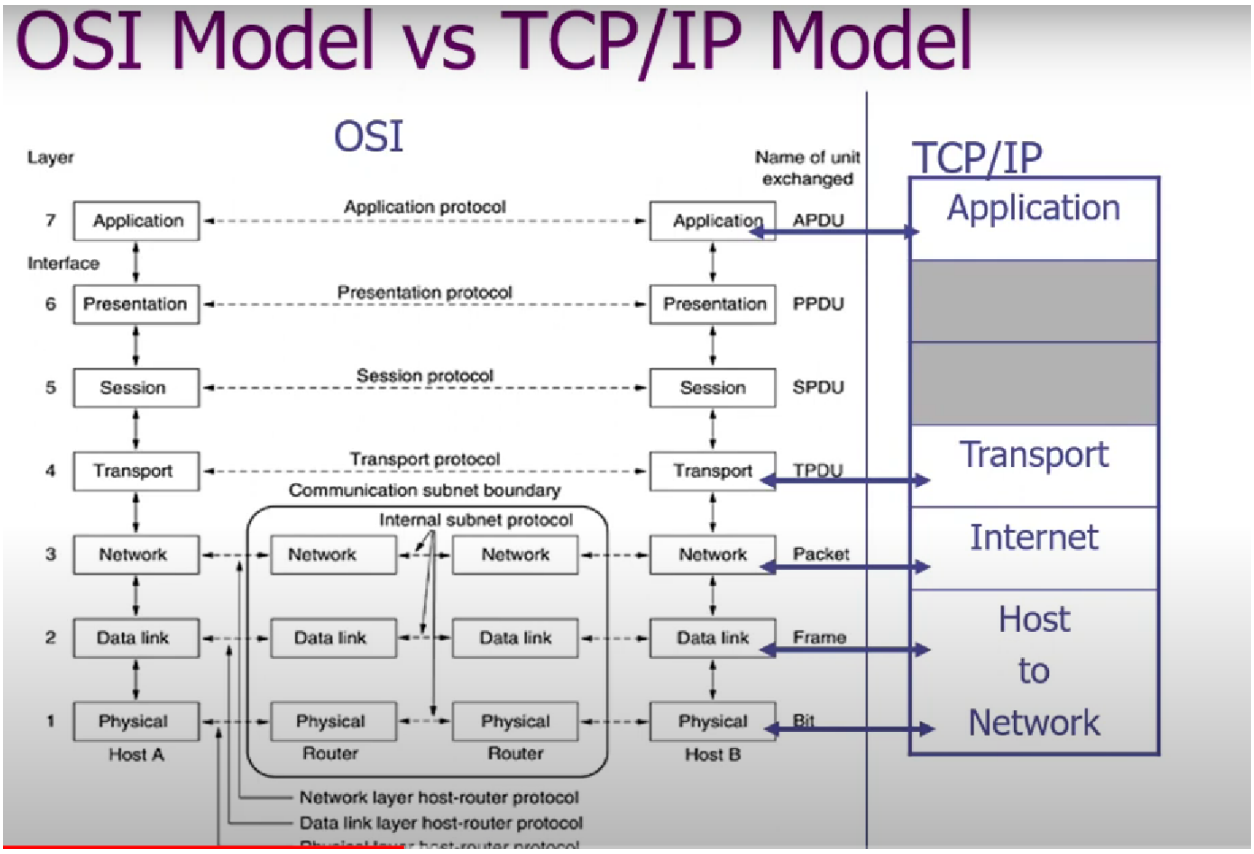


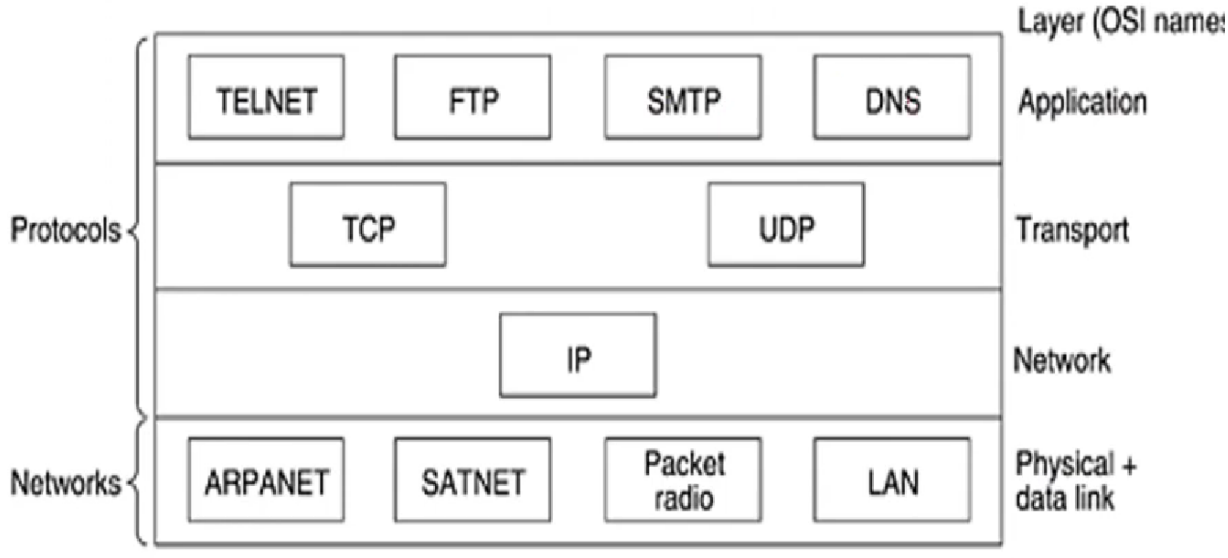
- telnet - remote command
- ssh - remote command
- smtp - mail exchange, linked to dns, tcp port 25
  - text protocol
  - end-to-end encryption (tls,-993, ssl-665)
  - allows for offline message exchanging
  - need to have a mail server and a valid dns



- proprietary protocols
  - kazaA

The TCP/IP Reference Model - The Internet Protocol

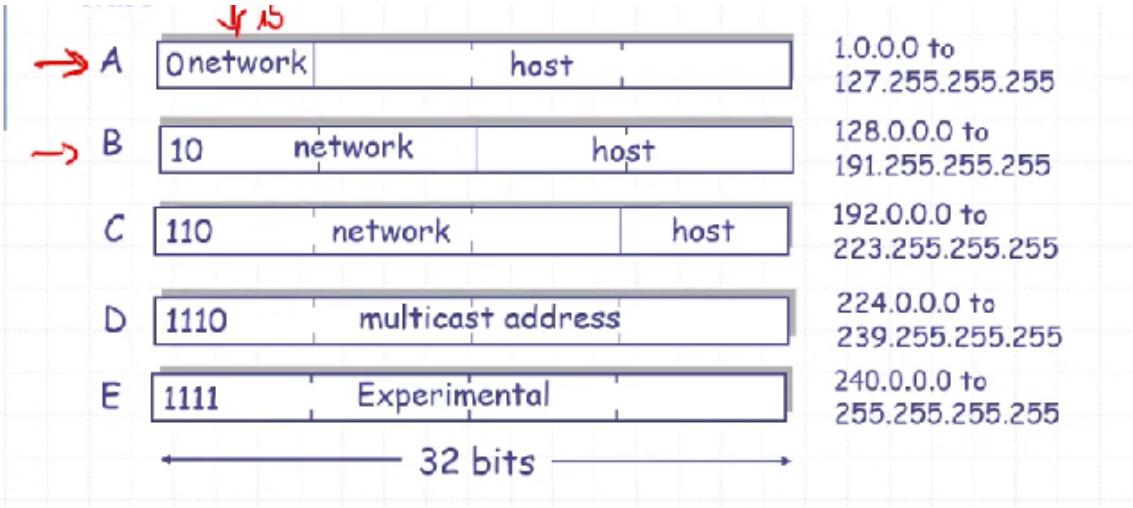


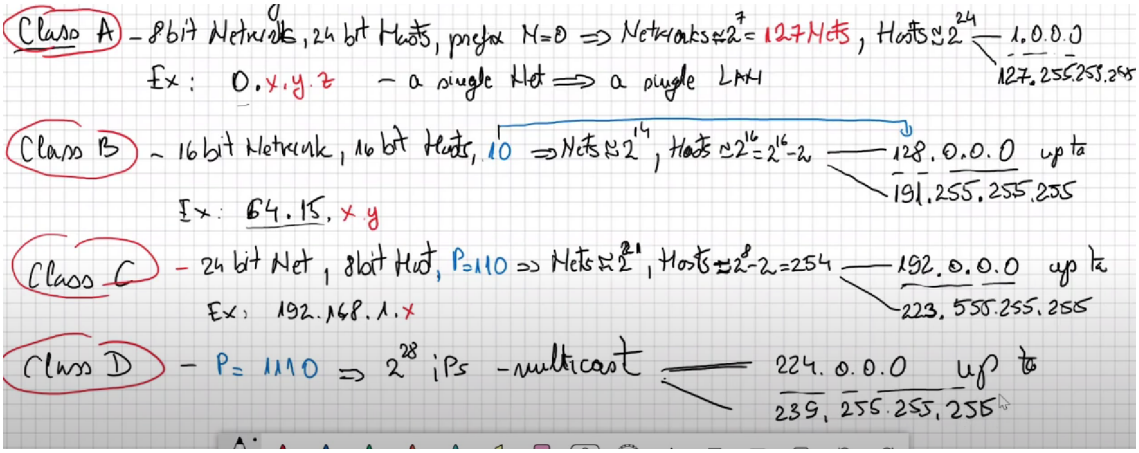


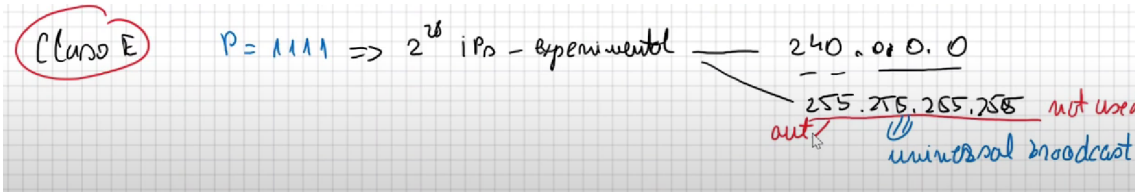
IPv4 addressing

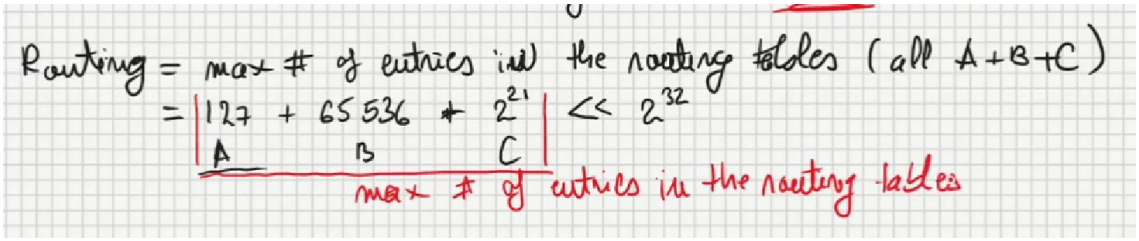
- 32-bit identifier for host, router interface

- 2^32 ip addresses
- interface(network card): connection between host/router and physical link
  - router typically have multiple interfaces
  - host may have multiple interfaces
  - IP adresses associated with each interface
- parts
  - network part → high order bits
  - host part (low order bits)
- a network from the IP adress perspective LAN(local area network)
  - device interfaces with same network part of IP address
  - can physically reach each other without intervening router
- Addressing
  - Class Full





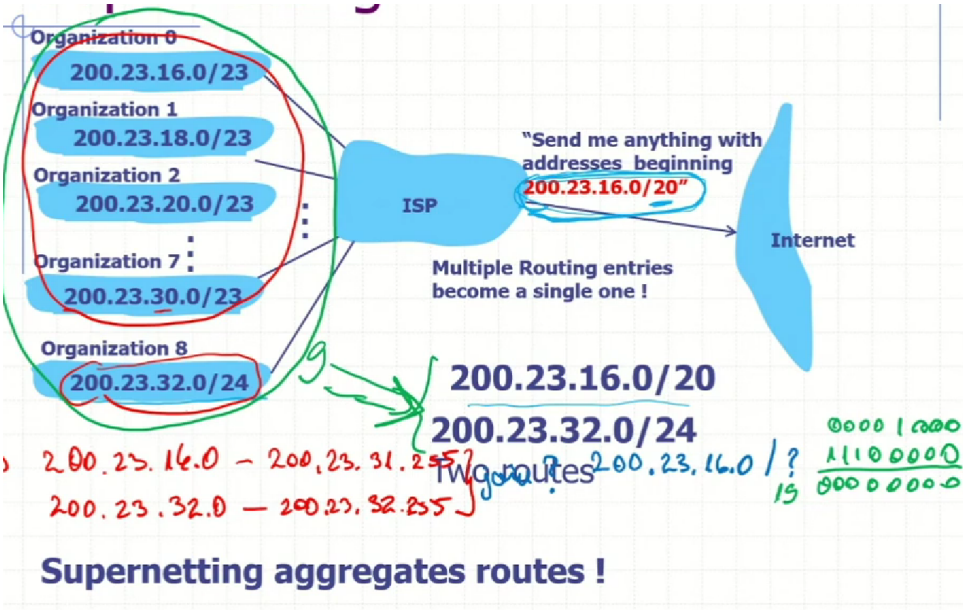




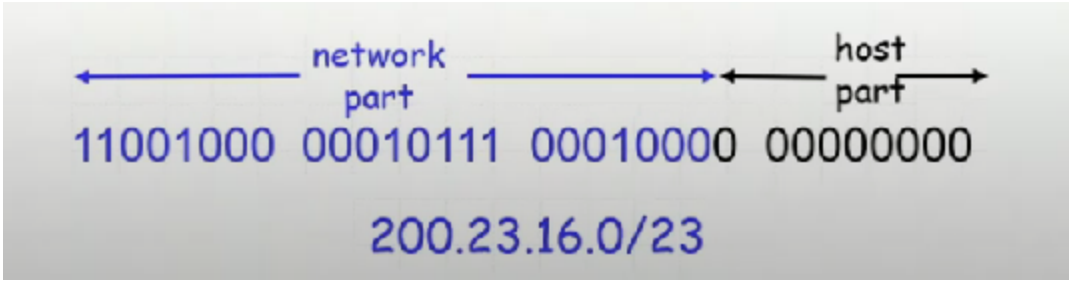
- inefficient use of address space, address space exhaustion
- lots of unused adrdresses

- CIDR: Classless InterDomain Routing
  - IP Subnet
    - introduces third level hierachy + subnet portion
    - allows more efficient and structured utilization of the addresses
    - subnetting
      - the same network can be configured with different masks
      - can have subnets of different sizes
    - supernetting

- how does a network get the network part of IP adrr → it gets allocated from the portion of its provider ISP's address space

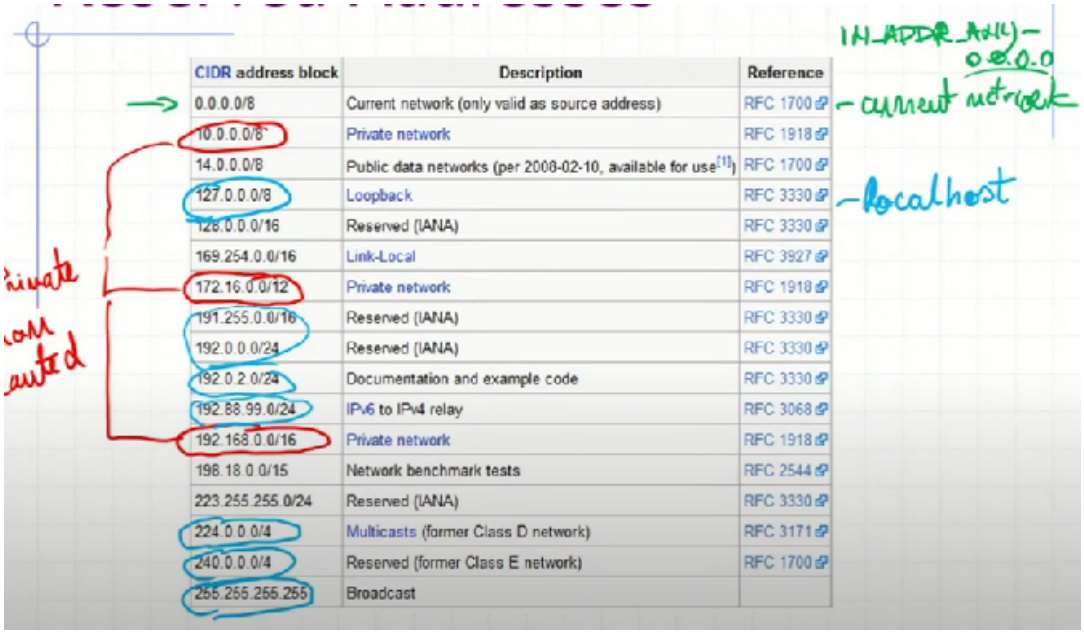


- network portion of the address of arbitary length
- network format a.b.c.d/x
  - a.b.c.d prefix
  - x - number of lefmost contiguous bits to be used for the network mask



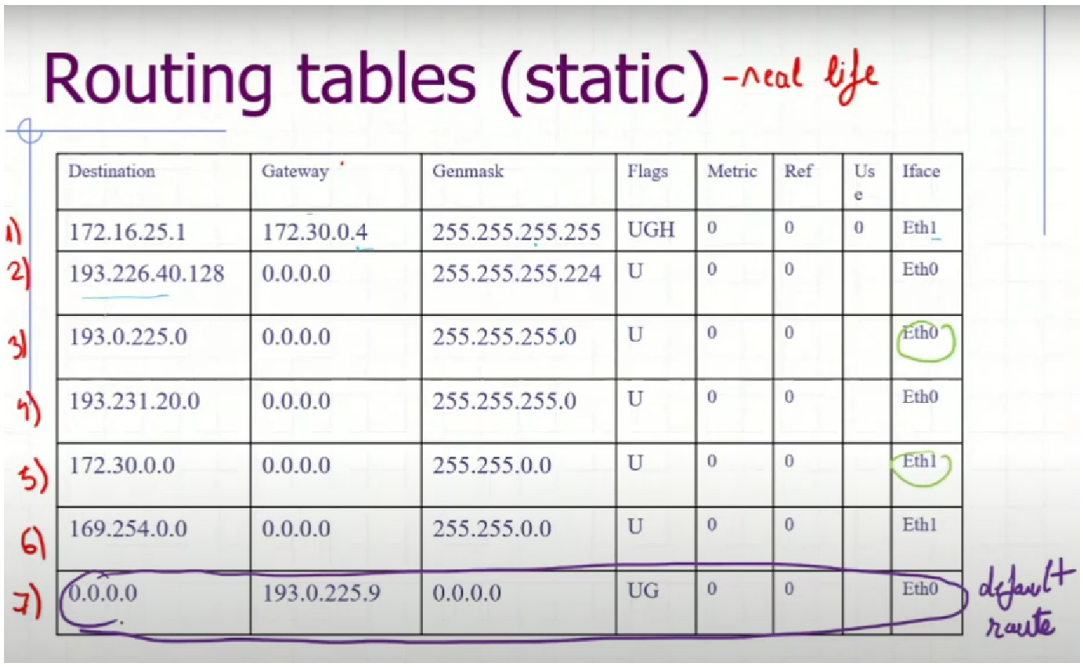
- rules
  - the number of addresses in each block must be a power of 2
  - the beginning address in each block must be divisible by the number o addresses in the block

        - a block that contains 16 addresses cannot have beginning address as 193.226.49.36, but 193.226.40.64 is possible
        - to check to and between network address and mask → network address back
    - the first ip = network address, the last ip = broadcast address
    - /31 possible but useless
- natural masks
    - class A,B,C have fixed division of network/host → can be expressed as masks
    - Class A: 255.0.0.0
    - Class B: 255.255.0.0
    - Class C: 255.255.255.0
- reserved addresses



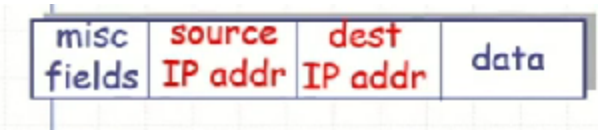
CIDR address block	Description	Reference
0.0.0.0/8	Current network (only valid as source address)	RFC 1700
10.0.0.0/8	Private network	RFC 1918
14.0.0.0/8	Public data networks (per 2008-02-10, available for use[1])	RFC 1700
127.0.0.0/8	Loopback	RFC 3330
128.0.0.0/16	Reserved (IANA)	RFC 3330
169.254.0.0/16	Link-Local	RFC 3927
172.16.0.0/12	Private network	RFC 1918
191.255.0.0/16	Reserved (IANA)	RFC 3330
192.0.0.0/24	Reserved (IANA)	RFC 3330
192.0.2.0/24	Documentation and example code	RFC 3330
192.88.99.0/24	IPv6 to IPv4 relay	RFC 3068
192.168.0.0/16	Private network	RFC 1918
198.18.0.0/15	Network benchmark tests	RFC 2544
223.255.255.0/24	Reserved (IANA)	RFC 3330
224.0.0.0/4	Multicasts (former Class D network)	RFC 3171
240.0.0.0/4	Reserved (former Class E network)	RFC 1700
255.255.255.255	Broadcast	

    - private address → not reachable
- routing tables (static)

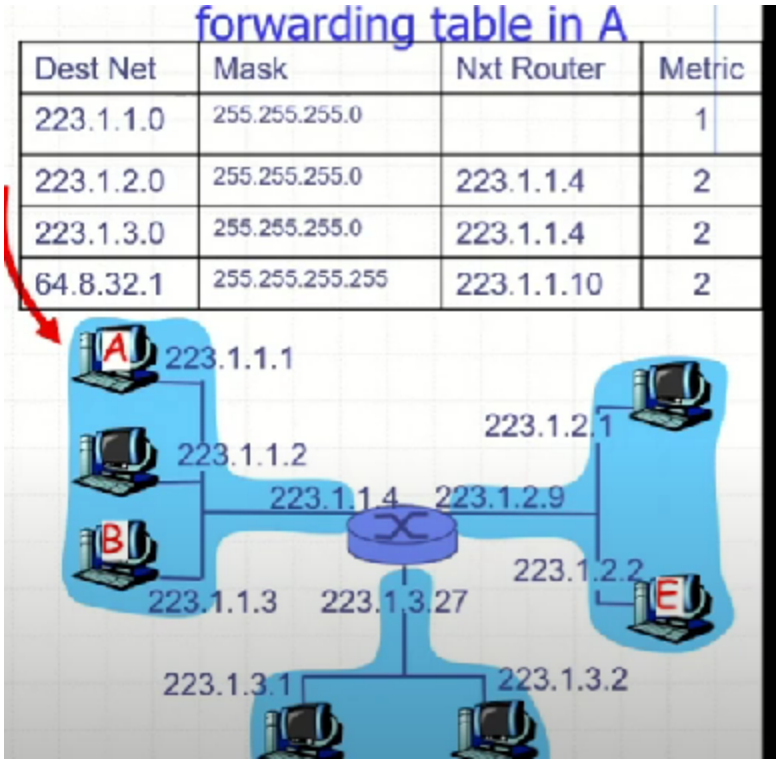


Moving a datagram from source to destination

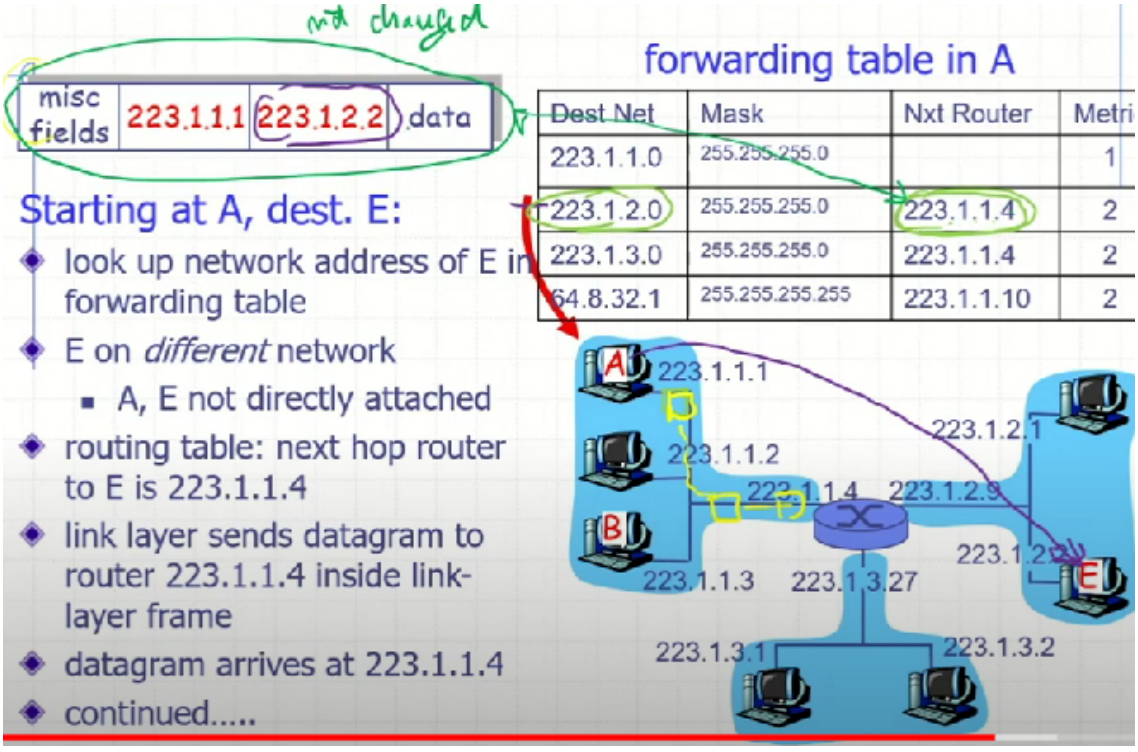
- ip datagram

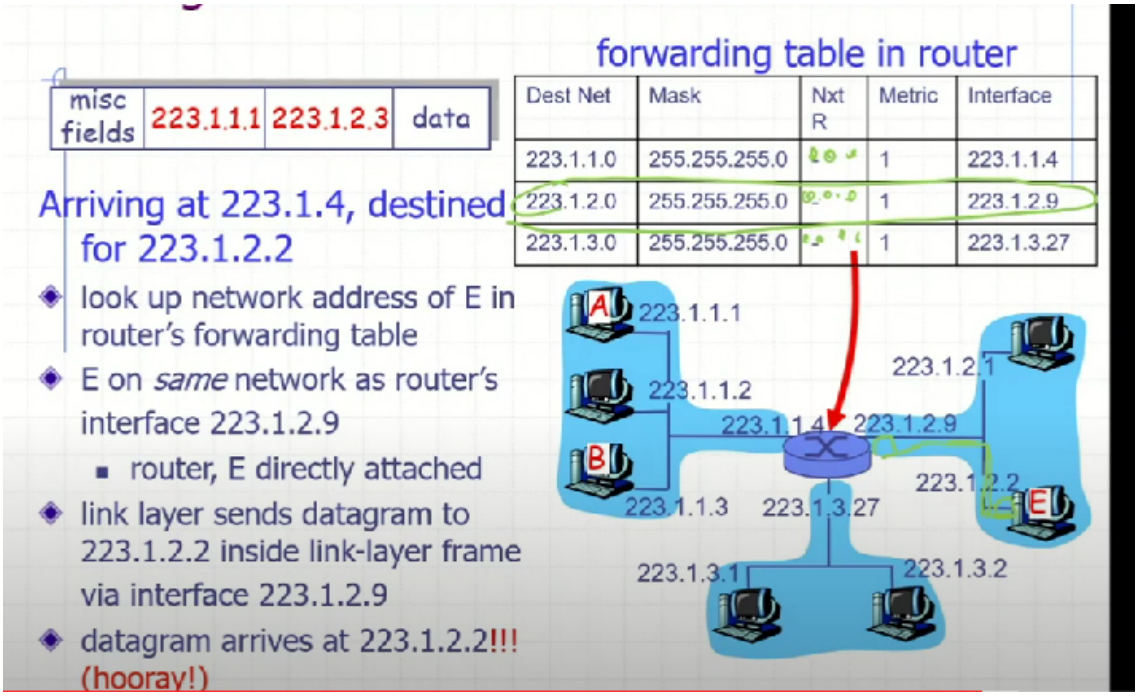


- datagram remain unchanged, as it travels source to destination
- Examples
  - A → B

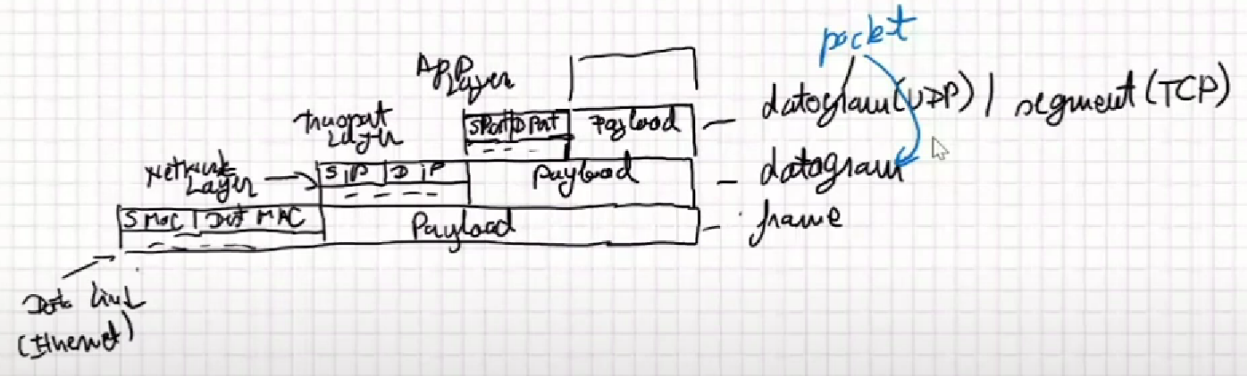

forwarding table in A
Dest Net | Mask | Nxt Router | Metric
223.1.1.0 | 255.255.255.0 | | 1
223.1.2.0 | 255.255.255.0 | 223.1.1.4 | 2
223.1.3.0 | 255.255.255.0 | 223.1.1.4 | 2
64.8.32.1 | 255.255.255.255 | 223.1.1.10 | 2
A
B
E
223.1.1.1
223.1.1.2
223.1.1.4
223.1.2.9
223.1.2.1
223.1.2.2
223.1.1.3
223.1.3.27
223.1.3.1
223.1.3.2

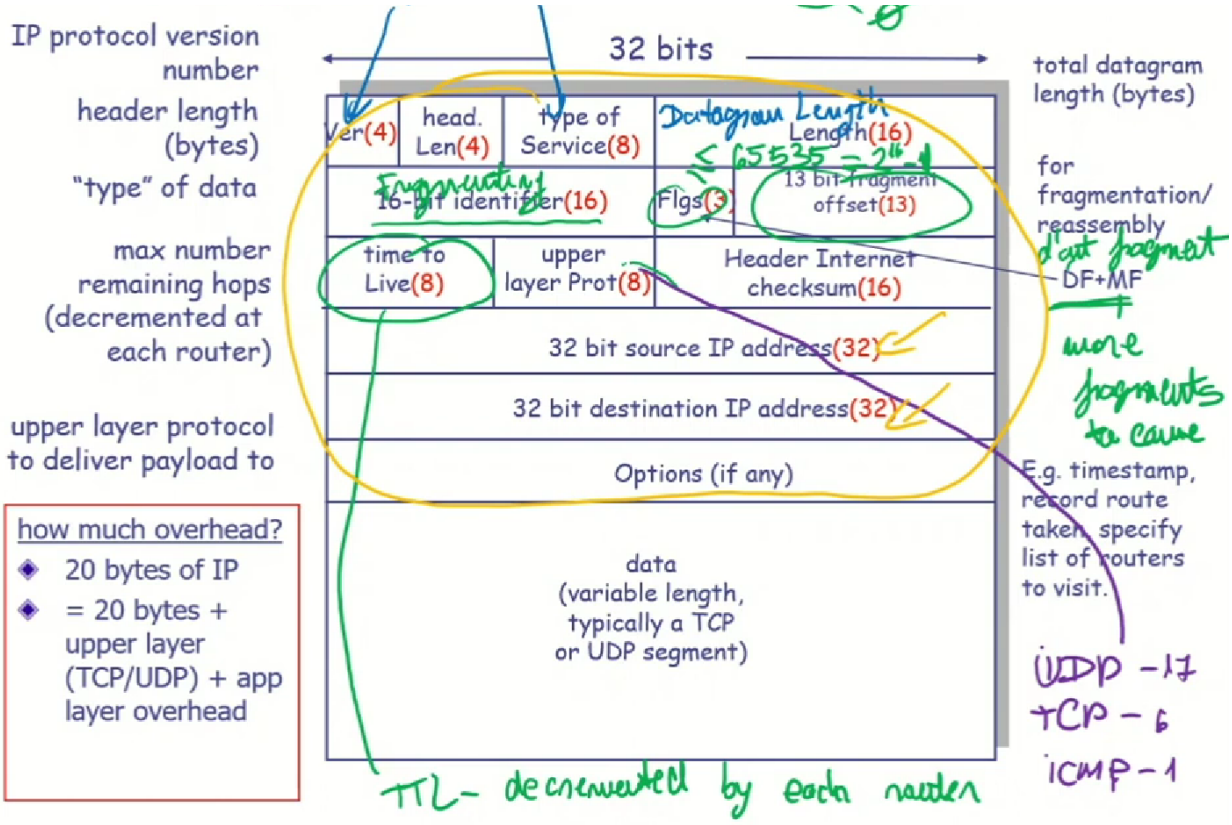
- A → E


not changed
misc fields
223.1.1.1
223.1.2.2
data
forwarding table in A
Dest Net | Mask | Nxt Router | Metri
223.1.1.0 | 255.255.255.0 | | 1
223.1.2.0 | 255.255.255.0 | 223.1.1.4 | 2
223.1.3.0 | 255.255.255.0 | 223.1.1.4 | 2
64.8.32.1 | 255.255.255.255 | 223.1.1.10 | 2
Starting at A, dest. E:
look up network address of E in forwarding table
E on different network
A, E not directly attached
routing table: next hop router to E is 223.1.1.4
link layer sends datagram to router 223.1.1.4 inside link-layer frame
datagram arrives at 223.1.1.4
continued.....
A
B
E
223.1.1.1
223.1.1.2
223.1.1.4
223.1.2.9
223.1.2.1
223.1.1.3
223.1.3.27
223.1.3.1
223.1.3.2

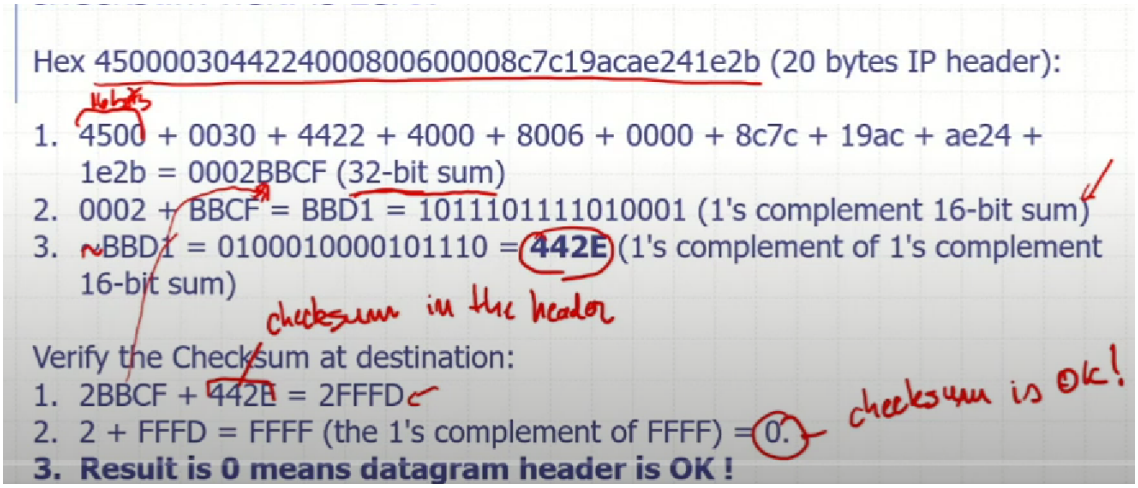


Datagram Format



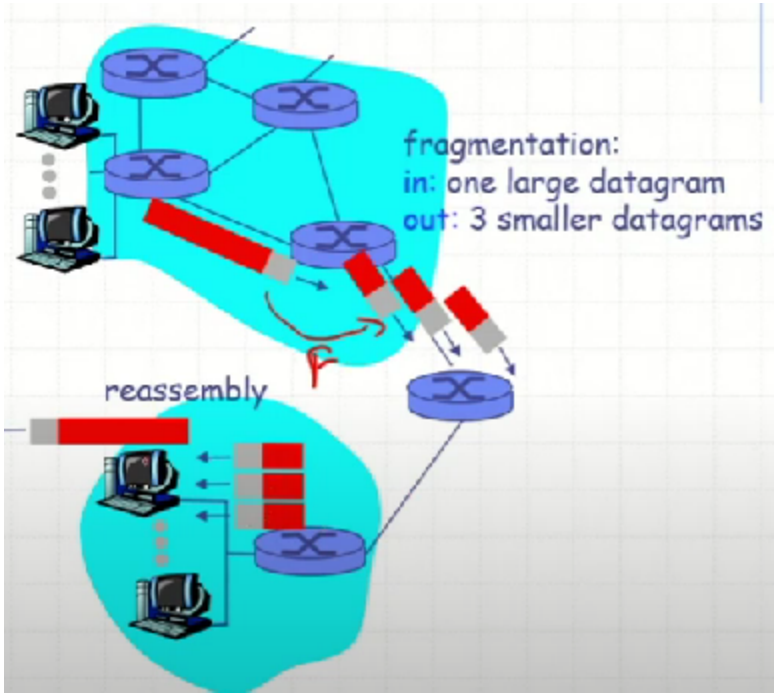


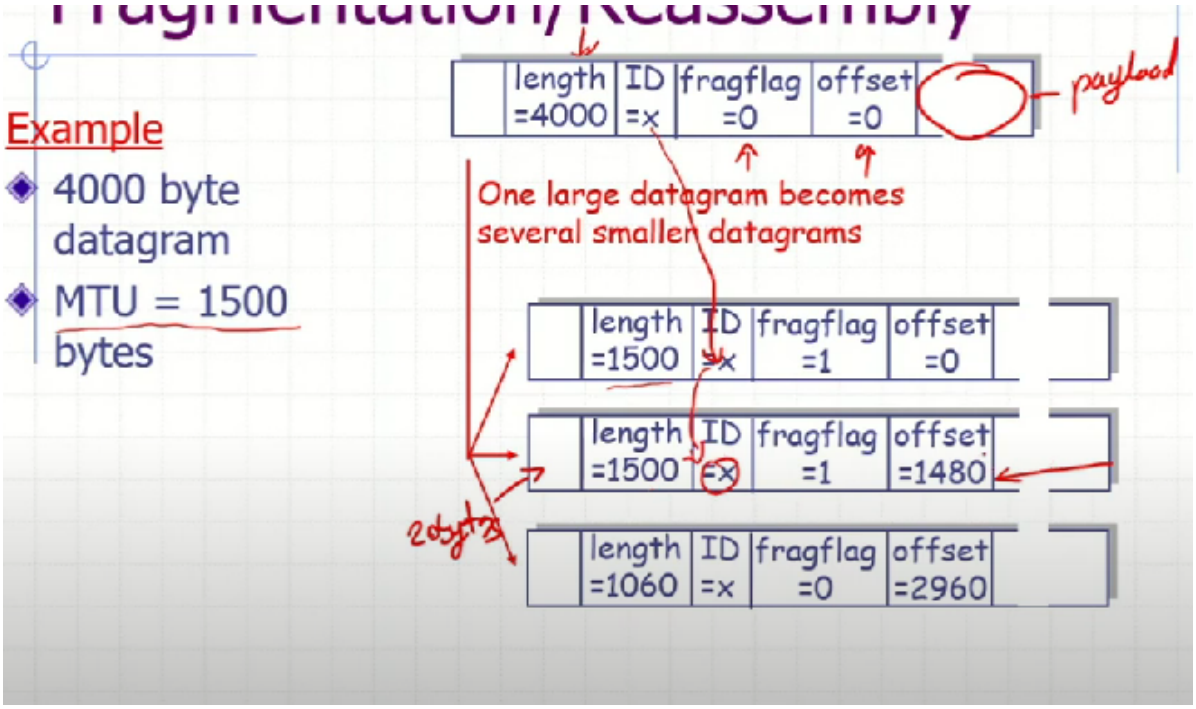
- IP checksum
  - the checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero



IP fragmentation/ reassembly

- only IPv4 has fragmentation implemented
- network links have MTU (max transfer size) - largest possible link-level frame → different link type → different MTU
- large IP datagram is divided within the net
  - one datagram becomes several datagrams
  - reassembled only at final destination
  - IP Header bits used to identify, order related fragments

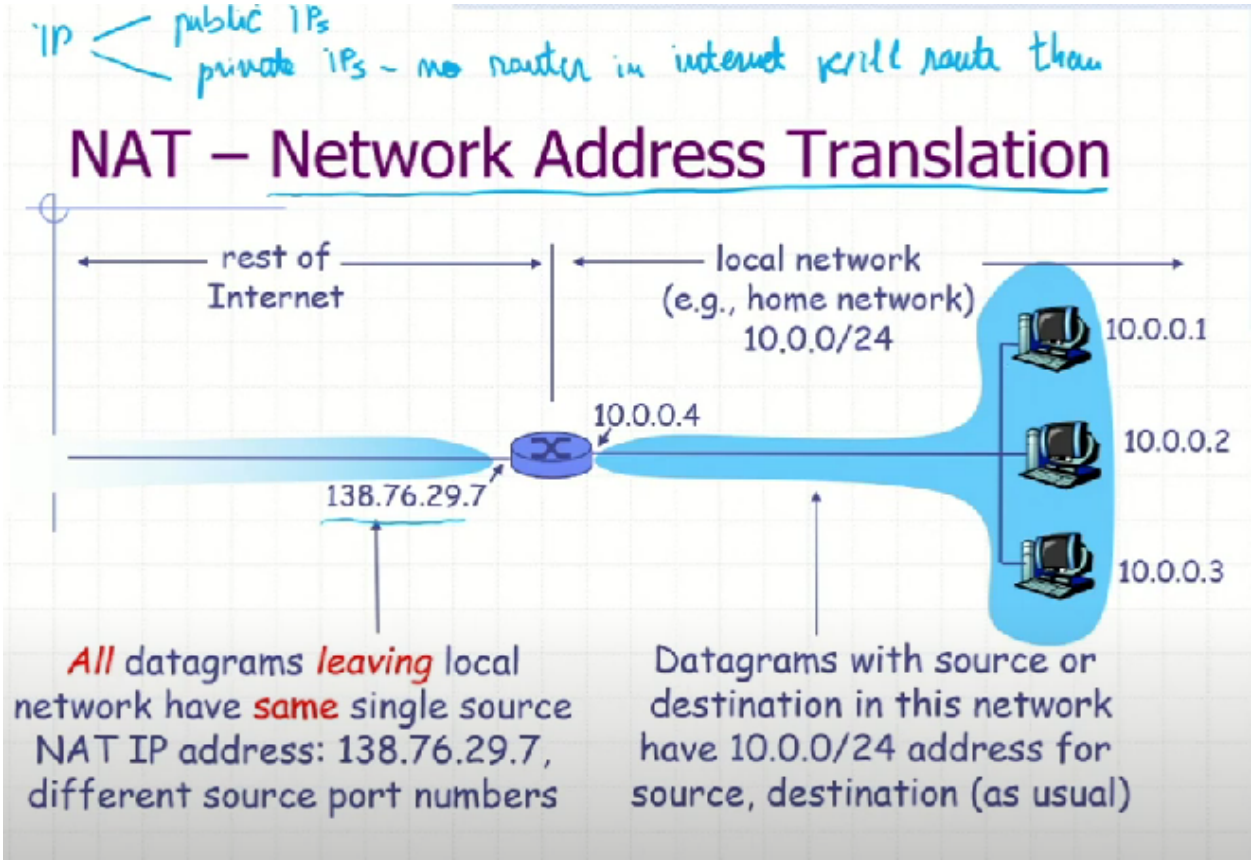




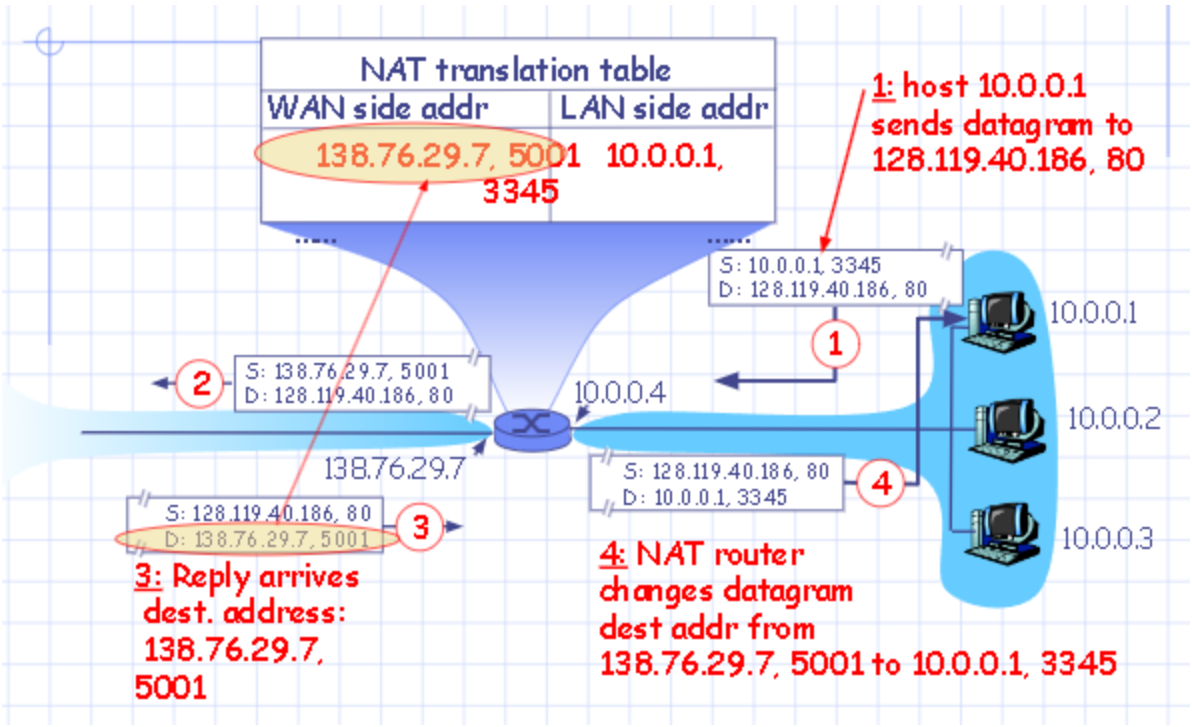
ARP: IP address vs Network Adapter address

- IP datagram: comm IP → IP → network layer
- Ehternet: MAC → MAC → Data link layer
- ARP: Address resolution protocol
- translates between IP to MAC, works at the physical layer

NAT - Network Address Translation



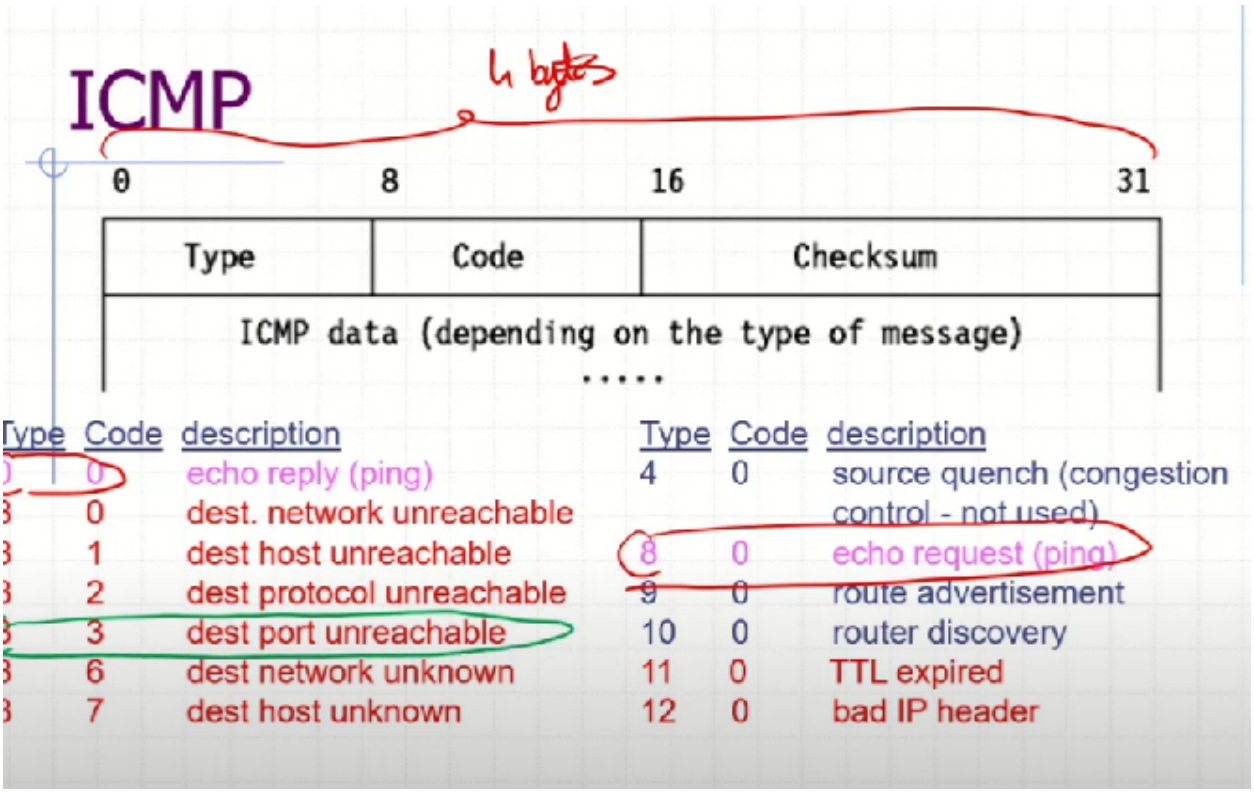
- local network uses just one IP address as far as the outside world is concerned
- no need to be allocated range of addresses from ISP: just one IP address is used for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)



- 16-bit port-nmber field
  - roughlt 60,000 for each TCP and UDP simultaneous connection with a single LAN-side address

ICMP Protocol

- signal things
- used by hosts, routers, gateways to communcation network-level information
  - error reporting: unreachable host, network, port, protocol
  - echo request/reply (used by ping)
- network-layer above IP:
  - ICMP msgs carried in IP datagrams
- ICMP message: type, code plus first 8 bytes of IP datagram causing error



Routing

- goal: determine “good” path (sequence of routers) thru network from source to dest
- graph abstraction for routing alg
  - graph nodes are routers
  - graph edgs are physical links
    - link cost: delay, $ cost or congstion level
- good path
  - minimum cost path
  - other def’s possible
- Classificaton 1
  - Static
    - routes change slowly over time
  - Dynamic

        - routes change more quickly
        - periodic update in response to link cost changes
- Classification 2
    - Global
        - all routers have complete topology, link cost info
        - link state alg → djisktra
    - Decentralized
        - router knows only physically connected neighbours, link cost to neighbours
        - iterative process of computation, exchange info with neighbours
            - distance vectors alg → RIP (routing information protocol)
                - problem: couting to infinity when link goes down → to limit infinity = 16
                - RIP v1 - classfull addressing only
                - RIP v2 - CIDR + poison reverse(solution to counting infinty, take out rows in routing table where next hop is the routing they send the table)

Address processes

- for a process to receive messages, it must have an identifier

IP address and ports

- every host has a unique 32-bit IP address
- 0.0.0.0 - local machine
- 255.255.255.255 -universal broadcast adresss
- assigned locally usually
- port number
    - used for identifying processes running on the host

- 2 bytes
- max 65535 max sockets for TCP and UDP each
- example: http server: 80, mail server: 25, 443: https

MAC address

- physical address, bound to the factor
- 48 bit size
- the first 24 are registered for the manufactures
- last 24 are serial number

Network programming

Socket programming with TCP

- socket: a door between application process and end-end-transport protocl (UDP or TCP)
- BSD socket library

```
#include <sys.socket.h>

//Client + Server :
int socket (int family, int type, int protocol);
//family = AF_INET
//type = SOCK_STREAM - TCP, SOCK_DGRAM - UDP

struct sockaddr {
   uint8_t sa_len;
   sa_family_t sa_family; /* address family: AF_xxx value */
   char sa_data[14];
};
sa_data is a: (like when using unions)
struct sockaddr_in {
   uint8_t sin_len; /* length of structure (16)*/
```

```
   sa_family_t sin_family; /* AF_INET*/
   in_port_t sin_port; /* 16 bit TCP or UDP port number */
   struct in_addr sin_addr; /* 32 bit IPv4 address*/
   char sin_zero[8]; /* not used but always set to zero - rer
};

struct in_addr{
in_addr_t s_addr; /*32 bit IPv4 network byte ordered address
};

//Client:
int connect (int sfd, const struct sockaddr *servaddr, sockle

//Server:
int bind(int sfd, const struct sockaddr *servaddr, socklen_t
int listen(int sockfd, int backlog);
int accept(int sockfd, struct sockaddr *cliaddr, socklen_t *a

//Client + Server:
#include <unistd.h>
int close(int sockfd);

int shutdown(sock, direction); //closes either sending or rec

//Once a connection has been setup
#include <sys/socket.h>


ssize_t send(int sockfd, const void *buf, size_t nbytes, int
int write(int sockfd, …..) – works as well

ssize_t recv(int sockfd, void *buf, size_t nbytes, int flags)
int read(int sockfd,……) – works as well

struct hostent *gethostbyname(const char *name);
struct hostent *gethostbyaddr(const void *addr, int len, int
```

```
//hostent structure is defined in <netdb.h> as follows:
struct hostent {
    char  *h_name;           /* official name of host */
    char **h_aliases;        /* alias list */
    int    h_addrtype;       /* host address type */
    int    h_length;         /* length of address */
    char **h_addr_list;      /* list of addresses */
}
#define h_addr h_addr_list[0]
```

    - info on man linux, msdn windows
- connection-oriented
    - send - return N bytes that have successfully delivered to OS and maybe to destination. Error will be signaled on next operation
    - recv - return N bytes that have been received
    - data is like a stream, there is no data loss
    - guaranteed data delivery, data ordering delivery
- type = sock_stream - when creating socket
- client-server TCP/IP apps
    - server listens for client’s requests, executes then and answers
    - server types
        - iterative servers (blocking - paradigm)
            - listen queue + request queue
            - one client at a time
        - concurrent servers (blocking) (processes, threads) or m
            - multiple clients at the same time
        - conccurent multiplexed servers(select)
            - multiple clients at the same time

- Steps of basic apps
  - client must contact server
    - server process must be running
    - server must already have a created socket (door) - rendevous socket, always listening that welcomes client’s contact
  - client contacts server by
    - creating client-local TCP socket
    - specifying IP adress, port
  - when client creates socket: client TCP establishes connection to server TCP
  - when contacted by cliend, server TCP creates a new socket for server process to communicate with that particular client
    - allows server to talk with multiple clients
    - source port numbers used to distinguish clients

Socket Programming with UDP

- connection-less - datagram oriented
- type = SOCK_DGRAM when creating socket
- 64kb - 8 UDP - 20 IP - nr byte for ethernet frame ↔ max IP datagram size
- UDP API
  - socket(..,SOCK_DGRAM,..)
  - bind - similar to TCP
  - data exchange - sendto, recvfrom
  - close - same meaning as TCP
  - no listen and accespt are required/permitted

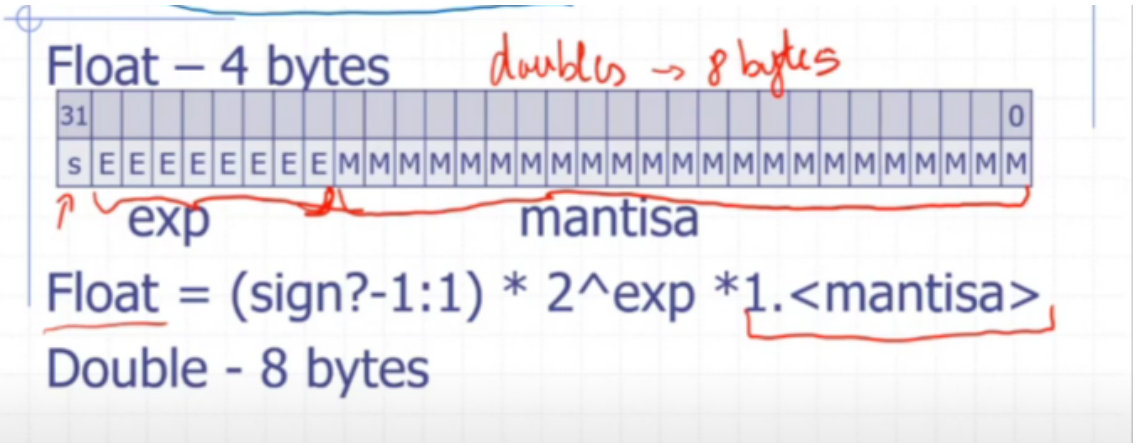
```
ssize_t sendto(int sockfd, const void *buf, size_t len, int f
/*Return:
```

```
no of bytes transmitted to the OS/network if positive
-1 in case of locally detected error (errno set appropriately
No error signaling for undelivered data !
Data size must fit into a transmission unit(datagram) < 64Kb
One sendto operation must be consumed by exactly one recvfrom
*/

size_t recvfrom(int sockfd, void *buf, size_t len, int flags,
                struct sockaddr *src_addr, socklen_t *addrle
/*Return:
no of bytes received from the OS/network and stored into buf
-1 in case of error (errno set appropriately)
Call blocks until a datagram arrives
If not all data is read in ONE call – the remaining is discar
Cannot receive more than the maximum transfer unit (datagram)
*/

setsockopt(int s, int level, int optname, void *optval,
getsockopt(…..)
//man 7 socket/ip/udp/tcp
/*Optname
    SO_REUSEADDR – reuse local addresses – use it to get rid
    SO_BROADCAST – enables broadcast = one sender – Universal
*/
```

Little Endian / Big Endian

- in memory data represenation
- big endian - most significant byte first
- little endian - least significant byte first
- network rep = Big endian
- conversions
  - no need for string

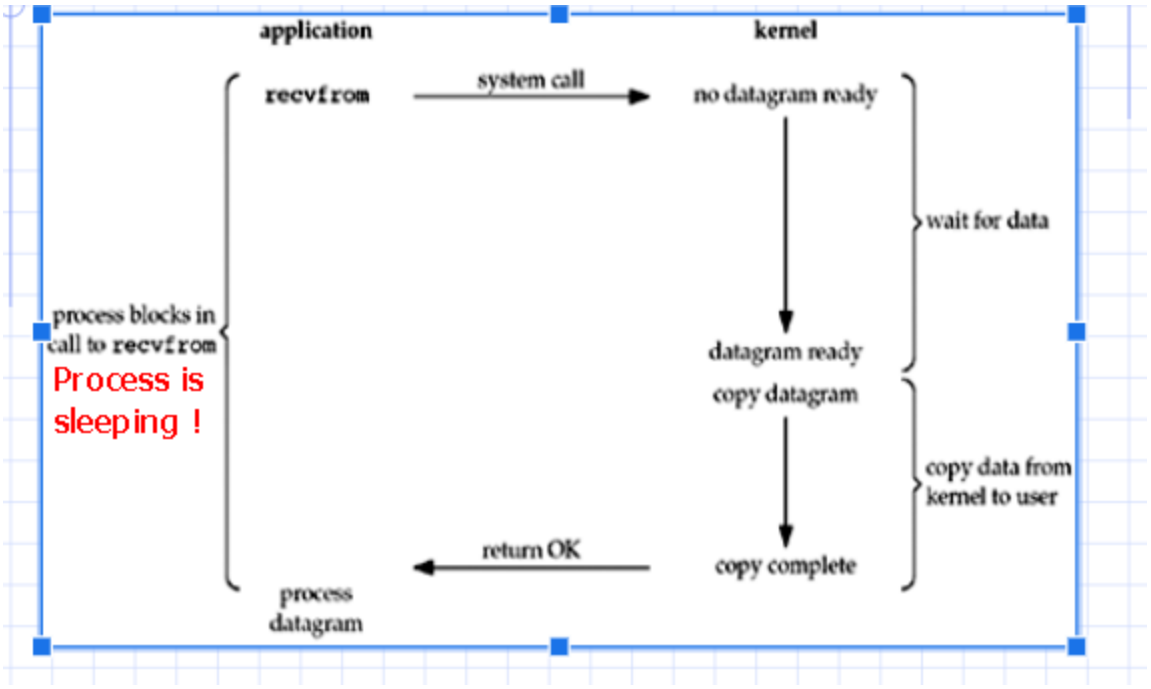
- int/long
  - c/c++: htons, htonl, ntohs, ntohl
  - python: pack/unpack with ! - network
- float and double



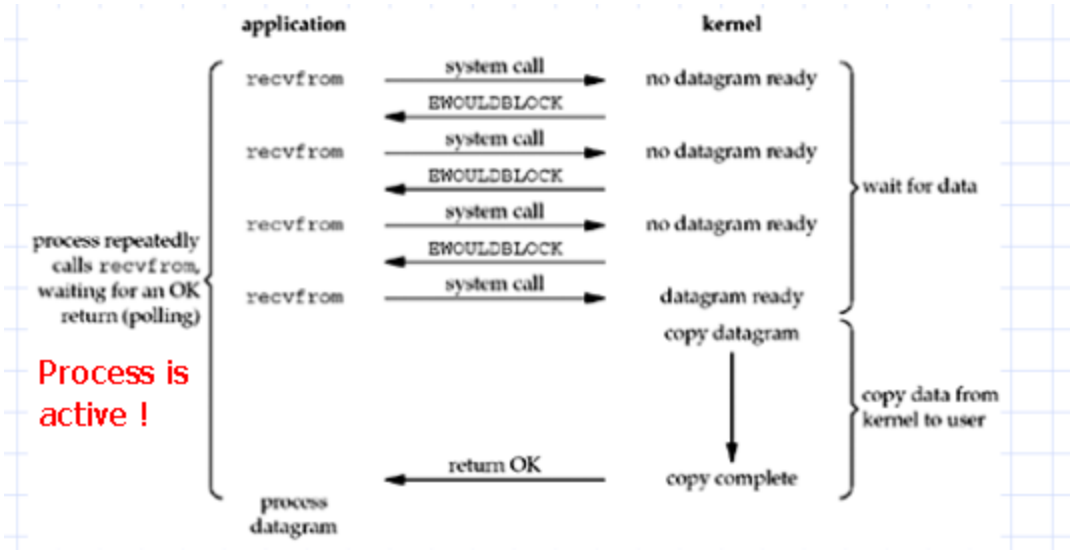
  - for float just convert to int and convert
  - for long convert to char and flip the bytes

I/O Modes

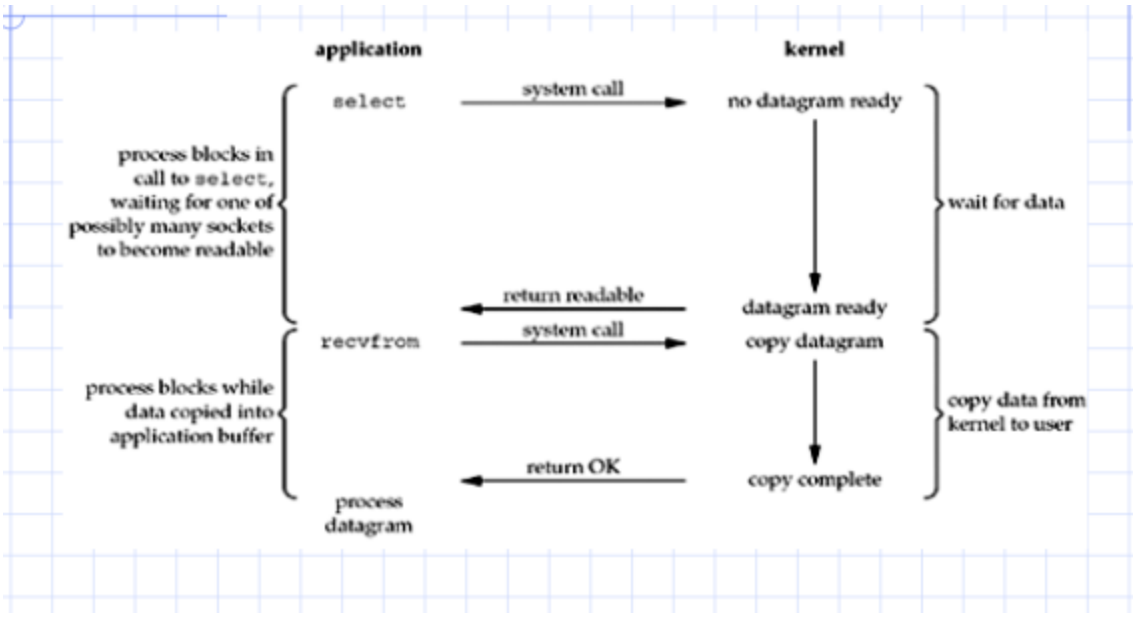
- blocking

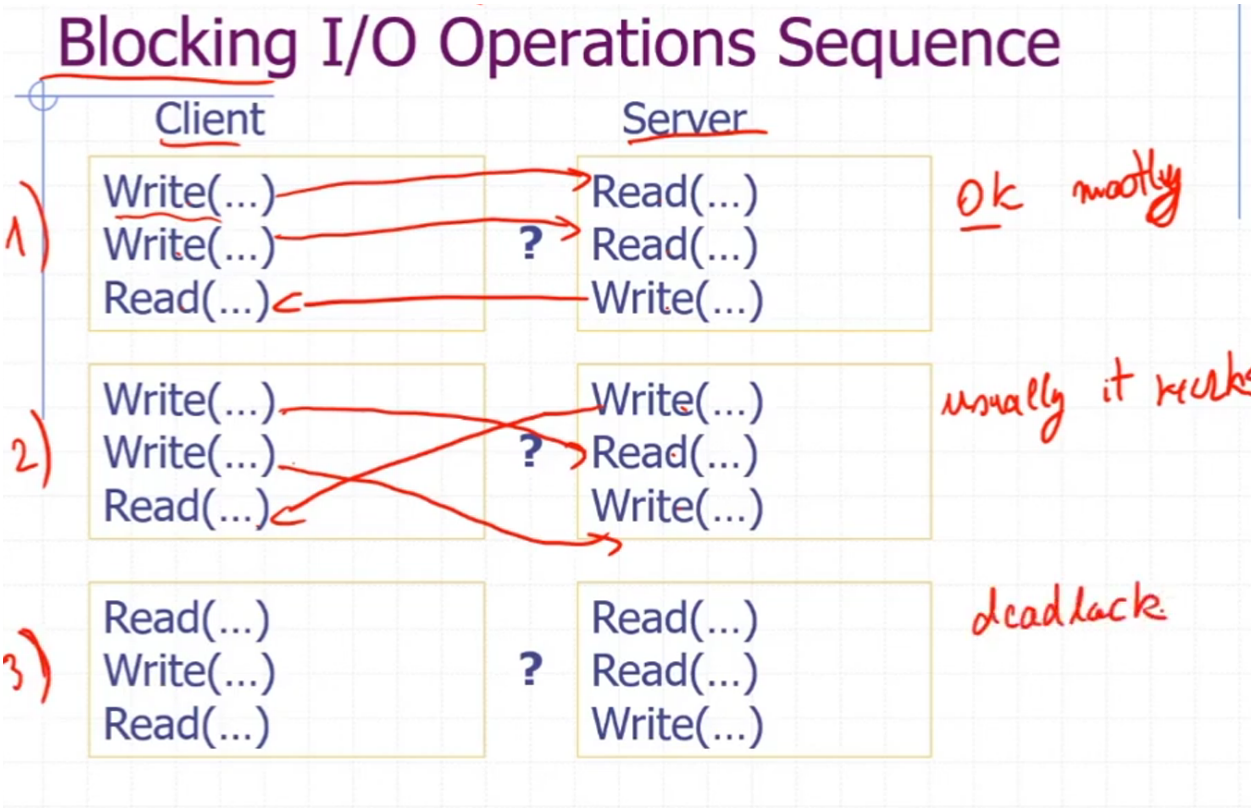


- nonblocking

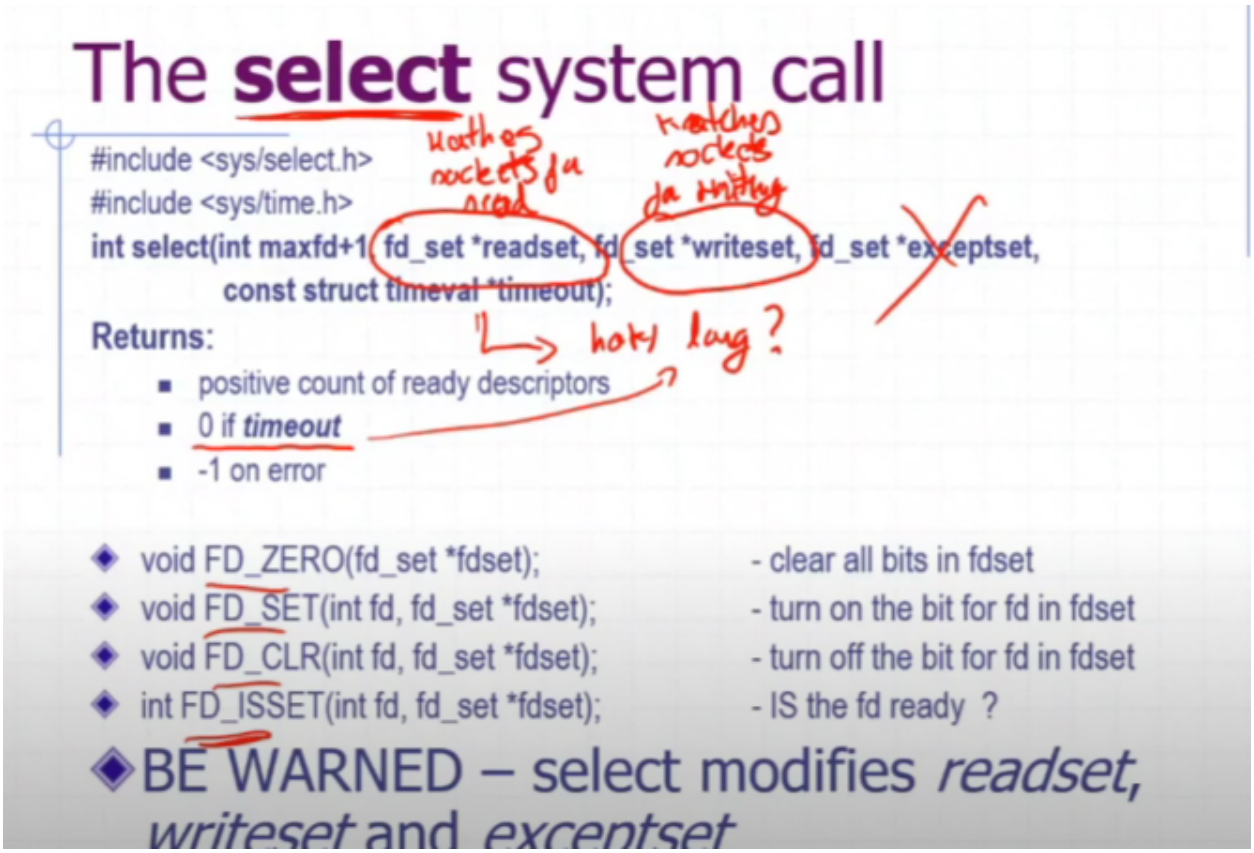


- muliplexing (select and pool)





The select system call



DNS

-

Default Gateway

- usually router address
- when a host wants to communicate with the internet

Broadcast

- 1 → everyone in the LAN
- destination MAC Address: FFFFFFFFFFF

Unicast

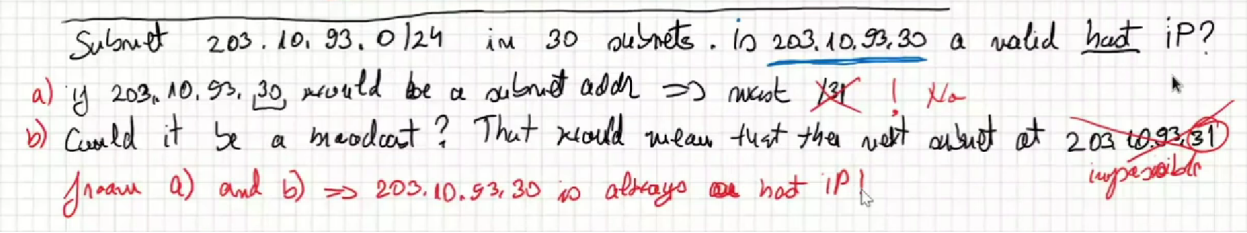
- 1 → 1

Multicast

- 1 → multiple
- when outside of LAN

Problems / Tips

- always allocate ips from the largest subnet to the smallest



- 50+ question
  - multiple choice(even all of them can be valid)
  - short answer
  - vznegative points
  - problem solving

- network-assisted congestion control
  - routers provide feedback to end systems
    - single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
    - explicit rate sender should send at