Real-time Scheduling

Tanenbaum

Section 2.5, Section 7.4.2-7.4.4

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

Real Time System (RTS)

- RTS accepts an activity *A* and guarantees its requested (timely) behaviour *B* if and only if
  - RTS finds a *schedule*
    - that includes all already accepted activities *Ai* and the new activity *A*,
    - that guarantees all requested timely behaviour *Bi* and *B*, and
    - that can be enforced by the RTS.
- Otherwise, RT system rejects the new activity *A.*

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:

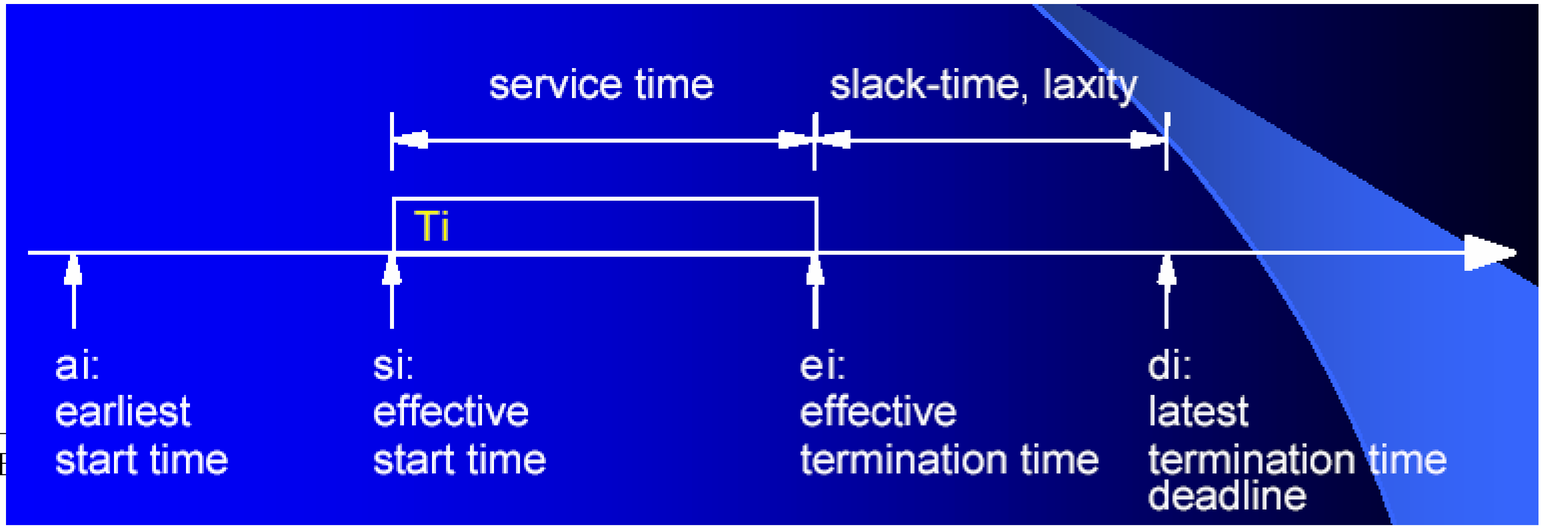
Must *mostly* meet all deadlines, e.g. 99.9% of cases

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.

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



Categories of Real time tasks

- Periodic
  - Each task is repeated at a regular interval
  - Max execution time is the same each period
  - Arrival time is usually the start of the period
  - Deadline is usually the end
- Aperiodic (sporadic)
  - Each task can arrive at any time

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

Real-time scheduling approaches

- Dynamic scheduling
  - Task arrives prior to execution
  - The scheduler determines whether the new task can be *admitted*
    - Can all other admitted tasks and the new task meet their deadlines?
      - If no, reject the new task
  - Can handle both *periodic* and *aperiodic* tasks

Scheduling in Real-Time Systems

- We will only consider periodic systems

Schedulable real-time system

- Given
  - *m* periodic events
  - event *i* occurs within period P_i and requires C_i seconds
- Then the load can only be handled if

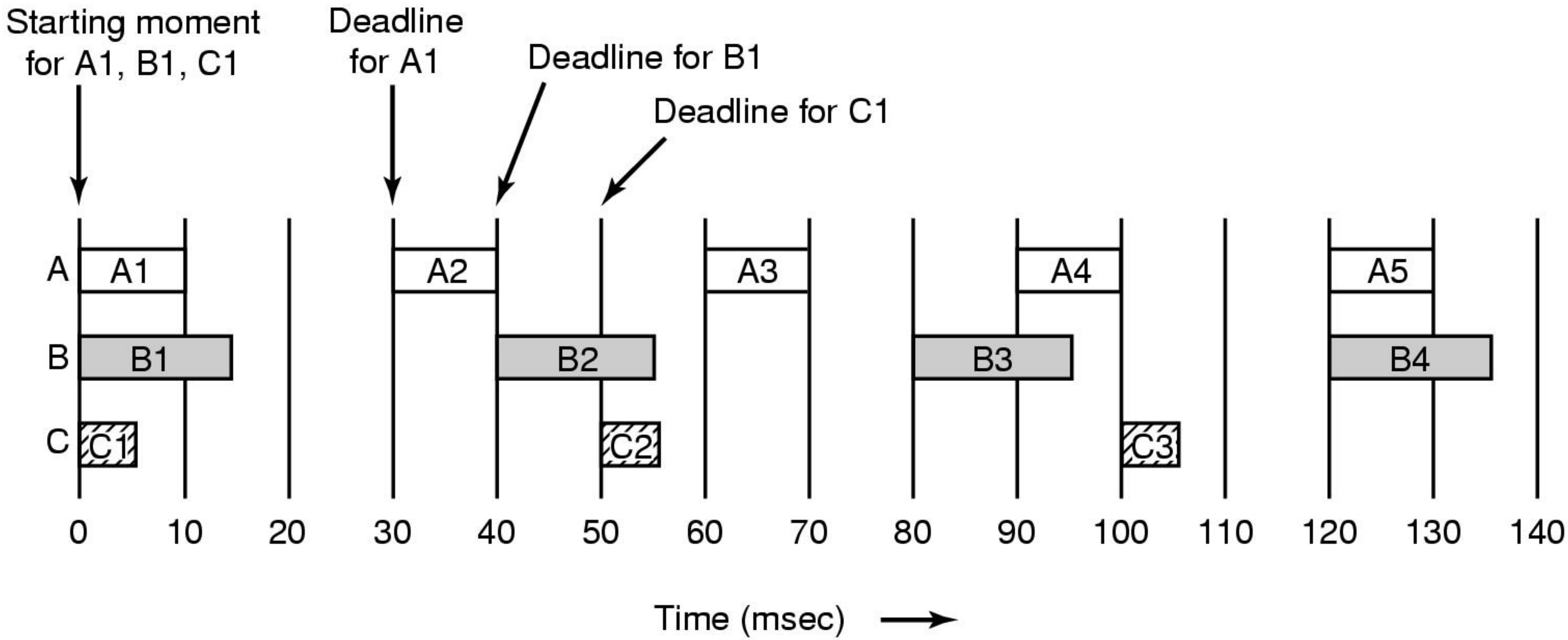
$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

Two Typical Real-time Scheduling Algorithms

- Rate Monotonic Scheduling
  - Static Priority priority-driven scheduling
  - Priorities are 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



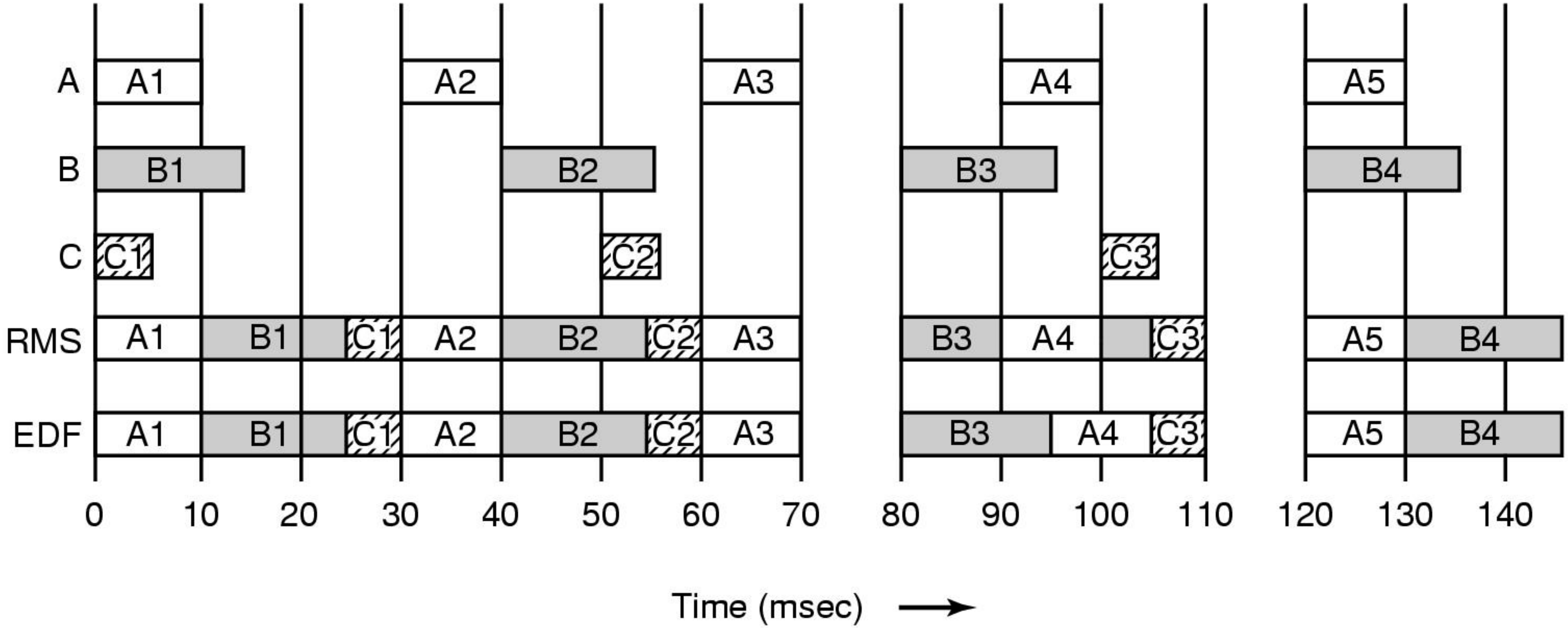
Is the Example Schedulable

$$\sum_{i=1}^{m} \frac{C_i}{P_i} \leq 1$$

$$\frac{10}{30} + \frac{15}{40} + \frac{5}{50} = 0.808$$

- YES

Two Schedules: RMS and EDF

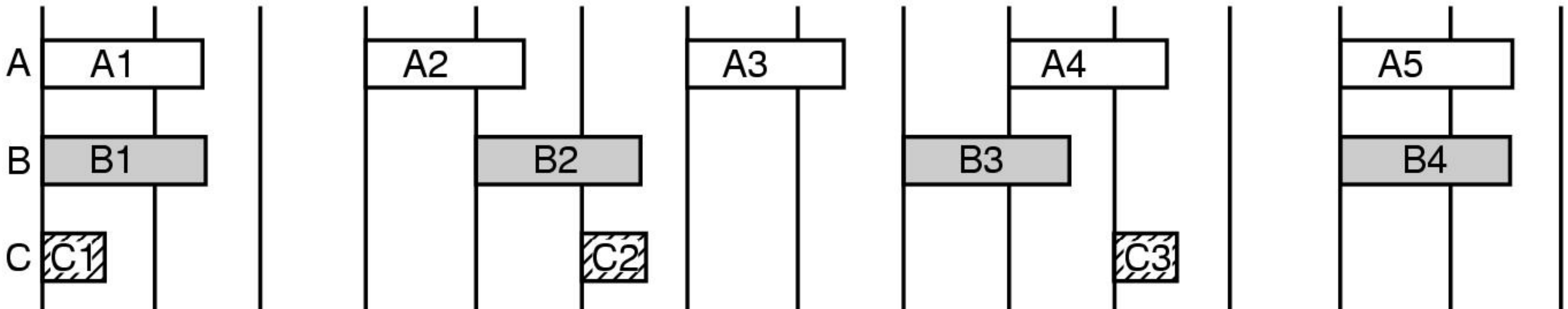


THE UNIVERSITY OF
NEW SOUTH WALES

Let's Modify the Example Slightly

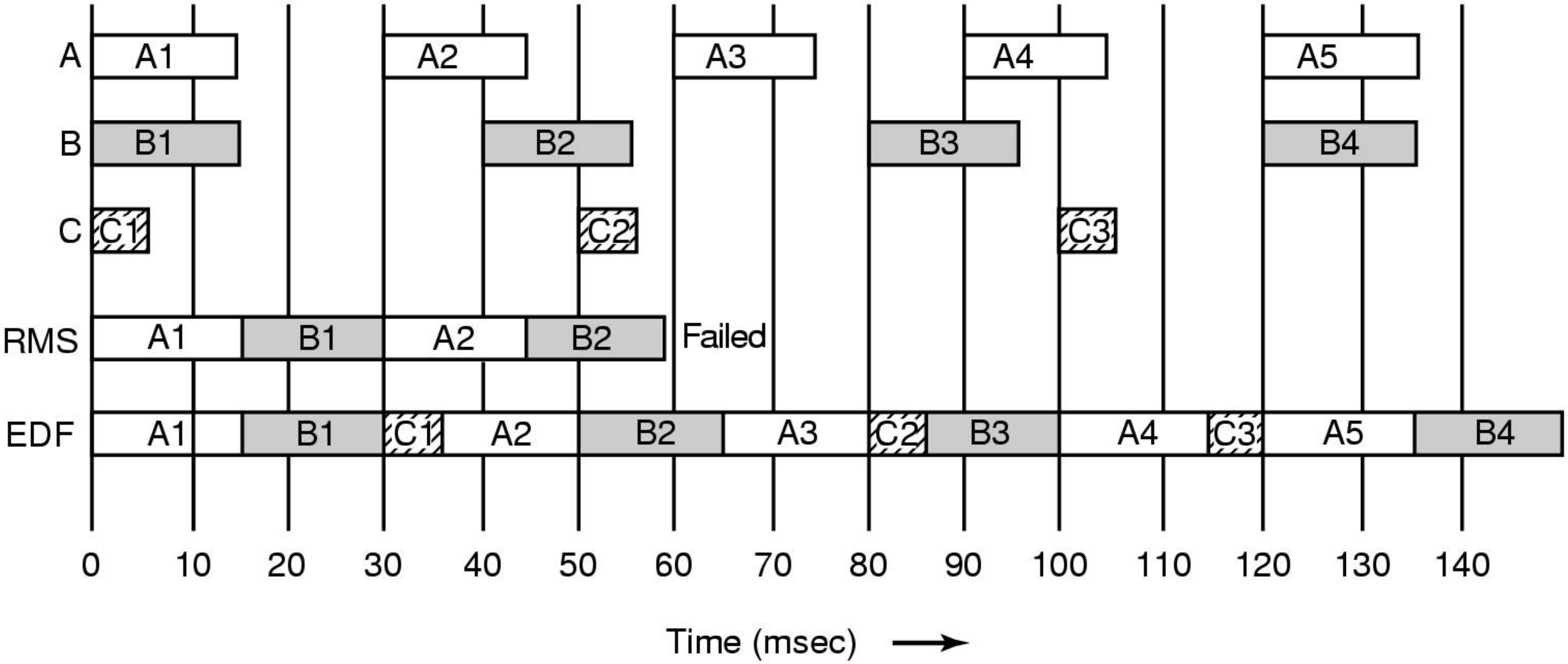
- Increase A's CPU requirement to 15 msec
- The system is still schedulable

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



THE UNIVERSITY OF NEW SOUTH WALES

RMS and EDF


A
B
C
A1
A2
A3
A4
A5
B1
B2
B3
B4
C1
C2
C3
RMS
A1
B1
A2
B2
Failed
EDF
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

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

EDF

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

- Summary
  - If CPU utilisation is low
    - Can use RMS which is simple and easy to implement
  - If CPU utilisation is high
    - Must use EDF