Chapter 8 Communication Networks and Services

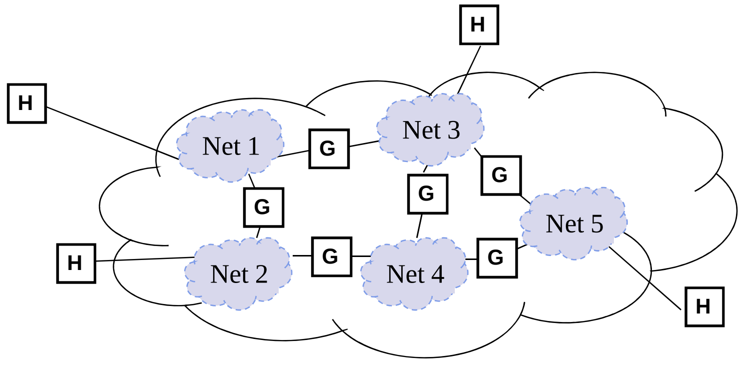
The TCP/IP Architecture
The Internet Protocol
IPv6
Transport Layer Protocols
Internet Routing Protocols
Multicast Routing
DHCP, NAT, and Mobile IP

Chapter 8 Communication Networks and Services

The TCP/IP Architecture

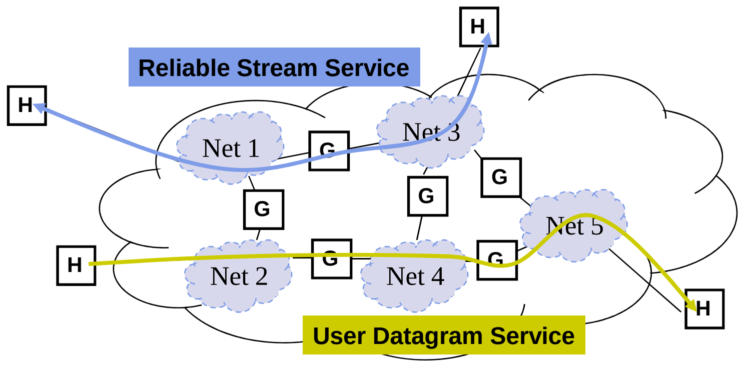
Why Internetworking?

- To build a “network of networks” or internet
  - operating over multiple, coexisting, different network technologies
  - providing ubiquitous connectivity through IP packet transfer
  - achieving huge economies of scale



Why Internetworking?

- To provide *universal communication services*
  - independent of underlying network technologies
  - providing common interface to user applications



Why Internetworking?

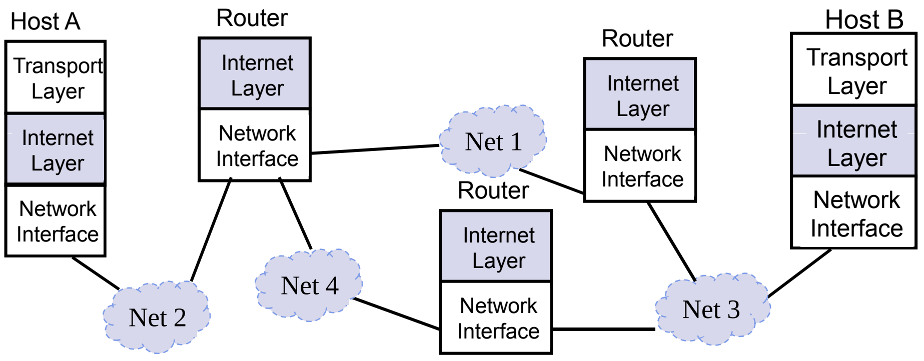
- To provide *distributed applications*
  - Any application designed to operate based on Internet communication services immediately operates across the entire Internet
  - Rapid deployment of new applications
    - Email, WWW, Peer-to-peer
  - Applications independent of network technology
    - New networks can be introduced below
    - Old network technologies can be retired

Internet Protocol Approach

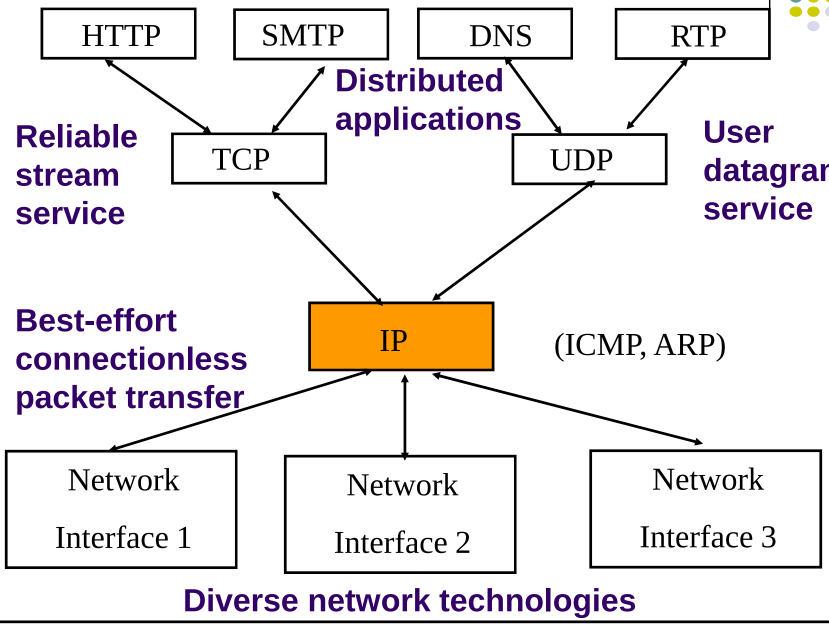
- IP packets transfer information across Internet

  Host A IP → router→ router…→ router→ Host B IP
- IP layer in each router determines next hop (router)

  Routing + encapsulation + segmentation/assembly
- Network interfaces transfer IP packets across networks



TCP/IP Protocol Suite



Internet Names & Addresses

Internet Names

- Each host has a unique name
  - Independent of physical location
  - Facilitate memorization by humans
  - Domain Name
  - Organization under single administrative unit
- Host Name
  - Name given to host computer
- User Name
  - Name assigned to user

leongarcia@comm.utoronto.ca

Internet Addresses

- Each host has globally unique *logical* 32 bit IP address
- Separate address for each physical connection to a network
- Routing decision is done based on destination IP address
- IP address has two parts:
  - *netid* and *hostid*
  - *netid* unique
  - *netid* facilitates routing
- Dotted Decimal Notation:

  int1.int2.int3.int4

  (intj = jth octet)

  128.100.10.13

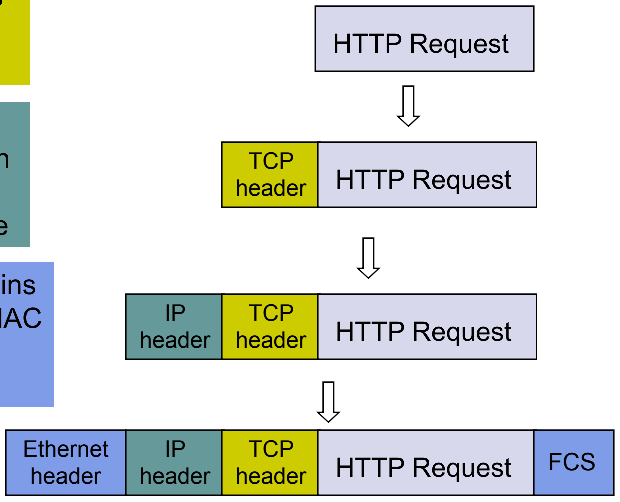
DNS resolves IP name to IP address

Encapsulation

TCP Header contains source & destination port numbers

IP Header contains source and destination IP addresses; transport protocol type

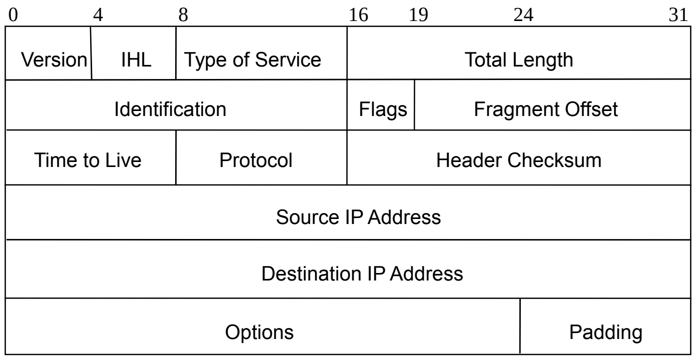
Ethernet Header contains source & destination MAC addresses; network protocol type



Internet Protocol

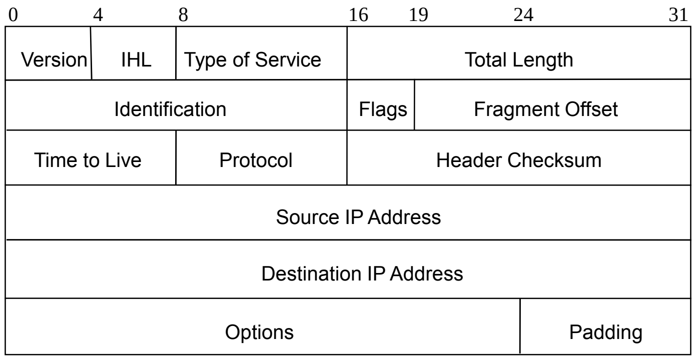
- Provides best effort, connectionless packet delivery
  - motivated by need to keep routers simple and by adaptibility to failure of network elements
  - packets may be lost, out of order, or even duplicated
  - higher layer protocols must deal with these, if necessary
- RFCs 791, 950, 919, 922, and 2474.
- IP is part of Internet STD number 5, which also includes:
  - Internet Control Message Protocol (ICMP), RFC 792
  - Internet Group Management Protocol (IGMP), RFC 1112

IP Packet Header



- Minimum 20 bytes
- Up to 40 bytes in options fields

IP Packet Header

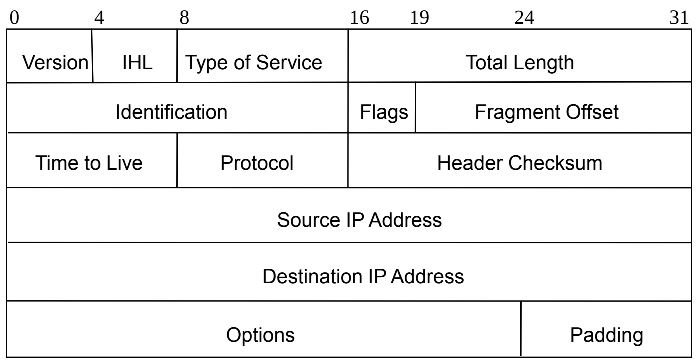


Version: current IP version is 4.

Internet header length (IHL): length of the header in 32-bit words.

Type of service (TOS): traditionally priority of packet at each router. Recent Differentiated Services redefines TOS field to include other services besides best effort.

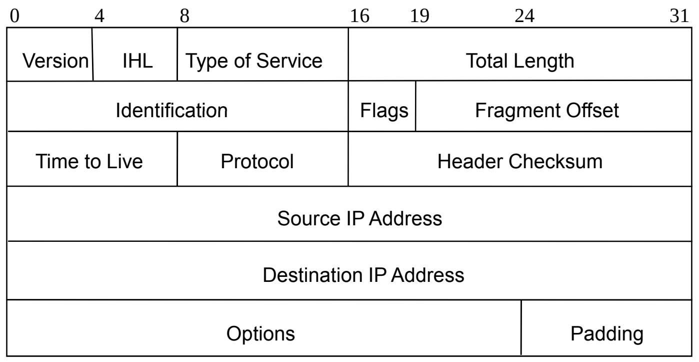
IP Packet Header



Total length: number of bytes of the IP packet including header and data, maximum length is 65535 bytes.

Identification, Flags, and Fragment Offset: used for fragmentation and reassembly (More on this shortly).

IP Packet Header



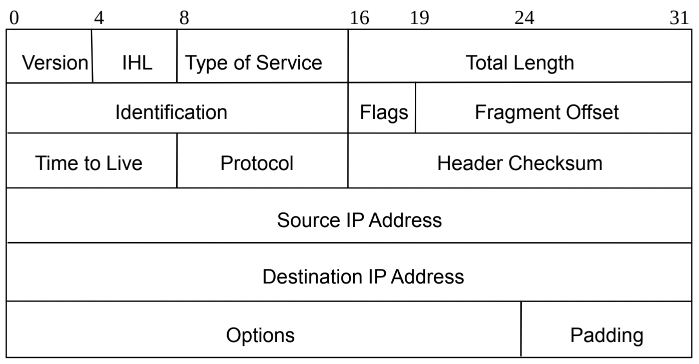
Time to live (TTL): number of hops packet is allowed to traverse in the network.

- Each router along the path to the destination decrements this value by one.
- If the value reaches zero before the packet reaches the destination, the router discards the packet and sends an error message back to the source.

Why not use actual time in TTL?

Unpredictable; very large #;
more complex to track and update

IP Packet Header

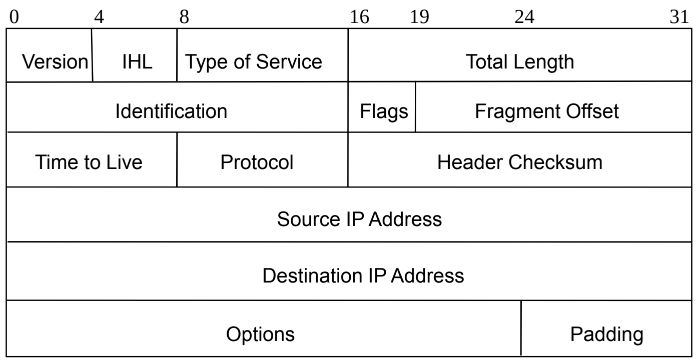


Protocol: specifies upper-layer protocol that is to receive IP data at the destination. Examples include TCP (protocol = 6), UDP (protocol = 17), and ICMP (protocol = 1).

Header checksum: verifies the integrity of the IP header.

Source IP address and **destination IP address:** contain the addresses of the source and destination hosts.

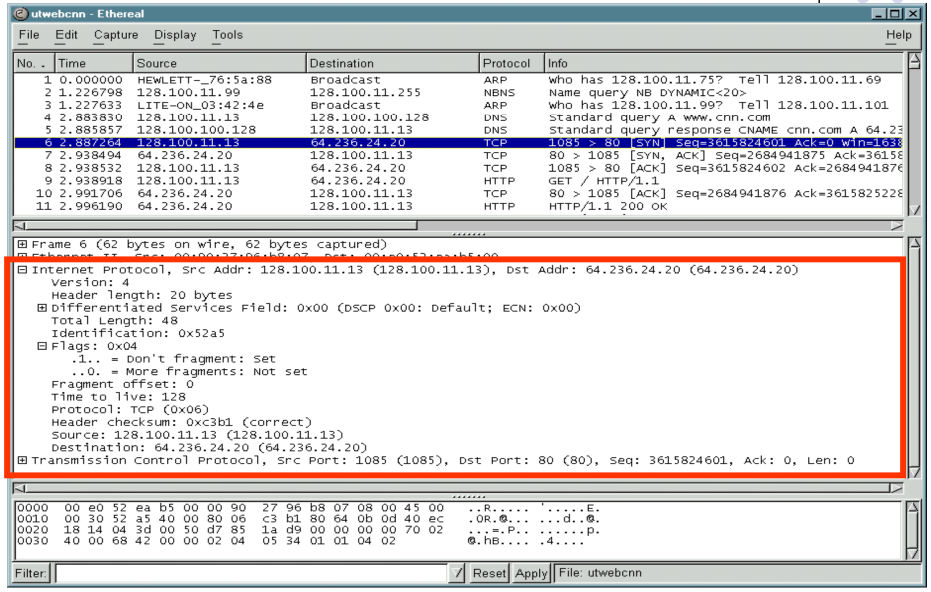
IP Packet Header



Options: Variable length field, allows packet to request special features such as security level, route to be taken by the packet, and timestamp at each router. Detailed descriptions of these options can be found in [RFC 791].

Padding: This field is used to make the header a multiple of 32-bit words.

Example of IP Header


utwebcnn - Ethereal
File Edit Capture Display Tools Help
Filter: Reset Apply File: utwebcnn

```
No. .  Time      Source              Destination        Protocol  Info
  1 0.000000  HEWLETT-_76:5a:88   Broadcast          ARP       Who has 128.100.11.75?  Tell 128.100.11.69
  2 1.226798  128.100.11.99       128.100.11.255     NBNS      Name query NB DYNAMIC<20>
  3 1.227633  LITE-ON_03:42:4e    Broadcast          ARP       Who has 128.100.11.99?  Tell 128.100.11.101
  4 2.883830  128.100.11.13       128.100.100.128    DNS       Standard query A www.cnn.com
  5 2.885857  128.100.100.128     128.100.11.13      DNS       Standard query response CNAME cnn.com A 64.23
  6 2.887264  128.100.11.13       64.236.24.20       TCP       1085 > 80 [SYN] Seq=3615824601 Ack=0 Win=1638
  7 2.938494  64.236.24.20        128.100.11.13      TCP       80 > 1085 [SYN, ACK] Seq=2684941875 Ack=36158
  8 2.938532  128.100.11.13       64.236.24.20       TCP       1085 > 80 [ACK] Seq=3615824602 Ack=268494187
  9 2.938918  128.100.11.13       64.236.24.20       HTTP      GET / HTTP/1.1
 10 2.991706  64.236.24.20        128.100.11.13      TCP       80 > 1085 [ACK] Seq=2684941876 Ack=361582522
 11 2.996190  64.236.24.20        128.100.11.13      HTTP      HTTP/1.1 200 OK
```

```
⊞ Frame 6 (62 bytes on wire, 62 bytes captured)
⊟ Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 64.236.24.20 (64.236.24.20)
    Version: 4
    Header length: 20 bytes
  ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 48
    Identification: 0x52a5
  ⊟ Flags: 0x04
      .1.. = Don't fragment: Set
      ..0. = More fragments: Not set
    Fragment offset: 0
    Time to live: 128
    Protocol: TCP (0x06)
    Header checksum: 0xc3b1 (correct)
    Source: 128.100.11.13 (128.100.11.13)
    Destination: 64.236.24.20 (64.236.24.20)
⊞ Transmission Control Protocol, Src Port: 1085 (1085), Dst Port: 80 (80), Seq: 3615824601, Ack: 0, Len: 0
```

```
0000   00 e0 52 ea b5 00 00 90  27 96 b8 07 08 00 45 00   ..R..... '.....E.
0010   00 30 52 a5 40 00 80 06  c3 b1 80 64 0b 0d 40 ec   .0R.@... ...d..@.
0020   18 14 04 3d 00 50 d7 85  1a d9 00 00 00 00 70 02   ...=.P.. ......p.
0030   40 00 68 42 00 00 02 04  05 34 01 01 04 02         @.hB.... .4....
```

Header Checksum

- IP header uses check bits to detect errors in the ***header***
- A checksum is calculated for header contents
- Checksum recalculated at every router, so algorithm selected for ease of implementation in software
- Let header consist of L, 16-bit words,

  $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{L-1}$
- The algorithm appends a 16-bit *checksum* $\mathbf{b}_L$

Checksum Calculation

The checksum $\mathbf{b}_L$ is calculated as follows:

- Treating each 16-bit word as an integer, find

  $\mathbf{x} = \mathbf{b}_0 + \mathbf{b}_1 + \mathbf{b}_2 + \ldots + \mathbf{b}_{L-1}$ modulo $2^{15}-1$
- The checksum is then given by:

  $\mathbf{b}_L = -\mathbf{x}$ modulo $2^{15}-1$
- This is the 16-bit 1's complement sum of the **b's**
- If checksum is 0, use all 1's representation (all zeros reserved to indicate checksum was not calculated)
- *Thus, the headers must satisfy the following* ***pattern***:

  $\mathbf{0} = \mathbf{b}_0 + \mathbf{b}_1 + \mathbf{b}_2 + \ldots + \mathbf{b}_{L-1} + \mathbf{b}_L$ modulo $2^{15}-1$

IP Header Processing

1. Compute header checksum for correctness and check that fields in header (e.g. version and total length) contain valid values
2. Consult routing table to determine next hop
3. Change fields that require updating (TTL, header checksum)

Header checksum vs. body checksum

IP Addressing

- RFC 1166
- Each host on Internet has unique 32 bit IP address
- Each address has two parts: *netid* and *hostid*
- *netid* unique & administered by
  - American Registry for Internet Numbers (ARIN)
  - Reseaux IP Europeens (RIPE)
  - Asia Pacific Network Information Centre (APNIC)
- Facilitates routing
- A separate address is required for each physical connection of a host to a network; "multi-homed" hosts
- Dotted-Decimal Notation:

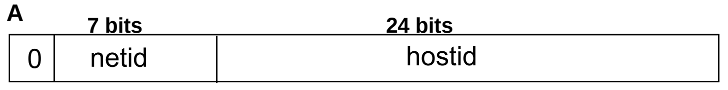
  int1.int2.int3.int4 where intj = integer value of jth octet

  IP address of 10000000 10000111 01000100 00000101

  is 128.135.68.5 in dotted-decimal notation

Classful Addresses

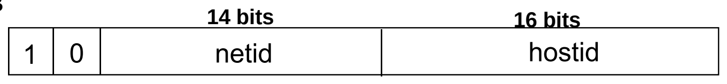
Class A



- 126 networks with up to 16 million hosts

1.0.0.0 to 127.255.255.255

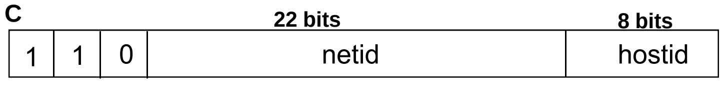
Class B



- 16,382 networks with up to 64,000 hosts

128.0.0.0 to 191.255.255.255

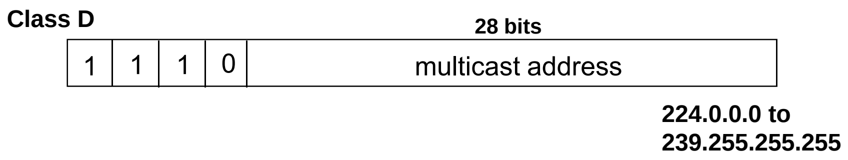
Class C



- 2 million networks with up to 254 hosts

192.0.0.0 to 223.255.255.255

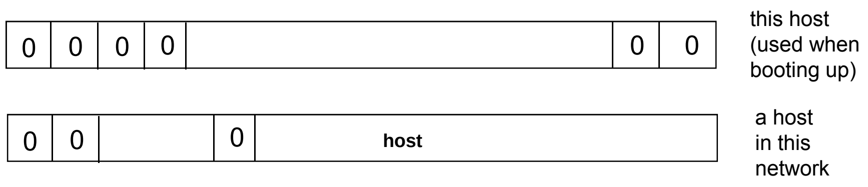
Class D Addresses



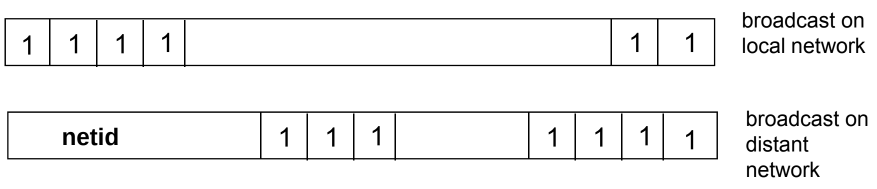
- Up to 250 million multicast groups at the same time
- Permanent group addresses
  - All systems in LAN; All routers in LAN;
  - All OSPF routers on LAN; All designated OSPF routers on a LAN, etc.
- Temporary groups addresses created as needed
- Special multicast routers

Reserved Host IDs (all 0s & 1s)

Internet address used to refer to network has hostid set to all 0s



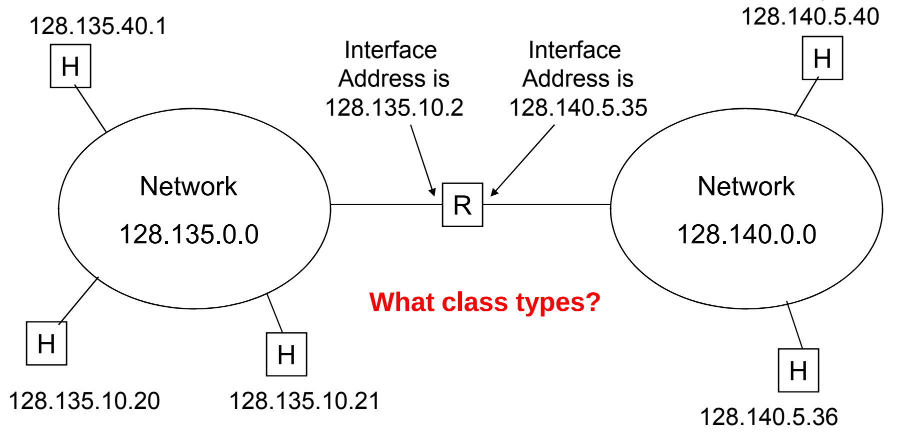
Broadcast address has hostid set to all 1s



Private IP Addresses

- Specific ranges of IP addresses set aside for use in private networks (RFC 1918)
- Use restricted to private internets; routers in public Internet discard packets with these addresses
- Range 1: 10.0.0.0 to 10.255.255.255
- Range 2: 172.16.0.0 to 172.31.255.255
- Range 3: 192.168.0.0 to 192.168.255.255
- Network Address Translation (NAT) used to convert between private & global IP addresses

Example of IP Addressing



What class types?

Address with host ID=all 0s refers to the network

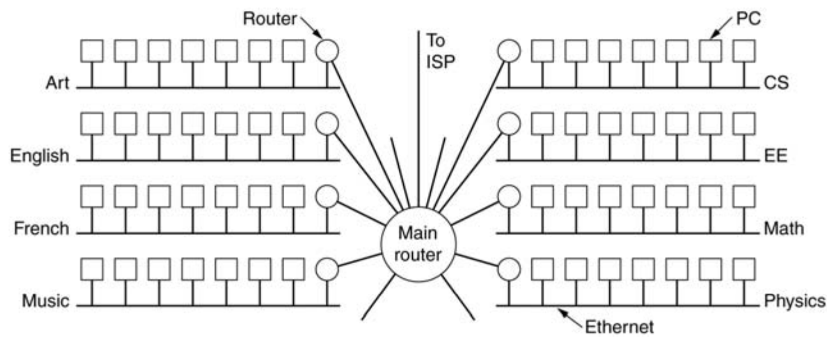
Address with host ID=all 1s refers to a broadcast packet

R = router

H = host

Subnets

A campus network consisting of LANs for various departments.



Subnetting: how to allow a network to be split into several parts for internal use but still act like a single network to the outside

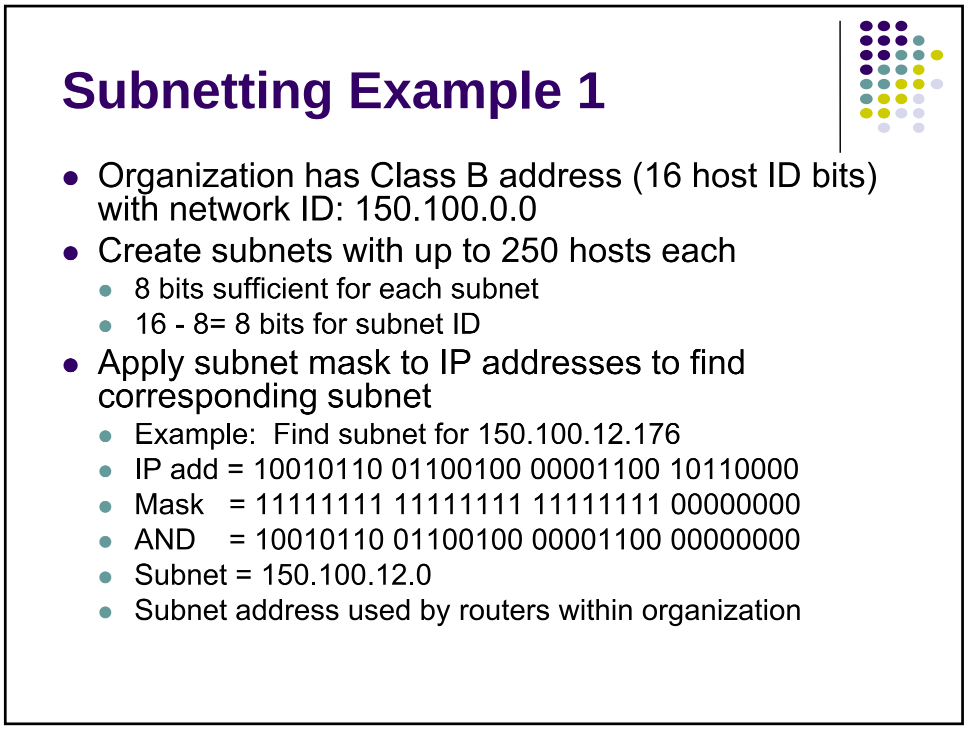
- When a packet comes into the main router, how does it know which subnet to give the packet to?

Subnet Addressing

Does a LAN need a unique network address?

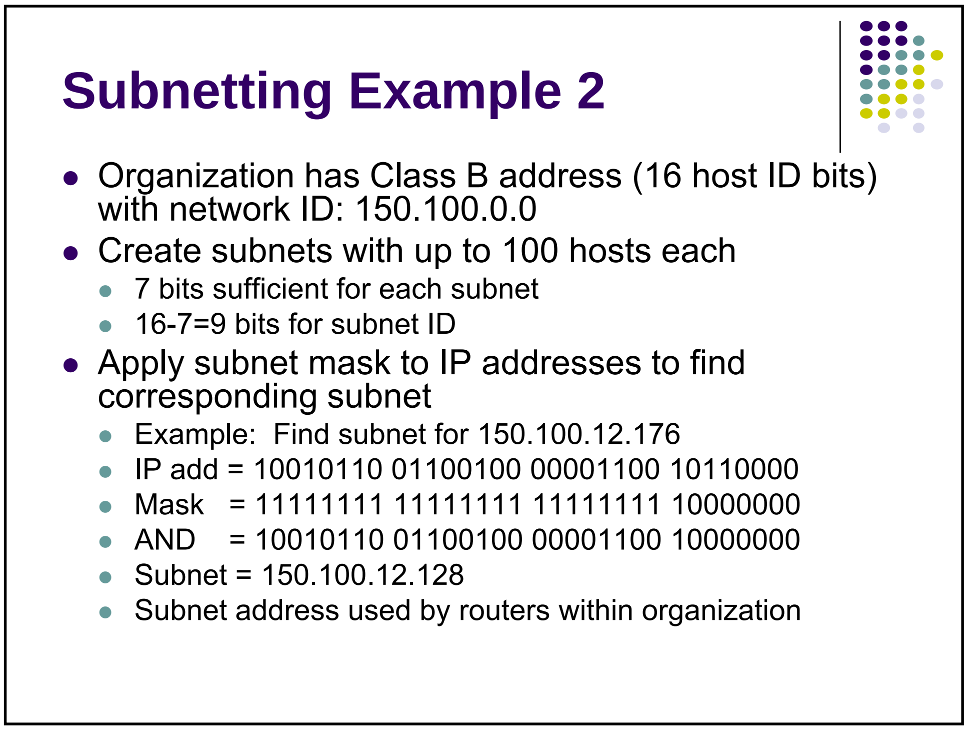
- Subnet addressing introduces another hierarchical level
- Transparent to remote networks
- Simplifies management of multiplicity of LANs
- Masking used to find subnet number

Original address	1	0	Net ID	Host ID	
Subnetted address	1	0	Net ID	Subnet ID	Host ID



Subnetting Example 1

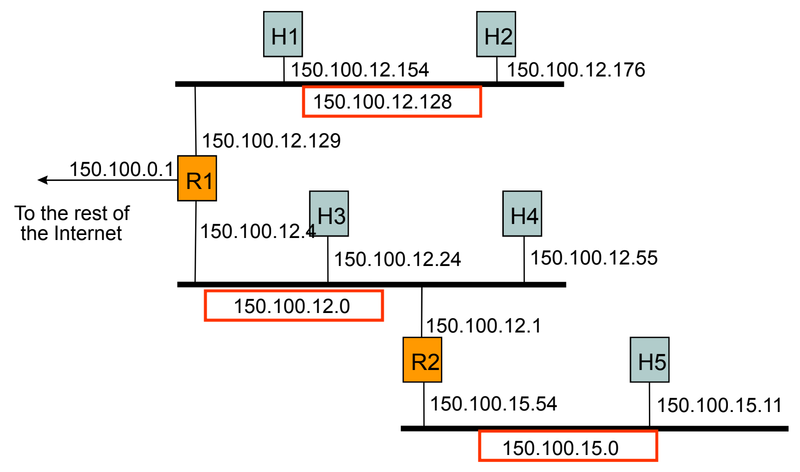
- Organization has Class B address (16 host ID bits) with network ID: 150.100.0.0
- Create subnets with up to 250 hosts each
  - 8 bits sufficient for each subnet
  - 16 - 8= 8 bits for subnet ID
- Apply subnet mask to IP addresses to find corresponding subnet
  - Example: Find subnet for 150.100.12.176
  - IP add = 10010110 01100100 00001100 10110000
  - Mask   = 11111111 11111111 11111111 00000000
  - AND   = 10010110 01100100 00001100 00000000
  - Subnet = 150.100.12.0
  - Subnet address used by routers within organization



Subnetting Example 2

- Organization has Class B address (16 host ID bits) with network ID: 150.100.0.0
- Create subnets with up to 100 hosts each
  - 7 bits sufficient for each subnet
  - 16-7=9 bits for subnet ID
- Apply subnet mask to IP addresses to find corresponding subnet
  - Example: Find subnet for 150.100.12.176
  - IP add = 10010110 01100100 00001100 10110000
  - Mask   = 11111111 11111111 11111111 10000000
  - AND   = 10010110 01100100 00001100 10000000
  - Subnet = 150.100.12.128
  - Subnet address used by routers within organization

Subnet Example



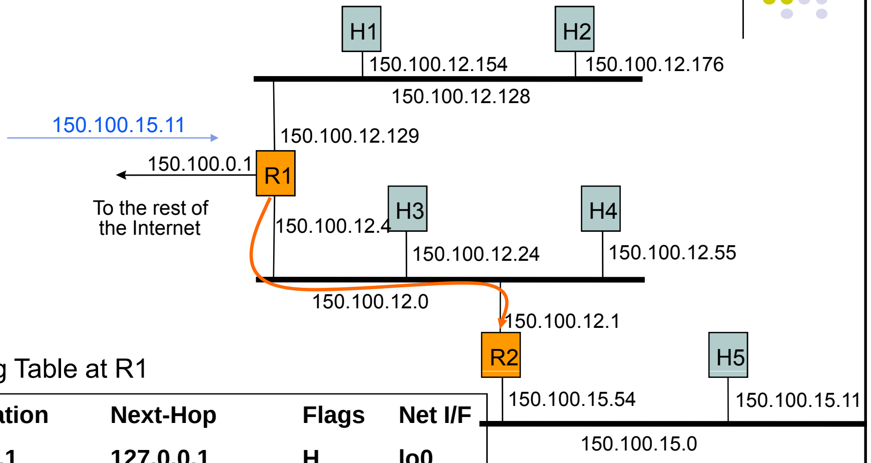
Routing with Subnetworks

- IP layer in hosts and routers maintain a routing table
- Originating host: To send an IP packet, consult routing table
  - If destination host is in same network, send packet *directly* using appropriate network interface
  - Otherwise, send packet indirectly; typically, routing table indicates a default router
- Router: Examine IP destination address in arriving packet
  - If dest IP address not own, router consults routing table to determine next-hop and associated network interface & forwards packet

Routing Table

- Each row in routing table contains:
  - Destination IP address
  - IP address of next-hop router
  - Physical address
  - Statistics information
  - Flags
    - H=1 (0) indicates route is to a host (network)
    - G=1 (0) indicates route is to a router (directly connected destination)
- Routing table search order & action
  - Complete destination address; send as per next-hop & G flag
  - Destination network ID; send as per next-hop & G flag
  - Default router entry; send as per next-hop
  - Declare packet undeliverable; send ICMP "host unreachable error" packet to originating host

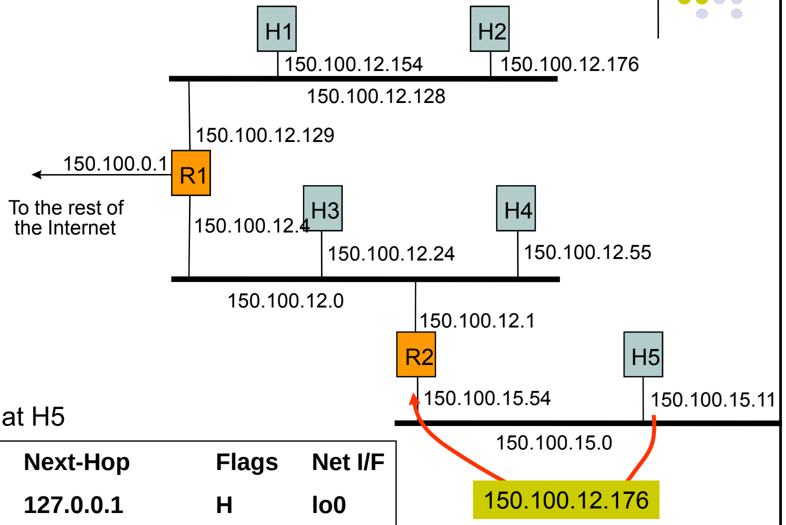
Example 1: A packet with 150.100.15.11 arrives at R1



Routing Table at R1

Destination	Next-Hop	Flags	Net I/F
127.0.0.1	127.0.0.1	H	lo0
150.100.12.128	150.100.12.129		emd0
150.100.12.0	150.100.12.4		emd1
150.100.15.0	150.100.12.1	G	emd1

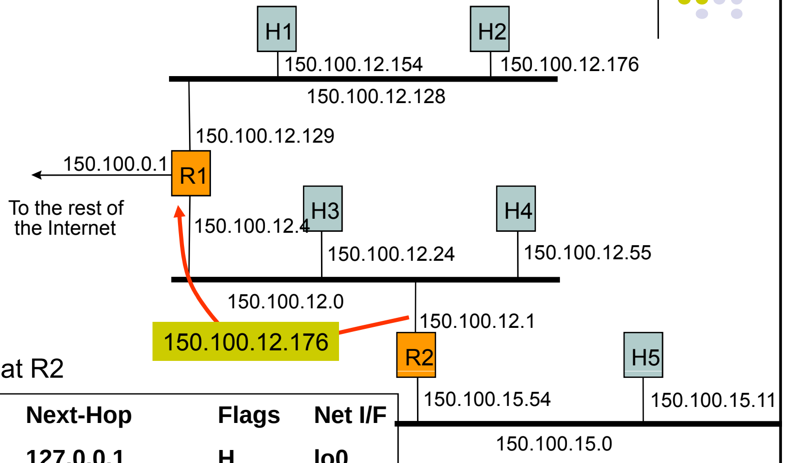
Example 2: Host H5 sends packet to host H2



Routing Table at H5

Destination	Next-Hop	Flags	Net I/F
127.0.0.1	127.0.0.1	H	lo0
default	150.100.15.54	G	emd0
150.100.15.0	150.100.15.11		emd0

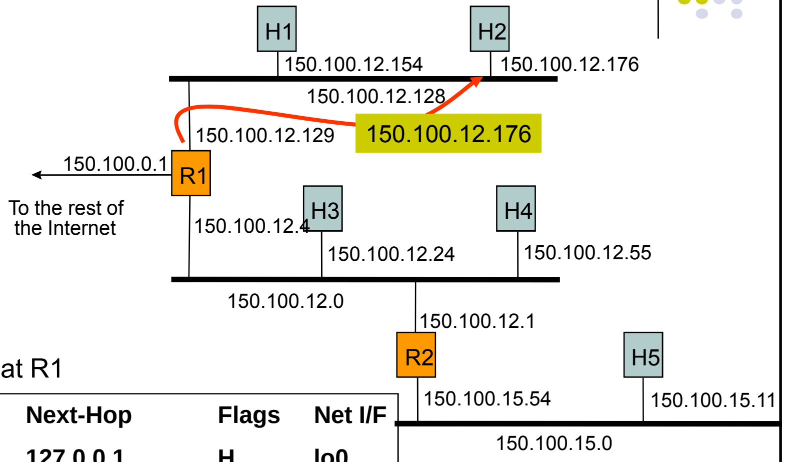
Example: Host H5 sends packet to host H2



Routing Table at R2

Destination	Next-Hop	Flags	Net I/F
127.0.0.1	127.0.0.1	H	lo0
default	150.100.12.4	G	emd0
150.100.15.0	150.100.15.54		emd1
150.100.12.0	150.100.12.1		emd0

Example: Host H5 sends packet to host H2



Routing Table at R1

Destination	Next-Hop	Flags	Net I/F
127.0.0.1	127.0.0.1	H	lo0
150.100.12.128	150.100.12.129		emd0
150.100.12.0	150.100.12.4		emd1
150.100.15.0	150.100.12.1	G	emd1

IP Address Problems

- In the 1990, two problems became apparent
  - IP addresses were being exhausted
  - IP routing tables were growing very large
- IP Address Exhaustion
  - Class A, B, and C address structure inefficient
    - Class B too large for most organizations, but future proof
    - Class C too small
    - Rate of class B allocation implied exhaustion by 1994
- IP routing table size
  - Growth in number of networks in Internet reflected in # of table entries
    - From 1991 to 1995, routing tables doubled in size every 10 months
    - Stress on router processing power and memory allocation
- Short-term solution:
- Classless Interdomain Routing (CIDR), RFC 1518
- New allocation policy (RFC 2050)
- Private IP Addresses set aside for intranets (NAT)
- Long-term solution: IPv6 with much bigger address space

Motivating Classless Inter-Domain Routing (CIDR)

- A company is allocated the following four /24 networks. At some router, it is often true that all of the four networks use the same outgoing line. CIDR aggregation can be done to reduce the number of entry at the router.
  - 128.56.24.0/24;
  - 128.56.25.0/24;
  - 128.56.26.0/24;
  - 128.56.27.0/24.

Pre-CIDR: Network with range of 4 contiguous class C blocks requires 4 entries

Post-CIDR: Network with range of 4 contiguous class C blocks requires 1 entry

Classless Inter-Domain Routing (CIDR)

- CIDR deals with Routing Table Explosion Problem
  - Networks represented by prefix and mask
  - Summarize a contiguous group of class C addresses using variable-length mask, if all of them use the same outgoing line
- Solution: *Route according to prefix of address*, not class
  - Routing table entry has <IP address, network mask>
  - Example: 192.32.136.0/21
  - `11000000 00100000 10001000 00000001` min address
  - `11111111 11111111 11111--- --------` mask
  - `11000000 00100000 10001--- --------` IP prefix
  - `11000000 00100000 10001111 11111110` max address

Supernetting (1)

- Summarize a contiguous group of class C addresses using variable-length mask
- Example: 150.158.16.0/20
  - IP Address (150.158.16.0) & mask length (20)
  - IP add = 10010110 10011110 00010000 00000000
  - Mask = 11111111 11111111 11110000 00000000
  - Contains 16 Class C blocks:
  - From 10010110 10011110 00010000 00000000
  - i.e. 150.158.16.0
  - Up to 10010110 10011110 00011111 00000000
  - i.e. 150.158.31.0

Supernetting (2)

- A router has the following CIDR entries in its routing table:

Address/mask	Next hop
128.56.24.0/22	Interface 0
128.56.60.0/22	Interface 1
default	Router 2

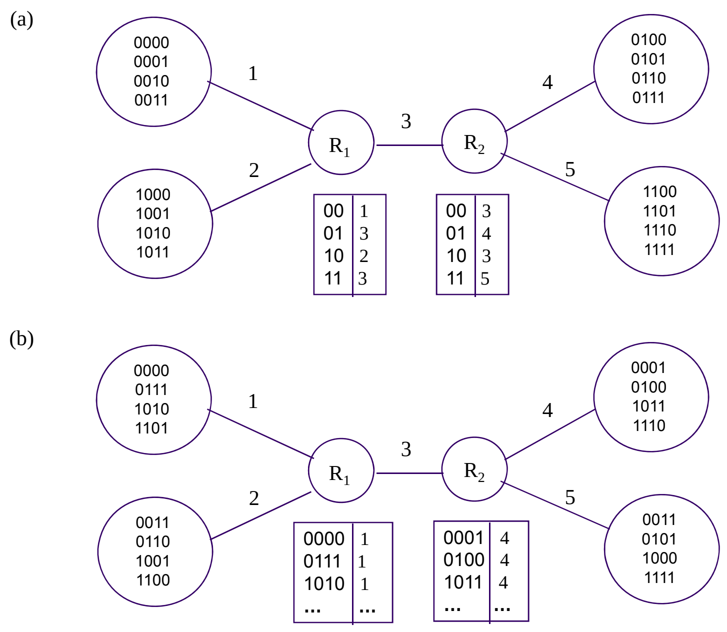
A packet comes with IP address of 128.56.63.10. What does the router do?

New Address Allocation Policy

- Class A & B assigned only for clearly demonstrated need
- *Consecutive* blocks of class C assigned (up to 64 blocks)
  - All IP addresses in the range have a common **prefix**, and every address with that prefix is within the range
  - Arbitrary prefix length for network ID improves efficiency
- Lower half of class C space assigned to regional authorities
  - More hierarchical allocation of addresses
  - Service provider to customer

Address Requirement	**Address Allocation**
< 256	1 Class C
256<,<512	2 Class C
512<,<1024	4 Class C
1024<,<2048	8 Class C
2048<,<4096	16 Class C
4096<,<8192	32 Class C
8192<,<16384	64 Class C

Hierarchical Routing & Table Efficiency



CIDR Allocation Principles (RFC 1518-1520)

- IP address assignment reflects physical topology of network
- Network topology follows continental/national boundaries
  - IP addresses should be assigned on this basis
- Transit routing domains (TRDs) have unique IP prefix
  - carry traffic between routing domains
  - interconnected non-hierarchically, cross national boundaries
  - Most routing domains single-homed: attached to a single TRD
  - Such domains assigned addresses with TRD's IP prefix
  - All of the addresses attached to a TRD aggregated into 1table entry
- Implementation primarily through BGPv4 (RFC 1520)

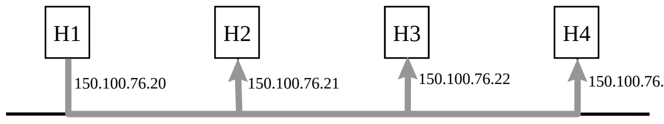
Longest Prefix Match

- CIDR impacts routing & forwarding
- Routing tables and routing protocols must carry IP address and mask
- Multiple entries may match a given IP destination address
- Example: perform CIDR on the following three /24 IP addresses (but 128.56.24.0/24 to a different port)
  - 128.56.25.0/24;
  - 128.56.26.0/24;
  - 128.56.27.0/24;
  - What if a packet with dest. IP address 128.56.24.0 comes?
- Packet must be routed using the *more specific route*, that is, the longest prefix match
- Several fast longest-prefix matching algorithms are available

Address Resolution Protocol

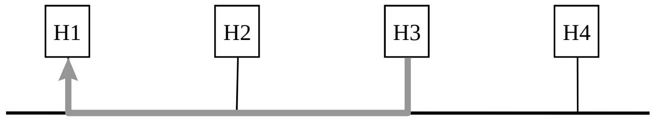
Although IP address identifies a host, the packet is physically delivered by an underlying network (e.g., Ethernet) which uses its own *physical address* (MAC address in Ethernet). How to map an IP address to a physical address? How to speed up? How fresh?

H1 wants to learn physical address of H3 -> broadcasts an ARP request



ARP request (what is the MAC address of 150.100.76.22?)

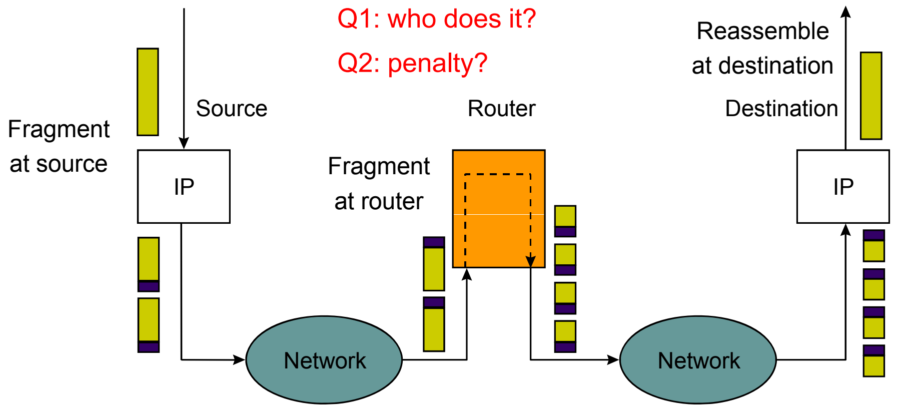
Every host receives the request, but only H3 reply with its physical address



ARP response (my MAC address is 08:00:5a:3b:94)

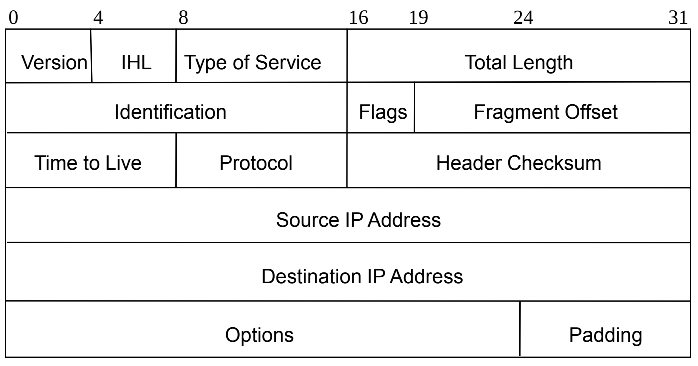
Fragmentation and Reassembly

- Identification identifies a particular packet
- Flags = (unused, don't fragment/DF, more fragment/MF)
- Fragment offset identifies the location of a fragment within a packet



Q3: Does it make sense to do reassembly at intermediate routers? Why?

RE: IP Packet Header



Identification, Flags, and Fragment Offset: used for fragmentation and reassembly

Fragment offset is 13 bits; total length is 16 bits, what does it imply?

Example: Fragmenting a Packet

- A packet is to be forwarded to a network with MTU of 576 bytes. The packet has an IP header of 20 bytes and a data part of 1484 bytes. and of each fragment.
- Maximum data length per fragment = 576 - 20 = 556 bytes.
- We set maximum data length to 552 bytes to get multiple of 8.

	Total Length	Id	MF	Fragment Offset
Original packet	1504	x	0	0
Fragment 1	572	x	1	0
Fragment 2	572	x	1	69
Fragment 3	400	x	0	138

Internet Control Message Protocol (ICMP)

- RFC 792; Encapsulated in IP packet (protocl type = 1)
- Handles error and control messages
- If router cannot deliver or forward a packet, it sends an ICMP "host unreachable" message to the source
- If router receives packet that should have been sent to another router, it sends an ICMP "redirect" message to the sender; Sender modifies its routing table
- ICMP "router discovery" messages allow host to learn about routers in its network and to initialize and update its routing tables
- ICMP echo request and reply facilitate diagnostic and used in "ping"

Chapter 8 Communication Networks and Services

IPv6

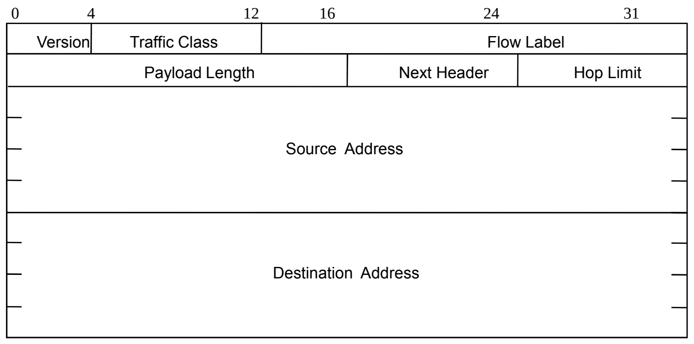
IPv6

- **Longer address field:**
  - 128 bits can support up to 3.4 x 10^{38} hosts
- **Simplified header format:**
  - Simpler format to speed up processing of each header
  - All fields are of fixed size
  - IPv4 vs IPv6 fields:
    - Same: Version
    - Dropped: Header length, ID/flags/frag offset, header checksum
    - Replaced:
      - Datagram length by Payload length
      - Protocol type by Next header
      - TTL by Hop limit
      - TOS by traffic class
    - New: Flow label

Other IPv6 Features

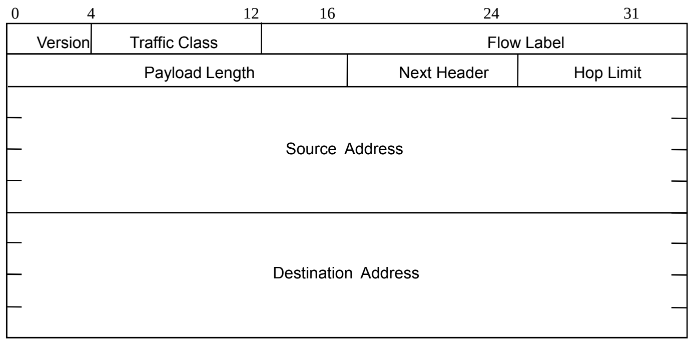
- **Flexible support for options (Next header):** more efficient and flexible options encoded in optional *extension headers (immediately following this header)*
- **Flow label capability:** “flow label” to identify a packet flow that requires a certain QoS
- **Security:** built-in authentication and confidentiality
- **Large packets:** supports payloads that are longer than 64 K bytes, called *jumbo* payloads.
- **Fragmentation at source only:** source should check the minimum MTU along the path
- **No checksum field:** removed to reduce packet processing time in a router

IPv6 Header Format



- Version field same size, same location
- Traffic class to support differentiated services
- Flow: sequence of packets from particular source to particular destination for which source requires special handling

IPv6 Basic Header Format



- Payload length: length of data excluding header, up to 65535 B
- Next header: type of extension header that follows basic header
- Hop limit: # hops packet can travel before being dropped by a router

Why fragmentation at source only? Relieving load at routers.

Extension Headers

- Allows an arbitrary number of extension headers be placed between the basic header and the payload (the extension headers are chained by the next header field)
- Large Packet: payload>64K

0	8	16	24 31
Next header	0	194	Opt len = 4
Jumbo payload length			

- Fragmentation: At source only
  - Source performs “path MTU discovery” (a fragment extension header for each packet fragment)

0	8	16	29	31
Next header	Reserved	Fragment offset	Res	M
Identification				

Extension Headers

- Source Routing: strict/loose routes

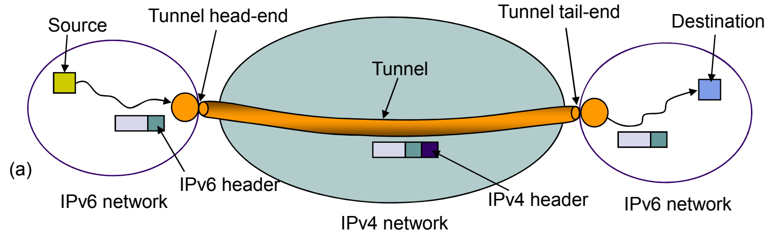
0	8	16	24 31
Next header	Header length	Routing type = 0	Segment left
Reserved	Strict/loose bit mask		
Address 1			
Address 2			
• • •			
Address 24			

IPv6 Addressing

- Address Categories
  - Unicast: single network interface
  - Multicast: group of network interfaces, typically at different locations. Packet sent to all.
  - Anycast: group of network interfaces. Packet sent to only one interface in group, e.g. nearest.
- Hexadecimal notation
  - Groups of 16 bits represented by 4 hex digits
  - Separated by colons
    - 4BF5:AA12:0216:FEBC:BA5F:039A:BE9A:2176
  - Shortened forms:
    - 4BF5:0000:0000:0000:BA5F:039A:000A:2176
    - To 4BF5:0:0:0:BA5F:39A:A:2176
    - To 4BF5::BA5F:39A:A:2176
  - Mixed notation:
    - ::FFFF:128.155.12.198

Migration from IPv4 to IPv6

- Gradual transition from IPv4 to IPv6
- Dual IP stacks: routers run IPv4 & IPv6
  - Type field used to direct packet to IP version
- IPv6 islands can tunnel across IPv4 networks
  - Encapsulate user packet insider IPv4 packet



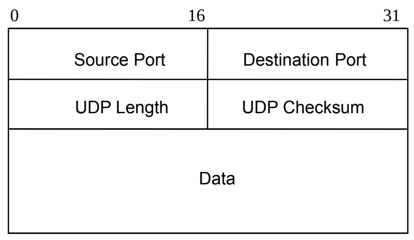
Chapter 8 Communication Networks and Services

Transport Layer Protocols: UDP and TCP

UDP

- Best effort datagram service
- Multiplexing enables sharing of IP datagram service
- Simple transmitter & receiver
  - Connectionless: no handshaking & no connection state
  - Low header overhead
  - No flow control, no error control, no congestion control
  - UDP datagrams can be lost or out-of-order
- Applications
  - multimedia (e.g. RTP)
  - network services (e.g. DNS, RIP, SNMP)

UDP Datagram



0-255
- Well-known ports

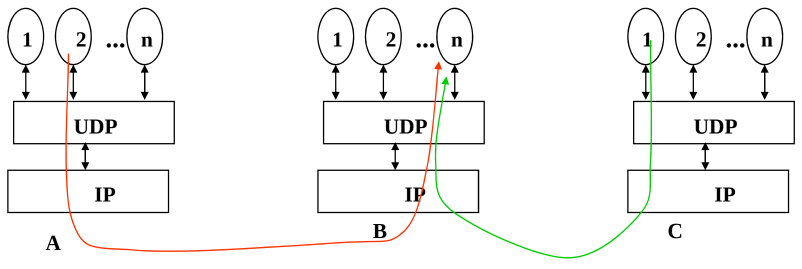
256-1023
- Less well-known ports

1024-65536
- Ephemeral client ports

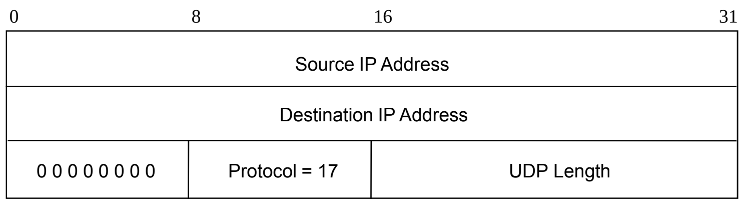
- Source and destination port numbers
  - Client ports are ephemeral
  - Server ports are well-known
  - Max number is 65,535
- UDP length
  - Total number of bytes in datagram (including header)
  - 8 bytes ≤ length ≤ 65,535
- UDP Checksum
  - Optionally detects errors in UDP datagram

UDP Multiplexing

- All UDP datagrams arriving to IP address B and destination port number *n* are delivered to the same process
- Source port number is not used in multiplexing



UDP Checksum Calculation



UDP pseudo-header

- UDP checksum detects for end-to-end errors
- Covers pseudoheader followed by UDP datagram
- IP addresses included to detect against misdelivery
- IP & UDP checksums set to zero during calculation
- Pad with 1 byte of zeros if UDP length is odd

UDP Receiver Checksum

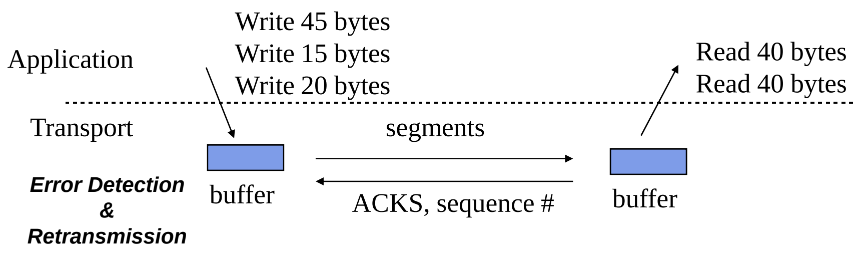
- UDP receiver recalculates the checksum and silently discards the datagram if errors detected
  - “silently” means no error message is generated
- The use of UDP checksums is optional
- But hosts are required to have checksums enabled

TCP

- Reliable byte-stream service
- More complex transmitter & receiver
  - Connection-oriented: full-duplex unicast connection between client & server processes
  - Connection setup, connection state, connection release
  - Higher header overhead
  - Error control, flow control, and congestion control
  - Higher delay than UDP
- Most applications use TCP
  - HTTP, SMTP, FTP, TELNET, POP3, …

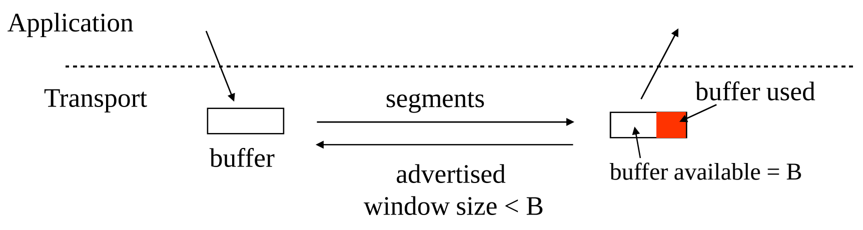
Reliable Byte-Stream Service

- Stream Data Transfer
  - transfers a contiguous stream of bytes across the network, with no indication of boundaries
  - groups bytes into segments
  - transmits segments as convenient (Push function defined)
- Reliability
  - error control mechanism to deal with IP transfer impairments



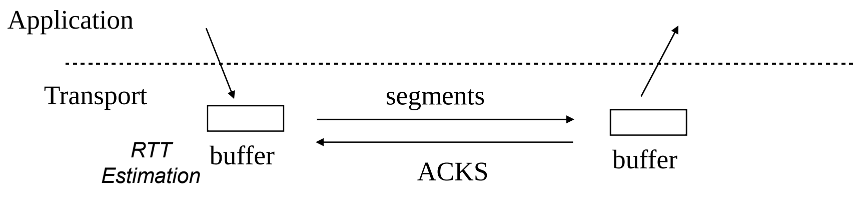
Flow Control

- Buffer limitations & speed mismatch can result in loss of data that arrives at destination; p2p issue
- Receiver controls rate at which sender transmits to prevent receiver’s buffer overflow



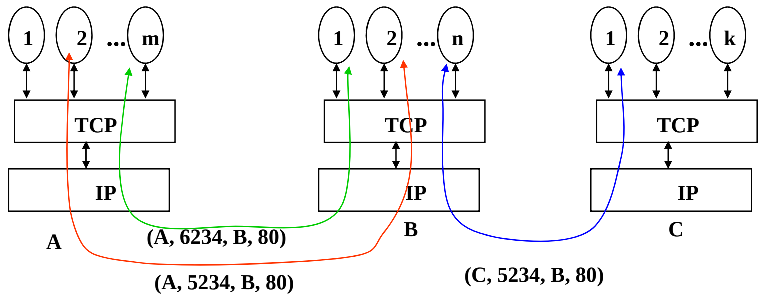
Congestion Control

- Available bandwidth to destination varies with activity of other users; aggregation issue
- Transmitter dynamically adjusts transmission rate according to network congestion as indicated by RTT (round trip time) & ACKs
- Elastic utilization of network bw. & router buffer

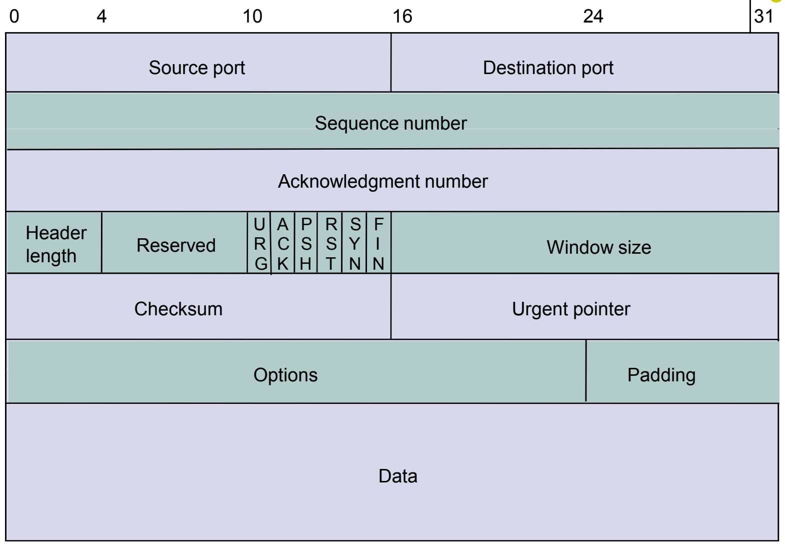


TCP Multiplexing

- A *TCP connection* is specified by a *4-tuple*
  - (source IP address, source port, destination IP address, destination port)
- TCP allows multiplexing of multiple connections between end systems to support multiple applications simultaneously
- Arriving segment directed according to connection 4-tuple



TCP Segment Format



- Each TCP segment has header of 20 or more bytes + 0 or more bytes of data

TCP Header

Port Numbers

- A socket identifies a connection endpoint
  - IP address + port
- A connection specified by a *socket pair*
- Well-known ports

FTP	20
Telnet	23
DNS	53
HTTP	80

Sequence Number

- Byte count
- First byte in segment
- 32 bits long
- $0 \leq SN \leq 2^{32}-1$
- Initial sequence number (ISN) selected during connection setup (SYN flag bit is 1);

TCP Header

Acknowledgement Number

- SN of next byte expected by receiver
- Acknowledges that all prior bytes in stream have been received correctly
- Valid if ACK flag is set

Header length

- 4 bits
- Length of header in multiples of 32-bit words
- Minimum header length is 20 bytes
- Maximum header length is 60 bytes

TCP Header

Reserved

- 6 bits

Control

- 6 bits
- URG: urgent pointer flag
  - Urgent message end = SN + **urgent pointer**
- ACK: ACK packet flag
- PSH: override TCP buffering
- RST: reset connection
  - Upon receipt of RST, connection is terminated and application layer notified
- SYN: establish connection
- FIN: close connection

TCP Header

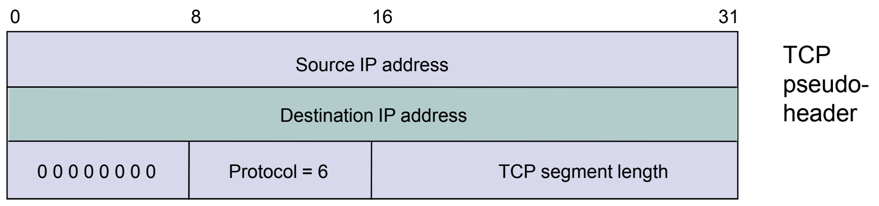
Window Size

- 16 bits to advertise window size
- Used for flow control
- Sender will accept bytes with SN from ACK to ACK + window
- Maximum window size is 65535 bytes

TCP Checksum

- Internet checksum method
- TCP pseudoheader + TCP segment

TCP Checksum Calculation



- TCP error detection uses same procedure as UDP

TCP Header

Options

- Variable length
- NOP (No Operation) option is used to pad TCP header to multiple of 32 bits
- Time stamp option is used for round trip measurements

Options

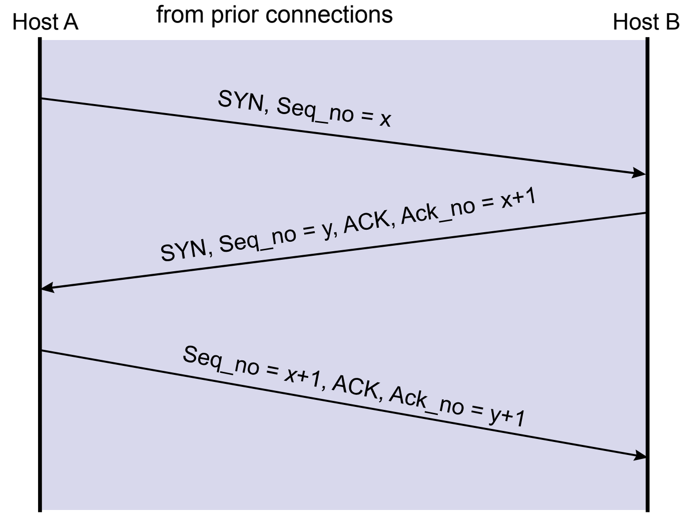
- Maximum Segment Size (MSS) option specifices largest segment a receiver wants to receive
- Window Scale option increases TCP window from 16 to 32 bits

TCP Connection Management

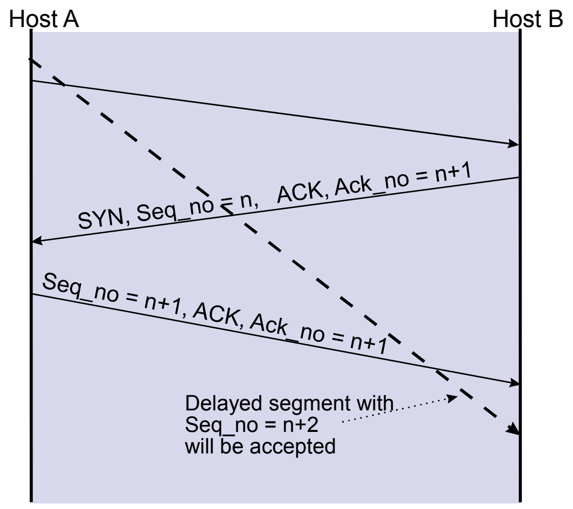
- Select initial sequence numbers (ISN) to protect against segments from prior connections (that may circulate in the network and arrive at a much later time; delayed duplicates)
- Select ISN to avoid overlap with sequence numbers of prior connections
- Use local clock to select ISN sequence number
- Time for clock to go through a full cycle should be greater than the maximum lifetime of a segment (MSL); Typically MSL=120 seconds
- High bandwidth connections pose a problem
- $2^n > 2 *$ max packet life * R bytes/second

TCP Connection Establishment

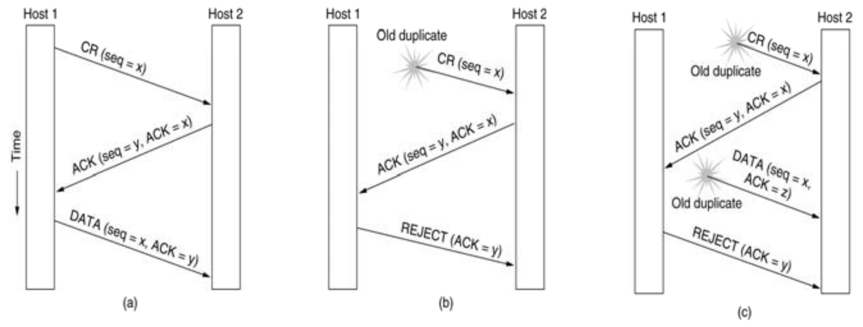
- "Three-way Handshake"
- ISN's protect against segments from prior connections


Host A
Host B
SYN, Seq_no = x
SYN, Seq_no = y, ACK, Ack_no = x+1
Seq_no = x+1, ACK, Ack_no = y+1

If host always uses the same ISN


Host A
Host B
SYN, Seq_no = n,   ACK, Ack_no = n+1
Seq_no = n+1, ACK, Ack_no = n+1
Delayed segment with
Seq_no = n+2
will be accepted

Connection Establishment (2)

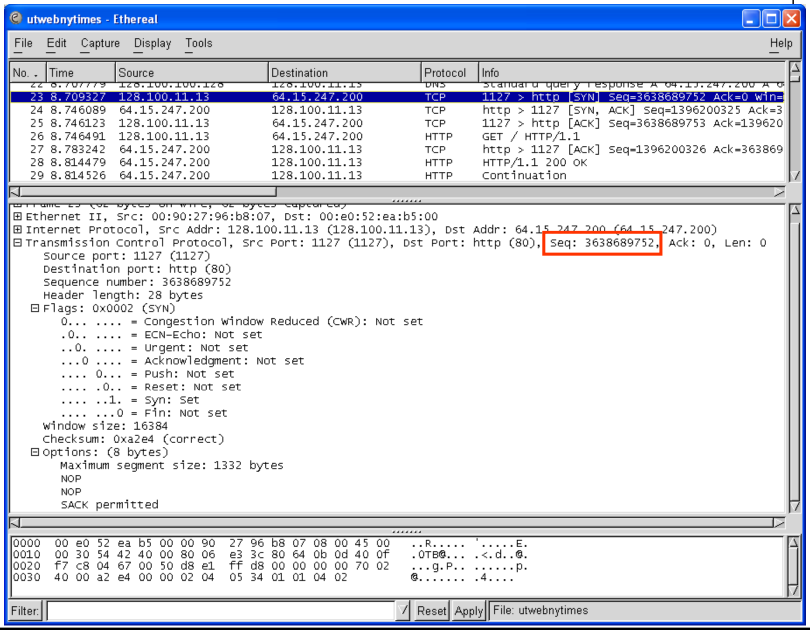


Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST.
(a) Normal operation,
(b) Old CONNECTION REQUEST appearing out of nowhere.
(c) Duplicate CONNECTION REQUEST and duplicate ACK.

Maximum Segment Size

- Maximum Segment Size
  - largest block of data that TCP sends to other end
- Each end can announce its MSS during connection establishment
- Default is 576 bytes including 20 bytes for IP header and 20 bytes for TCP header
- Ethernet implies MSS of 1460 bytes
- IEEE 802.3 implies 1452

Near End: Connection Request

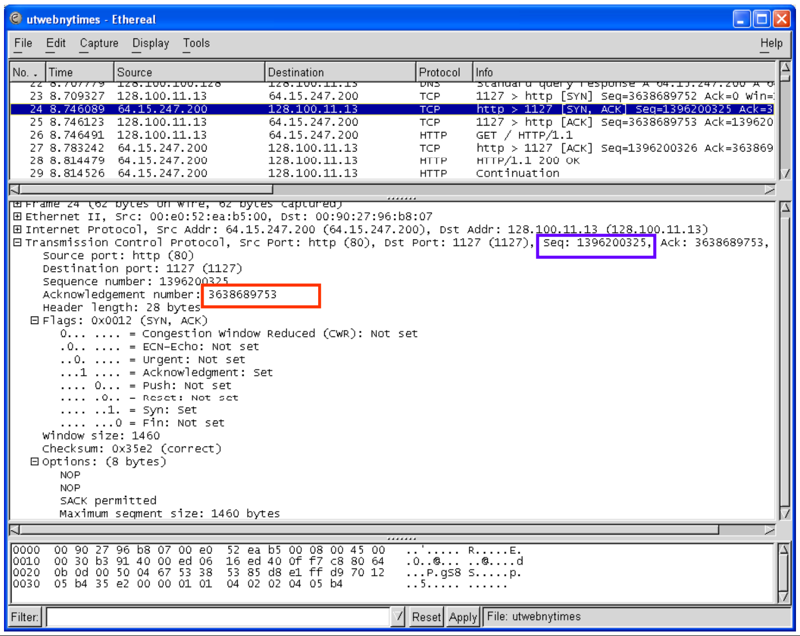


```
No. . Time      Source           Destination     Protocol Info
 22 8.707779  128.100.100.128  128.100.11.13   DNS      Standard query response A 64.15.247.200 A 6
 23 8.709327  128.100.11.13    64.15.247.200   TCP      1127 > http [SYN] Seq=3638689752 Ack=0 Win=
 24 8.746089  64.15.247.200    128.100.11.13   TCP      http > 1127 [SYN, ACK] Seq=1396200325 Ack=3
 25 8.746123  128.100.11.13    64.15.247.200   TCP      1127 > http [ACK] Seq=3638689753 Ack=139620
 26 8.746491  128.100.11.13    64.15.247.200   HTTP     GET / HTTP/1.1
 27 8.783242  64.15.247.200    128.100.11.13   TCP      http > 1127 [ACK] Seq=1396200326 Ack=363869
 28 8.814479  64.15.247.200    128.100.11.13   HTTP     HTTP/1.1 200 OK
 29 8.814526  64.15.247.200    128.100.11.13   HTTP     Continuation
```

```
⊞ Frame 23 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00
⊞ Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 64.15.247.200 (64.15.247.200)
⊟ Transmission Control Protocol, Src Port: 1127 (1127), Dst Port: http (80), Seq: 3638689752, Ack: 0, Len: 0
    Source port: 1127 (1127)
    Destination port: http (80)
    Sequence number: 3638689752
    Header length: 28 bytes
  ⊟ Flags: 0x0002 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgment: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 16384
    Checksum: 0xa2e4 (correct)
  ⊟ Options: (8 bytes)
      Maximum segment size: 1332 bytes
      NOP
      NOP
      SACK permitted
```

```
0000  00 e0 52 ea b5 00 00 90  27 96 b8 07 08 00 45 00   ..R..... '.....E.
0010  00 30 54 42 40 00 80 06  e3 3c 80 64 0b 0d 40 0f   .0TB@... .<.d..@.
0020  f7 c8 04 67 00 50 d8 e1  ff d8 00 00 00 00 70 02   ...g.P.. ......p.
0030  40 00 a2 e4 00 00 02 04  05 34 01 01 04 02         @....... .4....
```

Far End: Ack and Request

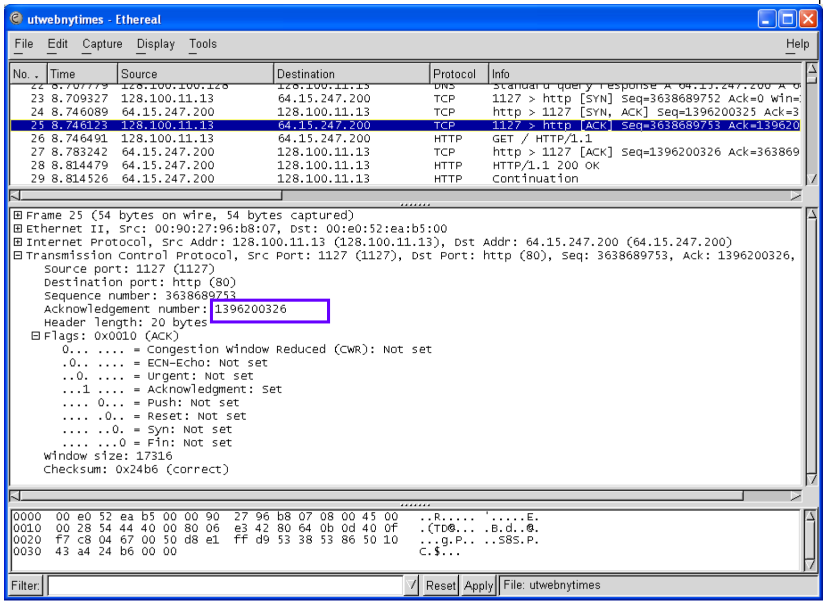

utwebnytimes - Ethereal
File Edit Capture Display Tools Help
Filter: Reset Apply File: utwebnytimes

```
No. . Time      Source           Destination      Protocol  Info
 22 8.707779  128.100.100.128  128.100.11.13    DNS       Standard query response A 64.15.247.200 A 6
 23 8.709327  128.100.11.13    64.15.247.200    TCP       1127 > http [SYN] Seq=3638689752 Ack=0 Win=
 24 8.746089  64.15.247.200    128.100.11.13    TCP       http > 1127 [SYN, ACK] Seq=1396200325 Ack=3
 25 8.746123  128.100.11.13    64.15.247.200    TCP       1127 > http [ACK] Seq=3638689753 Ack=139620
 26 8.746491  128.100.11.13    64.15.247.200    HTTP      GET / HTTP/1.1
 27 8.783242  64.15.247.200    128.100.11.13    TCP       http > 1127 [ACK] Seq=1396200326 Ack=363869
 28 8.814479  64.15.247.200    128.100.11.13    HTTP      HTTP/1.1 200 OK
 29 8.814526  64.15.247.200    128.100.11.13    HTTP      Continuation
```

```
⊞ Frame 24 (62 bytes on wire, 62 bytes captured)
⊞ Ethernet II, Src: 00:e0:52:ea:b5:00, Dst: 00:90:27:96:b8:07
⊞ Internet Protocol, Src Addr: 64.15.247.200 (64.15.247.200), Dst Addr: 128.100.11.13 (128.100.11.13)
⊟ Transmission Control Protocol, Src Port: http (80), Dst Port: 1127 (1127), Seq: 1396200325, Ack: 3638689753,
    Source port: http (80)
    Destination port: 1127 (1127)
    Sequence number: 1396200325
    Acknowledgement number: 3638689753
    Header length: 28 bytes
  ⊟ Flags: 0x0012 (SYN, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 1460
    Checksum: 0x35e2 (correct)
  ⊟ Options: (8 bytes)
      NOP
      NOP
      SACK permitted
      Maximum segment size: 1460 bytes
```

```
0000  00 90 27 96 b8 07 00 e0  52 ea b5 00 08 00 45 00   ..'..... R.....E.
0010  00 30 b3 91 40 00 ed 06  16 ed 40 0f f7 c8 80 64   .0..@... ..@....d
0020  0b 0d 00 50 04 67 53 38  53 85 d8 e1 ff d9 70 12   ...P.gS8 S.....p.
0030  05 b4 35 e2 00 00 01 01  04 02 02 04 05 b4         ..5..... ......
```

Near End: Ack

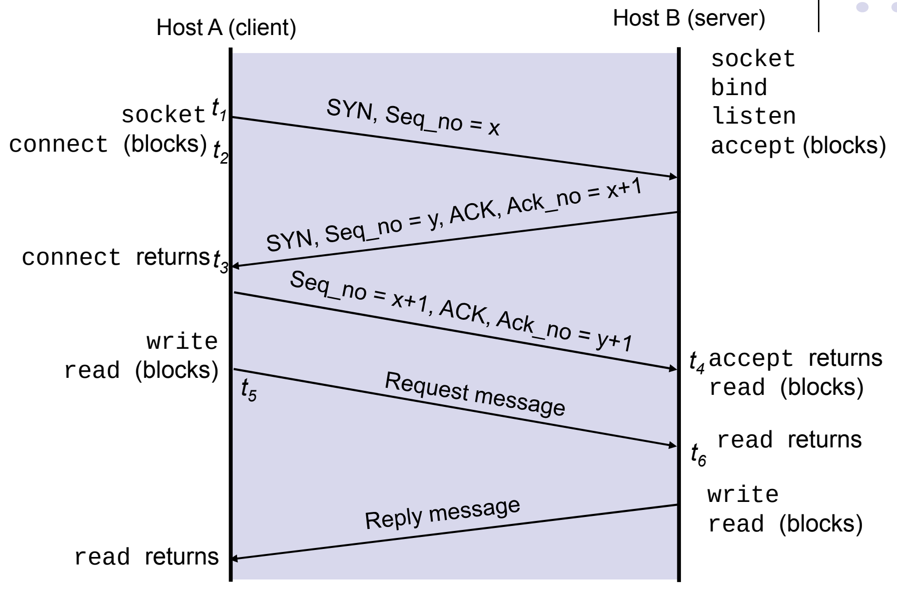

utwebnytimes - Ethereal
File Edit Capture Display Tools Help
Filter: Reset Apply File: utwebnytimes

```
No. . Time      Source           Destination      Protocol  Info
 22 8.707779  128.100.100.128  128.100.11.13    DNS       Standard query response A 64.15.247.200 A 6
 23 8.709327  128.100.11.13    64.15.247.200    TCP       1127 > http [SYN] Seq=3638689752 Ack=0 Win=
 24 8.746089  64.15.247.200    128.100.11.13    TCP       http > 1127 [SYN, ACK] Seq=1396200325 Ack=3
 25 8.746123  128.100.11.13    64.15.247.200    TCP       1127 > http [ACK] Seq=3638689753 Ack=139620
 26 8.746491  128.100.11.13    64.15.247.200    HTTP      GET / HTTP/1.1
 27 8.783242  64.15.247.200    128.100.11.13    TCP       http > 1127 [ACK] Seq=1396200326 Ack=363869
 28 8.814479  64.15.247.200    128.100.11.13    HTTP      HTTP/1.1 200 OK
 29 8.814526  64.15.247.200    128.100.11.13    HTTP      Continuation
```

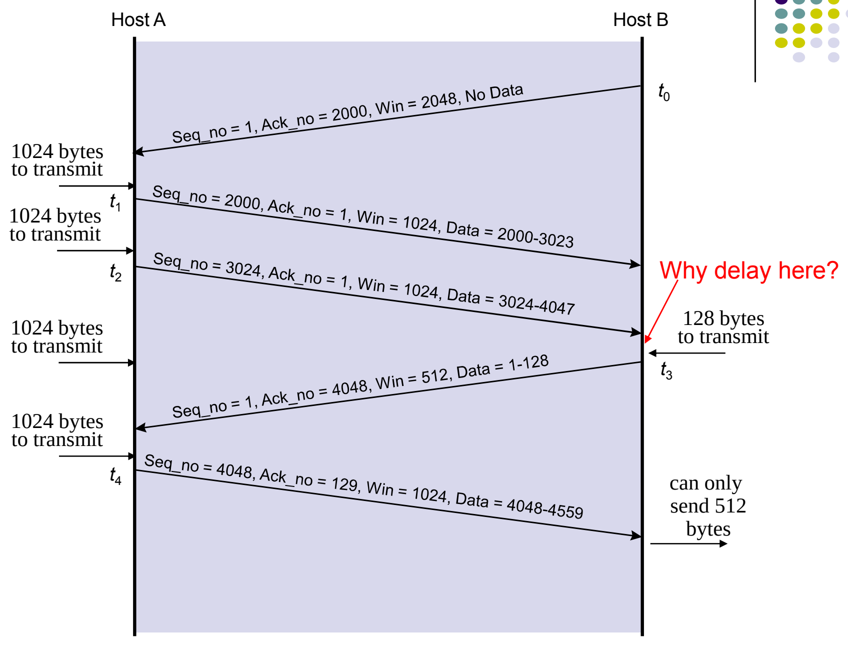
```
⊞ Frame 25 (54 bytes on wire, 54 bytes captured)
⊞ Ethernet II, Src: 00:90:27:96:b8:07, Dst: 00:e0:52:ea:b5:00
⊞ Internet Protocol, Src Addr: 128.100.11.13 (128.100.11.13), Dst Addr: 64.15.247.200 (64.15.247.200)
⊟ Transmission Control Protocol, Src Port: 1127 (1127), Dst Port: http (80), Seq: 3638689753, Ack: 1396200326,
    Source port: 1127 (1127)
    Destination port: http (80)
    Sequence number: 3638689753
    Acknowledgement number: 1396200326
    Header length: 20 bytes
  ⊟ Flags: 0x0010 (ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgment: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
      .... ..0. = Syn: Not set
      .... ...0 = Fin: Not set
    Window size: 17316
    Checksum: 0x24b6 (correct)
```

```
0000  00 e0 52 ea b5 00 00 90  27 96 b8 07 08 00 45 00   ..R..... '.....E.
0010  00 28 54 44 40 00 80 06  e3 42 80 64 0b 0d 40 0f   .(TD@... .B.d..@.
0020  f7 c8 04 67 00 50 d8 e1  ff d9 53 38 53 86 50 10   ...g.P.. ..S8S.P.
0030  43 a4 24 b6 00 00                                  C.$...
```

Client-Server Application



TCP Window Flow Control


Host A
Host B
t_0
Seq_no = 1, Ack_no = 2000, Win = 2048, No Data
1024 bytes to transmit
t_1
Seq_no = 2000, Ack_no = 1, Win = 1024, Data = 2000-3023
1024 bytes to transmit
t_2
Seq_no = 3024, Ack_no = 1, Win = 1024, Data = 3024-4047
Why delay here?
128 bytes to transmit
1024 bytes to transmit
t_3
Seq_no = 1, Ack_no = 4048, Win = 512, Data = 1-128
1024 bytes to transmit
t_4
Seq_no = 4048, Ack_no = 129, Win = 1024, Data = 4048-4559
can only send 512 bytes

Nagle Algorithm

- Situation: user types 1 character at a time
  - Transmitter sends TCP segment per character (41B)
  - Receiver sends ACK (40B)
  - Receiver echoes received character (41B)
  - Transmitter ACKs echo (40 B)
  - 162 bytes transmitted to transfer 1 character!
- Solution:
  - TCP sends data & waits for ACK
  - New characters buffered instead
  - Send new characters when ACK arrives
  - Algorithm adjusts to RTT
    - Short RTT send frequently at low efficiency
    - Long RTT send less frequently at greater efficiency

Silly Window Syndrome

- Situation:
  - Transmitter sends large amount of data
  - Receiver buffer depleted slowly, so buffer fills
  - Every time a few bytes read from buffer, a new advertisement to transmitter is generated
  - Sender immediately sends data & fills buffer
  - Many small, inefficient segments are transmitted
- Solution:
  - Receiver does not advertize window until window is at least ½ of receiver buffer or maximum segment size
  - Transmitter refrains from sending small segments

Sequence Number Wraparound

- $2^{32} = 4.29 \times 10^9$ bytes = 34.3×10^9 bits
  - At 1 Gbps, sequence number wraparound in 34.3 seconds (MSL = 120 seconds).
- Timestamp option: Insert 32 bit timestamp in header of each segment
  - Timestamp + sequence no → 64-bit seq. no
  - Timestamp clock must:
    - tick forward at least once every 2^{31} bits
    - Not complete cycle in less than one MSL
    - Example: clock tick every 1 ms @ 8 Tbps wraps around in 25 days

Where this timestamp can be filled in?

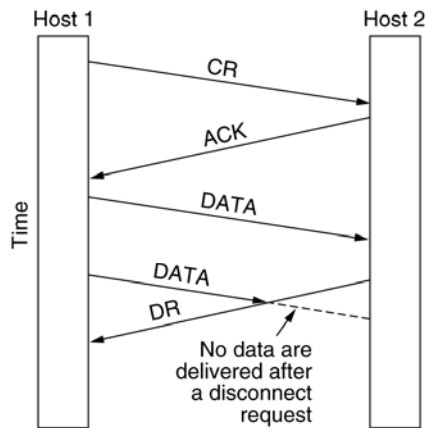
Delay-BW Product & Advertised Window Size

- Suppose RTT=100 ms, R=2.4 Gbps
  - # bits in pipe = 3 Mbytes
- If single TCP process occupies pipe, then required advertised window size is
  - RTT x Bit rate = 3 Mbytes
  - Normal maximum window size is 65535 bytes
- Solution: Window Scale Option
  - Window size up to 65535 x 2^{14} = 1 Gbyte allowed
  - Requested in SYN segment

Where the information can be filled in?

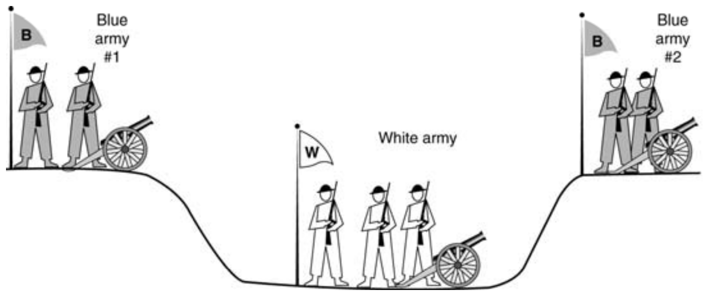
Connection Release

- Symmetric release vs. asymmetric release



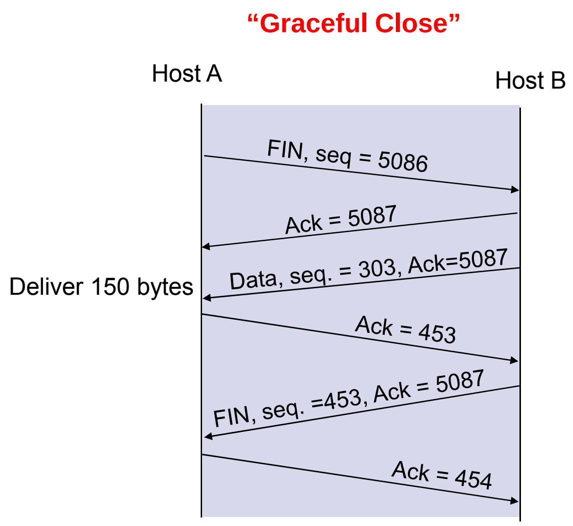
Abrupt asymmetric disconnection with loss of data.

Connection Release (2)



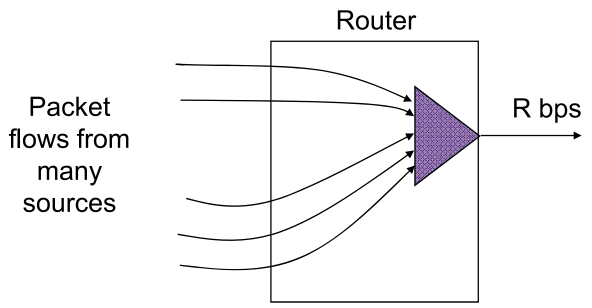
The two-army problem.

TCP Connection Closing



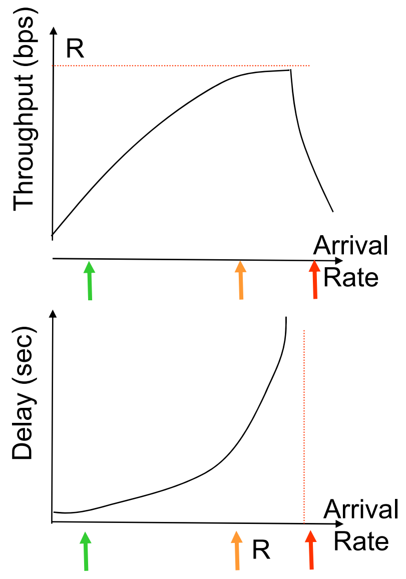
TCP Congestion Control

- *Advertised window* size is used to ensure that receiver's buffer will not overflow
- However, buffers at intermediate routers between source and destination may overflow



- Congestion occurs when total arrival rate from all packet flows exceeds R over a sustained period of time
- Buffers at multiplexer will fill and packets will be lost

Phases of Congestion Behavior



1. Light traffic
   - Arrival Rate << R
   - Low delay
   - Can accommodate more
2. Knee (congestion onset)
   - Arrival rate approaches R
   - Delay increases rapidly
   - Throughput begins to saturate
3. Congestion collapse
   - Arrival rate > R
   - Large delays, packet loss
   - Useful application throughput drops

Congestion Window

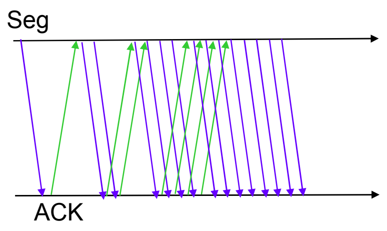
- Desired operating point: just before knee
  - Sources must control their sending rates so that aggregate arrival rate is just before knee
- TCP sender maintains a *congestion window* (cwnd) to control congestion at intermediate routers
- Effective window is minimum of congestion window and advertised window
- Problem: source does not know what its “fair” share of available bandwidth should be
- Solution: adapt dynamically to available BW
  - Sources probe the network by increasing cwnd
  - When congestion detected, sources reduce rate
  - Ideally, sources sending rate stabilizes near ideal point

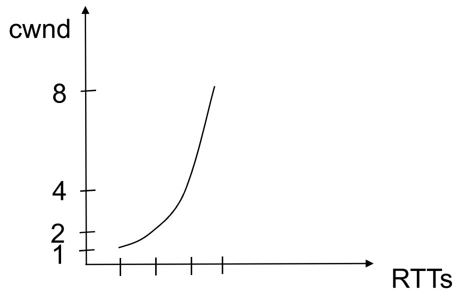
Congestion Window (Cont.)

- How does the TCP congestion algorithm change congestion window dynamically according to the most up-to-date state of the network?
- At light traffic: each segment is ACKed quickly
  - Increase cwnd aggresively
- At knee: segment ACKs arrive, but more slowly
  - Slow down increase in cwnd
- At congestion: segments encounter large delays (so retransmission timeouts occur); segments are dropped in router buffers (resulting in duplicate ACKs)
  - Reduce transmission rate, then probe again

TCP Congestion Control: Slow Start

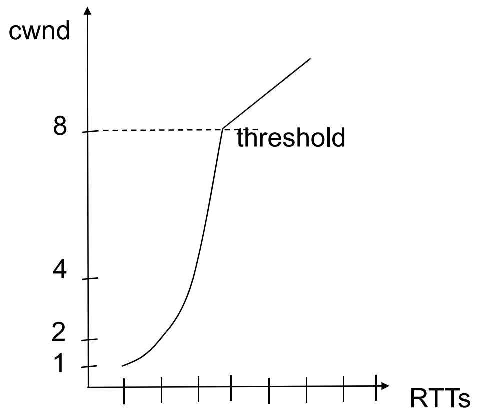
- **Slow start**: increase congestion window size by one segment upon receiving an ACK from receiver
  - initialized at ≤ 2 segments
  - used at (re)start of data transfer
  - congestion window increases exponentially



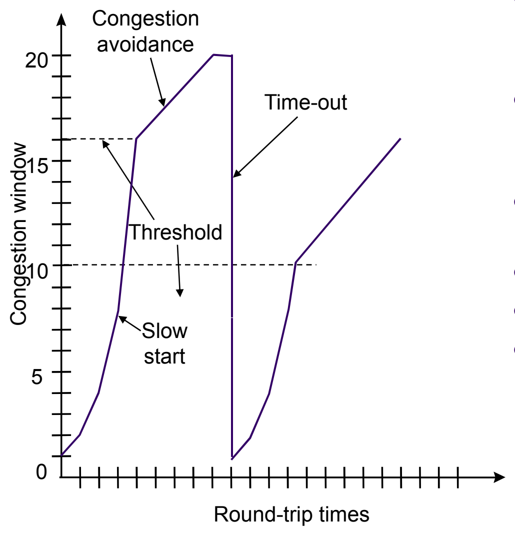


TCP Congestion Control: Congestion Avoidance

- Algorithm progressively sets a *congestion threshold*
  - When cwnd > threshold, slow down rate at which cwnd is increased
- Increase congestion window size by one segment per round-trip-time (RTT)
  - Each time an ACK arrives, cwnd is increased by 1/cwnd
  - In one RTT, cwnd segments are sent, so total increase in cwnd is cwnd x 1/cwnd = 1
  - cwnd grows linearly with time



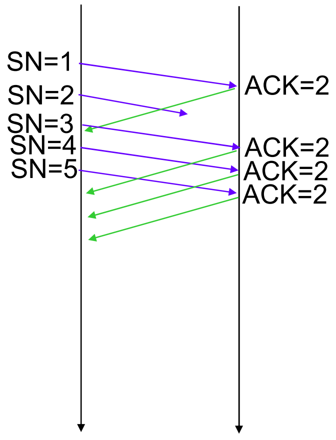
TCP Congestion Control: Congestion



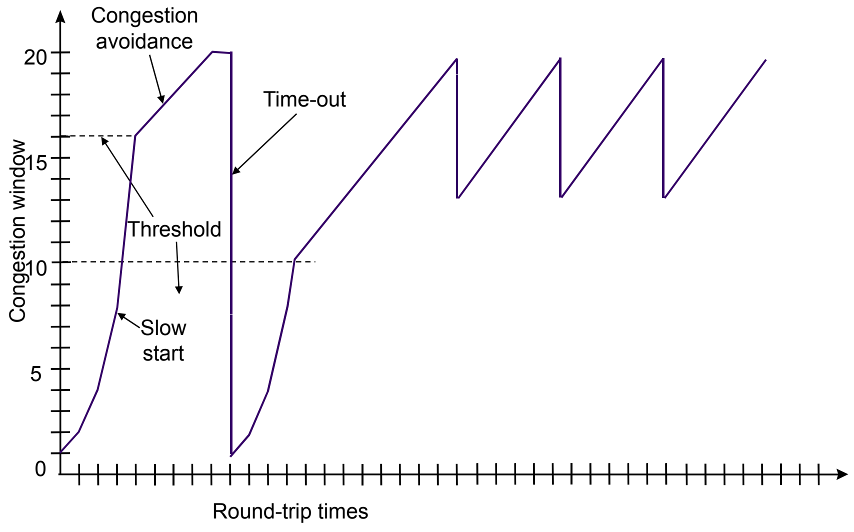
- Congestion is detected upon timeout or receipt of duplicate ACKs
- Assume current cwnd corresponds to available bandwidth
- Adjust congestion threshold = ½ x current cwnd
- Reset cwnd to 1
- Go back to slow-start
- Over several cycles expect to converge to congestion threshold equal to about ½ the available bandwidth

Fast Retransmit & Fast Recovery

- Congestion causes many segments to be dropped
- If only a single segment is dropped, then subsequent segments trigger duplicate ACKs before timeout
- Can avoid large decrease in cwnd as follows:
  - When three duplicate ACKs arrive, retransmit lost segment immediately
  - Reset congestion threshold to ½ cwnd
  - Reset cwnd to congestion threshold + 3 to account for the three segments that triggered duplicate ACKs
  - Remain in congestion avoidance phase
  - However if timeout expires, reset cwnd to 1
  - In absence of timeouts, cwnd will oscillate around optimal value



TCP Congestion Control: Fast Retransmit & Fast Recovery



Chapter 8

Communication Networks and Services

Internet Routing Protocols

Outline

- Basic Routing
- Routing Information Protocol (RIP)
- Open Shortest Path First (OSPF)
- Border Gateway Protocol (BGP)

Host Behavior

- Every host must do IP forwarding
- For datagram generated by own higher layers
  - if destination connected through point-to-point link or on shared network, send datagram directly to destination
  - Else, send datagram to a default router
- For datagrams received on network interface
  - if destination address, own address, pass to higher layer
  - if destination address, not own, discard “silently”

Router Behavior

Router’s IP layer

- can receive datagrams from own higher layers
- can receive datagram from a network interface
  - if destination IP address own or broadcast address, pass to layer above
  - else, forward the datagram to next hop
- routing table determines handling of datagram

Routers exchange information using routing protocols to develop the routing tables

Forwarding Procedure

- Does routing table have entry that matches complete destination IP address? If so, use this entry to forward
- Else, does routing table have entry that matches the longest prefix of the destination IP address? If so, use this entry to forward
- Else, does the routing table have a default entry? If so, use this entry.
- Else, packet is undeliverable

Autonomous Systems

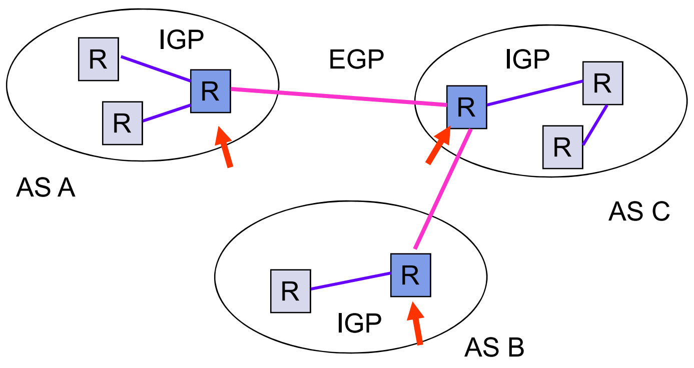
- Global Internet viewed as collection of autonomous systems.
- **Autonomous system (AS)** is a set of routers or networks administered by a single organization
- Same routing protocol need not be run within the AS
- But, to the outside world, an AS should present a *consistent picture of what ASs are reachable* through it
- **Stub AS:** has only a single connection to the outside world.
- **Multihomed AS:** has multiple connections to the outside world, but refuses to carry transit traffic
- **Transit AS:** has multiple connections to the outside world, and can carry transit and local traffic.

Inter and Intra Domain Routing

Interior Gateway Protocol (IGP): routing within AS
- RIP, OSPF

Exterior Gateway Protocol (EGP): routing between AS's
- BGPv4

Border Gateways perform IGP & EGP routing



Routing Information Protocol (RIP)

- RFC 1058
- **RIP** based on routed, "route d", distributed in BSD UNIX
- Uses the **distance-vector algorithm**
- Runs on top of UDP, port number 520
- Metric: number of hops
- Max limited to 15
  - suitable for small networks (local area environments)
  - value of 16 is reserved to represent infinity
  - small number limits the *count-to-infinity* problem

RIP Operation

- Router sends update message to neighbors every 30 sec
- A router expects to receive an update message from each of its neighbors within 180 seconds in the worst case
- If router does not receive update message from neighbor X within this limit, it assumes the link to X has failed and sets the corresponding minimum cost to 16 (infinity)
- Uses ***split horizon with poisoned reverse***
- Convergence speeded up by triggered updates
  - neighbors notified immediately of changes in distance vector table

RIP Protocol

- Routers run RIP in active mode (advertise distance vector tables)
- Hosts can run RIP in passive mode (update distance vector tables, but do not advertise)
- RIP datagrams broadcast over LANs & specifically addressed on pt-pt or multi-access non-broadcast nets
- Two RIP packet types:
  - *request* to ask neighbor for distance vector table
  - *response* to advertise distance vector table
    - periodically; in response to request; triggered

Key disadvantages of RIP protocol?

Flooding

- Used in OSPF to distribute link state (LS) information
- Forward incoming packet to all ports except where packet came in
- Packet eventually reaches destination as long as there is a path between the source and destination
- Generates exponential number of packet transmissions
- Approaches to limit # of transmissions:
  - Use a TTL at each packet; won't flood if TTL is reached
  - Each router adds its identifier to header of packet before it floods the packet; won't flood if its identifier is detected
  - Each packet from a given source is identified with a unique sequence number; won't flood if sequence number is same

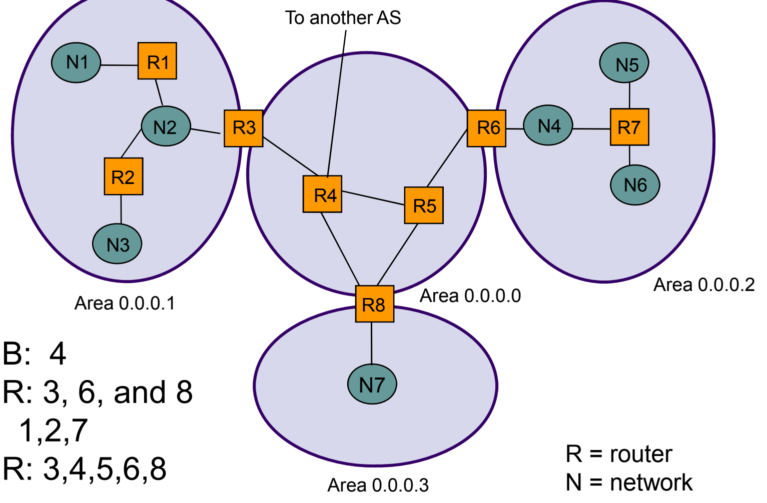
Open Shortest Path First

- RFC 2328 (v2)
- Fixes some of the deficiencies in RIP
- Enables each router to learn complete network topology
- Each router monitors the *link state* to each neighbor and floods the link-state information to other routers
- Each router builds an identical *link-state database*
- Allows router to build shortest path tree with router as root
- OSPF typically converges faster than RIP when there is a failure in the network

OSPF Network

- To improve scalability, AS may be partitioned into *areas*
  - Area is identified by 32-bit Area ID
  - Router in area only knows complete topology inside area & limits the flooding of link-state information to area
  - *Area border routers* summarize info from other areas
- Each area must be connected to *backbone area* (0.0.0.0)
  - Distributes routing info between areas
- *Internal router* has all links to nets within the same area
- *Area border router* has links to more than one area
- *backbone router* has links connected to the backbone
- *Autonomous system boundary (ASB) router* has links to another autonomous system.

OSPF Areas



ASB: 4
ABR: 3, 6, and 8
IR: 1,2,7
BBR: 3,4,5,6,8

R = router
N = network

Neighbor & Adjacent Routers

- *Neighbor routers*: two routers that have interfaces to a common network
  - Neighbors are discovered dynamically by *Hello protocol*
- Each neighbor of a router described by a state
  - down, attempt, init, 2-way, Ex-Start, Exchange, Loading, Full
- *Adjacent router*: neighbor routers become adjacent when they synchronize topology databases by exchange of link state information
  - Neighbors on point-to-point links become adjacent
  - Routers on multiaccess nets become adjacent only to *designated & backup designated routers*
    - Reduces size of topological database & routing traffic

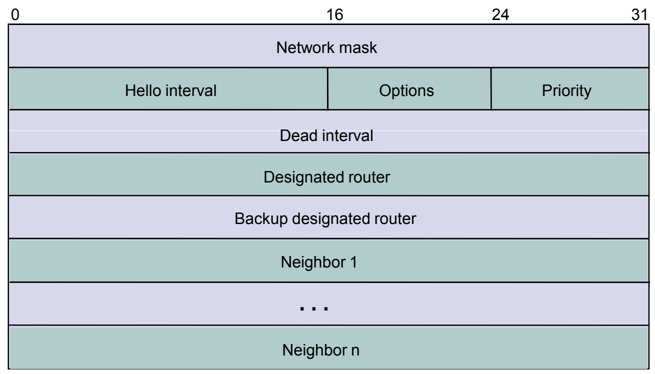
Designated Routers

- Reduces number of adjacencies
- Elected by each multi-access network after neighbor discovery by hello protocol
- Election based on priority & id fields
- Generates link advertisements that list routers attached to a multi-access network
- Forms adjacencies with routers on multi-access network
- Backup prepared to take over if designated router fails

OSPF Stages

1. Discover neighbors by sending Hello packets (every 10 sec) and designated router elected in multiaccess networks
2. Adjacencies are established & link state databases are synchronized
3. Link state information is propagated & routing tables are calculated

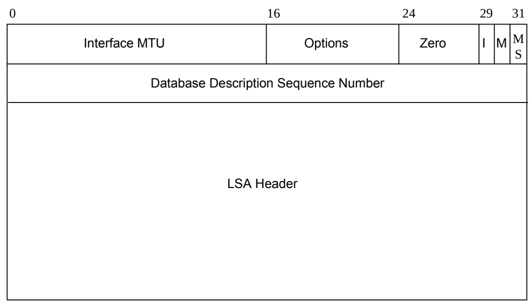
Stage 1: OSPF Hello Packet



- Send Hello packets to establish & maintain neighbor relationship

- Hello interval: number of seconds between Hello packets; 10
- Priority: used to elect designated router & backup
- Dead interval: # sec before declaring a non-responding neighbor down.
- Neighbor: the Router ID of each neighbor from whom Hello packets have recently been received

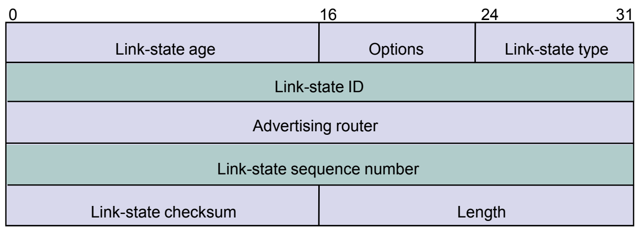
S2: OSPF Database Description



- Once neighbor routers become adjacent, they exchange database description packets to synchronize their link-state databases.

- Init bit 1 if pkt is first in sequence of database description packets
- More bit 1 if there are more database description packets to follow
- Master/Slave bit indicates that the router is the master.
- Link state ad (LSA) header describes state of router or network; contains info to uniquely identify entry in LSA (type, ID, and advertising router).
- Can have multiple LSA headers

LSA Header

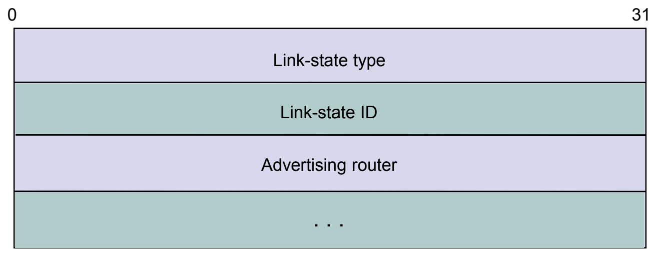


- LS type: Router LSAs generated by all OSPF routers; Network LSAs generated by designated routers; Summary LSAs by area border routers; AS-external LSAs by ASBRs
- LS id: identifies piece of routing domain being described by LSA
- LS Seq. Number: numbers LSAs to detect old/duplicate LSAs
- LS checksum: covers contents of LSA except link state age

Database Synchronization

- LS Database (LSDB): collection of the Link State Advertisements (LSAs) accepted at a node.
  - This is the “map” for Dijkstra algorithm
- When the connection between two neighbors comes up, the routers must wait for their LSDBs to be synchronized.
  - Else routing loops and black holes due to inconsistency
- OSPF technique:
  - Source sends only LSA headers, then
  - Neighbor requests LSAs that are more recent
  - Those LSAs are sent over
  - After sync, the neighbors are said to be “fully adjacent”

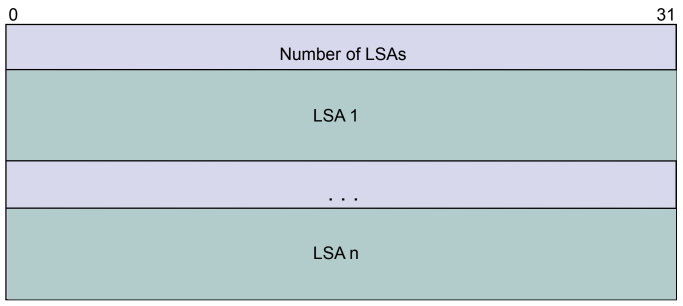
Stage 3: OSPF Link State Request



- Router sends a LS request packet to neighbor to update part of its link-state database
- Each LSA request is specified by the link state type, link state ID, and the advertising router.

Estimate the size of the Hello messages and the bandwidth consumed.

OSPF Link State Update



- In response to LS request or trigger, router will send new LS info using the LS update message
- Contents are composed of link state advertisements (LSAs)
- LS update message is acknowledged using LS ack pkt to ensure that the flooding algorithm is reliable; Link state acknowledgement packets consist of a list of LSA headers.

Exterior Gateway Protocols

- Within each AS, there is a consistent set of routes connecting the constituent networks
- The Internet is woven into a coherent whole by *Exterior Gateway Protocols (EGPs)* that operate between AS's
- EGP enables two AS's to exchange routing information about:
  - The networks that are contained within each AS
  - The AS's that can be reached through each AS
- EGP path selection guided by policy rather than path optimality
  - Trust, peering arrangements, etc

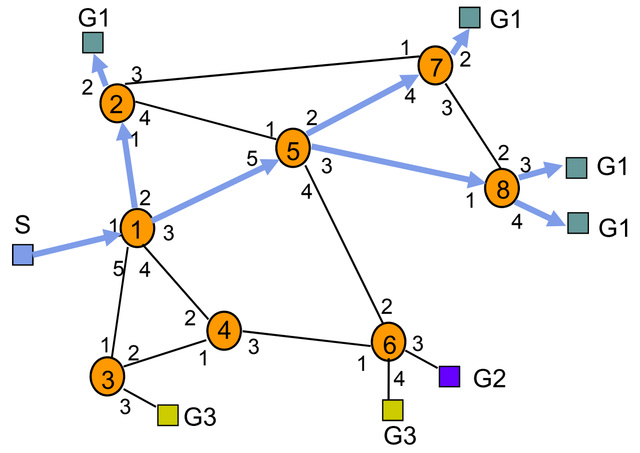
Chapter 8
Communication Networks and Services

Multicast Routing

Multicast/Broadcast Routing

- Broadcast: send a message to all (in a group) simultaneously!
  - how about the source sends a distinct message to each destination as Point-to-Point?
  - how about flooding?
  - Multi-destination routing: each message contains a list of destinations
  - Sink tree, or *spanning tree*, for directing routing
    - Excellent bandwidth utilization: minimal # of packets
    - Requiring knowledge of tree at each router

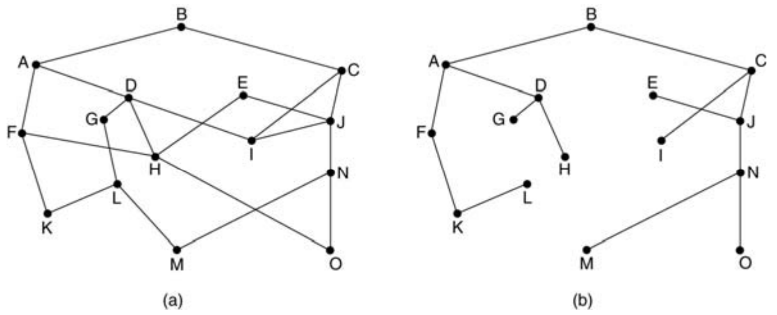
Multicasting



- Source S sends packets to multicast group G1

Reverse Path Forwarding and Sink Tree

- How many packets are generated by a broadcast from B, using
  - reverse path forwarding
  - the sink tree.



(a) A subnet. (b) A *sink tree* for router B.

Reverse-Path Forwarding/ Broadcasting (RPB)

- Fact: Set of shortest paths to the source node S forms a tree that spans the network
  - Approach: Follow paths in *reverse* direction
- Assume each router knows current shortest path to S
  - Upon receipt of a multicast packet, router records the packet's source address and the port it arrives on
  - If shortest path to source is through same port ("parent port"), router forwards the packet to all other ports
  - Else, drops the packet
- Loops are suppressed; each packet forwarded a router exactly once
- Implicitly assume shortest path to source S is same as shortest path from source
  - If paths asymmetric, need to use link state info to compute shortest paths from S

Internet Group Management Protocol (IGMP)

- *Internet Group Management Protocol:*
  - Host can join a multicast group by sending an IGMP message to its router
- Each multicast router periodically sends an IGMP query message to check whether there are hosts belonging to multicast groups
  - Hosts respond with list of multicast groups they belong to
  - Hosts randomize response time; cancel response if other hosts reply with same membership
- Routers determine which multicast groups are associated with a certain port
- Routers only forward packets on ports that have hosts belonging to the multicast group

DHCP

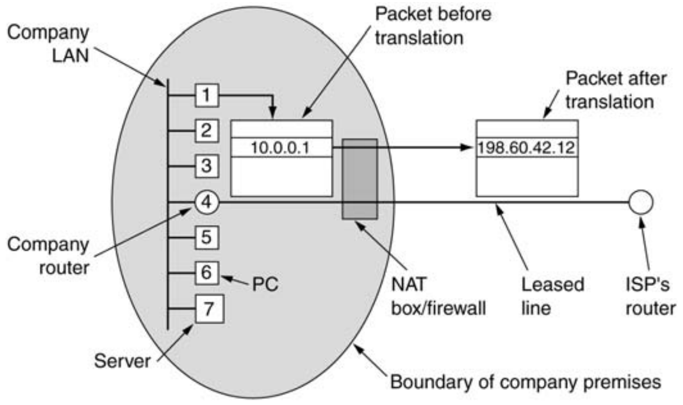
- Dynamic Host Configuration Protocol (RFC 2131)
- BOOTP (RFC 951, 1542) allows a diskless workstation to be remotely booted up in a network
  - UDP port 67 (server) & port 68 (client)
- DHCP builds on BOOTP to allow servers to deliver configuration information to a host
  - Used extensively to assign temporary IP addresses to hosts
  - Allows ISP to maximize usage of their limited IP addresses

Network Address Translation (NAT)

- Class A, B, and C addresses have been set aside for use within private internets
  - Packets with private ("unregistered") addresses are discarded by routers in the global Internet
- NAT (RFC 1631): method for mapping packets from hosts in private internets into packets that can traverse the Internet
  - A device (computer, router, firewall) acts as an agent between a private network and a public network
  - A number of hosts can share a limited number of registered IP addresses
    - Static/Dynamic NAT: map unregistered addresses to registered addresses
    - Overloading: maps multiple unregistered addresses into a single registered address (e.g. Home LAN)

NAT – Network Address Translation

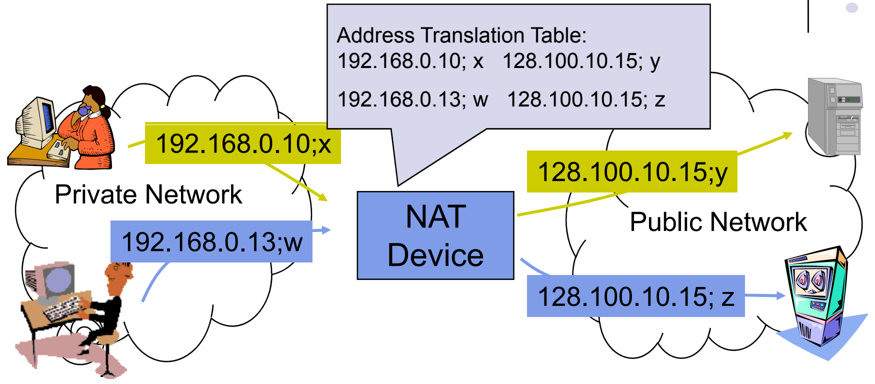
- NAT: pubic IP addresses and private IP addresses



Placement and operation of a NAT box.

How to translate when the reply comes back? What are its problems?

NAT Operation (Overloading)

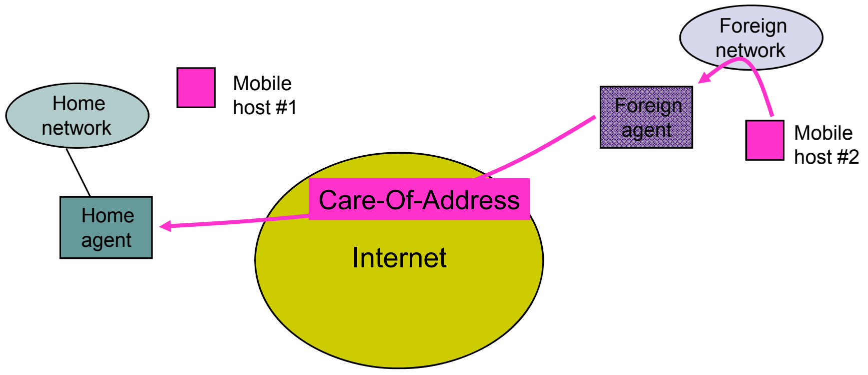


- Hosts inside private networks generate packets with private IP address & TCP/UDP port #s
- NAT maps each private IP address & port # into shared global IP address & available port #
- Translation table allows packets to be routed unambiguously

Mobile IP

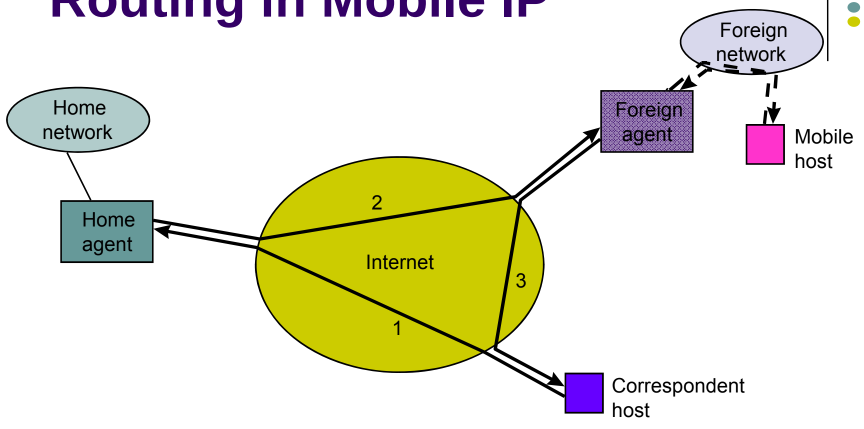
- Proliferation of mobile devices: PDAs, laptops, cellphones, …
- As user moves, point-of-attachment to network necessarily changes
- Problem: IP address specifies point-of-attachment to Internet
  - Changing IP address involves terminating all connections & sessions
- *Mobile IP (RFC 2002)*: device can change point-of-attachment while retaining IP address and maintaining communications

Routing in Mobile IP



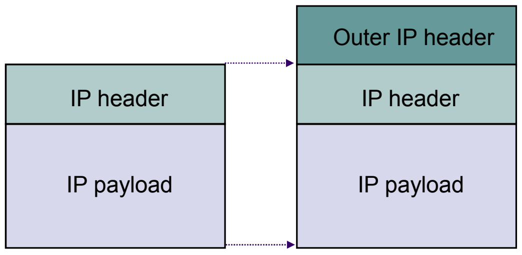
- Home Agent (HA) keeps track of location of each Mobile Host (MH) in its network; HA periodically announces its presence
- If an MH is in home network, e.g. MH#1, HA forwards packets directly to MH
- When an MH moves to a Foreign network, e.g. MH#2, MH obtains a care-of-address from foreign agent (FA) and registers this new address with its HA

Routing in Mobile IP



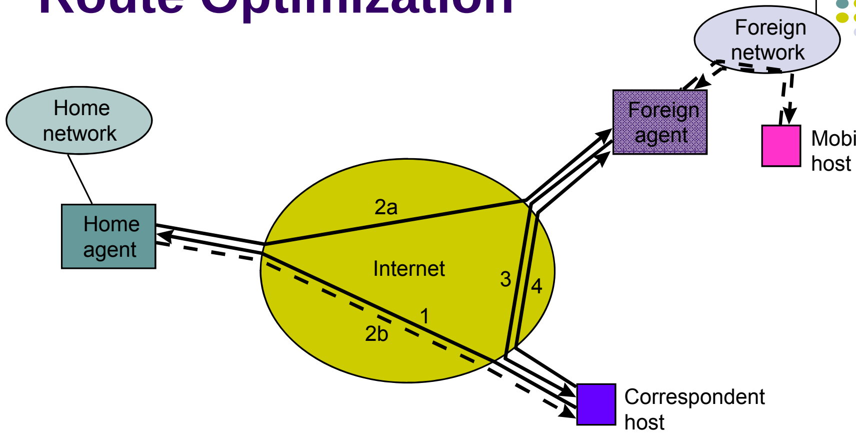
- Correspondent Host (CH) sends packets as usual (1)
- Packets are intercepted by HA which then forwards to Foreign Agent (FA) (2)
- FA forwards packets to the MH
- MH sends packet to CH as usual (3)
- How does HA send packets to MH in foreign network?

IP-to-IP Encapsulation



- HA uses IP-to-IP encapsulation
- IP packet has MH IP address
- Outer IP header has HA's address as source address and care-of-address as destination address
- FA recovers IP packet and delivers to MH

Route Optimization



- Going to HA inefficient if CH and MH are in same foreign network
- When HA receives pkt from CH (1), it tunnels using care-of-address (2a); HA also sends care-of-address to CH (2b)
- CH can then send packets directly to care-of-address (4)

Chapter 1 Communication Networks and Services

Future Network Architectures and Services

144

Trends in Network Evolution

- It’s all about services
  - Building networks involves huge expenditures
  - Services that generate revenues drive the network architecture
- Current trends
  - Packet switching vs. circuit switching
  - Multimedia applications
  - More versatile signaling
  - End of trust
  - Many service providers and overlay networks
  - Networking *is* a business

145

Packet vs. Circuit Switching

- Architectures appear and disappear over time
  - Telegraph (message switching)
  - Telephone (circuit switching)
  - Internet (packet switching)
  - Commonness and differences
- Trend towards packet switching at the edge
  - IP enables rapid introduction of new applications
  - New cellular voice networks packet-based
  - IP will support *real-time* voice and telephone network will gradually be replaced
  - However, large packet flows easier to manage by circuit-like methods

146

Optical Circuit Switching

- Optical signal transmission over fiber can carry huge volumes of information (Tbps)
- Optical signal processing very limited
  - Optical logic circuits bulky and costly
  - Optical packet switching will not happen soon
- Optical-to-Electronic conversion is expensive
  - Maximum electronic speeds << Tbps
  - Parallel electronic processing & high expense
- Thus, trend towards optical circuit switching in the core

147

Multimedia Applications

- Trend towards digitization of *all* media
- Digital voice standard in cell phones
- Music cassettes replaced by CDs and MP3's
- Digital cameras replacing photography
- Video: digital storage and transmission
  - Analog VCR cassettes largely replaced by DVDs
  - Analog broadcast TV to be replaced by digital TV
  - VCR cameras/recorders to be replaced by digital video recorders and cameras
- High-quality network-based multimedia applications now feasible

148

End of Trust

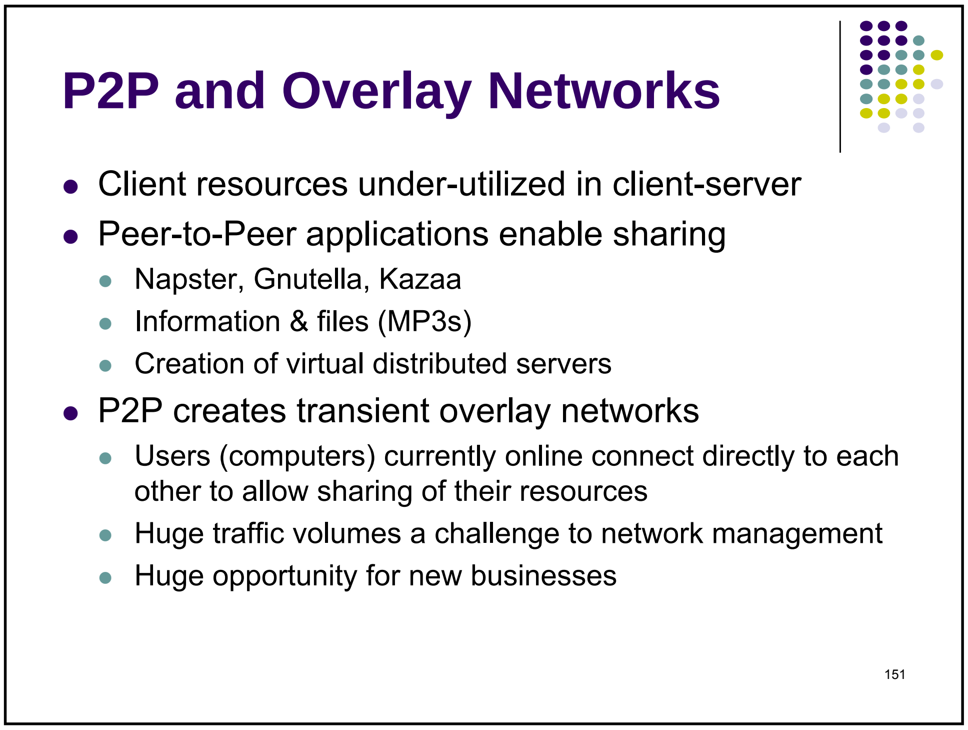
- Security Attacks
  - Spam
  - Denial of Service attacks
  - Viruses
  - Impersonators
- Firewalls & Filtering
  - Control flow of traffic/data from Internet
- Protocols for privacy, integrity and authentication

149

Servers & Services

- Many Internet applications involve interaction between client and server computers
  - Client and servers are at the edge of the Internet
  - SMTP, HTTP, DNS, …
- Enhanced services in telephone network also involve processing from servers
  - Caller ID, voice mail, mobility, roaming, . . .
  - These servers are inside the telephone network
  - Internet-based servers at the edge can provide same functionality
- In future, multiple service providers can coexist and serve the same customers

150



P2P and Overlay Networks

- Client resources under-utilized in client-server
- Peer-to-Peer applications enable sharing
  - Napster, Gnutella, Kazaa
  - Information & files (MP3s)
  - Creation of virtual distributed servers
- P2P creates transient overlay networks
  - Users (computers) currently online connect directly to each other to allow sharing of their resources
  - Huge traffic volumes a challenge to network management
  - Huge opportunity for new businesses

151