

TCP/IP and TCP protocols

Ethernet

- Data Link Layer protocol
- Ethernet (IEEE 802.3) is widely used.
- Supported by a variety of physical layer implementations.
- Multi-access (shared medium).

CSMA/CD

- *Carrier Sense Multiple Access with Collision Detection*
- *Carrier Sense* : can tell when another host is transmitting
- *Multiple Access* : many hosts on 1 wire
- *Collision Detection* : can tell when another host transmits at the same time.

An Ethernet Frame



- The preamble is a sequence of alternating 1s and 0s used for synchronization.
- CRC is Cyclic Redundancy Check

Ethernet Addressing

- Every Ethernet interface has a unique 48 bit address (a.k.a. *hardware address*).
 - Example: **C0:B3:44:17:21:17**
 - The broadcast address is all 1's.
 - Addresses are assigned to vendors by a central authority.
- Each interface looks at every *frame* and inspects the destination address. If the address does not match the hardware address of the interface (or the broadcast address), the frame is discarded.


Internet Protocol

- IP is the network layer
 - packet delivery service (host-to-host).
 - translation between different data-link protocols
- IP provides connectionless, unreliable delivery of *IP datagrams*.
 - Connectionless: each datagram is independent of all others.
 - Unreliable: there is no guarantee that datagrams are delivered correctly or even delivered at all.

IP Addresses

- IP addresses are not the same as the underlying data-link (MAC) addresses.
- IP is a network layer - it must be capable of providing communication between hosts on different kinds of networks (different data-link implementations).
- The address must include information about what *network* the receiving host is on. This is what makes routing feasible.

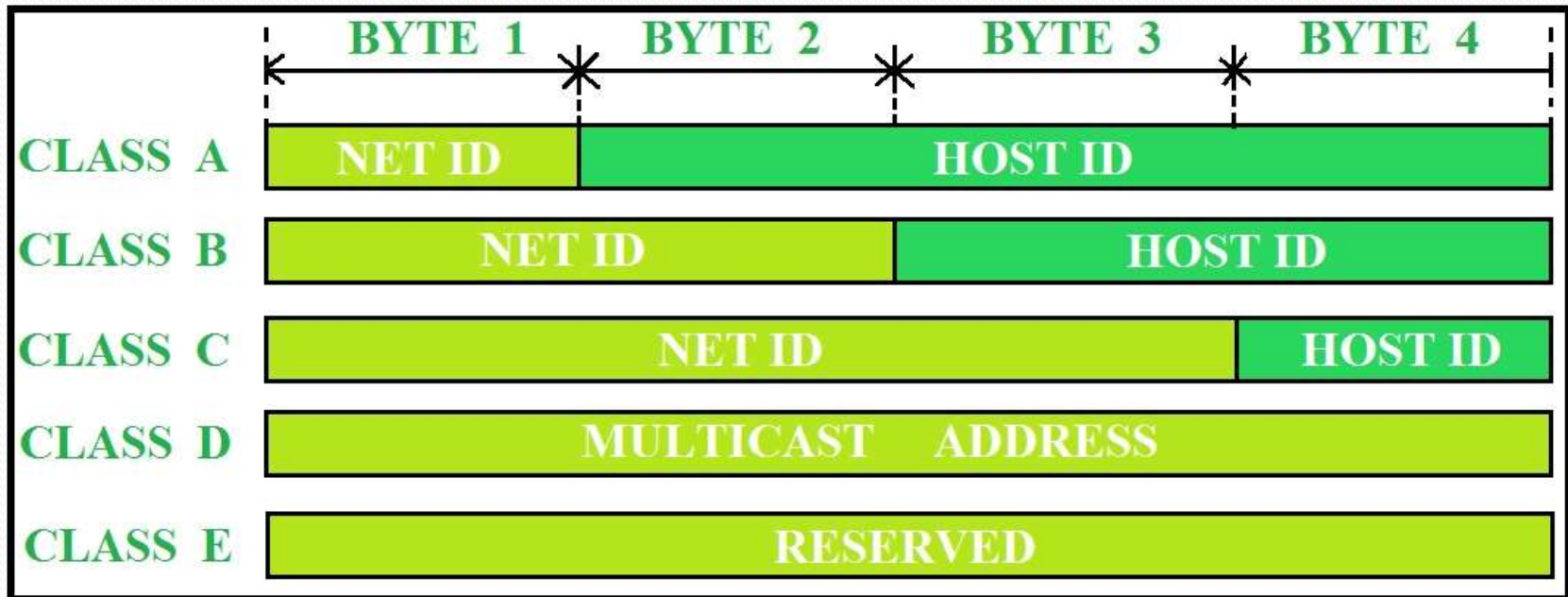
IP Addresses

- IP addresses are *logical* addresses (not physical)
- 32 bits.  IPv4 (*version 4*)
- Includes a network ID and a host ID.
- Every host must have a unique IP address.
- IP addresses are assigned by a central authority (*American Registry for Internet Numbers* for North America).

Network and Host IDs

- A Network ID is assigned to an organization by a global authority.
- Host IDs are assigned locally by a system administrator.
- Both the Network ID and the Host ID are used for routing.

CLASSES



The *four* formats of IP Addresses

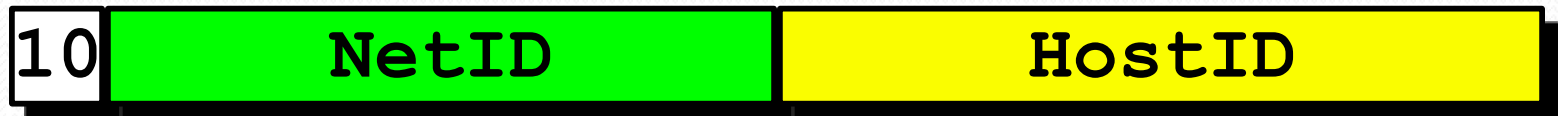
Class

A



128 possible network IDs, over 4 million host IDs per network ID

B



16K possible network IDs, 64K host IDs per network ID

C



Over 2 million possible network IDs, 256 host IDs per network ID

D



8 bits

8 bits

8 bits

8 bits

CLASS IP ADDRESS RANGE

The 5 Different Classes Of IP Address

Class A : 1.0.0.0 to 127.255.255.255

Class B : 128.0.0.0 to 191.255.255.255

Class C : 192.0.0.0 to 223.255.255.255

Class D : 224.0.0.0 to 239.255.255.255

Class E : 240.0.0.0 to 255.255.255.255

*The IP Classes listed above are not all usable by hosts!
Here we are simply looking at the range each Class covers*

IP Addresses

- IP Addresses are usually shown in *dotted decimal* notation:

1.2.3.4

00000001 00000010 00000011 00000100

- cse.unr.edu is 134.197.40.3

10000110 11000101 00101000 00000010



CSE has a class B network

Host and Network Addresses

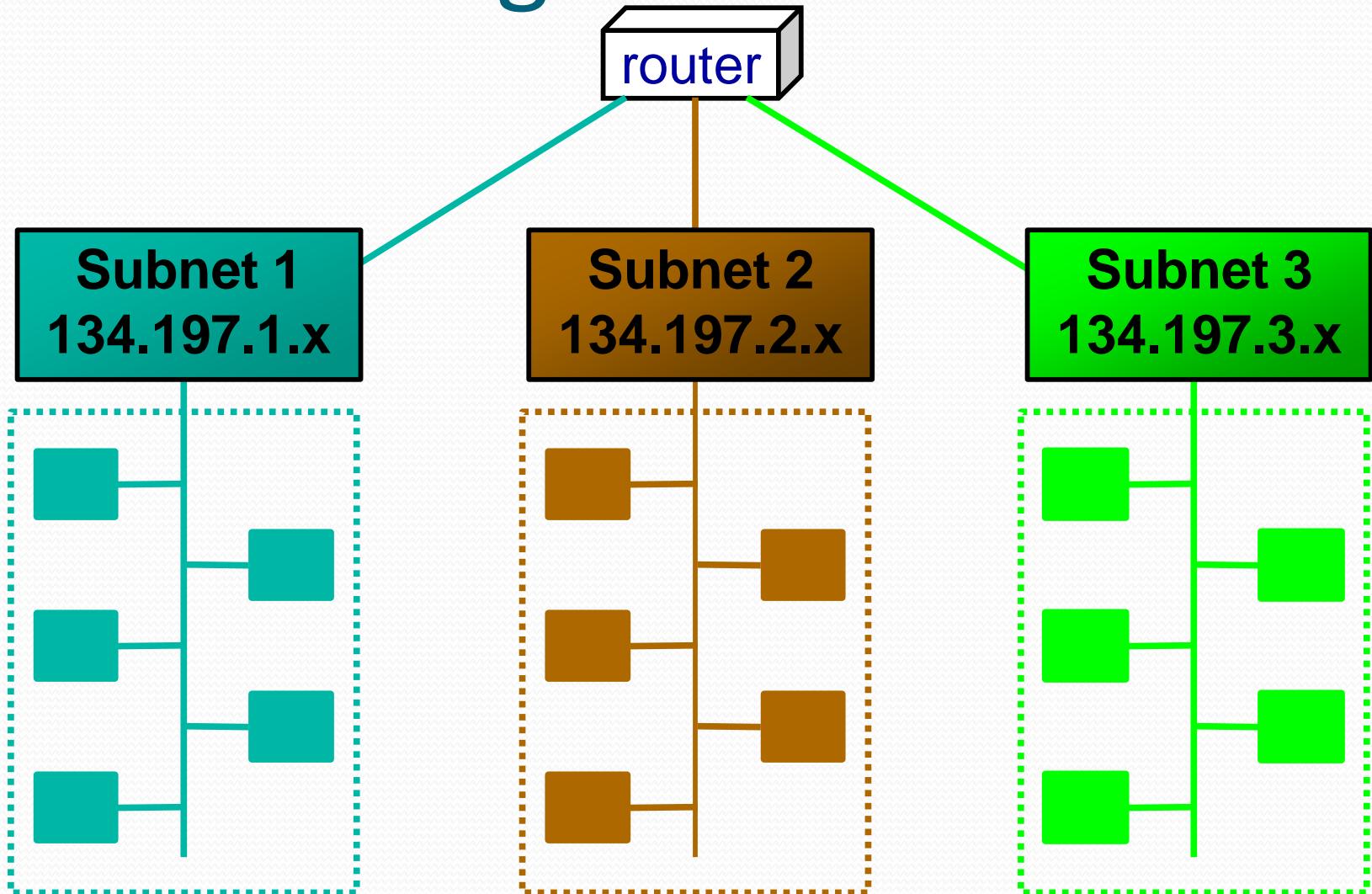
- A single network interface is assigned a single IP address called the *host* address.
- A host may have multiple interfaces, and therefore multiple *host* addresses.
- Hosts that share a network all have the same IP *network* address (the network ID).
- An IP address that has a host ID of all 0s is called a *network address* and refers to an entire network.

Subnet Addresses

- An organization can subdivide its host address space into groups called subnets.
- The subnet ID is generally used to group hosts based on the physical network topology.

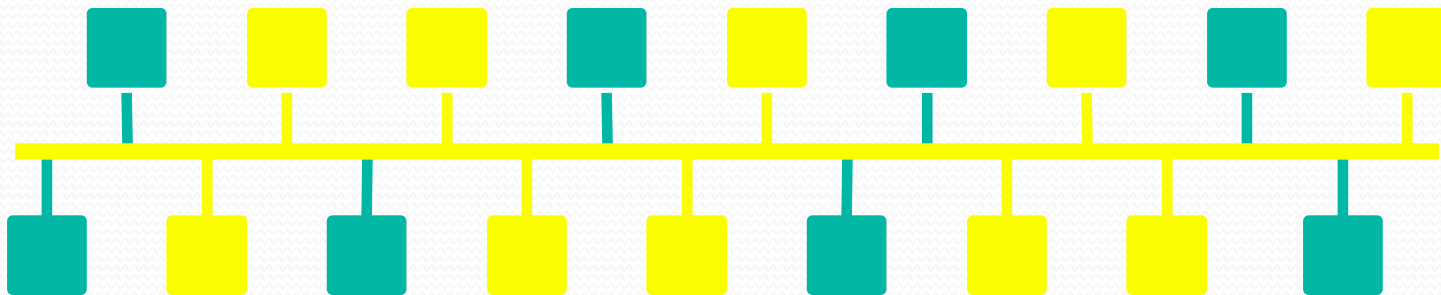


Subnetting



Subnetting

- Subnets can simplify routing.
- IP subnet broadcasts have a hostID of all 1s.
- It is possible to have a single wire network with multiple subnets.



Mapping IP Addresses to Hardware Addresses

- IP Addresses are not recognized by hardware.
- If we know the IP address of a host, how do we find out the hardware address ?
- The process of finding the hardware address of a host given the IP address is called

Address Resolution

ARP

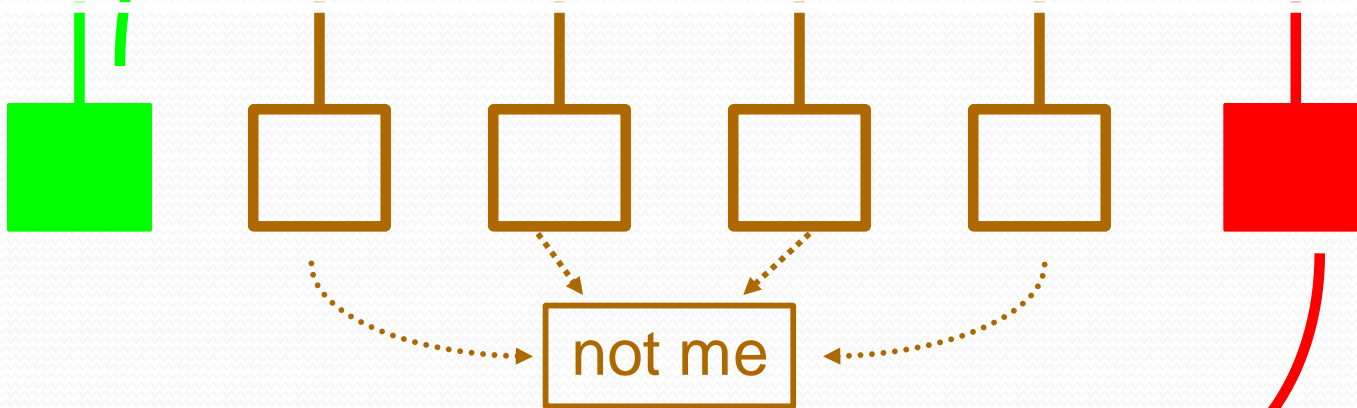
Arp Arp!



- The *Address Resolution Protocol* is used by a sending host when it knows the IP address of the destination but needs the Ethernet (or whatever) address.
- ARP is a broadcast protocol - every host on the network receives the request.
- Each host checks the request against its IP address - the right one responds.
- hosts *remember* the hardware addresses of each other.

ARP conversation

HEY - Everyone please listen!
Will 128.213.1.5 please send me
his/her Ethernet address?



Hi Green! I'm 128.213.1.5, and
my Ethernet address is
87:A2:15:35:02:C3

ICMP

Internet Control Message Protocol

- ICMP is a protocol used for exchanging control messages.
- ICMP uses IP to deliver messages.
- ICMP messages are usually generated and processed by the IP software, not the user process.

ICMP Message Types

- Echo Request
- Echo Response
- Destination Unreachable
- Redirect
- Time Exceeded
- Redirect (route change)
- there are more ...

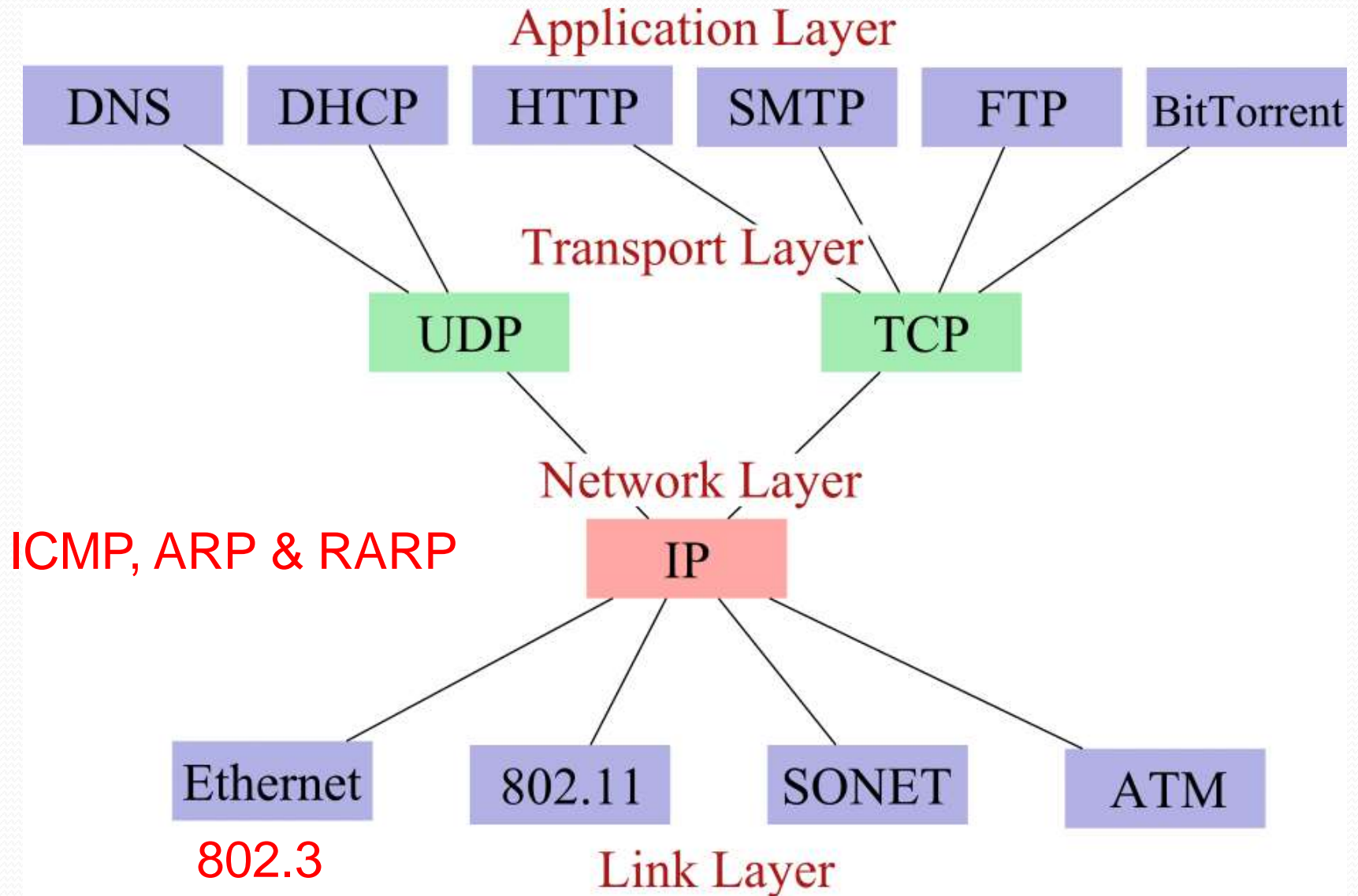
Transport Layer & TCP/IP

Q: We know that IP is the network layer - so TCP must be the transport layer, right ?

A: No... well, almost.

TCP is only part of the TCP/IP transport layer - the other part is UDP (User Datagram Protocol).

The Internet Hourglass

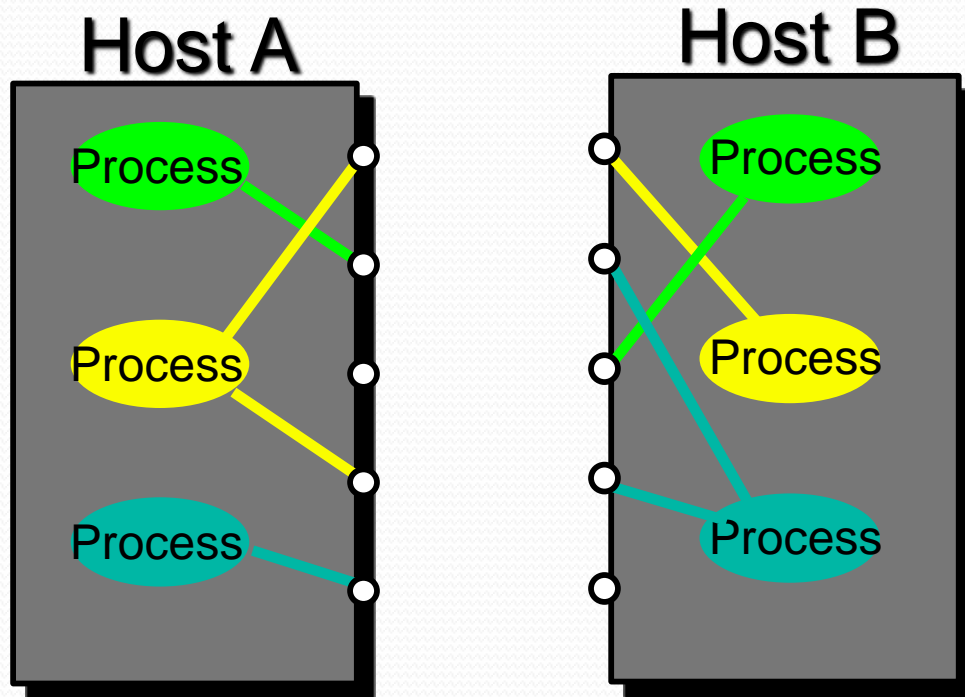


UDP User Datagram Protocol

- UDP is a transport protocol
 - communication between processes
- UDP uses IP to deliver datagrams to the right host.
- UDP uses *ports* to provide communication services to individual processes.

Ports

- TCP/IP uses an abstract destination point called a protocol port.
- Ports are identified by a positive integer.
- Operating systems provide some mechanism that processes use to specify a port.



UDP

- Datagram Delivery
- Connectionless
- Unreliable
- Minimal

UDP Datagram Format

Source Port	Destination Port
Length	Checksum
Data	

TCP

Transmission Control Protocol

- TCP is an alternative transport layer protocol supported by TCP/IP.
- TCP provides:
 - Connection-oriented
 - Reliable
 - Full-duplex
 - Byte-Stream

Connection-Oriented

- *Connection oriented* means that a virtual connection is established before any user data is transferred.
- If the connection cannot be established, the user program is notified (finds out).
- If the connection is ever interrupted, the user program(s) finds out there is a problem.

Reliable

- *Reliable* means that every transmission of data is acknowledged by the receiver.
- Reliable does not mean that things don't go wrong, it means that we find out when things go wrong.
- If the sender does not receive acknowledgement within a specified amount of time, the sender retransmits the data.

Byte Stream

- *Stream* means that the connection is treated as a stream of bytes.
- The user application does not need to package data in individual datagrams (as with UDP).

Buffering

- TCP is responsible for buffering data and determining when it is time to send a datagram.
- It is possible for an application to tell TCP to send the data it has buffered without waiting for a buffer to fill up.

Full Duplex

- TCP provides transfer in both directions (over a single virtual connection).
- To the application program these appear as 2 unrelated data streams, although TCP can piggyback control and data communication by providing control information (such as an ACK) along with user data.

TCP Ports

- Interprocess communication via TCP is achieved with the use of ports (just like UDP).
- UDP ports have no relation to TCP ports (different name spaces).

TCP Segments

- The chunk of data that TCP asks IP to deliver is called a *TCP segment*.
- Each segment contains:
 - data bytes from the byte stream
 - control information that identifies the data bytes

TCP Segment Format

1 byte

1 byte

1 byte

1 byte

Source Port			Destination Port		
Sequence Number					
Request Number					
offset	Reser.	Control		Window	
Checksum			Urgent Pointer		
Options (if any)					
Data					

TCP process

- When a client requests a connection, it sends a “SYN” segment (a special TCP segment) to the server port.
- SYN stands for *synchronize*. The SYN message includes the client’s ISN.
- ISN is Initial Sequence Number.

More...

- Every TCP segment includes a *Sequence Number* that refers to the first byte of *data* included in the segment.
- Every TCP segment includes a *Request Number* (*Acknowledgement Number*) that indicates the byte number of the next data that is expected to be received.
 - All bytes up through this number have already been received.

And more...

- There are a bunch of control flags:
 - URG: urgent data included.
 - ACK: this segment is (among other things) an acknowledgement.
 - RST: error - abort the session.
 - SYN: synchronize Sequence Numbers (setup)
 - FIN: polite connection termination.

And more...

- MSS: Maximum segment size (A TCP option)
- Window: Every ACK includes a Window field that tells the sender how many bytes it can send before the receiver will have to toss it away (due to fixed buffer size).

Addressing in TCP/IP

- Each TCP/IP address includes:
 - Internet Address
 - Protocol (UDP or TCP)
 - Port Number

NOTE: TCP/IP is a *protocol suite* that includes IP, TCP and UDP

TCP vs. UDP

Q: Which protocol is better ?

A: It depends on the application.

TCP provides a connection-oriented, reliable, byte stream service (lots of overhead).

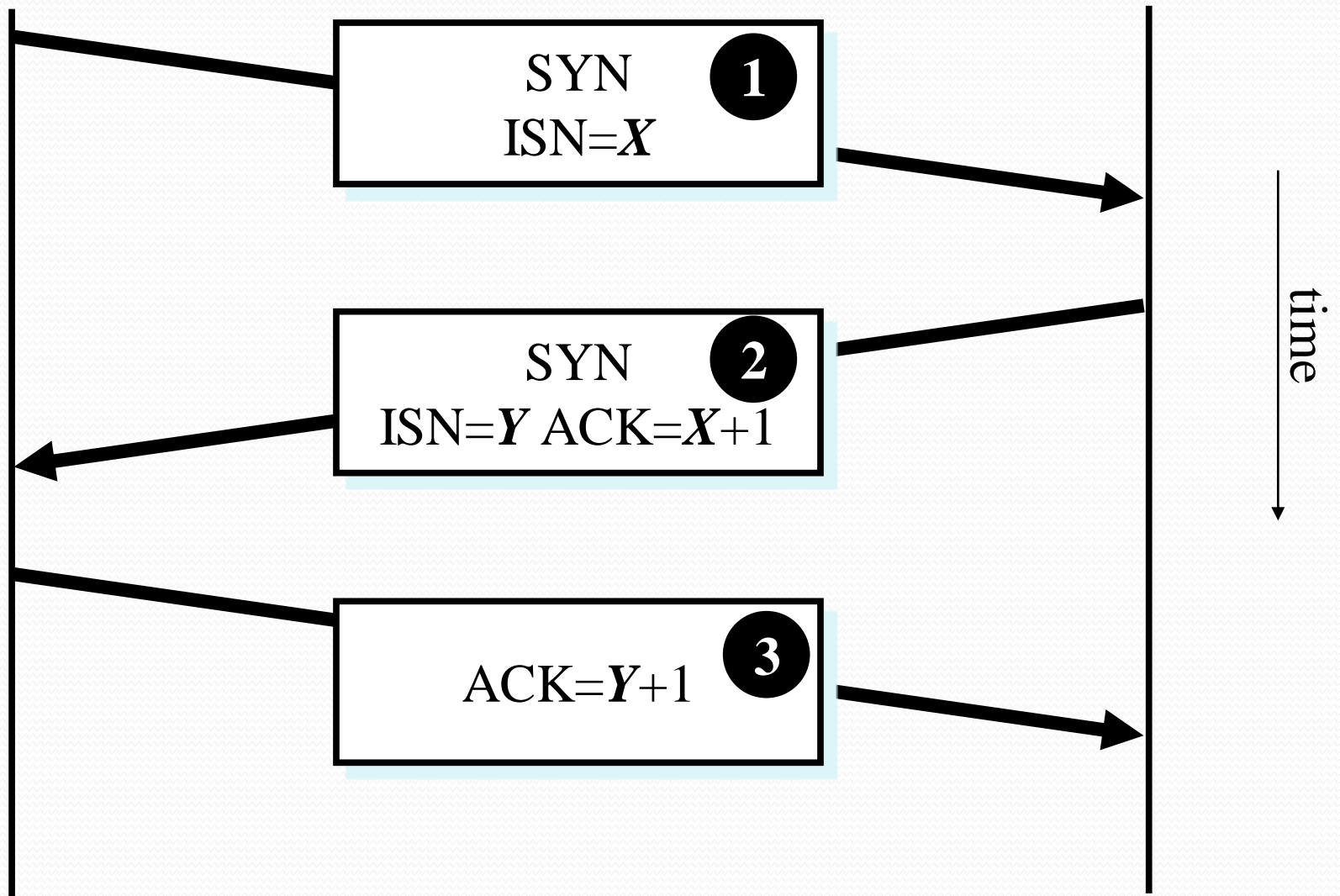
UDP offers minimal datagram delivery service (as little overhead as possible).

TCP Connection Creation

- A *server* accepts a connection.
 - Must be looking for new connections!
- A *client* requests a connection.
 - Must *know* where the server is!

Client

Server



TCP Data and ACK

- Once the connection is established, data can be sent.
- Each data segment includes a sequence number identifying the first byte in the segment.
- Each segment (data or empty) includes a request number indicating what data has been received.

TCP Buffers

- The TCP layer doesn't know when the application will ask for any received data.
 - TCP buffers incoming data so it's ready when we ask for it.
- Both the client and server allocate buffers to hold incoming and outgoing data
 - The TCP layer does this.
- Both the client and server announce with every ACK how much buffer space remains (the Window field in a TCP segment).

Send Buffers

- The application gives the TCP layer some data to send.
- The data is put in a send buffer, where it stays until the data is ACK'd.
 - it has to stay, as it might need to be sent again!
- The TCP layer won't accept data from the application unless (or until) there is buffer space.

ACKs

- A receiver doesn't have to ACK every segment (it can ACK many segments with a single ACK segment).
- Each ACK can also contain outgoing data (piggybacking).
- If a sender doesn't get an ACK after some time limit (MSL) it resends the data.