CS 310 Operating Systems

Lecture 23 Scheduling – Basics

Ravi Mittal
IIT Goa

Acknowledgements !

- Contents of the class -presentation have been taken from various sources. Thanks are due to the original content creators:
  - CS162, Operating System and Systems Programming, University of California, Berkeley
  - Book: Modern Operating Systems, Andrew Tenenbaum, and Herbert Bos, 4th Edition, Pearson
  - Book: Operating System: Three Easy Pieces, by Remzi H Arpaci-Dusseau, Andrea C Arpaci-Dusseau, Chapter 7, CPU Scheduling

Reading

- Book: Modern Operating Systems, Andrew Tenenbaum, and Herbert Bos, 4th Edition, Pearson
  - Chapter 2
- Book: Operating System: Three Easy Pieces, by Remzi H Arpaci-Dusseau, Andrea C Arpaci-Dusseau, Chapter 7, CPU Scheduling
  - https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched.pdf

In this lecture we will study

- Basic Concepts
- Categories of Scheduling Algorithms

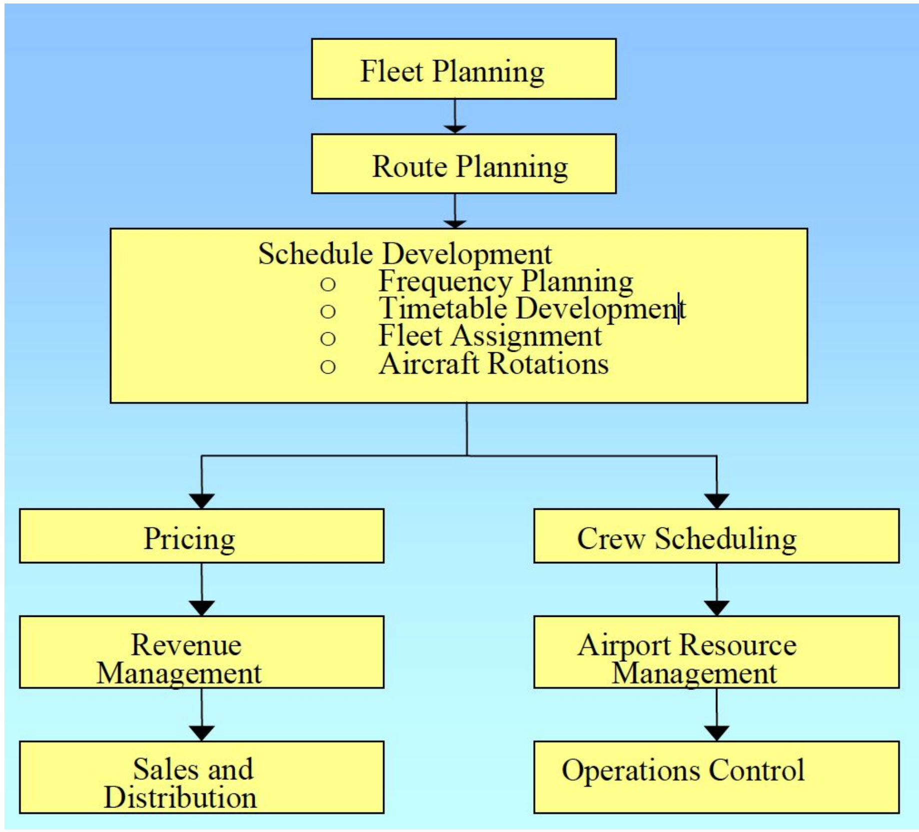
Basic Concepts

Scheduling: All About Queues

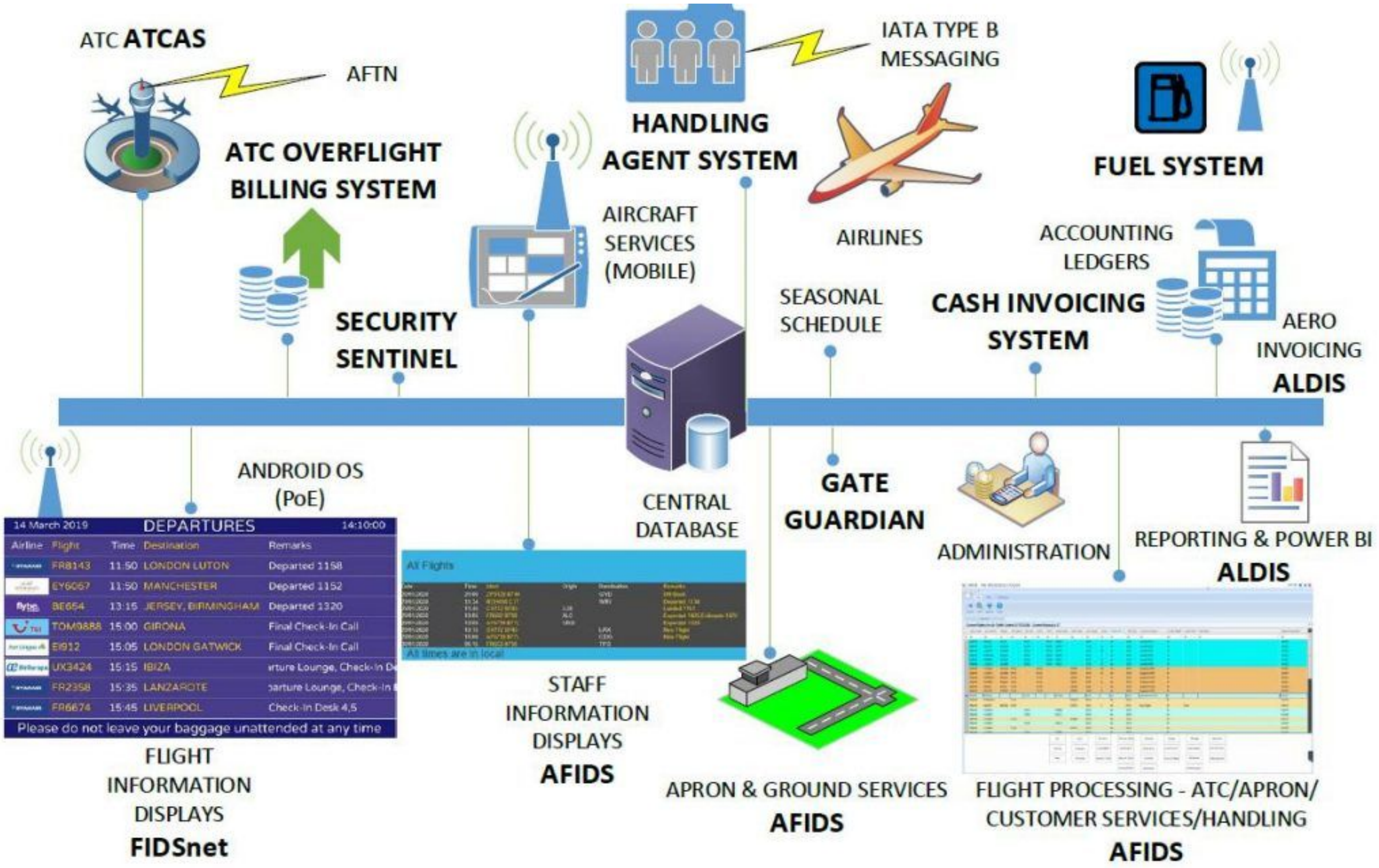
Airline scheduling

- Given a set of routes to be operated in an airline network, and a fleet of aircraft, schedule development involves
  - Frequency planning (how often?)
  - Timetable development (at what times?)
  - Fleet assignment (what type of aircraft?)
  - Aircraft rotation planning (network balance)

Complex ?



Airport Scheduling - Complexity



14 March 2019 DEPARTURES 14:10:00

Airline	Flight	Time	Destination	Remarks
	FR8143	11:50	LONDON LUTON	Departed 1158
	EY6067	11:50	MANCHESTER	Departed 1152
flybe	BE654	13:15	JERSEY, BIRMINGHAM	Departed 1320
TUI	TOM9888	15:00	GIRONA	Final Check-In Call
	EI912	15:05	LONDON GATWICK	Final Check-In Call
	UX3424	15:15	IBIZA	rture Lounge, Check-In De
	FR2358	15:35	LANZAROTE	parture Lounge, Check-In
	FR6674	15:45	LIVERPOOL	Check-In Desk 4,5

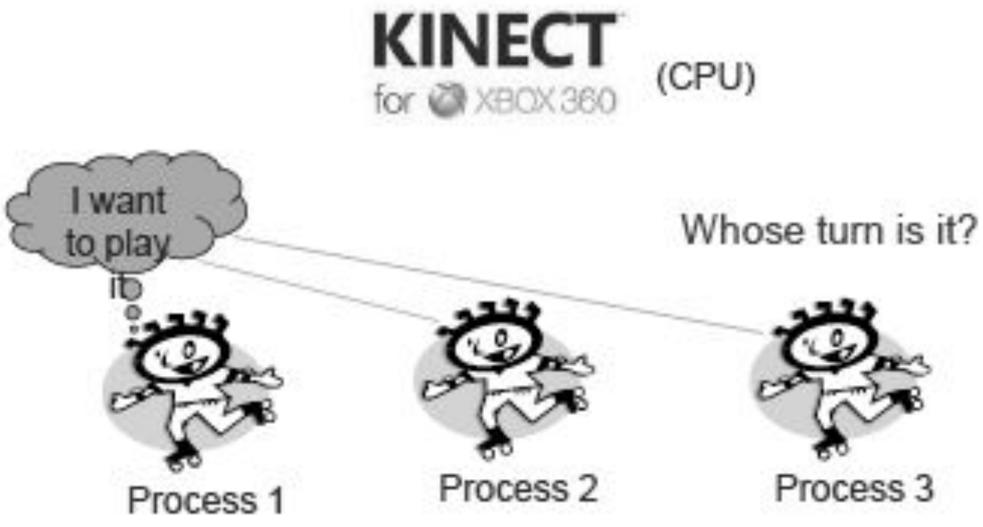
Please do not leave your baggage unattended at any time

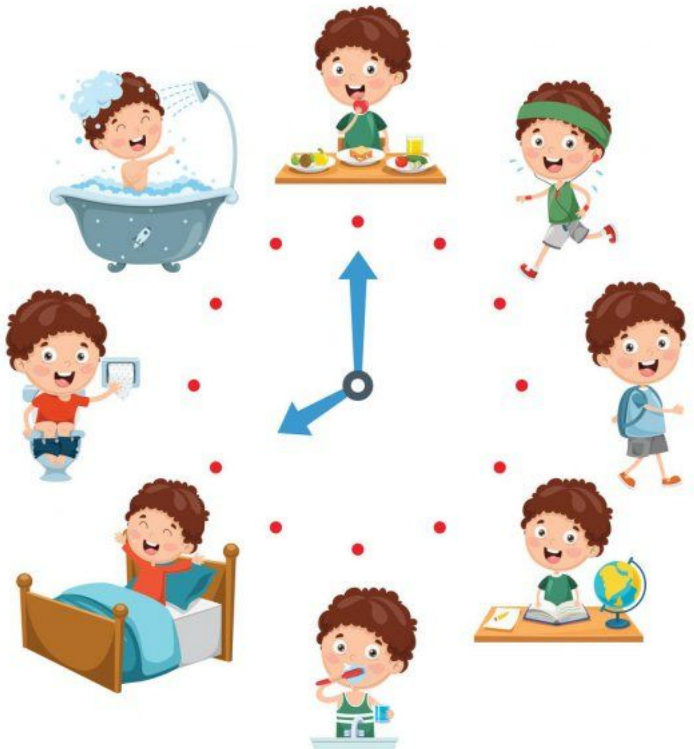
Lift scheduling

Lift Scheduling

- Parameters
  - Average waiting time for users
  - Lift utilization
  - Power consumption
  - Priority at certain floors
  - Lift speeds – Uniform vs variable

Real-life – Scheduling





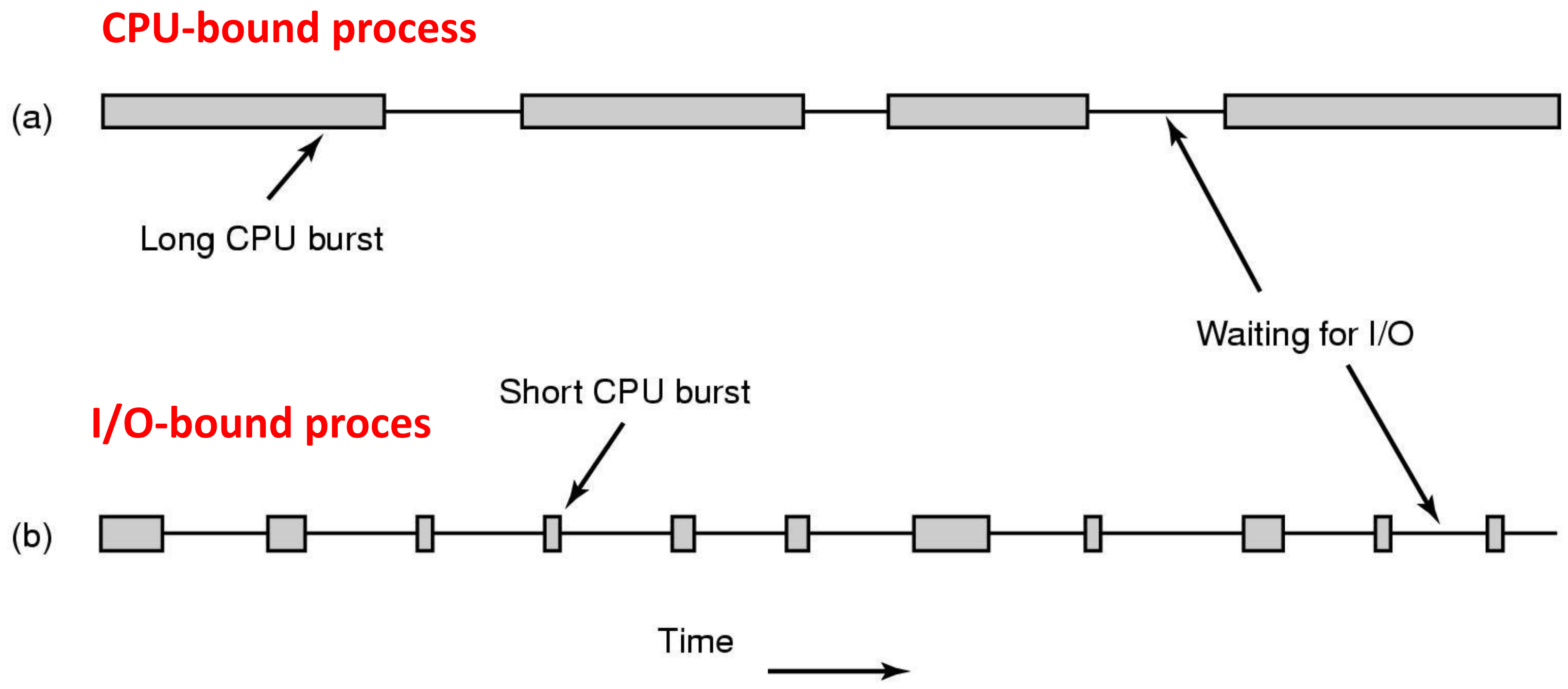
Scheduling is important activity

- PCs, Laptops
- Batch Servers
- Time Sharing Systems
- Networked Servers
- Mobile Devices
- Many other devices

Scheduling in PCs, Laptops

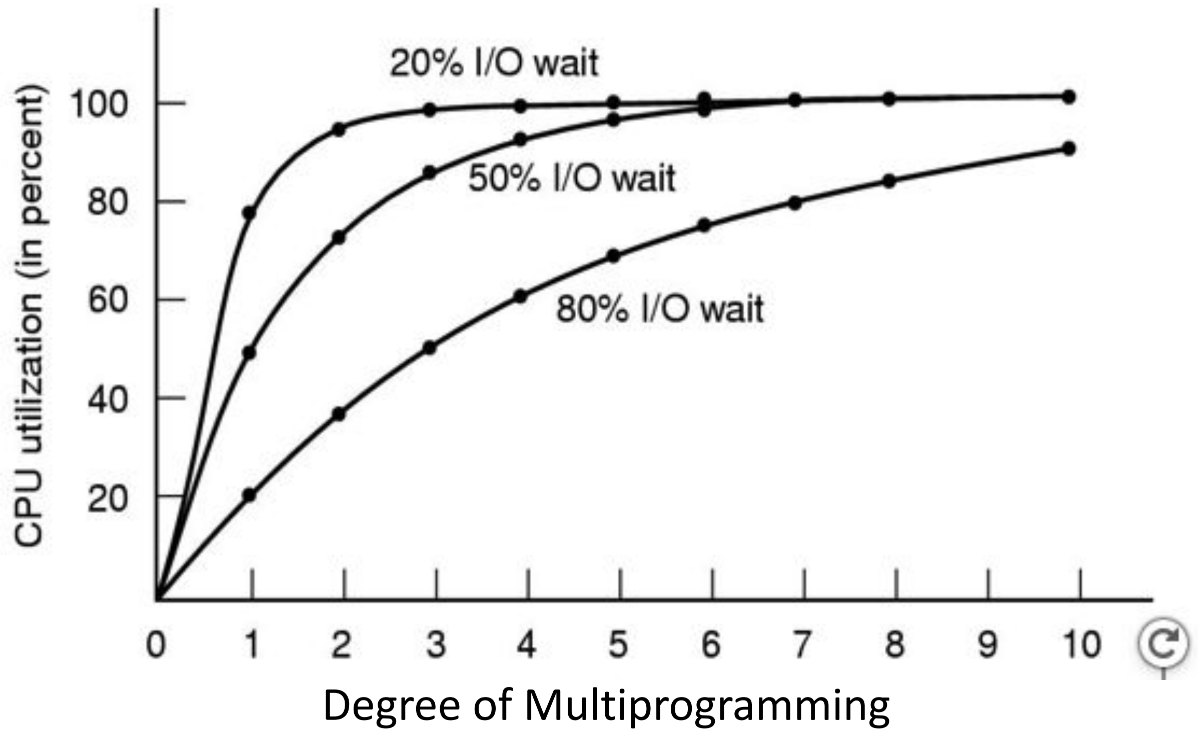
- Only one user
- Less important if user runs only one or two applications
- It becomes important if users runs multiple applications concurrently
  - Video Games
  - Music
  - Browsing
  - Etc
- With virtualized PC, there is a large number of processes running at any time
  - Scheduling becomes important

CPU Bursts



- As processors become faster, processes tend to become more I/O bound
  - Why?
  - CPU is becoming faster than the I/O

I/O bound processes vs Utilization



- I/O bound process must get a chance to run quickly
  - So that it can issue I/O request early keep disk busy
- When processes are I/O bound, it takes more number of processes to keep CPU fully occupied

When to make scheduling decisions?

- When a new process is created
  - Run a parent process or a child process as both processes are in ready state
- When a process exits
  - Which process from the ready list is chosen to run?
- When process blocks for I/O
  - On a condition variable or semaphore, or for some reason for blocking
  - Does scheduler know this dependency ? Eg Process A has made wait() system call
- I/O interrupt occurs
  - Which process to run next? Process that made I/O request or existing ready process or newly created process

When to make scheduling decisions? (continued)

- Timer Interrupt
    - OS gets control and scheduler can decide which process to run

Non-preemptive vs Preemptive scheduling

- Non-preemptive Scheduling Algorithm
  - Picks up a process to run
  - Let the process run until it blocks (for I/O or waiting for another process) or voluntarily releases the CPU
  - A process may run for hours; it will not be forcibly suspended
    - No scheduling decisions are made during clock interrupts
- Preemptive Scheduling
  - Picks up a process to run
  - lets the process run for a maximum of some fixed time
  - At the end of time period, timer interrupt occurs
  - In Kernel mode, scheduler picks up another ready process to run

Categories of Scheduling Algorithms

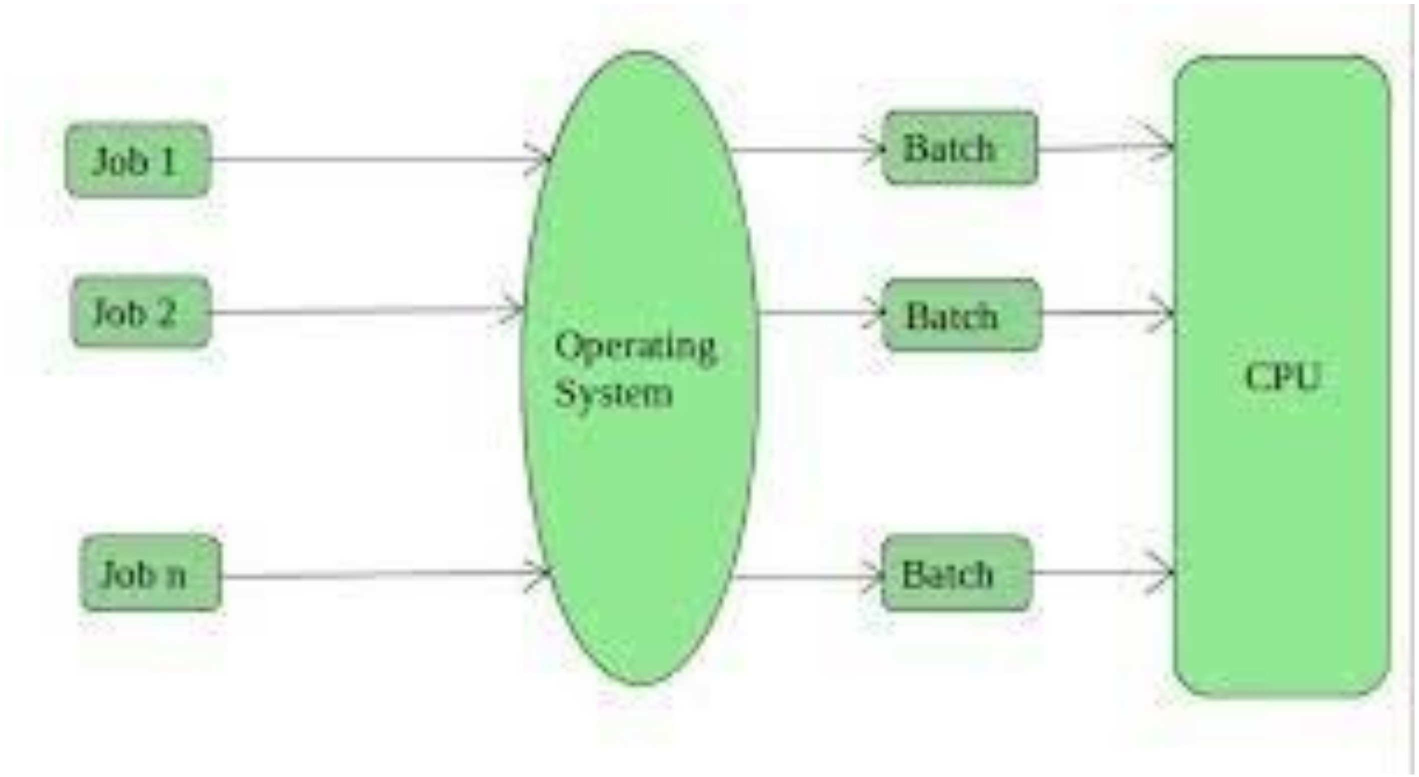
Scheduling Algorithms - types

- Different environment require different scheduling algorithms
  - Different applications have different goals requires appropriate scheduling algorithms
- Categories of Scheduling Algorithms
  - Batch
  - Interactive
  - Real time (deadlines)

Batch Systems

- Still in widespread use
- Example: Payroll, Inventory, Accounts receivable, Accounts payable, interest calculation (at banks), Claim processing (insurance) etc
  - periodic tasks
- No users impatiently waiting at their terminals for a quick response to a short request
- Non-preemptive algorithms, or preemptive algorithms with long time periods for each process
- Less context switches improved performance

Batch System



Interactive Systems

- Preemption is necessary in interactive systems
    - Otherwise one process may hog the CPU
    - Program bug may cause it to run indefinitely
- Servers also need preemption

Real-time system

- What is a real-time system?
  - Deadline
    - Hard
    - Soft
  - Deadlines must be met
  - Real-time OS (RTOS) ?
- Predictability is important
  - Example: Audio processing vs video processing

- How about Linux and Windows ?
  - Are they RTOS ?

In a modern multicore processor

- Processor Scheduling
- Bus scheduling
- Network Scheduling
- I/O scheduling

Role of Dispatcher vs. Scheduler

- Dispatcher
  - Low-level mechanism
  - Responsibility: context switch
    - Save previous process state in PCB
    - Load next process state from PCB to registers
    - Change scheduling state of process (running, ready, or blocked)
    - Migrate processes between different scheduling queues
    - Switch from kernel to user mode
- Scheduler
    - High-level policy
    - Responsibility: Deciding which process/thread to run

Scheduling Policy Goals/Criteria

- Minimize Response Time
  - Response time: Time between issuing a command and getting result
- Maximize Throughput
  - Maximize operations (or jobs) per second
- Minimize Turnaround time
  - Average elapsed time
    - primarily for batch system
- Fairness
  - Share CPU among users in some equitable way

Scheduling Policy Goal: Minimize Response Time

- The time between issuing a command and getting the result
- Response time is what the user sees:
  - Time to echo a keystroke in editor
  - Time to compile a program
  - Real-time Tasks: Must meet deadlines imposed by World
- Important for interactive systems
- Example: in PCs, there are many background processes. A user request (to open a file) must be given priority over background processes
- Response time – User perception plays a big role

Scheduling Policy Goal: Maximize Throughput

- Maximize operations (or jobs) per second

- Two parts to maximizing throughput
  - Minimize overhead (for example, context-switching)
  - Efficient use of resources (CPU, disk, memory, etc)

Scheduling Policy Goal: Turnaround time

- The turnaround time of a job
  - The time job completes minus the time at which the job arrived in the system
  - $T_{turnaround} = T_{completion} - T_{arrival}$
- It measures how long the average user has to wait for the output
- A scheduling algorithm may try to optimize turnaround time and throughput
  - Higher throughput doesn’t mean minimizing turnaround time
  - Example: A scheduler that always ran short jobs and never ran long jobs
    - Excellent throughput but terrible turnaround time for the long jobs

Scheduling Policy Goal: Fairness

- Share CPU among users in some equitable way

- Fairness is not minimizing average response time
- Performance and fairness are often at odds in scheduling
  - Scheduler may optimize performance but at the cost of preventing a few jobs from running,
    - decreasing fairness

Useful metrics

- Waiting time for process *P:* time before *P* got scheduled
- Average waiting time: Average of all processes’ wait time.
- Completion time: Waiting time + Run time.
- Average completion time: Average of all processes' completion time

Lecture Summary

- Scheduling is very important function of all Operating Systems
- As we increase degree of multiprogramming CPU utilization goes up (considering low context switching overheads)
- Scheduling decisions are made under many situations
  - Process Creation, Process exit, Process blocking, I/O Interrupts, Timer Interrupts
- Preemptive vs Non-preemptive Scheduling
- Scheduling Algorithms for
  - Batch Systems
  - Interactive Systems
  - Real-time Systems

Scheduling in Batch Systems

Scheduling in Batch Systems

- First-come First-served (FCFS)
- Shortest Job First (SJF)
- Shortest Remaining Time First (STRF)
  - Preemptive version of SJF

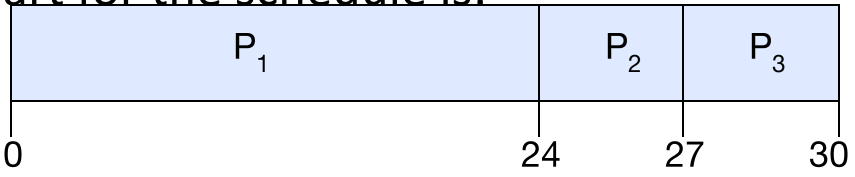
First-Come, First-Served (FCFS) Scheduling

- First-Come, First-Served (FCFS)
  - Also “First In, First Out” (FIFO) or “Run until done”
    - In Batch systems, FCFS meant one program scheduled until done (including I/O)
    - In interactive system, means keep CPU until thread blocks

- Example:

Process	Burst Time
P_1	24
P_2	3
P_3	3

  - Suppose processes arrive in the order: P_1 , P_2 , P_3
    The Gantt Chart for the schedule is:



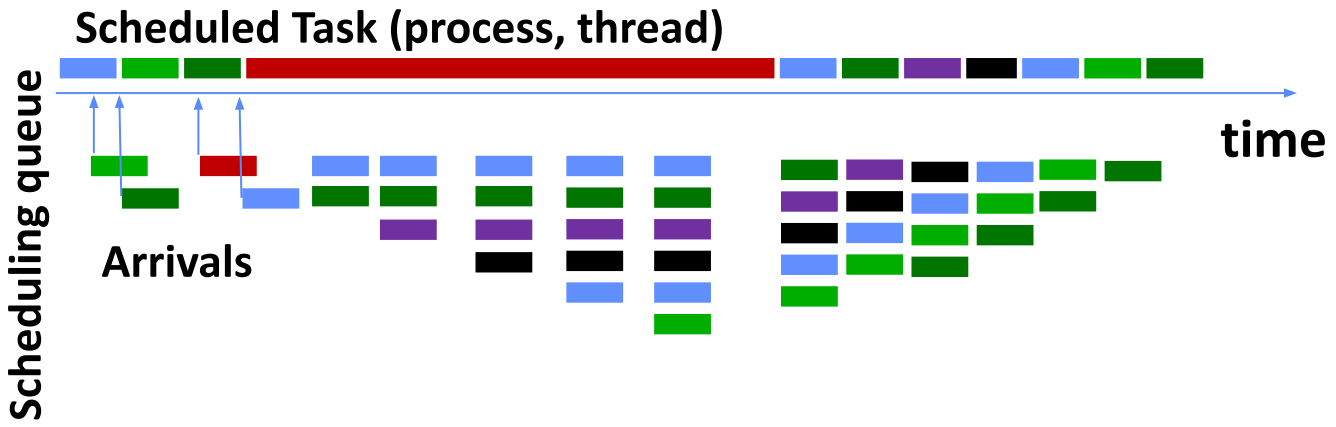
  - Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
  - Average waiting time: $(0 + 24 + 27)/3 = 17$
  - Average Completion time: $(24 + 27 + 30)/3 = 27$

First-Come, First-Served (FCFS) Scheduling

- Advantages
  - Simple algorithm
  - Easy to implement
- Disadvantage
  - Convoy Effect

FCFS: Convoy effect

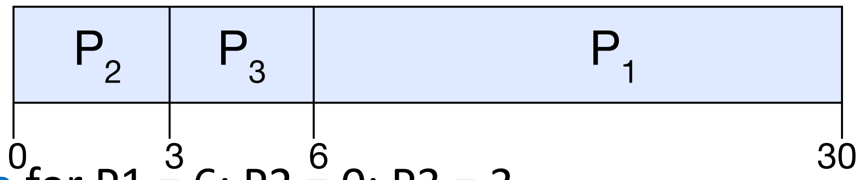
Short process stuck behind long process



With FCFS non-preemptive scheduling, convoys of small tasks tend to build up when a large one is running.

FCFS Scheduling (Cont.)

- Example continued:
  - Suppose that processes arrive in order: P2 , P3 , P1
    Now, the Gantt chart for the schedule is:



  - Waiting time for P1 = 6; P2 = 0; P3 = 3
  - Average waiting time: (6 + 0 + 3)/3 = 3
  - Average Completion time: (3 + 6 + 30)/3 = 13
- In second case:
  - Average waiting time is much better (before it was 17)
  - Average completion time is better (before it was 27)
- FCFS Pros and Cons:
  - Simple (+)
  - Short jobs get stuck behind long ones (-)
  - Non-preemptive (-)

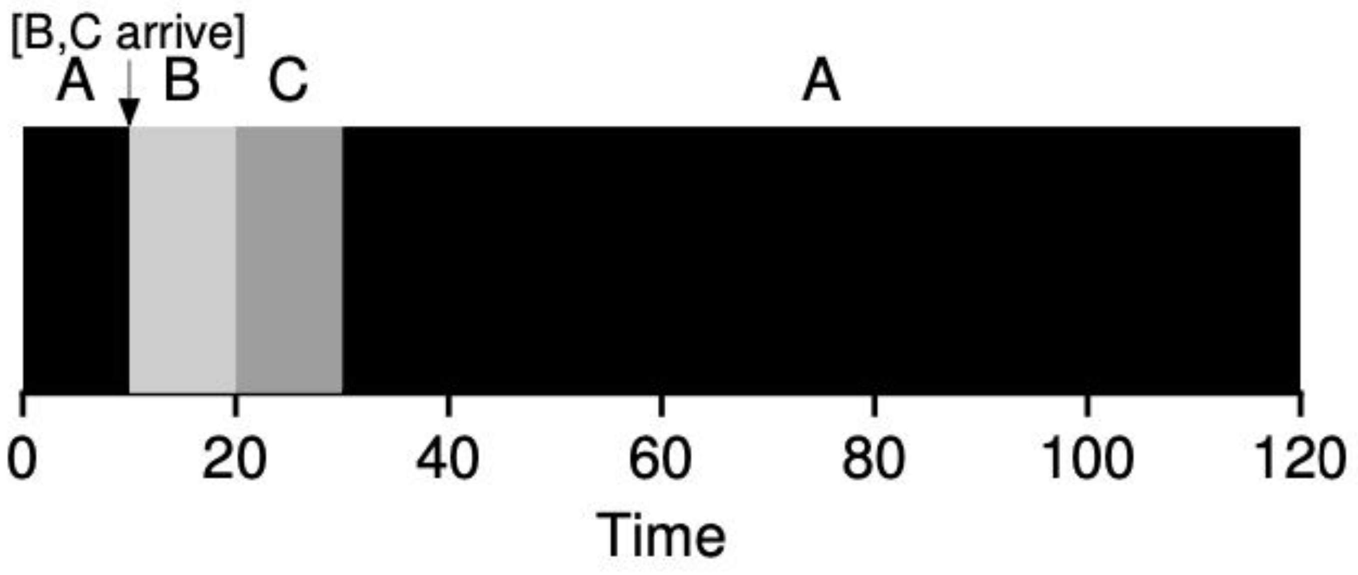
Shortest Job First (SJF)

- Non-preemptive
- Run whatever job has least amount of computation to do
  - Shortest job first scheduling runs a process to completion before running the next one
- Sometimes called "Shortest Time to Completion First" (STCF)
- Need to know run times in advance
- Provably optimal
  - 4 jobs with runs times of a,b,c,d
  - First finishes at a, second at a+b,third at a+b+c, last at a+b+c+d
  - Mean turnaround time is (4a+3b+2c+d)/4
  - Smallest time has to come first to minimize the mean turnaround time

Shortest Remaining Time First (SRTF)

- Preemptive version of Shortest job first
- If job arrives and has a shorter time to completion than the remaining time on the current job, immediately preempt CPU
- Sometimes called "Shortest Remaining Time to Completion First" (SRTCF)
- The queue of jobs is sorted by estimated job length so that short programs get to run first and not be held up by long ones
- Both SJF and SRTF:
  - These can be applied to whole program or current CPU burst
    - Idea is to get short jobs out of the system
    - Big effect on short jobs, only small effect on long ones
    - Result is better average response time

Shortest Remaining Time First (SRTF)



- Execution time of A = 100, B = 10, C =10 seconds
- A will be preempted when jobs B and C arrive
- Average Turn Around Time = 50 seconds

Lecture Summary

- Scheduling is very important aspect of overall computing system design
- Preemptive vs non-preemptive scheduling algorithm
- Scheduling algorithms differ from system to system
  - Batch System
  - Interactive System
  - Real-time System
- We have studied the following scheduling algorithms for Batch Systems
  - First Come First Served
  - Shortest Job First
  - Shortest Remaining Time First

Assumptions (Unrealistic)

- Each job runs for the same amount of time
- All jobs arrive at the same time
  - We will explicitly mention in case jobs don’t arrive at the same time
- Once started, each job runs to completion
- All jobs only use the CPU (i.e., they perform no I/O)
- The run-time of each job is known.