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 เซ็ทของงาน ประกอบด้วย งาน ขึ้นไป
  - Set of tasks for system to perform
- Preemptive scheduler มีปัจจัยภายนอกเข้ามา
  - If we can take resources away from a running task
- Work-conserving ตรงกันข้ามกับ non ถ้า มีงาน processer ประมวลผลทุกงาน หมด
  - Resource is used whenever there is a task to run
- Scheduling algorithm
  - takes a workload as input
  - decides which tasks to do first ② เลือกว่าจะทำงานต่อไป เหมาะตัดใจ ① เลือกว่าใครใช้ workload ได้เข้าไป ทำงานกับ processer ที่มีประสิทธิภาพ
  - Performance metric (throughput, latency) as output
  - Only preemptive, work-conserving schedulers to be considered ข้อแม้

First In First Out (FIFO)

- Schedule tasks in the order they arrive
  - Continue running them until they complete or give up the processor
- Example: memcached
  - Facebook cache of friend lists, …

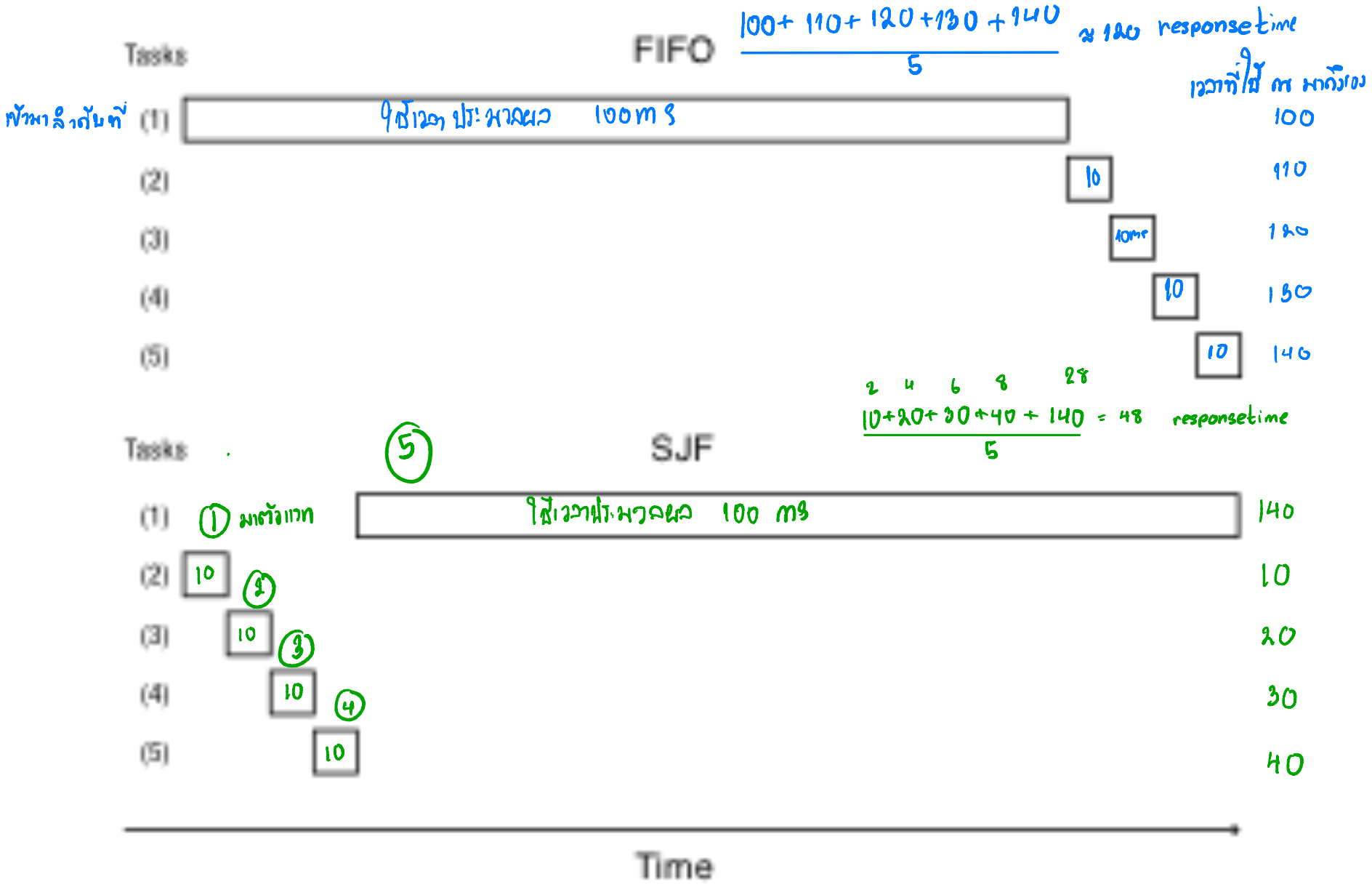
- On what workloads is FIFO particularly bad?

Shortest Job First (SJF)

- 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
100+110+120+130+140 / 5 ≈ 120 response time
เวลาที่ใช้ ๓ มาถึงของ
ฟังหาลำดับที่
(1) ใช้เวลาประมวลผล 100 ms
100
(2) 10
110
(3) 10ms
120
(4) 10
130
(5) 10
140
2 4 6 8 28
10+20+30+40+140 / 5 = 48 response time
Tasks
SJF
5
(1) ① มาถึงแรก ใช้เวลาประมวลผล 100 ms
140
(2) 10 ②
10
(3) 10 ③
20
(4) 10 ④
30
(5) 10
40
Time

Question

- Claim: SJF is optimal for average response time
  - Why? ทำไม response time เฉลี่ยถึงน้อยสุด

    เวลา เริ่มน้อยสุด ค่าเฉลี่ยจะต่ำ

- Does SJF have any downsides? ข้อเสีย มีข้อเสียไหม

- มีการถูกแซงคิวไปเรื่อย (เช่น: มี job เข้ามาใน work load) เรียกว่า ลักษณะ "starvation" ตายยาก

res time = response time

Question

มีสามารถให้ค่าเฉลี่ยที่ดีได้หรือไม่

- Is FIFO ever optimal? สำได้ กรณีใดบ้าง

  มี สำ Job เข้ามาแบบเรียงลำดับ โดยเวลาน้อย-มาก

  FIFO res time = SJF res time

มีสามารถให้ค่าเฉลี่ยที่แย่ที่สุดหรือไม่

- Pessimal? สำได้ กรณีใดบ้าง

  มี กรณี Job มาหนักที่สุด - น้อยสุด

Round Robin

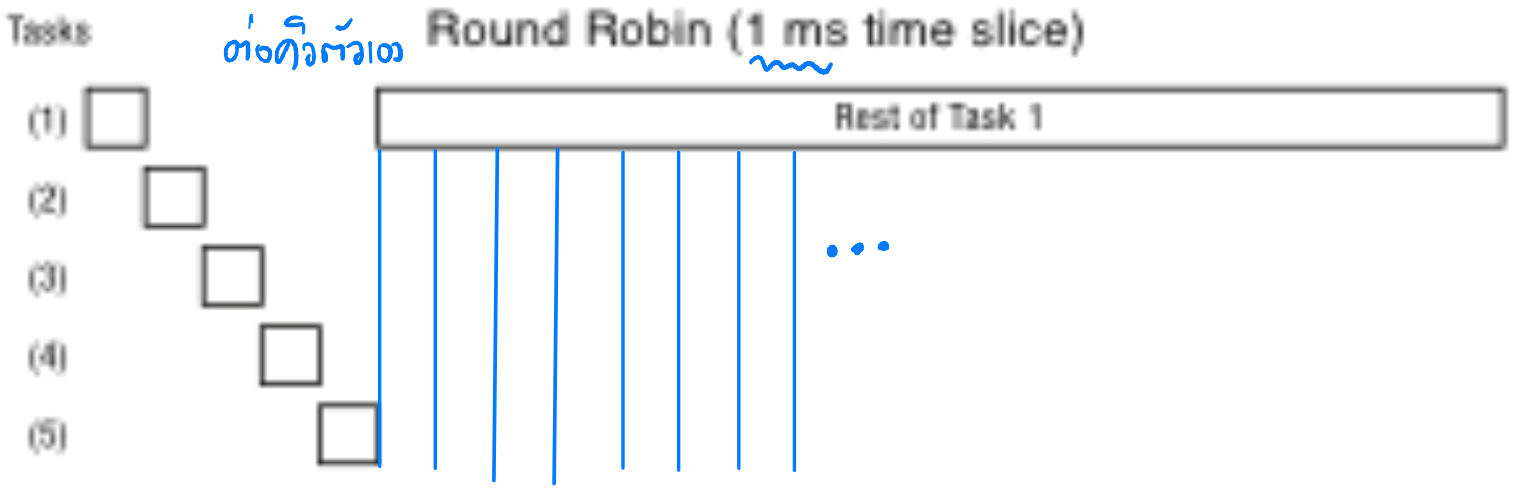
- Each task gets resource for a fixed period of time (time quantum)
  - If task doesn't complete, it goes back in line
- Need to pick a time quantum
  - What if time quantum is too long?
    - Infinite?
  - What if time quantum is too short?
    - One instruction?

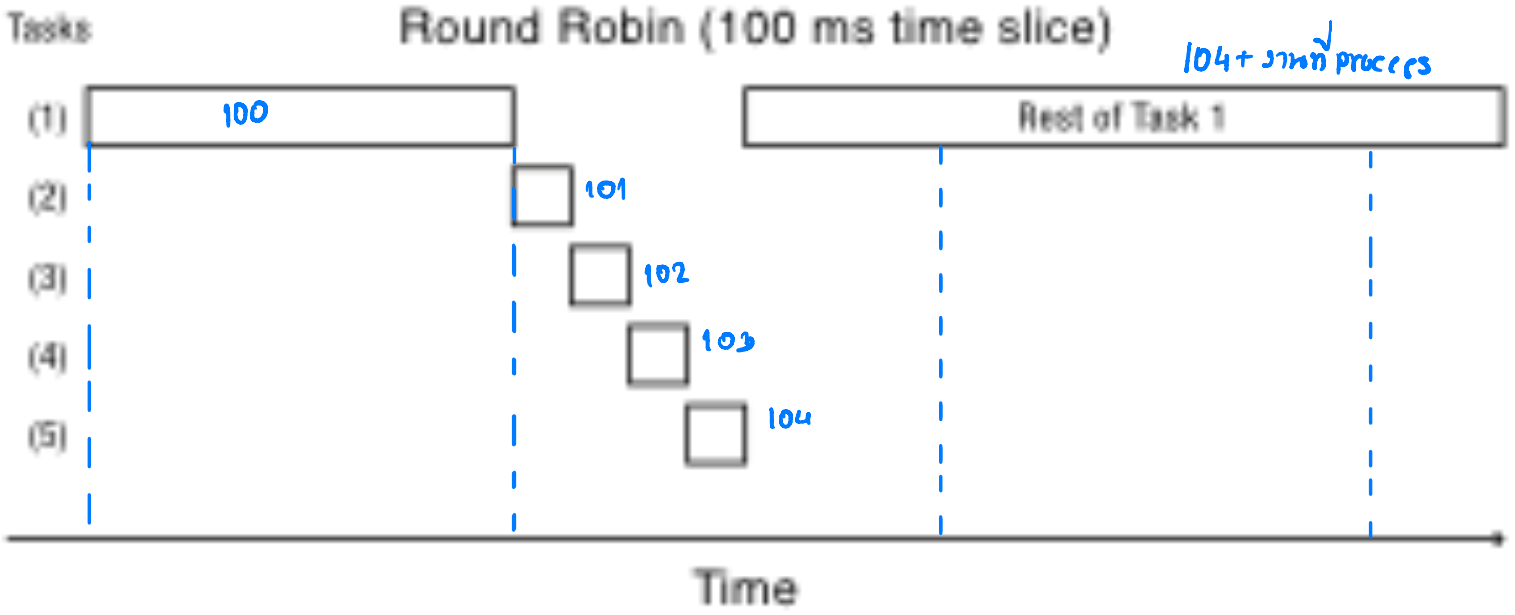
สร้างมาเพื่อจำลองแทน SJF

ต้องหาขนาด ของทำงาน Job แล้วหา time quantum ให้เหมาะสม

Round Robin

time quantum 1ms





Round Robin vs. FIFO

มันดีกว่า ตรงตามข้อ

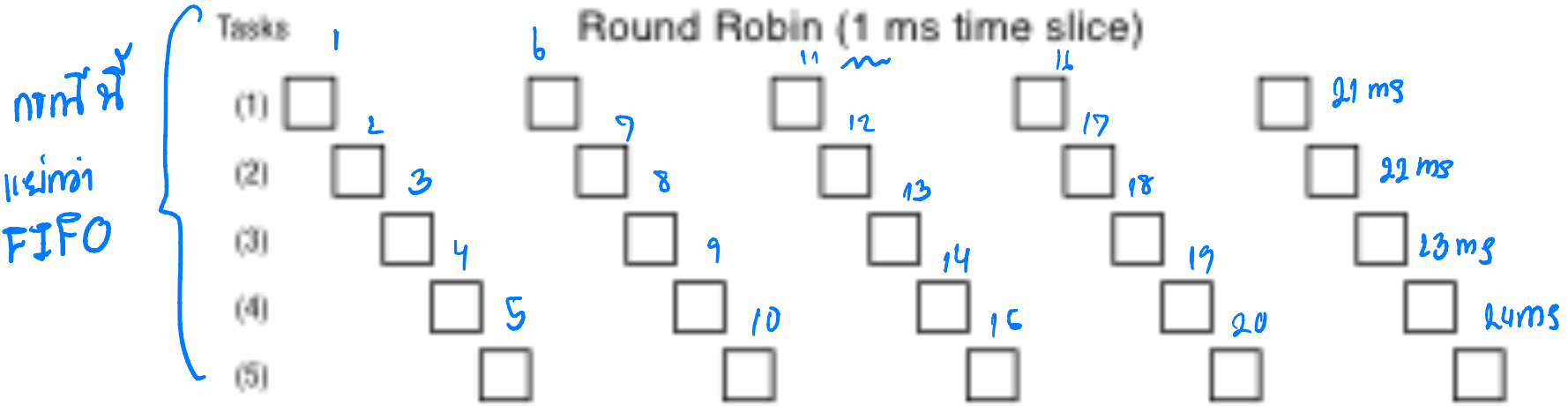
- Assuming zero-cost time slice, is Round Robin always better than FIFO?

ตอบหน้าต่อไป →

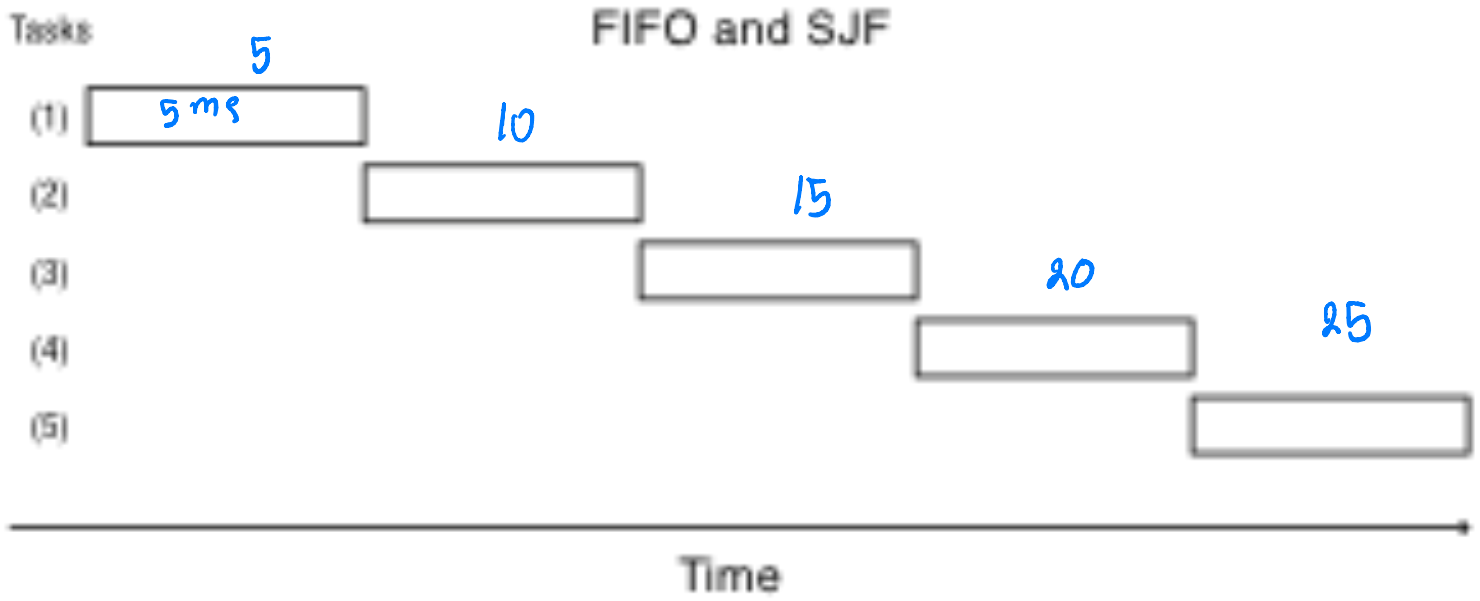
1 Job = 5 ms

Round Robin vs. FIFO

Job ถูก Slice ออกเป็น 5 ส่วน



กราฟนี้แสดงว่า FIFO



Round Robin = Fairness?

SJF เสียเวลา

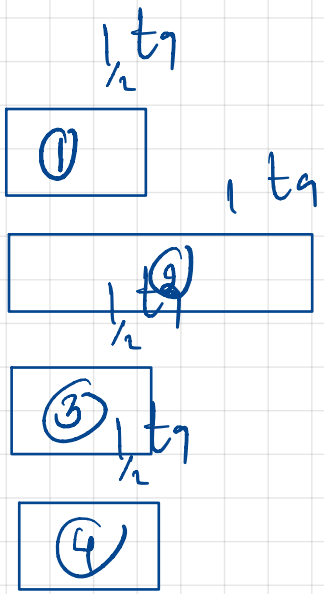
- Is Round Robin always fair?

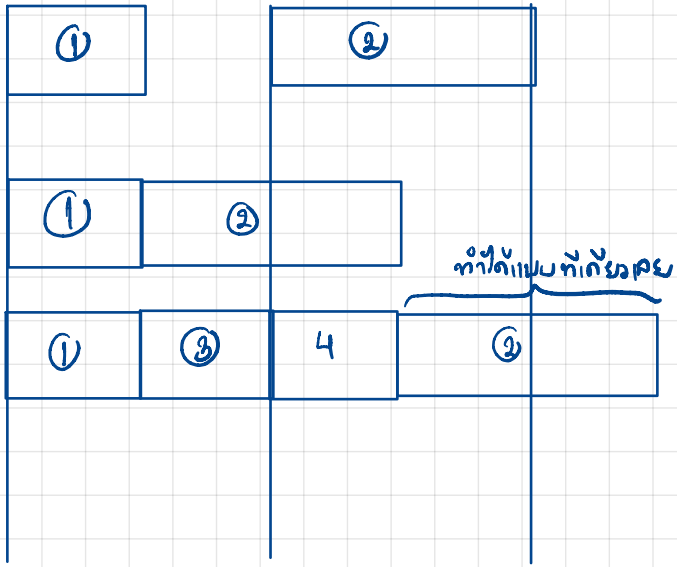
  คือ FIFO ที่ทำใน 1 time quantum ใครมาก่อนได้ใช้ก่อน

- What is fair?
  - FIFO? ใครมาก่อนได้ใช้ resource ก่อน
  - Equal share of the CPU? แม่ง หรือจัดสรร CPU ให้เท่ากัน
  - What if some tasks don't need their full share? ทำงานไม่เต็ม 1 time quantum มีการสลาชยาสมัย Round Robin ไม่มี
  - Minimize worst case divergence? ถ้าใน ตัว มีแค่ 1 Job จะวน time quantum ไหม (สรุป คือ หมด 1 tq ก็ต่อในม่แม้ต่างตัวเองก็ตาม)
    - Time task would take if no one else was running
    - Time task takes under scheduling algorithm

      เวลาที่ใช้ในการเลือก

    - ขนาด tq ที่เหมาะสม ถ้าเลือกเหมาะสม = ดี, ไม่เหมาะสม = ไม่ดี แล้วจะรู้ได้ไงล่ะ? ถือเป็นปัญหาอยู่

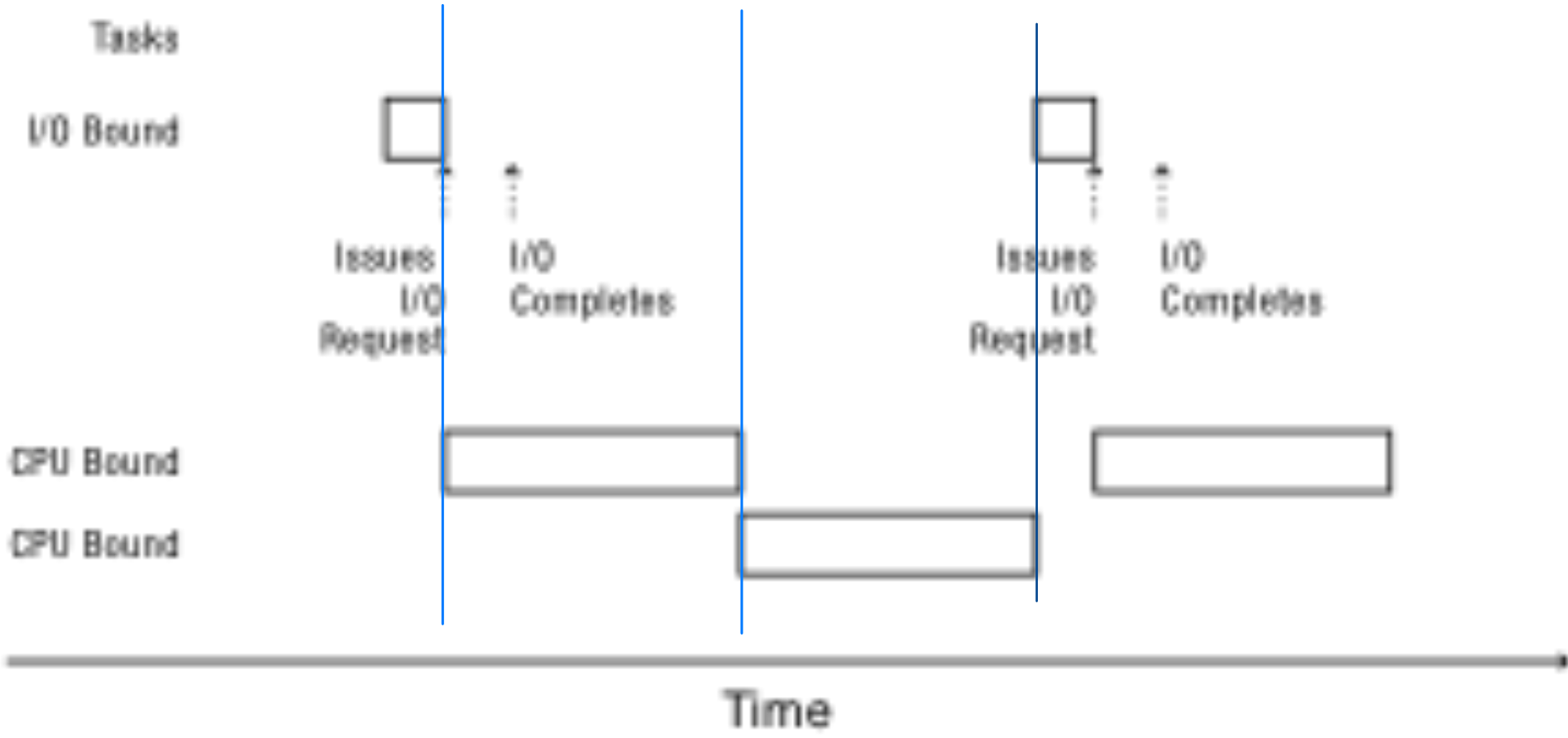




ไม่เฉยเวลาให้น้ำ:มน
เฉยเวลาให้น้ำ:มน
} RR fairness

คิดเวลาตอบสนองให้ตัวที่ถูกทำเสร็จ

Mixed Workload



เอา Job ที่ใช้เวลา เหลือให้ระบบ และ

Max-Min Fairness

-ใช้น้อย เริ่มก่อน
-ใช้มาก ระบบให้เยอะ แต่ได้เวลามากกว่า ใจ

WOW!

- 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

Multi-level Feedback Queue (MFQ)

- Goals:
  - Responsiveness
  - Low overhead
  - Starvation freedom
  - Some tasks are high/low priority
  - Fairness (among equal priority tasks)
- Not perfect at any of them!
  - Used in Linux (and probably Windows, MacOS)

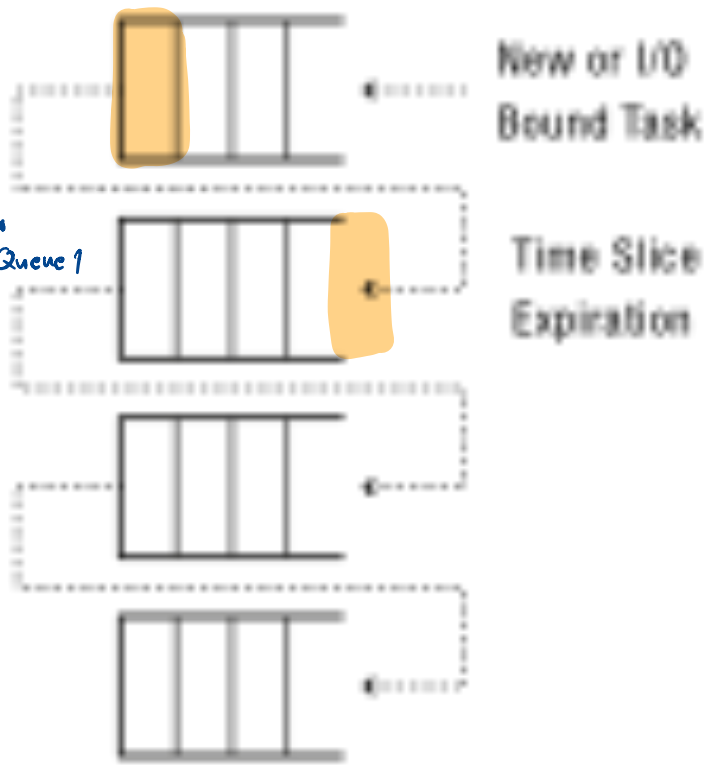
MFQ

- 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

Priority	Time Slice (ms)	Round Robin Queues
1	10	
2	20	
3	40	
4	80	

- ถ้า Job ใน Queue 1 emtry : ไปทำ Job ใน Queue 2
- ทำงานไม่จบต่อหลัง 2 & ลด priority เพื่อเวลา / tq จะได้มากขึ้น
- ถ้าทำไม่จบใน 1 tq ก็ต่อ Queue เดิม + คืนเวลาให้ระบบ
- มีการ interrupt if ยังไม่ได้ทำจักที่ (เป็น time out) : จะเลื่อนไป Queue ถัด
  *ไม่ทำจักก็เลื่อนเรื่อยๆ จนอยู่ Queue 1



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)

Job ขนาดไม่เท่ากัน

- If tasks are variable in size, Round Robin approximates SJF.

Job ขนาดเท่ากันหมด = แย่สุด

- If tasks are equal in size, Round Robin will have very poor average response time.

Job ที่มีการคำนวณก่อนจบ ถ้าไม่มีการขอเขียนไฟล์ แต่ max-min ขอเขียนไฟล์

- 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.
- Round Robin and Max-Min fairness both avoid starvation.
- By manipulating the assignment of tasks to priority queues, an MFQ scheduler can achieve a balance between responsiveness, low overhead, and fairness.