Computer Networks

15-1-15

BOOK: Computer Networks by Tannenbaum

→ Collection of computer systems/hosts (usually heterogenous) loosely interconnected by a single technology: copper, fiber, microwave, satellite, wireless etc. links

→ Networked nature of the system is clearly visible to the end user

→ The internet, or other large networked systems, formally speaking, are not a single computer network but a network of several separate comp. networks

→ Distributed systems also use no. of networked comp. systems, but are not computer networks (loosely coupled - significant delay in trans^n)

- Appears as single coherent system (subsystems) to end user (networked nature carefully hidden for user)
- Single model or paradigm is presented to all users
- Implemented by layer of software (usually called middleware) over existing OS. Normally diff on diff systems
- WWW could be an example (approx) of distributed system built on top of 'The Internet'

Why computer networks

→ Resource sharing: H/W, S/W, data info.

→ Communication: 1-1, 1-many (group), 1-all, many-many

→ On-line finance: banking, credit, card tx^n, share trading etc.

→ Electronic commerce: on-line pdt. catalogue, sales, purchase, exchange, competitive price bidding, etc.

- No general taxonomy / nomenclature to fit all computer networks
- 2 broad / imp. classifying features
  - → tx^m tech. used
  - → Scale / size of networks

- → Tx^m tech classified by type of physical / logical links:
  - Broadcast links
    - Use shared (by multiple hosts) links for comm.
    - Unicasting - 1 sending, 1 receiving
    - Multicasting - 1 " host and a selected group of receiving hosts (not all hosts)
    - Broadcasting - one host sending to all other hosts
    - high prob. of collision if more than one host tries to tx^m at same time
    - Local Area Networks normally use broadcast links
    - LANs are usually owned by individuals or pvt. organizations

→ Tx^m tech classified by type of phy / logical links
- Point to point:
  - Communication links not shared by more than 1 host
  - Many pairs of source - dest. hosts
  - Source to destination path may / usually goes through several other hosts, which act as routing agents
  - Proper / optimal routing is very imp.
  - Routing may be simple (static) or sophisticated (dynamic) depending on type of connecting links, nature & vol^m of network traffic, nature of apps running on the network, etc.

- ~~Two~~
- Size / scale of network classified by physical extent
  - Square meter - Personal Area Network (PAN)
  - Room / building / campus - Local " " (LAN)
  - City - Metropolitan " " (MAN)
  - Country / Continent - Wide " " (WAN)
  - Complete planet - The Internet

PAN

→ Bluetooth, wireless, NFC

LAN

→ Cu, wireless, fiber links

→ Links are shared, uni-cast, mult-cast, broadcast

→ Collision / contention very common (between 2 or more talking hosts)

→ Some arbitration technique necessary
- Static - simple, inefficient & ~~wasteful~~ wasteful
- Dynamic - more involved, but efficient
- Centralized - easy to manage, but vulnerable to single faults
- ~~Dec~~ Decentralized - more involved but more robust

→ Topology
- Ethernet - IEEE 802.3
- Token bus - IEEE 802.4
- Token ring - IEEE 802.5
- Operating speed - 1 to 10/100 Mbps for normal LANs
  10 Gbps or higher for fast/special purpose LANs

LAN

- Limitation of max length of LAN cable segment & no of hosts / other equipment connectable to a segment - problem mitigated using repeaters between cable segments
- Lan topology almost always decided or designed by owner

MAN

- Several hosts geographically distributed over city/town and suburbs (small, medium, large)
- Earliest CATV networks - ~~or~~ egs. of analog ~~m~~ MAN with single central host broadcasting to all others; present ones (CATV) are digital, allowing 2 way interactive multimedia & net access

- Topology:
  - Usually has several hosts on shared / broadcast channel (LAN)
  - Several of these LANs connected by point-point links through one or more switching nodes
  - Man may also be considered as collection of LANs rather than a single network

WAN

- Spread over several countries / continents
- Normally appears unstructured, at least topologically
- Has two broad components
  - Hosts / computers / nodes - normally owned by individual / user / organization
  - Comm. subnet - owned by specialized service providing companies, such as private or national ISPs, etc
  - 2 major components:
    - → Comm. links - owned by national telephone / data network companies
    - → Switching nodes / routers - owned by both national and pvt service providers

- WAN topology is almost always constrained by non technical considerations
  - → Geopolitical
  - → legal
  - → ~~Int~~. Intl. & other regional treaty / agreements, etc
  - → Almost always forced to use national telephone / data network of a country (predefn topology & b/w constraints)
  - → Normally not feasible (economically or otherwise) to lay dedicated comm. links

Size / scale of network classified by physical extent

- Complete planet – The internet
  - → Internet (interconnected network) of many diff geographically distributed WANs

- Usual topology – interconn. networks

  fig 1.9

  - → Single host can be replaced by a LAN
  - → Atleast 1 host in the LAN must be a router/switching node and must be connected to a router in the comm. subnet
  - → Comm. between ~~nod~~ hosts in same LAN (intra LAN) use comm. rules (protocols) of that LAN – routing is simple since broadcast / shared link is used
  - → Comm between hosts in diff LANs (inter-LAN) is more involved
    - → Source & dest LAN protocols as well as those for the diff p2p links / routers in subnet are used
    - → Routing is more complex

- Usual Topology - interconnected networks

fig 1.10

→ 2 routers may not be directly connected; path between them may go through intermediate routers (multihop) acting as forwarding agents

→ Multiple paths between subnet routers, so multiple paths between source - dest. hosts possible

→ Optimal route (min. length/delay) has to be used

Routing

→ Static routing:

→ Specific paths (ordered list of routers and links) for all possible source-dest host pair are computed at the time of design & remain in effect for entire life of the network

→ Real time overhead is low

→ Inefficient, since real time changes in network traffic, congestion & faults are not taken care of

Dynamic Routing

→ S-D path computed in real time

→ Each router examines dest. host-id of incoming message and determines outgoing link to use depending on network traffic /congestion, link bandwidth/availability, msg length / priority, etc.

(Accidental blank space. ignore)

→ More complex than static routing

→ More efficient, robust and responsive to changing traffic congestion and fault conditions

→ 2 broad types – message / packet switching & ckt. switching

→ Message / packet switching

- Messages of arbit. length may be diff. to handle because of fixed b/w of links
- Msgs usually split into smaller units called packets (normally fixed size)
- Each packet has dest-id & packet-no and source id and is routed separately
- Packet arrives at intermediate node, is temporarily stored and then fwded along appropriate outgoing link, when it becomes free / available
- Appropriateness of outgoing link decided by dynamic routing algo in use
- Diff. packets between same source–dest pair can travel through diff paths & end-to-end travel time may be diff
- Packets from a single message can and usually arrive out of sequence at the dest

→ Dest host stores packets as they arrive

→ Message is reconstructed by rearranging packets back in sequence after they have arrived

→ Store and fwd / packet switched subnet subnet using msg / packet switching type of routing

Circuit switching

→ Full path / ckt from source to dest host is estb. first (all links & intermediate hosts / routers)

→ All links & routers reserved for duration of message transm.

→ All packets of entire message sent in proper sequence through reserved path / ckt

→ Path / ckt terminated after successful transm of entire message (links & routers released for routing of other messages)

→ Routing method expensive and relatively slow

→ ~~High~~ reliability very high

→ Used only when out-of-sequence packet arrival packet re-sequencing delay, lost/corrupted packets are not tolerable

~~Sato~~

Satellite links

→ Inherently similar to shared broadcast links
→ Downlink trⁿ from sat. to a router also heard by all other routers in same radio range
→ Uplink transⁿ from router to sat. heard by neighbors routers within range

Most WANS use LANs, store and fwd./packet switched subnets and satellite links

→ Networks divided into groups
→ Group has several LANs interconnected by p2p links and uses store & fwd./packet switched routing between LANs

→ At least 1 router (possibly more) in every group has sat. uplink / downlink antenna
→ Routers in the same group comm. via their LANs & if necessary through the store & fwd/p. switched subnet
→ Intergroup. comm. is via satl. links

Protocol hierarchy - a layered approach

Fig 1.13

→ Networks use layered design to provide diff. levels of ~~at~~ abstraction and reduce design complexity
→ No. of layers, layer names, contents & services vary from N/W to N/W ~~to layer above, but hides implementation details~~
→ Each layer provides a well defined set of services to layer above, but hides implementation details
→ Layer n of a host communicates with layer n of another hosts by following a set of rules

→ This set of comm. rules ~~for~~ form layer n protocol & same throughout the network

→ Protocols for diff layers are diff

→ Entities on layer n. of diff hosts, communicating through layer n protocol are called peer ~~processes~~ process

→ Comm. channels used by layer n peer processes are virtual (dotted lines in fig.) except for lowest layer

→ Lowest layer of diff hosts are connected by actual physical channel

** → There can be multiple ~~#~~ protocols for a layer

→ Reliability / fault tolerance of virtual channels are higher (higher layers are more reliable)

→ Services: defines a well defined set of services provided by a layer to the one above it (by k to k+1)

→ Interface: def^n between each pair of layers at every host; precisely specifies set of rules / format for

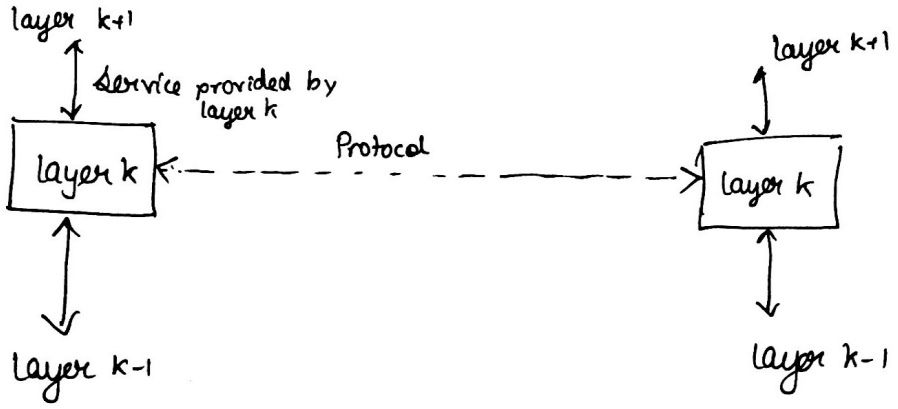
- layer k to avail services from layer k-1
- layer k-1 to return results to layer k

→ Protocols: used by ~~per~~ peer process on layer n of 1 host to carry out meaningful conversation with corr. process on another host

- ~~Per~~ Peer processes are not allowed aware of implementation details at lower layer

→ <u>Protocol stack</u>: an ordered list of layers and a collection of protocols, <u>at least one per layer</u>

- More than one protocol per layer can and do exist
- Also called '<u>protocol suit</u>'



1.10
3
47
75
122

Network architecture specification

→ Set of layers, & 1 or more protocols per layer

→ Just enough details provided to ~~allt~~ allow s/w, F/W, H/W design to correctly implement layer's protocol(s)

→ Layer's imp. details not part of network specification

→ Layer services (to layer above) & inter layer interface not included in network specs.

- Layer ~~specifications~~ services & inter-layer interface not visible to other hosts
- Particular layer in diff hosts may provide a diff. set of services, have diff inter layer interface, even diff. implementations

→ Any host pair communicate correctly if they have correct no of layers and each layer protocol(s) is/are correctly implemented

A simplified example

Fig 1.15

⇒

1) Message M generated at layer 5 in source host for transm.n to peer process in dest host via virtual channel

- M & some service request actually sent down to a process in source host layer 4 through layer interface

2) Source host layer 4 process enhances M (text/image reformatting/error checking/compression etc.) for peer process in dest. host; certain info. (text/image extraction info, reformatting info, msg ID, etc) added to M as header H_4 & sent via virtual channel

- Augmented M actually sent down to layer 3 with some service request

~~Handouting~~

3) Source host layer 3 process treats M & H4 as the total message, it may break up the var. size msg into no of fixed size for ease of processing at lower layers; it will add its own header H3 which contains packet ID and other info & transmit to layer 3 peer process via virtual channel

- packets however are sent down to layer 2

4) Layer 2 process may add routing info, i.e., forwarding link no, error detecting/corr. info etc. in header H2 (trailer T2 as well) & transmit to L2 peer process by virtual channel

- Aug. packets are actually sent down to layer 1 which is the actual phy. channel (possibly H/W one)

5) L1 p. process, does phy. trans. through phy. channel to dest. layer 1 p. process

- ~~At dest. host~~,

- At dest host ~~layer~~ each layer removes its own header from aug. packet/msg, performs any processing specified by the header content and delivers the rem

- Header/trailer associated with a layer is never fwded to layer above

- Packets/msgs received by processes at all dest. host layers in same format as generated by corr. peer processes in source host

- Dest host p. processes at each layer are unaware of processing details at lower layers & assume direct trans.^{n} of augmented packets/messages from corr. source host p. process (via v. links)

- Design issues at diff. layers

→ Host/process addressing, naming
  - Diff considerations & diff naming at diff layers

→ Data tx.f rules
  - simplex, half duplex, full duplex, etc

→ No of logical channels per connection
  - Usually atleast 2 (full duplex)
  - Several low b/w logical channel multiplexed through one high b/w connection & vice versa

→ Error control & recovery
  - Error detecting/correcting codes in header/trailer
  - Ack from dest host
  - Prob. of lost/duplicate packets

→ Message/Packet ordering
  - Recept/storage of out of ~~order~~ sequence packet
  - Ordering of packets in proper sequence

→ Flow control
  - Avoiding swamping of slow receiver by fast sender
  - Avoiding starvation of fast receiver because of slow sender

→ Managing large/small msg
  - Splitting large var. sized msg into fixed size packets
  - Packing small messages into one packet

→ Routing and congestion control

Computer Networks

22-1-15

Connection Oriented and Connectionless services (between hosts)

- Connection oriented service

→ Modeled after telephone network

→ Service user requests for / sets up a conn.
  - Sender-receiver negotiate on params such as msg size, Quality of service (QOS), i.e., end-to-end msg delay, delay between messages, reliability fault rate etc.
  - Proposal and counter proposals are offered till agreement is reached
  - This is the sender-receiver synchronization phase

→ Connection is actually setup using some routing algorithm
  - Conceptually an end-to-end / tube or circuit with no breaks or loss

→ Service user uses the connection
  - Requests packets sent in proper sequence from service user
  - Requests " received by service provider in proper sequence; no out of sequence packets or lost packets
  - Service reply packets sent by service provider in proper sequence & all of them received at service user end in proper order
  - Several rounds of request - reply packet sequences may follow

→ Service user (or service provider) terminates connection
  - On Completion of request-reply, service user / provider asks for dismantling of connection
  - All links & routers used for setting up conn. released / unreserved for use by other hosts

→ Connection-less service

- Modeled after postal / mail system
- Each msg / packet has full dest & source addresses
- Each " " " routed through the network independently in a stand alone ~~basis~~ basis
- Usually a sequence of messages / packets may ~~arrive~~ arrive in correct order at receiving end if n/w links have high b/w & congestion is low
- Network congestion or faulty links / routers may cause consecutive msgs / packets to be routed through diff paths resulting in out-of-sequence arrival at dest.
- Msg / packet re-ordering is a major overhead at dest. end (never happens for conn. oriented service)

→ Quality of Service (QoS)

→ Reliable (acknowledged)

- No data loss / info
- Receiver send ack. Packet to sender for all packet / msg. received
- Ack. packet tells sender if msg. / packet received ok, or with errors)
- For an erronous receipt, sender retransmits msg / packet till error free ack. is received
- Process repeated for all msgs / packets
- Add^nl packets generated; causes increase in traffic or delay
- Delay tolerated, for high reliability services, eg; secure file transfer

→ Reliable C. oriented service (acknowledged)

(2 varieties)

→ Message ~~streams~~ sequences

- Msg boundaries and ordering preserved from source to dest. – two 1024 bytes msgs will not be misrepresented as one 2048 byte msg or 4 512 bytes ~~long~~ msgs
- Example :- A newspaper with content generation, editorial, typesetting facilities in a central location & multiple online printing facilities at all ~~mg~~ major cities / towns
  - Pg. boundaries must be preserved
  - Pgs must arrive in corr. order
  - All pgs must arrive & must be error free
  - Delay between pg. arrivals not very imp.

→ Byte Streams

- Just a sequence of bytes with ordering & error control strictly maintained – a 2048 byte block can be interpreted as two 1024 bytes block or 1024 two byte blocks
- Example – A user at a terminal, carries out a query – response session with the server (a stream of bytes)
  - No need to recognize an ordered set of bytes (query & response all one byte long)
  - Query & response (bytes) ordering must be maintained
  - All bytes arrive error free, but delay is not very important

Unreliable conn. oriented service

→ Usual connection oriented approach used but receiver does not send back acknowledgement packets

- Used when delay because of acknowledgement cannot be tolerated
- A small % of garbled/erroneous msgs/byte tolerable
- Msgs/bytes should arrive strictly in correct order
- Delay between msgs/bytes should be less than given bound

Example - Digital voice traffic, video streaming
- Occasionally garbled voice perfectly acceptable compared to clear & noise free voice but long and variable periods of silence between words
- Slightly noisy video more acceptable than crystal clear pics with very jerky animations where frames tend to freeze frequently

→ Subnets used for implementing conn. oriented services are also called virtual circuit subnets

Unreliable connectionless service

→ For msg delivery with high success probability, but 100% correct delivery assurance not necessary
- Sender need not wait for ack to every msg. sent
- Msgs are standalone, i.e., delivery sequence need not be exactly same sending sequence
  - Eg: email

→ Unreliable connectionless services are also called Datagrams

→ Comm. subnet for implementing datagrams are called datagram subnet

Reliable (acknowledged) con. less service

→ Implementation same as for c. less service, but receiver sends ack. for every msg /pkt received & sender retransmits a msg /pkt until it is received correctly

- Used for msg. that are sent only once or very occasionally
- Msgs are very short
- Conn. set-up & dismantling overhead not tolerable, but assured error free delivery necessary
- Ack. datagrams used

Example - email with automatic ack. option when read by recipient

Request reply service

→ User sends datagram containing request to a server

→ Server sends back a datagram containing reply /response

- This exchange of info is normally done through unacknowledged datagrams i.e. unreliable connectionless service

→ Several such request-reply datagrams are exchanged during the session

→ Finally, user send final datagram containing logoff request and server responds by ending session

fig 1.16

- Protocol Stack
  - → Normally located within the OS

- Primitive (service) Calls
  - → These are sys. calls which trap to the kernel mode to get the system to send/receive necessary packets or perform other valid layer services

- ~~Service primitives~~
  - ~~→ formally specifies the service (provided by a layer)~~
  - ~~→ t~~

(ignore whitespace)

- Service primitives
  - → formally specifies the service (provided by a layer)
  - → lists set of primitives (operations) available to a peer process
  - → tells the service to perform an action taken by peer entity
  - → Available primitives depend on type of service - primitives for conn. or. service will be diff. from those for conn. less, etc.

- A simple illustration (minimal) - reliable Byte-stream in a client-server environment

fig 1.17 (LISTEN only sent by server)
CONNECT primarily sent by client

- A simple illustration (minimal) – reliable Byte stream in a client server env.

fig 1.18

Initial step: server executes a LISTEN (normally)
LISTEN is usually blocking sys. call; server process blocks, waiting for a conn. request from client side

Step 1 – Client process first executes CONNECT (must provide server ID), then ~~exees~~ executes a ~~receive~~ RECEIVE and blocks waiting for server's response
(as at client sends a packet to peer (server OS) requesting it to establish conn.)

Step 2 – Client packet received & processed by server OS
it finds that request is for connection
It checks for a listener process (initial LISTEN call insures this)

The listener is unblocked & it sends an ACK packet to the client (waiting, after a ~~receive~~ RECEIVE call); This ACK is generated by the protocol code itself, not in response to a client level primitive

Server process executes a RECEIVE & gets blocked waiting for the first service request from the client

Step 3 – Client issues SEND with details of service requested

[Client OS provides details of service requested with params, if any; protocol implementing SW/FW creates packets using these info & sends to the server (over communication subnet) by exec. send primitives]
Client executes RECEIVE, gets blocked waiting for reply from server

Step 4 - Arrival of request packet unblocks server process (for this client) ~~to~~ from ~~#~~ RECEIVE executed in step 2

Packet content inspected by server & required service fn. performed

Serv. reply/response encapsulated in a pkt & sent to client by exec. SEND primitive

Server executes RECEIVE & blocks waiting ~~& sent~~ for further service requests, if any from client

If there are further clients requests, steps 3 & 4 are repeated

Step 5: When client is done with current session, it executes a ~~disconnect~~ DISCONNECT primitive to request for disconnection i.e., dismantling of connection

[Client OS feeds request to client side network s/w/f/w which creates necessary packets and sends over to server by a SEND]

Client blocks by exec. ~~receive~~ RECEIVE and waits for disconnection confirmation from server

Step 6:
- On getting packet containing discon.
- Server discom. (ack/confirmation) sent as a packet from server network s/w/f/w to client s/w/f/w
- Disconnect pkt ~~b~~ may also be used by routing mechanism to unreserve links & routers, breaking the connection
- Disconnect pkt reaches client network s/w/f/w client OS informed & it terminates client process

Problems

- Timing may be wrong: CONNECT at client side is exec before LISTEN at the server and server may
  - immediately issue a 'request unrecognized, try later' response
  - queue the CONNECT request for sometime in anticipation of a LISTEN being exec. & initiate a conn. if this happens
  - Discard CONNECT request from queue after timeout in absence of a 'LISTENER'
- Lost and duplicate packets