

TCSS 422: OPERATING SYSTEMS

CPU Scheduling, Multi-level Feedback Queue (MLFQ)



Wes J. Lloyd
School of Engineering and Technology,
University of Washington - Tacoma

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

FEEDBACK FROM 10/8

- Why is interrupting an interrupt needed?
- What is a preemptive kernel (with respect to interrupts)?
- What is the importance of job preemption for CPU scheduling?
- Difference between turnaround time and execution time?
- Importance of all the scheduling metrics combined
 - Turnaround time, Response time, Jain's fairness index

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.2

FEEDBACK - 2

- Does the importance of a job matter when scheduling?
 - Instead of importance, operating systems track Job PRIORITY
 - The Linux “NICE” value provides a suggestion on which jobs should be scheduled
- Linux (NICE) value
 - Provides a suggestion regarding job priority
 - Does not map directly to Process PRIORITY
 - Values from -20 (high priority) to 19 (low priority)

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.3

FEEDBACK - 3

- `ps ax -o pid,ni,pri,cmd`
- `htop`
- Linux Job priority value
 - System maintains this value, influence by NICE value
 - Not user editable
 - Values: (higher is higher)
RT (Real Time), 0 to 99 (usr/krn), and 100 to 139 (sys?)

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.4

FEEDBACK - 4

- What sets the OS context switch time quantum?
 - Typically ~ 10ms
- How does job priority factor into fairness?
 - Fairness is generally considered among jobs of the same priority
 - Jain's fairness index is only calculated among jobs of the same priority level
 - **Higher priority Job(s) take precedence over lower priority jobs**

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.5

OBJECTIVES

- C Tutorial
- Quiz 1 – Active Reading
- Program 1 – MASH Shell
- **CPU Scheduling:**
 - Chapter 7 – Introduction to Scheduling
 - Chapter 8 – Multi-level Feedback Queue
 - Chapter 9 – Proportional Share Scheduler
 - Linux - Completely Fair Scheduler (CFS)

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma


L5.6

ASSIGNMENT #1 INTRODUCTION

[HTTP://FACULTY.WASHINGTON.EDU/WLLOYD/
COURSES/TCSS422/ASSIGNMENTS/
TCSS422_F2018_A1.PDF](http://FACULTY.WASHINGTON.EDU/WLLOYD/COURSES/TCSS422/ASSIGNMENTS/TCSS422_F2018_A1.PDF)

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma




L5.7

CHAPTER 7- SCHEDULING: INTRODUCTION

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma



L5.8

SCHEDULING METRICS

- **Metrics:** A standard measure to quantify to what degree a system possesses some property. Metrics provide repeatable techniques to quantify and compare systems.
- **Measurements** are the numbers derived from the application of metrics
- Scheduling Metric #1: **Turnaround time**
- The time at which the job completes minus the time at which the job arrived in the system

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

- How is turnaround time different than execution time?

October 10, 2018

TCCS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.9

SCHEDULING METRICS - 2

- Scheduling Metric #2: **Fairness**
 - Jain's fairness index
 - Quantifies if jobs receive a fair share of system resources

$$\mathcal{J}(x_1, x_2, \dots, x_n) = \frac{(\sum_{i=1}^n x_i)^2}{n \cdot \sum_{i=1}^n x_i^2}$$

- n processes
- x_i is time share of each process
- worst case = $1/n$
- best case = 1
- Consider $n=3$, worst case = .333, best case=1
- With $n=3$ and $x_1=.2, x_2=.7, x_3=.1$, fairness=.62
- With $n=3$ and $x_1=.33, x_2=.33, x_3=.33$, fairness=1

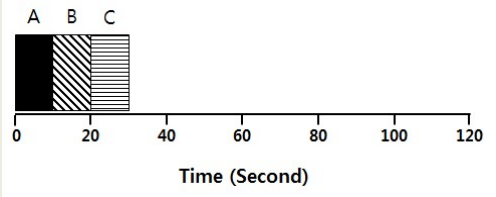
October 10, 2018

TCCS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.10

SCHEDULERS

- **FIFO: first in, first out**
 - Very simple, easy to implement
- **Consider**
 - 3 x 10sec jobs, arrival: A B C

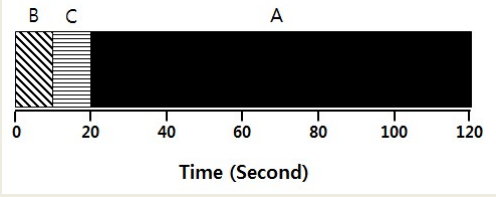


Average turnaround time = $\frac{10 + 20 + 30}{3} = 20 \text{ sec}$

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.11
------------------	---	-------

SJF: SHORTEST JOB FIRST

- **Given that we know execution times in advance:**
 - Run in order of duration, shortest to longest
 - Non preemptive scheduler
 - This is not realistic
 - Arrival: A B C



Average turnaround time = $\frac{10 + 20 + 120}{3} = 50 \text{ sec}$

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.12
------------------	---	-------

SJF: WITH RANDOM ARRIVAL

- If jobs arrive at any time:
- A @ t=0sec, B @ t=10sec, C @ t=10sec

[B,C arrive]

A

B C

0 20 40 60 80 100 120

Time (Second)

$$\text{Average turnaround time} = \frac{100 + (110 - 10) + (120 - 10)}{3} = 103.33 \text{ sec}$$

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.13
------------------	---	-------

STCF - 2

- Consider:
- $A_{len}=100, A_{arrival}=0$
- $B_{len}=10, B_{arrival}=10, C_{len}=10, C_{arrival}=10$

[B,C arrive]

A B C

A

0 20 40 60 80 100 120

Time (Second)

$$\text{Average turnaround time} = \frac{(120 - 0) + (20 - 10) + (30 - 10)}{3} = 50 \text{ sec}$$

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.14
------------------	---	-------

SCHEDULING METRICS - 3

- Scheduling Metric #3: **Response Time**
- Time from when job arrives until it starts execution

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

- STCF, SJF, FIFO
 - can perform poorly with respect to response time

What scheduling algorithm(s) can help minimize response time?

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.15

RR: ROUND ROBIN

- Run each job awhile, then switch to another distributing the CPU evenly (fairly)
- Scheduling Quantum is called a time slice
- Time slice is a multiple of the timer interrupt period.

RR is fair, but performs poorly on metrics such as turnaround time

Process	Burst Time
P1	12
P5	5

Round Robin scheduling algorithm Gantt chart

Scheduling Quantum = 5 seconds

➔

P1	P2	P3	P4	P5	P1	P2	P4	P1	
0	5	10	14	19	24	29	32	37	39

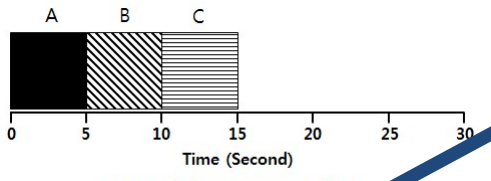
October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.16

RR EXAMPLE

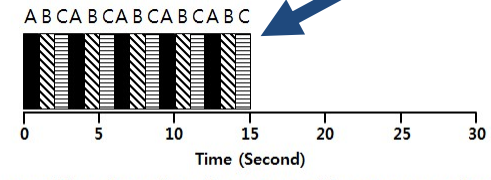
▪ ABC arrive at time=0, each run for 5 seconds



SJF (Bad for Response Time)

OVERHEAD not considered

$$T_{average\ response} = \frac{0 + 5 + 10}{3} = 5sec$$



RR with a time-slice of 1sec (Good for Response Time)

$$T_{average\ response} = \frac{0 + 1 + 2}{3} = 1sec$$

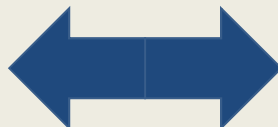
October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.17
------------------	---	-------

ROUND ROBIN: TRADEOFFS

Short Time Slice

Fast Response Time

High overhead from context switching



Long Time Slice

Slow Response Time

Low overhead from context switching

▪ **Time slice impact:**

- Turnaround time (for earlier example):
 $ts(1,2,3,4,5) = 14, 14, 13, 14, 10$
- Fairness: round robin is always fair, $J=1$

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.18
------------------	---	-------

SCHEDULING WITH I/O

- **STCF scheduler**
 - A: CPU=50ms, I/O=40ms, 10ms intervals
 - B: CPU=50ms, I/O=0ms
 - Consider A as 10ms subjobs (CPU, then I/O)

- **Without considering I/O:**

CPU utilization = $100/140 = 71\%$

Time (msec)

Poor Use of Resources

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.19
------------------	---	-------

SCHEDULING WITH I/O - 2

- **When a job initiates an I/O request**
 - A is blocked, waits for I/O to compute, frees CPU
 - STCF scheduler assigns B to CPU

- **When I/O completes → raise interrupt**
 - Unblock A, STCF goes back to executing A: (10ms sub-job)

Cpu utilization = $100/100 = 100\%$

Time (msec)

Overlap Allows Better Use of Resources

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.20
------------------	---	-------

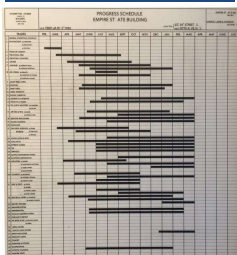
W Which scheduler, thus far, best address fairness and average response time of jobs?

Respond at PollEv.com/wesleylloyd641
Text **WESLEYLLOYD641** to **22333** once to join, then **1, 2, 3, 4, 5...**

- First In - First Out (FIFO) **1**
- Shortest Job First (SJF) **2**
- Shortest Time to Completion First (STCF) **3**
- Round Robin **4**
- None of the Above **5**
- All of the Above **6**

Start the presentation to see live content. Still no live content? Install the app or get help at PollEv.com/app Total Results

**CHAPTER 8 –
MULTI-LEVEL FEEDBACK
QUEUE (MLFQ) SCHEDULER**



October 10, 2018 TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma L5.22

MULTI-LEVEL FEEDBACK QUEUE

- Objectives:
 - Improve turnaround time:
Run shorter jobs first
 - Minimize response time:
Important for interactive jobs (UI)
- Achieve without a priori knowledge of job length

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.23
------------------	---	-------

MLFQ - 2

Round-Robin
within a Queue

- Multiple job queues
- Adjust job priority based on observed behavior
- Interactive Jobs
 - Frequent I/O → keep priority high
 - Interactive jobs require fast response time (GUI/UI)
- Batch Jobs
 - Require long periods of CPU utilization
 - Keep priority low

[High Priority] Q8 → (A) → (B)

Q7

Q6

Q5

Q4 → (C)

Q3

Q2

[Low Priority] Q1 → (D)

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.24
------------------	---	-------

MLFQ: DETERMINING JOB PRIORITY

- New arriving jobs are placed into highest priority queue
- If a job uses its entire time slice, priority is reduced (↓)
 - Jobs appears CPU-bound (“batch” job), not interactive (GUI/UI)
- If a job relinquishes the CPU for I/O priority stays the same

MLFQ approximates SJF

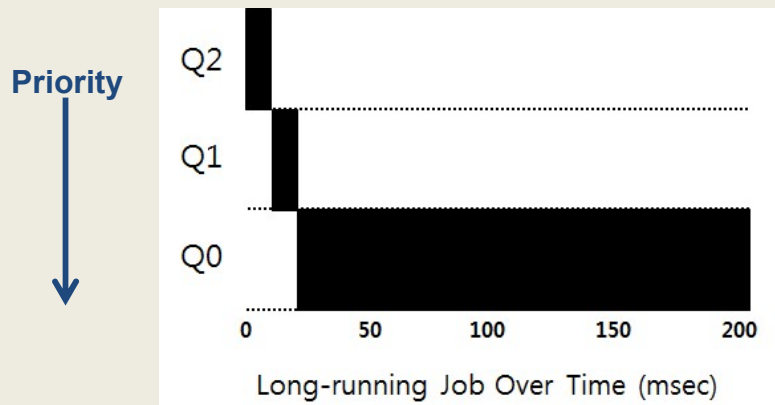
October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.25

MLFQ: LONG RUNNING JOB

- Three-queue scheduler, time slice=10ms



October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.26

MLFQ: BATCH AND INTERACTIVE JOBS

- $A_{arrival_time} = 0ms, A_{run_time} = 200ms,$
- $B_{run_time} = 20ms, B_{arrival_time} = 100ms$

Priority

Scheduling multiple jobs (ms)

A:

B:

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.27
------------------	---	-------

MLFQ: BATCH AND INTERACTIVE - 2

- Continuous interactive job (B) with long running batch job (A)
 - Low response time is good for B
 - A continues to make progress

The MLFQ approach keeps interactive job(s) at the highest priority

A Mixed I/O-intensive and CPU-intensive Workload (msec)

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.28
------------------	---	-------

MLFQ: ISSUES

- **Starvation**

[High Priority] Q8 → (A) → (B) → (C) → (D) → (E) → (F)

Q7

Q6

Q5

Q4

Q3

Q2

[Low Priority] Q1 → (G) → (H) *CPU bound batch job(s)*

October 10, 2018 TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma L5.29

MLFQ: ISSUES - 2

- **Gaming the scheduler**
 - Issue I/O operation at 99% completion of the time slice
 - Keeps job priority fixed – never lowered
- **Job behavioral change**
 - CPU/batch process becomes an interactive process

[High Priority] Q8 → (A) → (B) → (C) → (D) → (E) → (F)

Q7

Q6

Q5

Q4

Q3


Q2

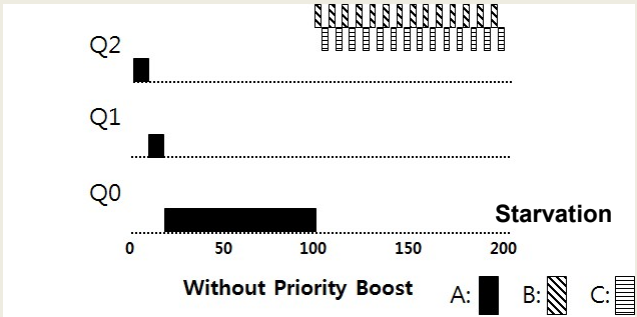
[Low Priority] Q1 → (G) → (H) *CPU bound batch job(s)*

Priority becomes stuck →


October 10, 2018 TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma L5.30

RESPONDING TO BEHAVIOR CHANGE






Without Priority Boost A: B: C:

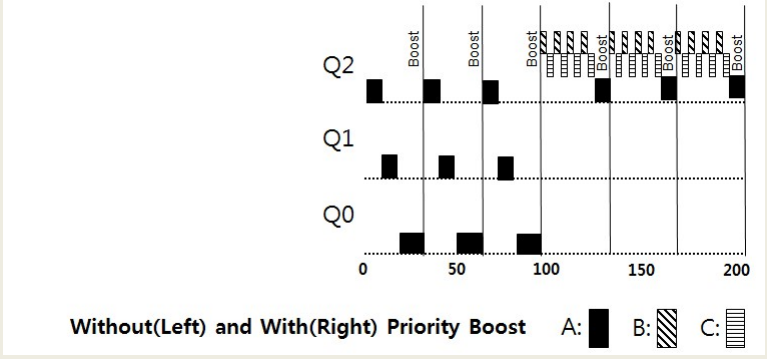


- **Priority Boost**
 - **Reset all jobs to topmost queue after some time interval S**

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.31
------------------	---	-------

RESPONDING TO BEHAVIOR CHANGE - 2





Without(Left) and With(Right) Priority Boost A: B: C:

- **With priority boost**
 - **Prevents starvation**

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.32
------------------	---	-------

PREVENTING GAMING

- Improved time accounting:
 - Track total job execution time in the queue
 - Each job receives a fixed time allotment
 - When allotment is exhausted, job priority is lowered

Without(Left) and With(Right) Gaming Tolerance

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.33
------------------	---	-------

MLFQ: TUNING

- Consider the tradeoffs:
 - How many queues?
 - What is a good time slice?
 - How often should we “Boost” priority of jobs?
 - What about different time slices to different queues?

Example) 10ms for the highest queue, 20ms for the middle, 40ms for the lowest

October 10, 2018	TCSS422: Operating Systems [Fall 2018] School of Engineering and Technology, University of Washington - Tacoma	L5.34
------------------	---	-------

PRACTICAL EXAMPLE

- Oracle Solaris MLFQ implementation
 - 60 Queues →
w/ slowly increasing time slice (high to low priority)
 - Provides sys admins with set of editable table(s)
 - Supports adjusting time slices, boost intervals, priority changes, etc.
- Advice
 - Provide OS with hints about the process
 - Nice command → Linux

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.35

MLFQ RULE SUMMARY

- The refined set of MLFQ rules:
- **Rule 1:** If $\text{Priority}(A) > \text{Priority}(B)$, A runs (B doesn't).
- **Rule 2:** If $\text{Priority}(A) = \text{Priority}(B)$, A & B run in RR.
- **Rule 3:** When a job enters the system, it is placed at the highest priority.
- **Rule 4:** Once a job uses up its time allotment at a given level (regardless of how many times it has given up the CPU), its priority is reduced (i.e., it moves down on queue).
- **Rule 5:** After some time period S , move all the jobs in the system to the topmost queue.

October 10, 2018

TCSS422: Operating Systems [Fall 2018]
School of Engineering and Technology, University of Washington - Tacoma

L5.36

Jackson deploys a 3-level MLFQ scheduler. The time slice is 1 for high priority jobs, 2 for medium priority, and 4 for low priority. This MLFQ scheduler performs a Priority Boost every 6 timer units. When the priority boost fires, the current job is preempted, and the next scheduled job is run in round-robin order.

Job	Arrival Time	Job Length
A	T=0	4
B	T=0	16
C	T=0	8

(11 points) Show a scheduling graph for the MLFQ scheduler for the jobs above. Draw vertical lines for key events and be sure to label the X-axis times as in the example. Please draw clearly. An unreadable graph will lose points.

