The University of Melbourne
School of Computing and Information Systems

COMP30023
Computer Systems

Semester 1, 2017

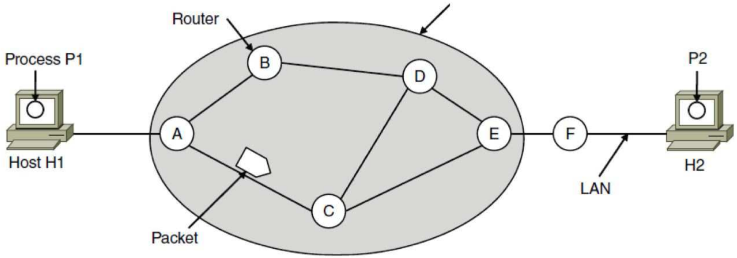
Section 8

TCP/IP

Topics (TCP/IP + more)

1. Packet switching
   - general overview
2. Network layer
   - IP Internet protocol
   - other control protocols for inter networking
3. Transport layer
   - services provided
   - UDP User datagram protocol
   - TCP Transmission control protocol
4. Routing
   - network layer task
5. An introduction to sockets

Store and forward packet switching



A key function of the network layer is to forward packets from source to destination.

Packet network

Data traffic divided into packets, each of which contains a header (with source and destination address). The packets travel separately through network.

- sender puts data into packets
- network interface hardware transforms data for physical transmission
- network delivers packets to variable destination (possibly along different paths)
- receiver converts physical signal back into a data packet
- receiver assembles packets back into data

It is possible that some packets may be lost, corrupted and/or delivered out of order. Error detection or correction provided by higher-level protocol.

Network layer services

Connectionless:

- packets (datagrams) injected into subnet independently and packets individually routed to destination
- *Internet*: move packets in a potentially unreliable subnet - QoS is not easily implemented

Connection-oriented:

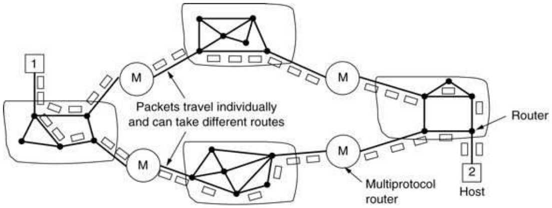
- packets travelling between destinations all use the same route
- *Telco*: guarantee reliability of subnet - QoS is important

Internetworking

Issues to be considered when connecting networks:

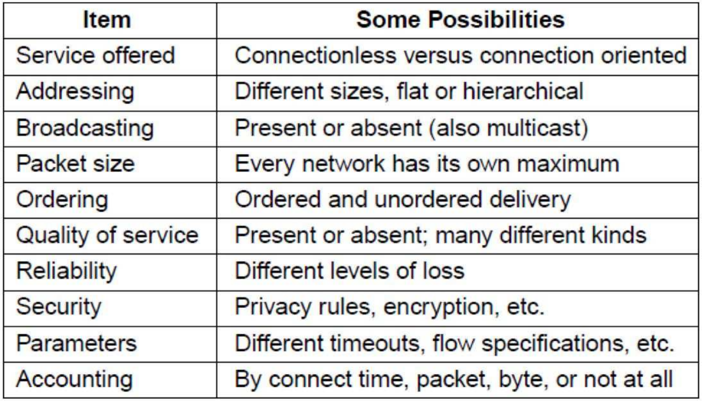
- different network types and protocols
- different motivations for network choices
- different technologies at both hardware and software levels

Connectionless networks can be connected using datagrams.



We will discuss *routing* in details towards the end of this section.

How networks differ . . .



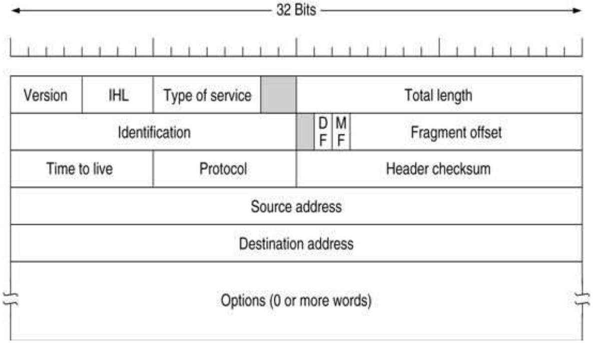
Item	Some Possibilities
Service offered	Connectionless versus connection oriented
Addressing	Different sizes, flat or hierarchical
Broadcasting	Present or absent (also multicast)
Packet size	Every network has its own maximum
Ordering	Ordered and unordered delivery
Quality of service	Present or absent; many different kinds
Reliability	Different levels of loss
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, packet, byte, or not at all

Some of the many ways networks can differ.

Internet Protocol (IP)

- Provides a "best-effort" service to route datagrams from source host to destination host. These hosts may be on same network or on different networks.
- Connectionless model
- Design goals:
  - services should be independent of router technologies
  - transport layer should be shielded from number, type and topology of routers
  - network addressing should use a uniform numbering plan

IPv4 header (frame structure)



IPv4 frame structure in detail

- IPv4 frame consists of a header and some text
- header is 20 byte fixed part + variable length optional part
- Version: IPv4 or IPv6
- IHL: Header Length - variable so this indicates actual
- Type: differentiates different classes of service
- Total Length: header and payload, maximum length 65535 bytes
- Identification: allows host to determine which datagram the new fragment belongs to - all fragments of same datagram have same ID
- DF: Don't Fragment byte

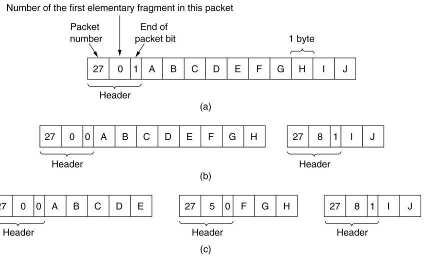
IPv4 frame structure in detail

- MF: More Fragment byte - are there more or is this the last one ?
- Fragment offset: where in the datagram the current fragment belongs
- TTL: limits packet lifetimes - hops or seconds
- Protocol: TCP, UDP, others ...
- Header Checksum: verifies the header only
- Source Address: IP - host/network
- Destination Address: IP - host/network
- Options: eg security, strict vs loose source routing, record route, timestamp

Fragmentation

- All networks have a maximum size for packets, could be motivated by:
  - hardware
  - OS
  - protocols
  - standards compliance
  - desire to reduce transmissions due to errors
  - desire for efficiency in communication channel
- Fragmentation (division of packets into fragments) allows network gateways to meet size constraints

Fragmentation

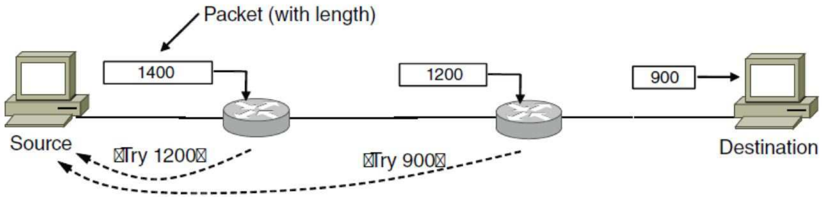


Fragmentation when the elementary data size is 1 byte.

a. original packet, containing 10 data bytes
b. fragments after passing through a network with maximum packet size of 8 payload bytes plus header
c. fragments after passing through a size 5 gateway

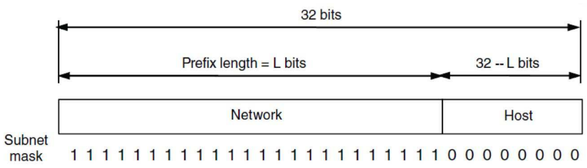
Fragmentation

An alternative approach: – IPv6

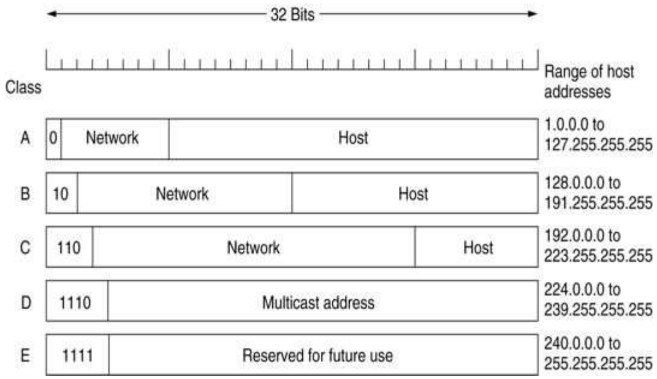


IP addresses

- IP Address is a 32-bit number that encodes the network and host number eg: 128.250.32.103 (dotted decimal notation)
- 0.0.0.0 is lowest, 255.255.255.255 is highest
- Overall IP allocation responsibility of Internet Corporation for Assigned Names and Numbers (ICANN) by delegation to IANA and Regional Internet Registries (RIR's)



IP address formats


32 Bits
Class
Range of host addresses
A
0
Network
Host
1.0.0.0 to 127.255.255.255
B
10
Network
Host
128.0.0.0 to 191.255.255.255
C
110
Network
Host
192.0.0.0 to 223.255.255.255
D
1110
Multicast address
224.0.0.0 to 239.255.255.255
E
1111
Reserved for future use
240.0.0.0 to 255.255.255.255

IP numbering schemes

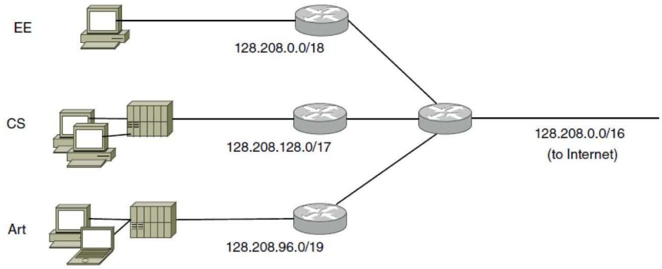
- Class A: 128 networks, 16m hosts each
- Class B: 16,384 networks, 64k hosts each
- Class C: 2m networks, 256 hosts each
- Class D: multicast

Subnets

- Subnetting allows networks to be split into several parts for internal uses whilst acting like a single network for external use
- Subnet masks can be written using:
  - "dotted decimal" (eg 255.255.255.128 indicates 2 internal networks) or
  - "slash" notation (eg /25)

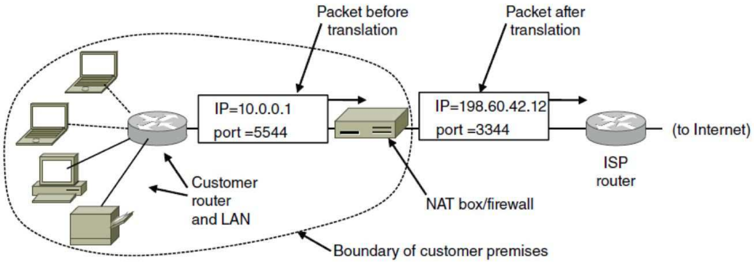
Number of Subnets	Subnet Mask	Slash Notation
1	255.255.255.0	/24
2	255.255.255.128	/25
4	255.255.255.192	/26
8	255.255.255.224	/27

IP address examples



University	First address	Last address	How many	Prefix
Cambridge	194.24.0.0	194.24.7.255	2048	194.24.0.0/21
Edinburgh	194.24.8.0	194.24.11.255	1024	194.24.8.0/22
(Available)	194.24.12.0	194.24.15.255	1024	194.24.12/22
Oxford	194.24.16.0	194.24.31.255	4096	194.24.16.0/20

IP address examples – NAT



Placement and operation of a NAT box.

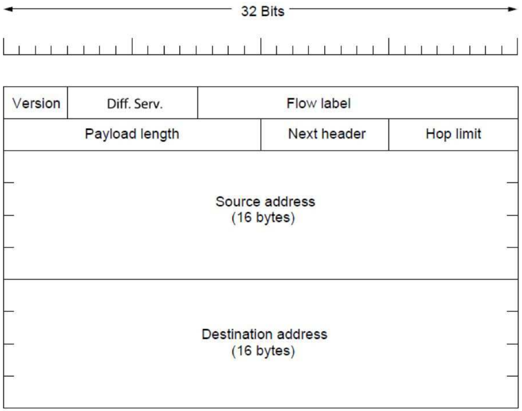
IPv4 and IPv6

Internet Protocol Version 4 is the most popular protocol in use today, although there are some questions about its capability to serve the Internet community much longer.

The main issue surrounding IPv4 is addressing – or, the lack of addressing – because many experts believe that we are nearly out of the four billion addresses available in IPv4. IPv6 could be the solution to many problems posed by IPv4

IPv6 uses 128 bit address instead of 32 bit address. (IPv6 addresses to be used based on geographical location).

IPv6 header



Goal for use include: reduce routing table size, simplify protocol, better security, roaming host without changing address allow future protocol evolution.

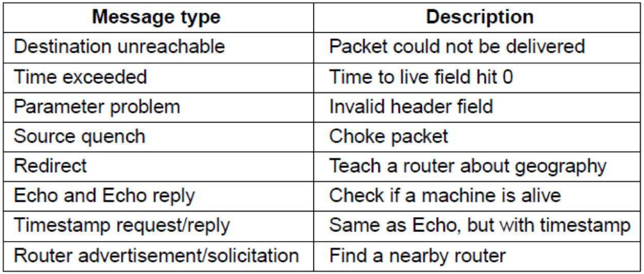
Internet control protocols

Control protocols in the network layer on the Internet:

- ICMP
- ARP (see slides in routing section)
- RARP (see slides in routing section)

Internet control message protocol

The **ICMP** Internet control message protocol is used for testing and monitoring ambient conditions between hosts and routers.



Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and Echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

Investigate `traceroute`

Transport layer

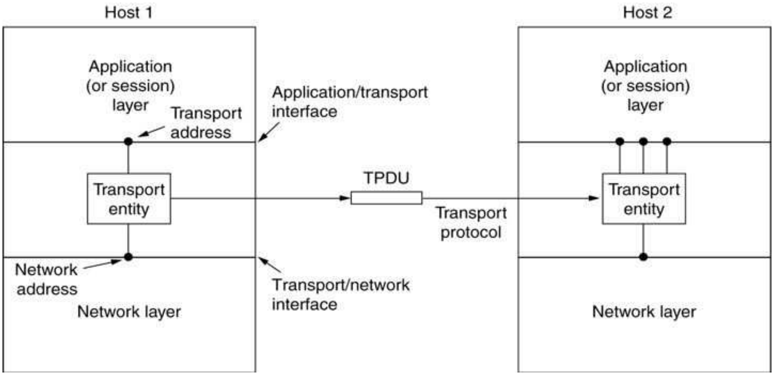
The primary function of the transport layer is to provide reliable cost-effective data transport from source to destination, independent of physical or data networks.

The Transport layer **services** provide interfaces between the Application layer and the Network layer. The Transport layer **entities** (the hardware or software which actually does the work eg. OS kernel, processes, NIC) can exist in multiple locations.

Services provided a logical communication between application processes running on different hosts:

- Connection-oriented: connection establishment, data transfer, connection release (TCP)
- Connectionless: data transfer (UDP)

Transport entity illustrated (Tanenbaum)



Note: connection-oriented transport services provide a reliable service on top of an unreliable network.

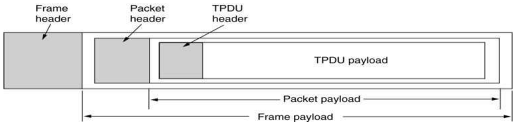
Transport vs Network layer services

If the transport and network layers are so similar, why are there two layers?

- Transport layer code runs entirely on hosts
- Network layer code runs almost entirely on routers
- Transport layer can fix *reliability* problems caused by the Network layer eg. delayed, lost or duplicated packets

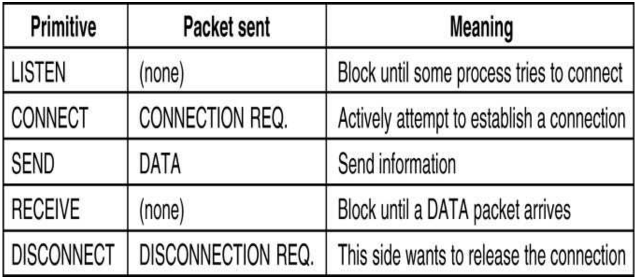
Transport layer encapsulation

- Abstract representation of messages sent to and from transport entities
  - Transport Protocol Data Unit (TPDU)
- Encapsulation of TPDUs (transport layer units) in packets (network layer units) in frames (data layer units)



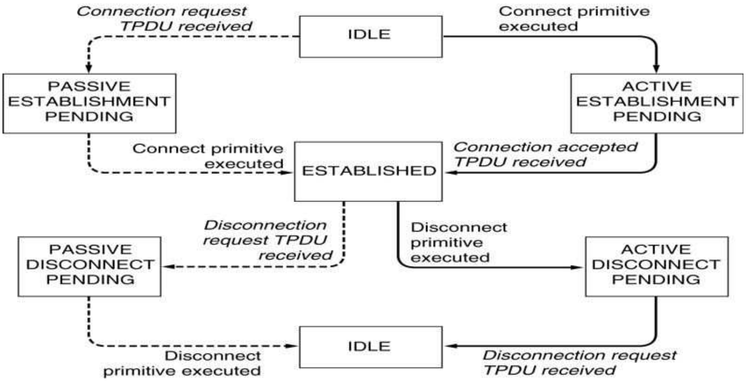
Transport service primitives

Primitives: core functions which allow interface with transport services



Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Connection process illustrated


Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

Transport vs Data-link protocols

Transport services are implemented as protocols used between transport entities.

- Transport services have some similarities with data link protocols
  - error control
  - sequencing
  - flow control
- Transport services have some differences with data link protocols
  - physical network vs physical/data/network
  - addressing
  - connection establishment
  - storage

Addressing

Specification of the remote process to “connect to” is required at both the application and transport layers.

- addressing in the Transport layer is typically done using port numbers (eg port 80).

    cf. Unix /etc/services, www.iana.org (well known ports)
    a process server intercepts inbound connections and spawns requested server and attaches inbound connection

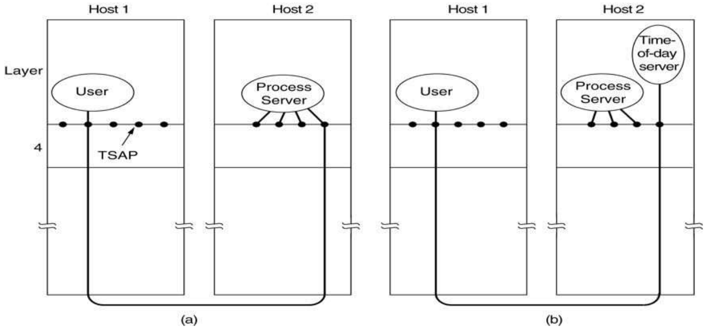
    - cf. Unix /etc/(x)inetd

- addressing in the network later is typically done using the IP address

Port allocations

- Port numbers can range from 0-65535
- Port numbers are regulated by IANA (http://www.iana.org/assignments/port-numbers)
- Ports are classified into 3 segments:
    - Well Known Ports (0-1023)
        - 21 FTP
        - 22 SSH
        - 23 Telnet
        - 25 SMTP
        - 80 HTTP
        - 110 POP3
        - 119 NNTP
    - Registered Ports (1024-49151)
    - Dynamic Ports (49152-65535)

Transport layer access points



(a) Process server intercepts inbound connection

(b) Process server attaches inbound connection to spawned server process

Connection establishment issues

- When networks can *lose, store and duplicate* packets, connection establishment can be complicated
  - congested networks may delay acknowledgements
  - incurring repeated multiple transmissions
  - any of which may not arrive at all or out of sequence - *delayed duplicates*
- Applications degenerate with such congestion (eg. imagine duplication of bank withdrawals)

UDP

The **User Datagram Protocol** provides a protocol whereby applications can transmit encapsulated IP datagrams *without a connection establishment*.

- UDP transmits in segments consisting of a header followed by the payload
- UDP headers contain source and destination ports, payload is handed to the process which is attached to the particular port at destination (using BIND primitive or similar)

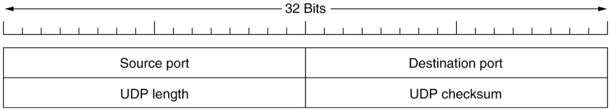
UDP

The main advantage of using UDP over raw IP is the ability to specify ports for source and destination pairs. Note: both source and destination ports are required - destination allows initial routing for incoming segments, source allows reply routing for outgoing segments.

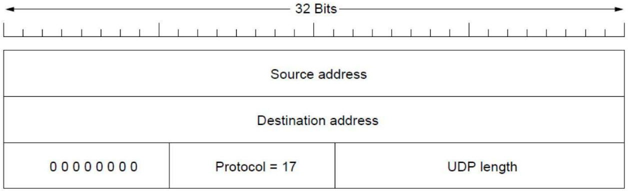
Strengths and weaknesses of UDP:

- **Strengths**: provides an IP interface with multiplexing/de-multiplexing capabilities and consequently, transmission efficiencies
- **Weaknesses**: UDP does not include support for flow control, error control or retransmission of bad segments
- **Conclusion**: where applications require a precise level of control over packet flow/error/timing, UDP is a good choice

UDP header

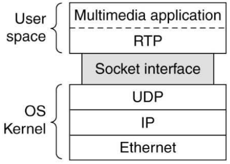


(top) UDP header

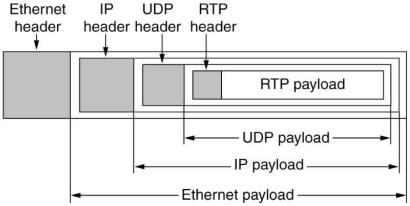


(bottom) The IPv4 pseudoheader included in the UDP checksum.

UDP example use



(a)



(b)

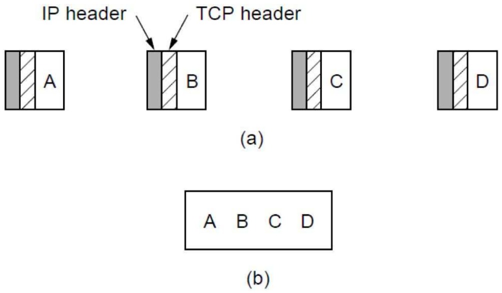
Example: (a) The position of real-time protocol in (on) the protocol stack. (b) Packet nesting

TCP

The **Transmission Control Protocol** provides a protocol by which applications can transmit IP datagrams within a *connection-oriented* framework, thus increasing reliability.

- TCP transport entity manages TCP streams and interfaces to the IP layer
- TCP entity accepts user data streams, and segments them into pieces <64Kb (often 1460b in order to fit the IP and TCP headers into a single Ethernet frame), and sends each piece as a separate IP datagram
- Recipient TCP entities reconstruct the original byte streams from the encapsulation

TCP service model



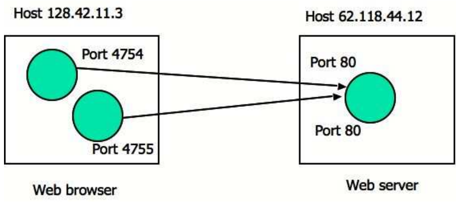
Example:
(a) Four 512-byte segments sent as separate IP datagrams
(b) The 2048 bytes of data delivered to the application in a single READ call

TCP service model

The sender and receiver both create "**sockets**", consisting of the IP address of the host and a port number

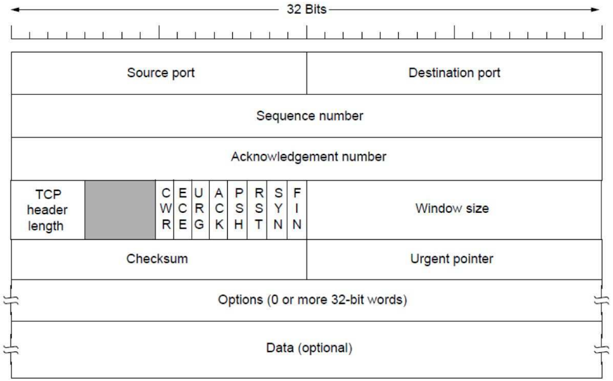
- For TCP service to be activated, connections must be explicitly established between a socket at a sending host **(src-host, src-port)** and a socket at a receiving host **(dest-host, dest-port)**

Example:



See end of this section for further details.

The TCP header



Features of TCP connections

- TCP connections are:
  - *Full duplex* - data in both directions simultaneously
  - *Point to point* - exact pairs of senders and receivers
  - *Byte streams*, not message streams - message boundaries are not preserved
  - *Buffer capable* - TCP entity can choose to buffer prior to sending or not depending on the context
    - PUSH flag - indicates a transmission not to be delayed
    - URGENT flag - indicates that transmission should be sent immediately (priority above in process data)

TCP details

- Data exchanged between TCP entities in segments - each segment has a fixed 20 byte header plus zero or more data bytes
- TCP entities decide how large segments should be within 2 constraints, namely:
  - 65,515 byte IP payload
  - Maximum Transfer Unit (MTU) - generally 1500 bytes
- *Sliding window protocol* - sender transmits and starts a timer, receiver sends back an acknowledgement which is the next sequence number expected - if sender's timer expires before acknowledgement, then the sender transmits the original segment again

Three-way handshake

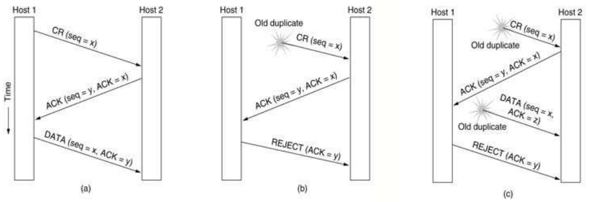
Goals of reliable connection establishment:

- Ensure packet lifetimes are bounded
- Assign sequence numbers that will not be reused within a packet lifetime
- Ensure initial send and receive sequence numbers are agreed at start of connection

Three-way handshake:

- A proposed solution, which avoids problems that can occur when both sides allocate same sequence numbers by accident (eg after host /router crash) (cf. Tomlinson, 1975).
- Sender and receivers exchange information about which sequencing strategy each will use, and agree on it before transmitting TPDU's

Three-way handshake



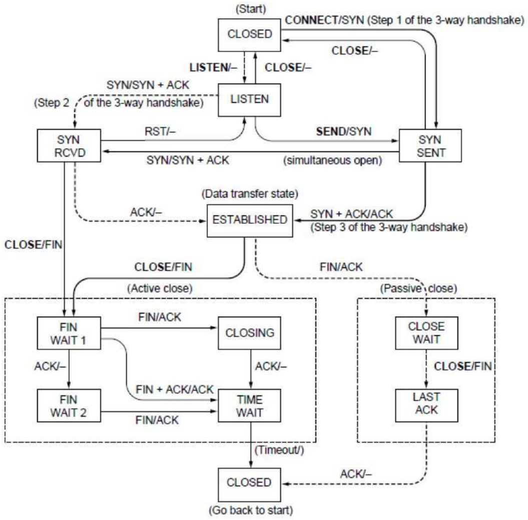
(a) Normal operation,

(b) Old CONNECTION REQUEST appearing out of nowhere.

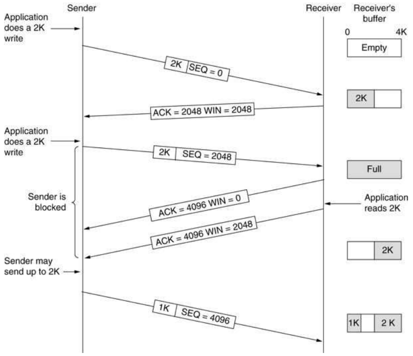
(c) Duplicate CONNECTION REQUEST and duplicate ACK.

Two simultaneous connection attempts results in only one connection (uniquely identified by end points). Connections released asynchronously (2 x asymmetric releases, one for each transmission direction). Timers used to lost connection releases

Connection management



TCP transmission policy



TCP acknowledges bytes, not packets. Receiver advertises window based on available buffer space.

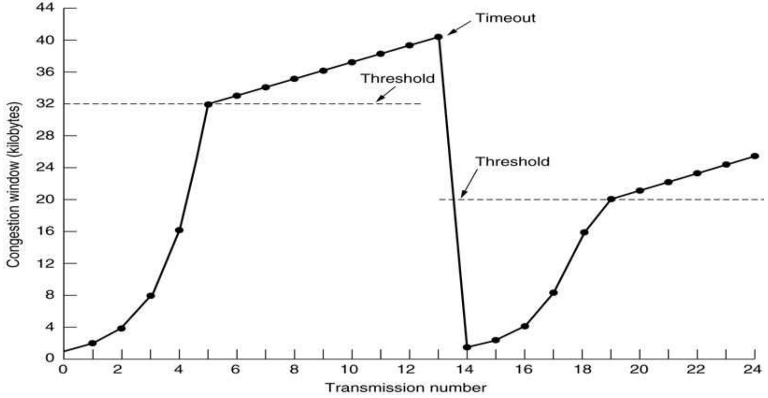
TCP and congestion control

- When networks are overloaded, congestion occurs, potentially affecting all layers
- Although lower layers (data and network) attempt to ameliorate congestion, in reality TCP impacts congestion most significantly because TCP offers methods to transparently reduce the data rate, and hence reduce congestion itself
- TCP adopts a defensive stance - open loop solution
    - at connection establishment, a suitable window size is chosen by the receiver based on its buffer size
    - if the sender is constrained to this size, then congestion problems will not occur due to buffer overflow at the receiver itself, but may still occur due to congestion within the network
    - ensure packet lifetimes are bounded

Incremental congestion control

- On connection establishment, the sender initializes the congestion window to the size of the maximum segment in use on the connection, and transmits one segment
- If this segment is acknowledged before the timer expires, the sender adds another segment's worth of bytes to the congestion window, making it two maximum size segments, and transmits two segments
- As each new segment is acknowledged, the congestion window is increased by one maximum segment size
- In effect, each acknowledgement doubles the congestion window - which grows until either a timeout occurs or the receiver's specified window is reached

Internet congestion control – an example



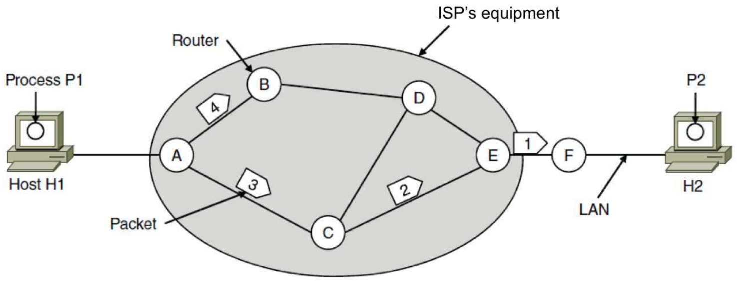
Packet forwarding – how does routing work?

Each router has a forwarding table. This maps destination addresses to outgoing interfaces. Upon receiving a packet:

- inspect the destination IP address in the header
- index into the table
- determine the outgoing interface
- forward the packet out that interface

Then, the next router in the path repeats the process – the packet travels along the path to destination.

Routing within a datagram subnet



A's table (initially)

A	
B	B
C	C
D	B
E	C
F	C

Dest. Line

A's table (later)

A	
B	B
C	C
D	B
E	D
F	D

C's Table

A	A
B	A
C	
D	E
E	E
F	E

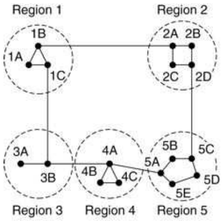
E's Table

A	C
B	D
C	C
D	D
E	
F	F

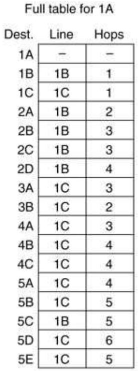
Routing algorithms

- The routing algorithm is responsible for deciding on which output line an incoming packet should be transmitted
- Routing algorithms examples:
    - non-adaptive: static decision making process
    - adaptive: dynamic decision making process
- Hierarchical routing
    - as networks grow in size, routing tables expand – but this impacts on performance and memory requirements
    - dividing all routers into regions allows efficiencies
        - each router knows everything about other routers in its region but nothing about routers in other regions
        - routers which connect to two regions act as exchange points for routing decisions

Hierarchical routing (an example)

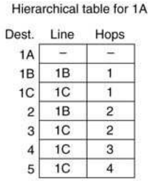


(a)


Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)


Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

(a) the networks; (b) the full router table for 1A; and (c) hierarchical table for 1A.

IP addressing and routing tables

- Routing tables are typically based around a triple:
    - IP Address
    - Subnet Mask
    - Outgoing Line (physical or virtual)
- eg: 203.32.8.0 255.255.255.0 Eth 0
- Longest mask always used when choosing a route

Where do forwarding tables come from?

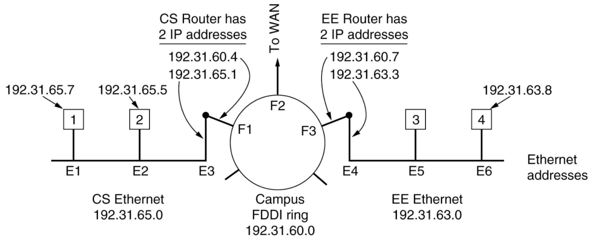
Routers have forwarding tables – map prefix to outgoing link(s)

Entries can be statically configured e.g., map 12.34.158.0/24 to Serial0/0.1

However, this does not adapt to failures, new equipment and the need to balance load. That is where other technologies such as routing protocols and DHCP (The Dynamic Host Configuration Protocol) come in.

Address resolution protocol

- Used to resolve IP allocation to MAC address by broadcasting IP address to all machines on network
- Proxy ARP – determination of IP allocation by a third party
- ARP Cache – temporary storage of ARP results



Client-Server

Client program

- running on end host; requests service eg. Web browser.
- does not communicate directly with other clients
- needs to know servers address

Server program

- running on end host; provides service eg. Web server
- does not initiate contact with the clients
- needs fixed, known address

Delivering the data ...

Network

- deliver data packet to the destination host based on the destination IP address

Operating system

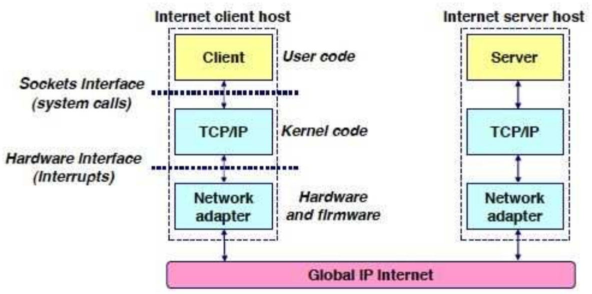
- deliver data to the destination socket based on the destination port number (e.g., port 80)

Application

- read data from and write data to the socket – then, interpret the data (e.g., render a Web page)

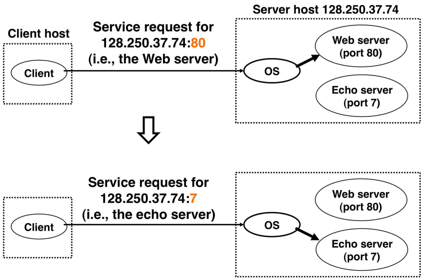
Sockets

When sending message from one process to another, the message must traverse the underlying network. A process sends and receives through a **socket** – in essence, the doorway leading in/out of the "application"



Using sockets

The socket API supports the creation of network applications. To develop inter-process communication using a network application, an **association** must exist: protocol, source-IP, source-port number, destination-IP, destination-port number.



Port numbers

A port number uniquely identifies the socket. You cannot use same port number twice with same address.

The OS enforces uniqueness – it keeps track of which port numbers are in use; it does not let a second program use the same port number.

Example: consider two Web servers running on a machine. Both servers cannot both use port “80”, the standard port number. Therefore, the second one might use a non-standard port number.

`http://www.unimelb.edu.au:8080`

Unix Socket API

Socket interface

- originally provided in Berkeley UNIX
- later adopted by all popular operating systems
- simplifies porting applications to different OSes

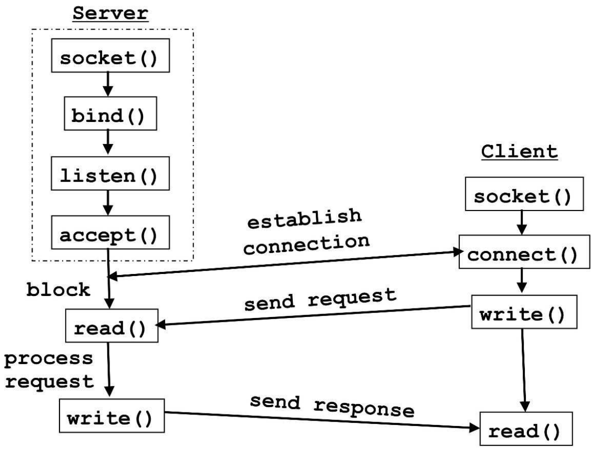
In UNIX, everything is like a file

- all input is like reading a file
- all output is like writing a file
- file is represented by an integer file descriptor

API implemented as system calls:

- example include connect, read, write, close,

Using sockets



Example C code

See LMS for sample C code introducing socket programming.