CS3.301 Operating Systems and Networks

Virtualization - Process

Karthik Vaidhyanathan

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

SERC
Software Engineering Research Centre

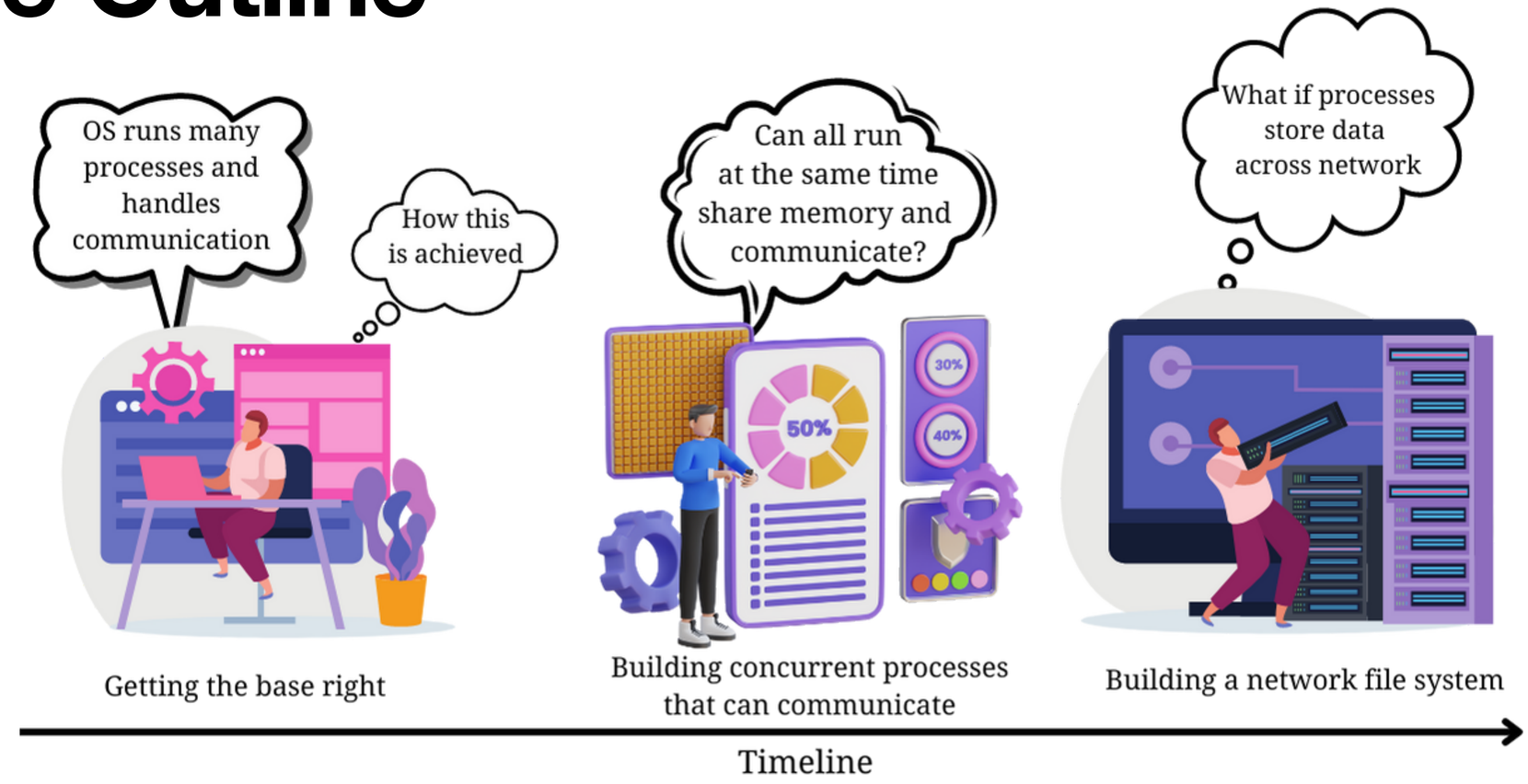
Acknowledgement

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.

Course Outline

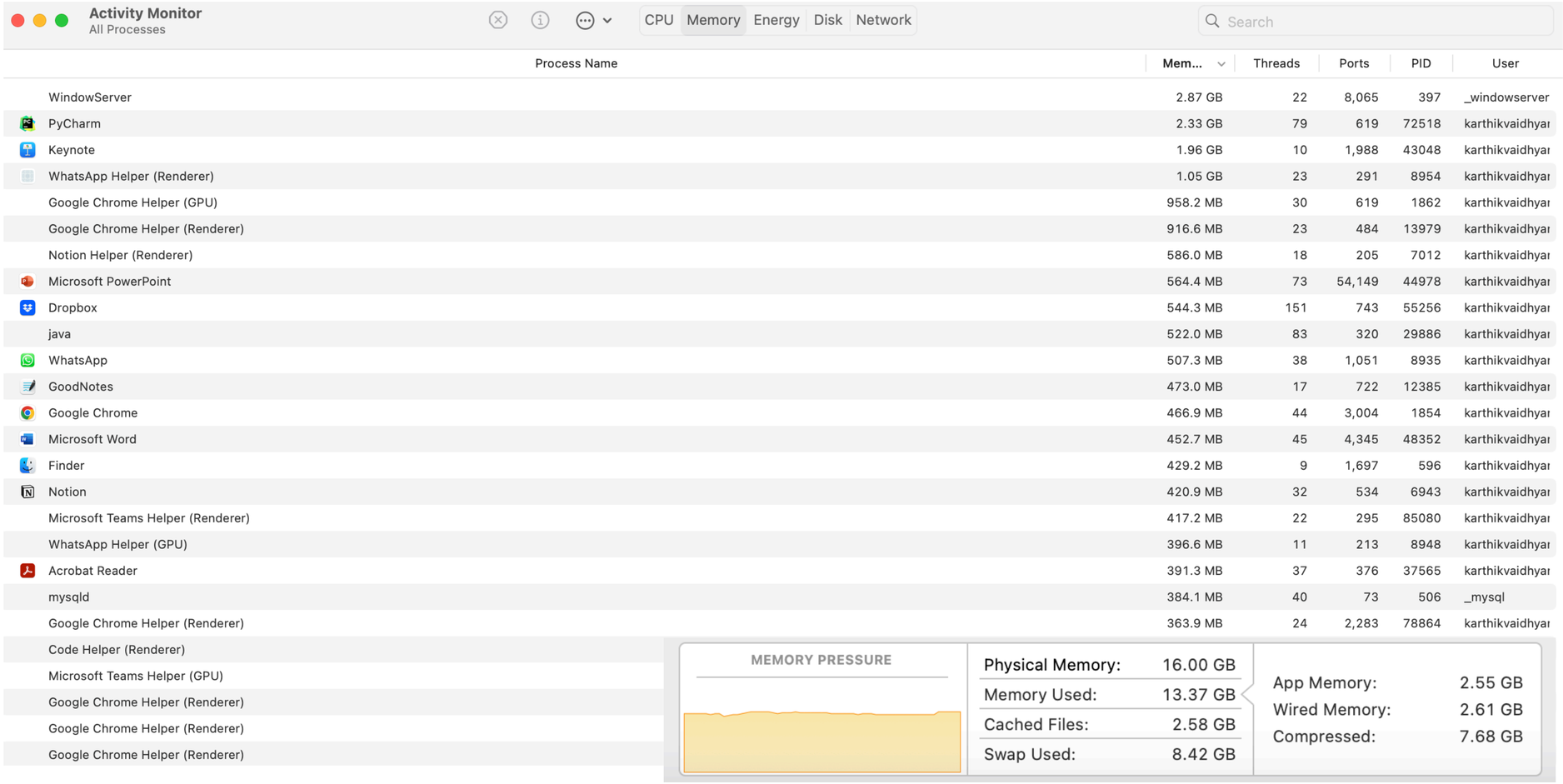


This Course

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

Many processes run at the same time!

- How many processes are currently running in your machine?



What is a Process?

- A Program is nothing but code
- Processes are **running program**
- There can be more than one process that are created per program

Process Virtualization

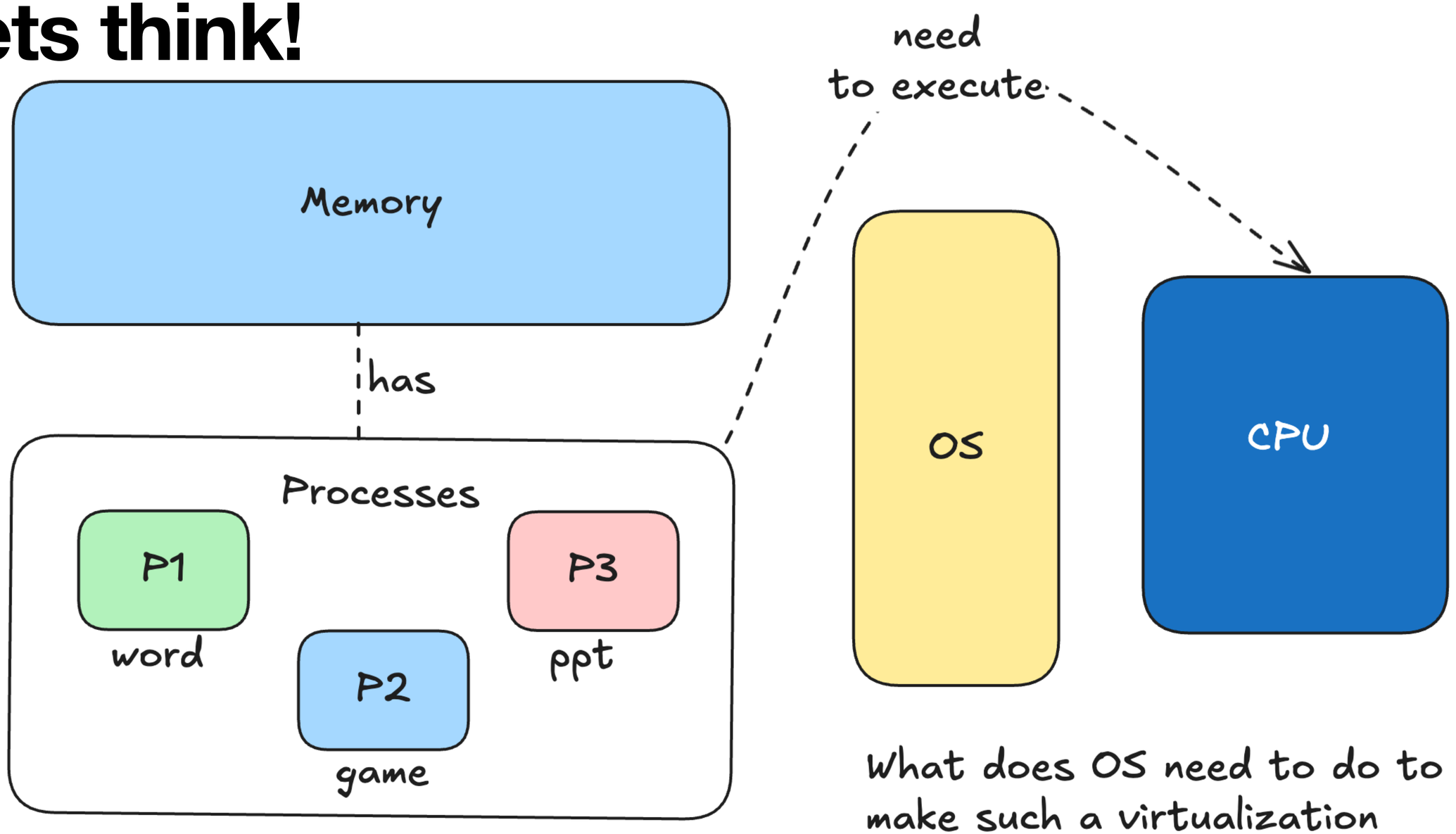
- Each process feels that it has its own CPU
- Even in Single core machines - There can be multiple process that run at the same time
- How is CPU handling this?

With limited CPU can we create an illusion that Endless CPU's are available?

OS achieves this using Virtualization of the CPU

Question: Can you of think of how such thing can be done?

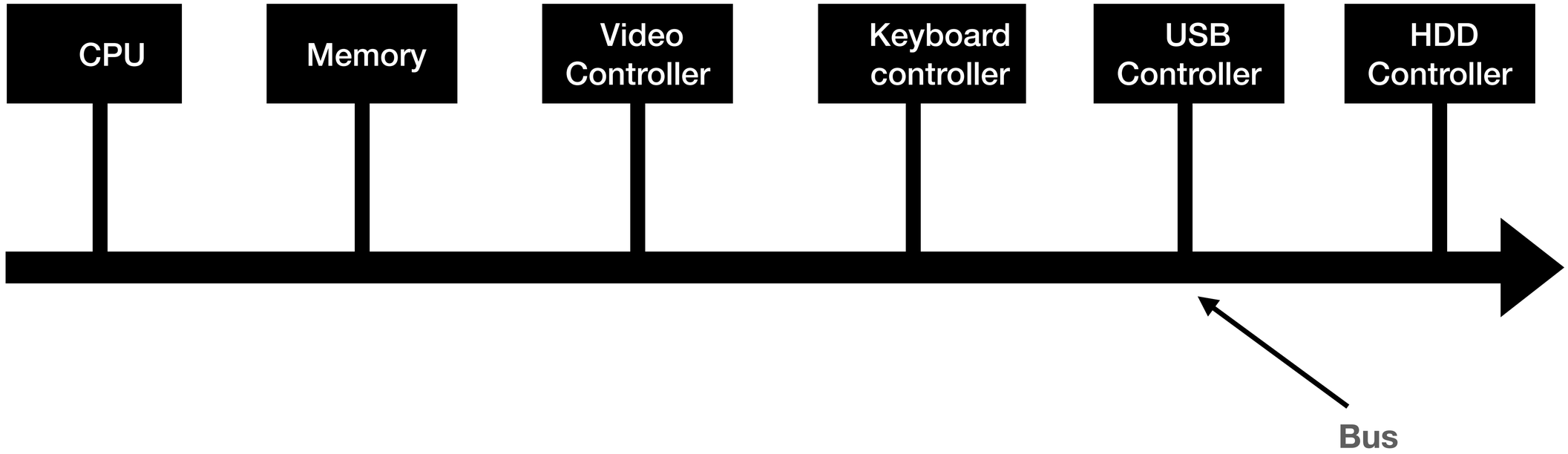
Lets think!



Each process is made to feel that it has its own CPU

What does OS need to do to make such a virtualization happen?

Some Prerequisite



As we go more away from CPU, the more time it takes

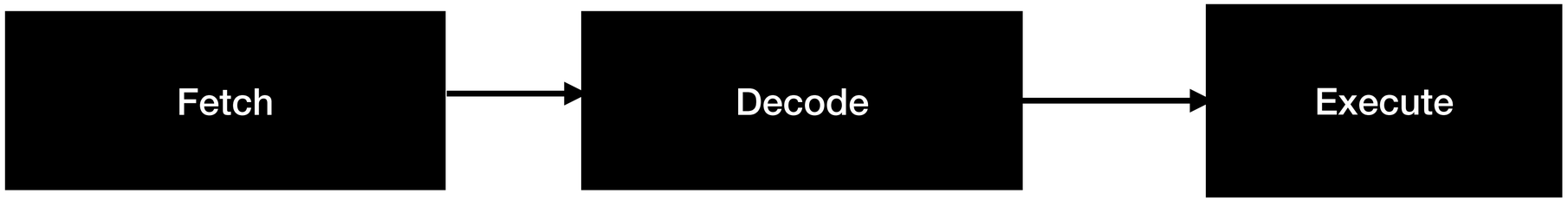
Some Prerequisite - Computer Hardware

- CPU contains some registers
  - Temporary registers
  - Program Counter (PC), Stack pointer (SP), Data register, address register,..
- Some key registers
  - Program counter - Points to the next instruction
  - Stack pointer - Points to top of the stack in the memory
  - Program Status word - Status of current state of CPU and program (condition bits)

Some Prerequisites

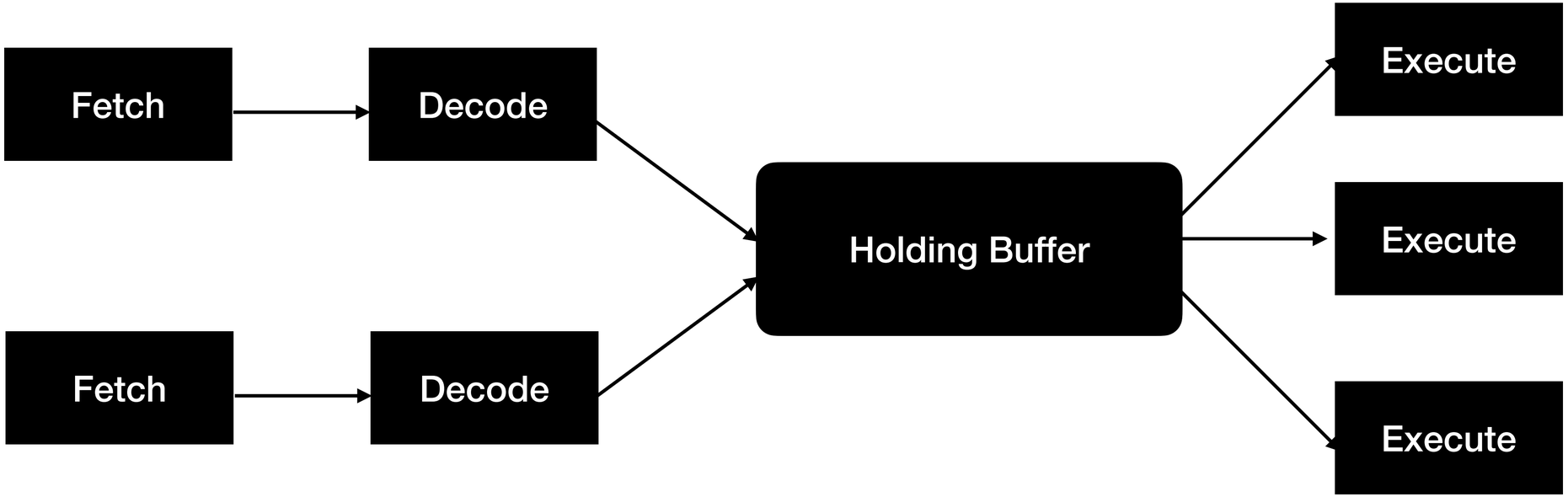
How does CPU execute a program?

- Three stage pipeline



Question: Do you believe that the current hardware structure is similar to this?

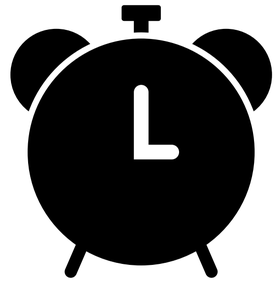
Some Prerequisite - Computer Hardware



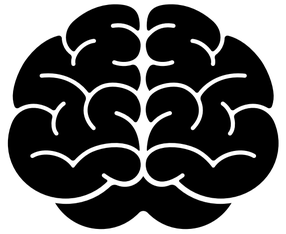
Superscalar CPU

How to make it at software level?

- **We do need support from the hardware**
  - Some mechanism to switch
  - Eg: Each process runs for a particular time and then we switch
  - Low-level mechanism (Context Switch)



- **We also need some intelligence in the software**
  - Some algorithm that can intelligently decide
  - Policies for switching



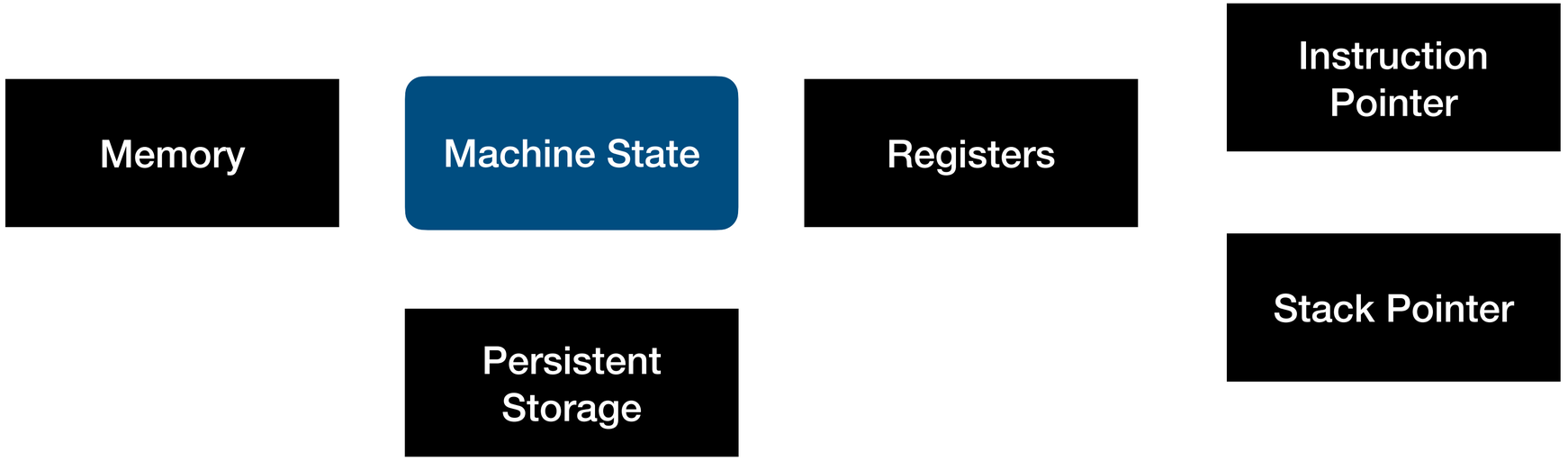
- Basically we need - **low level mechanisms** and **policies** (CPU Scheduler)

What Constitutes a Process?

Lets make it clear - Process is nothing but running program!!

- The Characteristics that make up a process (State)
  - What parts of the machine are important for execution?
- The most obvious component - Memory! Why?
  - Instructions lie in the memory, data (reads and writes) is in the memory
  - **Address space** is part of the process
- What else does a running program need?

What Constitutes a Process?



- **Memory** - address space (Memory that the process can address)
- **Instruction pointer, program counter** - which instruction is executed
- **Stack pointer**- local variables, functions and return addresses
- **Peristent storage** - I/O information

What Constitutes a Process?

- **Unique Identifier (Process ID)**
- **Memory Image**
  - Code and data (static)
  - Stack and Heap (Dynamic)
- **CPU Context: Registers**
  - Program Counter
  - Current Operands
  - Stack Pointer
- **File Descriptors**
  - Pointers to open files and devices

Creation of a Process by OS

- **Load program into memory**
  - Initially program resides on the disk
  - OS does **lazy loading**
- **Allocate runtime stack**
  - Use for local variables
  - Function parameters and return arguments

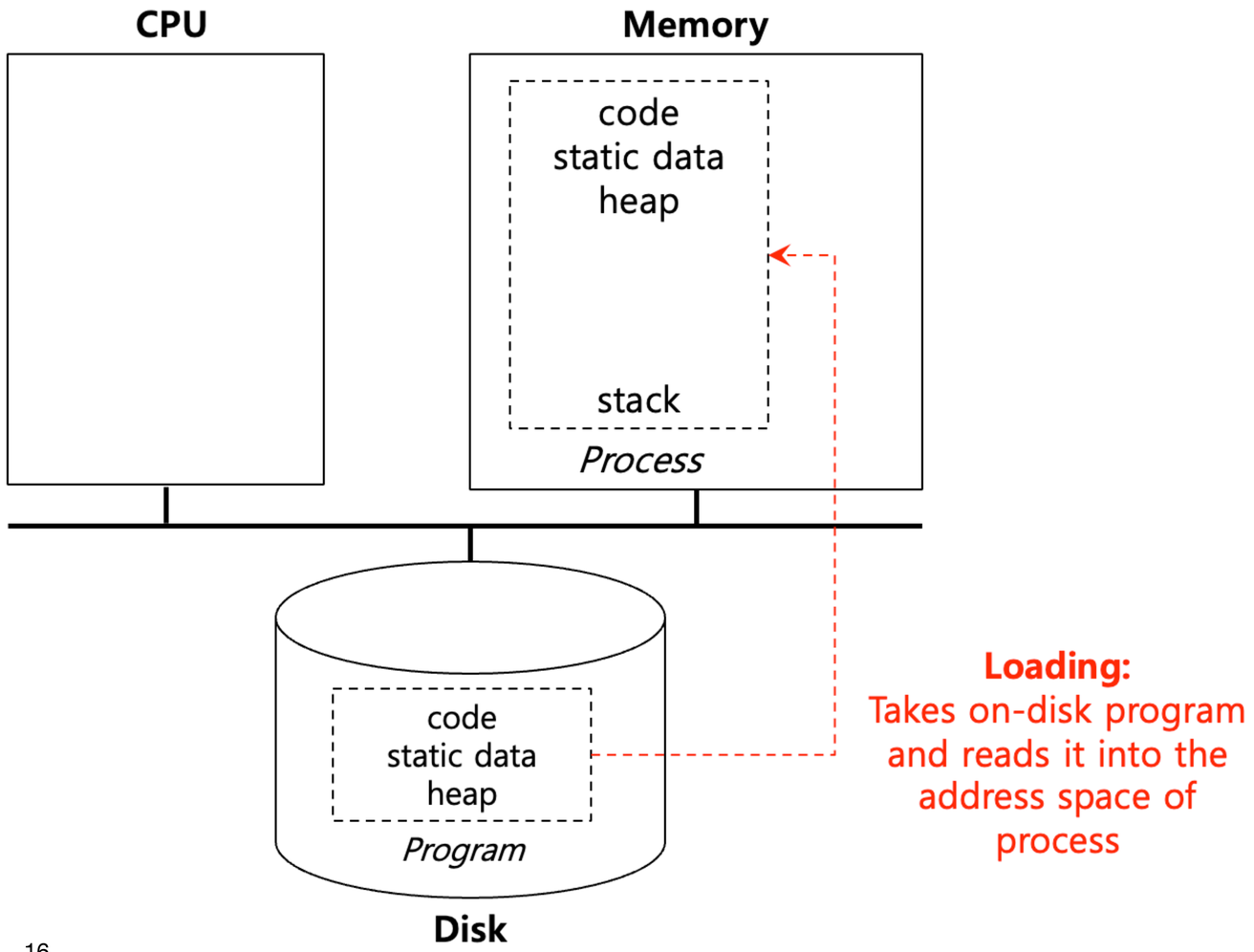


Image from OS lecture by Youjip Won

Creation of a Process by OS

- **Creation of Program heap**
  - Used for dynamically allocated data
  - malloc() and free()
- **Basic file setup**
  - STDIN, OUT, ERR
- **Initialise CPU registers**
  - PC to the first instruction
- **Start the program**

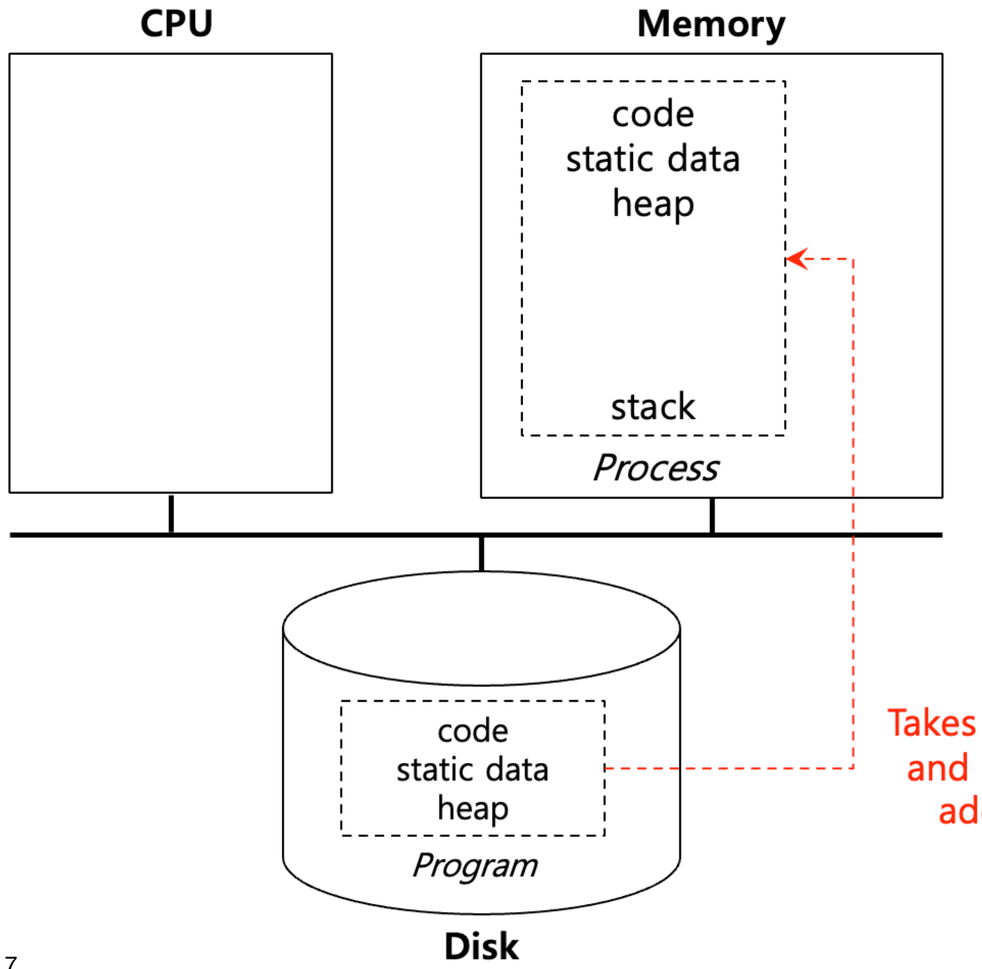


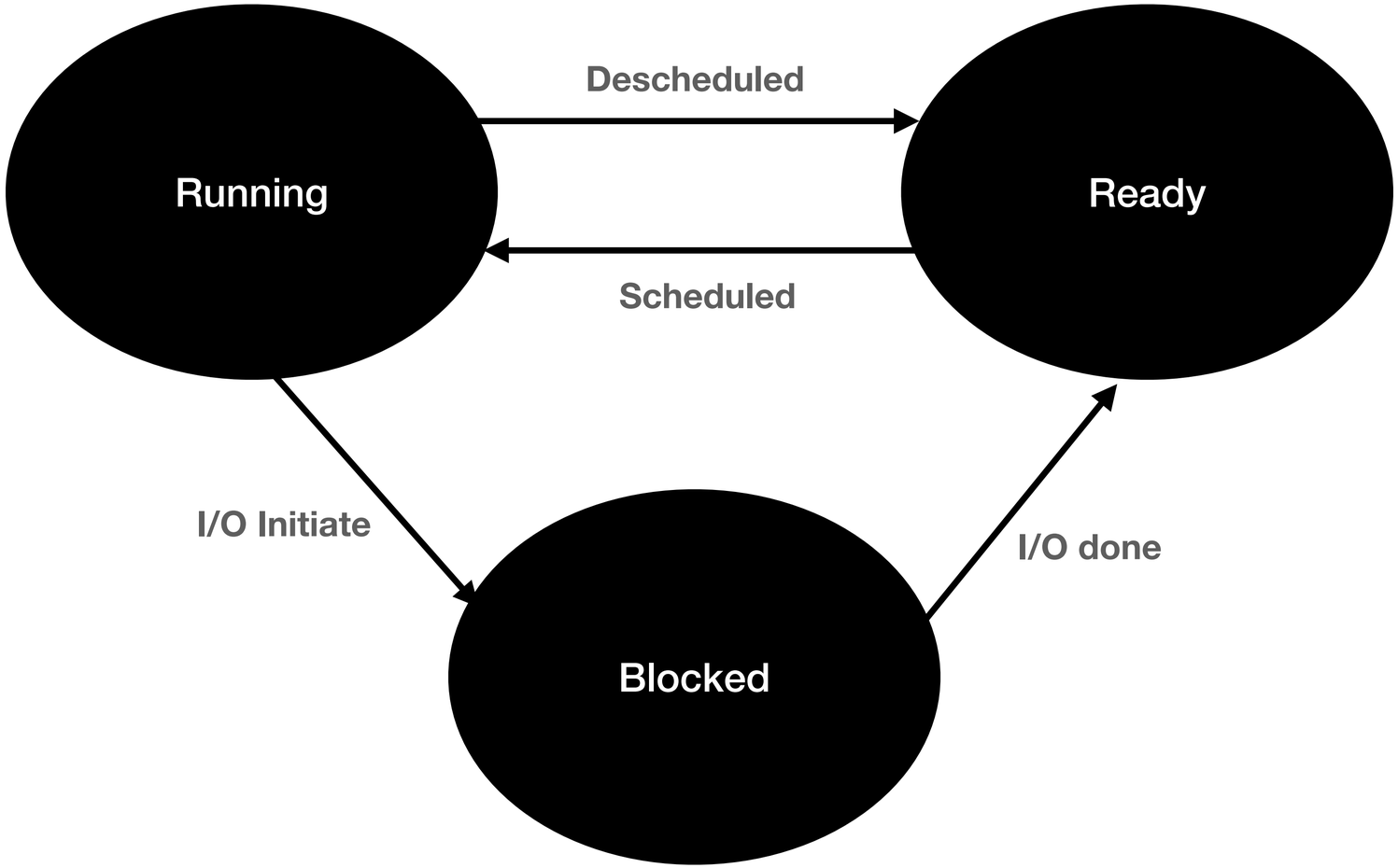
Image from OS lecture by Youjip Won

States of the Process

- At any point process can be in one of the following states
  - **Running** - Its running on the processor
  - **Ready** - Ready to run
  - **Blocked** - Not ready to run, something else is running
    - Any reason that you can think of?
  - **Think of I/O call** - Wait what does that mean?

States of the Process

Process State Transitions



Lets look at an Example

Time	Process 0	Process 1	What's happening
1	Running	Ready	
2	Running	Ready	
3	Running	Ready	Process 0 initiates I/O
4	Blocked	Running	Process 0 is blocked, 1 runs
5	Blocked	Running	
6	Blocked	Running	I/O of process 0 is done
7	Ready	Running	Process 1 is done
8	Running	-	Process 0 is done

How to store Metadata? - Use data structures

- Need for some mechanism to store the state of the process
- **Remember:** OS is a software
  - It leverages data structures to store the information
  - OS makes use of data structure called, **process list**
  - What to store inside each? - Process Control Block (PCB)
    - Process id? - Identification of the process
    - State of the process - ready, running or blocked
    - Address space of the process - the registers

Process List

Xv6 Operating System

Teaching OS developed by MIT - Replicate basic Unix

https://pdos.csail.mit.edu/6.828/2012/xv6.html

Process Structure in Xv6

```
// the information xv6 tracks about each process
// including its register context and state
struct proc {
    char *mem;                  // Start of process memory
    uint sz;                    // Size of process memory
    char *kstack;               // Bottom of kernel stack
                                // for this process
    enum proc_state state;      // Process state
    int pid;                    // Process ID
    struct proc *parent;        // Parent process
    void *chan;                 // If non-zero, sleeping on chan
    int killed;                 // If non-zero, have been killed
    struct file *ofile[NOFILE]; // Open files
    struct inode *cwd;          // Current directory
    struct context context;     // Switch here to run process
    struct trapframe *tf;       // Trap frame for the
                                // current interrupt
};
```

Process Structure in Xv6

```
// the registers xv6 will save and restore
// to stop and subsequently restart a process
struct context {
    int eip;    // Index pointer register
    int esp;    // Stack pointer register
    int ebx;    // Called the base register
    int ecx;    // Called the counter register
    int edx;    // Called the data register
    int esi;    // Source index register
    int edi;    // Destination index register
    int ebp;    // Stack base pointer register
};

// the different states a process can be in
enum proc_state { UNUSED, EMBRYO, SLEEPING,
                  RUNNABLE, RUNNING, ZOMBIE };
```

Thank you

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

CS3.301 Operating Systems and Networks

Process Virtualisation - API and Mechanisms

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 Remzi et al.
- Modern Operating Systems, Tanenbaum et al.
- Other online sources which are duly cited

What features should the OS Provide?

Consider that we should be able to run multiple processes!

- **Create a process**
  - Double click and something just runs
- **Destroy a process**
  - Force quit, task manager -> end process
- **Wait**
  - Wait before running
- **Suspend**
  - Keep the process in pause and resume (eg: Debugging an application!)
- **Status**
  - Can we get some status of the process (task manager, system monitor, top)

How to make it happen? - Heard of APIs?

- Application Programming Interface - What's that?
  - How does a travel website get information about different flights and allows booking?
  - What about payment services?
- API allows different programs/applications to communicate with each other
- Provides a software interface for accomplishment
- Comes with detailed documentation

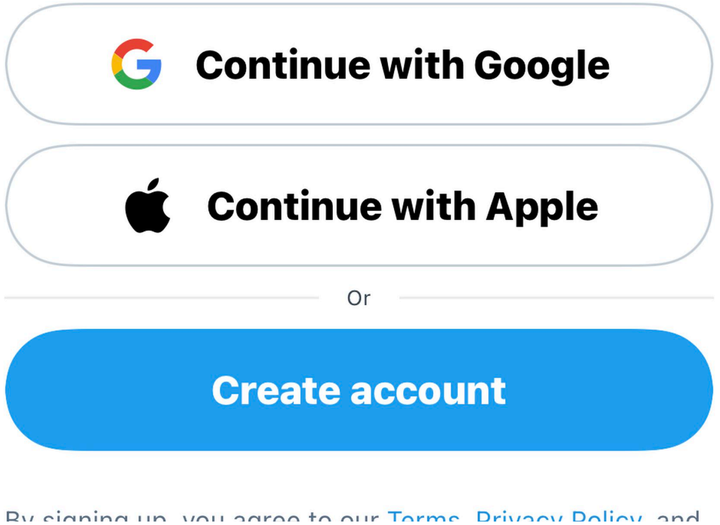
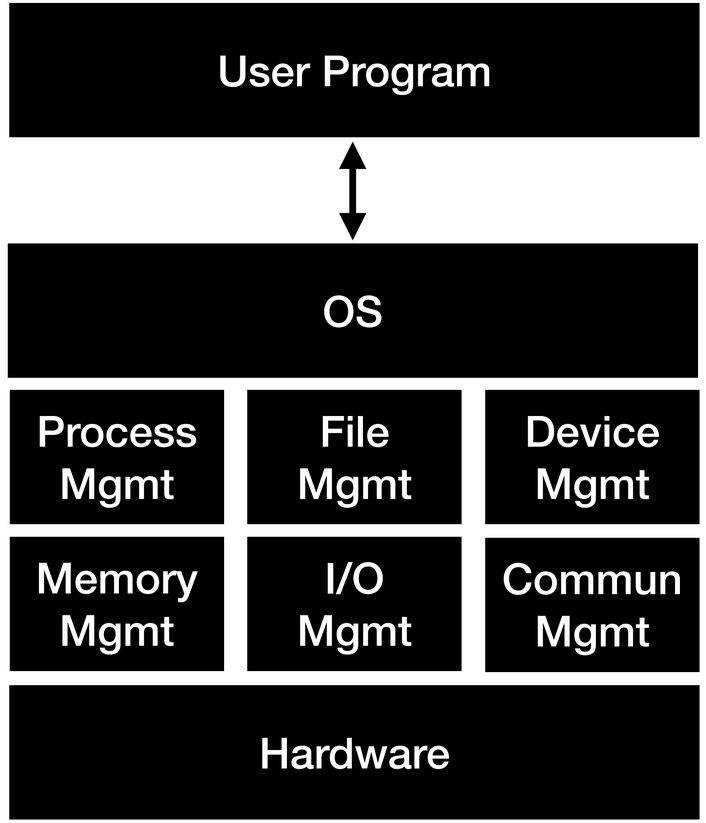


Image source: verge

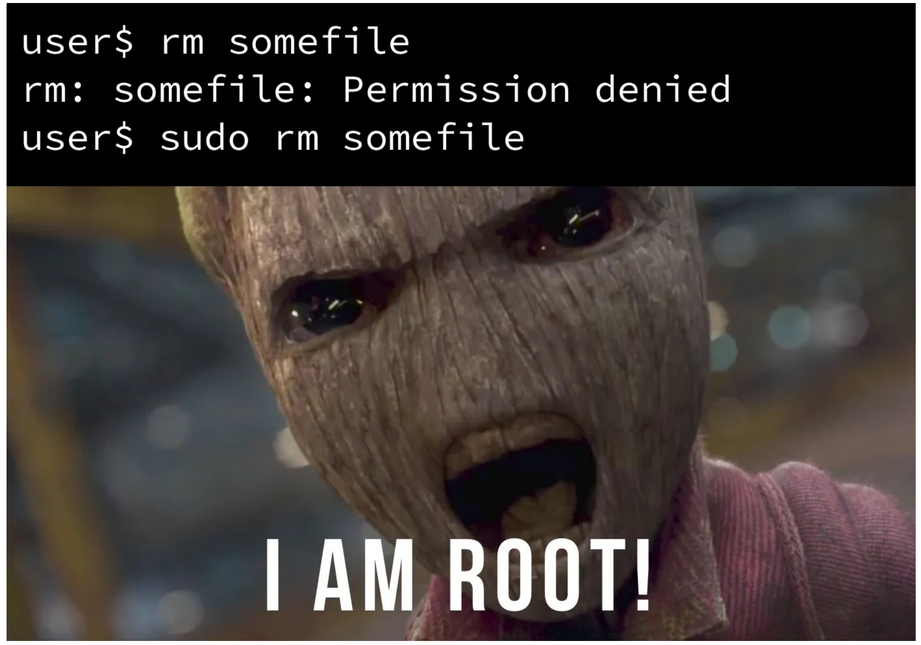
Does OS Provide API? - System Calls!

- Way for user program to interact with the OS
- OS provides some functions that can be leveraged by user programs
- Available in the form of “System calls”
  - Function call into OS code that runs at a higher privilege level
  - Think about access to hardware
- What if user wants to execute a process?



But you need Privileges!

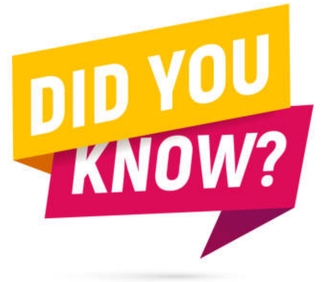
- What if a user gives a instruction to delete all files?
  - Should all the instructions be considered with equal priority?
  - When does the role of OS come in to the main picture?
    - Think about reading a file or writing a file - How to achieve it in C?
    - What if you just wanted to multiply two numbers?
    - What about the command to get list of available directories?
- Two modes of execution - **User mode** and **Kernel mode**

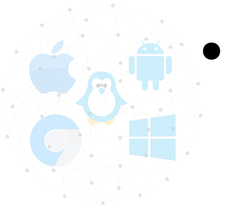


Source: reddit

For Each OS = Rewrite Programs?

- POSIX API (Portable Operating Systems Interface)
  - Standard set of System calls that an OS must implement
  - Most modern OS's are POSIX compliant
  - Ensures portability
- Programming language libraries abstract systems calls
  - printf() in C internally invokes write system call
  - User programs usually do not worry about system calls





Some System Calls

File Management	Process Management	Communication	Protection
fd =open(file,..)	fork()	Pipe()	chmod()
close(fd)	wait()	Shmget()	Unmask()
write(fd, …)	exec()	Mmap()	chown()
…	…	…	…

System Calls for Process (Unix)

System Call	Supports
fork()	Creates a new child process
exec()	Makes a process execute (runs an executable)
wait()	Causes a parent to block until child terminates
exit()	Terminates a process

- Many variants of the above calls exist
- **init** process is the ancestor of all processes

The Fork System Call

- A new process is created
  - Parent process image copy is made
- The new process is added to the list of processes and scheduled
- Parent and child start execution just after fork (with different return values)
- Parent and child execute and modify memory independently

The Wait API

- ***Wait()*** call blocks in parent until child terminates (options like ***waitpid()*** exists)
- Wait() also collects exit status of the terminated child process
  - Provides some visibility to the parent process
- Without wait, if process terminates - **Zombie process**
  - Exit status not collected by the parent
- Wait allows OS to reclaim the resources of the child - Prevent zombies
- What if Parent terminates before the child? - **Think!**

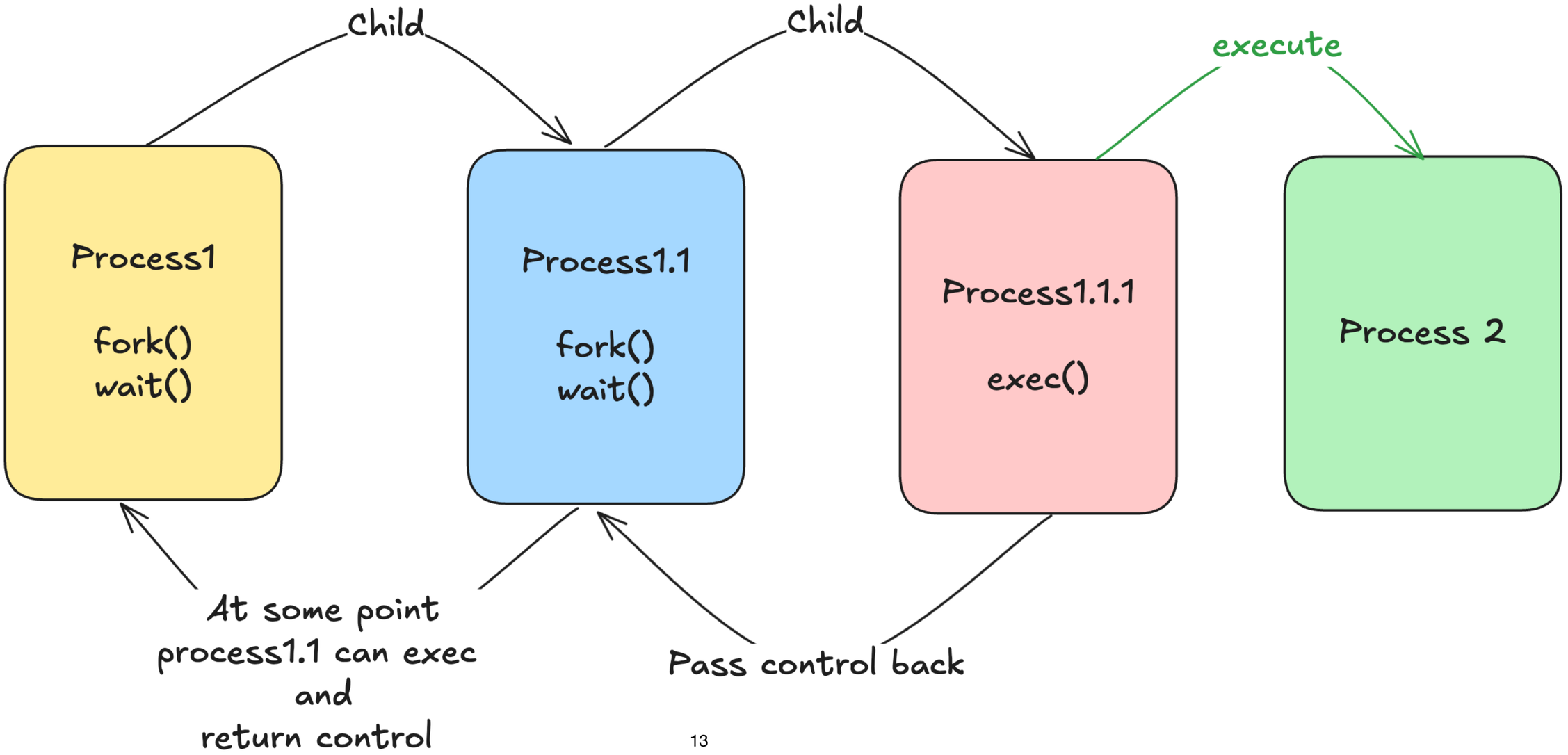
Remember: Init process, adopts orphans and reaps them

The Exec API

- When we perform a fork(), the parent and child execute the same code
  - Do you see some problem there?

- exec() comes to the rescue
  - Load a different executable to the memory
  - **Essence:** Child can run a different program from parent
  - The process ID of the process will remain the same
- In some variants of exec(), command lines to the executables can be passed!

Illustrative Flow



How does the Shell work? - Ever thought?

- Init process is started upon hardware initialisation
- The init process spawns a shell like bash
- Shell does the following
  - Read user command
  - Forks a child and exec the command
  - Wait for it to finish -> next command

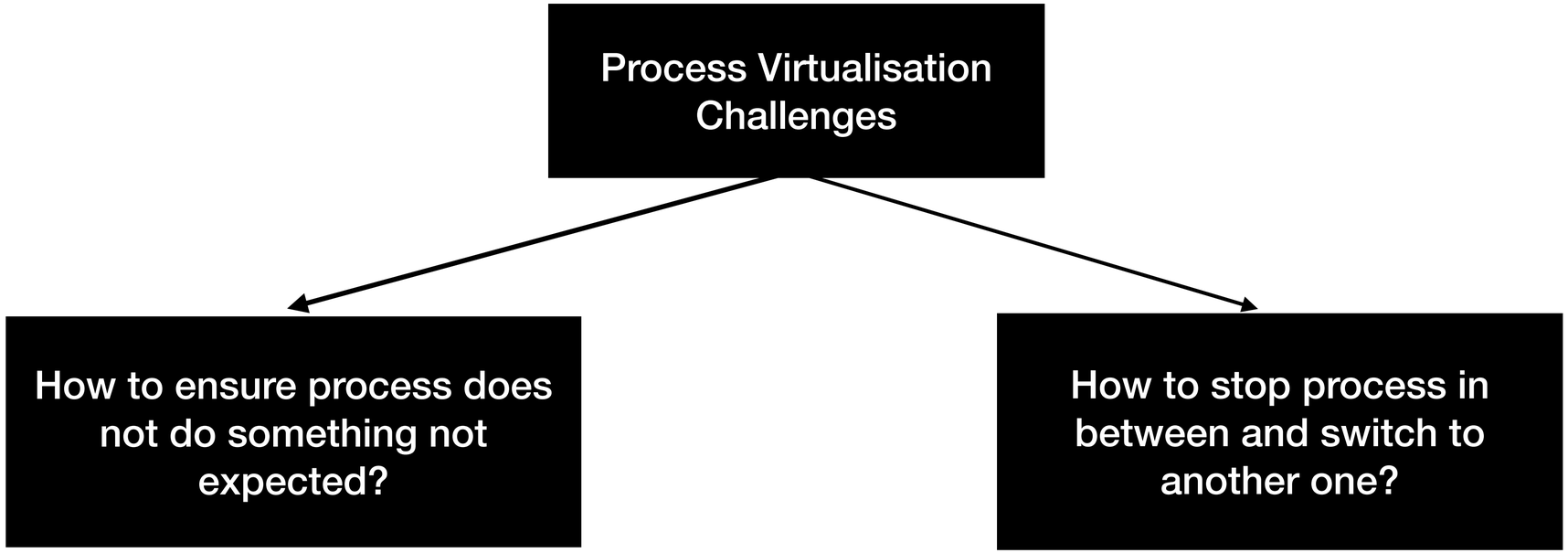
Can you think how this works?

- **> wc process_sample3.c > output.txt**
- Shell will fork a child
  - Rewires its standard output to text file (output.txt)
  - Calls exec on the child (wc process_sample.c)
  - The output will be redirected to output.txt
- Have you seen Unix pipes “|”
  - Output of one goes as input to the other

Note: fork(), exec() and wait() are required

The Big Question - How to run multiple Processes?

Two Major Problems to be Solved



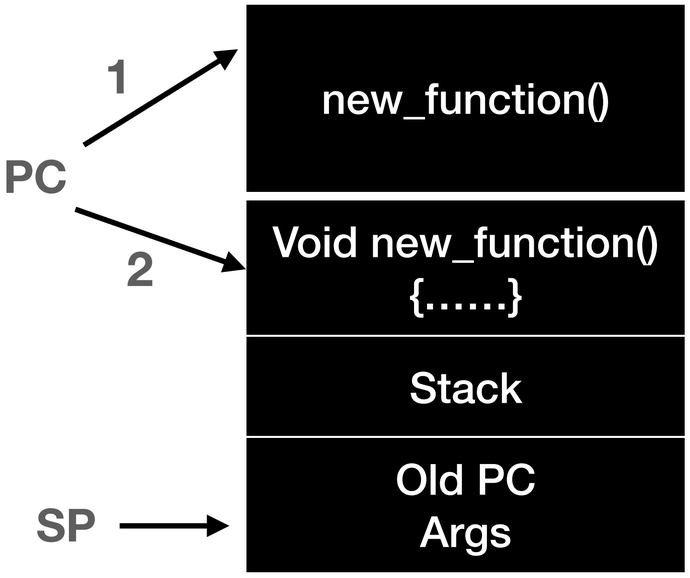
What if we allow process to do whatever it wants?

How can multiple processes run?

- **Hardware Support**
  - Have some low level mechanisms to switch process
  - What are the challenges?
    - Performance Overhead?
- **Software support**
  - Have some policies which decides what needs to be executed
  - What are some of the challenges?
    - Control overhead?

Normal Function call

- Function call translates to a jump instruction
  - One instruction to another instruction
- A new stack frame is pushed to the stack, Stack pointer is updated
- Old value of program counter (return value) pushed to stack and PC is updated
- Stack frame contains return value, function arguments, etc,



Is this enough?

OS	Program
1. Create an entry in process list 2. Allocate memory for the program 3. Load program into memory 4. Setup stack with argc/argv 5. Clear registers 6. Execute call main()	
	7. Run main () 8. Execute return from main()
9. Free memory of process 10. Remove process from process list	

What if?

- The process wants to perform operations such as:
  - Issuing I/O request to disk
  - Access to memory or other system resources
- Can we let the process do whatever it wants?

Idea: Can we think of limiting the access of a process?

Challenge 1: Prevent Unintentional behaviour

Limit Direct Execution

Only Kernel has access

https://tribuneindia.com

User program can go until this point

Thank you

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

CS3.301 Operating Systems and Networks

Process Virtualisation - Mechanisms and Policies (Part 1)

Karthik Vaidhyanathan

https://karthikvaidhyanathan.com

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 Remzi et al.
- Modern Operating Systems, Tanenbaum et al.
- Other online sources which are duly cited

Lets draw some Parallels

Library

Library Users

Reference Books

- As a visitor/user in the library - check sections, read books, magazines,..
- What about accessing the reference section and get access to some treasured books?

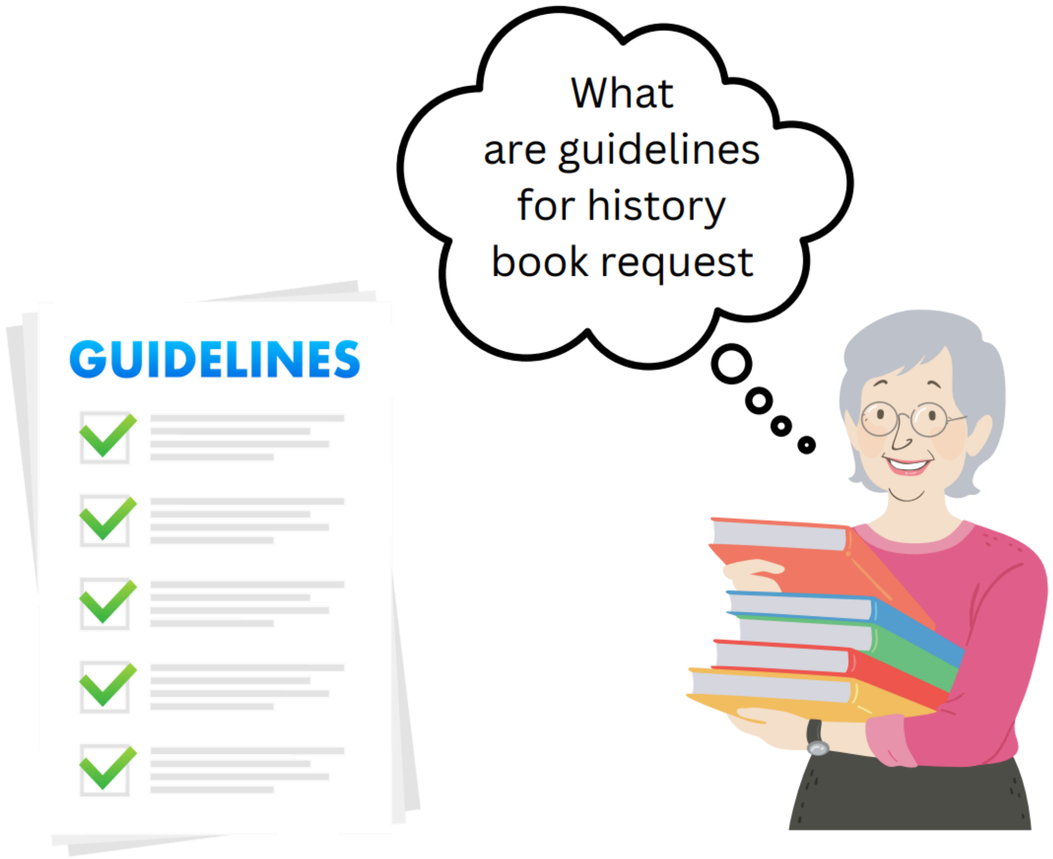
Lets draw some Parallels

Visitor/User

Librarian

Reference Books Section

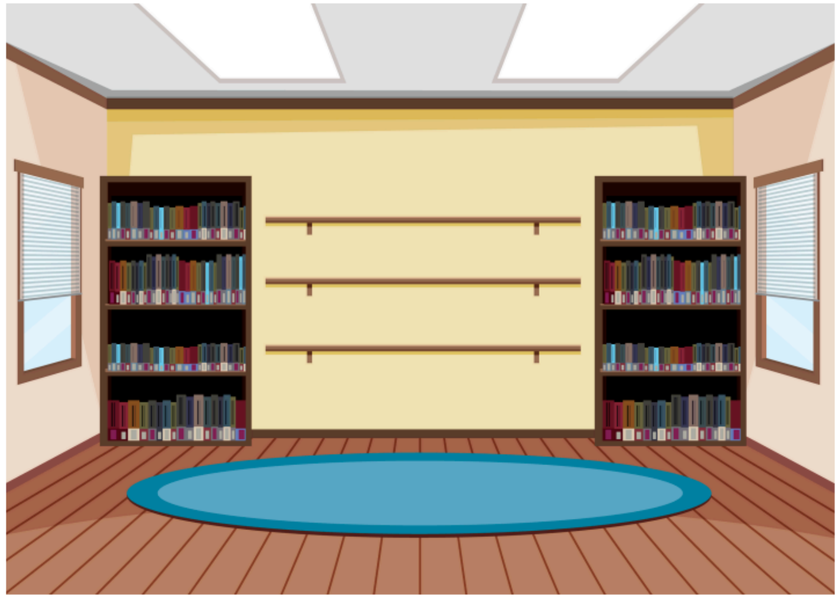
Lets draw some Parallels



Librarian

Provides access to the visitor/User as per guidelines

Lets draw some Parallels



Access completed

User is out of the reference section and continues normal access

Librarian waits for next request

Restricted Operations

- Bring hardware into the picture
  - Introduce a new processor mode
- **User mode**
  - Code is restricted in what it can do
  - Eg: no I/O request, Processor will raise an exception
- **Kernel mode**
  - Code can do whatever it likes to do
  - All privileged operations can be executed

Any challenges that you can think of?

Limited Direct Execution (LDE)

- Low level mechanism that separates the user space from kernel space
- Let the program directly run on the CPU
- Limits what process can do
- Offer privileged operations through well defined channels with the help of OS

At the end we need OS to be more than just a library!

How to move from User to Kernel?

- **System calls** - Kernel performs on behalf of user process
  - Key pieces of functionality exposed by the kernel
    - File system, process management, process communication, memory allocation, etc
  - Most OS provides few 100s of calls
  - Early unix - 20 calls
- Some privileged hardware instruction support is needed - Cannot use normal function call mechanism

System call works little differently

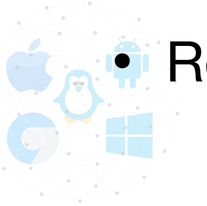
- Kernel does not trust the user stack - You don’t want to jump to random addresses
  - Maintains a separate **kernel stack (kernel mode)**
- Kernel cannot rely on user provided address
  - Uses a table - Interrupt Descriptor table (boot time) - **Guidelines in our example**
  - IDT consists of addresses of different kernel functions to run on system calls or other events

TRAP Instruction

- Special kind of instruction to switch mode from user to kernel
- Allows system to perform what it wants
- When a system call is made, the trap instruction allows to jump into kernel
  - Raise the privilege mode to kernel mode
  - **Return-from-trap** instruction allows switch back to user mode
  - Return into the calling user program
- Normal routine is interrupted

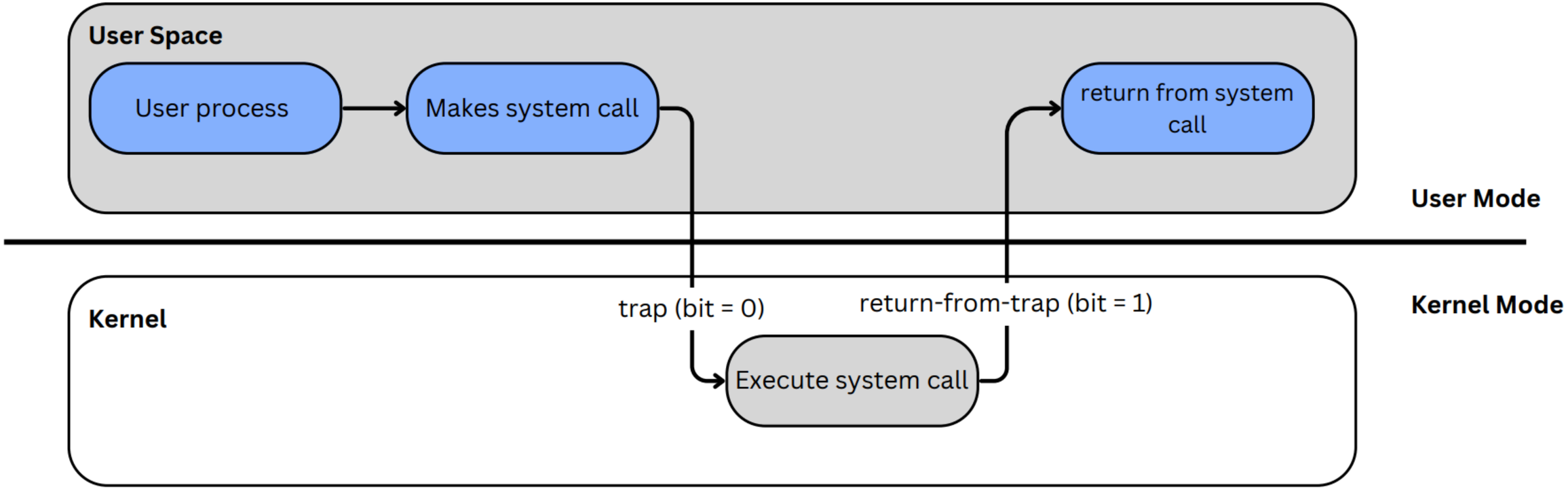
More about TRAP instruction

- During TRAP instruction execution
  - CPU to higher privilege level
  - Switch to Kernel Stack
  - Save context (old PC, registers) on Kernel Stack
  - Look up in IDT (Trap Table) and jump to trap handler function in OS code
  - Once in Kernel, privileged instructions can be performed
- Once done, OS calls a special **return-from-trap** instruction
- Returns into calling program, with back to **User mode**

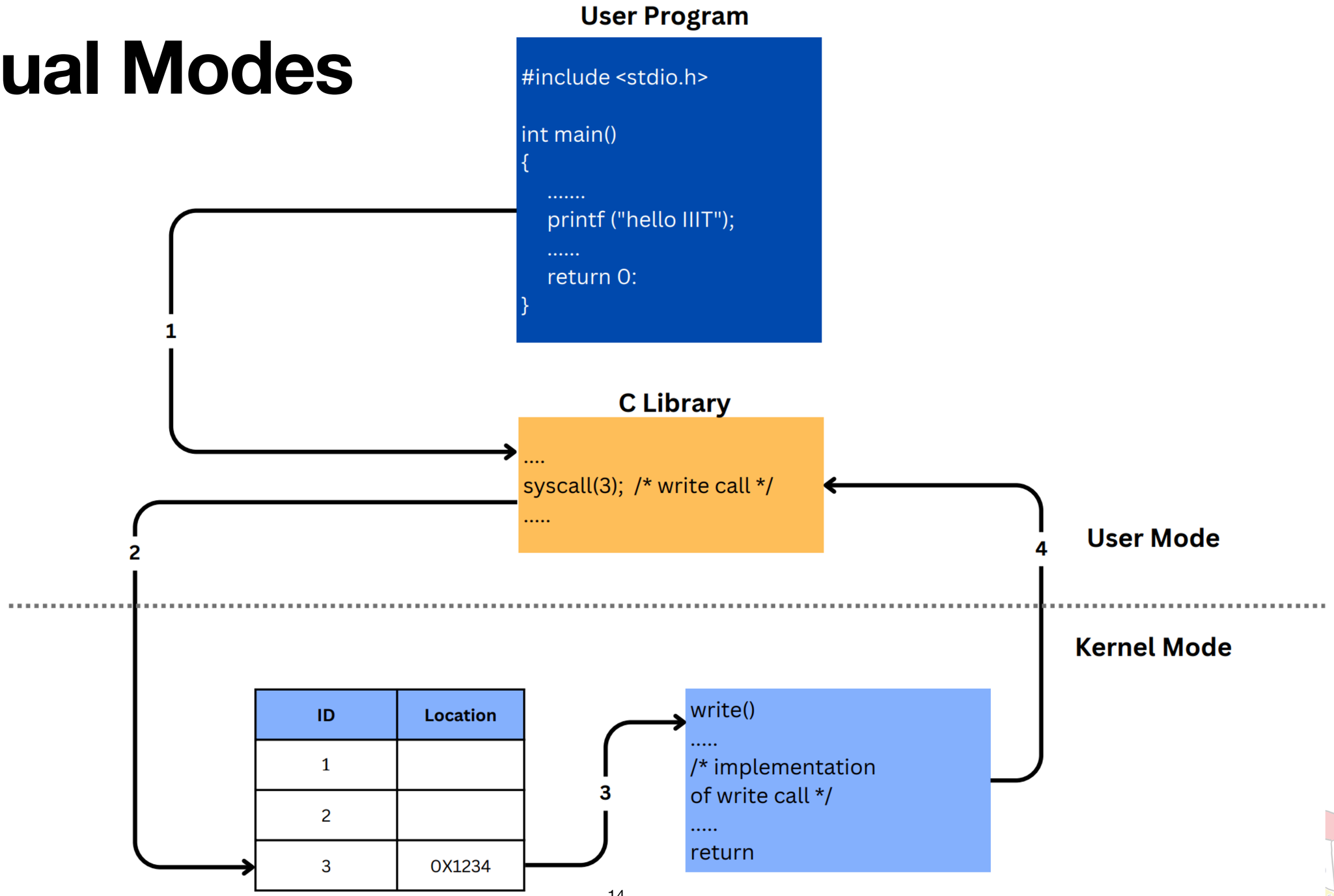


The dual modes

User Mode and Kernel Mode



The Dual Modes



Interrupt and Trap

- **Interrupt**
  - Signal sent to the CPU due to unexpected event
  - I/O Interrupt, clock Interrupt, Console Interrupt
  - From either Software or Hardware interrupt
    - Hardware may trigger an interrupt by signalling to the CPU
- **Trap**
  - Software generated interrupt caused by
    - Exception: Error from running program (divide by Zero)
    - System call: Invoked by user program

LDE Protocol

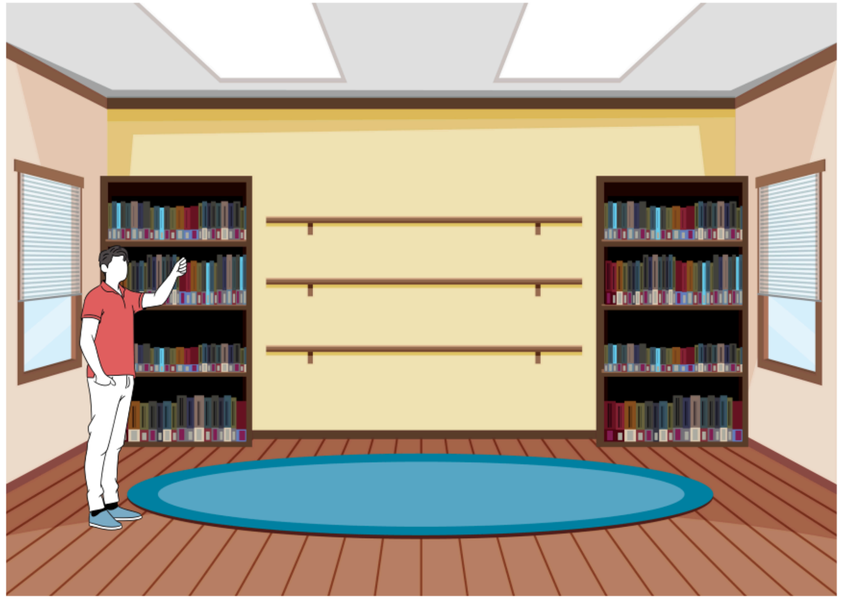
OS @ boot (Kernel mode)	Hardware	
Initialize trap table	Remember address of.. Syscall handler..	
OS @ run (Kernel mode)	**Hardware**	**Program (User mode)**
Create entry for process list Allocate memory for program Load program into memory Setup user stack with arg Fill kernel stack with reg/PC		
	Restore regs from kernel stack Move to user mode Jump to main	
		Run main() .. System call **trap** into OS

LDE Protocol

OS @ boot (Kernel mode)	Hardware	Program (User mode)
	…. Save regs to kernel stack Move to kernel mode Jump to trap handler	
Handle trap Execute the system call Return-from-trap		
	Restore regs from kernel stack Move to user mode Jump to PC after trap	
		… Return from main() trap (via exit())
Free memory of process Remove process from process list		

Problem 2: How to Switch between Process?

Lets draw some parallels



Librarian does not have a control
when the person is inside the reference section
(only one reference section and a person is already inside)

More users/visitors have requested
to access the reference section

How can this situation be handled? - What can be the possibilities?

Cooperative Approach

Non-Preemptive

- Wait for system calls
- OS trusts the processes to behave reasonably (Give control back - **Yield()** call)
- Process transfer the control to the CPU by making a system call
- There can be misbehaving process (They may try to do something they shouldn’t)
  - Divide by zero or attempting to access memory it shouldn’t
  - Trap to OS -> OS will terminate the process
- Used in initial versions of Mac OS, Old Xerox alto system
- What if there is an infinite loop & process never terminates? - **Reboot**

Non-Cooperative Approach

Preemptive

- **OS takes control**
  - The only way in cooperative approach to take control is reboot
  - Without Hardware support, OS can’t do much!
  - How can OS gain control?
- **Simple solution - Use interrupts**
  - Timer interrupt was invented many years ago
  - Every X milliseconds, raise an interrupt -> halt the process -> invoke interrupt handler -> OS regains control

Non-Cooperative Approach

Preemptive - Timer Interrupt

- During boot sequence, OS starts the timer
- The time raises an interrupt every “X” milliseconds
- The timer interrupt gives OS the ability to run again on CPU
- Two decisions are possible - Component called Scheduler comes into picture
  - Continue with current process after handling interrupt
  - Switch to a different process => OS executes **Context Switch**

Context Switch

- A low-level piece of assembly code
- **Save a few register values** from executing process registers to kernel stack
  - General purpose registers
  - Program counter
  - Kernel stack pointer
- Restore values for the next process
  - essentially return-from-trap will go to new process
- Switch to Kernel stack for the next process

LDE Protocol (Timer Interrupt)

OS @ boot (Kernel mode)	Hardware	
Initialise trap table	Remember address of.. Syscall handler.. Timer handler	
Start interrupt timer	Start timer Interrupt CPU every “X” milliseconds	
OS @ run (Kernel mode)	**Hardware**	**Program (User mode)**
		Process A ….
	Timer interrupt Save regs(A) to k-stack(A) Move to kernel mode Jump to trap handler	

LDE Protocol (Timer Interrupt)

OS @ boot (Kernel mode)	Hardware	Program (User mode)
Handle the trap Call switch() routine Save regs(A) to proc-struct(A) Restore regs(B) from proc-struct(B) Switch to k-stack(B) **Return-from-trap (into B)**		
	..... Restore regs(B) from k-stack(B) Move to user mode Jump to B's PC	
		Process B ...

What if?

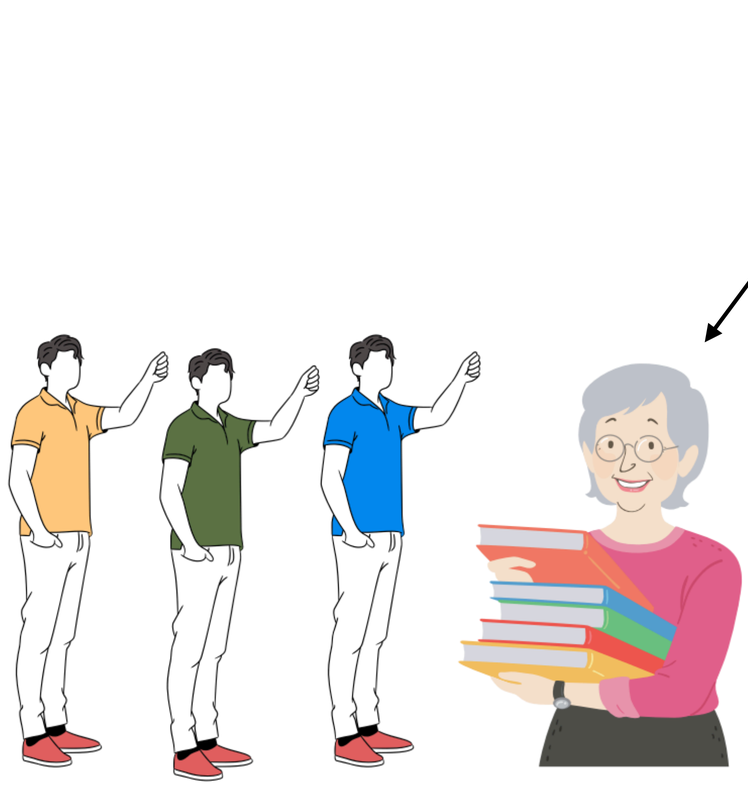
- During handling of one interrupt another interrupt occurs?
  - Disable interrupt during interrupt processing
  - Sophisticated locking mechanism to protect concurrent access to internal data structures

How to decide which process to run next?

Need for Policies (Scheduling)

Which process to schedule next on context switch?

The person is almost done with his reading and might come out soon (or time out!!)



More users/visitors have requested to access the reference section.How to decide whom to sent next?

Scheduling in the Library Scenario

What we need to know to ensure good policy?

- How many users want to go to the reference section?
- What's the purpose? - What type of book they want to read?
- How much time are they expected to be in the reference section?
- How frequently are new users coming in?

Essentially it would be good to have these estimates to make a good policy!

What does it mean Concretely?

- For scheduling we need an idea of **workload**
  - Assumptions about processes running in the system
    - Number of processes
    - RAM required
    - CPU utilisation
    - Any Input/Output, if yes what kind?
    - ….

Lets start with some workload assumptions

Each process that is ready/needs to be executed and those executing - **Job!**

Some Assumptions:

1. Each job runs for **same amount of time**
2. All jobs **arrive** at the **same time**
3. All jobs **only use the CPU** (No I/O)
4. The **run time** or execution time of each job is **known**

How good is the policy?
Some Key Scheduling Metrics

- Metric is something we used to measure
- **Performance metric:** Turnaround time
  - Time difference between job completion time and the arrival time

$$T_{turnaround} = T_{completion} - T_{arrival}$$

- Another **metric is fairness** - Jains fairness index: How fair is the scheduling?
  - May not go hand in hand with performance

Scenario 1

All Assumptions in tact

- Imagine three jobs - Whatsapp, Skype and Teams update arriving at same time
- Each of them take same time to complete

Process	Arrival	Time to Complete
Whatsapp (w)	~0	20
Skype (S)	~0	20
Teams (T)	~0	20

How to go about this?

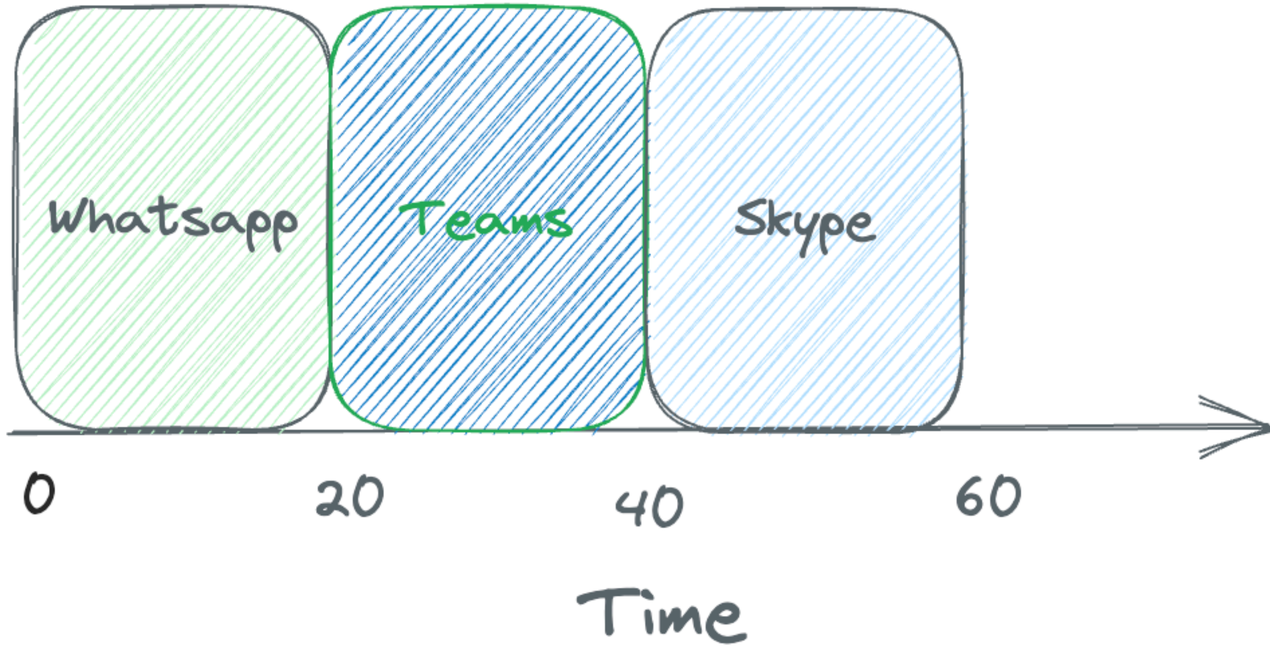
First Come First Serve Policy

- The most basic algorithm a scheduler can implement
  - Whoever comes first, give them the access

- Assume that they arrive at the same time - At time = 0
  - For sake of simplicity W just arrived before T which just arrived before S

First Come First Serve (FCFS) Policy

- **Policy:** Schedule the job came first
- As soon as it is done, schedule the job that came next, continue
- There is an assumption here that each job runs for the same time
  - What if that's not the case?
  - Let us relax this assumption

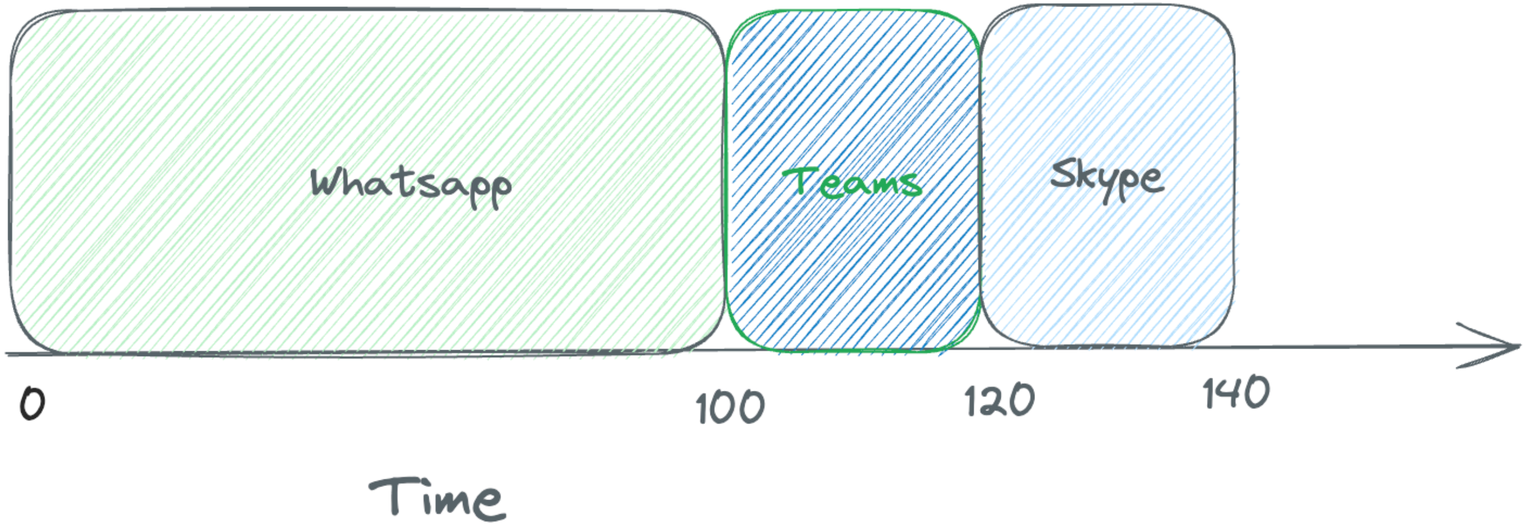


$$Avg(T_{turnaround}) = \frac{20 + 40 + 60}{3}$$

$$= 40$$

What if each job no longer runs for same time?

Relaxing assumption 2



$$Avg(T_{turnaround}) = \frac{100 + 120 + 140}{3}$$

$$= 120$$

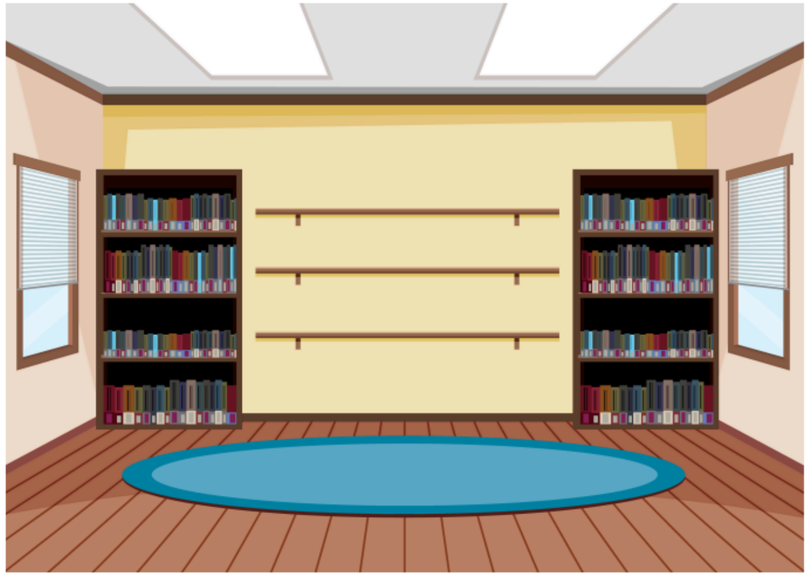
FCFS is not that great

Convoy Effect

- Waiting time can go very high
  - Convoy effect!
  - Think about waiting in single line in grocery store where you just have one item to purchase

What if?

Visitors/Users need to use the reference room, but who to give access to now? How to determine whom to give access to?



- Every one said that they will need this much time for accessing the reference section
- Librarian schedules based on the time they say

Shortest Job First (SJF) Policy

- Idea originating from operations research
- **Policy:** Run the shortest job first

Process	Arrival	Time to Complete
W	0	100
S	0	20
T	0	20

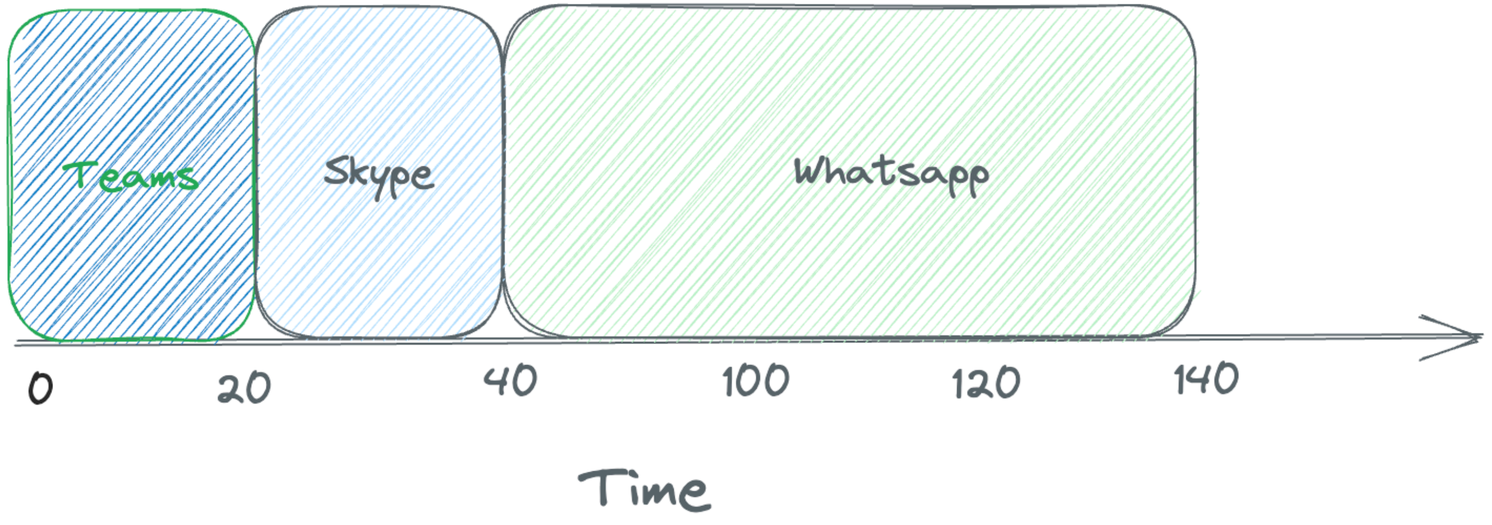
How to go about this?

Shortest Job First (SJF) Policy

- Assume that all jobs came at the same time
- Clearly whatsapp takes most amount of time

$$Avg(T_{turnaround}) = \frac{20 + 40 + 140}{3}$$

$$= 66.3$$



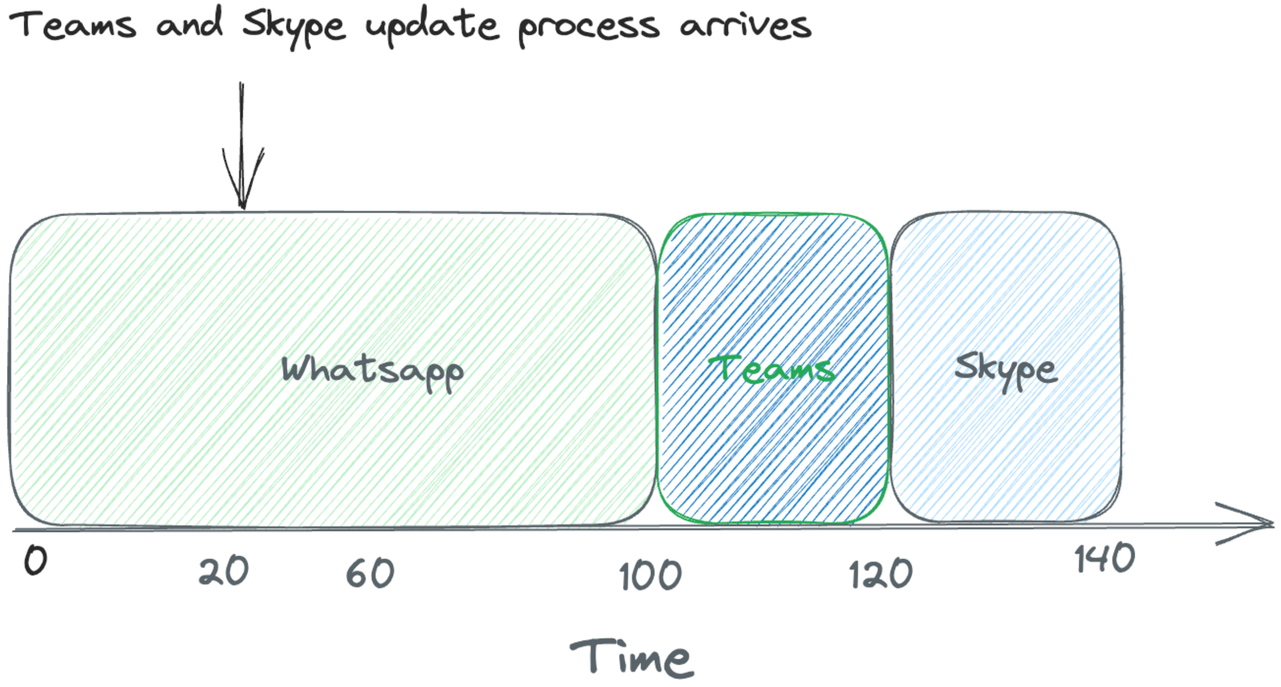
Is that a bit too unrealistic? - In reality jobs can arrive at any time

Shortest Job First (SJF) Policy

- Whatsapp job arrives first
- Teams and Skype jobs arrives around t = 20

$$Avg(T_{turnaround}) = \frac{100 + 100 + 120}{3}$$

$$= 106.6$$

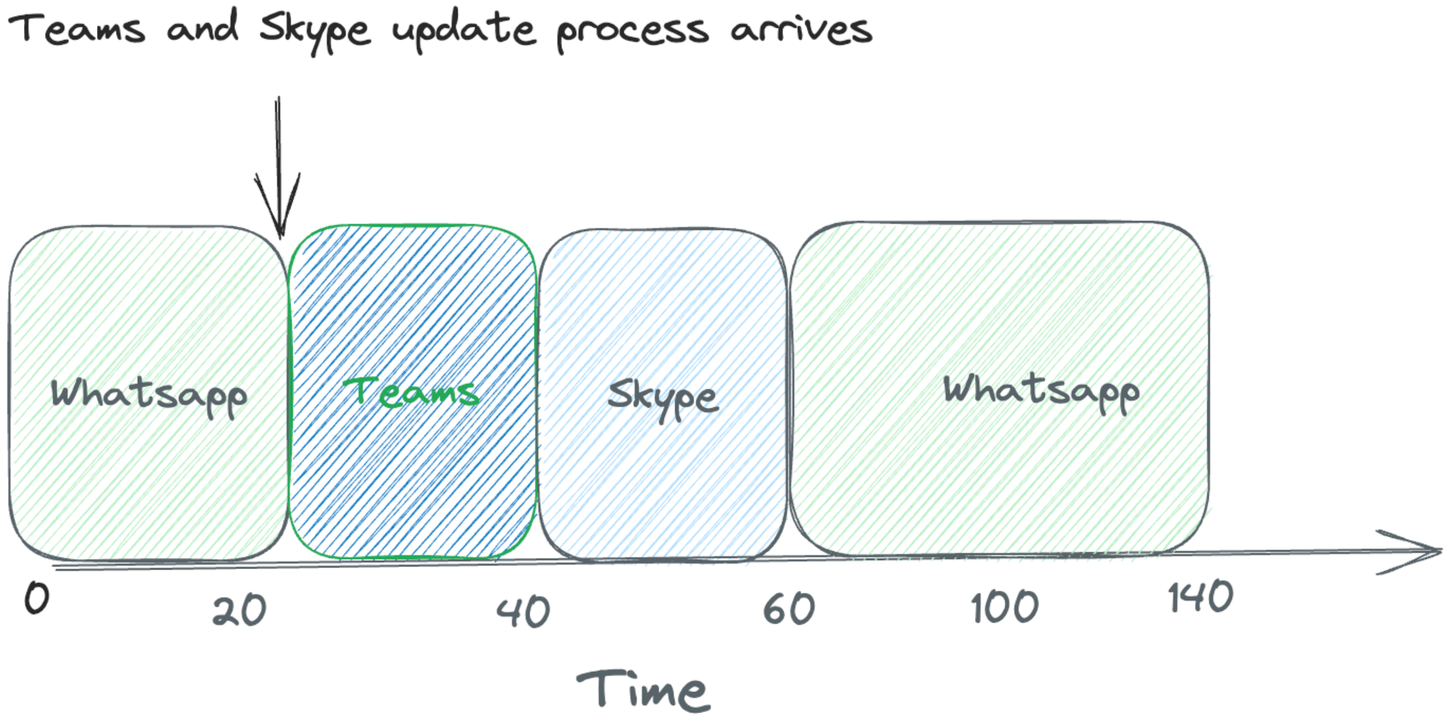


Even worst!! How to improve?

Shortest Time to Completion First (STCF)

- Adding preemption to Shortest Job First (SJF) Policy
  - More like preemptive SJF
- **Policy:** Any time a new job enters the system,
  - Check how much time is remaining for existing jobs
  - Check the time that is required for the new one
  - Execute the one that shall complete first

Shortest Time to Completion First (STCF)



$$Avg(T_{turnaround}) = \frac{(140 - 0) + (40 - 20) + (60 - 20)}{3}$$

$$= 66.3$$

Can we improve this a bit more?

- What about the user side?
  - What if this is an interactive process?
    - Think about going to Amazon or Working with some desktop application
    - Imagine a user sitting in front of the machine and executing the command
  - The machine identifies the nature of the job and schedules it
  - What about response time?

$$T_{response} = T_{firstrun} - T_{arrival}$$

Thank you

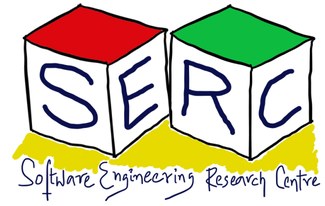
Course site: karthikv1392.github.io/cs3301_osn
Email: karthik.vaidhyanathan@iiit.ac.in
Twitter: @karthi_ishere

CS3.301 Operating Systems and Networks

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

Karthik Vaidhyanathan

https://karthikvaidhyanathan.com



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

Can we improve this a bit more?

- What about the user side?
  - What if this is an interactive process?
    - Think about going to Amazon or Working with some desktop application
    - Imagine a user sitting in front of the machine and executing the command
  - The machine identifies the nature of the job and schedules it
  - What about response time?

$$T_{response} = T_{firstrun} - T_{arrival}$$

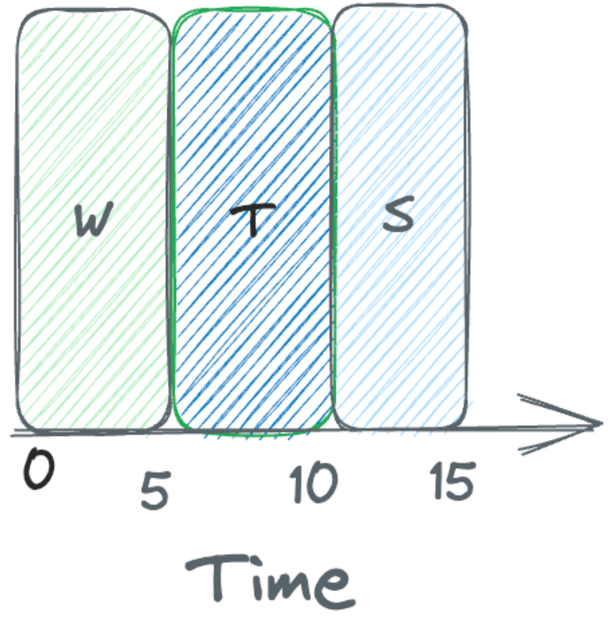
Round Robin Scheduling

- Instead of running jobs for completion
  - Can we run jobs for time intervals?
- **Policy:** Run jobs for a time slice -> switch to next job -> repeat till all are done!
- Key idea: Use the notion of time slice, considering timer interrupts
  - Take into consideration the overhead of Context Switch

Round Robin Scheduling

- What if we used SJF for the below scenario?

Process	Arrival	Time to Complete
W	0	5
S	0	5
T	0	5



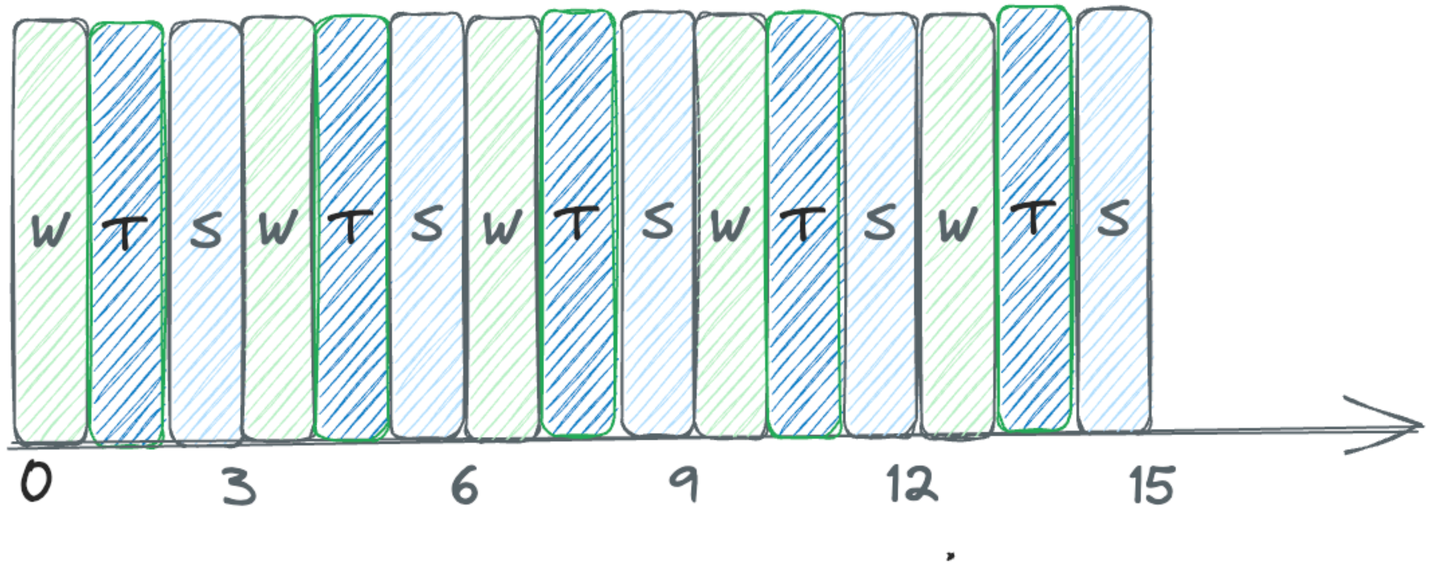
$$Avg(T_{turnaround}) = \frac{5 + 10 + 15}{3} = 10$$

$$Avg(T_{response}) = \frac{0 + 5 + 10}{3} = 5$$

Can we do better?

Round Robin Scheduling

- What if we do round robin with a time slice = 1 sec?



$$Avg(T_{response}) = \frac{0+1+2}{3}$$
$$= 1$$

W is added in the 0th Second
T in the 1st second
S in the second second

Do we foresee some issue?

Round Robin Scheduling

- Time slice plays a critical role in response time part
  - Too small time slice can result in an overhead - Too much Context Switch!
- RR is a good scheduling method
  - Key thing is to find an optimal time slice
  - Response time is the only metric
  - What about turnaround?

$$Avg(T_{turnaround}) = \frac{13 + 14 + 15}{3}$$

$$= 14!$$

Remember: Trade-off

- **Turnaround time only cares about completion**
  - Fairness of scheduling does not come into the picture - Given!
  - Fairness here comes at the cost of turnaround time
- The key aspect is to consider trade-off's
  - Very important in system design
  - Often among quality attributes
  - Eg: security vs performance

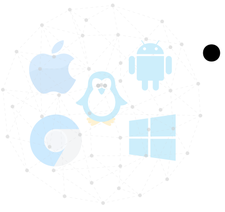
Continuing on the Assumptions

- Jobs don’t perform I/O
- Run-time of each job is known

What can be done to consider I/O? Can we do RR Scheduling by considering I/O?

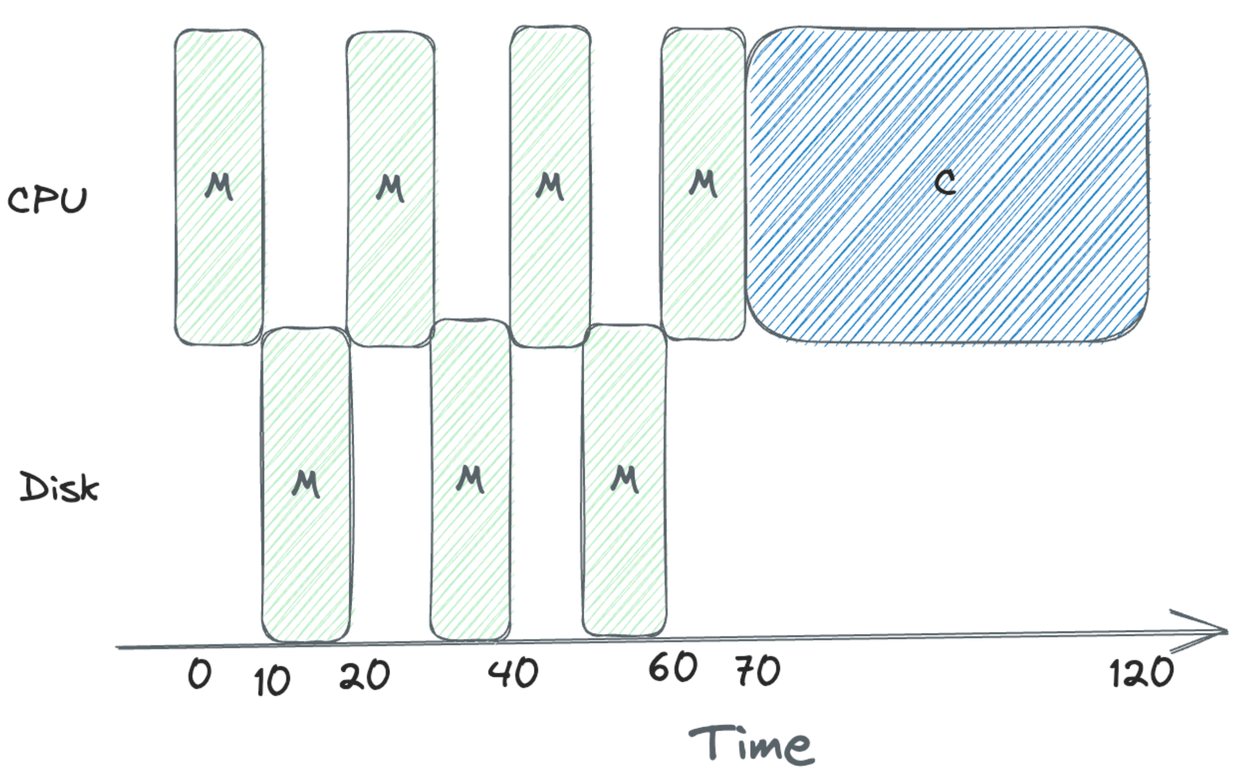
Incorporating I/O

- When there is a job doing I/O, Scheduler needs to be more decisive
  - During I/O what will usually happen?
    - The job will be **blocked** for I/O completion
    - If I/O is hard disk dependant then it may require more time
    - What can be an easy way out?
- When I/O is done - Interrupt is raised
  - OS moves the process (Job) from **blocked** to **ready** state



Lets consider a scenario

- Assume two process: Microsoft Word (autosave), your C program executing some numerical computation (No I/O access)
  - Microsoft Word (**M**): autosaves every 20 seconds => I/O access
  - C program (**C**): No I/O access



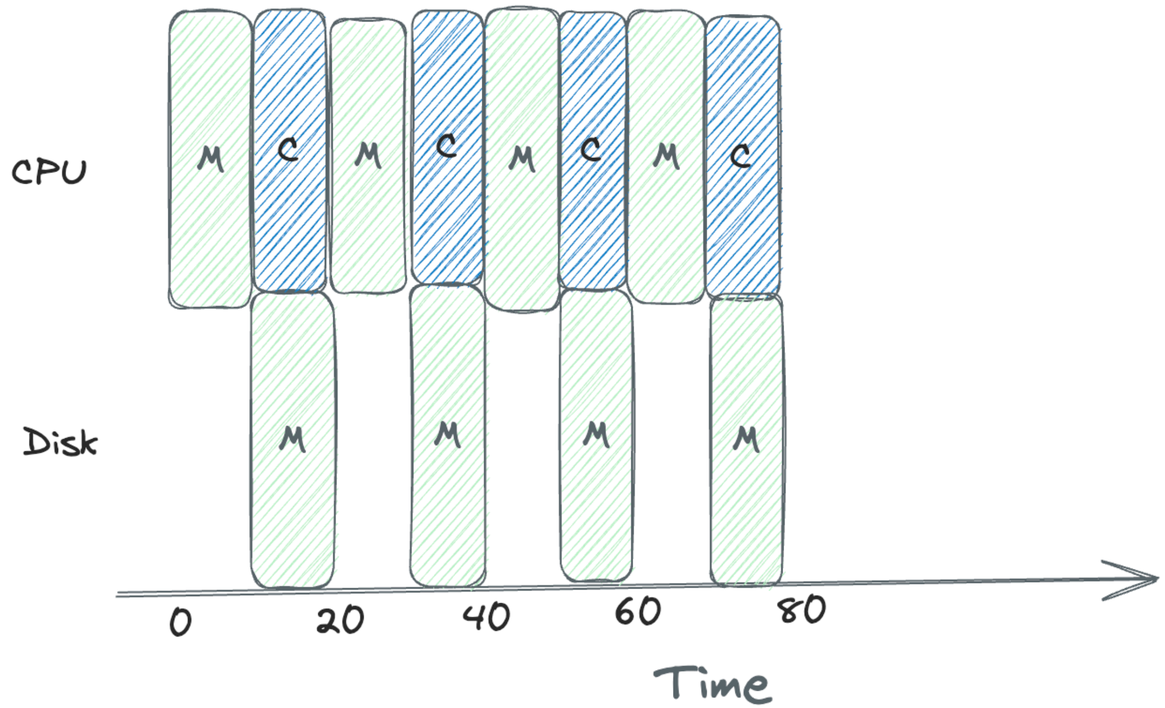
Can we do better?

Shortest Time to Completion First (STCF)

Can we leverage STCF?

- **Policy:** CPU is used by one process while the other one uses the disk
- Each CPU burst can be treated of as separate job
- Better utilization of the processor

Do we miss something?

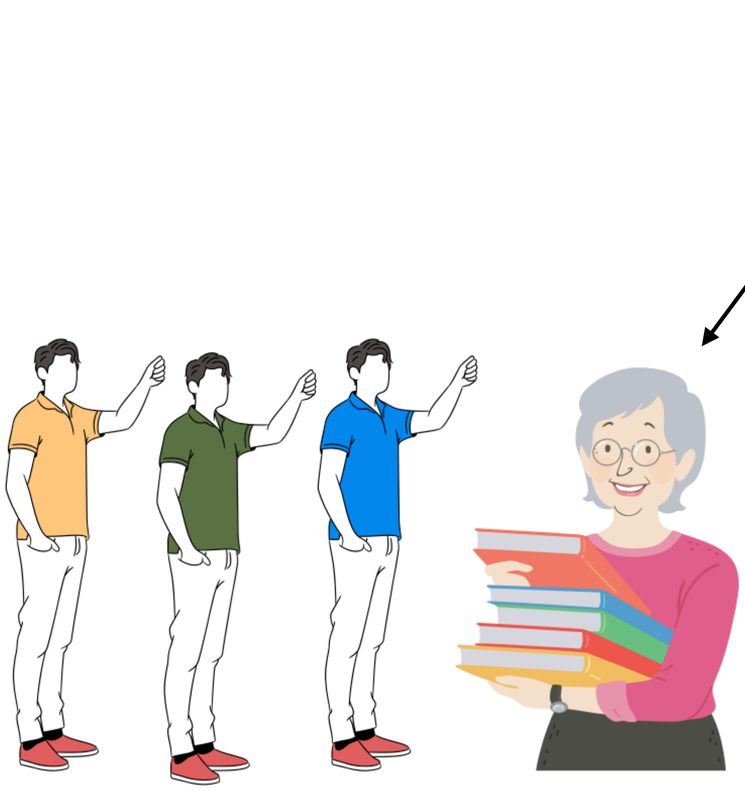


We may not know the length or expected time of completion of a job? — How to handle?

Lets go back to the example

How can Librarian take a guess?

The person is almost done with his reading and might come out soon (or time out!!)



More users/visitors have requested to access the reference section.How to decide whom to sent next?

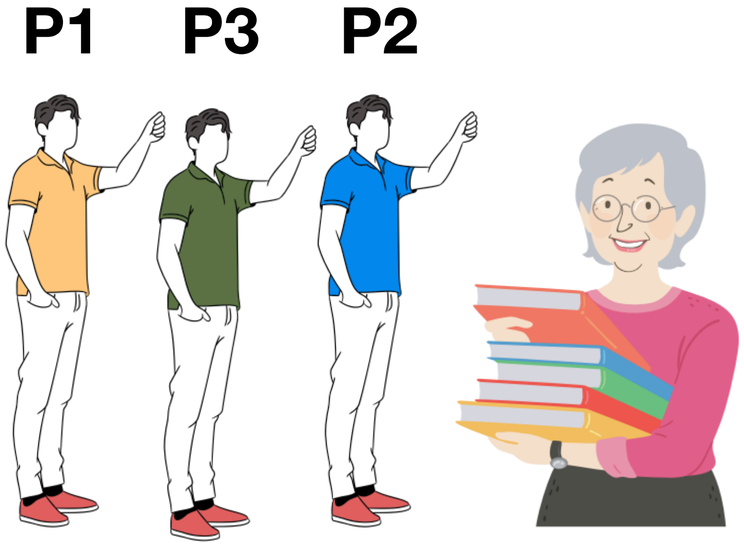
Why don’t we Prioritise?

Source: https://imgur.com

Lets go back to the example

Introduce Priority - Give priority, Observe and Improve

The person is almost done with his reading and might come out soon (or time out!!)



More users/visitors have requested to access the reference section.How to decide whom to sent next?

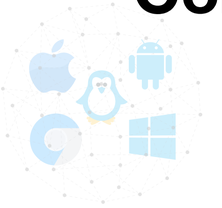
Multi Level Feedback Queues (MLFQ)

Two main features

- Reduce turn around times
  - Run shortest jobs first
- Reduce response time

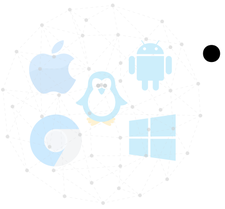
Can the policy learn continously to optimise response time and turnaround time?

Constraint: No apriori knowledge of the job length!

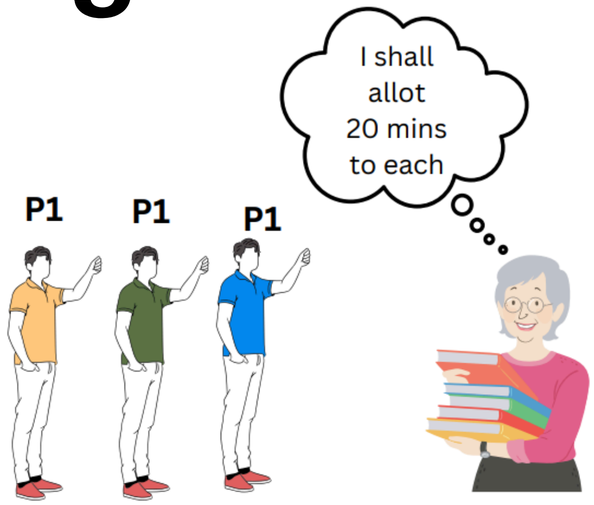


MLFQ: Basic Rules

- Use **n** number of distinct queues
  - Each queue has a different priority level
- Use priority to decide which job should be run at a given time
  - A job with a higher priority => job on a higher queue
- **Key idea:**
  - Scheduler sets priority to different jobs
  - Keep updating the priority based on observed behaviour



Going back to the example



Put all the visitors initially on the same priority
Observe how they behave and then decide!

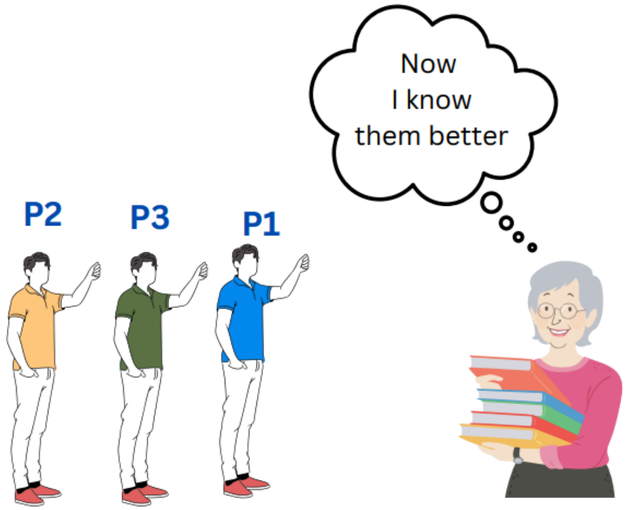
First person gets the chance!

out in 18 mins

Next person goes in

20 mins!!

0ut in 15 mins!



For the next visit, priorities have been updated

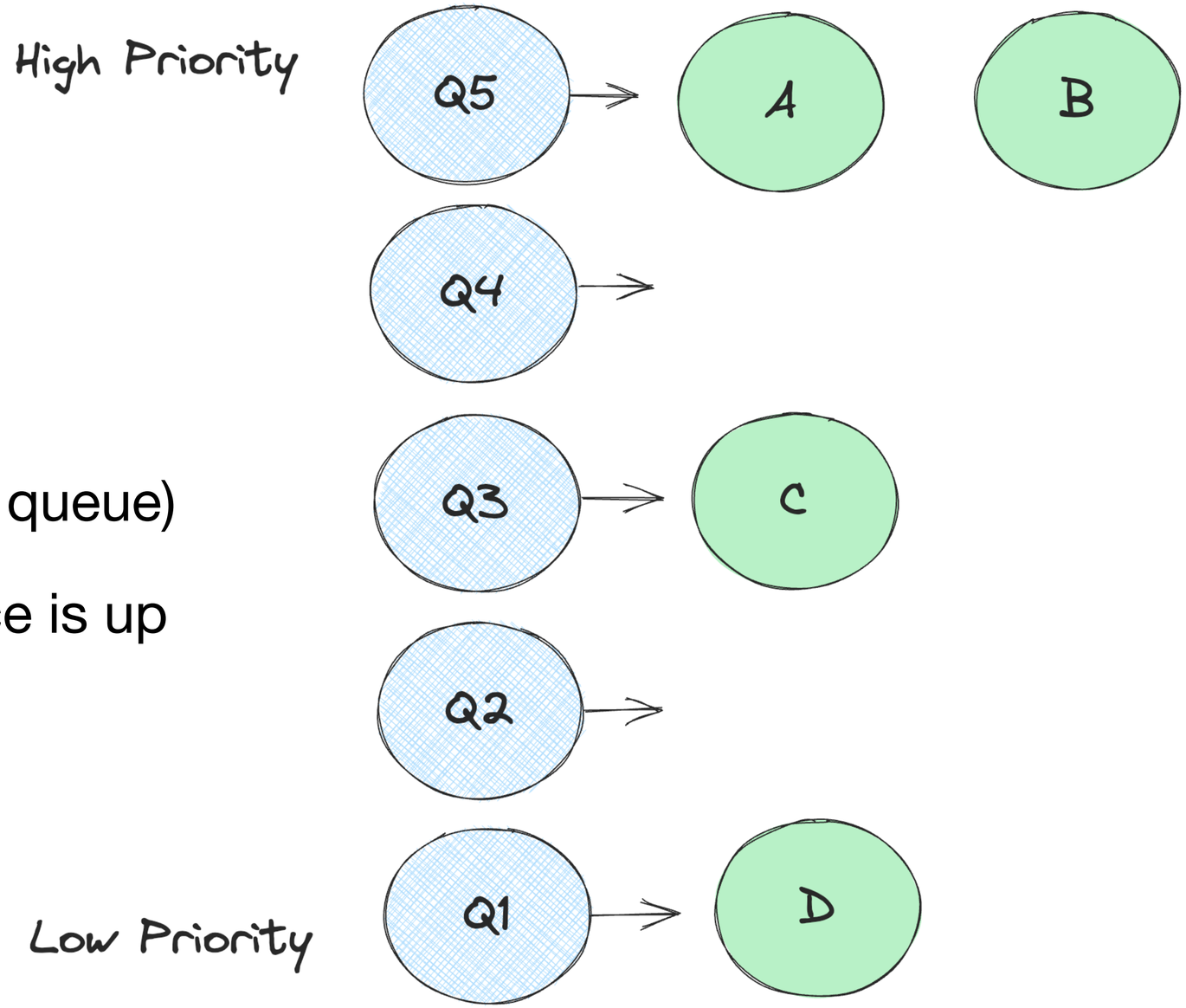
MLFQ

- Jobs that keep giving back the CPU - interactive jobs (higher priority)
- Jobs that uses CPU for more time - Reduce priority
- Learn about the processes as they run and **predict future** (Not using AI)
- **Two basic rules:**
    1. If priority (A) > Priority (B), A runs
    2. If priority (A) = Priority (B), A&B run in Round Robin

MLFQ: Visualisation

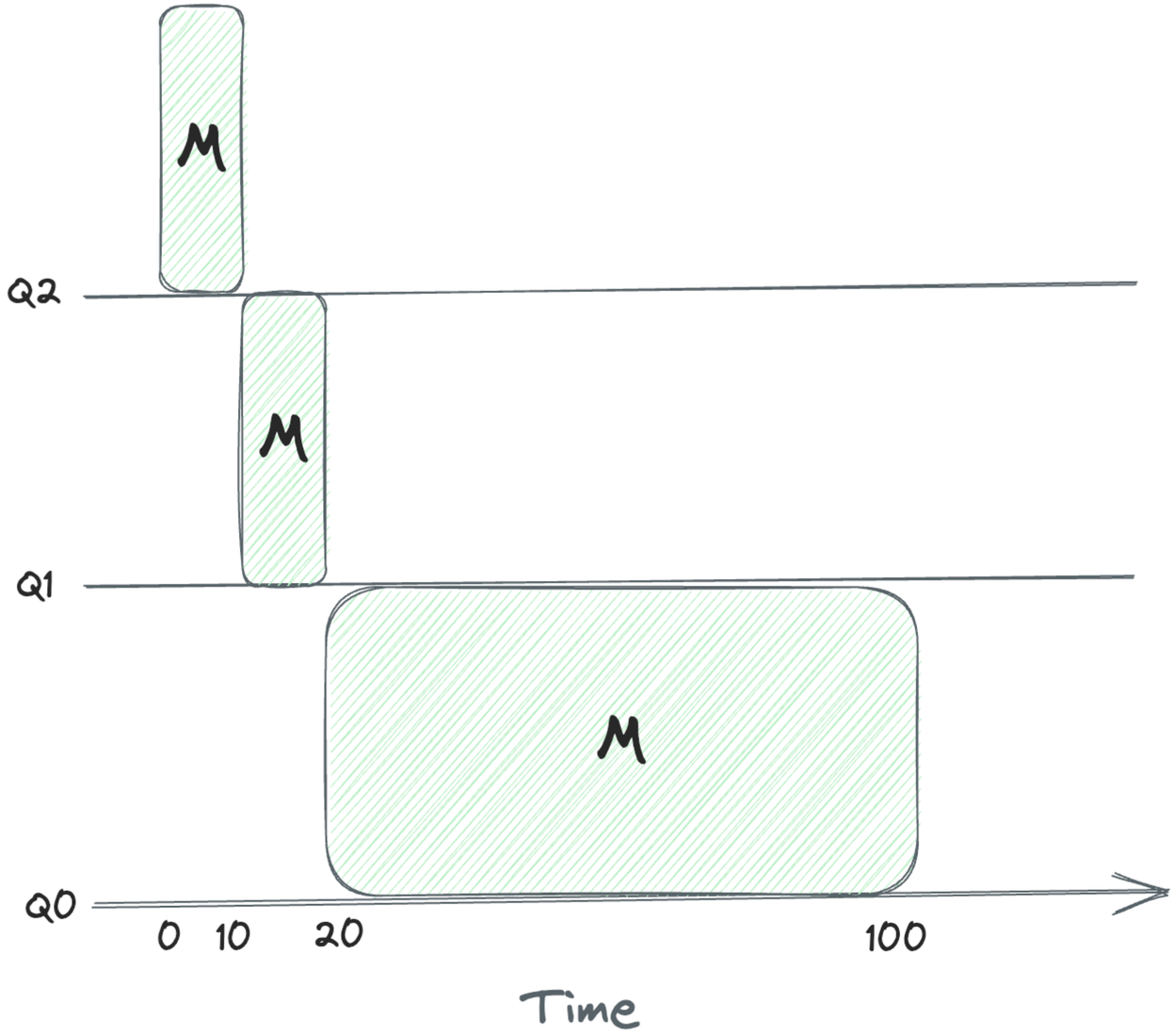
Five queues and 4 jobs

- Use notion of time slices
- When job enters, high priority
- If job uses up the entire time slice
  - Reduce priority (move down one queue)
- If job gives up CPU before time slice is up
  - Keep it at the same priority level



Example: Single Long Running Job

- Three queues
- Priority: Q2 > Q1 > Q0
- Single job M enters at t=0
- Time slice = 10 seconds
- After running for 10 seconds, priority is lowered

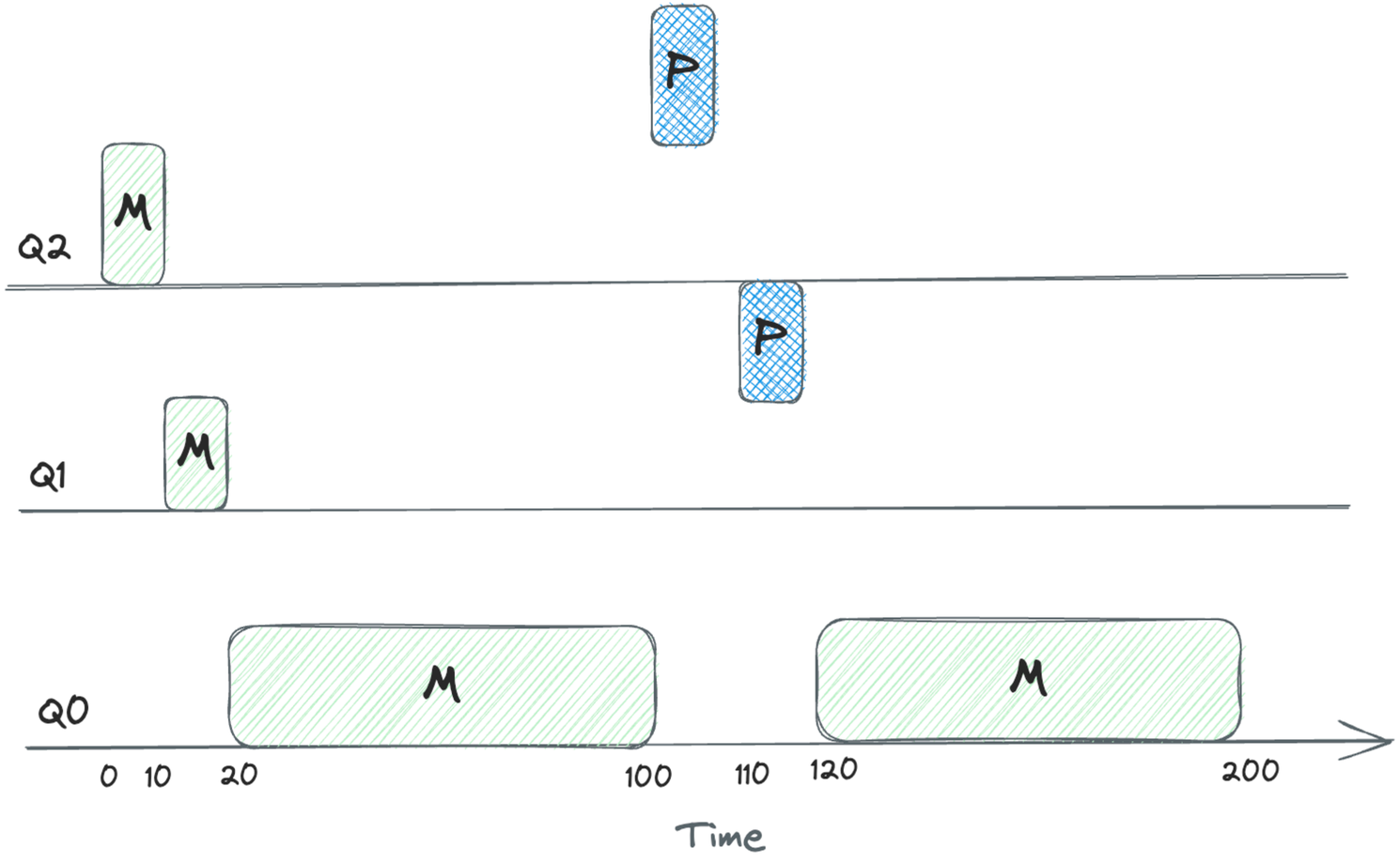


Example: Short job enters

- **P (interactive job)**
  - Enters at t=100
  - Runs for 10 second
  - Goes to the second queue
- Incoming job is considered

short => placed in higher queue

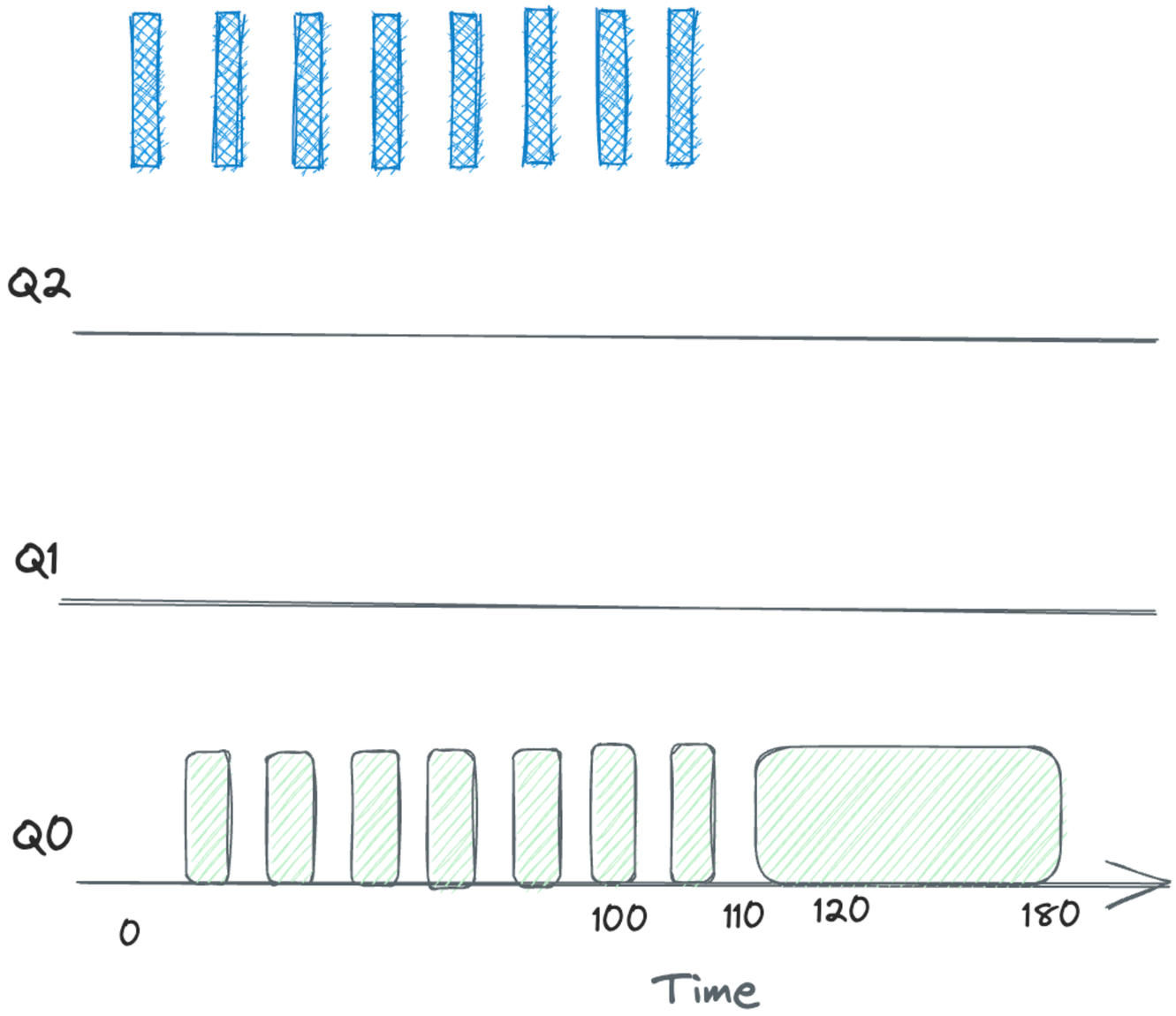
- If its short, it will keep running
  - Else moves down



Incorporating I/O

- Highly interactive job => relinquishes CPU faster
  - Priority is kept at same level
- Long term job can take more time and execute in lower queue

What could be the challenges?



What are the challenges with previous model?

It can lead to two major challenges

- **Starvation**
  - Too many interactive jobs will keep consuming CPU
  - Long running job will never get any CPU - **Starve!!**
- **Gaming of Scheduler**
  - The process can trick the scheduler into giving more than fair share - How?
  - **Idea:** Give an I/O request and relinquish the CPU before time slice is over
    - Priority does not change!! (Monopolise the CPU)

Another Challenge to Consider

- Program may also change behaviour over time
  - Suddenly there may be more CPU intensive phases
  - There may be also phases of interactiveness
  - Eg: Some long numerical computation followed by interactiveness

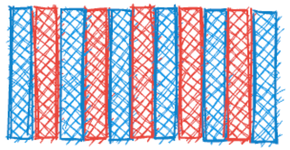
Can you think of some way to handle this?

Priority Boost

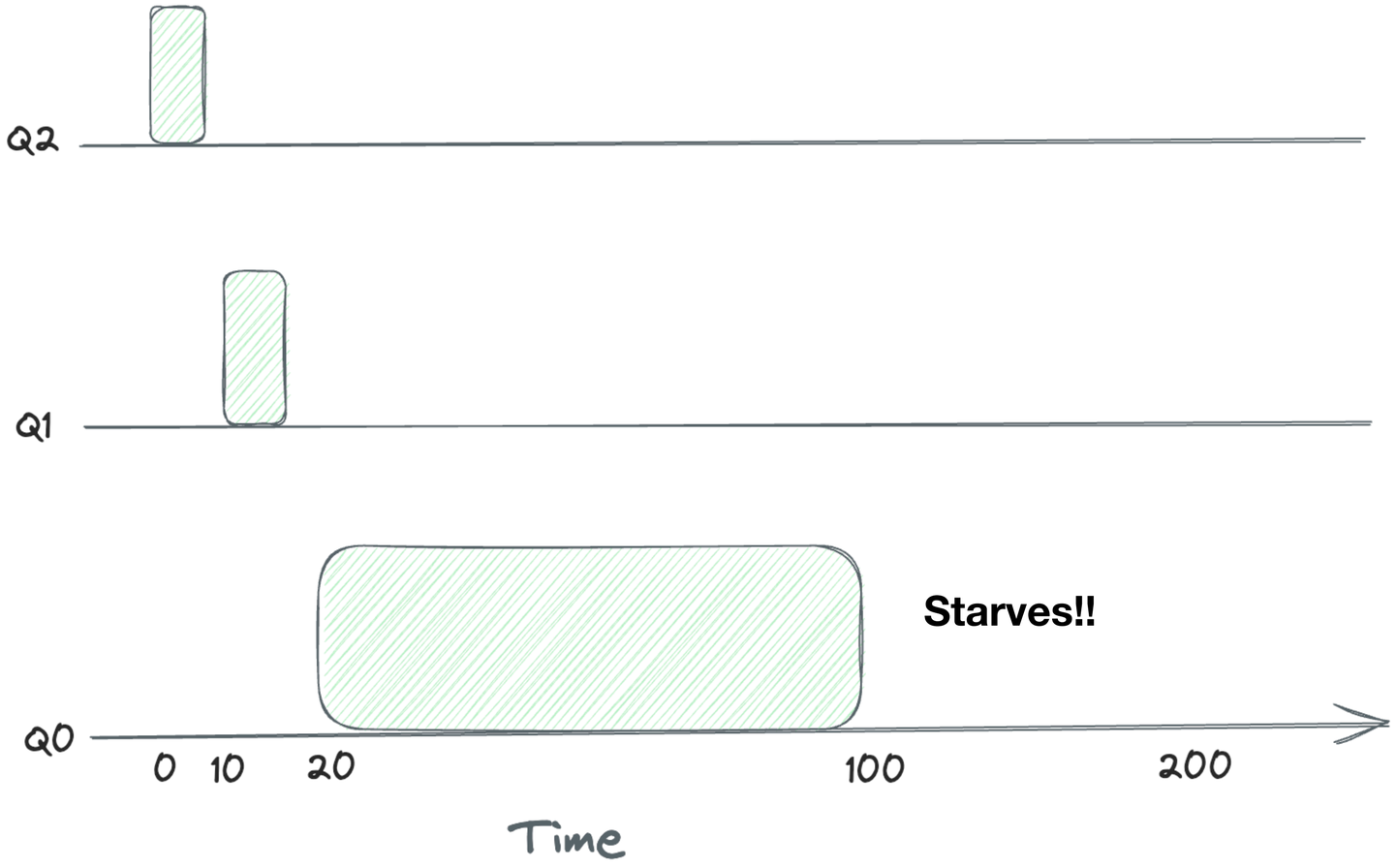
- Periodically boost the priority of all the jobs
  - This can **prevent starvation**
- **Rule:** After a time interval ***S***, move all jobs to top most queue
- **Provides two key guarantees**
  - Processes are guaranteed not to starve
  - If CPU bound has become interactive, scheduler will give it chances
    - Thanks to the periodic priority updates!

Case 1

Non priority Boost Scenario

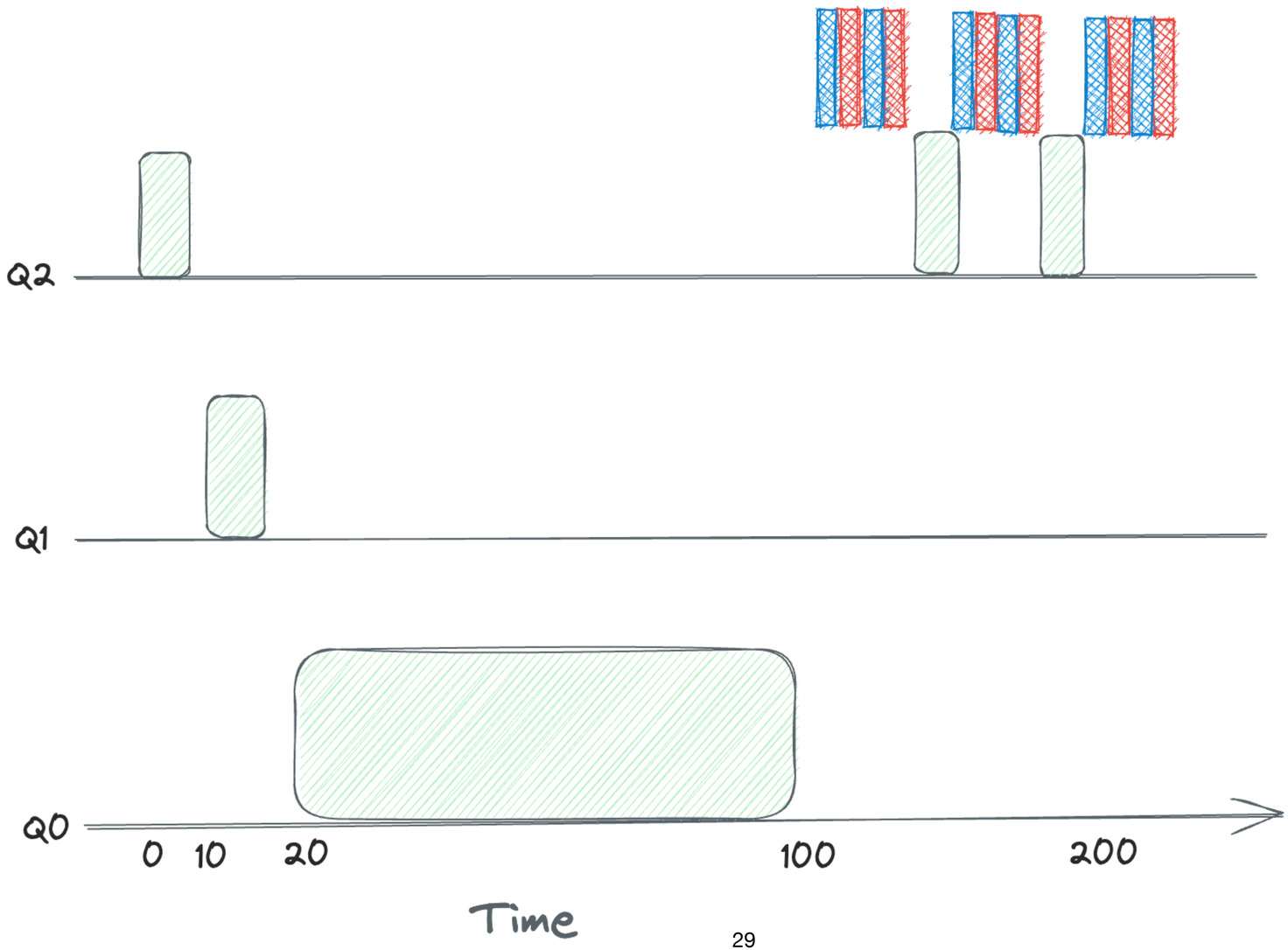


Interactive jobs keep getting the CPU



Case 2

With Priority Boost (S = 50)



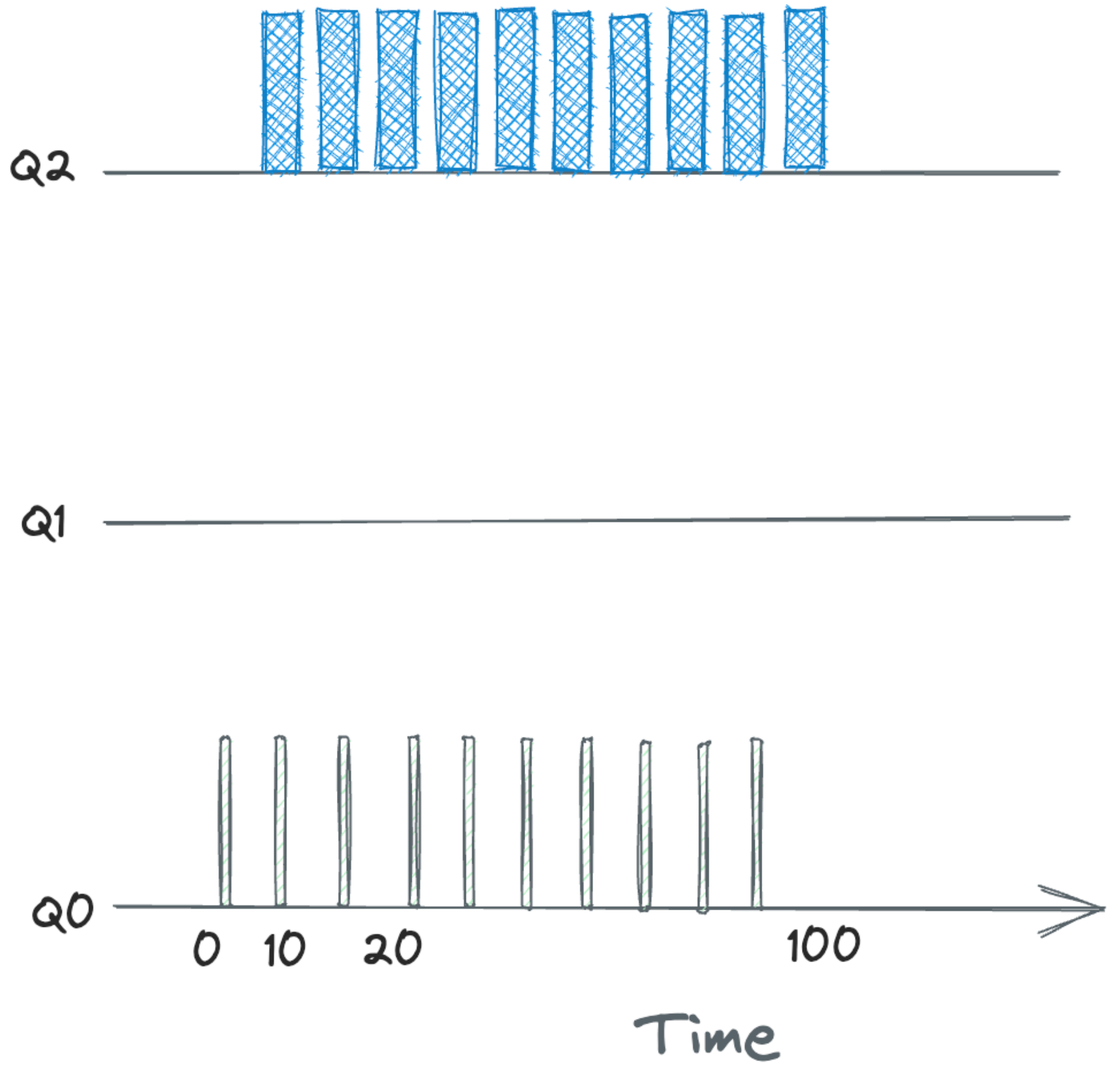
One additional Challenge

- How to determine the value of **S**?
  - If S is very small, interactive jobs may not get proper share of CPU
  - If S is too high, long running jobs could starve
- Tricky part is to come up with a value of S
  - Voo-doo constants - Named by John Ousterhout
- What about gaming the scheduler? - S by itself cannot solve it!

Better Accountability

How to prevent gaming of the scheduler?

- Introduce one more rule
  - Once a job uses its time allotment, it needs to be moved down
  - No consideration if the job has relinquished CPU ahead of time slice
  - Prevent the gaming since no matter what, priority reduces



Tuning MLFQ

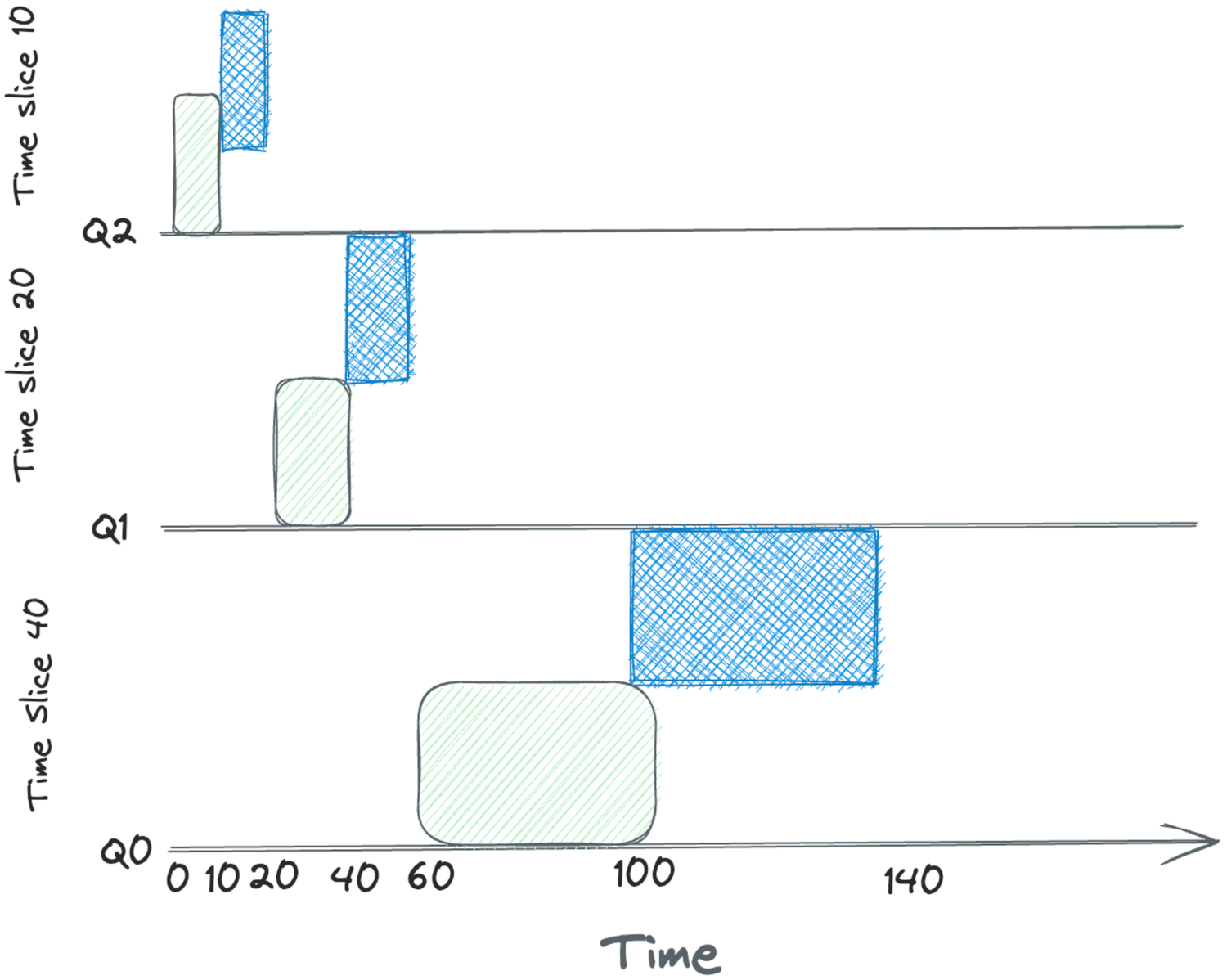
- How to parametrise the scheduler?
  - How many queues?
  - What should be the time slice?
  - How often the priority boost needs to be done?
- **High priority queues:** Interactive jobs with shorter time slices (10 ms)
- **Low priority queues:** CPU bound jobs with longer time slices (100 ms or less)

Example Scenario

- Different queues
- Low priority and high

Priority queues

- Each queue has a different slice interval
- Based on scenario, S changes (boost interval)



Solaris MLFQ

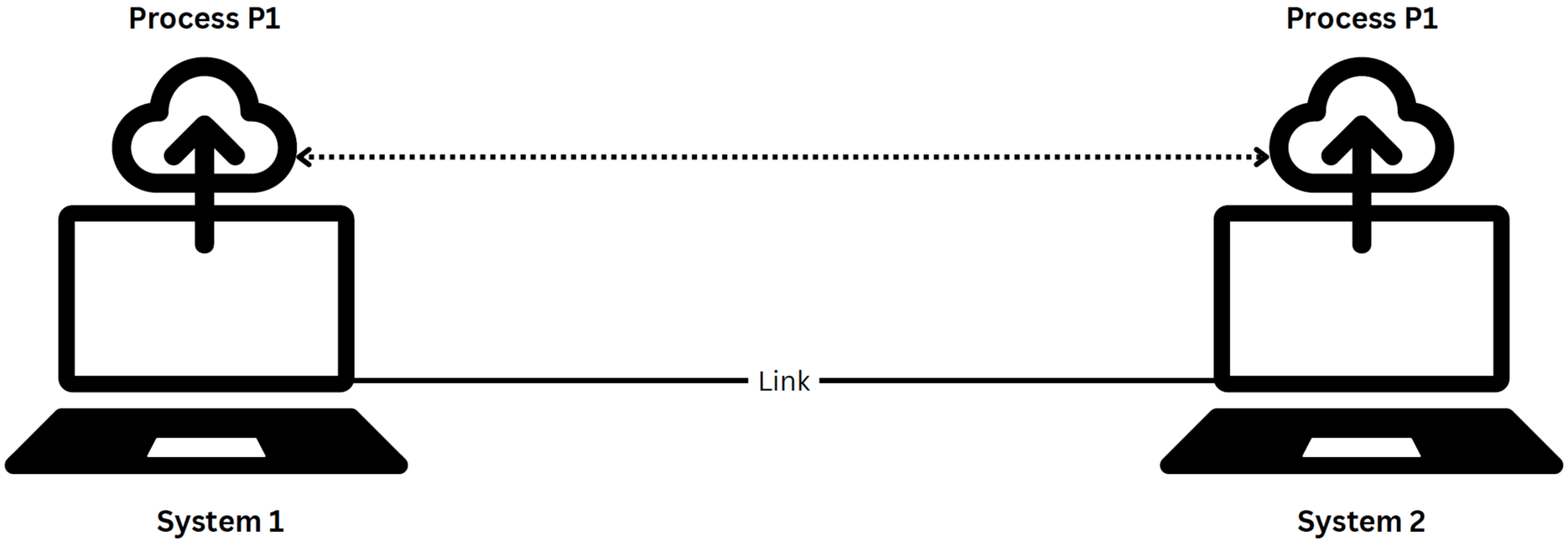
- Tables to configure:
  - How long should be the time slice?
  - How often to boost?
- 60 queues
  - Time slice length from 20ms to few 100 milliseconds
  - Priority boosted every 1 second
- Free BSD scheduler uses formula to calculate priority

MLFQ: Summing Up

- Five key rules are used by MLFQ:
  - If P(A) > P(B), A runs (B not)
  - If P(A) >= P(B), A & B runs in Round Robin using time slice of queue
  - When job enters, its placed in the highest priority
  - Once job uses its time allotment at a given level => priority is reduced
  - After a period, S move all jobs to top most priority queue
- BSD Unix derivatives, Windows NT and Solaris use a form of MLFQ in their basic scheduler

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?

Thank you

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

CS3.301 Operating Systems and Networks

Networking - Introduction (Sockets and Networking Layers)

Karthik Vaidhyanathan

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

SERC
Software Engineering Research Centre

Acknowledgement

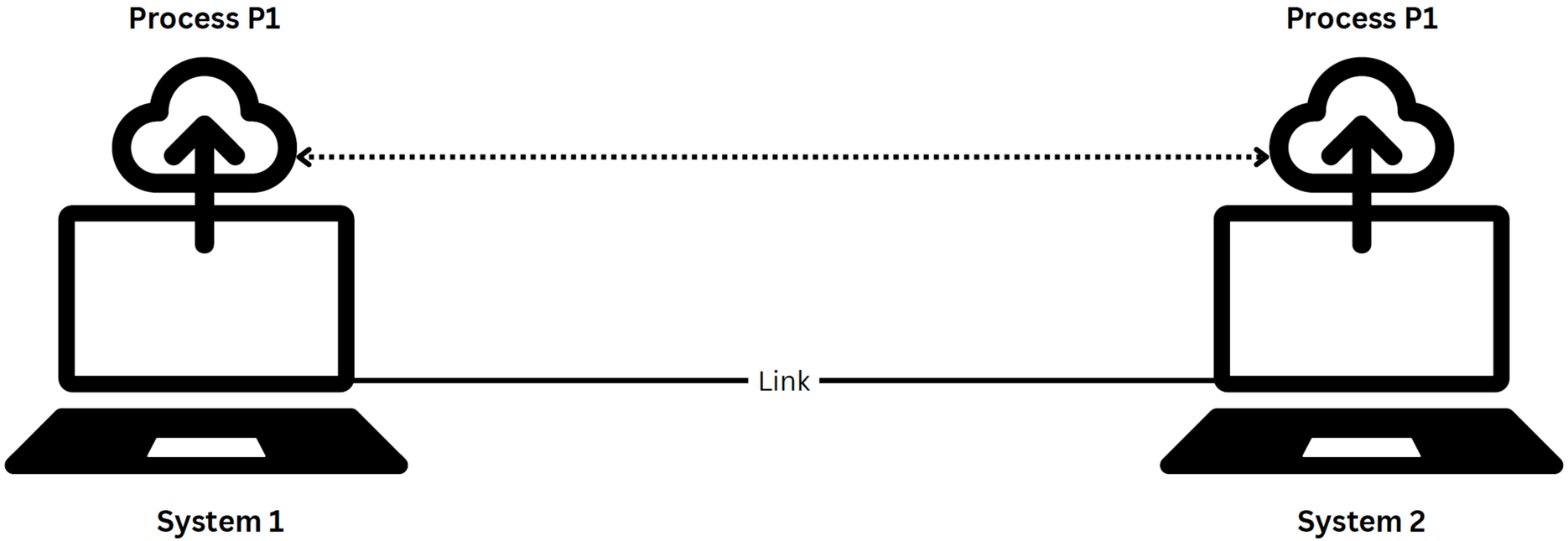
The materials used in this presentation have been gathered/adapted/generate from various sources as well as based on my own experiences and knowledge -- Karthik Vaidhyanathan

Sources:

- 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

What about processes in different machines?

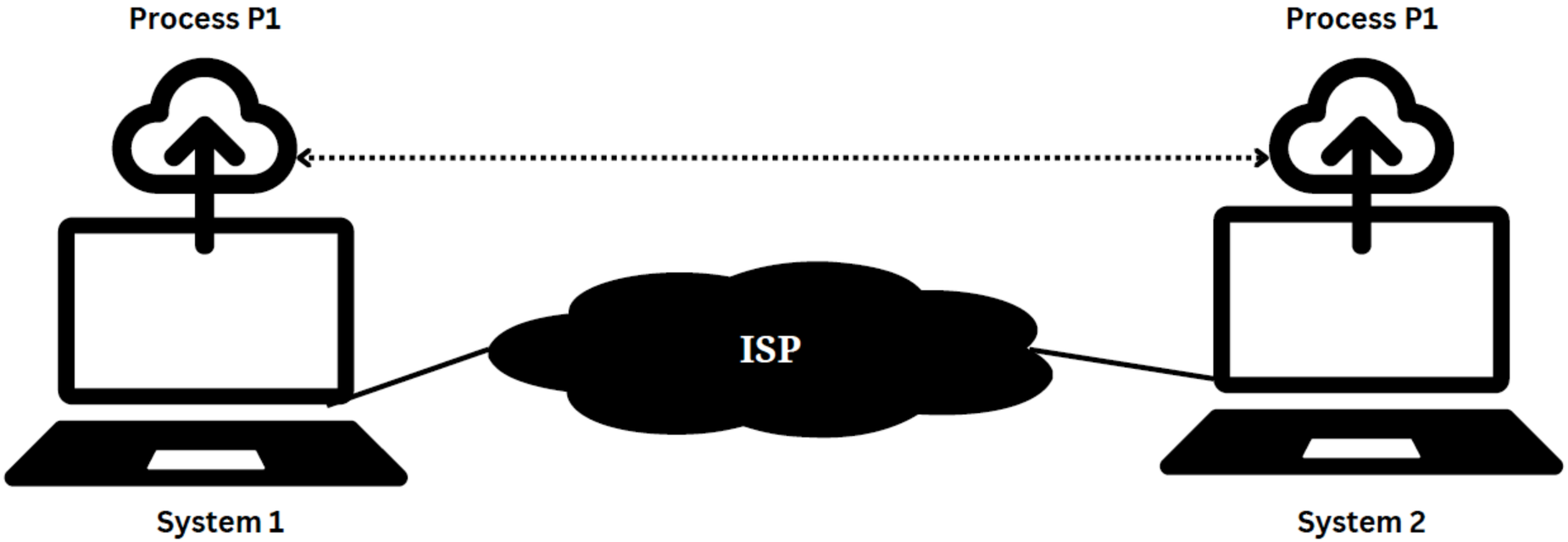
How can they communicate?



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?

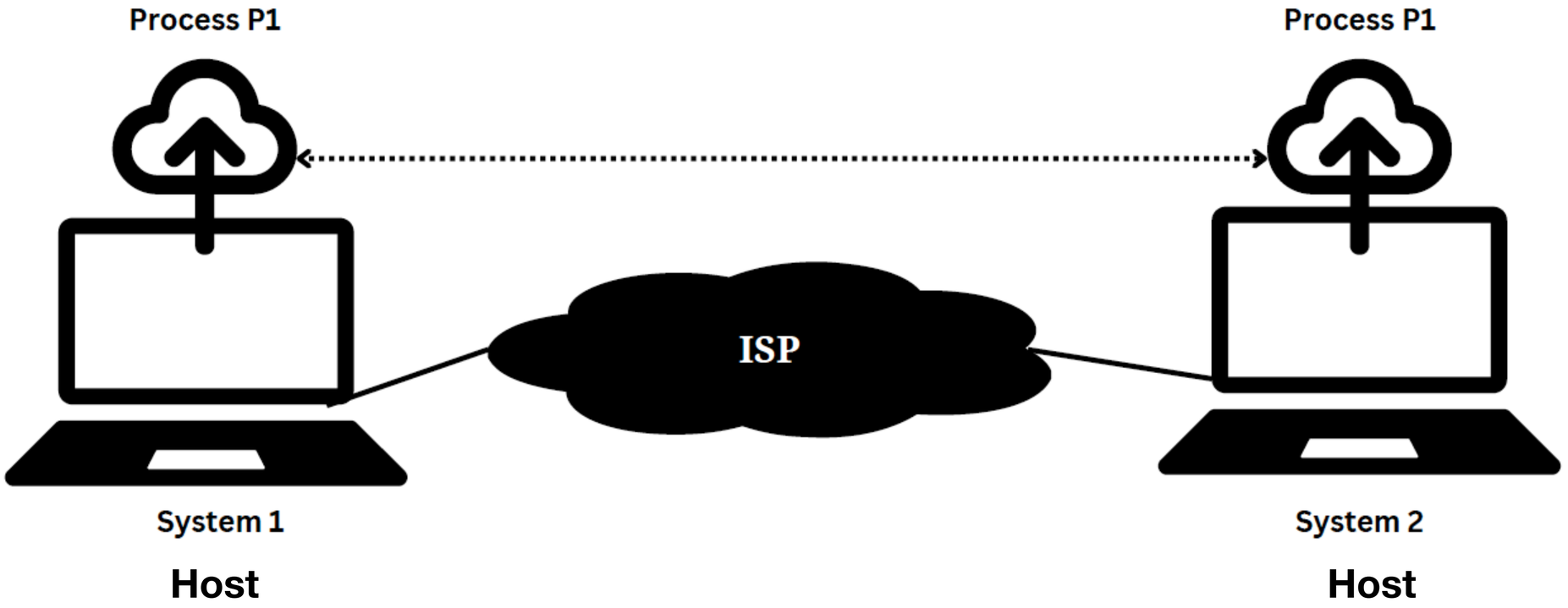
Let us expand it



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

Network Components

Host

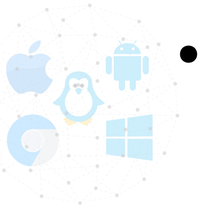


- Any device that send or receive traffic: Computer, laptop, smartwatch, phone, etc
- Host can be client or server
- Servers can sometime be clients too

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

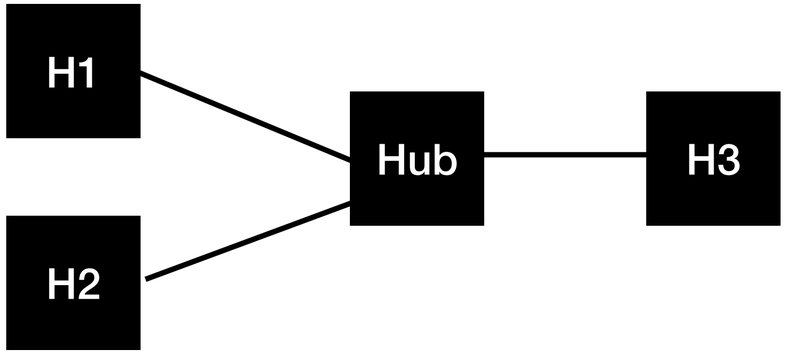
Repeater

- 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

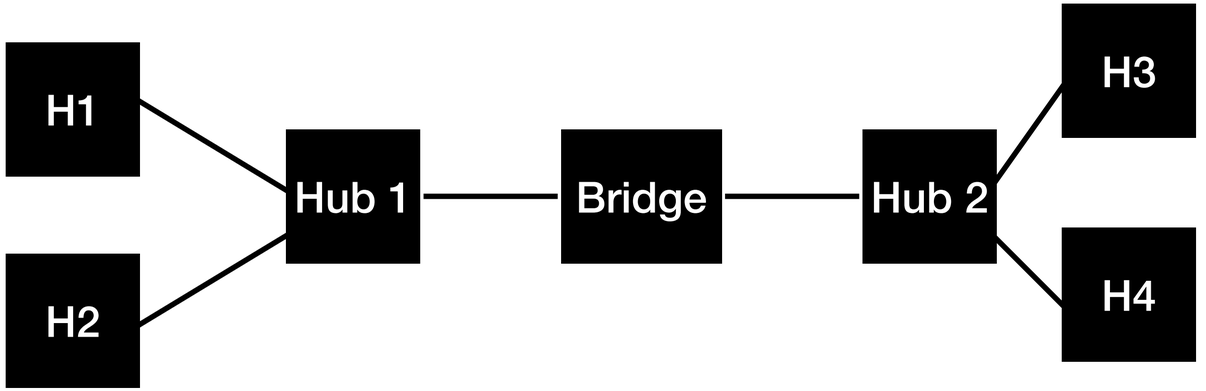
Hubs

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



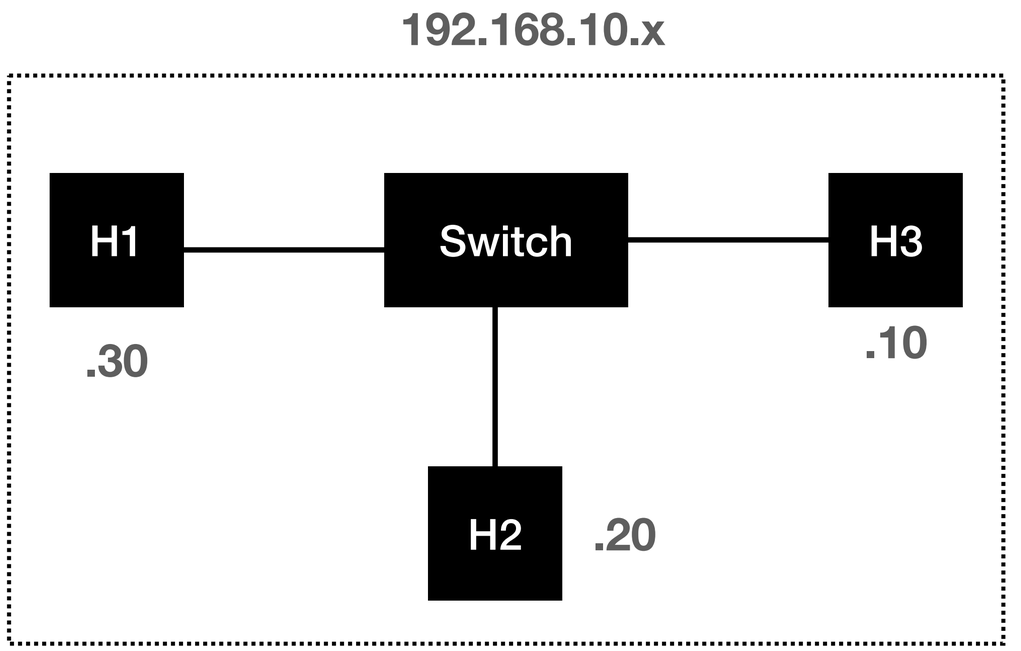
Bridges

- 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

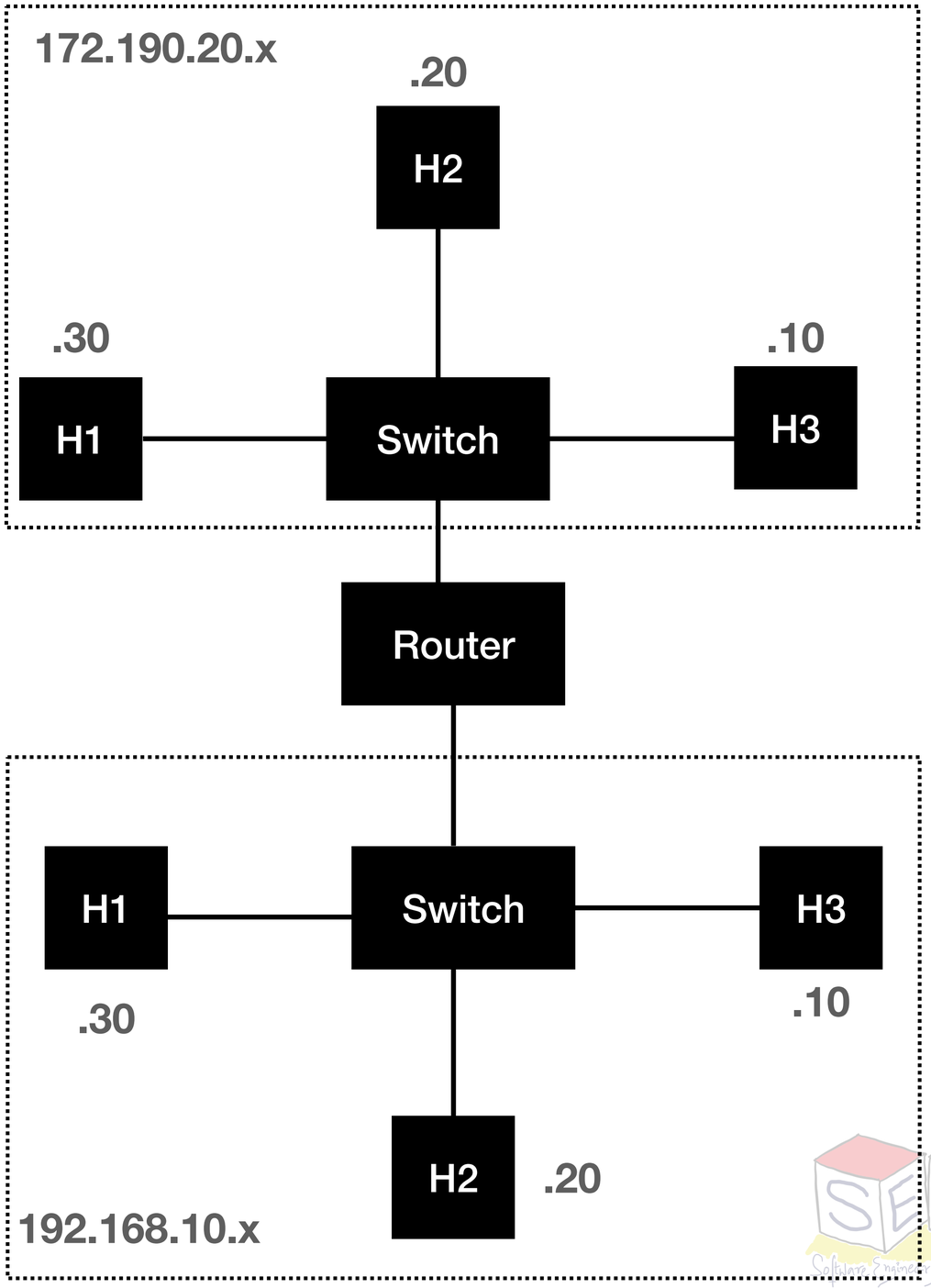
- Devices which facilitate communication within a network
- Combination of hubs and ports
- Knows or infers which hosts are on each port
- They have multiple ports
- Whatever connected to switch becomes part of one network



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

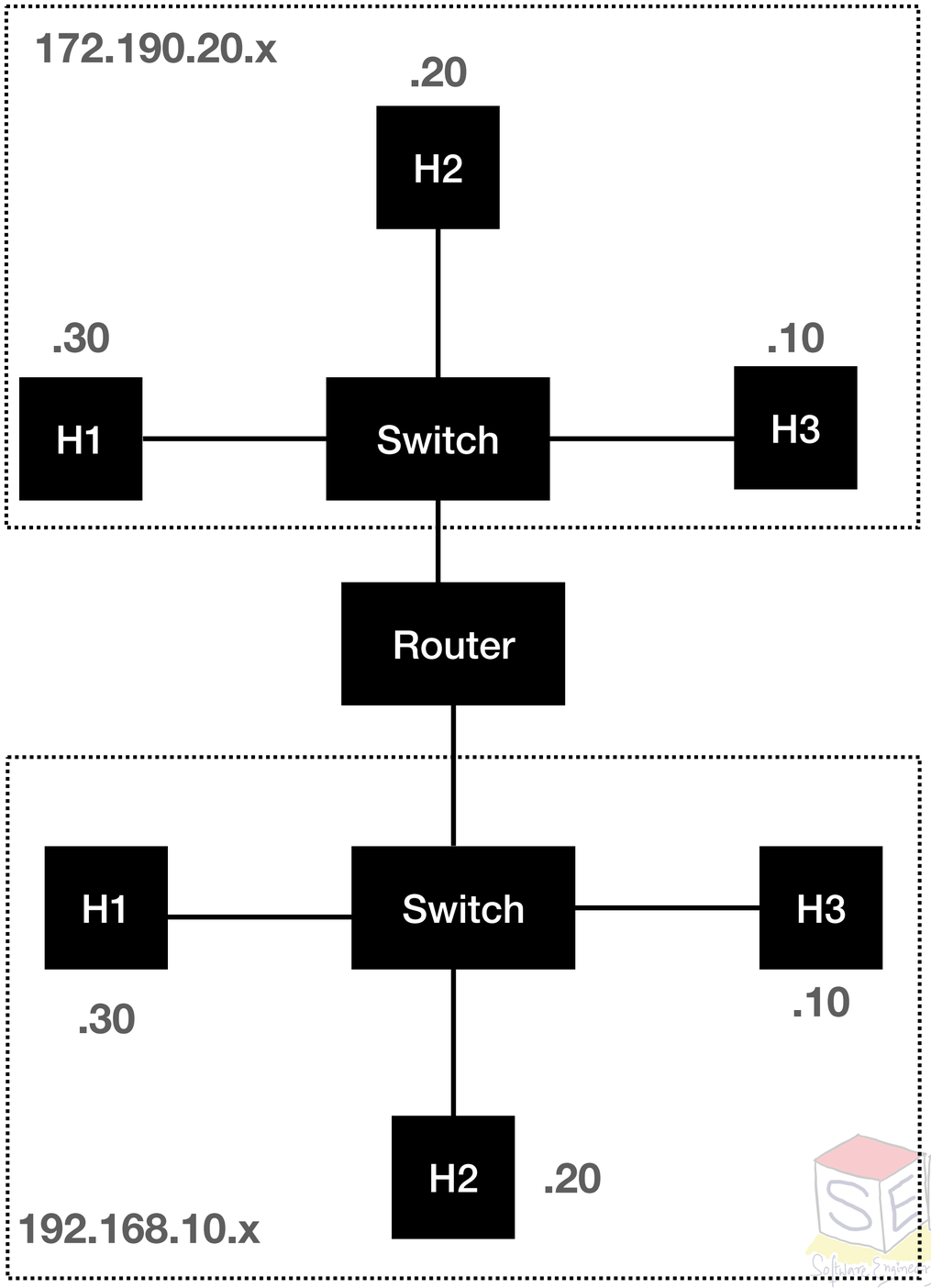
Routers

- 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

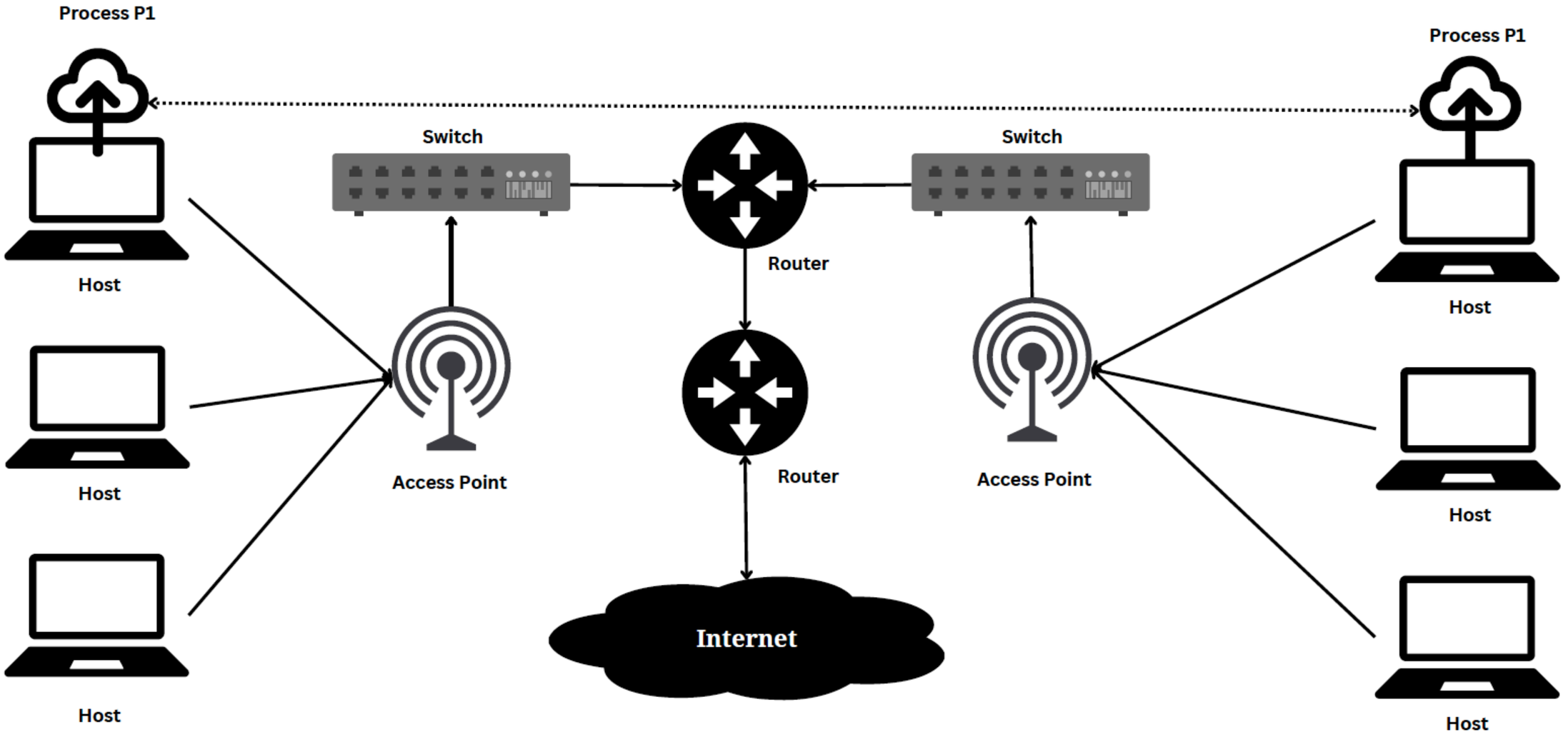


Routers

- 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

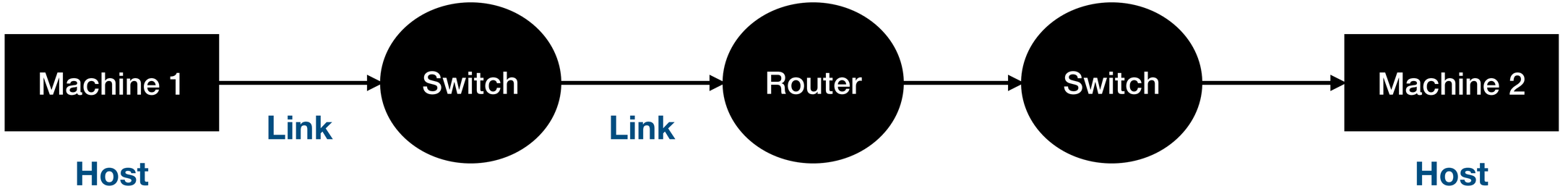


The Bigger Picture



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

Components of a network



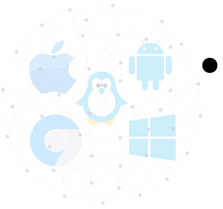
Process Communicating over network

- 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

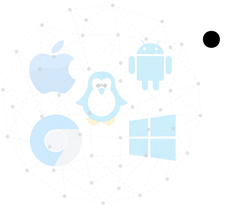
- **Personal Area Network - PAN**
  - Devices communicate over a very short range
  - Eg: Bluetooth (master slave paradigm)
- **Local Area Network - LAN**
  - Private network operating within and nearby a single building (home, office, factory, etc.)
  - Wireless LANs runs speed from 11 Mbps to 7 Gbps (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

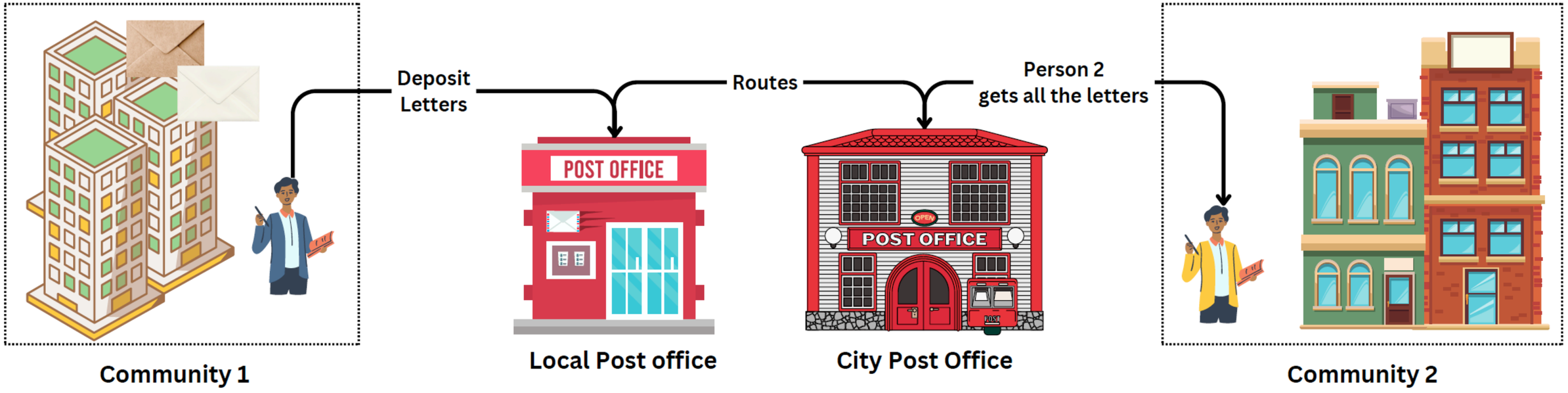
Two large communities in different parts of the city, two people in each community to collect and deliver the letters/couriers

An Illustrative Scenario

People in community send letters/couriers

One person is collecting all the letters/couriers

Person delivers the letter to each person

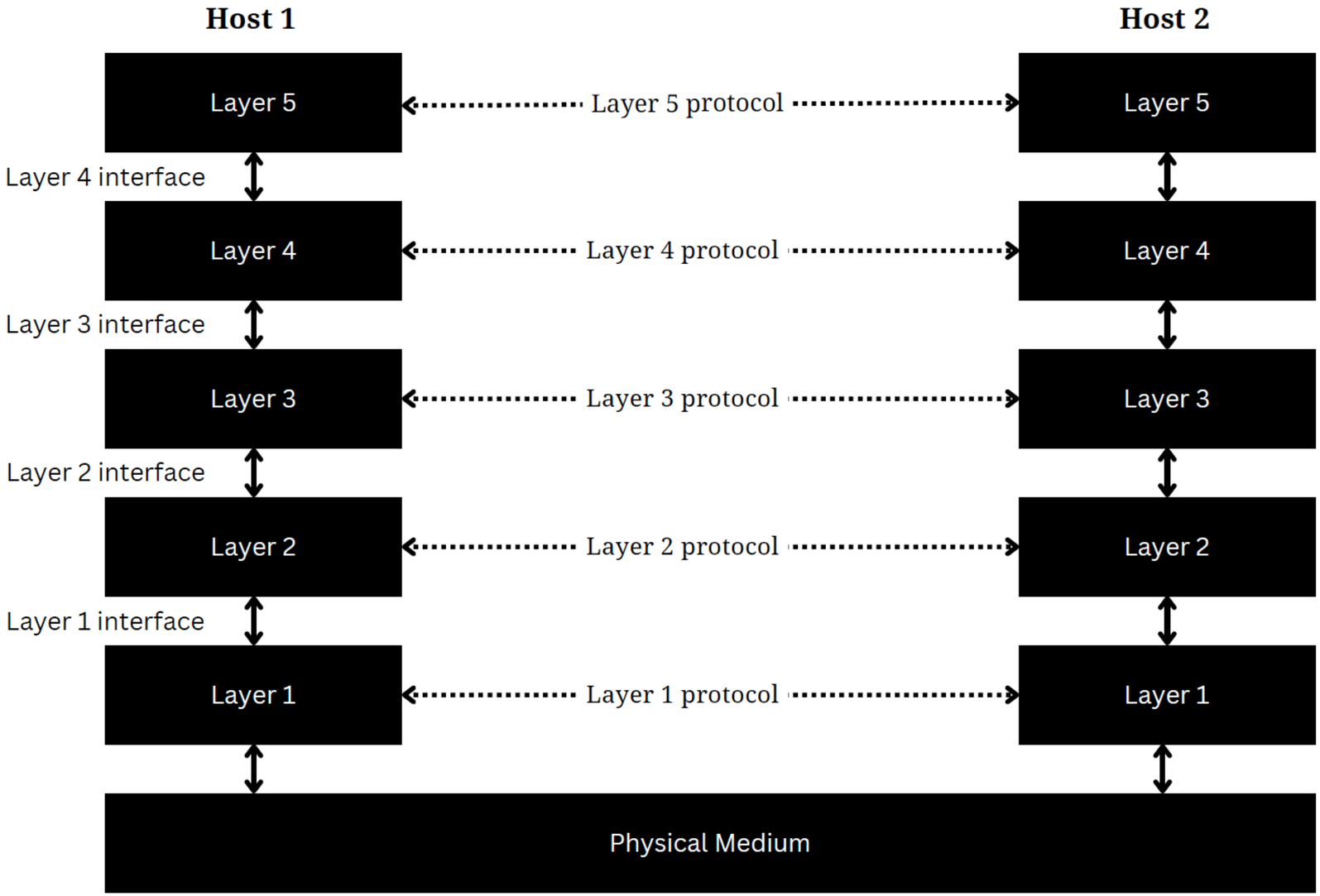


Key points: Door number, building number, Post office has an identifier, etc.

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

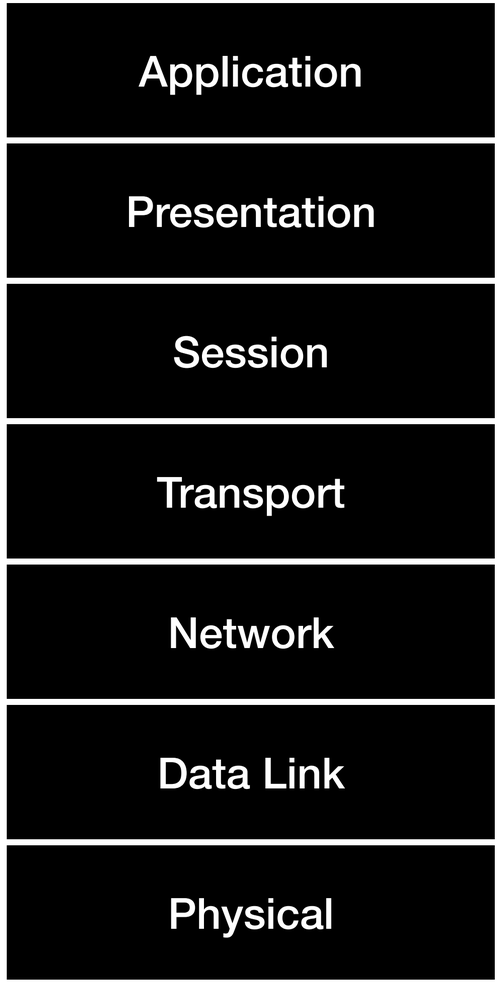
Networking Layers



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

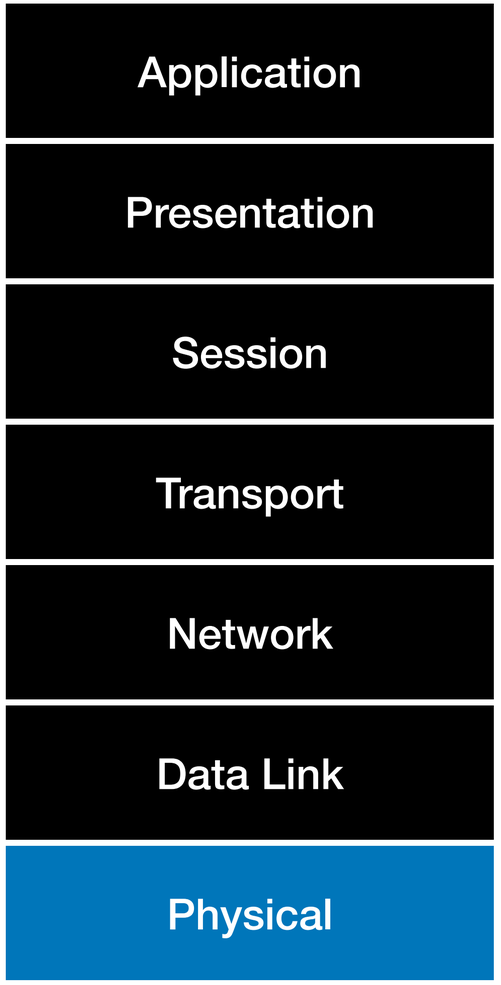


- Open System Interconnection (OSI)
- A Conceptual framework used to understand how communication works through different layers
- Divides the network communication process into seven layers
- Developed to facilitate interoperability between different technologies
- Each layer has a specific function.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

Physical Layer (L1)

Ultimately everything is 0's and 1's

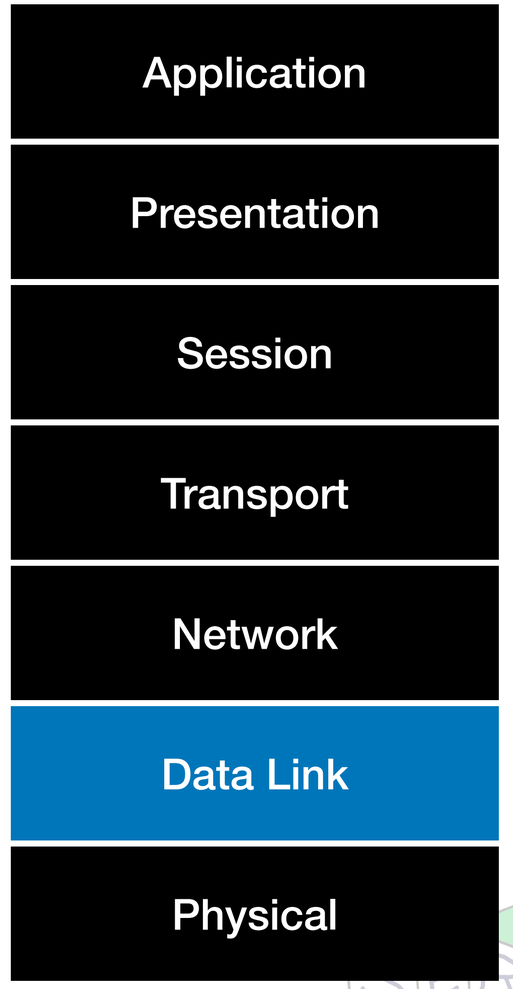
- Data is in the form of bits - 0s and 1s
- Something has to transport the bits from one machine to another - Physical layer
- Concerned with transmission of raw bits over physical medium, like a cable
- L1 technologies: Ethernet cables, Optical fiber, Coaxial cable, etc.
    - Even WiFi is L1 technology,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



Data Link Layer (L2)

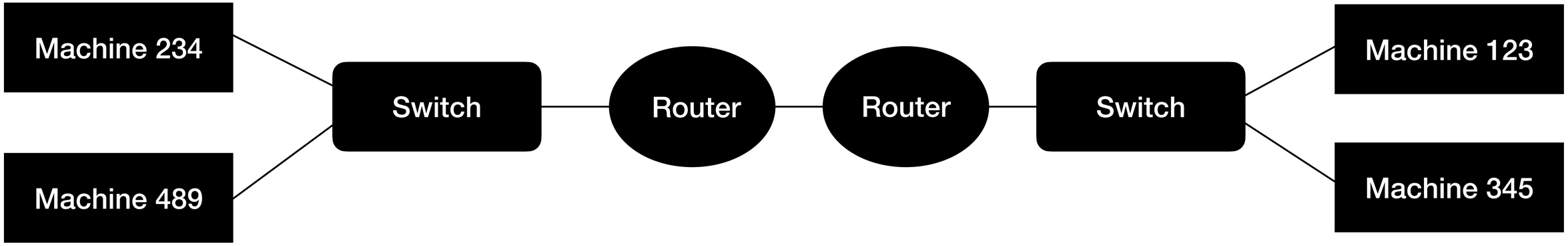
Hop to Hop connectivity

- 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

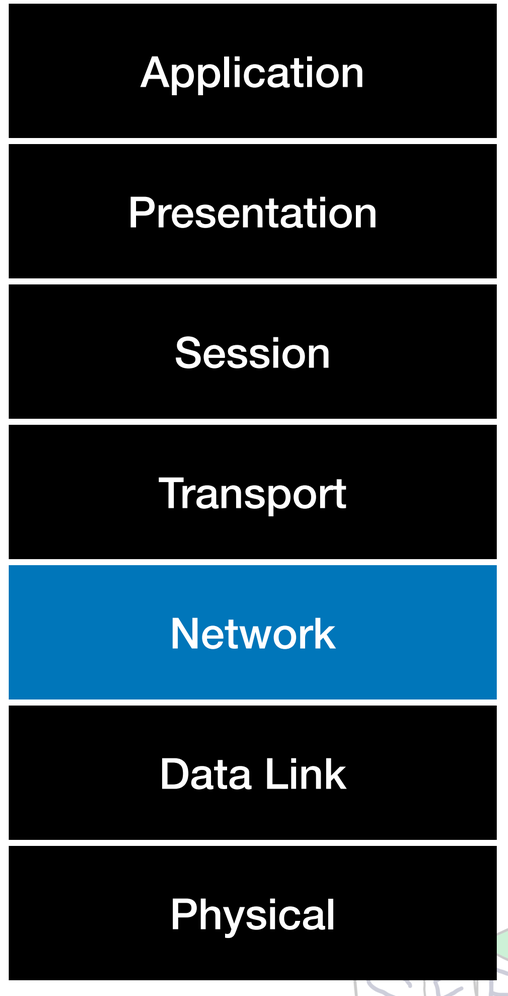
What about communication from 234 to 345?



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

Network Layer (L3)
End-to-end Communication

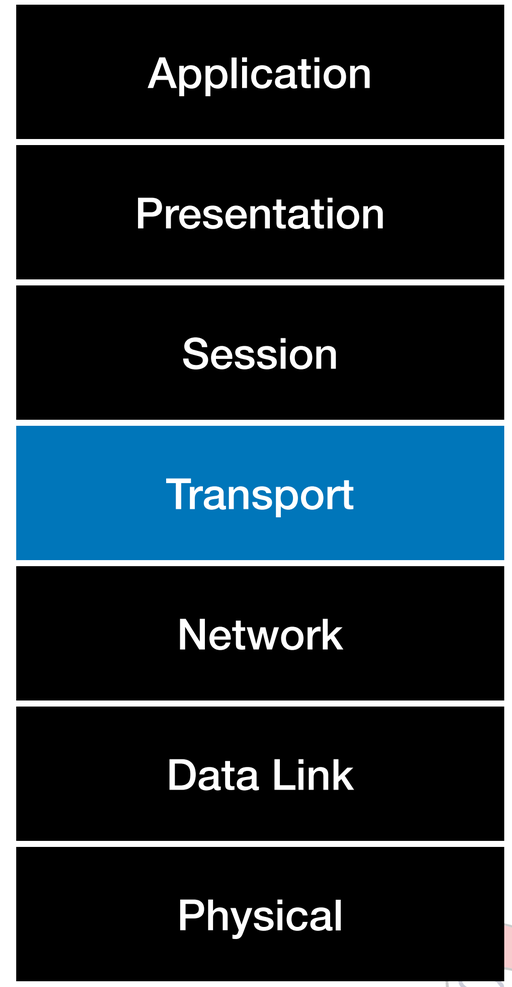
- Manages routing through different routes in a large network
- Uses an addressing scheme - IP addressing
  - 32 bits represented as 4 octets (IPv4)
- Performs functionalities such as Logical addressing (IP), Path selection and packet forwarding
- L3 technologies: routers, even hosts are L3, L3 switches



Transport Layer (L4)

Service to Service

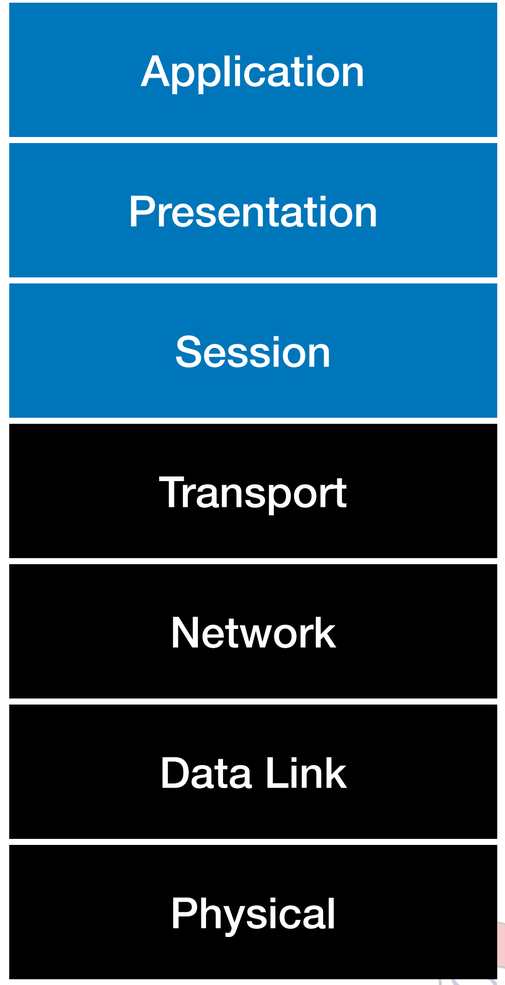
- How to ensure that the right process receives the data?
  - Many process will be executing or are waiting to execute?
- Ensures data transfer is reliable, sequential and free from others
- Manages flow control and error correction
- Layer 4 has an addressing scheme to guarantee message delivery
  - 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



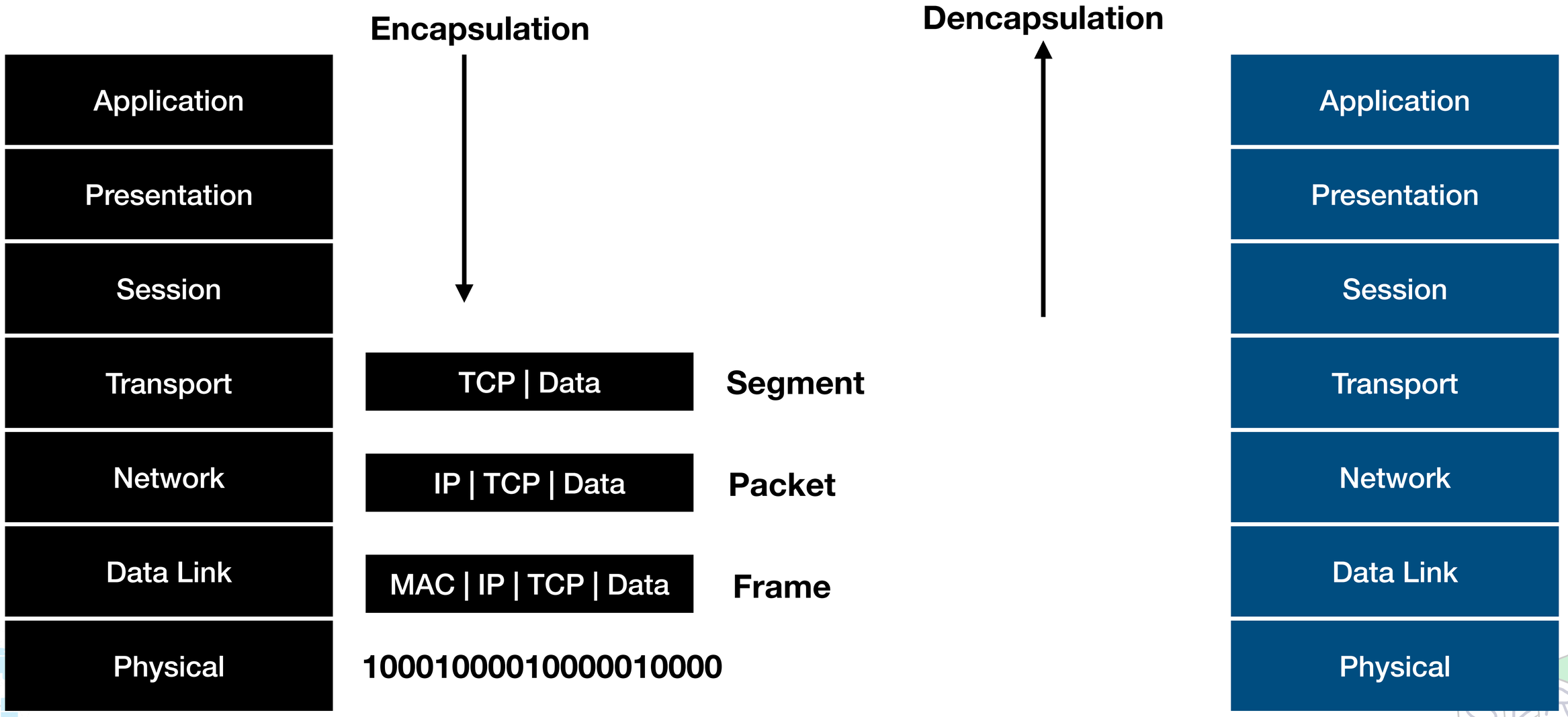
Session, Presentation and Application

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



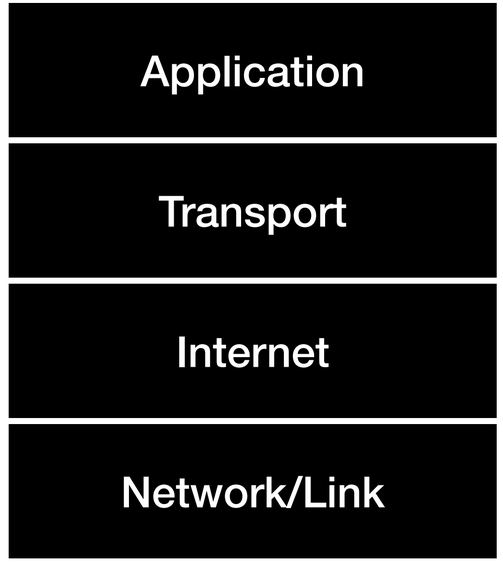
Putting It Together



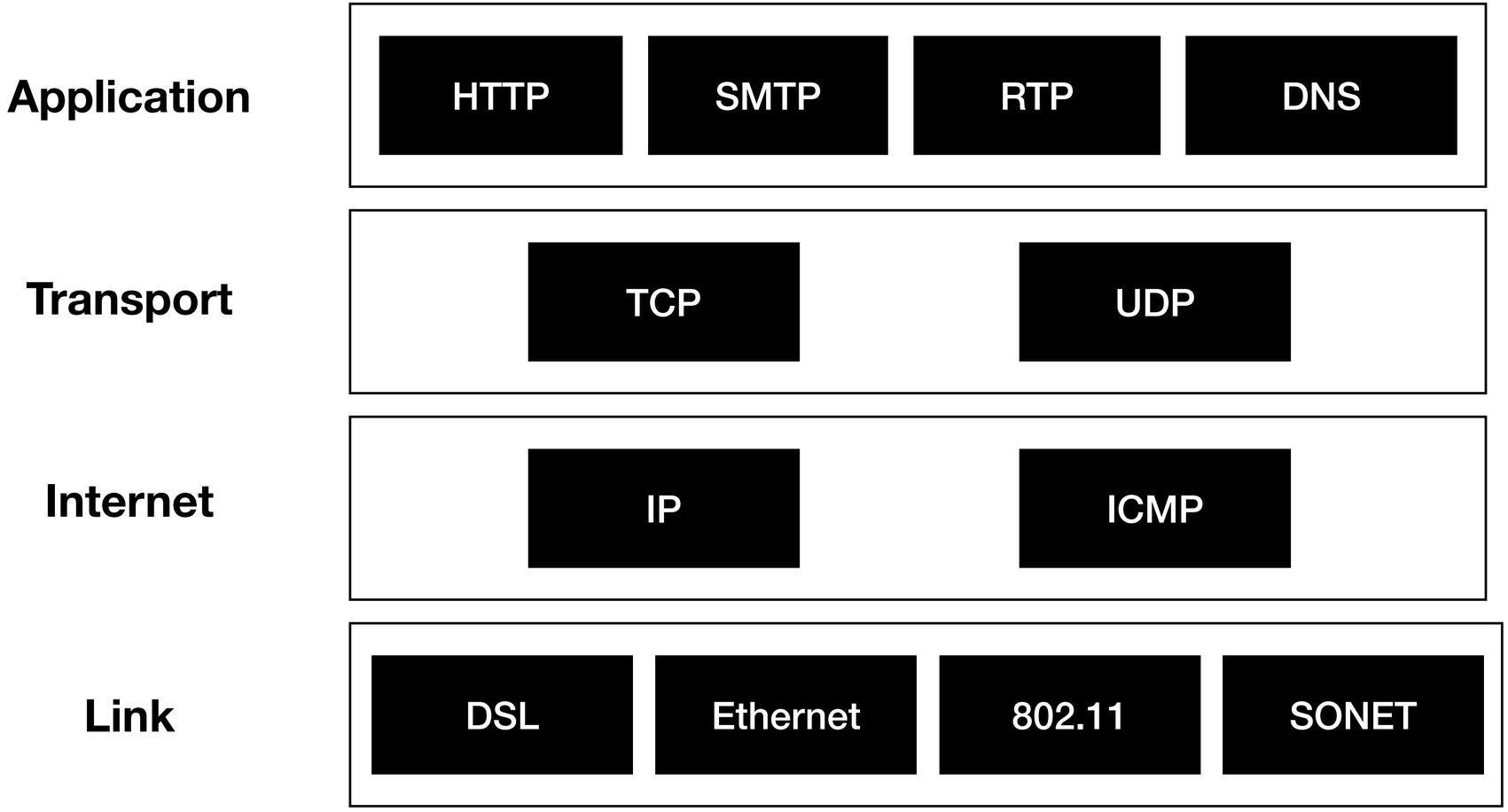
The 4 Layer Model

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



Network Protocol Stack



Thank you

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

CS3.301 Operating Systems and Networks

Transport Layer and how it works!

Karthik Vaidhyanathan

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

SERC
Software Engineering Research Centre

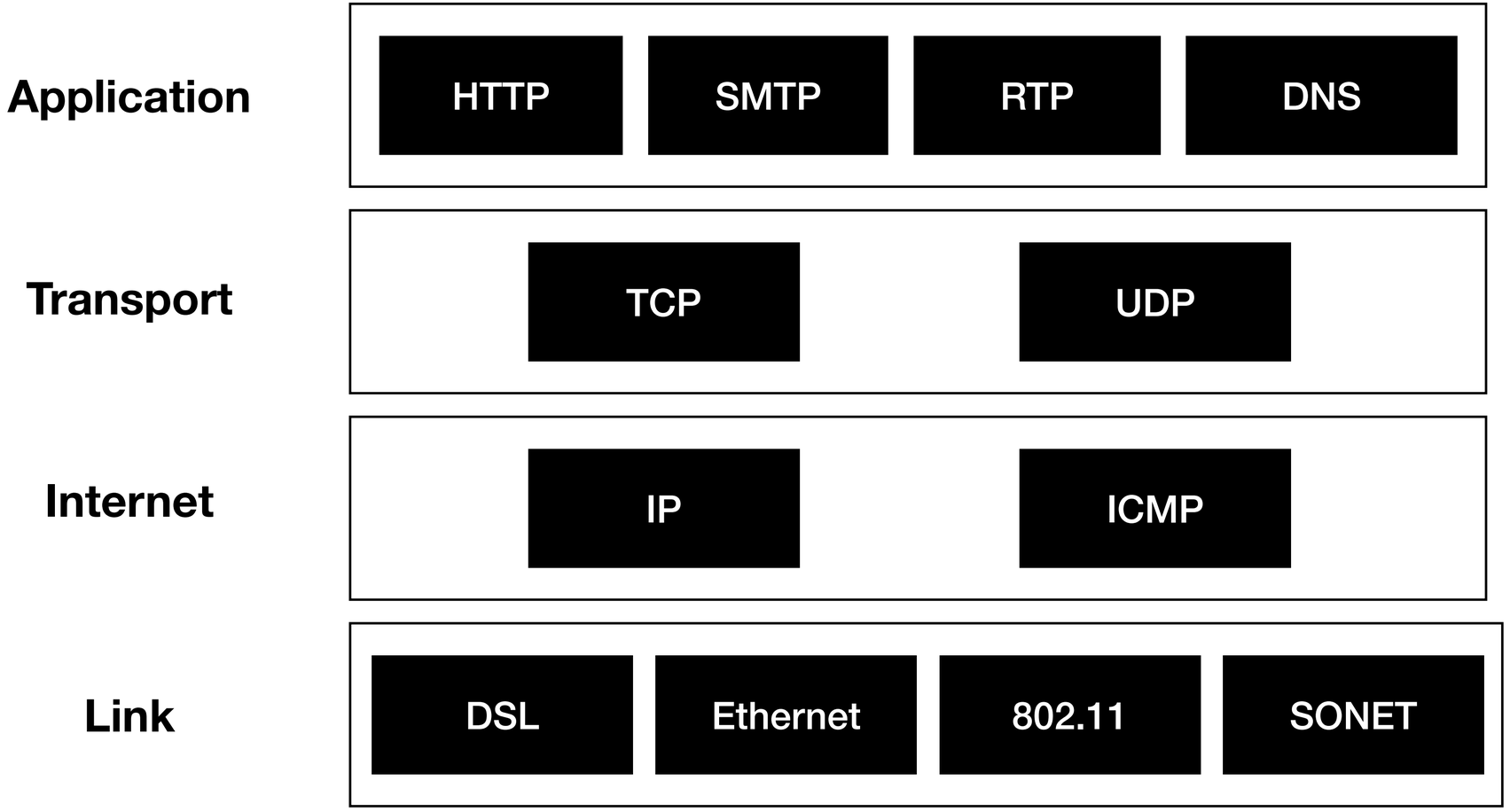
Acknowledgement

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:

- 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

Network Protocol Stack



Onto Transport Layer

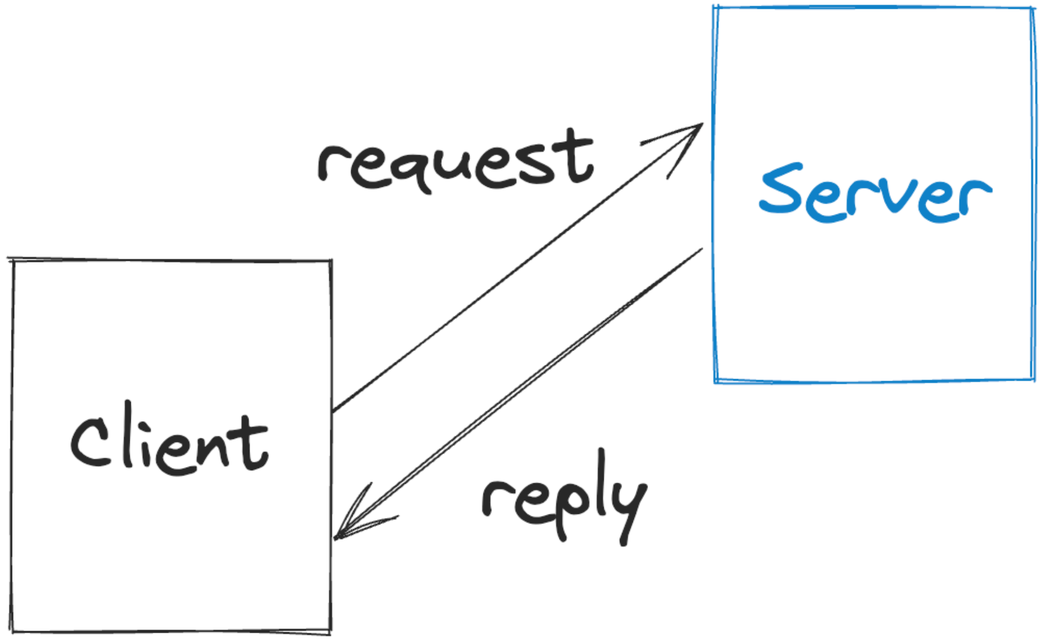
Service to Service Delivery

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

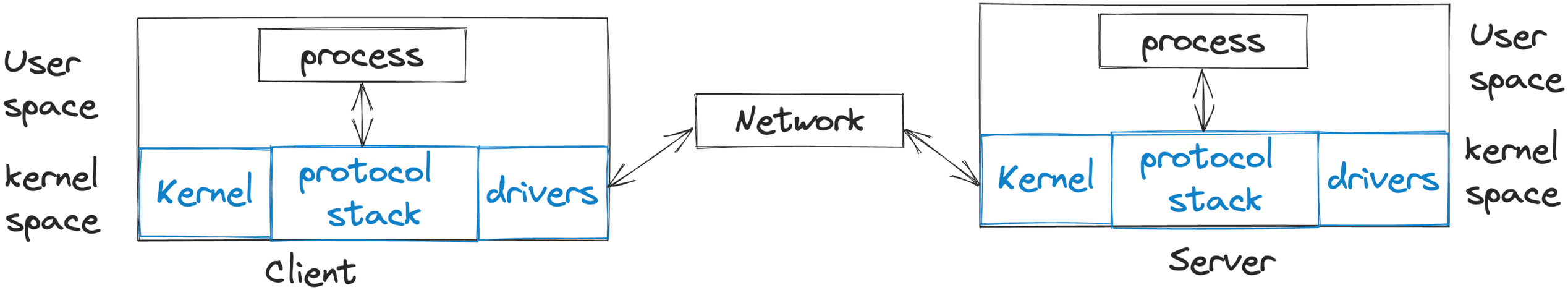
Lets go back

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



- 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

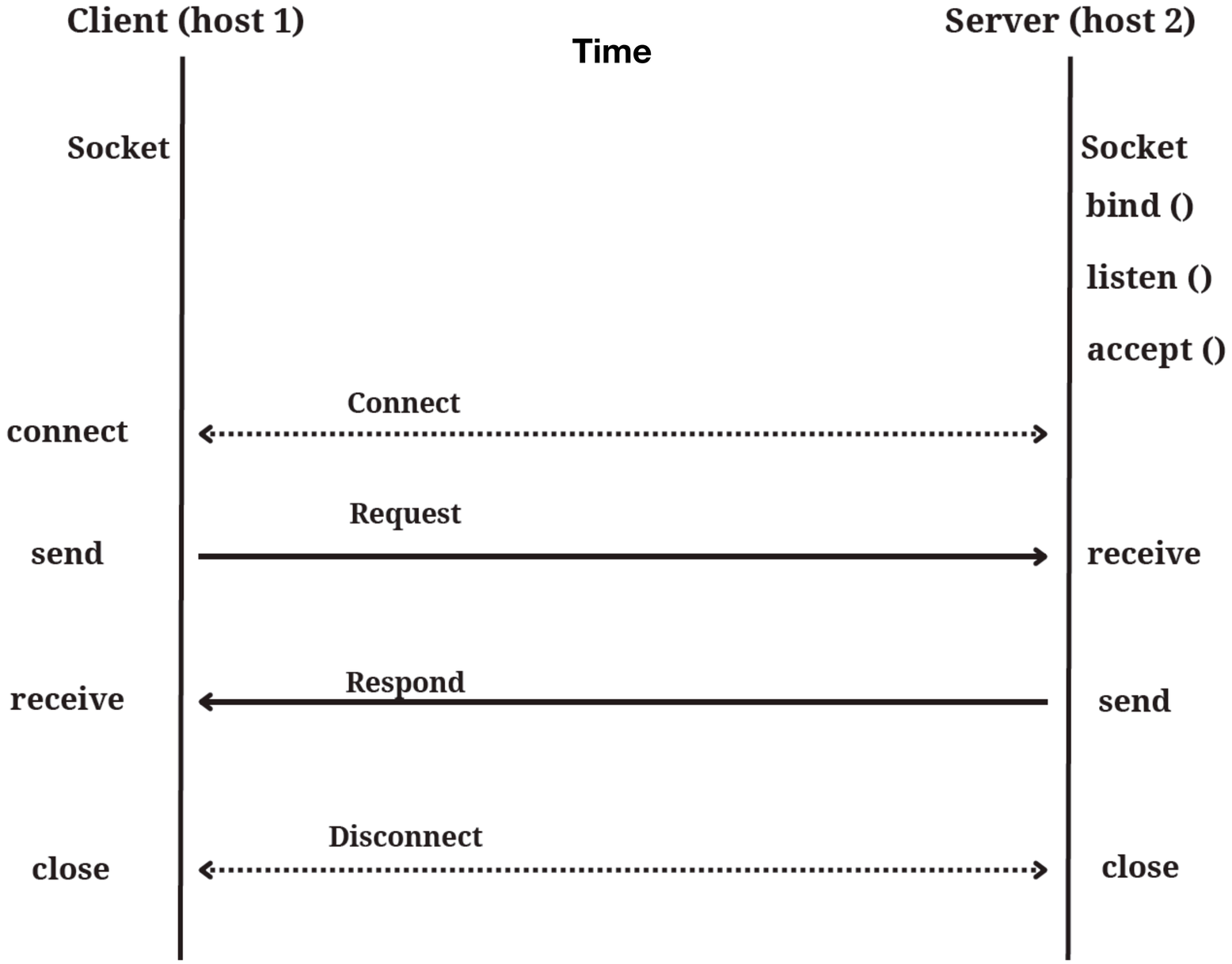
The Socket API

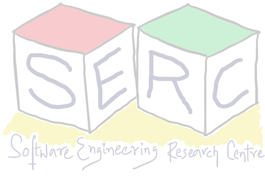
- Simple abstraction to use the network
- The network service API used to write all network applications
- Part of all OS and all language [Berkley Unix, 1983]
  - Allows user space applications to interact with networking subsystem
- Two services:
  - **Streams:** Reliable (Connection-oriented)
  -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

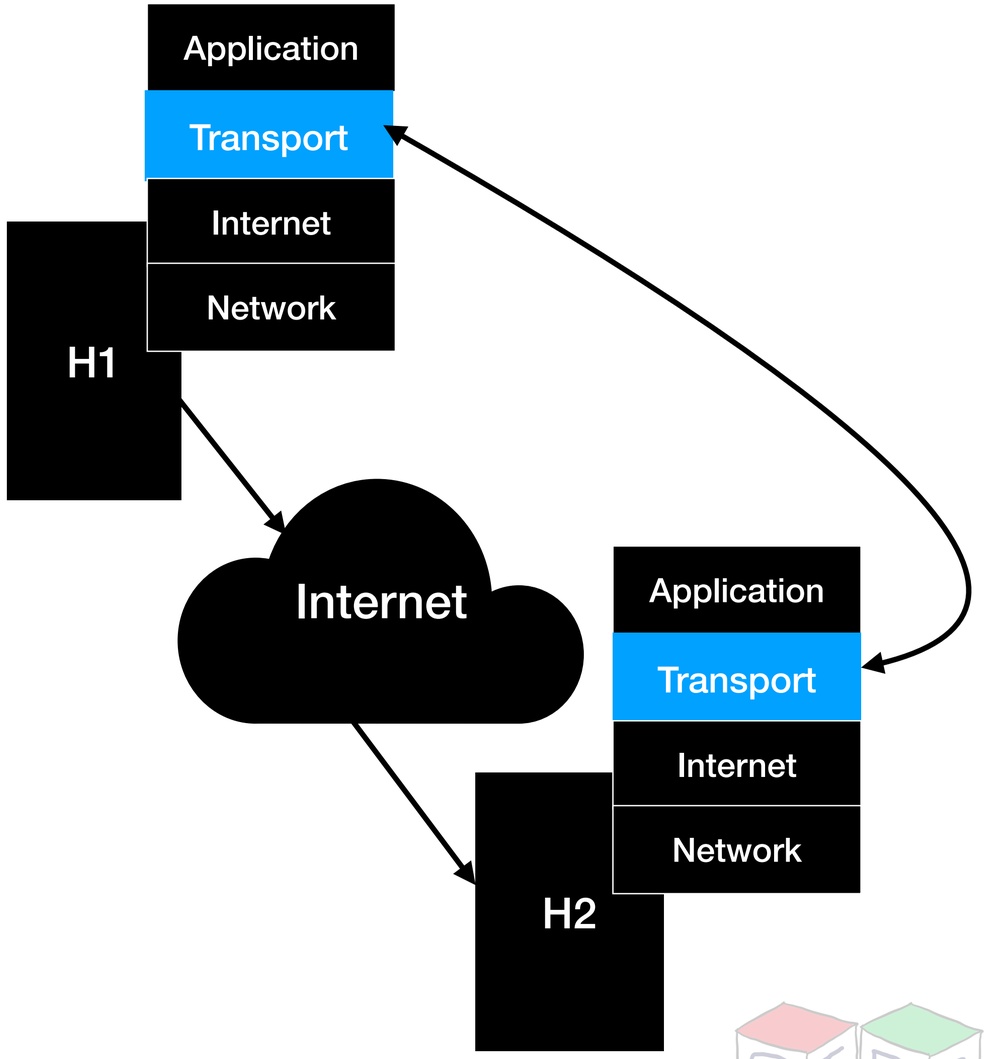
Let us take a step back

Types of Links

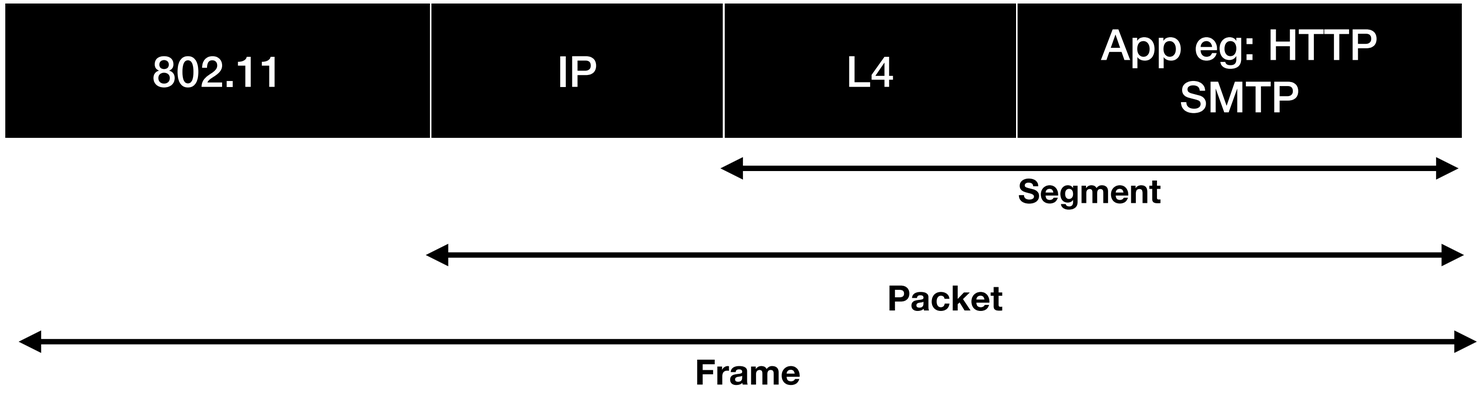
- **Full Duplex**
  - Bidirectional
  - Both sides at the same time
- **Half-duplex**
  - Bidirectional
  - Both the sides but only one direction at a time (eg: walkie talkies)
- **Simplex**
  - Unidirectional

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



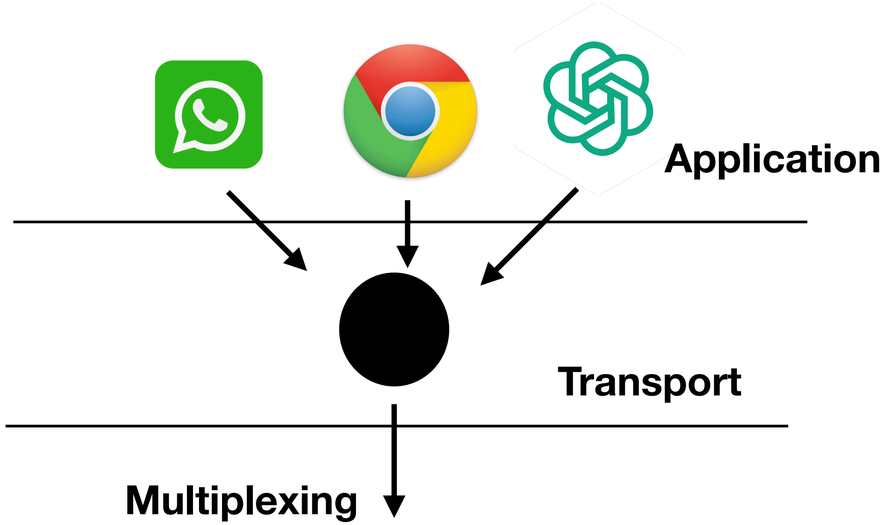
Quick Recap



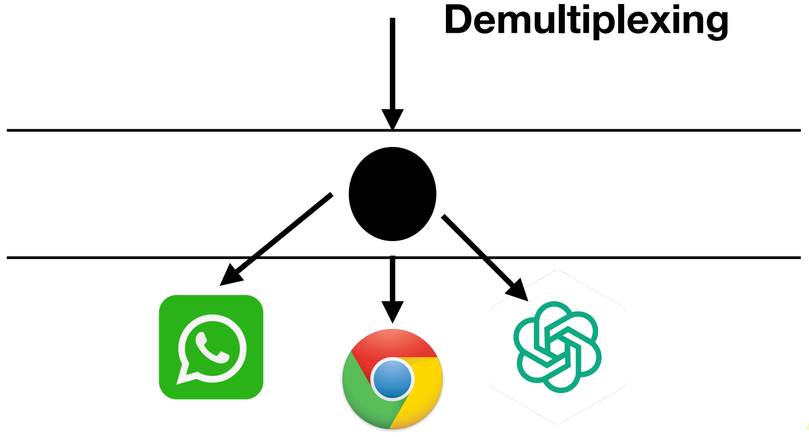
- Segments carry data across the network
- **Segments** are carried within the packets, within frames
- Each layer adds a **header** (Above L4 will be replaced by its header)

Multiplexing and Demultiplexing

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

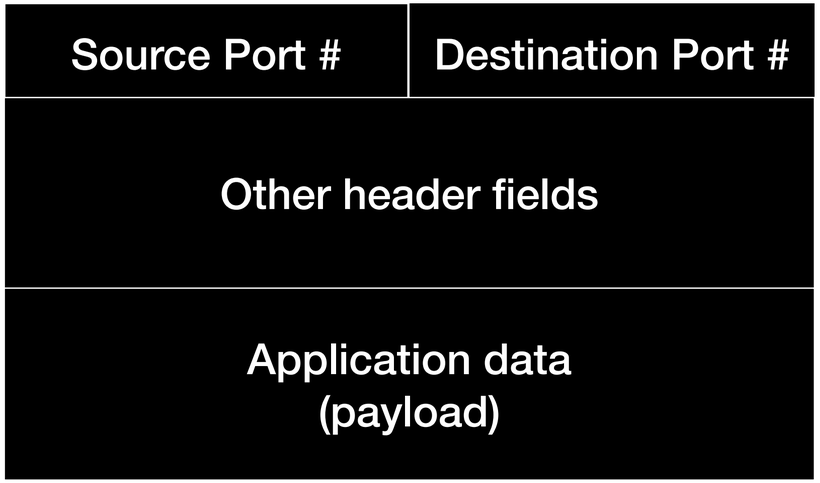


- **Demultiplexing as receiver**: Use header info to deliver received segments to correct socket



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



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

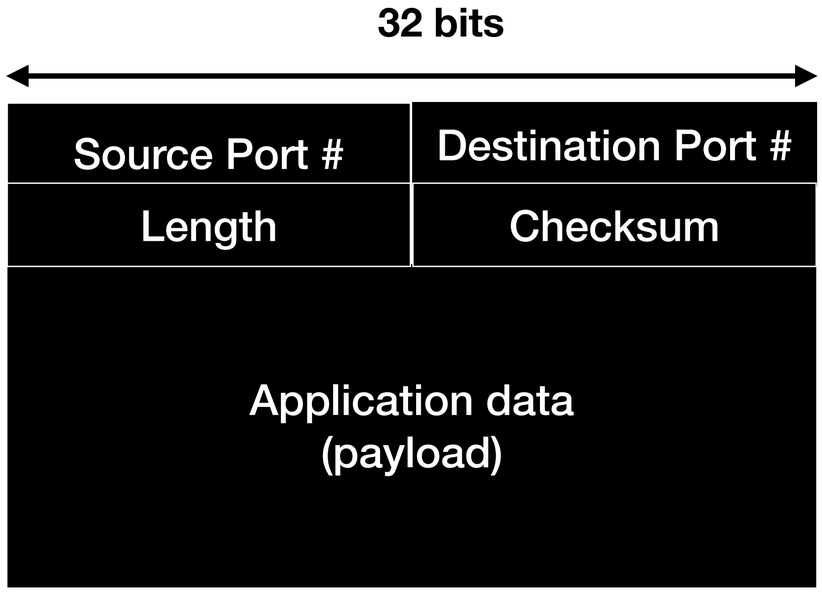
Flow Control and Overhead

- 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

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



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?

Thank you

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