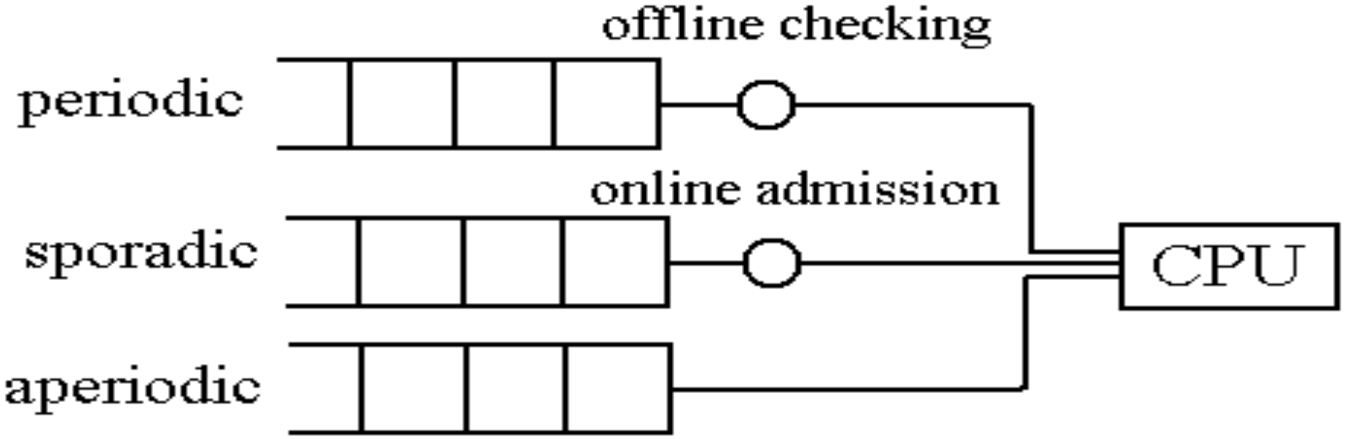
Introduction to
Real-Time Computer Systems

Sporadic Servers

Sporadic and Aperiodic

- A periodic task arrives in fixed time interval
- An aperiodic job is not periodic
  - An aperiodic job has no deadline, but we often want to minimize the response time
- What is sporadic?
  - A sporadic job has a well-defined maximum workload
    - Can be off-line guaranteed for successful execution
    - Has a defined (exec. time, minimum inter-arrival time)

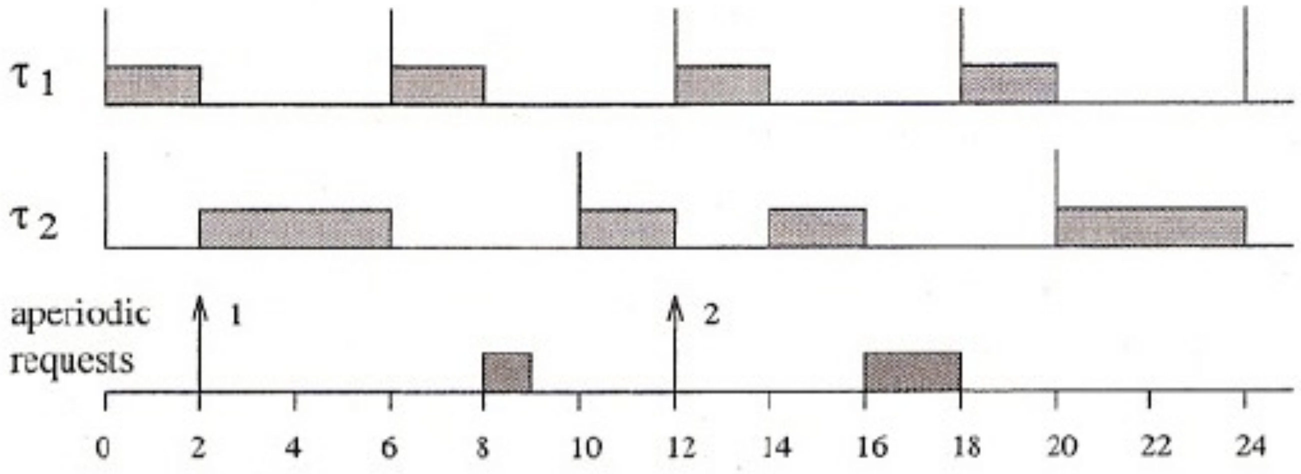
Scheduling Scheme



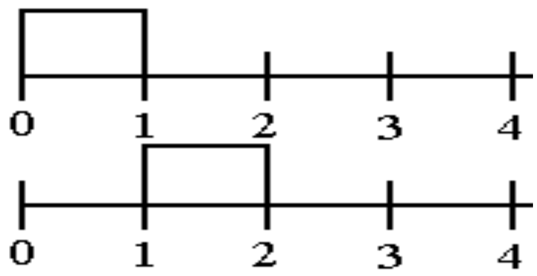
- Want to reject the sporadic job before we start execution if it cannot be serviced.

Background Scheduling Algorithm

- When the processor is free and there are no real-time jobs to schedule, then schedule an aperiodic job using FCFS
- The major advantage of background scheduling is its simplicity
  - However, we may not minimize the response time



Aperiodic Scheduling Idea



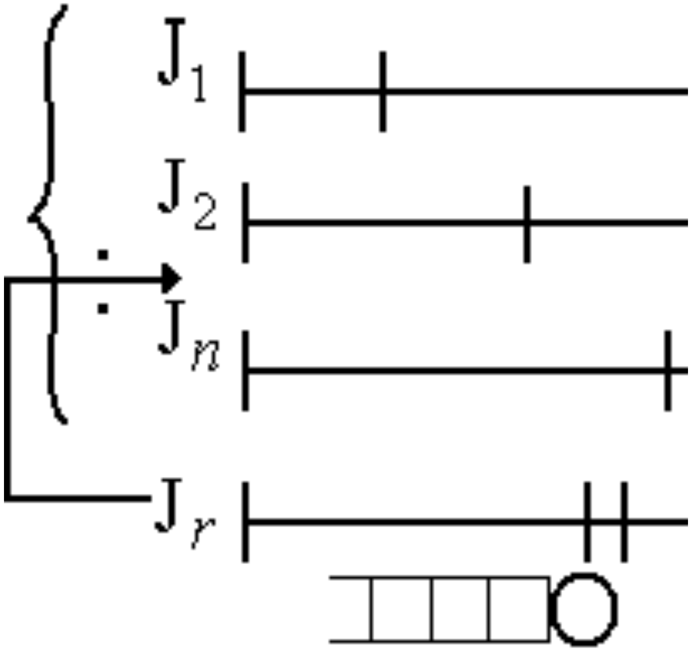
- periodic
- aperiodic
  - If we have 1 periodic job, 1/4
  - If we make the periodic job wait, the response time is 1.
  - Since periodic is pass/fail, there is no benefit to running it first
    - Should schedule aperidoic first to let it finish earlier
    - But we can't let aperiodic jobs run first all the time, or we may miss a deadline

Polling Server Algorithm

- The average response time of aperiodic tasks can be improved with respect to background scheduling through the use of a *server*, I.e., set up a periodic task whose purpose is to serve aperiodic requests as early as possible.
- Like any periodic task, a server is characterized by a period *Ts* and a computation time *Cs,* called server *capacity.*
- In general, the server is scheduled with the same algorithm used for periodic tasks and, once active, serves aperiodic requests within the limit of its server capacity.

Aperiodic Polling Server

- In every period, you allow aperiodic jobs time to execute. At runtime, poll the queued server



Polling Server Replenishment

- If no aperiodic requests are pending, PS suspends itself until the beginning of its next period, and the time originally allocated for aperiodic service is not preserved but is used by periodic tasks
- Note that if an aperiodic request arrives just after the server has suspended, it must wait until the beginning of the next polling period, when the server capacity is replenished at its full value.
- Schedulability condition:

  $\sum u_i + u_s \leq \text{Schedulability-bound}(n+1)$

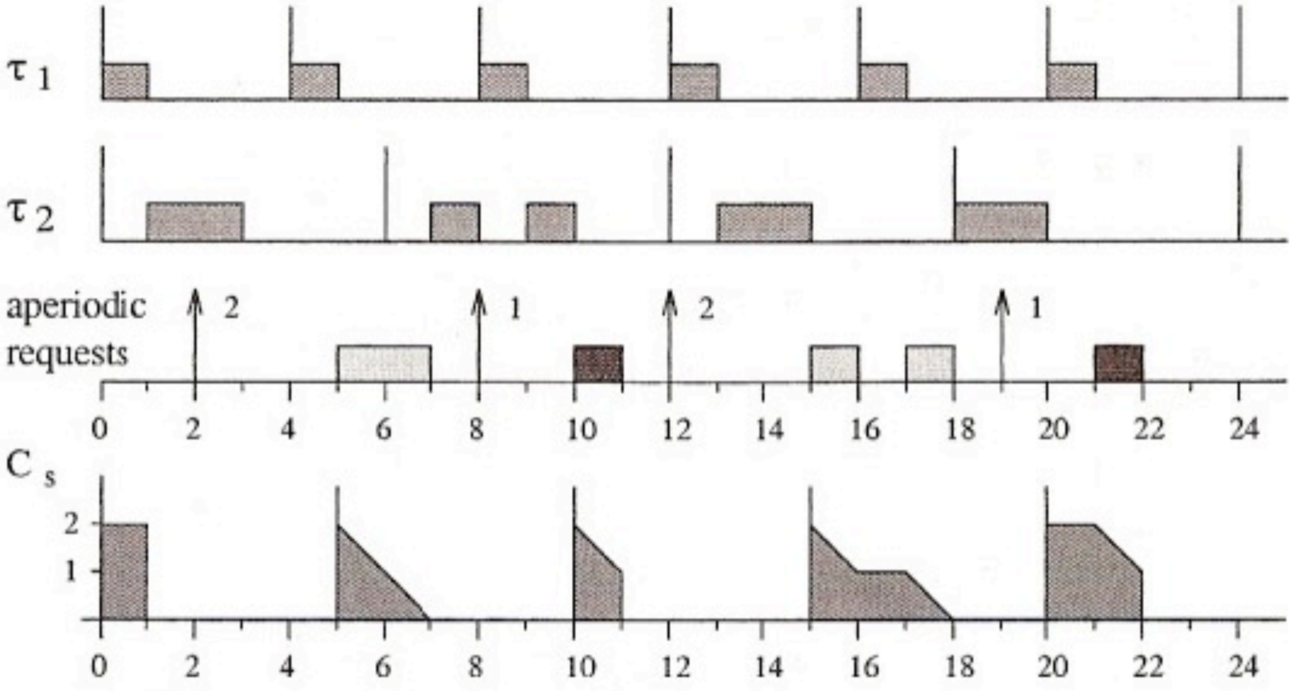
Polling Server Algorithm

	C_i	T_i
τ_1	1	4
τ_2	2	6

Server

$C_s = 2$

$T_s = 5$



Aperiodic Job's Response Time using PS

- After an aperiodic job's arrival (needs C_a) :

1. It has to wait for next server period
2. It can use only the fixed capacity of each period C_a/C_s
3. It may need to wait until the end of the last period to finish
4. Total delay is $(1+\lceil C_a/C_s \rceil)T_s$

Deferrable Server

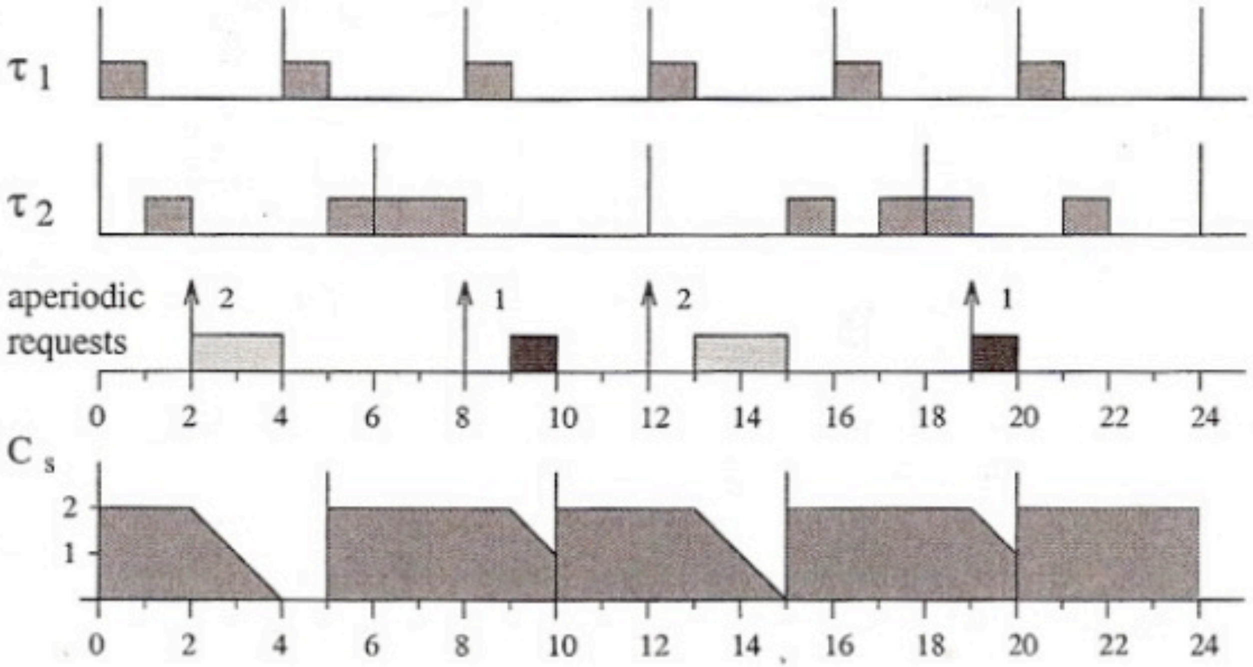
- Deferrable Server preserves its capacity in every period if no aperiodic job uses it immediately
- At end of period, the capacity is replenished to the full

	C_i	T_i
τ_1	1	4
τ_2	2	6

Server

$C_s = 2$

$T_s = 5$



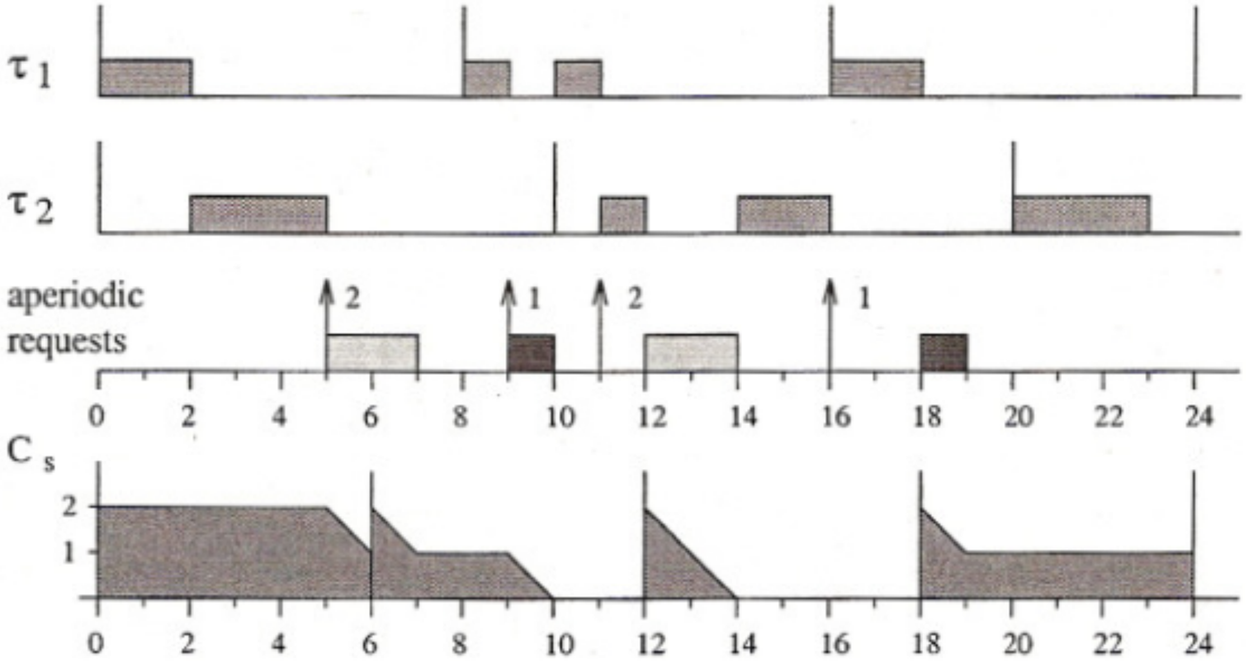
High Priority Deferrable Server

	C_i	T_i
τ_1	2	8
τ_2	3	10

Server

$C_s = 2$

$T_s = 6$



Example on DS Issue

- DS may compete with lower priority jobs unfairly, causing them to miss deadlines.
- For example, DS = (2, 4), Periodic Task = (2, 5)
- Aperiodic job arrival (with C = 2), [0, 2, 10, 12, 16]
- Periodic task would miss deadline at t = 15.

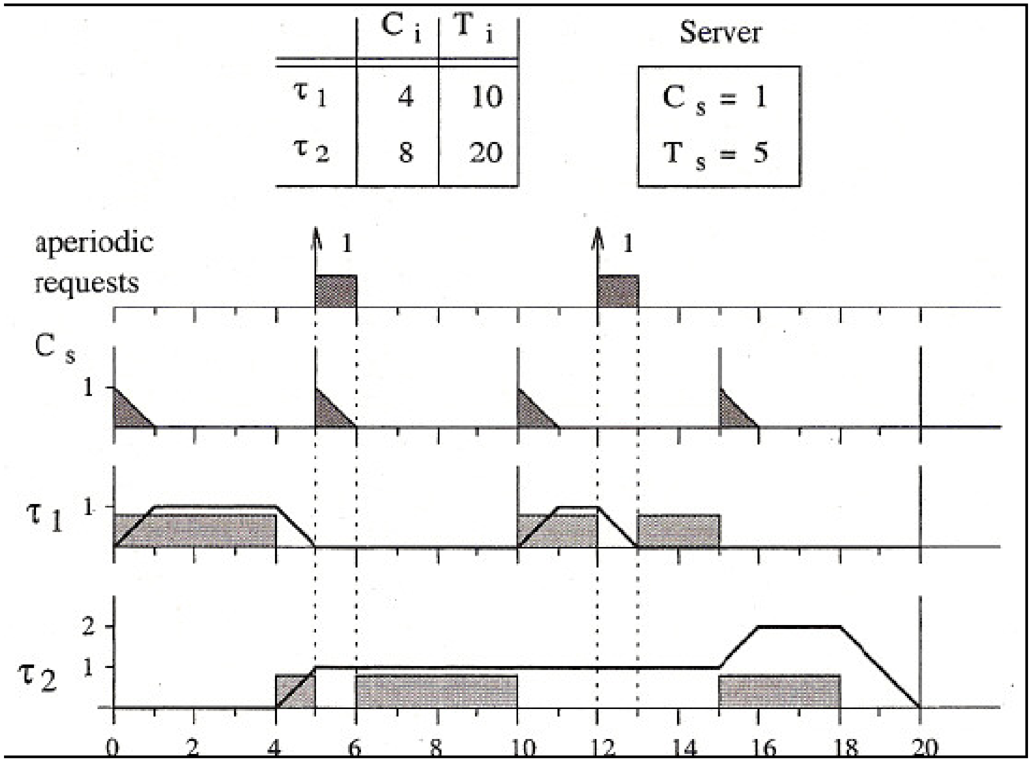
[Try to show the schedule yourself]

Priority Exchange

- One way to avoid the DS problem is to lower the *priority* of the aperiodic server *capacity*
  - The capacity can be used if it doesn't affect other lower priority jobs
  - But which priority should we place the unused capacity?
- The capacity should be degraded to a lower priority, if no aperiodic requests are pending
  - Let the person behind you go first if you are not ready at the supermarket checkout
- The exchange continues until the end of the period

Priority Exchange Examples

- The least upper bound of the processor utilization factor in the presence of PE is calculated by assuming that PE is the highest-priority task in the system.

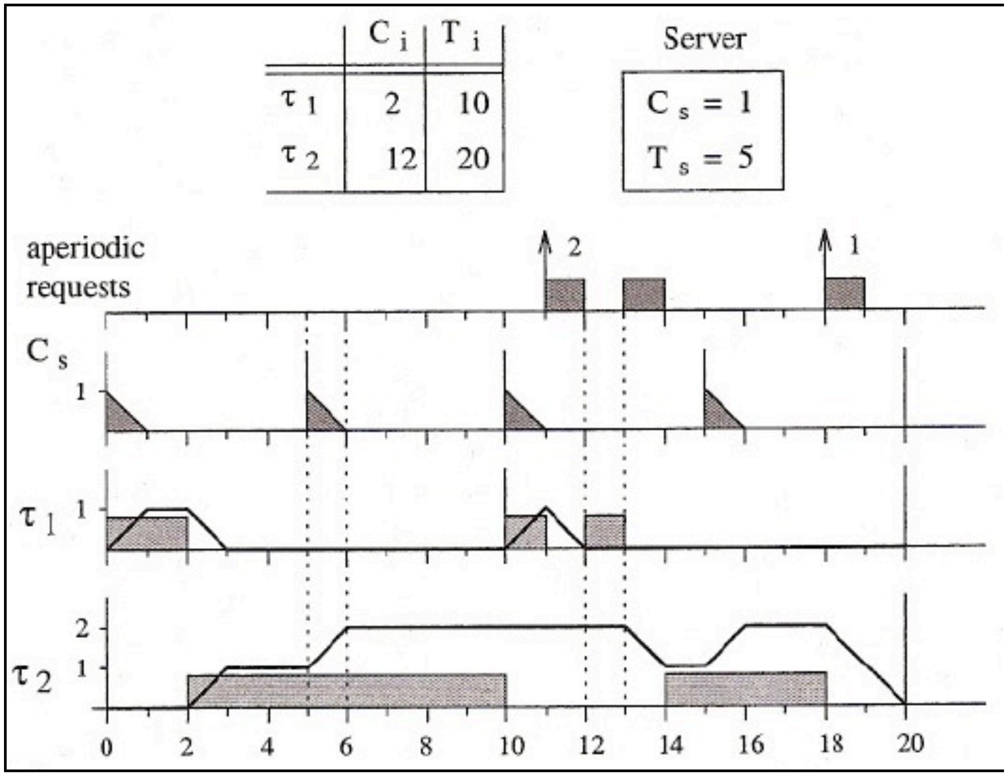


	C_i	T_i
τ_1	4	10
τ_2	8	20

Server

$C_s = 1$

$T_s = 5$



	C_i	T_i
τ_1	2	10
τ_2	12	20

Server

$C_s = 1$

$T_s = 5$