Real-Time Systems

Combined Scheduling of
Periodic and Aperiodic Tasks

Assumptions & Issues

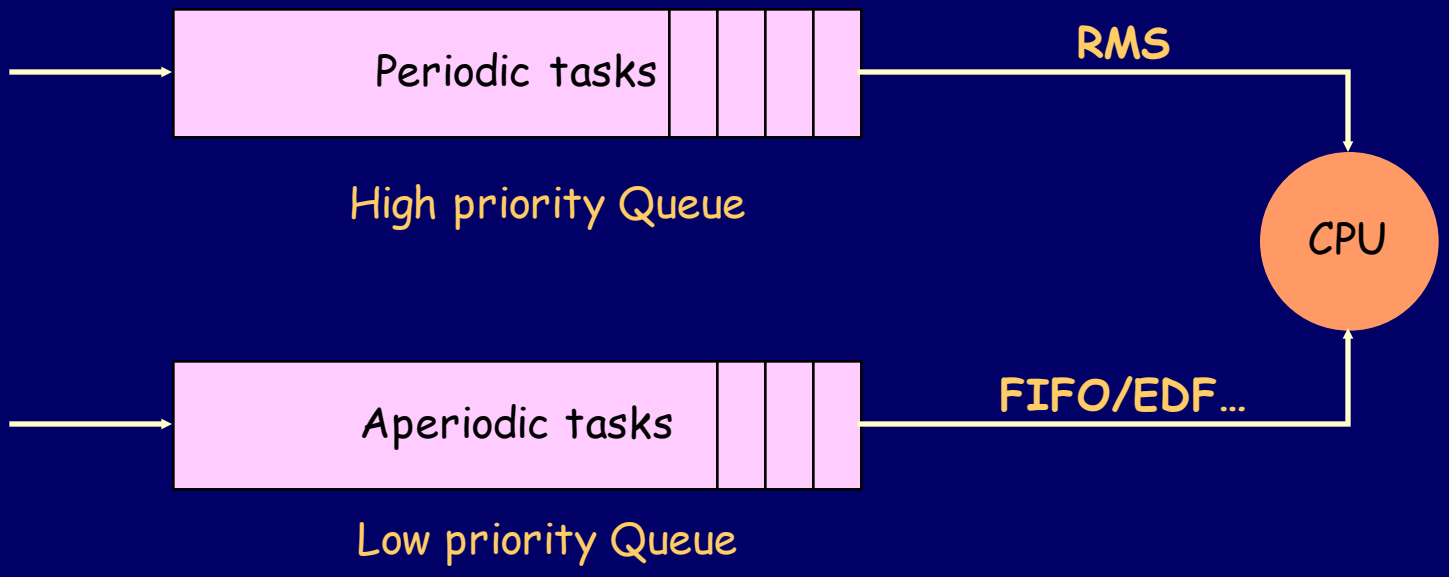
- RMS scheduling algorithm used
- All periodic tasks start at time t=0 (same as before)
- Periodic tasks relative deadlines are equal to end of period
- Arrival times of aperiodic tasks unknown

- Schedulability of periodic tasks
- Response time for aperiodic tasks
- Implementation considerations

Background Scheduling Algorithm

- No server is created.

- Aperiodic tasks are executed when there is no periodic task to execute.
- Simple, but no guarantee on aperiodic schedulability



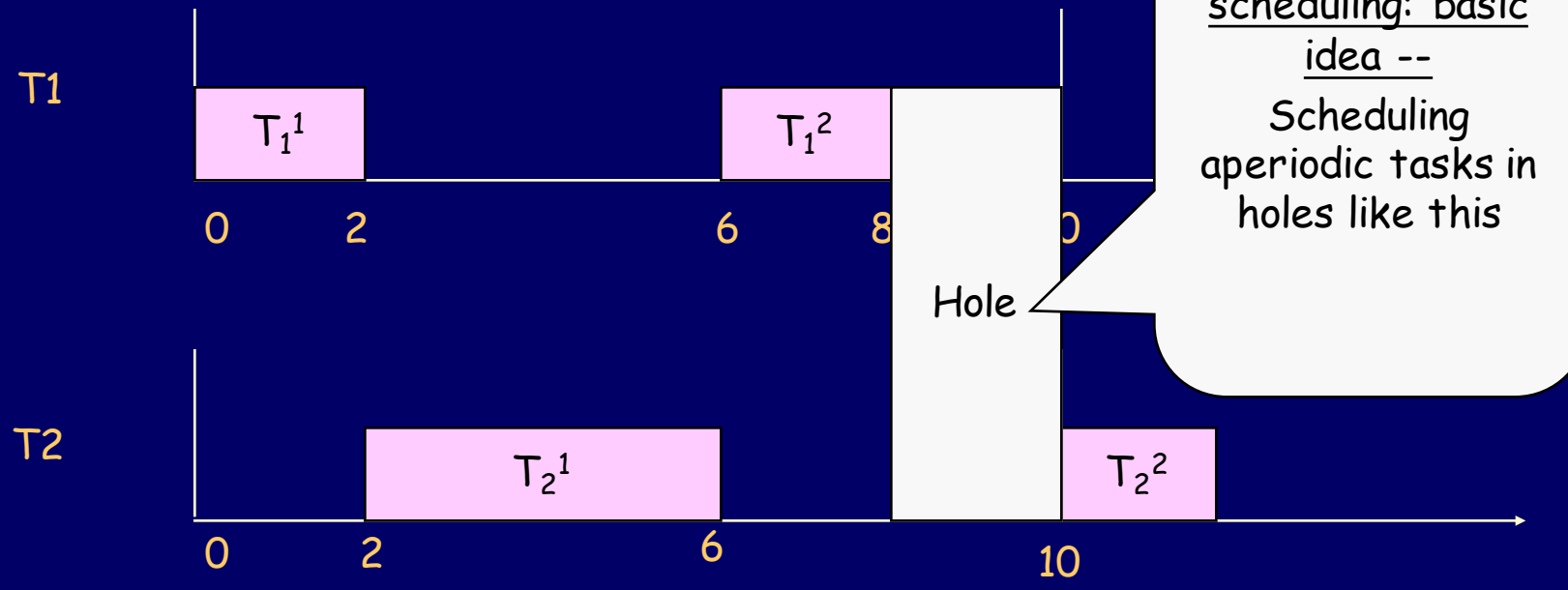
Normal RMS schedule: Notice the holes

Task set: $T_i = (c_i, p_i)$

T1 = (2,6) and T2 = (4,10)

Schedulability check:

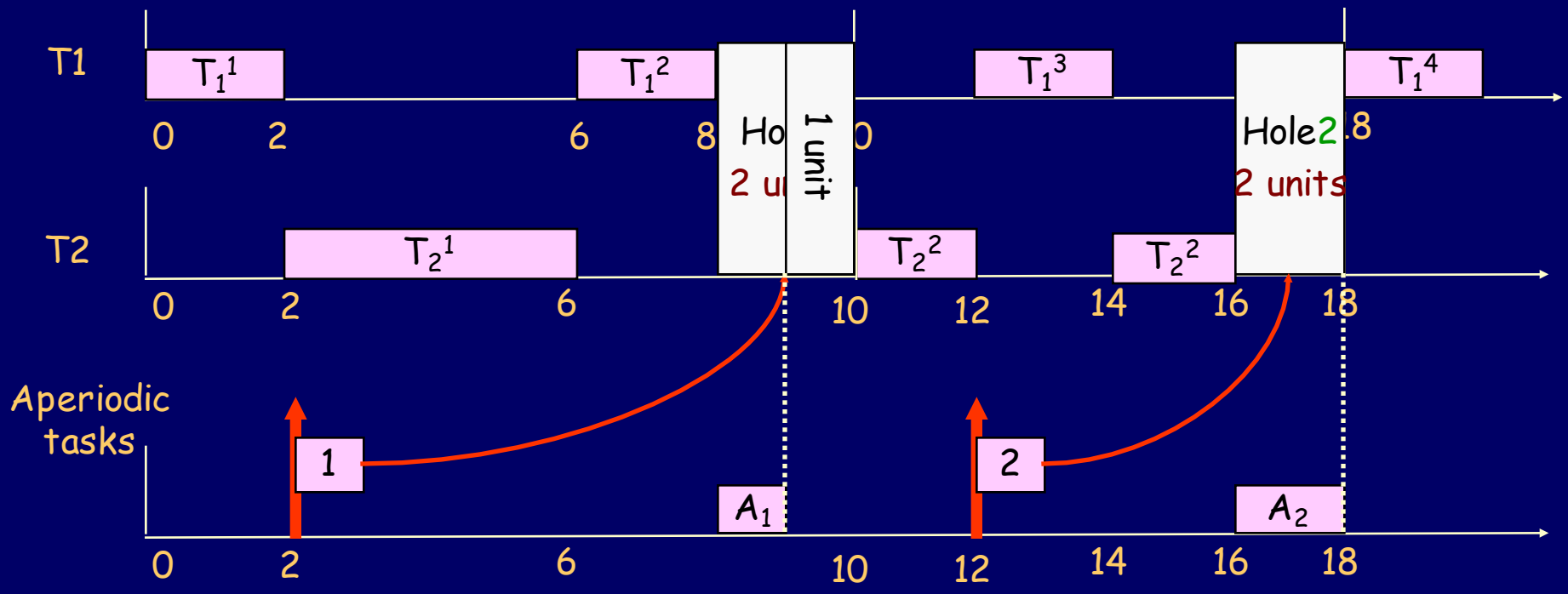
$2/6 + 4/10 = 0.33 + 0.40 = 0.73 \leq 2(\sqrt{2} - 1) = 0.82$

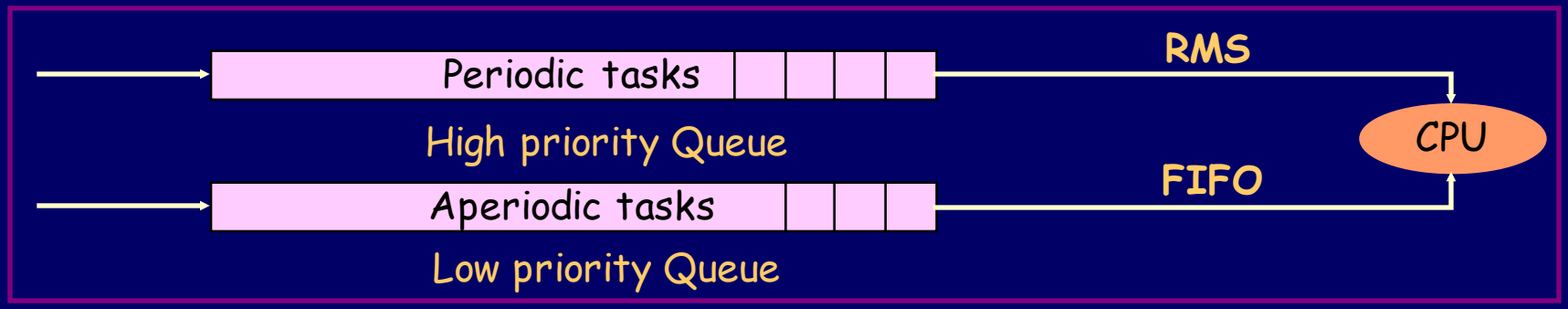


Background scheduling: basic idea --

Scheduling aperiodic tasks in holes like this

Background Scheduling: Example





Combined Scheduling

- Creating a *periodic server* Ts=(Cs, Ps) for processing aperiodic workload. Create one or more server tasks.

- Aperiodic tasks are scheduled in the periodic server's time slots. This policy could be based on deadline, arrival time, or computation time.

- Algorithms – all algorithms behave the same manner when there are enough aperiodic tasks to execute
  - Polling Server (bandwidth non-preserving)
  - Deferrable Server (bandwidth preserving)
  - Priority Exchange Server (bandwidth preserving)
  - Sporadic Server (bandwidth preserving)

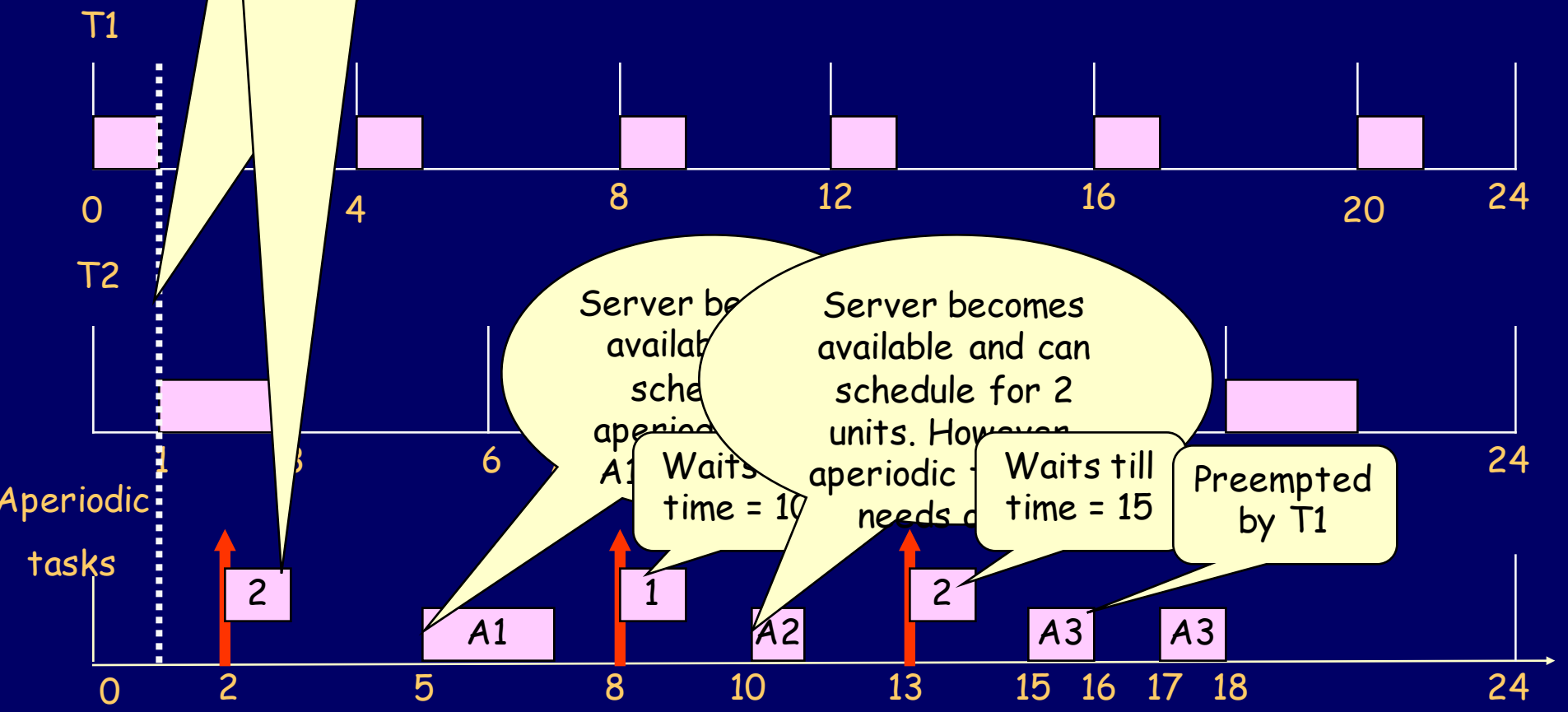
Polling Server

- A periodic server is created.
- If there are no aperiodic tasks at an invocation of the server (as per RMS), the server suspends itself during its current period and gets invoked again at its next period.
- If there are enough aperiodic tasks in an invocation, it serves up to Cs capacity.
- The computation time allowance for the server is replenished at the start of its period
- Include Ts in the task set and do schedulability test
- Poor response time for aperiodic tasks

ample

sk set: $T_i = (c_i, p_i)$	set: $T_i = (c_i, p_i)$
, T2 = (2,4) and Ts = (2,5)	and Ts = (2,5)

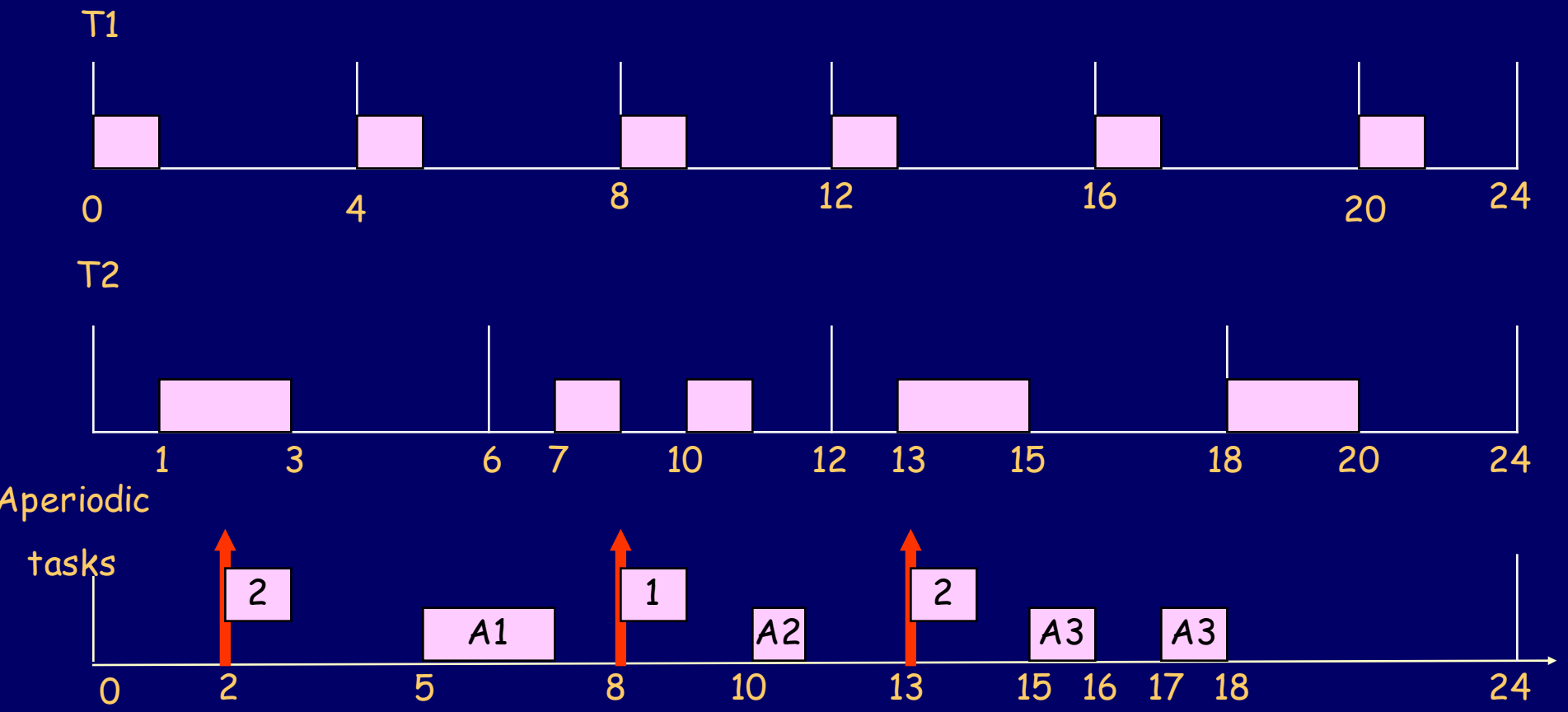
This aperiodic task cannot be scheduled now as there is no server available till time = 5



Polling server: Example (no animations)

Task set: $T_i = (c_i, p_i)$

T1 = (1,4) , T2 = (2,6) and Ts = (2,5)



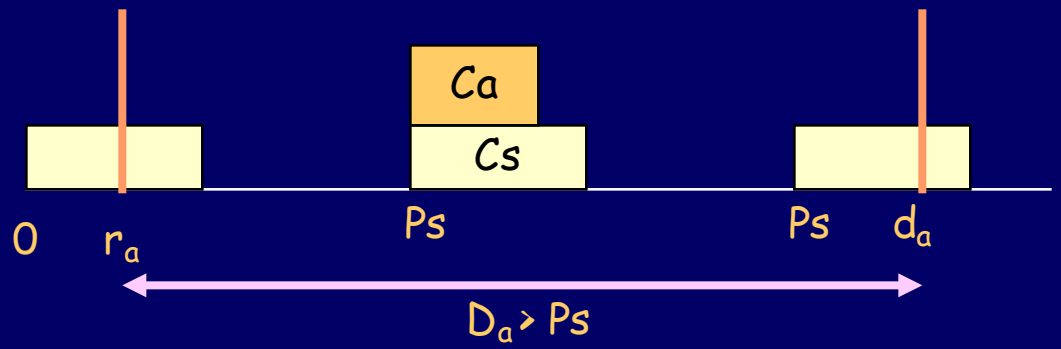
Polling server: Schedulability Analysis

- Schedulability analysis involves
  - Schedulability of periodic tasks
  - Schedulability of Aperiodic tasks
- Schedulability of periodic tasks can be evaluated by introducing a periodic task equivalent to the server. Therefore, the schedulability test is:

$$\sum_{i=1 \text{ to } n} (Ci / Pi) + (Cs / Ps) \leq (n+1)[2^{1/(n+1)} - 1]$$

Polling server: Schedulability Analysis

- Aperiodic task guarantees:
  - Consider a single aperiodic task A_i, arrived at r_a, with computation time C_a and deadline D_a. Since an aperiodic task can wait at most for one period before receiving service, if $C_a \leq Cs$ the request is certainly completed within two server periods. Thus it is guaranteed if $2P_s \leq D_a$



Polling server: Schedulability Analysis

- Aperiodic task guarantees:
  - For arbitrary computation times, the aperiodic task is certainly completed in ceil(Ca/Cs) server periods; hence it is guaranteed if
    - Ps + ceil(Ca/Cs) * Ps ≤ Da

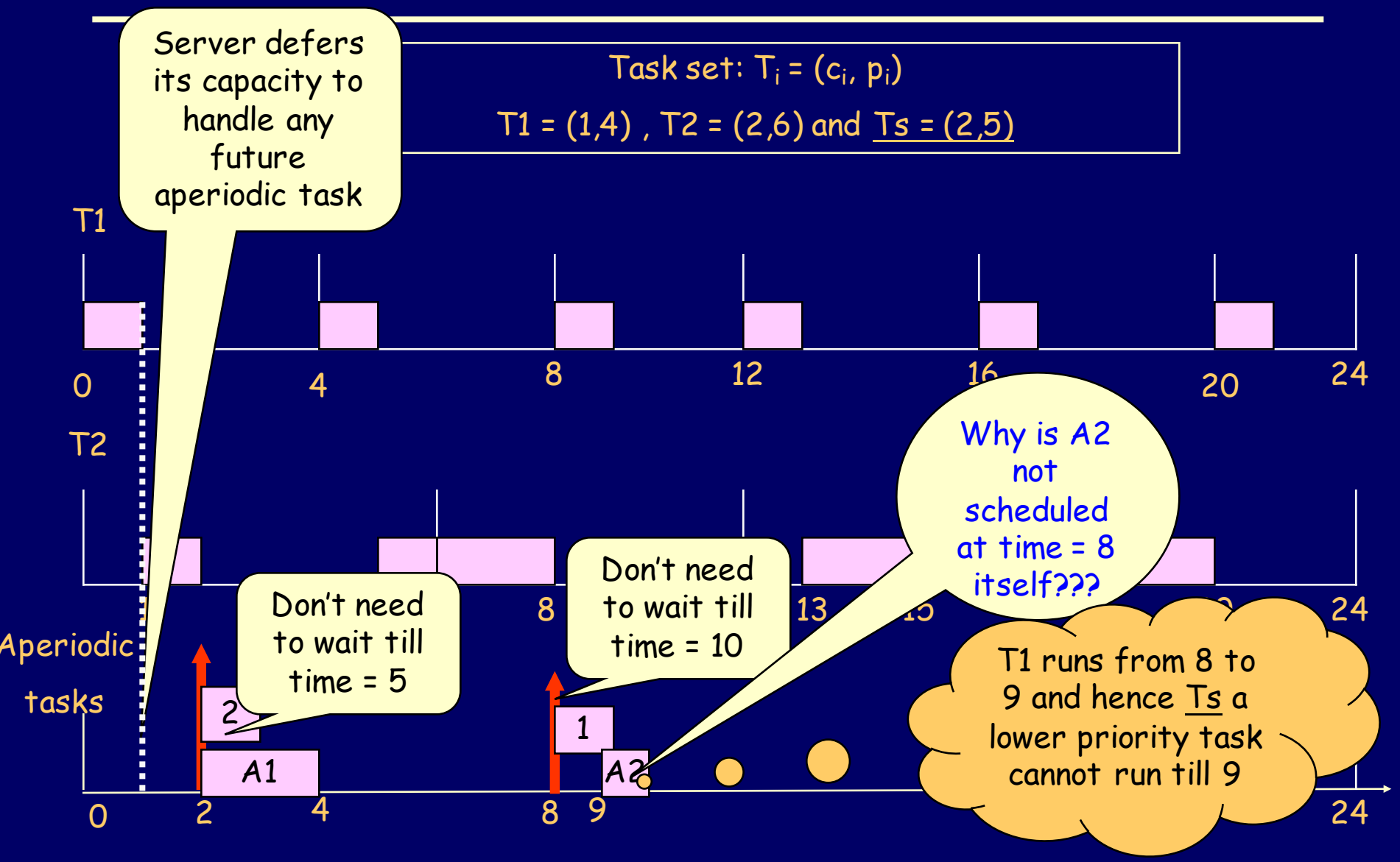
Deferrable Server

- A periodic server task is created.
- When the server is invoked with no outstanding aperiodic tasks, the server does not execute but defers its assigned time slot.
- When an aperiodic task arrives, the server is invoked (as per RMS) to execute aperiodic tasks and maintains its priority.

Deferrable Server (Contd.)

- The computation time allowance for the server is replenished at the start of its period.
- Provides better response time for aperiodic tasks than Polling server
- Under overload, deadlines are missed predictably.
- Similar schedulability test like polling server

Deferrable Server: Example



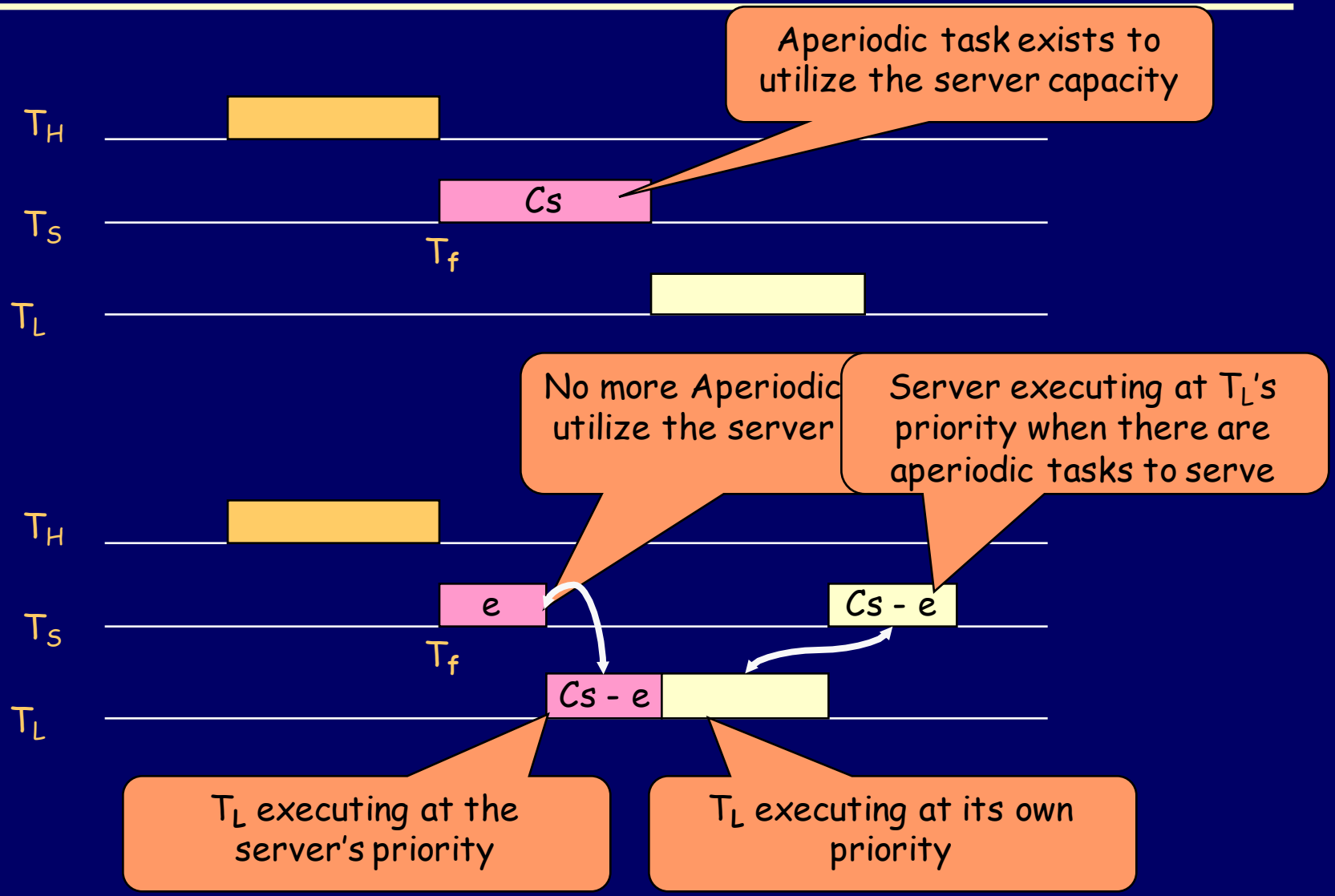
Priority Exchange Server

- A periodic server task is created.
- When the server invoked, the server runs if there are any outstanding aperiodic tasks.
- If no aperiodic task exists, the high priority server exchanges its priority with a lower priority periodic task for a duration of Cs', where Cs' is the remaining computation time of the server.

Priority Exchange Server (Contd.)

- In this way, the priority of the server decreases, but its computation time is maintained.
- The computation time allowance for the server is replenished at the start of its period.
- As a consequence, the aperiodic tasks get low preference for execution. Offers worse response time compared to Deferrable Server.
- Better schedulability bound for periodic task set compared to Deferrable Server.

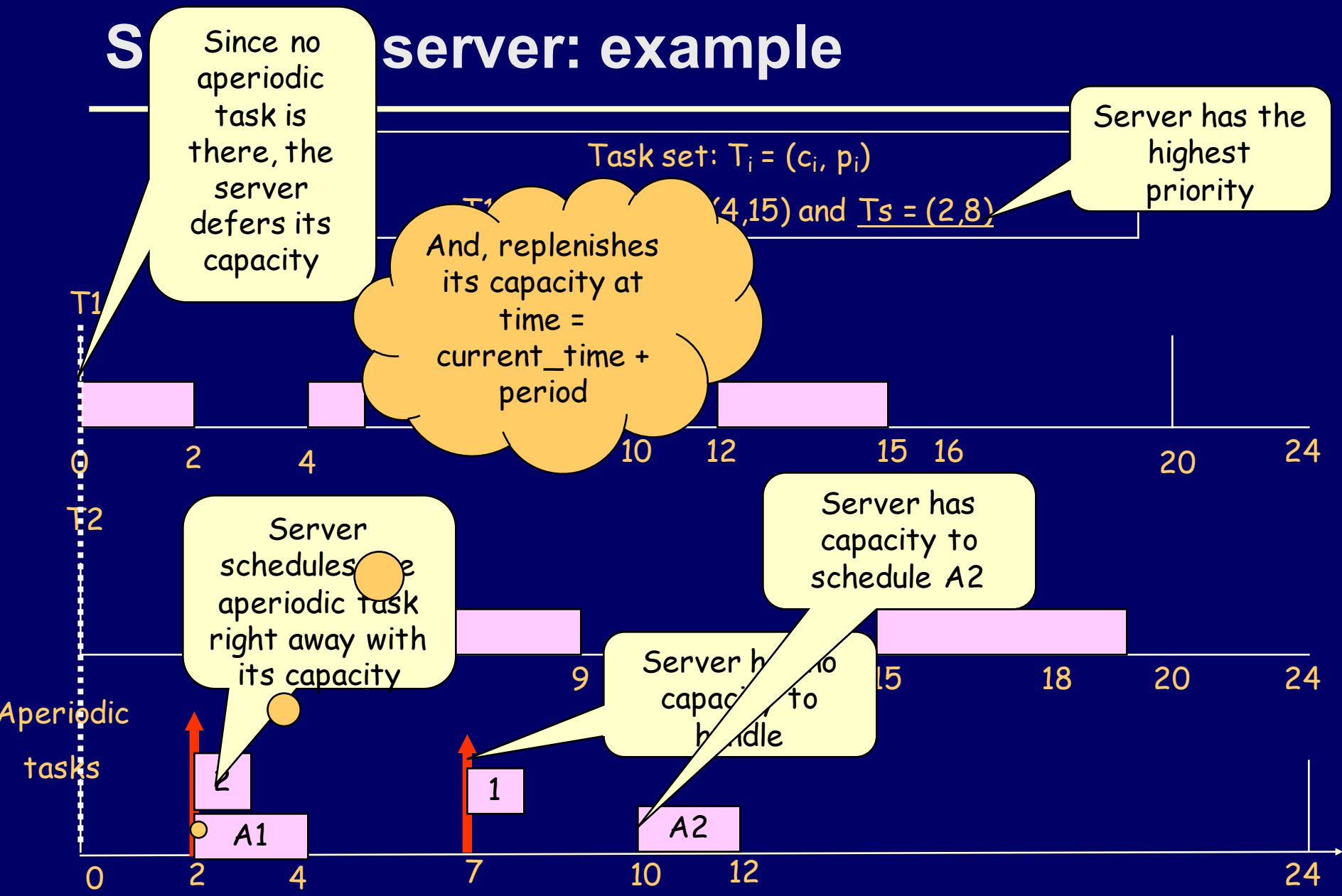
Priority Exchange server: example



Sporadic Server

- This algorithm allows to enhance the average response time for aperiodic tasks without degrading the utilization bound for periodic task set
- This is achieved by varying the points at which the computation time of the server is replenished, rather than merely at the start of each server period.
- In other words, any spare capacity (i.e., not being used by periodic tasks) is available for an aperiodic task on its arrival.

S server: example



Priority-driven preemptive scheduling- summary

- Scheduling algorithms
  - RMS & EDF utilization test
  - RMS & DMS Exact analysis
- Combined Scheduling
  - Polling, Deferrable, PE, Sporadic servers
- Resource Access Control
  - Priority Inversion
  - Priority Inheritance & Pri. Ceiling Protocols
  - Schedulability tests accounting Blocking