Revision notes - CS2105

Ma Hongqiang

October 24, 2017

Contents

1 Introduction

1.1 What is the Internet?

Definition 1.1 (Internet)**.**
The *Internet* is a network of connected computing devices known as **hosts** or **end systems**. Hosts run **network applications**.

Examples of hosts include PC, laptops, servers and smartphones.
Examples of network applications include Web, Pokemon Go, Skype.

1.2 Network Edge

Hosts access the Internet through access network. Hosts connect to the access network over different physical media.

- **Guided media**: signals propagate in solid media, e.g. fiber
- **Unguided media**: signals propagate freely, e.g., Wi-Fi, cellular

1.3 Network Core

Network core consists of a mesh of interconnected routers. The data transmit through net via either **circuit switching** or **packet switching**.

Definition 1.2 (Circuit Switching)**.**
In **circuit switching**, end-end resources allocated to and reserved for "call" between source and destination.

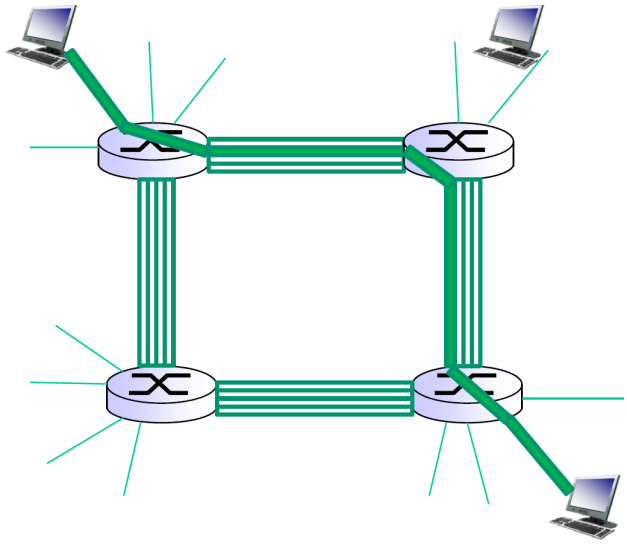


Figure 1: In above diagram, each link has four circuits. A call gets 2nd circuit in top link and 1st circuit in right link.

Some properties of **circuit switching** include

- call setup required
- circuit-like (guaranteed) performance
- circuit segment idle if not used by call (no sharing)
- *commonly used in traditional telephone networks*
- divide link bandwidth into "pieces"
  - frequency division
  - time division

Definition 1.3 (Packet Switching)**.**
In **packet switching**, host sending function by

- breaking application message into smaller chunks known as **packets**, of length L bits; and
- transmitting packets onto the link at **trasmission rate**[1] R bits/sec.

Definition 1.4 (Packet Trasmission Delay)**.**
Packet transmission delay, d_{trans} is defined as time nedded to transmit L-bit packet onto link.

$$d_{\text{trans}} = \frac{L}{R}$$

Packet switching adopts **store-and-forward** approach when a *router* is encountered.

- Packets are passed from one *router* to the next, across links on path from source to destination.
- **Store and forward**: entire packet must arrive at a router before it can be transmitted on the next link.

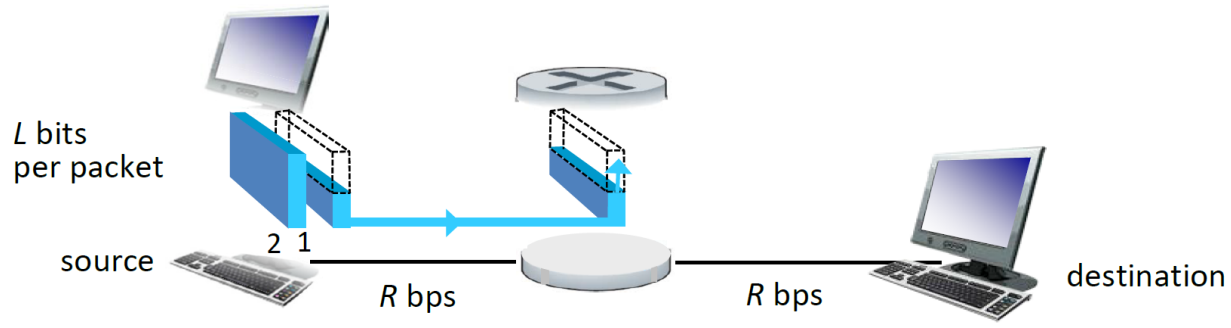


Figure 2: End to end $d_{\text{trans}} = 2 \times \frac{L}{R}$, assuming no other delays

[1]link transmission rate is also known as **link capacity** or **link bandwidth**

Definition 1.5 (Routing and Addressing)**.**
Routers determines source-destination route taken by packets by routing algorithms.
Addressing is achieved by each packet carrying source and destinaton information.

Some properties of packet switching include

- The Internet **is** a packet switching network.
- User A, B, ...'s packets *share* network resources
- Resources are used on demand.
- Excessive congestion is possible.[2]

Definition 1.6 (Internet Structure: Network of Networks)**.**

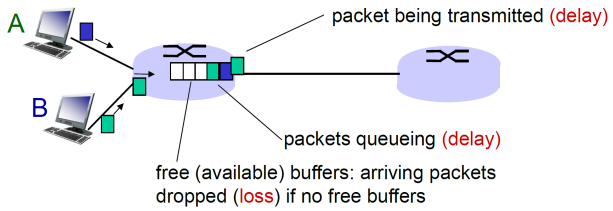
- Hosts connect to Internet via access **ISPs**(Internet Service Providers)
- Access ISPs in turn must be interconnected.
- Resulting network of networks is very complex.
  Evolution was driven by *economics* and *national policies*.
- Therefore, the Internet is a "network-of-networks", organised into autonomous systems(AS), each is owned by an organisation.

1.4 Delay, Loss and Throughput in Networks

Recall, the algorithm of sneding a packet in a packet switching network is

1. Sender transmit a packet onto the link as a sequence of bits.
2. Bits are propagated to the next node(e.g. a router) on the link.
3. Router stores, processes and forwards the packet to the next link.
4. Steps 2 and 3 repeat till the packet arrives at the receiver.

Delay and Loss could occur as packets need to queue in router buffers, waiting for turn to be send out one by one.



[2]These properties suggest bandwidth division into "pieces", dedicated allocation and resource reservation are not available.

- Queue(aka **buffer**) of a router has finite capacity.
- Packet arriving to full queue will be dropped(aka lost).
- Lost packet may be retransmitted by previous node, by source host, or not at all.

Definition 1.7 (Four Sources of End-to-end Packet Delay)**.**
End-to-end packet delay is the time taken for a packet to travel from source to destination.
It consists of:

- transmission delay
- propagation delay
- processing delay
- queueing delay

d_{proc}: nodal processing delay	d_{queue}: queueing delay	d_{trans}: transmission delay	d_{prop}: propagation delay
• check bit errors • determine output link • typically $<$ msec	• time waiting in the queue for transmission • dependes on congestion level of router	• L: packet length • R: link bandwidth • $d_{\text{trans}} = \frac{L}{R}$	• d: length of physical link • s: propagation speed in medium ($\sim 2 \times 10^8$m/sec) • $d_{\text{prop}} = \frac{d}{s}$

Definition 1.8 (Throughput)**.**
Throughput is defined as how many bits can be transmitted per unit time.

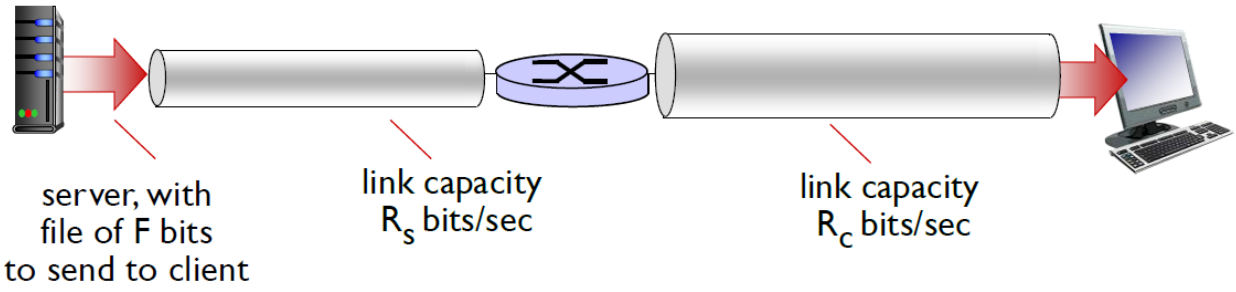


Figure 3: The throughput is $\frac{F}{\left(\frac{F}{R_s}+\frac{F}{R_c}\right)} = \frac{R_s R_c}{R_s+R_c}$

The difference between throughput and link capacity is that

- Throughput is measured for end-to-end communication.
- Link capacity(bandwidth) is meant for a specific link.

1.5 Protocol Layers and Service Models

The Internet supports various network applications. Network applications exchange messages and communicate among peers according to **protocols**.

Definition 1.9 (Protocol)**.**
Protocols define *format* and *order* of messages exchanged and the *actions* taken after messages are sent or received.

Protocols in the Internet are logically organised into "layers" according to their purposes.

- Each layer provides a service.
- Simple interfaces between layers
- Hide details from each other

Definition 1.10 (Internet Protocol Stack)**.**
The Internet consists of five layers.

- **Application**: supporting network applications, e.g. FTP, SMTP, HTTP
- **Transport**: process-to-process data transfer, e.g. TCP, UDP
- **Network**: routing of datagrams from source to destination, e.g. IP
- **Link**: data transfer between neighbouring network elements,e.g. Ethernet, 802.11
- **Physical**: bits "on the wire"

2 Application Layer

2.1 Principle of Network Applications

Application architectures refer to possible structures of network applications, which include:

- Client-server
- Peer-to-peer(P2P)
- Hybrid of client-server and P2P

Definition 2.1 (Client-Server Architecture)**.**
In **client-server** architecture,
Server:

- Waits for incoming requests
- Provides requested service to client

Client:

- Initiates contact with server("speak first")
- Typically requests service from server
- For Web, client is usually implemented in browser

Definition 2.2 (P2P Architecture)**.**
In **P2P** Architecture,

- *No* always-on server
- Arbitrary end systems directly communicate.
- Peers request service from other peers, provide service in return to other peers.

P2P Architecture is highly scalable but difficult to manage.

An example of Hybrid of Client-Server and P2P architecture is the instant messaging, where

- Chatting between two users is **P2P**
- Presence detection/location is **centralised**

Usually for applications, some considerations of *transport* service include

- Data integrity
- Throughput
- Timing

- Security

Definition 2.3 (Two Internet Transport Protocols)**.**
Two Internet Transport Protocols are `TCP` and `UDP`. The main difference is that `TCP` is reliable while `UDP` is not. In detail,
`TCP` service offers:

- **reliable transport** between sending and receiving pocess
- **flow control**: sender won't overwhelm receiver
- **congestion control**: throttle sender when network is overloaded

However, `TCP` does *not* provide: timing, minimum throughput guarantee, security.
In contrast, `UDP` service provide **unreliable** data transfer between sending and receiving process. It does *not* provide: reliability, flow control, congestion control, timing, throughput guarantee or security.

For **application layer protocol** which is over transport layer protocol, instead it defines

- Types of messages exchanged: e.g., request, response
- Message syntax: what field in messages and how fields are delineated
- Message semantics: meaning of information in fields
- Rules: for when and how applications send and respond to messages
- Open protocols: defined in RFCs; allow for interoperability; e.g., `HTTP`, `SMTP`
- Proprietary protocols, e.g., Skype

2.2 Web and `HTTP`

Definition 2.4 (Web Page)**.**
A **Web page** typically consists of:

- base `HTML` file, and
- several referenced *objects*
  An object can be HTML file, JPEG image etc.
  Each object is addressable by a **URL**,e.g.,

$$\underbrace{\texttt{www.comp.nus.edu.sg}}_{\text{host name}}\underbrace{\texttt{/ cs2105/img/doge.jpg}}_{\text{path name}}$$

Definition 2.5 (`HTTP`)**.**
`HTTP` refers to **Hypertext transfer protocol**.
`HTTP`

- is Web's application layer protocol
- usees client/server model
  - **client**: usually is browser that requests, receives and displays Web objects
  - **server**: Web server sends objects in response to requests

Note that `HTTP` uses `TCP` as transport service.

2.2.1 Non-persistent `HTTP`

In **non-persistent** `HTTP/1.0`,

1. Client initiates `TCP` connection to server.
2. Server accepts `TCP` connection request from client.
3. `HTTP` messages are exchanged between browser(`HTTP` client) and Web server(`HTTP` server) over `TCP` connection.
4. `TCP` connection closed.
5. `HTTP` client receives response message containing HTML file, displays HTML. Parsing HTML file, client notices referenced objects.
6. Step 1∼5 repeated for the referenced objects.

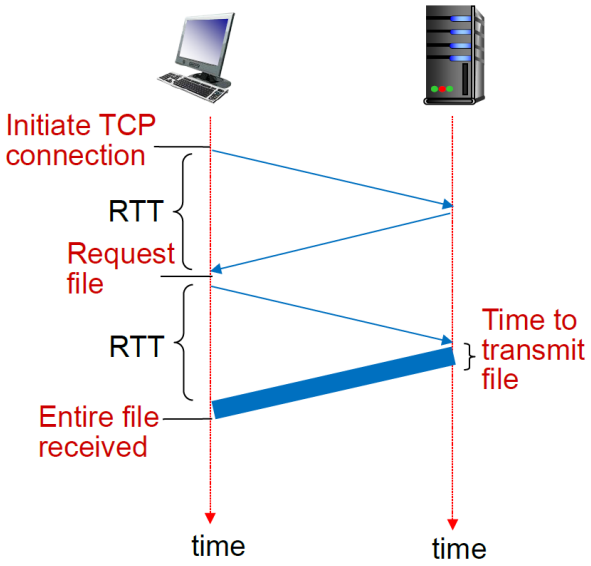
Definition 2.6 (Round Trip Time(RTT))**.**
Round trip time(RTT) refers to the time for a packet to travel from client to server and go back.

Clearly, the `HTTP` **response time** for one object transfer equals the sum of:

- one RTT to establish `TCP` connection
- one RTT for `HTTP` request and the first few bytes of `HTTP` response to return
- file transimission time

As a result, the **non-persistent** `HTTP` response time =

$$2 \times \text{RTT} + \text{file trasmission time}$$



.

2.2.2 Persistent HTTP

The problems of non-persistent HTTP include:

- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP/1.1 solves these issues by:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over the same TCP connection
- client sends requests as soon as it encounters a referenced object(*persistent with pipelining*)
- as little as one RTT for all the referenced objects

Example HTTP request message usually entails both **request** and **response**.
A HTTP **request** message looks like:

```
GET /~cs2105/demo.html HTTP/1.1
Host: www.comp.nus.edu.sg
User-Agent: Mozilla/5.0
Connection: close
\r\n
```

The 1st line is the request line. Subsequent 3 lines are header lines. Last extra blank line indicates end of header lines.
A HTTP **response** message looks like:

```
HTTP/1.1 200 OK
Date: Thu, 15 Jan 2015 13:02:41 GMT
Server: Apache/2.4.6 (Unix)
Content-Type: text/html
\r\n
data data data data ...
```

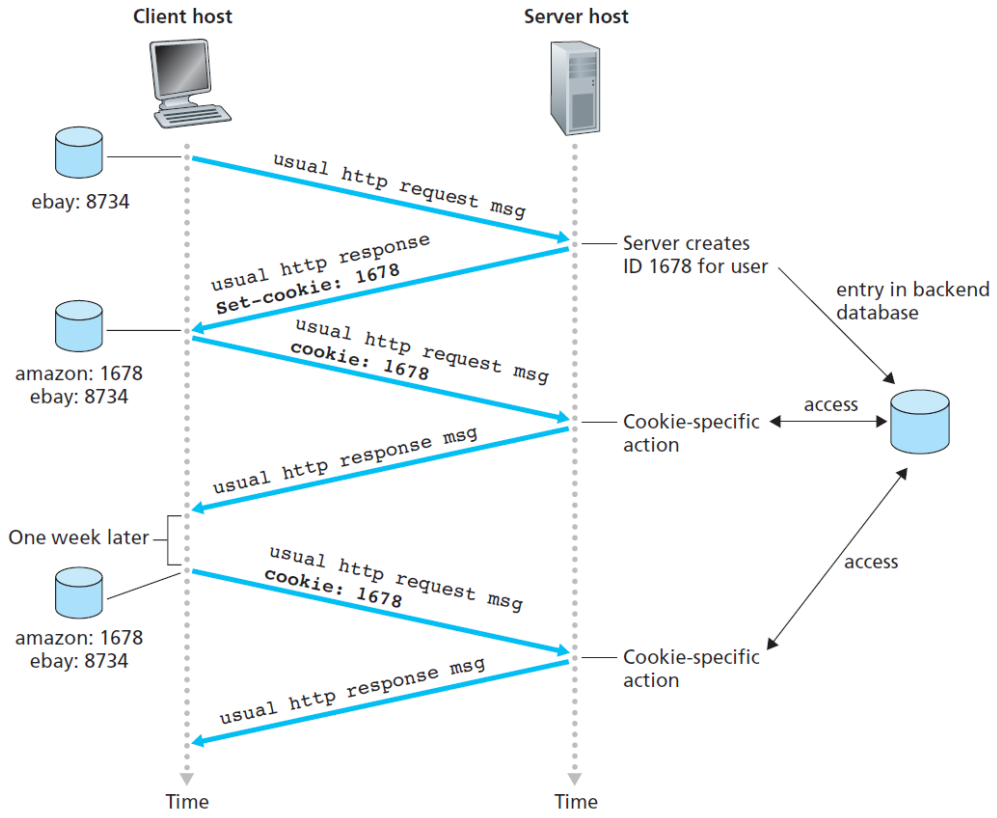
The 1st line is the protocol status line. Subsequent 3 lines are header lines. After the extra blank line is the data of the requested HTML file.

2.2.3 Cookies

HTTP is desisned to be **stateless**, in a sense that server maintains *no* information about past client requests.
However, sometimes it is good to maintain states(history) at server/client over multiple transactions.

Definition 2.7 (Cookies)**.**
Cookies allow sites to keep track of users.
Cookie technology has four components:

- A cookie header line in the HTTP response message
- A cookie header line in the HTTP request message
- A cookie file kept on the user's end system and managed by the user's browser
- A back-end database at the Web site



2.2.4 Conditional GET

The goal of conditional GET is to **not** send object if client cache has **up-to-date** cached version.

In the client cache, date of cached copy in `HTTP` request is specified by `If-modified-since: <date>`.
In the server response, no object will be contained in the response if cached copy is up-to-date, and return `HTTP/1.0 304 Not Modified`.

2.3 Domain Name System

There are two ways to identify a host:

1. **Hostname**, e.g., www.comp.nus.edu.sg
2. `IP` **address**, e.g., 137.132.80.57

Definition 2.8 (DNS)**.**
Domain Name System(DNS) translates between hostname and `IP` address.

A client must carry out a DNS query to determine the `IP` address corresponding to the server name (e.g., www.comp.nus.edu.sg) prior to the connection.

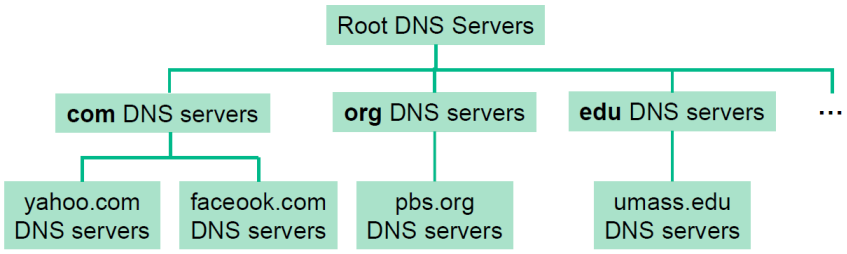
Definition 2.9 (DNS Resource Records)**.**
DNS resource records are mappings between host names and `IP` addresses and others. The format of resource records is

`(name, value, type, TTL)`

There are 4 `type` if resource records.

1. `type` = A: `(hostname, IP address, A, TTL)`
2. `type` = CNAME: `(caconical name, real name, CNAME, TTL)`
3. `type` = NS: `(domain name, hostname of authoritative name server for this domain, NS, TTL)`
4. `type` = MX: `(domain name, mail server, MX, TTL)`

DNS stores resource records in distributed databases implemented in hierarchy of many name servers. For example, if a client wants `IP` address for www.facebook.com:



- Client queries root server to find .com DNS server

- Client queries .com DNS server to get facebook.com DNS server
- Client queries facebook.com DNS server to get `IP` address for www.facebook.com

Here, **root servers** answer requests for records in the root zone by returning a list of the authoritative name servers for the appropriate top-level domain(TLD).
Top level domain(TLD) servers us responsible for com,org,net.edu and all top-level country domains and returns a list of authoritative servers.
Authoritative servers are organisation's own DNS servers, providing authoritative hostname to `IP` mappings for organisation's named hosts(e.g., web, mail). They can be maintained by organisation or service provider.
Local DNS server is also called *default name server*. It does not strictly belong to the hierarchy; each ISP has one local DNS server. When host makes a DNS query, query is sent to its local DNS server and

- Retrieve name-to-address translation from local cache;
- if answer is not found *locally*, acts as proxy and forwards query into hierarchy if answer is not found locally
- Once it learns mapping, it caches mapping. Cache entries will expire after Time to Live(TTL) countdown.

Note that DNS runs over `UDP` for speed. `UDP` is faster since compared to `TCP`, it avoids one RTT induced by `TCP` handshake.

3 Socket Programming

3.1 Processes

Definition 3.1 (Process)**.**
Applications runs in hosts as **processes**.

- Within the *same* host, two processes communicate using **inter-process communication** (defined by OS).
- Processes in *different* hosts communicate by exchanging **messages** (according to protocols).

In Client/Server model, **server process** waits to be contacted; **client process** initiates the communication.

Theorem 3.1 (Addressing processes)**.**
A process is identified by (**IP address**, **port number**), where **port number** is a 16-bit integer.

3.2 Sockets

Definition 3.2 (Socket)**.**
Socket is the software interface between app processes and transport layer protocols.

- Process sends/receives messages to/from its socket.
- Programming-wise: a set of **API** calls

Refer to Lecture Notes for programming details.

4 Transport Layer Protocol

4.1 Transport-layer Services

Internet transport layer protocols mainly are:

- `TCP` : connection-oriented and reliable
- `UDP` : connection-less and unreliable

Transport layer protocols run in hosts.

- Sender side:
  - breaks app message into **segments**
  - passes them to network layer
- Receiver side:
  - reassembles segments into message
  - passes it to application layer

4.2 `UDP` : User Datagram Protocol

`UDP` adds very little service on top of `IP`:

- Connectionless multiplexing/de-multiplexing
- Checksum

`UDP` transmission is **unreliable**. Therefore, to achieve reliable transmission over `UDP` , application layer protocol should implement error detection and recovery mechanism.
`UDP` has advantage over `TCP` in

- No connection establishment (No handshake delay)
- Simple: no connection state at sender and receiver
- Small header size
- No congestion control

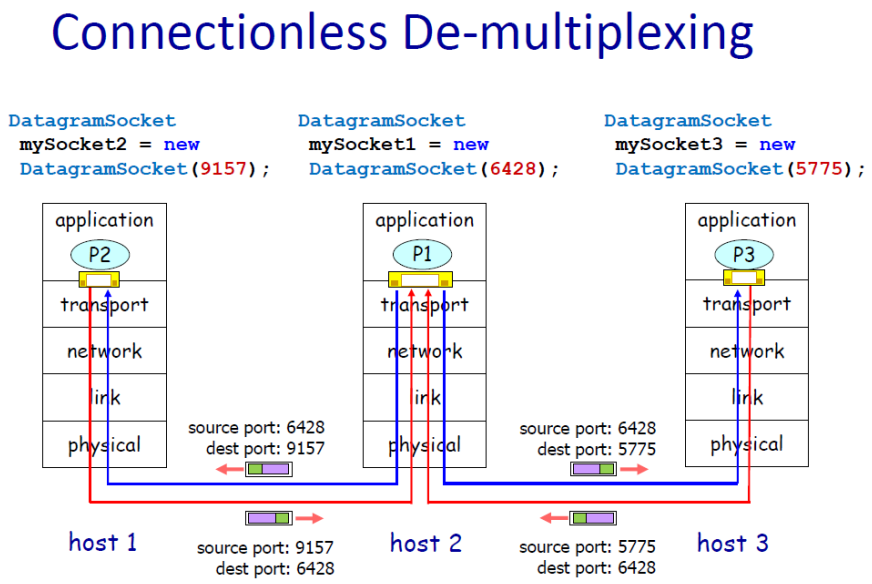
4.2.1 Connectionless De-multiplexing

Connectionless de-multiplexing is achieved by information on port numbers.

- `UDP` sender:
  - creates a socket with **local port number**.

- When creating a datagram to send to UDP socket, sender must specify **destination IP address** and **port number**.

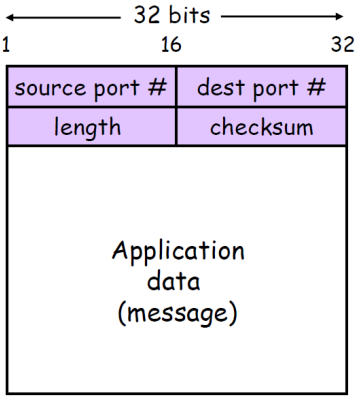
- When UDP receiver receives a UDP segment:
  - Checks **destination port number** in segment.
  - Directs UDP segment to the socket with that port number.
  - IP datagram from different sources with the **same destination port number** will be directed to the same UDP socket at destination.



4.2.2 UDP Header

Definition 4.1 (UDP Header)**.**
UDP **header** consists of 64 bits:

- bit 1 $\sim$ 16: source port number (0 $\sim$ 65535)
- bit 17 $\sim$ 32: destination port number (0 $\sim$ 65535)
- bit 33 $\sim$ 48: length of UDP segment in **bytes**, including header
- bit 49 $\sim$ 64: checksum



4.2.3 `UDP` Checksum

`UDP` Checksum is used to detect errors in transmitted segment.

- Sender
  1. Compute checksum value
  2. Put checksum value into `UDP` checksum field
- Receiver
  1. Computer checksum of received segment
  2. Check if computed checksum equals checksum field value
     - NO – error detected
     - YES – no error with respect to the checksum check

Theorem 4.1 (Checksum Computation)**.**
CHecksum is computed accordingly

1. Treat `UDP` segment as a sequence of **16-bit** integers.
2. Apply binary addition on every 16-bit integer. If the last integer is of length less than 16 bit, add zeroes to the right of last integer until 16 bit is reached.
3. Carry (if any) from the most significant bit will be added to the result
4. Compute 1's complement to get `UDP` checksum.

4.3 Principle of Reliable Data Transfer 3.0

In this section, we assume the underlying channel

- *may* flip bits in packets
- *may* lose packets
- *may* incur arbitrarily long packet delay
- *will NOT* reorder packets

Reliable data transfer involves a sender and a receiver.

- The sender sends out **segments**;
- the receiver replies **ACK** message.

A **sequence ID** is assigned by sender to each packet. **Acknowledgements** contains the sequence ID of the last successfully received packet.

4.3.1 Behavior of Sender

In reliable data transfer, sender maintains a **timer**. The timer will start when a segment is sent and stops by receiving ACK with sequence ID of this segment. Until then, another packet with a different sequence ID will be sent, with the whole process again. Otherwise, the timer will timeout, in which case this same packet is sent again.
All incoming packets(ACK) will be ignored if the sender waits to send, in which case there is no timer running.

4.3.2 Behaviour of the receiver

The data transfer is initiated by sender sending the first segment. On receiving a packet, *corrupted* copy will trigger an ACK of last correctly received copy. Receiver will also check the sequence ID for duplicate segment for duplicated copy. However, ACK of that copy's sequence ID will be sent regardless of whether it is a duplicate or not; the only difference is that duplicated copy will be discarded. If it is not corrupted and not a duplicate packet, it will be delivered to application layer.

4.3.3 Bit Error

Bit error occurs when some bits of the segment are flipped.
Bit error can occur in

- **segment**, in which case
    1. receiver detects bit error using checksum
    2. receiver ignores packet
    3. receives send ACK with sequence ID of last successfully transmitted segment
- ACK message, in which case
    1. sender will ignore the corrupted ACK

4.3.4 Packet Loss

Packet loss occurs in

- Segment, in which case receiver will do nothing;
- ACK message, in which case sender will do nothing.

4.3.5 Arbitrarily Long Packet Delay

Arbitrarily long packet delay can occur in

- Segment, in which case receiver reply ACK accordingly once received;
- ACK message, in which case sender will only act according to timer.

4.3.6 Finite State Machine Diagram of Reliable Transfer Protocol

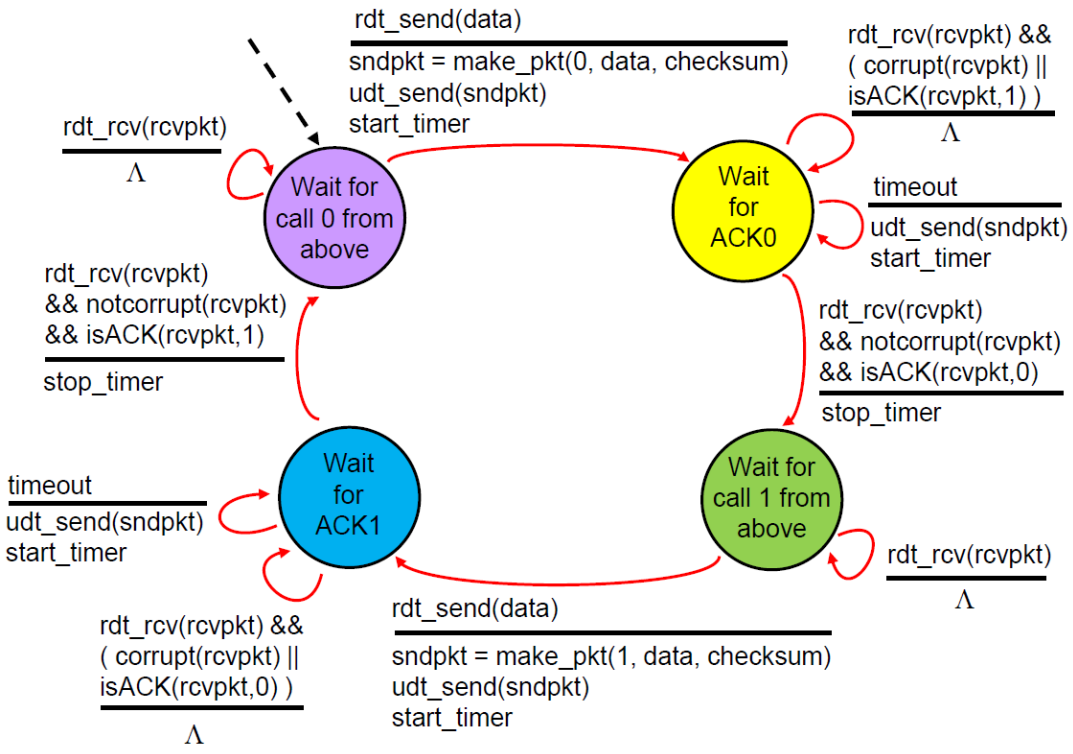


Figure 4: Reliable Data Transfer Sender Finite State Machine

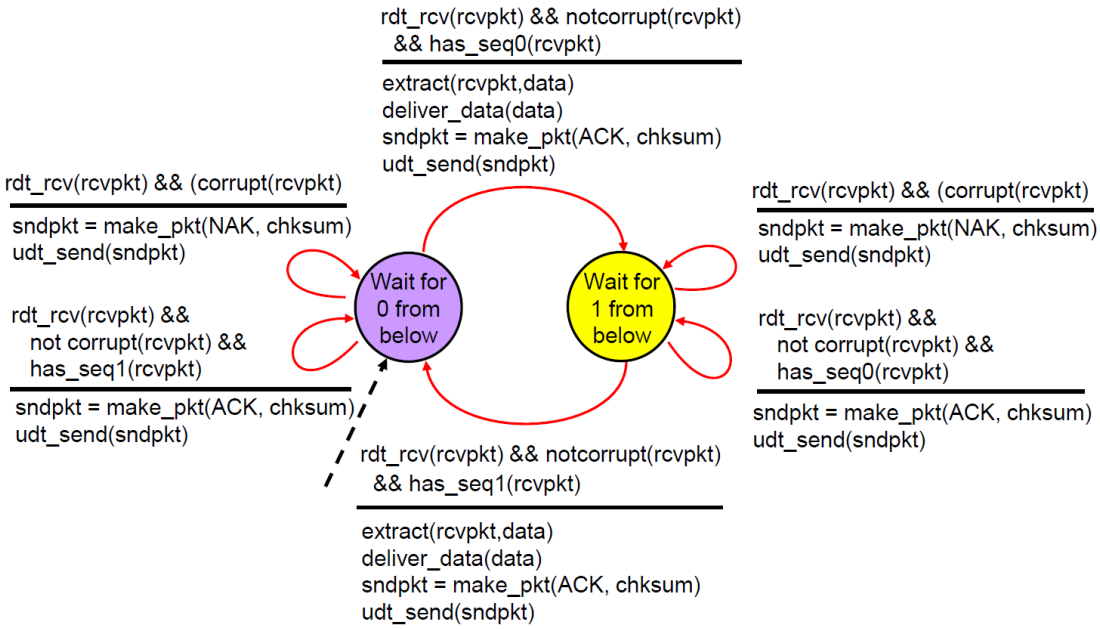


Figure 5: Reliable Data Transfer Receiver Finite State Machine

4.3.7 Performance of rdt 3.0

Definition 4.2 (Utilisation)**.**
Utilisation is defined as the fraction of time sender is busy sending.

rdt 3.0 stinks in performance, given its **stop-and-wait** nature. **Pipelining** can be used to increase utilisation, which allows multiple "in-flight", yet-to-be-acknowledged packets.
Two generic form of pipelined protocols are

- Go-Back-N
- Selective Repeat

4.4 Go-back-N

4.4.1 Sender Behaviour

- Sender maintains a window size of N, i.e., there can be N packets in pipeline, where the *first* is unACKed.
- There will be a timer for the first packet in the window.
- Packets in the window will be *cumulatively* ACKed.
- If the sender receive ACK of a packet with sequence ID k, all packet no later than sequence ID k will be ACKed; the window will slide until the next packet of sequence ID $k+1$.[3] Sender will send the packets newly entered in the window. If there is no timer running, start the timer to take the oldest unACKed packet.
- Else, the first packet's timer will timeout by receiving no ACK with sequence ID in the window. The sender will retransmit all the packets in the window.

4.4.2 Receiver Behaviour

- Same as rdt 3.0, corrupted packets will be discarded.
- Acknowledge packet in order, by maintaining an `expectedSeqNum`. Any packet of different sequence ID from `expectedSeqNum` will be discarded.
- In all cases, acknowledgement is **cumulative**.
  When the expected packet is received, ACK with sequence ID `expectedSeqNum` will be sent and `expectedSeqNum++`.
  When other packet is received, ACK with sequence ID `expectedSeqNum-1` will be sent.
  As such, ACK k means all packets up to k are received.

The benefit of cumulative ACK is that lost ACK packet will not cause retransmission as long as one ACK packet after this lost packet is received by sender before timeout. Therefore,

[3] This is due to cumulative acknowledgement.

timeout value shall be delicately chosen.

However, Go-back-N will waste network resources in a sense corrupt-free packet will be discarded merely for being out-of-order.

4.5 Selective Repeat

Selective repeat solves the problem by **buffering** the out-of-order packets.

4.5.1 Sender Behaviour

- Sender maintains a window size of N. Window will be maintained such that the first packet of the window is unACKed.
- There will be a timer for *every* packet in the window.
- Packets in the window will be *individually* ACKed.
- If one timer timeout, *only* the timed-out packet will be resent.
- Upon sliding of window due to acknowledgement of first packet, packets newly entered the window will be sent.

4.5.2 Receiver Behaviour

- Same as rdt 3.0, corrupted packets will be discarded.
- Acknowledge packet individually.
- Maintain a **receiver window** of equal length N to the sender window. The first packet in receiver window is not received.
- All the packets received *correctly*, which either are inside the receiver window, or outside the receiver window but within just one window length N before the first UnACKed packet, will fire an individual ACK message.
- If packets correctly received is in-order, all consecutive packets buffered in the receiver window will be delivered to application layer. The receiver window will slide to maintain the "first UnACKed" property.
- If packets correctly received is out-of-order, it will be buffered.
- All the packets *corrupted*, or out of the specified range before will not be buffered, and will not fire ACK.

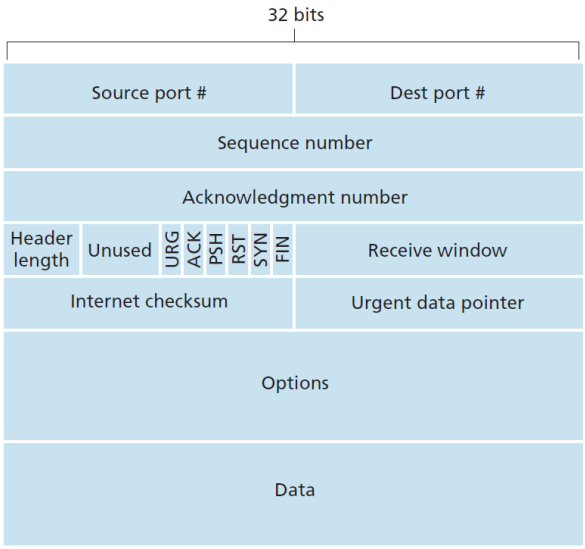
5 TCP : Transport Control Protocol

TCP is **connection-oriented** in a sense handshaking is required before sending application data.
TCP supports a **reliable, in-order** byte stream: Application passes data to TCP and TCP forms packets in view of **MSS**(maximum segment size).[4]
TCP also offers **flow control** and **congestion control**.

5.1 Connection-oriented De-mux

De-multiplexing in TCP is achieved by driecting a segment to the appropriate socket.
A TCP connection socket is identified by 4-tuple: `(srcIPAddr, srcPort, destIPAddr, destPort)`.

5.2 TCP Header



5.2.1 Sequence Number

TCP sequence number refers to the "byte number" of the first byte of data transferred in the segment in the file. Note the sequence number is 0-indexed.
The first sequence number used in handshake is randomly selected and this randomly selected number will be used as offset for all subsequent sequence number.

5.2.2 TCP ACK Number

TCP Acknowledgement number records the sequence number of the *next* byte of data *expected* by receiver.

[4]MSS excludes the length of TCP header.

TCP implements **cumulative** acknowledgement, in a sense TCP ACKs up to the first missing byte in the stream.

However, TCP spec doesn't say how receiver should handle out-of-order segments - it's up to implementer.

The A bit in the header:

- $A = 0$ denotes a data packet
- $A = 1$ denotes a ACK packet

There is only 1 timer running on the sender side.

5.2.3 TCP Sender Behaviour

- Event: `data` received from application above
  1. Create TCP segment with sequence number `NextSeqNum`
  2. Start timer if it is not running
  3. Send the packet
  4. Update sequence number used by next segment:`NextSeqNum = NextSeqNum + length(data)`
  5. Send until sender window is full.
- Event: Timer timeout
  1. Retransmit not-yet-acknowledged segment with *smallest* sequence number[5]
  2. Start timer
- Event: ACK received, with ACK field value `y`
  1. If `y>send_base`, shift sender window: `send_base=y`.
  2. If there are currently any not-yet-acknowledged segment, start timer.

5.2.4 TCP Receiver Behaviour

- Event: Arrival of in-order segment with sequence number `seqNum`
  - if all data up to `seqNum` are already ACKed, wait up to 500ms for next segment. Send ACK if no next segment arrival.
  - if one other segment is awaiting to send ACK, send single cumulative ACK, to ACK both in-order segment

[5]Only the oldest unACKed packet is retransmitted

- Event: Arrival of out-of-order segment with higher than expected `seqNum`(gap formed and detected)
  - Immediately send **duplicate** ACK, indicating `seqNum` of next expected segment[6]
- Event: Arrival of segment that partially or completely fills gap
  - Immediately send ACK, provided that the segment starts at lower end of gap.

5.2.5 TCP Timeout Value

TCP computes and keeps updating timeout interval based on estimated RTT.

$$\begin{aligned}
\text{Estimated RTT} &= (1-\alpha) \times \text{ Estimated RTT } + \alpha \times \text{ Sample RTT} \\
&\quad \text{typical value of } \alpha = 0.125 \\
\text{DevRTT} &= (1-\beta) \times \text{ DevRTT} + \beta \times |\text{SampleRTT} - \text{EstimatedRTT}| \\
&\quad \text{typical value of } \beta = 0.25 \\
\text{TimeoutInterval} &= \text{EstimatedRTT } + 4 \times \text{ DevRTT}
\end{aligned}$$

5.2.6 TCP Fast Retransmission

TCP adopts **fast retransmission** if sender receives 3 duplicate ACKs for the same data[7], it supposes that segment after the last ACKed segment is lost. Thus it will resend segment before timer expires[8].

5.2.7 Establishing Connection

Before exchanging data, TCP sender and receiver "shake hands".

1. Client chooses initial sequence number x, and send TCP SYN msg, by setting $S = 1$.
2. Server chooses initial sequence number y, and send TCP SYNACK msg, by setting $A = 1$,$S = 1$.
3. Client sends ACK msg by setting $A = 1$. In this message, it can contain client-to-server data.

5.2.8 Closing Connection

Client and server will close their side of connection by

1. Client sends FIN msg by setting $F = 1$. After this segment sent, client can no longer send application data, but can receive data.

[6] i.e. ACK the last correctly received packet
[7] 1+3 = 4 ACKs of the same data in total
[8] Retransmission will not reset the timer

2. Server send ACKFIN msg by setting $A = 1$, $F = 1$. Server can no longer send application data after this segment.

3. Client sends ACK mssg by setting $A = 1$.

6 Network Layer

Network layer is responsible for delivering packets to receiving hosts. Routers will examine header fields of IP datagrams passing it and direct it to the right destination.

6.1 IP Address

Definition 6.1 (IP v4 address)**.**
IP v4 address is a 32-bit binary integer used to identify a host.

A host can get an IP address either

- manually configured by system administrator, or
- automatically assigned by a DHCP server.

6.2 Dynamic Host Configuration Protocol(DHCP)

DHCP allows a host to dynamically obtain its IP address from DHCP server when it joins network. It has the following benefits:

- IP address is renewable
- allow reuse of address (only hold address while connected)
- support mobile users who want to join network

Theorem 6.1 (DHCP IP assignment)**.**
Assignment of IP address by DHCP server involves a 4-step process:

1. Host broadcasts "DHCP **discover**" message
   Format:
   `src: 0.0.0.0:68`
   `dest: 255.255.255.255:67`
   `yiaddr: 0.0.0.0`
   `transaction ID: 654`

2. DHCP server responds with "DHCP **offer**" message
   Format:
   `src: 223.1.2.5:67` (IP address of DHCP server)
   `dest: 255.255.255.255:68`
   `yiaddr: 223.1.2.4`
   `transaction ID:654`
   `lifetime: 3600 secs`

3. Host requests IP address: "DHCP **request**" message
   Format:
   `src: 0.0.0.0:68`

```
dest:  255.255.255.255:67
yiaddr:  223.1.2.4
transaction ID: 655
lifetime:  3600 secs
```

4. DHCP server sends address: "DHCP **ACK**" message
   Format:
   ```
   src:  223.1.2.5:67
   dest:  255.255.255.255:68
   yiaddr:  223.1.2.4
   transaction ID: 655
   lifetime:  3600 secs
   ```

In addition to host IP address assignment, DHCP may also provide a host additional network information, such as

- IP address of first-hop router
- IP address of local DNS server
- Network Mask(network prefix)

Note, DHCP runs over UDP

- DHCP server port number: 67
- DHCP client port number: 68

6.3 Special IP address

Special Addresses	Present Use
0.0.0.0/8	**Non-routable meta-address** for special use
127.0.0.0/8	**Loopback address** A datagram sent to an address within this block loops back inside the host. This is ordinarily implemented using only 127.0.0.1/32.
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	**Private addresses** They can be used without any coordination with IANA or an Internet registry.
255.255.255.255/32	**Broadcast address** All hosts on the same subnet receive a datagram with such a destination address.

6.4 IP address and Network Interface

An IP address is associated with a **network interface**.

- A host usually has one or two network interfaces(e.g. wired Ethernet and WiFi).
- A router typically has multiple interfaces (e.g. subnet)

6.4.1 IP address and Subnet

An IP address logically comprises two parts:

$$\overbrace{\underbrace{\text{Network (subnet) prefix}}_{n \text{ bits}} \mid \underbrace{\text{host ID}}_{32-n \text{ bits}}}^{32\text{bits}}$$

Definition 6.2 (Subnet)**.**
Subnet is a network formed by a group of "directly" interconnected hosts.

Theorem 6.2 (Properties of subsets)**.**

- Hosts in the same subnet have the *same* network prefix in their IP address.
- Hosts in the same subnet can physically reach each other *without* intervening router.
- They connect to the outside world through a router.

6.5 IP address assignment in Internet

Internet's IP address assignment strategy is known as **Classless Inter-domain Routing**(CIDR). For corporations, they will be offered with a continuum of IP address with fixed subnet prefix by an ISP.

- Subnet prefix of IP address is of arbitrary length.
- Address format: `a.b.c.d/x`, where `x` is the number of bits in subnet prefix of IP address.

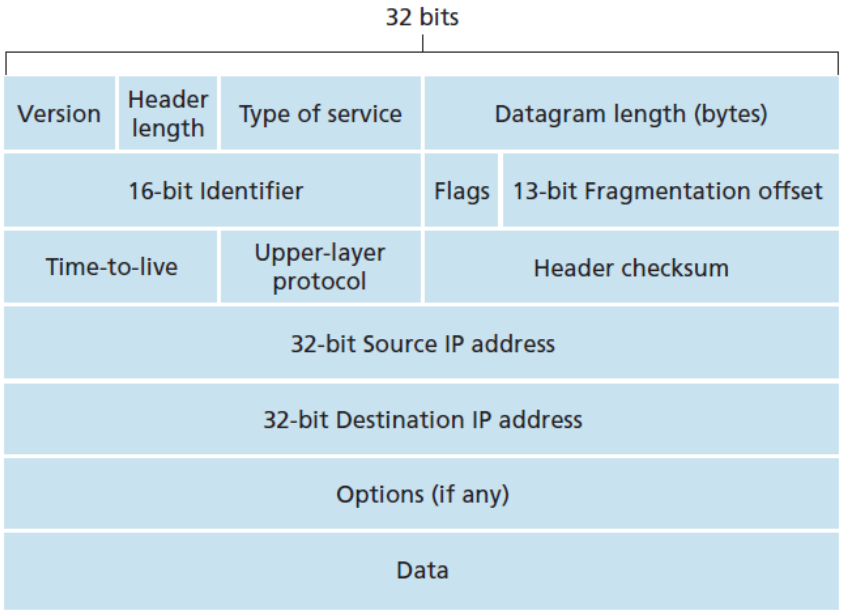
Subnet mask is used to determine which subnet an IP address beong to.
Subnet mask is calculated by setting all subnet prefix bits to 1 and hosts ID bits to 0.

6.6 Hierarchical Addressing

This hierarchical addressing via restricting subnet prefix at all levels allows efficient advertisement of routing information.
Router uses **longest prefix match** in forwarding table when determining to which next hop a IP datagram is sent.

7 IP and Routing

7.1 IP v4 Datagram Format



The key fields in the IP v4 datagram are the following

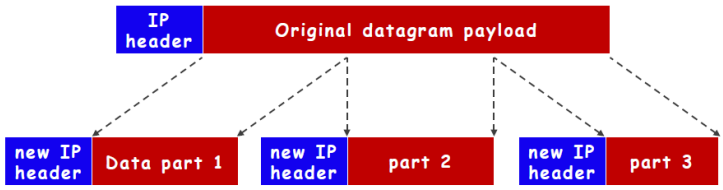
- Version Number: These 4 bits specify the IP protocol version.
- Datagram length: Total length of IP datagram (header *plus* data), measured in **bytes**.
- Identifier, flags, fragmentation offset are used in IP fragmentation.
- Time-to-live(TTL) refers to number of remaining hops, decremented at each router
- Header checksum is used to detect error for the IP header
- source and destination IP address refers to the initial sender and final receiver's IP address

A typical IP v4 header is 20 bytes long.

7.2 IP Fragmentation and Reassembly

IP fragmentation is used as different links may have different **maximum transfer unit**(MTU) specified by underlying link layer, and IP datagrams could be so large that need to be fragmented to fit into MTU. When destination hosts reassembles the packet, IP header fields are used to identify fragments and their relative order. **Flag**: flag is set to

- 1 if there is next fragment from the same segment



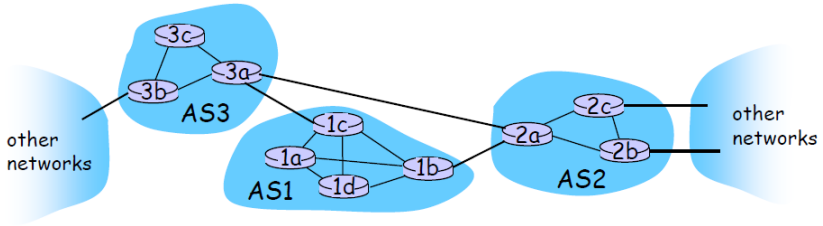
- 0 if this is the last fragment

Offset: offset is expressed in unit of 8-bytes, indicating the first byte of the data of this datagram in relative to the original fragmented datagram.
ID: Same datagram will use the same ID.
Length: Length field will change accordingly in smaller packets.

7.3 Intra-AS Routing

The Internet is a network-of-networks, organised in a hierarchy of **autonomous systems**(AS). Due to the size of the Internet and the decentralised administration of the Internet, routing on the Internet is done **hierarchically**:

- Intra-AS routing
  - Finds a good paths between two routers within an AS.
  - Commonly used protocols: RIP, OSPF
- Inter-AS routing
  - Handles the interfaces between ASes.
  - The de facto standard protocol: BGP



The aim of routing may differ between intra-AS routing and inter-AS routing.

- Intra-AS routing
  - Single admininistrator, so no policy decisions in needed.
  - Routing mostly focus on performance.
- Inter-AS routing

- Administrator often wants to control over how its traffic is routed, who routes through its net, etc.
- Policy may dominate over performance

We can abstractly view a network of routers as a *graph*, where vertices are routers and edges are physical links between routers.
Costs can be associated to each link, represented as the weight of respective edge.
In this sense, routing is the problem of finding a least cost path between two vertices in a graph.

7.3.1 Link State Algorithm

For link state algorithm, *all* routers have the complete knowledge of network topology and link cost. Routers will periodically broadcast link costs to each other.
The least cost path can be found using dijkstra algorithm.

7.3.2 Distance Vector Algorithm

Routers know *physically-connected neighbours* and *link costs to neighbours*. Router exchange **local views** with neighbours and update own local view, based on neighbours' view.
It admits an iterative process of computation converging slowly:

- Swap local view with *direct* neighbours.
- Update own's local view.
- Repeat 1 – 2 till no more change to local view.

Let $c(x, y)$ be the cost of link between routers x and y. $c(x, y) = \infty$ if x and y are note direct neighbours.
Let $d_x(y)$ be the cost of the *least*-cost path from x to y, from x point of view. This d is called teh distance vector.
Distance vector algorithm then utilises Bellman-Ford Equation:

$$d_x(y) = \min_{v \text{ neighbour of } x} \{c(x, v) + d_v(y)\}$$

The first term is easily obtained since it is mostly manually configured. The distance vector in second term is obtained from exchange of local views between neighbours.

7.3.3 Bellman-Ford behaviour

Router will maintain a $N \times N$ table where N is number of routers in the network, storing $d_{\text{from}}(\text{to})$.

- In the beginning, router x has only $c(x, v)$ known. $c(x, v)$ is ∞ if v is not a neighbour. Therefore, $d_x(v)$ is initialised as $c(x, v)$ and others to infinity.
- Router will exchange $d.(v)$ with each other. After each exchange, x will update $d_x(v)$ according to Bellman-Ford Equation.

- In addition, x will note down the next hop router to every destination, based on the least cost path. This information will be used to create forwarding table of x.
- Eventually this table will converge to provide optimal routing service.

7.4 Routing Information Protocol(RIP)

- Routing Information Protocol implements the Distance Vector Algorithm. It uses **hop counts** as the cost metric.[9]

- Entries in the routing table are aggregated subnet masks.
- Routers exchange routing table every 30 seconds over `UDP` port 520.
- Self-repair mechanism: if there is no update from a neighbour router for 3 minutes, the neighbour is assumed to be failed.

7.5 Network Address Translation(NAT)

Network Address Translation is used by routers connecting to both local network and the internet, for translating `IP` addresses with the aid of port number.
It is necessary since private `IP` addresses cannot be used for routing on the Internet.

7.5.1 Implementation of NAT

NAT enabled routers must:

- Maintain a NAT translation table, in which mapping between (source `IP` address, port #) of the private network and (NAT `IP` address, *new* port #) used on the Internet are stored.
- For **outgoing** datagrams, replace (source `IP` address, port #) to (NAT `IP` address, *new* port #).
- For **incoming** datagrams, replace (NAT `IP` address, *new* port #) in destination fields with coresponding (source `IP` address, port #) by NAT translation table.

The benefits of NAT include:

- There is no need to rent a range of public `IP` addresses from ISP, just one public `IP` for the NAT router.
- All hosts use private `IP` addresses. Addresses of hosts in local network without notifying the outside world.
- ISP can be changed without changing addresses of hosts in local network.
- Hosts inside local network are not explicitly addressable and visible by outside world.

[9]Therefore, it has the drawback of being insensitive to network congestion.

7.6 Internet Control Message Protocol(ICMP)

Internet Control Message Protocol(ICMP) is used by hosts and routers to communicate network-level information.

- Error reporting: unreachable host / netowrk / port / protocol
- Echo request/reply (used by `ping`)

ICMP messages are carried in `IP` datagrams. ICMP header starts **after** `IP` header.

Type	Code	Description
8	0	echo request(`ping`)
0	0	echo reply(`ping`)
3	1	dest host unreachable
3	3	dest port unreachable
11	0	TTL expired
12	0	bad `IP` header

7.6.1 `ping` and `traceroute`

Command `ping` sees if a remote host will respond to us – check for connection.
Command `traceroute` sends a series of small packets across a network and attempts to display the route (or path) that the messages would take to get to a remote host.

8 Network Security

Network security concerns about the following aspects:

- Message confidentiality
- Message integrity
- Message authenticity
- Service availability

Let us denote the two parties of communication as Alice and Bob, and the intruder as Trudy.

8.1 Principles of Cryptography

Cryptography is aimed to make it difficult for an unauthorised third party to understand private communications.
Cryptography involves both **algorithms**, which is known public, and **keys**, which are dependent on the type of algorithms used, and is kept secret.
Suppose Alice has an encryption key K_A and Bob has a corresponding decryption key K_B, the following relationship is preserved

$$K_B(K_A(m)) = m$$

where m is the message Alice wants to send to Bob.
There are two types of cryptography, namely **symmetric key** cryptography and **public key** cryptography.

8.1.1 Symmetric Key Cryptography

In symmetric key cryptography, Alice and Bob share and use the **same**(**symmetric**) key: K_{A-B}.
Popular algorithms include Advanced Encryption Standard(AES).
One simplistic example of symmetric key cryptography is **monoalphabetic cipher**. In this scheme, Alice and Bob share an identical mapping table from a permutation of 26 letters to another permutation, for translation between plaintext and ciphertext.
Symmetric key cryptography has the following problem: sender and receiver cannot agree on the shared key in the first place.

8.1.2 Public Key Cryptography

In public key cryptography, each party has a pair of keys, namely the **public key**, K^+, and **private key**, K^-. The public key is known public and the private key is kept confidental. The choice of this pair of key requires:

1. $K^-(K^+(m)) = m$
2. Given public key K^+, it should be very difficult to find private key K^-.

The most popular algorithm of public key cryptography is **RSA**, for which, the public key is the product of two *very large prime* numbers and the private key is derived from these two large primes.

Theorem 8.1 (Property of RSA)**.**
An important property of RSA is that the public key and private key are commutative:

$$K^- K^+ = K^+ K^- = 1$$

where 1 is the identity operator.

RSA, although resolves the problem of key sharing, is *computationally intensive*, thus slower. Therefore in practive, the following standard is used:

1. Use public key cryptography to establish secure connection, by exchanging the symmetric session key K_S.
2. Then use this session key for encryption and decryption.

Note that K_S is only valid for *one* session.

8.2 Message Integrity and Digital Signatures

Two ways to ensure message integrity and meesage authenticity are **message authentication code**(MAC) and **digital signature**.

8.2.1 Cryptographic Hash Function

Cryptographic hash function is the basis of both methods above.

Definition 8.1 (Cryptographic Hash Function)**.**
Cryptographic hash function is a hash function H which takes an input m, and generate a *fixed size* string $H(m)$, known as **message digest**(hash or finger print).
Cryptographic hash function must adhere the property that it is computationally infeasible to find two different messages m and m' such that $H(m) = H(m')$.

The following property makes it impossible for Trudy to forge another message m' with the same message digest as m.
Popular cryptographic hash functions include MD5 and SHA-1.

Theorem 8.2 (Work Flow of hash function)**.**
For Alice, the sender of the message.

1. Alice creates message m and calculates the hash $H(m)$.
2. Alice then appends $H(m)$ to the message m, creating an extended message $(m, H(m))$, and sends the extended message to Bob.

For Bob, the receiver,

1. Bob receives an extended message (m', h').

2. Bob calculates $H(m')$. If $H(m') = h'$, Bob concludes that message integrity is preserved.

However, cryptographic hash function does not guarantee message authenticity, which leads to the invention of message authentication code.

8.2.2 Message Authentication Code

To ensure message authenticity, Alice and Bob agrees on a **message authentication code**, s. The extended work flow is as follows:

Theorem 8.3 (Work Flow of message authentication code)**.**
For Alice, the sender of the message.

1. Append, by method of string concatenation, s after m to form the modified message $m + s$.

2. Calculate the hash $H(m + s)$.

3. Alice then appends $H(m+s)$ to the message m, creating an extended message $(m, H(m+s))$, and sends the extended message to Bob.

For Bob, the receiver,

1. Bob receives an extended message (m', h').

2. Bob calculates $H(m' + s)$. If $H(m' + s) = h'$, Bob concludes that message integrity **and** message authenticity are preserved.

8.2.3 Digital Signature

The problem of message authentication code is that either Bob and Alice could produce the same $H(m + s)$. Therefore, the producer of the message cannot be uniquely identified.
Digital signature, which solves this problem, has the following properties:

- **verifiable**: recipient(Bob) can verify that Alice, and **no one else**, has signed the document.

- **non-repudiation**: when presented the document and digital signature to a third party, the third party is confident that this document is indeed signed by Alice but no one else.

One way to produce the digital signature is to utilise the **private key** of the public key encryption scheme.

Theorem 8.4 (Work flow of Digital Signature)**.**
For Alice,

- Alice signs m be encrypting the message digest $H(m)$ with her private key K_A^-, creating a "signed" message digest $K_A^-(H(m))$.
- Alice sends both m and $K_A^-(H(m))$ to Bob.

For Bob, he receives $(m', \text{signature})$

- Verify the sender by computing the message digest $H(m)$.
- and compare with $K^+(\text{signature})$.

One problem of the above workflow is that, the public key Bob uses may not definitely be Alice's. To resolve this issue, certificate authority is used.
Certificate authority is an entity that issues digital certificates, which is Alice's public key encrypted by Certificate Authority's private key, i.e. $K_{\text{CA}}^- K_A^+$.
If Bob trusts the certificate authority's public key, it will trust Alice's public key, by applying $K_{\text{CA}}^+(DC)$.

8.3 Security Layer

8.4 SSL: Secure Socket Layer

Secure socket layer is an layer interfaced between application and transport layer. It is applicable to `TCP` conenctions.

8.5 IPSec: Internet Protocol Security

IPsec is a suite of protocols that secure communications by authenticating and ecrypting each `IP` packet of a communication session. It is interfaced between Transport layer and `IP` layer.
Both SSL and IPsec can be used to build VPN.

9 Link Layer

Link Layer is responsible of sending datagrams between adjacent nodes over a *single* link.

- `IP` datagrams are encapsulated in link-layer **frames** for transmission.
- Different link-layer protocols may be used on different links;
  each protocol may provide a different set of services.

Link layer may offer the following services

- **Framing**: encapsulate `IP` datagram into link-layer fram, adding header and trailer
  Remark: different protocols have different header and trailer format; each header and trailer is valid for *only one* hop
- **Link access control**: Coordinate which nodes can send frames at a certain point of time, when multiple nodes *share* a single link.
- **Reliable delivery**
- **Error detection**
  - Errors are usually caused by signal attenuation or noise.
  - Receiver detedcts presence of errors. Receiver, when error is detected, may signal sender for retransmission or simply drop frame.
- **Error correction**: Receiver may identify and correct bit error without resorting to retransmission.

Link layer is implemented in adapters, which contain both link and physical layer.

9.1 Error Detection and Correction

Commonly used error detection schemes include

- Checksum(used in `TCP` /`UDP` /`IP`)
- Parity Checking
- CRC(commonly used in link layer)

Note that error detection schemes are not 100% reliable.

9.1.1 Parity Checking

Definition 9.1 (Single Bit Parity)**.**
Suppose the data is of the form of $d_n \cdots d_1$, the parity bit p is chosen such that

$$\sum_{i=1}^{n} d_i + p \equiv 0 \mod 2$$

The sender, after computing the parity, will send data and parity bit, in the form of $d_n \cdots d_1 p$.

Definition 9.2 (Two-dimensional Bit Parity)**.**
In two-dimensional bit parity, the data is arranged in a two-dimensional array, and the single bit parity is computed for each row and column. The augmented array with both data and parity bits will be sent.
Two-dimensional bit parity can

- Detect **and correct** single bit errors in data.
- Detect two bits errors in data.

9.1.2 Cyclic Redundancy Check(CRC)

Cyclic Redundancy check is a power error-detection coding. It consists of

- D: data bits viewed as a binary number
- G: generator of length $(r+1)$ bits, agreed by sender and receiver beforehand
- R: CRC checksum in length of r bits

CRC checksum is computed as follows:

1. Append r bits of 0's after the data D.
2. Perform bitwise `XOR` operation on the augmented data $D\underbrace{0\cdots0}_{r \text{ bits}}$.

   This `XOR` operation is done withour carry or borrow, between augmented data and G. The operations starts from the MSBs of augmented data and cascading to the LSBs.
3. The remainder will be set as R, which replaces the trailing 0's in augmented data, to form the CRC checksum $D+R$, where $+$ is the string contacenation.

If $G \nmid D+R$, the error is detected.

9.2 Multiple Access Links and Protocols

There are two types of network links, namely

1. **Point-to-point link**, where a sender and receiver connected by a **dedicated** link, which do not need multiple access contol
2. **Broadcast link**, where
   - Multiple nodes connected to a shared broadcast channel
   - When a node transmits a frame, the channel broadcasts the frame and each other node receives a copy

In a broadcast channel, if two or more nodes transmit simultaneously, frames will **collide** at nodes and **none** will be correctly read.
Therefore, multiple access protocol is required, to serve as an distributed algorithms that determines how the nodes share the channel, using the channel itself.
Multiple access protocols can be categorised into three broad classes:

1. **Channel Partitioning**: divide channel into smaller "pieces"; pieces are allocated to nodes for **exclusive** use.
2. **Taking turns**: nodes take turns to transmit
3. **Random access**: channel are not divided so collisions are still possible; the mechanism focuses on recovery from collision.

9.2.1 Channel Partitioning Protocols

Definition 9.3 (Time Division Multiple Access(TDMA))**.**

- The channel is divided into equal time slots.
- Access to channel will be in "rounds", where each node gets **fixed** length slots(equivalently, equal transmission time) in each round.
- Unused slots go idle.

Definition 9.4 (Frequency Division Multiple Access(FDMA))**.**

- Channel spectrum is divided into frequency bands.
- Each node is assigned a **fixed** frequency band.
- Unused transmission frequency bands go idle.

9.2.2 "Taking Turns" Protocols

Definition 9.5 (Polling)**.**

- In polling, there is a **master** and rest of the nodes are the **slaves**.
- Master nodes invites slave nodes to transmit in turn.

Concerns of polling include (1)polling overhead and (2)single point of failure (at master node).

Definition 9.6 (Token Passing)**.**

- In token passing, there is a **control token**.
- Control token is passed from one node to next sequentially.
- Only the node with the control token can send.

Concern of token passing include (1)token overhead and (2)single point of failure (at control token).

9.2.3 Random Access Protocols

For random access protocols, when node has packet to send, there is no *a priori* coordination among nodes. A consequence is the presence of **collision**.
Random access protocols instead specify:

- How to detect collisions
- How to recover from collisions

Definition 9.7 (Slotted ALOHA)**.**
In slotted ALOHA,

- All frames require to have equal size.
- Time is divided into slotes of equal length.
- Node start to transmit only at the beginning of a slot.

The operation of each node includes

- **Collision Detection**: listens to the channel while transmissing
- **Collision resolution**: if collision happens, node retransmits a frame in *each subsequent* slot with probability p until success/

Definition 9.8 (Pure ALOHA)**.**
In pure ALOHA, there is no requirement of slot nor synchronisation: when there is a fresh frame, transmit immediately.
Therefore, the chance of collision increases.