Scheduling

Main Points

- Scheduling policy: what to do next, when there are multiple threads ready to run
  - Or multiple packets to send, or web requests to serve, or …
- Definitions
  - response time, throughput, predictability
- Uniprocessor policies
  - FIFO, round robin, optimal
  - multilevel feedback as approximation of optimal
- *Multiprocessor policies*
  - *Affinity scheduling, gang scheduling*
- *Queueing theory*
  - *Can you predict/improve a system's response time?*

Definitions

- Task/Job
  - User request: e.g., mouse click, web request, shell command, ...
- Latency/response time
  - How long does a task take to complete?
- Throughput
  - How many tasks can be done per unit of time?
- Overhead
  - How much extra work is done by the scheduler?
- Fairness
  - How equal is the performance received by different users?
- Predictability
  - How consistent is the performance over time?

More Definitions

- Workload
  - Set of tasks for system to perform
- Preemptive scheduler
  - If we can take resources away from a running task
- Work-conserving
  - Resource is used whenever there is a task to run
- Scheduling algorithm
  - takes a workload as input
  - decides which tasks to do first
  - Performance metric (throughput, latency) as output
  - Only preemptive, work-conserving schedulers to be considered

First In First Out (FIFO)

Fairness มี queue

แต่ไม่มี response time กับ throughput

- Schedule tasks in the order they arrive
  - Continue running them until they complete or give up the processor

workloads แต่ละอันใช้เวลาน้อย | มากไม่เท่ากัน

ใครมาก่อนได้ก่อน เช่น จ่ายเงิน เซเว่น

Lantacy แปรผันตามการ Load

- Example: memcached
  - Facebook cache of friend lists, …

- On what workloads is FIFO particularly bad?

Algorithmในอุตมคติ

Shortest Job First (SJF)

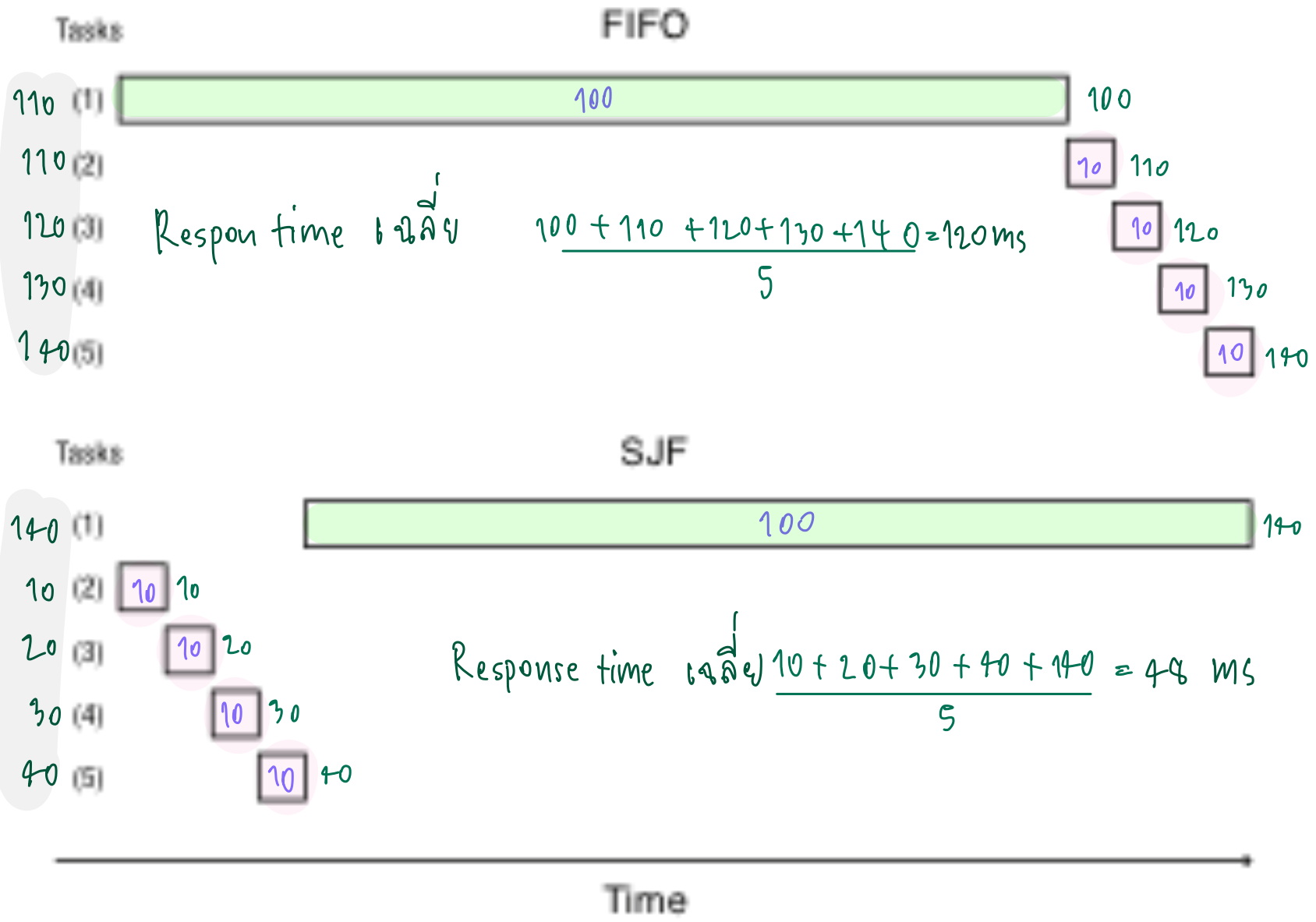
วัดยาก

เอางานที่ใช้เวลาน้อยที่สุด ถูกเลือกมาก่อน

- Always do the task that has the shortest remaining amount of work to do
  - Often called Shortest Remaining Time First (SRTF)

- Suppose we have five tasks arrive one right after each other, but the first one is much longer than the others
  - Which completes first in FIFO? Next?
  - Which completes first in SJF? Next?

FIFO vs. SJF


Tasks
FIFO
110 (1)
110 (2)
120 (3)
130 (4)
140 (5)
100
10
10
10
10
100
110
120
130
140
Respon time เฉลี่ย
100 + 110 + 120 + 130 + 140 = 120 ms
5
Tasks
SJF
140 (1)
10 (2)
20 (3)
30 (4)
40 (5)
100
10
10
10
10
140
10
20
30
40
Response time เฉลี่ย 10 + 20 + 30 + 40 + 140 = 48 ms
5
Time

Question

- Claim: SJF is optimal for average response time (ดีที่แทบทุกกรณี)
  - Why? (จริง เพราะ มัก จัดลำดับ ให้เวลาน้อยทำก่อน)

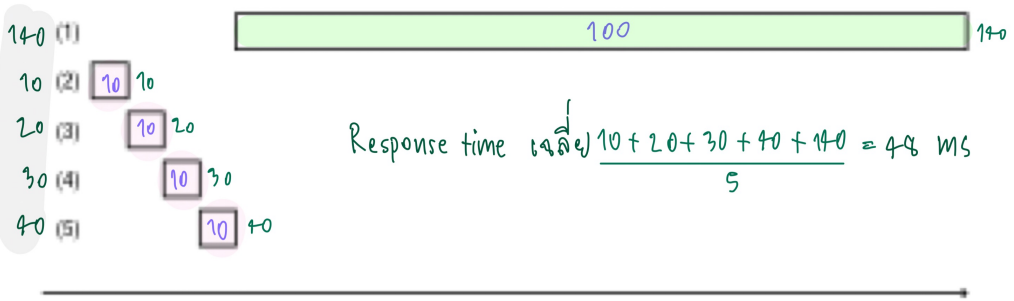
- Does SJF have any downsides? (ข้อเสีย/ข้อด้อย)

(มีงานขนาดเล็กที่แซงคิวงานขนาดใหญ่ จน ไม่ได้ทำงานสักที เกิด Starvation (ค.อดอยาก))

Question

- Is FIFO ever optimal? ดีที่สุด

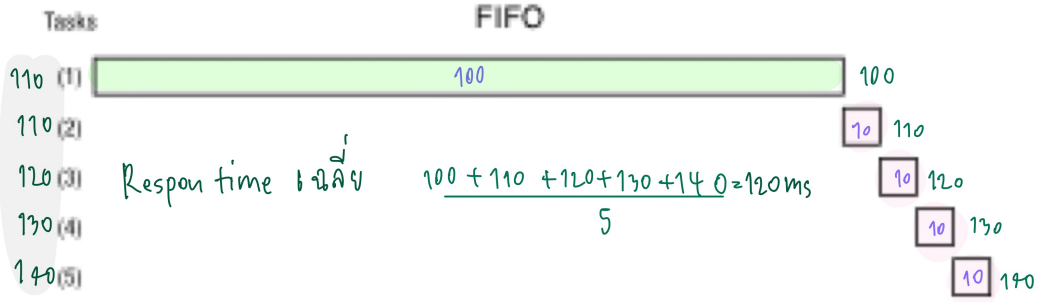
ในกรณีนี้มีค่าเท่ากับ SJF



แย่ที่สุด

- Pessimal?

แย่ที่สุด



ถูกออกแบบให้ใกล้เคียง SJF

Round Robin

คล้าย SJF

กำหนดช่วงเวลา ให้แต่ละ job ไปใช้งาน ถ้าหมดต้องคืนเวลาให้ job ต่อไปทำงาน

- Each task gets resource for a fixed period of time (time quantum)
  - If task doesn't complete, it goes back in line

เวลาที่กำหนด

ทำเสร็จไม่ทัน 1 time quantum ก็ต้องต่อ queue ใหม่

- Need to pick a time quantum
  - What if time quantum is too long?
    - Infinite?

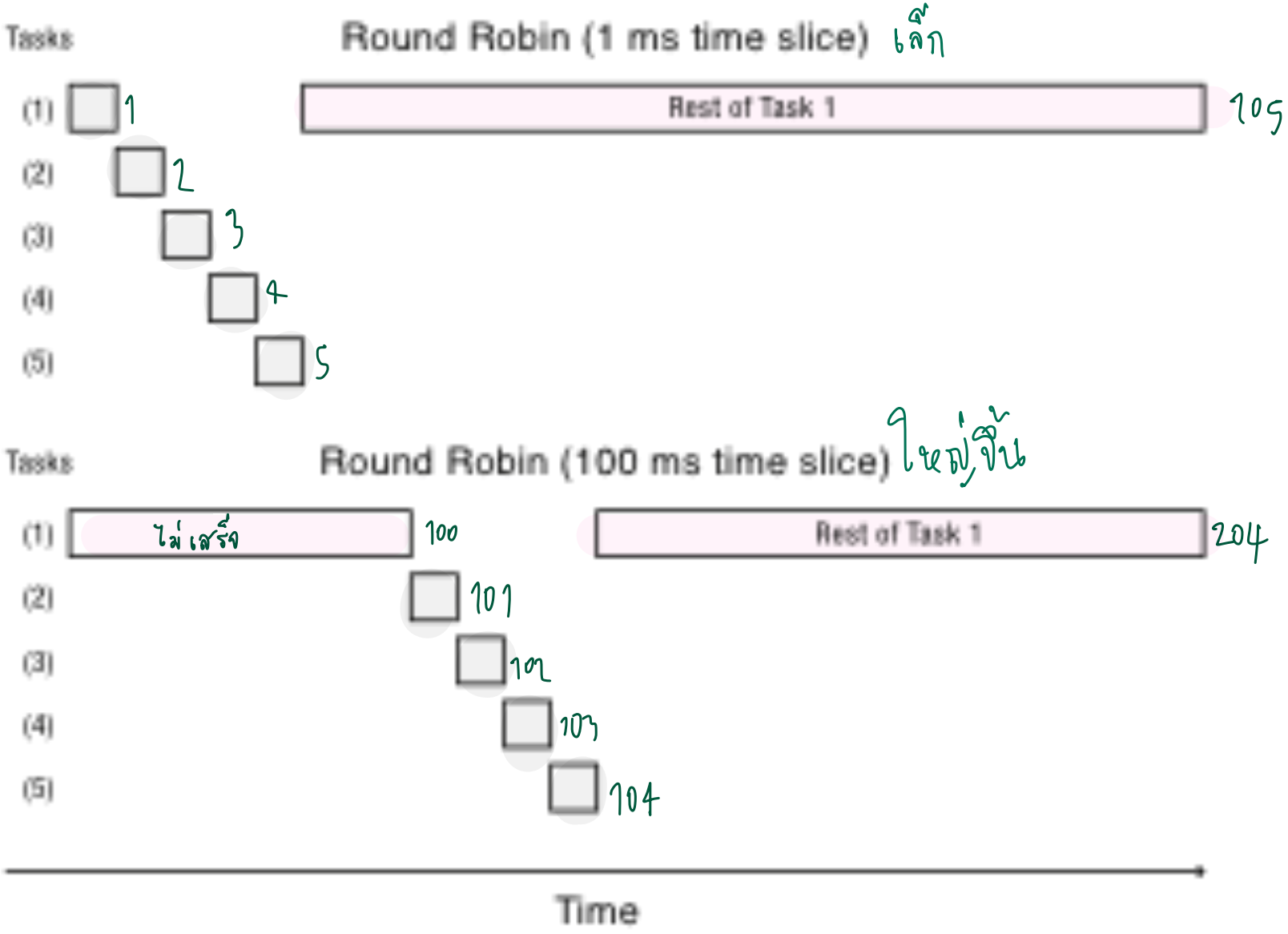
FIFO = ทำงานจนจบไม่มีใครมาไล่ออกได้ (จนกว่าจะเสร็จ job ต่อไปค่อยทำงาน)

  - What if time quantum is too short?
    - One instruction?

ขนาดเล็กมากๆ ที่เป็นไปได้

ทำงาน 1 อัน เกิด context switch หลาย instruction แล้วกลับมาทำงาน (พลีกมากใช้เวลานานเกิน)

Round Robin


Tasks
Round Robin (1 ms time slice) เล็ก
(1)
1
Rest of Task 1
105
(2)
2
(3)
3
(4)
4
(5)
5
Tasks
Round Robin (100 ms time slice) ใหญ่ขึ้น
(1)
ไม่เสร็จ
100
Rest of Task 1
204
(2)
101
(3)
102
(4)
103
(5)
104
Time

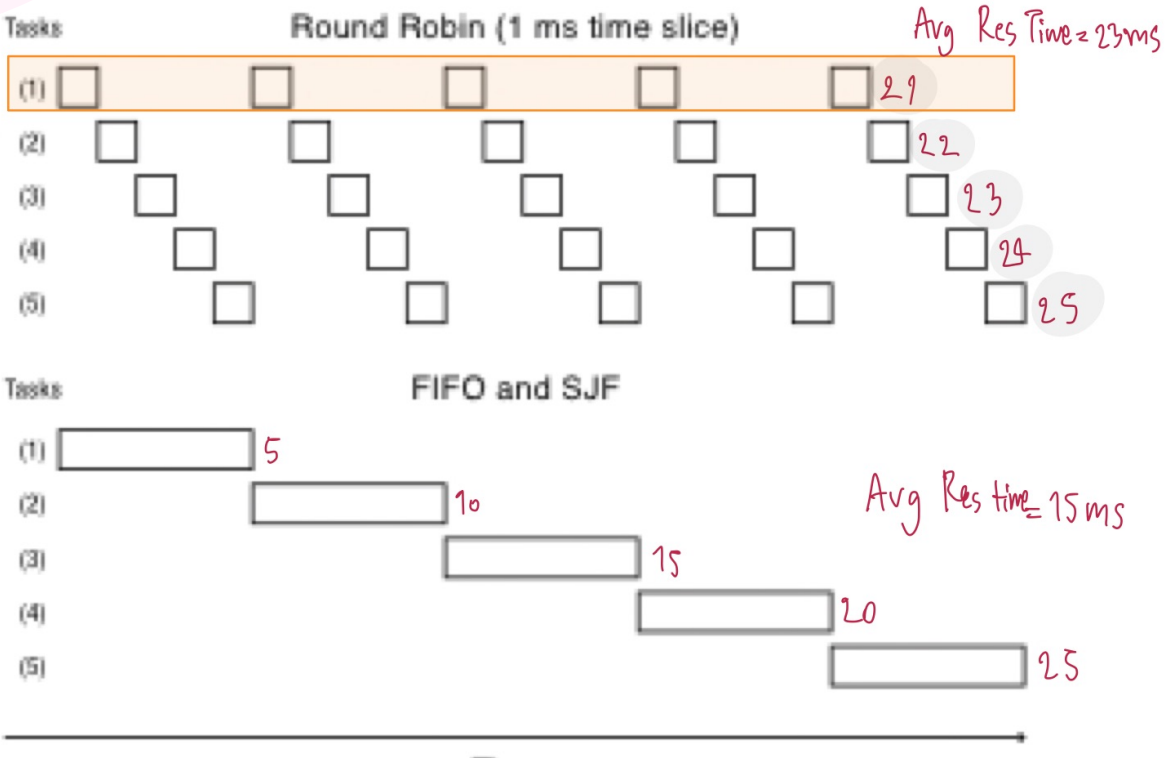
Round Robin vs. FIFO

switch job = 0

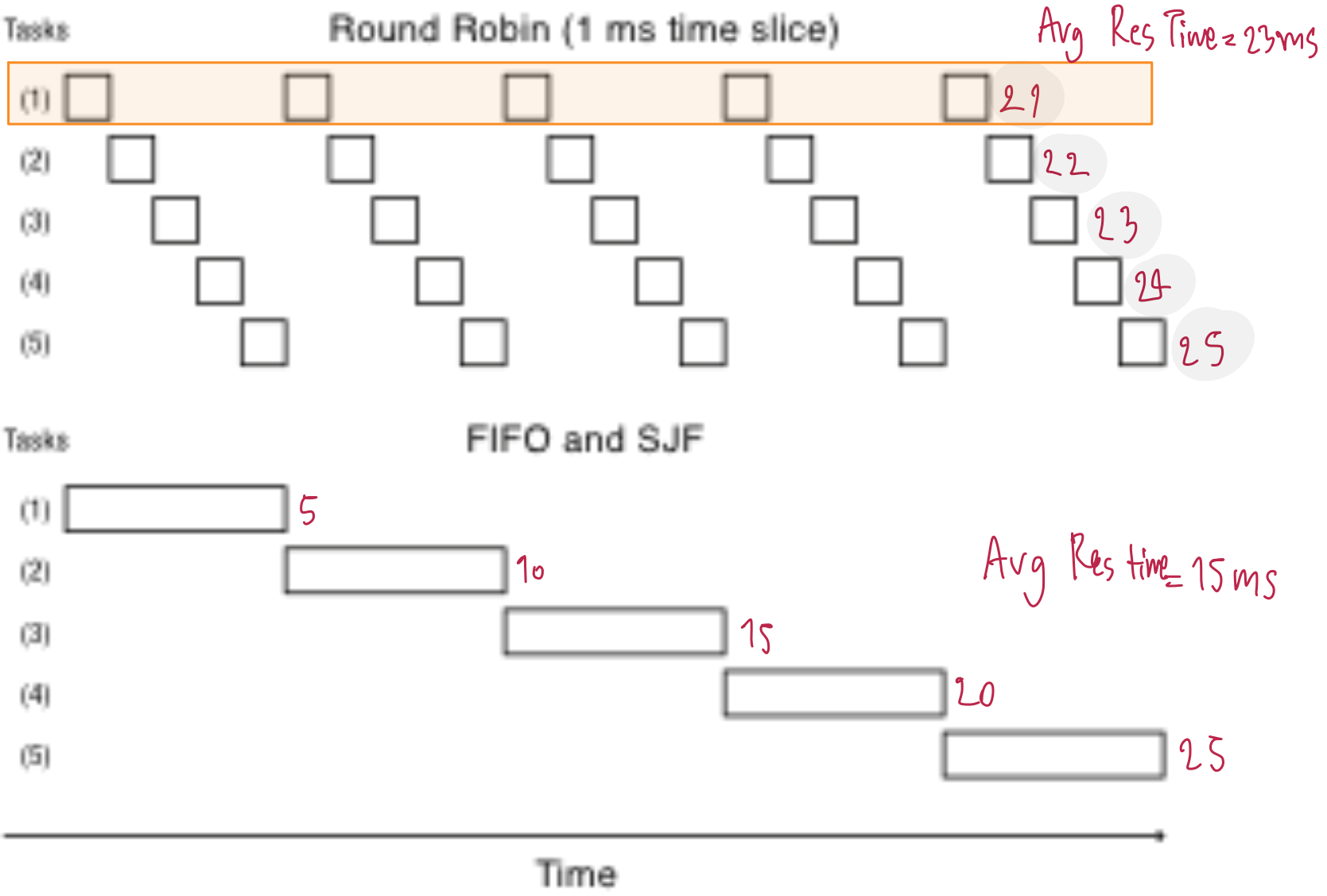
- Assuming zero-cost time slice, is Round Robin always better than FIFO?

Case 1

RR แย่กว่า FIFO

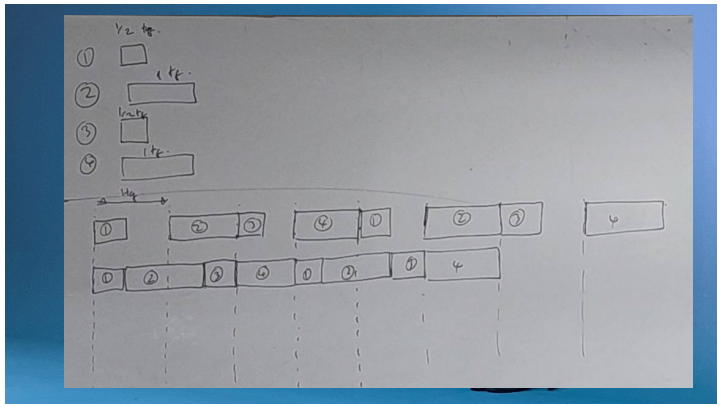


Round Robin vs. FIFO



แต่ SJF มีทรัพยากรเชิงคิว

Round Robin = Fairness?



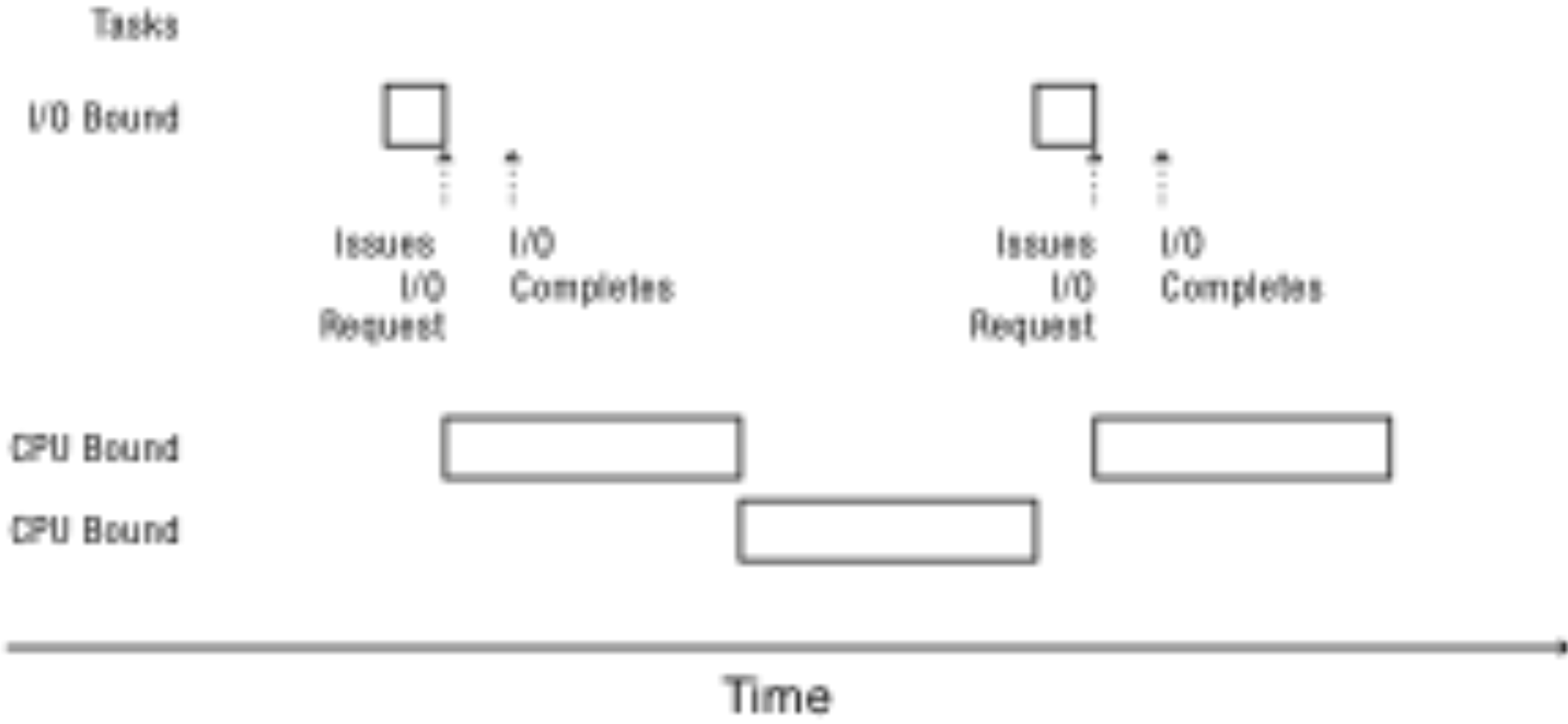
- Is Round Robin always fair?

ใช่

- What is fair?
  - FIFO? ลำดับทรัพยากรรับบริการ ใครมาถึงก่อนได้ใช้ก่อน
  - Equal share of the CPU? มีทรัพยากรใช้ CPU ไม่เหมือน FIFO SJF
  - What if some tasks don't need their full share? บางjobใช้เวลาไม่หมด เช่น read file เสร็จ ต้องทำให้เสร็จ ถึงไปต่อเวลาใหม่
  - Minimize worst case divergence? หลังจากนั้นทำอะไรต่อไม่ได้ ต้องรอเวลาให้หมด งานไม่เสร็จทำต่อไม่ได้ ต้องคืนเวลาให้ระบบ ไม่มีรถเช่า ต้องทำใหม่
    - Time task would take if no one else was running
    - Time task takes under scheduling algorithm

ขนาดของ Time quantum ที่เหมาะสม → ดี
ไม่เหมาะสม → ไม่ดี

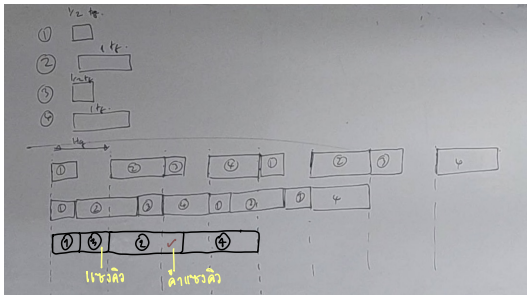
Mixed Workload



Algorithmในอุดมคติ

Max-Min Fairness

เวลาวัดยาก/ลำบาก



- How do we balance a mixture of repeating tasks:
  - Some I/O bound, need only a little CPU
  - Some compute bound, can use as much CPU as they are assigned
- One approach: maximize the minimum allocation given to a task
  - If any task needs less than an equal share, schedule the smallest of these first
  - Split the remaining time using max-min
  - If all remaining tasks need at least equal share, split evenly

จะรวบรวม ลักษณะ ของ RR + Max Min Fairness → ได้ผลลัพธ์ใกล้เคียงกัน

Multi-level Feedback Queue (MFQ)

- Goals:
  - Responsiveness การตอบสนองที่รวดเร็ว
  - Low overhead ต้อง สั้น, กระชับ
  - Starvation freedom ไม่เกิด Starvation
  - Some tasks are high/low priority มีการลำดับ ค. สำคัญ
  - Fairness (among equal priority tasks)
- Not perfect at any of them!
  - Used in Linux (and probably Windows, MacOS) คาดการณ์

MFQ

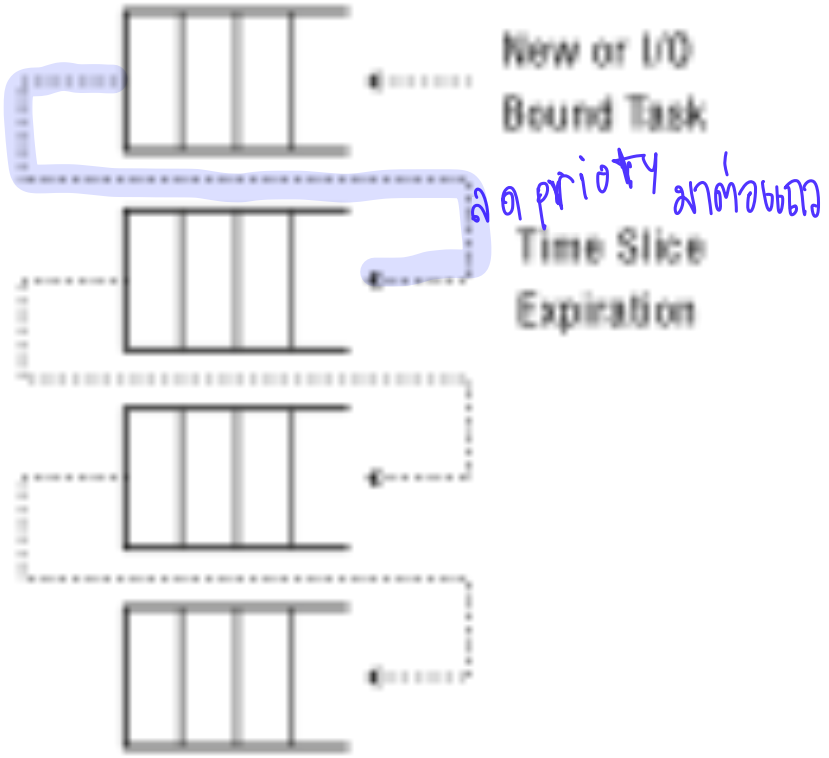
- Set of Round Robin queues
  - Each queue has a separate priority
- High priority queues have short time slices
  - Low priority queues have long time slices
- Scheduler picks first thread in highest priority queue
- Tasks start in highest priority queue
  - If time slice expires, task drops one level

MFQ

1 time Quantum = 10 ms

Priority	Time Slice (ms)
1 ทำงานก่อนเสมอ	10
2	20
3	40
4	80

Round Robin Queues



Uniprocessor Summary (1)

- FIFO is simple and minimizes overhead.
- If tasks are variable in size, then FIFO can have very poor average response time.
- If tasks are equal in size, FIFO is optimal in terms of average response time.
- Considering only the processor, SJF is optimal in terms of average response time.
- SJF is pessimal in terms of variance in response time. variance ไม่ดี

Uniprocessor Summary (2)

- If tasks are variable in size, Round Robin approximates SJF.
- If tasks are equal in size, Round Robin will have very poor average response time.
- Tasks that intermix processor and I/O benefit from SJF and can do poorly under Round Robin.

Uniprocessor Summary (3)

- Max-Min fairness can improve response time for I/O-bound tasks. ช่วยให้ชอบ
- Round Robin and Max-Min fairness both avoid starvation. หลีกเลี่ยง
- By manipulating the assignment of tasks to priority queues, an MFQ scheduler can achieve a balance between responsiveness, low overhead, and fairness.