CPU Scheduling (4)

Dr. Jun Zheng

CSE325 Principles of Operating Systems

9/16/2019

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

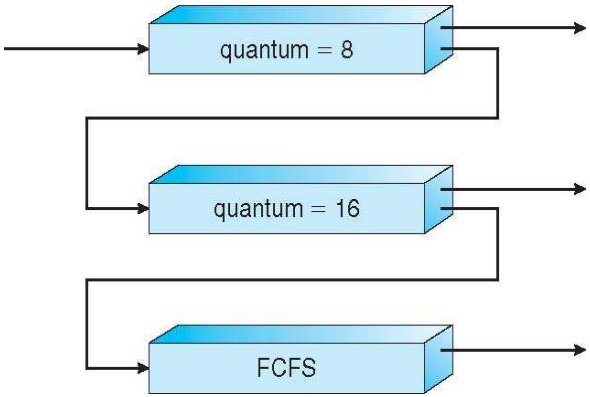
Example of Multilevel Feedback Queue

Three queues:

- $Q0$ – RR with time quantum 8 milliseconds
- $Q1$ – RR time quantum 16 milliseconds
- $Q2$ – FCFS

Scheduling

- A new job enters queue $Q0$ which is served FCFS
  - When it gains CPU, job receives 8 milliseconds
  - If it does not finish in 8 milliseconds, job is moved to queue $Q1$
- At $Q1$ job is again served FCFS and receives 16 additional milliseconds
  - If it still does not complete, it is preempted and moved to queue $Q2$



Question

- Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithms for each queue, the criteria used to move processes between queues, and so on. These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum).What (if any) relation holds between the following pairs of algorithm sets?
  - Priority and SJF
  - Multilevel feedback queues and FCFS
  - Priority and FCFS
  - RR and SJF

Answer

- The shortest job has the highest priority.
- The lowest level of MLFQ is FCFS.
- FCFS gives the highest priority to the job having been in existence the longest.
- None.

Multiple-Processor Scheduling

- CPU scheduling more complex when multiple CPUs are available
- **Homogeneous processors** within a multiprocessor
- **Asymmetric multiprocessing** – only one processor accesses the system data structures, alleviating the need for data sharing
- **Symmetric multiprocessing (SMP)** – each processor is self-scheduling, all processes in common ready queue, or each has its own private queue of ready processes
  - Currently, most common
- **Processor affinity** – process has affinity for processor on which it is currently running
  - **soft affinity**
  - **hard affinity**
  - Variations including **processor sets**

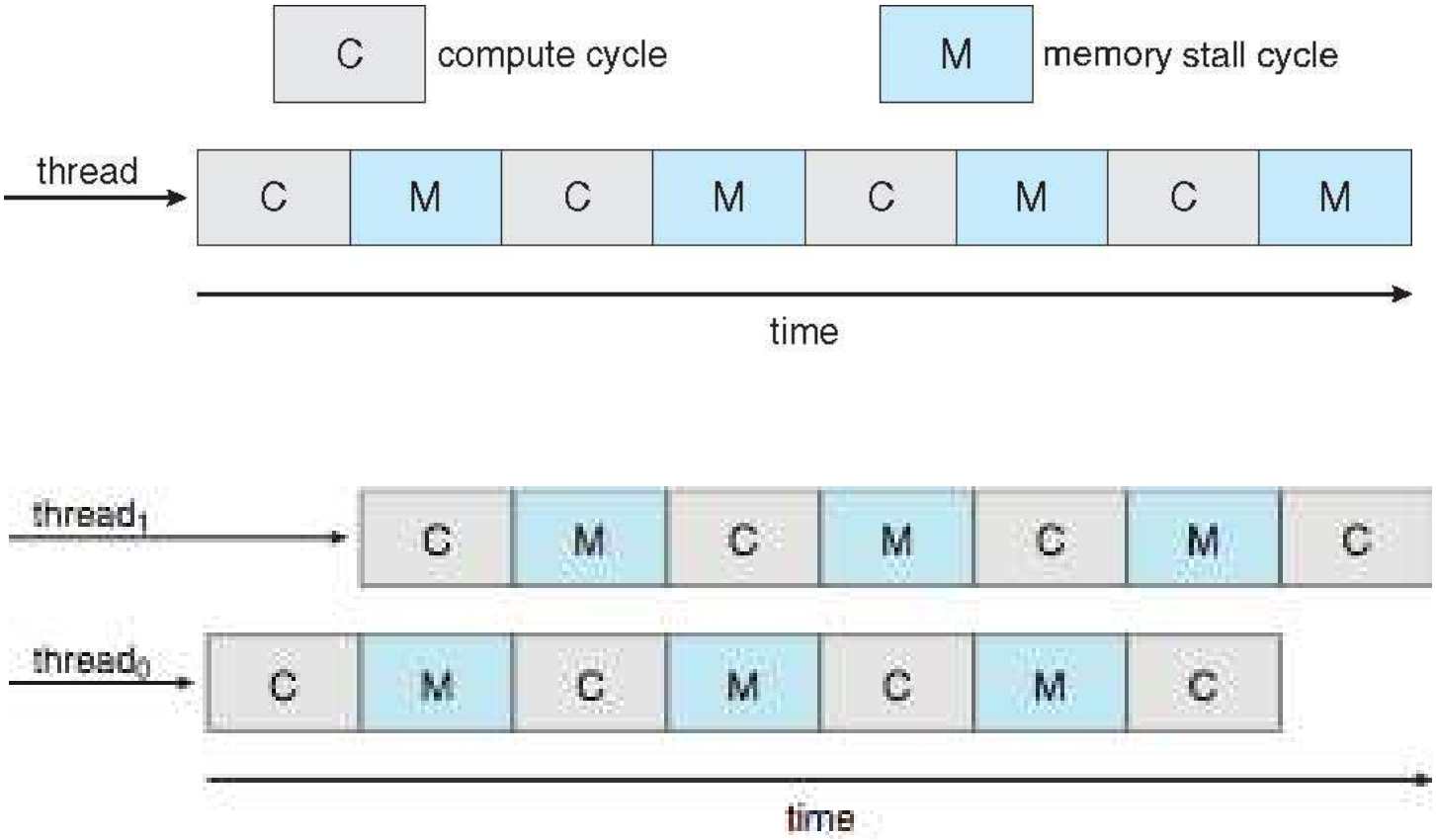
Multiple-Processor Scheduling – Load Balancing

- If SMP, need to keep all CPUs loaded for efficiency
- **Load balancing** attempts to keep workload evenly distributed
- **Push migration** – periodic task checks load on each processor, and if found pushes task from overloaded CPU to other CPUs
- **Pull migration** – idle processors pulls waiting task from busy processor

Multicore Processors

- Recent trend to place multiple processor cores on same physical chip
- Faster and consumes less power
- Multiple threads per core also growing
  - Takes advantage of memory stall to make progress on another thread while memory retrieve happens

Multithreaded Multicore System



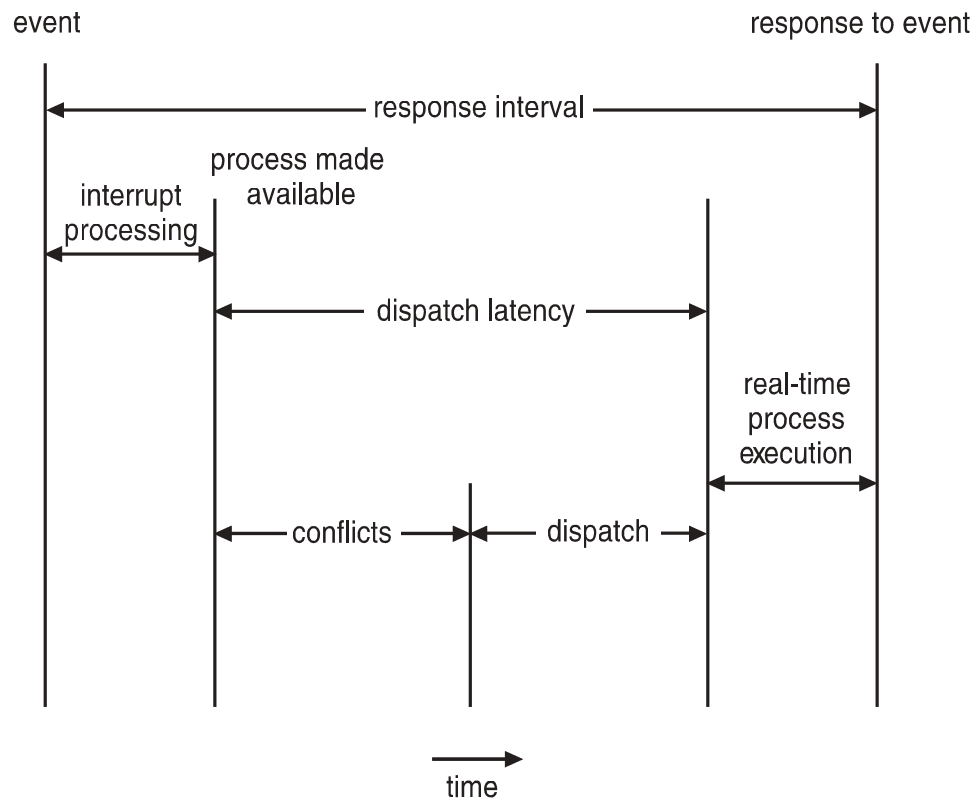
NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Real-Time CPU Scheduling

- Can present obvious challenges
- **Soft real-time systems** – no guarantee as to when critical real-time process will be scheduled
- **Hard real-time systems** – task must be serviced by its deadline
- Two types of latencies affect performance
  - Interrupt latency – time from arrival of interrupt to start of routine that services interrupt
  - Dispatch latency – time for schedule to take current process off CPU and switch to another

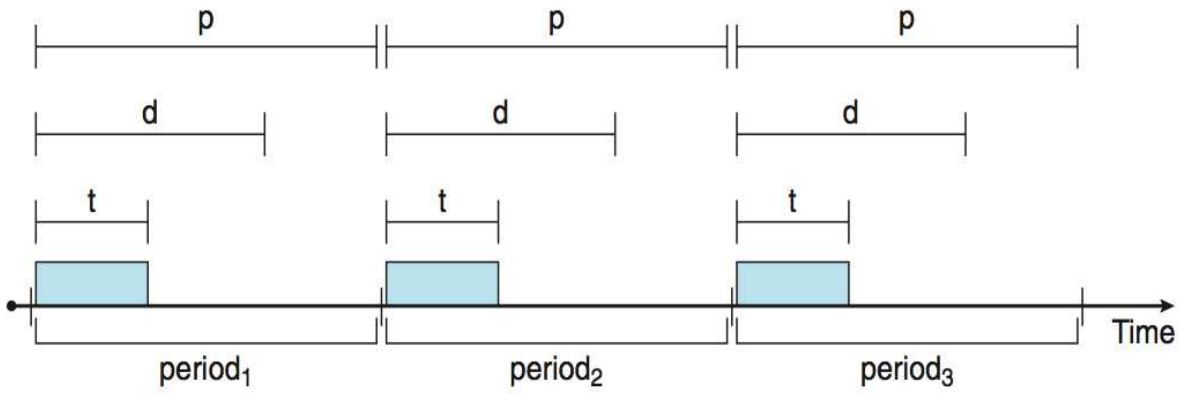
Real-Time CPU Scheduling (Cont.)

- Conflict phase of dispatch latency:
  - Preemption of any process running in kernel mode
  - Release by low-priority process of resources needed by high-priority processes



Priority-based Scheduling

- For real-time scheduling, scheduler must support preemptive, priority-based scheduling
- But only guarantees soft real-time
- For hard real-time must also provide ability to meet deadlines
- Processes have new characteristics: **periodic** - processes require CPU at constant intervals
- Has processing time *t*, deadline *d*, period *p*
- $0 \leq t \leq d \leq p$
- **Rate** of periodic task is $1/p$



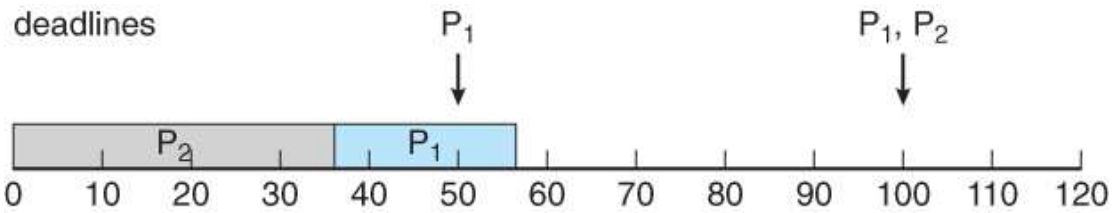
NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Rate Monotonic Scheduling

- A priority is assigned based on the inverse of its period
- Shorter periods = higher priority;
- Longer periods = lower priority

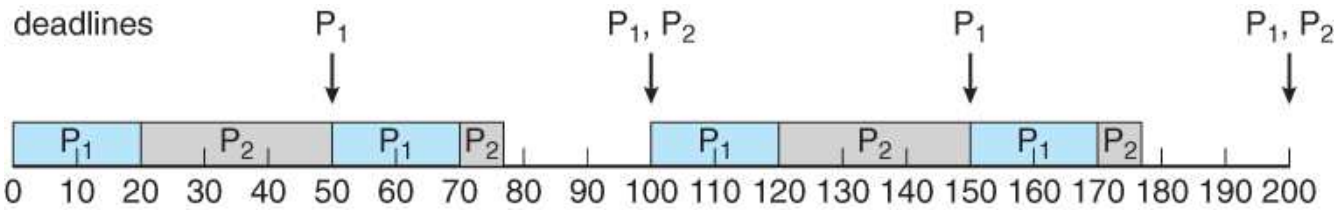
Example

- Suppose that process P1 has a period of 50, an execution time of 20, and a deadline that matches its period (50).
- Similarly suppose that process P2 has period 100, execution time of 35, and deadline of 100.
- The total CPU utilization time is 20 / 50 = 0.4 for P1, and 35 / 100 = 0.35 for P2, or 0.75 (75%) overall.
- However if P2 is allowed to go first (FCFS), then P1 cannot complete before its deadline:



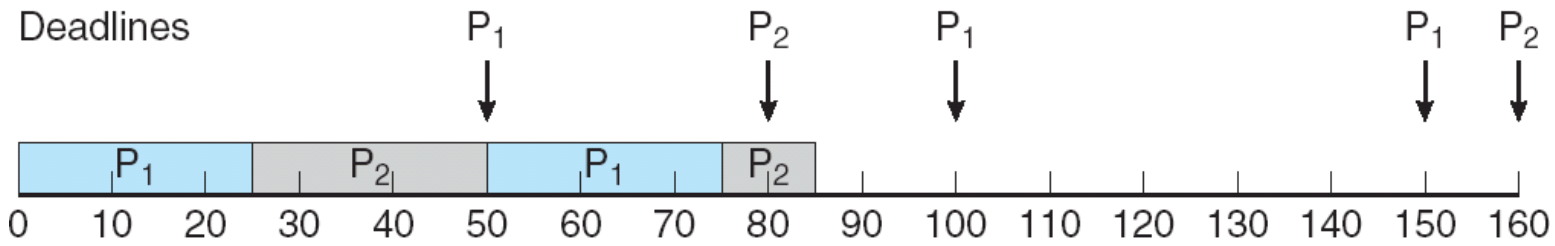
Example

- if P1 is given higher priority, it gets to go first, and P2 starts after P1 completes its burst.
- At time 50 when the next period for P1 starts, P2 has only completed 30 of its 35 needed time units, but it gets pre-empted by P1.
- At time 70, P1 completes its task for its second period, and the P2 is allowed to complete its last 5 time units.
- Overall both processes complete at time 75, and the cpu is then idle for 25 time units, before the process repeats.



NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Missed Deadlines with Rate Monotonic Scheduling

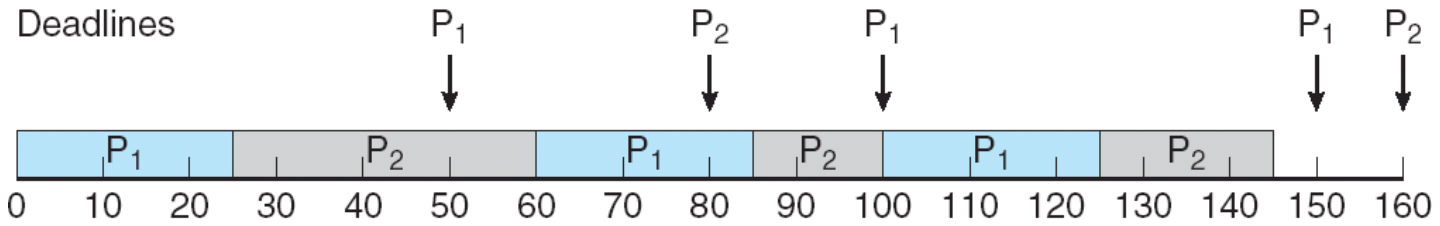


P1 =50, T1 = 25, P2 = 80, T2 = 35, and the deadlines match the periods.

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Earliest Deadline First Scheduling (EDF)

- Priorities are assigned according to deadlines:
  - the earlier the deadline, the higher the priority;
  - the later the deadline, the lower the priority
- Previous example:



P1 =50, T1 = 25, P2 = 80, T2 = 35, and the deadlines match the periods.

NEW MEXICO TECH
SCIENCE · ENGINEERING · RESEARCH · UNIVERSITY

Proportional Share Scheduling

- T shares are allocated among all processes in the system
- An application receives N shares where $N < T$
- This ensures each application will receive $\boldsymbol{N / T}$ of the total processor time
- Must work in conjunction with an admission-control policy to guarantee that an application receives its allocated share of time.