→ TCP/IP layers
→ OSI layers
→ Sync / Async / Isoch.
→

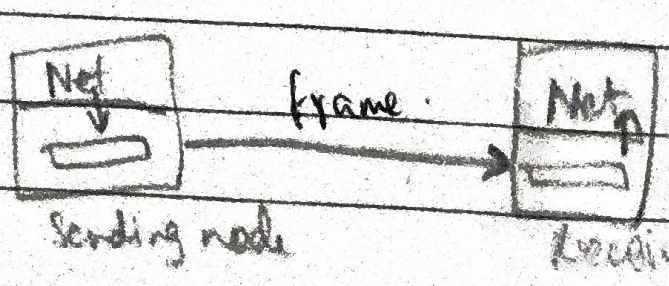
→ Multiplexing:
- Time division
- Freq. ,,
- Code ,,
- Wavel. ,,
- Statistical

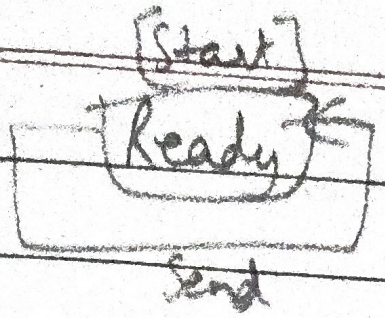
→ MAC
- 2nd layer
- fixed
- ~~Defined~~ Physical addr.
- 00-00-00-0C-00-00 Hex = 000.000.000.000
- Hard coded

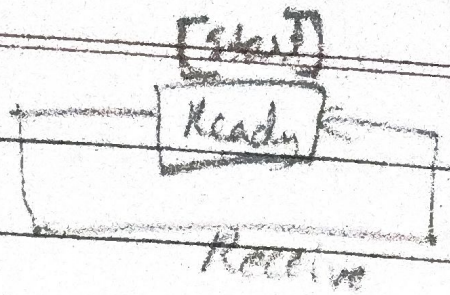
IP
- 3rd layer
-
-
-

→ Data Link layer Services
- Flow cont.
- Err. rectf.
- Phys. Addr.ing
- Framing
- Connection less/oriented

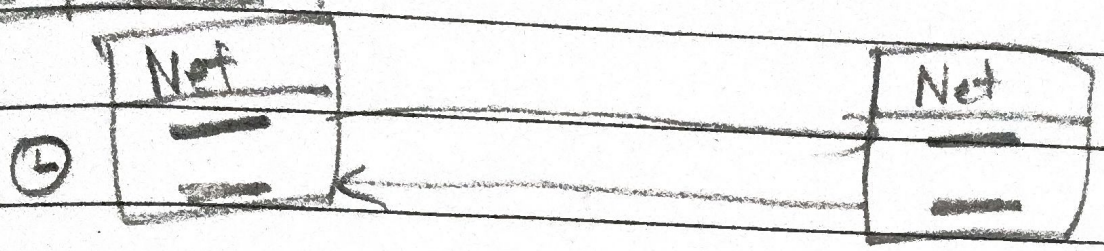
→ DLL protocols:
↳ Simple:

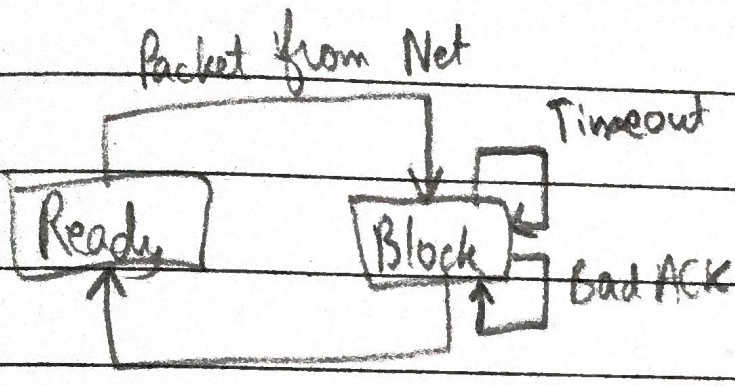


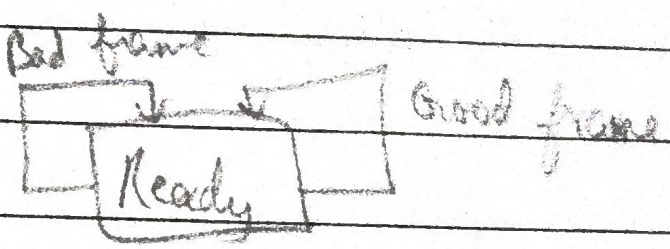



↳ Stop & Wait:



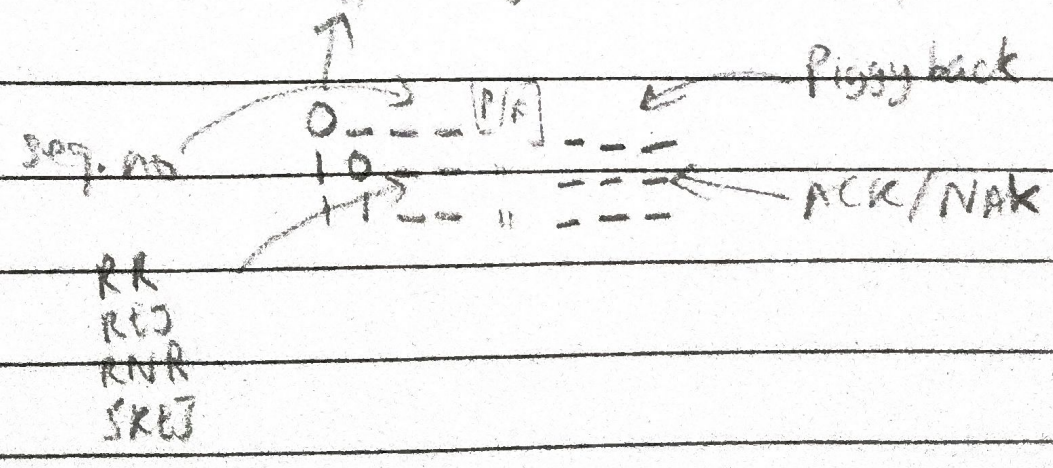




→ HDLC:


I: Flag Addr Cntr. Info FCS Flag

S: " " X " "

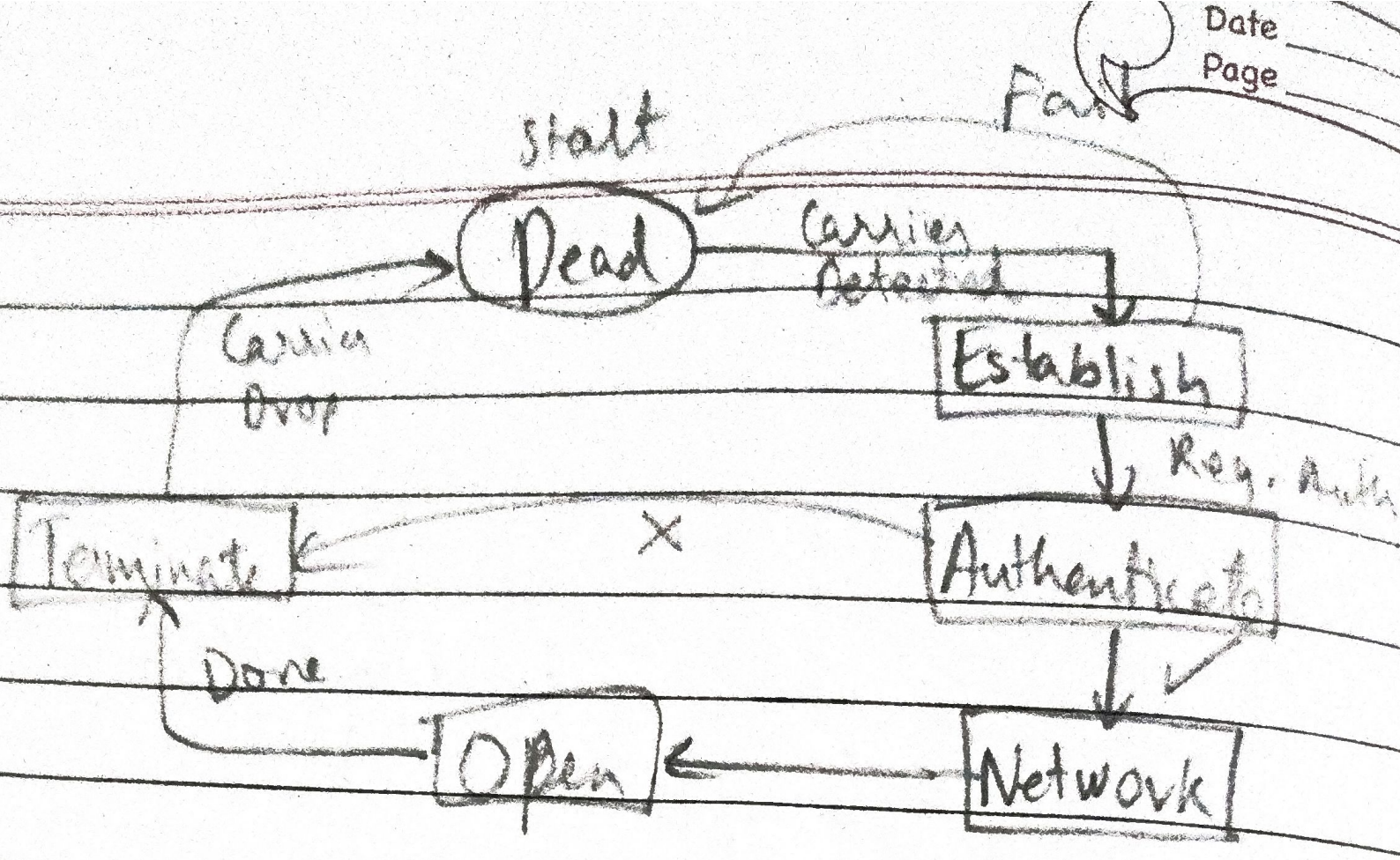
U: " " Info " "



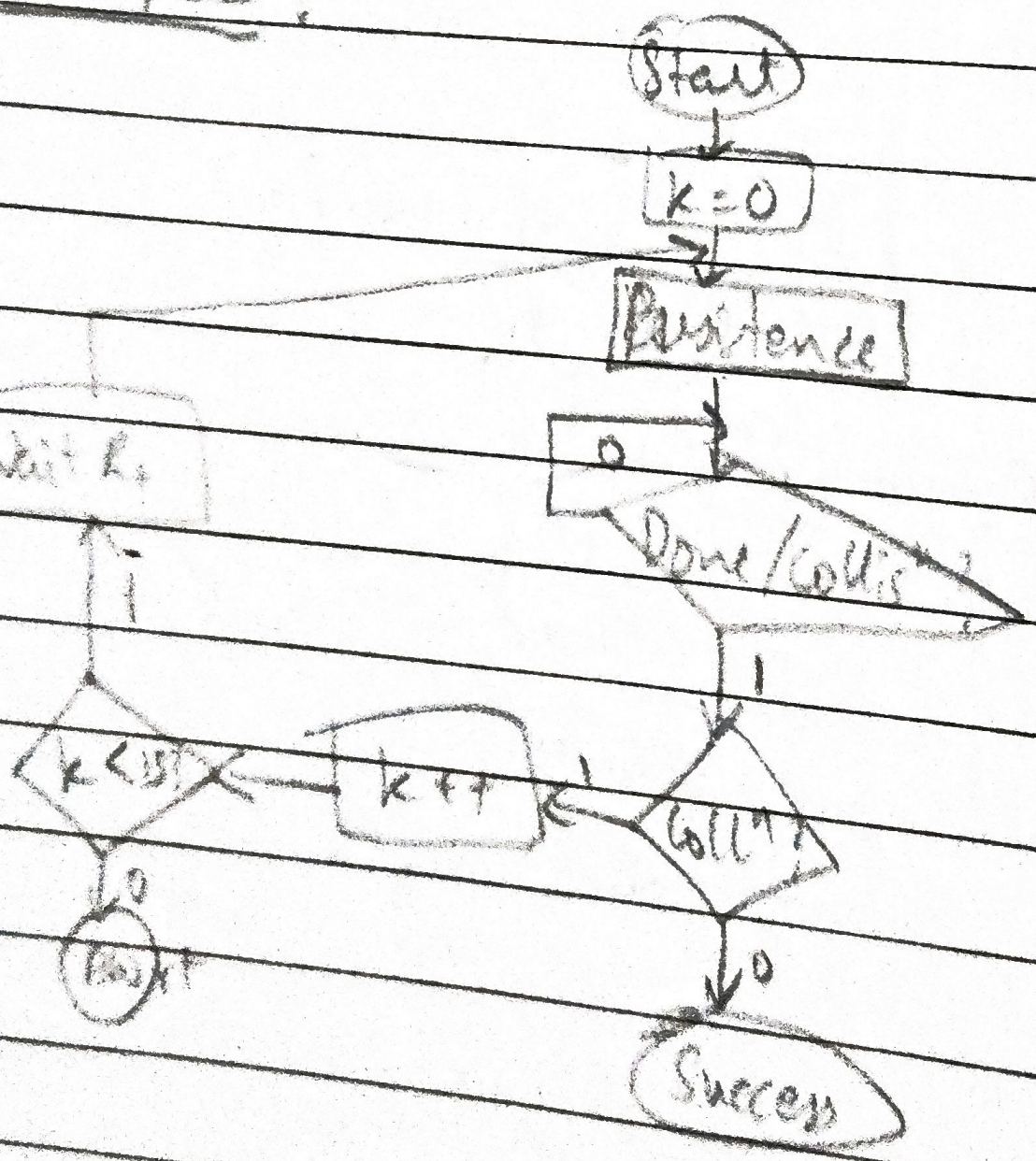

→ PPP



(Same) Flag Addr Cont Prob. Info FCS Flag

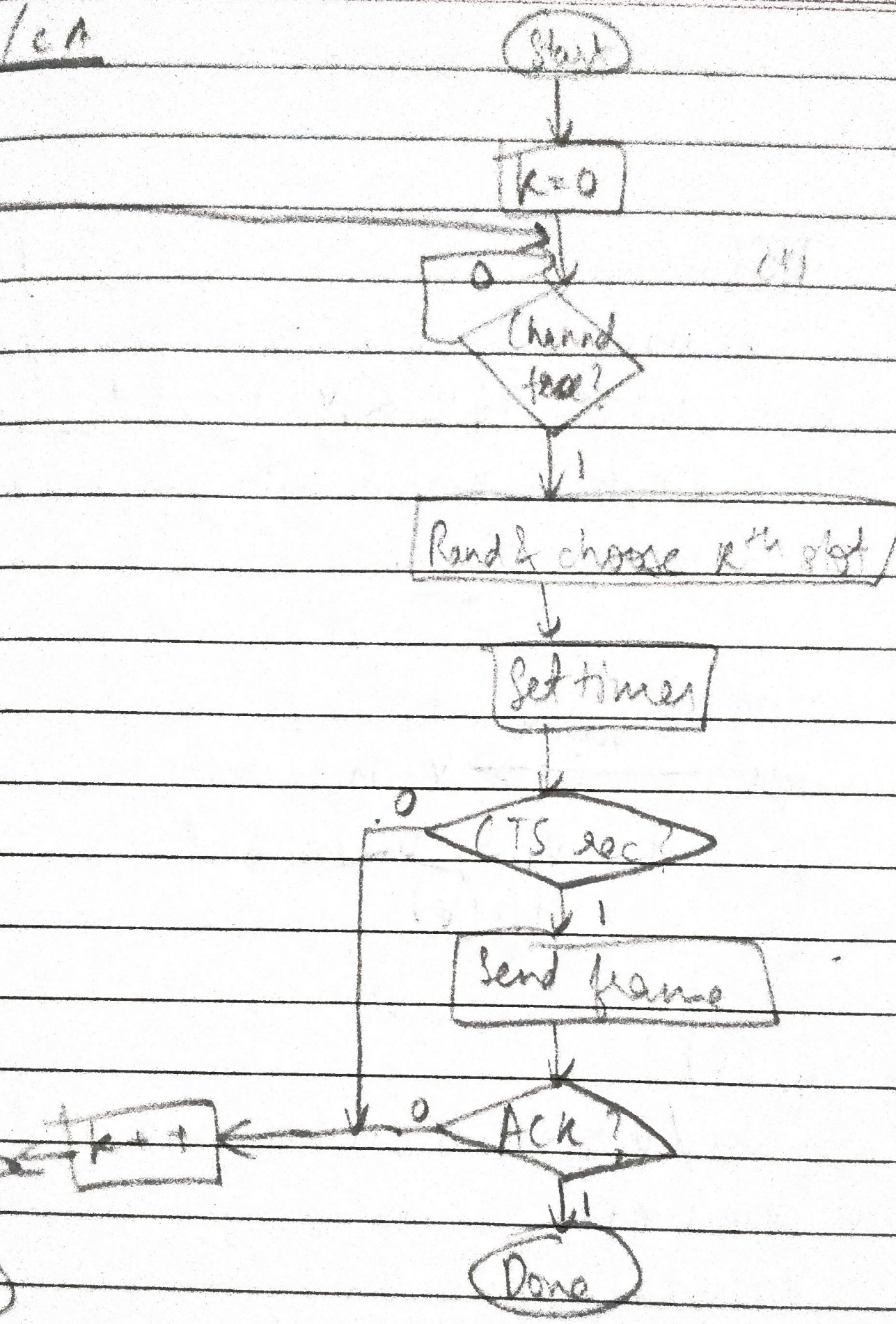

Start
Fail
Dead
Carrier Detected
Carrier Drop
Establish
Req. Auth.
Terminate
Authenticate
Done
Open
Network

→ CSMA/CD :


Start
k=0
Persistence
0
Wait k
Done/Coll?
1
i
k<15?
k++
Coll?
1
0
Fail
0
Success

Date
Page

→ CSMA/CA




Packet switching
- store & forward
- Cut through

CN

→ **TCP/IP model**: Network, Internet, Transport, Application
~~HTTP/FTP~~ Wifi, Ethernet IPv4, IPv6 TCP, UDP HTTP, FTP

→ **OSI**: P D N T S P A

- P - bits (raw), [USB, ~~B~~ ✱]
- D - error free, detects, bits → frames, MAC handling, [Ethnet, wifi]

N - Routing, logical addr:ing, best path [IPv4, IPv6]
T - Complete data transfer, Error correct$^{n}$ [TCP, UDP]
S - Manage sessions [✗]
P - Translates data, encrypts, compresses [ASCII]
A - Interface [HTTP, FTP]

→ **Data Link Control (DLC)**

→ Framing
→ Byte Stuffing
flag — Data (+esc) — flag (flag = 01111110)
→ Bit
01111100 → 011111010

→ Flow control
→ Err. control → Connection + less
→ Protocols: Simple, Stop & Wait, Go Back N, Selective Repeat

→ Store & forward packet switching : rec, err$^{?}$, forward
Cut-through " " : rec, forward

classmate
Date
Page

## Services to Transport layer:

Logical addrsing, routing, flow cntl, data → packet, connec" + servi
datagram, virt. ckt.
Connec". + service - connec" ID-en tables (Virt.)
" - " - full routing " (Datag.)

→ Routing algo's:
- Non-adaptive:
  - Flooding
    - Uncontrolled
    - Controlled
    - Selective
    - ~~Adapt~~ Random Walk.
- Adaptive
  - Isolated
  - Centralized
  - Distributed
- Hybrid
  - Dist. Vec. routing: routing table shared w/ neighbours. Bellman Ford, time-based updates, slow
  - Link state " " " " " all routers, Dijkstra's, change- " " " fast.
- Hierarchical, Broadcast, Multicast.

---

→ Congestion Cntl: What?, Benefits, Types - Open loop (Admi", Window, Ack, Retrans, discarding policies), Closed loop (Backpres, Choke, Implicit, Expl" (forw & backw signals)

- Virt. subnets : Closed loop. No new connec.ⁿ, congested router doesn't participate, negotiate parameters.

- Datagram subnets :
  - Closed loop
    - Warning bit : Piggyback bit on ACK
    - Choked packet : What?, steps?
    - Load Shedding : Deliberate pack discard, due to resorces, when?, working?,
  - Open loop
    - Random Early Detection : Drop/mark packets before buffers fill (avoid cong.)
      - How : Check avg. buffer size def. min. & max thresh, randomly discr packs past a threash., (prob.).
      - + : no onoff, no global sync, smooth traffic,
      - − : thresh. tune, only for big buffers,

→ QoS : Prioritizing traffic
- Components : Traffic classif.ⁿ & priorit.ⁿ, Shaping/Policing, Cong.ⁿ mgmt, resource resv.ⁿ, Avoidance
- ± : perf.ᶜ, UX, Netw. Util.ⁿ, cost, flexible, reliable & stable
- Param.'s : Bandw, Lat.ᶜʸ, Jitter, Packetloss, Throughput, Priority, Classif.ⁿ

→ Leaky bucket :
- Working : Buffer with fixed cap., packets arrive, buffer fills, at fixed rate packets are sent, if full discard packet
- Used : Shapes traffic, limits rate, cong.ⁿ cntl.
- + : Simple, smooth, no bursts
- − : delays, nonadaptive, no priority system

→ Packet sched.: FCFS, priority, RR, DRR, FQ, WFQ

→ Token bucket

→ Resource Rsrv^n: Pre-allocating resources for perfomance.
eg. Bandw, Buffs, Queue.

→ Admiss^n cntl: Check current resources before deciding to allocate some for a process (Components, implementation
policies, points, protocols, monitoring — static, dynamic, QoS, Policing)

→ Intserv: Reserves resources along path before transmission
↳ Comps: RSVP (PATH & RESV), Classify traffic, Admns. cntl.

→ Best effort: No guarantees, simple, IP-level

→ Diffserv: Each packet is marked based on type, depending on which nodes decide handling
↳ Perhop behaviors: Expedited, Assured forwarding & Best effort (aka regular traffic)

→ IPv4

| | | | |
|---|---|---|---|
| Version (4) | HLEN (4) | Service type (8) | Tot. len. (16) |
| Datagram ID (16) | | (3) DF MF | Fragment ID (13) |
| TTL (8) | Prot (8) | | Checksum (16) |
| Src. addr. | | | |
| Dst. " | | | |
| Options | | | |

→ **IP addr**: Class A: ¼ Network, ¾ Host; B: ½; C: ¾; D: Multicast; E: RESV

→ **Tunnelling** - Encap. a protoc. into anoth (for sec. & compatibility | Eg. VPN, SSH, IPsec)

→ **Fragmentn** - 2 types: transpt non-.

→ **Subnets**

→ **Classless IP (CIDR)** - Inter domain routing, variable /

→ **Network addr. Transl$^n$**: Private netw's access internet; table is kept

→ **ICMP** (msg): Error rep$^n$ packets (echo req & reply, unreachable, TLE, re-dir., param. prob.)

→ **Addr. Resol.$^n$ Protoc. (ARP)**: Interface w/ MAC addr, broadcast req, unicast response
(Req. → reply → cache)

→ **IPv6**:

| Version (4) | Prio. (8) / Traffic class | Flow label (20) |
|---|---|---|
| Payload len (16) | | Next header (8) | Hop limit (8) |
| Src | | |
| Dst. | | |

→ **RARP** - to get IP addr, send req to RARP server (broadcast) & receive IP (unic)

→ **Dynamic Host Configuration Protoc**: Leases IP's for some time.
Discover → Offer → Request → Ack.

→ Bootstrap protoc.: Assign IP. | Device send IP req (broad) → router → Relay Agent → Other subnet → Bootp server (uni) → relay agent (uni) → device | Static table

→ Open Shortest Path First (OSPF): Link-state, Hierarchical design (areas), Uses cost metric (converges) | Link state ad's, Dijkstra's, Discovery of neigh. → DB sync (internal) → Topo. calc. → Update

→ Border Gateway Protoc.: Dist-vec, connect Autonomous (names) syst.'s, external

---

→ UDP: x-connec", no order

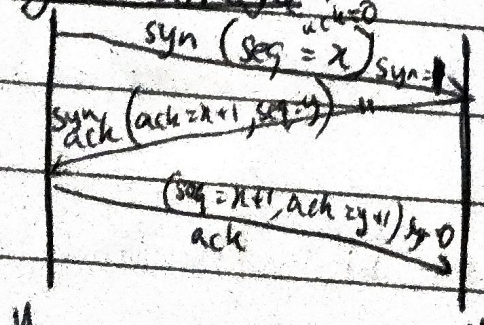
| Src port (16) | Dst. port (16) |
|---|---|
| len (..) | checksum (..) |

→ TCP: connec". + full duplex, header - 20 bytes, payload - $2^{16}$ bytes (Disassembles packets)

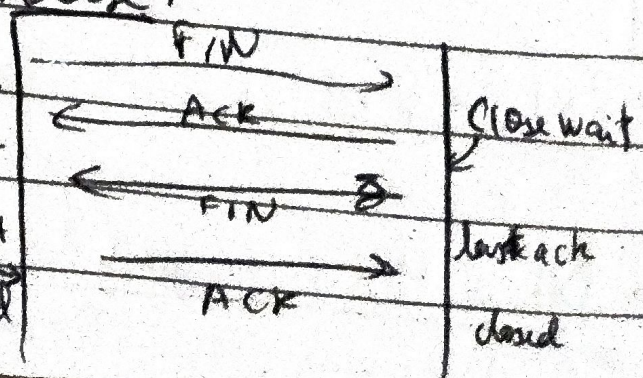
CNTL
URG
ACK
PUSH
RST
SYN
FIN

| Src. port | Dst. Port |
|---|---|
| Seq. no (pos" of 1st byte in seg.) | |
| Ack. no (next expected byte) | |
| HLEN (4) \| RES (6) | Cntl. flags (6) \| Win Size (Size of buffer) |
| Checksum | Urgent ptr. |
| Options. | |

→ 3-way handshake:




Release:



Date
Page

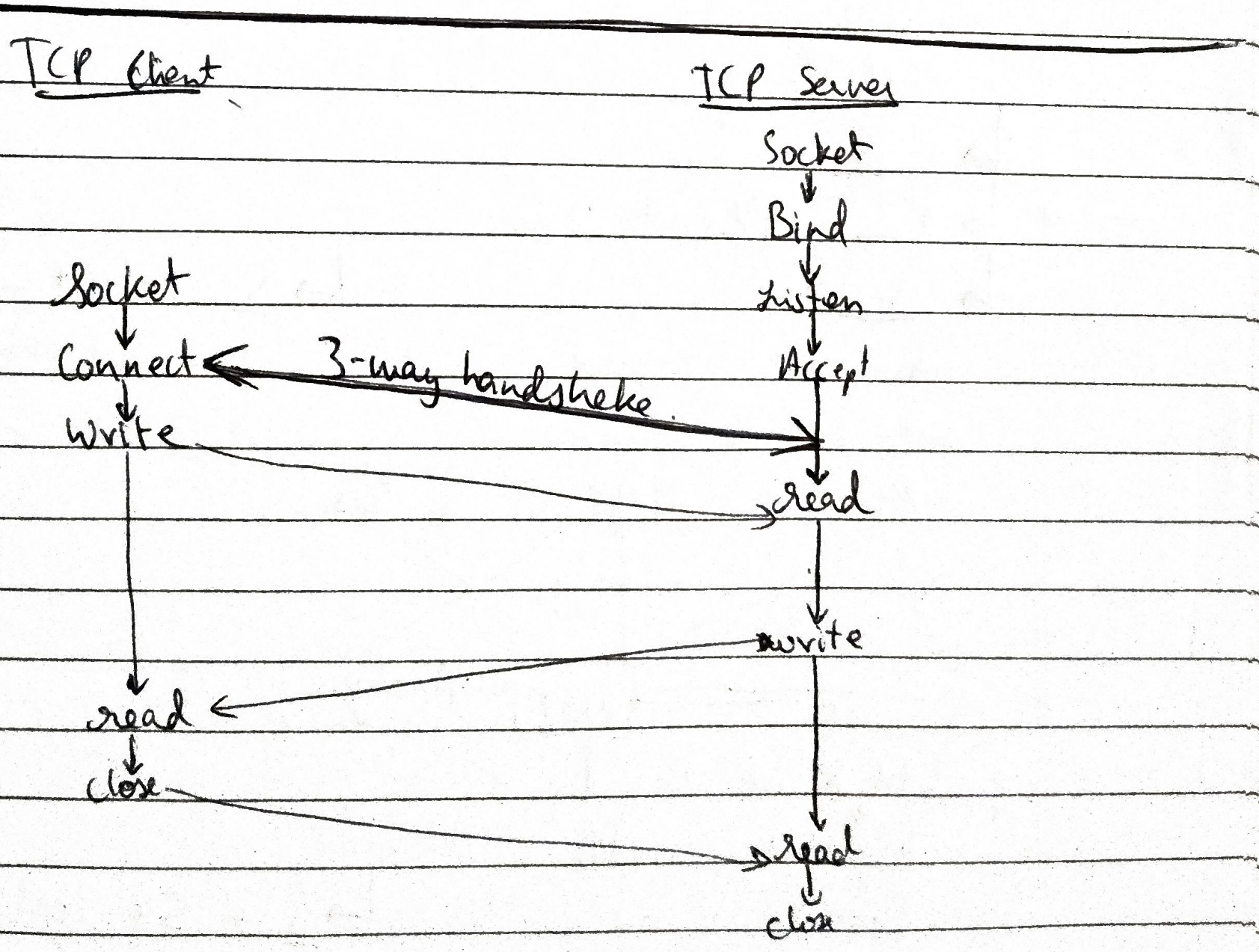
→ **WWW architecture**: Browser, server, internet, HTTP, URL, DNS, Webpages

→ **URL**:
↳ **Structure**: Protocol, Domainname, Port(80 std), path (/...), query (?...), fragment (#...)

→ **Cookie**: Name, val, dom, path, expiry, secure/httponly

→ **HTTP**: Get, post, put, delete, options, head, patch (info, success, redir, client, server)

→ **Telnet**: text based protocol & cmnd line tool, remotely accesses & controls comp's via TCP | Connect, Authenticate, exec commands, close sess. | eg. telnet ...com 23

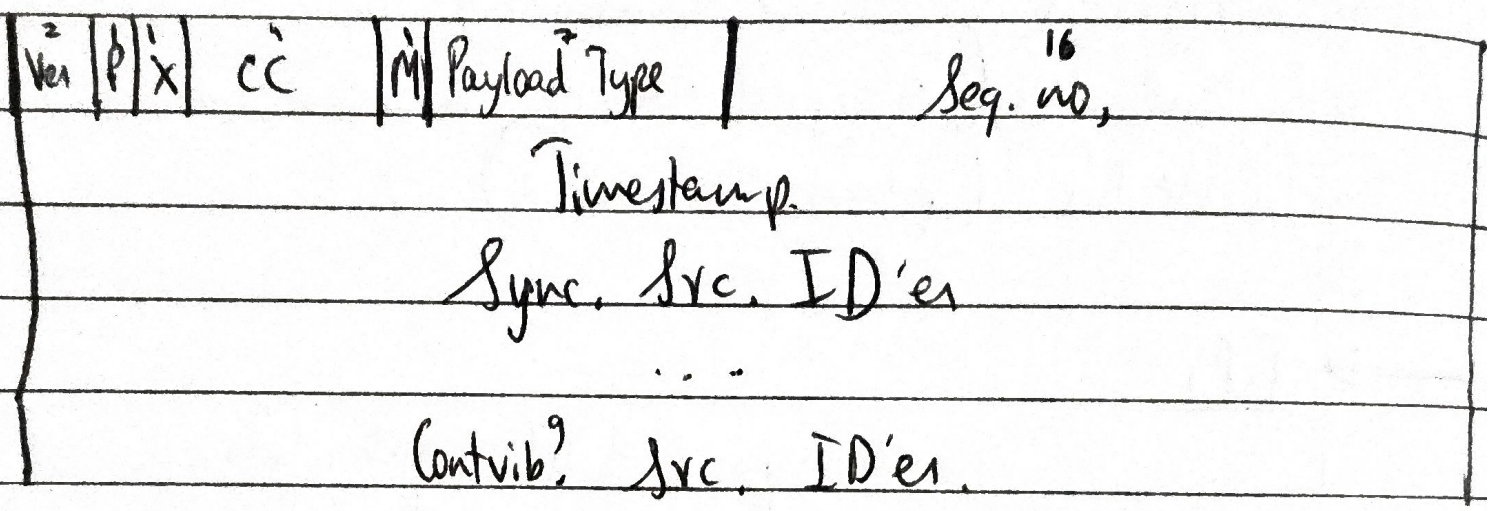


## Remote Procedure Call (RPC):

- Working - (Client call → (stub) Marshalling (serialize) → Send → Server receive → unmarshall → exec. → return → receive response.

## R.T. Transm^n. protoc. (RTP): Data & A/V over IP

- Working - Packetize → send on UDP → Sync w/ timestamps → Seq. no → payload type.




## TCP cong^n. ctrl.

- Steps
  - Slow start
  - Cong^n. avoidance
  - Fast retransmit.
  - Fast recovery

## TCP timers:

- Retransm^n
- Persistence
- Keepaline
- Time-wait
- Delayed ACK