Fixed-Priority Servers

CPEN 432 Real-Time System Design

Arpan Gujarati
University of British Columbia

Periodic Tasks with Background Workload

- Until now, we studied **homogeneous** set of tasks
  - All tasks are either periodic or aperiodic
- Typical real-time systems have **hybrid** task sets
  - Periodic tasks
    - Time-driven with regular activation rates
    - Hard timing constraints
    - Execute critical control activities
  - Aperiodic tasks
    - Event-driven
    - Hard, soft, or non-real-time requirements
    - E.g., monitoring, environment-driven, fault tolerance, etc.
- Twofold objectives
  - Guarantee the **schedulability** of all critical tasks in worst-case conditions
  - Provide **good average response times** for soft and non-real-time activities

Background Scheduling

Background Scheduling

- Simple design
  - Aperiodic tasks picked only if the periodic queue is empty
  - New periodic task immediately preempts aperiodic task
  - The guarantee test for periodic tasks does not change

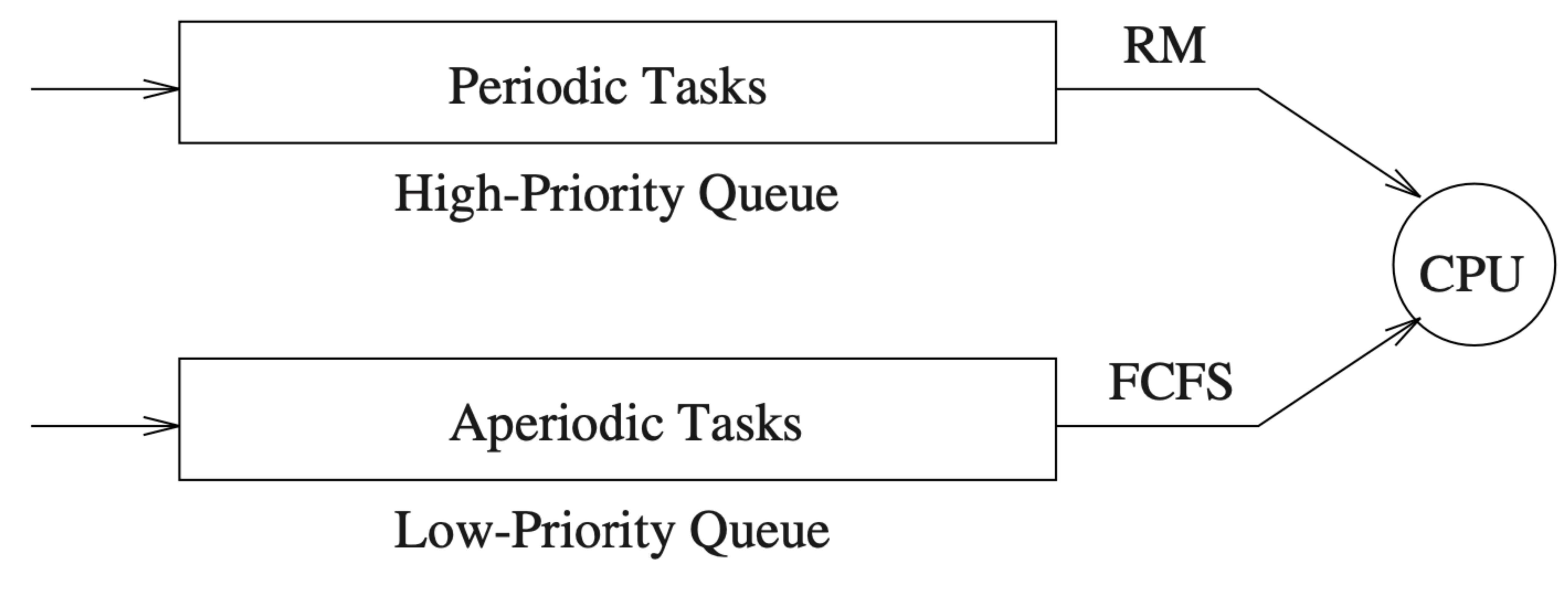
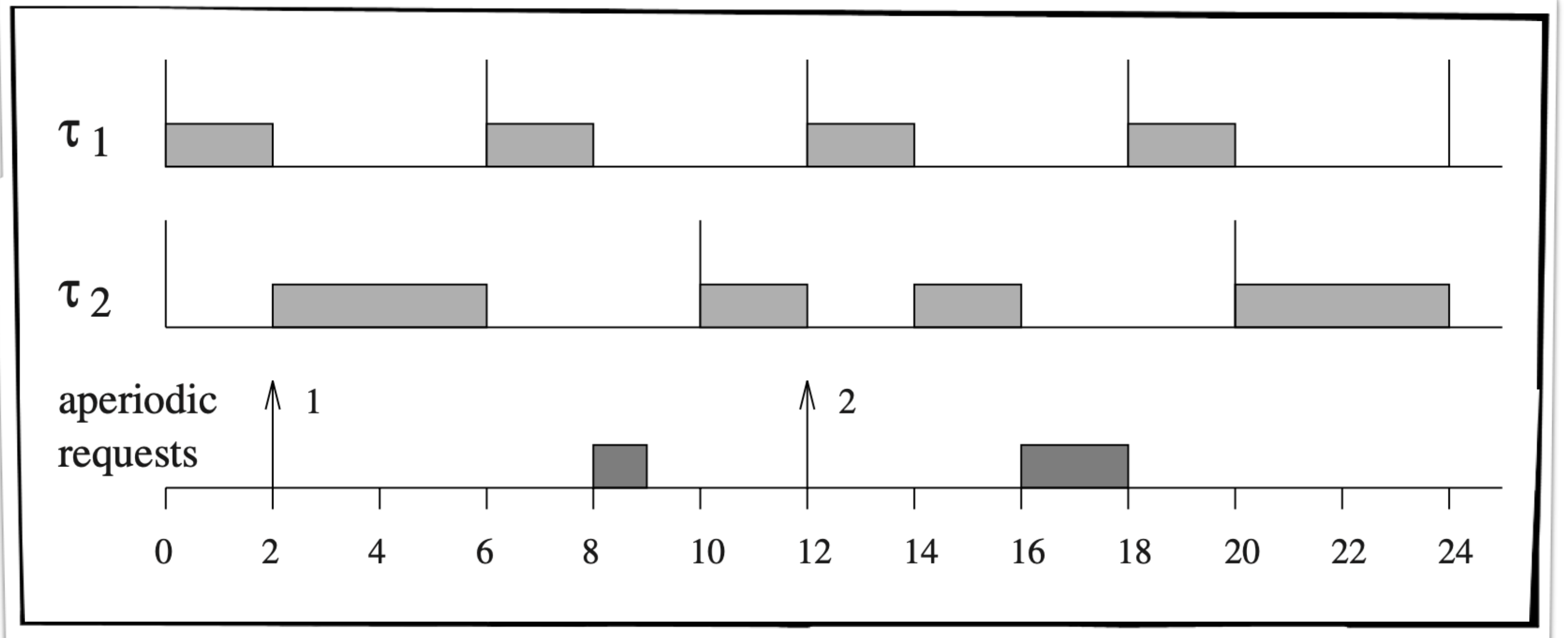


Figure 5.2 Scheduling queues required for background scheduling.

Example



Can we **further improve** the average response time of aperiodic jobs?

Polling Server

Periodic Task to Serve Aperiodic Jobs

- Task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\} \cup \{\tau_{polling}\}$
- Like any periodic task, $\tau_{polling}$ is characterized by $T_{polling}$ and $C_{polling}$
  - $C_{polling}$ is often referred to as **server capacity** or **server budget**
- Fixed-priority scheduling (RM, DM, etc.)
- When $\tau_{polling}$ is scheduled, i.e., when it becomes **active**
  - Schedules pending aperiodic jobs as long as $C_{polling}$ is not exhausted
  - If no pending aperiodic jobs, suspends itself until it activated again
  - Upon suspension, any pending budget is immediately discharged

Example (RM)

Periodic Tasks

	T_i	C_i	a_i
τ_1	4	1	0
τ_2	6	2	0

Polling Server

	$T_{polling}$	$C_{polling}$
$\tau_{polling}$	5	2

Aperiodic Jobs

	Workload	*Arrival*
J_1	2	2
J_2	1	8
J_3	2	12
J_4	2	19

Advantages

- Schedulability analysis
  - Plug in $T_{polling}$ and $C_{polling}$ into utilization-based or response time analyses
- Implementation

Dimensioning a Polling Server

- Given $\{\tau_1, \tau_2, \ldots, \tau_n\}$, how can we compute $T_{polling}$ and $C_{polling}$?
- Step 1: What is the maximum utilization $U_{polling} = \dfrac{C_{polling}}{T_{polling}}$?
  - Recall the hyperbolic bound $\prod_{i=1}^{n} (U_i + 1) \leq 2$
- Step 2: How can we compute $T_{polling}$ and $C_{polling}$?
  - Given an upper bound on $U_{polling}$, infinite possibilities!

Disadvantages

- Budget $C_{polling}$ is immediately discarded if no pending aperiodic jobs
  - Server capacity is wasted!
  - Average response time of aperiodic jobs may be unnecessarily high
    - E.g., a job that arrives immediately after the budget is discarded has to wait until the next time period

Deferrable Server

Similar to the Polling Server ...

- Task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\} \cup \{\tau_{deferrable}\}$
  - Like any periodic task, $\tau_{deferrable}$ is characterized by $T_{deferrable}$ and $C_{deferrable}$
- Fixed-priority scheduling (RM, DM, etc.)
- When $\tau_{deferrable}$ is scheduled, i.e., when it is becomes **active**
  - Schedules pending aperiodic jobs as long as $C_{deferrable}$ is not exhausted
  - ~~If no pending aperiodic jobs, suspends itself until it activated again~~
  - ~~Upon suspension, any pending budget is immediately discharged~~
  - If no pending aperiodic jobs, preserves budget until the end of the time period

Example (RM)

Periodic Tasks

	T_i	C_i	a_i
τ_1	4	1	0
τ_2	6	2	0

Polling Server

	$T_{polling}$	$C_{polling}$
$\tau_{polling}$	5	2

Aperiodic Jobs

	Workload	*Arrival*
J_1	2	2
J_2	1	8
J_3	2	12
J_4	2	19

Advantages, Disadvantages?

Sporadic Server

Best of Both Worlds …

- Task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\} \cup \{\tau_{sporadic}\}$
  - Like any periodic task, $\tau_{sporadic}$ is characterized by $T_{sporadic}$ and $C_{sporadic}$
- Fixed-priority scheduling (RM, DM, etc.)
- Like deferrable server, preserve the budget until an aperiodic job arrives
- Like polling server, ensure that task remains equivalent to the periodic task
  - Replenishes capacity only after it has been consumed by aperiodic job execution
- Replenishment protocol
  - If the current task has a lower priority, the sporadic server is **idle**, else it is **active**
  - If the sporadic server becomes active at time t_{active} and $C_{sporadic} > 0$ at that time
    - The next replenishment time of the server is set to $t_{replenishment} = t_{active} + T_{sporadic}$
  - The replenishment amount is decided at time $t_{idle\_or\_exhausted}$ when the server is idle again or its budget has exhausted
    - The replenishment amount is the capacity consumed in the interval $[t_{active}, t_{idle\_or\_exhausted}]$

Example (RM)

Periodic Tasks

	T_i	C_i	a_i
τ_1	10	3	0
τ_2	15	4	0

Polling Server

	$C_{polling}$	$T_{polling}$
$\tau_{polling}$	2	8

Aperiodic Jobs

	Workload	*Arrival*
J_1	2	2
J_2	2	7

Example (RM)

Periodic Tasks

	T_i	C_i	a_i
τ_1	5	1	0
τ_2	15	4	0

Polling Server

	$C_{polling}$	$T_{polling}$
$\tau_{polling}$	5	10

Aperiodic Jobs

	Workload	*Arrival*
J_1	2	4
J_2	2	8

Advantages

- The replenishment rule compensates for any deferred execution
  - From a scheduling point of view, sporadic server task is a **normal periodic task**
  - Dimensioning a sporadic server is **similar to a polling server**

Dynamic-Priority Servers

CPEN 432 Real-Time System Design

Arpan Gujarati
University of British Columbia

Dynamic Sporadic Server

Protocol

- Task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\} \cup \{\tau_{sporadic}\}$
  - Like any periodic task, $\tau_{sporadic}$ is characterized by $T_{sporadic}$ and $C_{sporadic}$
- Replenishment protocol
  - If an aperiodic job is pending, the sporadic server is **active**, else it is **idle**
  - If the sporadic server becomes active at time t_{active} and $C_{sporadic} > 0$ at that time
    - The next replenishment time of the server is set to $t_{replenishment} = t_{active} + T_{sporadic}$
    - The absolute deadline of the server is also set to $d_{sporadic} = t_{active} + T_{sporadic}$
  - The replenishment amount is decided at time $t_{idle\_or\_exhausted}$ when the server is idle again or its budget has exhausted
    - The replenishment amount is the capacity consumed in the interval $[t_{active}, t_{idle\_or\_exhausted}]$
- The dynamic sporadic server behaves **like a periodic task**
  - Schedulability analysis: $U_{periodic\_tasks} + U_{sporadic\_server} \leq 1$

Example (RM)

Periodic Tasks

	T_i	C_i	a_i
τ_1	8	2	0
τ_2	12	3	0

Polling Server

	$C_{polling}$	$T_{polling}$
$\tau_{polling}$	5	10

Aperiodic Jobs

	Workload	*Arrival*
J_1	2	3
J_2	2	6
J_3	2	14
J_4	1	15

Constant Bandwidth Server

Protocol

- A CBS is characterized by a budget C_{CBS} and by an ordered pair (Q_{CBS}, T_{CBS})
  - Q_{CBS} is the maximum budget and T_{CBS} is the period of the server
  - The ratio $U_{CBS} = C_{CBS}/T_{CBS}$ is denoted as the server bandwidth
  - At any time, CBS has a fixed deadline $d_{CBS,k}$ (initially, $d_{CBS,0} = 0$)

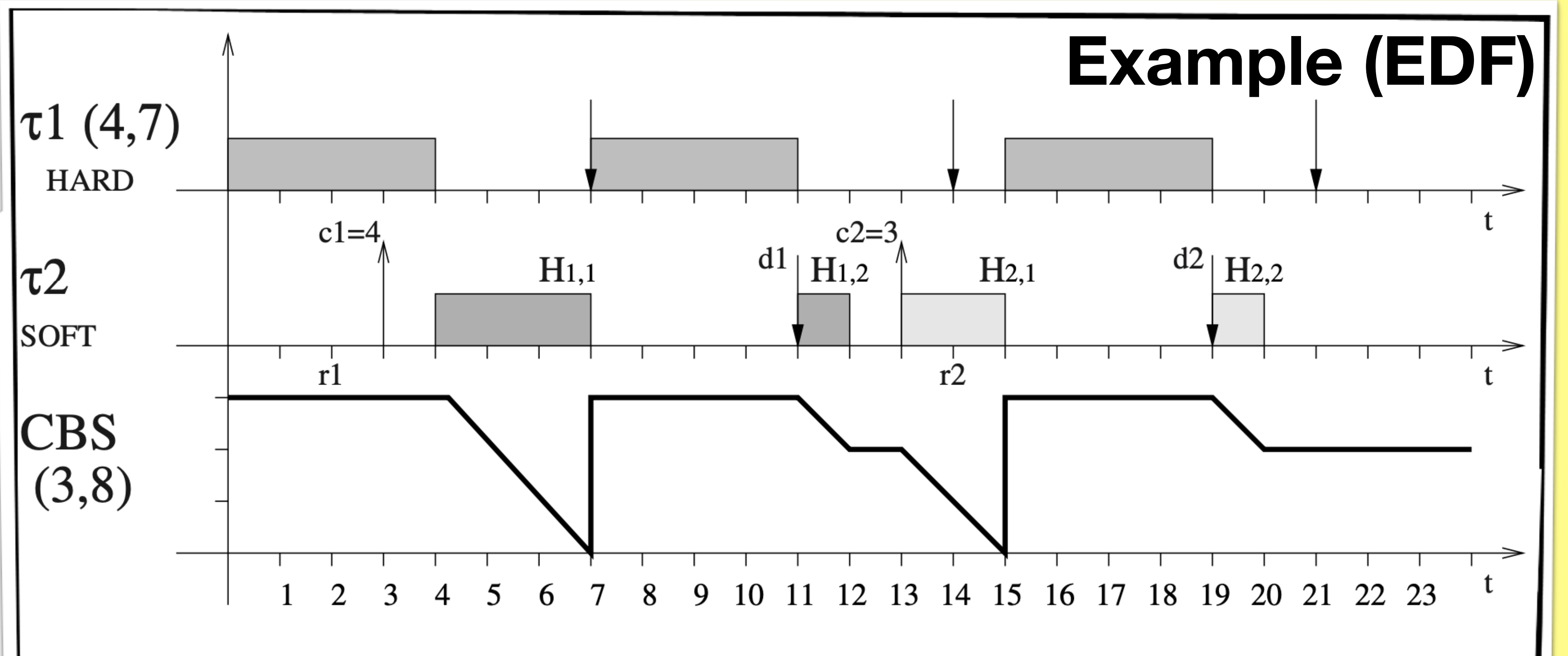


Figure 6.14 An example of CBS scheduling.

- Whenever $C_{CBS} = 0$
  - The server budget is recharged at the maximum value Q_{CBS}
  - A new server deadline is generated as $d_{CBS,k+1} = d_{s,k} + T_{CBS}$
  - As a result, there are no finite intervals of time in which the budget is equal to zero

- When a job J_i arrives and there are no other pending jobs (CBS is idle)
  - If $C_{CBS} \geq (d_{s,k} - a_i)U_{CBS}$, the deadline is updated to $d_{CBS,k+1} = a_i + T_{CBS}$ and C_{CBS} is recharged to Q_{CBS}
  - Otherwise, the job is served with the last server deadline $d_{CBS,k}$ using the current budget