240-229: SDA (Operating Systems session)

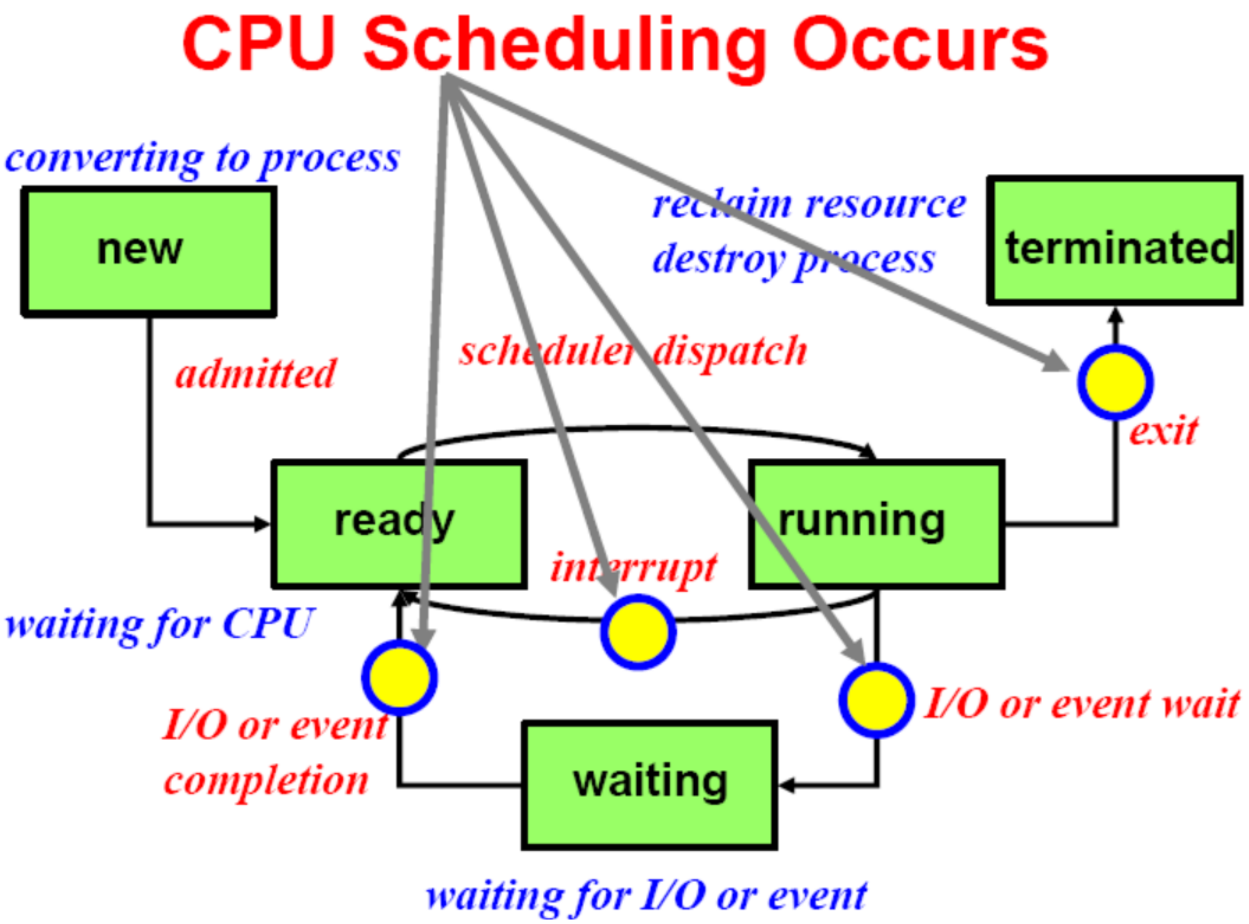
Lecture 4: CPU Scheduling

Associate Professor Dr. Sangsuree Vasupongayya
sangsuree.v@psu.ac.th
Department of Computer Engineering
Prince of Songkla University

Outline

- Basic concepts
- Scheduling criteria
- Scheduling algorithms
  - FCFS
  - SJF
  - Priority
  - Round Robin
- Multiple-processor scheduling

CPU Scheduler



- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates
- Scheduling under 1 and 4 is *nonpreemptive*
- All other scheduling is *preemptive*

Scheduling Criteria

CPU utilization – keep the CPU as busy as possible

Throughput – # of processes that complete their execution per time unit

Turnaround time – amount of time to execute a particular process

Waiting time – amount of time a process has been waiting in the ready queue

Response time – amount of time it takes from when a request was submitted until the first response is produced, **not** output (for time-sharing environment)

Scheduling Algorithms

- First-Come First-Served Scheduling
- Shortest-Job-First scheduling
- Priority Scheduling
- Round-Robin Scheduling

First-Come, First-Served (FCFS) Scheduling

Process	Burst Time
P_1	24
P_2	3
P_3	3

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

Waiting time

Average waiting time

Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst. Use these lengths to schedule the process with the shortest time
- Two schemes:
  - nonpreemptive – once CPU given to the process it cannot be preempted until completes its CPU burst
  - preemptive – if a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is know as the Shortest-Remaining-Time-First (SRTF)
- SJF is optimal – gives minimum average waiting time for a given set of processes

Example of Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (non-preemptive)

- Average waiting time

Example of Preemptive SJF

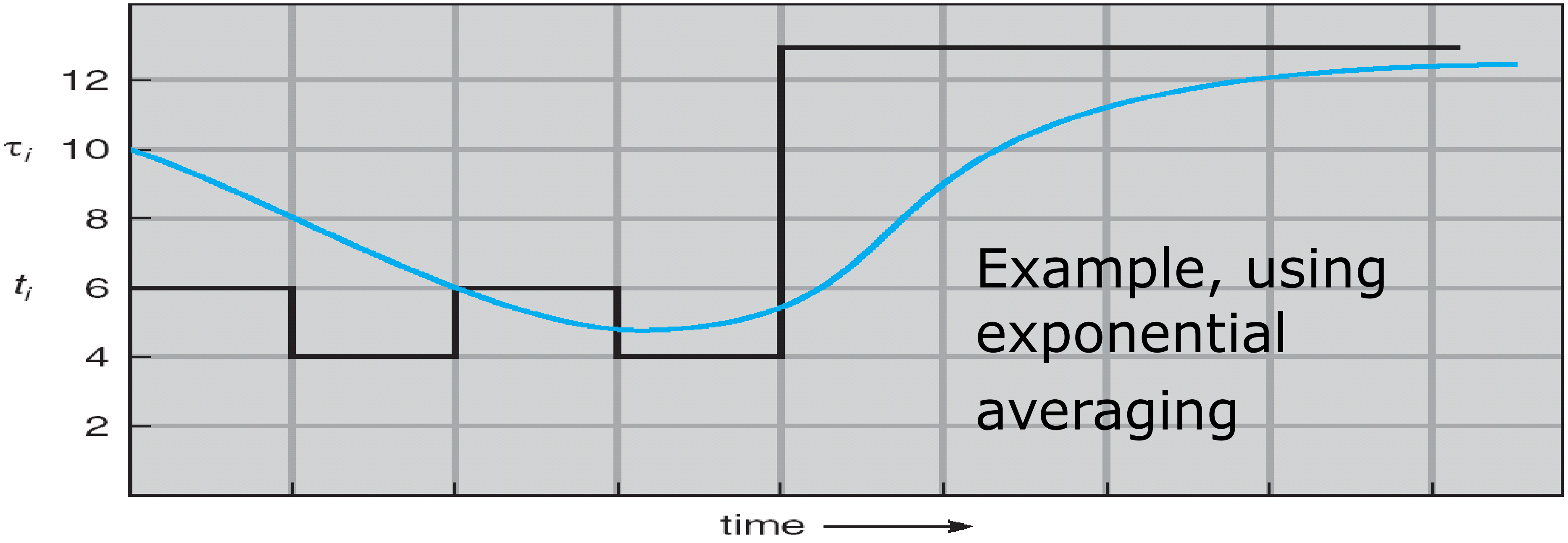
Process	Arrival Time	Burst Time
P_1	0	7
P_2	2	4
P_3	4	1
P_4	5	4

- SJF (preemptive)

- Average waiting time

Problem with SJF

- Determining Length of Next CPU Burst (Can only estimate the length)
- Usually the prediction is done by using the length of previous CPU bursts



CPU burst (t_i)		6	4	6	4	13	13	13	...
"guess" (τ_i)	10	8	6	6	5	9	11	12	...

Priority Scheduling

- A priority number (integer) is associated with each process
- The CPU is allocated to the process with the highest priority (smallest integer ≡ highest priority)
  - Preemptive
  - nonpreemptive
- SJF is a priority scheduling where priority is the predicted next CPU burst time
- Problem ≡ Starvation – low priority processes may never execute
- Solution ≡ Aging – as time progresses increase the priority of the process

Round Robin (RR)

- Each process gets a small unit of CPU time (*time quantum*), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are n processes in the ready queue and the time quantum is q, then each process gets $1/n$ of the CPU time in chunks of at most q time units at once. No process waits more than $(n-1)q$ time units.
- Performance
  - q large $\Rightarrow$ FIFO
  - q small $\Rightarrow$ q must be large with respect to context switch, otherwise overhead is too high

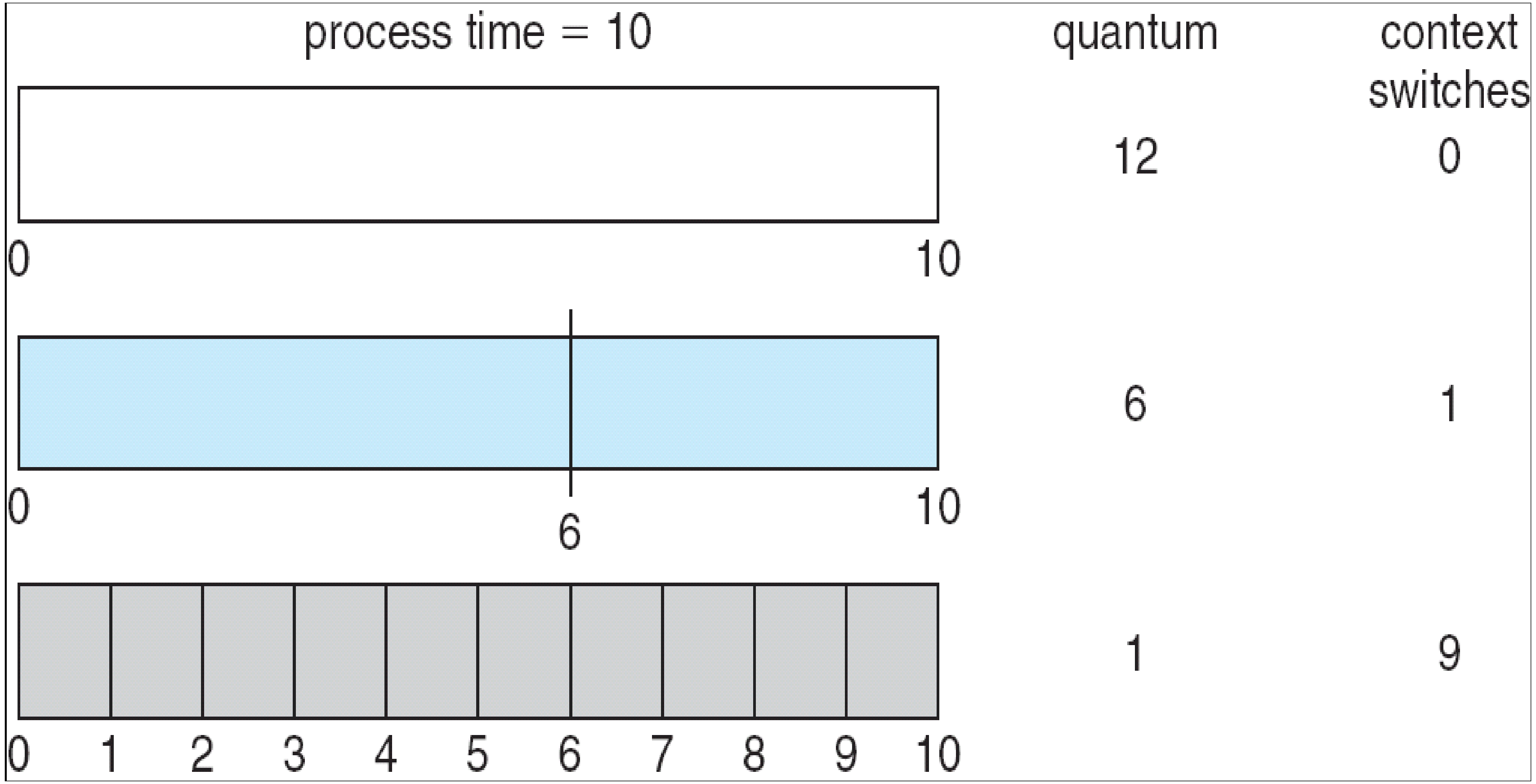
Example of RR with Time Quantum = 20

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

- The Gantt chart is:

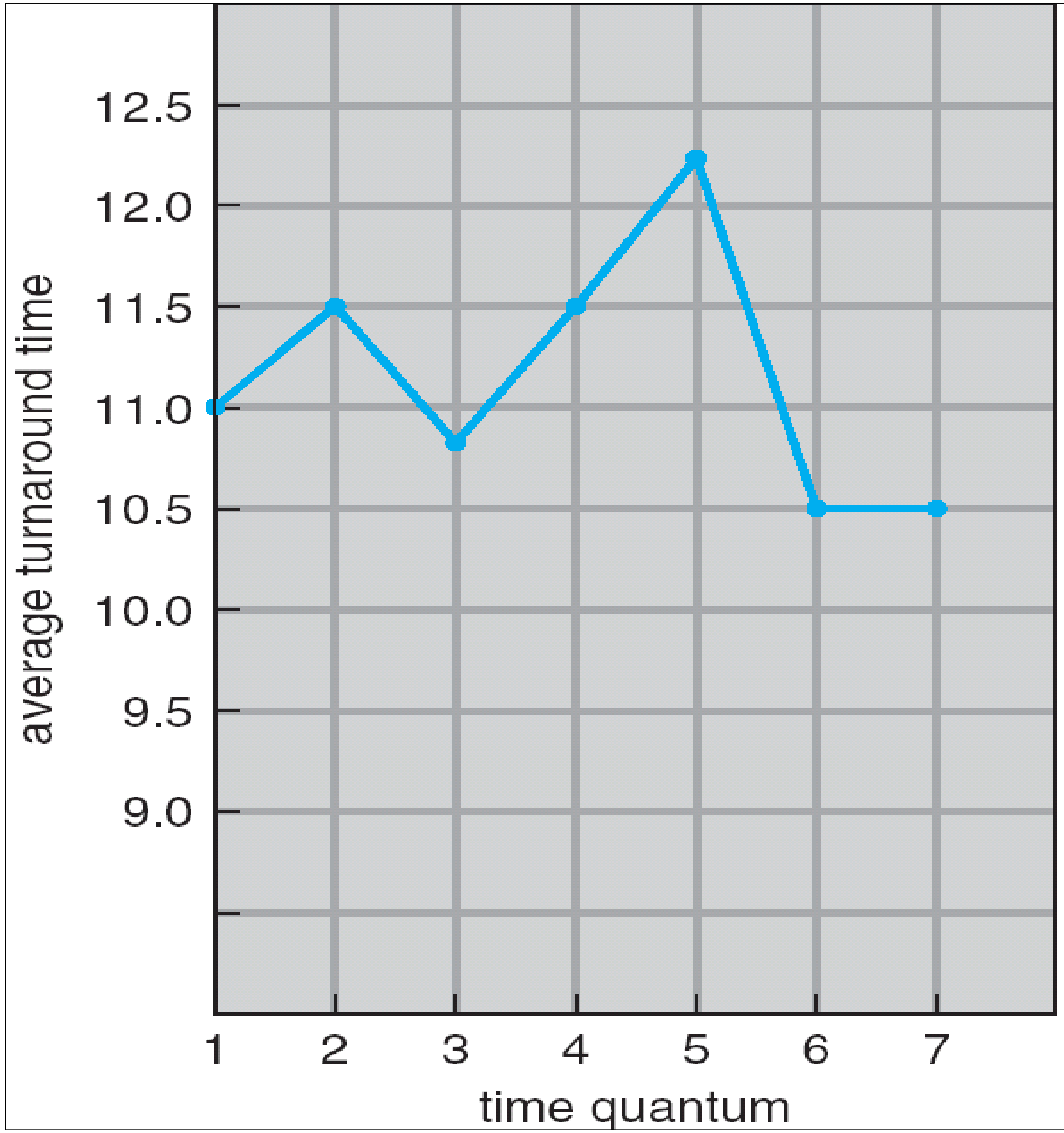
- Typically, higher average turnaround than SJF, but better *response*

Time Quantum and Context Switch Time



small time quantum → more context switches

Turnaround Time Varies With The Time Quantum



process	time
P_1	6
P_2	3
P_3	1
P_4	7

Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available
- Homogeneous processors within a multiprocessor (e.g., CPU-GPU)
- Approaches
  - Asymmetric multiprocessing – all scheduling decisions, I/O processing, and other system activities handled by a single processor. Thus, only one processor accesses the system data structures, alleviating the need for data sharing. Other processors execute only user code
  - Symmetric multiprocessing – each processor is self-scheduling