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

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

HTTP | SMTP | DNS | RTP

TCP/IP Protocol Suite

TCP: Reliable stream service

UDP: User datagram service

IP: (ICMP, ARP)

Network Interface 1

Network Interface 2

Network Interface 3

Diverse network technologies
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:
  \[
  \text{int}_1.\text{int}_2.\text{int}_3.\text{int}_4
  \]
  (int\(j\) = jth octet)
  128.100.10.13

DNS resolves IP name to IP address

Internet Protocol

- Provides best effort, connectionless packet delivery
  - motivated by need to keep routers simple and by adaptability 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

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>24</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **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.

#### Minimum 20 bytes
- Up to 40 bytes in options fields
**IP Packet Header**

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0</td>
</tr>
<tr>
<td>IHL</td>
<td>4</td>
</tr>
<tr>
<td>Type of Service</td>
<td>8</td>
</tr>
<tr>
<td>Total Length</td>
<td>16</td>
</tr>
<tr>
<td>Identification</td>
<td>19</td>
</tr>
<tr>
<td>Flags</td>
<td>24</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>31</td>
</tr>
<tr>
<td>Time to Live</td>
<td>0</td>
</tr>
<tr>
<td>Protocol</td>
<td>4</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>8</td>
</tr>
<tr>
<td>Source IP Address</td>
<td>16</td>
</tr>
<tr>
<td>Destination IP Address</td>
<td>24</td>
</tr>
<tr>
<td>Options</td>
<td>31</td>
</tr>
<tr>
<td>Padding</td>
<td>31</td>
</tr>
</tbody>
</table>

**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).

---

**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**

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>19</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>IHL</td>
<td>Type of Service</td>
<td>Total Length</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Fragment Offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time to Live</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source IP Address

Destination IP Address

Options

Padding

**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.

**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.
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, $b_0, b_1, b_2, \ldots, b_{L-1}$
- The algorithm appends a 16-bit checksum $b_L$

Checksum Calculation

The checksum $b_L$ is calculated as follows:
- Treating each 16-bit word as an integer, find $x = b_0 + b_1 + b_2 + \ldots + b_{L-1}$ modulo $2^{15-1}$
- The checksum is then given by: $b_L = -x$ 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: $0 = b_0 + b_1 + b_2 + \ldots + b_{L-1} + b_L$ 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:
  \[ \text{int1} \cdot \text{int2} \cdot \text{int3} \cdot \text{int4} \quad \text{where intj} = \text{integer value of jth octet} \]
  IP address of 10000000 10000111 01000100 00000101 is 128.135.68.5 in dotted-decimal notation
### Classful Addresses

#### Class A
- **24 bits**
- **7 bits**

<table>
<thead>
<tr>
<th>netid</th>
<th>hostid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

- 126 networks with up to 16 million hosts
- **1.0.0.0** to **127.255.255.255**

#### Class B
- **16 bits**
- **14 bits**

<table>
<thead>
<tr>
<th>netid</th>
<th>hostid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>

- 16,382 networks with up to 64,000 hosts
- **128.0.0.0** to **191.255.255.255**

#### Class C
- **8 bits**
- **22 bits**

<table>
<thead>
<tr>
<th>netid</th>
<th>hostid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0</td>
<td></td>
</tr>
</tbody>
</table>

- 2 million networks with up to 254 hosts
- **192.0.0.0** to **223.255.255.255**

---

### 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

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

<table>
<thead>
<tr>
<th></th>
<th>Net ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subnetted address

<table>
<thead>
<tr>
<th></th>
<th>Net ID</th>
<th>Subnet ID</th>
<th>Host ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Subnetting Example

- 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 00000110 10000000
    - Subnet = 150.100.12.128
- Subnet address used by routers within organization
**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

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next-Hop</th>
<th>Flags</th>
<th>Net I/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>$H$</td>
<td>lo0</td>
</tr>
<tr>
<td>150.100.12.128</td>
<td>150.100.12.129</td>
<td></td>
<td>emd0</td>
</tr>
<tr>
<td>150.100.12.0</td>
<td>150.100.12.4</td>
<td>$G$</td>
<td>emd1</td>
</tr>
<tr>
<td>150.100.15.0</td>
<td>150.100.12.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example 2: Host H5 sends packet to host H2

Routing Table at H5

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next-Hop</th>
<th>Flags</th>
<th>Net I/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>H</td>
<td>lo0</td>
</tr>
<tr>
<td>default</td>
<td>150.100.15.54</td>
<td>G</td>
<td>emd0</td>
</tr>
<tr>
<td>150.100.15.0</td>
<td>150.100.15.11</td>
<td></td>
<td>emd0</td>
</tr>
</tbody>
</table>

Example: Host H5 sends packet to host H2

Routing Table at R2

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next-Hop</th>
<th>Flags</th>
<th>Net I/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>H</td>
<td>lo0</td>
</tr>
<tr>
<td>default</td>
<td>150.100.12.4</td>
<td>G</td>
<td>emd0</td>
</tr>
<tr>
<td>150.100.15.0</td>
<td>150.100.15.54</td>
<td></td>
<td>emd1</td>
</tr>
<tr>
<td>150.100.12.0</td>
<td>150.100.12.1</td>
<td></td>
<td>emd0</td>
</tr>
</tbody>
</table>
Example: Host H5 sends packet to host H2

Routing Table at R1

<table>
<thead>
<tr>
<th>Destination</th>
<th>Next-Hop</th>
<th>Flags</th>
<th>Net I/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>H</td>
<td>lo0</td>
</tr>
<tr>
<td>150.100.12.128</td>
<td>150.100.12.129</td>
<td>emd0</td>
<td></td>
</tr>
<tr>
<td>150.100.12.0</td>
<td>150.100.12.4</td>
<td>emd1</td>
<td></td>
</tr>
<tr>
<td>150.100.15.0</td>
<td>150.100.12.1</td>
<td>G</td>
<td>emd1</td>
</tr>
</tbody>
</table>

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
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

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
CIDR (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 0001 0000
  - i.e. 150.158.16.0
  - Up to 10010110 10011110 00011111 00000000
  - i.e. 150.158.31.0

CIDR (Supernetting-2)

- A router has the following CIDR entries in its routing table:
  
<table>
<thead>
<tr>
<th>Address/mask</th>
<th>Next hop</th>
</tr>
</thead>
<tbody>
<tr>
<td>128.56.24.0/22</td>
<td>Interface 0</td>
</tr>
<tr>
<td>128.56.60.0/22</td>
<td>Interface 1</td>
</tr>
<tr>
<td>default</td>
<td>Router 2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Address Requirement</th>
<th>Address Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 256</td>
<td>1 Class C</td>
</tr>
<tr>
<td>256&lt;,&lt;512</td>
<td>2 Class C</td>
</tr>
<tr>
<td>512&lt;,&lt;1024</td>
<td>4 Class C</td>
</tr>
<tr>
<td>1024&lt;,&lt;2048</td>
<td>8 Class C</td>
</tr>
<tr>
<td>2048&lt;,&lt;4096</td>
<td>16 Class C</td>
</tr>
<tr>
<td>4096&lt;,&lt;8192</td>
<td>32 Class C</td>
</tr>
<tr>
<td>8192&lt;,&lt;16384</td>
<td>64 Class C</td>
</tr>
</tbody>
</table>

Hierarchical Routing & Table Efficiency

(a) and (b) diagrams showing hierarchical routing and table efficiency with different prefix lengths and allocation classes.
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 1 table 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 diagram]

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

![ARP response diagram]

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

Q1: who does it?  
Q2: penalty?  
Q3: Does it make sense to do reassembly at intermediate routers? Why?
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.
**Internet Control Message Protocol (ICMP)**

- RFC 792; Encapsulated in IP packet (protocol 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”

---

**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 inside 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**

<table>
<thead>
<tr>
<th>Source Port</th>
<th>Destination Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>31</td>
</tr>
</tbody>
</table>

- 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

**0-255**
- Well-known ports

**256-1023**
- Less well-known ports

**1024-65536**
- Ephemeral client ports

**UDP Checksum Calculation**

<table>
<thead>
<tr>
<th>Source IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Destination IP Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0 0 0 0 0 0 0 0</th>
<th>Protocol = 17</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UDP Length</td>
</tr>
</tbody>
</table>

- 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
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

![Diagram of data transfer and error detection with buffer and ACKS, sequence #]
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

![Diagram of Flow Control]

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

![Diagram of Congestion Control]
### 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 Multiplexing Diagram](image)

### TCP Segment Format

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>10</th>
<th>16</th>
<th>24</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source port</td>
<td>Destination port</td>
<td>Sequence number</td>
<td>Acknowledgment number</td>
<td>Window size</td>
<td>Checksum</td>
</tr>
</tbody>
</table>

- 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 selected during connection setup

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

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 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 specifies 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
- $2^n > 2 \times \text{max packet life} \times R \text{ bytes/second}$

Three-way handshaking

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.
Client-Server Application

Host A (client)  Host B (server)

- socket
- connect (blocks)
- connect returns
- write (blocks)
- read (blocks)
- read returns

SYN, Seq_no = x
SYN, Seq_no = y, ACK, Ack_no = x+1
Seq_no = x+1, ACK, Ack_no = y+1
Request message
Reply message

TCP Window Flow Control

Host A  Host B

1024 bytes to transmit
 Seq_no = 1, Ack_no = 2000, Win = 2048, No Data
0

1024 bytes to transmit
 Seq_no = 2000, Ack_no = 1, Win = 1024, Data = 2000-3023
1

1024 bytes to transmit
 Seq_no = 3024, Ack_no = 1, Win = 1024, Data = 3024-4047
2

128 bytes to transmit
 Why delay here?

1024 bytes to transmit
 Seq_no = 1, Ack_no = 4048, Win = 512, Data = 4048-4559
3

can only send 512 bytes

1024 bytes to transmit
 Seq_no = 4048, Ack_no = 129, Win = 1024, Data = 4048-4559
4
Sequence Number Wraparound

- $2^{32} = 4.29 \times 10^9$ bytes = $34.3 \times 10^9$ 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 $\rightarrow$ 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 $65535 \times 2^{14} = 1$ 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.

TCP Connection Closing

“Graceful Close”
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.

![Router diagram](image)

- 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

![Graph showing slow start](image)

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

![Graph showing congestion avoidance](image)
TCP Congestion Control:
Congestion

- Congestion is detected upon timeout or receipt of duplicate ACKs
- Assume current cwnd corresponds to available bandwidth
- Adjust congestion threshold
  \[ = \frac{1}{2} \times \text{current cwnd} \]
- Reset cwnd to 1
- Go back to slow-start
- Over several cycles expect to converge to congestion threshold equal to about \( \frac{1}{2} \) 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 \( \frac{1}{2} \) 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

- Congestion avoidance
- Time-out
- Threshold
- Slow start

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

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: public 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
Chapter 1
Communication Networks and Services

Future Network Architectures and Services

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
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

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
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

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