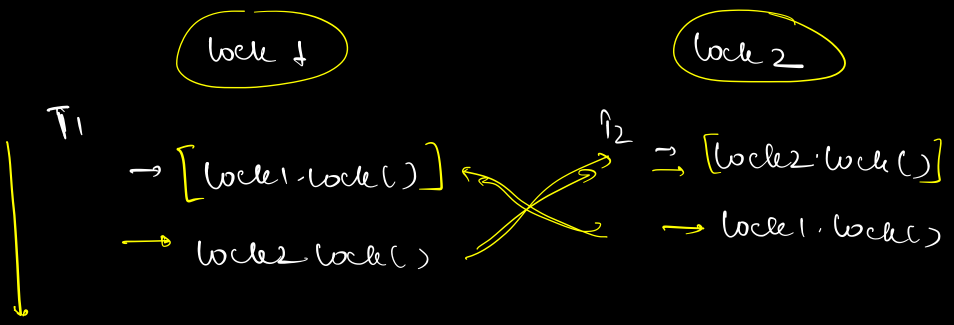
Agenda
- → Intro to computer networks
- → Protocols
- → How Internet is structured
- → Layering architecture
- → OSI model

⇒ **Deadlock :**

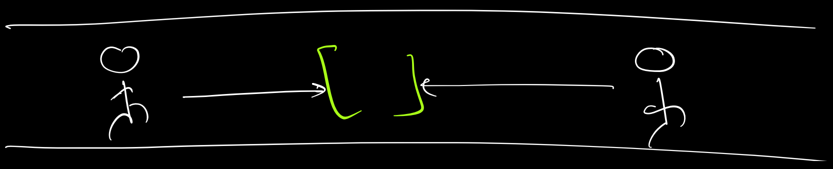
Scenario



Two threads, both waiting because they dont have locks they need to execute and the required lock is taken by the corresponding thread.

So, they end up waiting indefinitely for each other to release locks and complete execution.

This state of indefinite wait is called ⇒ Deadlock



Deadlock :- Situation where the complete system comes to a halt because of threads waiting on locks that they will never be able to get.

=> Sol^n to deadlock :-

* Prevention is better than cure

i) Deadlock avoidance → take locks in order

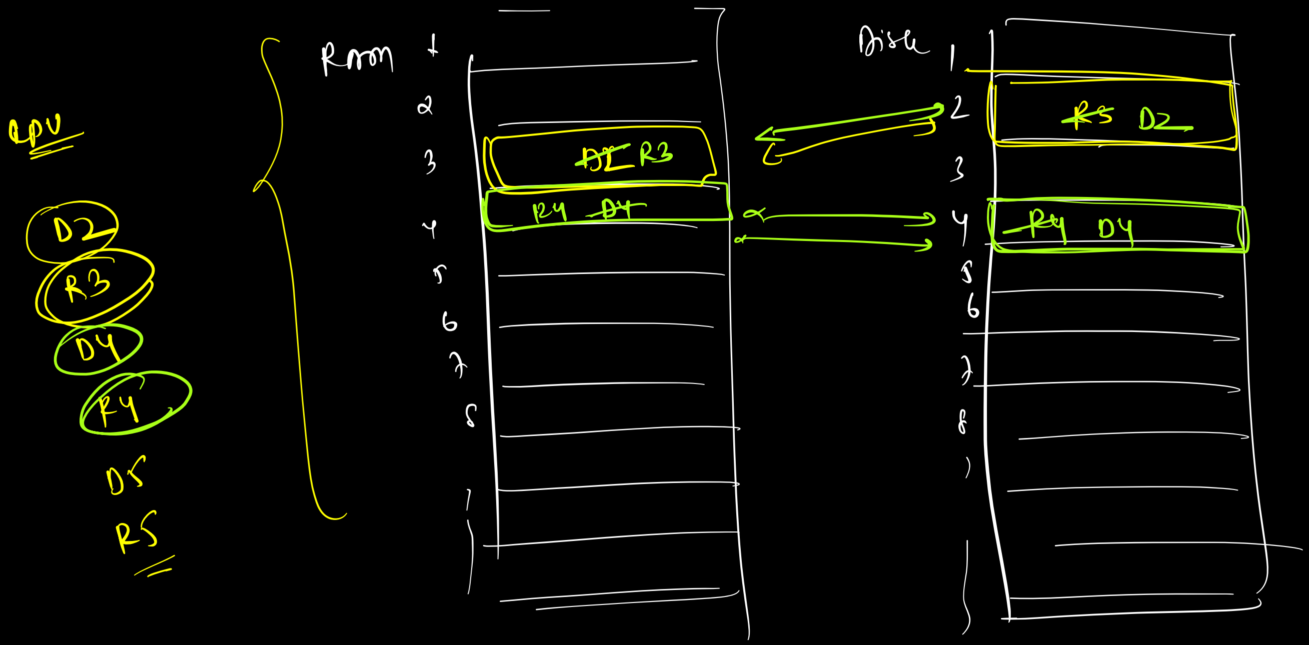
ii) Deadlock recovery → i) wait for a timeout

iii) Deadlock ignorance   ii) graph cyclic dependency → kill

Thrashing :- When you have high amount of page fault, leading to the system only worrying about page faults and not getting any actual done is called "Thrashing".

When it happens : i) Small RAM

ii) bad page replacement algo.



: Intro to computer networks :-

Computer Networks
↳ group of interconnected items

- network → humans
- networks → machines
- network → railway

Network of interconnected computers ⇒ Internet

⇒ why computers need to connect to each other?

i) Communication

ii) Sharing resources
- Aws → CPU | Ram | HDD - —
- data

⇒ History of internet :-

i) US Army

ii) ARPA ⇒ Advanced Research Projects Agency

iii) ARPANET

iv) Started due to the need of army to communicate from one end to other.

⇒ why is studying about internet is interesting :

Java ⇒ System.out.println("Hello world");

C++ ⇒ cout "Hello world"

Python ⇒ print("Hello world")

Js ⇒ console.log("Hello world").

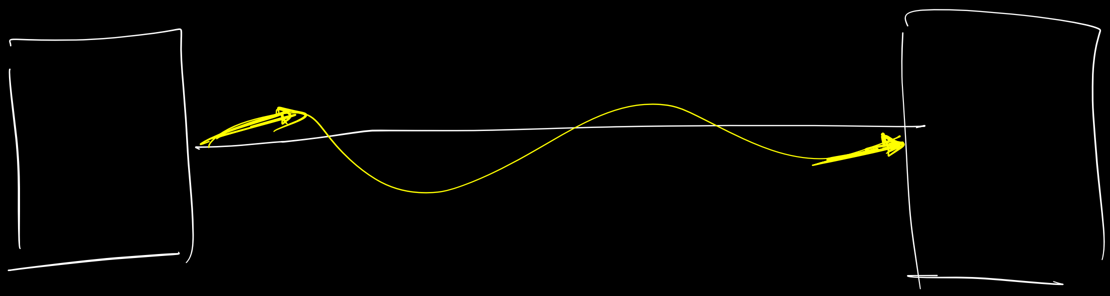
i) Computers understand only binaries.

ii) Computers are not smart | dumb. → cant handle uncertainty

Specified language, set of rules ⇒ grammar

$\downarrow$

protocols



Protocols: Set of rules that govern the transmission of data, b/w 2 computers

* Protocol - be known on both ends.

ex ⇒ TCP / IP → Transmission control protocol
Internet protocol

FTP - file transfer protocol

HTTP - Hypertext transfer protocol

HTTP(S) → " " " Secured

SMTP - Simple mail transfer protocol

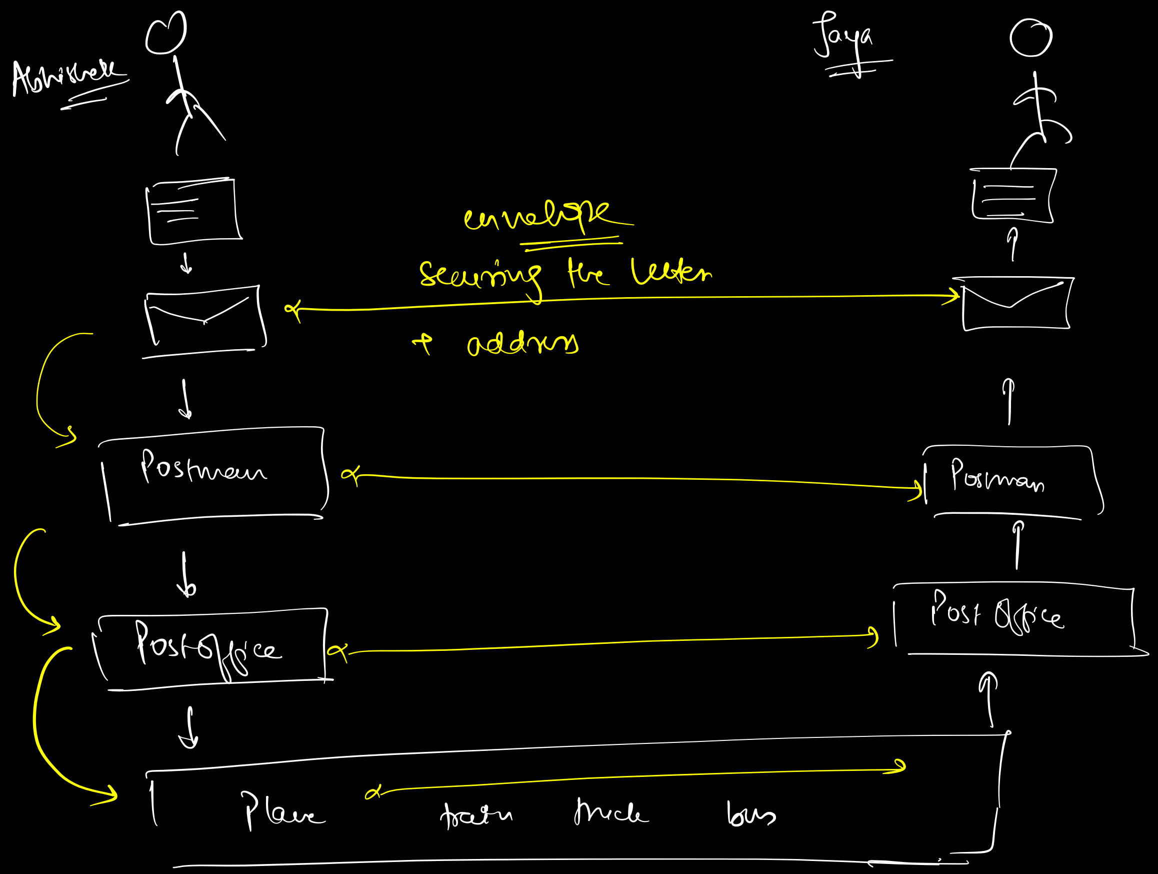
UDP - User datagram protocol

IETF website :- Internet Engineering Task force

↳ maintains a list of RFC
↓
Request for change / comments
↓
proposal of change

All protocols that we use have RFCs, all changes are done via RFCs.

How IP works ⟶ How post service works.



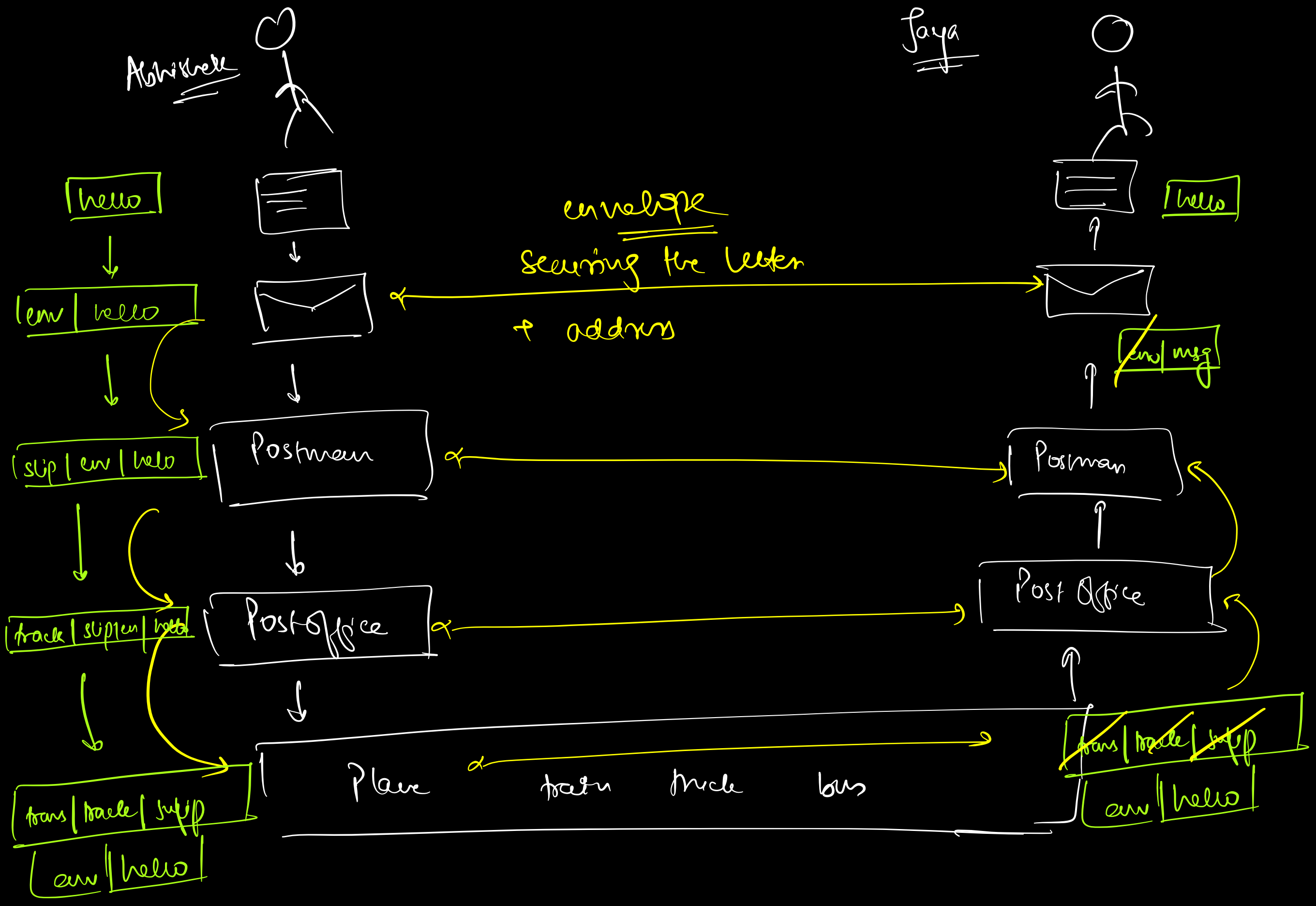
layered architecture :- Complex systems are broken into several layers, every layer data does some amount and delegates the rest.

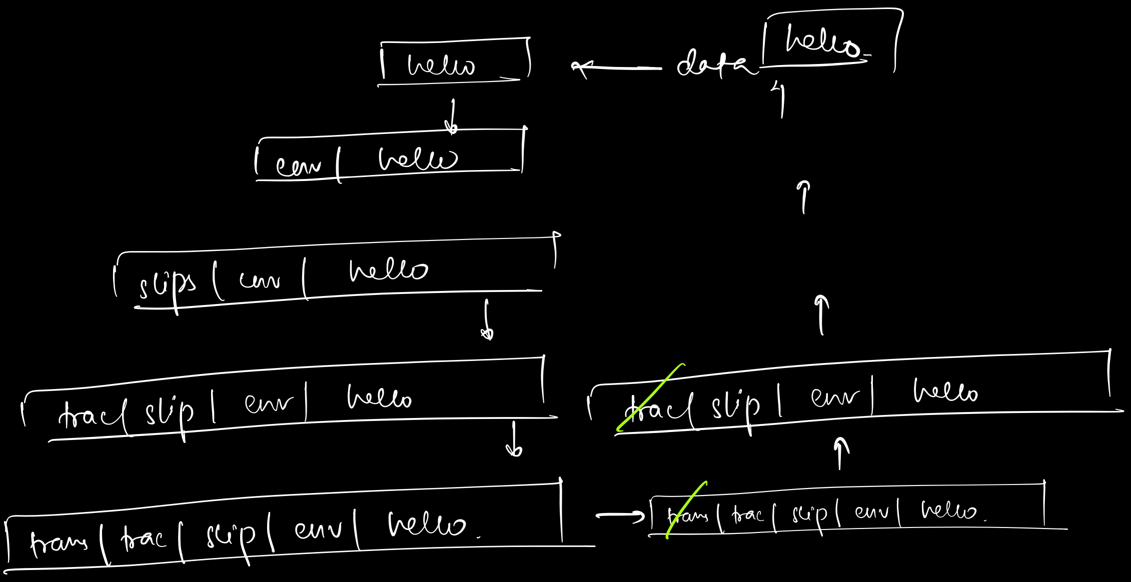
→ Upper layer does the work it specialises in, and passes the info to the lower layer to do the remaining work

→ As each layer doesnt need to know the impl of other layers, it allows abstraction

⇒ Because of well broken down responsibilities across each layer, and abstraction, it becomes easier to under complex systems.

INTERNET ALSO WORKS ON LAYERED ARCHI.


Abhishek
Jaya
hello
env | hello
slip | env | hello
track | slip | env | hello
trans | track | slip
env | hello
envelope
securing the letter
+ address
Postman
Post Office
Plane
train
truck
bus
~~env~~ | msg
~~trans~~ | ~~track~~ | ~~slip~~
env | hello
hello


hello
data
hello
env | hello
slips | env | hello
track | slip | env | hello
trans | trac | slip | env | hello.
~~trac~~ | slip | env | hello
~~trans~~ | trac | slip | env | hello.

⇒ OSI model

OSI ⇒ open systems interconnected

Application → generates data to be sent forward

↓

Presentation → In what format data will be sent
→ encrypt / decrypt

↓

Session → manages user session
→ login / logout

↓

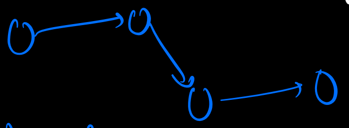
POSTMAN

Transport → how to send the data
→ features for sending the data

ex ⇒ guaranteed delivery, In order delivery

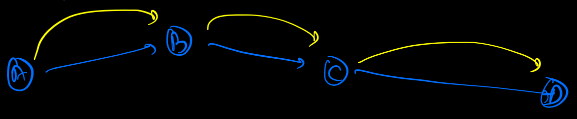
↓

Network IP ⇒ Routing
→ creates the map



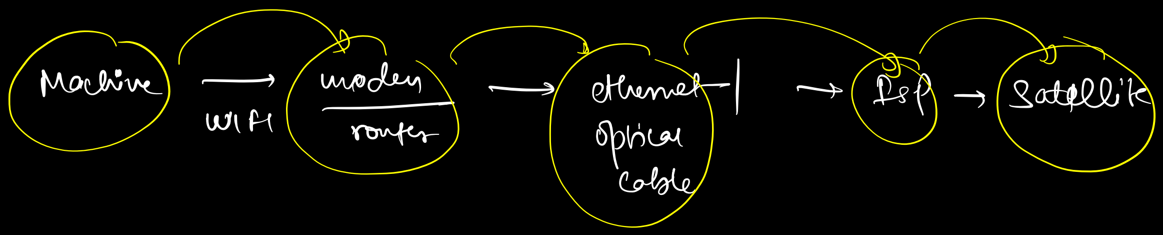
↓

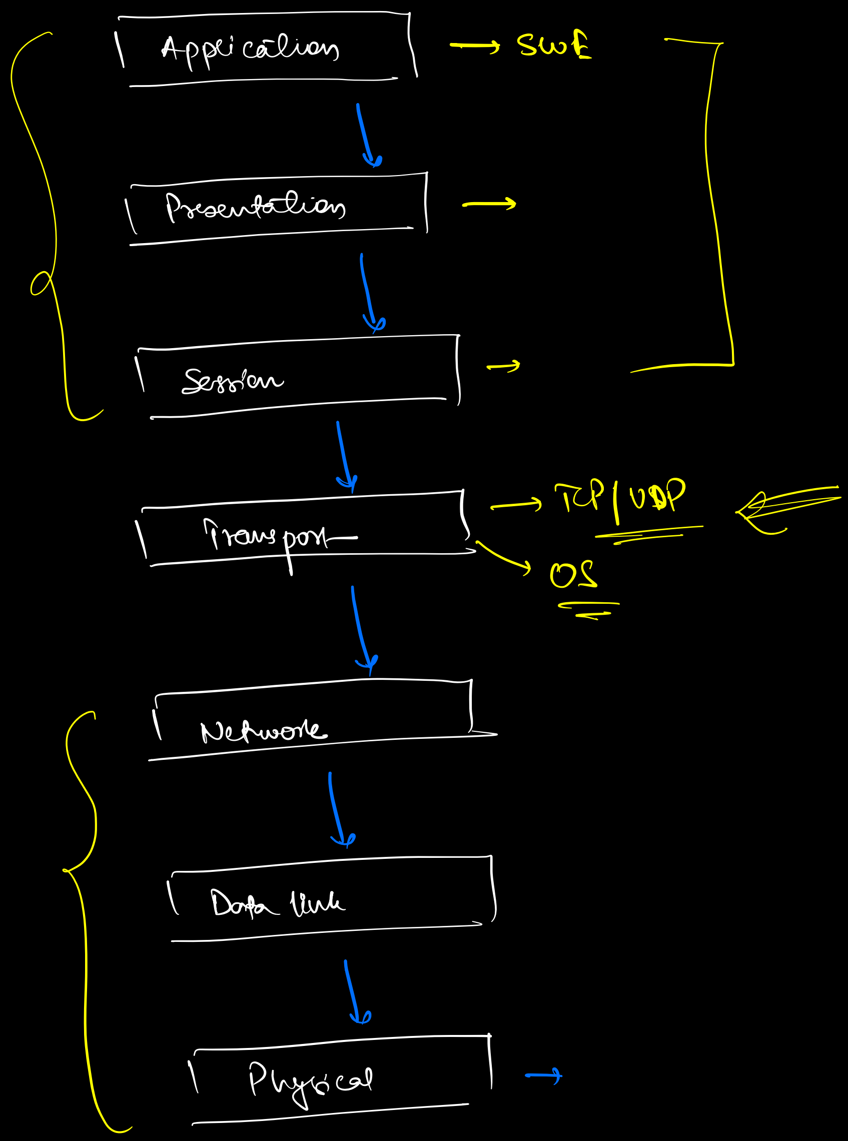
Data link → transfer the data b/w 2 nodes



↓

Physical → travelling through devices
wifi —
ethernet —




Application
SWE
Presentation
Session
Transport
TCP / UDP
OS
Network
Data link
Physical

TCP/IP model → practical model

$$\begin{tikzcd}
\boxed{\text{Application}} \arrow[r] \arrow[d] & \text{app}^n + \text{presentation} + \text{session} \\
 & \text{(HTTP, SMTP, FTP)} \\
\boxed{\text{Transport}} \arrow[r] \arrow[d] & \text{(TCP / UDP)} \\
\boxed{\text{Network}} \arrow[r] \arrow[d] & \text{IP} \Leftrightarrow \text{internet protocol} \\
\boxed{\text{Data Link}} \arrow[r] \arrow[d] & \text{ARP, VLAN, STP} \\
\boxed{\text{Physical}} \arrow[r] & \text{Hubs, Fiber, Ethernet.}
\end{tikzcd}$$