Real Time Systems and Control Applications

Contents

Priority Inversion

Review

Among CE, RM, DM, EDF, which are static priority schedulers?

Which is optimal scheduler for real-time applications? What is the drawback of it?

One of the Assumptions of RM, DM, EDF is that the tasks are preemptive! What happens if it is not always true? E.g., the resources may not always be available to higher priority tasks due to resource contention.

Critical Sections

- When two or more processes are competing to use the same resource - race condition - conflict
- Resources can be used by one task at a time - mutual exclusion
- Use of a resource cannot be interrupted - serially reusable
- Code that interacts with serially reusable resources - critical section
- We can avoid race conditions by making sure that no two processes/threads enter their **critical sections** at the same time.

Mutex: at most one process can hold it.

- Locking thread gets ownership of the mutex when first locked
- Only an owner can release a mutex

```
Process 1                                   Process2
   .                                           .
   .                                           .
   .                                           .
Mutex_Lock(s)                             .
critical section                            Mutex_Lock(s)
Mutex_Unlock(s)                             critical section
   .                                        Mutex_Unlock(s)
   .                                           .
   .                                           .
```

Deadlocks

```
Task A
      .
      .
Mutex_Lock(S)
use resource 1
      .
      .
Mutex_Lock(R)
stuck here
use resource 2
      .
      .
Mutex_Unlock(R)
Mutex_Unlock(S)
```

```
Task B
    .
    .
Mutex_Lock(R)
use resource 2
    .
    .
Mutex_Lock(S)
stuck here
use resource 1
    .
    .
 Mutex_Unlock(S)
 Mutex_Unlock(R)
```

Resource Access Control

- One processor
- n serially reuseable resources R1;R2; …; Rn, typically used by processes in a mutually exclusive manner without preemption
- A resource once allocated to a job cannot be used by another job until the previous job frees it
- Resources that can be used by more than one jobs at the same time (e.g. file) are modeled as a resource type that has many units, each used in a mutually exclusive manner.

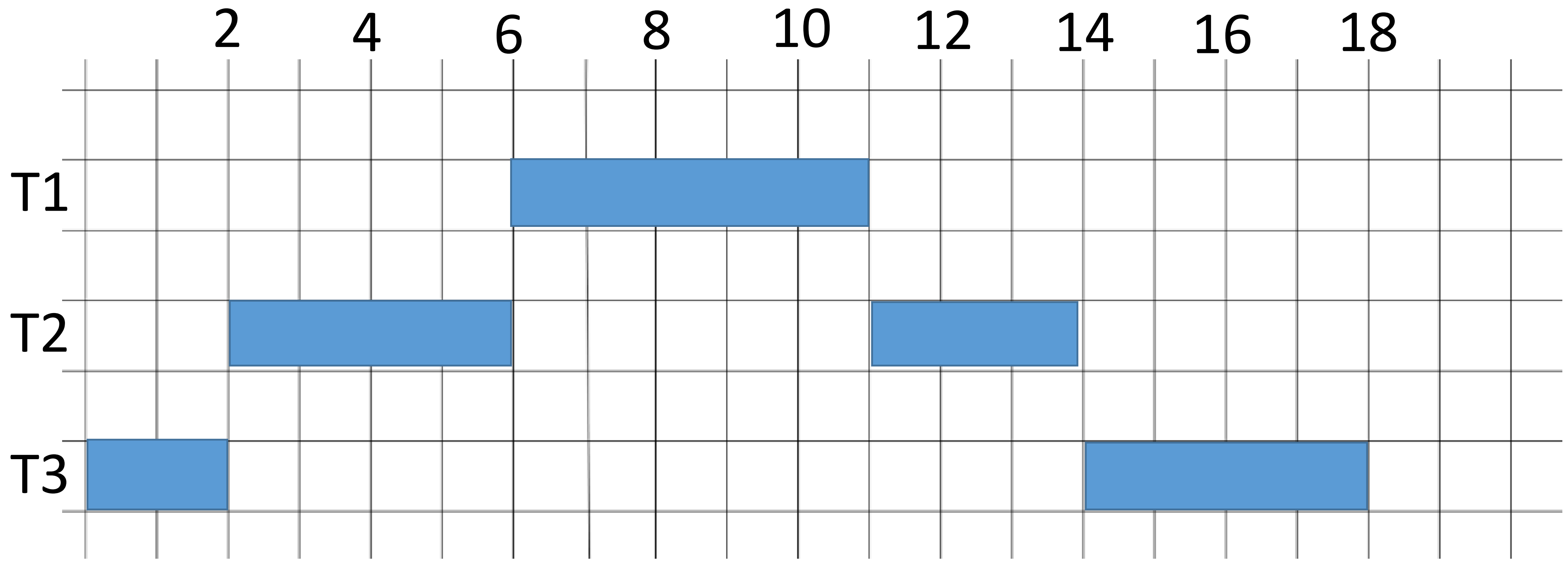
Mutual Exclusion and Critical Sections

- When a job wants to use k_i units of resource Ri, it executes a lock to request them, denoted by L(Ri; k_i). An unlock is denoted by U(Ri; k_i).
- In case Ri has just one unit, a simpler notation L(Ri) and U(Ri) is used.
- A segment of a job that begins at a lock and ends at a matching unlock is called a critical section, denoted by [R; t]
- A critical section that is not included in any other critical sections is called outermost critical section

Problem caused by Resource Contention

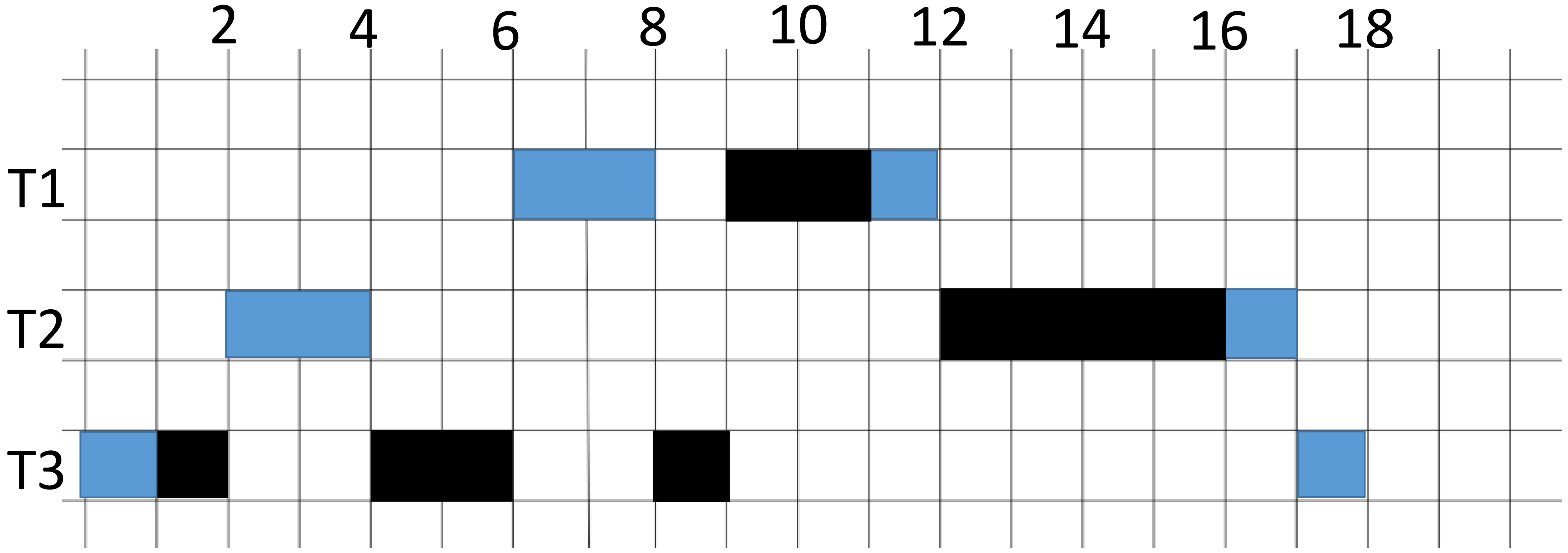
- Consider 3 tasks: T1(6; 12; 5; 8); T2(2; 16; 7; 15), T3(0; 18; 6; 18)
- Assume EDF scheduling algorithm
- For the first jobs in the tasks, T1 has the highest priority and T3 the lowest priority
- If we do not consider resource contention, what is a feasible schedule (assuming the tasks are preemptable)?
- What happens if the tasks T1; T2; T3 each has critical sections [R; 2]; [R; 4]; [R; 4] after 2, 2, 1 time units, respectively?

A Possible Schedule w/o Resource Contention



A Possible Schedule w. Resource Contention

Black boxes indicate critical sections of each task



Priority Inversion

- It shows how resource contention can delay completion of a higher priority task.
- Tasks T1; T2 could be completed by time 11 and 14 if there was no resource contention

Higher priority task can be blocked by a lower priority task due to resource contention

Priority Inversion

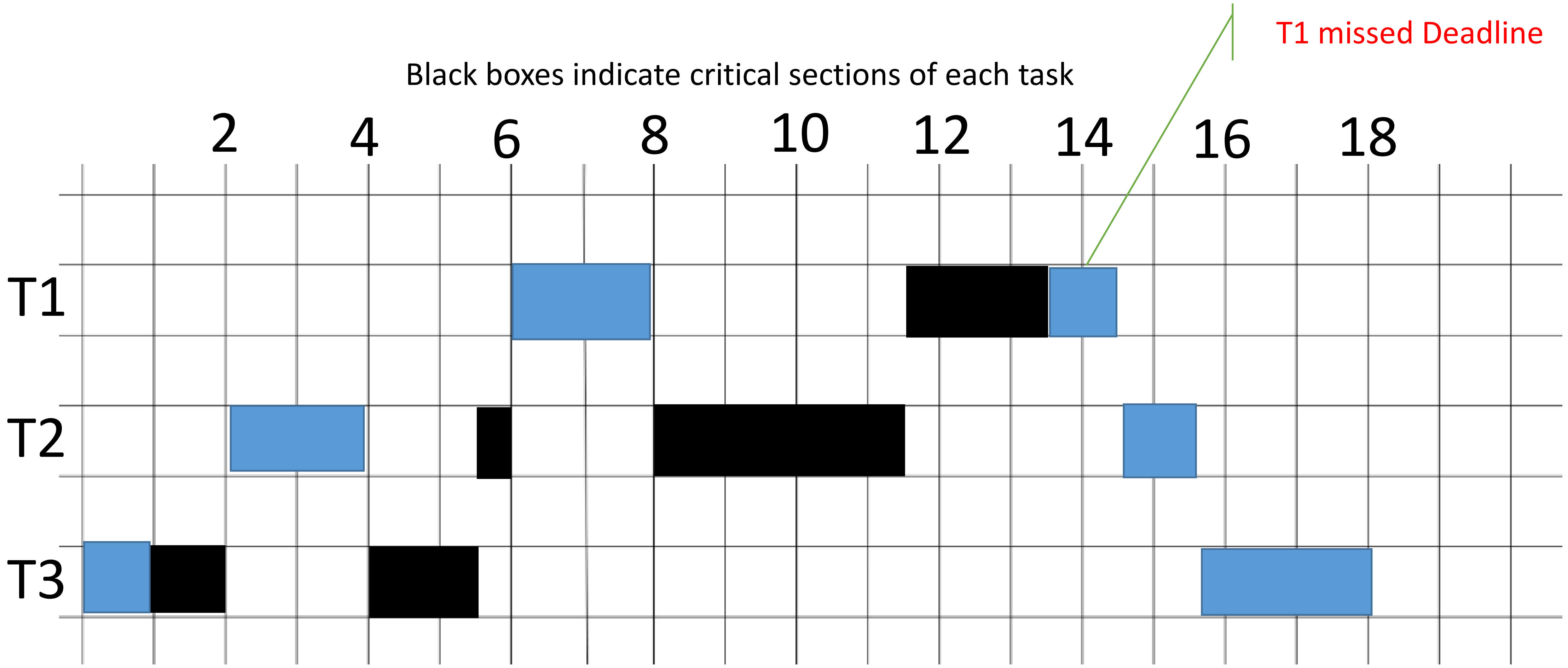
- This is because even if the tasks are preemptable, resources are allocated on nonpreemptive basis
- This phenomenon is called priority inversion (ref. time intervals (4, 6) and (8, 9))
- Sources of priority inversion
  - Access shared resources guarded by mutex or semaphores.
  - Access non-preemptive subsystems, e.g., storage, networks.
- Non preemptivity of resource allocation can also cause deadlocks e. g. when there are two jobs that both require resources X and Y.

Timing Anomalies

- Priority inversion may result in timing anomalies i. e. some tasks may not be able to meet their deadlines. For example, if we reduce CS of T3 to 2.5, so that T1(6; 12; 5; 8); T2(2; 16; 7; 15), T3(0; 18; 6; 18), and the tasks T1; T2; T3 each has critical sections [R; 2]; [R; 4]; [R; 2.5], respectively. What does the schedule look like if EDF is used?

Note that if without critical section, the first instance of T1, T2 and T3 will be completed at 11, 14, and 18 respectively. All deadlines are met.

Timing Anomalies if T3 Requires [R;2.5]

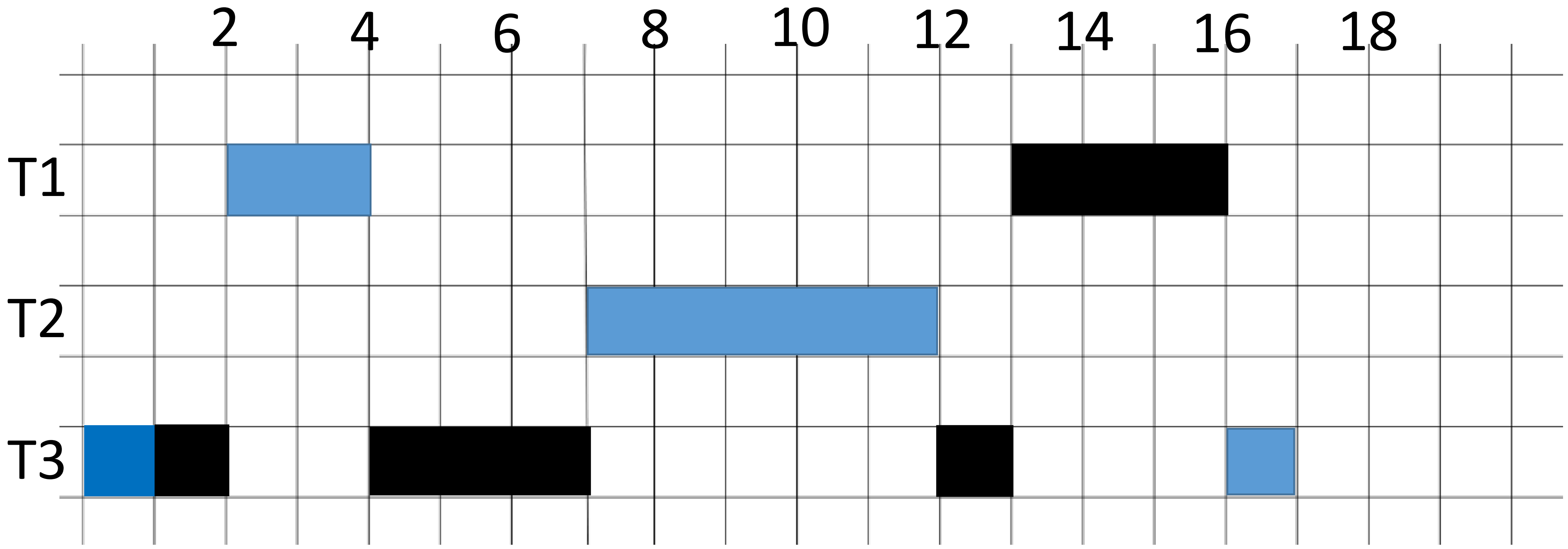


Example

- Give the RM schedule of the task set, if
  - T1(2, 16, 5, 16)   [R; 3] (Assuming T1 tries to lock R, 2 time units after it is scheduled)
  - T2(7, 17, 5, 17)   No CS
  - T3(0, 18, 7, 18)   [R; 5] (Assuming T3 locks R, 1 time unit after it is scheduled)

Priority Inversion May Be Unbounded

Black boxes indicate critical sections of each task



- Without a good resource access control protocol, duration of a priority inversion may be infinite

What really happened on Mars?

- In the year 1997, the Pathfinder was landed on Mars, and began to transmit data back to Earth. Days later transmission of data and images was interrupted by a series of total systems resets.

- Diagnosing the issue:

  T1. Data gathering task ran as an infrequent, low priority thread, and used the information bus synchronized with mutual exclusion locks (mutexes).

  T2. Long-run communication tasks (with priority in between)

  T3. Bus Management task has high priority

  Task T2 blocks T3, and system detected something had gone wrong, and ordered a total system reset.

Solution --- Non-preemptive Critical Section Protocol (NPCS)

- Schedule all critical sections non-preemptively
- **While a task holds a resource it executes at a priority higher than the priorities of all tasks**
- In general, a higher priority task is blocked only when some lower priority job is in critical section
- Once the blocking critical section completes, no lower priority task can get the processor and/or acquire any resource until the higher priority task completes.

Advantage and Disadvantage

- Advantages:
  - Does not need prior knowledge about resource requirements of tasks
  - Simple to implement
  - Can be used in both static and dynamic priority schedulers
  - Good protocol when most critical sections are short and most tasks conflict with one another

- Disadvantage:
  - A task can be blocked by a lower priority task for a long time even without a resource conflict

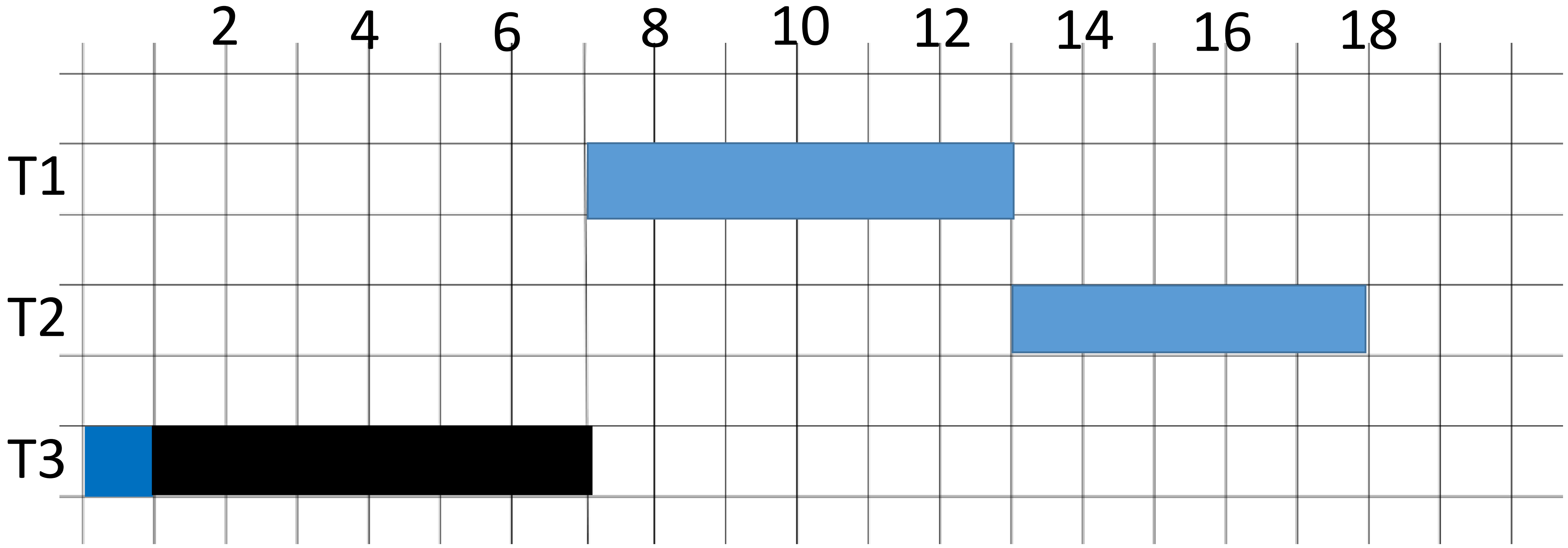
Example

- Give The Schedule of The Task Set
  - T1(2, 8, 5, 8) No CS
  - T2(5, 10, 5, 10) No CS
  - T3(0, 18, 7, 18) [R; 6] (Assuming T3 locks R at time t=1)

- By NPCS, what will be the schedule using RM/EDF (consider the first instances of the tasks only)?

Deadlines of T1 and T2 are missed!

Black boxes indicate critical sections of each task


2
4
6
8
10
12
14
16
18
T1
T2
T3