# CS3.301 Operating Systems and Networks

## Process Virtualisation - Policies (Scheduling) and Process Communication (Intro to networks)

**Karthik Vaidhyanathan**

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

SERC

Software Engineering Research Centre

# Acknowledgement

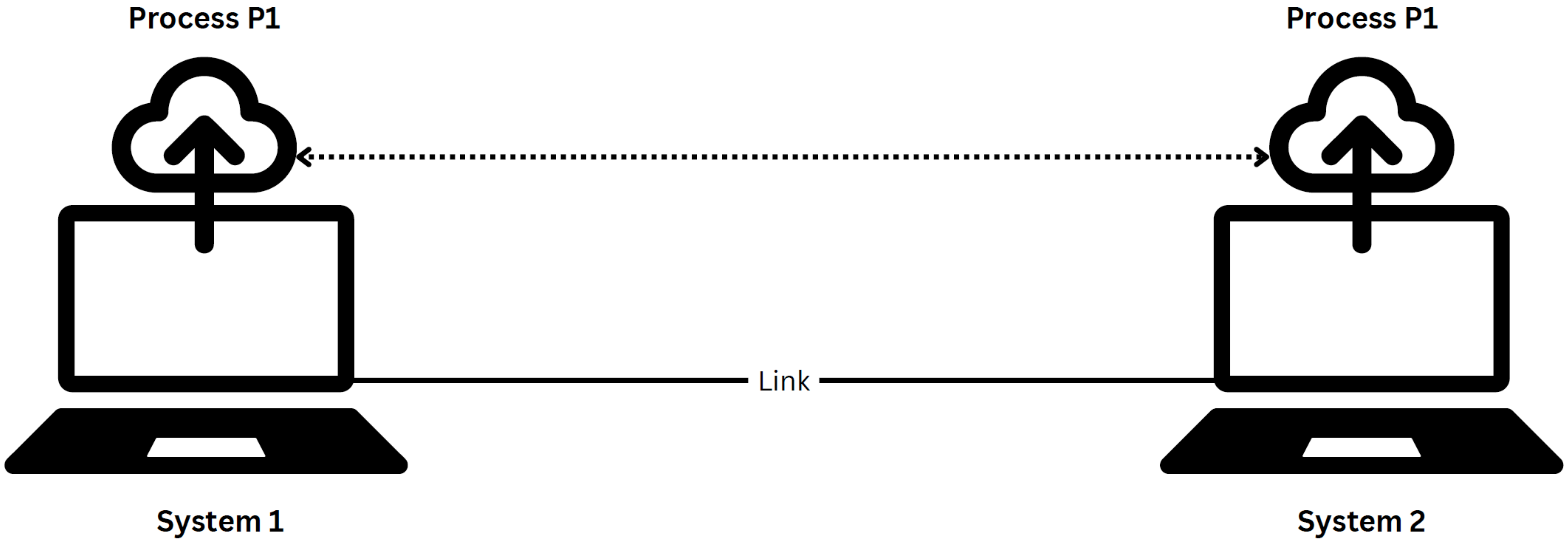
The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidhyanathan

Sources:

- OSTEP Educator Materials, Remzi et al.
- OSTEP Book by Renzi et al.
- Modern Operating Systems, Tanenbaum et al.
- Networking Fundamentals by Practical Networking (Youtube Channel)
- Other online sources which are duly cited

# What about processes in different machines?

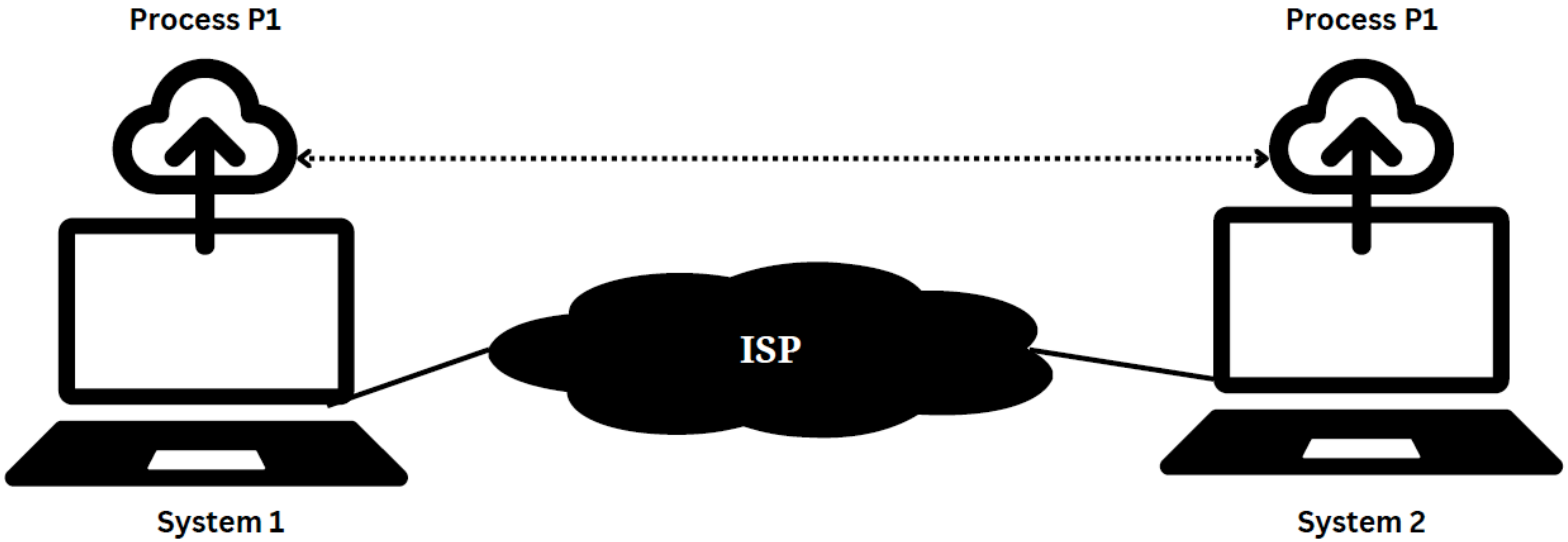
## How can they communicate?



How does message/data from P1 in System 1 reach P1 in System 2?

What is the role of the OS in this and how does it contribute to the effectiveness?

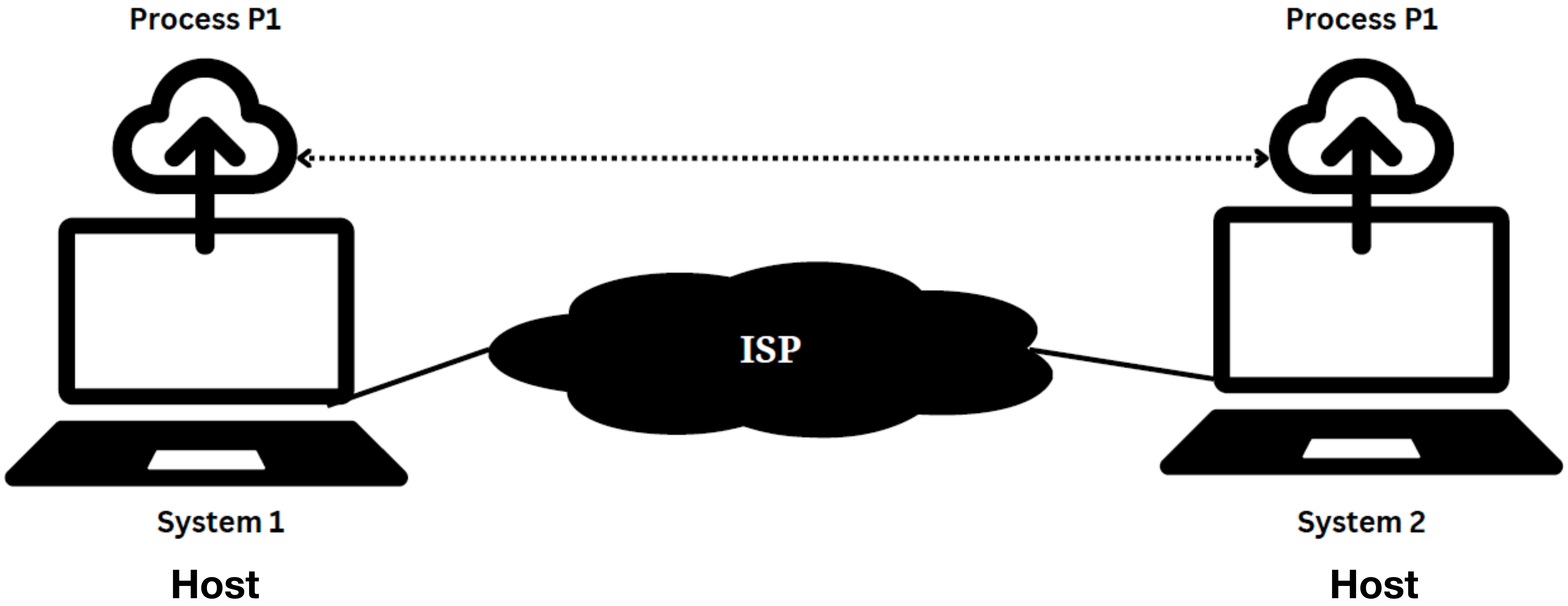
# Let us expand it



- Between the process and network there needs to be an interface
- Between the network components there needs to be some interface

# Network Components

## Host



- Any device that send or receive traffic: Computer, laptop, smartwatch, phone, etc
- Host can be client or server
- Servers can sometime be clients too

# How does host send data?

## IP Address

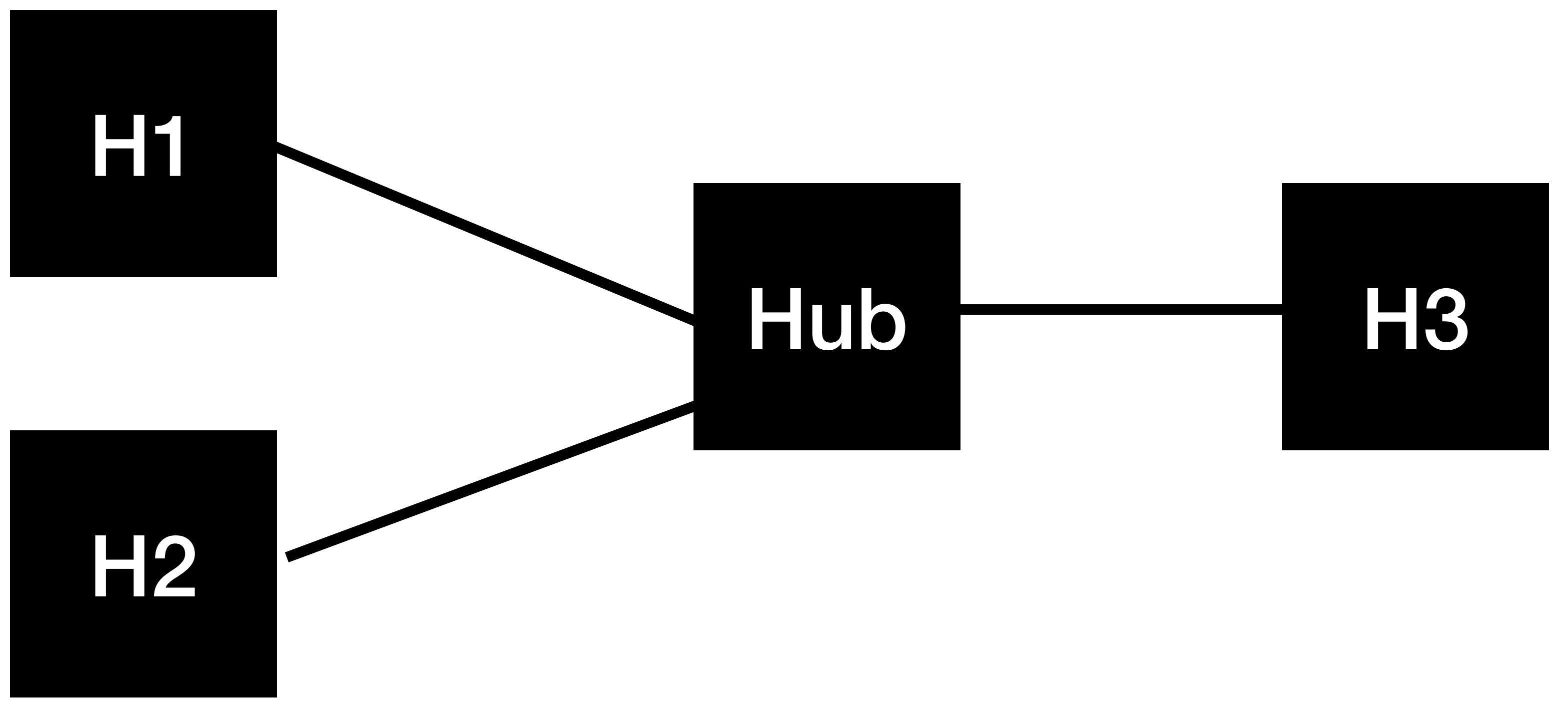
- Host needs address to send the data
  - This address is known as **IP address**
  - When client sends data it provides both source IP and destination IP
  - IP addresses are 32 bits. Each bit can be 0 or 1
  - Total of 4 octets. Each octet ranges from **0-255 (8 bits)**
  - IP addresses are usually hierarchically assigned
  - **Eg:** 192.168.1.1

# Sometimes we need more signal Strength

## Repeater

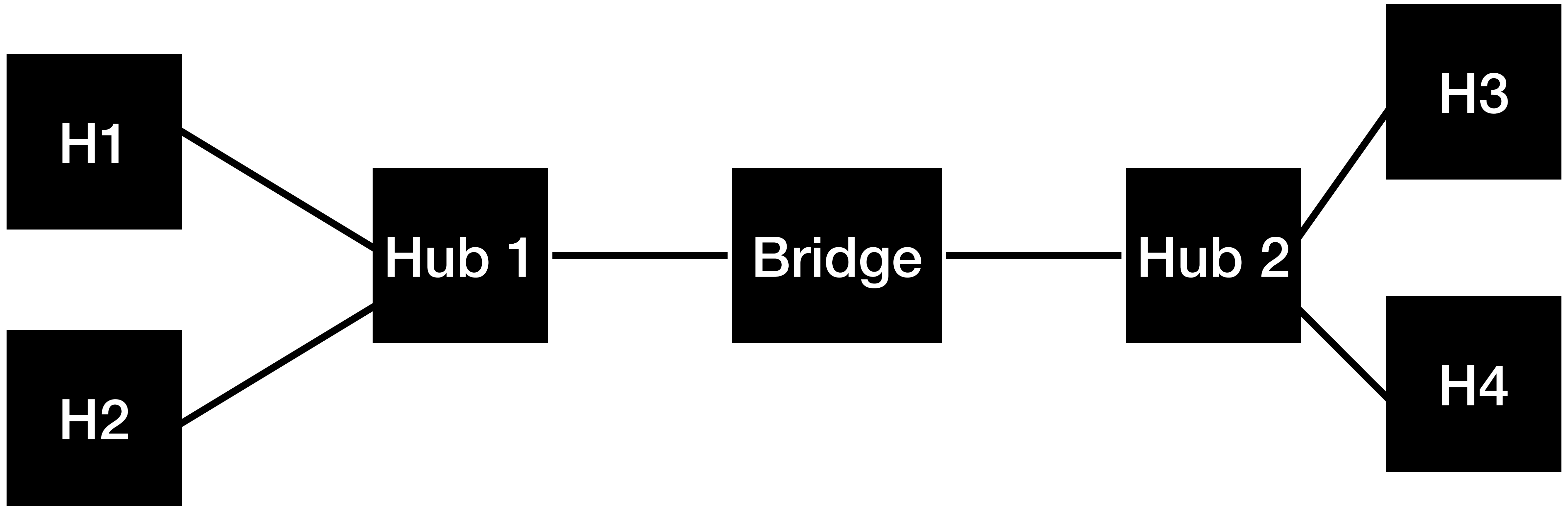
- Repeater allows regeneration of signals for long distance communication
- Think of wifi in home
  - Signal strength may not be there
  - Repeater might be needed to transmit to longer ranges
- But we cannot just connect one host to another - **Not Scalable!**

# Hubs and Bridges



**Hubs**

- Multi-port repeaters
- Key issue: Everyone receives everyones data

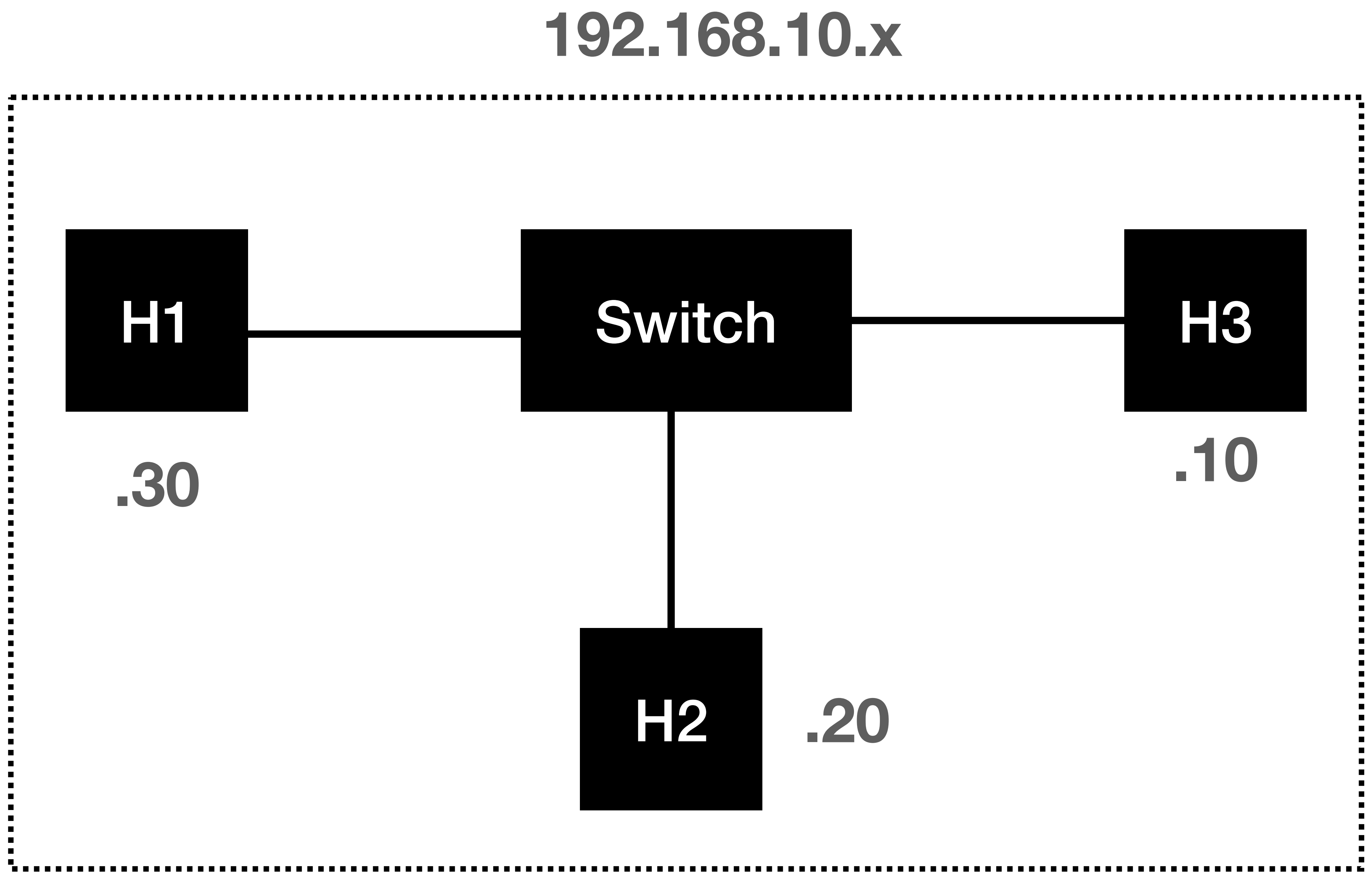


**Bridges**

- Sit between two hubs
- They have only two ports
- They learn which hosts are on either side (for routing)
- Eg: H1 wants to communicate with H2

# Switches

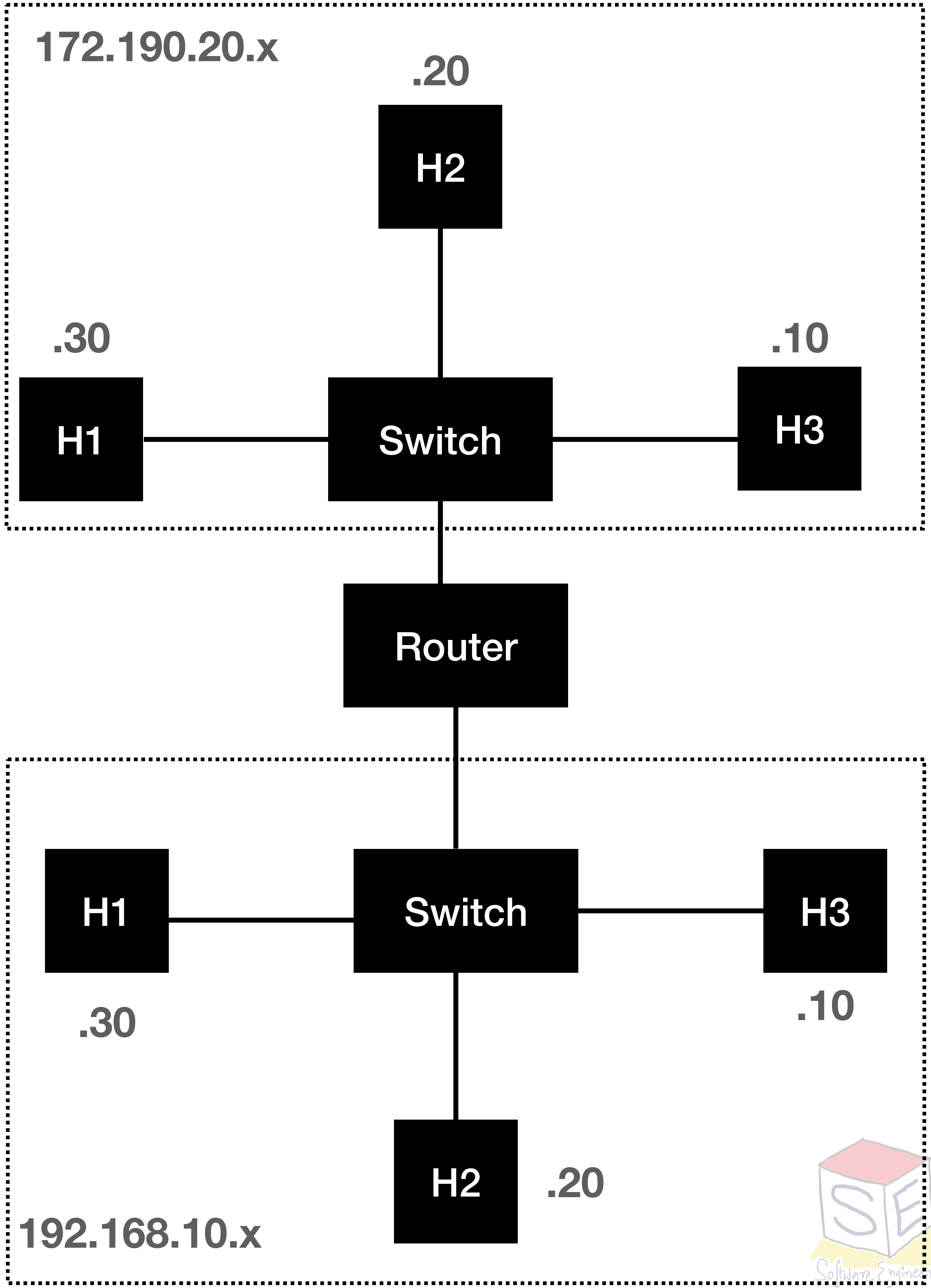
- Devices which facilitate communication within a network
- Combination of hubs and ports
- Knows or infers which hosts are on each port
- They have multiple ports
- Whatever connected to switch becomes part of one network



What if the host 192.168.10.20 wants to communicate with Another host in different network?

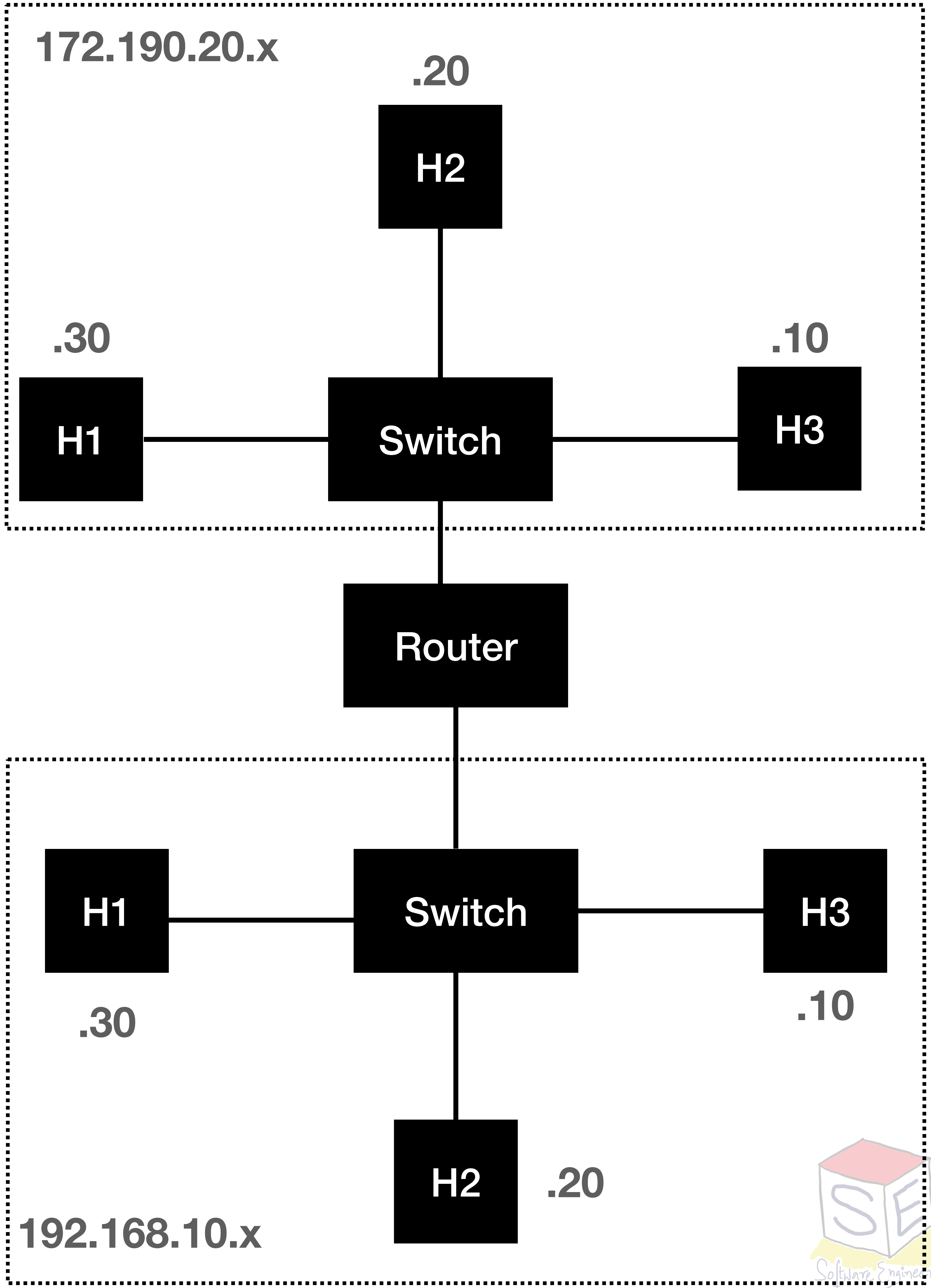
# Routers

- Facilitate communication between the networks
- They provide like a traffic control point
  - Security, filtering, redirection
- Routers learn which networks they are attached to
  - Known as routes
  - Stored in a routing table
- Routers have their IP address in the network they are attached to

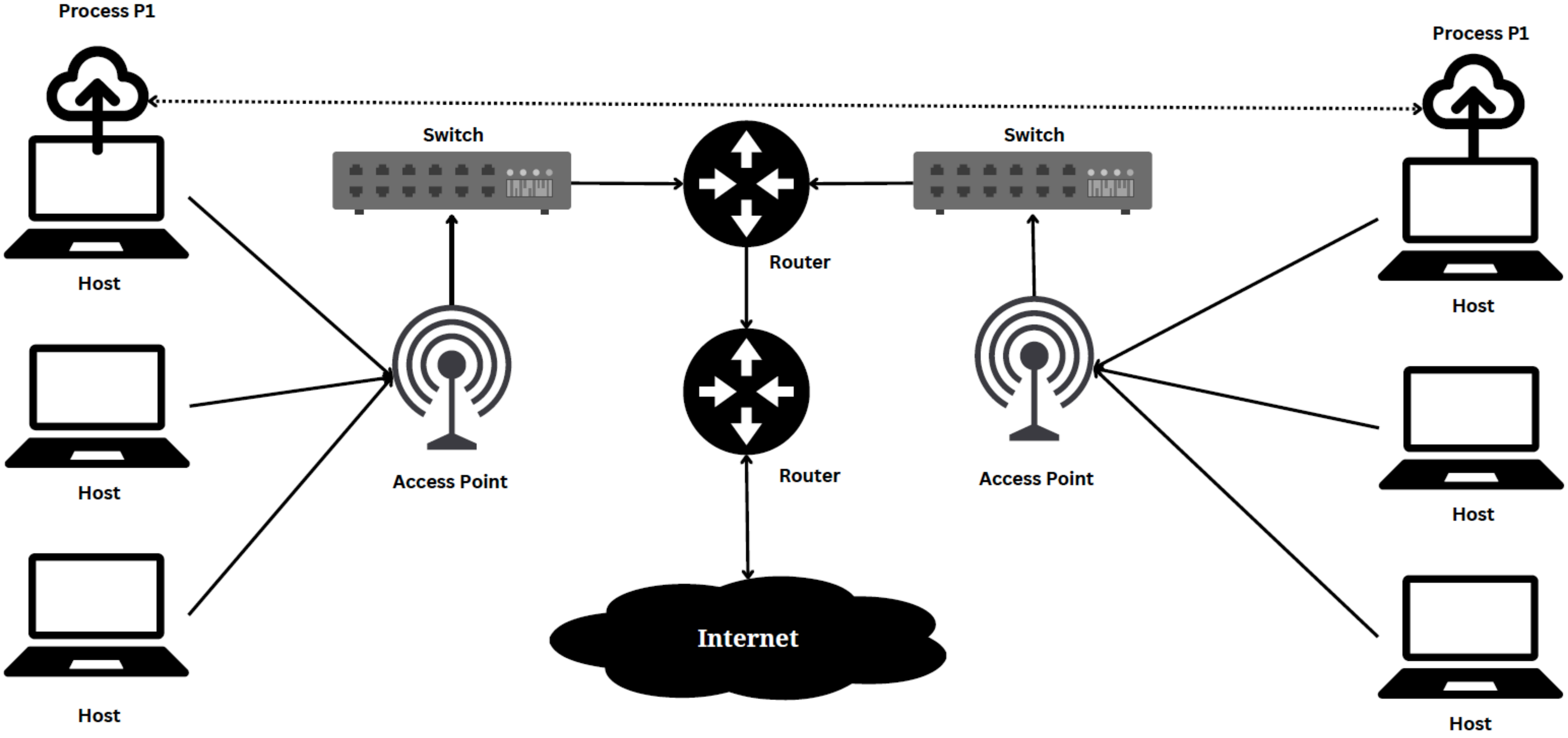


# Routers

- Allows creation of hierarchy in the networks
- Every time hosts wants to go out of the network, it goes through router
- Internet is nothing but bunch of routers
- Gateway (IP address of router in the given network)
  - Becomes exit point of hosts outside their network

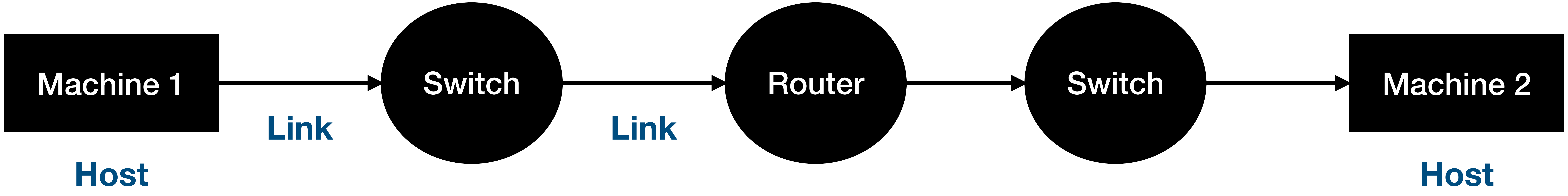


# The Bigger Picture



# What happens internally when one process wants to share/receive data from another host?

# Components of a network



# Process Communicating over network

- Process A (eg: Whatsapp) is executing in Host 1
- Process B (whatsapp) executing in Host 2
- Process A wants to communicate with Process B
    - Some system call is made to access the network
        - Leads to an interrupt
        - Process A should know where process B is
        - Hardware support is needed (some network device with drivers)

# Network Types

## Multiple types of networks exist

- **Personal Area Network - PAN**
  - Devices communicate over a very short range
  - Eg: Bluetooth (master slave paradigm)
- **Local Area Network - LAN**
  - Private network operating within and nearby a single building (home, office, factory, etc.)
  - Wireless LANs runs speed from 11 Mbps to 7 Gbps (1 Mbps is 1,000,000 bits per sec) - Power in 10 (not in terms of 2)
  - Wired LANs operate at speed ranging from 100 Mbps to 40 Gbps (low latency)
  - Eg: connecting personal computers and consumer electronics (eg: printer)

# Network Types

## We have come a long way!

- One large Physical LAN can be divided into smaller logical LANs (Virtual LANs)
- Earlier days it was about broadcasting on a single line!
  - At most one computer could transmit successfully at a time
  - Use static allocation techniques
    - Every machine gets some time to transmit or receive
    - Round robin was used for scheduling
  - Packet collision used to happen (Wait for some random time and try again)

# Network Types

- **Metropolitan Area Network - MAN**
  - Covers a city (City wide networks)
  - Think of cable TV networks (Earlier Antenna on top of the house and from the top of the hill the communication used to happen)
  - Idea of cable TV was used for providing two services (Internet and TV)
- **Wide Area Network - WAN**
  - Spans a large geographical area (often country, continent, etc.)
  - Eg: Internet is a large WAN (Dedicated WANs also exist)
    - There can be WAN that connects offices of organisation in different locations
  - Higher latency and lower transmission speeds. Cost for dedicated ones can be high

# An Illustrative Scenario

## Sending Letter/Courier

Community 1

Local Post office

City Post Office

Community 2

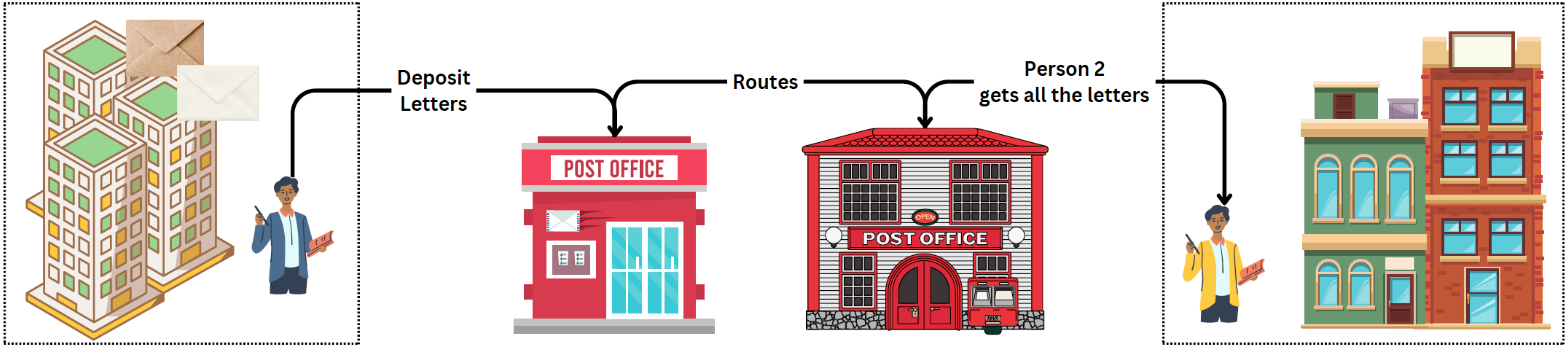
Two large communities in different parts of the city, two people in each community to collect and deliver the letters/couriers

# An Illustrative Scenario

People in community send letters/couriers

One person is collecting all the letters/couriers

Person delivers the letter to each person



Deposit Letters

Routes

Person 2 gets all the letters

Community 1

Local Post office

City Post Office

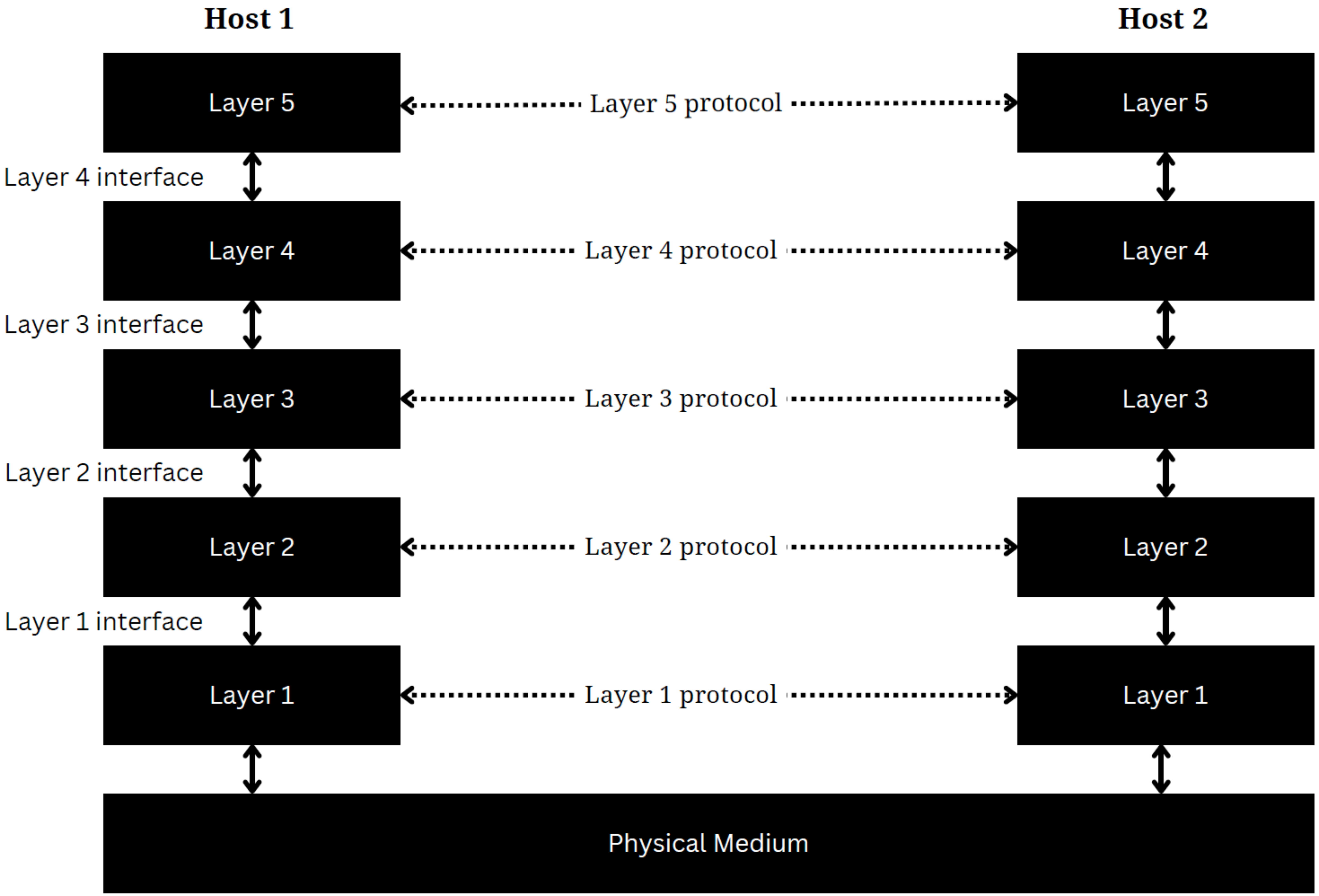
Community 2

Key points: Door number, building number, Post office has an identifier, etc.

# Lets take this to Process communication

- 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

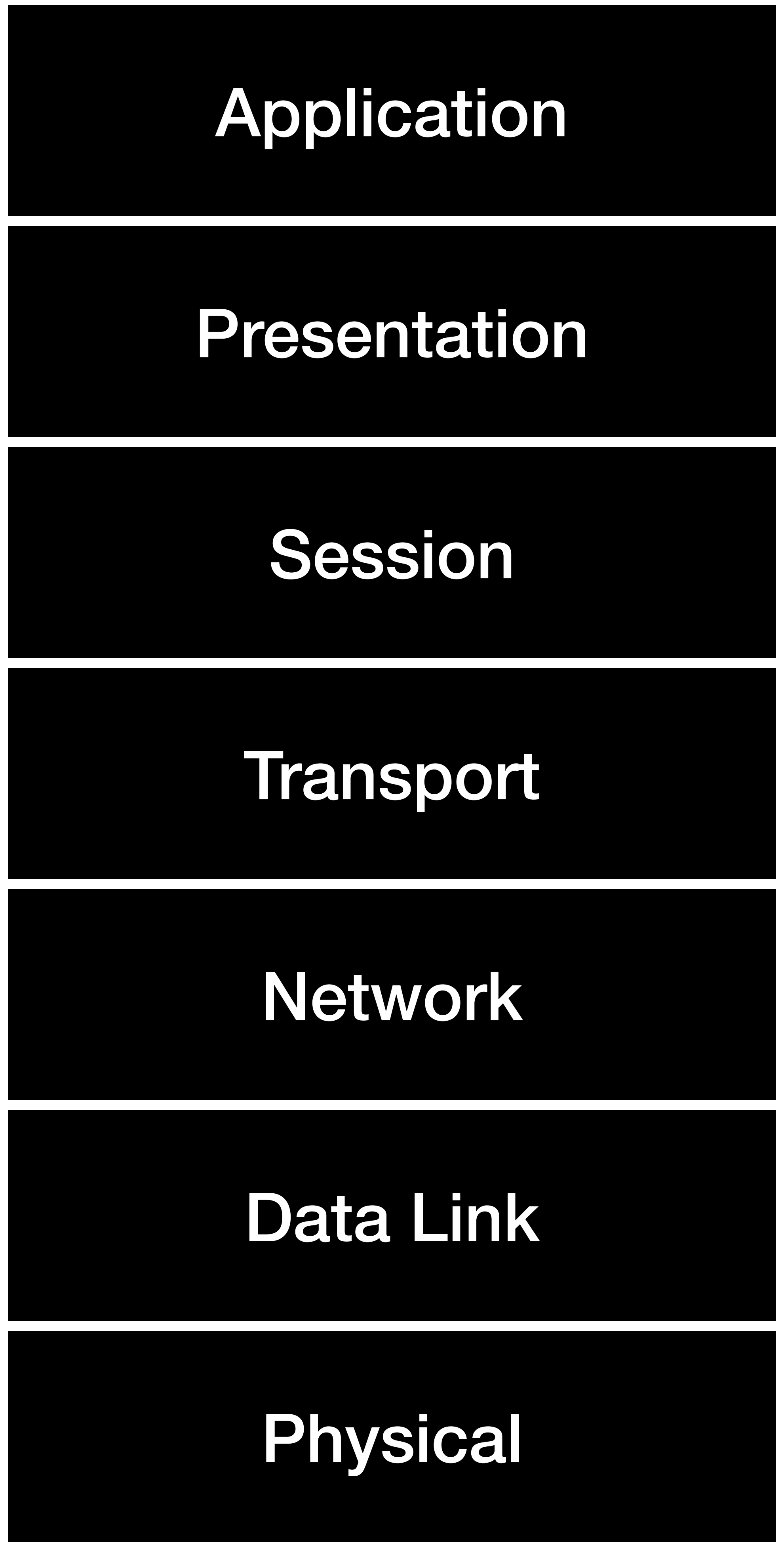
# Networking Layers



# Networking Layers

- Reduce design complexity, network organised a stack of layers or levels
- The layering provides abstraction in and to other layers
- Communication (in some sense) happens between the corresponding layers
    - Layer n one machine communicates with layer n of another using a **protocol**
    - Protocol is just like agreement between communicating practices
- Set of layers along with protocols => **Network Architecture**
- Between each pair of adjacent layer their is an **interface**
    - Defines the primitive operations and services the lower layer makes available

# The OSI Model



Application

Presentation

Session

Transport

Network

Data Link

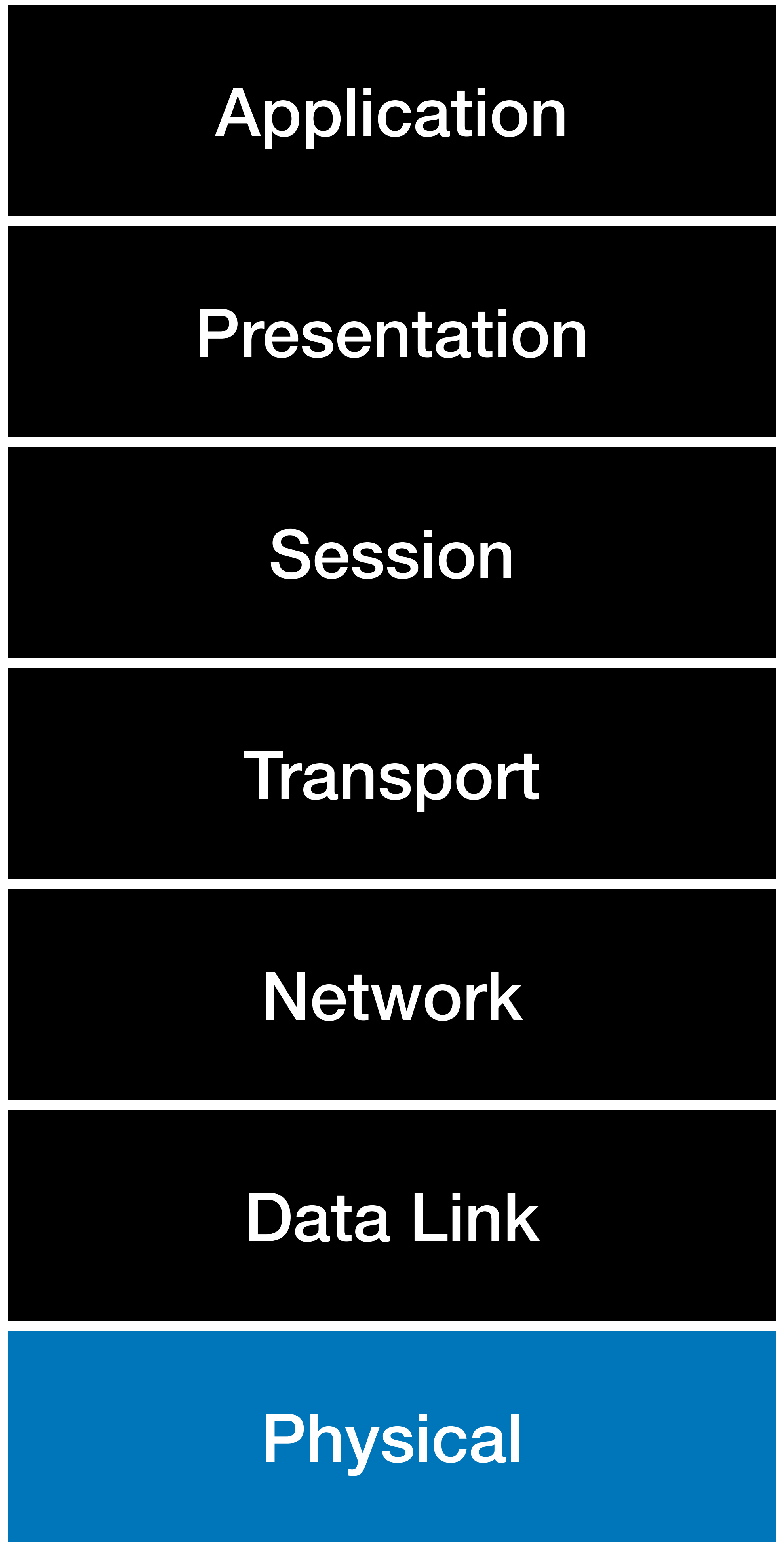
Physical

- 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

# Physical Layer (L1)

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

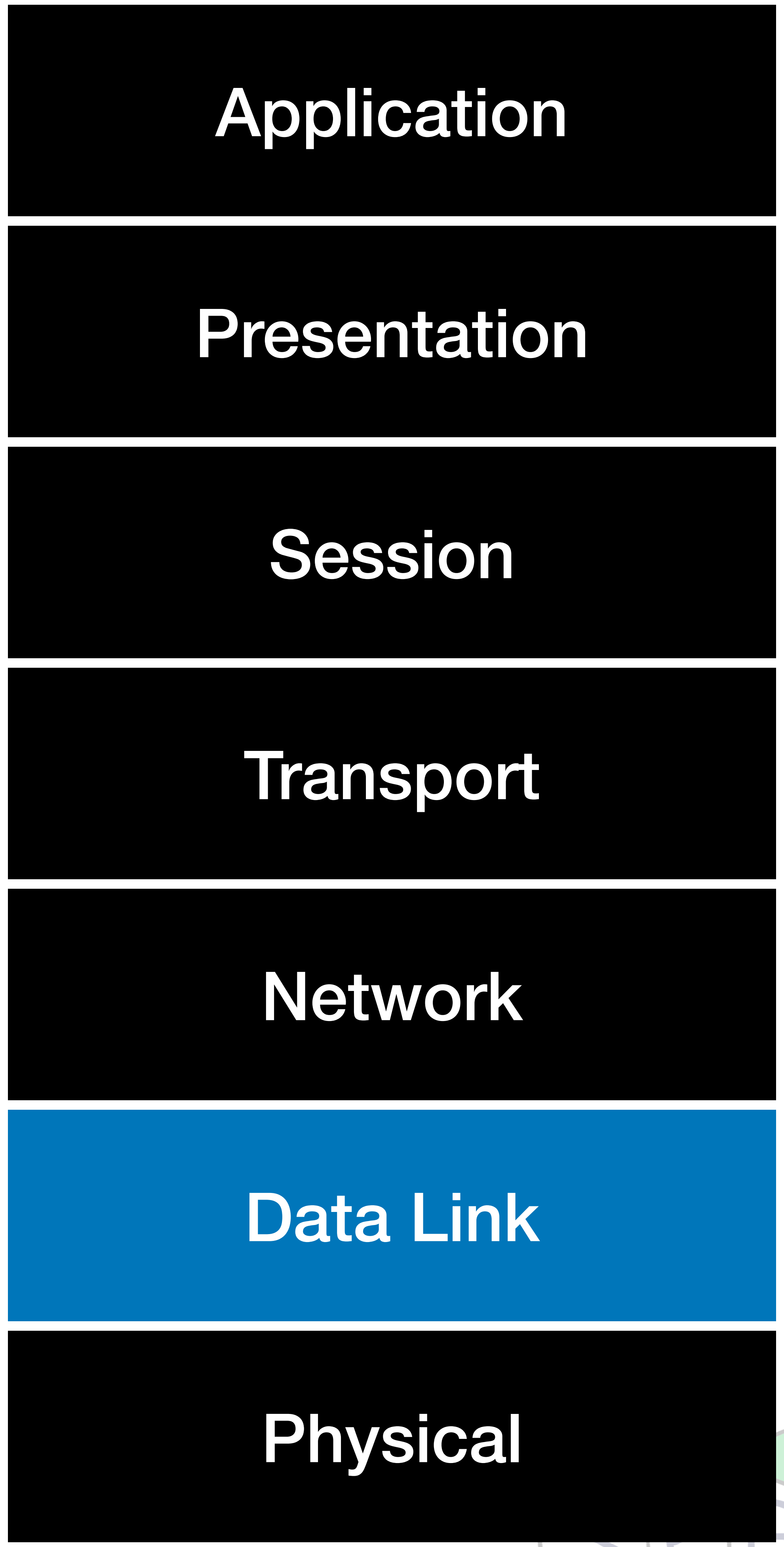
- Data is in the form of bits - 0s and 1s
- Something has to transport the bits from one machine to another - Physical layer
- Concerned with transmission of raw bits over physical medium, like a cable
- L1 technologies: Ethernet cables, Optical fiber, Coaxial cable, etc.
  - Even WiFi is L1 technology,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



# Data Link Layer (L2)

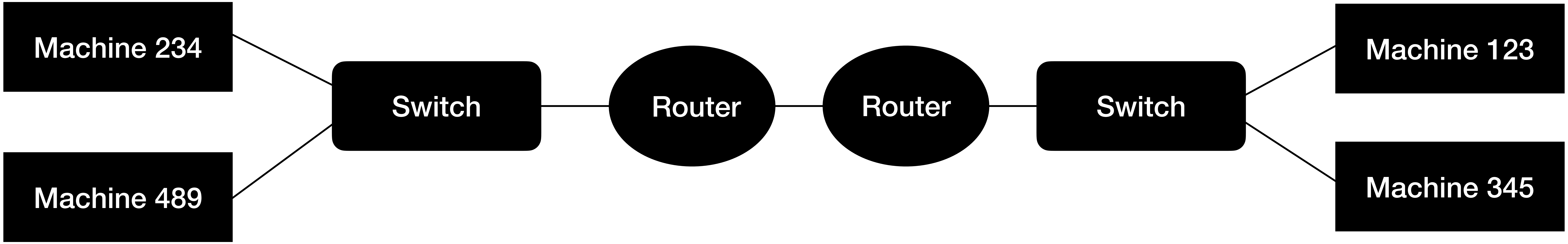
## Hop to Hop connectivity

- Interacts with the physical medium
- Adds/takes data to/from L1 technology
- Responsible for creating a reliable link between two directly connected nodes, error correction, etc.
- L2's responsibility is mainly taking data from one hop to another
  - Uses an addressing scheme - MAC addressing
  - 48 bits represented as 12 hex digits
- L2 Technologies : NIC, WiFi access cards
- Switches are also L2 technology (help in moving data)



# Communication is just more than Hop to Hop

## What about communication from 234 to 345?

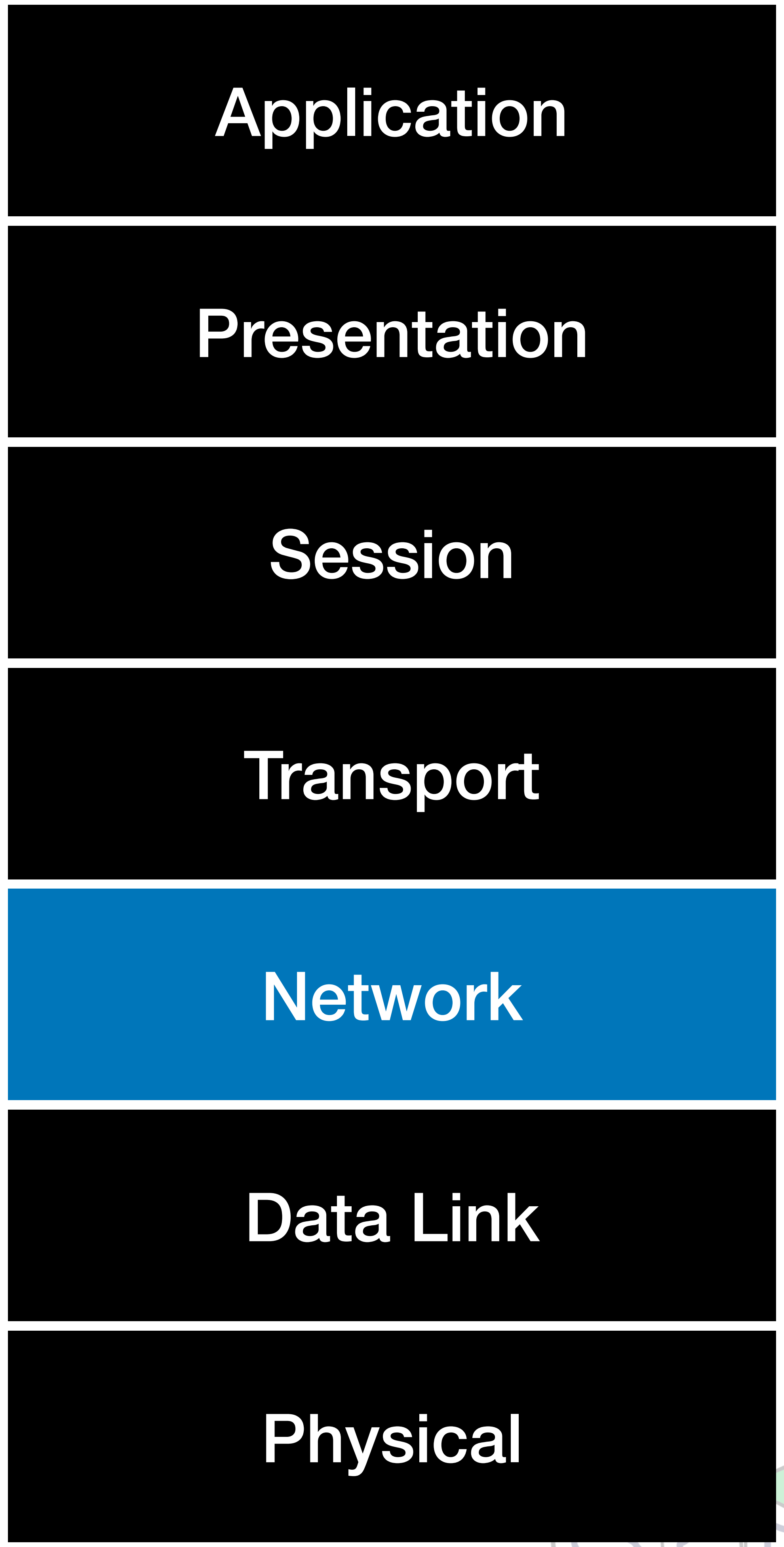


- Hop to hop is guaranteed by L2
- What about end-to-end delivery?

# 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



Application

Presentation

Session

Transport

Network

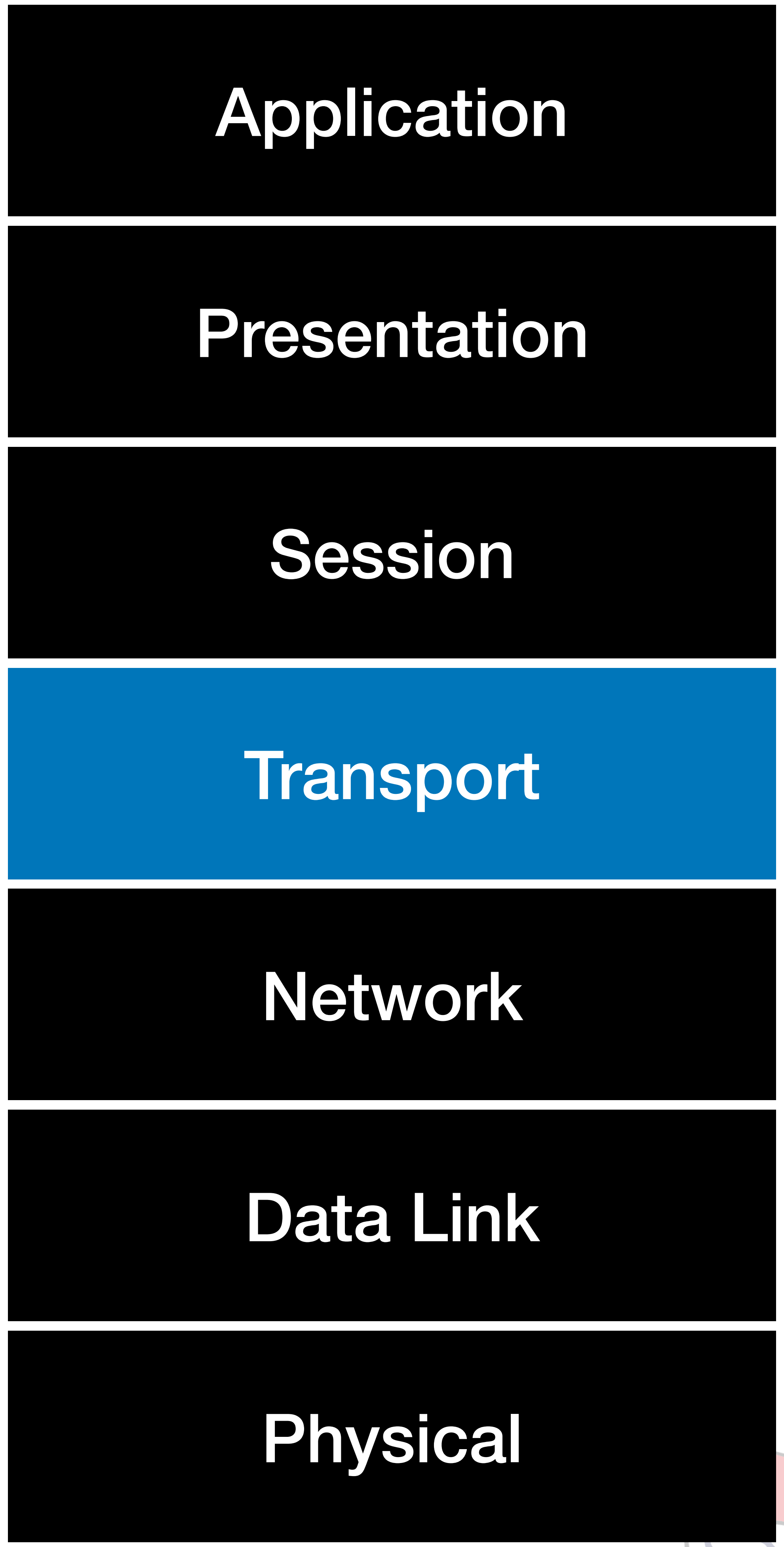
Data Link

Physical

# Transport Layer (L4)

## Service to Service

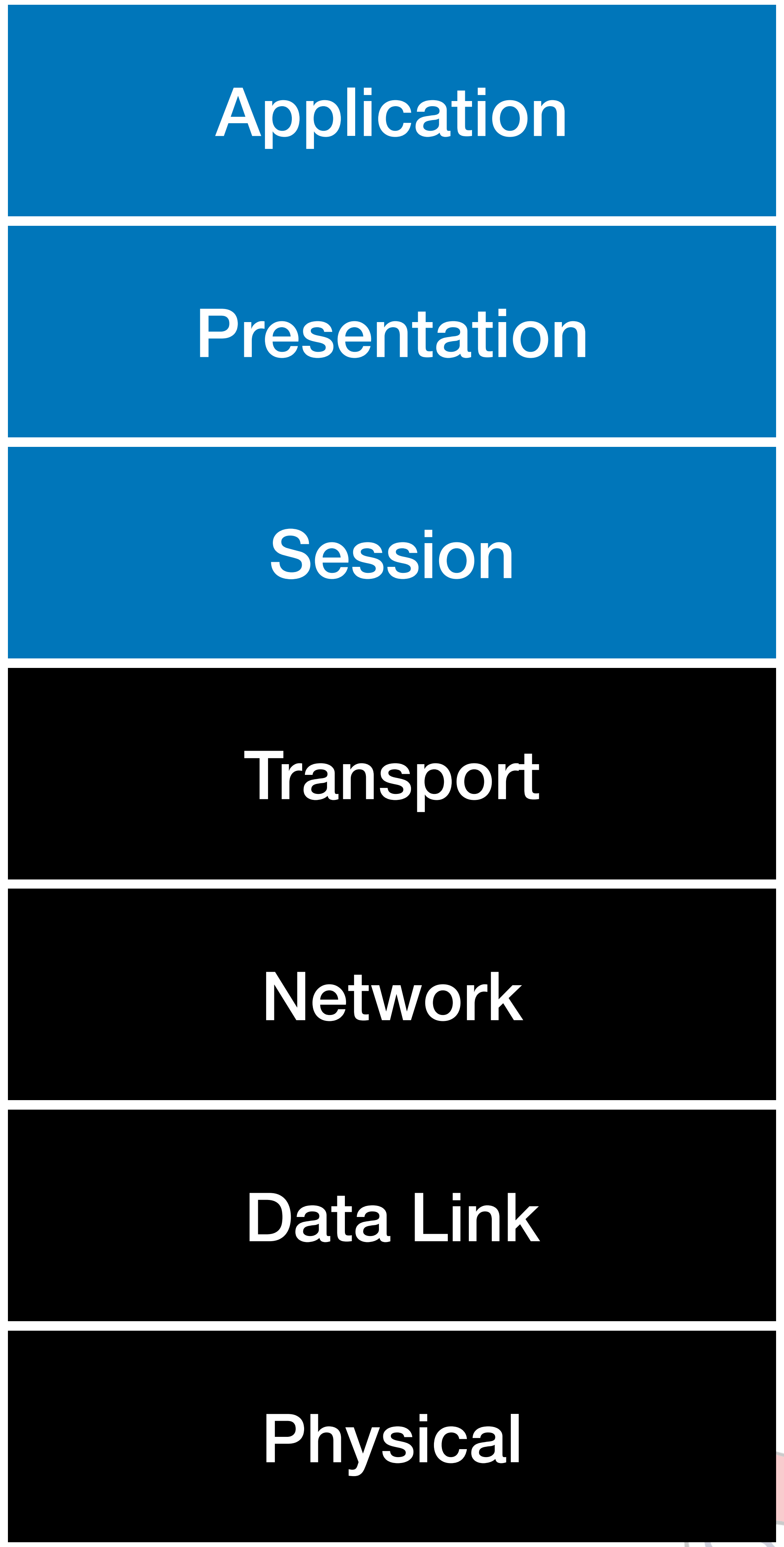
- How to ensure that the right process receives the data?
  - Many process will be executing or are waiting to execute?
- Ensures data transfer is reliable, sequential and free from others
- Manages flow control and error correction
- Layer 4 has an addressing scheme to guarantee message delivery
  - Ports! (0 - 65535), Privileged: 0-1023, Registered: 1024 - 49151
  - Every process will have a port through which sending or receiving data
- L4 Technologies: TCP, UDP



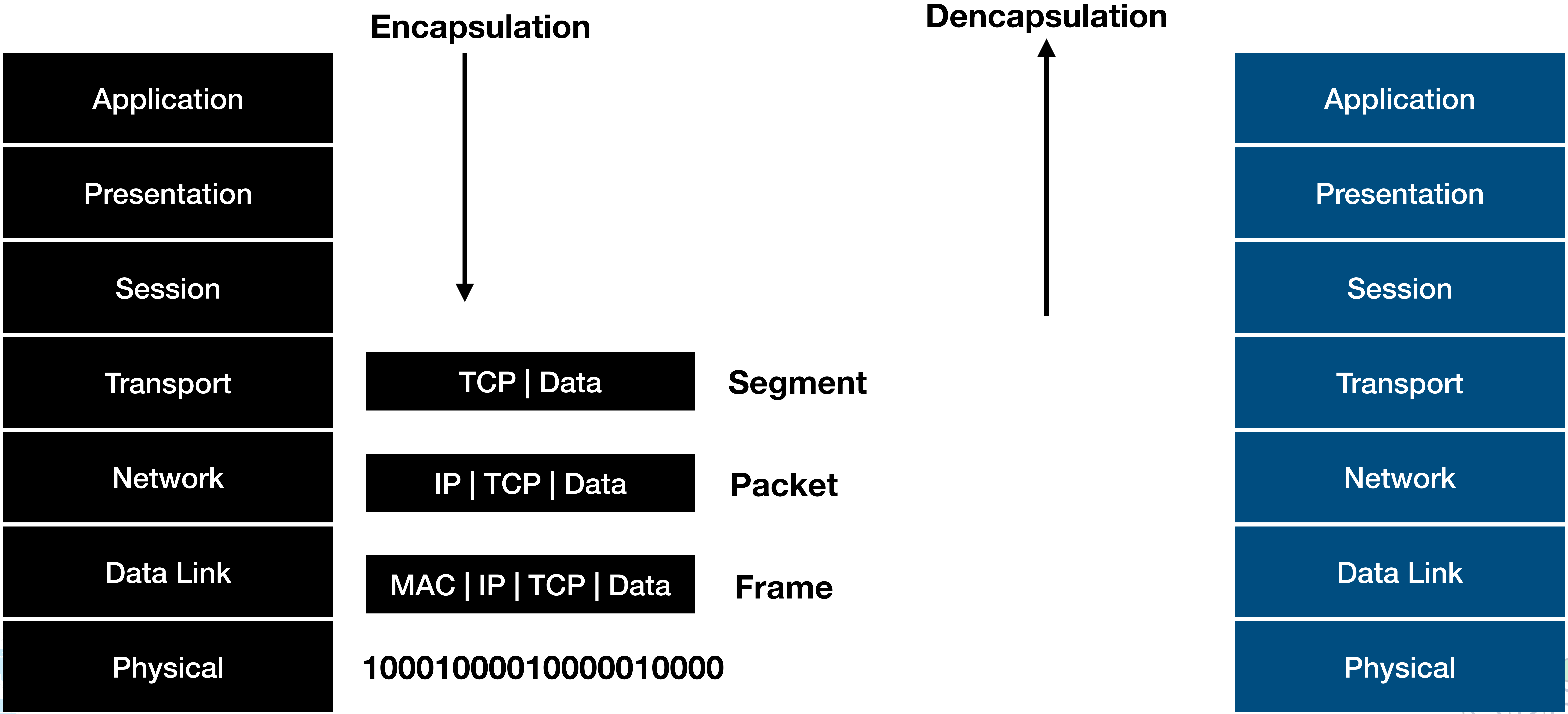
# Session, Presentation and Application

## 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



# Putting It Together



Encapsulation

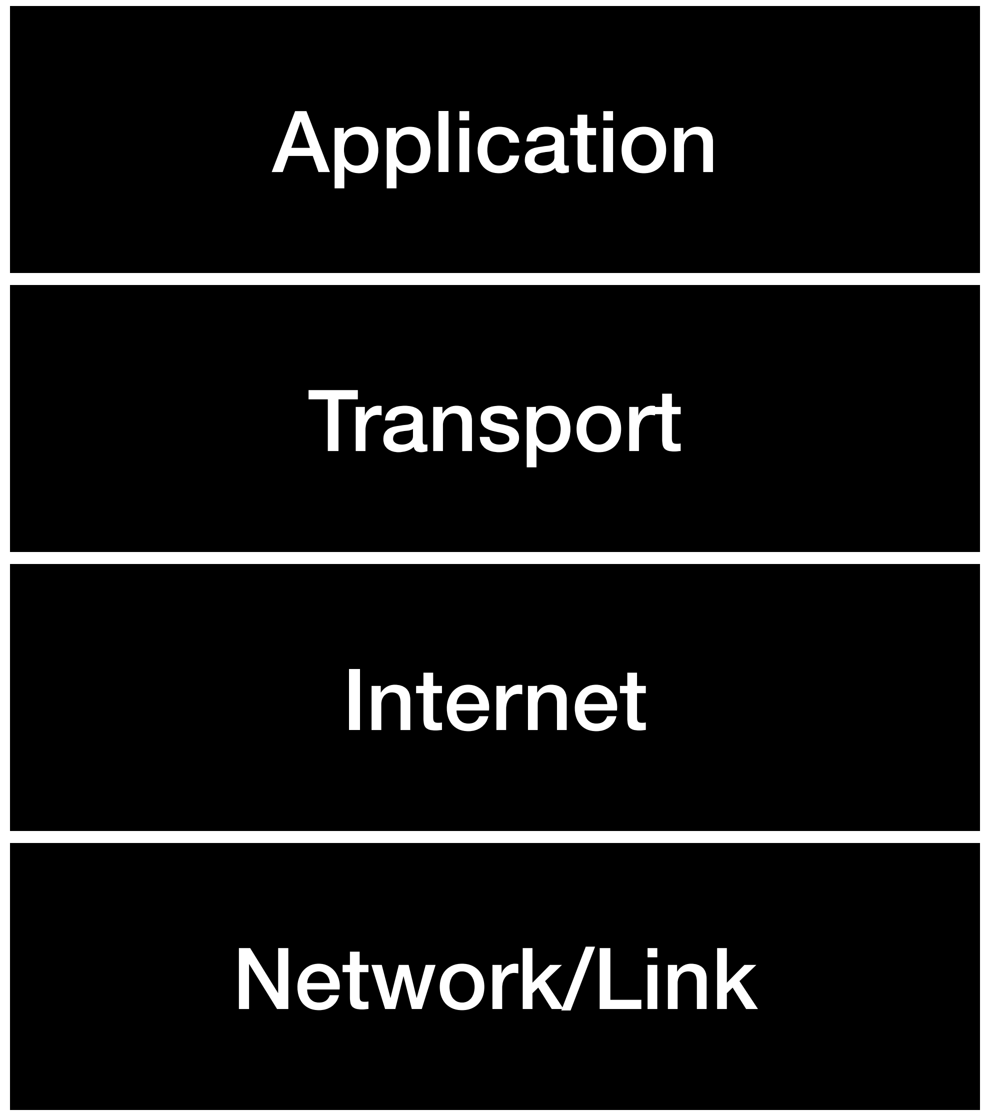
Dencapsulation

| | | | |
|---|---|---|---|
| Application | | | Application |
| Presentation | | | Presentation |
| Session | | | Session |
| Transport | TCP \| Data | Segment | Transport |
| Network | IP \| TCP \| Data | Packet | Network |
| Data Link | MAC \| IP \| TCP \| Data | Frame | Data Link |
| Physical | 10001000010000010000 | | Physical |

# 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



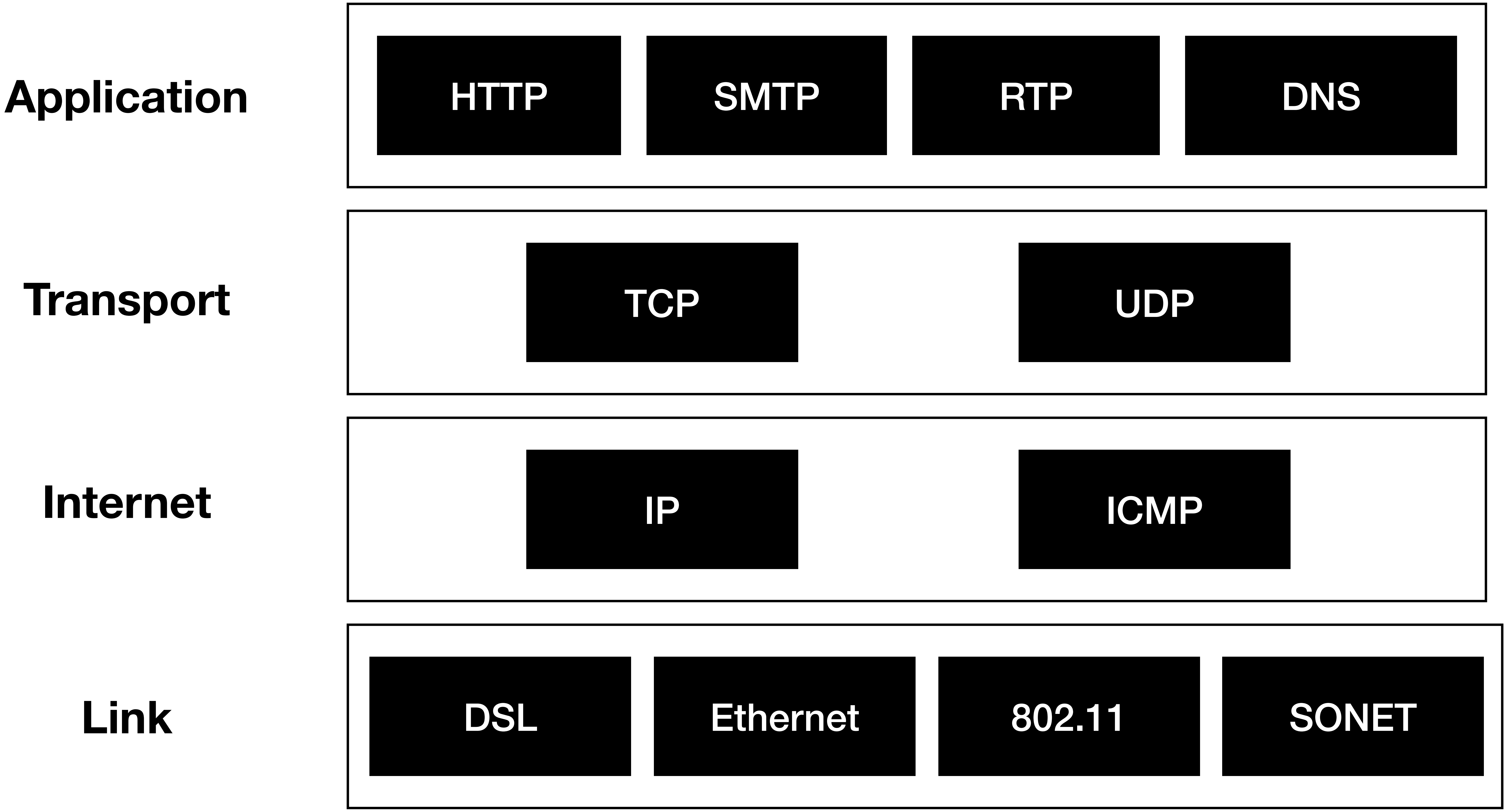
Application

Transport

Internet

Network/Link

# Network Protocol Stack



Application

HTTP SMTP RTP DNS

Transport

TCP UDP

Internet

IP ICMP

Link

DSL Ethernet 802.11 SONET

# Onto Transport Layer

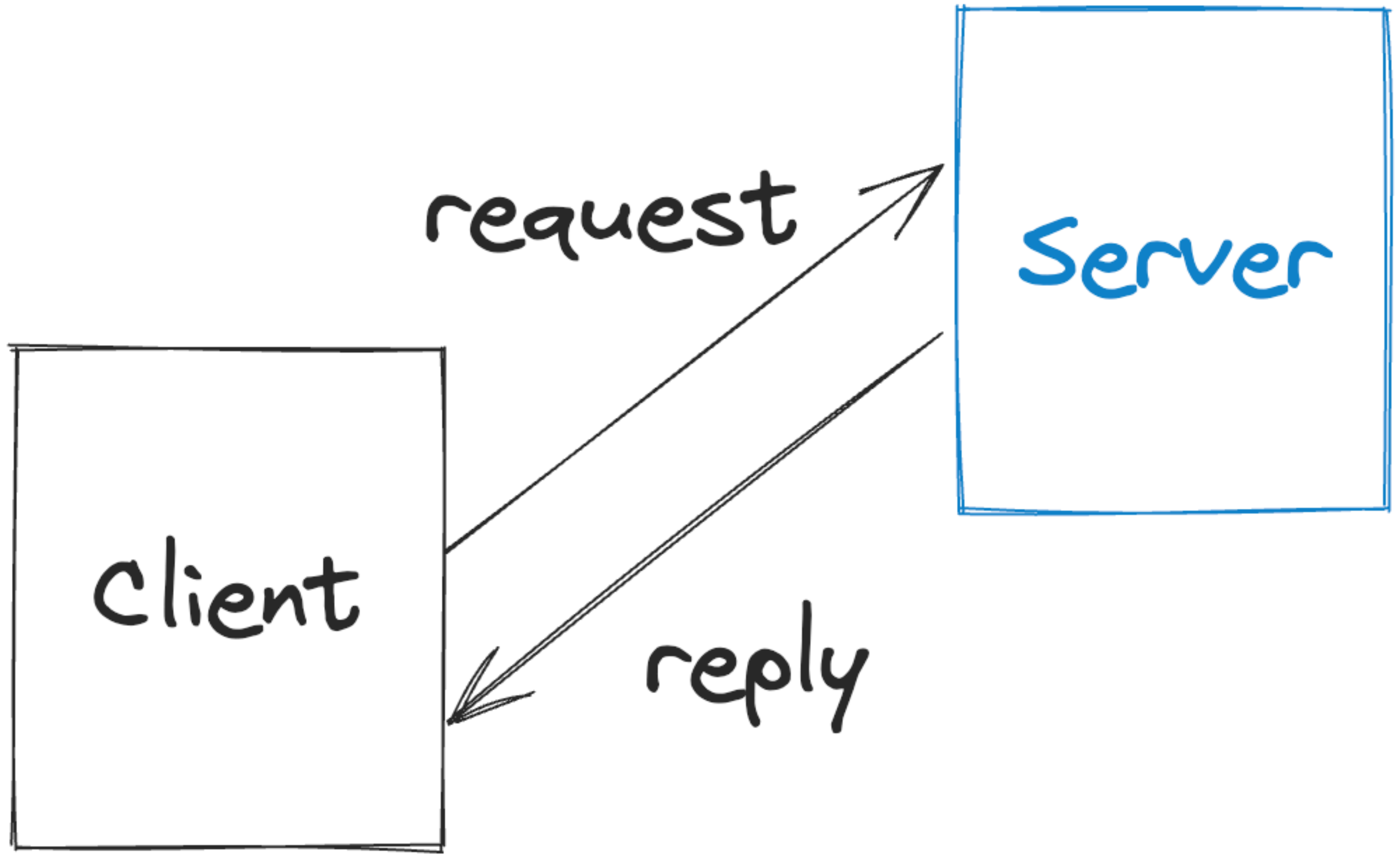
## Service to Service Delivery

- **Multiplexing** and **Demultiplexing**
- Addressing scheme: Ports
- Two strategies/protocols that allows this
  - Transmission Control Protocol (TCP) -  favours reliability
  - User Datagram Protocol (UDP) - favours efficiency

# Lets go back

## Two process wants to communicate

- Client sends a request to the server
- The server provides a reply based on the request made by client
- Eg: File transfer to get files
- Eg: web browsing, sent a url and get page
- How to transport data from process to process?



# The role of Operating System

User space
kernel space
process
Kernel
protocol stack
drivers
Client
Network
process
Kernel
protocol stack
drivers
Server
User space
kernel space

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

## Any idea on what should be some functionalities that should be made available?

**Hint:** **Think of process API**

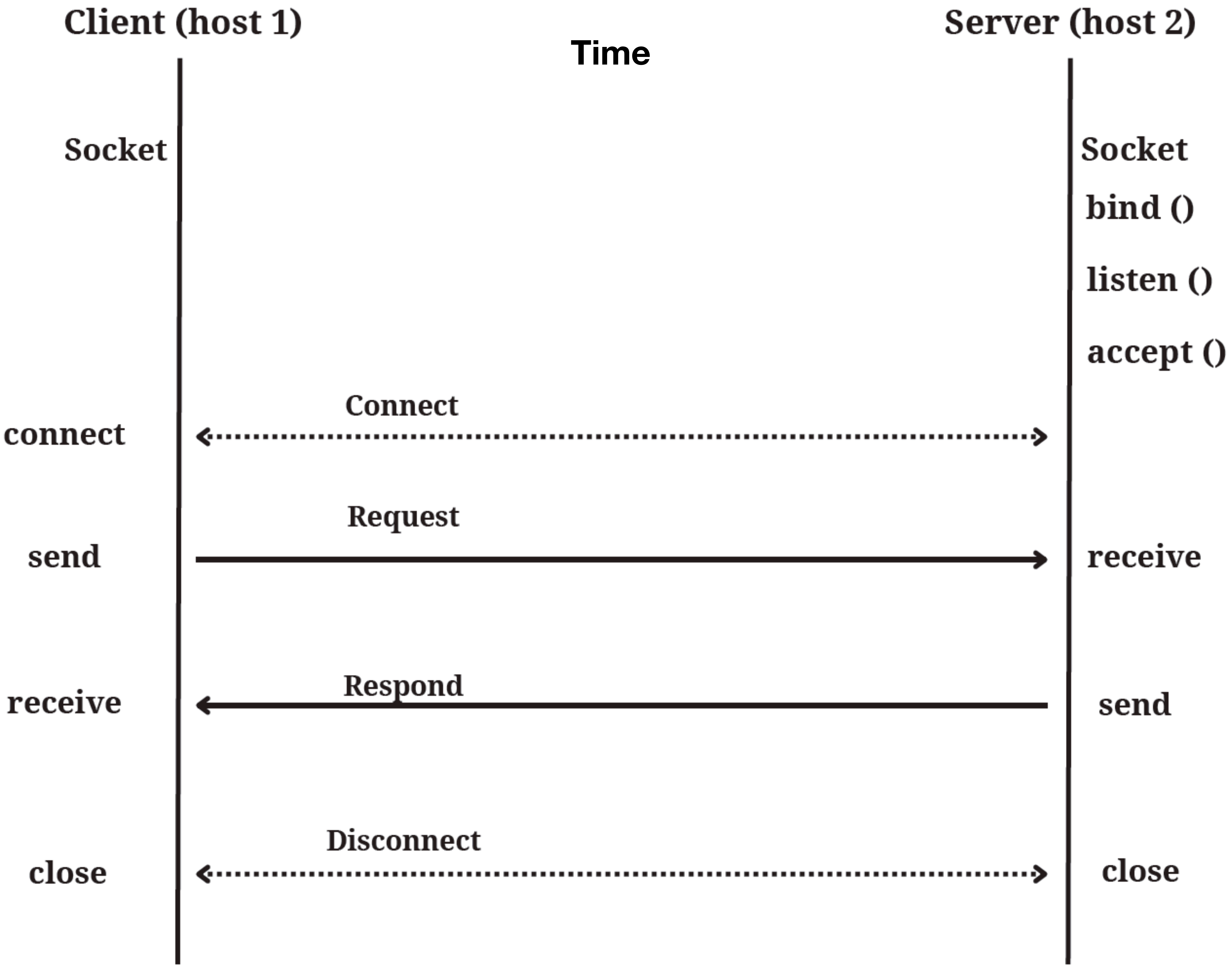
# The Socket API

- Simple abstraction to use the network
- The network service API used to write all network applications
- Part of all OS and all language [Berkley Unix, 1983]
    - Allows user space applications to interact with networking subsystem
- Two services:
    - Streams: Reliable (Connection-oriented)
    - Datagram: Unreliable (Connection-less)
- Allows applications to attach to the network at different ports

# The Socket API

| Function | Description |
|---|---|
| **socket()** | Creates a new socket of a certain type (depending on TCP or UDP) and returns file descriptor |
| **bind()** | Associates the socket with a specific IP and port |
| **listen()** | For server sockets, it allows sockets to listen for incoming connections |
| **accept()** | For server sockets, it waits for client to connect and then return a new file descriptor |
| **connect()** | For client sockets, it initiates a connection to a server. |
| **send() / receive()** | Transmit data or receive data |
| **close()** | Terminate the connection |

# Using Sockets



# More about Ports

- Application process is identified by tuple (IP address, Protocol, Port)
    - Port are 16-bit integers representing “mailboxes” that process leases
- Servers are often bind to “well-known-ports”
- Clients are assigned ephemeral ports
    - Chosen by the OS temporarily

# Some well Known Ports

| Port | Protocol | Use |
|---|---|---|
| **20, 21** | FTP | File Transfer |
| **22** | SSH | Remote login |
| **25** | SMTP | Email |
| **80** | HTTP | World wide web |
| **443** | HTTPS | Secured web |
| **543** | RTSP | Media Player Control |

# An Opportunity for a Context Switch?

- The calls of establishing socket are blocking calls
  - connect(), accept(), receive()
  - Once the call is made, OS halts the program to wait to receive some response
  - They are essentially **System calls**
  - Trap instruction is called and there is an opportunity for a context switch

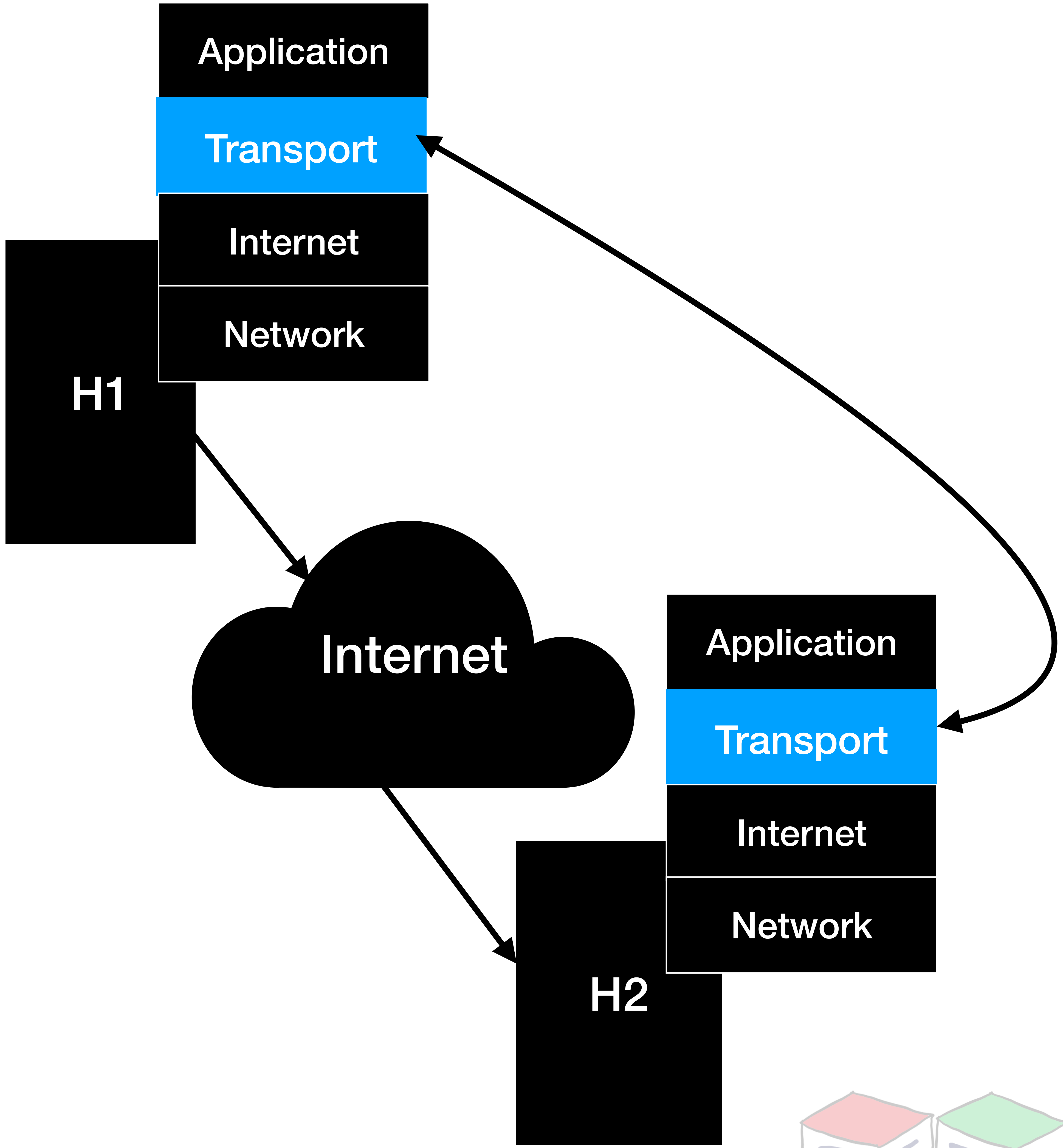
# Let us take a step back

## Types of Links

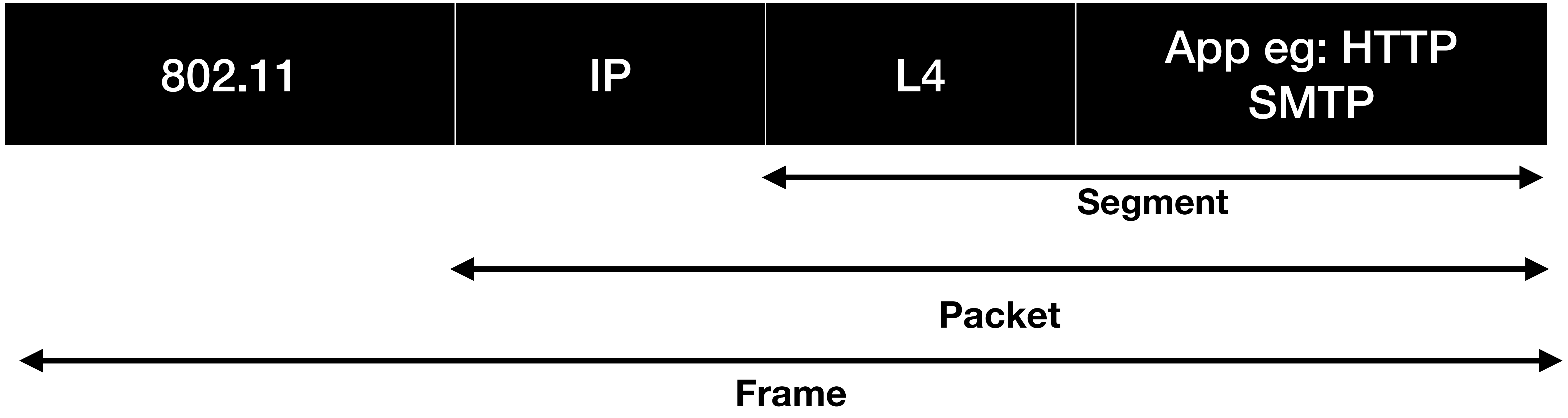
- **Full Duplex**
  - Bidirectional
  - Both sides at the same time
- **Half-duplex**
  - Bidirectional
  - Both the sides but only one direction at a time (eg: walkie talkies)
- **Simplex**
  - Unidirectional

# Transport Services and Protocols

- Provides logical communication between application processes running on different hosts
- Transport protocols actions in the end systems:
  - Sender: breaks application messages into segments, passes to network layer
  - Receiver: reassembles messages into messages, passes to application layer
- Protocols: TCP, UDP



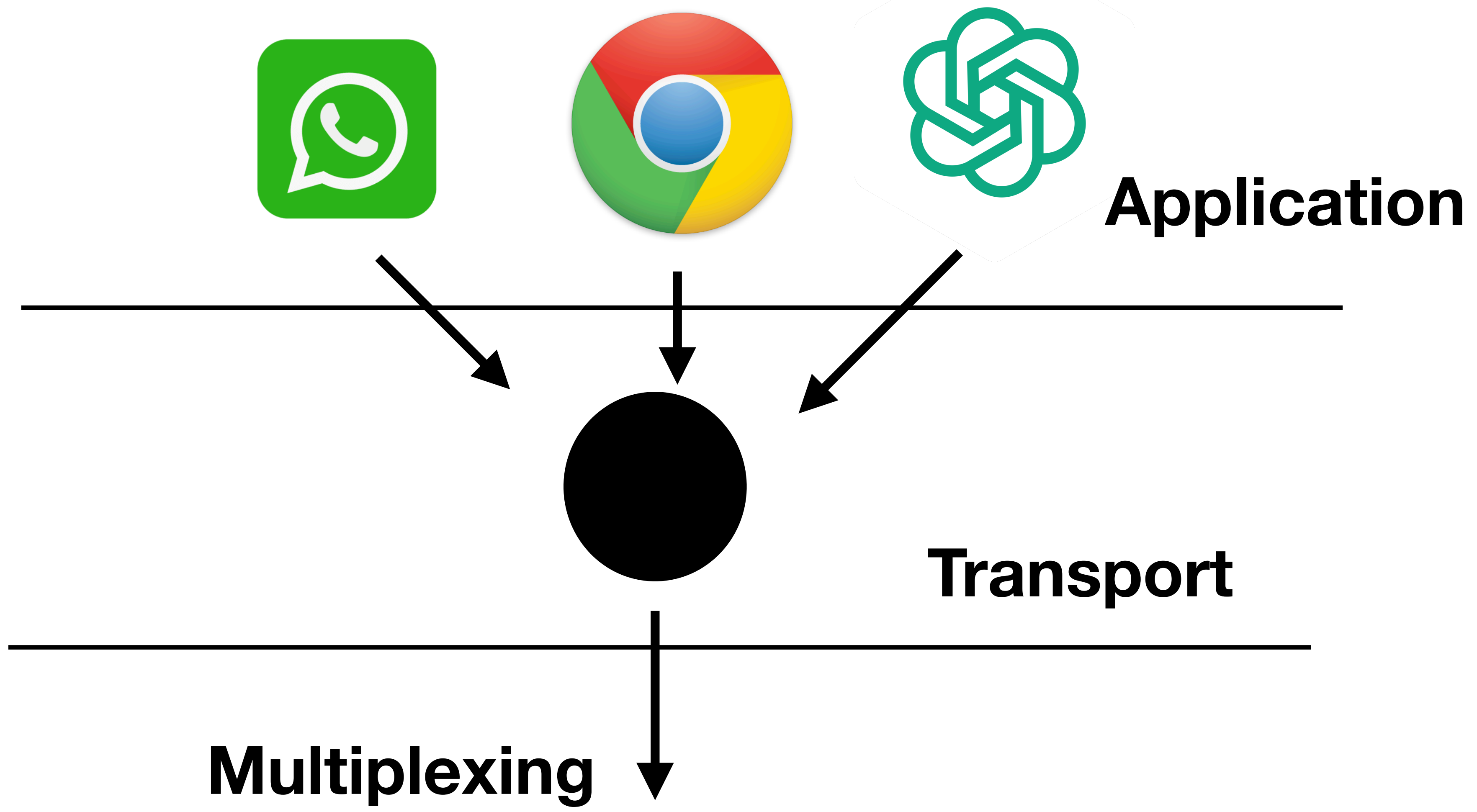
# Quick Recap



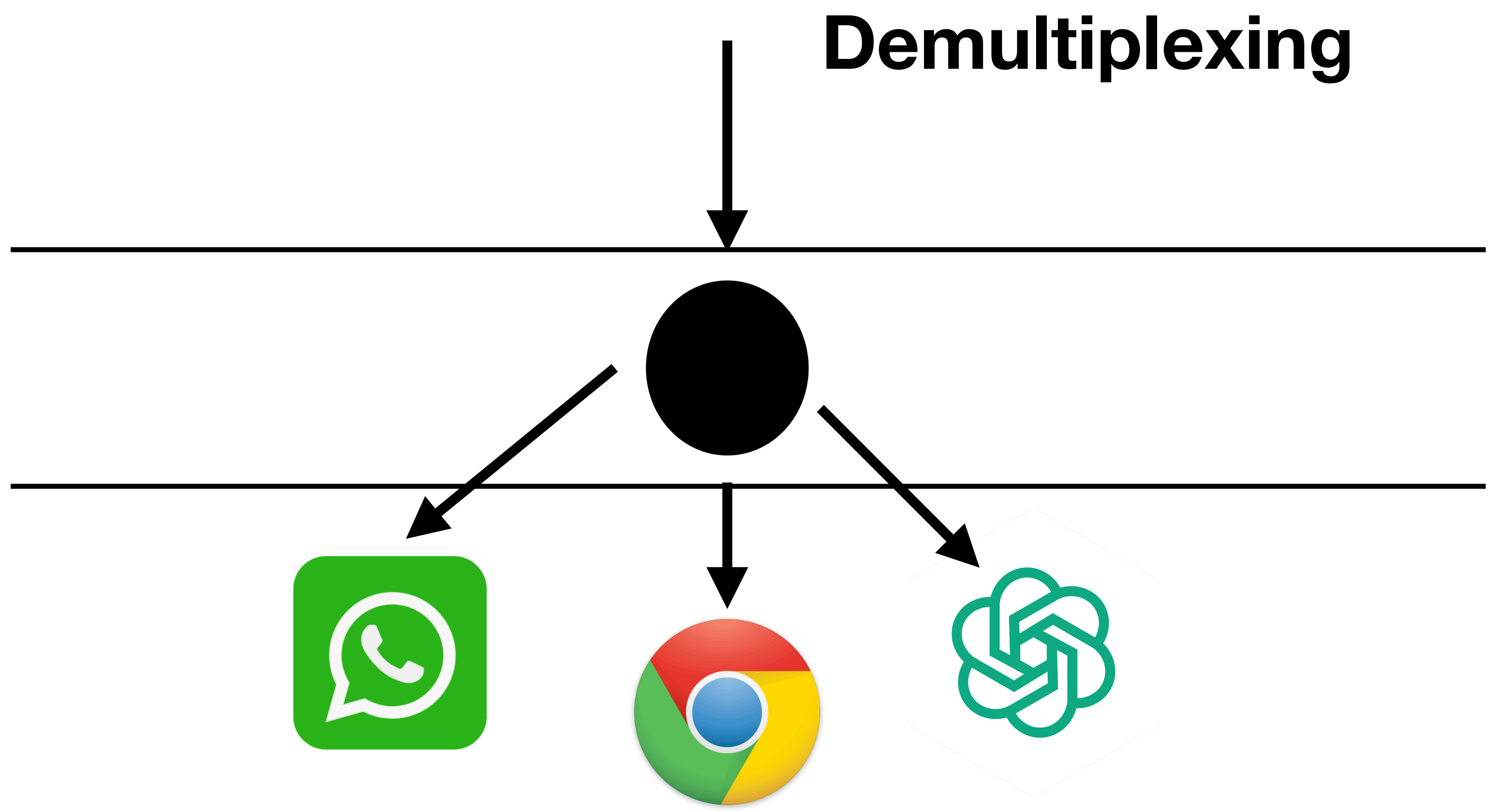
- Segments carry data across the network
- **Segments** are carried within the packets, within frames
- Each layer adds a **header** (Above L4 will be replaced by its header)

# Multiplexing and Demultiplexing

- **Multiplexing as sender**: Handle data from multiple sockets, add transport header

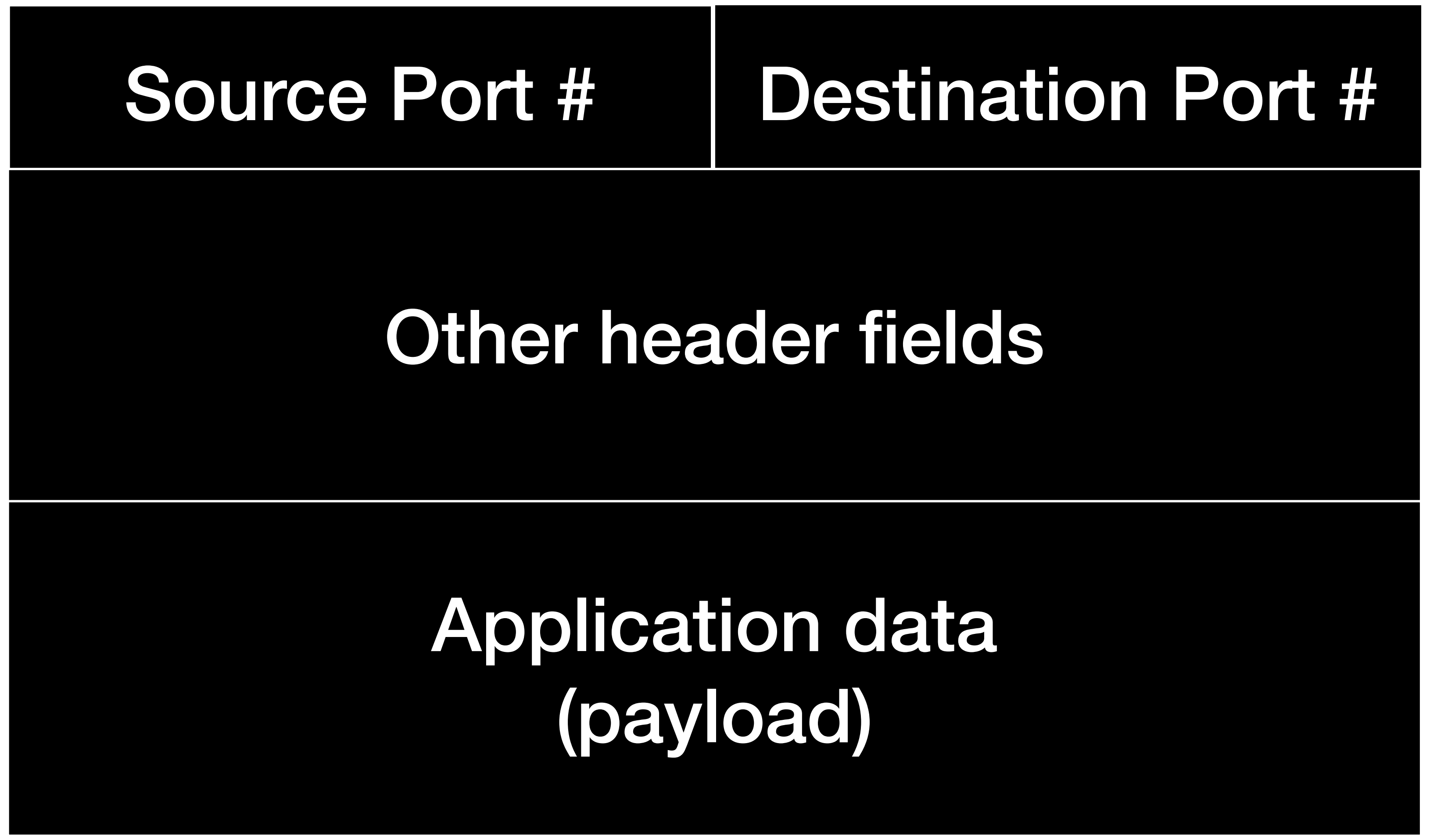


- **Demultiplexing as receiver**: Use header info to deliver received segments to correct socket



# Working of Demultiplexing

- Host receives IP datagrams
  - Each datagram has source IP address, destination IP address
  - Each datagram carries one transport layer segment
  - Each segment has source and destination port number
- IP addresses and ports are used to direct segment to appropriate socket



Source Port #

Destination Port #

Other header fields

Application data
(payload)

**TCP/UDP Segment format**

# Connection Oriented vs Connectionless

## Demultiplexing Scenarios

- **Connection oriented (TCP)**
  - TCP socket identified by 4 tuple
    - Source IP, destination IP, source port and destination port
  - Receiver uses all 4 to direct segment to appropriate socket
  - Server may support many TCP sockets
    - **Each socket has it own client**

- **Connectionless (UDP)**
  - UDP socket identified by 2 tuple
    - Destination IP and port
  - Receiver uses the port to redirect to the corresponding socket
  - UDP segments with same destination port but different IP or source port
    - **Redirected to same socket**

# TCP vs UDP

| TCP | UDP |
|---|---|
| Connection Oriented | Not Connection Oriented |
| Reliability (order is maintained and retransmission) | Unreliable |
| 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, .. |

# Connection Oriented and Reliability

- **Connection Oriented**
    - In TCP, the connection is first established before the data is transmitted
    - In UDP there is no notion of connection starting and ending (use timeout)
- **Reliability**
    - Confirmation of data delivery (Acknowledgement is there) in TCP
        - Order is preserved or maintained
        - Error can be handled  (Awareness). TCP can handle it.
    - In UDP there is no confirmation, the client trusts that there is someone to receive the data (Fire and Forget)
        - No error awareness (at L4). Protocol does not handle it

# Flow Control and Overhead

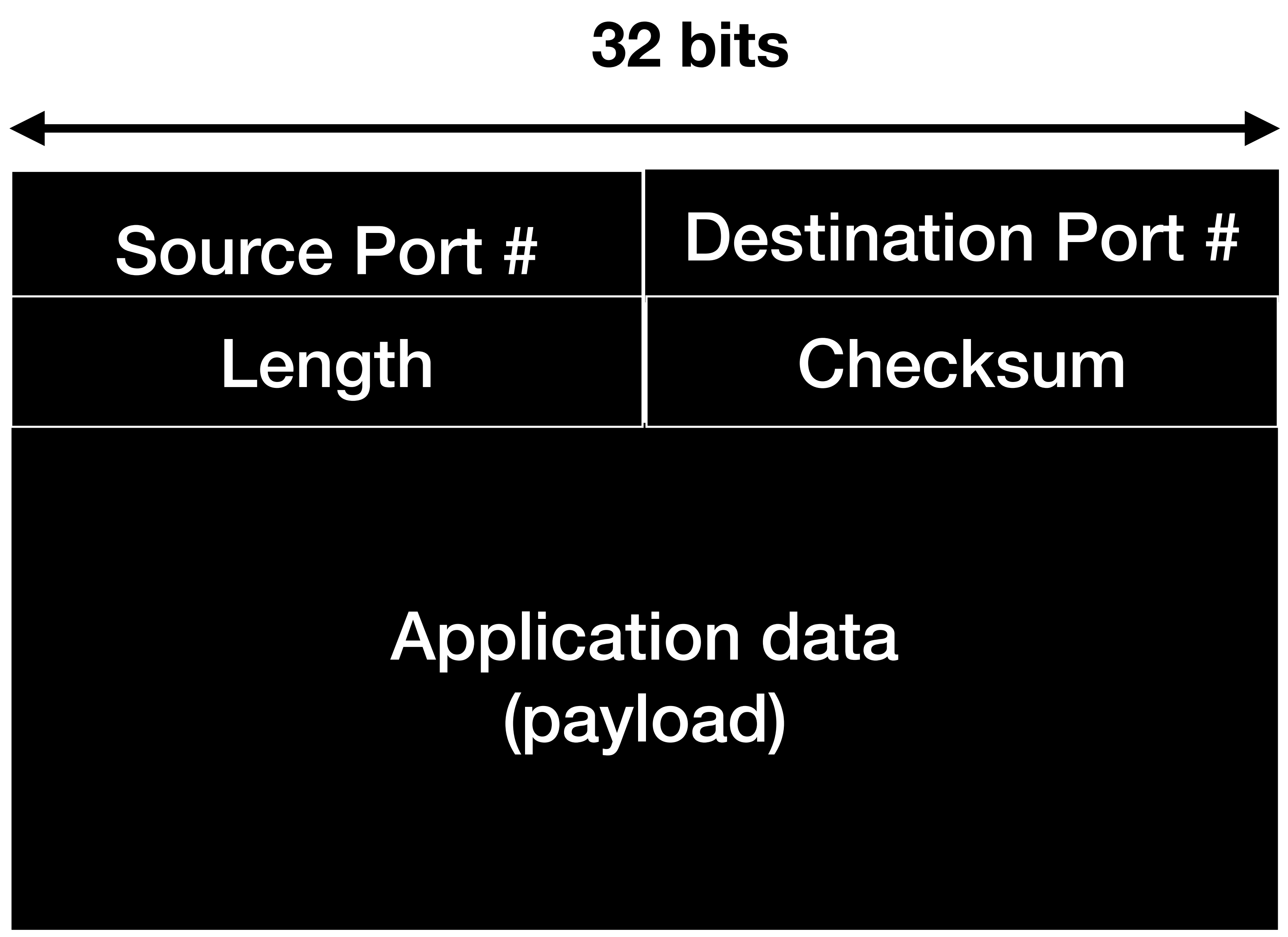
- Flow Control
  - TCP can adjust the transmission rate to use maximum available bandwidth
  - Check how much the receiver can receive and adjust accordingly

- Overhead
  - TCP Adds a larger header to the data ~ 20 bytes or even more
  - TCP has more features that does not exist in UDP
  - In UDP the header length is ~ 8 bytes

L4 | Data

# UDP Segment Header

- Length: In bytes of the UDP segment including the header
- Checksum: For error detection (16 bit value which represents the sum of UDP header, payload and Pseudo header from IP layer)
  - Supports Error detection
  - Makes use of 1's compliment arithmetic to find the sum


32 bits

Source Port # | Destination Port #

Length | Checksum

Application data
(payload)

UDP Segment Format

# Checksum Process

- **Sender**
  - All contents of the header including IP addresses are treated as sequence of 16 bit integers
  - Checksum: addition (one’s complement) of segment content
- **Receiver**
  - Compute checksum of received content
  - Check if received and header checksum are equal - No error
  - Else, Error detected

**Example**

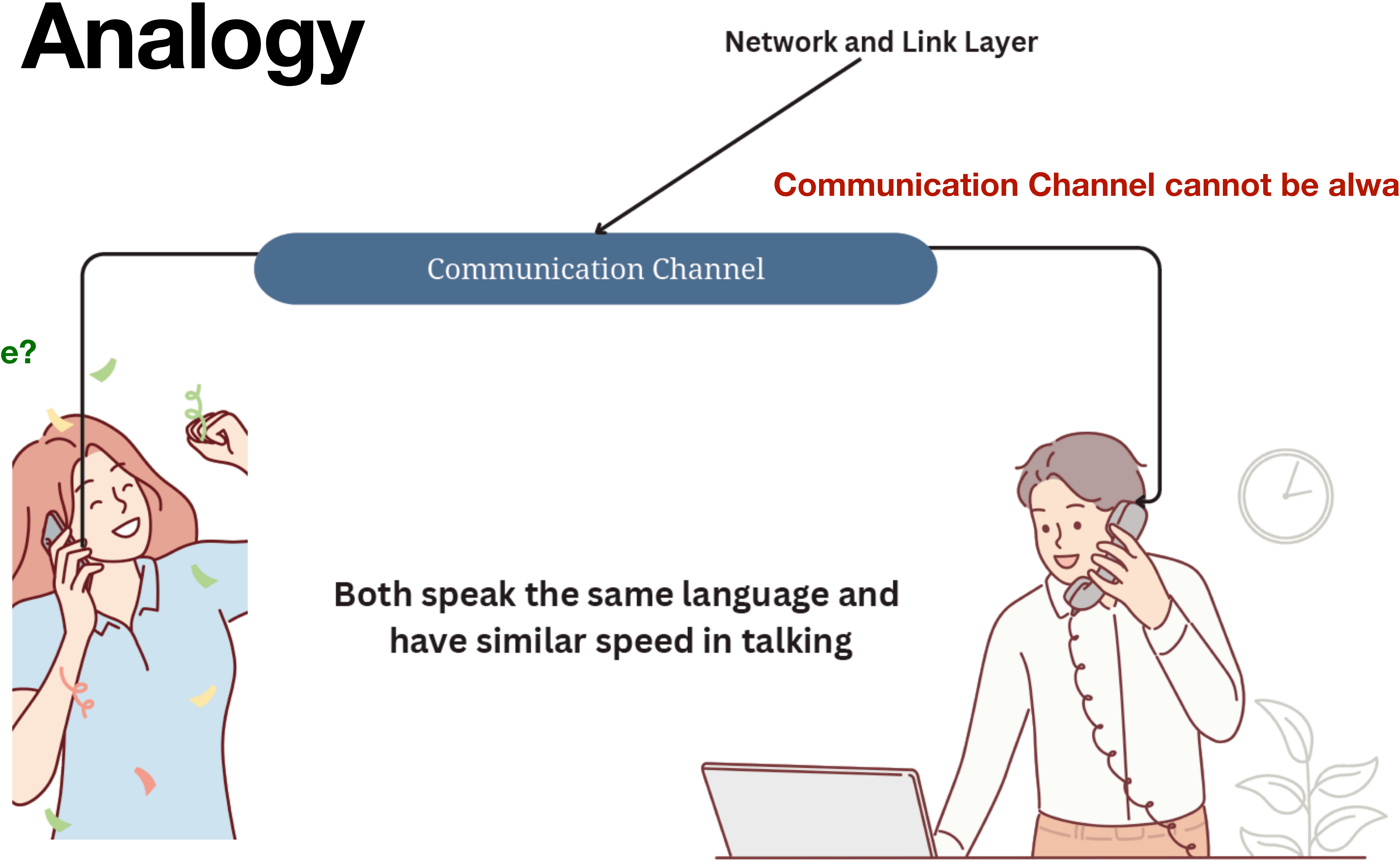
| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| | | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| **Add it back** | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| **Sum** | | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| **Checksum** | | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

## TCP is the most used protocol on the internet. How does TCP work?

## What all you need to provide some features that TCP provides?

# A Small Analogy

What can we do from the protocol perspective?



Network and Link Layer

Communication Channel cannot be always reliable!!

Communication Channel

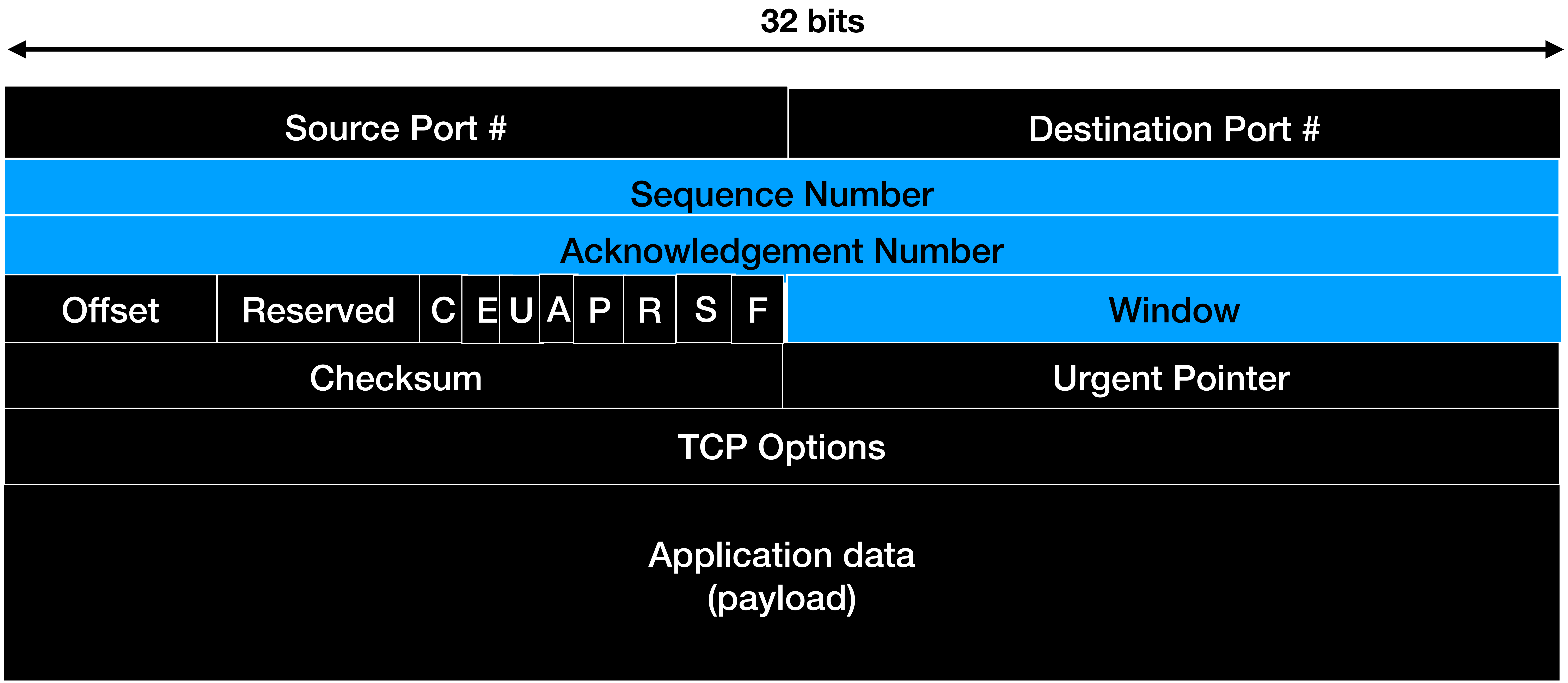
Both speak the same language and have similar speed in talking

Person 1 Talking
(Process in a host A)

Person 2 Talking
(Process in a host B)

**Do we foresee some challenges?**

# Lets go into TCP - Header



32 bits

Source Port # | Destination Port #

Sequence Number

Acknowledgement Number

Offset | Reserved | C | E | U | A | P | R | S | F | Window

Checksum | Urgent Pointer

TCP Options

Application data
(payload)

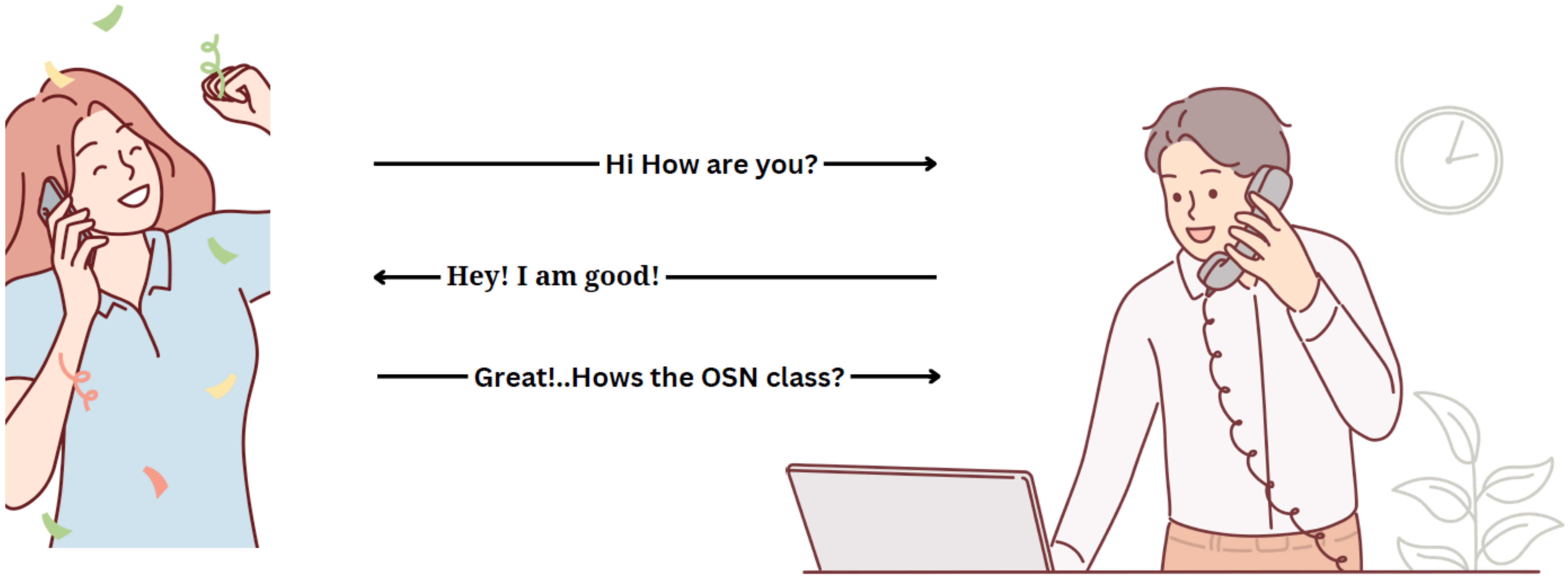
TCP Segment Header

# Header Elements

- **Sequence number:** Tracks bytes that are sent (# of bytes that are sent)
- **Acknowledgement number:** Tracks bytes that are received  (Sequence number of the next expected byte)
- **Window/Receive Window:** Number of bytes the receiver can accept (Flow control)
- **A:** Acknowledgement bit
- **R, S, F:** Connection management
- **C, E:** Congestion notification
- **Offset:** Length of the TCP header

# What do ACK and Sequence Number do?

## Reliability!!



Hi How are you? →

← Hey! I am good!

Great!..Hows the OSN class? →

Person 1 Talking
(Process in a host A)

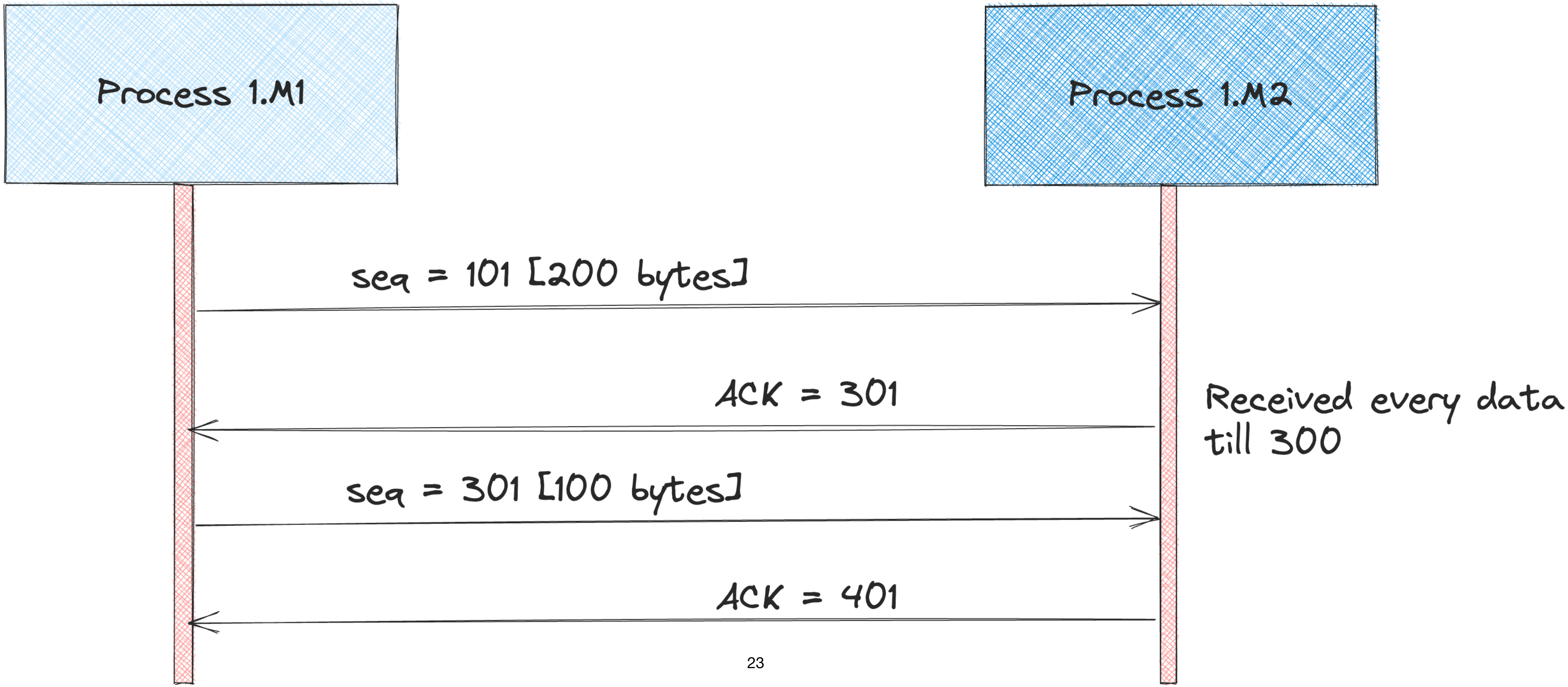
Person 2 Talking
(Process in a host B)

Each word the Person 1 says reaches person 2 in the same order

Whatever Person 1 Says, Person 2 acknowledges before adding new points to the conversation

# What do ACK and Sequence Number do?

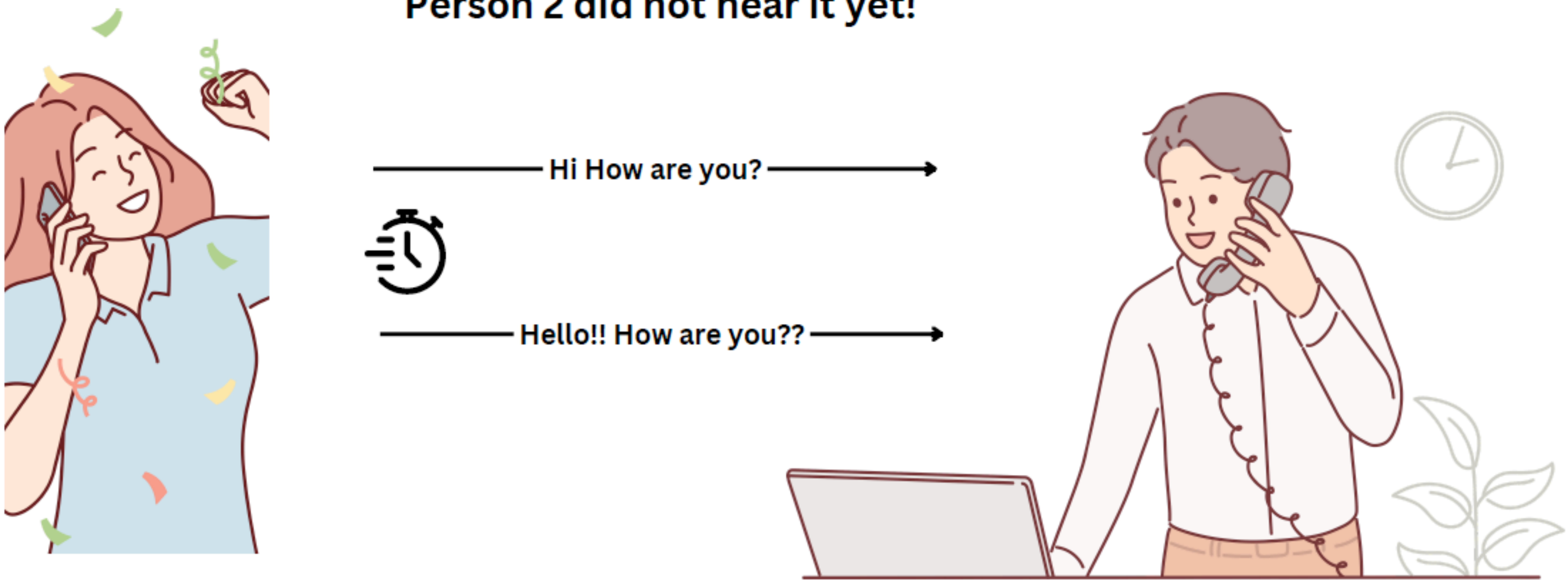
## Reliability!!



# How to handle if data is lost?

## Can we retransmit?

Person 1 is trying to Speak
Person 2 did not hear it yet!



Hi How are you?

Hello!! How are you??

Person 1 Talking
(Process in a host A)

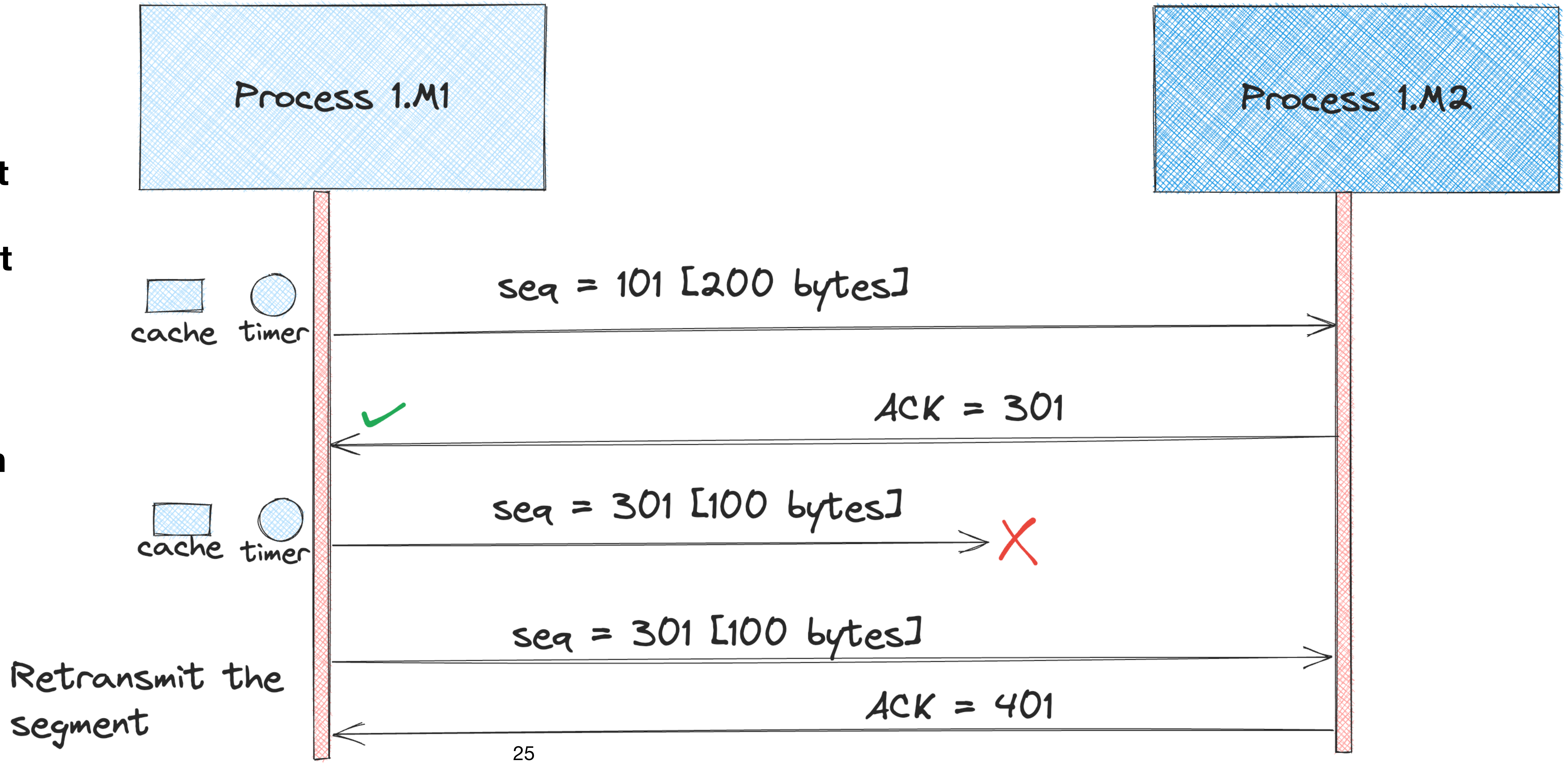
Person 2 Talking
(Process in a host B)

# How to handle if data is lost?

## Retransmission timeout also known as Round Trip Timeout (RTT)

TCP caches every data sent
in a buffer (OS supports)
Until retransmission timeout

What if ACK does not reach
Back Process 1.M1?



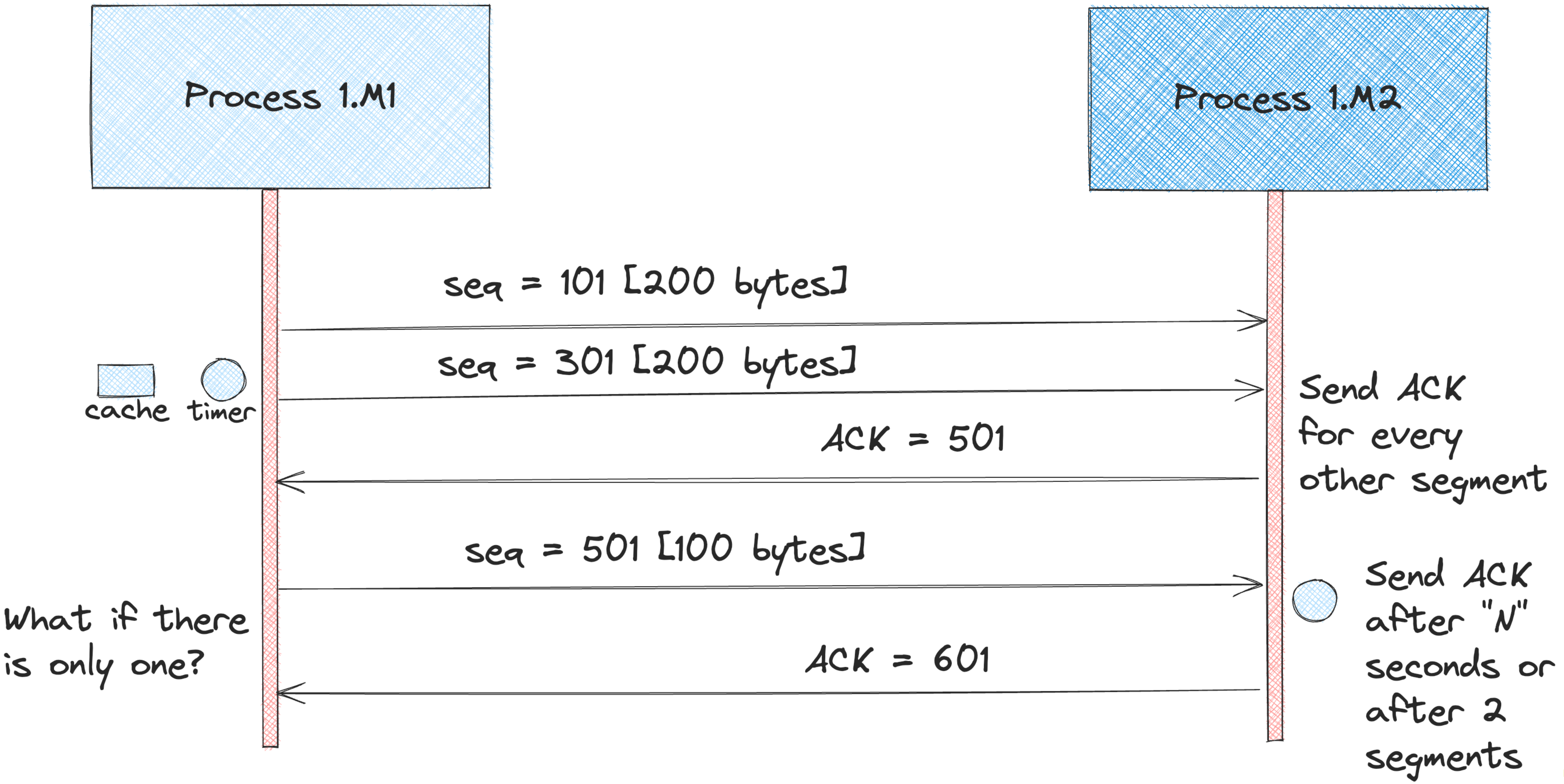
# How to calculate RTT?

```
EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT
  DevRTT = (1-β)*DevRTT + β*|SampleRTT-EstimatedRTT|
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

- **SampleRTT:** Time measured from segment transmission until ACK receipt
- **EstimatedRTT:** Estimated weighted moving average (EWMA) $\alpha = 0.25$
- **DevRTT:** EWMA of sampleRTT deviation from EstimatedRTT $\beta = 0.75$
- **TimeoutInterval:** Estimated Time plus some kind of safety margin

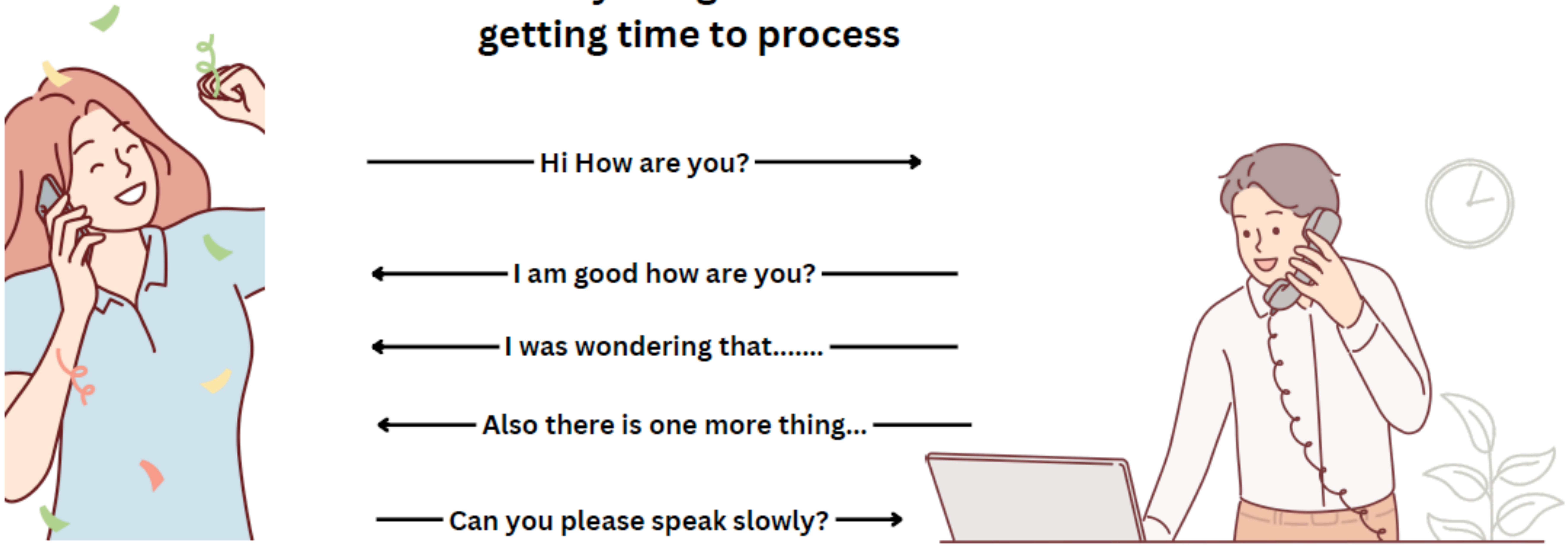
# Do We need to Send ACK for each segment?

## Use delayed acknowledgements



# What if the speed is high?

Person 2 is speaking
fast and many things..Person 1 is not
getting time to process



Hi How are you? →

← I am good how are you?

← I was wondering that.......

← Also there is one more thing...

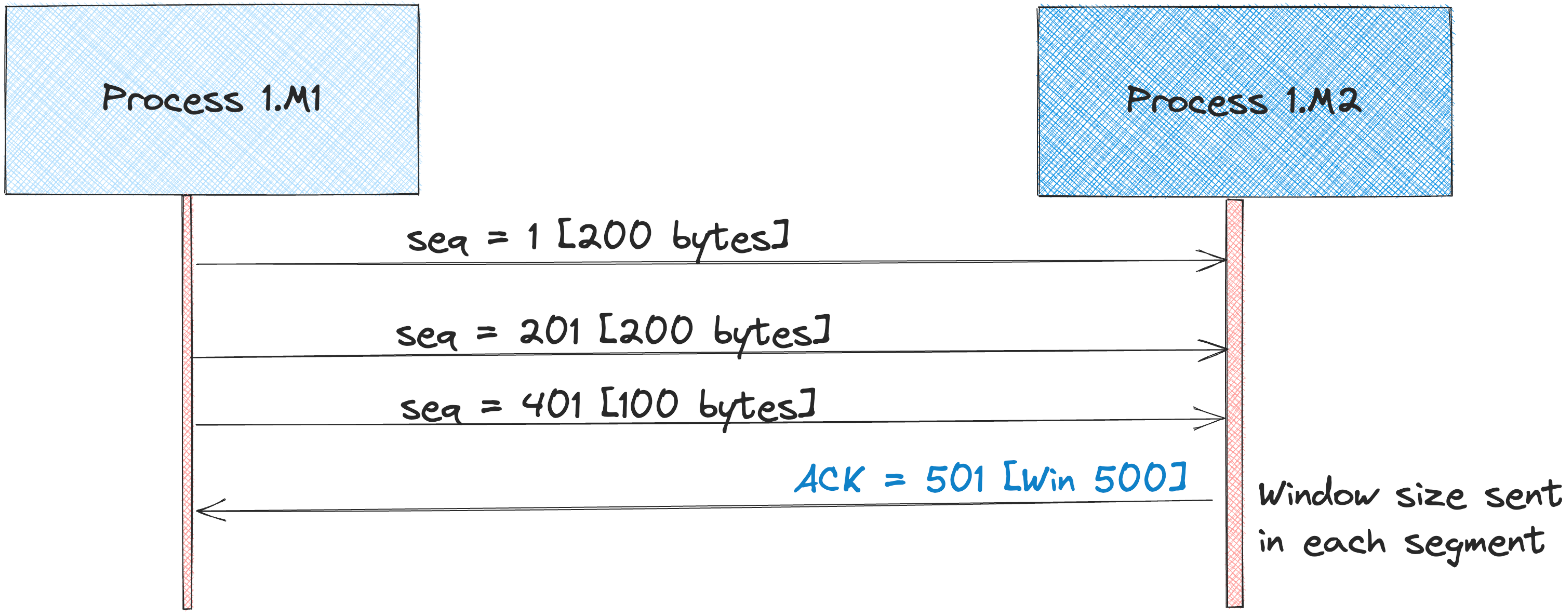
Can you please speak slowly? →

Person 1 Talking
(Process in a host A)

Person 2 Talking
(Process in a host B)

# Sending too much data is also problem

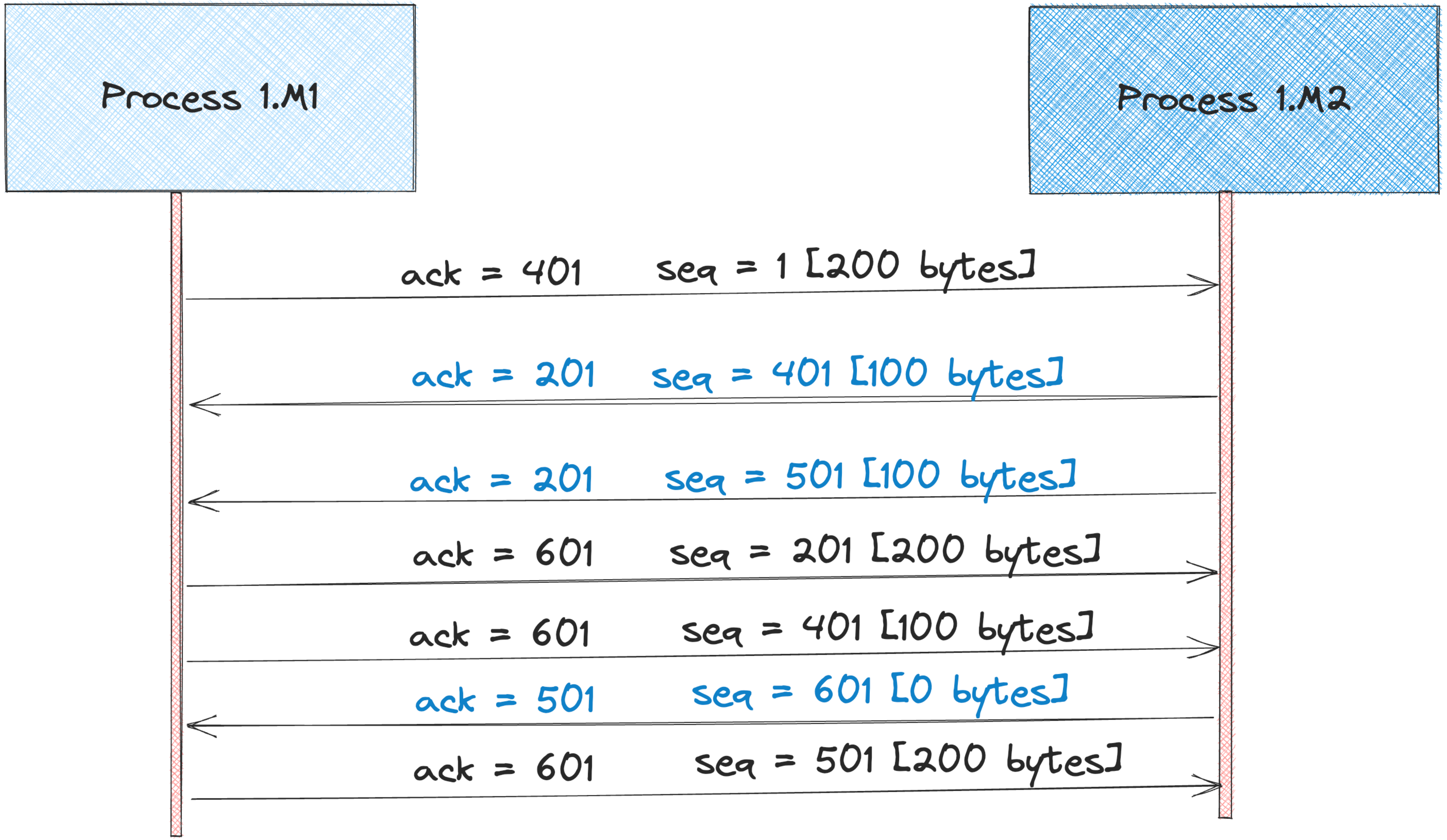
## Window Size - Flow Control



- Dynamic update of Window size will enable flow control
- What if Process 1.M2 sends a windows size of 0?

# TCP is bidirectional

## Both Senders can send data


Process 1.M1
Process 1.M2
ack = 401 seq = 1 [200 bytes]
ack = 201 seq = 401 [100 bytes]
ack = 201 seq = 501 [100 bytes]
ack = 601 seq = 201 [200 bytes]
ack = 601 seq = 401 [100 bytes]
ack = 501 seq = 601 [0 bytes]
ack = 601 seq = 501 [200 bytes]

# How to choose sequential numbers?

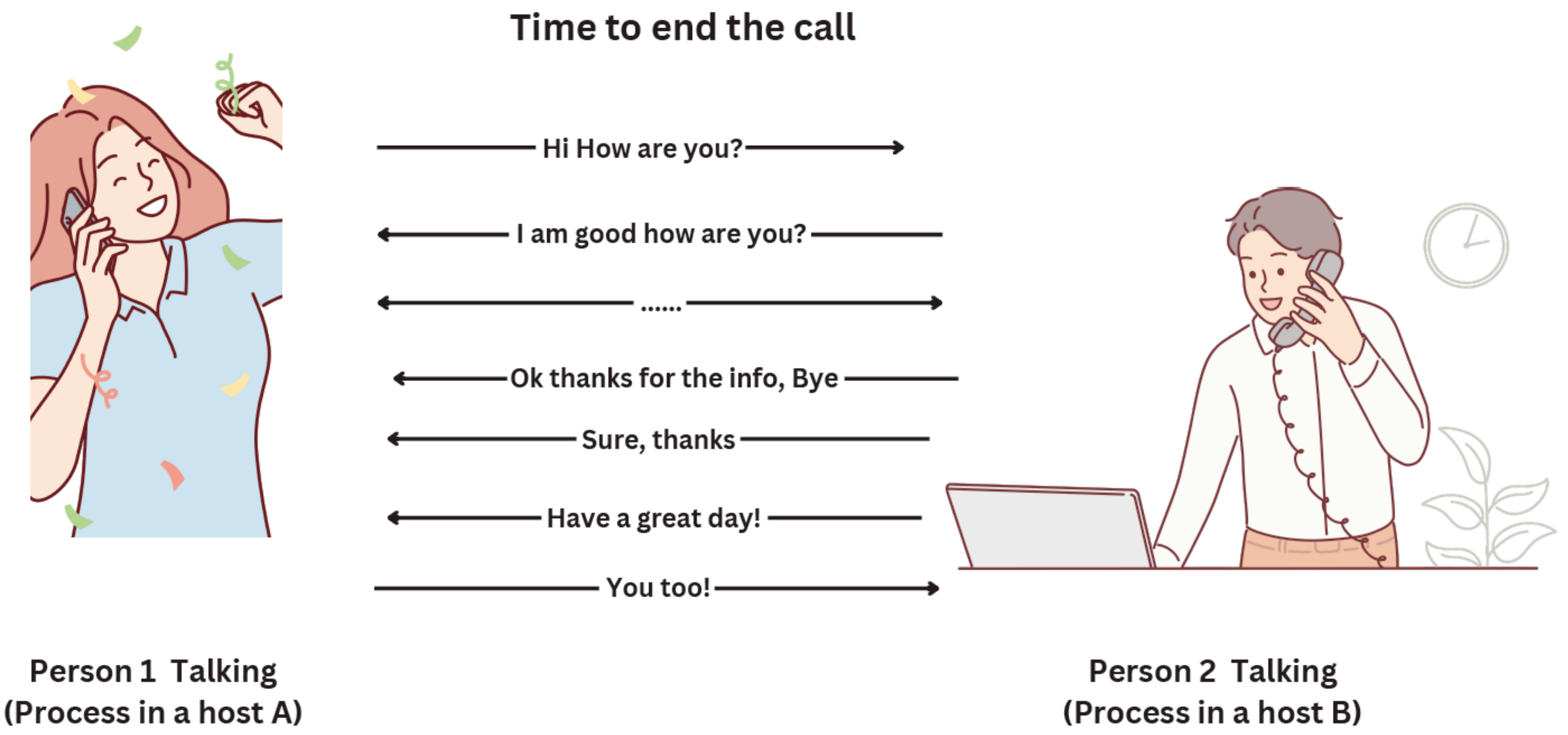
- Initial sequence numbers are randomly chosen by the senders
- Each can select a sequence number during the connection establishment
- Connection establishment in TCP happens through 3-way handshake
- The 3-way handshake consist of 4 events:
  - Process 1.M1 sends a connection request with SYN bit set and sequence number of X
  - Process 1.M2 acknowledges the connection request and sends back an ACK with X+1
  - Process 1.M2 also sends a request with the SYN bit set and sequence number [Y]
  - Process 1. M1 acknowledges the receipt by sending ACK [Y+1]

# Three Way Handshake

## Establishing Connection

Process 1.M1
Process 1.M2
set syn bit = 1
ack bit =0
seq = 100 [0 bytes] aCK = 0 [WIN = 100]
aCK = 101 seq = 300 [0 bytes] [WIN = 400]
set syn bit = 1
ack bit =1
set syn bit = 1
ack bit =0
ack = 301 seq = 101 [0 bytes] [WIN = 500]

# Closing Connection



Time to end the call

Hi How are you?

I am good how are you?

......

Ok thanks for the info, Bye

Sure, thanks

Have a great day!

You too!

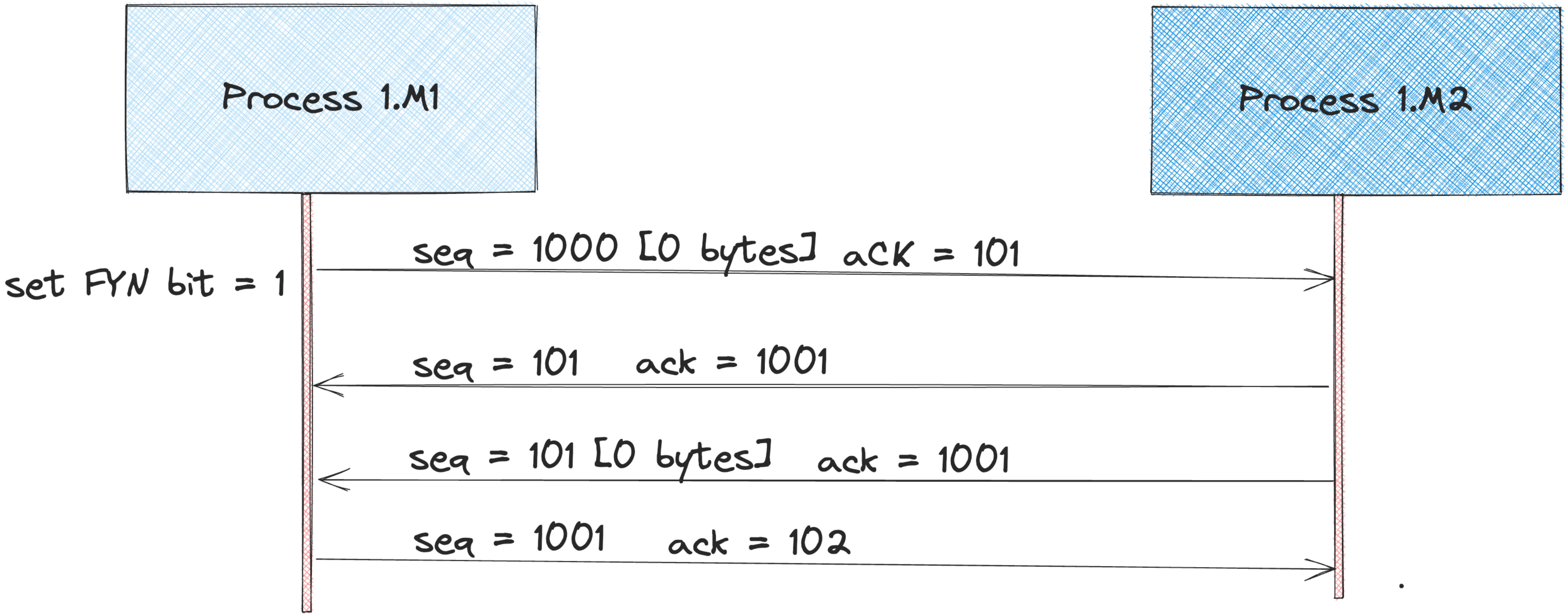
Person 1 Talking
(Process in a host A)

Person 2 Talking
(Process in a host B)

- TCP has two ways to close connection: **FIN** and **RST** flags

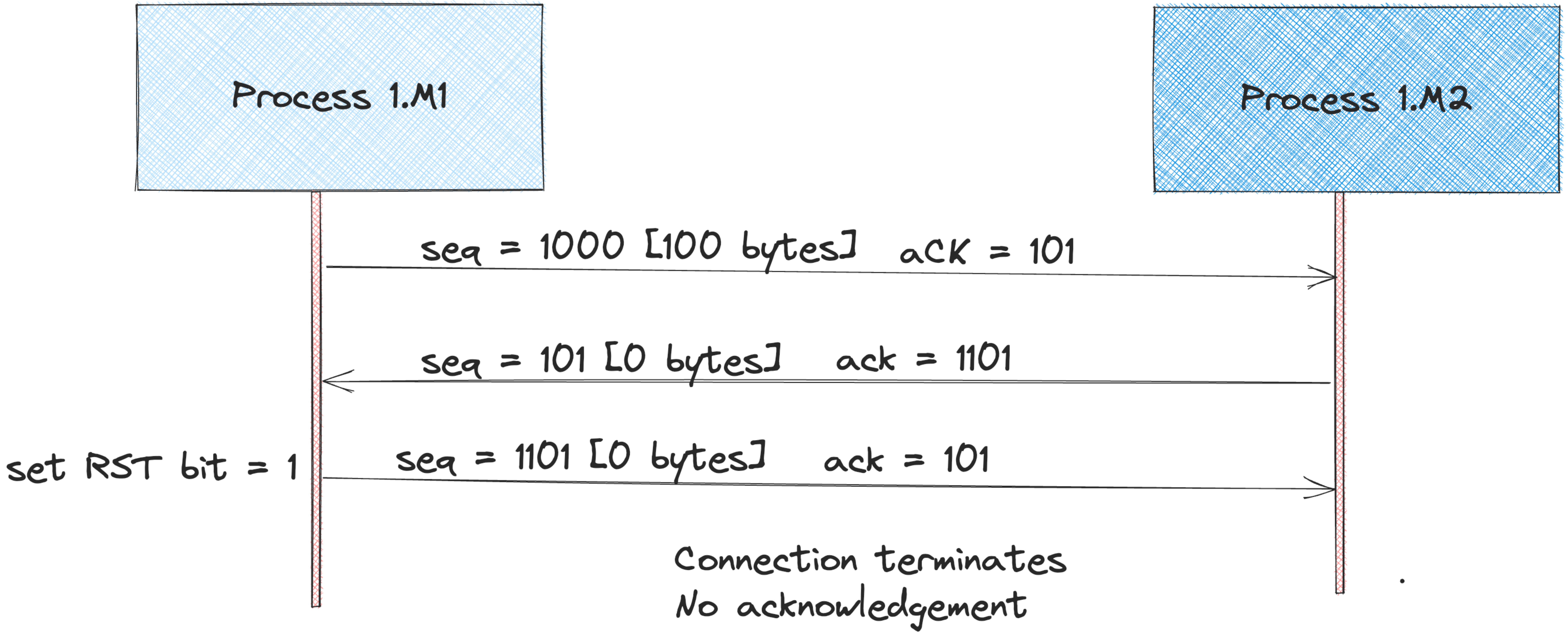
# Using FYN bit

## Graceful termination



# Using RST Flags

## Ungraceful closing


Process 1.M1
Process 1.M2
seq = 1000 [100 bytes] aCK = 101
seq = 101 [0 bytes] ack = 1101
set RST bit = 1
seq = 1101 [0 bytes] ack = 101
Connection terminates
No acknowledgement

# But we need Memory!

**How does OS handle the memory requirements of all these?**

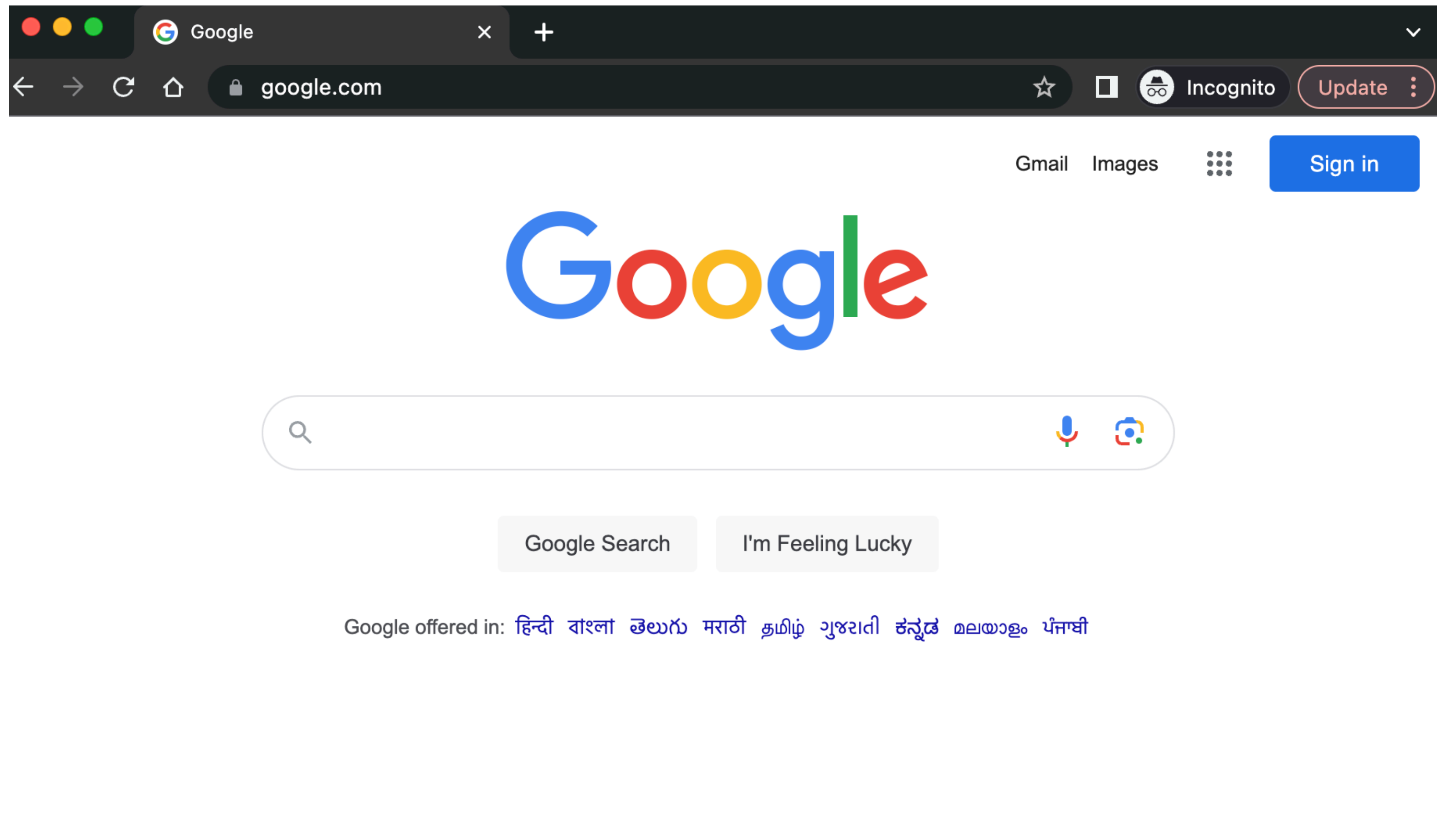
**Where is the process stored? What about network buffer?**

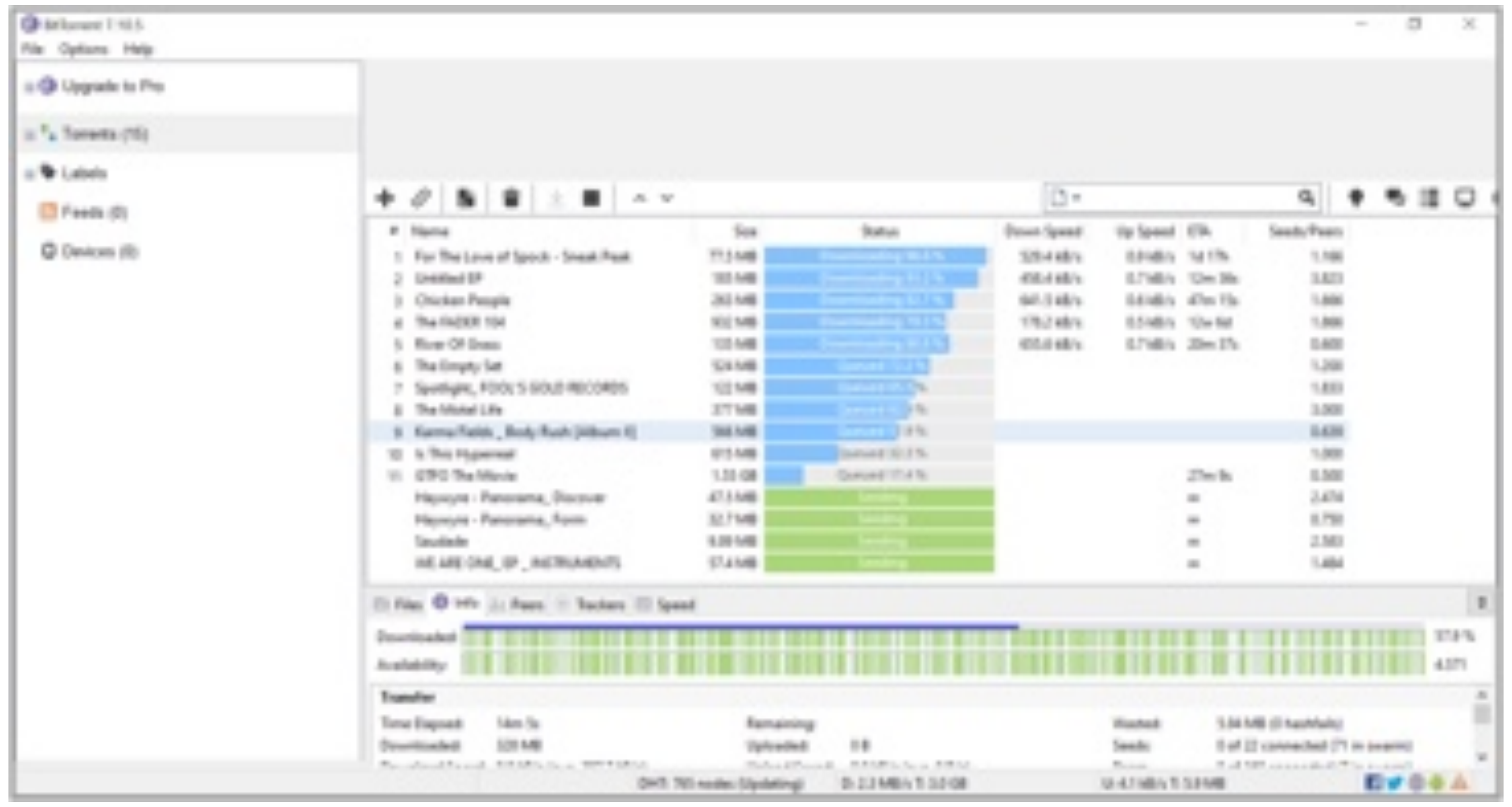
# What are the different Network Applications that you come across?

- Streaming stored video (eg: Netflix, Youtube)
- VOIP apps (eg: Skype)
- Web browser (eg: Chrome, Firefox)
- Social Networking (eg: Facebook)
- Instant Messaging (eg: Whatsapp)
- P2P File Sharing (eg: Bit Torrent)
- Real-time Video Conferencing (eg: Zoom)
- ….

# Many Processes run on the OS

## Rather applications!





Browser - Type any URL and you get the page - How?

The process on the other side needs to provide the page

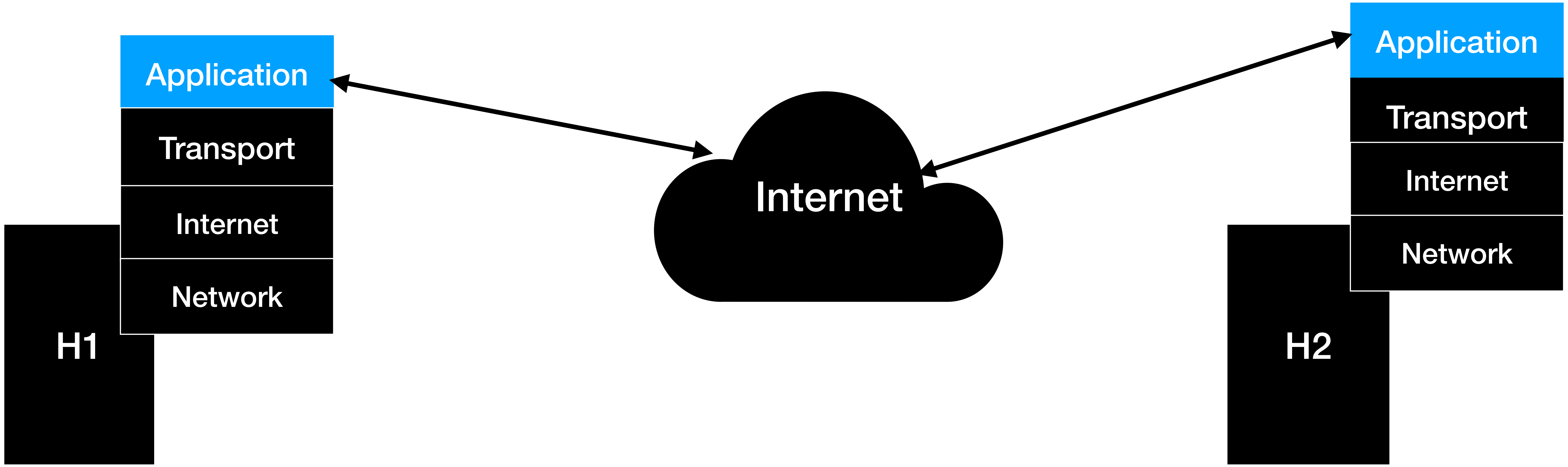
Large videos may have to be served to different users

There are also programs like torrent -

How does that work?

# Is transport layer enough?

- Process communicates over the network
- The real communication happens through the application layer



# Building a Network Application

- **Write programs:**
  - That can run on different systems
  - Communicate over the network
  - Eg: Web server communicates with browser (Apache)
- **Core-network device software:**
  - Network devices do not run user applications

APACHE
HTTP SERVER PROJECT

httpd://apache.org

# Network Application Architectures

- From a developer perspective, network architecture is fixed
  - Application architecture is something that can be controlled
- Two main types are available: Client-server and P2P
- **Client-Server**
  - Host that is always on, serving the clients - Server
  - Host that requests for services - Clients
  - Eg: Web client and Web server

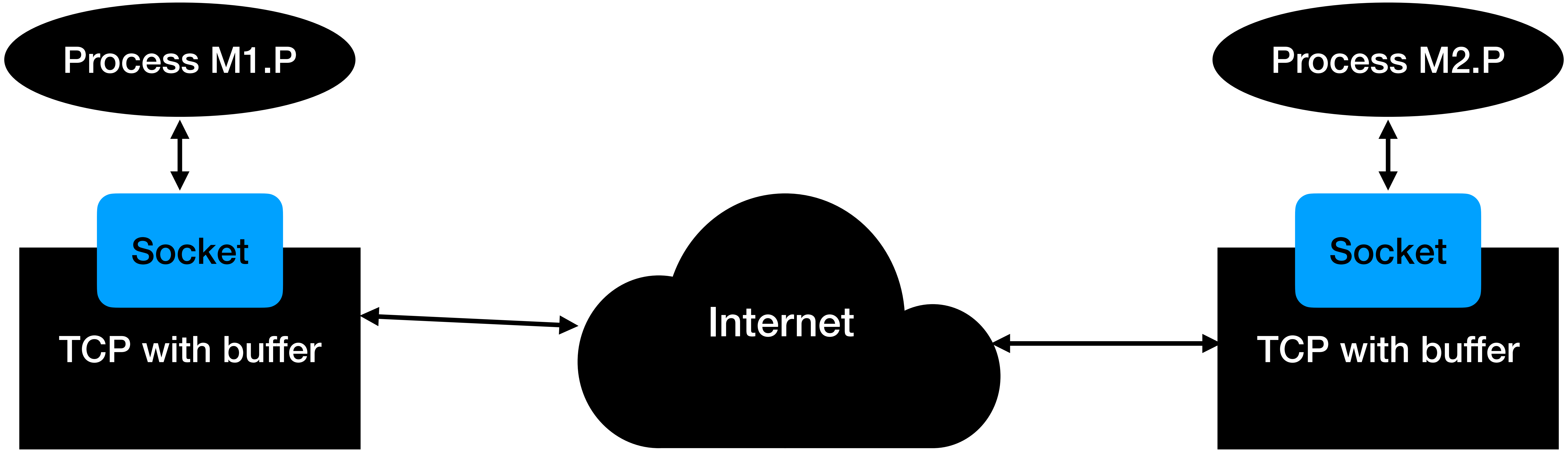
# Network Application Architecture

- Server has mostly a fixed IP address (or rather domain)
- Clients can always connect by sending packet to server IP address
  - Eg: Web browser, FTP, e-mail
- Often a single server may not be enough - leverage data centres
- Data centres have hundreds or thousands of servers that must be processed and maintained - **Energy!!**
  - **Spare a thought for the carbon foot print - Can we do something?**

# Network Application Architecture

- **Peer-to-Peer (P2P) Architecture**
  - There is minimal or no reliance on dedicated servers (No always on Servers)
  - Peers communicate among each other
  - Peer technically acts as a client and a server
  - Not owned by any service provider and does not pass through dedicated server
  - Advantage: Self-scalability, cost effective
  - Challenges: Security, Performance, reliability, etc.
  - Eg: Bit Torrent, Skype

# Lets take a step back



- Process sends to and receives messages from network using socket interface
- The developer has more control on the application side than socket
- Application just sends message with IP address and port - Rest other layers

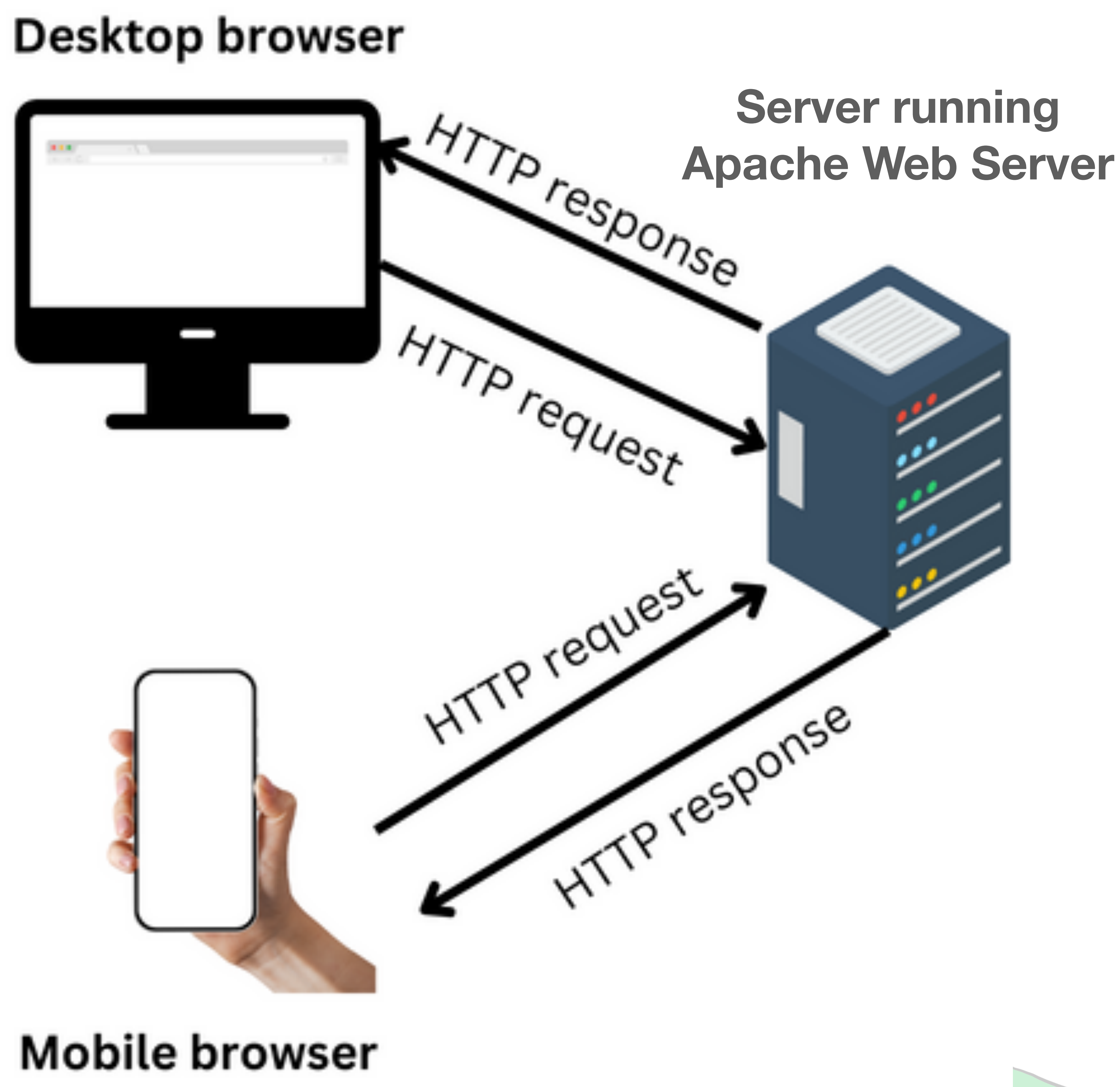
# Application Layer Protocol

## 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.

# HTTP: Hyper Text Transfer Protocol

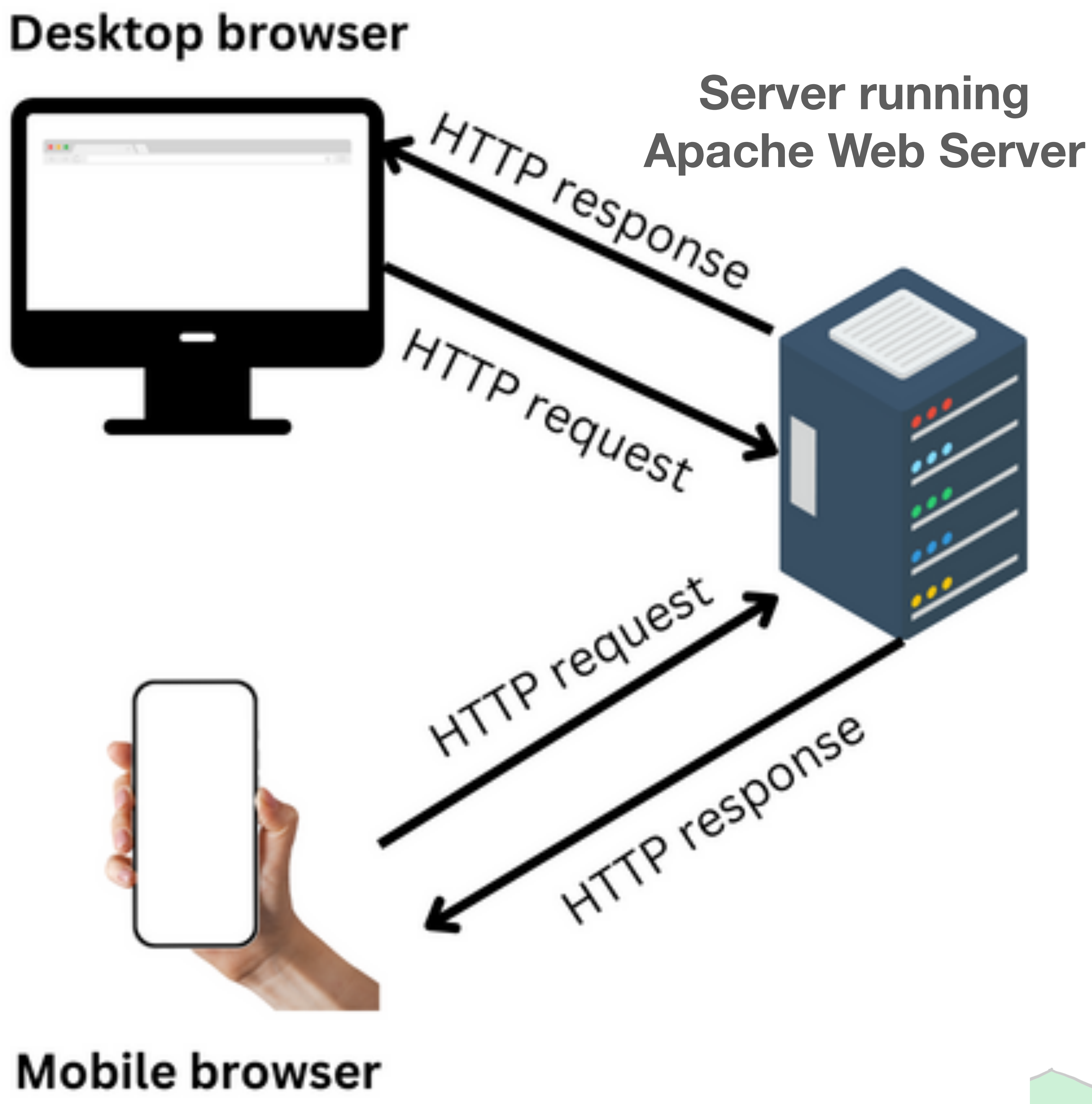
- Application layer protocol of the web
- Implemented in two programs: Client and Server
- HTTP protocol defines structure of messages
- **Client:** browser that sends requests, receives and displays web objects (using HTTP protocol)
- **Server:** Web server that sends objects in response to requests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



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

# HTTP: Hyper Text Transfer Protocol

- HTTP uses **TCP at transport layer**
  - Client initiates TCP to server (Port 80)
  - Server accepts TCP connection from client
  - HTTP messages are exchanges
  - Connection is closed
  - HTTP is reliable - why?
- HTTP is **stateless**
  - Server maintains no information about client



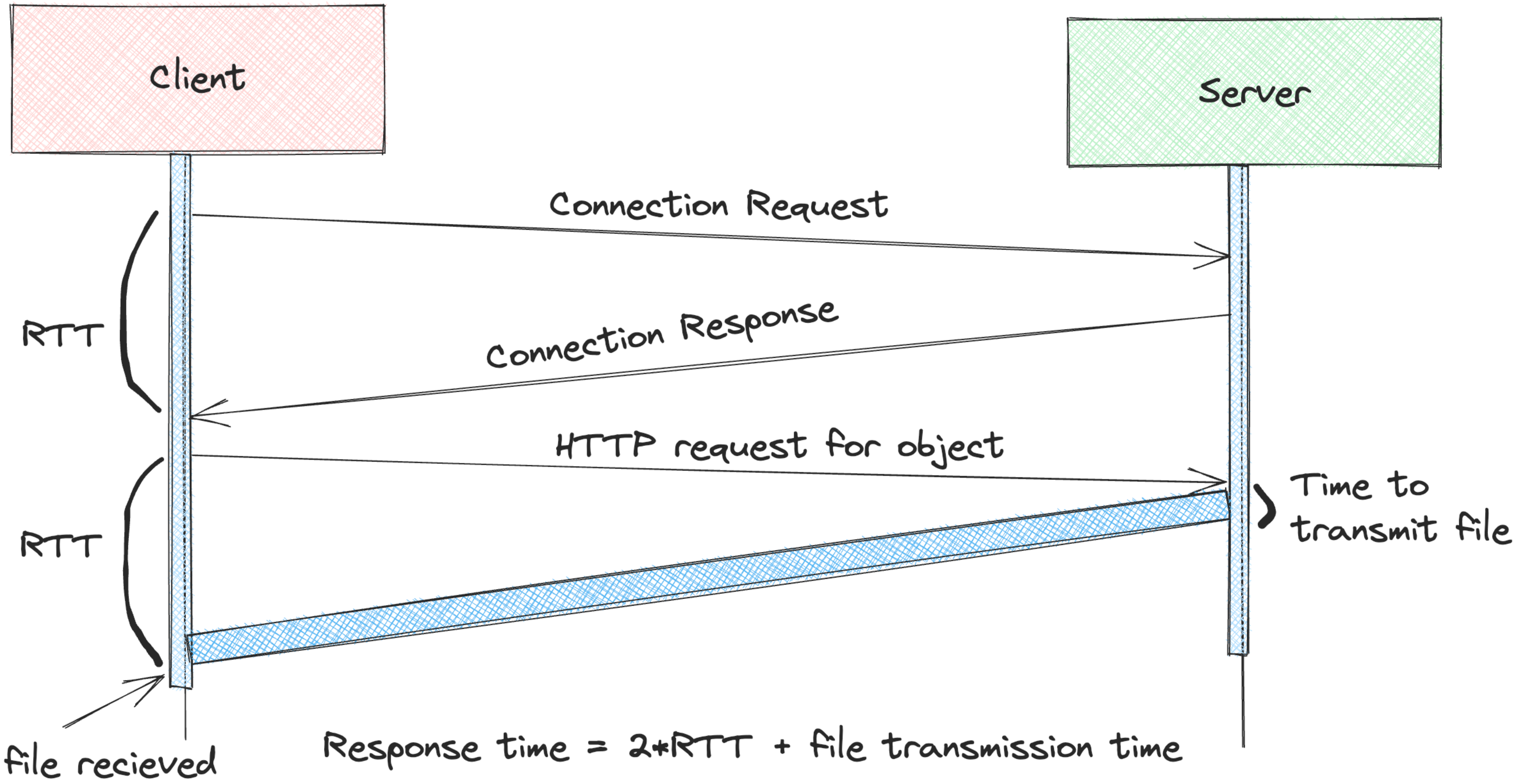
# Types of HTTP Connection

## Persistent and Non-Persistent Connections

- **Non-persistent HTTP Connection (HTTP/1.0)**
    - For every connection, the client has to create a request (one page may require multiple objects)
    - Downloading multiple objects requires multiple connections
    - Opened connection is closed after each request response
- **Persistent HTTP Connection (HTTP/1.1)**
    - One connection for all the objects
    - The opened connection is maintained

# Non Persistent Connection: Response time

## HTTP 1.0



Response time = 2*RTT + file transmission time

# HTTP Request Message

- Two types of HTTP messages: **request** and **response**
- HTTP request message: ASCII (human-readable format)
- HTTP supports different methods - GET, POST, PUT, HEAD, DELETE
- Request line: Method, URL and Version

Request line

Header lines

```
GET /index.html HTTP/1.1\r\n
Host: www.iiit.ac.in/centers \r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X
   10.15; rv:80.0) Gecko/20100101 Firefox/80.0 \r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Connection: keep-alive\r\n
\r\n
```

# HTTP Response Message

Status line

Header lines

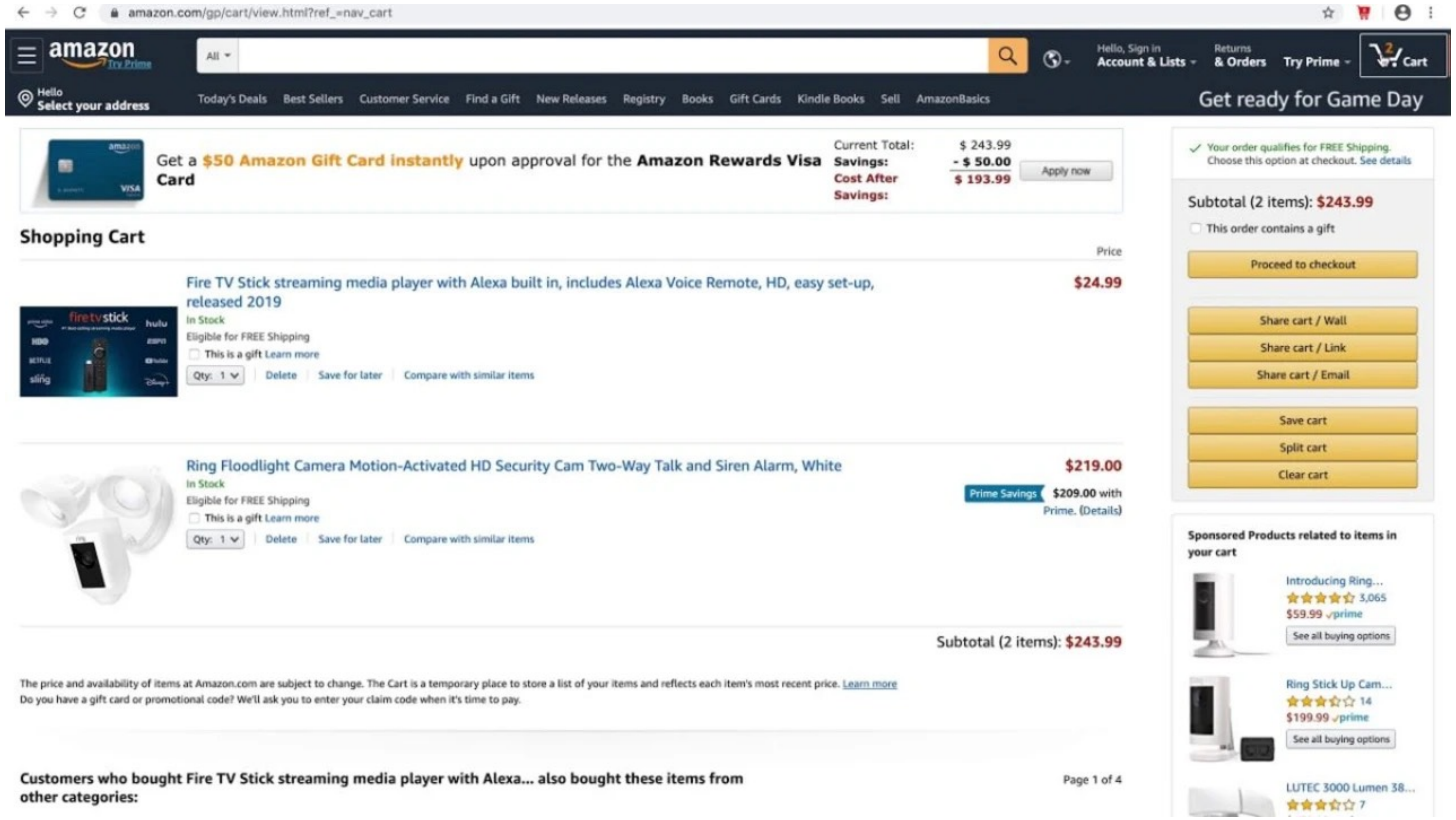
```
HTTP/1.1 200 OK
Date: Tue, 08 Sep 2020 00:53:20 GMT
Server: Apache/2.4.6 (CentOS)
   OpenSSL/1.0.2k-fips PHP/7.4.9
   mod_perl/2.0.11 Perl/v5.16.3
Last-Modified: Tue, 01 Mar 2016 18:57:50 GMT
ETag: "a5b-52d015789ee9e"
Accept-Ranges: bytes
Content-Length: 2651
Content-Type: text/html; charset=UTF-8
\r\n
data data data data data ...
```

- Status line: version and status code
- Last-modified in header can help in caching - **How?**

# HTTP Status Codes

| Status Code | Description |
|---|---|
| **200** | Request succeeded, status ok |
| **301** | Moved permanently, requested object moved, new location in the location: field (header) |
| **400** | Bad request, request message not understood by server |
| **404** | Document does not exist |
| **505** | HTTP version not supported |

# Can this happen in stateless protocol



Get a **$50 Amazon Gift Card instantly** upon approval for the **Amazon Rewards Visa Card**

|  |  |
|---|---|
| Current Total: | $ 243.99 |
| **Savings:** | **- $ 50.00** |
| **Cost After Savings:** | **$ 193.99** |

✓ Your order qualifies for FREE Shipping. Choose this option at checkout. See details

Subtotal (2 items): **$243.99**

□ This order contains a gift

**Shopping Cart**

Price

Fire TV Stick streaming media player with Alexa built in, includes Alexa Voice Remote, HD, easy set-up, released 2019 — **$24.99**

In Stock

Eligible for FREE Shipping

□ This is a gift Learn more

Qty: 1 | Delete | Save for later | Compare with similar items

Ring Floodlight Camera Motion-Activated HD Security Cam Two-Way Talk and Siren Alarm, White — **$219.00**

Prime Savings $209.00 with Prime. (Details)

In Stock

Eligible for FREE Shipping

□ This is a gift Learn more

Qty: 1 | Delete | Save for later | Compare with similar items

Subtotal (2 items): **$243.99**

The price and availability of items at Amazon.com are subject to change. The Cart is a temporary place to store a list of your items and reflects each item's most recent price. Learn more

Do you have a gift card or promotional code? We'll ask you to enter your claim code when it's time to pay.

**Customers who bought Fire TV Stick streaming media player with Alexa... also bought these items from other categories:**

Page 1 of 4

**Sponsored Products related to items in your cart**

Introducing Ring...
★★★★☆ 3,065
$59.99 ✓prime

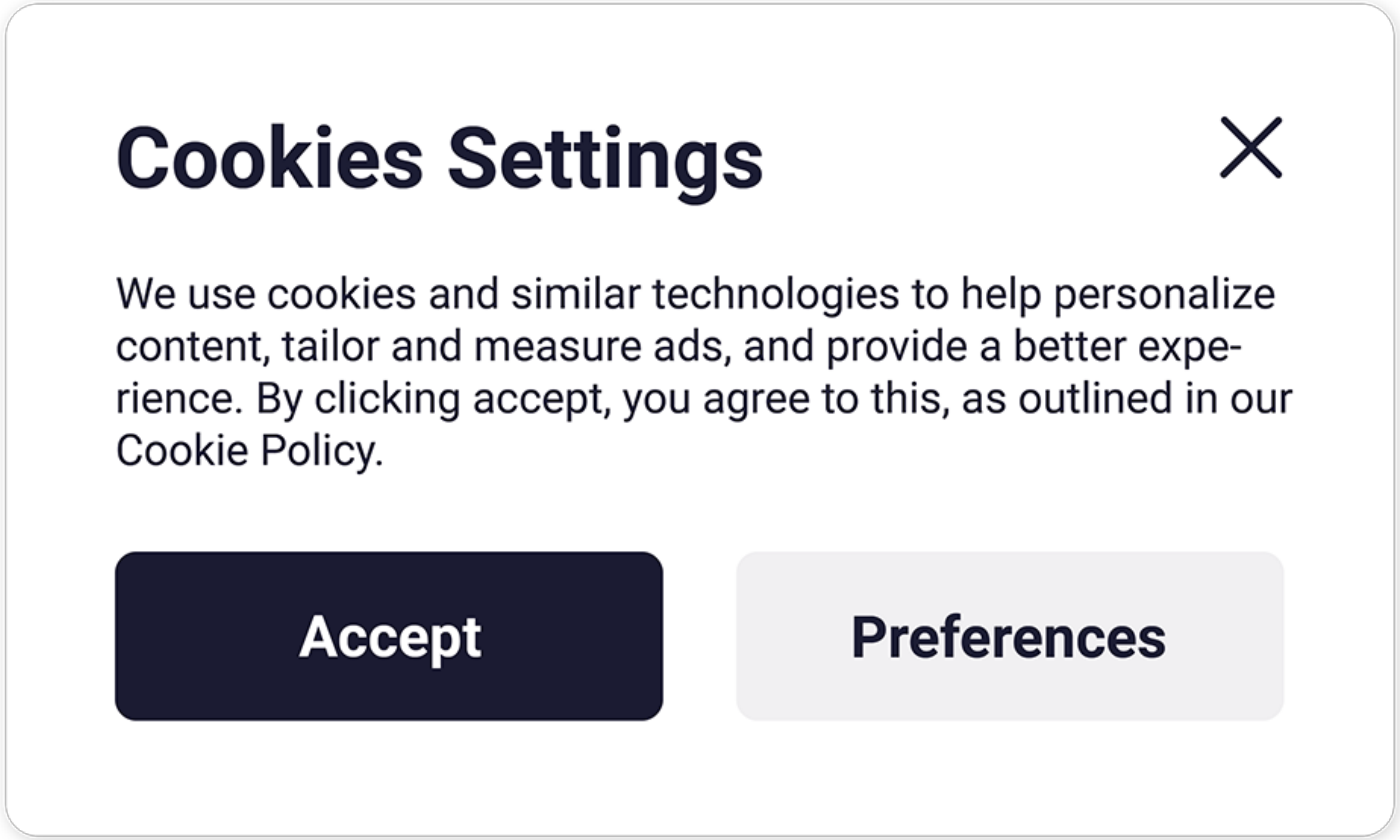
Ring Stick Up Cam...
★★★☆☆ 14
$199.99 ✓prime

LUTEC 3000 Lumen 38...
★★★☆☆ 7

- Even after closing and opening the website, the cart has the items
- How's website like Amazon, ebay, etc able to do this?

Image source: https://en.softonic.com/articles/how-to-use-share-a-cart-for-amazon

# Why not cookies!



## Cookies Settings

We use cookies and similar technologies to help personalize content, tailor and measure ads, and provide a better experience. By clicking accept, you agree to this, as outlined in our Cookie Policy.

**Source:** https://boom visibility.com

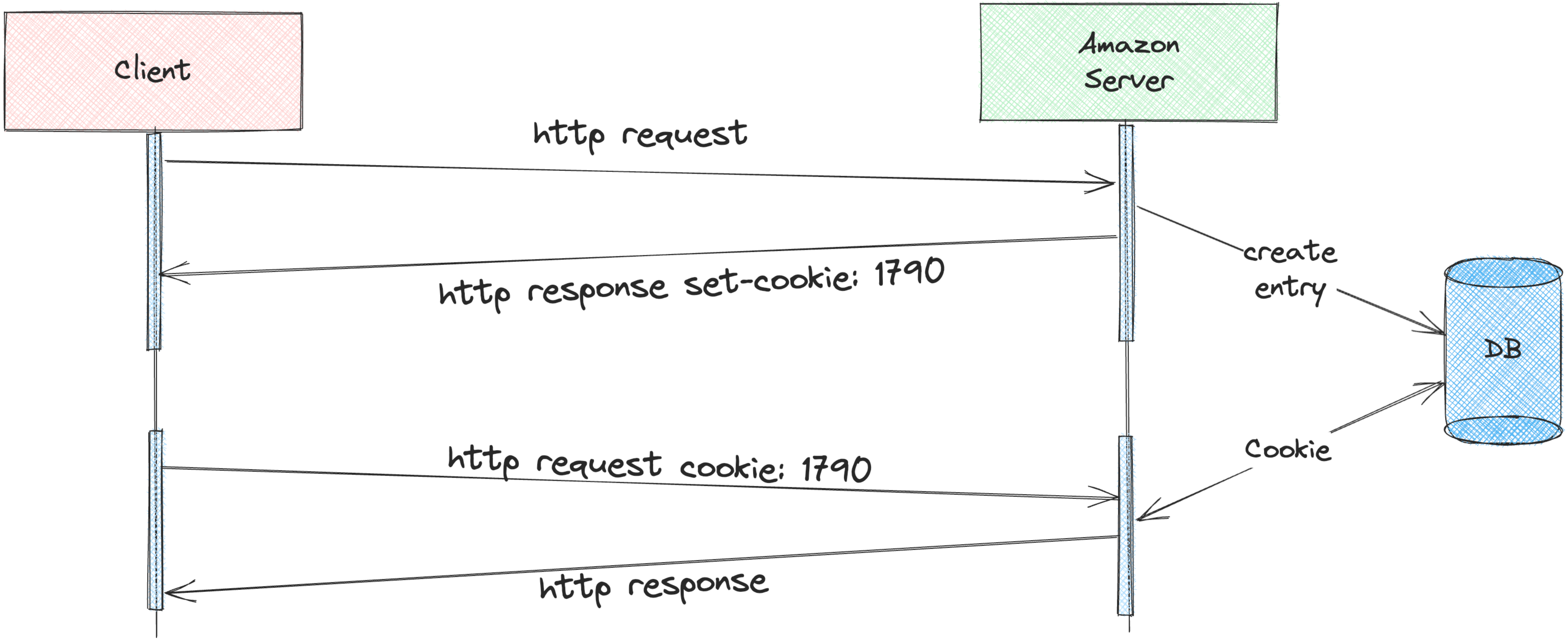
# Maintaining User/Server State: Cookies

- HTTP server is **stateless**
- Helps in supporting thousands of simultaneous connections
- Each connection is treated separately
- Website may want to identify users for various reasons
  - Keep session information
  - Recommend similar products

# Cookies

- HTTP header consists of information for cookies
- Consists of four components:
    - Cookie header line in HTTP response
    - Cookie header line in HTTP request message
    - Cookie file kept in clients system
    - Backend database on the server/website
    - Cookies can be used to create user session on top of HTTP - **Can be invasion of Privacy!**

# Cookie Illustration



# Web Caches

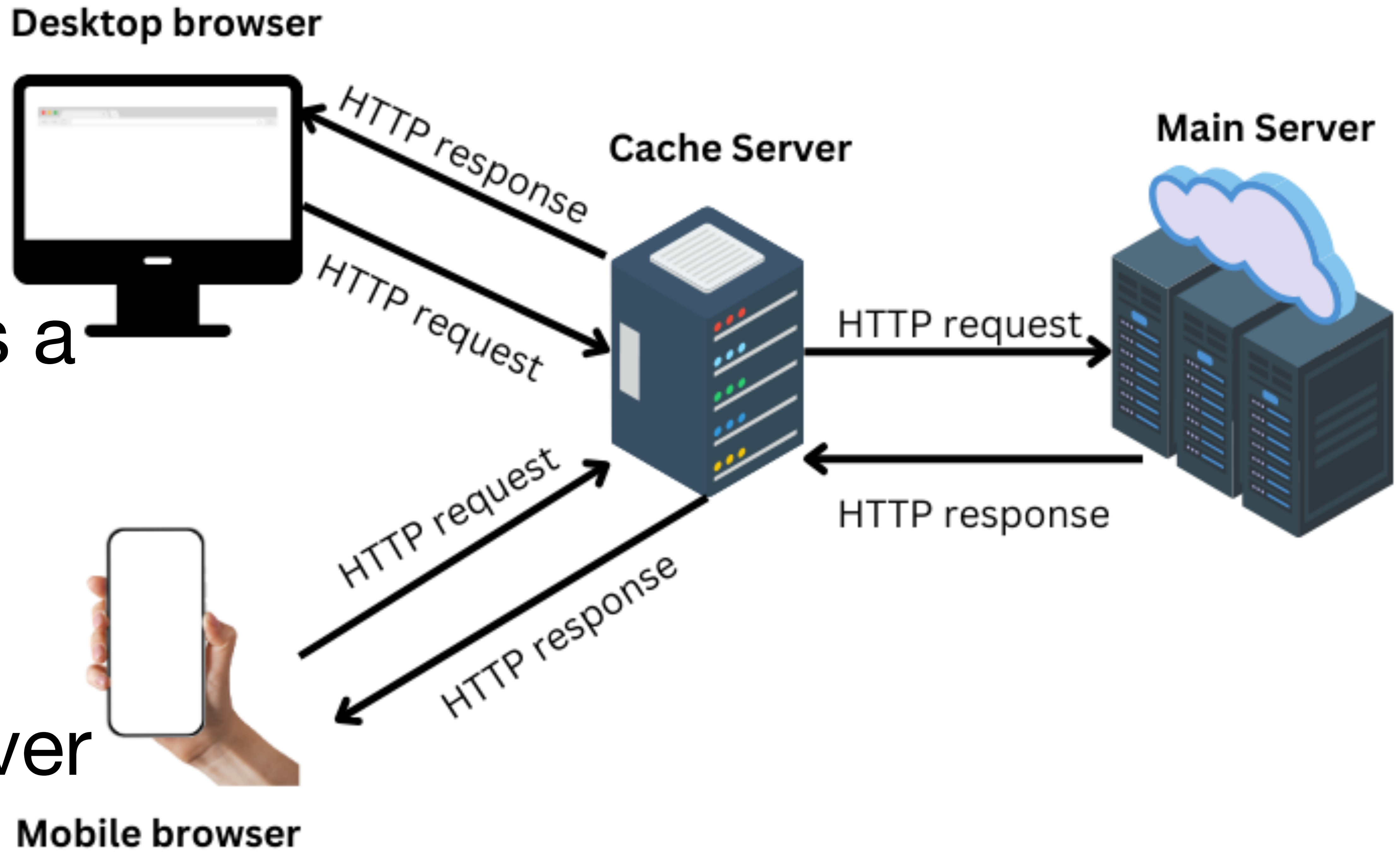
- Not every time we need to access the main (original) web server

- 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

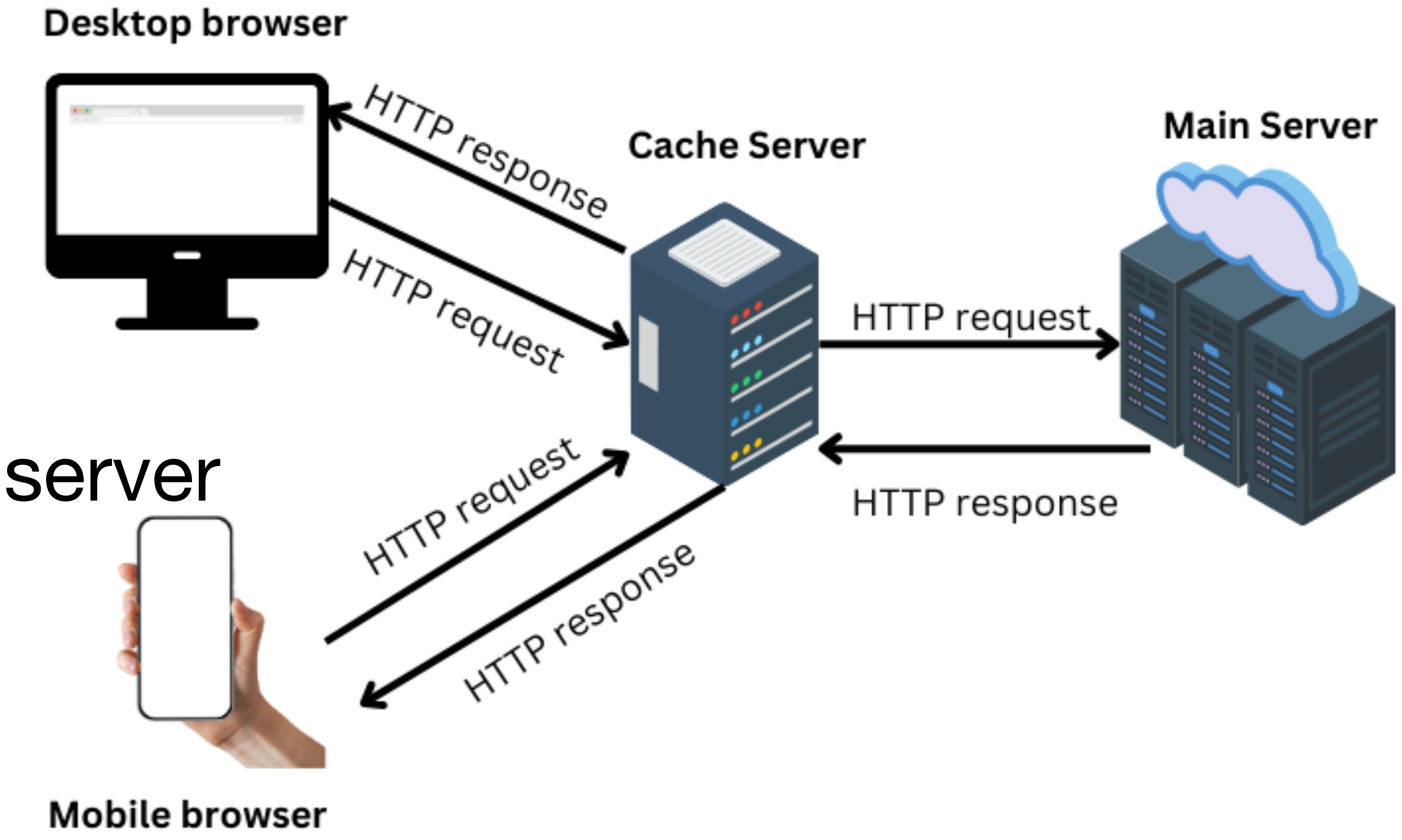


```
Cache-Control: max-age=<seconds>
```

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

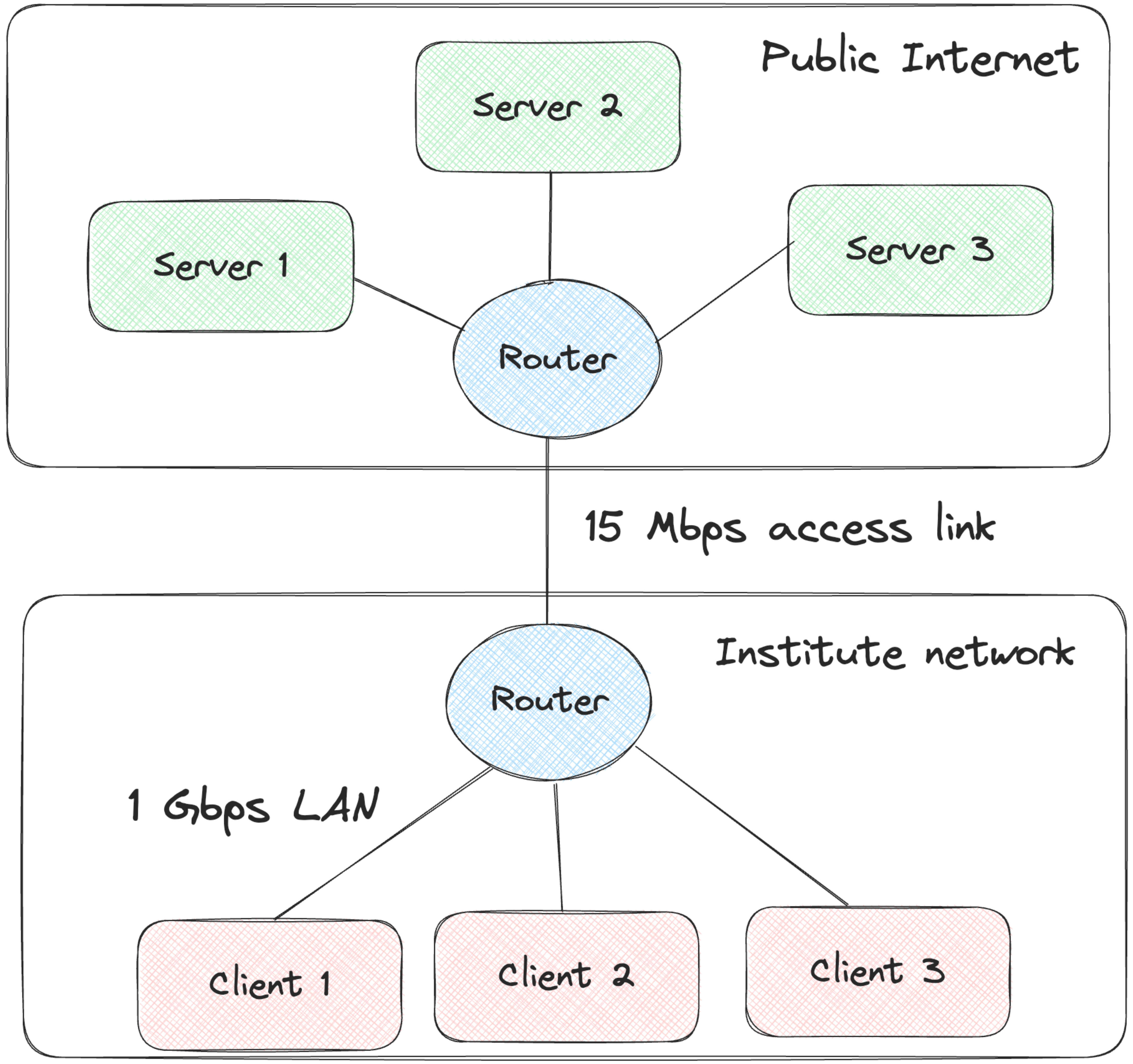
# Caching Benefits

- Reduce response time for a client request
  - Cache is closer to client
  - There may be bandwidth constraints
- Reduce traffic on the internet and to main server
  - Significantly improve performance
- Internet is dense with caches
  - Effective delivery of content



# University Scenario

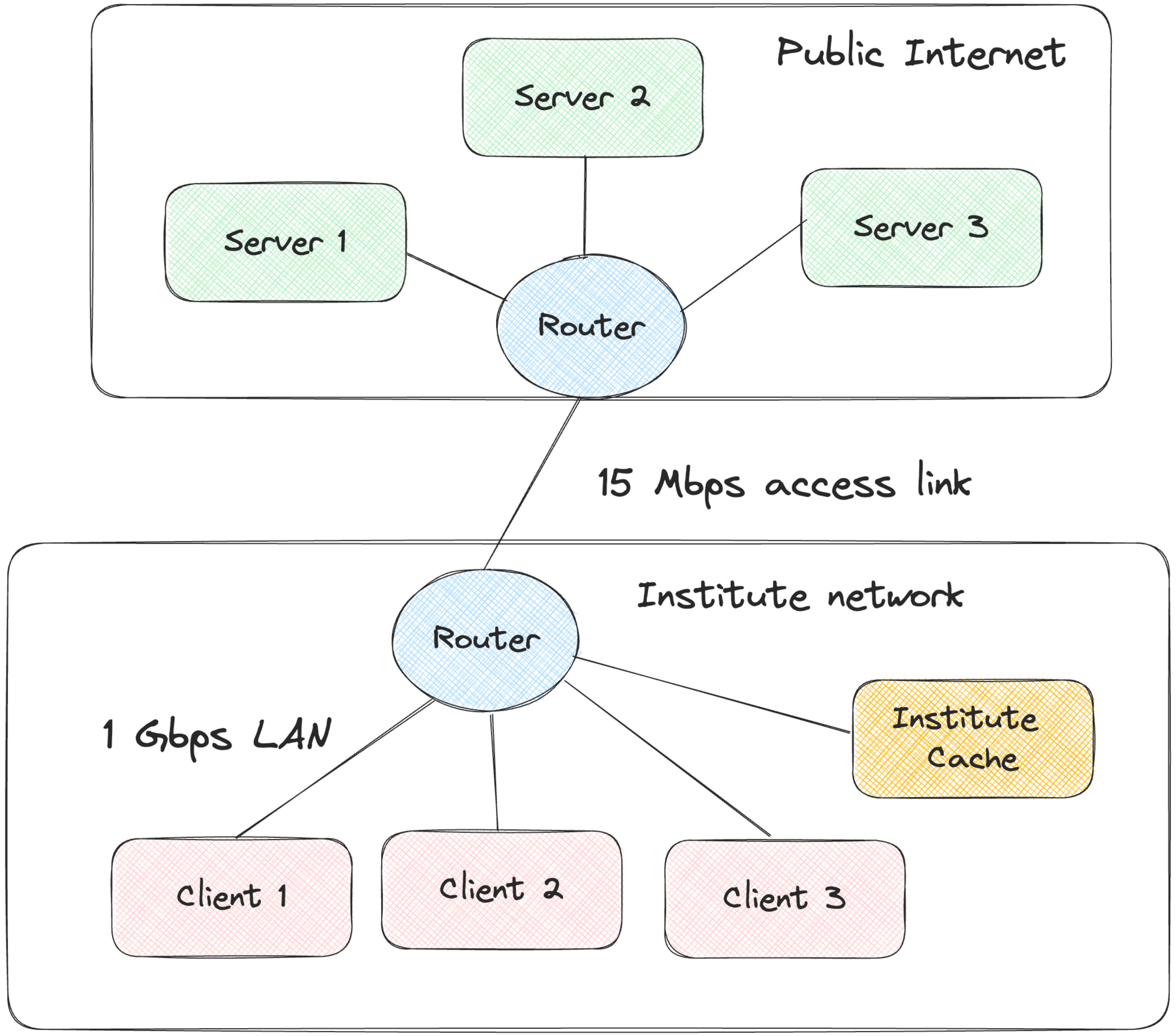
- Assume that:
    - servers to router in public internet incurs ~ 2 sec delay
    - Data is 1Mbits
    - Request interval from client ~ 15 requests/ second
- End-to-end delay = Internet delay + access link delay + LAN delay
    - ~ 2 sec + ~ 1 min + ~ u sec (High!!)



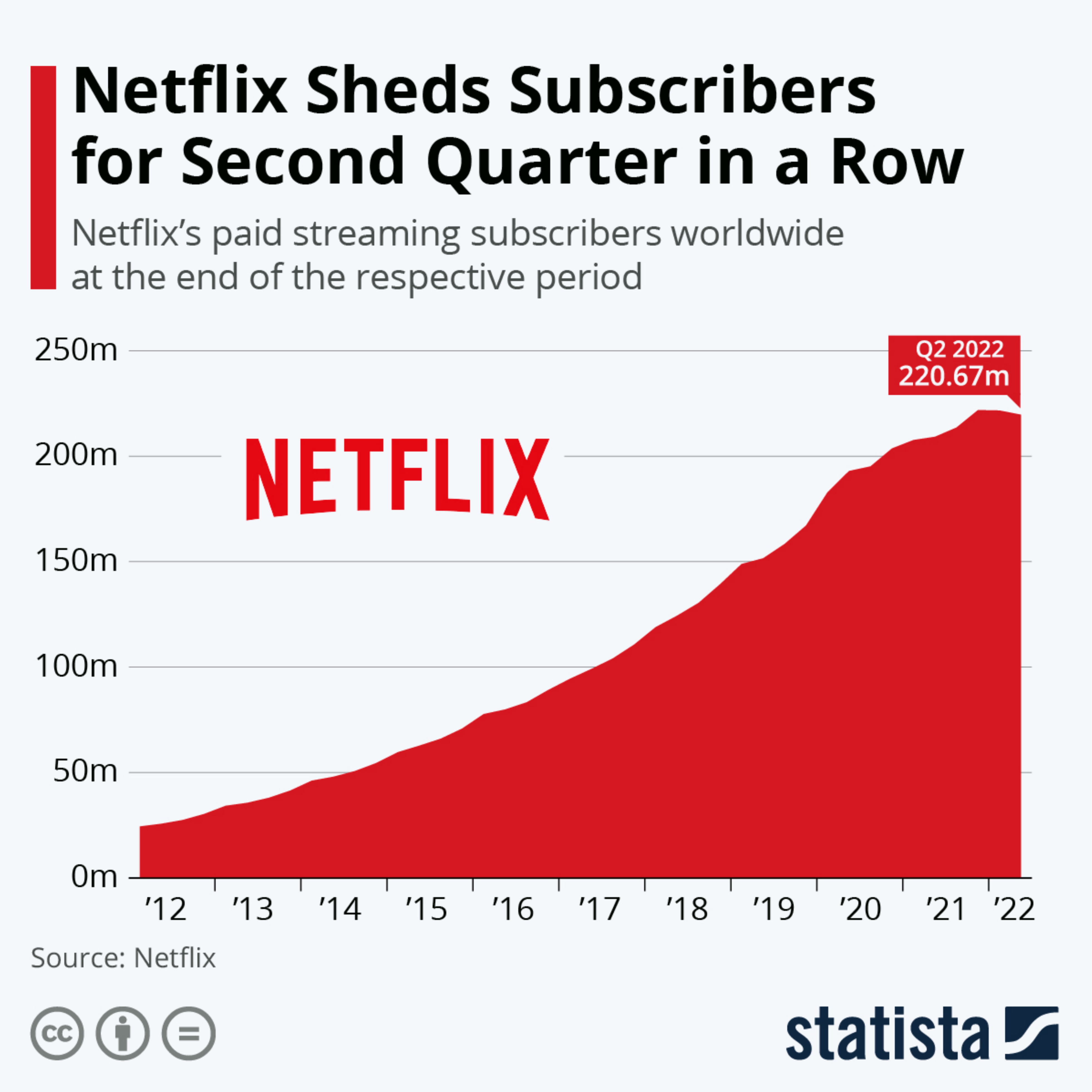
# University Scenario

- Easy way out is to increase bandwidth
  - Costly solution, request rate may increase
- Introduce Cache server at Institute level
  - Even if there is 40% hits => significant improvement
- Delay = 0.6*(2) + (0.4)*(0.01)

  = ~1.2 secs
- Cheaper solution, lower end-to-end delay



# Web Cache has broader Applications

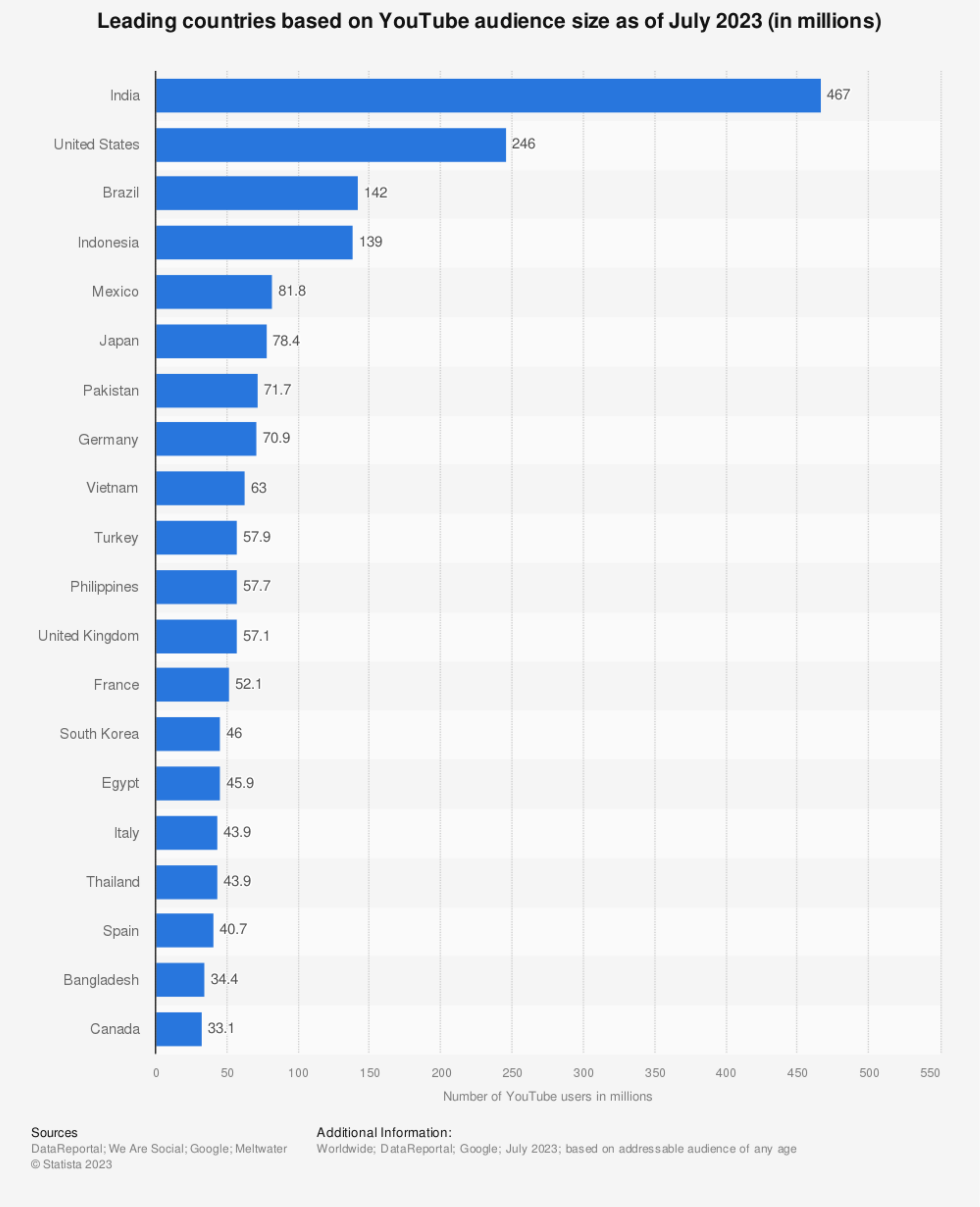


**Netflix Sheds Subscribers for Second Quarter in a Row**

Netflix's paid streaming subscribers worldwide at the end of the respective period

Source: Netflix

statista



**Leading countries based on YouTube audience size as of July 2023 (in millions)**

Sources
DataReportal; We Are Social; Google; Meltwater
© Statista 2023

Additional Information:
Worldwide; DataReportal; Google; July 2023; based on addressable audience of any age

# Content Distribution Networks (CDN)

- Many internet companies are distributing on-demand video streams to millions of users on daily basis
  - Youtube (~2.7 Billion users), Netflix (~238 Million subscribers), ..
  - Distributed across the world - Having one large data centre may not work - Why?
- To support growing demand, scale - distribute over different CDNs
  - Servers are distributed among different geographical locations
  - Requests are redirected to CDNs (private or third party)

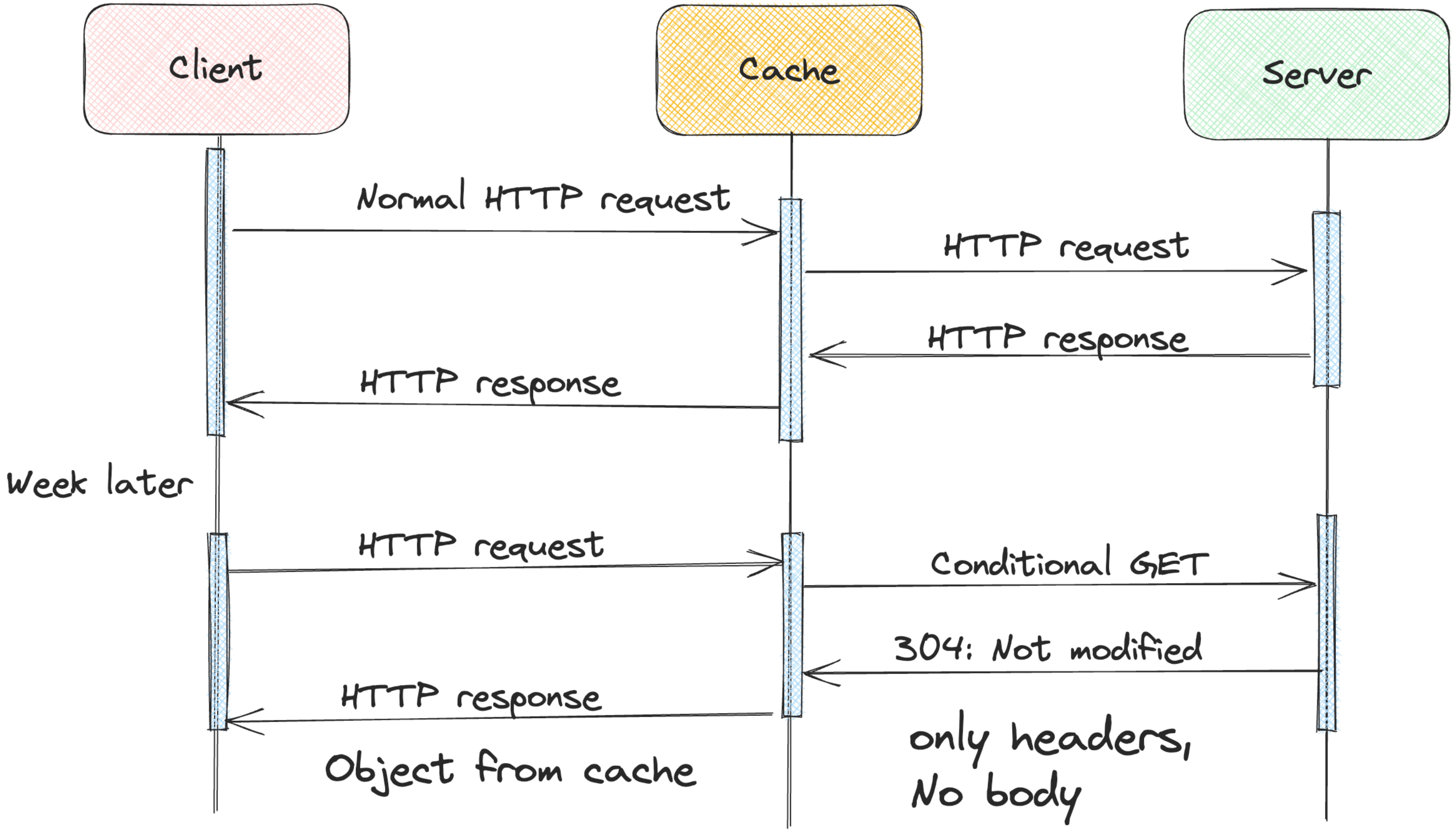
# Content Distribution Networks (CDN)

- CDNs adopt two different server placement strategies
  - **Enter Deep:** Deploy server clusters in all access ISPs
    - High maintenance, higher throughput and lower delays
  - **Bring Home:** Building larger cluster at smaller number of sites (eg: 10s)
    - Lower maintenance, lower throughput and higher delays
- Client needs to pick a cluster - Geographically closest or using real-time information (dynamic choices based on heuristics)

# Conditional GET

## Cache does not contain originals

- If things are kept in cache, the copy of original may become stale!
- HTTP provides mechanism to verify if object is up-to-date (conditional GET)
- HTTP request header - use field “If-modified-since: “
- Server provides “Last-Modified” in the response header



# HTTP 1.1 and Beyond

- **HTTP 2.0**
  - Standardised in 2015
  - As of 2020, 40% of top 10 million websites support HTTP/2
  - Primary goals
    - Enable request response multiplexing over single TCP
    - Request prioritisation
    - Server push
    - Compression of HTTP header fields
- **HTTP 3.0** underway (drafts as of 2020)

# Domain Name System (DNS)

- Students have roll numbers but addressed by names
  - Similar is the case with citizens - Aadhar, SSN
- What about the internet?
  - Google.com, Facebook.com, YouTube.com - Are they enough?
  - How to locate them? Eg: iiit.ac.in - Provides info that it is in India but exactly where?
  - IP address can help - They have a hierarchical representation

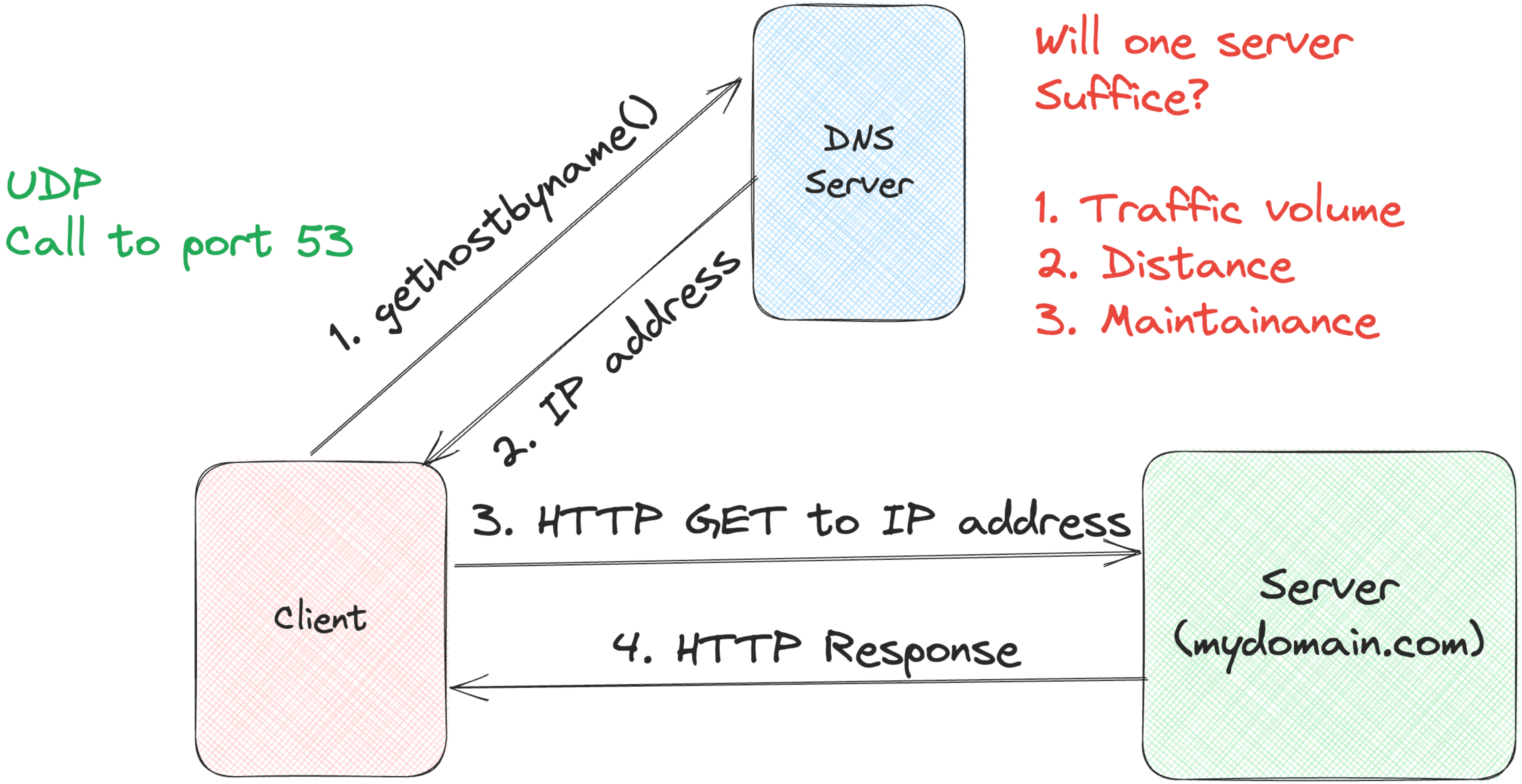
# Domain Name System (DNS)

- Directory service of the internet that translates hostnames to IP addresses
- People prefer mnemonic names, routers prefer IP addresses
- Application layer protocol, allows hosts to query a distributed database
- DNS servers are UNIX machines running **Berkley Internet Name Domain** (BIND) software
- DNS runs over **UDP** and uses port **53**
- Leveraged by protocols like HTTP and SMTP to translate hostnames to IP addresses

# Domain Name System (DNS) Services

- **Host aliasing**
  - Same hosts can have multiple aliases, resolve the names (get canonical names of host)
  - Eg: 1231242s-us-west.aws.com: mydomain.com, e-markt.com
- **Mail Server Aliasing**
  - Mail servers may also have aliases
  - DNS can provide canonical names of mail server to mail clients
- **Load distribution**
  - Perform distribution among replicated servers
  - Eg: Amazon may be replicated to multiple servers (IP address), keep giving back IP in different order

# DNS: How does it work?



UDP
Call to port 53

Will one server Suffice?

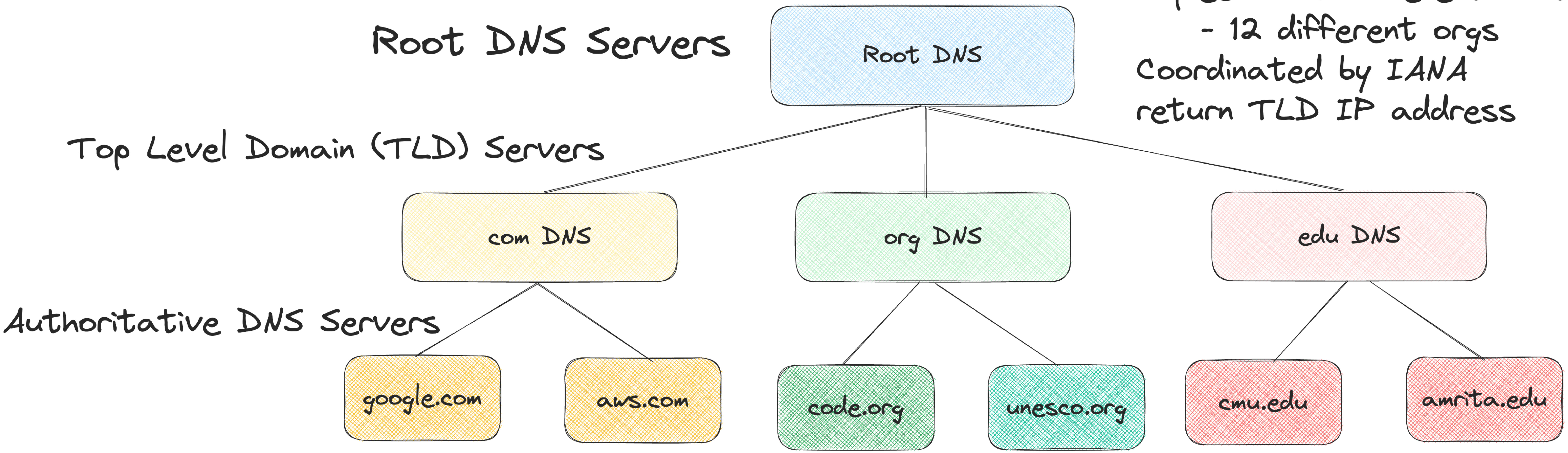
1. Traffic volume
2. Distance
3. Maintainance

wants to
send HTTP request
to mydomain.com

# 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

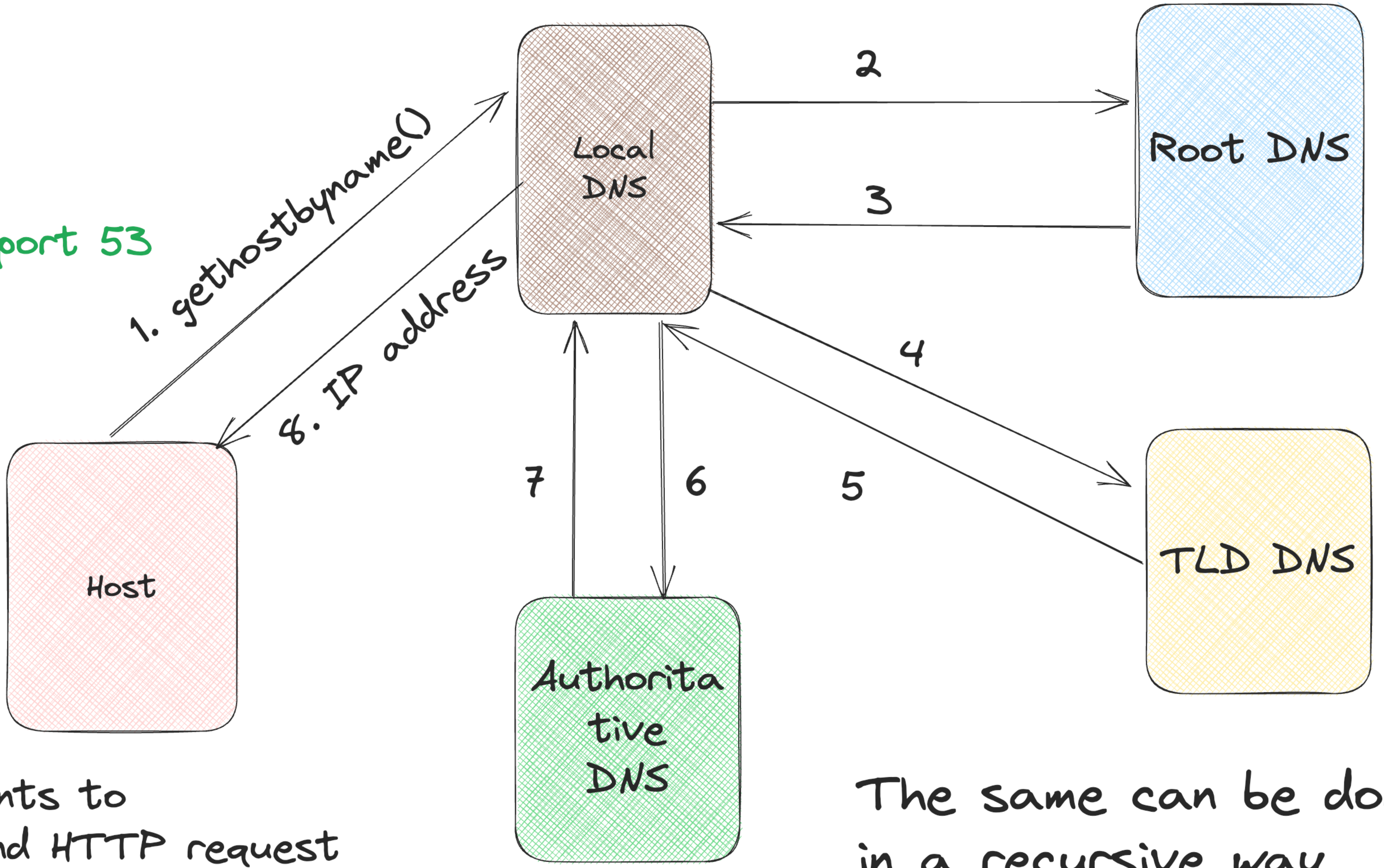


TLD Servers - can be maintained by orgs, provide IP of authoritative DNS Servers
Authoritative DNS servers - Orgs can choose to implement their own or go for third party
All DNS records have to be made public - that maps hosts to IP address

# Local DNS

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

UDP
Call to port 53



Caching can be done at the ISP Local DNS

wants to
send HTTP request
to mydomain.com

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

# DNS Records

## DNS Servers stores Resource Records (RR)

- Each RR is a tuple: **(name, value, type, ttl)**

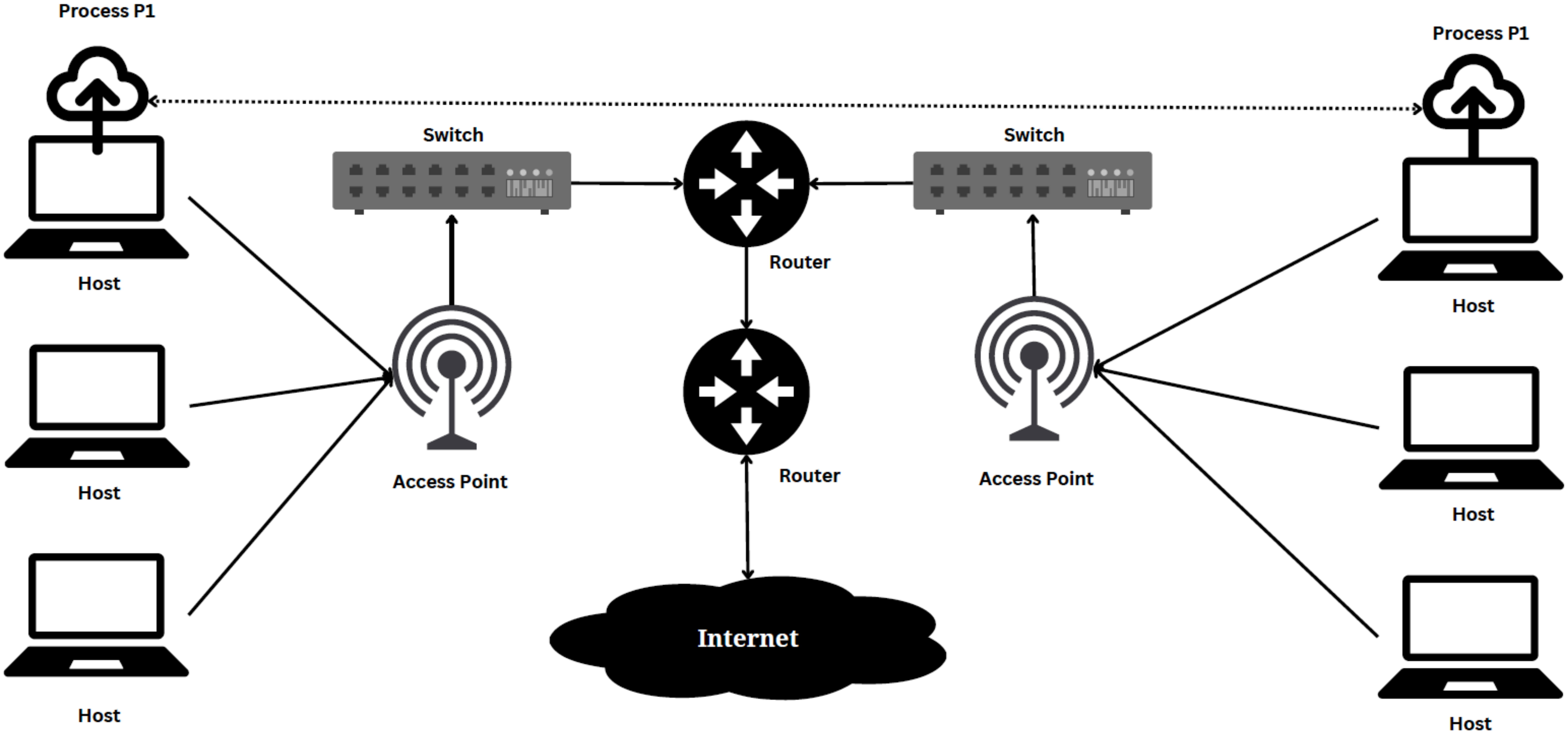
- **type = A**
  - name is hostname
  - value is IP address
  - (abc.com, 122.x.x.x, A, 3600)
- **type = NS**
  - name is domain
  - value is hostname of authoritative
  - (abc.com, ns.host.com, NS, 86400)
- **type = CNAME**
  - name is alias of canonical name
  - value is canonical name
  - (abc.com, west-abc.com, CNAME, 86400)
- **type = MX**
  - name is domain
  - value is name of SMTP mail server
  - (abc.com, mail.abc.com, MX, 3600)

# More questions needs to be answered

**How does Network layer route the traffic?**

**Before that: Its not that only one process will be running at one time?**

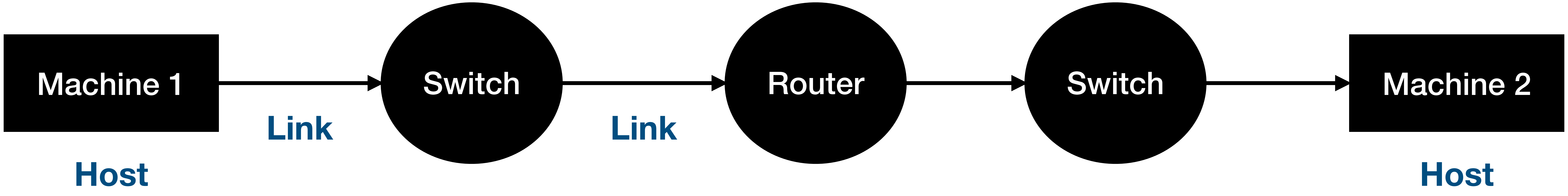
# The Bigger Picture



# How does host to host communication work?

# What’s the contribution of the link layer?

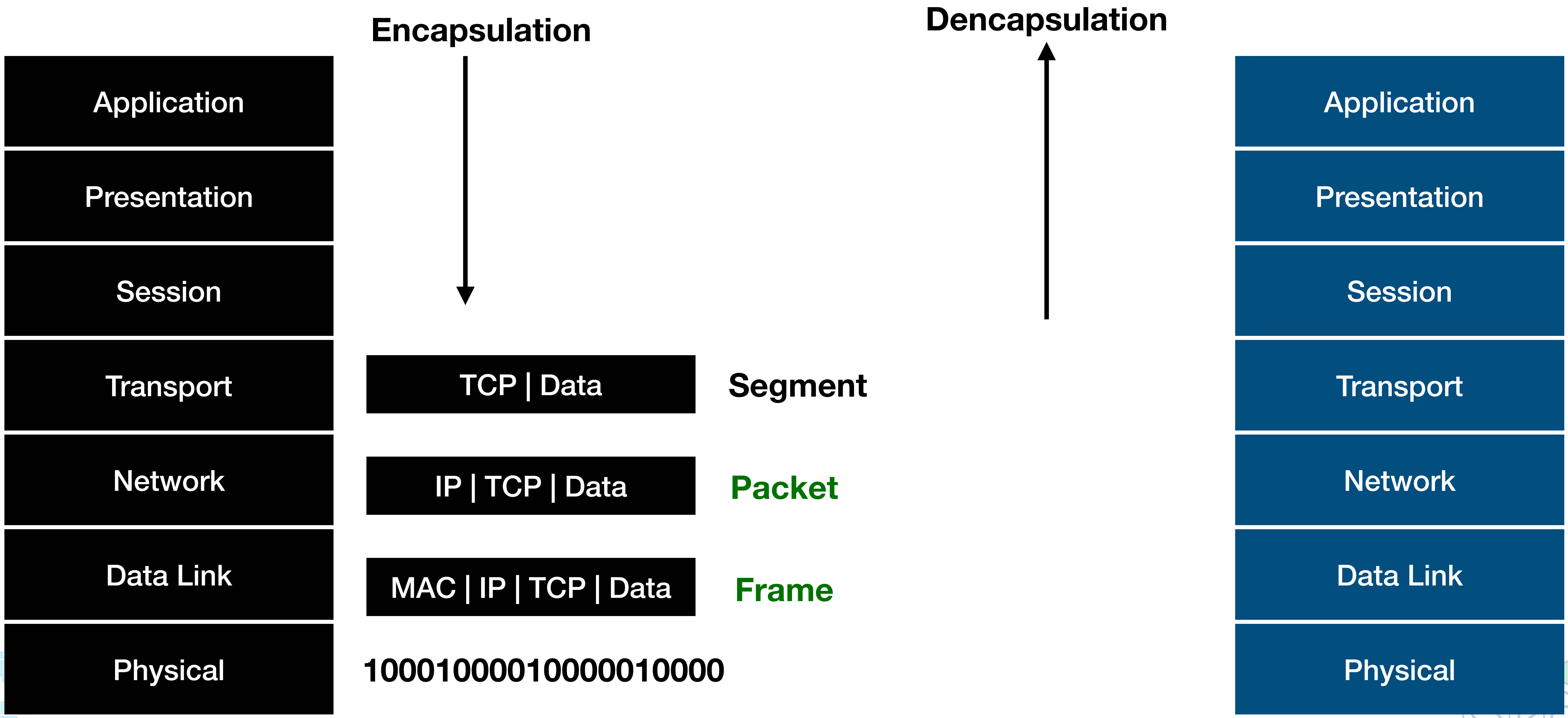
# Remember the Components?



# What we have seen so far

- **Application layer**
  - Provides support for end applications to format and manage data
  - HTTP, DNS, SMTP, etc.
  - In turn they make use of transport layer protocols
- **Transport layer**
  - Provides support for communication between services
  - TCP, UDP
  - Ports helps in identifying the right services/process
  - But transport layer by itself is not enough! - **Requires underlying support - Why?**

# Putting It Together



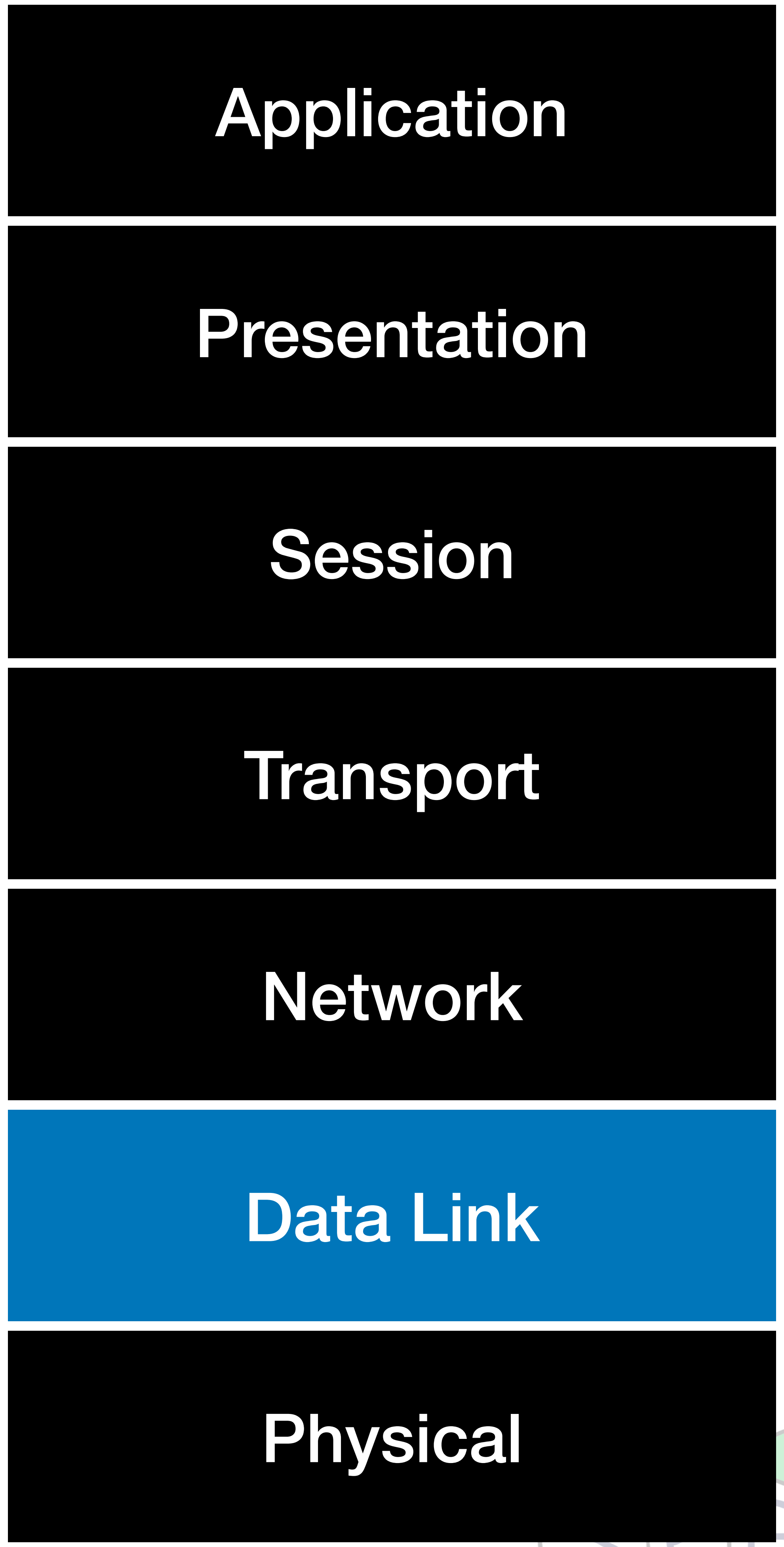
Encapsulation

Dencapsulation

| | | | |
|---|---|---|---|
| Application | | | Application |
| Presentation | | | Presentation |
| Session | | | Session |
| Transport | TCP \| Data | Segment | Transport |
| Network | IP \| TCP \| Data | Packet | Network |
| Data Link | MAC \| IP \| TCP \| Data | Frame | Data Link |
| Physical | 10001000010000010000 | | Physical |

# Link Layer

- 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



Application

Presentation

Session

Transport

Network

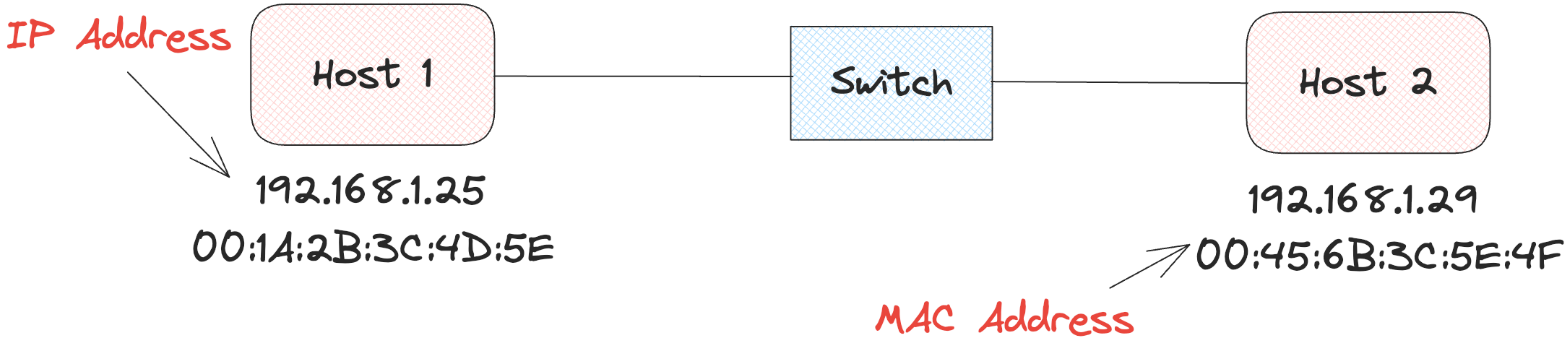
Data Link

Physical

# Two process want to communicate with each other

- Two main possibilities
  - Two processes are running in machines in the same network
  - Two processes are running in machines in two different networks
- Same network
  - Two machines are connected through switches
  - It can be a large intranet but the machines are still in same network

# When machines are in same network

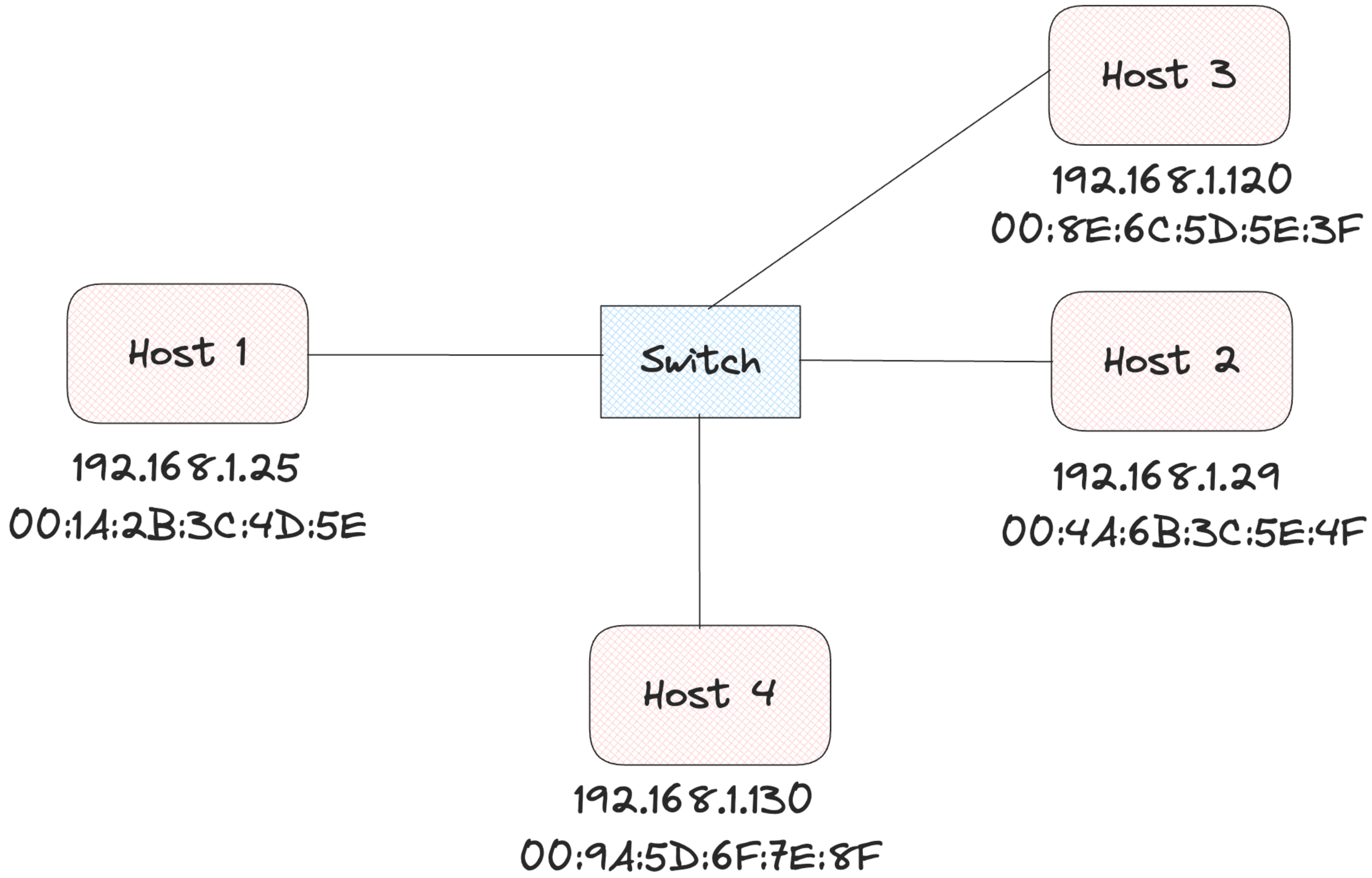


- Assume Host 1 wants to communicate to Host 2 known by domain host2.com
- Host 1 will know the IP address of Host 2 - **How?**
- IP address will help with L3 communication
- Will just the IP address suffice? What about L2? How to get the MAC address?

# Why MAC and not just IP

- **Analogy:**
    - MAC address is like Aadhar card number
    - IP address is like Postal address
- **MAC address is unique to each interface**
    - Each interface on a LAN has globally unique MAC address
    - Each interface has a **locally unique IP address**
- How to determine interface’s MAC address knowing its IP address?

# ARP - Address Resolution Protocol

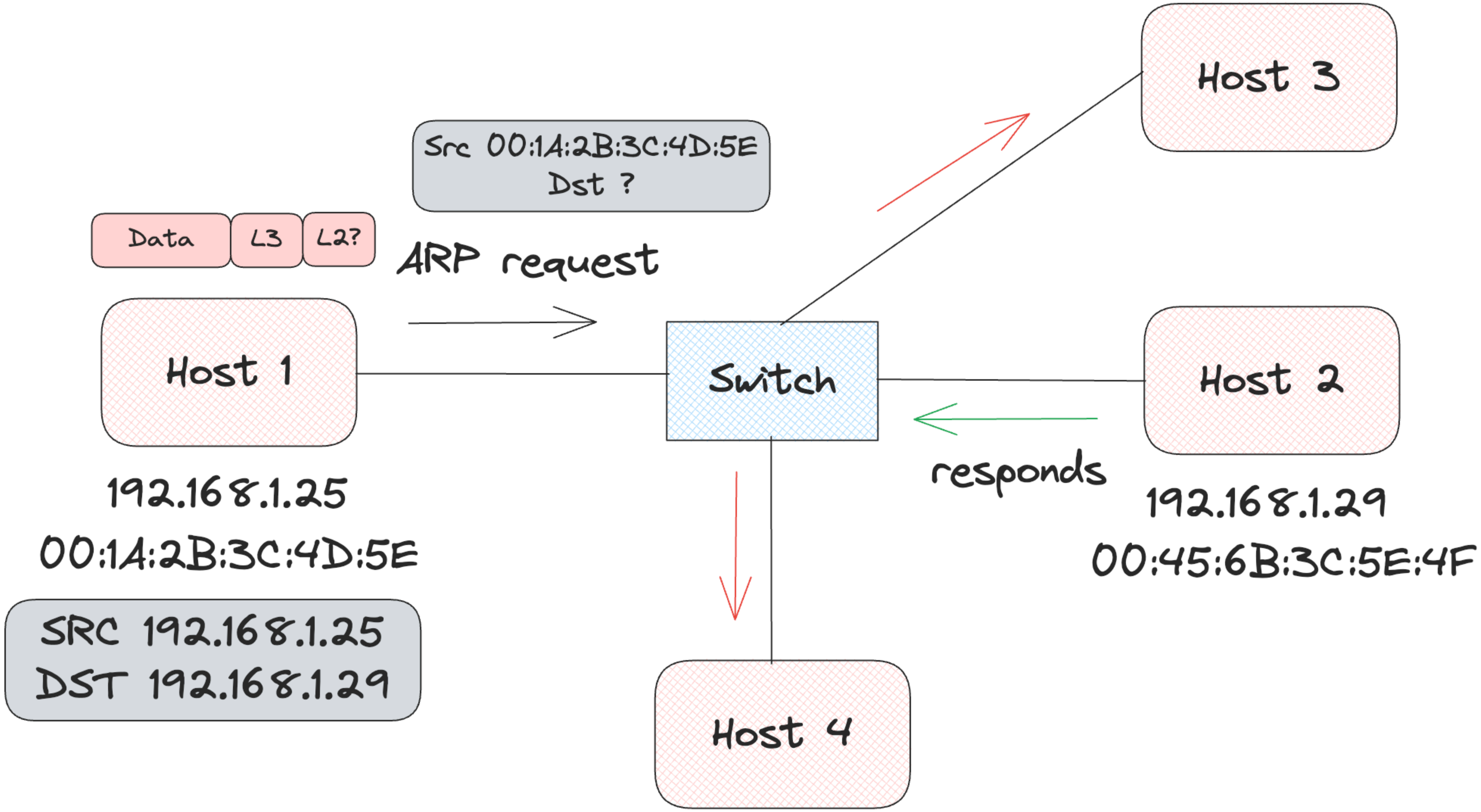


- Each IP node (router, host) on the LAN has a table - **ARP Table**
- IP/MAC address mappings for some LAN nodes
  - **<ip address, MAC address, TTL>**
- **TTL:** Time to live, time after which the mapping will be forgotten (20 mins)

# ARP - Address Resolution Protocol

**ARP Table in Host 1**

| IP | MAC | TTL |
|---|---|---|
| | | |
| | | |
| | | |
| | | |



- Host 1 wants to send data to Host 2
- Sends ARP request to all the hosts in the network and updates table

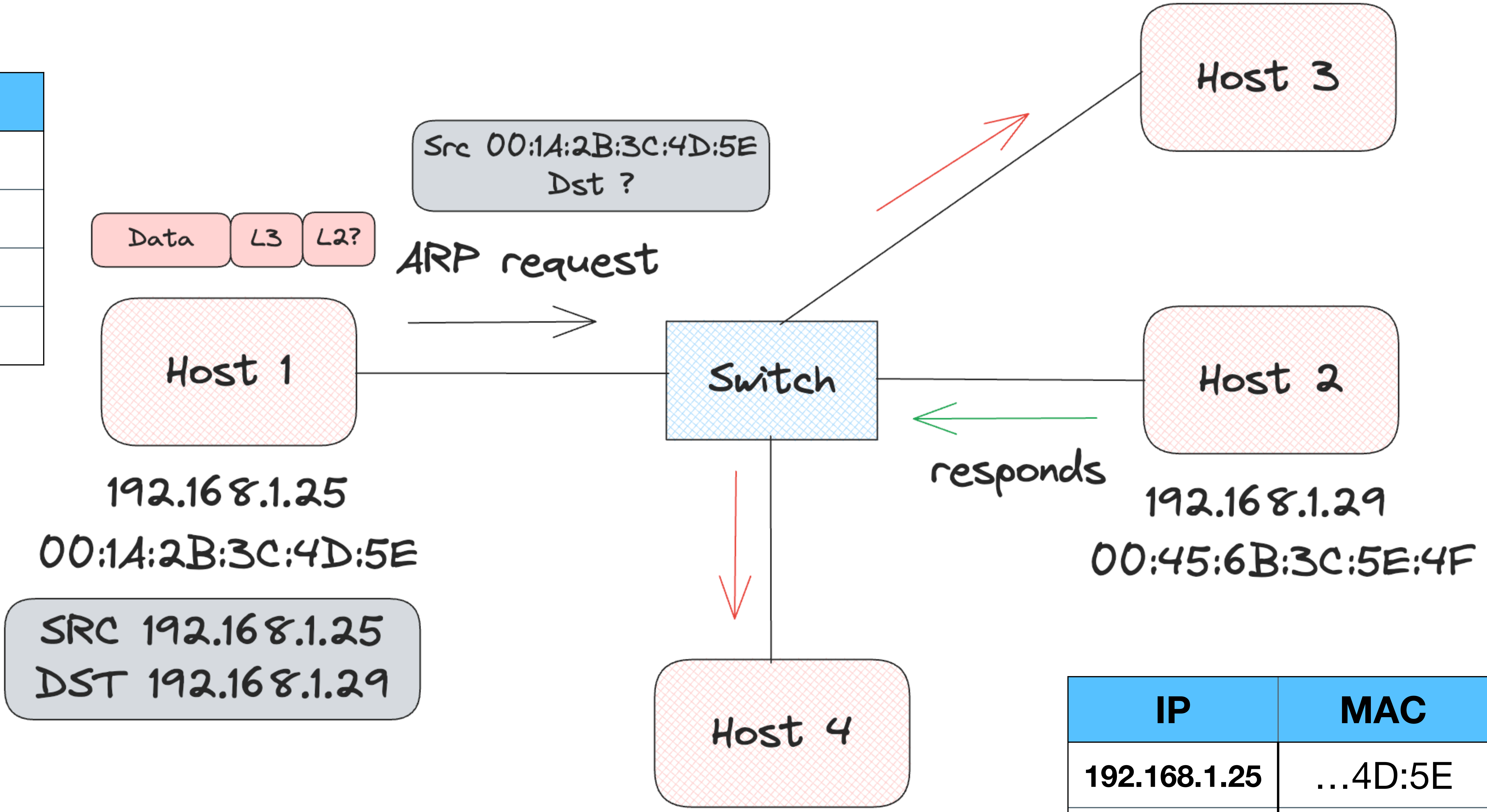
# ARP Query

- 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)
- 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 Table

| IP | MAC | TTL |
|---|---|---|
| 192.168.1.29 | …:5E:4F | 20 |
| | | |
| | | |
| | | |

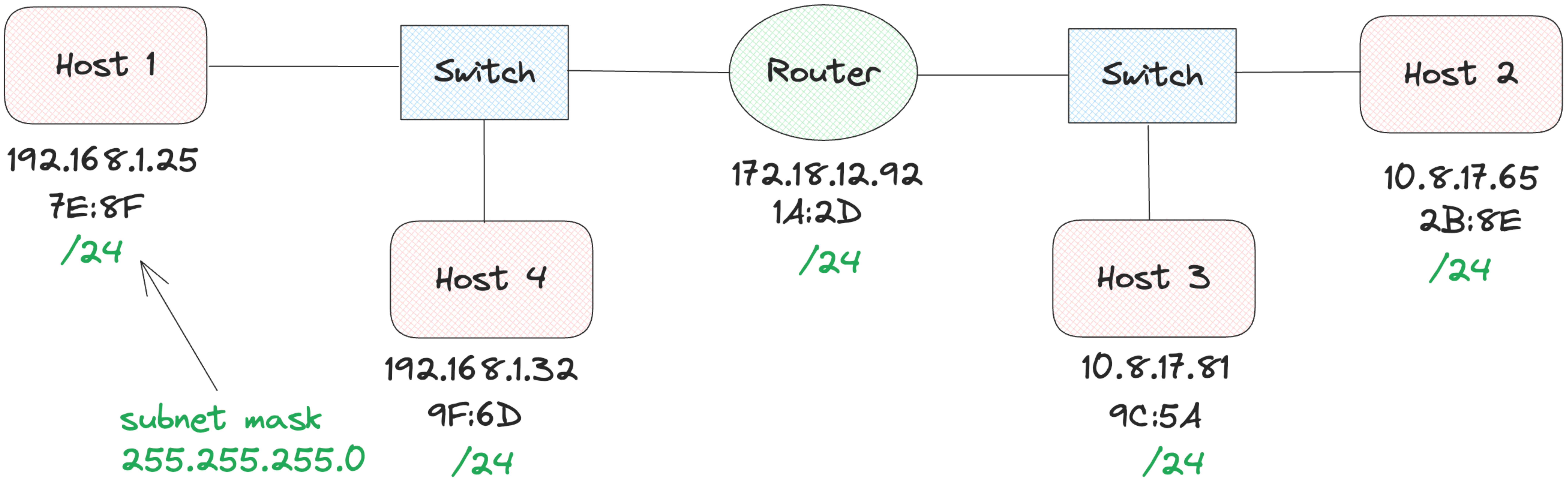
ARP Table at Host 1



| IP | MAC | TTL |
|---|---|---|
| 192.168.1.25 | …4D:5E | 20 |
| | | |
| | | |
| | | |

**Host 2 is saving the MAC of Host 1 request in its ARP table**

# When Machines are in Different Network

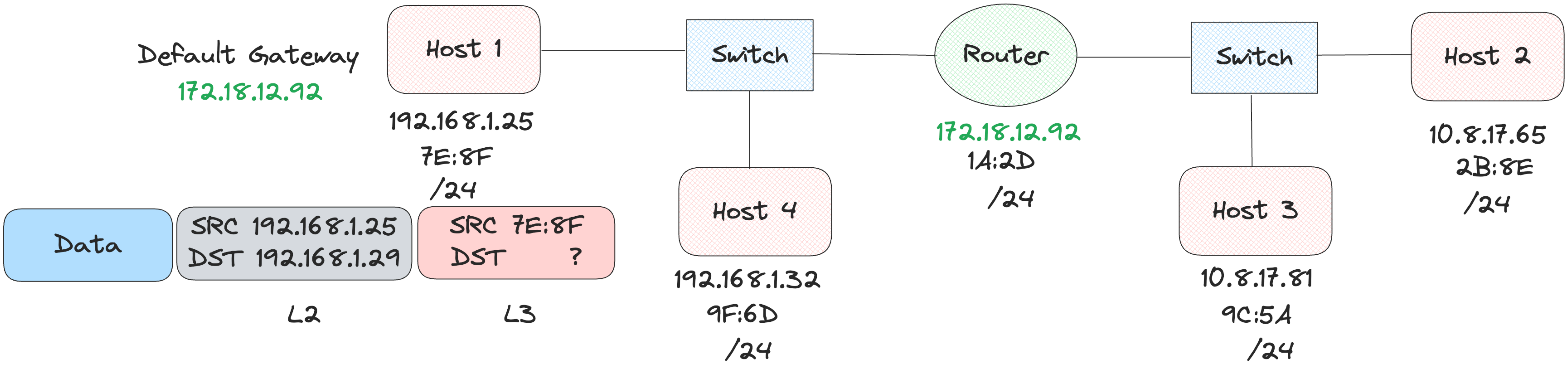


- Host 1 wants to communicate to Host 2 - Knows the IP address
- Host 1 also knows that Host 2 is not in the same network? - **How?**
  - /24 - Subnet mask, what do we mean by that?
  - Router **bridges** the two different networks

# Subnets

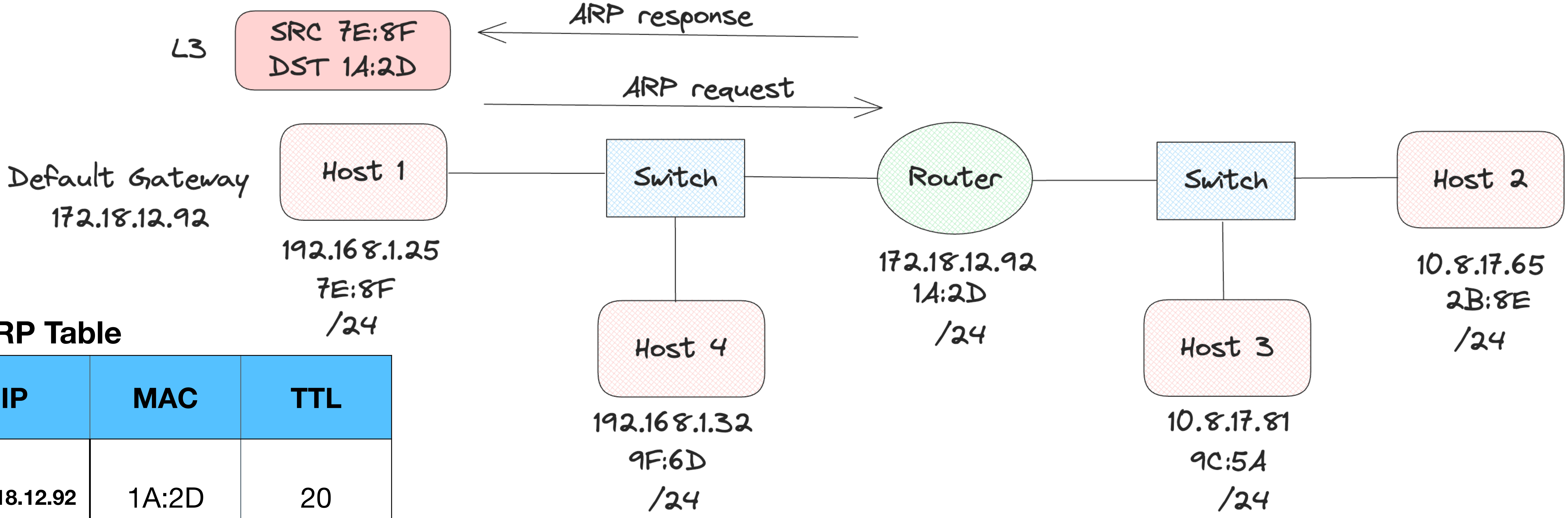
- Practice of dividing a network into one or more networks is subnetting
- Allows to create hierarchy within an organisation
  - Think about 172.18.21.x (country.organization.department.machine)
  - Another set of IP within organisations 172.18.y.x
- Consider an IP address 172.18.21.0 with a subnet mask of 255.255.255.0 or /24
  - Implies one network that can contain 254 host addresses (only the last one can change)
  - /24 - CIDR Notation (Classless Inter domain routing) - Number of 1s in the address
  - **What about subnet mask 255.255.0.0? Or /16?**

# When Machines are in Different Network



- Host 1 does not know the destination MAC address
- It need not broadcast to all MAC in the same network - **Why?**
- It can make use of the default gateway - IP address of the router

# When Machines are in Different Network



**ARP Table**

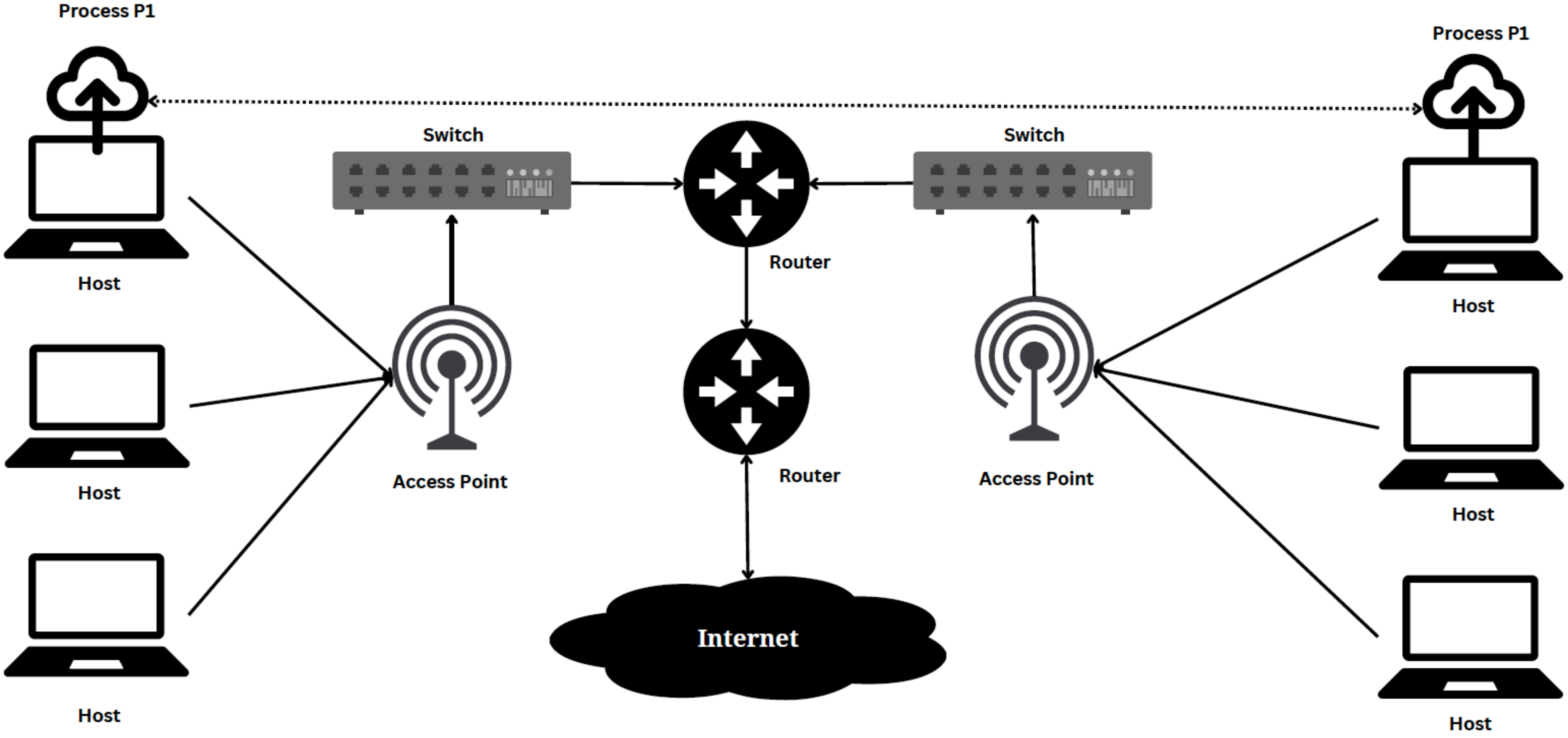
| IP | MAC | TTL |
|---|---|---|
| 172.18.12.92 | 1A:2D | 20 |

- 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)

## How does end-to-end communication work?

## What is the role of the network layer?

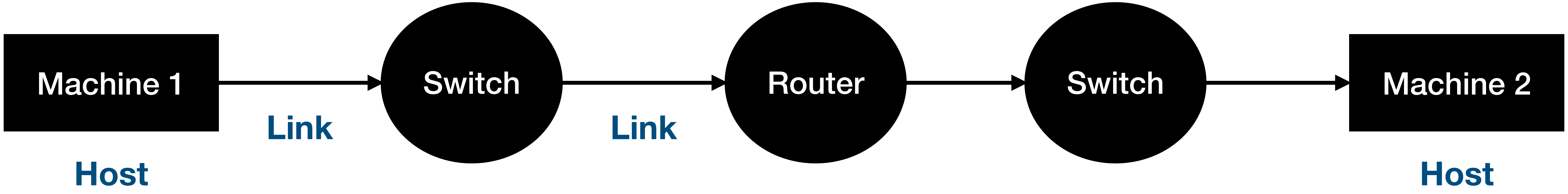
# The Bigger Picture



## How does end-to-end communication work?

## What is the role of the network layer?

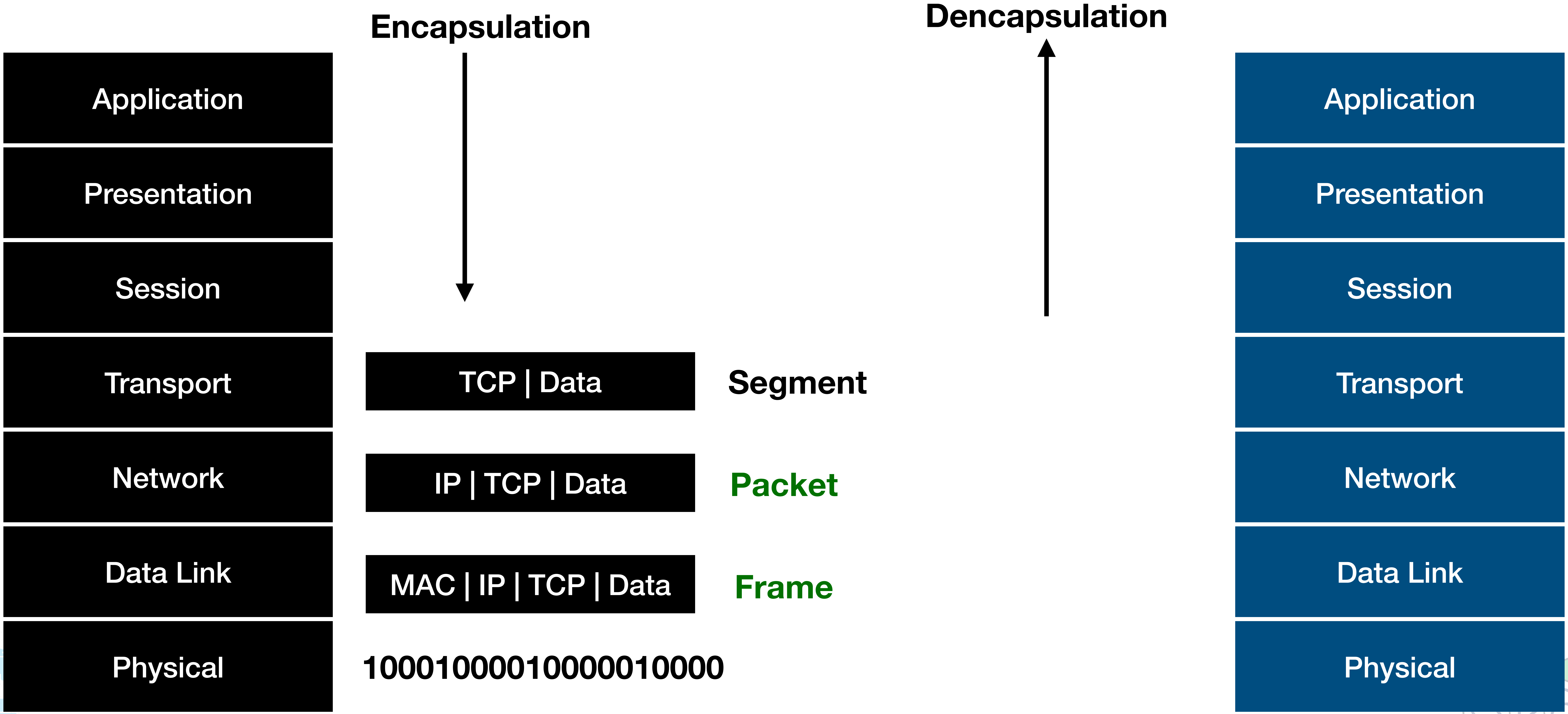
# Remember the Components?



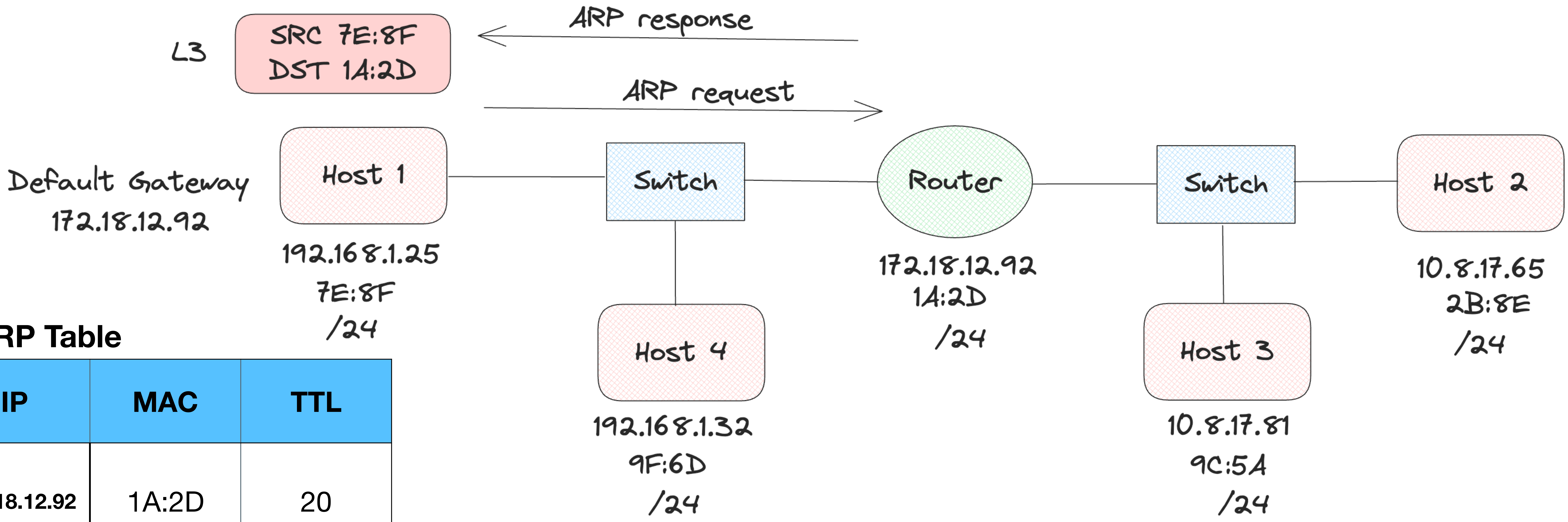
# What we have seen so far

- **Application layer**
  - Provides support for end applications to format and manage data
  - HTTP, DNS, SMTP, etc.
  - In turn they make use of transport layer protocols
- **Transport layer**
  - Provides support for communication between services
  - TCP, UDP
  - Ports helps in identifying the right services/process
  - But transport layer by itself is not enough! - **Requires underlying support - Why?**

# Putting It Together



# Link Layer Working



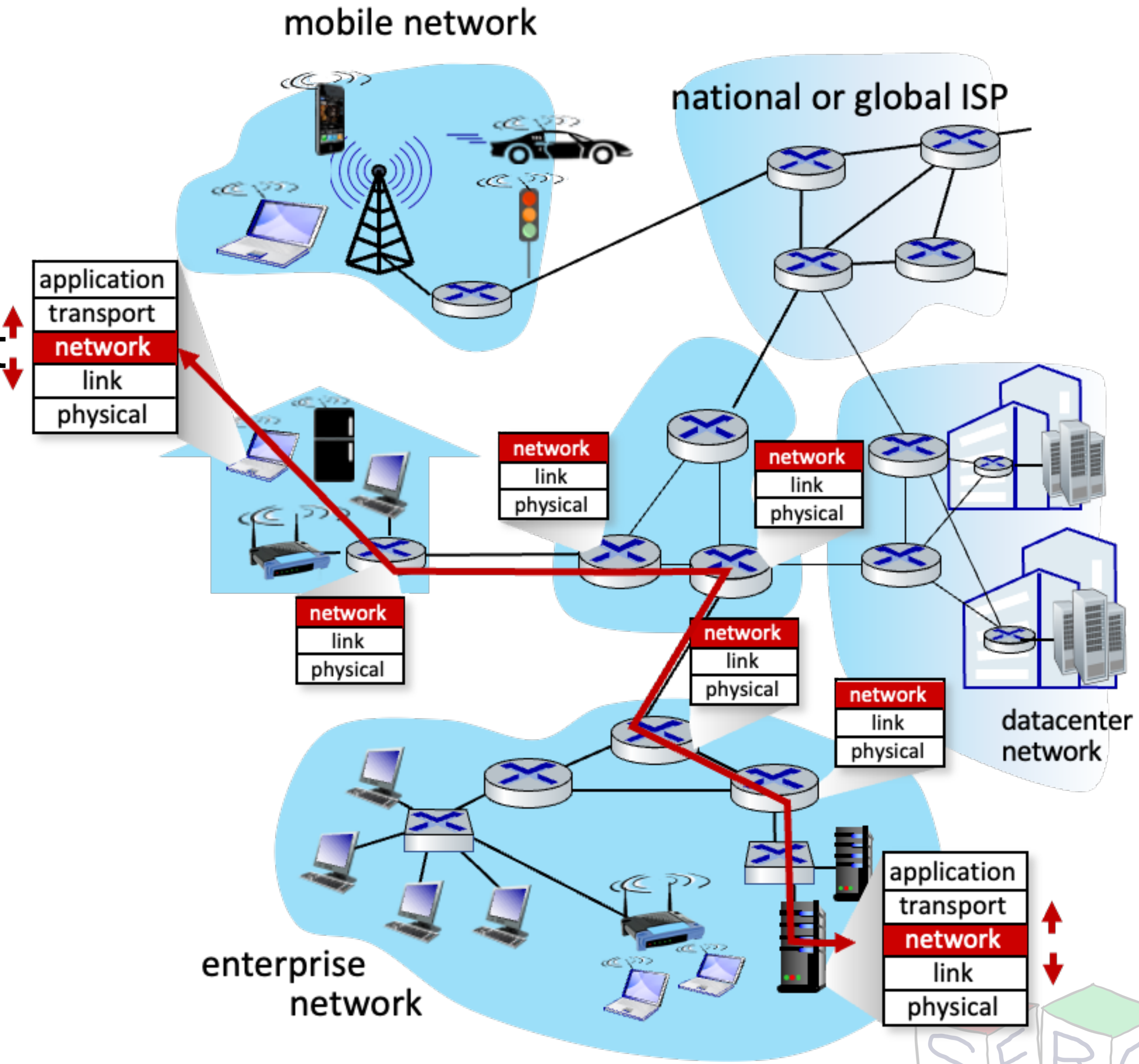
**ARP Table**

| 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)

# Network Layer - Functionalities

- Plays key role in end-to-end communication
    - Link layer is concerned about just hop to hop
- Transport segment from sending to receiving host
    - **Sender:** Encapsulates segments into datagrams, passes to Link layer
    - **Receiver:** Delivers segments to transport layer protocol
- Network layer protocols in every internet device
    - Hosts and routers



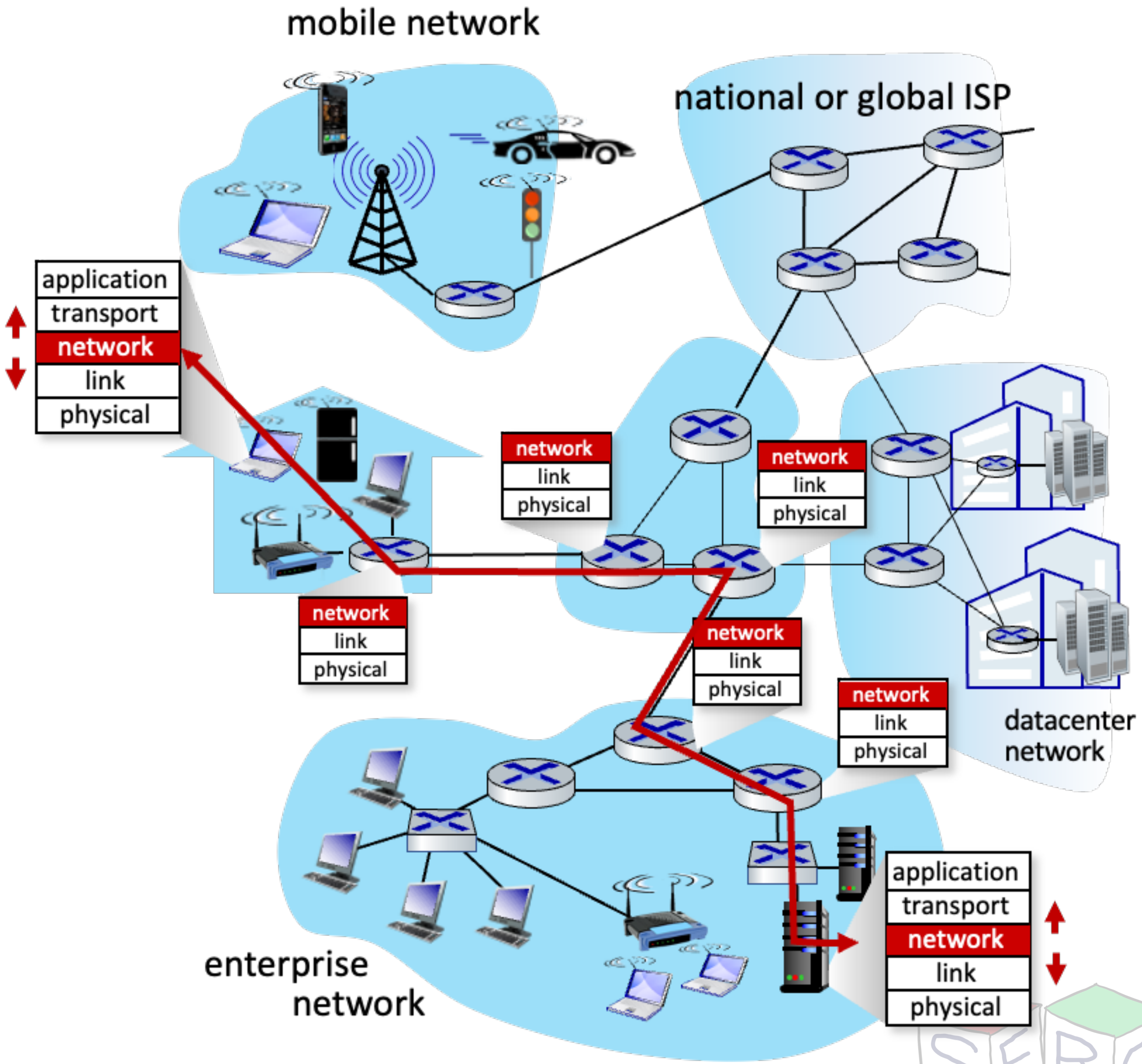
# Network Layer - Functionalities

## Addressing

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

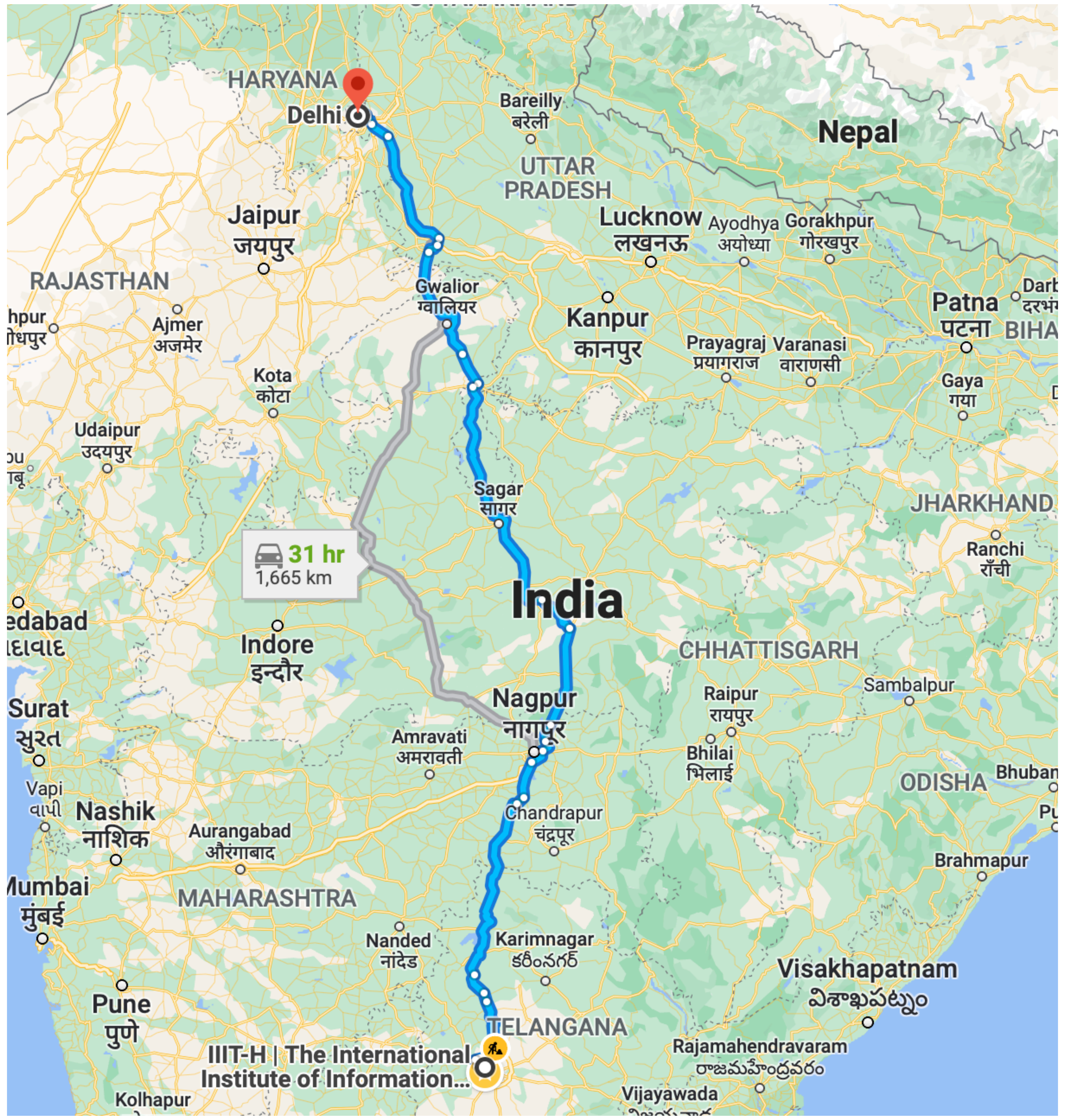
## Route Determination

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



# Two Key Network Layer Functions

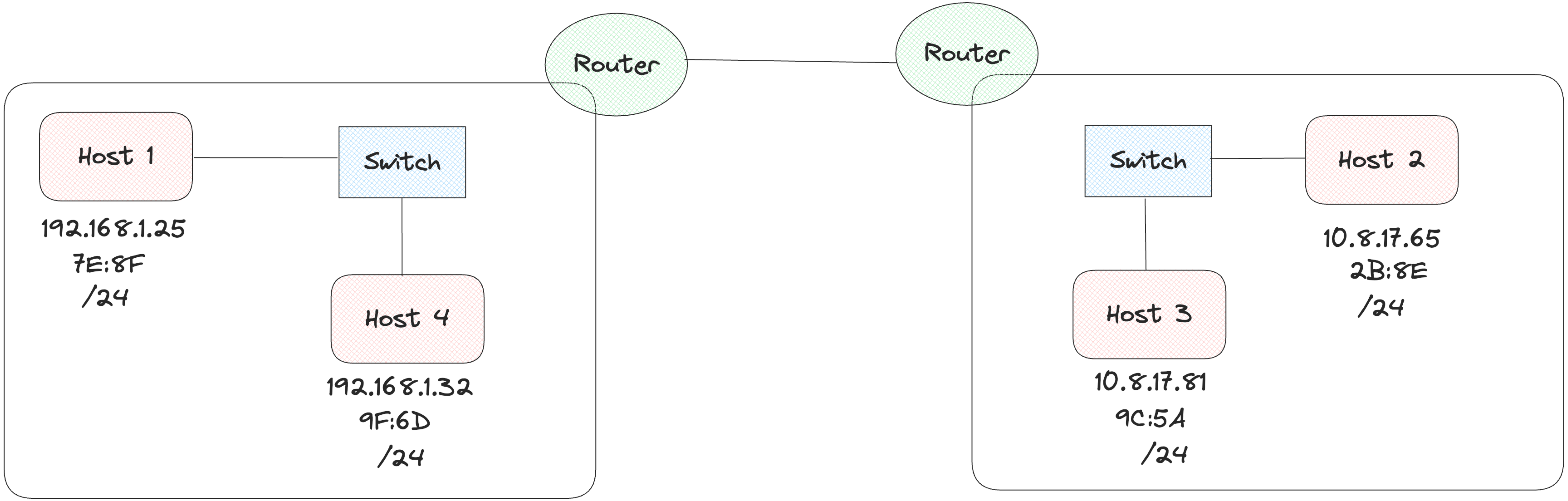
Forwarding (Interchanges)



Routing (Source to destination route)

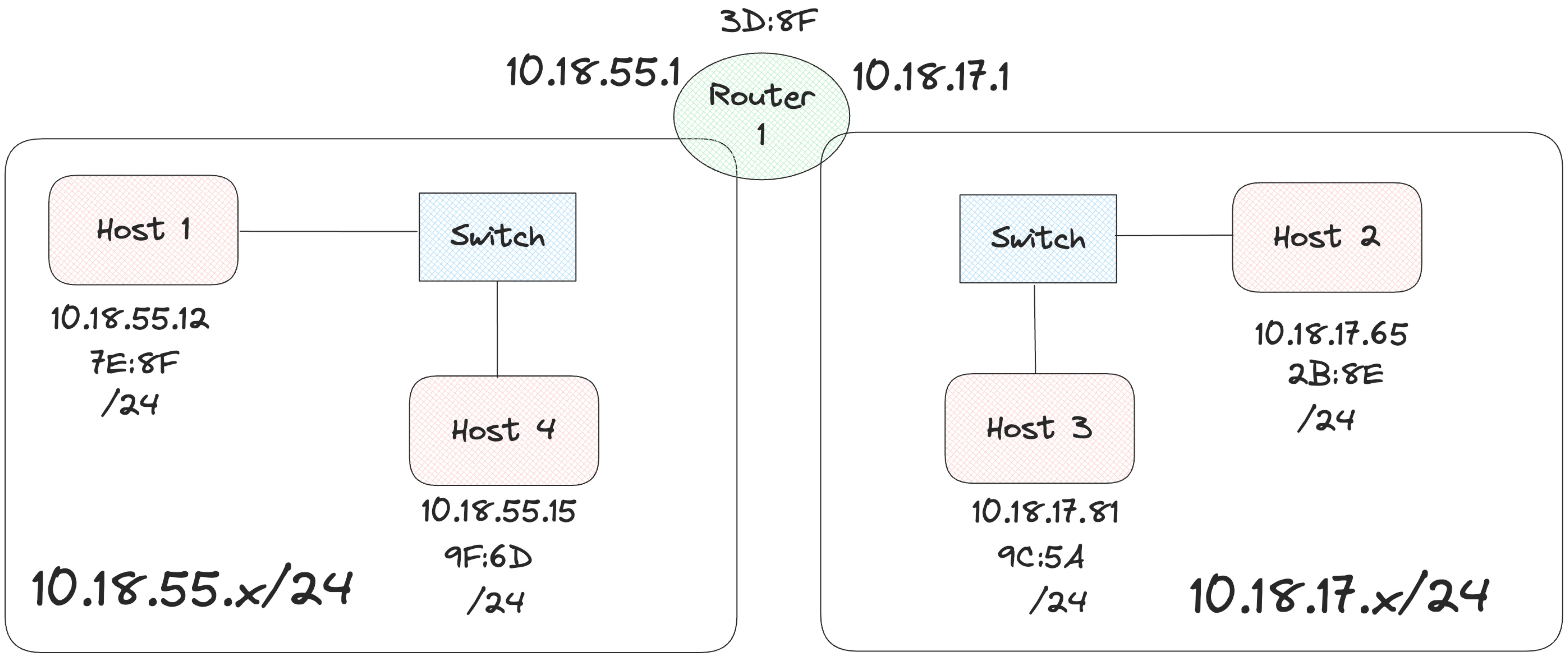
- **Forwarding**
  - Move packets from routers input link to output link
- **Routing**
  - Determine route to be taken by packets from source to destination

# Routers - Devices in L3 that makes things happen



- Routers are connected to a network (have IP and MAC)
- Routers are node that **forwards packets** not explicitly addressed to itself
- Hosts are any nodes that are not a router (RFC 2460) - They can discard packets

# More about Routers

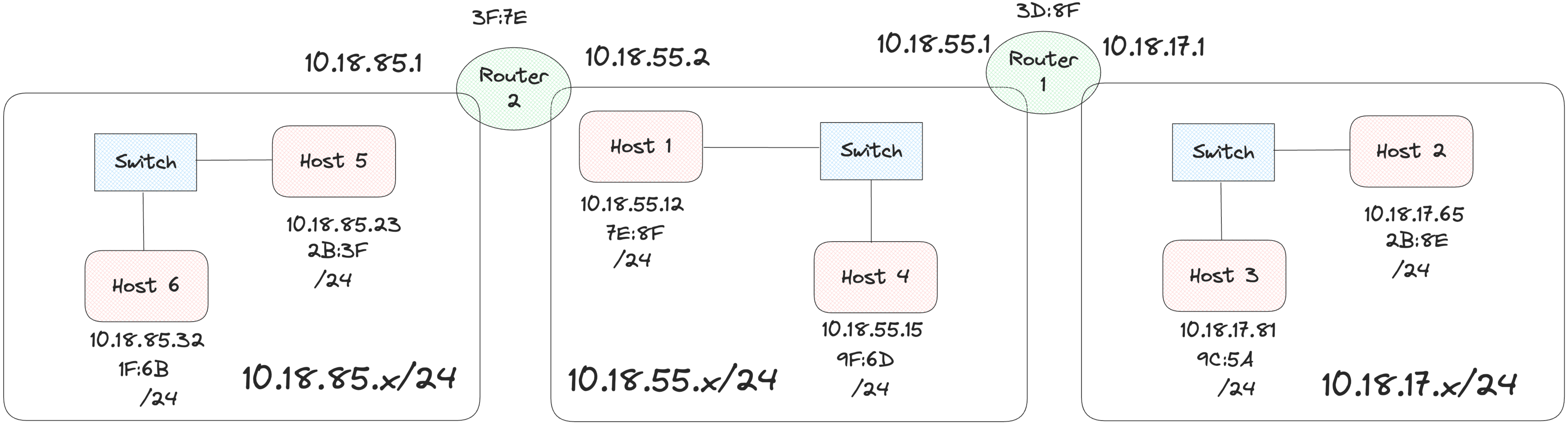


**Router 1 routing table**

| Destination | Interface |
|---|---|
| **10.18.55.x/24** | Left |
| **10.18.17.x/24** | Right |

- Routers maintain a map of all networks they know about
  - **Routing Table**: Used by routers as a map to connect to the networks they know about given the destination IP
  - **Note** the table is just a sample, in reality instead of left and right it can be eth/0, eth/1, etc.

# Simple Example



| Destination | Interface |
|---|---|
| 10.18.85.x/24 | Left |
| 10.18.55.x/24 | Right |

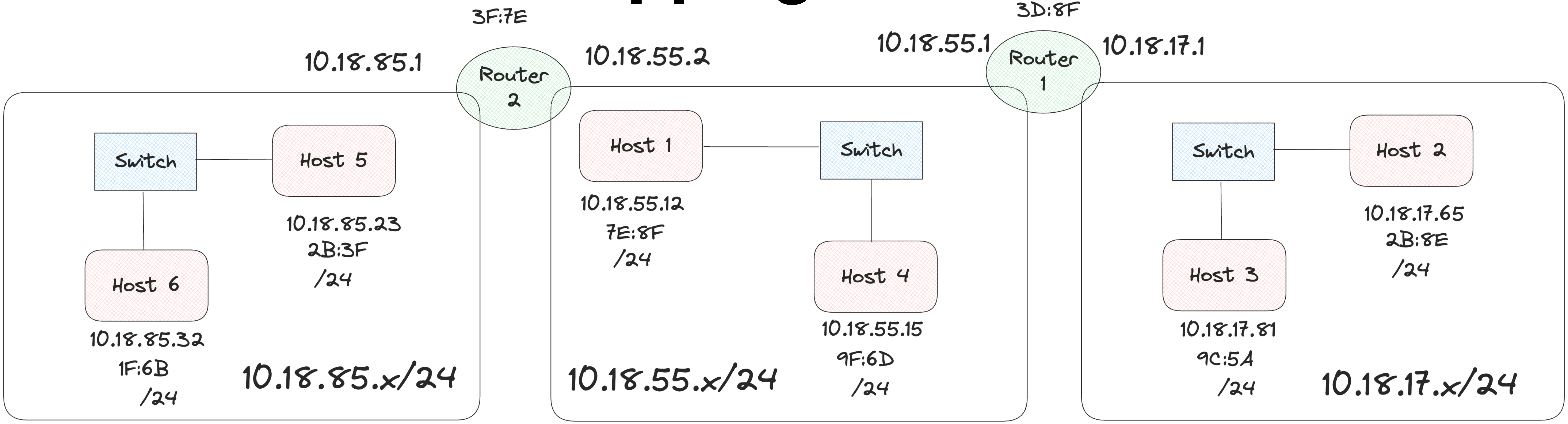
**Router 2 routing table**

| Destination | Interface |
|---|---|
| 10.18.55.x/24 | Left |
| 10.18.17.x/24 | Right |

**Router 1 routing table**

**How can Host 6 communicate with Host 2?**

# Admin can add mappings to table!



| 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 |

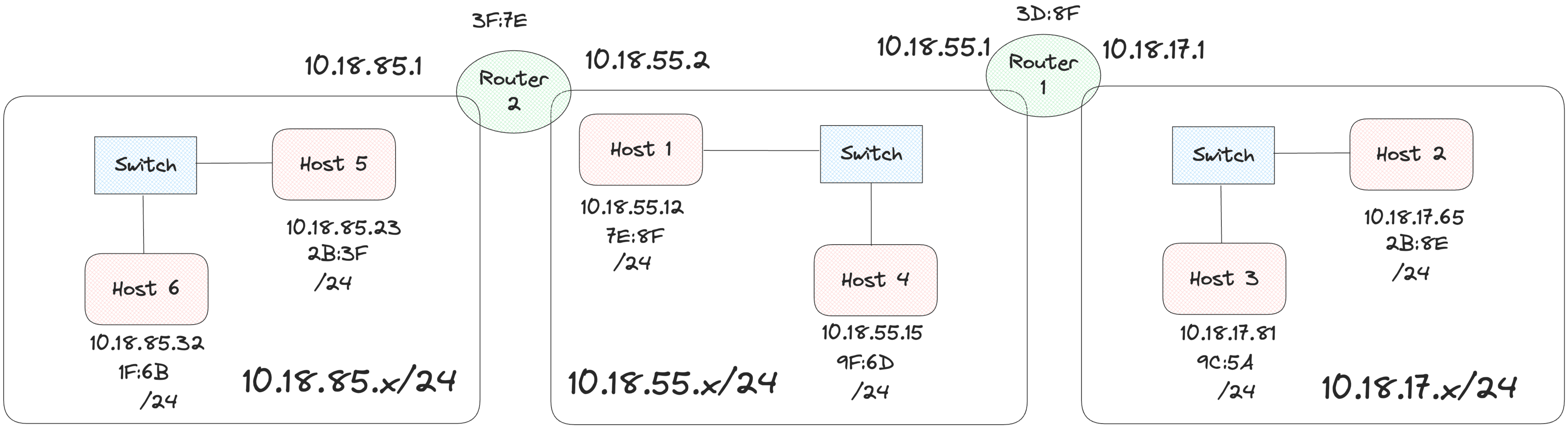
**Router 2 routing table**

| 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**

**Admin can add it!!**

# What if Routers can learn by themselves?



| Type | Destination | Interface |
|---|---|---|
| DC | 10.18.85.x/24 | Left |
| DC | 10.18.55.x/24 | Right |
| Dyn | 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 |
| Dyn | 10.18.85.x/24 | 10.18.55.2 |

Router 1 routing table

Dynamic discovery
And addition

# But how to send data to the host

- Routers have an IP and MAC
- Routers have routing tables - Map to every network
- Routers also have ARP tables
  - Mapping of L3 address to L2 address
  - Anything in network with IP will have an ARP table
  - ARP table is populated on the fly - Why?
  - Routing tables needs to be ready apriori - Routers may drop packets if IP is not known

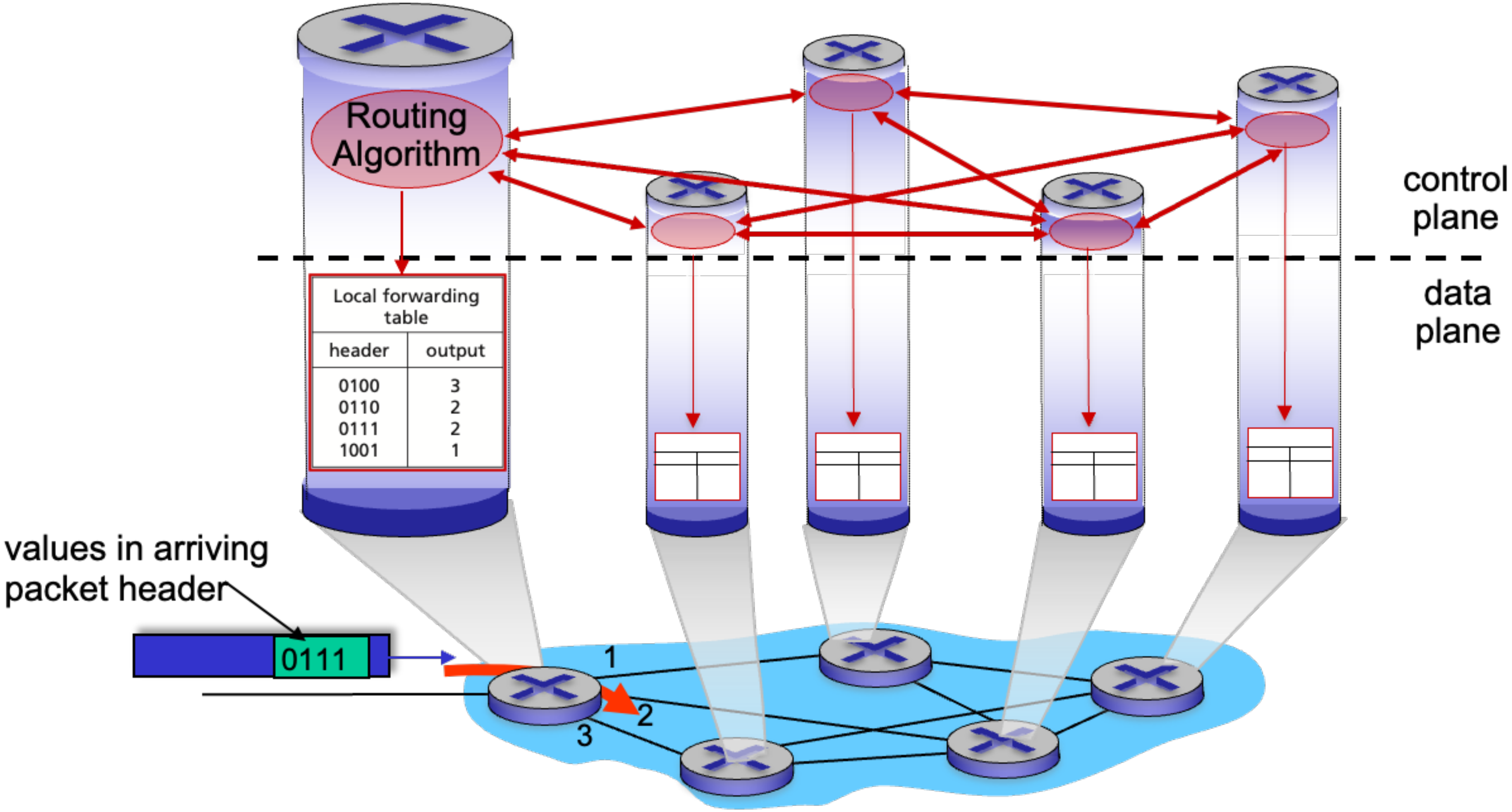
# Routing Table

- Three methods to populate routing table
  - **Directly connected:** Networks to which the router is directly attached to
  - **Static routes:** Routes manually provided by an administrator
  - **Dynamic routes:** Routes automatically learned from other routers
    - Routers communicate with each other to know about different networks
    - Different protocols: OSPF, BGP, EIGRP, IS-IS
    - Used by routers to inform about the different networks they are connected to

# Network Layer: Data Plane and Control Plane

- One can divide network layer functions into two planes: Data and control
- **Data plane:** Local per router function
    - Determines how datagram arriving on router input port is forwarded to router output port
- **Control plane:** Network wide logic
    - Determine how datagram is routed along end to end path form source to destination
    - Two approaches: Traditional routing algorithms, or Software defined networking (SDN)

# Traditional Control Plane Approach



Local forwarding table

| header | output |
|---|---|
| 0100 | 3 |
| 0110 | 2 |
| 0111 | 2 |
| 1001 | 1 |

# How does one router know whom to send to?

- Routers have forwarding table consisting of routes
- But there are **billions of destinations** - Not everything can be stored in each router!!!
- Sending so many links with each other can itself bring down the network
- There are two parts to it:
  - Internet: network of networks
  - Each network admin may want to control routing in its own network

# Protocols used in routing

- Intra-AS routing protocols:
    - **OSPF (Open Shortest Path First) Protocol**
        - Classic link state routing (Dijkstra’s algorithm)
    - Others include: RIP, EIGRP  (RIP: Routing Information Protocol)
- Inter-AS routing protocols:
    - **BGP (Broader Gateway protocol)**
        - Path vector protocol
    - Considered as “glue that holds internet together”

# 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

Gateway Routers
AS 1
Router A.1
Router A.3
Router A.5
Router A.5
AS 2
Router A.2
Router A.1
Router A.3
Router A.5
AS 3
Router B.1
Router B.4
Router B.3

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

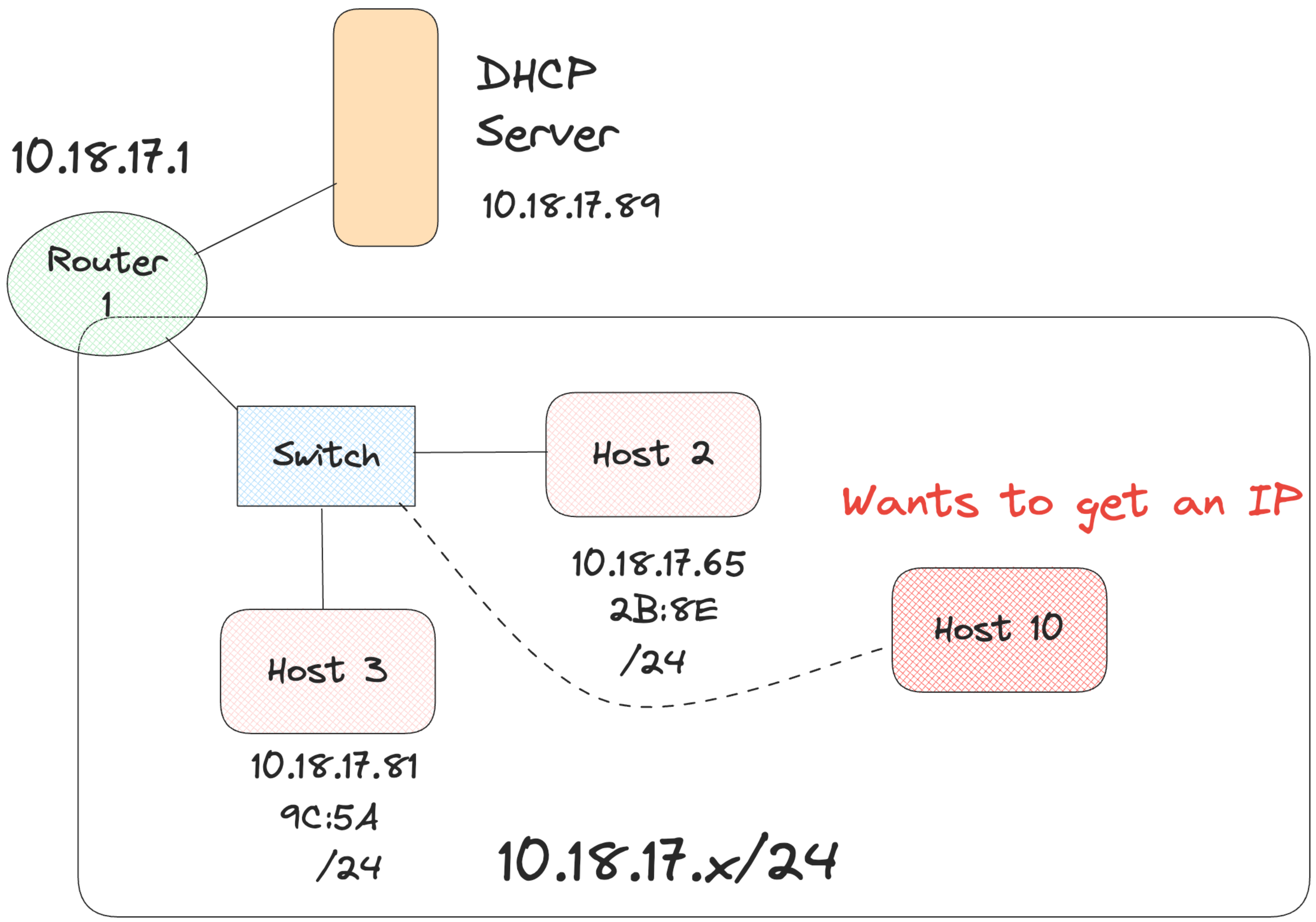
Inter AS routing protocols determine entries for destination outside AS

Eg: BGP, EGP

# Taking a step back: How to get IP address?

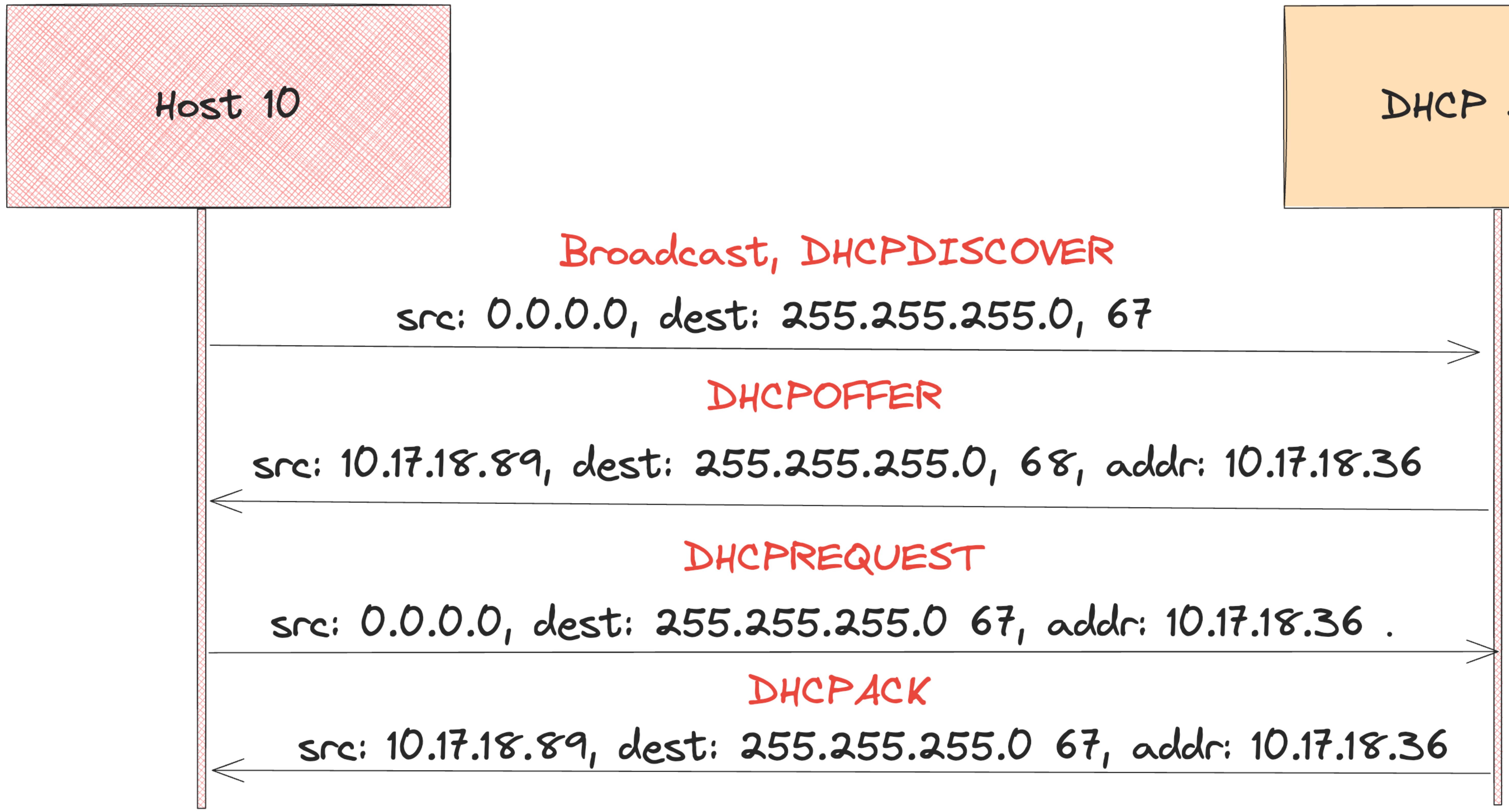
- Two questions needs to be answered:
  - How does host get IP address within its network?
  - What about the network address?
- How does host get an IP address?
  - Hard-coded by sysadmin in config file (e.g., /etc/rc.config in UNIX)
  - **DHCP:** Dynamic Host Configuration Protocol - Dynamically get IP address when joining from a server

# Working of 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)

# Working of DHCP



Host 10

DHCP Server

Broadcast, DHCPDISCOVER

src: 0.0.0.0, dest: 255.255.255.0, 67

DHCPOFFER

src: 10.17.18.89, dest: 255.255.255.0, 68, addr: 10.17.18.36

DHCPREQUEST

src: 0.0.0.0, dest: 255.255.255.0 67, addr: 10.17.18.36 .

DHCPACK

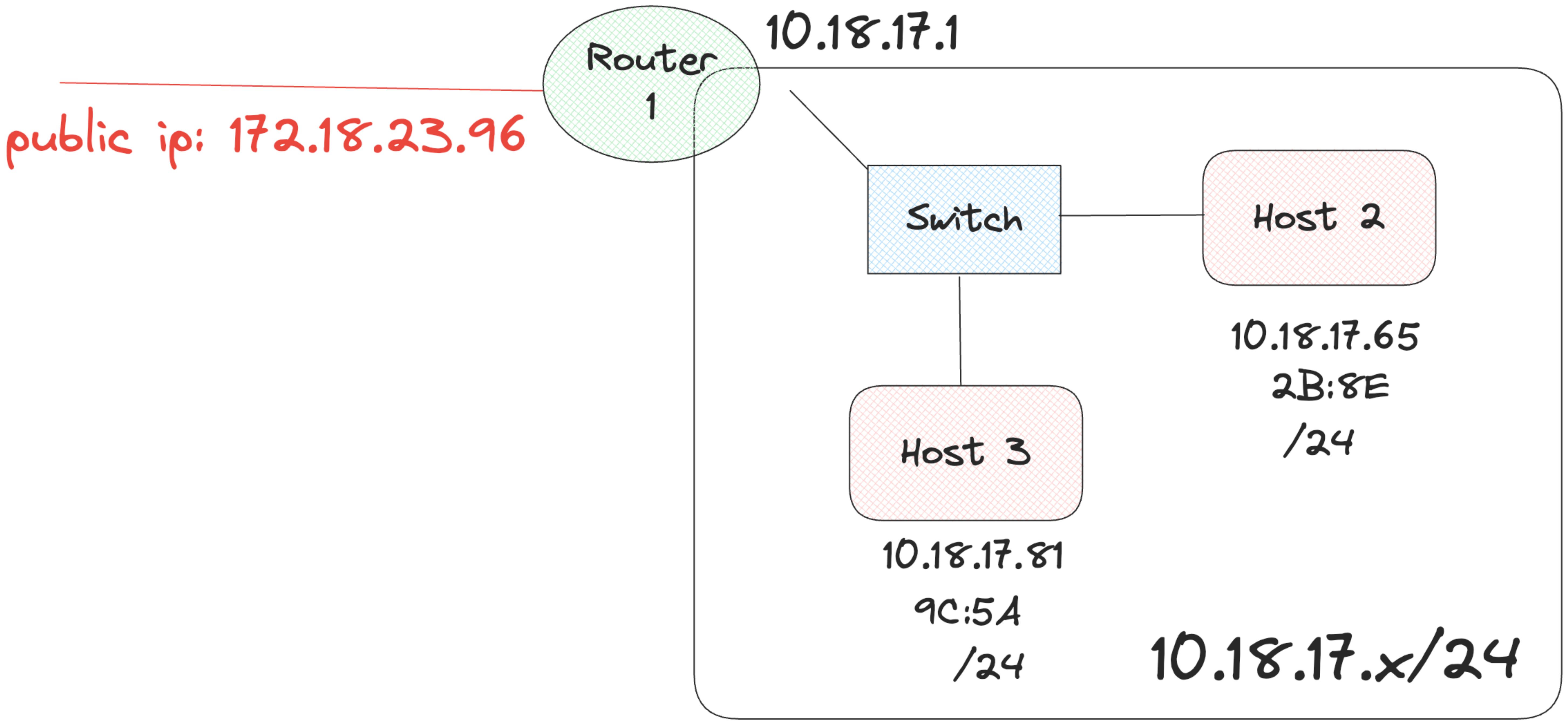
src: 10.17.18.89, dest: 255.255.255.0 67, addr: 10.17.18.36

- DHCPDISCOVER will be broadcasted to all the DHCP servers
- The IP address offer will be given by multiple DHCP servers, client chooses one (first response) and broadcasts the acceptance
- DHCP server can also give details like address of DNS server, address of first hop router, network mask, etc.

# How to get IP Address

- ISP gets IP address block from ICANN (Internet Corporation for Assigned Names and Numbers) - http://ican.org
  - Allocates IP addresses through 5 regional registries (RRs)
  - There are not enough IPV4 addresses - Last chunk was allocated to RRs in 2011
  - IPV6 - 128 bit address space
- We are still able to function with IPV4 due to NAT (Network Address Translation)

# Network Address Translation (NAT)

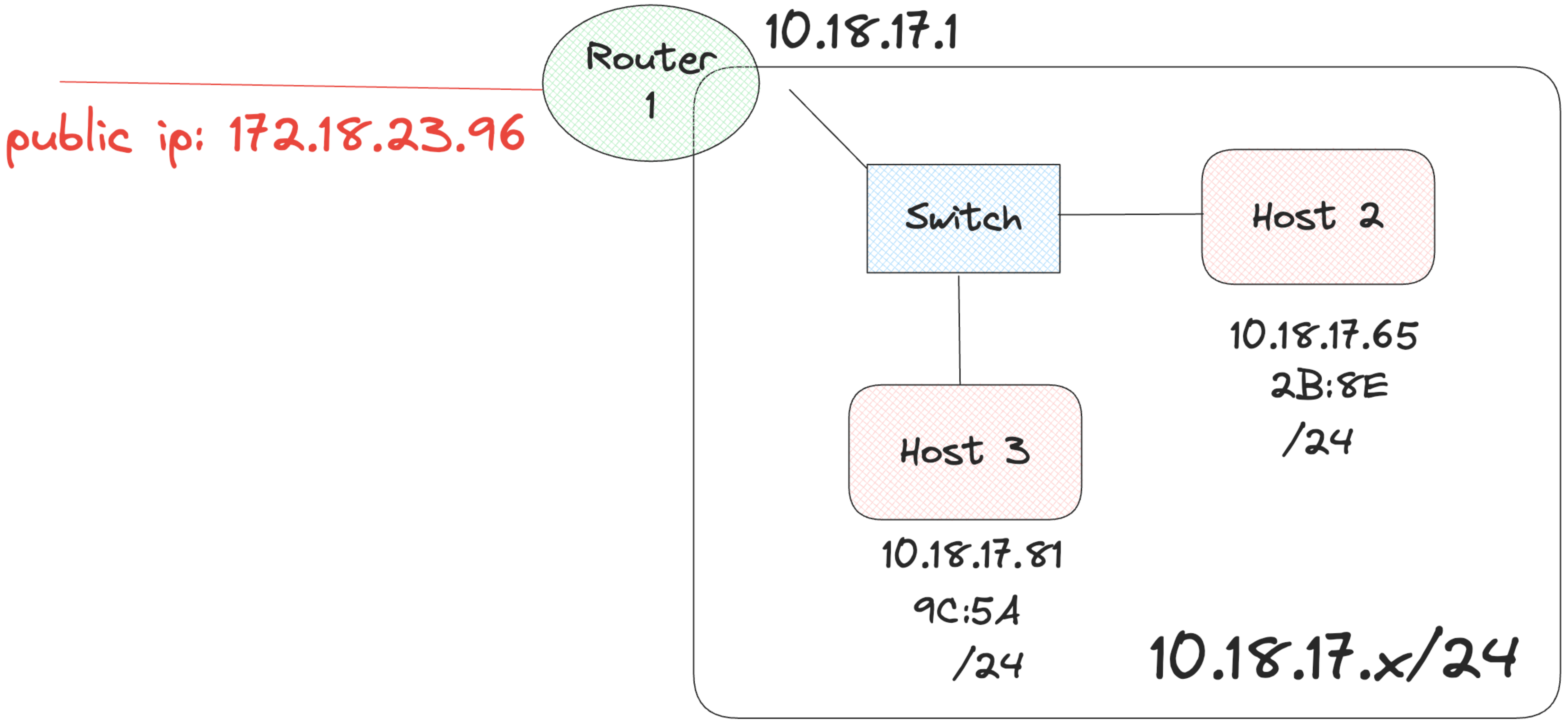


- All devices in the network share just one IPV4 address as far as the outside world is concerned
  - They can still communicate with different hosts outside the network with one public IP
  - How is that possible and how to make this work?

# Network Address Translation (NAT)

NAT Translation Table

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



- 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

# Putting it together

1. Machine in a network gets an IP using DHCP

2. Process makes use of app protocol (HTTP, SMTP, etc.) to connect to process in abc.com

3. First does a DNS query to get IP address

Leverages UDP

4. Make use of TCP/UDP to send data to IP address and a port

5. IP address is used to find the route -> next router (BGP, OSPF)

6. Get the MAC address of the next node using ARP and send data

SCAN ME

## Thank you

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