# CS3.301 Operating Systems and Networks

## OS and Networks: Run Down and Concluding Thoughts!

**Karthik Vaidhyanathan**

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

SERC
Software Engineering Research Centre

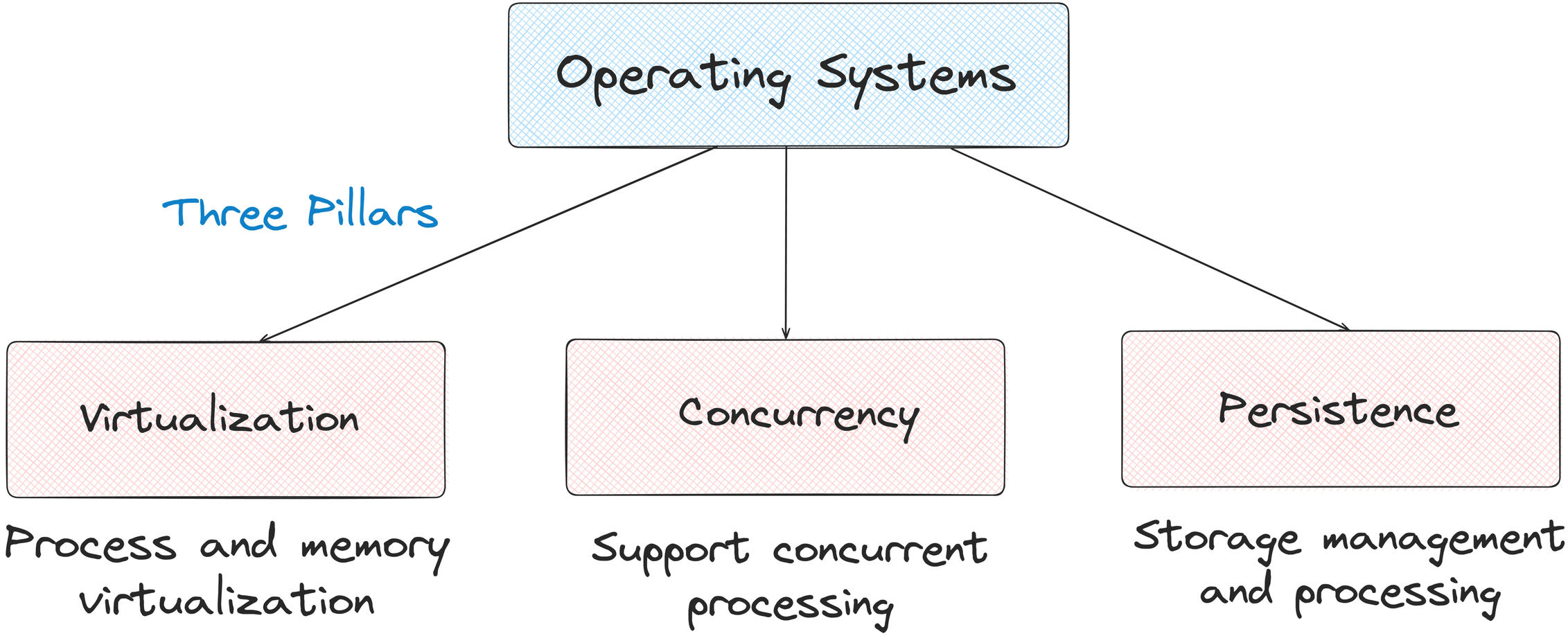
# Acknowledgement

The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidhyanathan

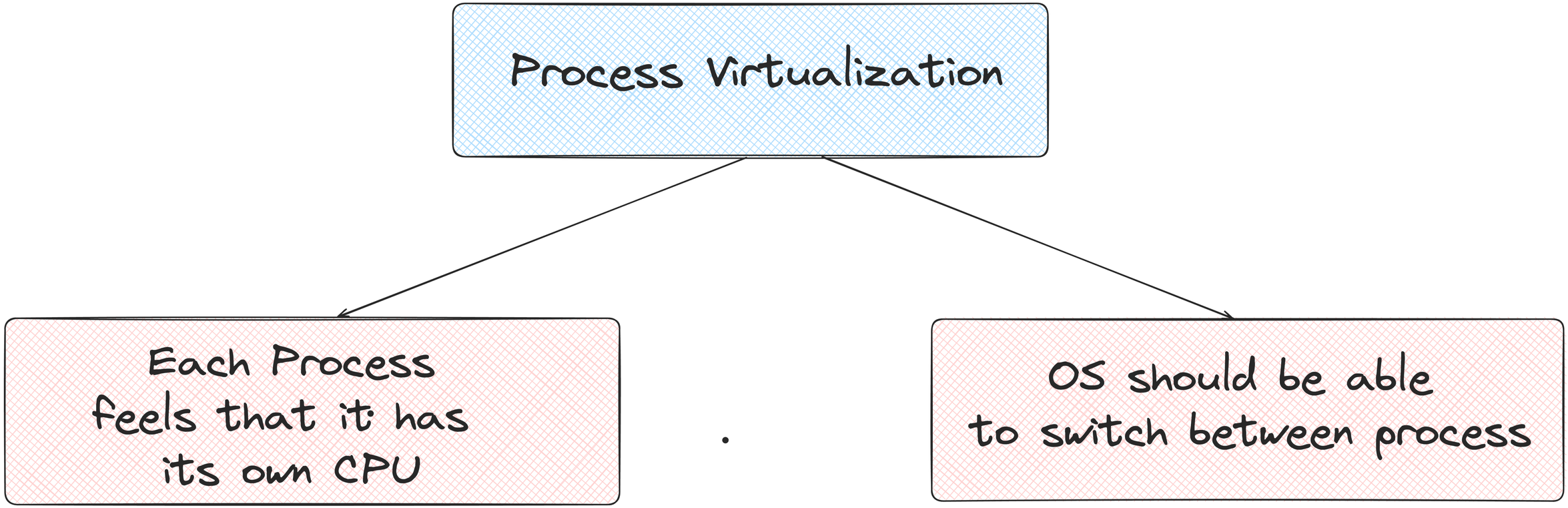
Sources:

- Operating Systems in Three Easy Pieces by Remzi et al.
- Computer Networks, 6e by Tanebaum, Teamster and Wetherall
- Computer Networks: A Top Down Approach by Kurose and Ross
- Computer Networking essentials, Youtube Channel
- Other online sources which are duly cited
- Different materials used throughout the course!

# OS: An Overview

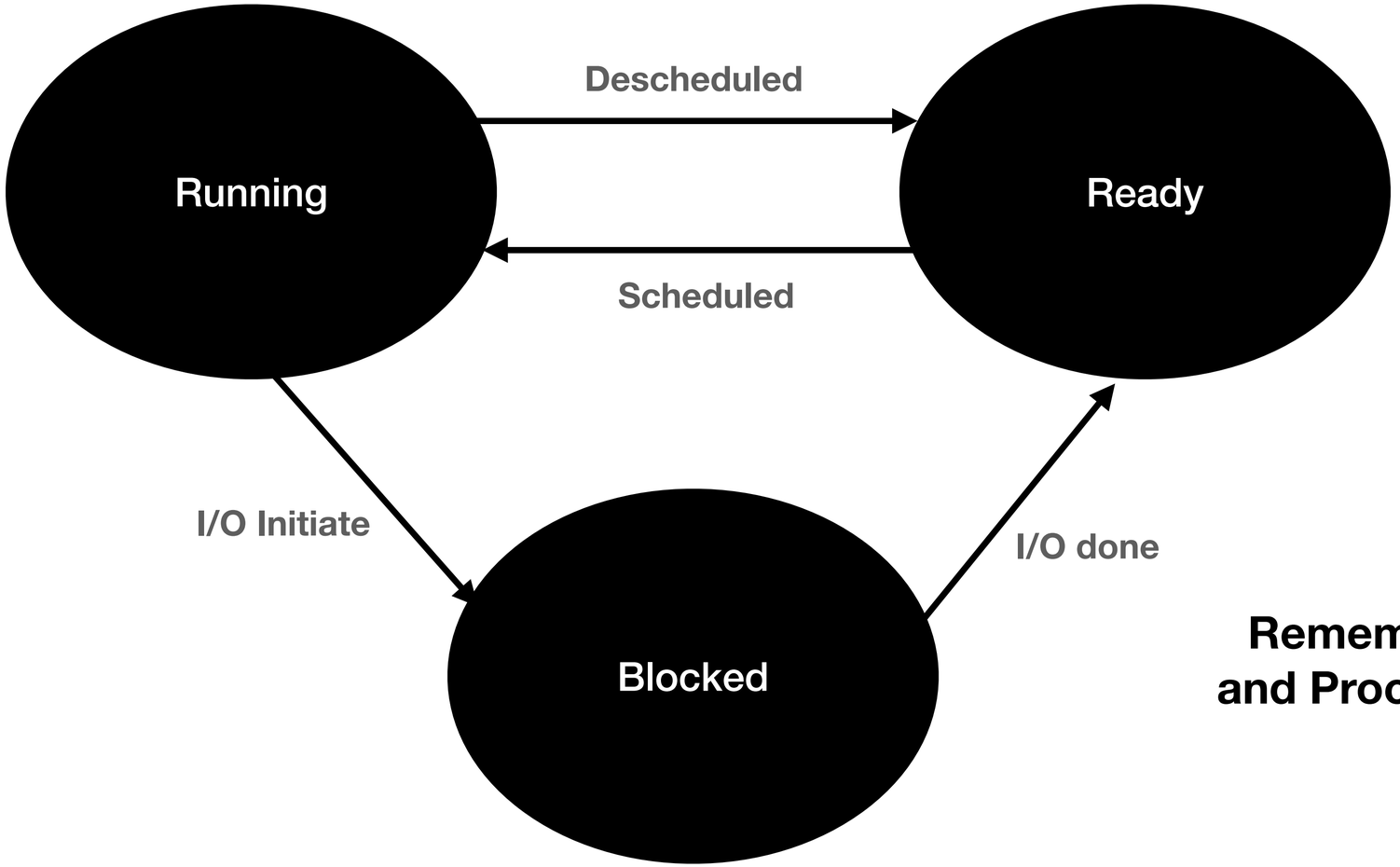


# Process Virtualization



# States of the Process

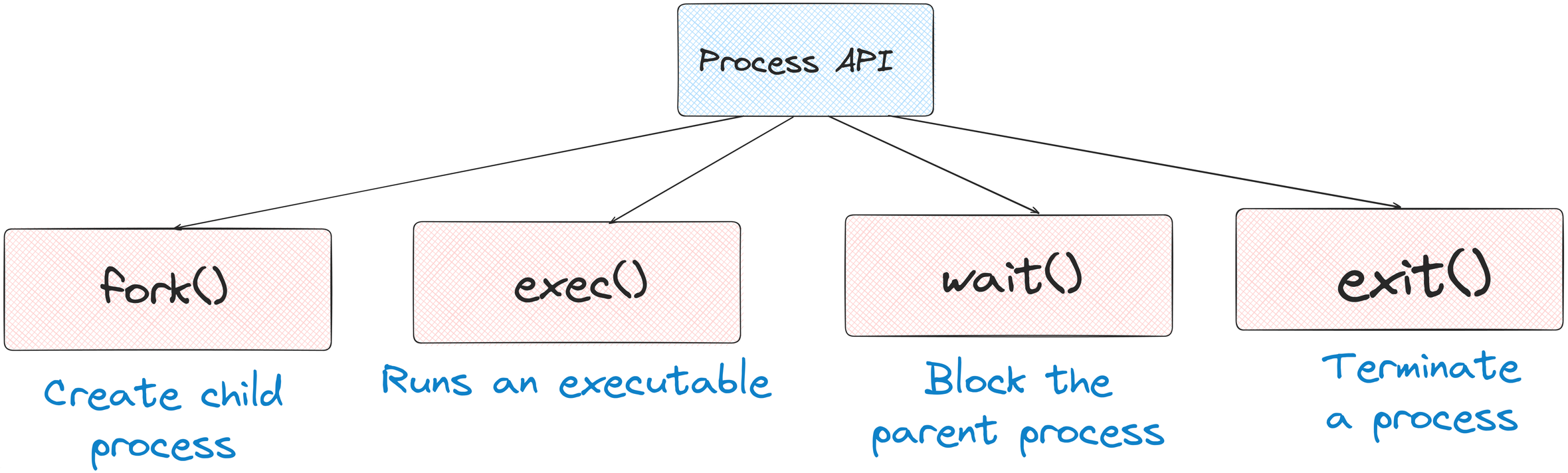
## Process State Transitions



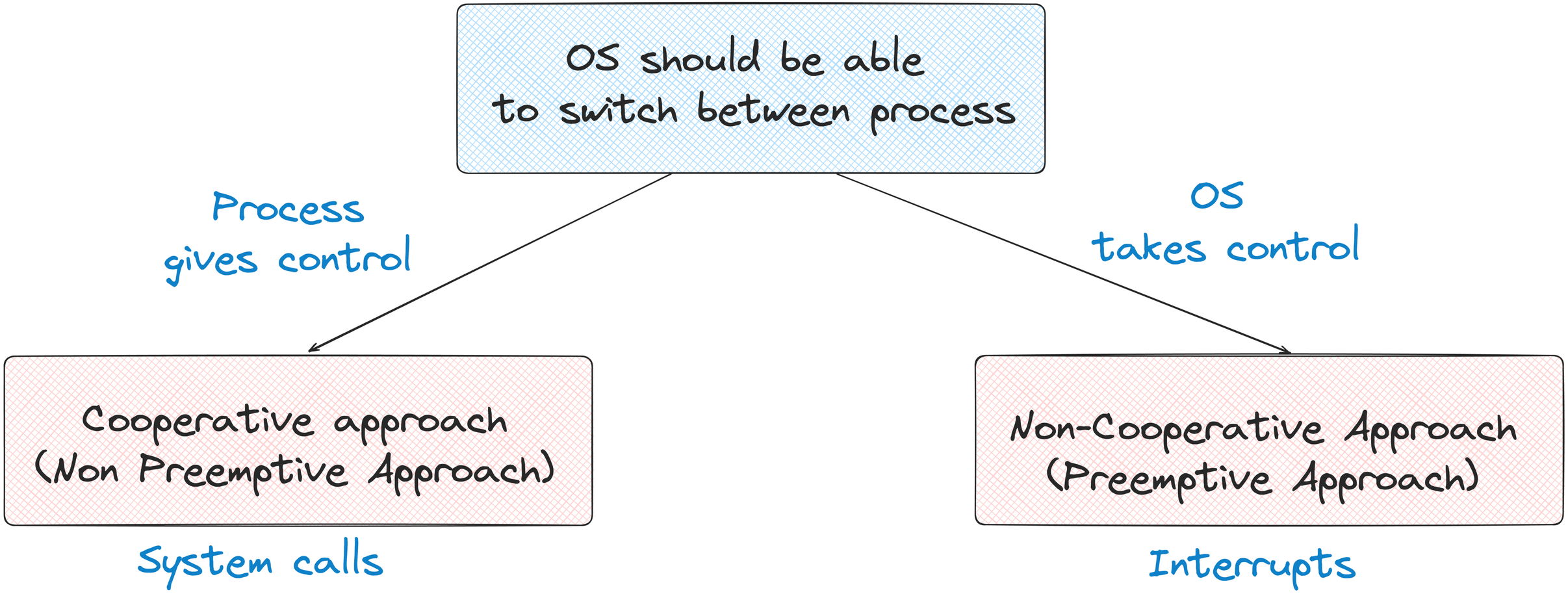

**Remember: Process lists and Process Control Blocks**

# Process Management API

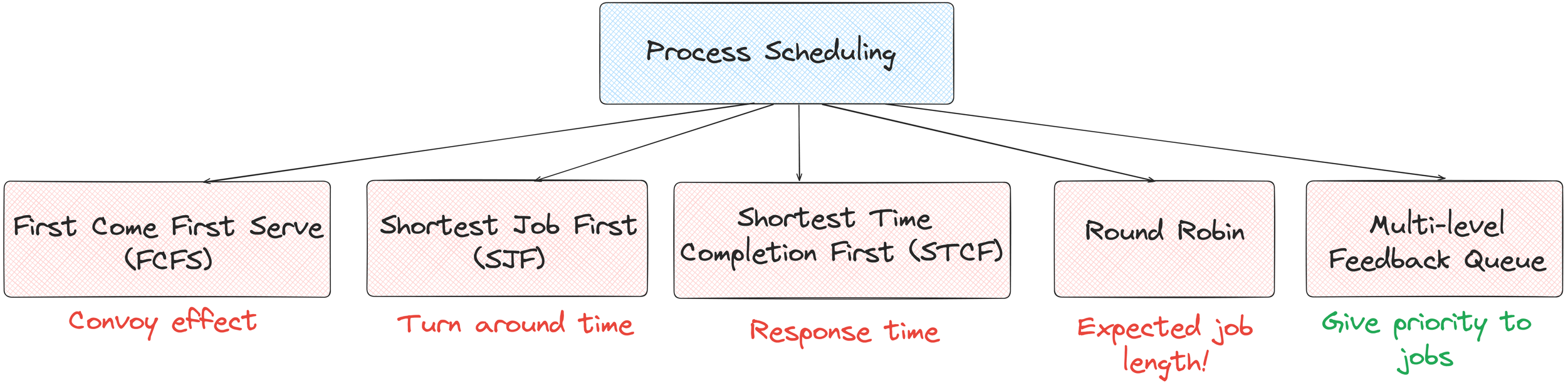
Memory image of a process - **Code, data, stack,** and **heap**



# Switching Between Process



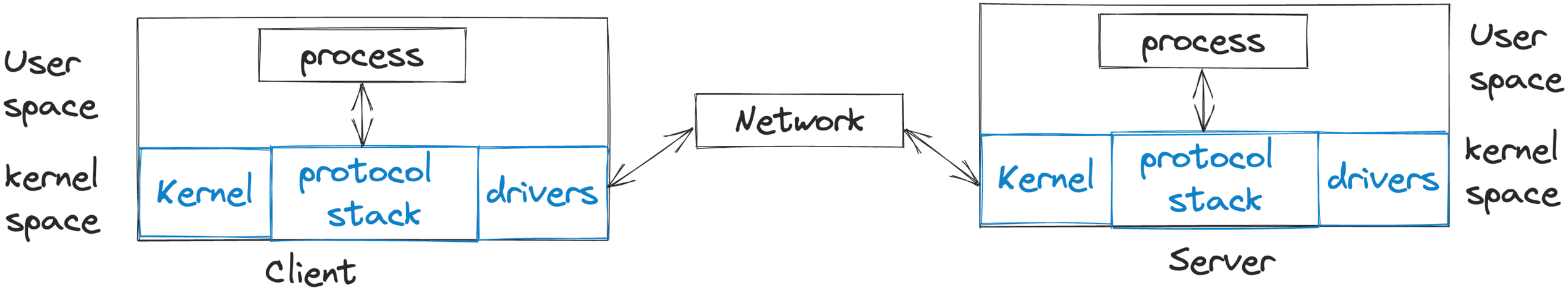
# Process Scheduling



# What if Process wants to communicate?

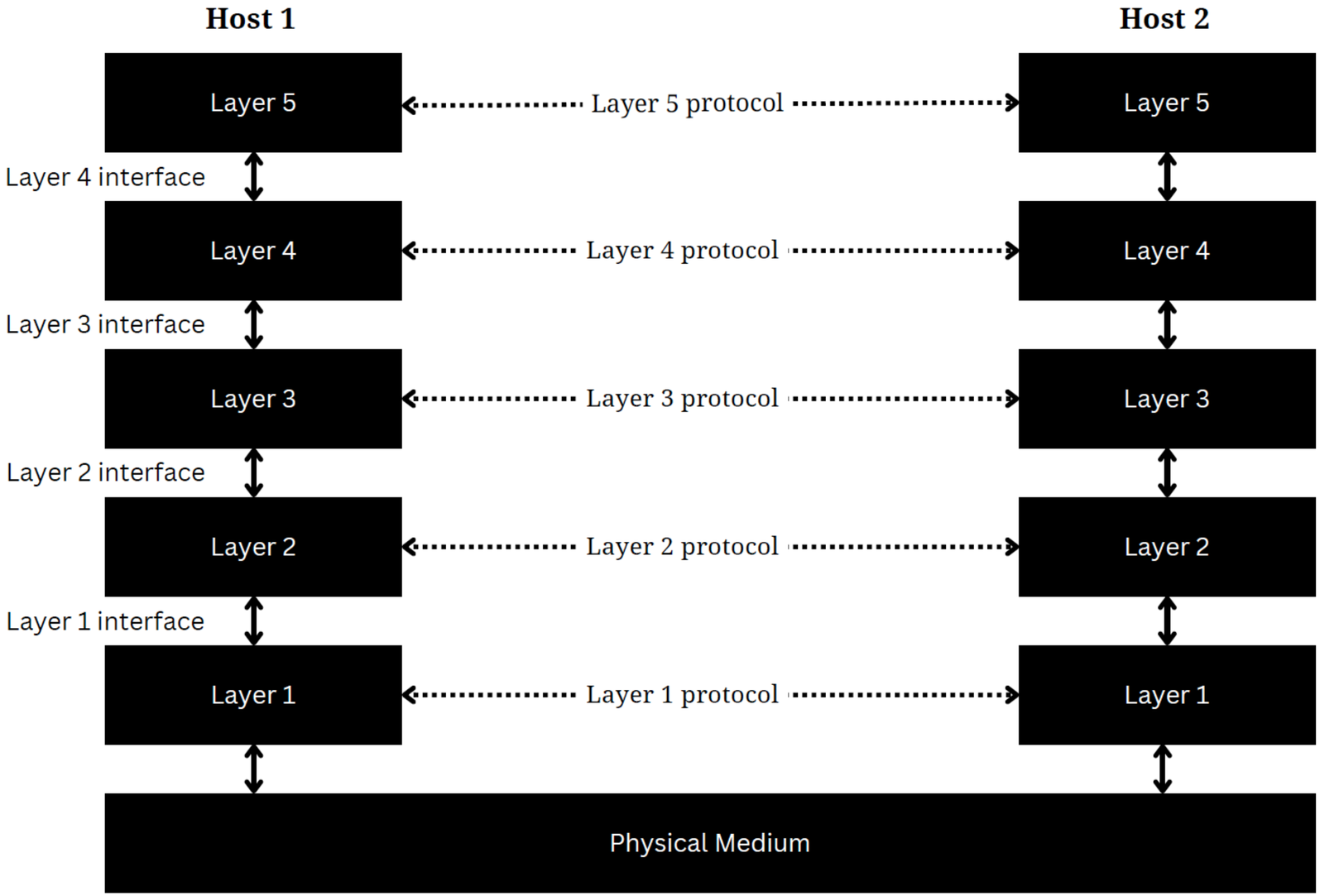
- Process A (eg: Whatsapp) is executing in Host 1
  - Process B (Whatsapp) is executing in Host 2
- Host 1 will have an address, same is the case with host 2
- How to ensure the data reaches from Host 1 to Host 2?
  - What all needs to be considered?
  - Remember: There will be multiple processes that are executing in a host

# The role of Operating System

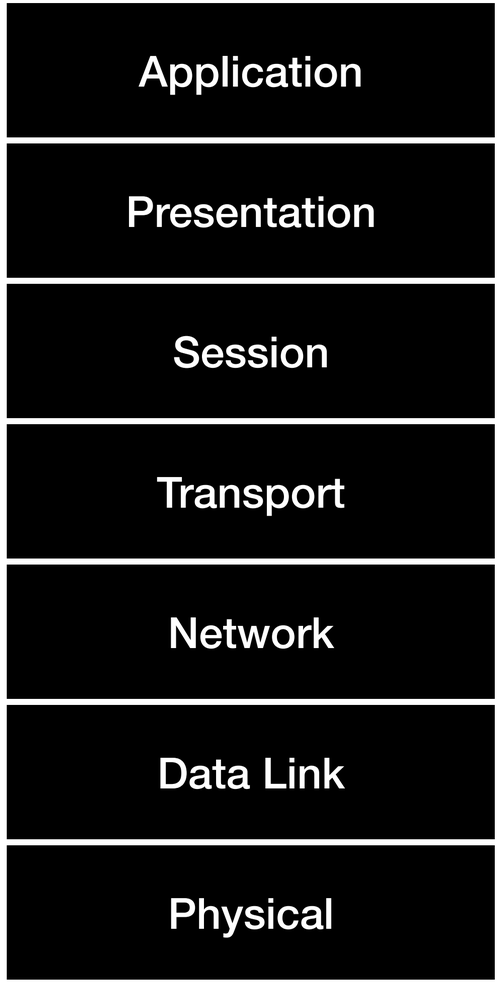



- Software component in the OS that supports network calls - **Protocol stack**
- Provides Service primitives which are nothing but system calls - **Some API?**

# Networking Layers



# The OSI Model

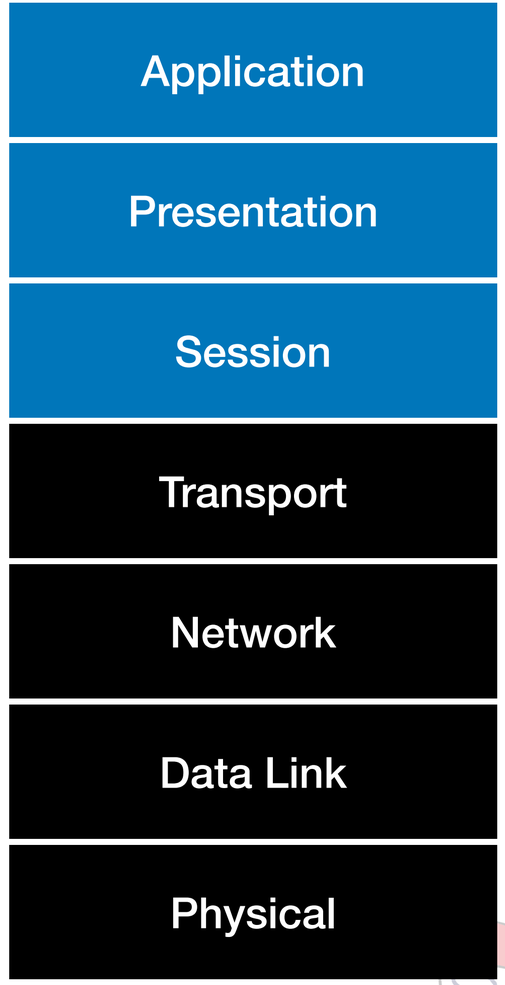



- Open System Interconnection (OSI)
- A Conceptual framework used to understand how communication works through different layers
- Divides the network communication process into seven layers
- Developed to facilitate interoperability between different technologies
- Each layer has a specific function. If they all do what they are supposed to do => sharing of data

# Session, Presentation and Application

## Application to Application

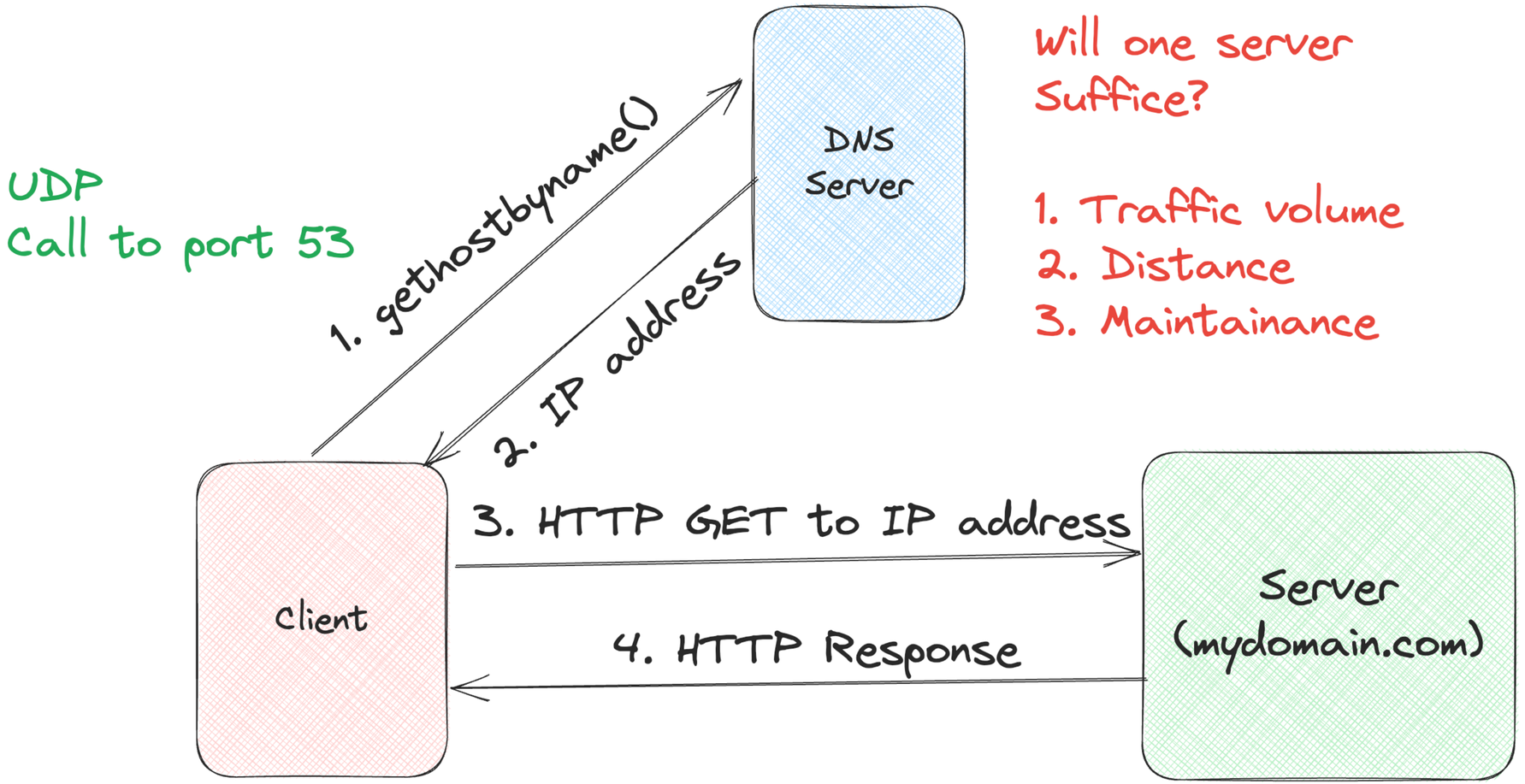
- **Session Layer (L5)**
  - Manages connection between different devices
  - Establishing, maintaining and terminating connections
- **Presentation Layer (L6)**
  - Ensures that data is in format that sender and receiver can understand
  - Manages data encryption, compression
- **Application Layer  (L7)**
  - Provides network services to the application processes
  - Eg: web browser, email clients, other softwares/apps



# Application Layer Protocol
## What does it mean?

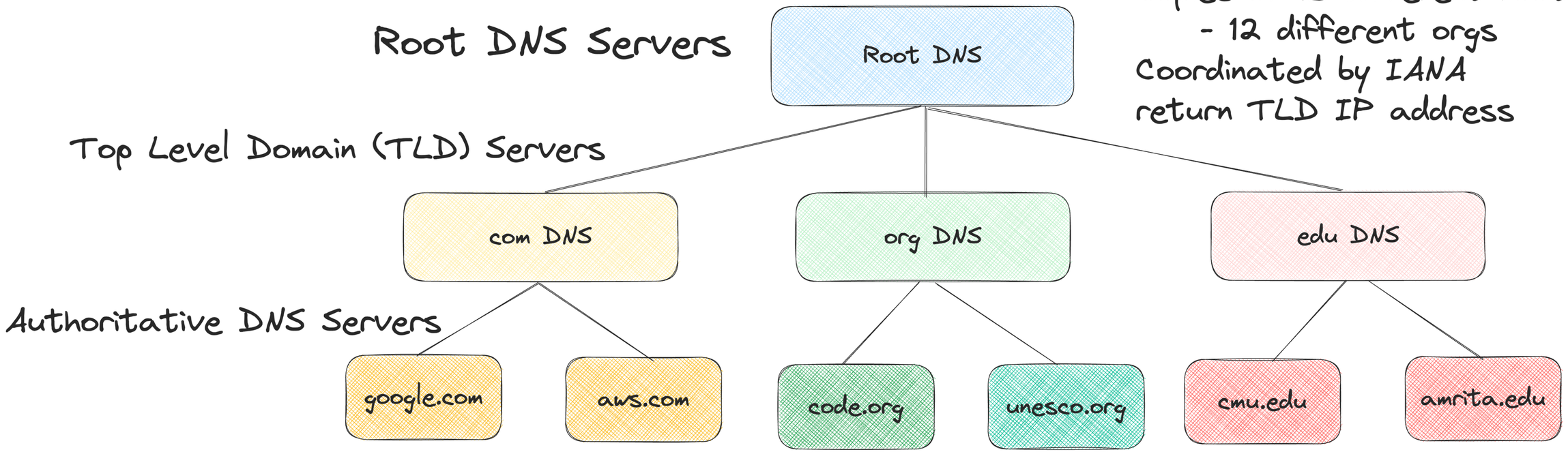
- Application layer protocol defines the following:
  - Types of message exchanges (request/response)
  - Syntax of various message types
  - Semantics of the fields
  - When and how the process sends and responds to messages
- Some protocols: HTTP, SMTP, DNS, etc.

# Mapping domain name to IP: DNS



# DNS: Distributed Hierarchical Database

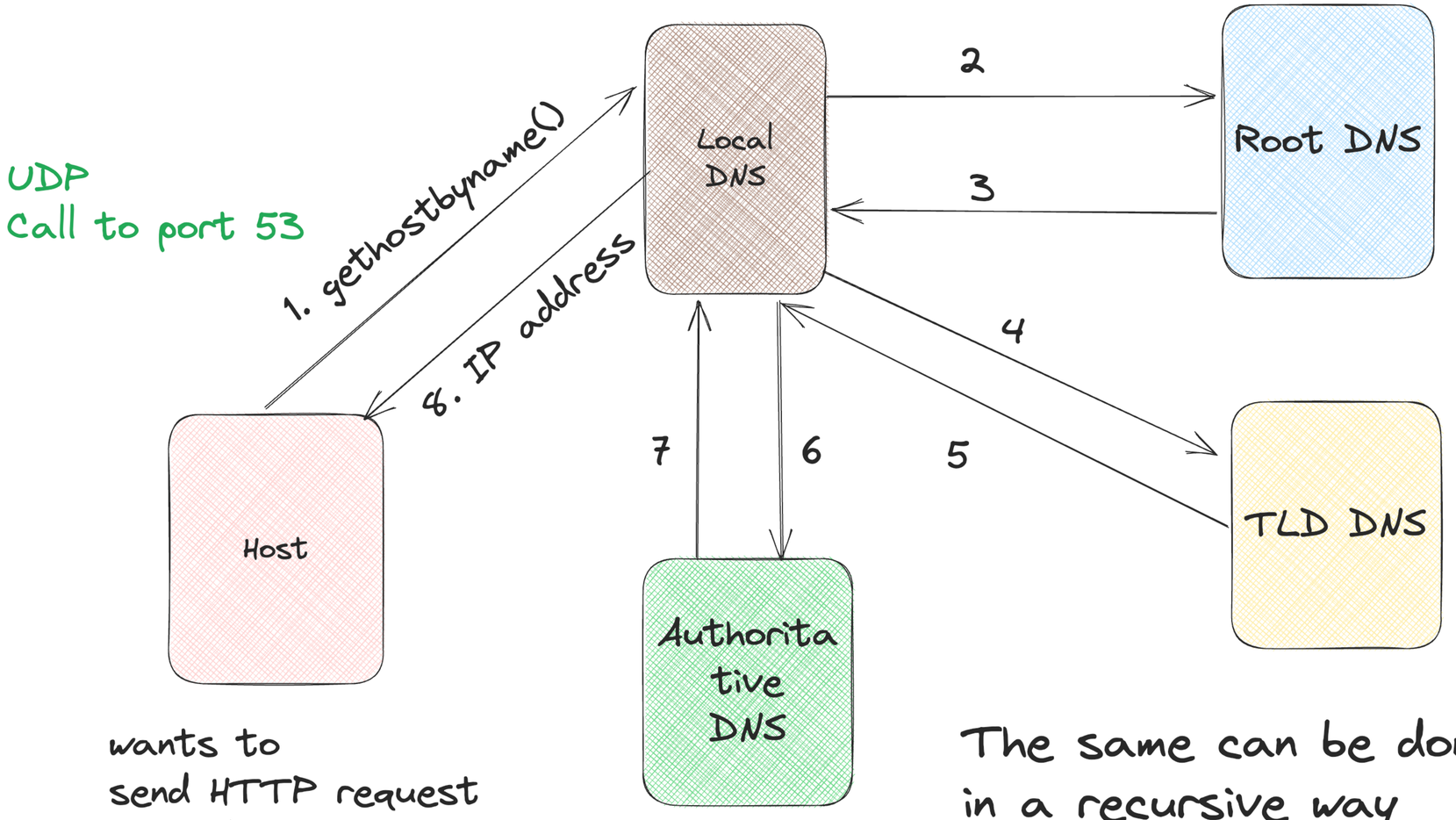
More than 1000 root servers
Copies of 13 different root servers
- 12 different orgs
Coordinated by IANA
return TLD IP address




TLD Servers - can be maintained by orgs, provide IP of authoritative DNS Servers
Authoritative DNS servers - Orgs can choose to implement their own or go for third party
All DNS records have to be made public - that maps hosts to IP address

# Local DNS

## Each ISP can have DNS and clients can connect to that

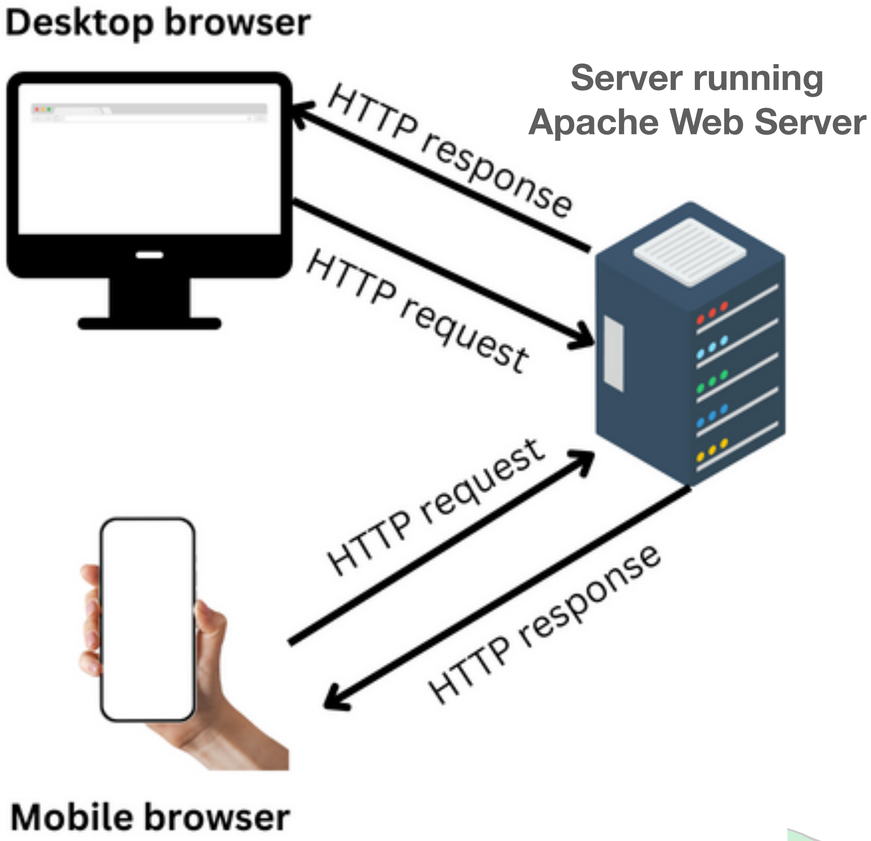



Caching can be done at the ISP Local DNS

The same can be done in a recursive way putting overhead on contacted server

# HTTP: Hyper Text Transfer Protocol

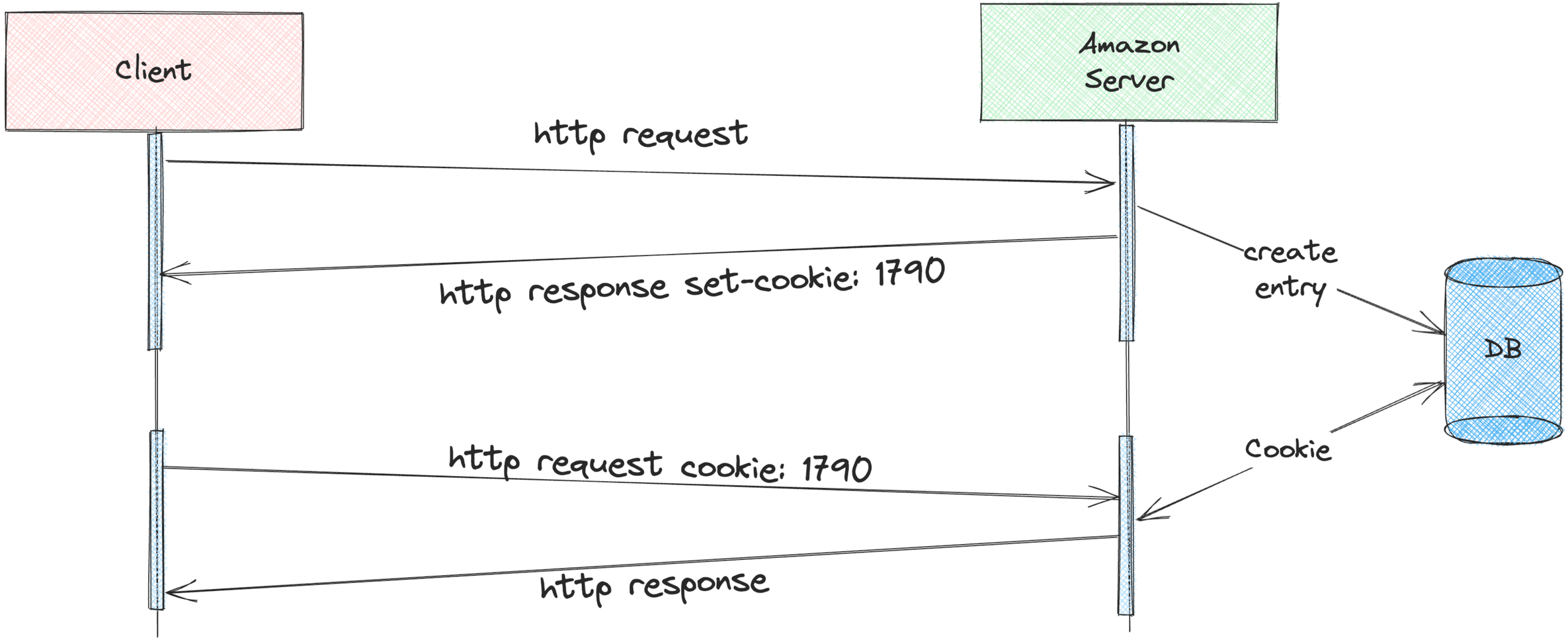
- Application layer protocol of the web
- Implemented in two programs: Client and Server
- HTTP protocol defines structure of messages
- **Client:** browser that sends requests, receives and displays web objects (using HTTP protocol)
- **Server:** Web server that sends objects in response to requests (using HTTP protocol)
- Uses **TCP** and it is stateless




**https://iiit.ac.in/samplePage.html**
**Host name: iiii.ac.in**
**Object: samplePage.html**

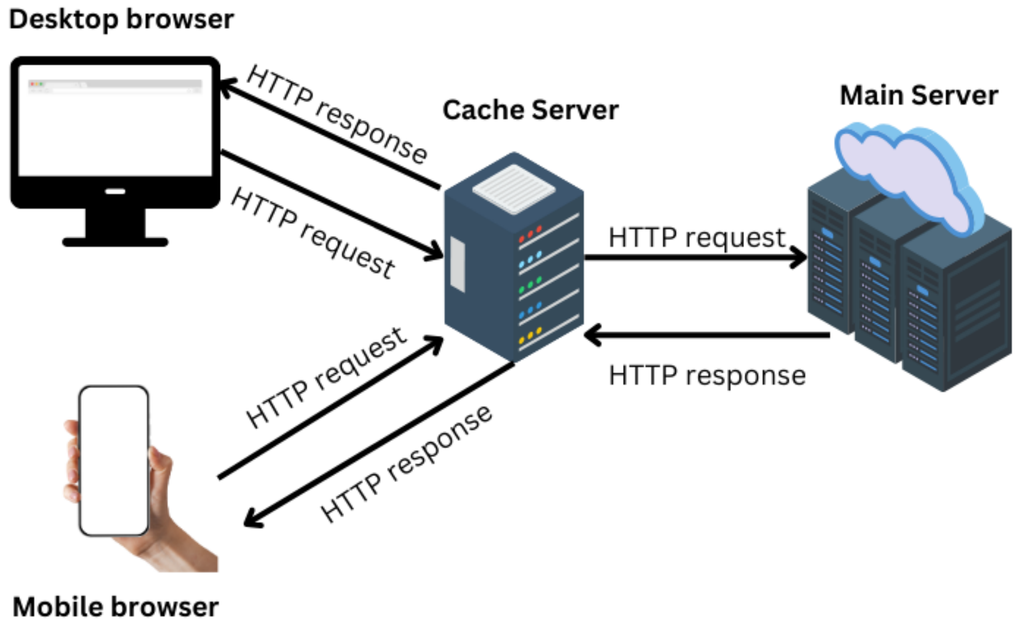
# How to store information in stateless Protocol?

## Using Cookies



# Web Caches

- Not every time we need to access the main (original) web server
- We can have proxy server that satisfies request on behalf of main server
- Browser can be controlled to point towards a cache (mentioned in response header)
  - If cache hit: return object from cache
  - Else cache request object from main server and returns it
- Conditional GET is used to update Cache (“if-modified-since”)

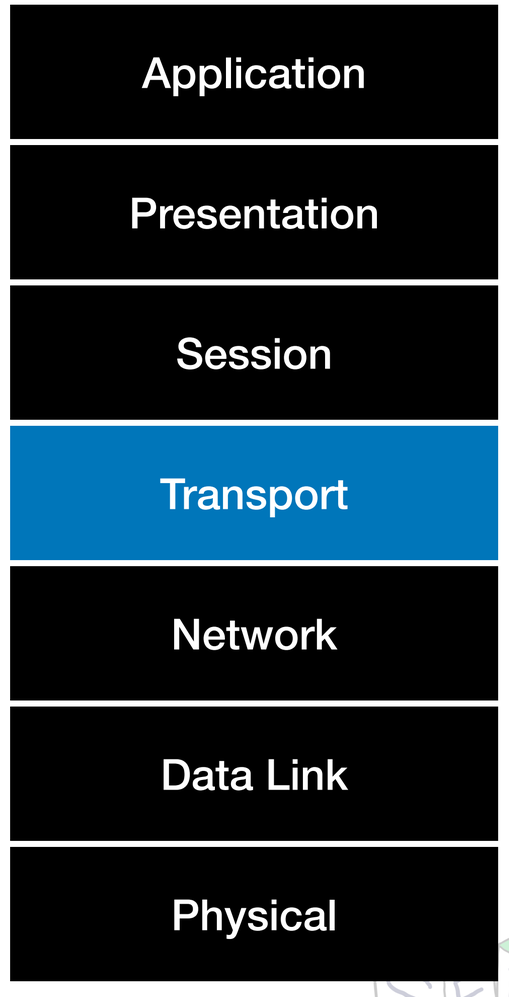



```
Cache-Control: max-age=<seconds>
```

```
Cache-Control: no-cache
```

# On to Transport Layer (L4)

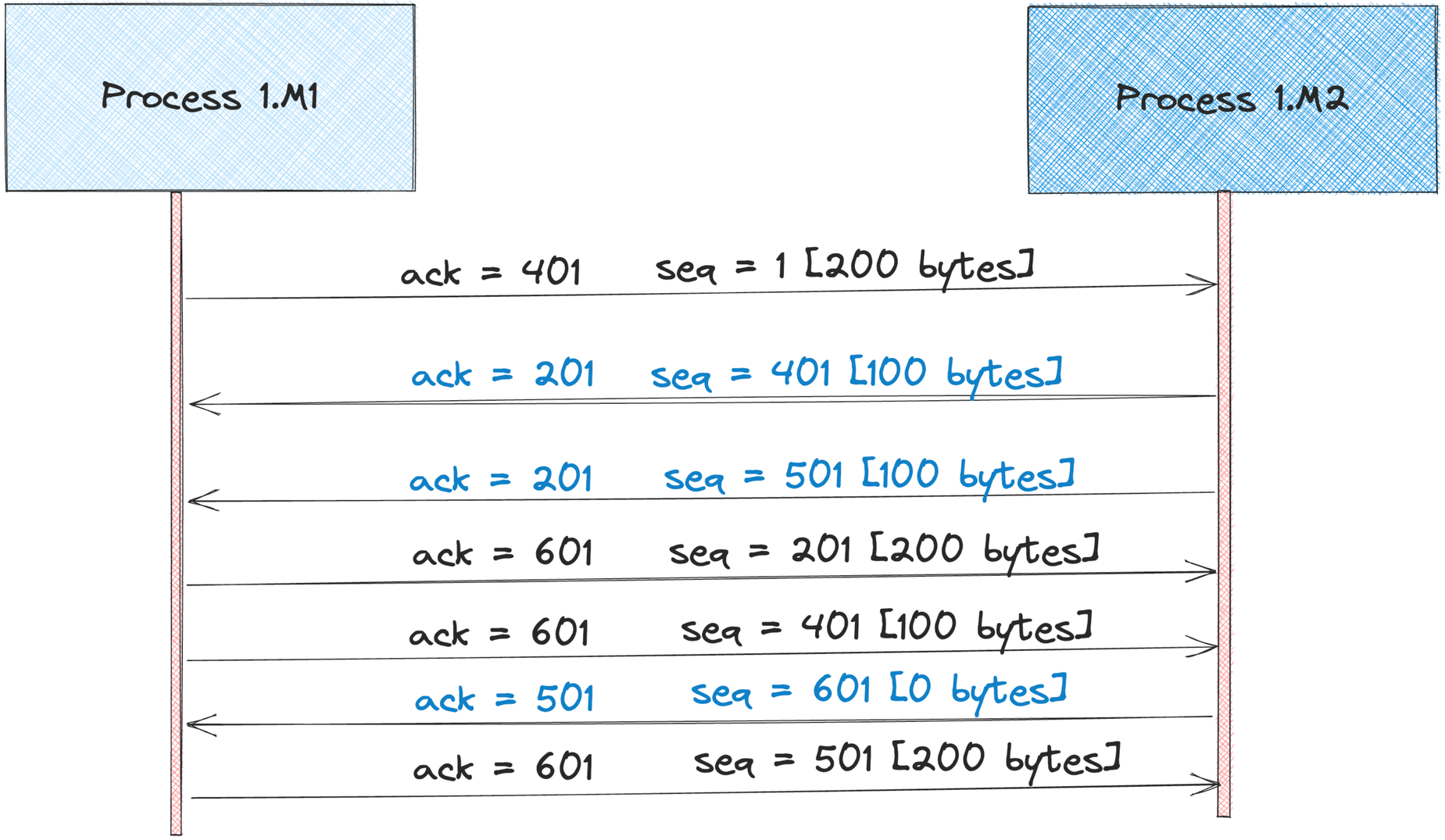
- Process gets the data delivered through support of transport layer
- Addressing scheme: Ports
- Layer 4 has an addressing scheme to guarantee message delivery
  - Ports! (0 - 65535), Privileged: 0-1023, Registered: 1024 - 49151
- Two strategies/protocols that allows this
  - **Transmission Control Protocol (TCP)** - favours reliability
  - **User Datagram Protocol (UDP)** - favours efficiency



# TCP vs UDP

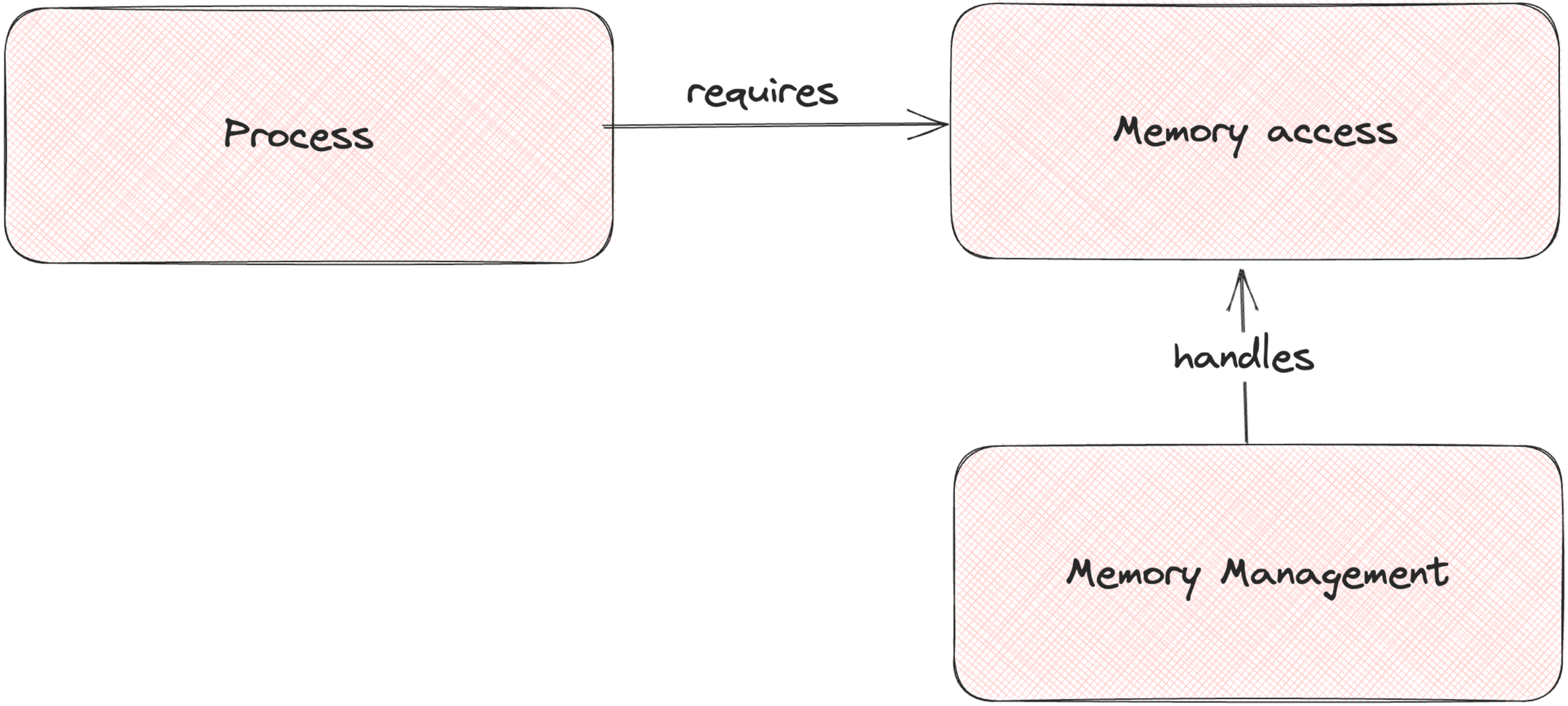
| TCP | UDP |
|---|---|
| Connection Oriented | Not Connection Oriented |
| Reliability (order is maintained and retransmission) | Unreliable (At L4) |
| Higher overhead - reliability, error checking, etc | Low overhead |
| Flow control (based on network) | No implicit flow control |
| Error detection - retransmit erroneous packets | Has some error checking - Erroneous packets are discarded without notification |
| Congestion Control | No Congestion Control |
| Use cases: HTTP/HTTPS, File transfer, Mail | Use cases: Streaming data, VoIP, DNS queries, .. |

# TCP and how it works!

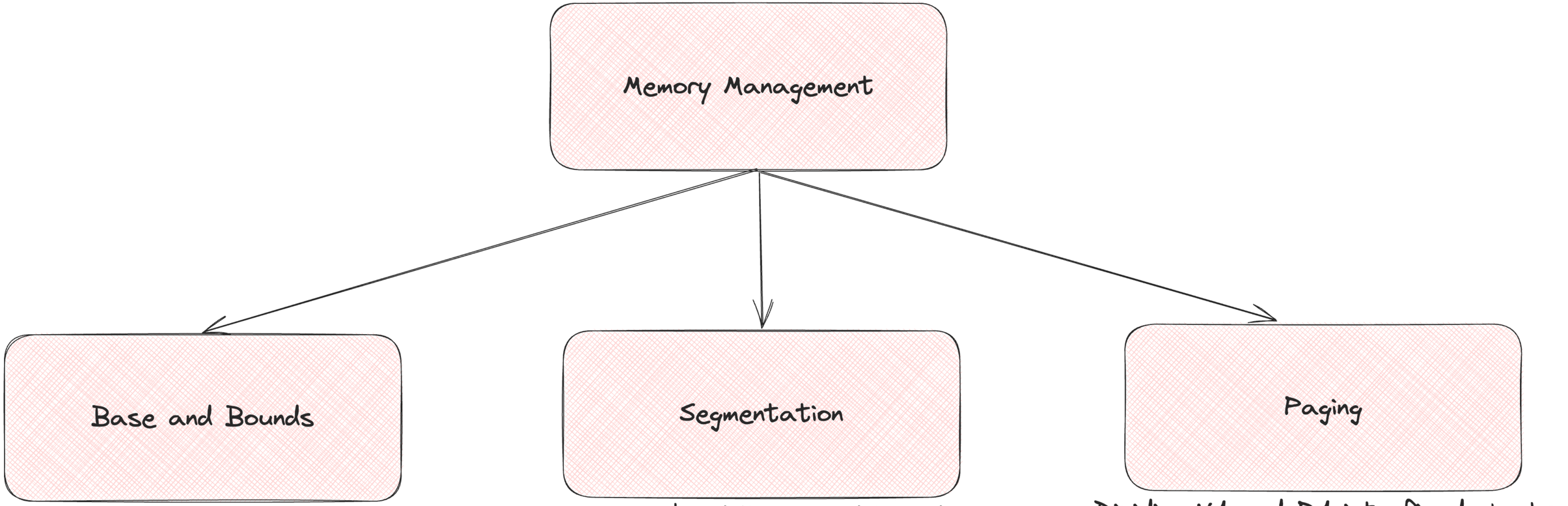





# Process requires Memory: Memory Virtualization



# Memory Management




Base register
and bounds register

(just add and check
if less than bounds for
translation)

Generalized Base and bounds

(for each segment there is a
base and bounds that needs to
be assigned)

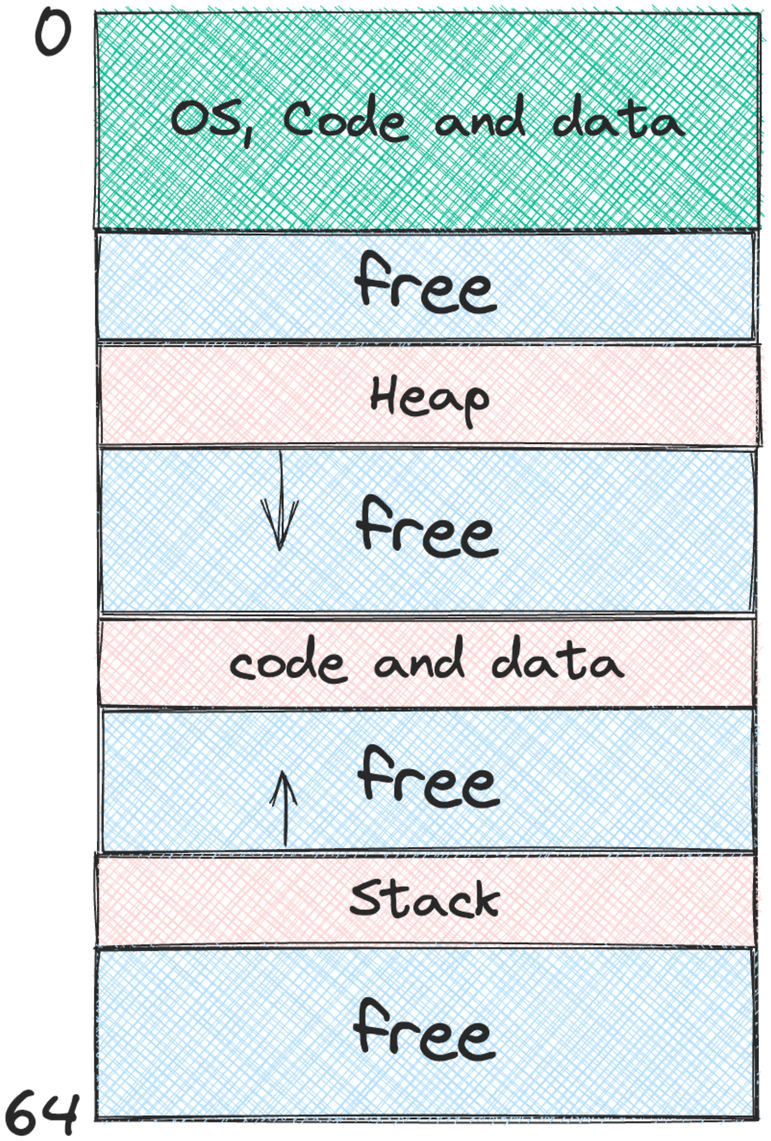
Dividing VA and PA into fixed chunks

(Translation with the help of a page
table)

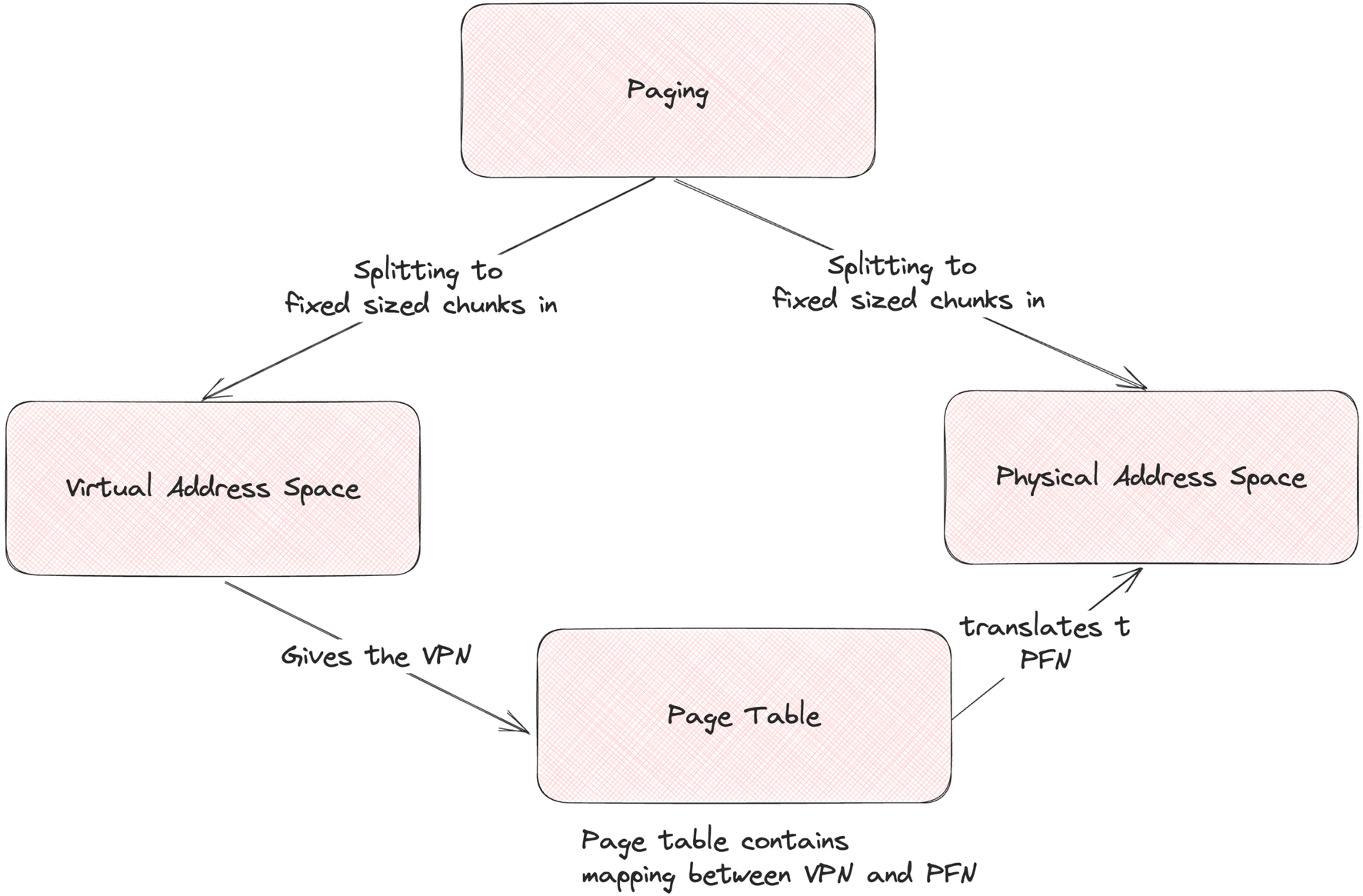
# Segmentation

## Generalized Base and Bounds

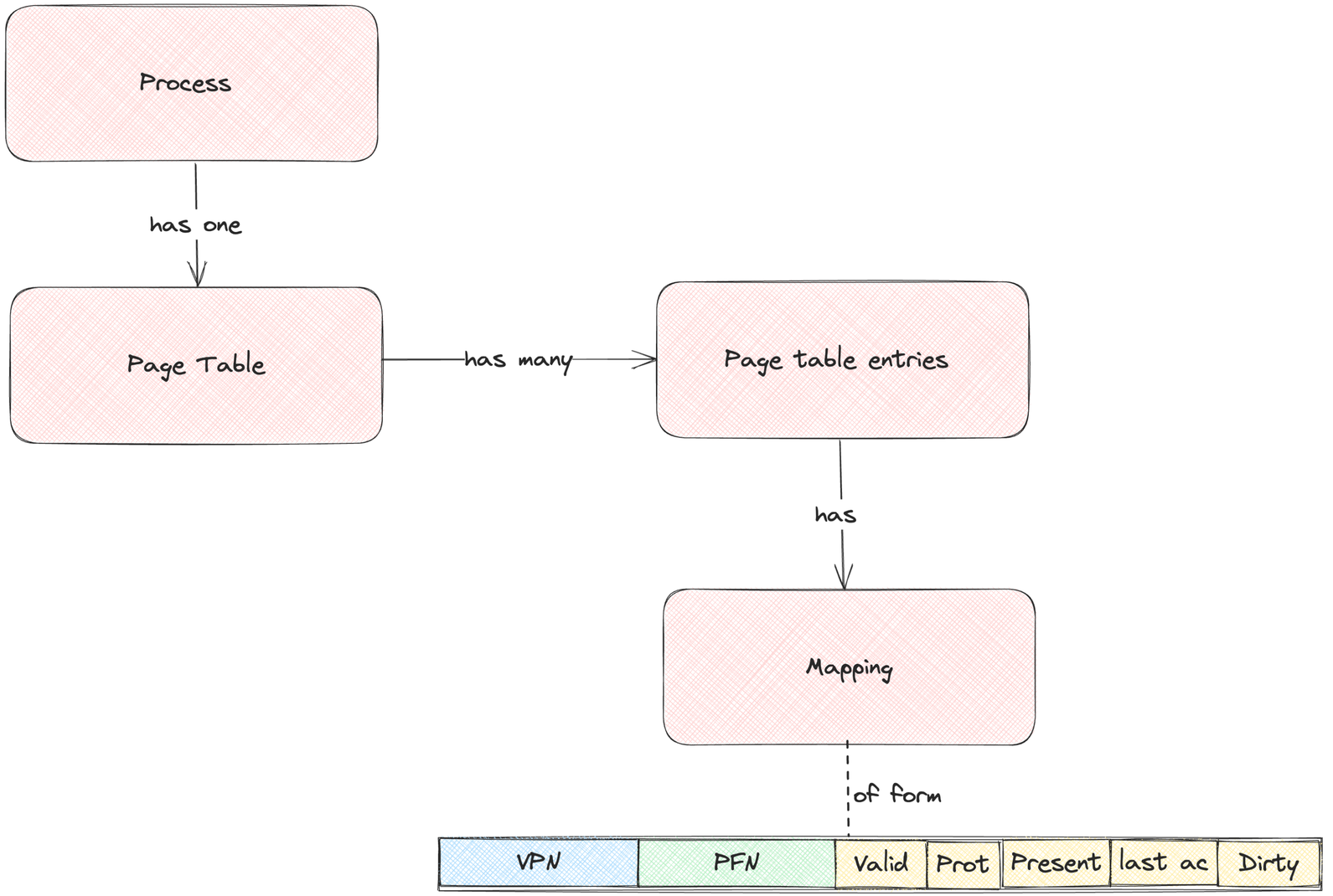
- Only used memory is allocated in physical memory
  - Allows allocating large address space
  - Sparse address space
- Different segments per process - code, stack, heap
- **For translation:** use first bits to identify segments and perform translation
- Results in **External fragmentation**



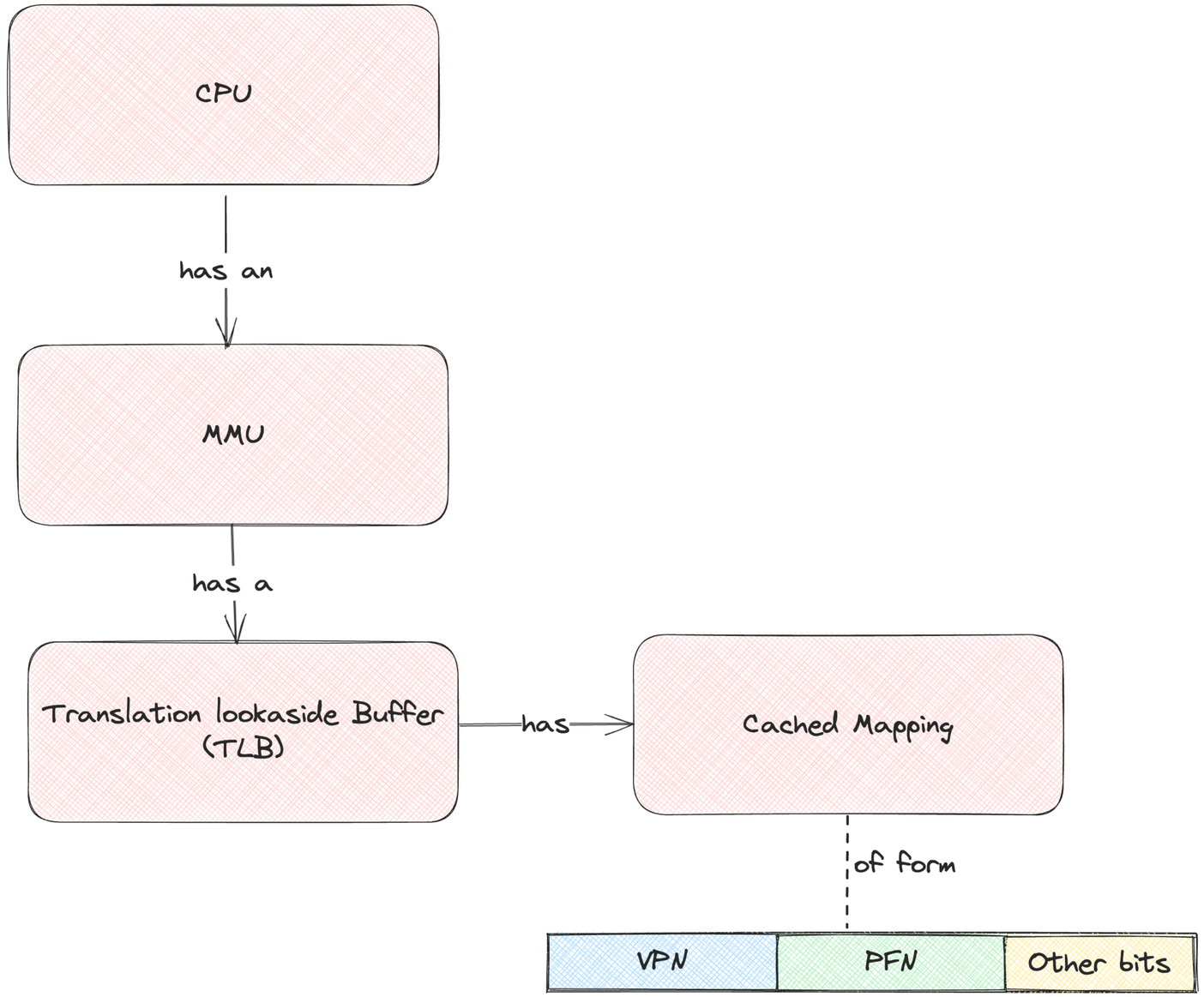
# Paging

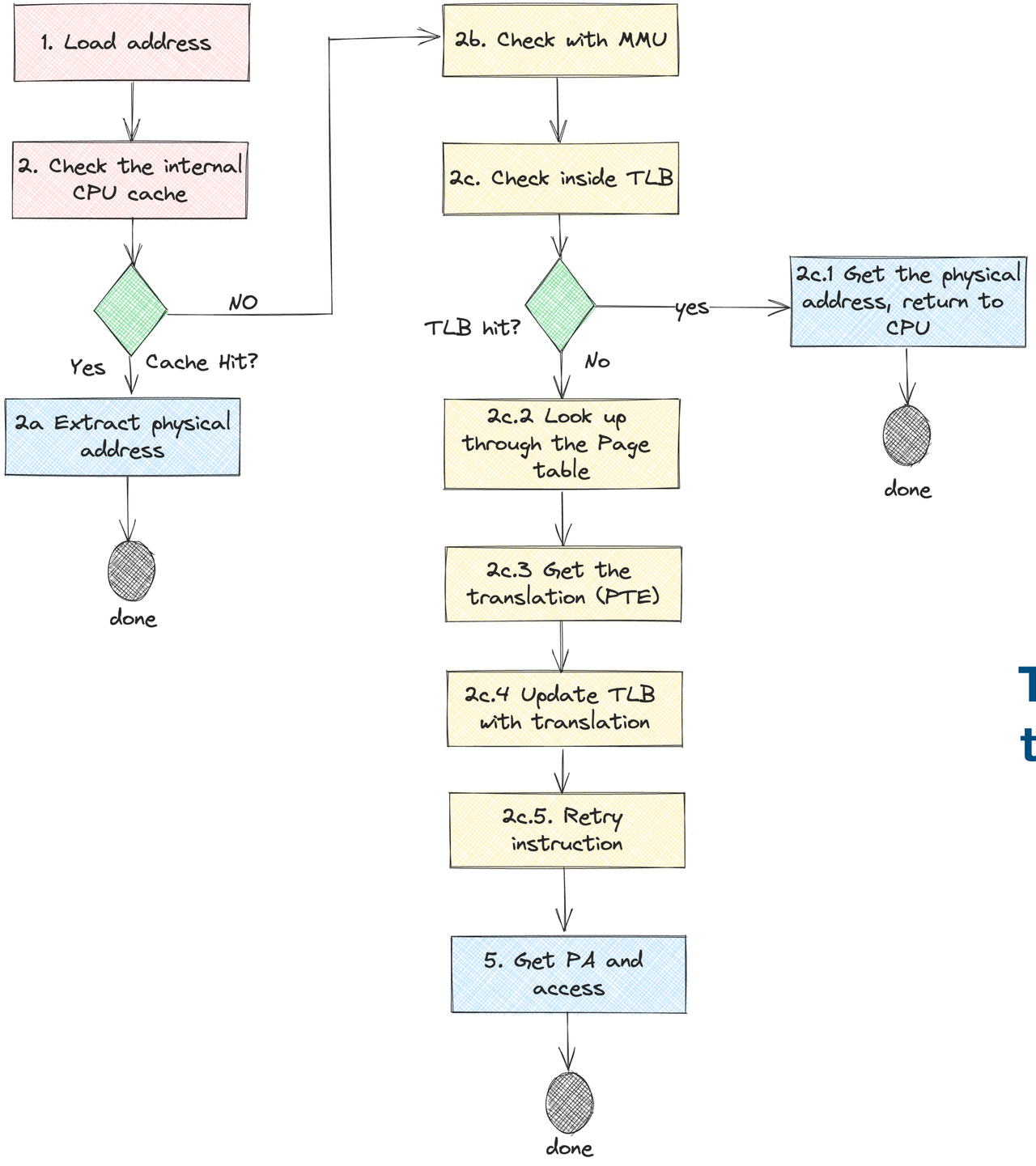


# Page Tables


Process
has one
Page Table
has many
Page table entries
has
Mapping
of form
VPN
PFN
Valid
Prot
Present
last ac
Dirty

# Paging - TLB



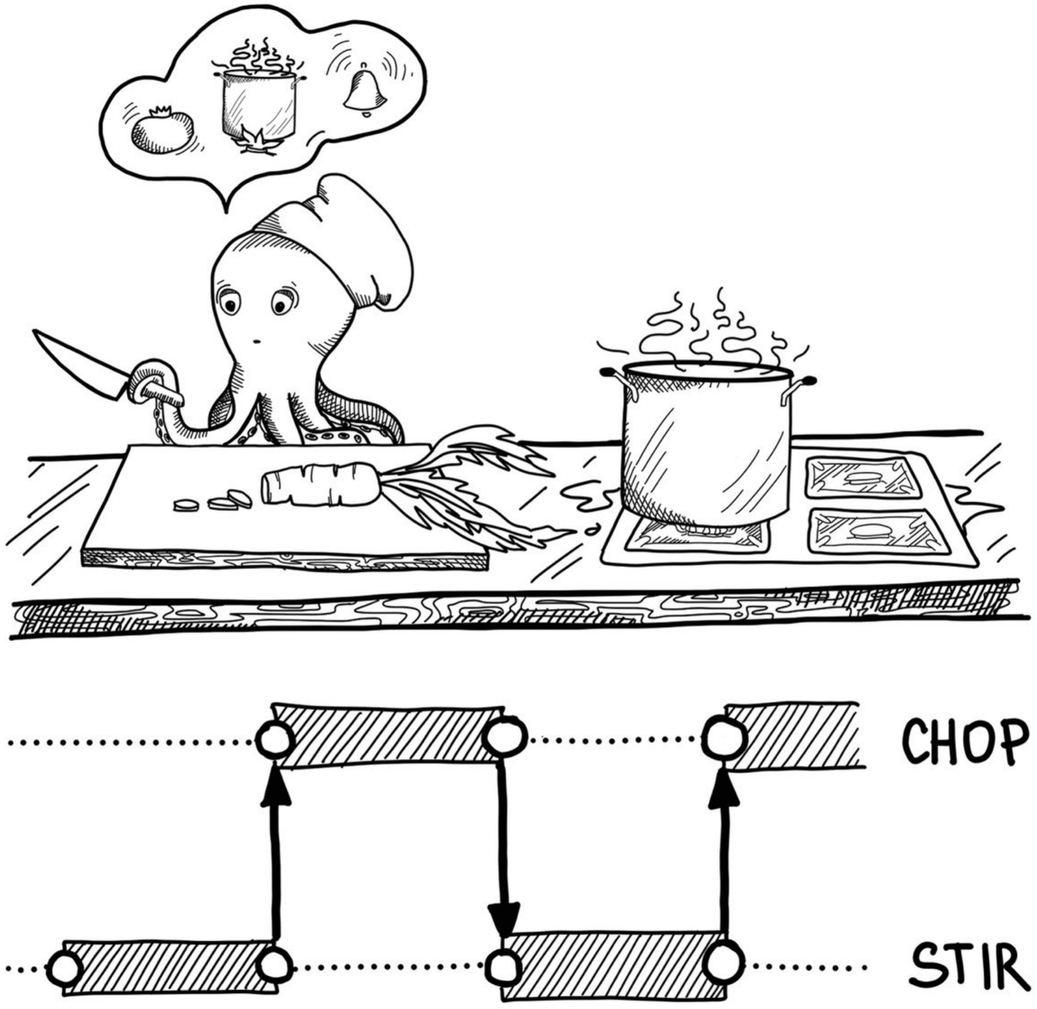


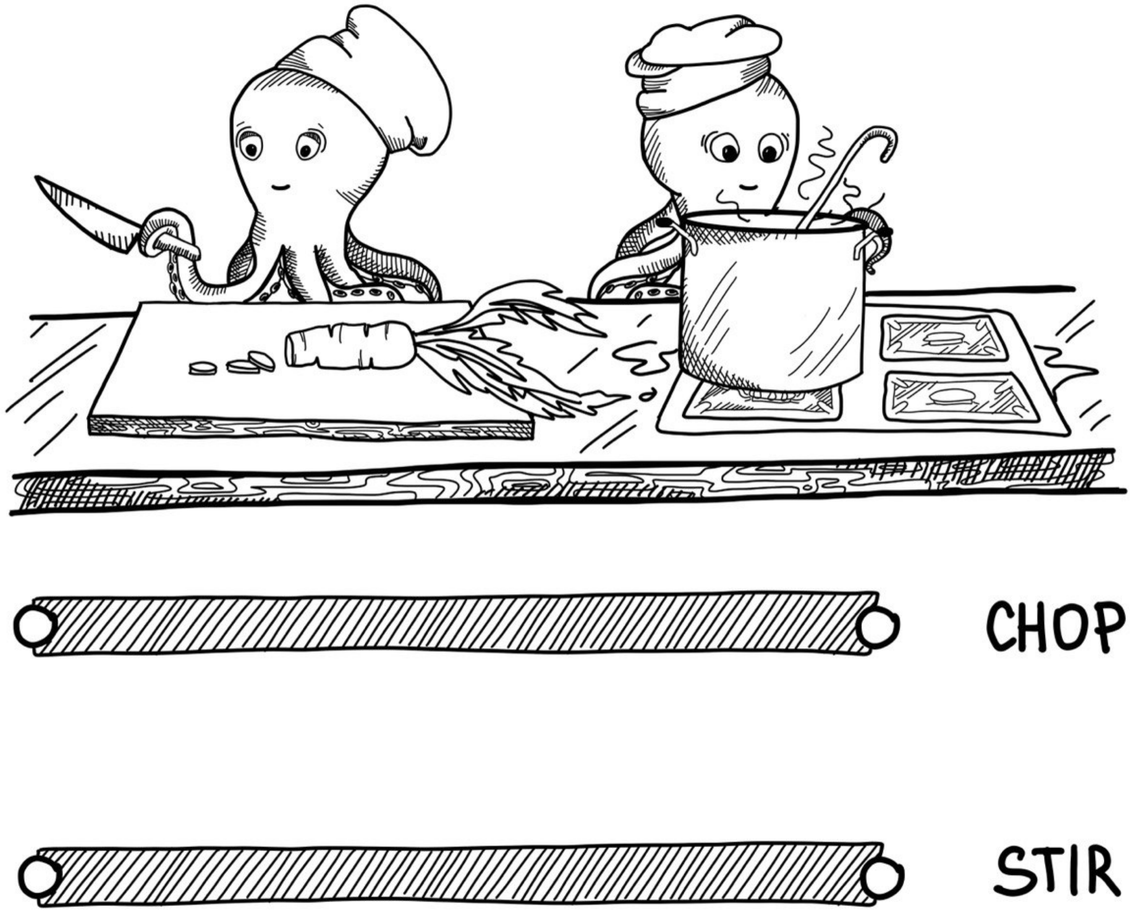

# The overall address translation Process



# Concurrency and Parallelism

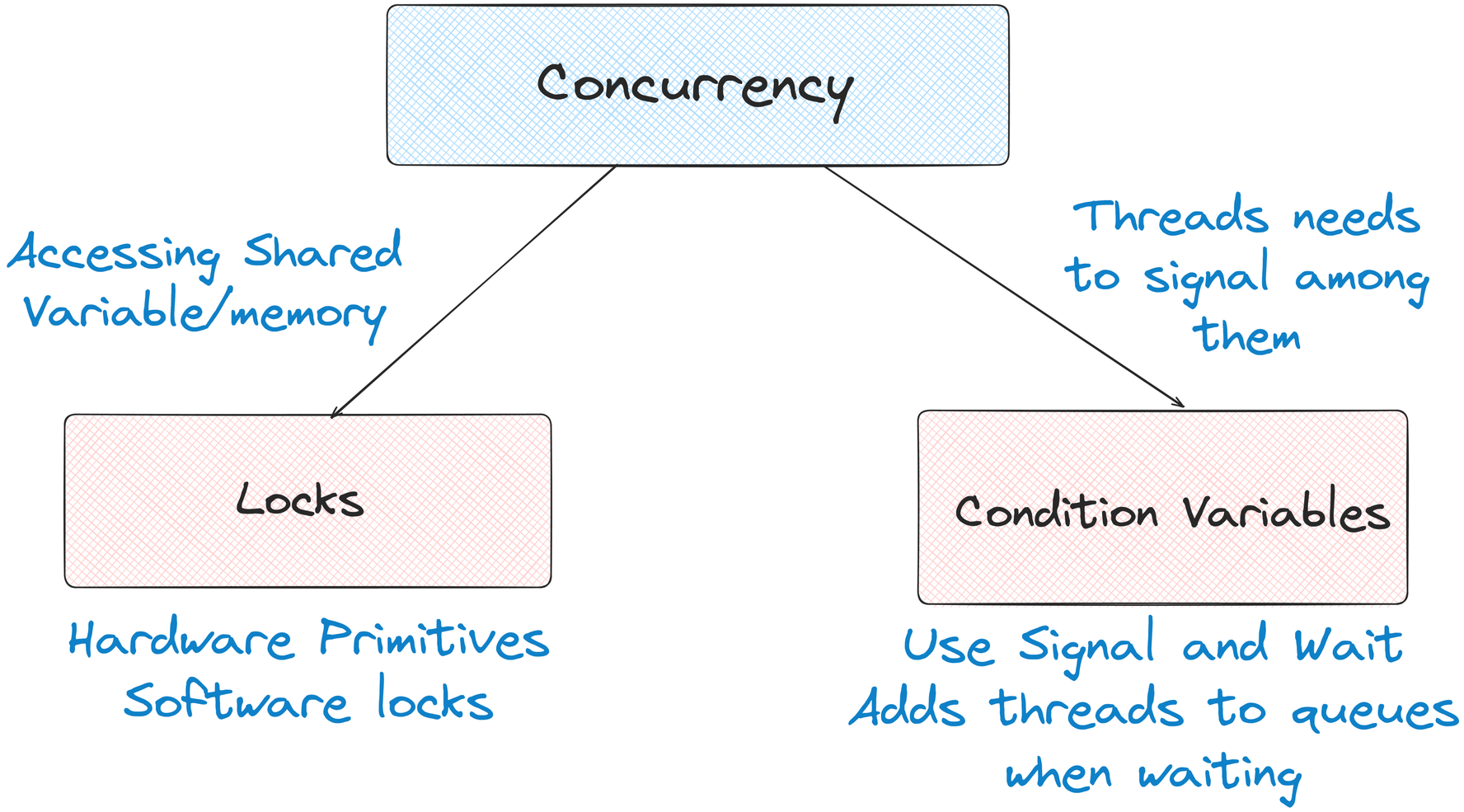
## What is what?




Source: https://freecontent.manning.com/concurrency-vs-parallelism/

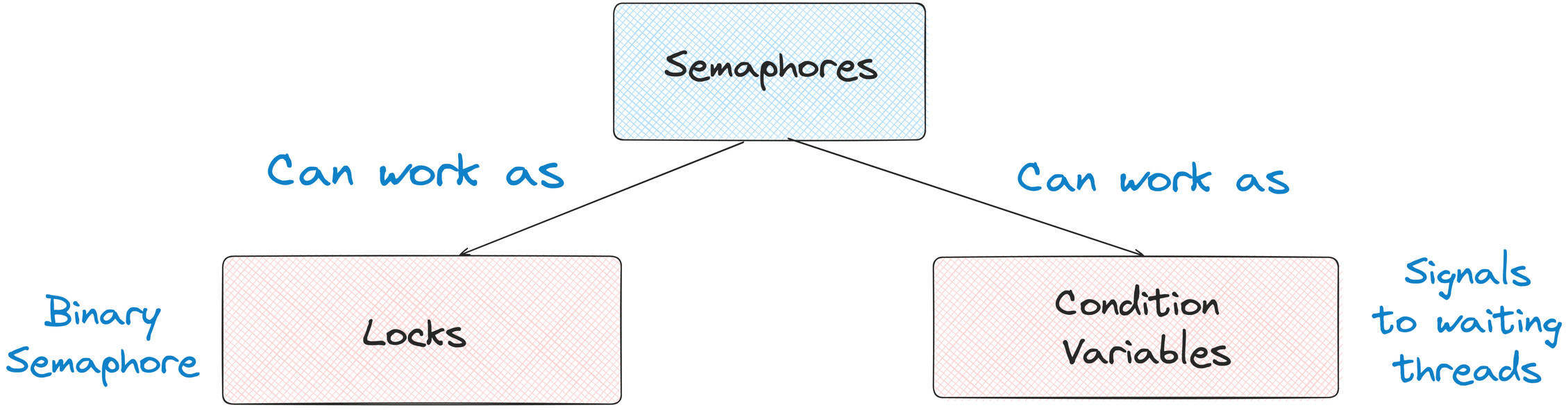
# Locks and Condition Variables




Combination of Locks and Condition variables may be required to accomplish different tasks



# Semaphores




Two key operations: wait () and post()
Initialization of semaphore holds the key

# Producer Consumer Problem Using Semaphores

- Let us start with 2 semaphores: empty and wait, Buffer with MAX = 1

Get and Put for large sized buffer

```
int buffer[MAX];
int fill = 0;
int use = 0;
int count = 0;

void put (int value)
{
  buffer[fill] = value;
  fill = (fill + 1)%MAX;
  count ++;
}

int get()
{
  int tmp = buffer[use];
  use = (use + 1)%MAX;
  count --;
  return tmp;
}
```

Producer-Consumer with buffer

```
sem_t empty;
sem_t full;

void *producer(void *arg)
{
  int i;
  int maxLoops = (int)arg;
  for (i=0;i<maxLoops;i++)
  {
    sem_wait(&empty);
    put (i);
    sem_post(&full);
  }
}

void *consumer(void *arg)
{
  int i;
  int maxLoops = (int)arg;
  for (i=0;i<maxLoops;i++)
  {
    sem_wait(&full);
    int tmp = get();
    sem_post(&empty);
    printf("%d\n", tmp);
  }
}
```

# Producer Consumer Problem Using Semaphores

## The Solution

Producer

```
sem_wait (&empty);
sem_wait (&mutex);
put(i);
sem_post (&mutex);
sem_post (&full);
```

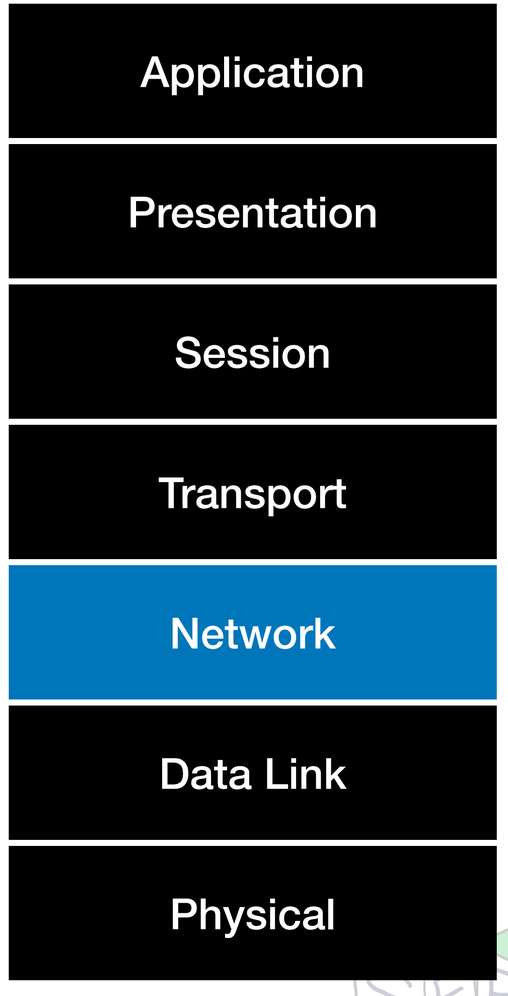
Consumer

```
sem_wait (&full);
sem_wait (&mutex);
get();
sem_post (&mutex);
sem_post (&empty);
```

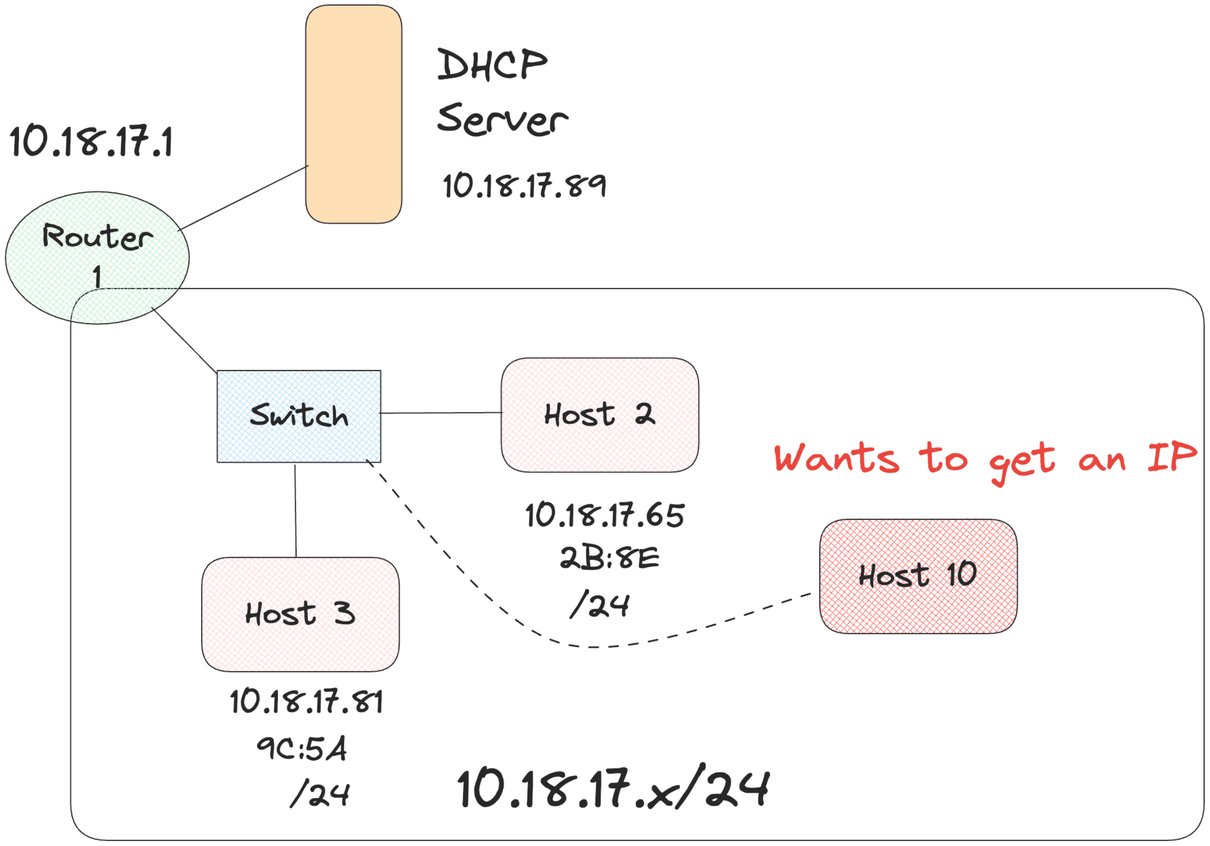
- Add mutex lock around put and get - Avoid deadlocks!
- Let producer and consumer get the signal and then lock when entering CS

# Network Layer (L3)
## End-to-end Communication

- Manages routing through different routes in a large network
- Uses an addressing scheme - IP addressing
  - 32 bits represented as 4 octets (IPv4)
- Performs functionalities such as Logical addressing (IP), Path selection and packet forwarding
- L3 technologies: routers, even hosts are L3, L3 switches



# Getting IP Address - DHCP

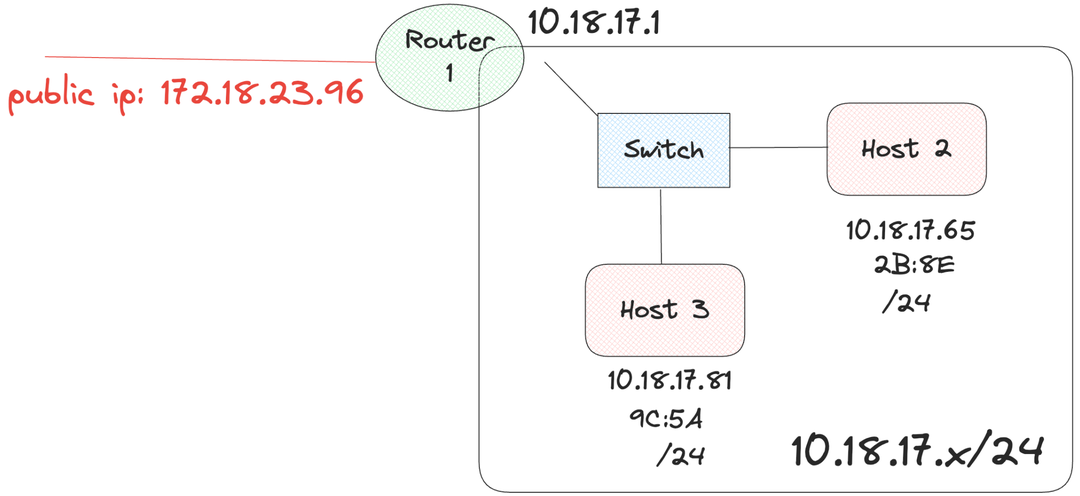



- Host 10 is the client here
- It sends out a broadcast DHCP request to every node in the network to get DHCP server
- Every device in the network will get the request
- DHCP runs over UDP
- Client uses port 68 and server port (listens on port 67)

# Network Address Translation (NAT)

**NAT Translation Table**

| WAN side address | LAN side address |
|---|---|
| **172.18.23.96 5501** | 10.18.17.81 3801 |
| **....** | ..... |




- All devices in the network share just one IPV4 address as far as the outside world is concerned
- NAT allows a router (similar device) to translate private IP addresses to its own public IP address
- When devices from network wants to communicate with outside network:
  - NAT modifies the source IP to make it appear that communication is from the larger public IP
  - A translation table is used for managing the translations
- **Multiple types:** Static NAT, Dynamic NAT, Port Address Translation or NAT Overload

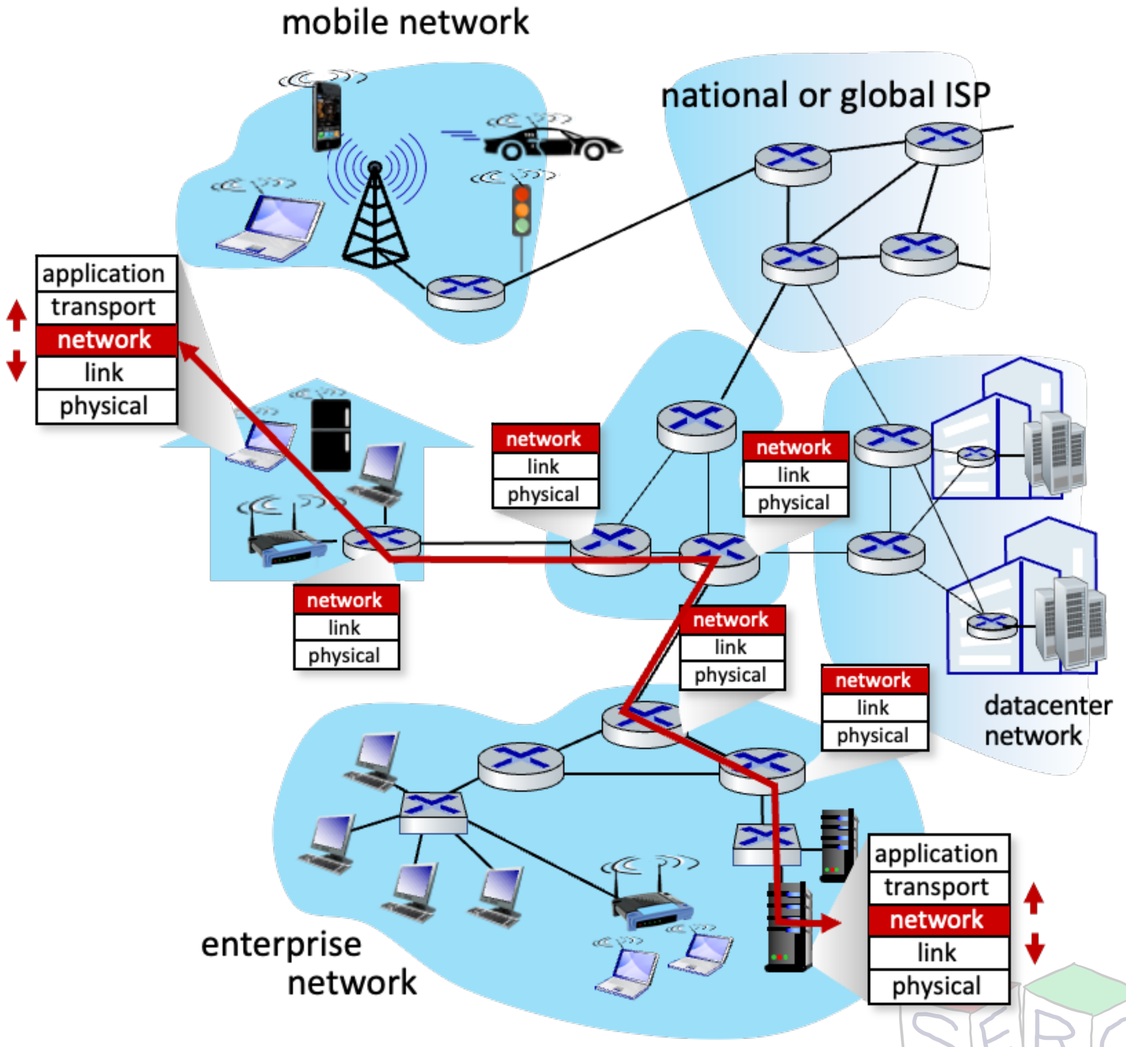
# Network Layer - Functionalities

## Addressing

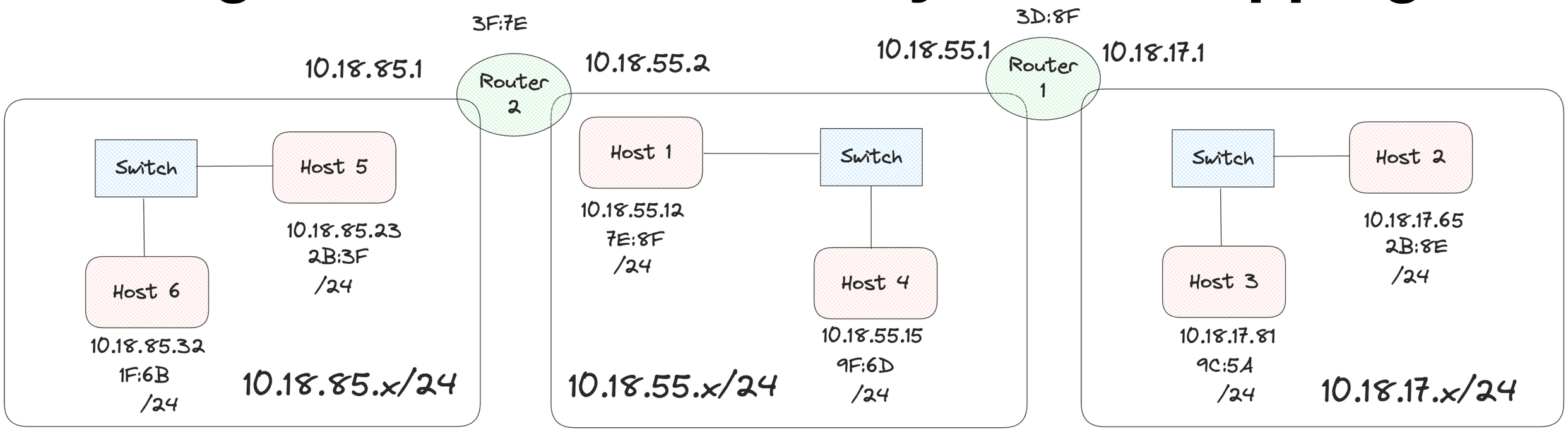
- Devices in network are assigned logical address for unique identification - IP
- Network layer uses IP to forward packets to the intended destinations

## Route Determination

- Identifies best path for packets to reach to destination
- This process is dynamic and changes based on network conditions



# Routing Tables - Static and Dynamic Mappings




| Type | Destination | Interface |
|---|---|---|
| **DC** | **10.18.85.x/24** | Left |
| **DC** | **10.18.55.x/24** | Right |
| **Static** | **10.18.17.x/24** | 10.18.55.1 |

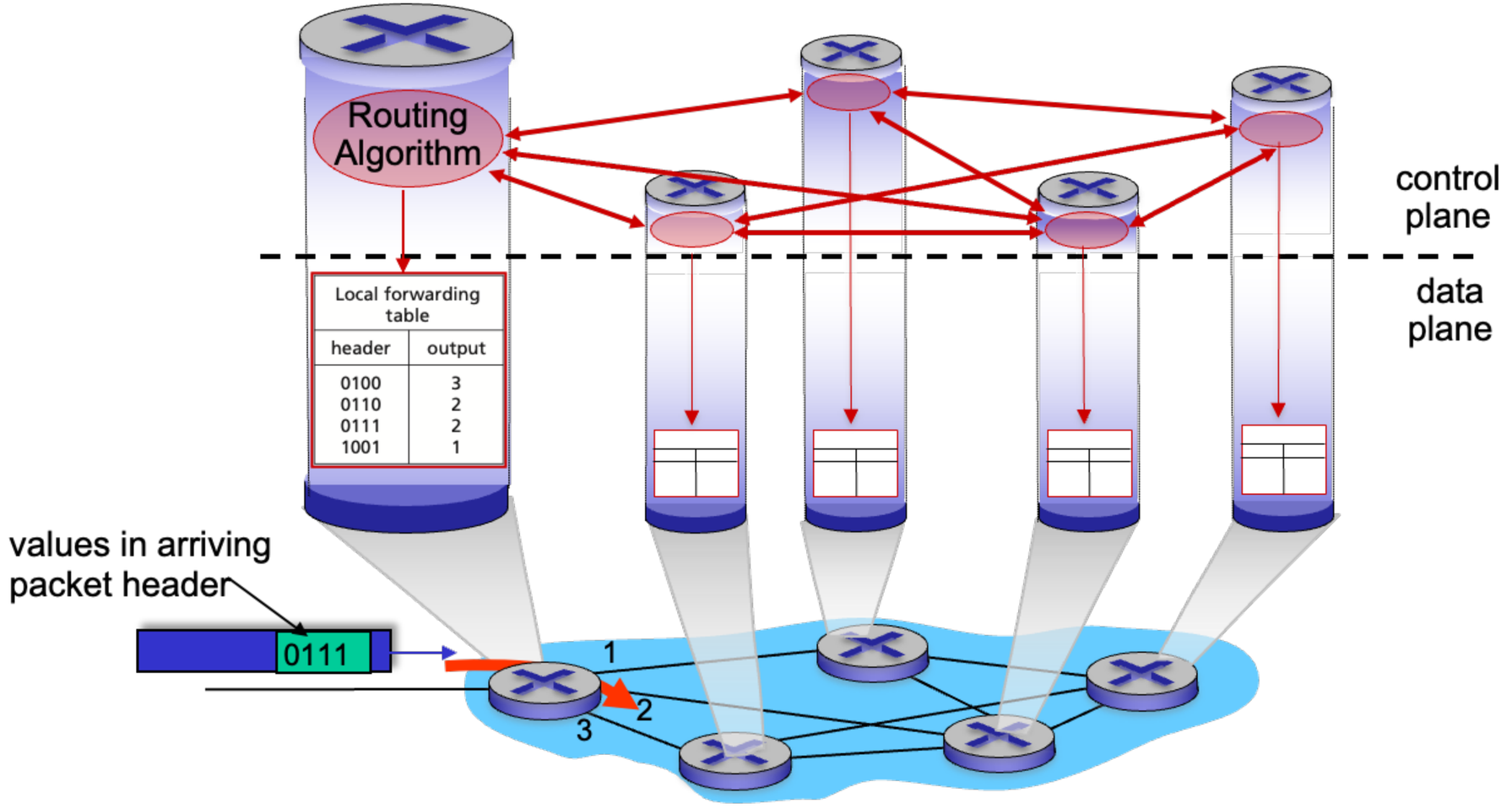
**Router 2 routing table**

| Type | Destination | Interface |
|---|---|---|
| **DC** | **10.18.55.x/24** | Left |
| **DC** | **10.18.17.x/24** | Right |
| **Static** | **10.18.85.x/24** | 10.18.55.2 |

**Router 1 routing table**

**Routers can also Learn about the address**

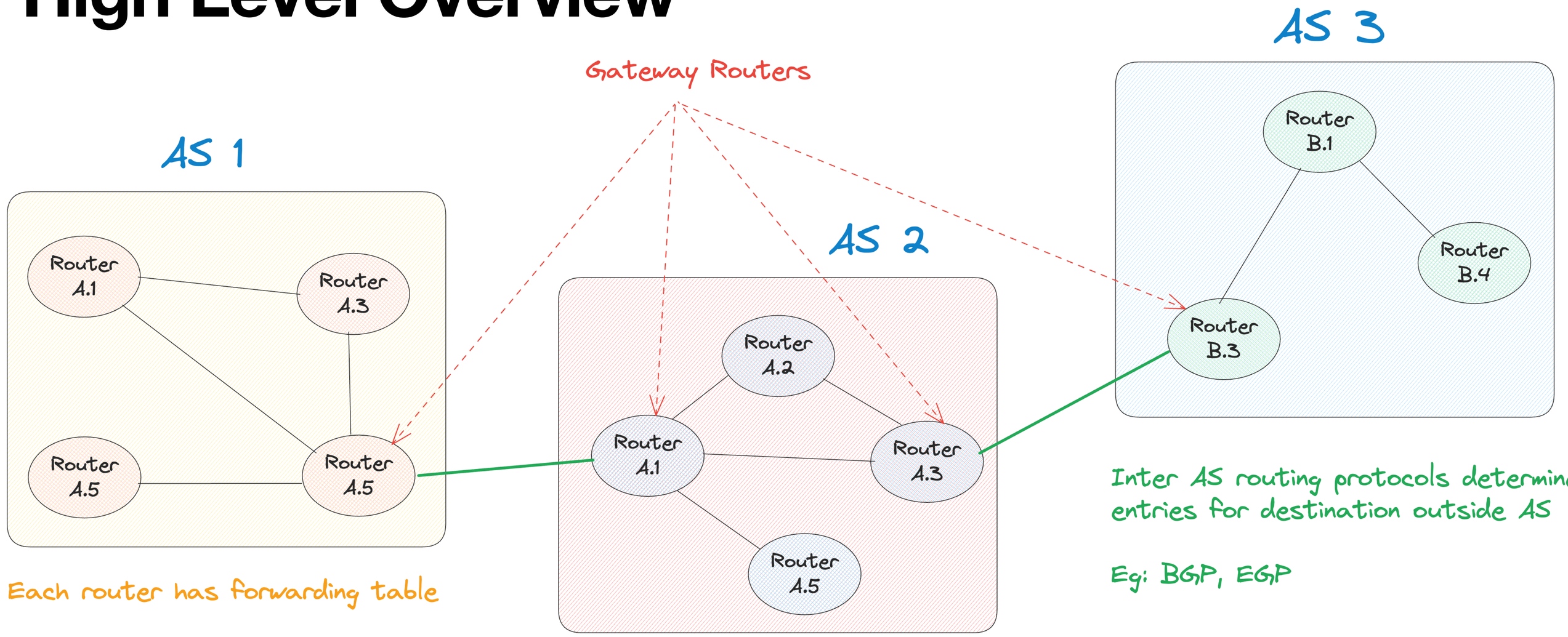
# Traditional Control Plane Approach



# Internet approach to scalable routing

- Aggregate routers into regions known as **“Autonomous Systems” (AS)** a.k.a “domains”
  - Total of around 70,000 AS’s have been assigned not all are active
- There are mechanisms for handling routing within the domain and across AS
- **Intra-AS or Intra-domain**
  - All routers in AS must run the same intra-domain protocol
  - There is a **gateway router** at the edge of each AS which connects with router in another AS
- **Inter-AS or Inter-domain**
  - Routing among AS’s
  - Gateways perform inter-domain as well as intra-domain within their network

# High Level Overview




Each router has forwarding table

Configured by both intra and inter AS routing algorithms

Intra AS routing protocols determine entries for destination within AS

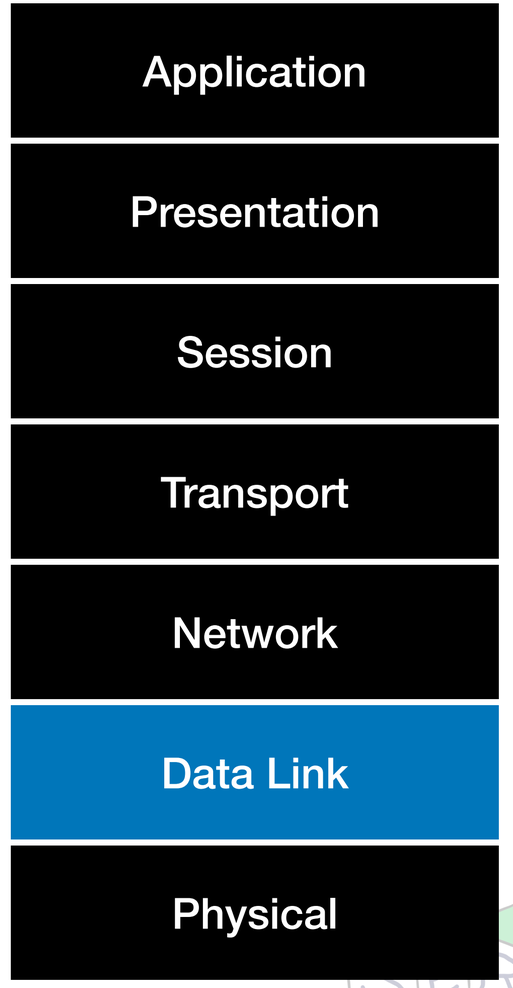
Eg: OSPF, EIGP, RIP

Inter AS routing protocols determine entries for destination outside AS

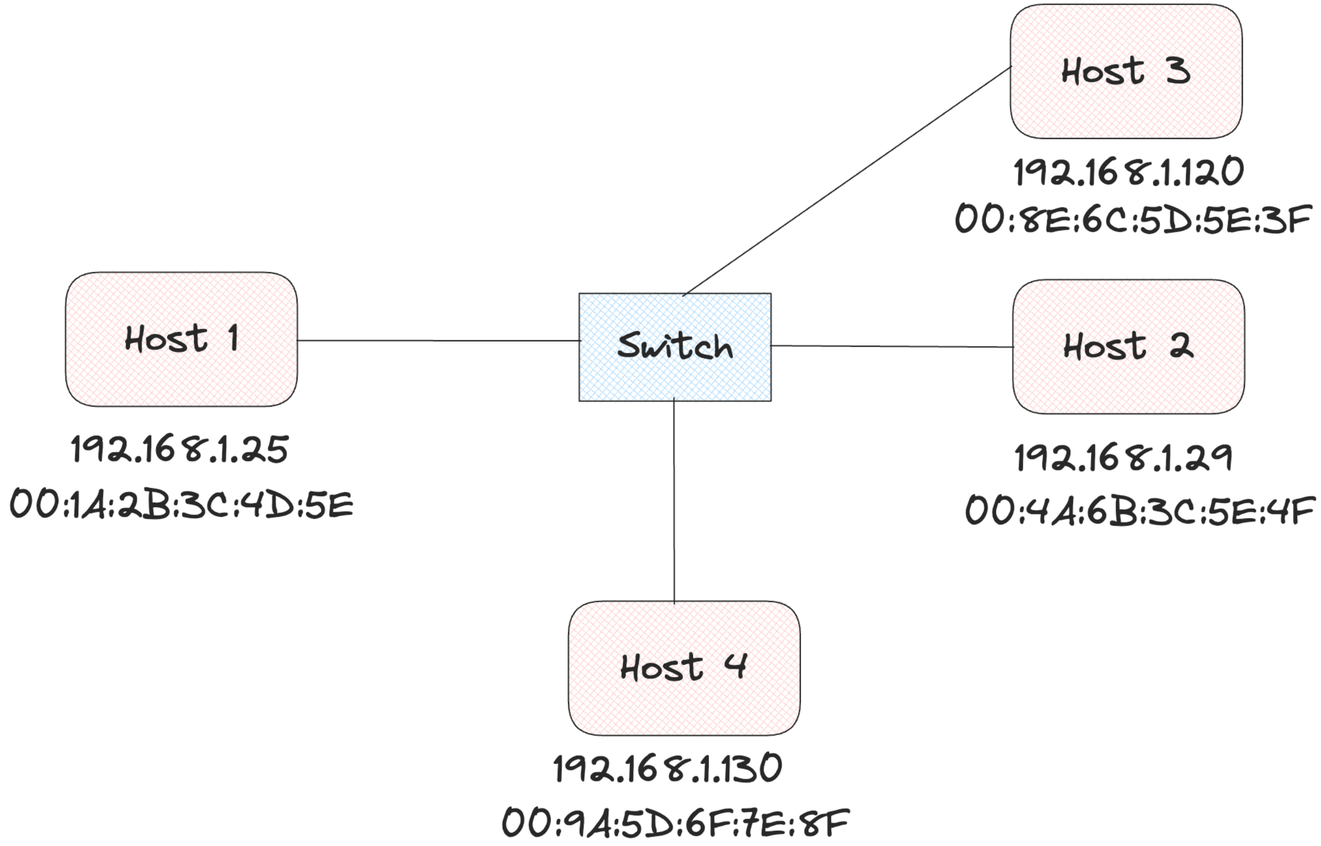
Eg: BGP, EGP

# Data Link Layer (L2)

- Responsibility of transferring datagram from one node to a **physically adjacent node** over a link (no intermediate L3 routers)
- Supports **hop-to-hop** communication
- Ensures reliable connection link between two directly connected nodes (flow control, error correction and detection, etc.)
- Supported by **Media Access Control (MAC)** addressing
- Addressing scheme: MAC addressing (48 bit address, 12 hex digits, 6 bytes)
    - Eg: **00:1A:2B:3C:4D:5E**
    - First three identify manufacturer (IEEE)
    - Next three are assigned by manufacturer and should be unique



# ARP - Address Resolution Protocol

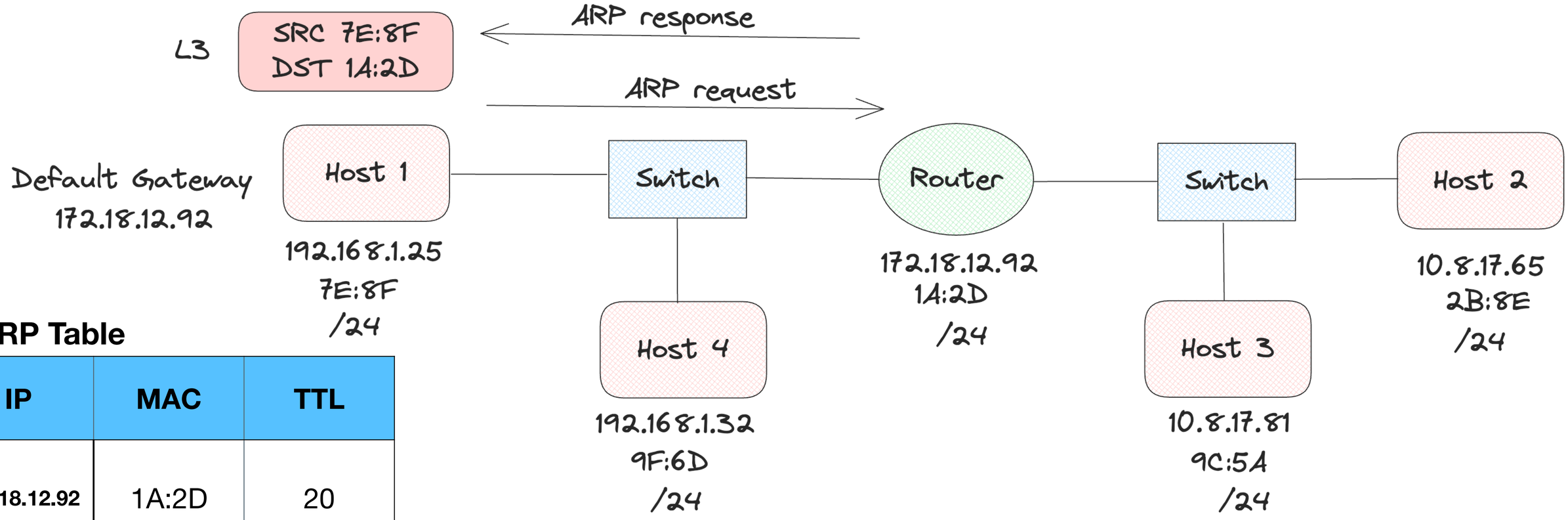



- Each IP node (router, host) on the LAN has a table - **ARP Table**
- IP/MAC address mappings for some LAN nodes
  - **<ip address, MAC address, TTL>**
- **TTL:** Time to live, time after which the mapping will be forgotten (20 mins)

# ARP Query

- When ARP query is sent initially, it is broadcast to all the nodes in the network
- The request includes senders IP address and MAC address
- It also includes the target IP address
  - Destination MAC is set as **FF:FF:FF:FF:FF:FF** (Reserved to send packet to all in the network)
  - If different network then send to the IP address of the gateway router
- All the nodes will have an ARP cache or ARP table
  - It stores the mapping, when the initial request is send from one host, all other hosts stores the incoming mapping as well

# ARP Working




**ARP Table**

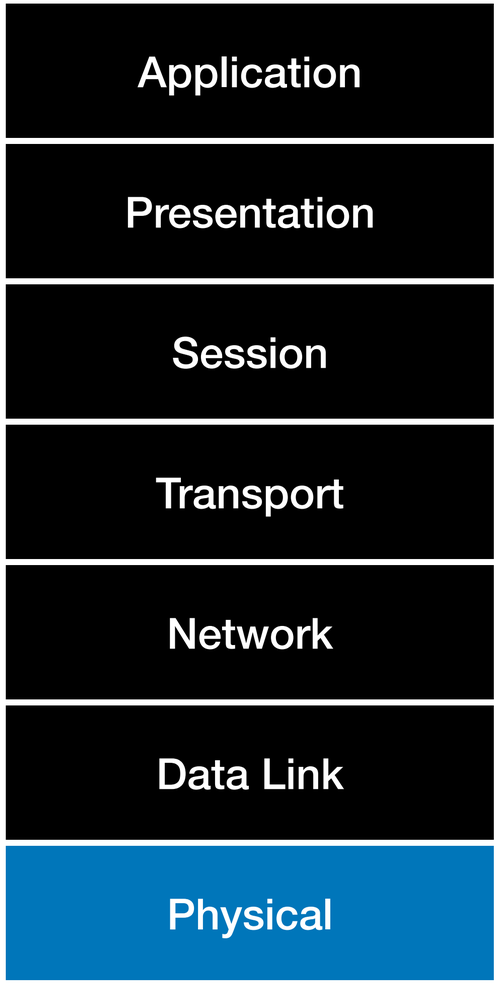
| IP | MAC | TTL |
|---|---|---|
| 172.18.12.92 | 1A:2D | 20 |

- The ARP process needs to happen only once, since router is the gateway
- First step - Check if the IP of the receiver is in the same or different network
- If different network => Send ARP to gateway else, send ARP to all nodes in the network (FF:FF….:FF)

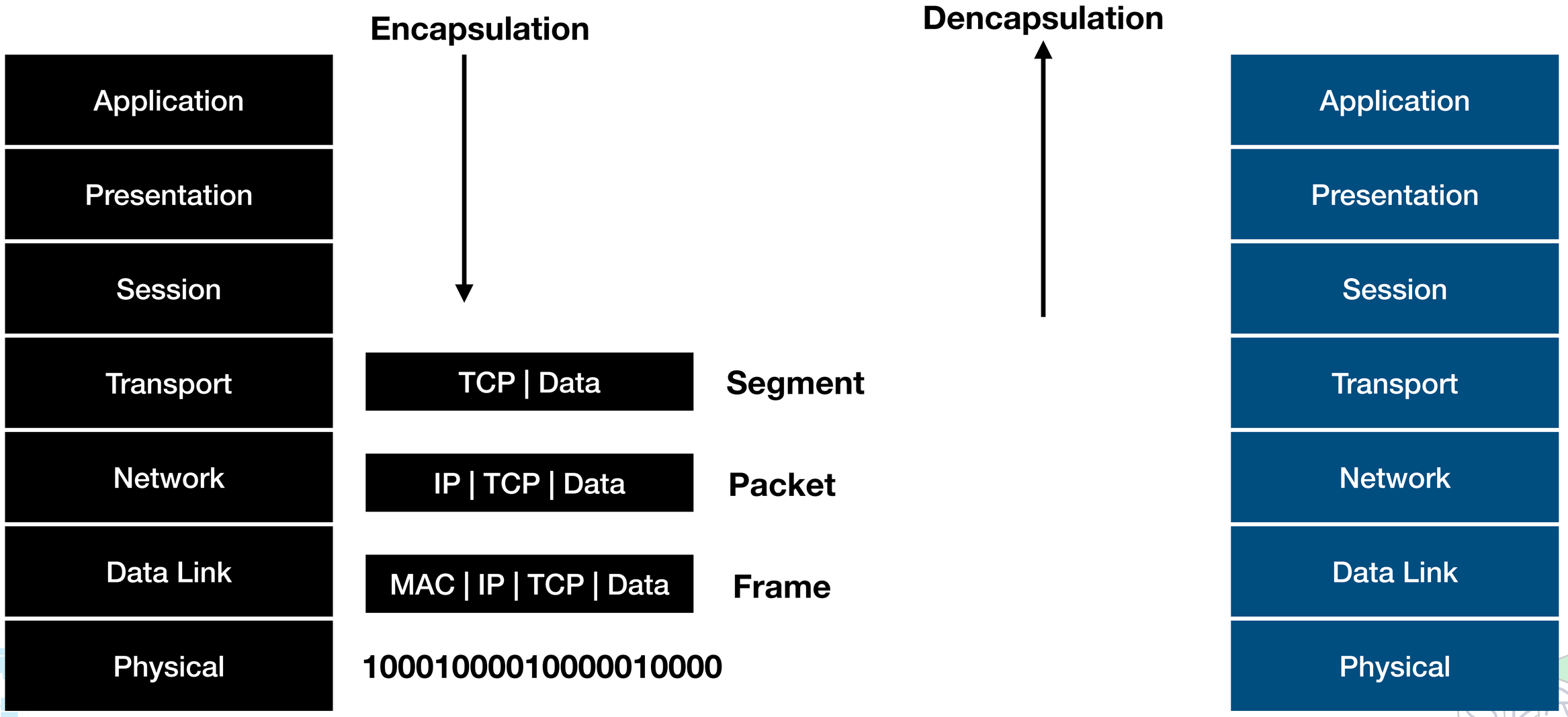
# Physical Layer (L1)

## Ultimately everything is 0's and 1's

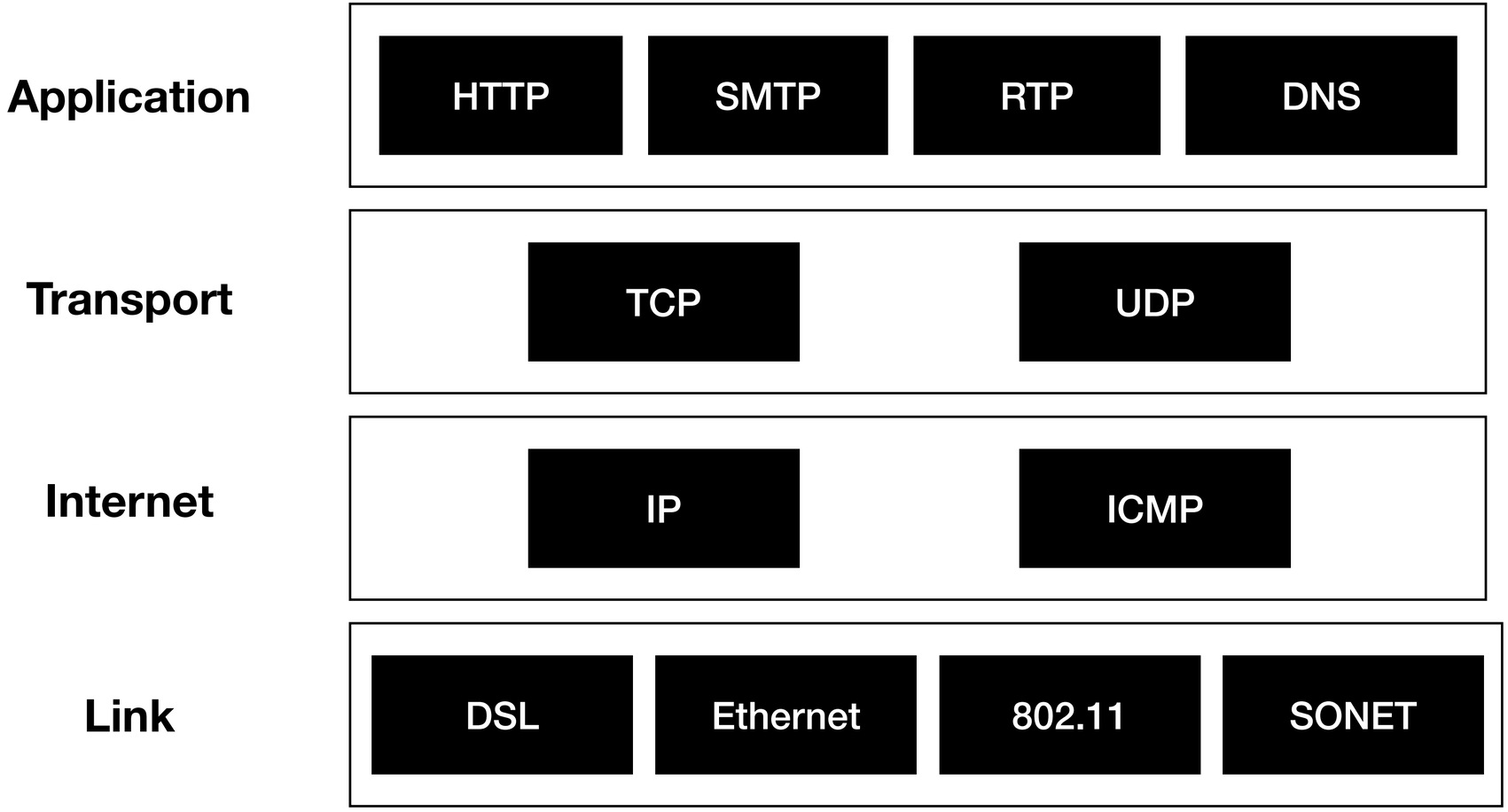
- Data is in the form of bits - 0s and 1s
- Something has to transport the bits from one machine to another - Physical layer
- Concerned with transmission of raw bits over physical medium, like a cable
- L1 technologies: Ethernet cables, Optical fiber, Coaxial cable, etc.
    - Even WiFi is L1 technology, hub, repeater, etc.




# End-to-End OSI View



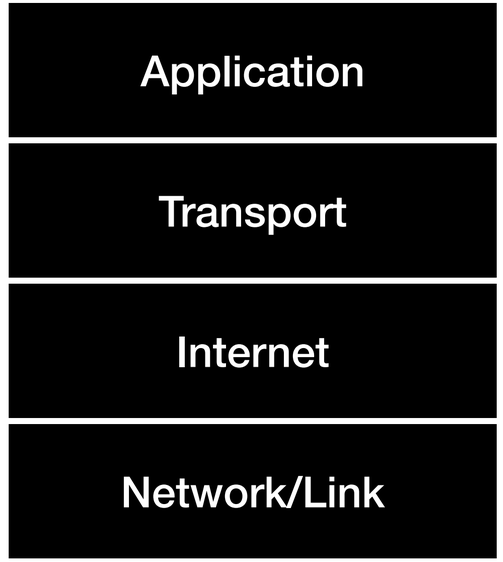
# Network Protocol Stack



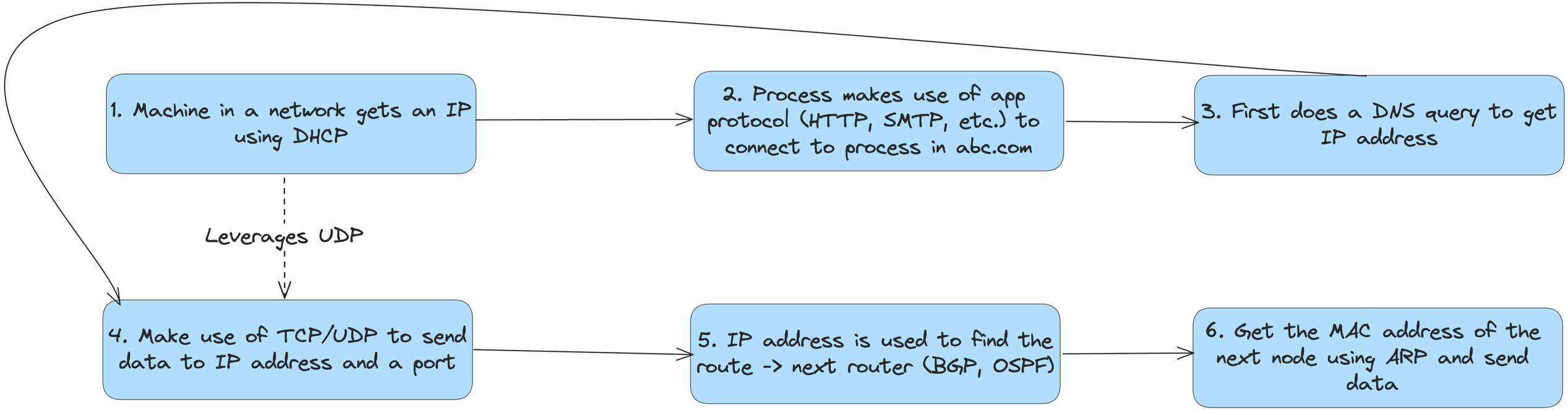
# The 4 Layer Model

## Internet Model or TCP/IP model,

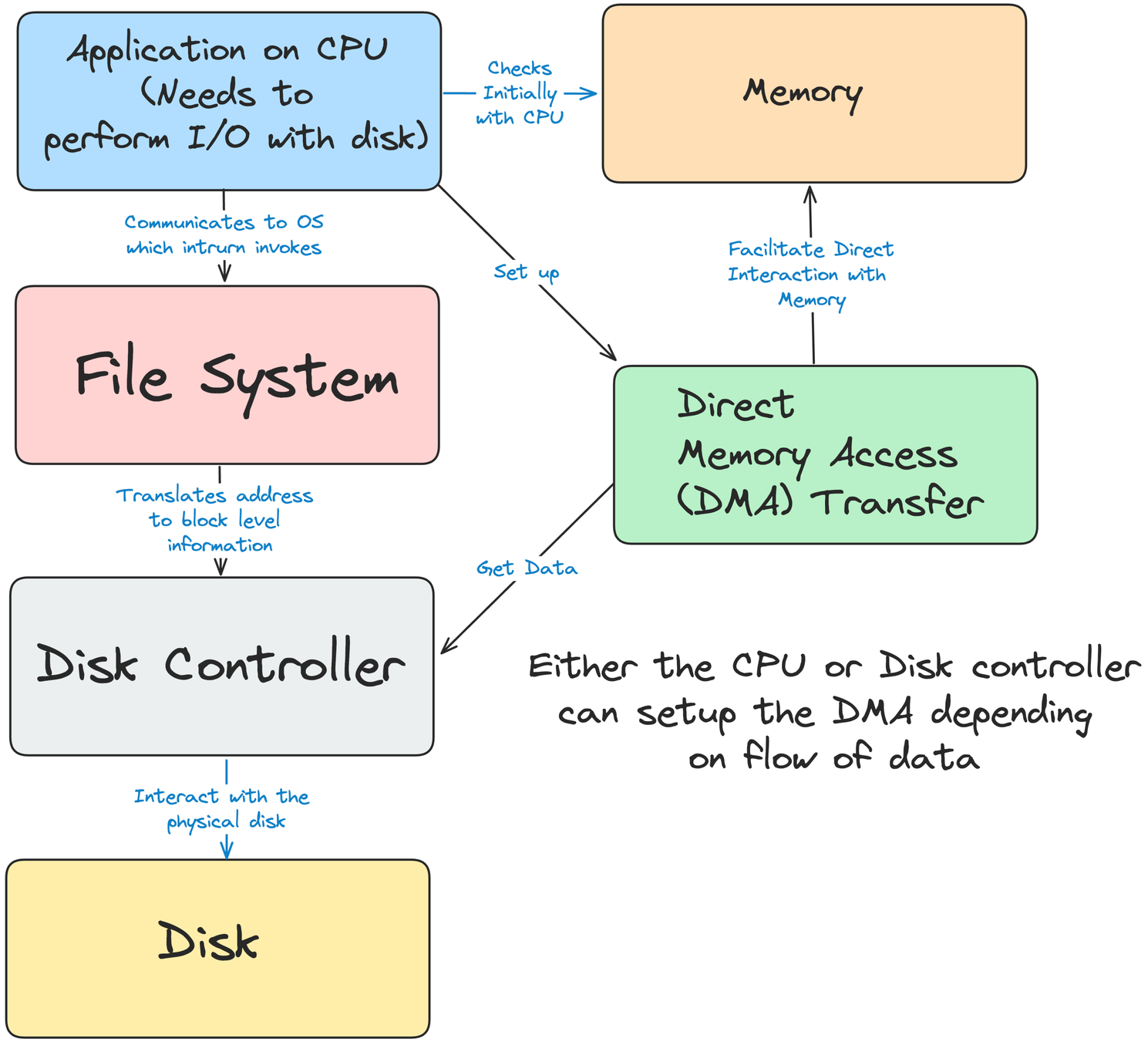
- OSI model is more educational purpose
- 4 layer model more used in reality
- Application layer - Corresponds to application, presentation and session
- Transport layer  - Transport layer of OSI
- Internet layer - Network layer of OSI
- Network - Physical and data link layers of OSI



# Putting it together

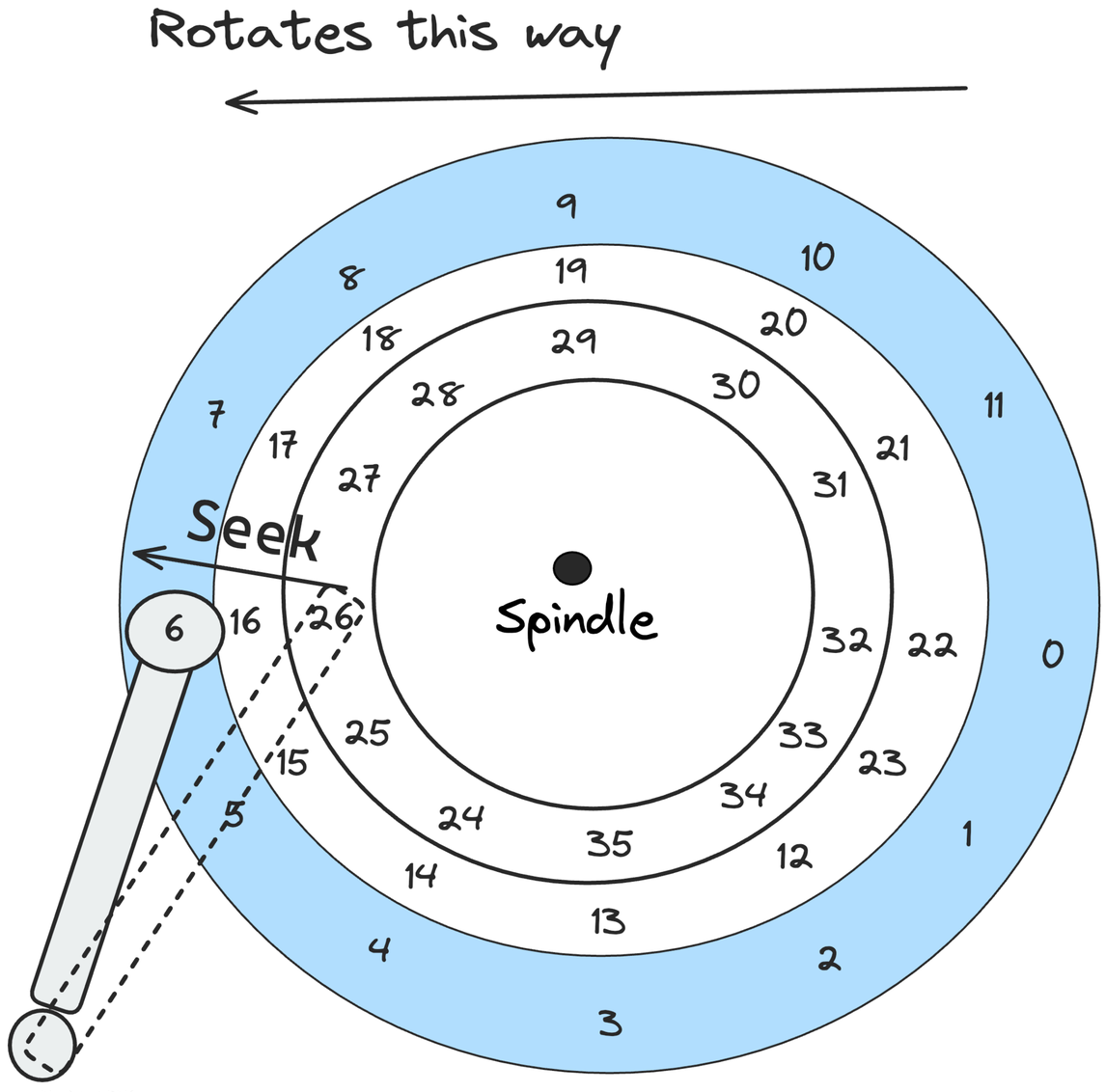


# Persistence



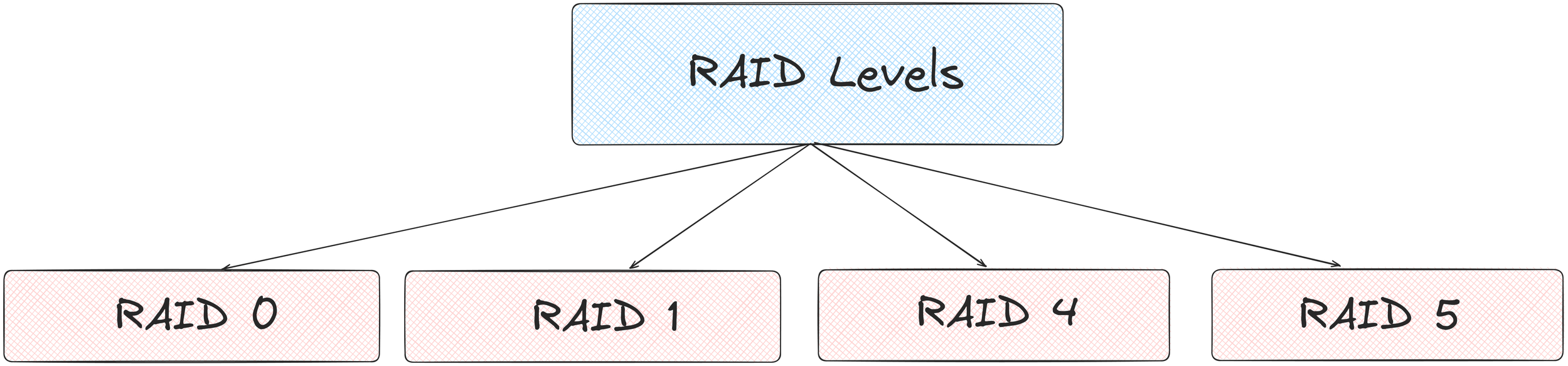

Either the CPU or Disk controller can setup the DMA depending on flow of data

# Disks: An Overview




- Disk rotates on a spindle
  - The arm can move across (seek) or stay as the disk rotates
  - The head is used to read/write
- Data is arranged in tracks as blocks/ sectors
- There are 100s of tracks on a single disk
- **Seek, rotate and transfer** - three key phases

# RAIDs




1. Round robin
2. Capacity N
3. No redundancy

1. Mirroring
2. Capacity N/2
3. Redundancy 1

1. Parity block
2. Capacity N-1
3. Redundancy 1
4. Small writes

1. Distributed Parity
2. Redundancy of 1
3. Capacity N-1

# Breaking down into two main aspects

- We worked on building a **Very Simple File System (VSFS)**
- In any FS, two key things make the difference

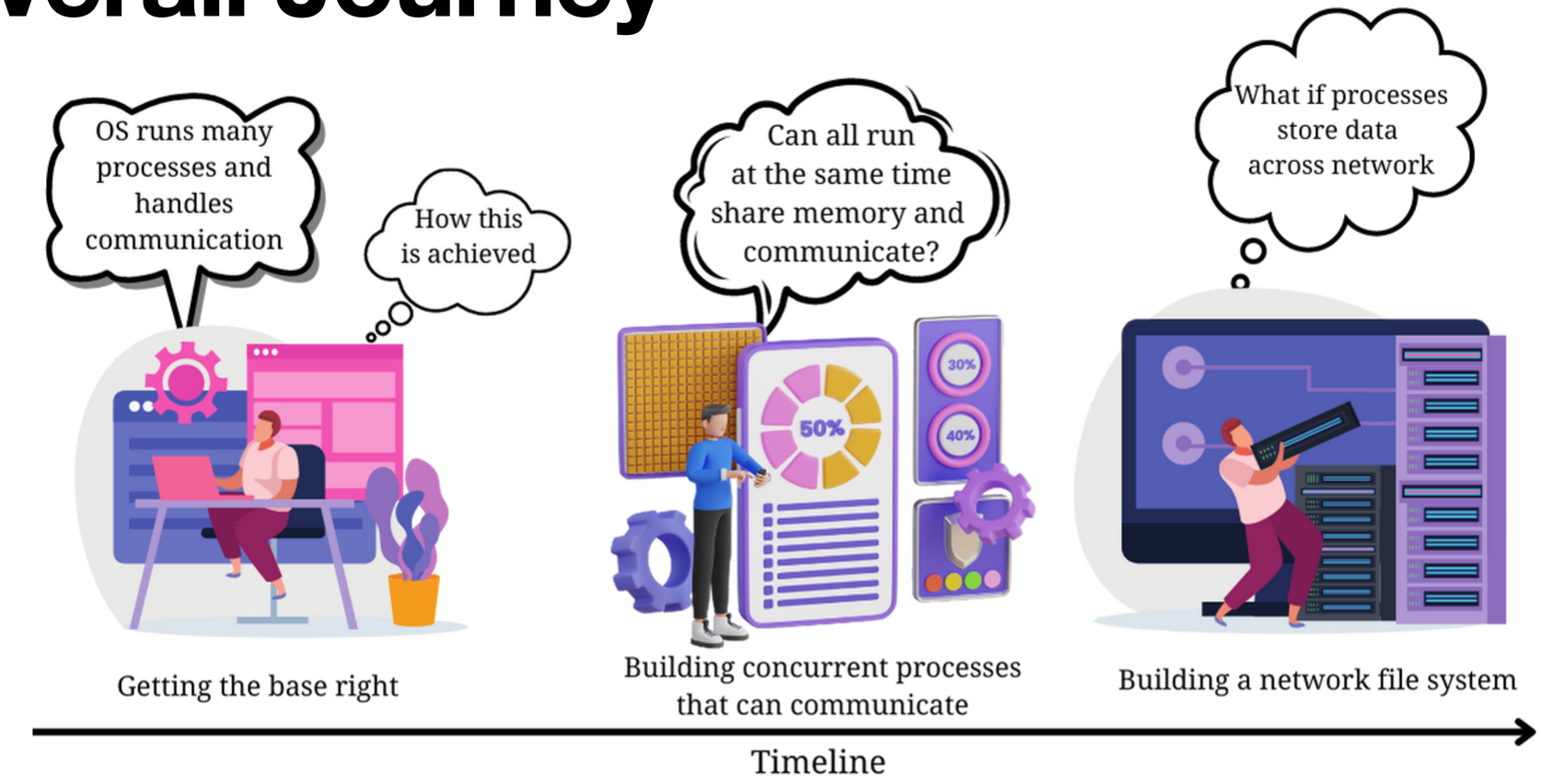
**Data Structures**

- Inode - Data structure for each file
- Store inodes, data, mapping to inodes, etc in a large array

**Access Methods**

- Start with the root
- Traverse through the path using inode mapping
- Caching can be used to improve efficiency

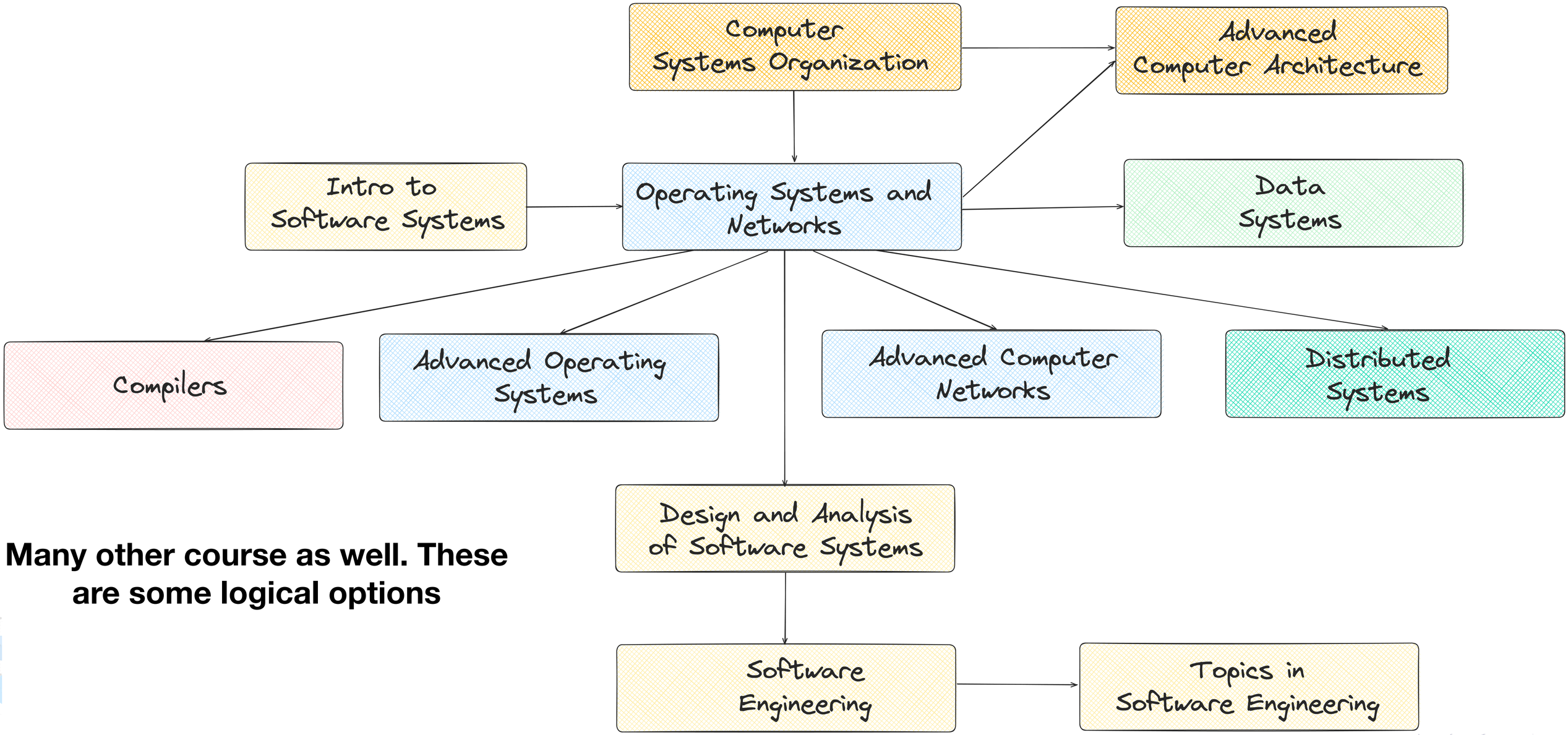
# The overall Journey




**This Course**

| | | |
|---|---|---|
| Process and Memory Virtualization | Concurrency | Persistence |
| Networking intro | Addressing and Routing | Network file Systems |

# What next?




**Many other course as well. These are some logical options**

# Course Restructuring

- **Restructured little more compared to last year! - Course Calendar is more intact!**
- Changes in course logistics and planning
  - Modified the grading scheme
  - Modified **groups in final project**
  - 2 Mini projects - A Lot of changes
  - Almost every project had a network component
  - OS + Networks were kept **more intertwined**
  - Regular meetings with TAs - Increased tutorials and TA meeting hours
  - Feedbacks are always welcome!!

# Thanks to the team!

Akhila Matathammal

Ananya Halagatti

Anirudh Vempati

Aviral Gupta

Karthik Vaidhyanathan

Divijh Mangtani

Eshaan Sharma

Kriti Gupta

Prakhar Jain

Prasoon Dev

Miryala Sathvika

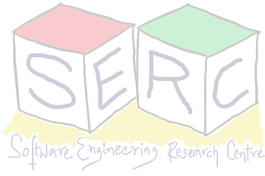
Shlok Sand

Shubham Goel

Varun Gupta

# Thanks to all of you!






# Thank you

**Course site:** karthikv1392.github.io/cs3301_osn
**Email:** karthik.vaidhyanathan@iiit.ac.in
**Twitter:** @karthi_ishere