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 งาน 1 งาน
  - User request: e.g., mouse click, web request, shell command, ...
- Latency/response time (ใช้เวลาน้อยๆ) ใช้เวลาแค่ไหนที่จะทำงานเสร็จ
  - How long does a task take to complete?
- Throughput (ได้มากๆ) อัตราการทำงานให้สำเร็จใน 1 หน่วยเวลา
  - How many tasks can be done per unit of time?
- Overhead งานที่ต้องทำเพิ่ม ในแง่ Scheduler
  - How much extra work is done by the scheduler?
- Fairness ความยุติธรรม มีหลายด้าน
  - How equal is the performance received by different users?
- Predictability ความสม่ำเสมอของประสิทธิภาพ
  - How consistent is the performance over time?

More Definitions

- Workload set ของงาน / กลุ่มงาน / งานหลายๆงาน
  - Set of tasks for system to perform
- Preemptive scheduler สามารถดึง Resource ที่กำลังทำงาน คืนจาก CPU
  - If we can take resources away from a running task
- Work-conserving = ถ้ามีงาน CPU จะทำงานอย่างรวดเร็วที่ไม่มีหยุดพัก
  - 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

รับก้อนของงานเข้ามา แล้วตัดสินใจว่าอะไรให้ทำก่อนหรือหลัง

อัตราการทำงานใน 1 หน่วยเวลา

เวลาที่ใช้ในการทำงานสำเร็จเสร็จจริงดี ต้องน้อยๆ

จัดลำดับงานให้เข้าไปทำงานใน CPU
ฉนกว่าจะหมดจึงหยุดพักได้

First In First Out (FIFO)

อัลกอริทึมแบบพื้นฐาน

= สนใจงานที่เข้าก่อนได้ทำงานก่อน

- Schedule tasks in the order they arrive (Schedule จะไม่ขัดจังหวะงาน)
  - Continue running them until they complete or give up the processor
- Example: memcached
  - Facebook cache of friend lists, ...
- On what workloads is FIFO particularly bad? (ให้ผลลัพธ์แย่คือ Throughput ต่ำ Latency สูง)

MOV Workload ที่ใช้เวลามากเข้ามาถึงก่อน แบบตัวอย่างหน้า 7 FIFO (Longtest job first)

Shortest Job First (SJF)

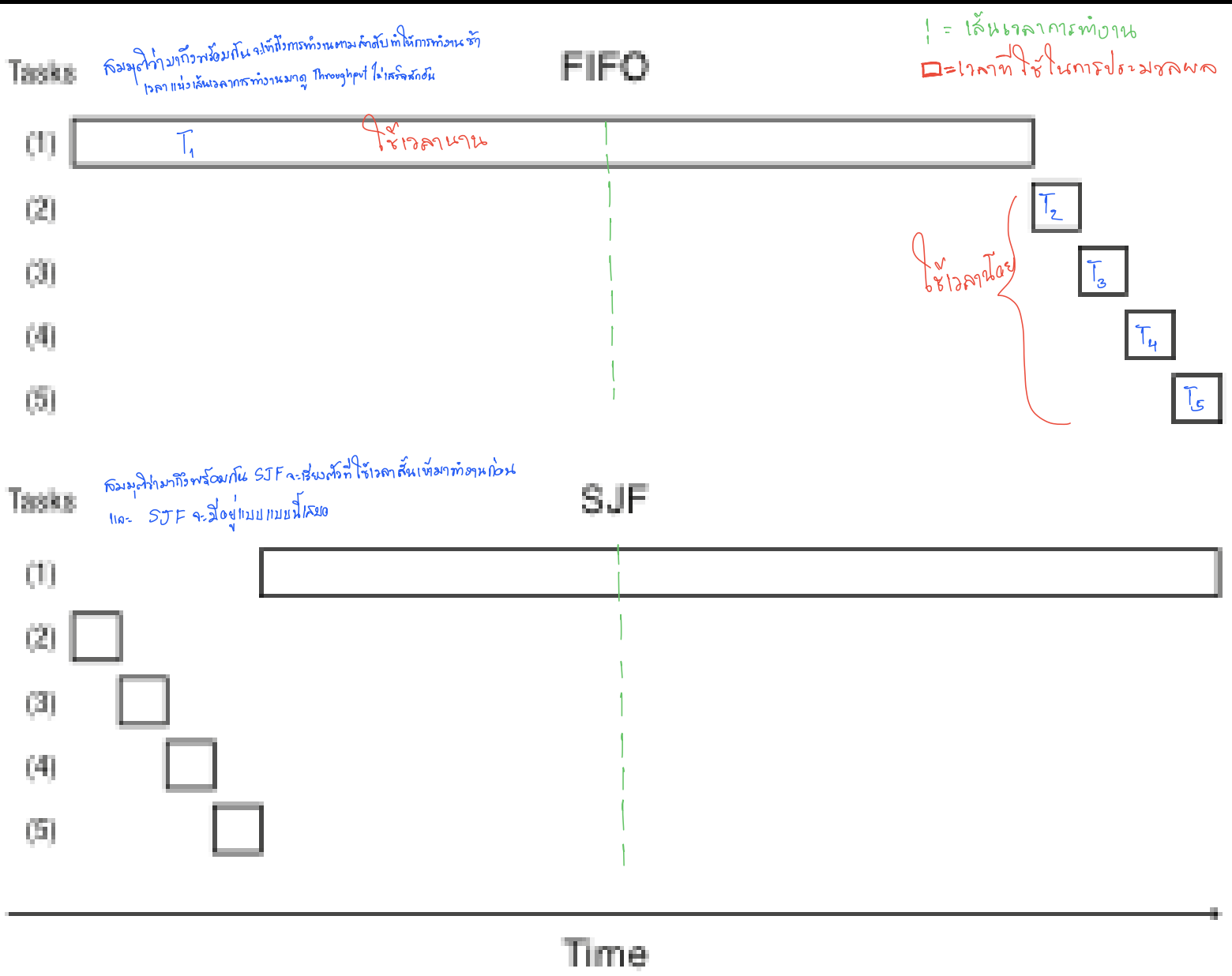
เป็น job ที่ใช้เวลาประมวลผลสั้นสุดจะได้เข้าทำงานก่อน ไม่สนใจการเข้าของงานก่อนหลัง

cpu time

- 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



Question

- Claim: SJF is optimal for average response time
  - Why? ตัวของ SJF จะเลือกตัวที่ทำงานสั้นที่สุด ซึ่งทำ Throughput สูง response time ต่ำ

- Does SJF have any downsides? ด้านลบ / ข้อเสีย

สรุป! การต่อแถวจ่ายเงิน ใครมีของน้อยชิ้นได้จ่ายเงินก่อน
ส่วนใครซื้อของเยอะไล่ไปต่อท้ายแถวซักที มีคนซื้อของ 1 ชิ้นมาเรื่อยๆ คนที่ซื้อ 10 ชิ้นก็รอไปเถอะไปได้จ่ายเงิน
เกิดเป็น starvation : job ขนาดใหญ่จะใดแทรงคิว

กรณีที่ 2 ที่เป็นข้อเสียคือไม่สามารถระบุเวลาในการทำงานได้

Question

- Is FIFO ever optimal?

- Pessimal?

SJF ดีสุด ถือเป็นจุดเด่นของมัน

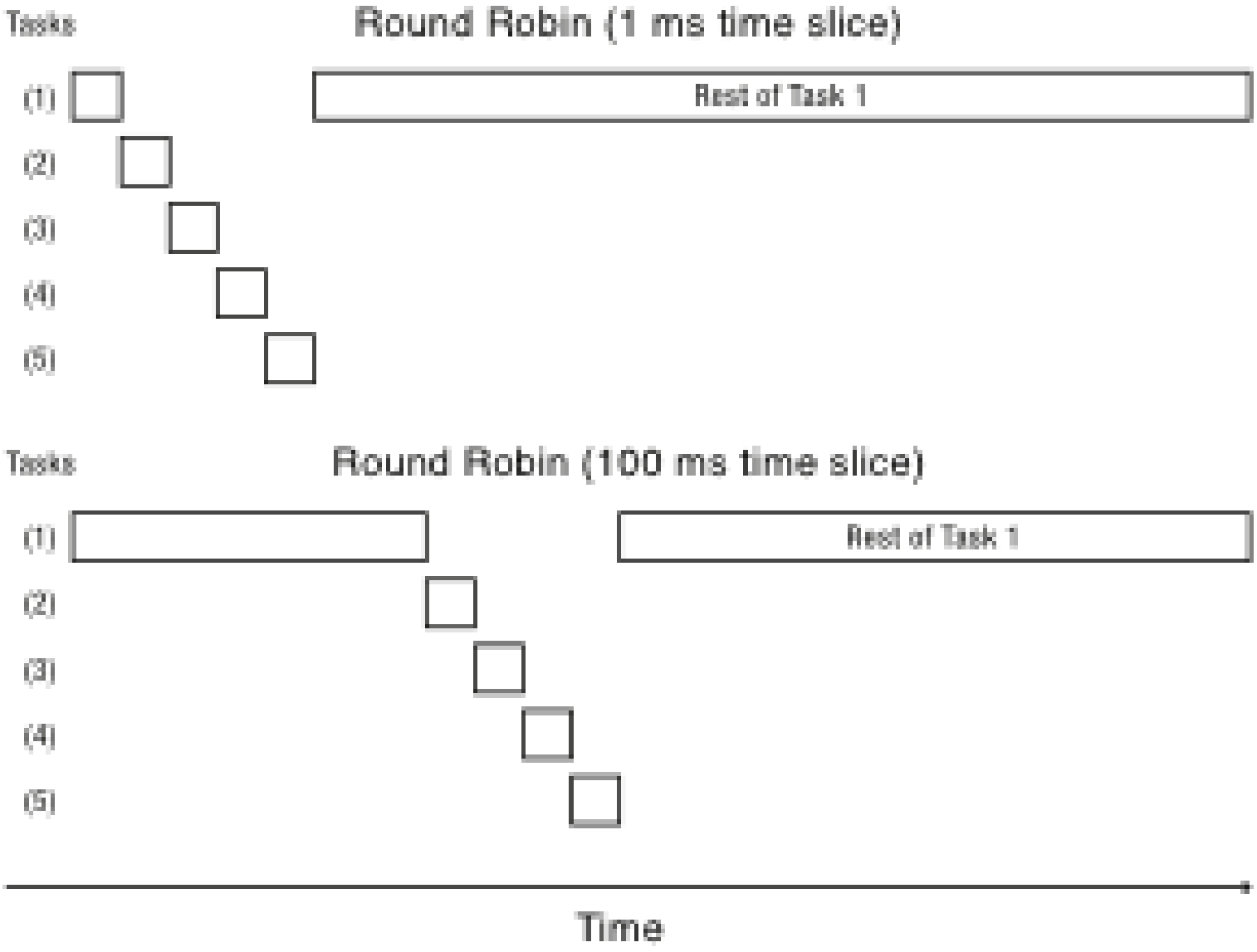
Round Robin

- Each task gets resource for a fixed period of time (time quantum) ถือเป็นจุดเด่น แบ่งเวลาในการเข้าใช้ cpu เป็น slote เท่าๆกัน ถ้าใช้จบใน 1 time quantum ก็หมดเวลาแล้วทำงานจบก็เสร็จ แต่ถ้า 1 time quantum ยังไม่เสร็จก็ไปต่อท้ายแถว
  - If task doesn't complete, it goes back in line
- Need to pick a time quantum
  - What if time quantum is too long?
    - Infinite? จะเหมือน FIFO มันจะใช้งานตามลำดับ
  - What if time quantum is too short?
    - One instruction? ประมวลผลสั่นไปจะเกิด Overhead เยอะ คือมี job เข้า แต่ให้ทำเสร็จแค่ 1 แล้วไล่ไปต่อท้าย วนอยู่แบบนี้

Round Robin

Ef ของ Time quantum ที่เหมาะสมใน RR

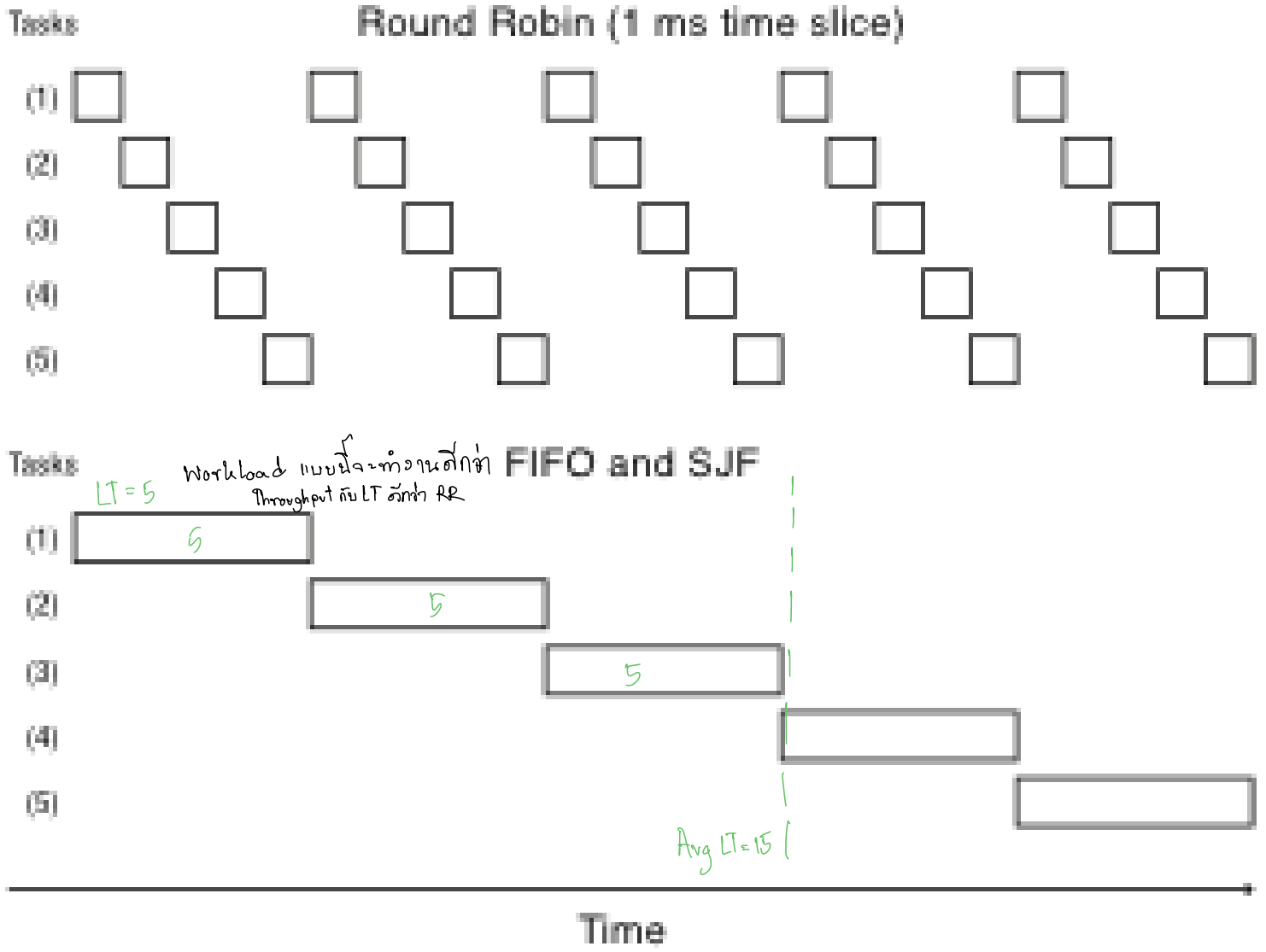
Time Quantum



Round Robin vs. FIFO

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

Round Robin vs. FIFO

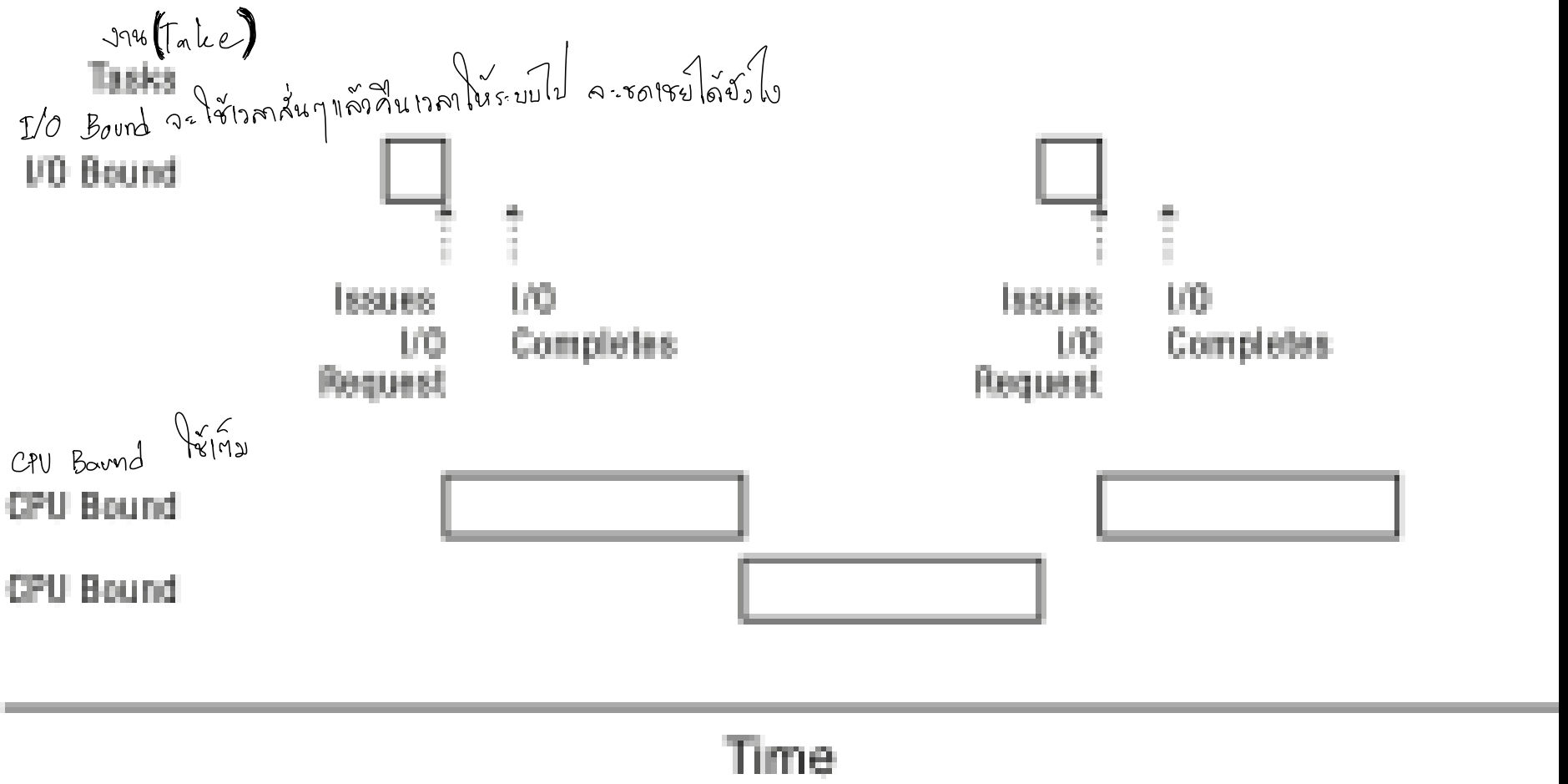


Round Robin = Fairness?

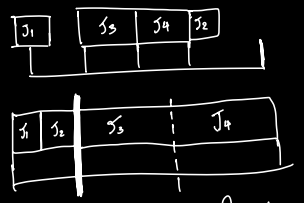
- Is Round Robin always fair? ไม่fair เพราะถ้า job
- What is fair?
  - FIFO? ในกรณีมาก่อนหลัง
  - Equal share of the CPU?
  - What if some tasks don't need their full share? จบที่ในเวลาไม่เต็ม TQ ต้องคืนเวลาก่อน ถ้าจะให้fair ต้องมีการชดเชยเวลาให้กับงานประเภทนี้
  - Minimize worst case divergence?
    - Time task would take if no one else was running ลืมมุมมี ถ้าเหลือแค่งานเดียว แต่ยังใช้กฎ 1 QT คือเมื่องานมี 10 จะต้องให้สิทธิ์ทุก 1 TQ ด้วย
    - Time task takes under scheduling algorithm

RR จะตอบคำถามได้แค่ 2 ข้อนี้

Mixed Workload



Max-Min Fairness



คือ Max/Min fairness จะให้ I/O Bound (ใช้เวลาสั้นมาทำงานก่อน) แล้วเวลาที่ J06 ของ I/O Bound คืนมา จะทำเวลานั้น มาใส่ให้ CPU Bound ทำให้ J3, J4 มีเวลาในการทำงานมากขึ้น คือเอาเวลาที่ใช้คืนเพิ่ม

- 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

ให้ความสำคัญกับงานที่ใช้เวลาสั้นสุดก่อน (จบก่อน) 1 Time Quantum

Multi-level Feedback Queue (MFQ)

- Goals:
  - Responsiveness โปรแกรมที่มีลักษณะ : I/O Bound ตอบสนองได้ดีขึ้น
  - Low overhead
  - Starvation freedom job ยาวไม่ได้ใช้ CPU
  - Some tasks are high/low priority สามารถ add just cpu ได้
  - Fairness (among equal priority tasks)
- Not perfect at any of them!
  - Used in Linux (and probably Windows, MacOS)

เป็น open source

MFQ

- Set of Round Robin queues จะมีหลายคิว
  - Each queue has a separate priority แต่ละคิวมี priority ไม่เท่ากัน
- 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

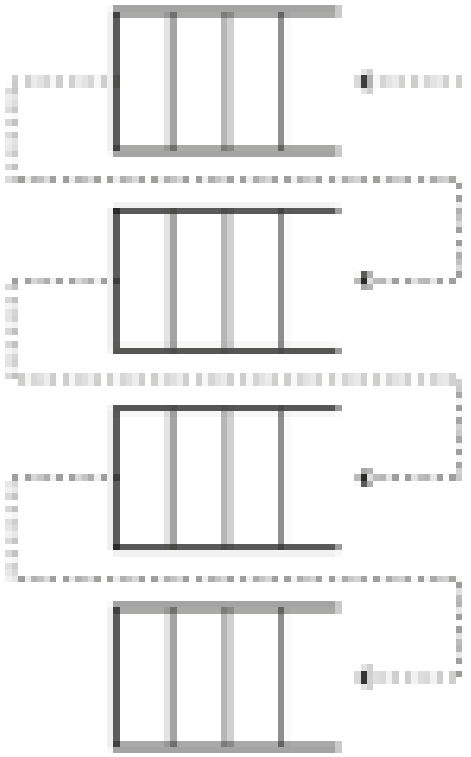
MFQ

Time Quantum

Priority	Time Slice (ms)
1 สูงสุด	10 เล็ก
2	20
3	40
4 ต่ำ	80 ใหญ่

Round Robin Queues

New or I/O Bound Task



Note! ถ้าใช้ RR (Time Slice) ไม่เหมาะกับระบบที่ต้องการคืนเวลาให้ระบบ ขึ้นอยู่กับคน Implement ว่าต้องการแบบไหนต้องทำให้ได้ทั้งสองอย่างเหมือนกัน

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.

Uniprocessor Summary (2)

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

SJF ให้ Job สั้น แทรกคิว RR จะเสียประโยชน์ตรงที่ถ้าใช้เต็มเวลา TQ จะโดนตัดแล้วไปต่อคิวท้ายใหม่ เลยเกิดเป็น max/min fairness

Uniprocessor Summary (3)

SJF Job มีขนาดใหญ่จะเกิด starvation

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