Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Real-Time Systems

4-Servers

Antonio Camacho Santiago
antonio.camacho.santiago@upc.edu

Introduction to hybrid task sets

Homogeneous task set: tasks are either periodic or aperiodic

Hybrid task set: tasks join **PERIODIC AND APERIODIC** activities

- Periodic tasks are usually time-driven with hard real time constraints
- Aperiodic tasks are usually event-driven with hard, soft or non-real time constraints
  - Sporadic tasks: aperiodic tasks with minimum interarrival time (MIT)
  - Firm tasks: aperiodic tasks that require online acceptance test

Objective of RT schedulers for hybrid tasks sets:

- Guarantee schedulability of critical tasks under worst case condition
- Provide good response time for soft and non-real time tasks
- Feasibility of the scheduling are based on periods and deadlines, and maximum interarrival times of aperiodic tasks
- How to: using servers

Classification of Servers

Server: Implements a **periodic task to service aperiodic requests**
It is characterized by a server computing time C_s and a server period T_s

No server: Background Scheduling (BS)

Servers based on Rate Monotonic (fixed priorities):

- Polling Server (PS)
- Deferrable Server (DS)
- Sporadic Server (SS)

Servers based on Earliest Deadline First (dynamic priorities):

- Dynamic Sporadic Server (DSS)
- Total Bandwidth Server (TBS)
- Constant Bandwidth Server (CBS)

BACKGROUND SCHEDULLING (BS)

Background Scheduling Requisites

The approach for the Background Scheduling (not a pure server) is based on:

Periodic tasks: are scheduled based on Rate Monotonic
arrive at t=0
$T_i=D_i$
Scheduled by RM

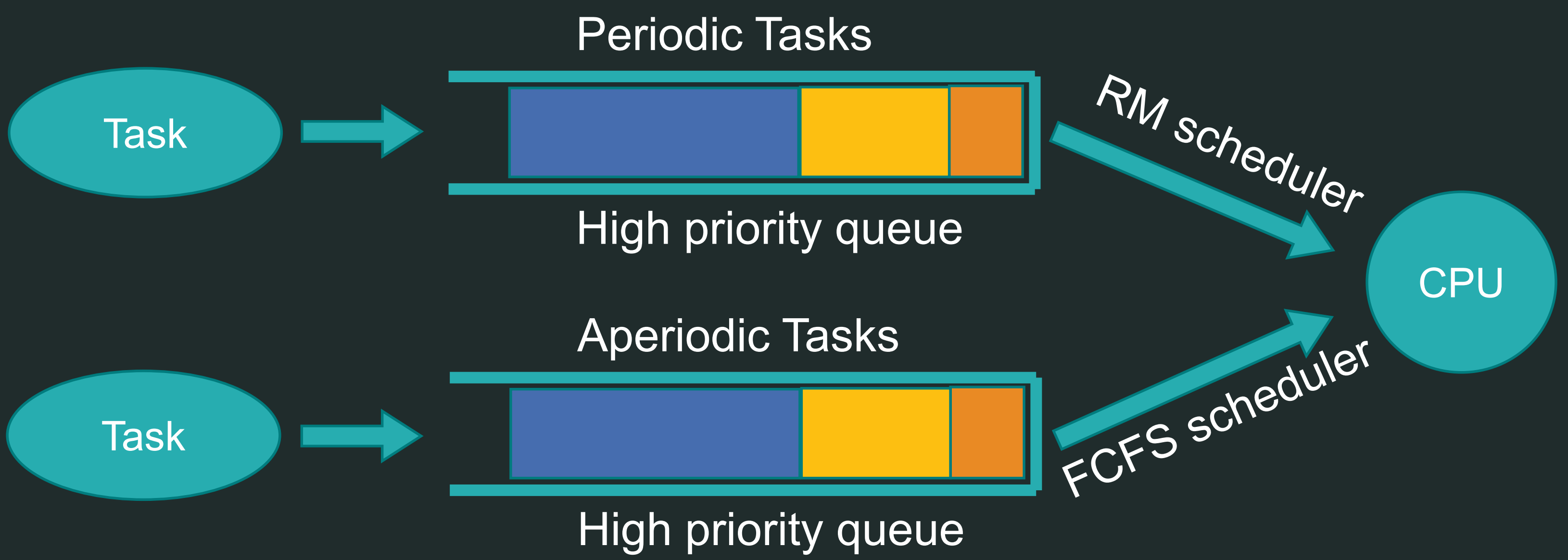
Aperiodic tasks: are scheduled based on First Come First Served
arrive at unknown time
T_i(minimum interarrival time)$=D_i$

All tasks can be preempted

Background Scheduling Queues

The periodic tasks are scheduled based on fixed priorities schedulers (RM or DM)

The aperiodic tasks are scheduled in background (no periodic task in ready state)



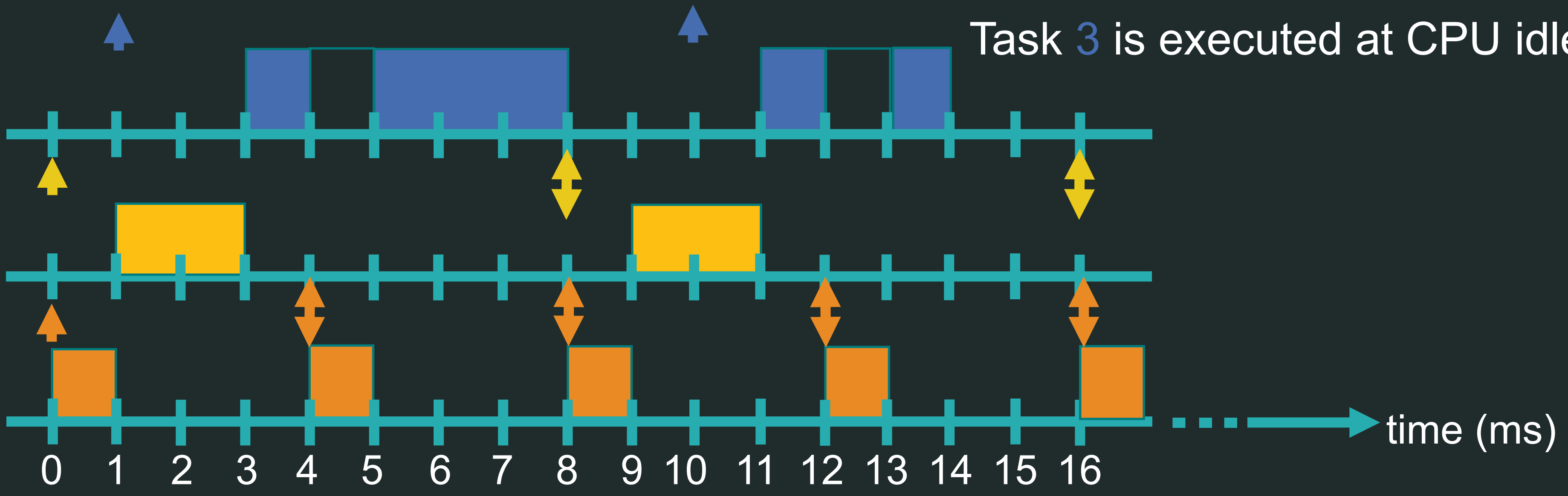
Background Scheduling Example

Task 1 and 2 are periodic

Task 3 is aperiodic and arrives at unknown time.

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	2
τ_2	2	8	1
Aperiodic τ_3	x	MIT=9	0

Task 3 is executed at CPU idle times



Background Scheduling

Pros:

- Simplicity
- Rate Monotonic for periodic tasks
- First Come First Served for aperiodic tasks
- Isolation among queues

Cons:

- Response time of aperiodic tasks can be high
- Activation of a periodic task preempts aperiodic task
- It is only useful for soft and non-critical aperiodical tasks

POLLING SERVER (PS)

Polling Server Requisites

Recall: A server is a periodic task to service aperiodic tasks

The approach for the Polling Server is based on:

Periodic tasks: scheduled based on Rate Monotonic
arrive at t=0
$T_i=D_i$

Aperiodic tasks: are scheduled based on Polling Server under RM
arrive at unknown time
T_i(minimum interarrival time)$=D_i$
The server is characterized by a period T_s and a computing time C_s

All tasks can be preempted

Polling Server

The periodic tasks are scheduled based on fixed priorities schedulers (RM or DM)

The aperiodic tasks are scheduled with a polling server (PS)

At the beginning of each T_s period, the server activates and its budget is recharged at its maximum value C_s

When the server becomes active and there are no pending jobs, C_s is discharged to zero.

When the server becomes active and there are pending jobs, they are served until $C_s > 0$.

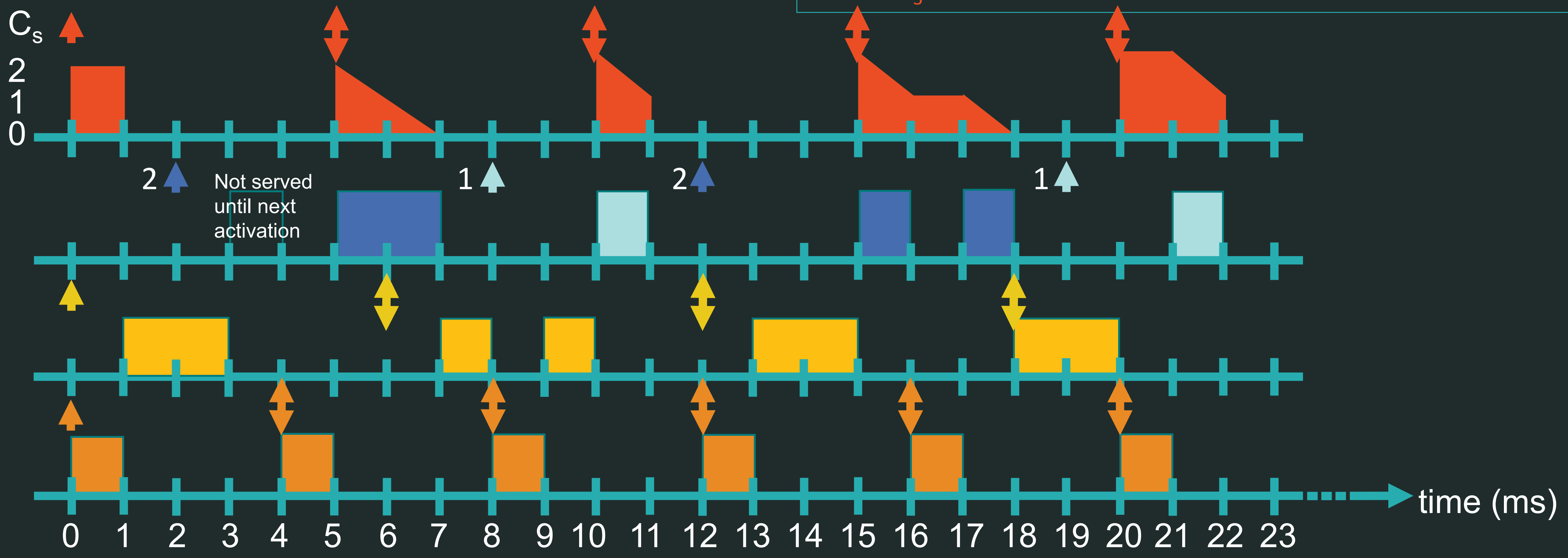
Example 1

Task 1 and 2 are periodic

Red represents the server capacity

Blue and cyan represents aperiodic tasks arriving at unknown time

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	3
τ_2	2	6	1
Server τ_3	2	5	2!!!



Polling Server Schedulability

A periodic RM with PS is schedulable if (sufficient condition)

$$U_p + U_s \leq U_{lub}(n+1)$$

or checking the hyperbolic bound as

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{2}{U_s + 1}$$

where U_p is the utilization factor of the n periodic tasks

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

where U_s is the utilization factor of the periodic server

$$U_s = \frac{C_s}{T_s}$$

and U_{lub} is the least upper bound utilization factor.

Recall that for pure periodic RM, $U_{lub}(n) = n\left(2^{1/n} - 1\right)$ being *n* the number of tasks

Polling Server Schedulability

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	3
τ_2	2	6	2
Server τ_3	2	15	1

Is the task set schedulable?

$$U_p + U_s \leq U_{lub}(n+m)$$

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{C_1}{T_1} + \frac{C_2}{T_2} = \frac{1}{4} + \frac{2}{6} = 0.5833$$

$$U_s = \frac{C_s}{T_s} = \frac{2}{15} = 0.1333$$

$$U_{lub}(n+m) = U_{lub}(2 \text{ periodic tasks} + 1 \text{ periodic server}) = U_{lub}(3) = 3\left(2^{1/3} - 1\right) = 0.7798$$

Joining everything:

$0.5833 + 0.1333 = 0.7166 \ \leq 0.7798$ →Schedulable

Hyperbolic bound condition: $\prod_{i=1}^{n}(U_i + 1) \leq \frac{2}{U_s+1}$ gives $1.66 \leq 1.7648$→ Schedulable

Polling Server Schedulability

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	3
τ_2	2	6	2
Server τ_3	3	15	1

Is the task set schedulable?

$$U_p + U_s \leq U_{lub}(n+m)$$

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{C_1}{T_1} + \frac{C_2}{T_2} = \frac{1}{4} + \frac{2}{6} = 0.5833$$

$$U_s = \frac{C_s}{T_s} = \frac{3}{15} = 0.2$$

$$U_{lub}(n+m) = U_{lub}(2 \text{ periodic tasks} + 1 \text{ periodic server}) = U_{lub}(3) = 3\left(2^{1/3} - 1\right) = 0.7798$$

Joining everything:

$0.5833 + 0.2 = 0.7833 \geq 0.7798$ →Nothing can be said. However the hyperbolic condition is less restrictive…

Hyperbolic bound condition: $\prod_{i=1}^{n}(U_i + 1) \leq \frac{2}{U_s+1}$ gives $1.66 \leq 1.66$→ Schedulable

Polling Server Schedulability

Is the task set schedulable?

$U_p + U_s \leq U_{lub}(n+m)$

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{C_1}{T_1} + \frac{C_2}{T_2} = \frac{1}{4} + \frac{2}{6} = 0.5833$$

$$U_s = \frac{C_s}{T_s} = \frac{2}{5} = 0.4$$

$$U_{lub}(n+m) = U_{lub}(2 \text{ periodic tasks} + 1 \text{ periodic server}) = U_{lub}(3) = 3\left(2^{1/3} - 1\right) = 0.7798$$

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	3
τ_2	2	6	1
Server τ_3	2	5	2

Joining everything:

$0.5833 + 0.4 = 0.9833 \geq 0.7798$ →Nothing can be said

Hyperbolic bound condition: $\prod_{i=1}^{n}(U_i + 1) \leq \frac{2}{U_s+1}$ gives $1.66 \geq 1.42$ →Nothing can be said

Polling Server Dimensioning

How to select server characteristics T_s and C_s?
From

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{2}{U_s + 1}$$

One can obtain

$$U_s^{\max} = \frac{2}{\prod_{i=1}^{n}(U_i + 1)} - 1$$

Given a period T_s, then $C_s = U_s^{\max} \cdot T_s$

Given a budget C_s, then $T_s = \frac{C_s}{U_s^{\max}}$

For the top-right example: $U_s^{\max} = 0.2$

→ given $T_s = 5$ → $C_s = U_s^{\max} \cdot T_s = 0.2 \cdot 5 = 1$

→ given $C_s = 3$ → $T_s = \frac{C_s}{U_s^{\max}} = \frac{3}{0.2} = 15$

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	3
τ_2	2	6	2
Server τ_3	C_s=?	T_s=?	?

Polling Server

Pros:

- Simplicity
- Converts aperiodic jobs into periodic ones
- Rate Monotonic is therefore used for all the tasks
- It does not waste time if there is nothing aperiodic to do
- On the limit, in the worst case, PS behaves as a periodic task with $U_s = C_s / T_s$
- Schedulable if (suficient condition)

$$\prod_{i=1}^{n} (U_i + 1) \leq \frac{2}{U_s + 1}$$

Cons:

- Still long response time for aperiodic requests because they have to wait until next polling period

DEFERRABLE SERVER (DS)

Deferrable Server Requisites

Its main objective is to improve average response time of aperiodic requests compared to PS

The approach for the Deferrable Server is based on:

Periodic tasks: scheduled based on Rate Monotonic
arrive at t=0
$T_i = D_i$

Aperiodic tasks: are scheduled based on Deferrable Server under RM
arrive at unknown time
T_i(minimum interarrival time)=D_i
The server is characterized by a period T_s and a computing time C_s

All tasks can be preempted

Deferrable Server

Used to improve the average response time of aperiodic tasks compared with Polling Server

DS is implemented as a periodic task (T_s and C_s) to service aperiodic tasks

DS **preserves** its capacity when no requests are pending

Aperiodic request can be serviced as soon as they arrive provided that C_s>0

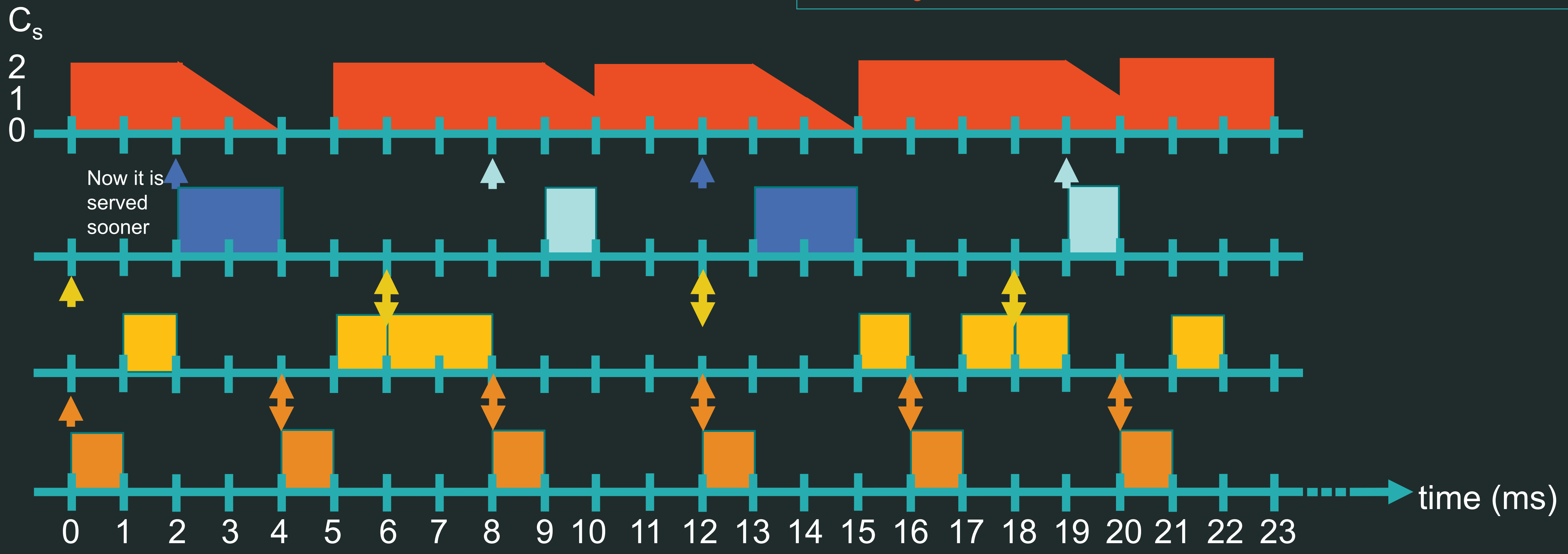
Deferrable Server Example

Task 1 and 2 are periodic

Red represents the server

Blue and cyan represents aperiodic tasks arriving at unknown time

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	3
τ_2	2	6	1
Server τ_3	2	5	2



Deferrable Server Schedulability

A periodic RM with DS is schedulable if (sufficient condition)

$$U_p \leq n\left[\left(\frac{U_s+2}{2U_s+1}\right)^{1/n} - 1\right]$$

or checking the hyperbolic bound as

$$\prod_{i=1}^{n}(U_i+1) \leq \frac{U_s+2}{2U_s+1}$$

where U_p is the utilization factor of the n periodic tasks

$$U_p = \sum_{i=1}^{n}\frac{C_i}{T_i}$$

where U_s is the utilization factor of the periodic server

$$U_s = \frac{C_s}{T_s}$$

Deferrable Server Dimensioning

How to select server characteristics T_s and C_s?
From

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{U_s + 2}{2U_s + 1}$$

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	3
τ_2	2	6	2
Server τ_3	C_s=?	T_s=?	?

One can obtain

$$U_s^{\max} = \frac{2 - \prod_{i=1}^{n}(U_i + 1)}{2\prod_{i=1}^{n}(U_i + 1) - 1}$$

Given a period T_s, then $C_s = U_s^{\max} \cdot T_s$

Given a budget C_s, then $T_s = \frac{C_s}{U_s^{\max}}$

For the top-right example: $U_s^{\max} = 0.14$ → given $T_s = 5$ → $C_s = U_s^{\max} \cdot T_s = 0.14 \cdot 5 = 0.71$

→ given $C_s = 3$ → $T_s = \frac{C_s}{U_s^{\max}} = \frac{3}{0.14} = 21$

Deferrable Server

Pros:

Improves the aperiodic time response

Schedulable if (suficient condition)

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{U_s + 2}{2U_s + 1}$$

Cons:

PS does not behave as a periodic task with $U_s = C_s/T_s$

Deferring a task can cause lower priority tasks to loose their deadlines

It jeopardises the schedulability of the periodic task set.

SPORADIC SERVER (SS)

Sporadic Server Requisites

Its main objective is to improve average response time of aperiodic requests
Compared with DS, it does not degrades the utilization factor of the periodic task set

The approach for the Sporadic Server is based on:

Periodic tasks: scheduled based on Rate Monotonic
arrive at t=0
$T_i=D_i$

Aperiodic tasks: are scheduled based on Sporadic Server under RM
arrive at unknown time
T_i(minimum interarrival time)=D_i
The server is characterized by a period T_s and a computing time C_s

All tasks can be preempted

Sporadic Server

Used to improve the average response time of aperiodic tasks compared with Polling Server

It preserves the budget like Deferrable Server, but it is less aggressive than DS, since the budget is replenished only T_s units after its consumption, not at the beginning of the server activation as in the DS.

SS is not activated periodically, but from the analysis point of view it behaves like a period task with computation time C_s and period T_s

Implementation rules:

The replenishment time (*RT*) of the server capacity occurs when SS is active and C_s>0 plus the server period: $RT=t_{(\text{SS active and } Cs>0)} + T_s$

The replenishment amount (*RA*) is computed when SS is idle or empty and its value is equal to the capacity consumed during the interval

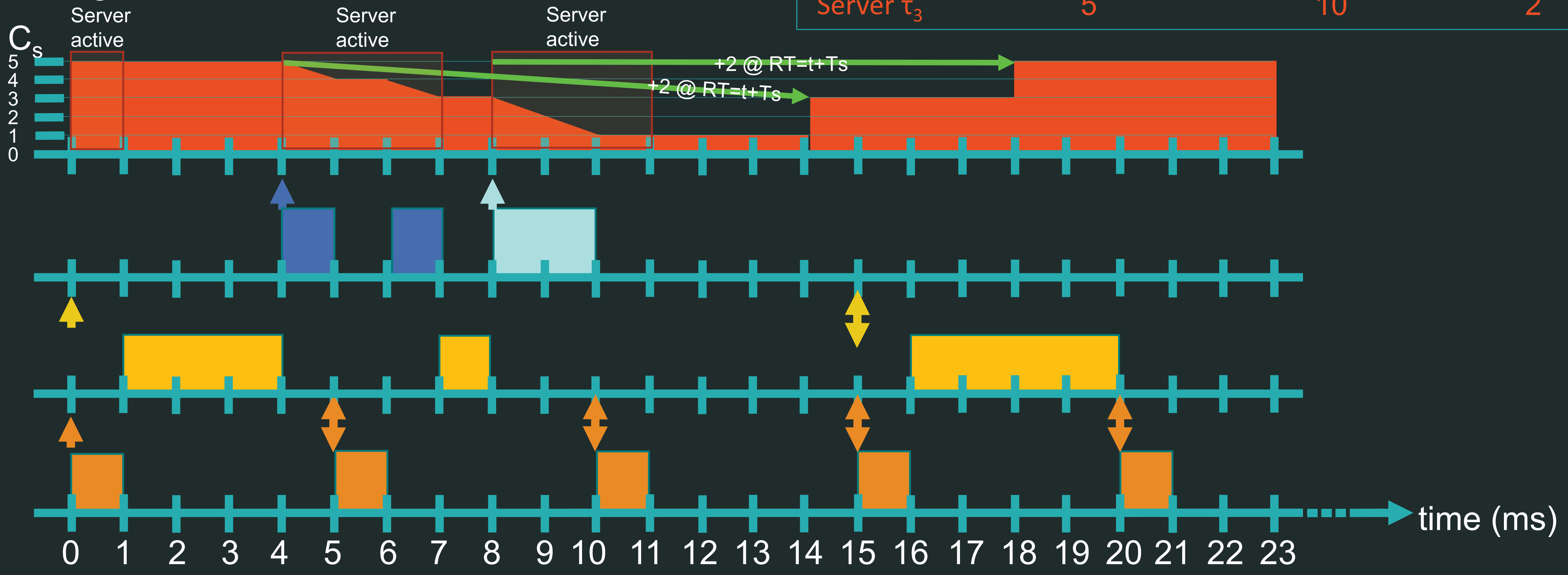
Sporadic Server Example

Task 1 and 2 are periodic

Red represents the server

Blue and cyan represents aperiodic tasks arriving at unknown time

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	5	3
τ_2	4	15	1
Server τ_3	5	10	2



Sporadic Server Schedulability

A periodic RM with SS is schedulable if (sufficient condition)

$$U_p \leq n\left[\left(\frac{2}{U_s+1}\right)^{1/n} - 1\right]$$

or checking the hyperbolic bound as

$$\prod_{i=1}^{n}(U_i+1) \leq \frac{2}{U_s+1}$$

where U_p is the utilization factor of the n periodic tasks

$$U_p = \sum_{i=1}^{n}\frac{C_i}{T_i}$$

where U_s is the utilization factor of the periodic server

$$U_s = \frac{C_s}{T_s}$$

Sporadic Server Dimensioning

How to select server characteristics T_s and C_s?
From

$$\prod_{i=1}^{n}(U_i + 1) \le \frac{2}{U_s + 1}$$

One can obtain

$$U_s^{\max} = \frac{2}{\prod_{i=1}^{n}(U_i + 1)} - 1$$

Given a period T_s, then $C_s = U_s^{\max} \cdot T_s$

Given a budget C_s, then $T_s = \frac{C_s}{U_s^{\max}}$

For the top-right example: $U_s^{\max} = 0.2$

→ given $T_s = 5$ → $C_s = U_s^{\max} \cdot T_s = 0.2 \cdot 5 = 1$

→ given $C_s = 3$ → $T_s = \frac{C_s}{U_s^{\max}} = \frac{3}{0.2} = 15$

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)	Priority RM
τ_1	1	4	3
τ_2	2	6	2
Server τ_3	C_s=?	T_s=?	?

Sporadic Server

Pros:

Improves the aperiodic time response

Less agressive than DS

Schedulable if (suficient condition)

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{2}{U_s + 1}$$

Cons:

SS behaves as a periodic task with $U_s = C_s / T_s$

Computation of online acceptance tests for firm tasks becomes difficult

A little bit complex, increases complexity due to replenishment time RT and replenishment amount RA rules

DYNAMIC SPORADIC SERVER (DSS)

Dynamic Sporadic Server Requisites

The approach for the Dynamic Sporadic Server is based on:

Periodic tasks: scheduled based on EDF
arrive at t=0
$T_i = D_i$

Aperiodic tasks: are scheduled based on EDF
arrive at unknown time
T_i(minimum interarrival time)$=D_i$

All tasks can be preempted

Next Servers are based on EDF (dynamic priorities) instead of RM (fixed priorities), which increases the overall utilization

Recall that pure periodic EDF guarantee condition is $U_{\text{total}} = \sum_{i=1}^{n} U_i \leq 1$ while pure periodic RM guarantee condition is $U_{\text{total}} = \sum_{i=1}^{n} U_i \leq n\left(2^{1/n} - 1\right)$

Dynamic Sporadic Server

Used to improve the CPU utilization compared with fixed priority servers

It is implemented with a computation time C_s and a pericd T_s

The server capacity is not replenished at its full value at the beginning of each server period but only when it has been consumed

The times at which the replenishments occur are chosen according to a replenishment rule, which allows the system to achieve full processor utilization.

Implementation rules for DSS:

- When created, the server has maximum C_s
- The replenishment time (*RT*) of the server capacity occurs when there is an aperiodic task pending and C_s>0 plus the server period. $RT=d_s=t_{(\text{aperiodictasspending and } Cs>0)} + T_s$
- The replenishment amount (*RA*) is computed when the last request is completed or C_s has been exhausted. Its value is equal to the capacity consumed during the interval

Note that SS implements a fixed priority depending on T_s, while DSS consists on a dynamic priority depending on deadlines

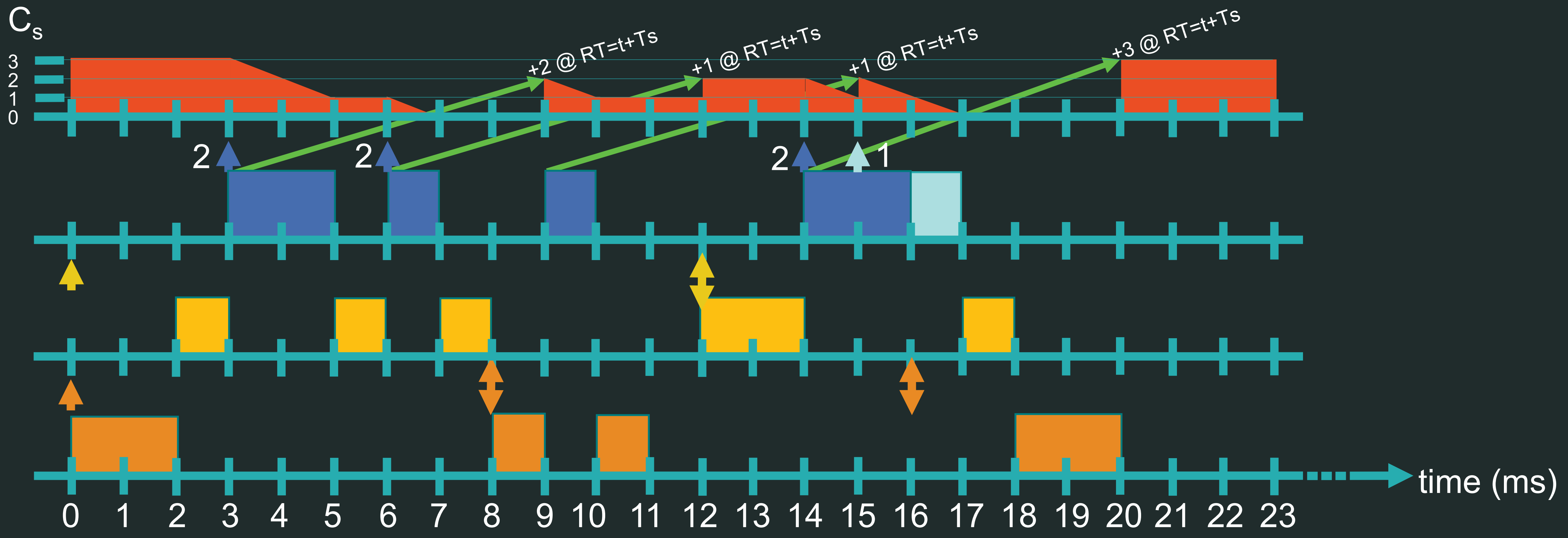
Dynamic Sporadic Server Example

Task 1 and 2 are periodic

Red represents the server

Blue and cyan represent aperiodic tasks arriving at unknown time

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	2	8
τ_2	3	12
Server τ_3	3	6



Dynamic Sporadic Server Schedulability

A periodic EDF with DSS is schedulable iff (necessary and sufficient condition)

$$U_p + U_s \leq 1$$

where U_p is the utilization factor of the n periodic tasks

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

where U_s is the utilization factor of the periodic server

$$U_s = \frac{C_s}{T_s}$$

Dynamic Sporadic Server

Pros:

Improves the aperiodic time response

Increases the CPU utilization

Schedulable iff (necessary and suficient condition)

$$U_p + U_s \leq 1$$

Cons:

Long period T_s causes long response time for aperiodic tasks.

This issue can be fixed by reducing T_s at the expense of more overhead, or by assigning an earlier deadline to the requests

TOTAL BANDWIDTH SERVER (TBS)

Total Bandwidth Server Requisites

The approach for the Total Bandwidth Server is based on:

Periodic tasks: scheduled based on EDF
arrive at t=0
$T_i = D_i$

Aperiodic tasks: are scheduled based on EDF
arrive at unknown time
T_i(minimum interarrival time)=D_i

All tasks can be preempted

Total Bandwidth Server

Used to improve the response time of aperiodic tasks due to the long period (long deadline) of DSS

It is implemented with a computation time C_s a pericd T_s and an earlier deadline d_s

The deadline computation must be assigned to avoid exceeding a maximum specified value U_s according to

$$d_k = \max(r_k, d_{k-1}) + \frac{C_k}{U_s}$$

where d_k is the deadline of the aperiodic request, d_{k-1} is the previous deadline, r_k is the arrival time, C_k is the computing time of the request, and U_s is the server utilization (or the bandwidth)

Once the deadline is assigned, the task goes to the EDF list as any other periodic task

Total Bandwidth Server Example

Task 1 and 2 are periodic

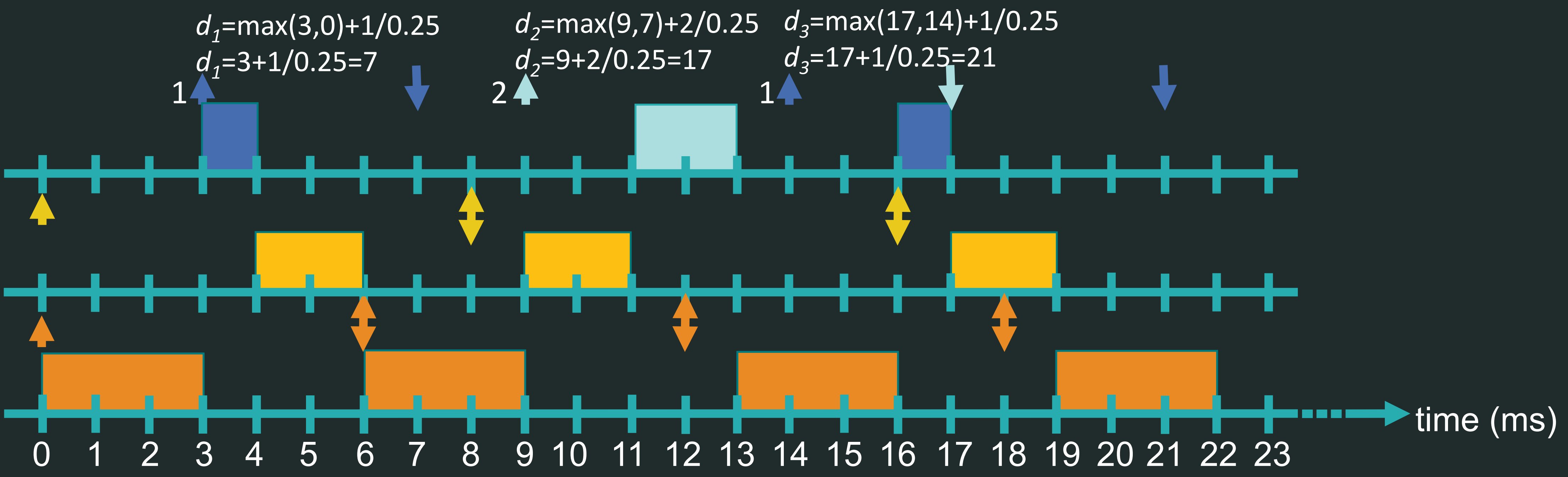
Red represents the server

Blue and cyan represent aperiodic tasks arriving at unknown time

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	3	6
τ_2	2	8

Server τ_3 → U_s=0.25

Note: 0.25=1-U_p=1-3/6-2/4



Total Bandwidth Server Schedulability

A periodic EDF with DSS is schedulable iff (necessary and sufficient condition)

$$U_p + U_s \leq 1$$

where U_p is the utilization factor of the n periodic tasks

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

where U_s is the utilization factor of the periodic server

$$U_s = \frac{C_s}{T_s}$$

Total Bandwidth Server

Pros:

Improves the aperiodic time response

Increases the CPU utilization

Schedulable iff (necessary and suficient condition)

$$U_p + U_s \leq 1$$

Cons:

Domino effect (missing deadlines) if some task lasts more than expected

CONSTANT BANDWIDTH SERVER (CBS)

Constant Bandwidth Server Requisites

The approach for the Constant Bandwidth Server is based on:

Periodic tasks: scheduled based on EDF
arrive at t=0
$T_i = D_i$

Aperiodic tasks: are scheduled based on EDF
arrive at unknown time
T_i(minimum interarrival time)=D_i

All tasks can be preempted

Constant Bandwidth Server

Used to avoid the domino effect caused by TBS during overrun by implementing a bandwidth reservation strategy that provides isolation between pure periodic and server tasks

Deadlines are computed as in TBS

The server is similar to TBS, but it is implemented with a maximum budget Q_s a period T_s and a server bandwidth U_s

The server tracks the budget, if it is exhausted, the deadline is postponed

Server rules:

Arrival of an aperiodic job: if (∃ pending aperiodic requests) then enqueue J_k

else if $q_s>(d_s-r_k)\cdot U_s$ then $q_s=Q_s$ //replenish the budget

$d_s=r_k+T_s$

Budget exhausted: if (budget is exhausted) then $q_s=Q_s$ //replenish the budget

$d_s=d_s+T_s$

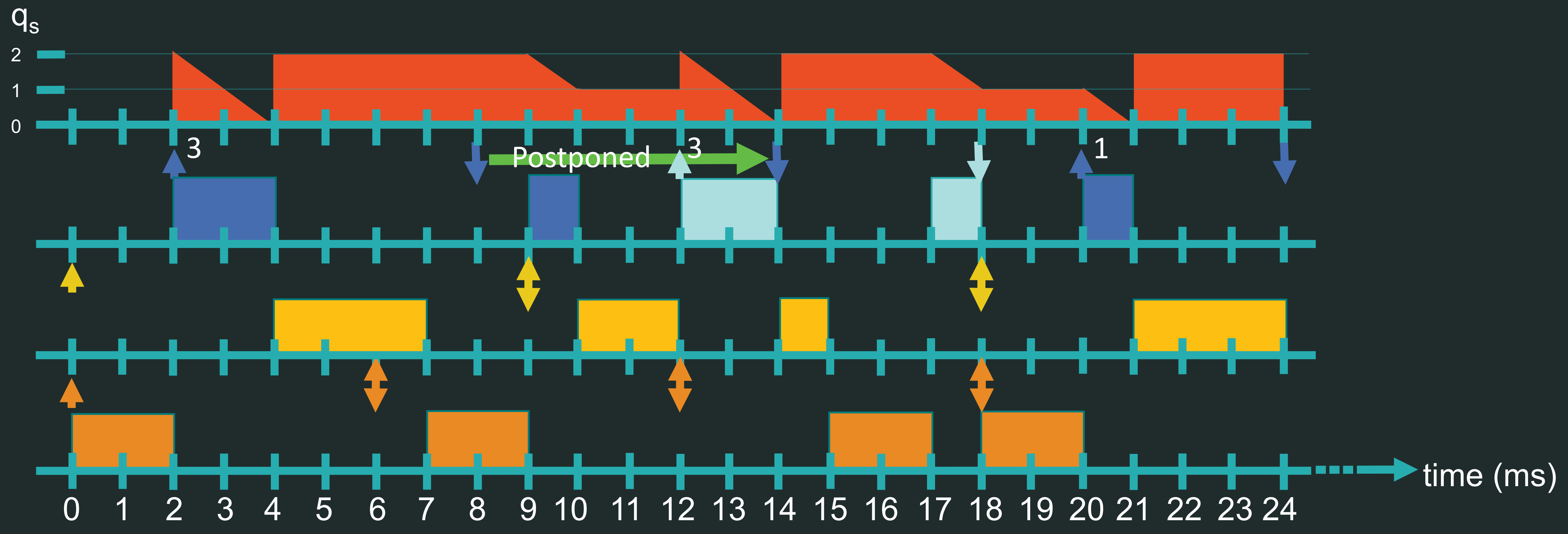
Constant Bandwidth Server Example

Task 1 and 2 are periodic

Red represents the server capacity

Blue and cyan represent aperiodic tasks arriving at unknown time

Task τ_i	Computing time c_i (ms)	Period T_i = Deadline D_i (ms)
τ_1	2	6
τ_2	3	9
Server τ_3	Qs=2	Ts=6



Constant Bandwidth Server Schedulability

A periodic EDF with CBS is schedulable iff (necessary and sufficient condition)

$$U_p + U_s \leq 1$$

where U_p is the utilization factor of the n periodic tasks

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

where U_s is the utilization factor of the periodic server

$$U_s = \frac{C_s}{T_s}$$

Constant Bandwidth Server

Pros:

The total utilization factor is no greater than U_s, even in the presence of overloads (this was the main problem of TBS)

Performs better than DSS, and similarly to TBS.

Inproves the CPU utilization

Schedulable iff (necessary and suficient condition)

$$U_p + U_s \leq 1$$

Cons:

Higher computational load

COMPARISON OF SERVERS

Comparing servers