Scheduling and Deadlines

Learning Outcomes

- Understand the role of the scheduler, and how its behaviour influences the performance of the system.
- Know the difference between I/O-bound and CPU-bound tasks, and how they relate to scheduling.

Goals of Scheduling Algorithms

- Interactive Algorithms
  - Minimise *response time* (*latency*)
    - Response time is the time difference between issuing a command and getting the result
      - E.g selecting a menu, and getting the result of that selection
    - Response time is important to the user’s perception of the performance of the system.
  - Performance
    - Prioritise I/O latency over CPU tasks

Goals of Scheduling Algorithms

- Real-time Algorithms
  - Must meet deadlines
    - Each job/task has a deadline.
    - A missed deadline can result in data loss or catastrophic failure
      - Aircraft control system missed deadline to apply brakes

Real-time Scheduling

Real Time Scheduling

- Correctness of the system may depend not only on the logical result of the computation but also ***on the time when*** these results are produced, e.g.
  - Tasks attempt to control events or to react to events that take place in the outside world
  - These external events occur in *real time* and processing must be able to keep up
  - Processing must happen in a timely fashion,
    - neither too late, nor too early

Typical Real Time Systems

  - Control of laboratory experiments
  - Robotics
  - (Air) Traffic control
  - Controlling Cars / Trains/ Planes
  - Telecommunications
  - Medical support (Remote Surgery, Emergency room)
  - Multi-Media
- Remark: Some applications may have only **soft-real time** requirements, but some have really **hard real-time** requirements

Hard-Real Time Systems

- Requirements:
  - **Must *always* meet all deadlines** (time guarantees)
  - You have to guarantee that in any situation these applications are done in time, otherwise dangerous things may happen

Examples:

1. If the landing of a fly-by-wire jet cannot react to sudden side-winds within some milliseconds, an accident might occur.
2. An airbag system or the ABS has to react within milliseconds

Soft-Real Time Systems

Requirements:

Must *mostly* meet all deadlines, e.g. 99.9% of cases

Examples:

1. Multi-media: 100 frames per day might be dropped (late)
2. Car navigation: 5 late announcements per week are acceptable
3. Washing machine: washing 10 sec over time might occur once in 10 runs, 50 sec once in 100 runs.

Hard-Real Time System OSs

- A typical "hard" real-time system has some kind of *responsibility* to operate on time.
  - In a "soft" real-time system this can be a trade-off, e.g. cost vs performance
- We are not allowed to trade safety
  - Someone has a *professional* responsibility give a *guarantee* that deadlines are kept.
  - This typically requires a specialised OS
    - FreeRTOS, QNX, Integrity

Predictability, not Speed

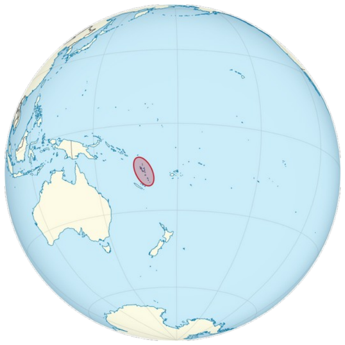
- Real time systems are NOT necessarily fast.
- Real time systems can be slow, as long as they are predictably slow.
  - What matters is that they meet their deadlines, not how fast they run.
- Real-time OSs and real-time hardware may be chosen in an unusual way.
  - Very simple caches and pipelines, for instance.

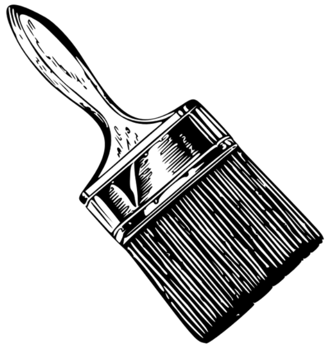
Unusual Soft Real-Time Systems

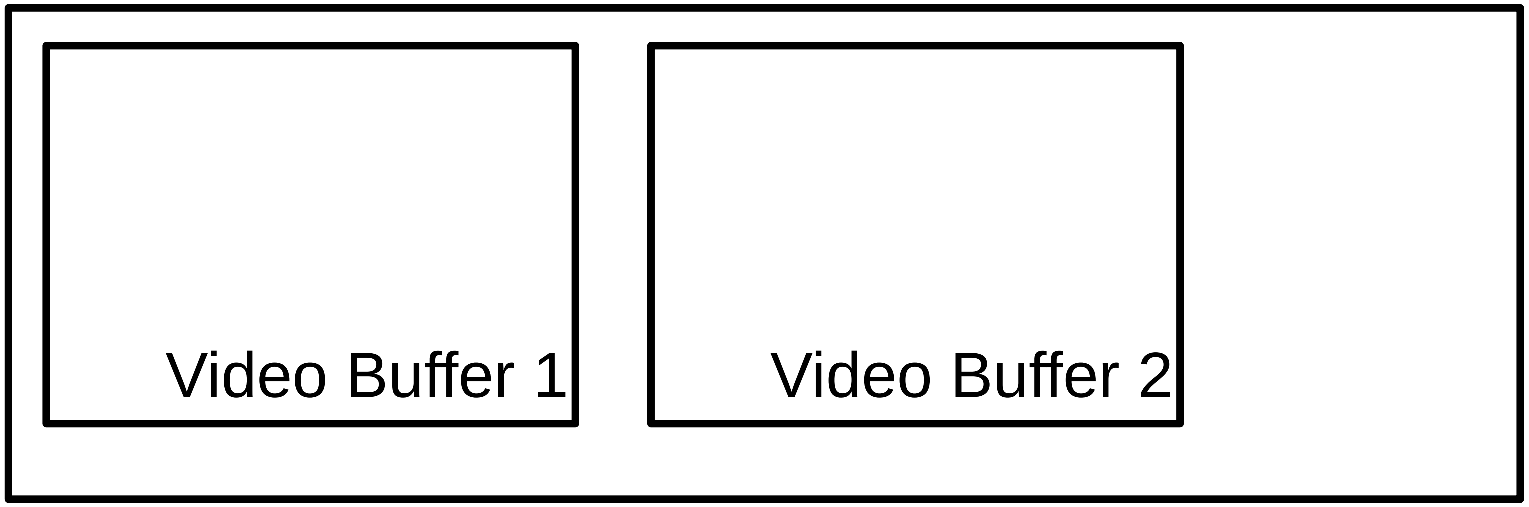
- An exception that proves the rule:
  - “Safety” critical systems that are not crewed.
  - Failure of the system leads to loss of capital, not loss of life.
  - Now “safety” can be involved in a trade-off.

- Recent space exploration.
- Some mining systems.

Another Example







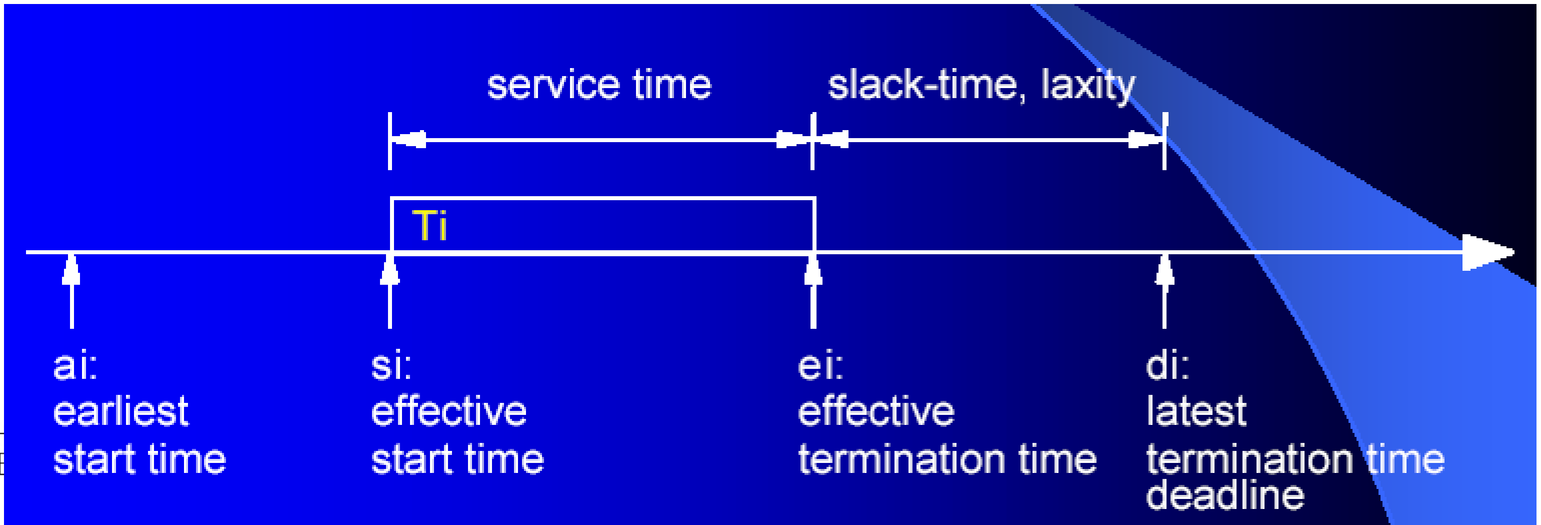
Explanation by Example

- The previous slide is about a graphical simulation
- “Deadline” is for new output to be sent to the hardware, which happens at a fixed frequency.

- Other hardware has similar properties.
  - Output control settings need to be sent at particular times.
  - Creates “deadline” for control decisions.

Properties of Real-Time Tasks

- To schedule a real time task, its properties must be known *a priori*
- The most relevant properties are
  - Arrival time (or release time) a_i
  - Maximum execution time (service time)
  - Deadline d_i



Real-time scheduling approaches

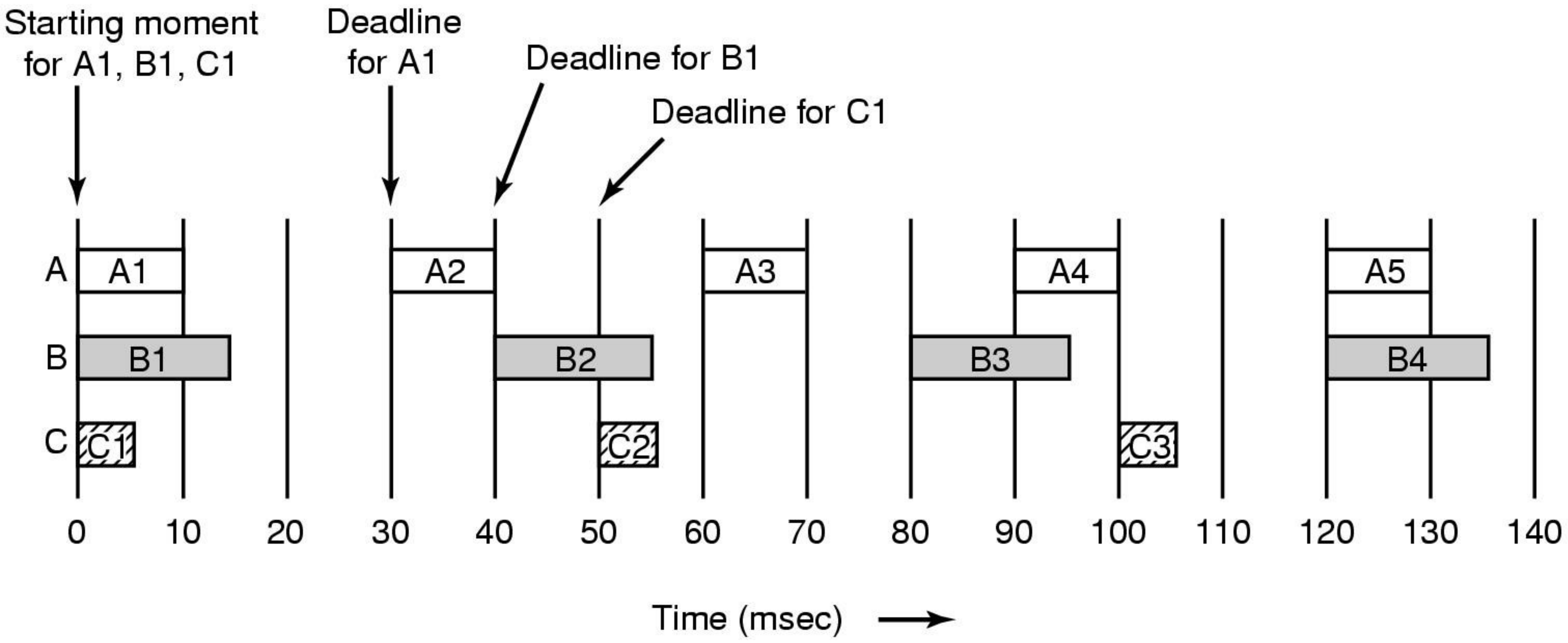
- Static table-driven scheduling
  - Given a set of tasks and their properties, a schedule (table) is precomputed offline.
    - Used for periodic task set
    - Requires entire schedule to be recomputed if we need to change the task set
- Static priority-driven scheduling
  - Given a set of tasks and their properties, each task is assigned a fixed priority
  - A preemptive priority-driven scheduler used in conjunction with the assigned priorities
    - Used for periodic task sets

Two Typical Real-time Scheduling Algorithms

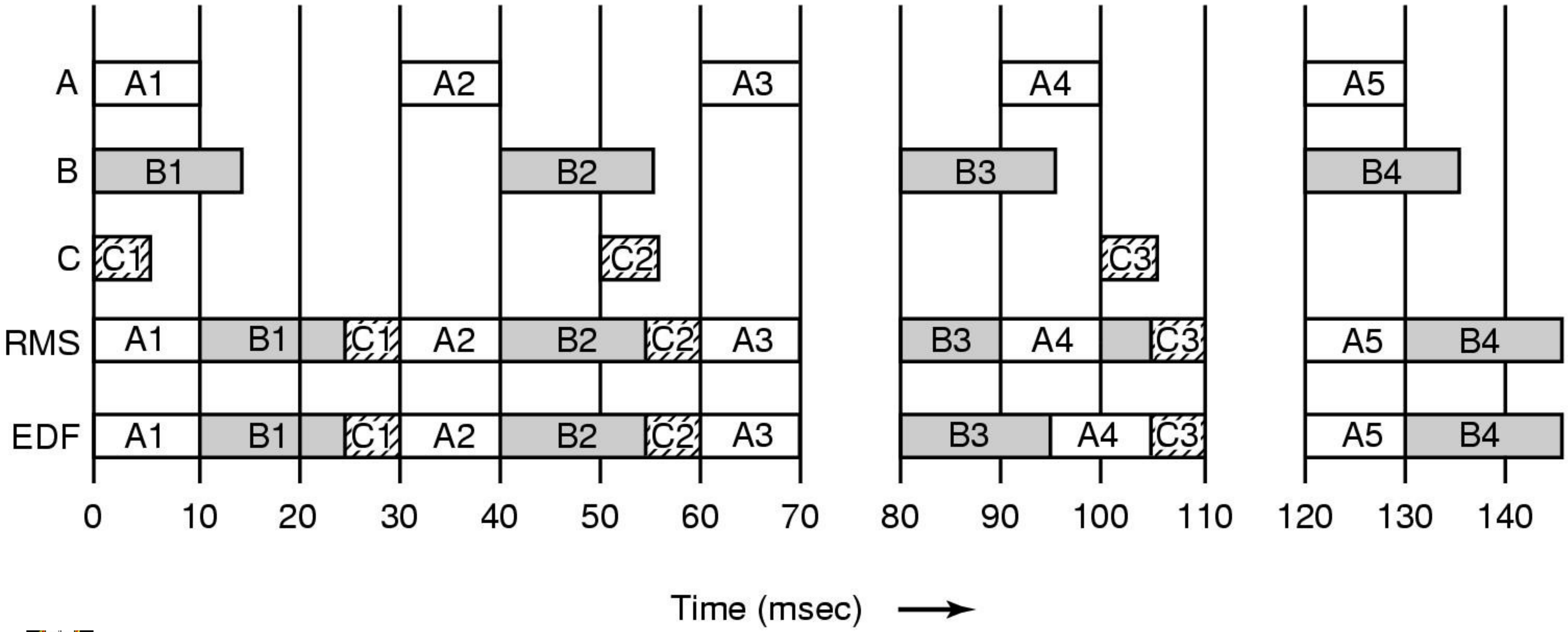
- Rate Monotonic Scheduling
  - Static Priority priority-driven scheduling
  - Priorities are usually assigned based on the period of each task
    - The shorter the period, the higher the priority
- Earliest Deadline First Scheduling
  - The task with the earliest deadline is chosen next

A Scheduling Example

- Three periodic Tasks



Two Schedules: RMS and EDF

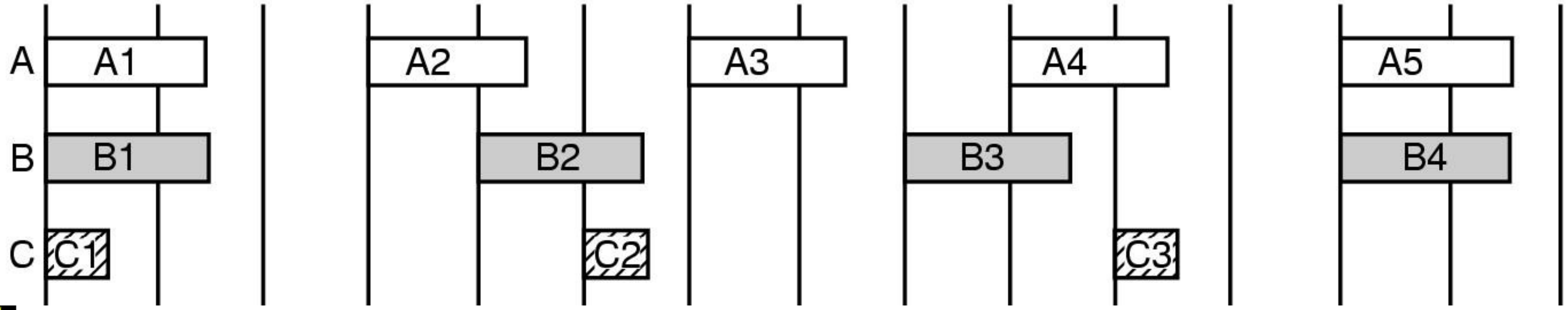


THE UNIVERSITY OF NEW SOUTH WALES

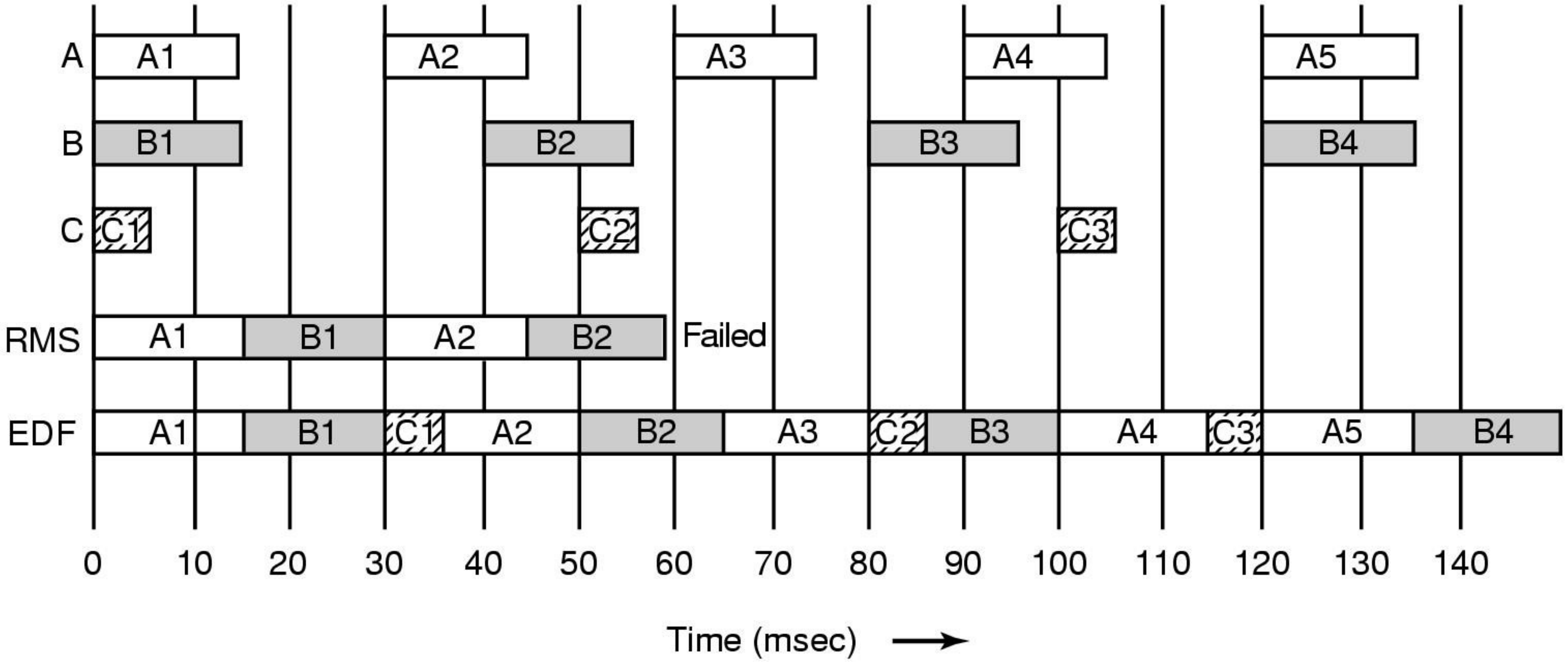
Let’s Modify the Example Slightly

- Increase A’s CPU requirement to 15 msec
- The system is still within CPU constraints

$$\frac{15}{30}+\frac{15}{40}+\frac{5}{50}=0.975$$



RMS and EDF



A
B
C
RMS
EDF
A1
A2
A3
A4
A5
B1
B2
B3
B4
C1
C2
C3
A1 B1 A2 B2
Failed
A1 B1 C1 A2 B2 A3 C2 B3 A4 C3 A5 B4
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140
Time (msec)

RMS failed, why?

- It has been proven that RMS is only guaranteed to work if the CPU utilisation is not too high
  - For three tasks, CPU utilisation must be less than 0.780
    - We were lucky with our original example

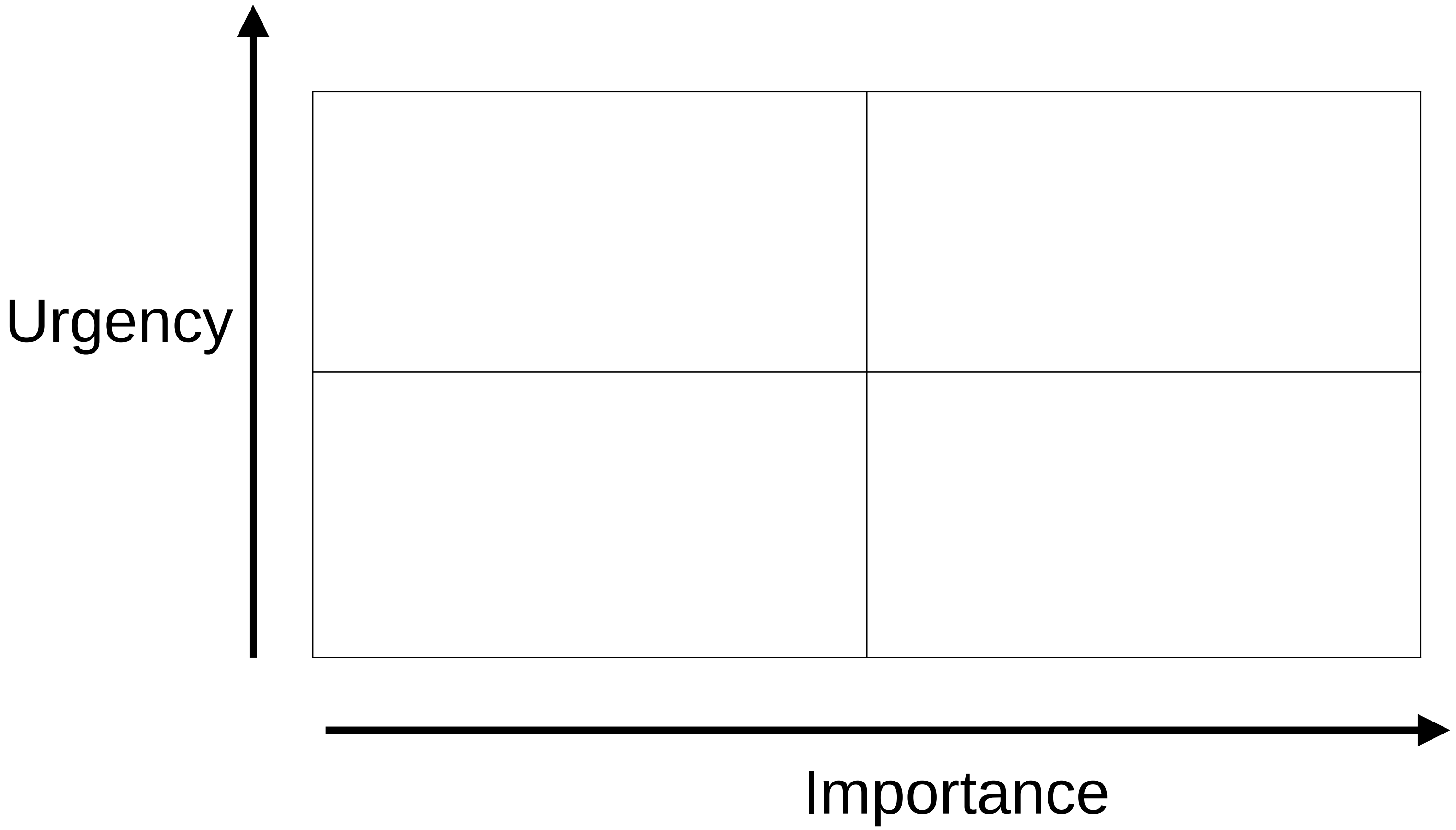
$$\sum_{i=1}^{m} \frac{C_i}{P_i} \le m(2^{1/m} - 1)$$

EDF

- EDF always works for any schedulable set of tasks, i.e. up to 100% CPU utilisation

- Massive caveat: what happens if it is not possible to complete everything by deadline?

Eisenhower Matrix



Today (Scheduling)

- A different kind of OS.
- Deadlines and deadline management.
- Earliest-deadline vs priority-based scheduling.