TCP/IP Fundamentals

OSI Seven Layer Model & Seminar Outline

Layer	Function
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

This seminar will present TCP/IP communications starting from Layer 2 up to Layer 4 (TCP/IP applications cover Layers 5-7)

- IP Addresses
- Data Link Layer
    - Network Frames
    - Address Resolution Protocol
- Network Layer
    - Internet Protocol
    - IP Routing
    - ICMP Error Reporting
- Transport Layer
    - User Datagram Protocol
    - Transmission Control Protocol
- Session through Application Layers
    - Domain Name System
- Final example tracing DNS transaction through a router
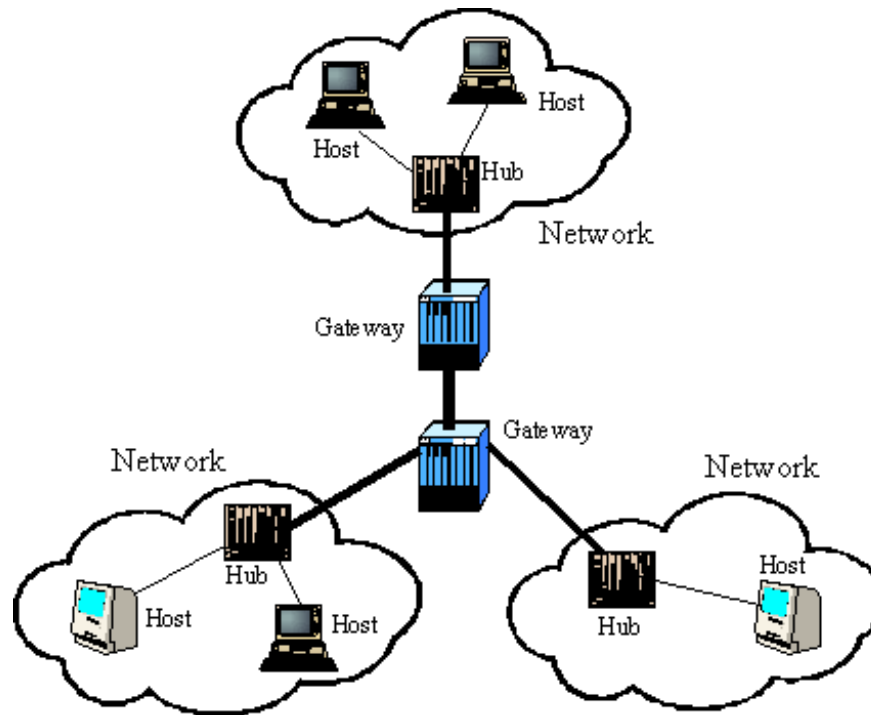
Definitions

- **Physical network** - a collection of computers, communications devices, wiring, etc. that communicate directly with one another (e.g., Ethernet, Token Ring)
- **Host** - A computer, connected to a physical network, that exchanges information with another computer via TCP/IP
- **Gateway** - A computer that interconnects two or more physical networks and that routes TCP/IP information among those networks (accurately referred to as a **router**)

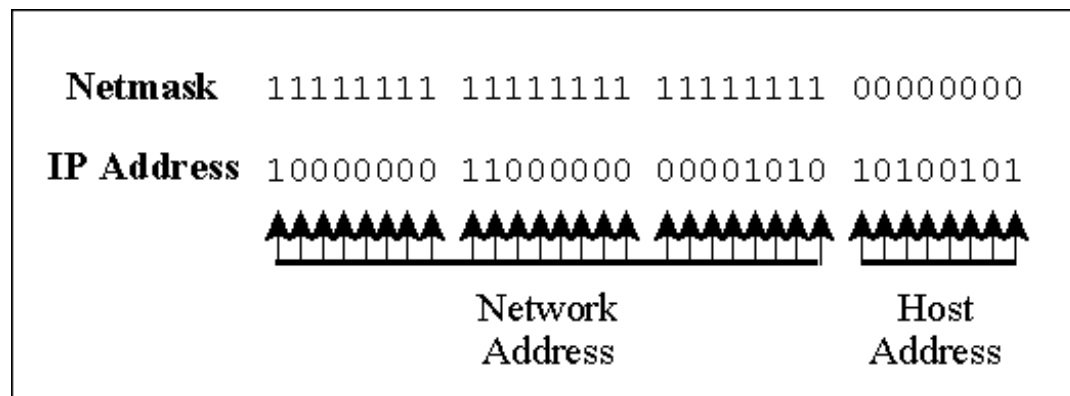## IP Addresses

IP addresses

- are unique, 32-bit addresses
- correspond to connections, not hosts (generally, move connection ==> change IP address)
- are referenced by humans via dotted decimal (or dotted quad) notation, one number per 8 bits (1 octet or byte), e.g., 128.192.6.7
- consist of three primary classes A, B, and C (class D is for multicast) of the form [netid,hostid]

Class formats
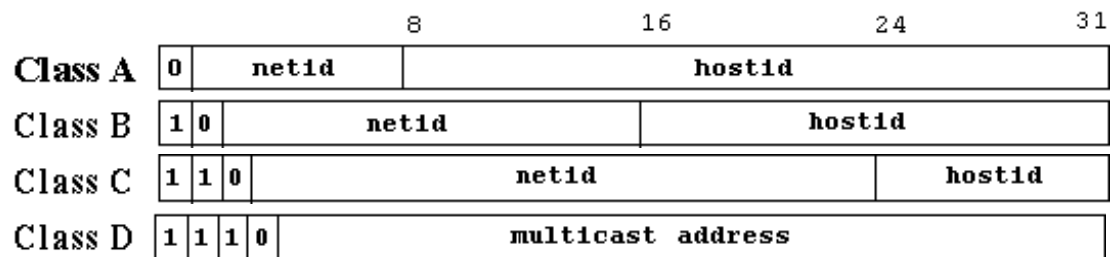
Subnet Mask (netmask)

- 32-bit value
- Generally used to subdivide (subnet) a given IP class network into smaller (sub)networks
- Netmask determines which portion of an IP address is the network address and which is the host address
    - An IP address bit is a **network** address bit if the corresponding netmask bit is 1
    - An IP address bit is a **host** address bit if the corresponding netmask bit is 0
- "Natural netmask" has all netid bit locations = 1 and all hostid bit locations = 0
  (e.g., 255.0.0.0, 255.255.0.0, and 255.255.255.0 for class A, B, and C networks, respectively)
- Netmask example:

Netid and hostid conventions:

- Network addresses have hostid with all bits = 0 (e.g., 128.192.0.0 with netmask=255.255.0.0 and 128.192.6.0 with netmask=255.255.255.0)
- Directed broadcast addresses have hostid with all bits = 1 (e.g., 128.192.255.255 with netmask=255.255.0.0 and 128.192.54.255 with netmask=255.255.255.0)
- "Limited" broadcast has all bits = 1 (e.g. 255.255.255.255)
- Loopback address 127.xxx.yyy.zzz used for internal testing, no traffic generated (typically 127.0.0.1)

IP network ranges by class:

- Class A ==> 1.0.0.0 - 126.0.0.0
- Class B ==> 128.xxx.0.0 - 191.xxx.0.0

- Class C ==> 192.xxx.yyy.0 - 223.xxx.yyy.0
- Class D ==> 224.xxx.yyy.zzz - 239.xxx.yyy.zzz (multicast IP)

Hosts with multiple IP addresses per interface and/or on multiple interfaces are called **multi-homed hosts**.

Data Link Layer

Network Frames

- The basic unit of a physical network is a **frame**
- General form of a network frame

Frame Header	Frame Data	Frame Trailer

- Network frame formats
  - Ethernet frames
    - Version 2 format

```
0               8              16             24             31

PREAMBLE (Bytes 1-4)
PREAMBLE (Bytes 5-8)
DEST MAC ADDRESS (Bytes 1-4)
DEST MAC ADDRESS (Bytes 5-6) | SRC MAC ADDRESS (Bytes 1-2)
SRC MAC ADDRESS (Bytes 3-6)
TYPE*          | DATA (Bytes 1-2)
DATA (Bytes 3-1500)
FRAME CHECK SEQUENCE
```
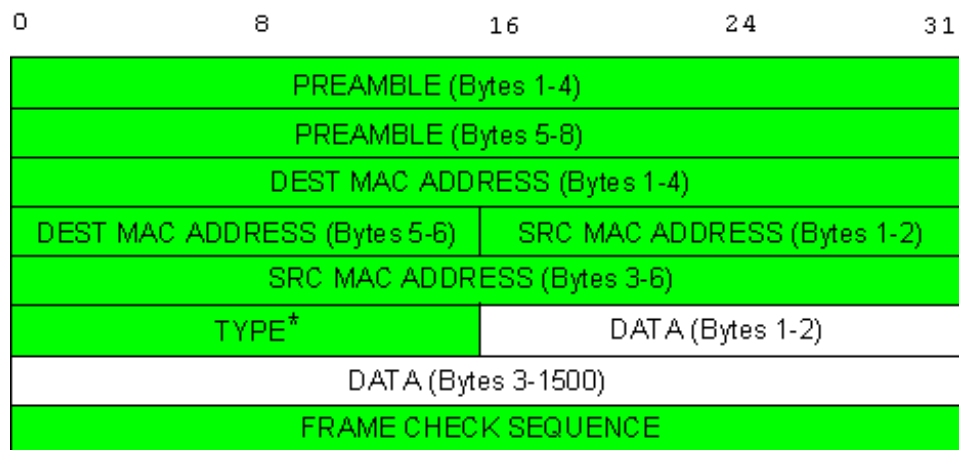
* Click for more information

    - IEEE 802 format

0	8	16	24	31
PREAMBLE (Bytes 1-4)				
PREAMBLE (Bytes 5-8)				
DEST MAC ADDRESS (Bytes 1-4)				
DEST MAC ADDRESS (Bytes 5-6)		SRC MAC ADDRESS (Bytes 1-2)		
SRC MAC ADDRESS (Bytes 3-6)				
LENGTH		DSAP (AA)	SSAP (AA)	
CONTROL (03)	ORGANIZATION CODE (000000)			
TYPE		DATA (Bytes 1-2)		
DATA (Bytes 3-1492)				
FRAME CHECK SEQUENCE				
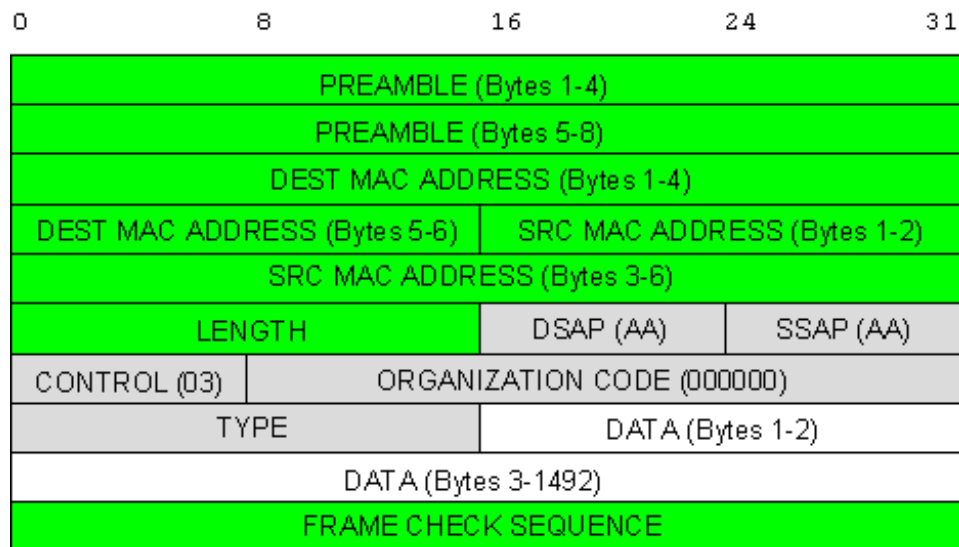
- ❍ Token Ring frames
  - ■ IEEE 802 format

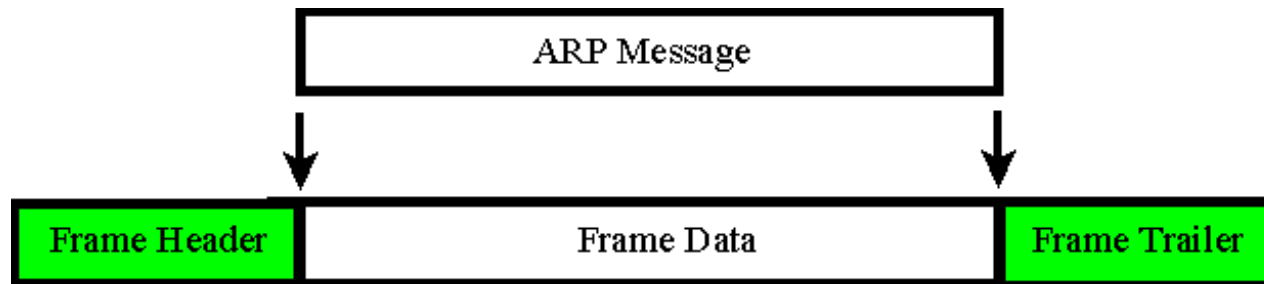| 0	8	16	24	31
SD	AC	FC	DMA (Byte 1)	
DEST MAC ADDRESS (Bytes 2-5)				
DMA (Byte 6)	SRC MAC ADDRESS (Bytes 1-3)			
SRC MAC ADDRESS (Bytes 4-6)			RIF (opt)	
DSAP (AA)	SSAP (AA)	CONTROL(03)	ORG CDE (00)	
ORG CDE (0000)		TYPE		
DATA (Bytes 1-4096)				
FRAME CHECK SEQUENCE				
ED	FS			
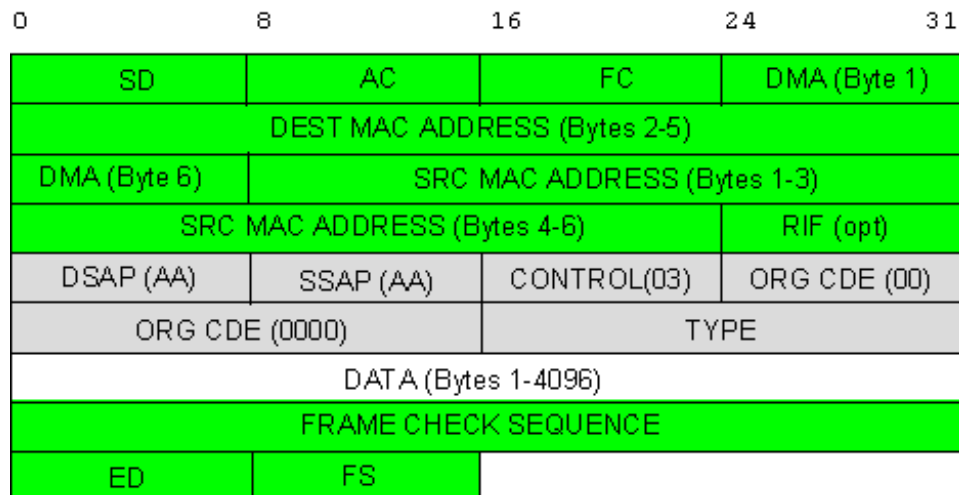
- Every physical network has a maximum frame size called the maximum transfer unit (MTU)
  - ❍ Ethernet MTU - 1500 bytes
  - ❍ Token Ring MTU - 4192 bytes
- Some physical networks have a minimum frame size and must be padded (typically with bytes of "zeroes") to that size when the actual data size is smaller (Ethernet has a minimum frame size of 60 bytes)

Address Resolution Protocol (ARP)

- The address resolution protocol (ARP) is used to associate physical network card addresses (MAC addresses) with IP addresses.
- Encapsulation of ARP message in a physical frame

- ARP message format

- ARP process
  1. ARP requestor sends a broadcast frame with the destination IP address, its source IP address and MAC address, asking for the destination MAC address.
  2. Host with destination IP address sends a directed frame back to ARP requestor filling in its MAC address and storing the MAC address of the sender in an ARP table (or cache).
  3. Optionally, all other hosts within the same broadcast domain add the sender's MAC and IP addresses to its ARP cache.

- Ethernet example:

  Source IP address = 128.192.6.7 (**80 C0 06 07**)
  Source MAC address = **00 00 C0 8D 9C FB**
  Destination IP address = 128.192.6.193 (**80 C0 06 C1**)
  Destination MAC address = **00 00 1D E5 A3 B9**

  1. ARP Request

PREAMBLE	FF FF FF FF FF FF	00 00 C0 8D 9C FB
08 06	HARDWARE TYPE (01)	PROTOCOL TYPE (0800)
HLEN (06)	PLEN (04)	OPERATION (01)
00 00 C0 8D 9C FB		80 C0 06 07
00 00 00 00 00 00		80 C0 06 C1
FRAME CHECK SEQUENCE		

2. ARP Response

| PREAMBLE	00 00 C0 8D 9C FB	00 00 1D E5 A3 B9
08 06	HARDWARE TYPE (01)	PROTOCOL TYPE (0800)
HLEN (06)	PLEN (04)	OPERATION (02)
00 00 1D E5 A3 B9		80 C0 06 C1
00 00 C0 8D 9C FB		80 C0 06 07
FRAME CHECK SEQUENCE		

ARP Table (or Cache)

- To minimize broadcasts due to ARP requests, IP hosts and gateways store tables of MAC and IP addresses (called ARP tables or cashes)
- Each entry in the cache contains (IP address, MAC address, Time to Live)
- Sample ARP cache:

IP address	MAC address	TTL
128.192.6.111	00:00:C0:B8:A5:E3	155s
128.192.10.141	08:00:20:7C:7F:7E	246s
128.192.106.159	00:05:02:E6:48:41	626s
128.192.153.21	00:00:1B:16:F7:FF	332s
128.192.237.52	00:00:0C:4E:60:8F	185s
128.192.26.126	00:00:94:21:66:14	439s

- The Time to Live (TTL) field has either a fixed or adjustable maximum (usually a workstation or router, respectively)
- TTL is set to maximum when ARP request received, not set when communicating with another host
- Tools to display ARP info (TTL usually not displayed):
  - arp (Windows 9x/NT/2000)
 - arp (Unix)

Proxy ARP

- Some devices (such as routers) respond to ARP requests for IP addresses connected to other networks by providing their MAC address in an ARP reply
- This behavior is called proxy (or promiscuous) ARP
- Proxy ARP example:
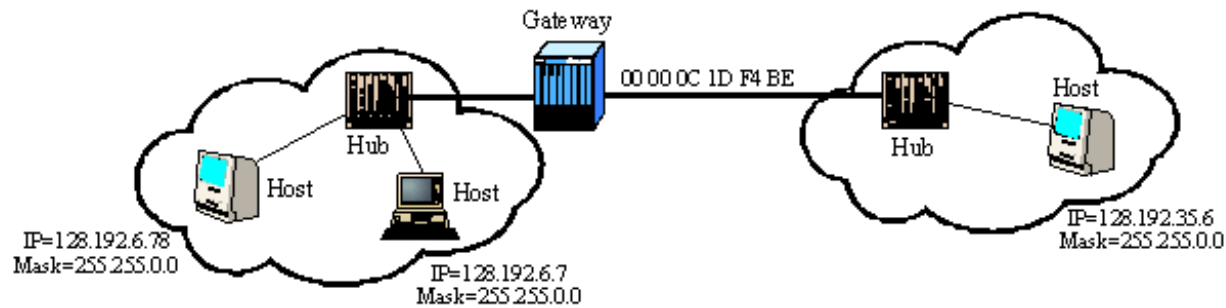
Network Layer

Internet Protocol

- The Internet Protocol (IP) is an:
  - **unreliable** (delivery not guaranteed)
  - **connectionless** (packets independent of one another)
  - **best-effort** (attempt to deliver packets)
  packet delivery mechanism
- Basic unit is the **datagram** (up to 65,535 bytes)
- Encapsulation of IP datagram in a physical frame
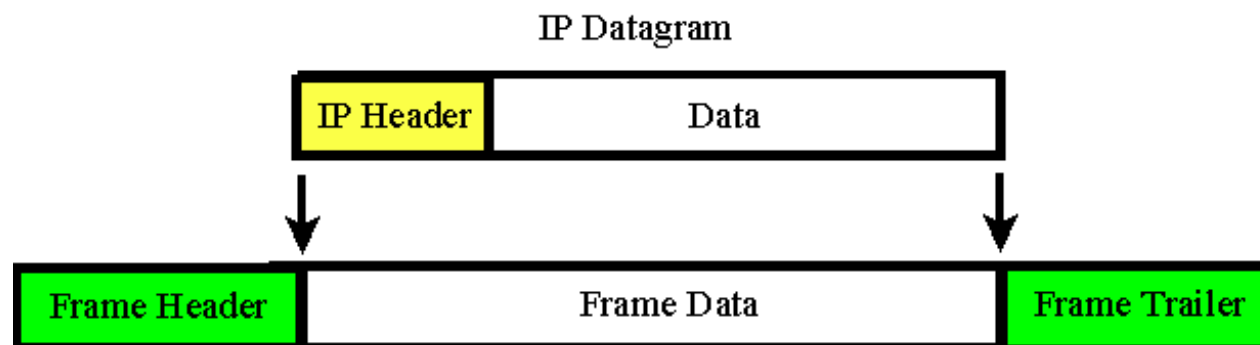
- IP datagram format

```
   0        4        8               16    19       24          31
  +--------+--------+-----------------+---------------------------+
  | VERS   | HLEN   | SERVICE TYPE    |      TOTAL LENGTH         |
  +--------+--------+-----------------+------+--------------------+
  |        IDENTIFICATION             |FLAGS |   FRAGMENT OFFSET  |
  +-----------------+-----------------+------+--------------------+
  |  TIME TO LIVE   |   PROTOCOL      |      HEADER CHECKSUM      |
  +-----------------+-----------------+---------------------------+
  |               SOURCE IP ADDRESS                               |
  +--------------------------------------------------------------+
  |             DESTINATION IP ADDRESS                            |
  +----------------------------------------------+----------------+
  |         IP OPTIONS (IF ANY)                  |    PADDING     |
  +----------------------------------------------+----------------+
  |                       DATA                                    |
  +--------------------------------------------------------------+
  |                       . . . .                                |
  +--------------------------------------------------------------+
```

- Network MTU and fragmentation
  - IP hosts send datagrams up to the MTU size of the physical network
  - Routers *may* have to fragment datagrams if outbound MTU smaller than inbound frame size
    - Each fragment has the format of an IP datagram
    - Fragments reassembled at receiving host (may be inefficient)
    - Higher probability of retransmission --> losing one fragment loses entire datagram
- IP Options
  - **Loose and strict source routing** - used to route a datagram along a specific path
  - **Record route** - used to trace a route
  - **Internet timestamp** - used to record timestamps along the route

IP Routing

- Both hosts and routers participate in routing
- **Direct routing** - transmitting a datagram from one computer directly to another on same physical network
- **Indirect routing** - destination host not on same network --> datagram sent to a router for delivery
- Routing based on IP routing table of the form (**netmask**, **net-address**, **next-hop**)
- Routing algorithm
  1. Extract destination IP address **ipdest** from datagram
  2. Starting at the beginning of the routing table (and for each entry)
     a. Calculate network portion of **ipdest** --> **ipnet** = AND(**netmask**, **ipdest**)
     b. If **ipnet** equals **net-address**, send datagram to **next-hop**
     c. If **ipnet** does not equal **net-address**, repeat steps a. and b. for next entry
     d. If no table entry matches, declare a routing error
- Routing table order:
  1. Directly connected networks (**DCN**)
  2. Host-specific routes (**HSR**)
  3. Net-specific routes (**NSR**)
  4. Default route (**DR**)
- Hosts have minimal routing tables (usually two entries - directly connected network and default route)
- Network for host routing examples
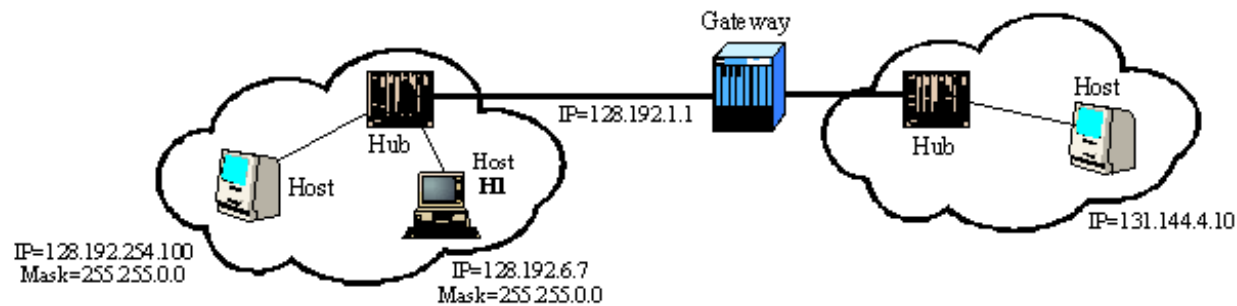
- Routing table for host **H1**

entry	netmask	net-address	next-hop	hop-count	(comments)
1	255.255.0.0	128.192.0.0	128.192.6.7	0	DCN
2	0.0.0.0	0.0.0.0	128.192.1.1	1	DR

- Host routing example 1:

    1. Datagram **ipdest**=128.192.254.100
    2. Calculate network portion of **ipdest** using 1st routing table entry **netmask**:

```
        ipdest  = 10000000 11000000 11111110 01100100 (128.192.254.100)
        netmask = 11111111 11111111 00000000 00000000 (255.255.0.0)
  AND operation   ----------------------------------
        ipnet   = 10000000 11000000 00000000 00000000 (128.192.0.0)
```

    3. Compare **ipnet** and 1st routing table entry **net-address**

```
        128.192.0.0 equals 128.192.0.0
```

    4. Since they match, send it to **next-hop**=128.192.6.7, i.e., communicate directly with destination host

- Host routing example 2:

    1. Datagram **ipdest**=131.144.4.10
    2. Calculate network portion of **ipdest** using 1st routing table entry **netmask**:

```
        ipdest  = 10000011 10010000 00000100 00001010 (131.144.4.10)
        netmask = 11111111 11111111 00000000 00000000 (255.255.0.0)
  AND operation   ----------------------------------
        ipnet   = 10000011 10010000 00000000 00000000 (131.144.0.0)
```

3. Compare **ipnet** with 1st routing table entry **net-address**

```
128.192.0.0 does *not* equal 131.144.0.0
```

4. Since they don't match, calculate network portion of **ipdest** using 2nd routing table entry **netmask**:

```
     ipdest = 10000011 10010000 00000100 00001010 (131.144.4.10)
    netmask = 00000000 00000000 00000000 00000000 (0.0.0.0)
AND operation   ----------------------------------
     ipnet  = 00000000 00000000 00000000 00000000 (0.0.0.0)
```
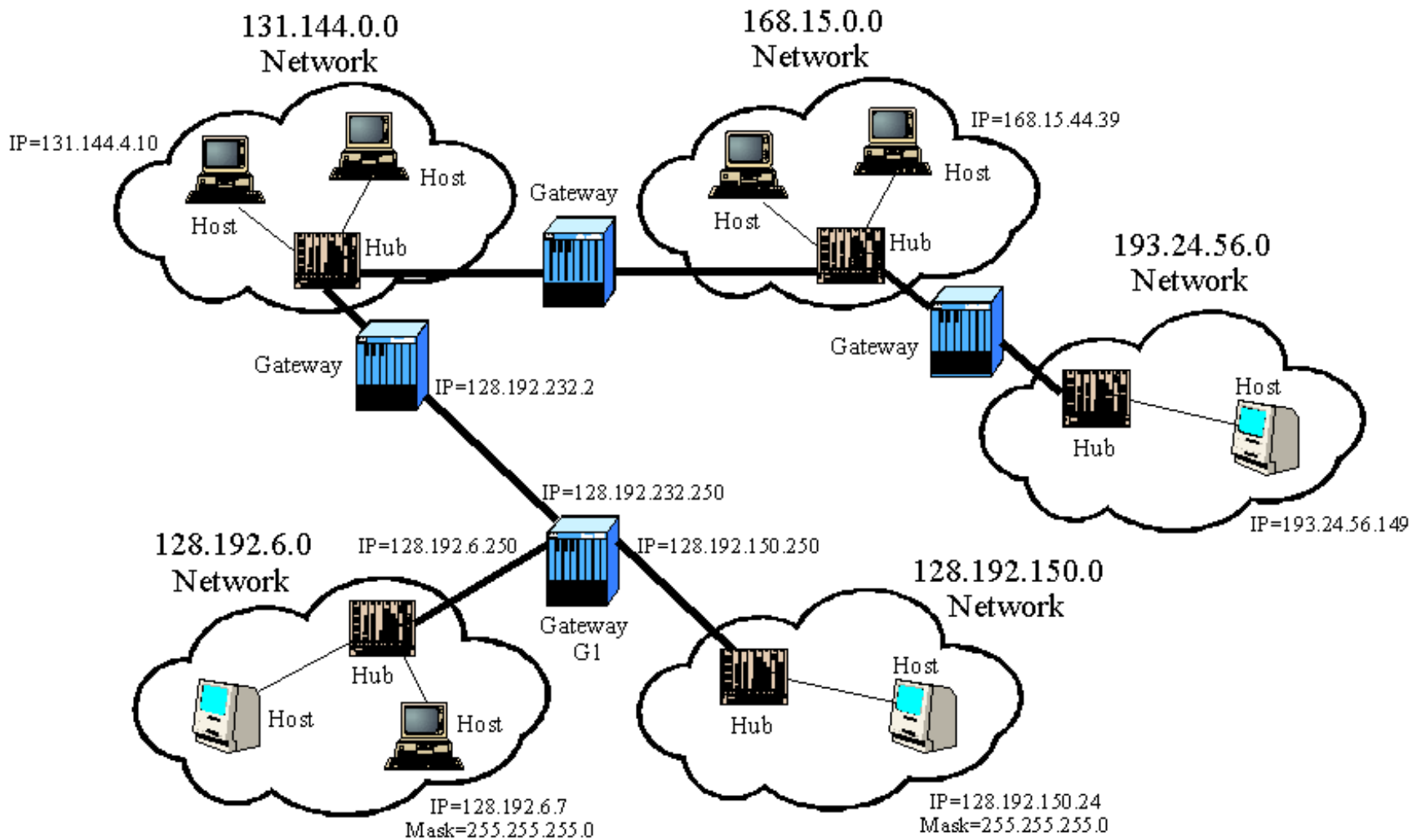
5. Compare **ipnet** with 2nd routing table entry **net-address**

```
0.0.0.0 does equal 0.0.0.0
```

6. Since they match, send it to **next-hop**=128.192.1.1, i.e., the default gateway

- Network for gateway routing examples

- Routing table for gateway **G1**

entry	netmask	net-address	next-hop	hop-count	(comments)
1	255.255.255.0	128.192.6.0	128.192.6.250	0	DCN
2	255.255.255.0	128.192.7.0	128.192.7.250	0	DCN
3	255.255.255.0	128.192.150.0	128.192.150.250	0	DCN
4	255.255.255.0	128.192.232.0	128.192.232.250	0	DCN
5	255.255.255.255	131.144.4.10	128.192.232.2	1	HSR
6	255.255.0.0	168.15.0.0	128.192.232.2	2	NSR
7	0.0.0.0	0.0.0.0	128.192.232.2	1	DR

- Gateway routing example 1:

  1. Datagram **ipdest**=128.192.150.24
  2. Find routing entry in table in which network portion of **ipdest** matches **net-address** and send datagram to corresponding **next-hop**

```
                          (netmask)
 entry     netmask   AND(128.192.150.24)  net-address    result
 -----     -------       -------------    -----------    ------
   1     255.255.255.0   128.192.150.0    128.192.6.0    no match
   2     255.255.255.0   128.192.150.0    128.192.7.0    no match
   3     255.255.255.0   128.192.150.0    128.192.150.0  send directly
```

- Gateway routing example 2:

  1. Datagram **ipdest**=168.15.44.39
  2. Find routing entry in table in which network portion of **ipdest** matches **net-address** and send datagram to corresponding **next-hop**

```
                          (netmask)
 entry     netmask    AND(168.15.44.39)    net-address    result
 -----     -------        ------------     -----------    ------
   1     255.255.255.0    168.15.44.0      128.192.6.0    no match
   2     255.255.255.0    168.15.44.0      128.192.7.0    no match
   3     255.255.255.0    168.15.44.0      128.192.150.0  no match
   4     255.255.255.0    168.15.44.0      128.192.232.0  no match
   5     255.255.255.255  168.15.44.39     131.144.4.10   no match
   6     255.255.0.0      168.15.0.0       168.15.0.0     send to 128.192.232.2
```

- Gateway routing example 3:

  1. Datagram **ipdest**=193.24.56.149
  2. Find routing entry in table in which network portion of **ipdest** matches **net-address** and send datagram to corresponding **next-hop**

```
                          (netmask)
 entry     netmask    AND(193.24.56.149)   net-address    result
 -----     -------        --------------   -----------    ------
   1     255.255.255.0    193.24.56.0      128.192.6.0    no match
   2     255.255.255.0    193.24.56.0      128.192.7.0    no match
   3     255.255.255.0    193.24.56.0      128.192.150.0  no match
   4     255.255.255.0    193.24.56.0      128.192.232.0  no match
   5     255.255.255.255  193.24.56.149    131.144.4.10   no match
   6     255.255.0.0      193.24.0.0       168.15.0.0     no match
   7     0.0.0.0          0.0.0.0          0.0.0.0        send to 128.192.232.2
                                                          (default gateway)
```

Routing Update Protocols

- Routing update protocols are used to modify the routing tables in gateways
- **Autonomous system** - a collection of networks and gateways controlled by a single authority
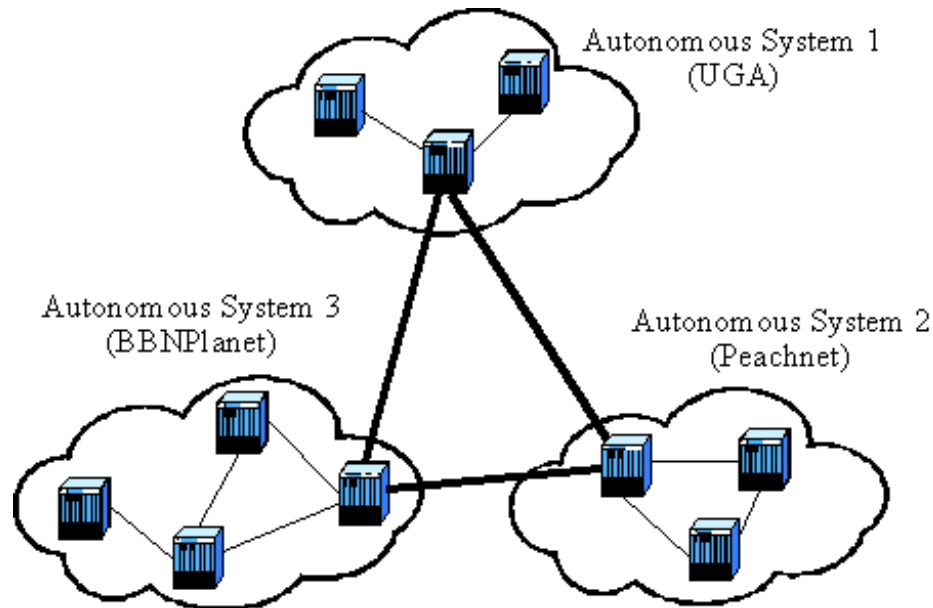

- Two general types of update protocols
  - **Interior gateway protocols** - Used among gateways within an autonomous system
    - Routing Information Protocol (RIP, RIP2)
    - Open Shortest Path First (OSPF)
  - **Exterior gateway protocols** - Used by gateways connecting autonomous systems
    - Exterior Gateway Protocol (EGP)
    - Border Gateway Protocol (BGP)
- OSPF & RIP used at UGA
  - Only Foundry core routers communicate via OSPF
  - Foundry routers broadcast RIP sometimes (don't listen)
  - RIP turned off all other routers
  - Hosts can listen to RIP but must *NOT* broadcast RIP

ICMP Error Reporting

- Since IP networks are inherently unreliable, need a mechanism for reporting IP datagram delivery problems
- Internet Control Message Protocol (ICMP) used to report errors
  - ICMP ***required*** implementation of any TCP/IP software
  - Usually initiated by gateways, but can also be initiated by hosts
  - ICMP messages are sent back to source IP host, not gateways

- ❍ ICMP messages delivered just like IP datagrams
- ❍ Problems delivering ICMP messages do *not* generate additional ICMP messages
- ❍ ICMP does not specify how to handle errors
- ICMP Message Encapsulation
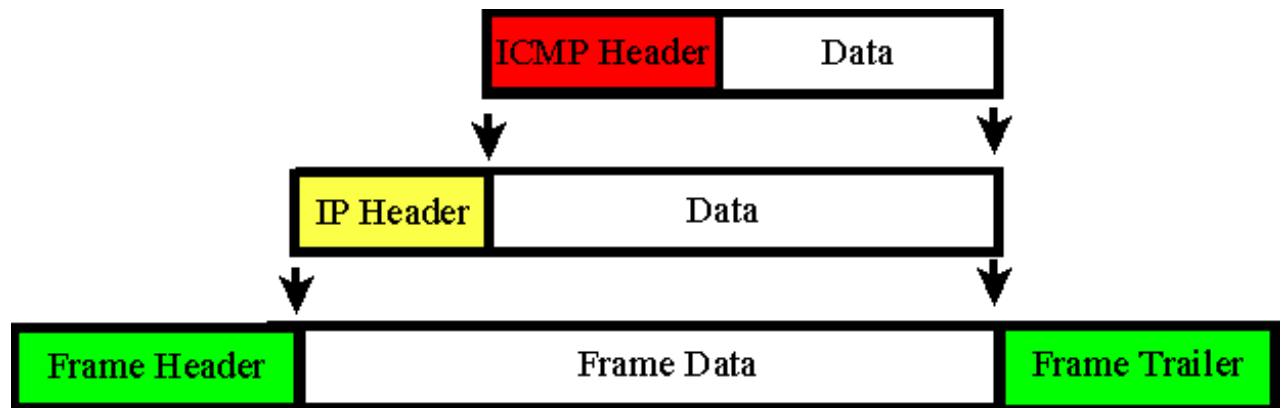
- ICMP Message Format

- **ICMP Echo Request/Reply** - Used to test whether a destination is reachable and responding (e.g., used by **p**acket **i**nter**n**et **g**roper or **ping**)

- **ICMP Destination Unreachable** - Used by a gateway to indicate that it cannot route or deliver an IP datagram

```
  0              8              16             24             31
┌──────────────┬──────────────┬────────────────────────────┐
│  TYPE (3)    │   CODE*      │         CHECKSUM           │
├──────────────┴──────────────┴────────────────────────────┤
│              UNUSED (MUST BE ZERO)                        │
├──────────────────────────────────────────────────────────┤
│        IP HEADER + 1ST 64 BITS OF DATAGRAM               │
├──────────────────────────────────────────────────────────┤
│                      . . . .                             │
└──────────────────────────────────────────────────────────┘
```

* Click for more information

- **ICMP Source Quench** - Used by a gateway to indicate that it is congested, source slows down rate it sends datagrams

```
  0              8              16             24             31
┌──────────────┬──────────────┬────────────────────────────┐
│  TYPE (4)    │  CODE (0)    │         CHECKSUM           │
├──────────────┴──────────────┴────────────────────────────┤
│              UNUSED (MUST BE ZERO)                        │
├──────────────────────────────────────────────────────────┤
│        IP HEADER + 1ST 64 BITS OF DATAGRAM               │
├──────────────────────────────────────────────────────────┤
│                      . . . .                             │
└──────────────────────────────────────────────────────────┘
```

- **ICMP Redirect** - Used by a gateway to tell a directly connected host that a more efficient gateway should be used for a specific IP address

```
  0              8              16             24             31
┌──────────────┬──────────────┬────────────────────────────┐
│  TYPE (5)    │   CODE*      │         CHECKSUM           │
├──────────────┴──────────────┴────────────────────────────┤
│            GATEWAY INTERNET ADDRESS                       │
├──────────────────────────────────────────────────────────┤
│        IP HEADER + 1ST 64 BITS OF DATAGRAM               │
├──────────────────────────────────────────────────────────┤
│                      . . . .                             │
└──────────────────────────────────────────────────────────┘
```

* Click for more information

- **ICMP Time Exceeded** - Used by a gateway to indicate that a routing loop has occurred (sent when the IP TTL value reaches 0 or when segment reassembly time exceeded)

```
  0              8              16             24             31
┌──────────────┬──────────────┬────────────────────────────┐
│  TYPE (11)   │   CODE       │         CHECKSUM           │
├──────────────┴──────────────┴────────────────────────────┤
│              UNUSED (MUST BE ZERO)                        │
├──────────────────────────────────────────────────────────┤
│        IP HEADER + 1ST 64 BITS OF DATAGRAM               │
├──────────────────────────────────────────────────────────┤
│                      . . . .                             │
└──────────────────────────────────────────────────────────┘
```

Transport Layer

- TCP/IP applications utilize client/server technology to exchange information
  - Some hosts can offer a number of TCP/IP services concurrently
  - Some hosts can execute multiple client TCP/IP applications concurrently
  - Some can execute both client and server applications
- When TCP/IP client and server applications want to send data to one another, they need a method of transporting that data to the correct application
- **Protocol ports** are used to determine which application receives the data
  - positive 16-bit integers
  - **well-known ports** assigned by central authority (range 1-2047)
  - User-defined ports (range 2048 or greater)
  - **destination port** - used to specify the application on the destination host to receive the data (usually one of the *well-known port assignments*)
  - **source port** - used to specify where replies on the source host are to be sent (generated by the source host)
- Two methods of transporting data
  - Connectionless, unreliable delivery (User Datagram Protocol)
  - Connection-oriented, reliable delivery (Transmission Control Protocol)

User Datagram Protocol (UDP)

- UDP provides an unreliable, connectionless delivery
- Application programs using UDP are responsible for message loss, duplication, delay, out-of-order delivery, and loss of connectivity
- Examples of applications that use UDP transport include Network Time Protocol (NTP), Sun's Network File System (NFS), and the Simple Network Management Protocol (SNMP)
- Each UDP message is called a **user datagram**
- UDP Message Encapsulation
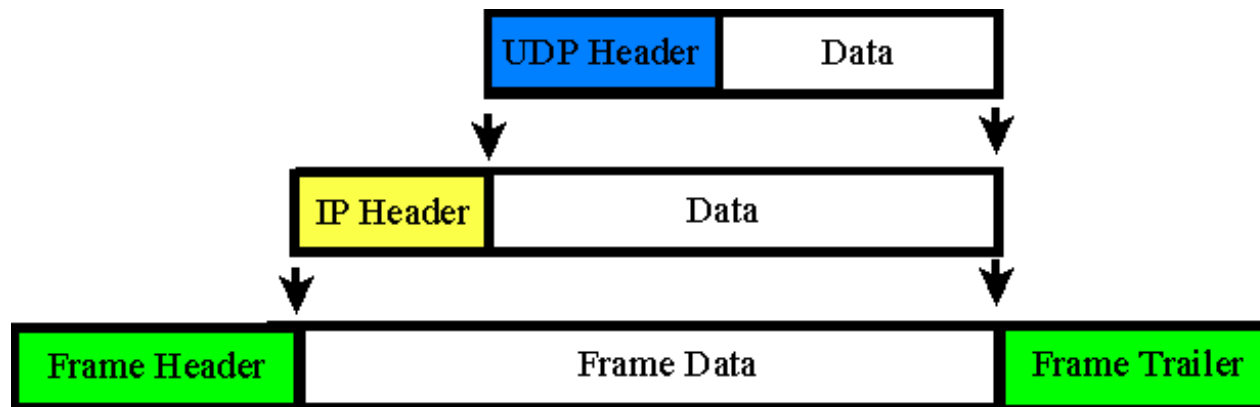
- UDP Message Format

0	16	31

UDP SOURCE PORT	UDP DESTINATION PORT
UDP MESSAGE LENGTH	UDP CHECKSUM
DATA	
. . . .	
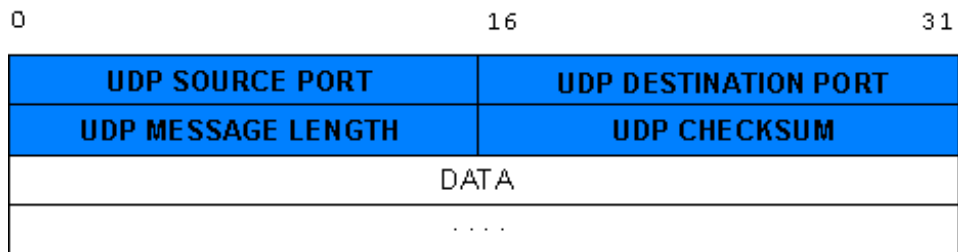
- UDP messages are stored in queues on destination host, one queue per destination port
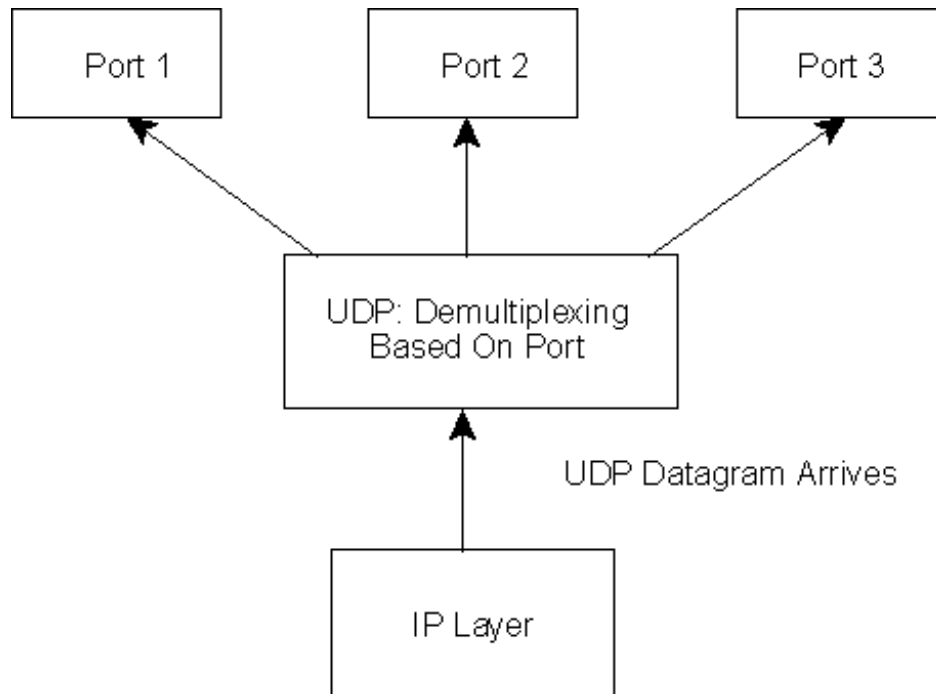
Transmission Control Protocol (TCP)

- TCP provides reliable, end-to-end data transmission with flow control
- Examples of TCP applications include Telnet, FTP, WWW, POP, IMAP, etc.
- Basic features of TCP transmission:
  - **Streamed Data** - Data from sender to receiver organized as a *stream* of bits divided into 8-bit bytes (data streams have no TCP imposed structure)

  - **Connection Oriented**
    - Client host "calls" server host at a specific destination port
    - If receiver accepts call, a connection is established between the corresponding client and server applications

- Information is transferred bi-directionally
- Connection is closed ("call terminated") when client or server application finished or when certain communications errors detected
  - **Buffered Transfer** - Applications send bytes to TCP software that delivers the stream of bytes in exactly the order sent (not necessarily grouped the same way)
  - **Full-duplex Transmission** - Both hosts can send and receive data and control information independently

- The unit of transfer in TCP is called a **segment**
- TCP Segment Encapsulation

- TCP Segment Format

* Click for more information

- Connection defined by the pair of numbers (**source IP**, **source port**) and (**dest IP**, **dest port**)
- Different connections can use the same destination port on server host as long as the source ports or source IPs are different

Client Host 1
128.192.6.7

Source Port
5082 ←
6043 ←

Server Host
128.192.252.2

Dest Port
→ 23
→ 23
↘ 23

Client Host 2
128.192.103.2

Source Port
5082 ↙
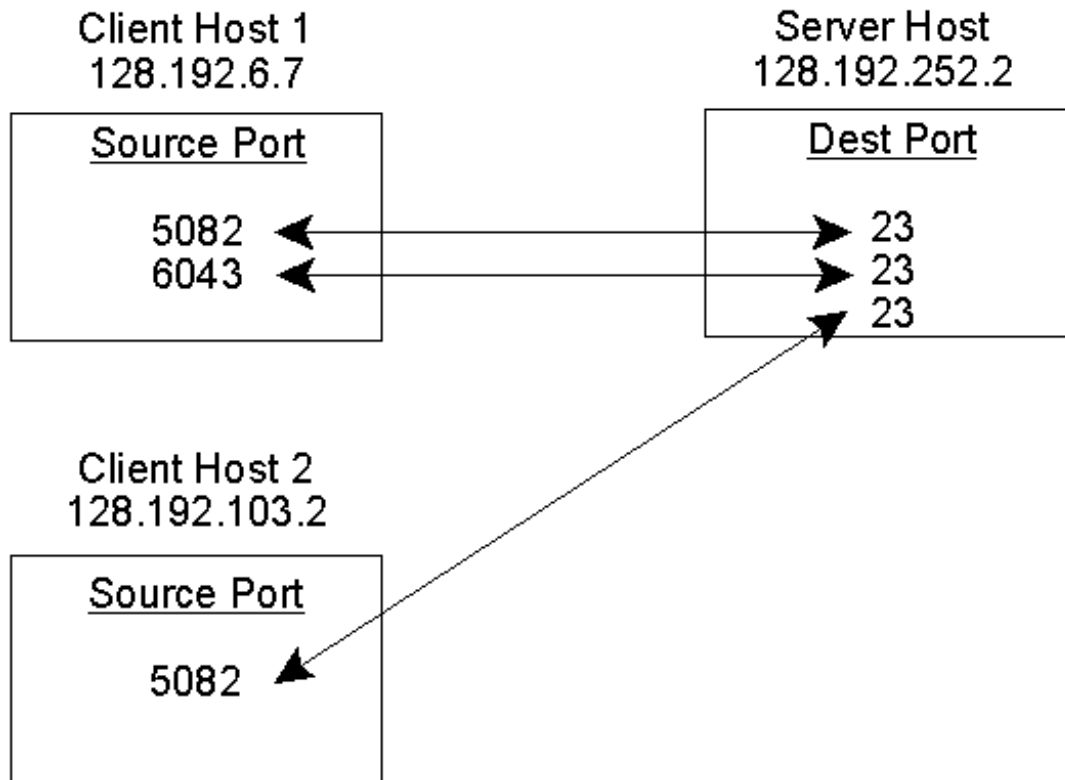
- TCP breaks data stream into segments

ISN

Segment 1			Segment 2					Segment 3				Segment 4		
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

. . .

- Sequence numbers used to place received segment data in the correct order
  - Initial sequence number (**ISN**) marks the beginning of data stream
  - ISN is random and negotiated when connection is established
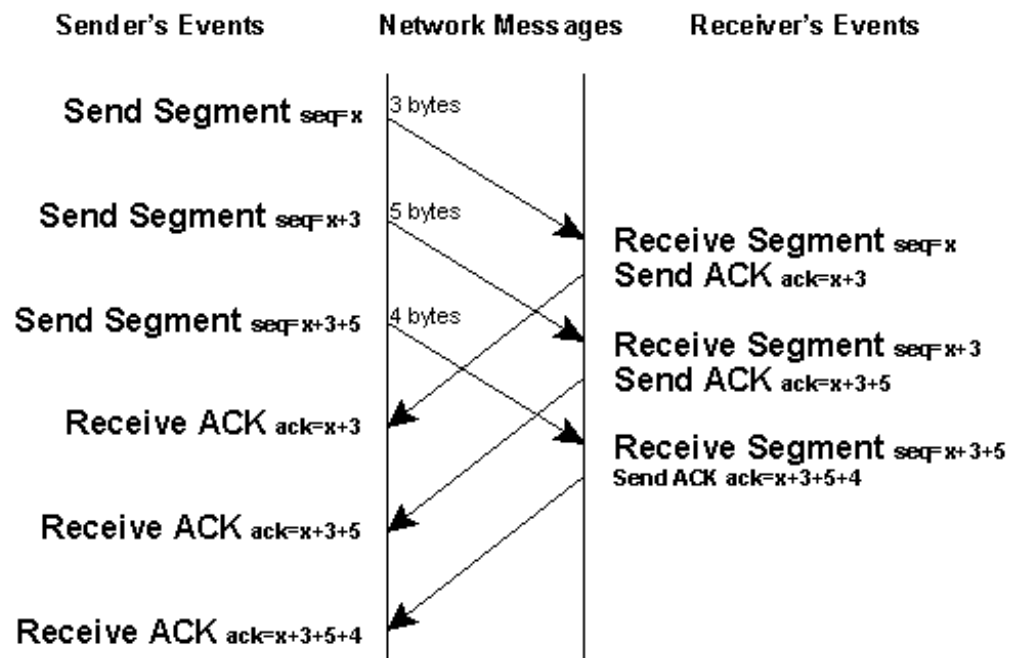- Acknowledgement numbers tell sender that receiver expects *next* segment
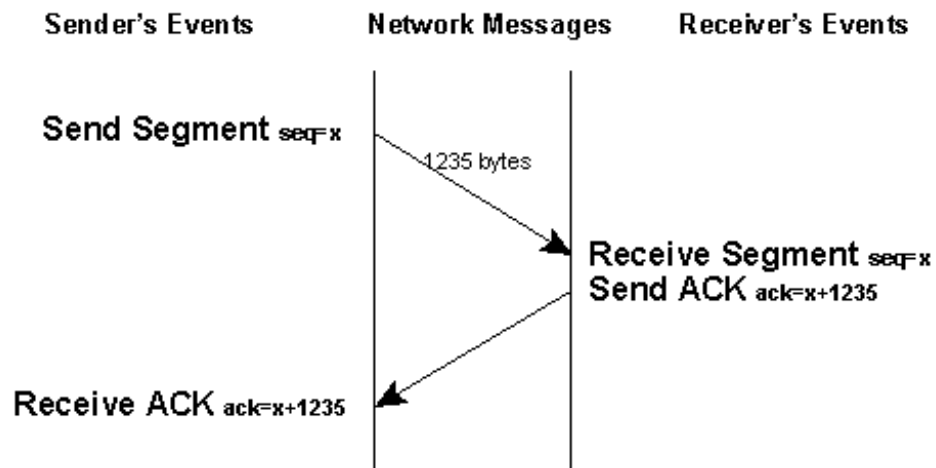
- When a segment is sent, a timer is started; if an ACK has not been received when the timer expires, the segment is resent
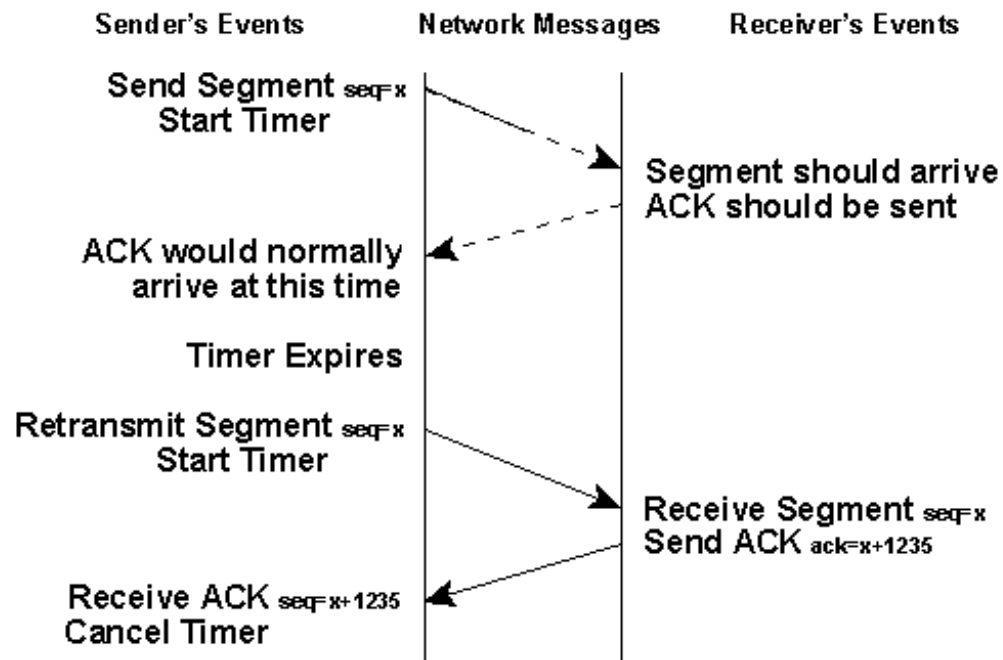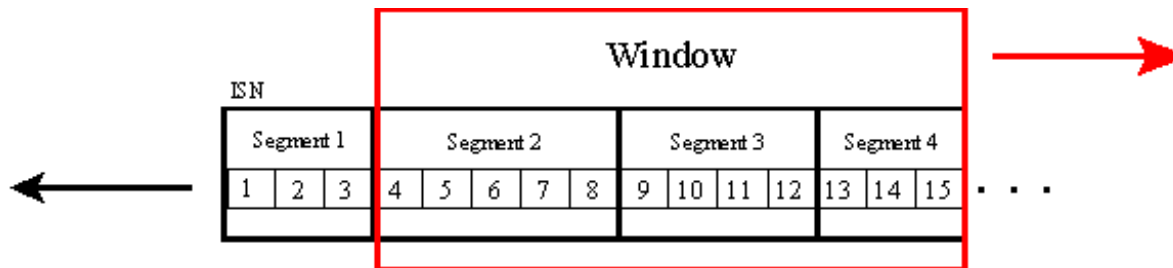

- Sliding windows are used to transmit data stream efficiently and for flow control
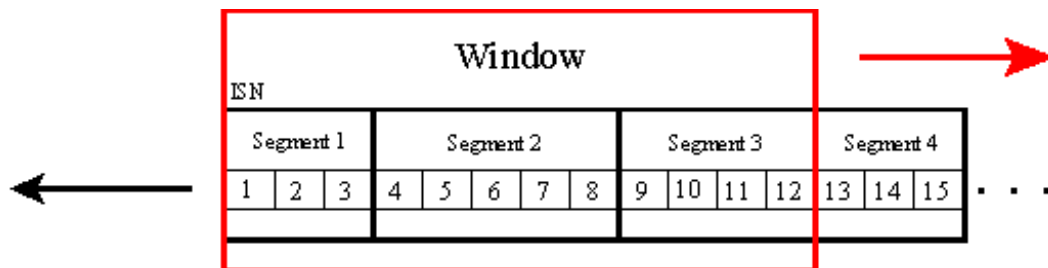
- All segments within the window are sent without waiting for acknowledgement (efficient transmission)

Sender's Events · Network Messages · Receiver's Events

Send Segment seq=x | 3 bytes

Send Segment seq=x+3 | 5 bytes | Receive Segment seq=x
Send ACK ack=x+3

Send Segment seq=x+3+5 | 4 bytes | Receive Segment seq=x+3
Send ACK ack=x+3+5

Receive ACK ack=x+3 | Receive Segment seq=x+3+5
Send ACK ack=x+3+5+4

Receive ACK ack=x+3+5

Receive ACK ack=x+3+5+4

- Window slides as acknowledgements received

ISN

Segment 1 | Segment 2 | Segment 3 | Segment 4

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | . . . |

- Receiver sends acceptable window size to sender during each segment transmission (flow control)
  - if too much data being sent, acceptable window size is reduced
  - if more data can be handled, acceptable window size is increased
- Actually four windows used, i.e., send and receive windows in both directions (full-duplex)
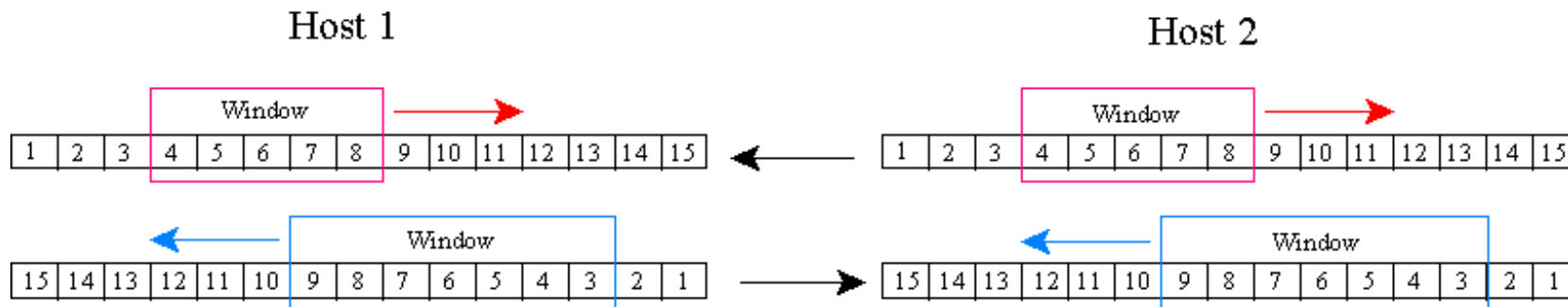
- A TCP connection is established using a "three-way handshake"
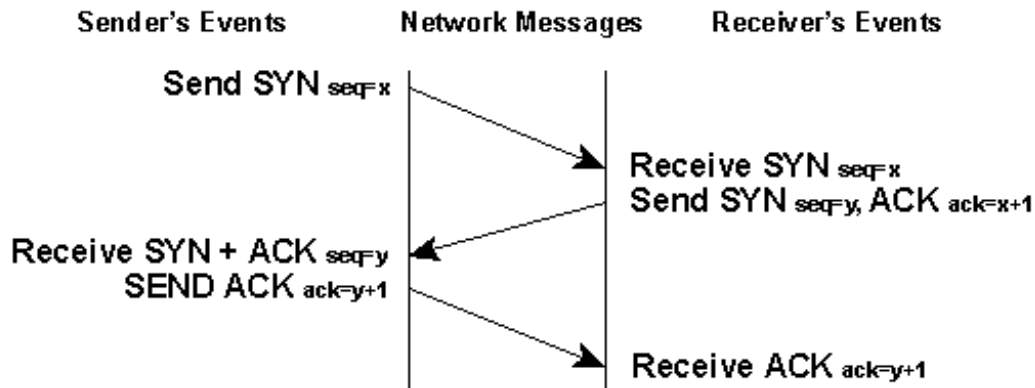
- A TCP connection is closed using a "modified three-way handshake"
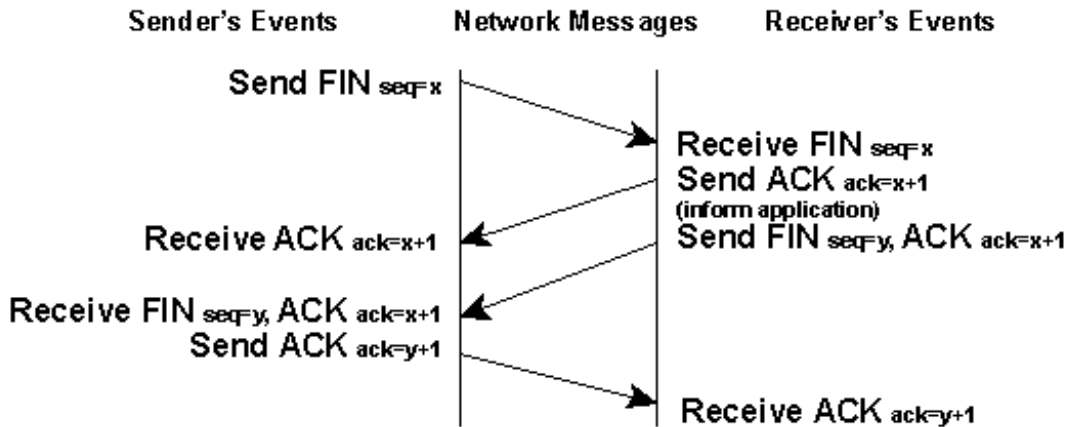
- A TCP connection is aborted via a connection reset (RST bit set in the CODE field)

Session through Application Layers

Domain Name System (DNS)

- For human beings wanting to access Internet resources, names are much easier to remember than IP addresses
- The Domain Name System (DNS) was created to provide a mapping between names for Internet resources and their associated IP addresses
- Characteristics of DNS:
  - Hierarchical naming scheme
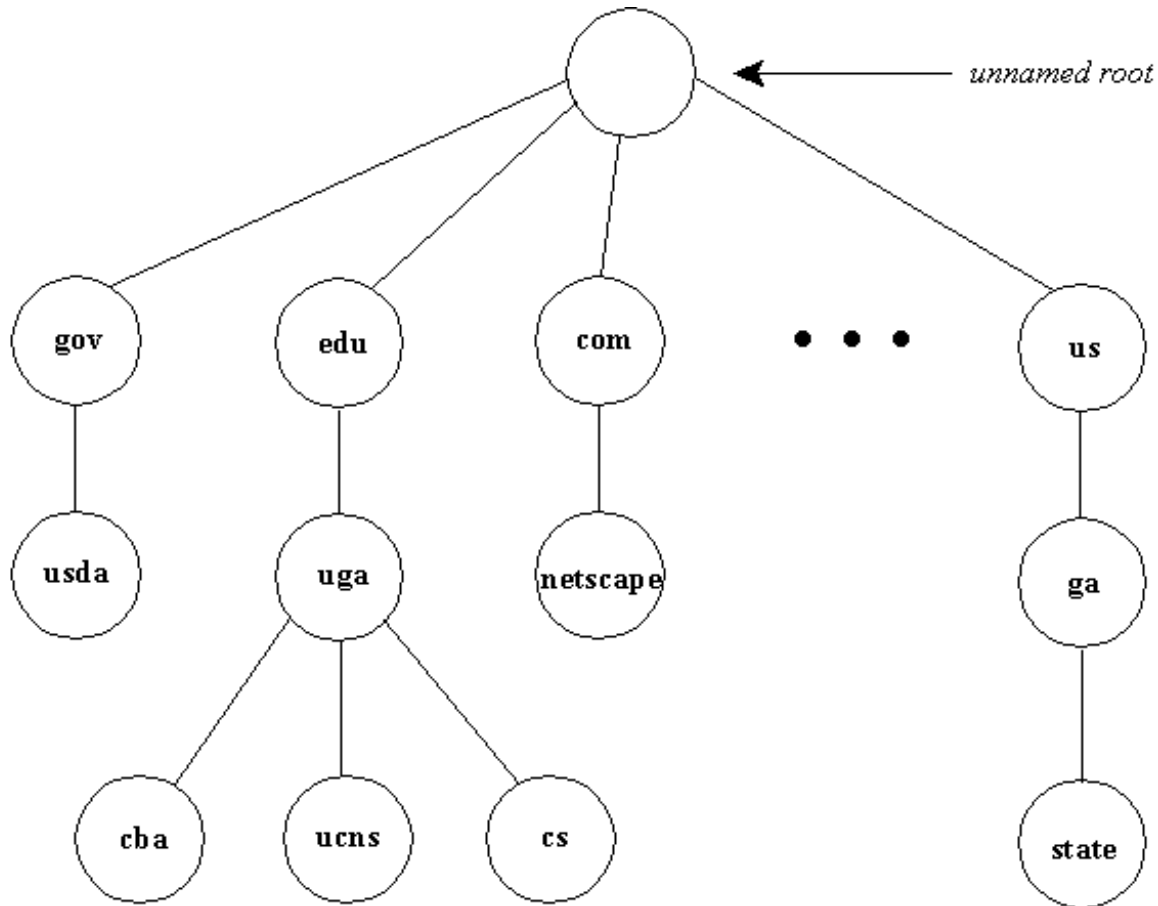
  - Delegation of authority for names
  - Distributed databases of name to IP address (and IP to name) mappings
- Each name authority must operate at least two DNS database servers (name servers) for their authorized domain
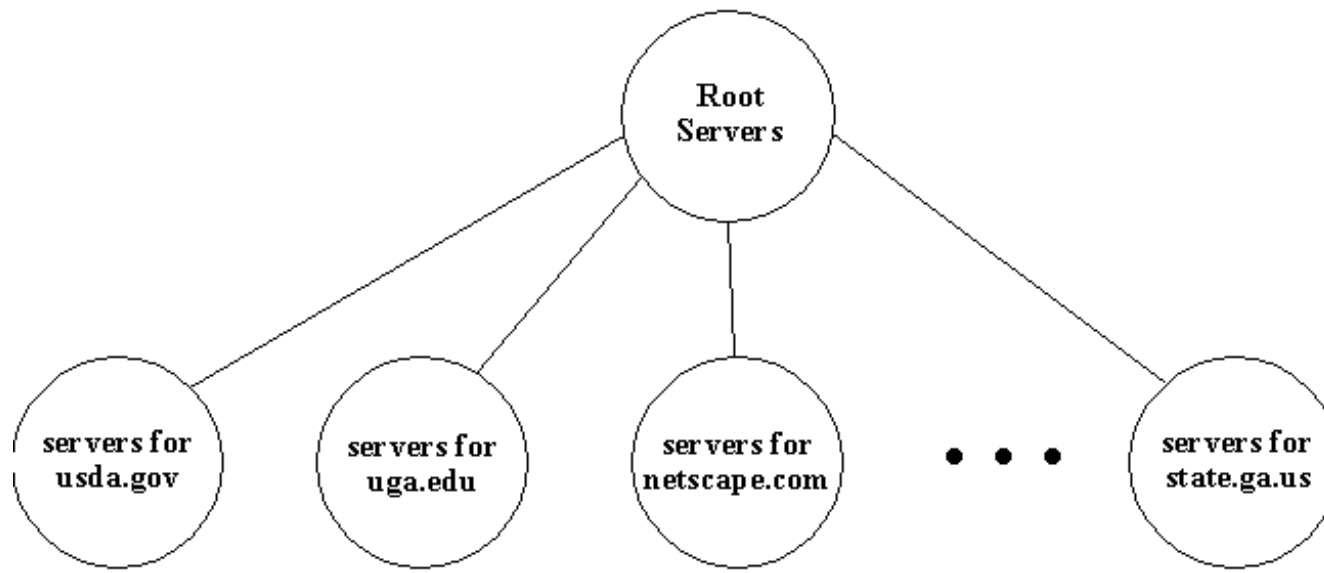
- Every TCP/IP implementation has a software routine called the *name resolver (NR)* to request a DNS lookup from a name server (NS)
- Two types of name resolution:
  - **Recursive resolution** - NR asks NS to resolve names for which it does not have an authoritative answer by querying other name servers (**predominant method**)
  - **Iterative resolution** - NR asks NS to provide the IP address of a NS that can provide an authoritative answer
- Typical name resolution process:
  1. NR receives a domain name from client TCP/IP application, formulates a DNS query and sends it to the first NS in its list
  2. NS determines if it is the authority for the domain name
  3. If so, it looks up the answer and sends an authoritative response back to client's NR
  4. If not, it (typically) queries other name servers for an authoritative response, sends a non-authoritative response to the client's NR and caches the response in case it receives a NR request for the same name
  5. The NR passes the response back to the application program and caches it for a period of time
  6. If the NR does not received a response from the NS in a specified period of time, it sends the query to the next NS in its list

- DNS message format:

```
0                           16                          31
```

IDENTIFICATION	PARAMETER*
NUMBER OF QUESTIONS	NUMBER OF ANSWERS
NUMBER OF AUTHORITY	NUMBER OF ADDITIONAL

QUESTION SECTION*

. . . .

ANSWER SECTION*

. . . .

AUTHORITY SECTION*

. . . .

ADDITIONAL INFORMATION SECTION*

. . . .

* Click for more information

- Pointer queries

  - Some TCP/IP server programs are configured to verify that an IP address has a corresponding domain name
  - IP address to domain name mapping is performed though a pointer query
  - For a given IP address of the form *www.xxx.yyy.zzz*, the format for a pointer query is:

    zzz.yyy.xxx.www.in-addr.arpa

  - NR sends the query (with a type of PTR) to name server
  - If it the authoritative NS for that IP address, it returns the corresponding domain name
  - If not, it queries a root NS for the authoritative NS, queries that NS, and returns the response to the NR

- Domain suffix lists

  - Some name resolvers append a suffix name to the domain name from a *domain suffix list* before formulating a DNS query
  - A example of a domain suffix list is:

    .ucns.uga.edu
    .dev.uga.edu
    .uga.edu
    null

- Some NRs automatically add domain suffixes one at a time to *all* domain names
- Specifying a period (.) at the end of a domain name usually stops the NR from adding suffixes

Final Example

- Establishing an FTP session between FTP client dmm.ucns.uga.edu (128.192.6.7) and FTP server ftp.uga.edu (128.192.252.5)
- Network topology:
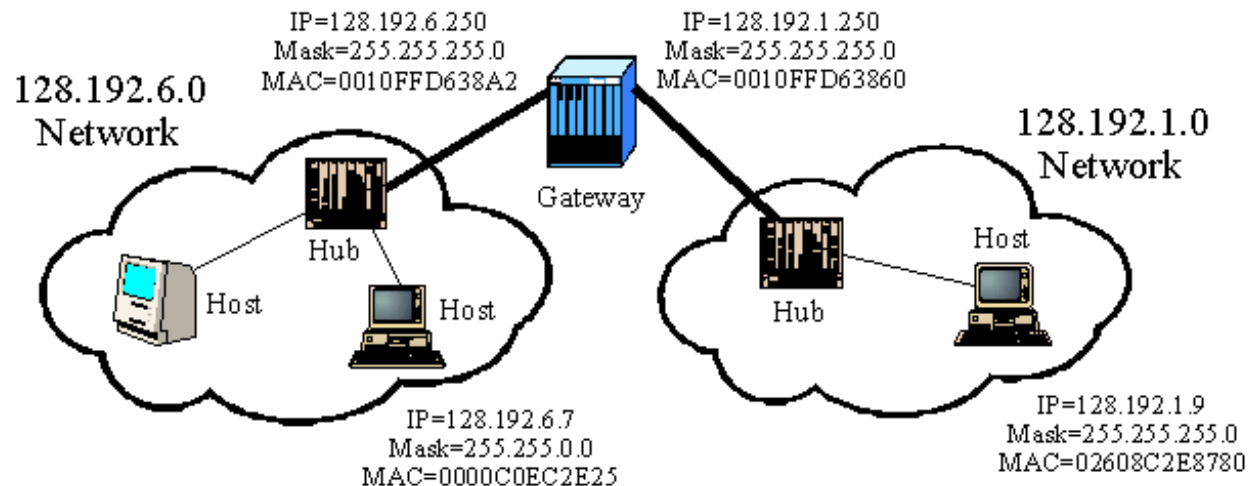
- Process of establishing FTP session:
  1. FTP client program sends domain name ftp.uga.edu to the name resolver (NR)
  2. NR first looks in its DNS cache to see if it has an IP address for ftp.uga.edu
  3. If so, NR passes the IP address back to the FTP client program
  4. If not, NR sends a DNS query to the first name server in its list
     a. NR formulates a DNS query asking for an address record (**A**) for the domain name ftp.uga.edu with an Internet class (**hex 01**)
     b. NR passes DNS query to UDP module along with NS IP address **(128.192.1.9)**

     c. UDP module places UDP header around data with destination protocol port of 53 (domain name service), passing UDP datagram to IP module along with NS IP address

     d. IP module places IP header around UDP datagram with PROTOCOL field set to UDP, determines whether to directly deliver datagram or forward to a

router (**direct in this case**), and passes the IP datagram and appropriate IP delivery address (**128.192.1.9**) to data link (DL) module

IP Header	UDP Header	DNS Query

  e. DL module checks its ARP cache for a MAC address corresponding to the delivery IP address
  f. If found, DL module places an Ethernet frame around IP datagram with destination MAC address (**0010FFD638A2**) from ARP cache and an IP Ethernet type (**hex 0800**) and queues the frame for delivery via the physical layer to the router

Frame Header	IP Header	UDP Header	DNS Query	Frame Trailer

  g. If not found, DL module issues an ARP request and router supplies its MAC address since it is performing proxy ARP functions
  h. When ARP response is received, DL module places an Ethernet frame around IP datagram with destination MAC address (**0010FFD638A2**) from ARP response and an IP Ethernet type (**hex 0800**) and queues the frame for delivery via the physical layer to the router

Frame Header	IP Header	UDP Header	DNS Query	Frame Trailer

5. Router receives frame and forwards to destination network
  a. DL handler of router receives a frame with an IP Ethernet type, removes frame, and passes IP datagram to IP routing module

IP Header	UDP Header	DNS Query

  b. IP routing module determines best route (**direct on interface 128.192.1.250**) and passes IP datagram (after decrementing TTL) to DL module along with IP delivery address (**128.192.1.9**)
  c. DL module checks its ARP cache for a MAC address corresponding to the delivery IP address
  d. If found, DL module places an Ethernet frame around IP datagram with destination MAC address (**02608C2E8780**) from ARP cache and an IP Ethernet type (**hex 0800**) and queues the frame for delivery via the physical layer

Frame Header	IP Header	UDP Header	DNS Query	Frame Trailer

  e. If not found, DL module issues an ARP request and when ARP response is received, DL module places an Ethernet frame around IP datagram with destination MAC address (**02608C2E8780**) from ARP response and an IP Ethernet type (**hex 0800**) and queues the frame for delivery via the physical layer

6. Name server looks up DNS query and sends a DNS response
   a. DL handler of name server receives a frame with an IP Ethernet type, removes frame, and forwards the IP datagram to its IP module

   b. IP module sees UDP in PROTOCOL field, removes the IP header, and passes the datagram to the UDP module

   c. UDP module looks at destination port number **(53)** and places data in the DNS port queue

   d. Name server program (BIND on Unix) reads DNS query from queue, looks up answer in its database, builds the DNS response (two in Answer section -- one CNAME **[ftp.uga.edu=cousteau.uga.edu]**, one A record **[cousteau.uga.edu=128.192.252.5]**; three in Authority section -- one for each name server **[dns1.uga.edu, dns2.uga.edu, dns3.uga.edu]**; and three in Additional Info section -- one A record for each name server **[dns1.uga. edu=128.192.1.9, dns2.uga.edu=128.192.1.193, dns3.uga.edu=168.24.242.249]**), and forwards response along with client IP address **(128.192.6.7)** and source and destination ports (reversed) to UDP module

   e. UDP module places UDP header around DNS response and forwards UDP datagram to IP module along with client IP address

   f. IP module places IP header around UDP datagram with PROTOCOL field set to UDP, determines whether to directly deliver datagram or forward to a router **(router in this case)**, and passes the IP datagram and appropriate IP delivery address **(128.192.1.250)** to data link (DL) module

IP Header	UDP Header	DNS Response

g. DL module checks its ARP cache for a MAC address corresponding to the delivery IP address (which should already be in cache from previous ARP request) and places an Ethernet frame around IP datagram with destination MAC address **(0010FFD63860)** from ARP cache and an IP Ethernet type **(hex 0800)** and queues the frame for delivery via the physical layer to the router

Frame Header	IP Header	UDP Header	DNS Response	Frame Trailer

7. Router receives frame and forwards to destination network
   a. DL handler of router receives a frame with an IP Ethernet type, removes frame, and passes IP datagram to IP routing module

IP Header	UDP Header	DNS Response

   b. IP routing module determines best route **(direct on interface 128.192.6.250)** and passes IP datagram (after decrementing TTL) to DL module along with IP delivery address **(128.192.6.7)**
   c. DL module checks its ARP cache for a MAC address corresponding to the delivery IP address
   d. If found, DL module places an Ethernet frame around IP datagram with destination MAC address **(0000C0EC2E25)** from ARP cache and an IP Ethernet type (hex 0800) and queues the frame for delivery via the physical layer

Frame Header	IP Header	UDP Header	DNS Response	Frame Trailer

   e. If not found, DL module issues an ARP request and when ARP response is received, DL module places an Ethernet frame around IP datagram with destination MAC address **(0000C0EC2E25)** from ARP response and an IP Ethernet type **(hex 0800)** and queues the frame for delivery via the physical layer

Frame Header	IP Header	UDP Header	DNS Response	Frame Trailer

8. Client host processes frame containing DNS response
   a. DL handler of client host receives a frame with an IP Ethernet type, removes frame, and forwards the IP datagram to its IP module

b. IP module sees UDP in PROTOCOL field, removes the IP header, and passes the datagram to the UDP module

c. UDP module looks at source and destination port numbers and passes data to name resolver (NR)

d. NR caches DNS response (domain name and IP address pair) and passes that information back to FTP client program

9. Client FTP program initiates three-way handshake to establish FTP session with destination host