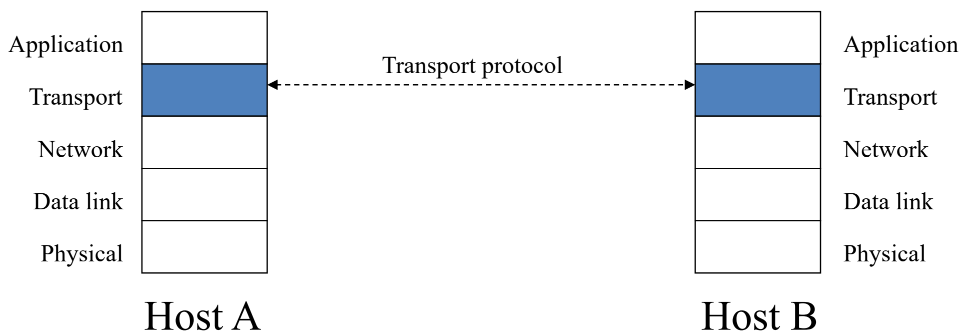
CSE 156
Network Programming

Lecture 1

***

Network Reference Architecture

- Layered architecture (protocol stack)
- Transport: end-to-end message delivery
  - User Datagram Protocol (UDP)
  - Transmission Control Protocol (TCP)



Ethernet

- Data Link Layer protocol
- Ethernet (IEEE 802.3) is widely used
- Supported by a variety of physical layer implementations
- Traditionally multi-access (shared medium)
  - CSMA/CD
- Nowadays usually used as point-to-point

An Ethernet Frame

Preamble	Destination Address	Source Address	Len	Data	CRC
8 bytes	6	6	2	0-1500	4

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

Ethernet Addressing

- Every Ethernet interface has a unique 48-bit address (i.e., hardware or MAC address)
  - Example: 0a:31:c1:cb:49:3a
  - 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 (i.e., different data-link instances)
- The address includes information about what *network* the receiving host is on
  – This is what makes routing scalable

IP Addresses

- IP addresses are *logical* addresses (not physical)
  - Assigned by software or user
- 32 bits in IP version 4
- Every host on a public network must have a unique IP address
- Includes a Network ID and a Host ID

The *four* formats of IP Addresses

Class

A | **0** | **NetID** | **HostID**

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

B | **10** | **NetID** | **HostID**

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

C | **110** | **NetID** | **HostID**

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

D | **1110** | **Multicast Address**

8 bits 8 bits 8 bits 8 bits

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 of the organization
- Both the Network ID and the Host ID are used for routing
- IP Addresses are usually shown in *dotted decimal* notation:

1.2.3.4

00000001 00000010 00000011 00000100

Host and Network Addresses

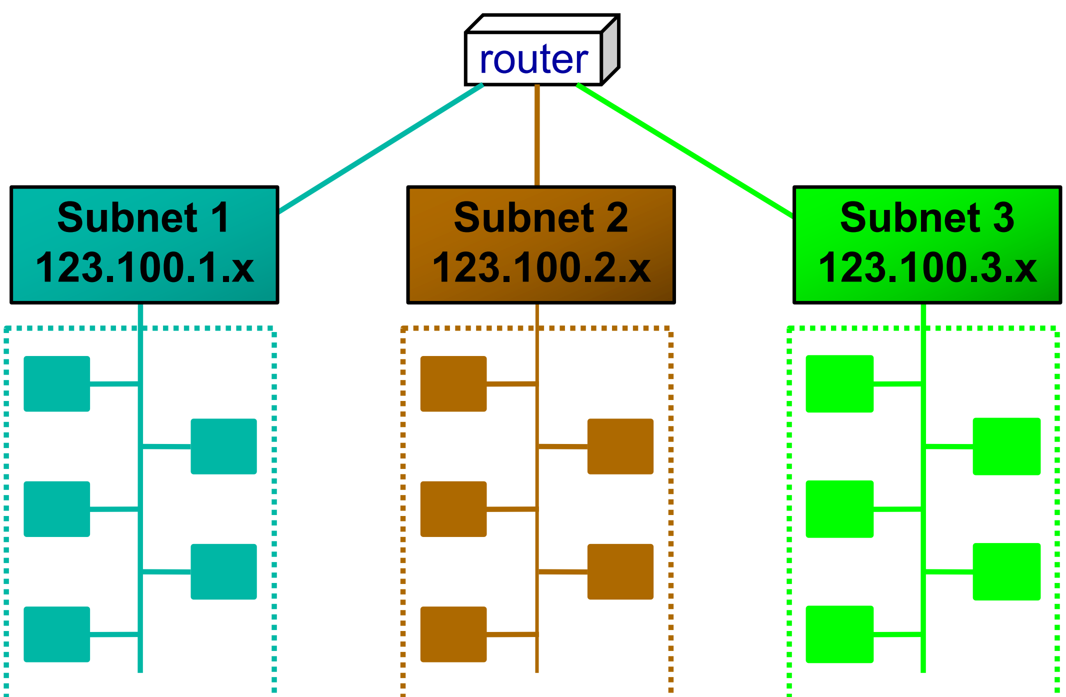
- A single network interface is assigned a single IP address called the *host* address.
- A host may have multiple interfaces, and therefore multiple *host* addresses.
- Hosts that share a network all have the same IP *network* address (the Network ID).
- An IP address that has a Host ID of all 0s is 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
- Use subnet mask to identify the boundary between Subnet ID and Host ID: bits of 1 cover the Network and Subnet ID, bits of 0 cover the Host ID. E.g., 128.x.x.x/24

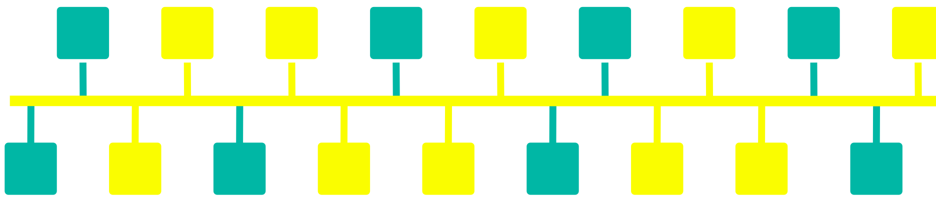
10	NetID	SubnetID	HostID
11	11111111111111	11111111	00000000

Subnetting



Subnetting

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



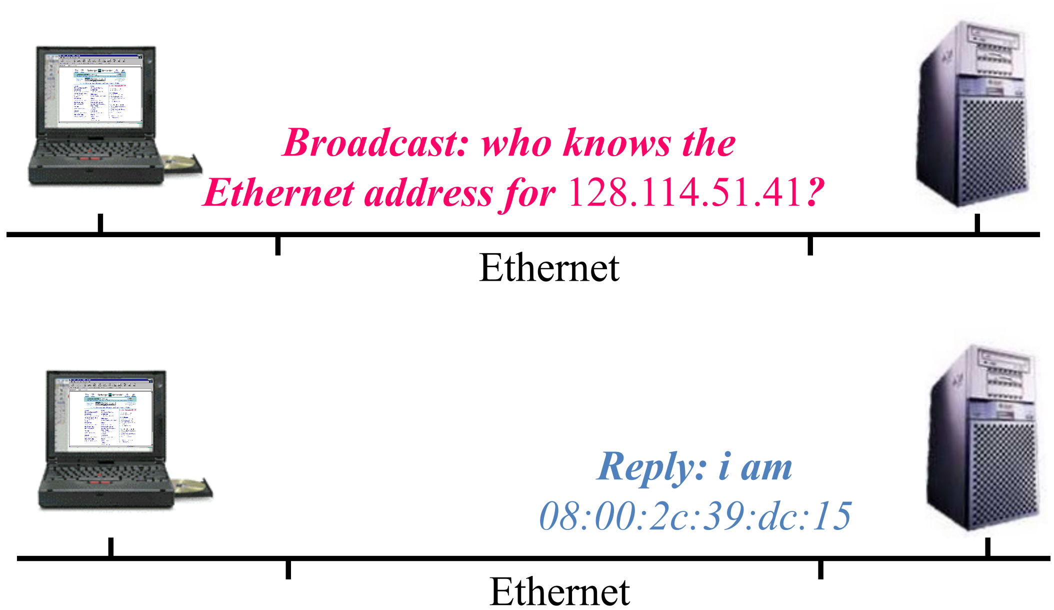
Mapping IP Addresses to Hardware Addresses

- 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

Address Resolution Protocol (ARP)

- Used by a sending host when it knows the IP address of the destination but needs the hardware (e.g., Ethernet) address
- Uses broadcast: 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 (e.g.)



IP Datagram

<table>
<tr><th colspan="2">1 byte</th><th colspan="2">1 byte</th><th colspan="2">1 byte</th><th colspan="2">1 byte</th></tr>
<tr><td>VERS</td><td>HL</td><td colspan="2">Service</td><td colspan="4">Total Length</td></tr>
<tr><td colspan="4">Datagram ID</td><td>FLAG</td><td colspan="3">Fragment Offset</td></tr>
<tr><td colspan="2">TTL</td><td colspan="2">Protocol</td><td colspan="4">Header Checksum</td></tr>
<tr><td colspan="8">Source Address</td></tr>
<tr><td colspan="8">Destination Address</td></tr>
<tr><td colspan="8">Options (if any)</td></tr>
<tr><td colspan="8">Data</td></tr>
</table>

IP Datagram Fragmentation

- Packets are fragmented due to a link's Maximum Transmission Unit (MTU)
- Each fragment (packet) has the same structure as the IP datagram
- IP specifies that datagram reassembly is done only at the destination (not on a hop-by-hop basis)
- If any of the fragments are lost, the entire datagram is discarded (and an ICMP message is sent to the sender)
- Allow for the ability to prevent fragmentation

Error or Informational Messages

- A mechanism is needed for IP to communicate error conditions found in the network

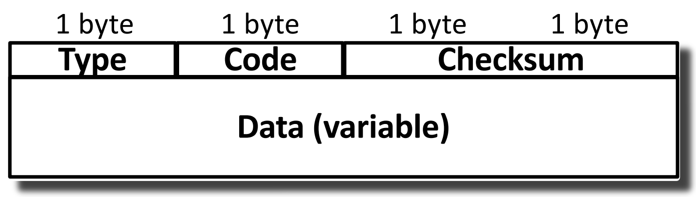
- If an error is found during the processing of a packet, it's discarded and generally a message is sent to the sender
  - E.g. header checksum problem
  - E.g. If fragmentation is required and the don't fragment bit is set

ICMP

- Internet Control Message Protocol
- Used for exchanging control messages
- Uses IP to deliver its messages (Layer 3?)
- ICMP messages are usually generated and processed by the IP software, not the user process
- Error responses to the sender include the IP header plus the first 8 bytes of the original datagram's data

ICMP Message Types

- Echo Request (Type 8, Code 0)
- Echo Response (Type 0, Code 0)
- Destination Unreachable (Type3, …)
  - Port Unreachable (Type 3, Code 3)
  - Fragmentation Needed (Type 3, Code 4)
- Redirect (Type 5)
- Time Exceeded (Type 11)
- And more ...



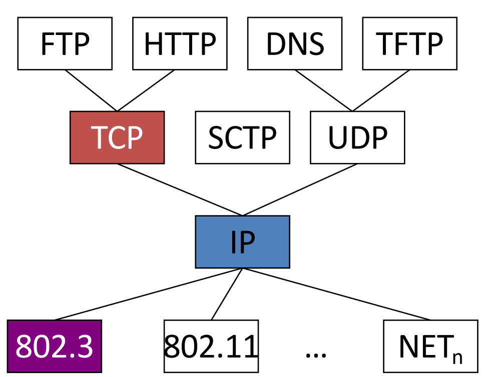
ICMP Application

- Ping
- Traceroute
- Path MTU Discovery (PMTUD)

Transport Layer

- Several transport layer protocols supported by TCP/IP stack
  - TCP, UDP, SCTP
- Applications select transport based on required functionality
  - Timing: e.g., low delay
  - Bandwidth: e.g., minimum bandwidth
  - Data loss: e.g., 100% reliable
  - E.g., reliable delivery vs. minimal delay

TCP/IP Hourglass



Application: protocols

Transport: provide logical channels to apps

Network: ICMP, ARP

Link: heterogeneous net hardware: ethernet, wifi

TCP/IP Addressing

Each TCP/IP address includes:

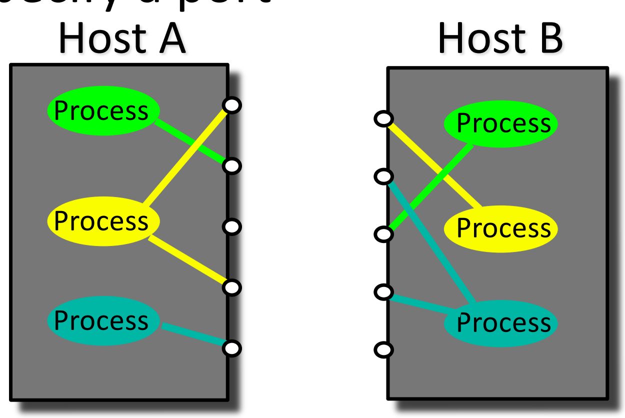
– Network: Internet Address

– *Transport*: Port Number

– Protocol (UDP or TCP or …)

TCP/IP Protocol Ports

- Abstract destination points used by processes
  - i.e., applications (well-known and otherwise)
- Ports are identified by a 16-bit positive integer
- Operating systems provide APIs that processes use to specify a port

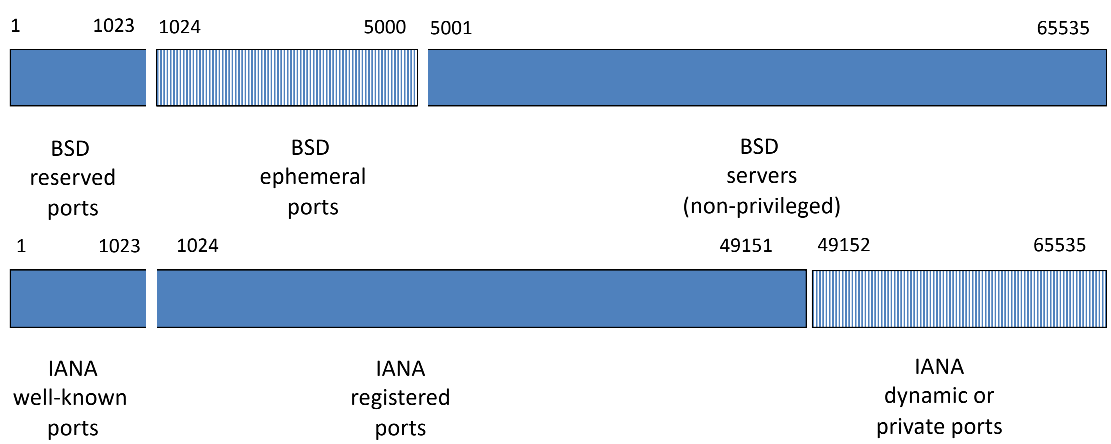


Port Numbers

- Multiple processes use TCP and UDP at any given time
  - Differentiate (multiplex) between processes
- Well known ports: 0 through 1023
  - Controlled and assigned by IANA
- Registered ports: 1024 through 49151
- Dynamic or private ports: 49152 through 65535

Port Numbers (2)

- Non-reserved port ranges can vary from OS to OS. E.g., linux
  linux$ cat /proc/sys/net/ipv4/ip_local_port_range
  32768   60999
- Standardized by IANA (Internet Assigned Numbers Authority)
- Example, Fig. 2-10: allocation of port numbers



Centos Linux /etc/services (E.g. 1)

```
# CentOS Linux release 7.9.2009 (Core)
# service-name  port/protocol  [aliases ...]   [# comment]

tcpmux          1/tcp                           # TCP port service multiplexer
tcpmux          1/udp                           # TCP port service multiplexer
rje             5/tcp                           # Remote Job Entry
rje             5/udp                           # Remote Job Entry
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
systat          11/udp          users
daytime         13/tcp
daytime         13/udp
qotd            17/tcp          quote
qotd            17/udp          quote
msp             18/tcp                          # message send protocol (historic)
msp             18/udp                          # message send protocol (historic)
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp-data        20/udp
ftp             21/tcp
ftp             21/udp          fsp fspd
ssh             22/tcp                          # The Secure Shell (SSH) Protocol
ssh             22/udp                          # The Secure Shell (SSH) Protocol
# ...
nimbusdbctrl    48005/tcp               # NimbusDB Control
3gpp-cbsp       48049/tcp               # 3GPP Cell Broadcast Service Protocol
isnetserv       48128/tcp               # Image Systems Network Services
isnetserv       48128/udp               # Image Systems Network Services
blp5            48129/tcp               # Bloomberg locator
blp5            48129/udp               # Bloomberg locator
com-bardac-dw   48556/tcp               # com-bardac-dw
com-bardac-dw   48556/udp               # com-bardac-dw
iqobject        48619/tcp               # iqobject
iqobject        48619/udp               # iqobject
matahari        49000/tcp               # Matahari Broker
```

Ubuntu Linux /etc/services (E.g. 2)

```
# Ubuntu 18.04.4 LTS
# service-name  port/protocol  [aliases ...]   [# comment]

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
# ...
nimbusdbctrl    48005/tcp               # NimbusDB Control
3gpp-cbsp       48049/tcp               # 3GPP Cell Broadcast Service Protocol
isnetserv       48128/tcp               # Image Systems Network Services
isnetserv       48128/udp               # Image Systems Network Services
blp5            48129/tcp               # Bloomberg locator
blp5            48129/udp               # Bloomberg locator
com-bardac-dw   48556/tcp               # com-bardac-dw
com-bardac-dw   48556/udp               # com-bardac-dw
iqobject        48619/tcp               # iqobject
iqobject        48619/udp               # iqobject
matahari        49000/tcp               # Matahari Broker
```

***

TCP Overview (1)

- Transmission Control Protocol
- Most widely used transport protocol on the Internet
- Multiplexing service through “ports”
- Used with IP over many data link layers and many types of networks
- Flow and congestion control
- Round-trip delay estimation

TCP Overview (2)

TCP provides:

- Connection-oriented
- Reliable delivery
- Full-duplex communication
- 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 finds out
- If the connection is ever interrupted, the user program learns there is a problem

Reliable

- *Reliable* means that every transmission of data is acknowledged by the receiver.
- Reliable also means recovering from errors:
  - Lost packets
  - Out of order packets
  - Duplicate packets
- If the sender does not receive acknowledgement within a specified amount of time, the sender retransmits the data.

Full Duplex

- 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.

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.

TCP Segments

- The chunk of bytes 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
- Includes a *Sequence Number* that refers to the first byte of *data* included in the segment
- Includes an *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

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: (IP, port)
  - Sends a "SYN" segment (a special TCP segment) to the server port
  - The SYN message includes the client's Initial Sequence Number (ISN)

Client Starts

- A client starts by sending a SYN segment with the following information:
  - Client's ISN (generated pseudo-randomly)
  - Maximum Receive Window for client
  - Optionally (but usually) MSS option (largest datagram accepted)
  - No payload! (Only TCP headers)

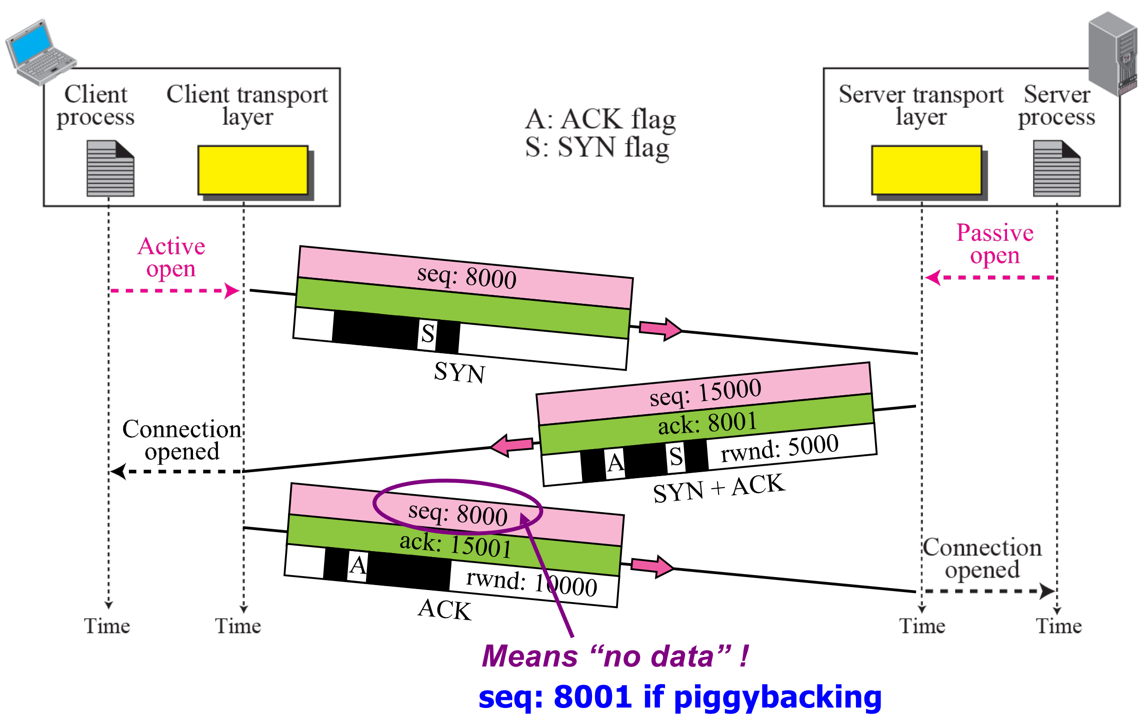
Server Response

- When a waiting server sees a new connection request, the server sends back a SYN segment with:
  - Server's ISN (generated pseudo-randomly)
  - Acknowledgement Number is Client ISN+1
  - Maximum Receive Window for server
  - Optionally (but usually) MSS
  - No payload! (Only TCP headers)

Finally

- When the Server's SYN is received, the client sends back an ACK with:
  - Request Number is Server's ISN+1

TCP Connection Establishment



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 an acknowledgement number indicating what data has been received

TCP Buffers

- 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 (i.e., the window field in a TCP segment)
- 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

Send Buffers

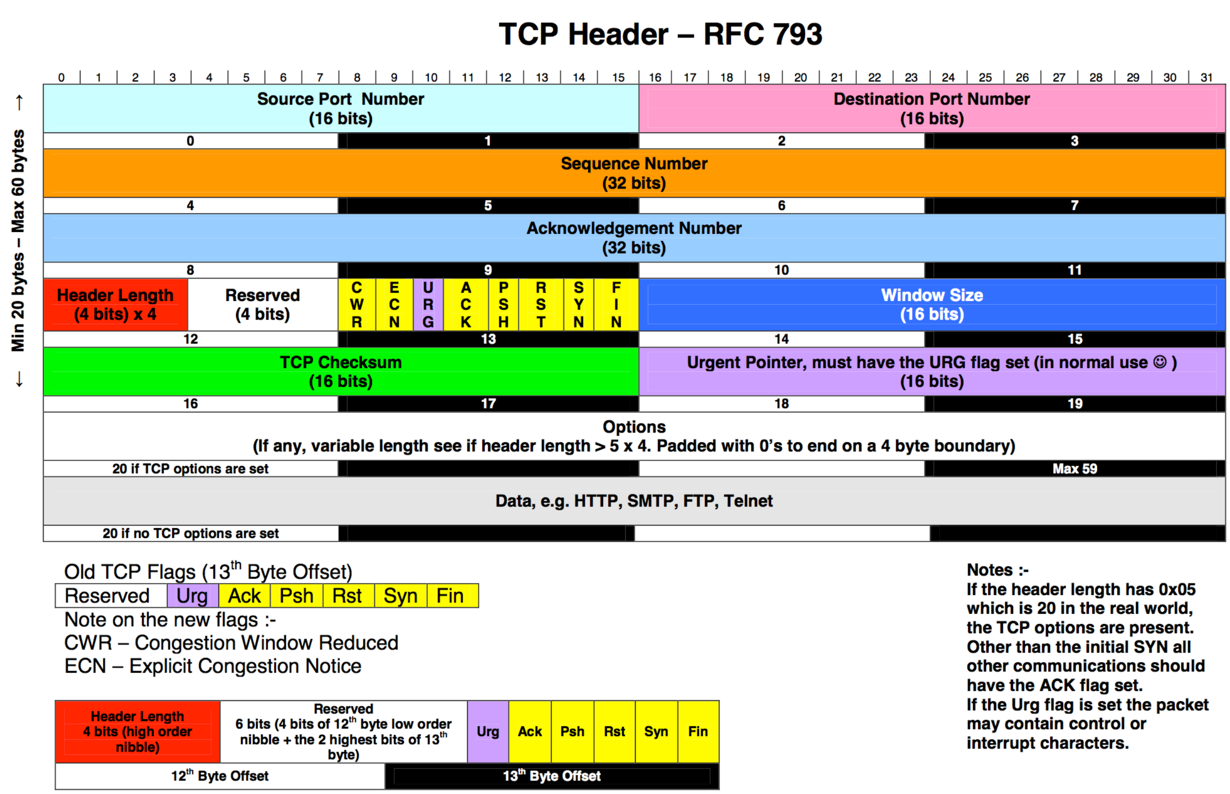
- The application gives the TCP layer some data to send.
- The data is put in a send buffer, where it stays until the data is ACK'd.
  - It has to remain, as it might need to be sent again!
- The TCP layer won't accept data from the application until it has buffer space.

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

TCP Segment Order

- Most TCP implementations will accept out-of-order segments (if there is room in the buffer).
- Once the missing segments arrive, a single ACK can be sent for the whole thing.
- Remember: IP delivers TCP segments, and IP in not reliable - IP datagrams can be lost or arrive out of order.



Notes :-
If the header length has 0x05 which is 20 in the real world, the TCP options are present. Other than the initial SYN all other communications should have the ACK flag set.
If the Urg flag is set the packet may contain control or interrupt characters.

TCP Control Flags

- **SYN**: Synchronize sequence numbers
- **ACK**: This segment includes an ack (Ack Number field valid)
- **FIN**: Normal termination of a connection
- **RST**: Reset connection because of an error
- **URG**: Urgent Pointer field valid in header
- **ECN, CWR**: Used to support Explicit Congestion Notification (ECN)
- **PSH**: Push flag to notify that data must be sent immediately by the sender, and be delivered to the receiving application immediately on arrival

Header Fields

- MSS: maximum segment size (A TCP option)

- Window: every ACK includes a window size 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)

Termination

- TCP can send a RST segment that terminates a connection if something is wrong
- Usually the application tells TCP to terminate the connection politely with a FIN segment

FIN

- Either end of the connection can initiate termination
- A FIN is sent, which means the application is done sending data
- The FIN is ACK'd
- The other end must now send a FIN
- That FIN must be ACK'd

TCP TIME_WAIT

- Once a TCP connection has been terminated (the last ACK sent) there is some unfinished business:
  - What if the ACK is lost? The last FIN will be resent and it must be ACK'd
  - What if there are lost or duplicated segments that finally reach the destination after a long delay?

- TCP keeps the state for connections for this period to handle these situations

TCP Connection Termination (1)

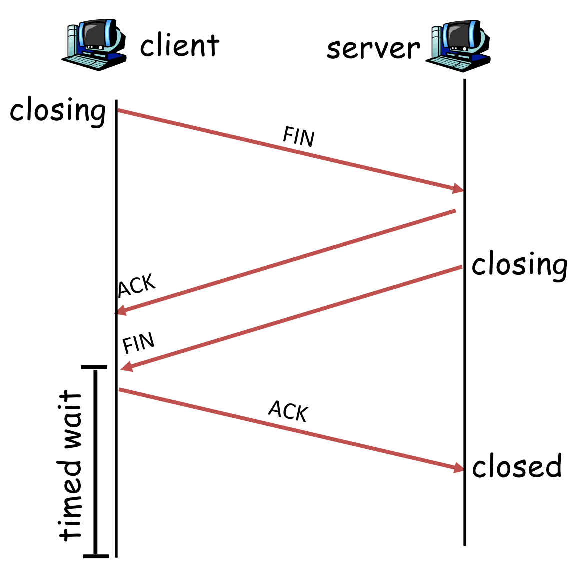
<u>Closing a connection:</u>

client closes socket:

close()

Step 1: client end system sends TCP FIN control segment to server

Step 2: server receives FIN, replies with ACK. Closes connection, sends FIN.



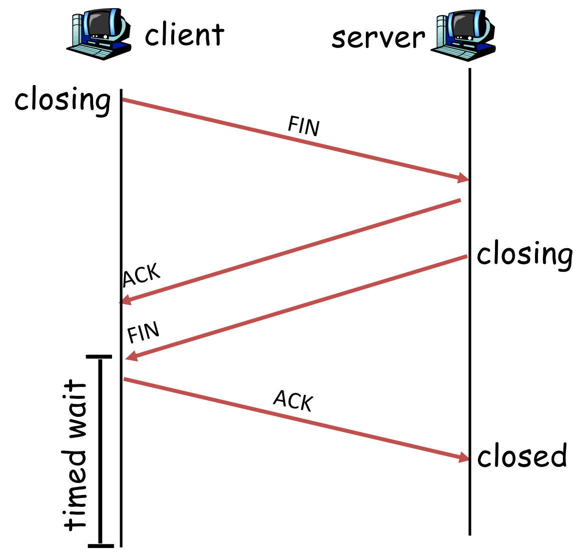
TCP Connection Termination (2)

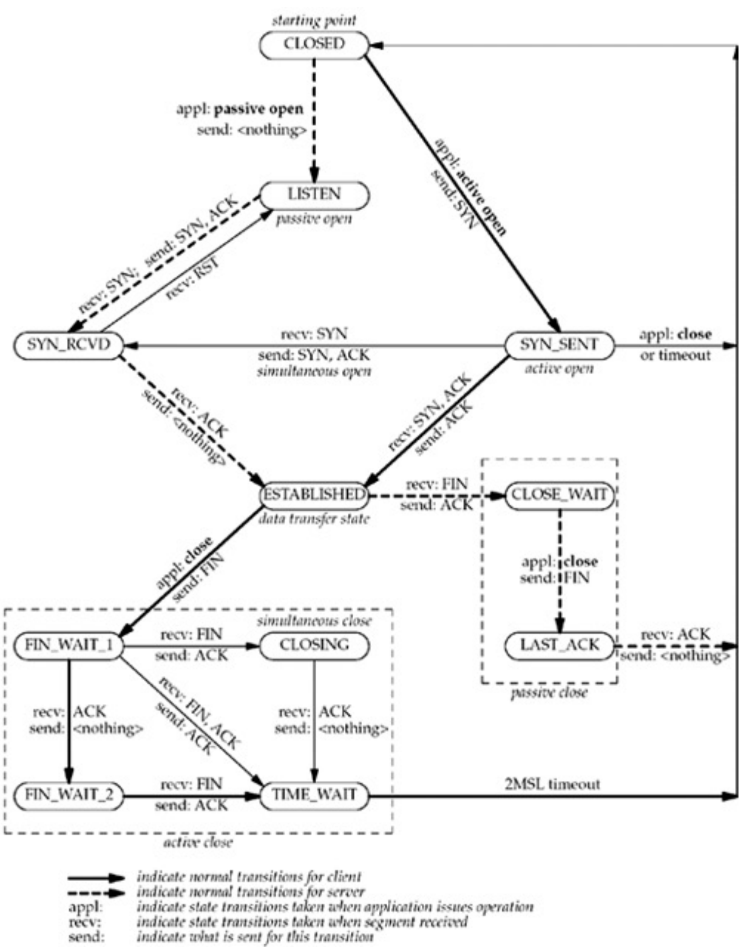
Step 3: client receives FIN:

- Replies with ACK.
- Enters “timed wait”
- Responds with ACK to received FIN (dups)

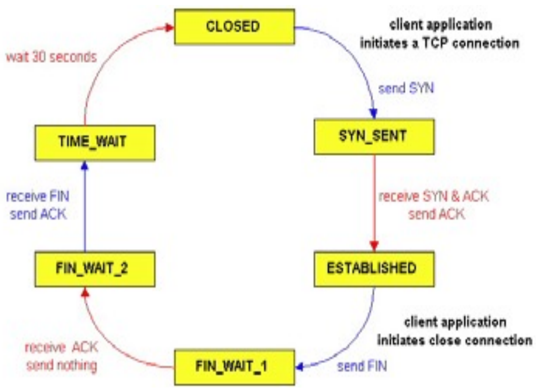
Step 4: server, receives ACK. Connection closed.

Note: with small modification, can handle simultaneous FINs.

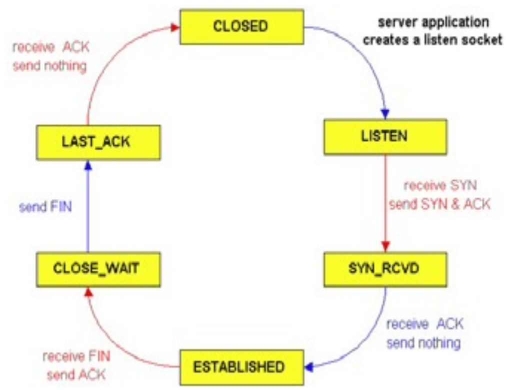




TCP Connection Lifecycle



TCP Client lifecycle



TCP Server lifecycle

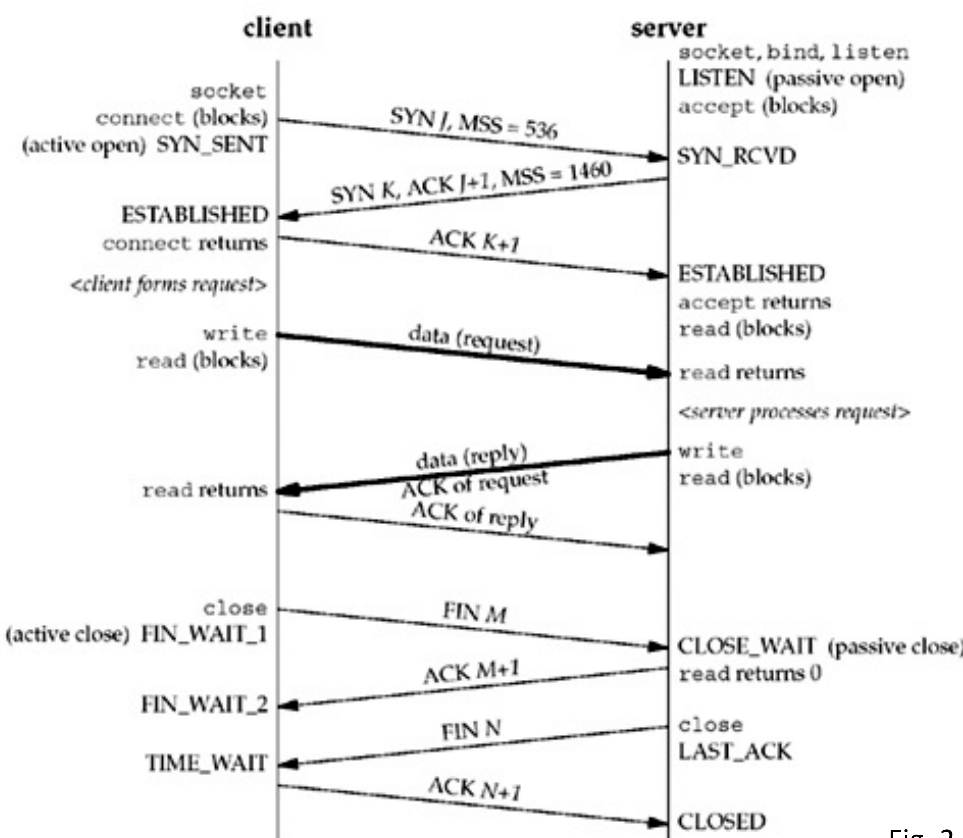


Fig. 2-5

TCP Daytime Client Server Example

- Let’s look at the implementation of a TCP time-of-day client (IPv4)
- Client establishes a TCP connection with server
- Server sends back current time and date
- Client: Figure 1.5 *daytimetcpcli.c*
- Server: Figure 1.9 *daytimetcpsrv.c*

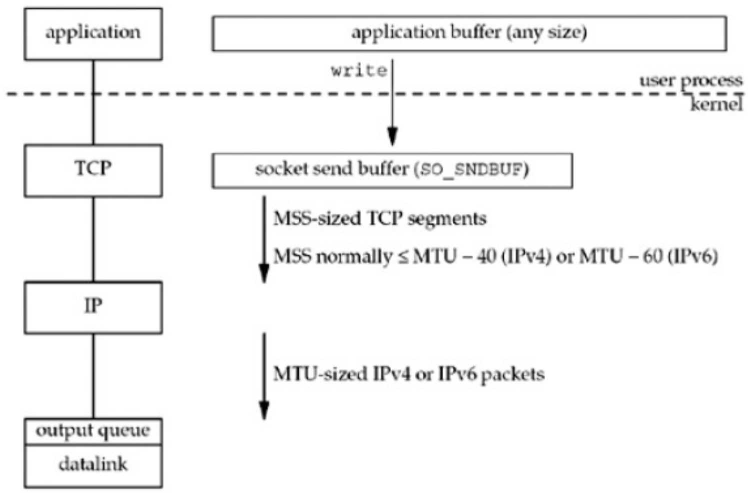
Buffer Sizes and Limitations

- Max size IPv4 datagram 65,535 bytes
- MTU – maximum transmission unit can be dictated by hardware
- Path MTU – the smallest MTU in path between hosts
- MSS – maximum segment size announced by TCP peers as the maximum amount of data sent per segment

Application writes to TCP

- An application writes data to a TCP socket:

Figure 2.15. Steps and buffers involved when an application writes to a TCP socket.



UDP

- User Datagram Protocol

- UDP is a transport protocol
  – Uses ports to provide communication services to individual processes
  – Communication between processes

- UDP uses IP to deliver datagrams to the right host

UDP

- Datagram delivery
- Connectionless
- Unreliable
- Minimal

UDP Datagram

Source Port	Destination Port
Length	Checksum
Data	

Protocol Usage by Application

Application	IP	ICMP	UDP	TCP	SCTP
ping		•			
traceroute		•	•		
OSPF (routing protocol)	•				
RIP (routing protocol)			•		
BGP (routing protocol)				•	
BOOTP (bootstrap protocol)			•		
DHCP (bootstrap protocol)			•		
NTP (time protocol)			•		
TFTP			•		
SNMP (network management)			•		
SMTP (electronic mail)				•	
Telnet (remote login)				•	
SSH (secure remote login)				•	
FTP				•	
HTTP (the Web)				•	
NNTP (network news)				•	
LPR (remote printing)				•	
DNS			•	•	
NFS (network filesystem)			•	•	
Sun RPC			•	•	
DCE RPC			•	•	
IUA (ISDN over IP)					•
M2UA,M3UA (SS7 telephony signaling)					•
H.248 (media gateway control)			•	•	•
H.323 (IP telephony)			•	•	•
SIP (IP telephony)			•	•	•