Problems I

Scheduling of processes

Scheduling review

- The scheduler is the kernel service that manages the CPU:
  - Allocation of the core(s);
  - Load balancing;
  - Monitors the load and informs other kernel services such as the power saving one.
  - Fixes conflicting situations, e.g., by priority inversion.

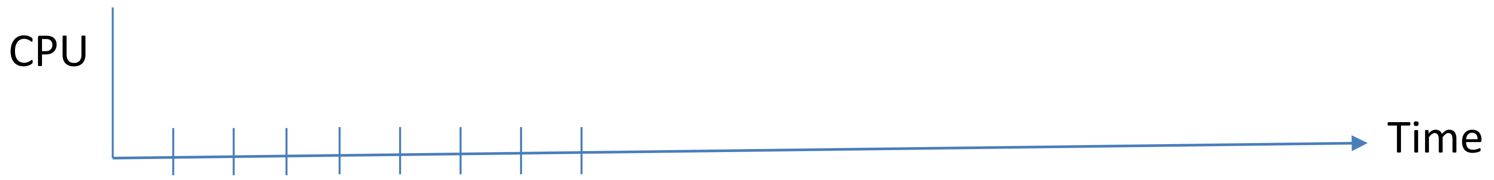
- There are different algorithms implemented by the scheduler, depending on the applications area of the computing system:
  - Earliest deadline first;
  - Multiple level feedback queues;
  - Group scheduling;
  - Domain scheduling.

1. Real-time scheduling

In real-time systems, execution of processes is dictated by their deadlines: the next process to run is that with the earliest deadline.

Let’s consider four processes, A, B, C and D, running in an embedded system. The deadlines are 200 for A, 400 for B, 500 for C and 700 for D. The time quantum is 100 and all processes complete within the allocated quantum – all figures correspond to time units.

If we consider the scheduling strategy “earliest deadline first”, follow the execution of these processes along the time axis starting with time = 0.



2. Multilevel feedback queues

A computer system is using 8 multilevel feedback queues for scheduling user processes (numbered from 0, highest priority, to 7, lowest priority). Consider three processes, A, B and C that start on level 2, 4 and 5 respectively and have the execution time 0.5 s, 0.9 s and 3 s respectively. If the time quantum, denoted by q, for level 0 is 10 ms, determine the time quantum for each level. Show the execution of processes A, B and C on the time axis and determine when and from which level they exit the system. We assume there is no I/O operation. The time quantum for each level i is determined by the equation $t = 2^i q$.

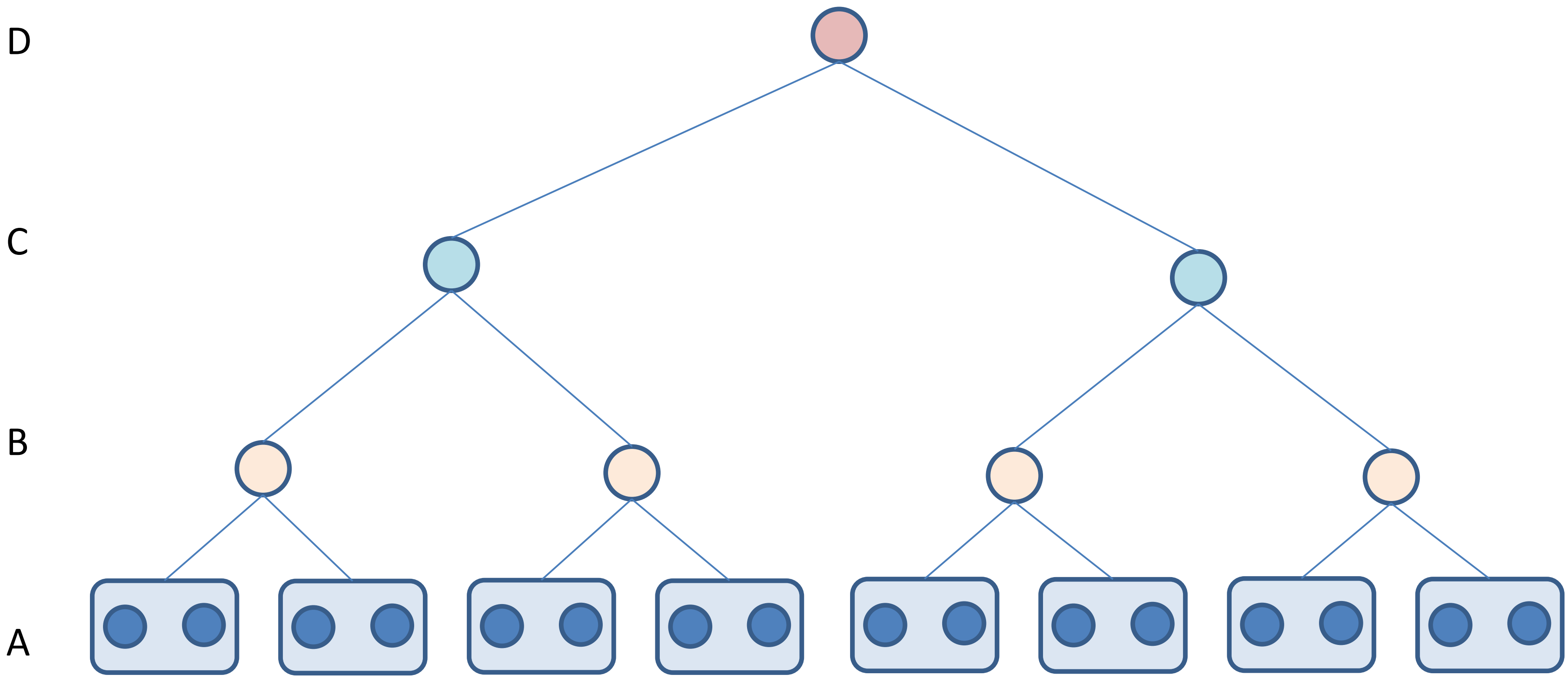
What is the impact of I/O operations on the priority of processes A, B and C ? As an example, consider that process B will run for 480 ms after which it will start disk operations that will take 2 s. Follow B’s execution and determine its exit queue.

3. Multi-core systems

1. Consider a 16-core system (see next slide). Define the concept of domain scheduling and illustrate it using a diagram of the 16-core homogeneous system. Discuss how different policies associated with domains can reduce the execution time of processes and save energy. Explain your choices in terms of policies.

2. What would change if the system has a LITTLE.big configuration of eight cores?

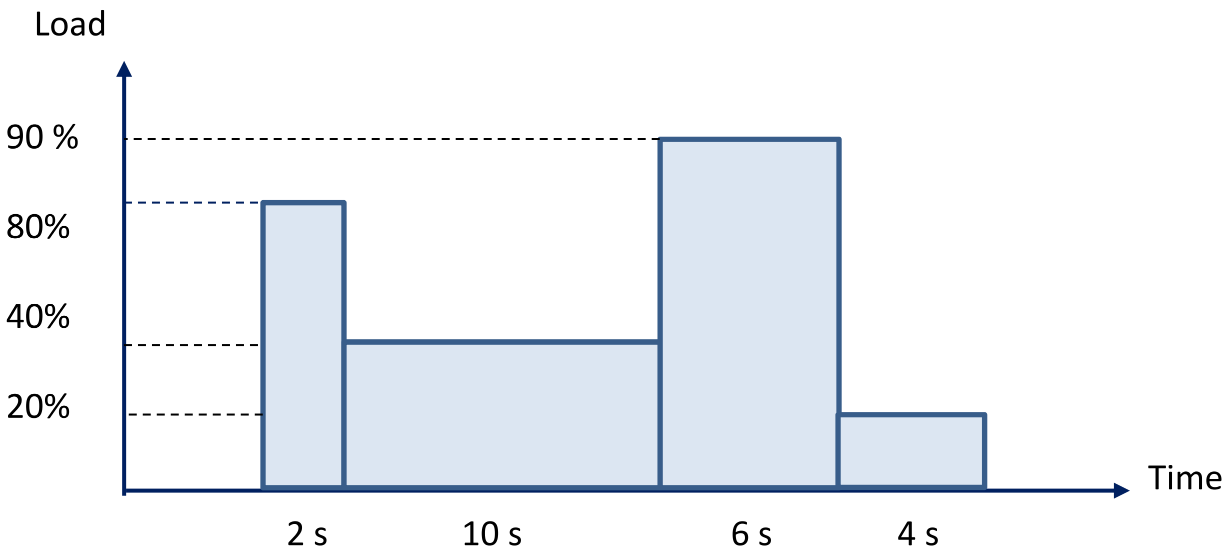
Cores domain hierarchy



4. big.Little at work

- A six-core system has four power saving (little) cores and two high performance (big) cores. The execution rate of a little core is 1MIPS (million instructions/sec) and that of a big core is 1.6 MIPS. A little core consumes 1 W and a big core consumes 1.5 W. The figure on the next slide shows a diagram of the load evolution with sustained values for short periods of time. The load is defined in terms of four million of instructions: e.g., a sustained load of 80% during 2 sec means .8 x 4 mil instructions x 2 sec = 3,200,000 instructions x 2 sec, therefore a total load of 6.4 million instructions to be executed during that interval of time.

1. What is the energy consumed by the CPU if all cores are powered all the time?
2. Propose a scheduling strategy that will save energy. Explain how the cores are used. What is the new value of the energy consumed?


Load
90 %
80%
40%
20%
Time
2 s
10 s
6 s
4 s