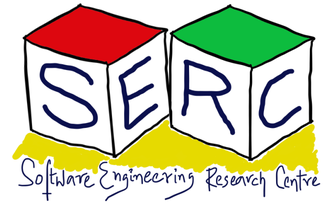
CS3.301 Operating Systems and Networks

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

Quick Recap of Scheduling

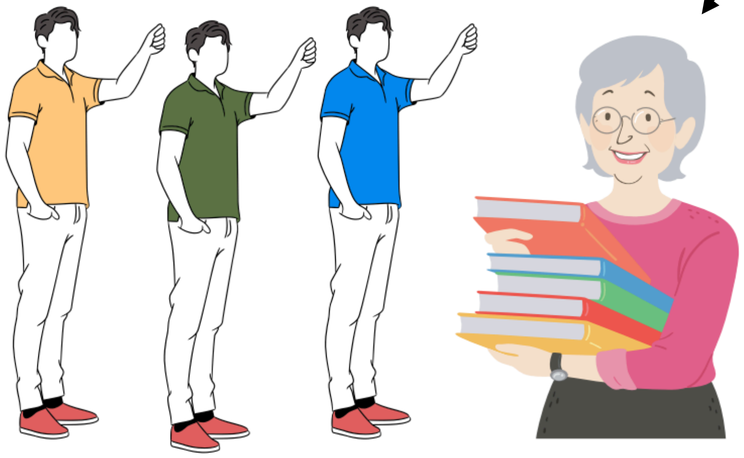
- Hardware provides support with some mechanisms - LDE
- OS switches between processes - Context switch
- Some policy is required to decide which process to execute - Scheduling
  - Different types of scheduling policies exist
  - Workload estimates help
  - Metrics - Turnaround time, response time, fairness
  - Policies - FCFS, SJF, STCF, RR, RR with I/O

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?

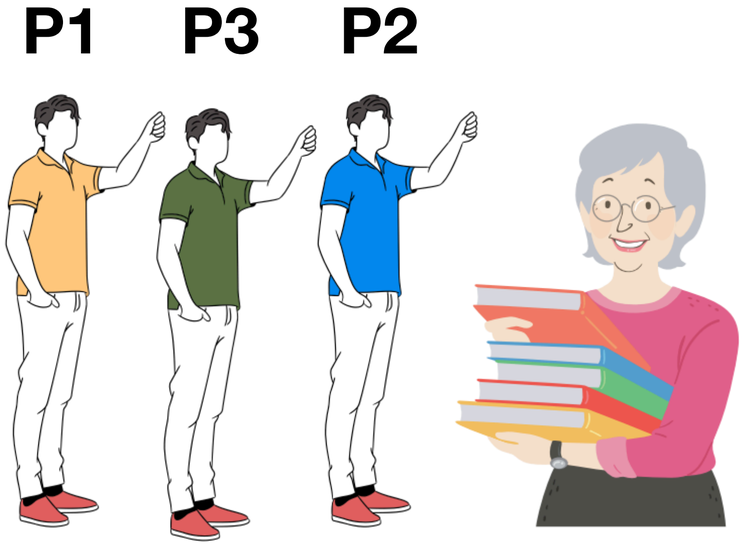
Why don’t we Prioritise?

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?

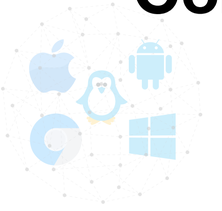
Multi Level Feedback Queues (MLFQ)

Two main features

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

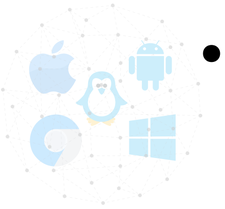
Can the policy learn continously to optimise response time and turnaround time?

Constraint: No apriori knowledge of the job length!

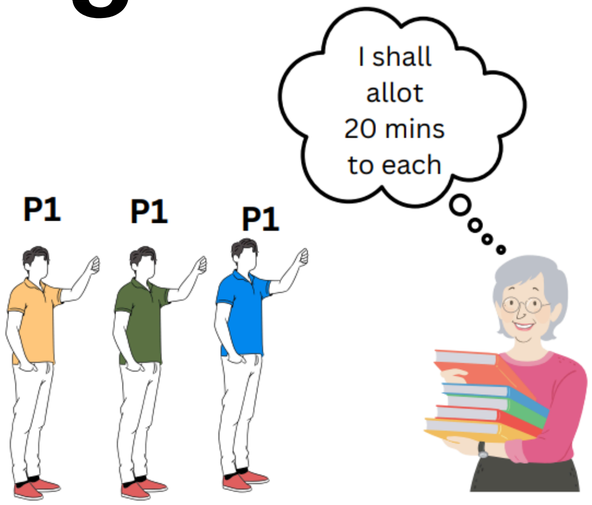


MLFQ: Basic Rules

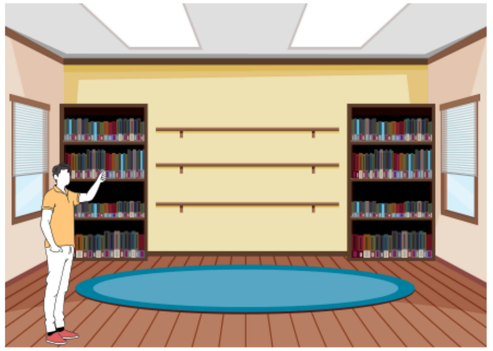
- Use **n** number of distinct queues
  - Each queue has a different priority level
- Use priority to decide which job should be run at a given time
  - A job with a higher priority => job on a higher queue
- **Key idea:**
  - Scheduler sets priority to different jobs
  - Keep updating the priority based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



Going back to the example



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

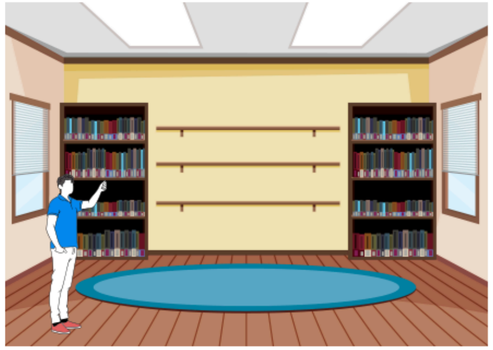


First person gets the chance!

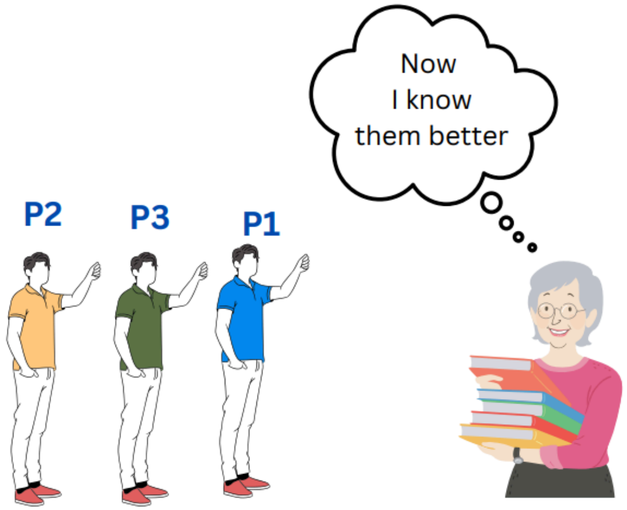
out in 18 mins

Next person goes in

20 mins!!



0ut in 15 mins!



For the next visit, priorities have been updated

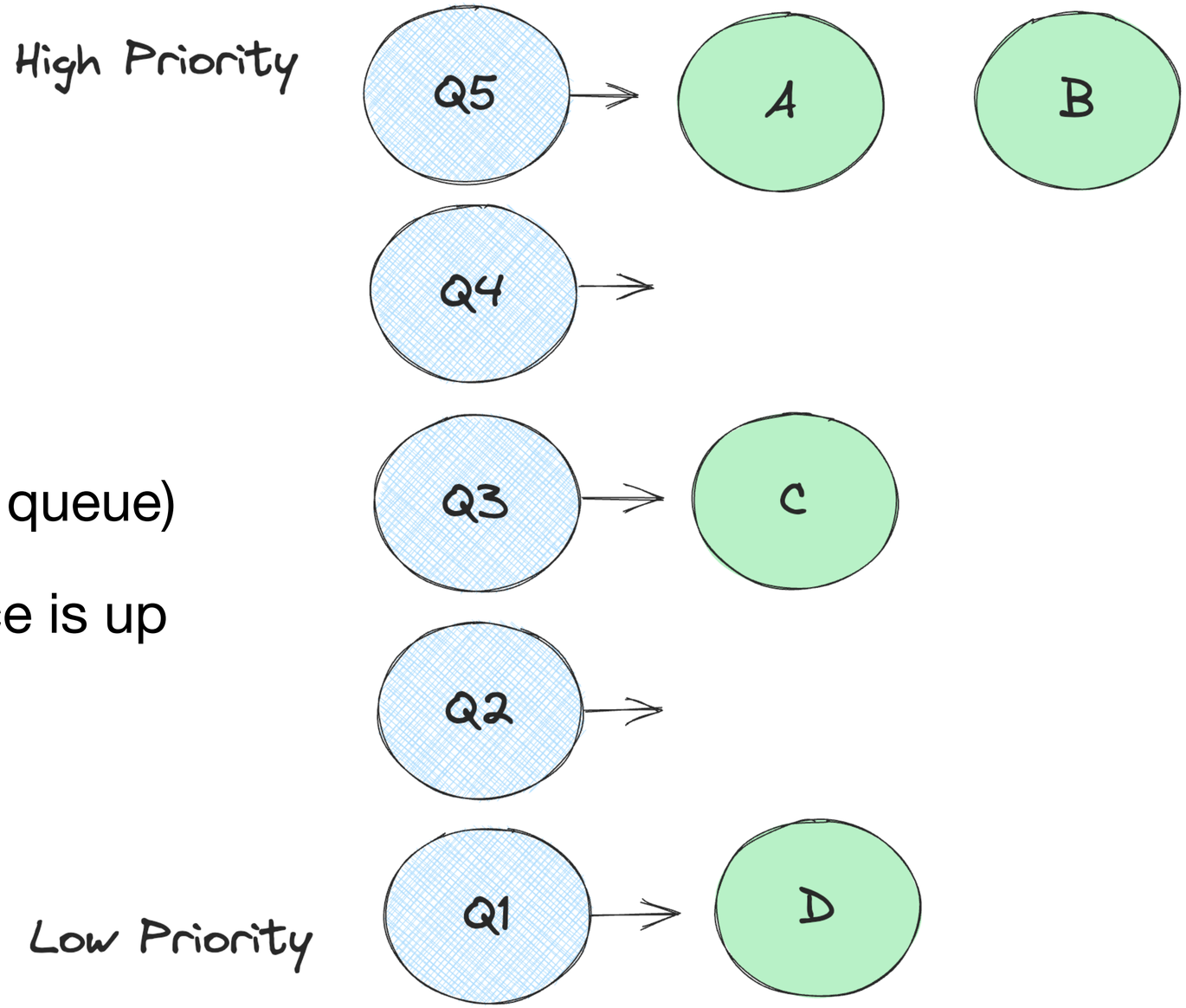
MLFQ

- 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

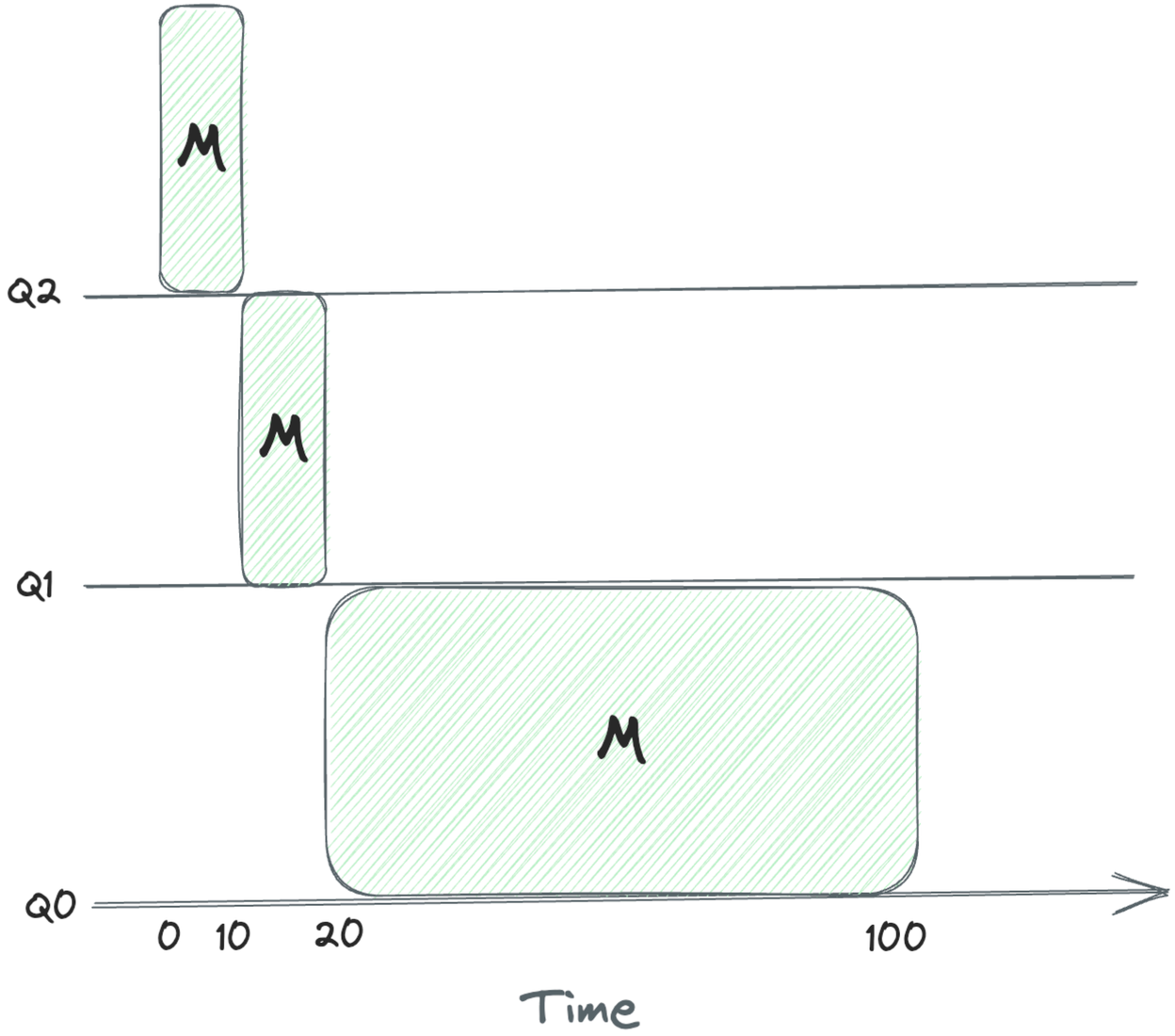
Five queues and 4 jobs

- 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

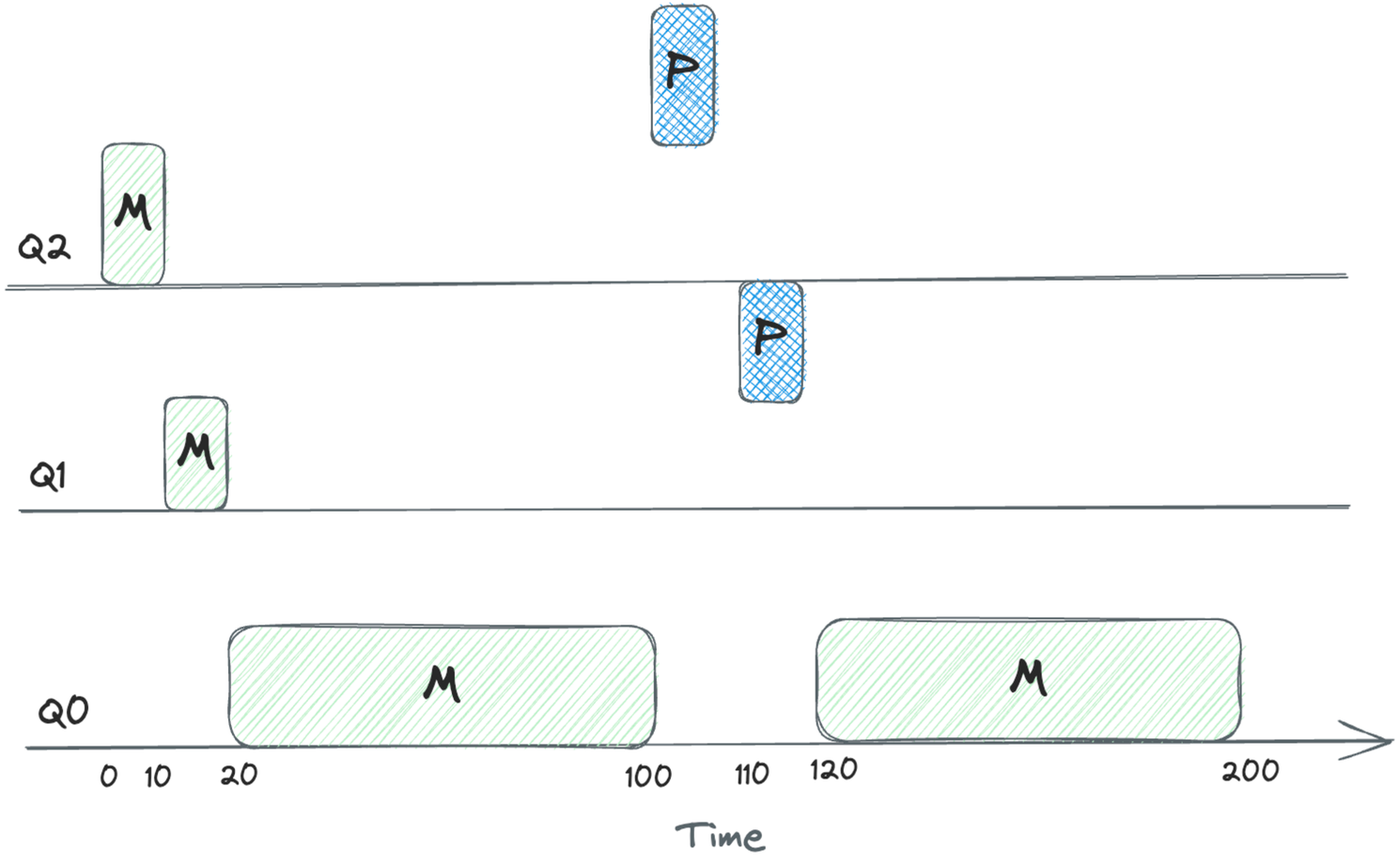


Example: Short job enters

- 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

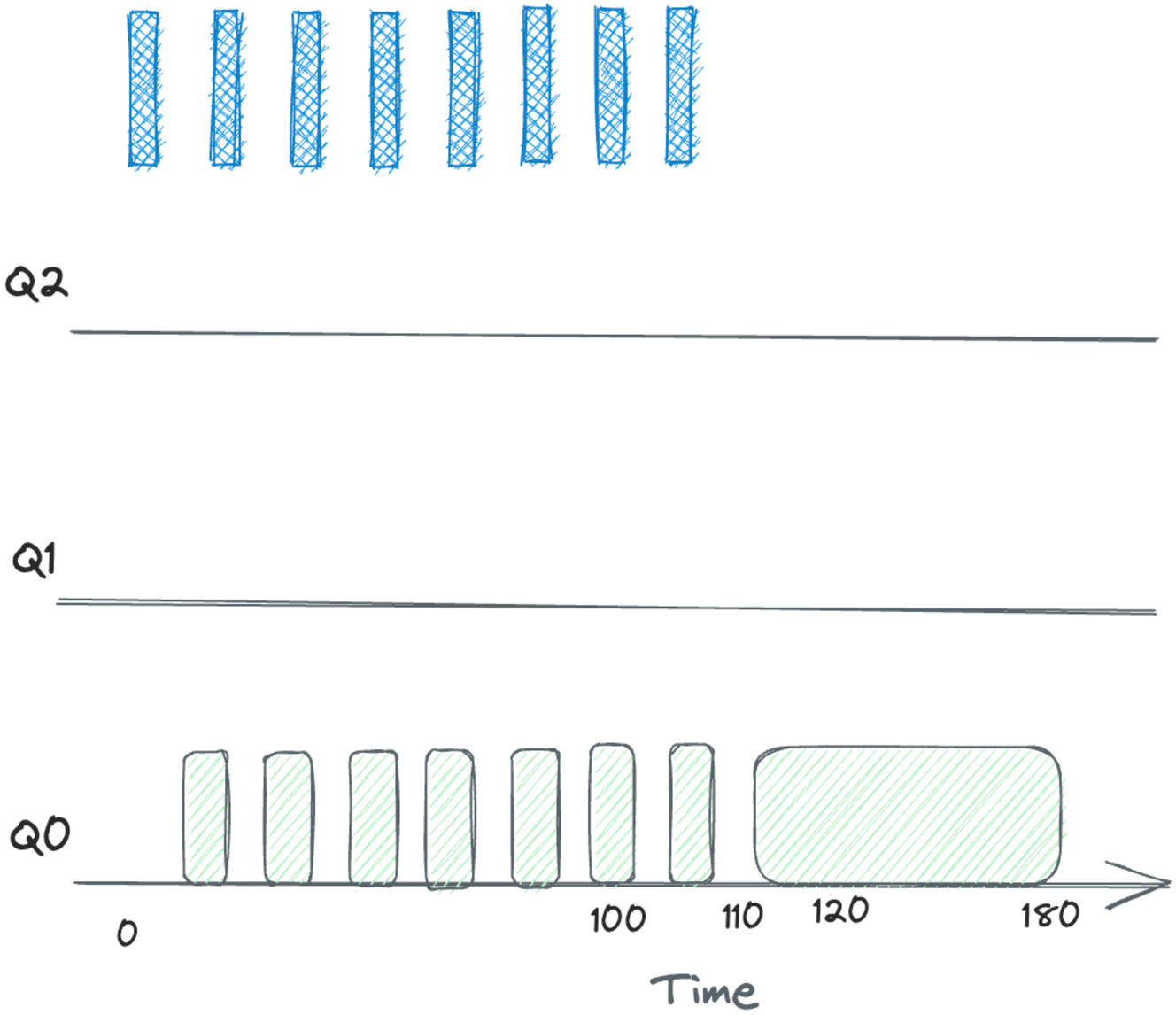
- If its short, it will keep running
  - Else moves down



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

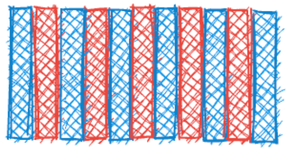
Can you think of some way to handle this?

Priority Boost

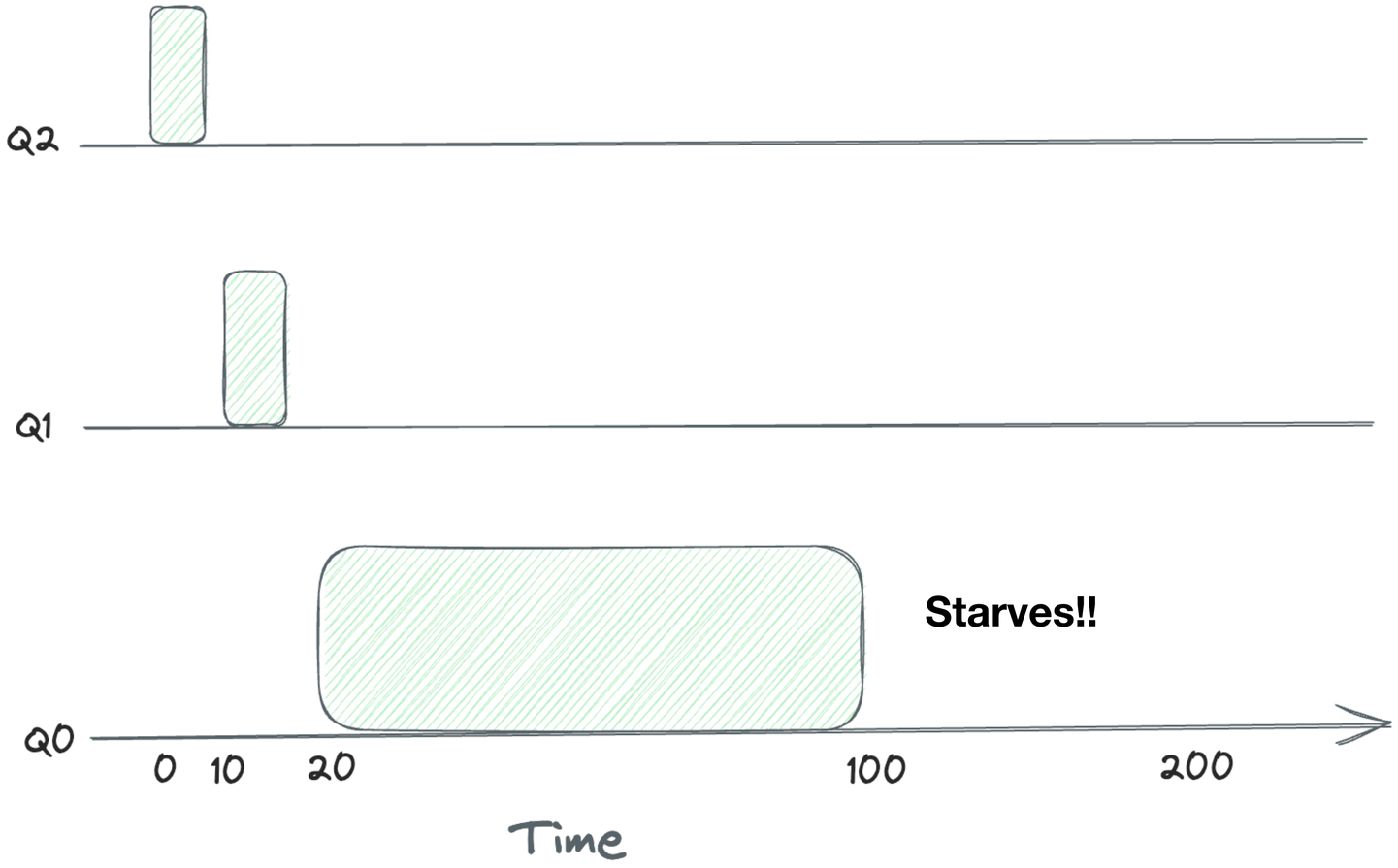
- Periodically boost the priority of all the jobs
  - This can **prevent starvation**
- **Rule:** After a time interval ***S***, move all jobs to top most queue
- Provides two key guarantees
  - Processes are guaranteed not to starve
  - If CPU bound has become interactive, scheduler will give it chances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

Case 1

Non priority Boost Scenario

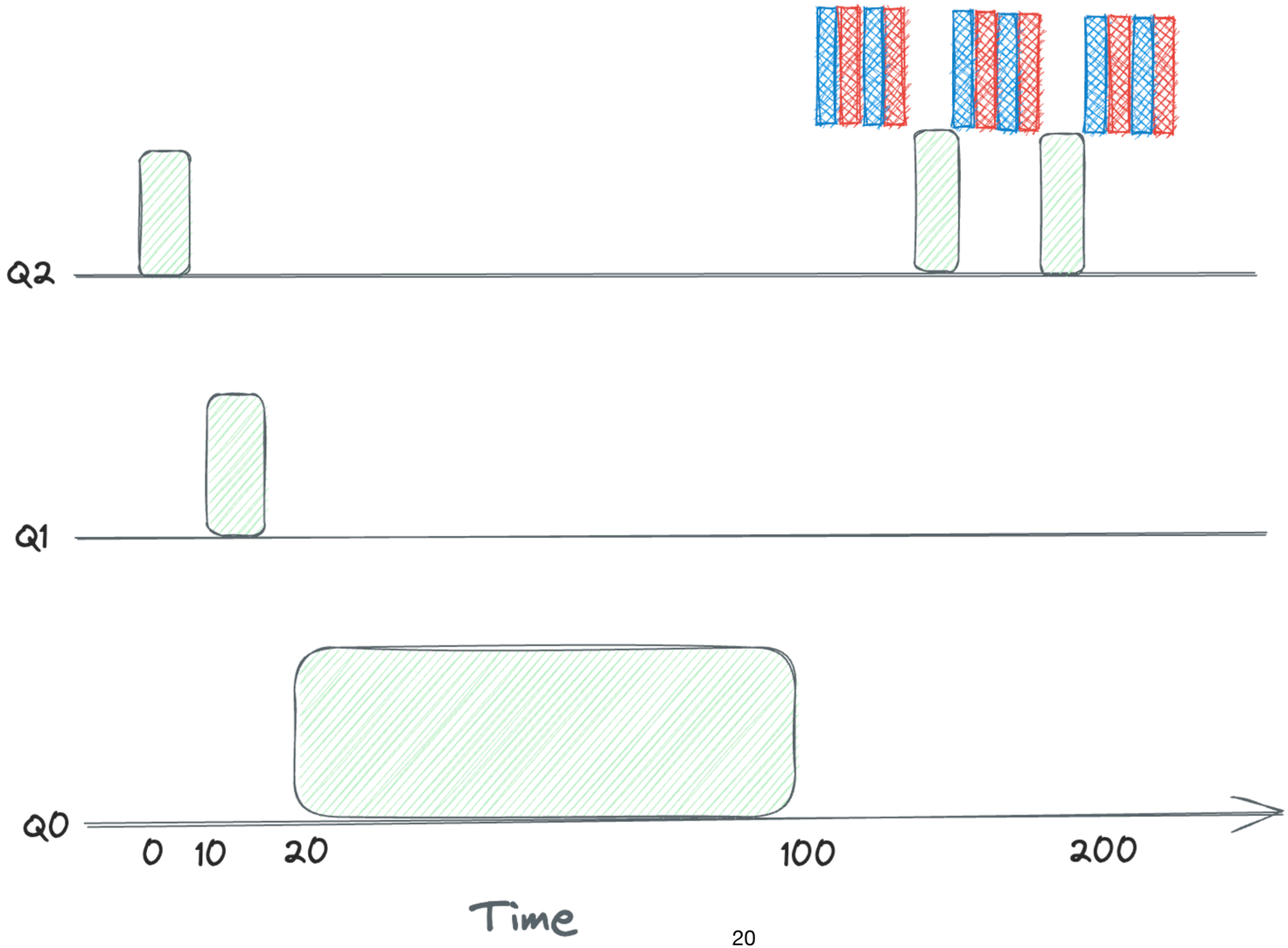


Interactive jobs keep getting the CPU



Case 2

With Priority Boost (S = 50)



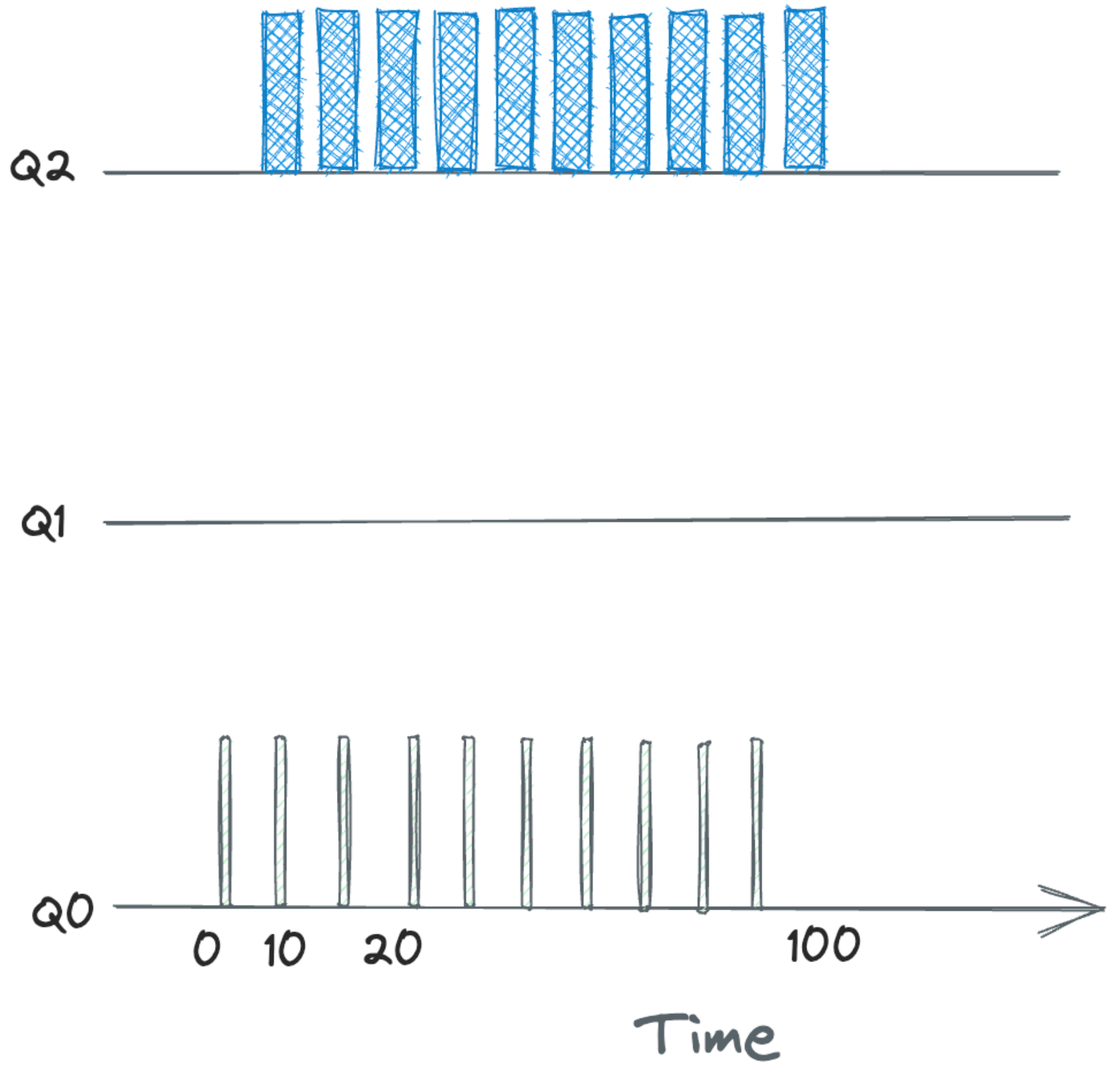
One additional Challenge

- 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

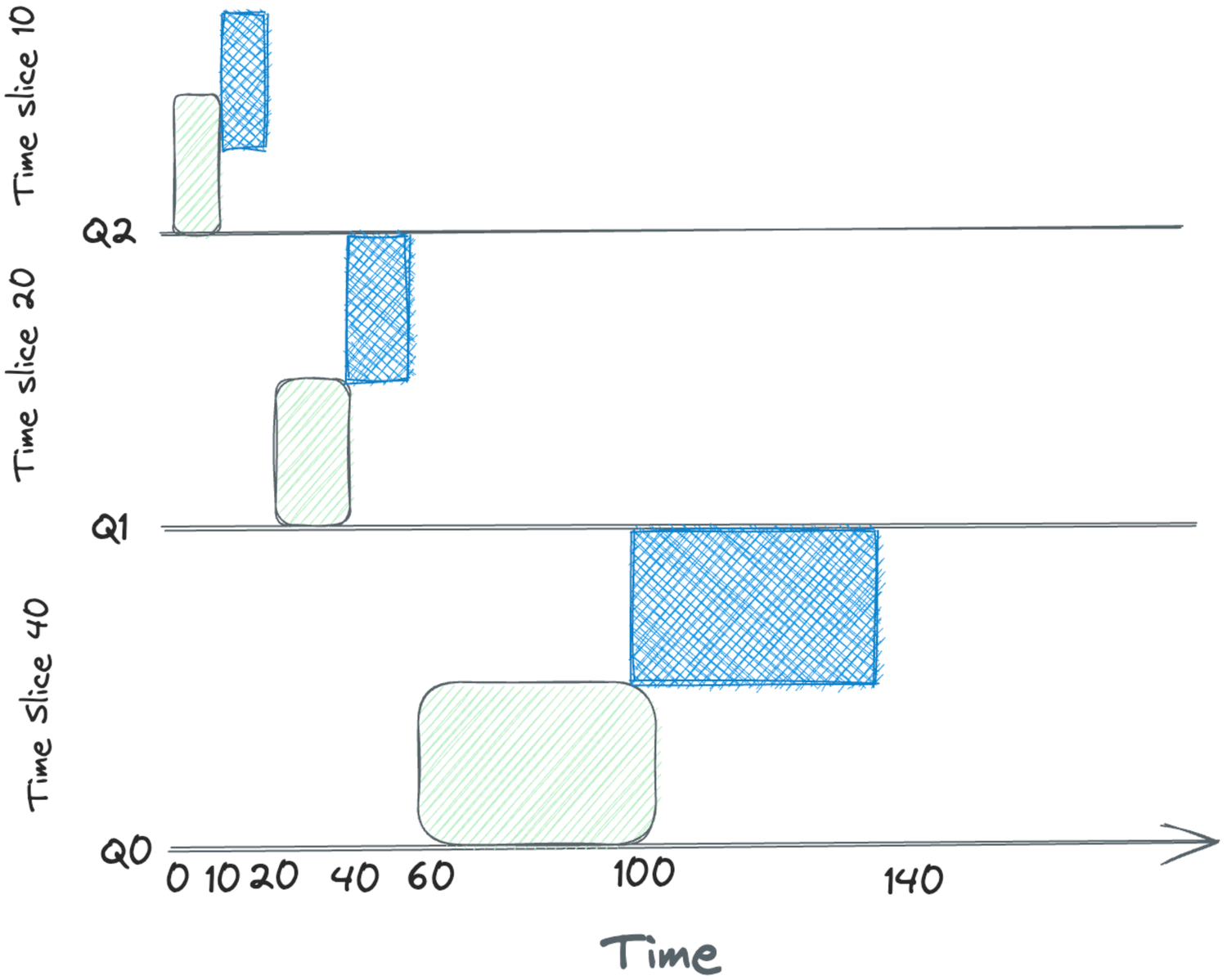
- How to parametrise the scheduler?
  - How many queues?
  - What should be the time slice?
  - How often the priority boost needs to be done?
- High priority queues: Interactive jobs with shorter time slices (10 ms)
- Low priority queues: CPU bound jobs with longer time slices (100 ms or less)

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

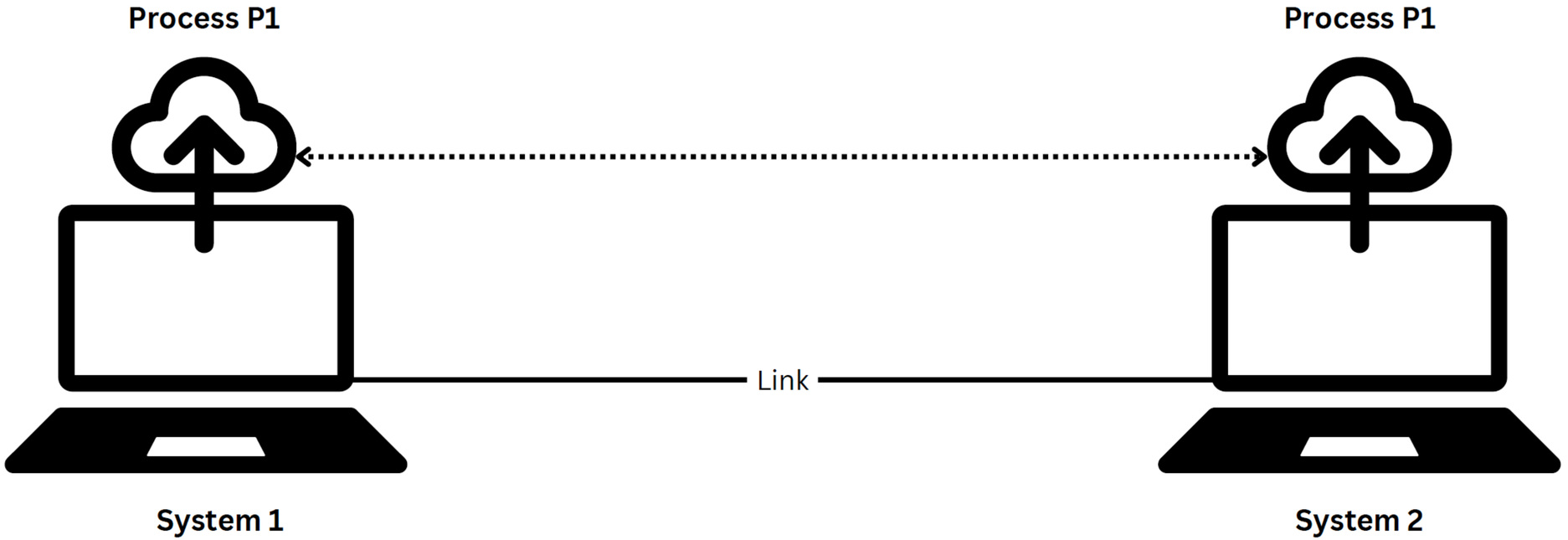
- Tables to configure:
  - How long should be the time slice?
  - How often to boost?
- 60 queues
  - Time slice length from 20ms to few 100 milliseconds
  - Priority boosted every 1 second
- Free BSD scheduler uses math formula to calculate priority

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?

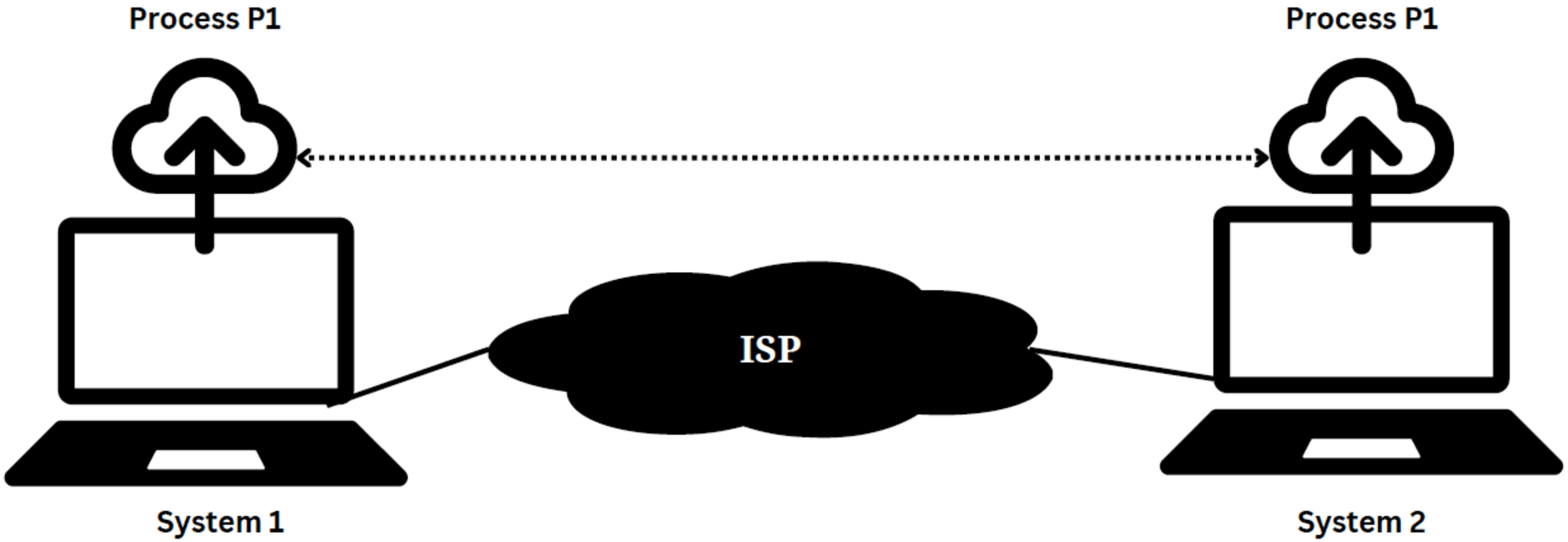
How can they communicate?



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?

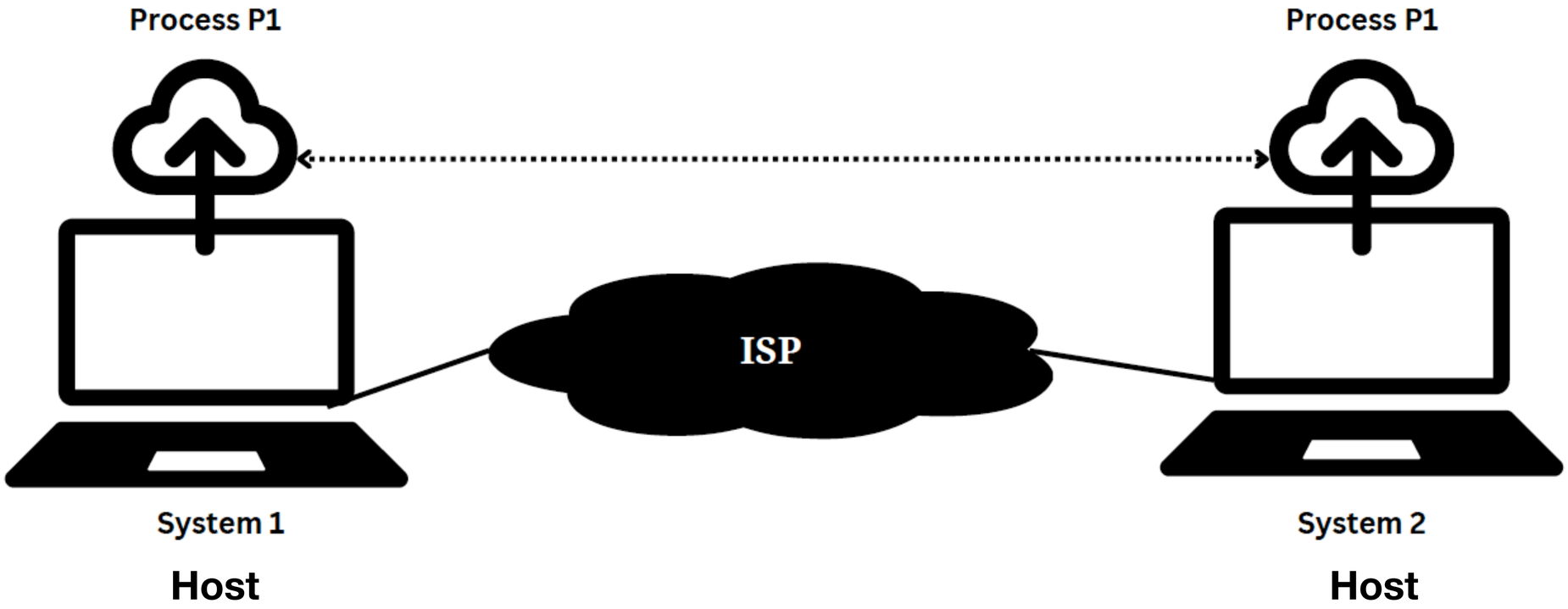
Let us expand it



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

Network Components

Host

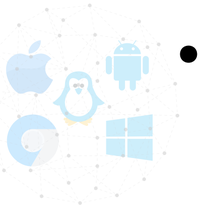


- Any device that send or receive traffic: Computer, laptop, smartwatch, phone, etc
- Host can be client or server
- Servers can sometime be clients too

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

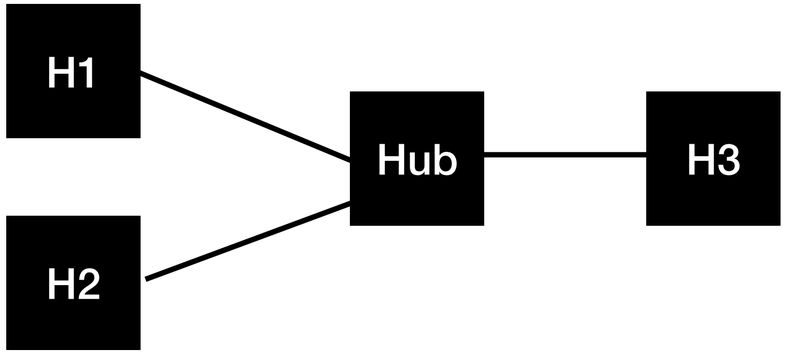
Repeater

- 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

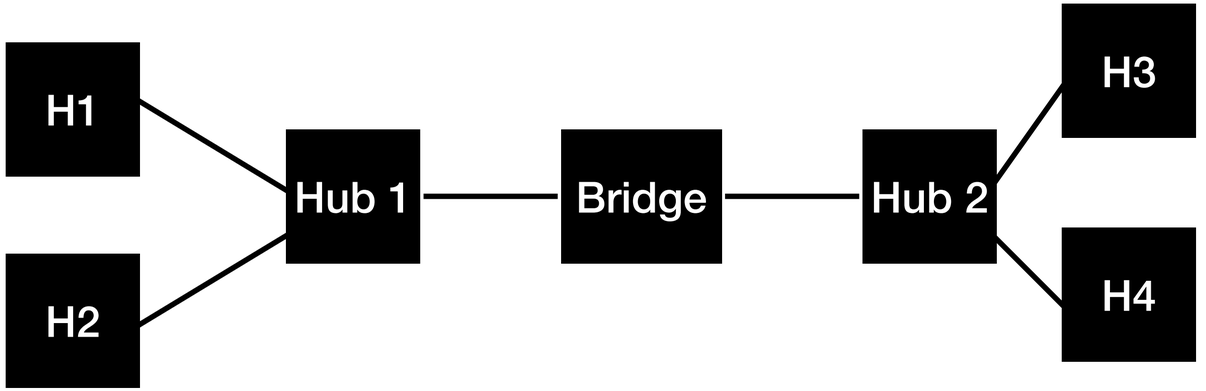
Hubs

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



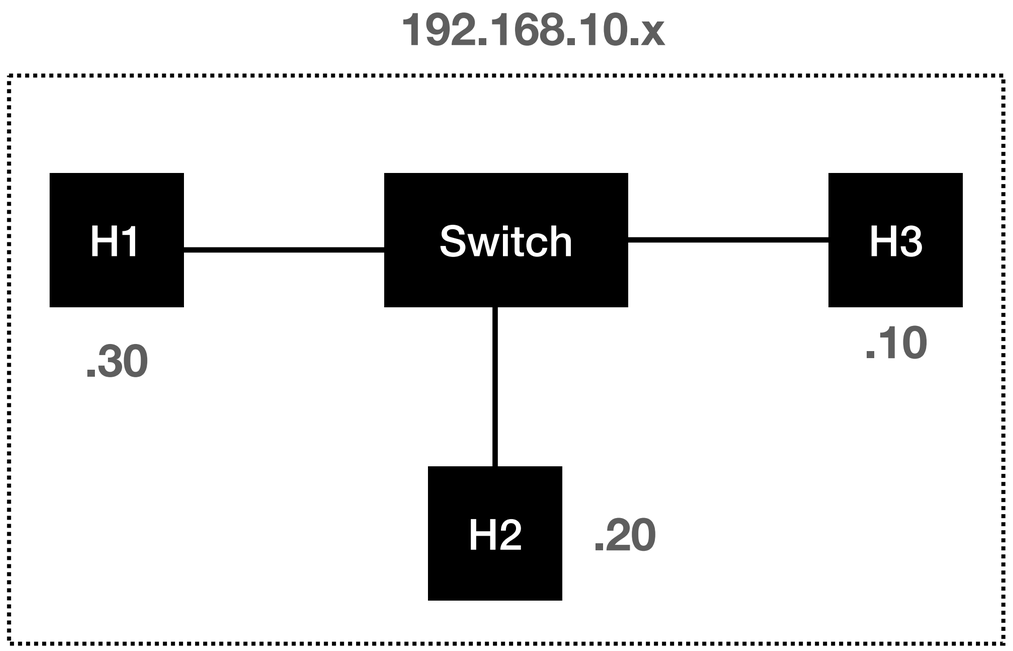
Bridges

- 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

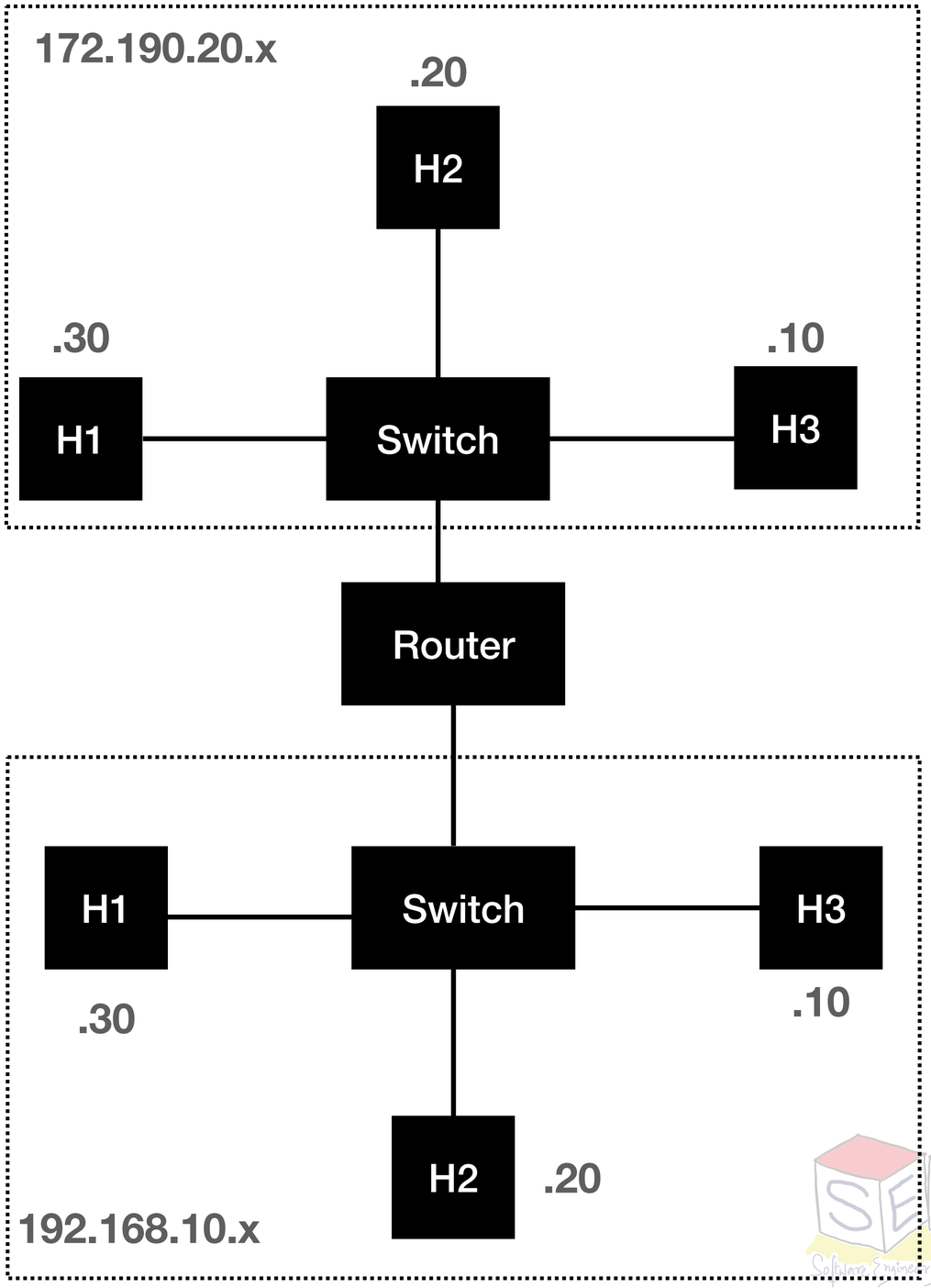
- Devices which facilitate communication within a network
- Combination of hubs and ports
- Knows or infers which hosts are on each port
- They have multiple ports
- Whatever connected to switch becomes part of one network



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

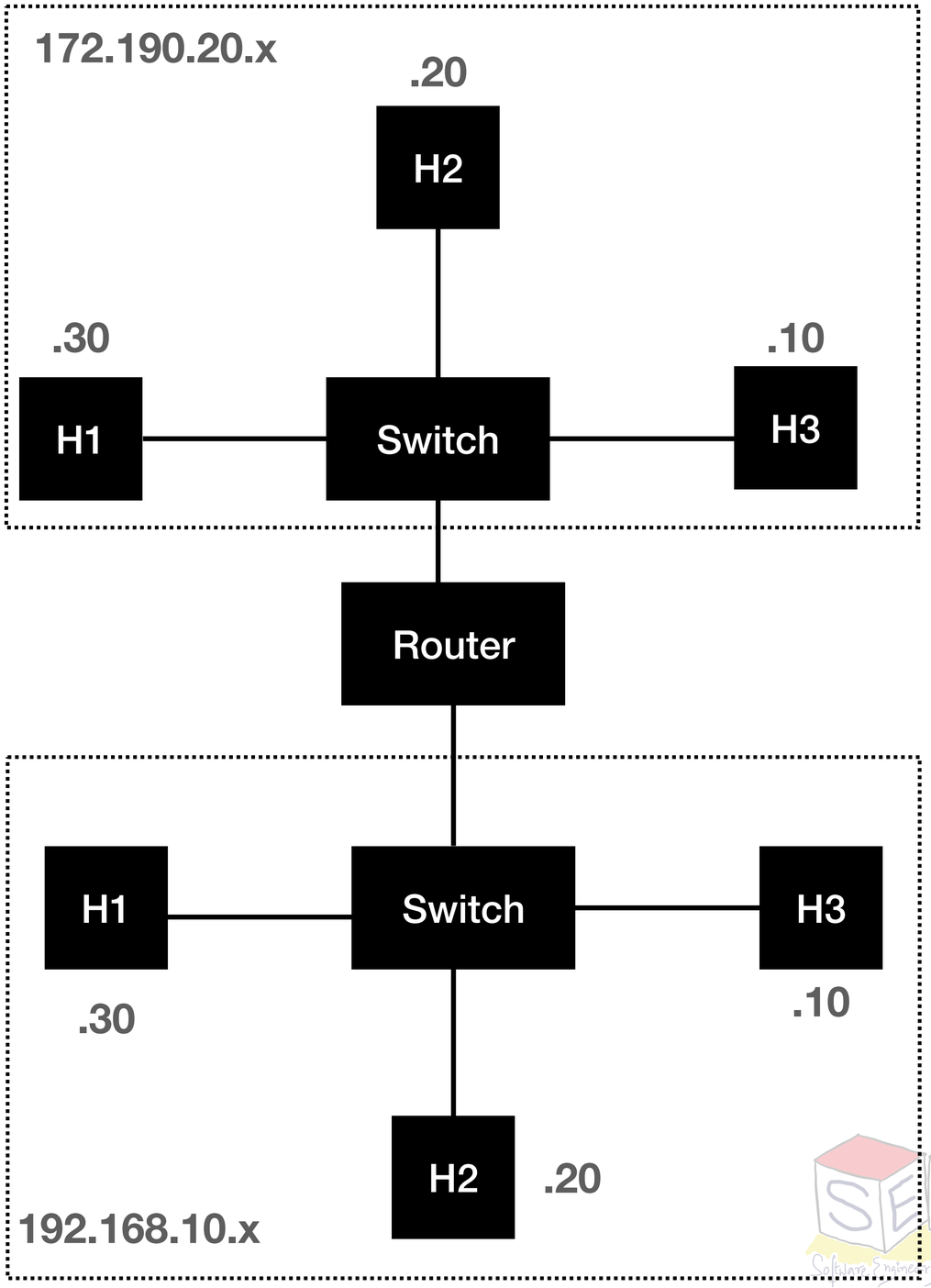
Routers

- 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

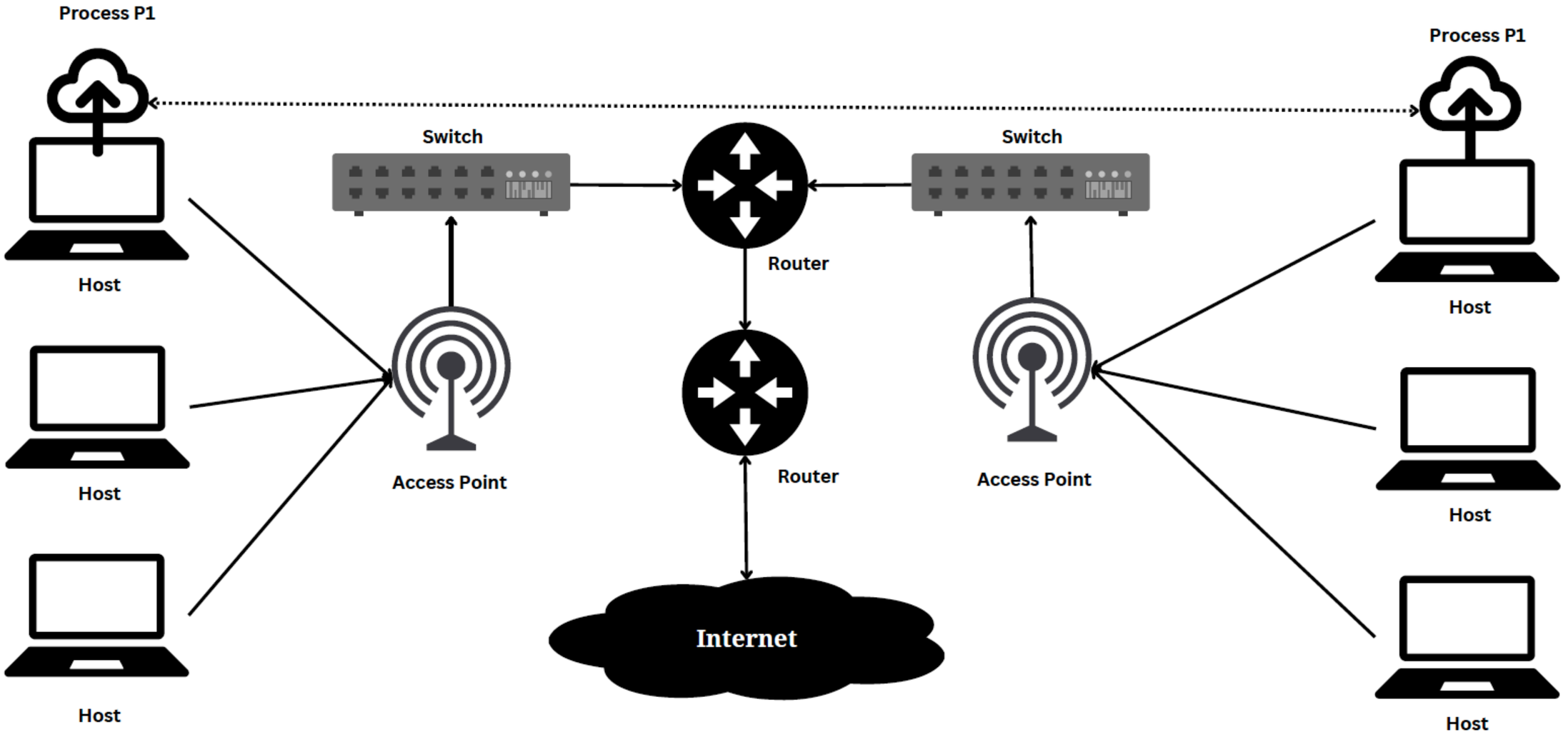


Routers

- 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



Thank you

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