Inovace tohoto kurzu byla v roce 2011/12 podpořena projektem CZ.2.17/3.1.00/33274 financovaným Evropským sociálním fondem a Magistrátem hl. m. Prahy.

Evropský sociální fond
Praha & EU: Investujeme do vaší budoucnosti

Embedded and Real-Time Systems

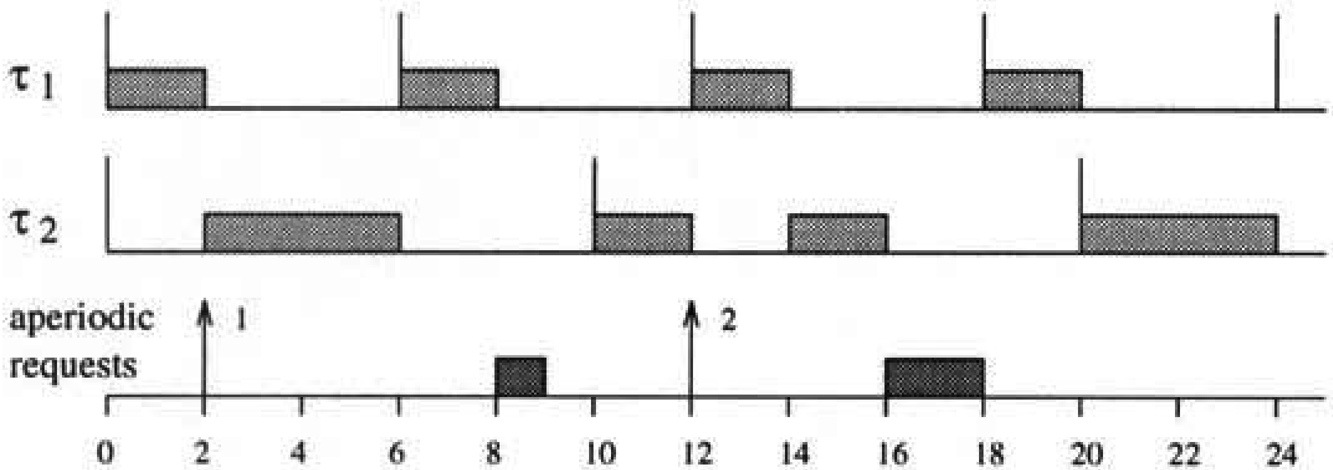
Fixed-Priority Servers

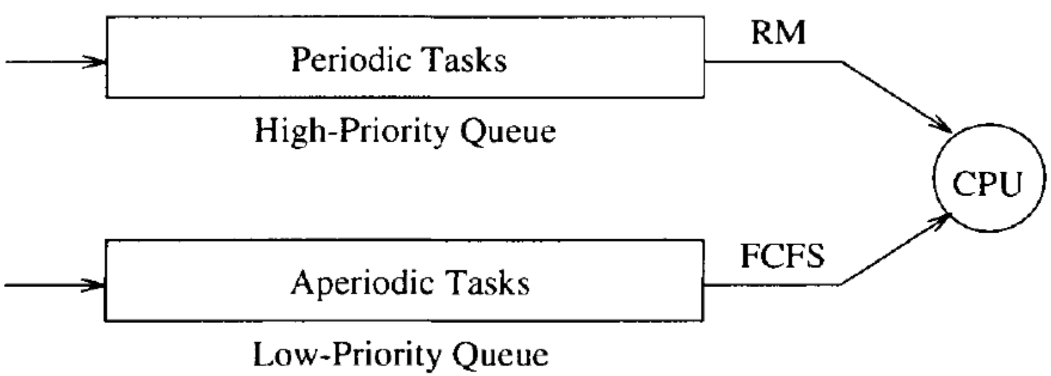
Scheduling of Aperiodic Tasks

- Often necessary to combine hard real-time periodic load with aperiodic ones
  - To provide off-line guarantee for aperiodic task, we need to know minimal inter-arrival time between its instances (i.e. sporadic task)
  - Aperiodic tasks requiring on-line guarantee on individual instances are called firm.
- In this part we assume
  - Periodic tasks scheduled by RM
  - All periodic tasks start at t=0 and relative deadlines are equal to periods
  - Minimum interarrival time is equal to the relative deadline
  - All tasks are fully preemptable

Background Scheduling

- Aperiodic load is scheduled when no periodic load is running





Figures taken from Buttazzo, G.:Hard Real-Time Computing Systems

Polling Server

- Aperiodic requests are processed in a task, which is scheduled according to RM

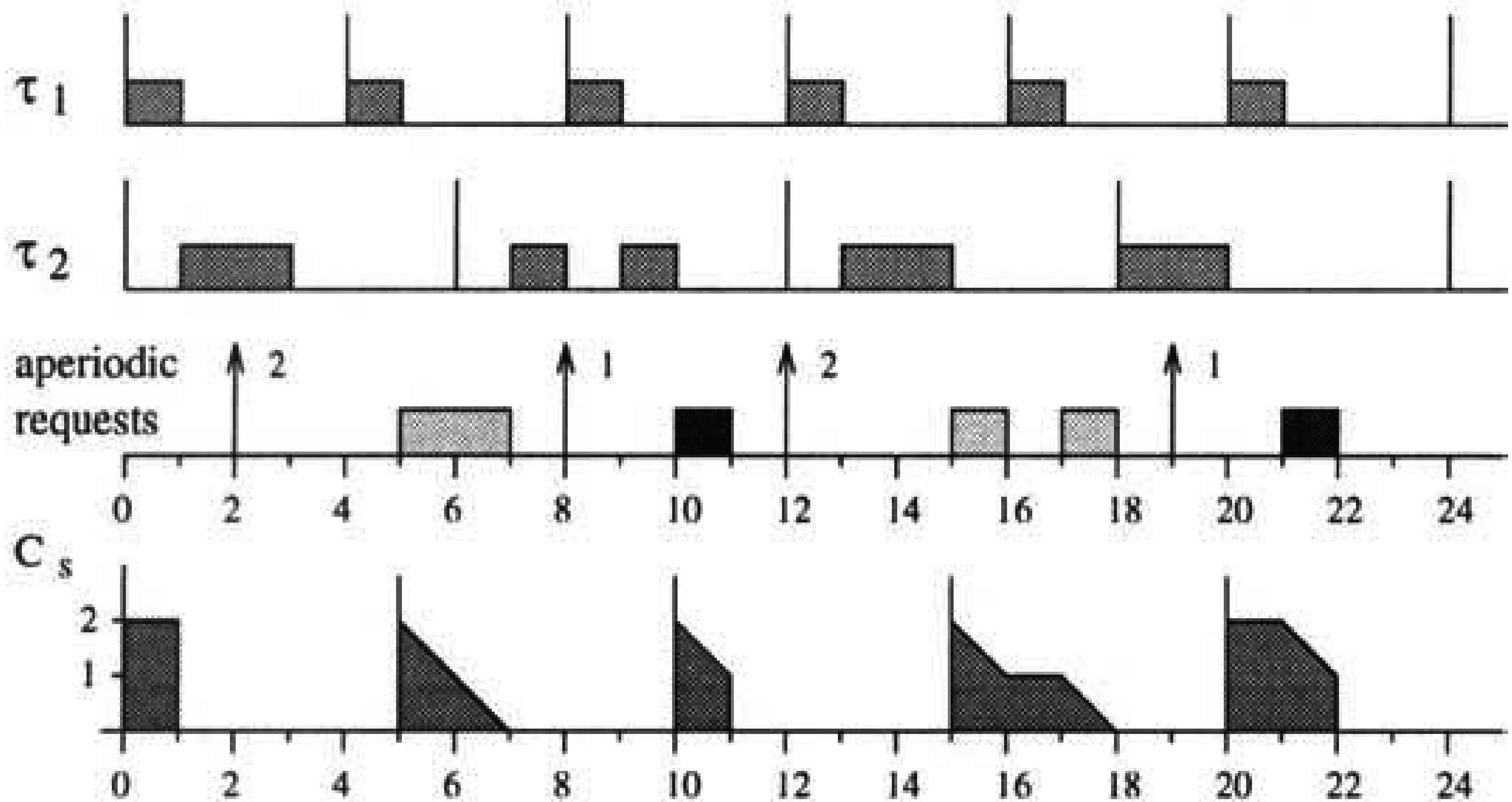


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Properties of Polling Server

- Improves the average response time of Background Scheduling
- Aperiodic load is constrained by the capacity of the server
  - Inside the server, it may be scheduled according to different criteria
- Possible to have more servers for aperiodic loads of different criticality
- If aperiodic load arrives after the start of the execution of the server, it has to wait till the next period

Polling Server – Analysis

- Schedulability analysis same as for RM
  - $\sum_{i=1}^{n} \frac{C_i}{T_i} + \frac{C_s}{T_s} \leq (n+1)[2^{1/(n+1)} - 1]$
- Aperiodic Guarantee
  - The task may wait up to one period until it starts executing
  - Further it is unknown where in the period it will execute

$$T_s + \left\lceil \frac{C_a}{C_s} \right\rceil T_s \leq D_a$$

Aperiodic Guarantee

- If the polling server is the highest priority task, the aperiodic guarantee for a task can be made more precise

$$F_a = \left\lfloor \frac{C_a}{C_s} \right\rfloor \qquad G_a = \left\lceil \frac{r_a}{T_s} \right\rceil \qquad R_a = C_a - F_a C_s$$

  - guarantee: $G_a T_s + F_a T_s + R_a \leq d_a$

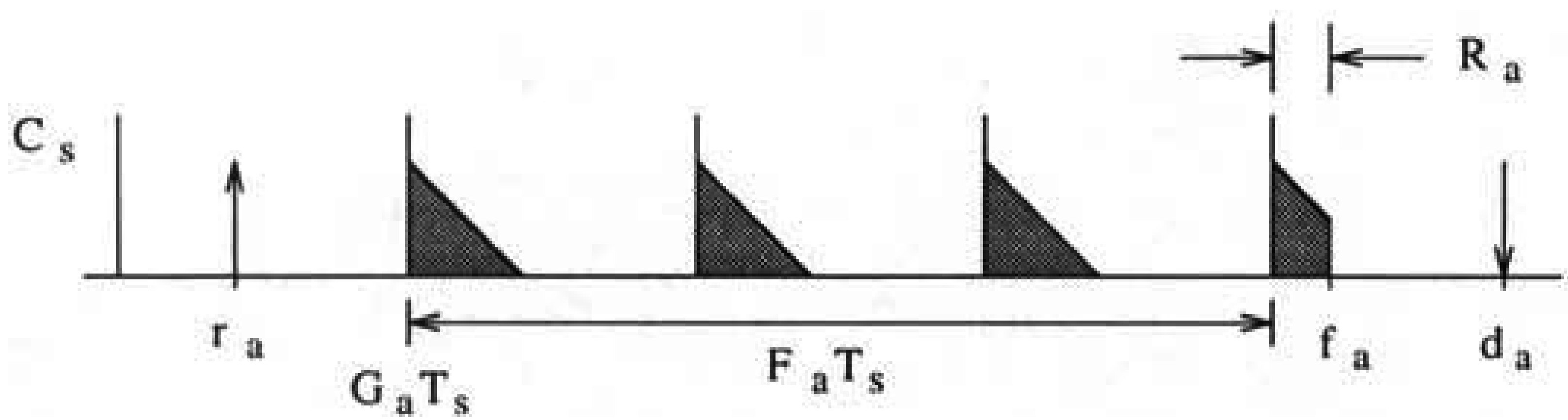


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Aperiodic Guarantee

- To generalize it on the set of firm tasks
  - the aperiodic computation to be processed at time *t*: $C_{ape}(t, d_k) = \sum_{i=1}^{k} c_i(t)$
- Guarantee: $f_k \leq d_k \quad \forall k = 1, \ldots, n$
  - where

$$f_k = \begin{cases} t + C_{ape}(t, d_k) & \text{if } C_{ape}(t, d_k) \leq c_s(t) \\ (F_k + G_k)T_s + R_k & \text{otherwise} \end{cases}$$

$$G_a = \left\lceil \frac{t}{T_s} \right\rceil$$

$$R_a = C_{ape}(t, d_k) - c_s - F_k C_s$$

$$F_a = \left\lfloor \frac{C_{ape}(t, d_k) - c_s(t)}{C_s} \right\rfloor$$

Deferrable Server

- Similar to polling server, but preserves capacity till the next period
  - I.e. if a task arrives after start of period, it is still serviced

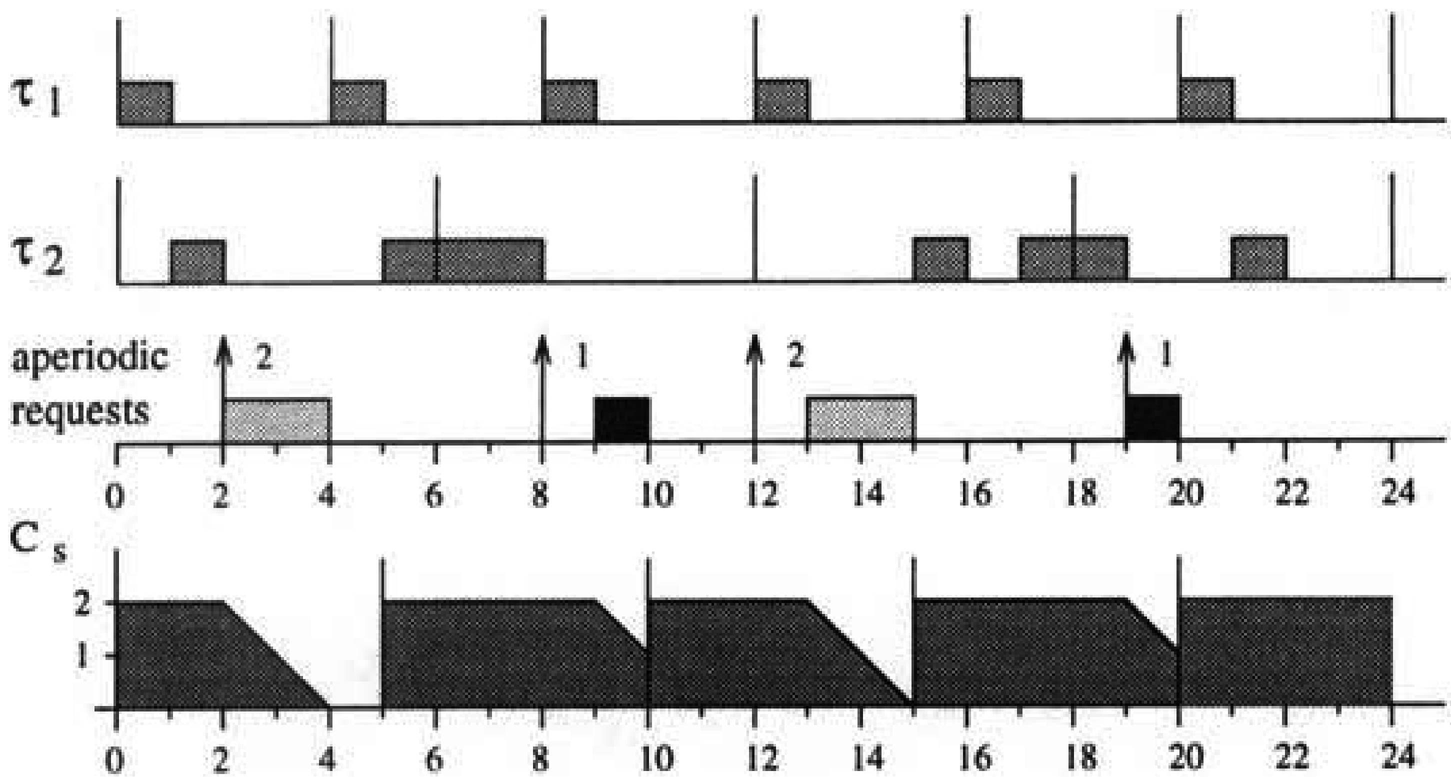


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Negative Impact on Schedulability

RM

DS

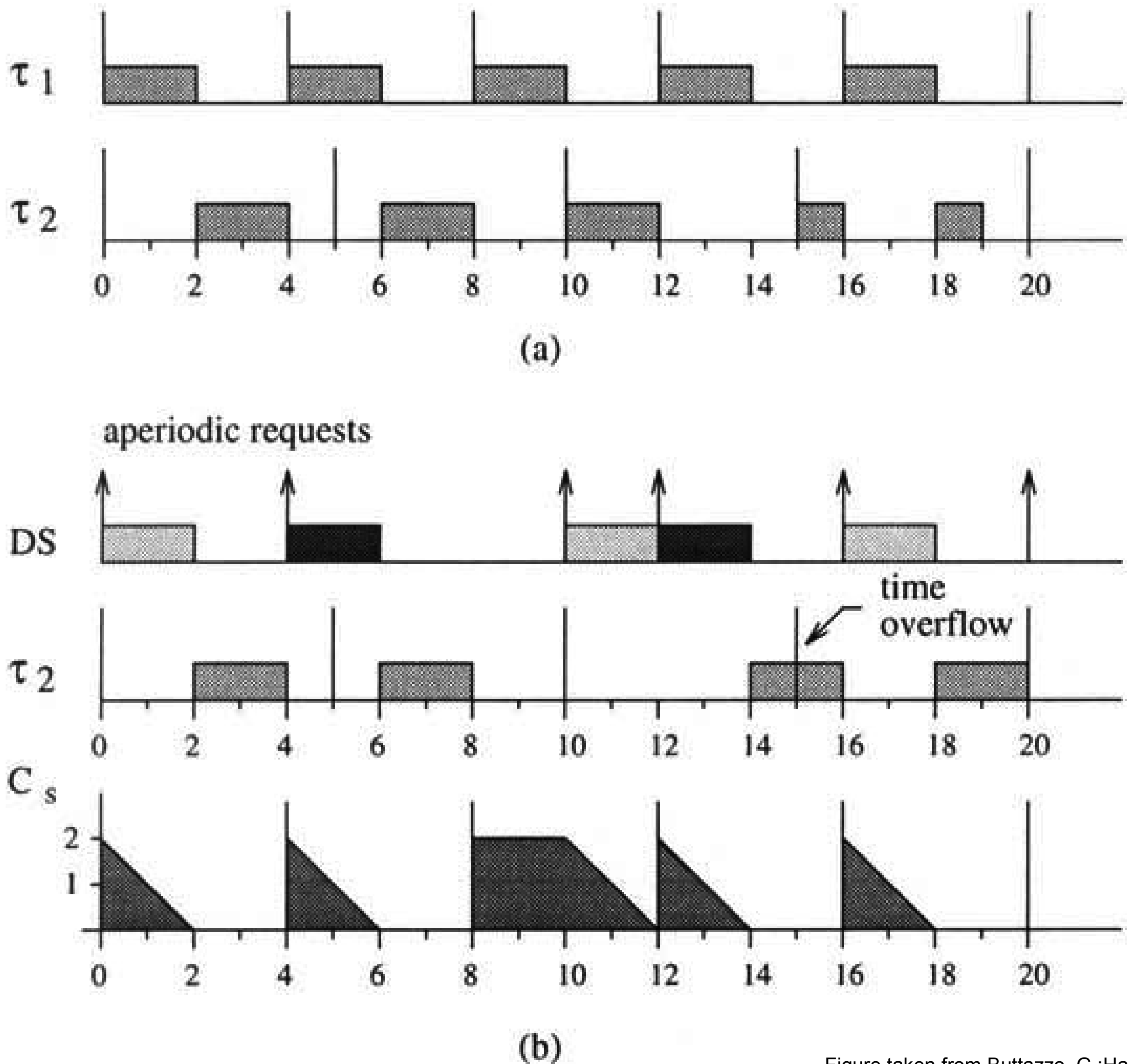


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Schedulability of DS

- Calculation of U_{lub} is different from RM

$$\lim_{n \to \infty} U_{lub} = U_s + \ln\left(\frac{U_s + 2}{2U_s + 1}\right)$$

  - thus:

$$U_p \leq \ln\left(\frac{U_s + 2}{2U_s + 1}\right)$$

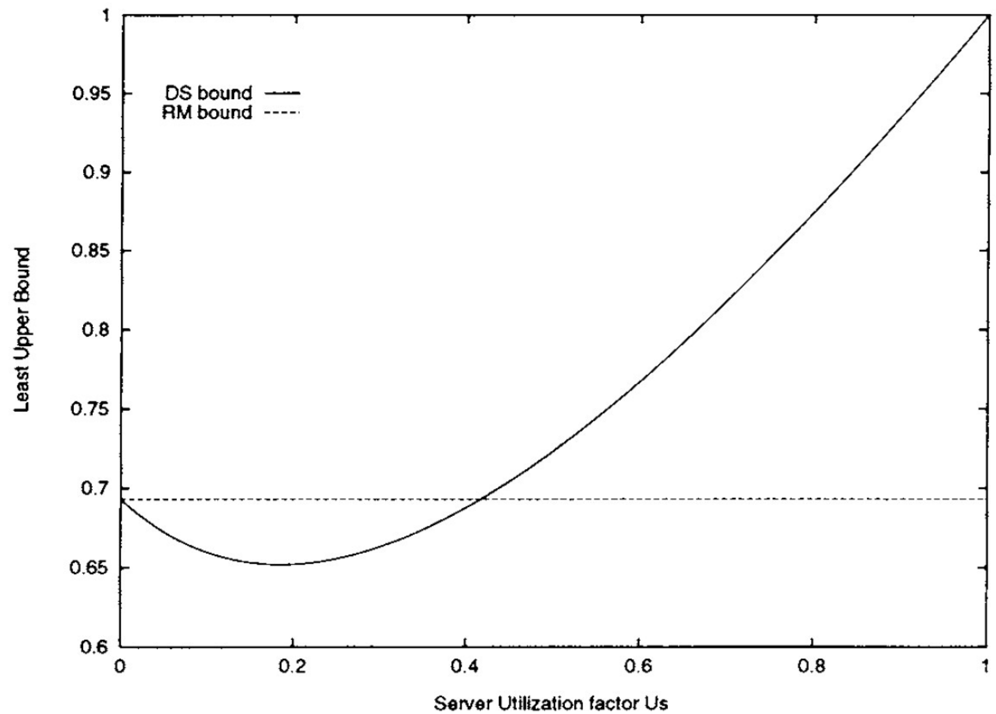


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Aperiodic Guarantee

- For one task guaranteed if and only if $f_a \leq r_a + D_a$
  - portion of J_a executed in the current server period:

$$\Delta_a = \min[c_s(t), (G_a T_s - r_a)]$$

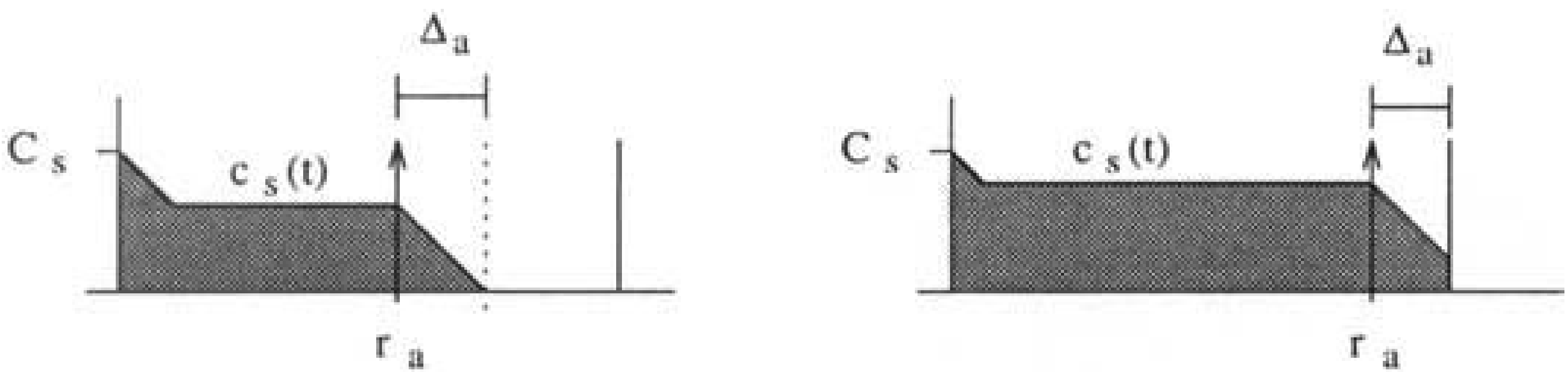


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Aperiodic Guarantee

- For one task guaranteed if and only if $f_a \leq r_a + D_a$
  - f_a is derived as follows:
  - $$f_a = \begin{cases} r_a + C_a & \text{if } C_a \leq c_s(t) \\ (F_a + G_a)T_s + R_a & \text{otherwise} \end{cases}$$
  - $F_a = \left\lfloor \frac{C_a - \Delta_a}{C_s} \right\rfloor$ $\quad G_a = \left\lceil \frac{r_a}{T_s} \right\rceil$ $\quad R_a = C_a - \Delta_a - F_a C_s$

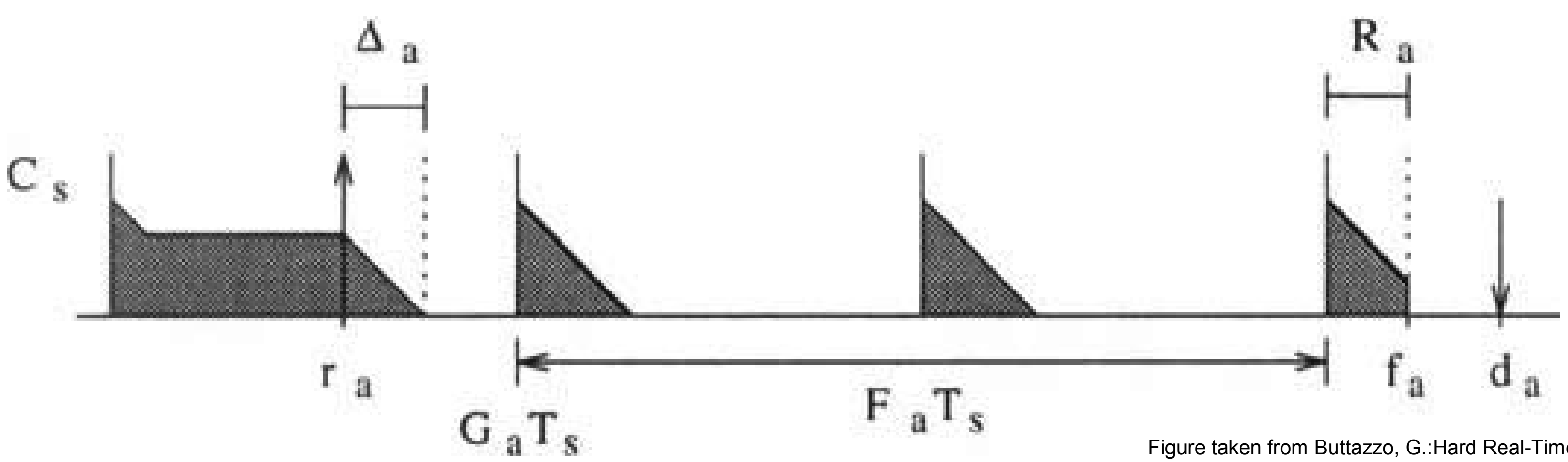


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Aperiodic Guarantee

- To generalize it on the set of firm tasks
  - the aperiodic computation to be processed at time *t*:

$$C_{ape}(t, d_k) = \sum_{i=1}^{k} c_i(t)$$

- Guarantee: $f_k \leq d_k \quad \forall k = 1, \ldots, n$
  - where

$$f_k = \begin{cases} t + C_{ape}(t, d_k) & \text{if } C_{ape}(t, d_k) \leq c_s(t) \\ (F_k + G_k)T_s + R_k & \text{otherwise} \end{cases}$$

$$G_a = \left\lceil \frac{t}{T_s} \right\rceil \qquad \Delta_a = \min[c_s(t), (G_a T_s - r_a)]$$

$$F_a = \left\lfloor \frac{C_{ape}(t, d_k) - \Delta_k}{C_s} \right\rfloor \qquad R_a = C_{ape}(t, d_k) - \Delta_k - F_k C_s$$

Priority Exchange Server

- Server replenishes its capacity to full value at beginning of each period
- It preserves its high-priority capacity by exchanging it for the execution time of a lower-priority periodic task
- The capacity is either eventually used for an aperiodic task or it degrades to priority level of background processing

- Compared to DS, it has worse responsiveness for aperiodic tasks but better schedulability bound for periodic tasks

Priority Exchange – Example 1

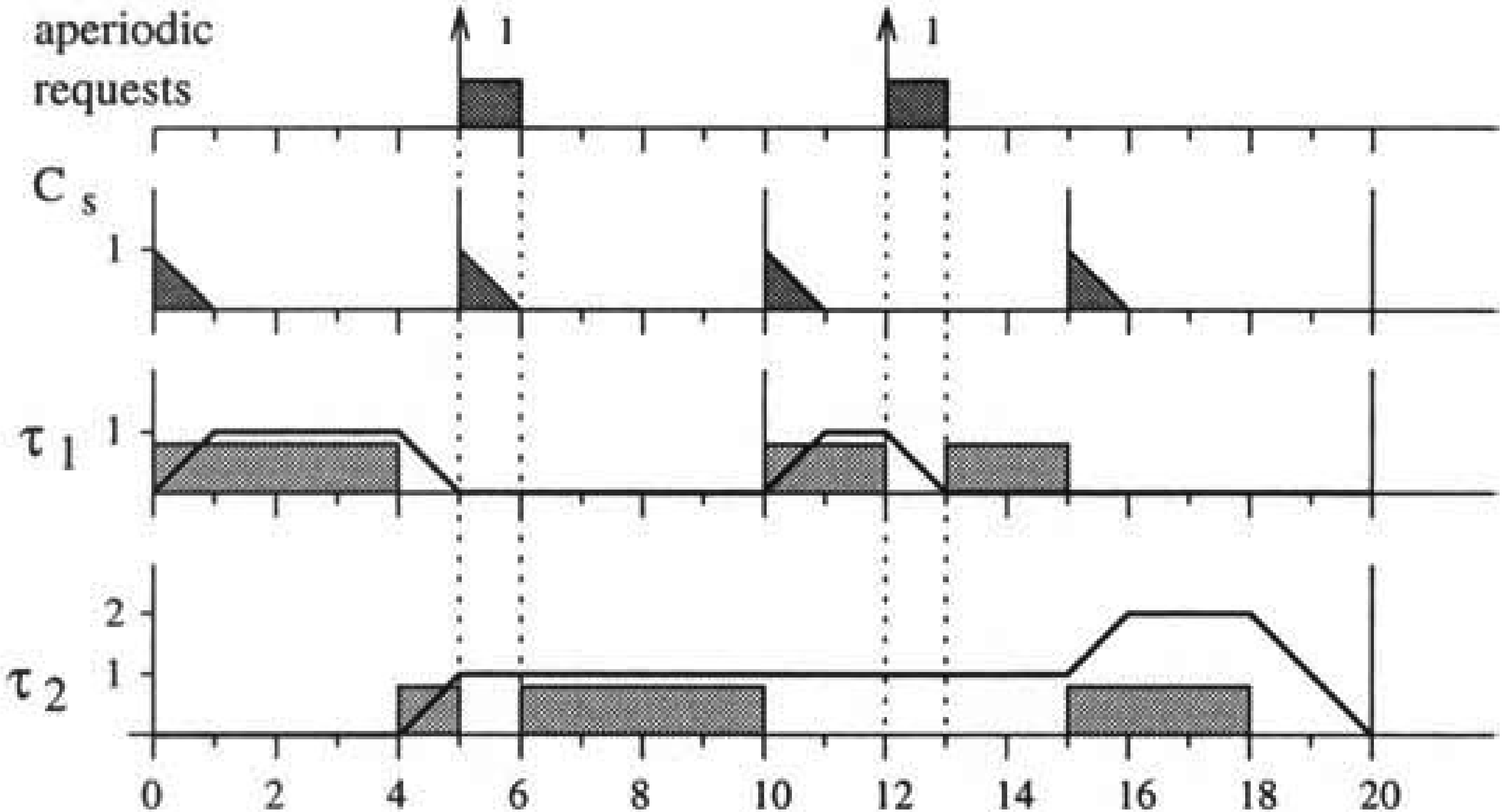


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Priority Exchange – Example 2

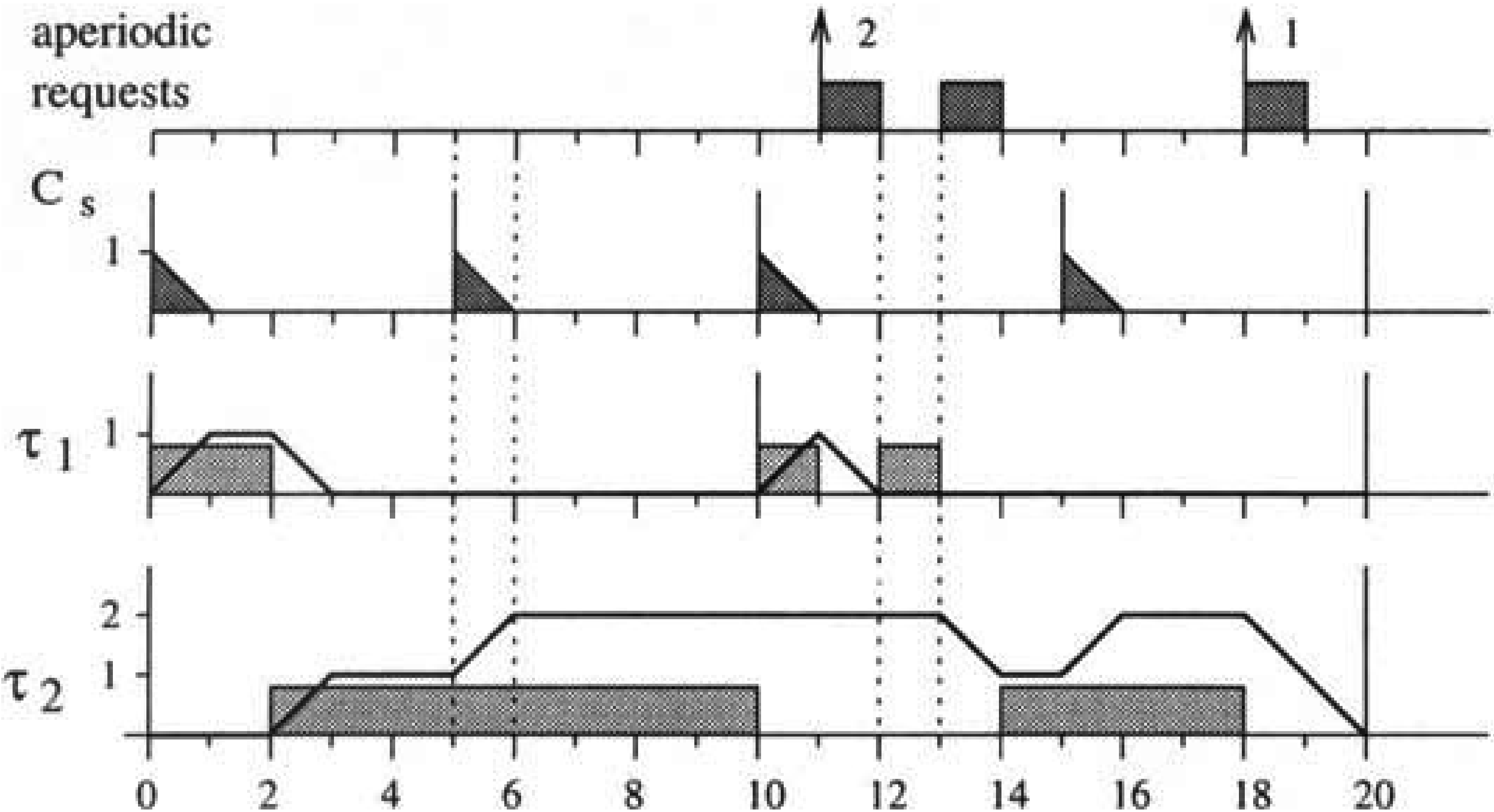


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Schedulability of PE

- Calculation of U_{lub}

$$\lim_{n \to \infty} U_{lub} = U_s + \ln\left(\frac{2}{U_s + 1}\right)$$

  - thus:

$$U_p \leq \ln\left(\frac{2}{U_s + 1}\right)$$

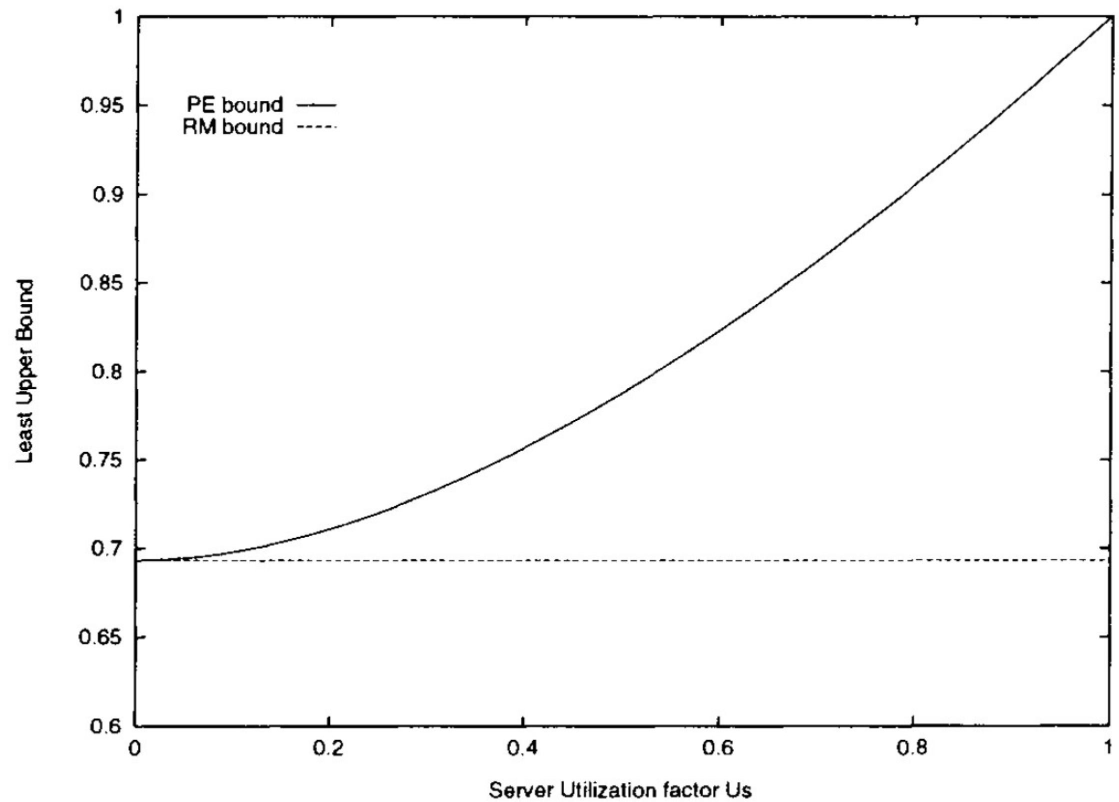


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Comparison of DS and PE

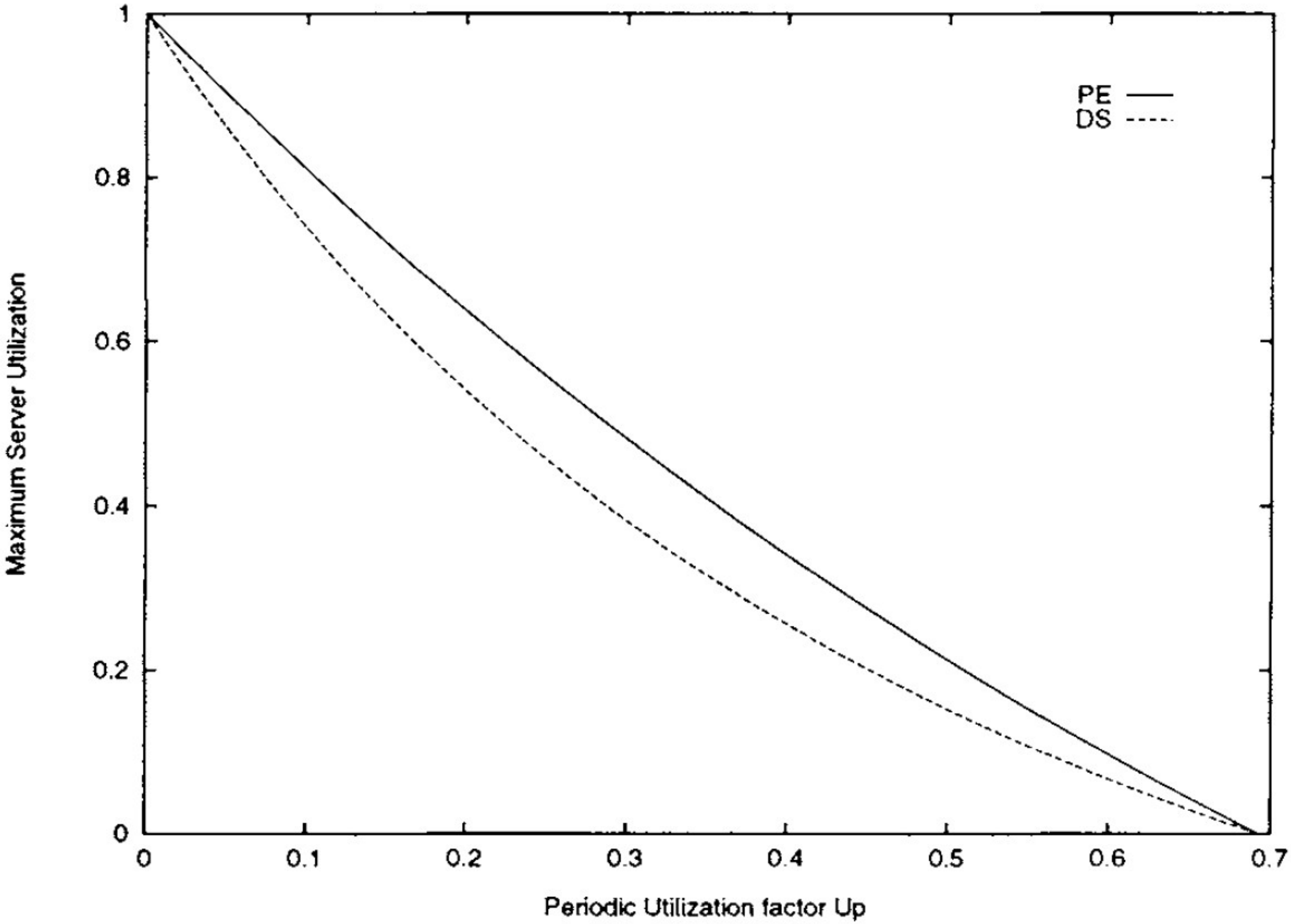


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Sporadic Server

- Enhances average response time of aperiodic tasks without degrading the utilization bound of the periodic task set
- SS creates a high-priority task for servicing aperiodic requests (like DS)
- Replenishes capacity only after it has been consumed by aperiodic task execution
  - The replenishement time RT is set as soon as SS becomes active and $C_s > 0$. Let t_A be such a time. The value of RT is set to $t_A + T_S$.
  - Replenishement amount RA to be done at RT is computed when SS becomes idle or C_S has been exhausted (t_I is such time) RA is set to the capacity consumed within $[t_A, t_I]$

Medium Priority SS – Example

	C_i	T_i
τ_1	1	5
τ_2	4	15

Server

$C_s = 5$

$T_s = 10$

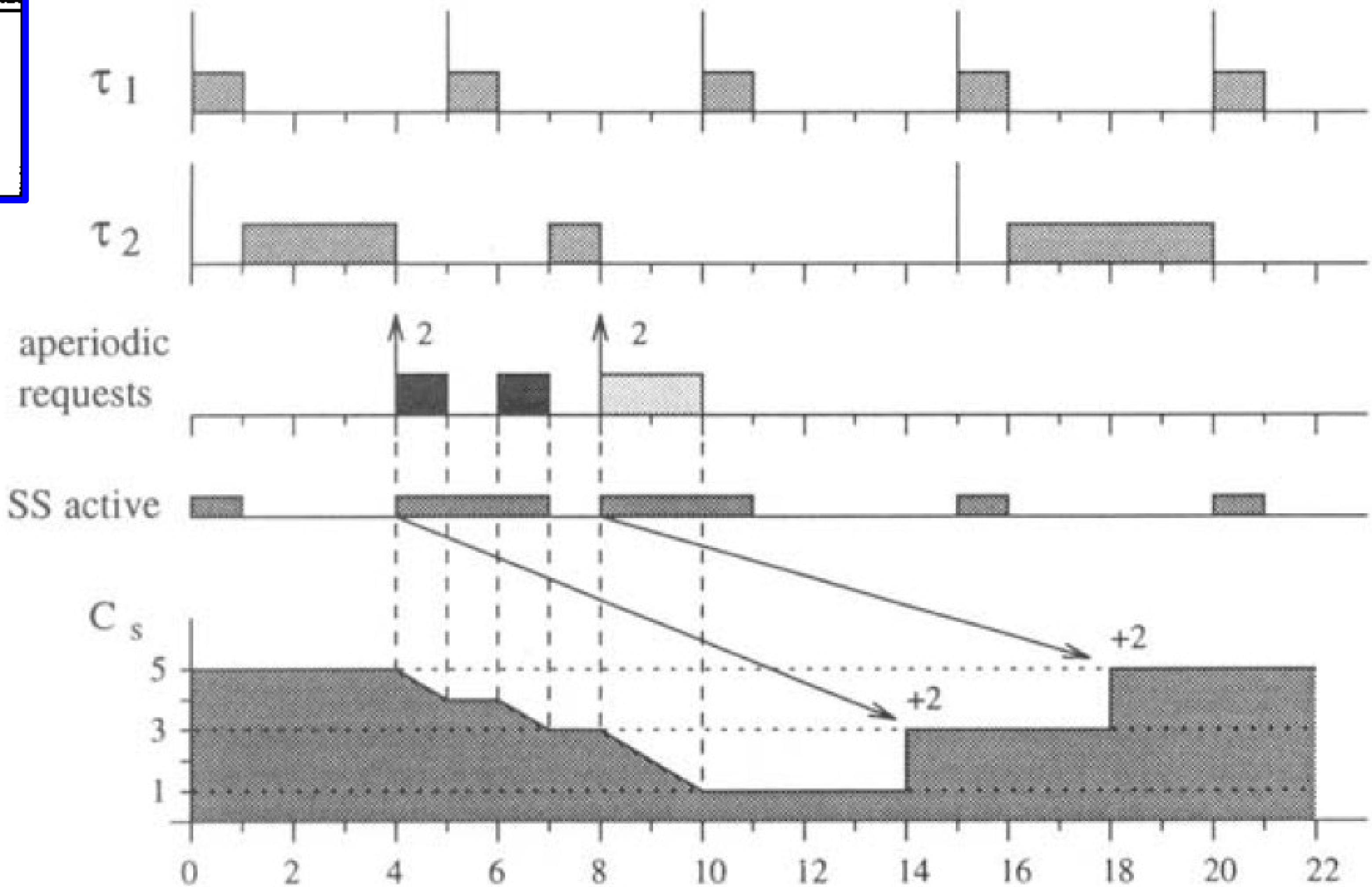


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

High Priority SS – Example

Server

$C_s = 2$

$T_s = 8$

	C_i	T_i
τ_1	3	10
τ_2	4	15

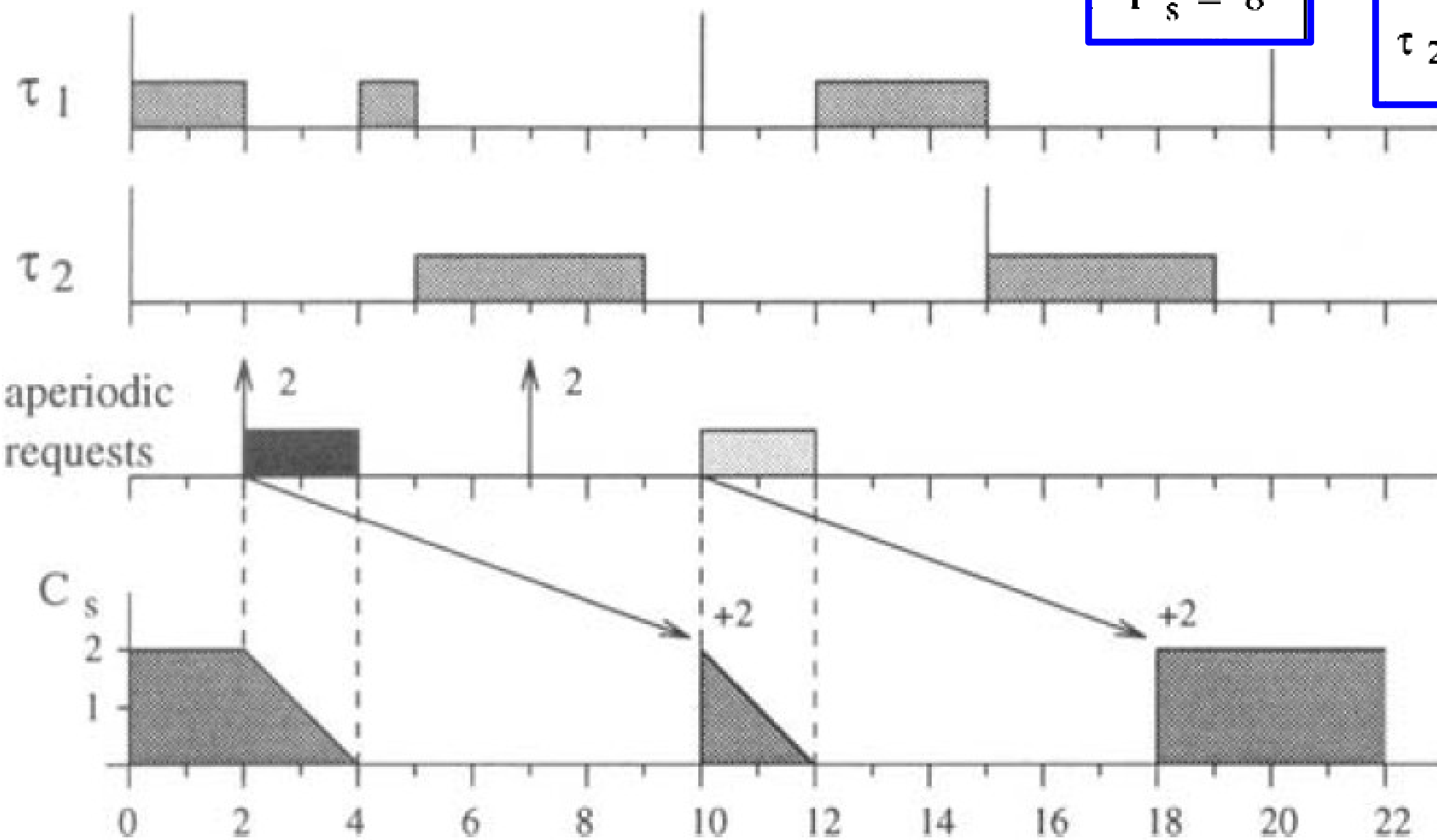


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Schedulability of SS

- Sporadic server violates the assumption that it runs if it can run.
- However, from schedulability point of view SS can be treated as standard periodic task.

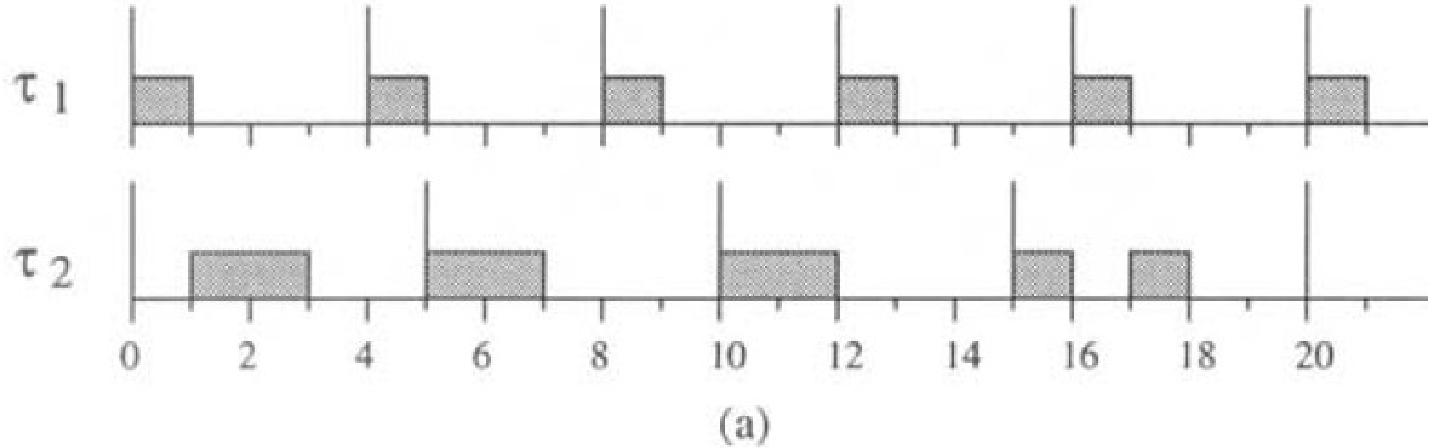
$$U_p \leq \ln\left(\frac{2}{U_s + 1}\right)$$

Slack Stealing

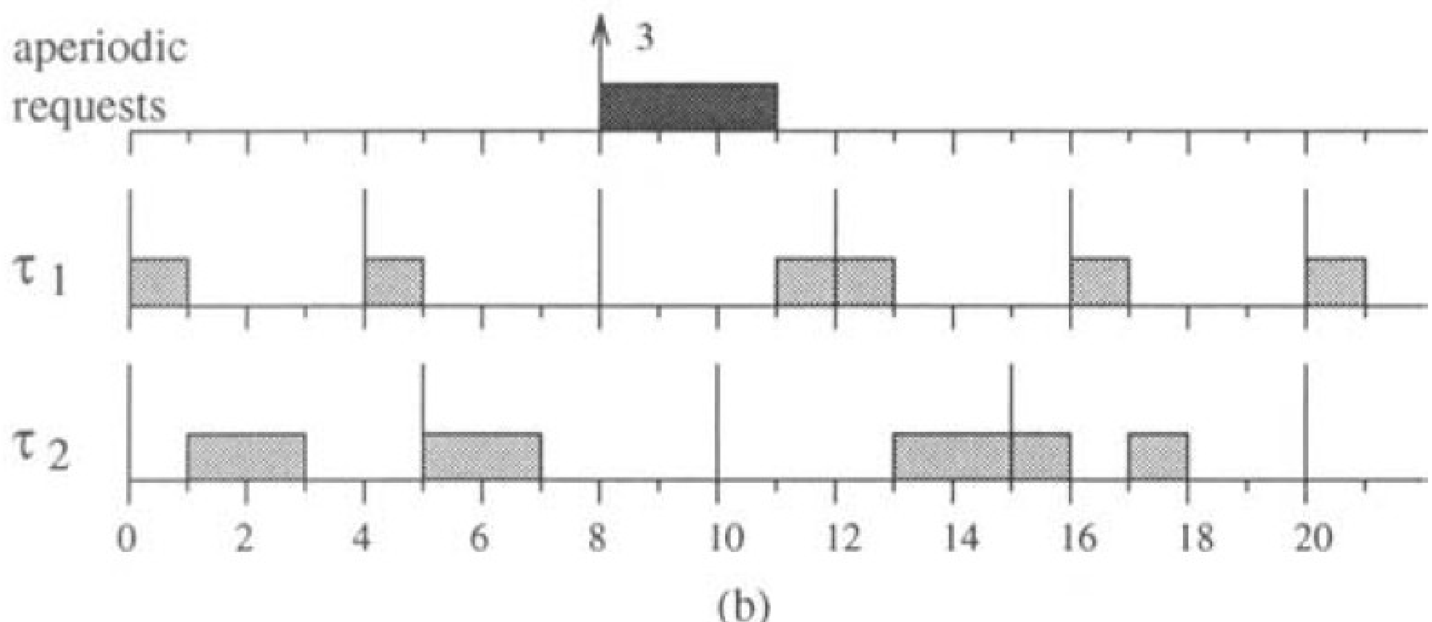
- Improvements of response time
- Creates a passive task (Slack Stealer)
  - Steals time from periodic tasks without causing deadlines to be missed
  - Works with the idea that there is typically no benefit in early completion of periodic tasks
  - Available slack computed as

$$slack_i(t) = d_i - t - c_i(t)$$

Slack Stealing – Example



(a)



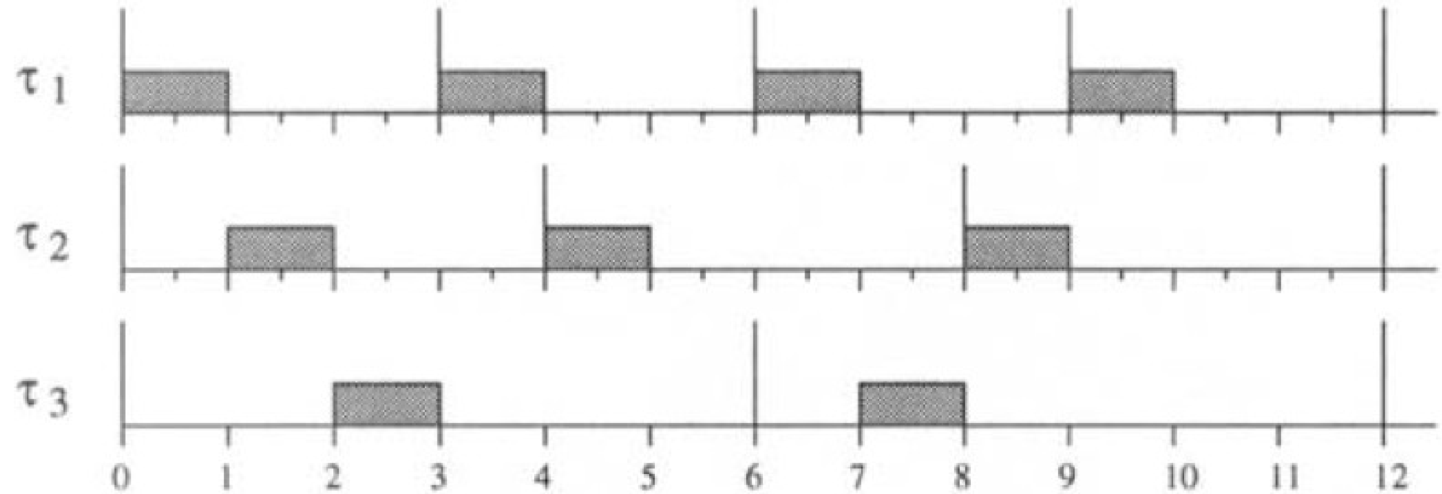
(b)

Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Non-Existence of Optimal Servers

- *For any set of periodic tasks ordered on a given fixed-priority scheme and aperiodic requests ordered according to a given aperiodic queueing discipline, there does not exist any valid algorithm that minimizes the response time of every soft aperiodic request.*
  - Applies to both clairvoyant and on-line algorithms

- *For any set of periodic tasks ordered on a given fixed-priority scheme and aperiodic requests ordered according to a given aperiodic queueing discipline, there does not exist any on-line valid algorithm that minimizes the average response time of soft aperiodic requests.*



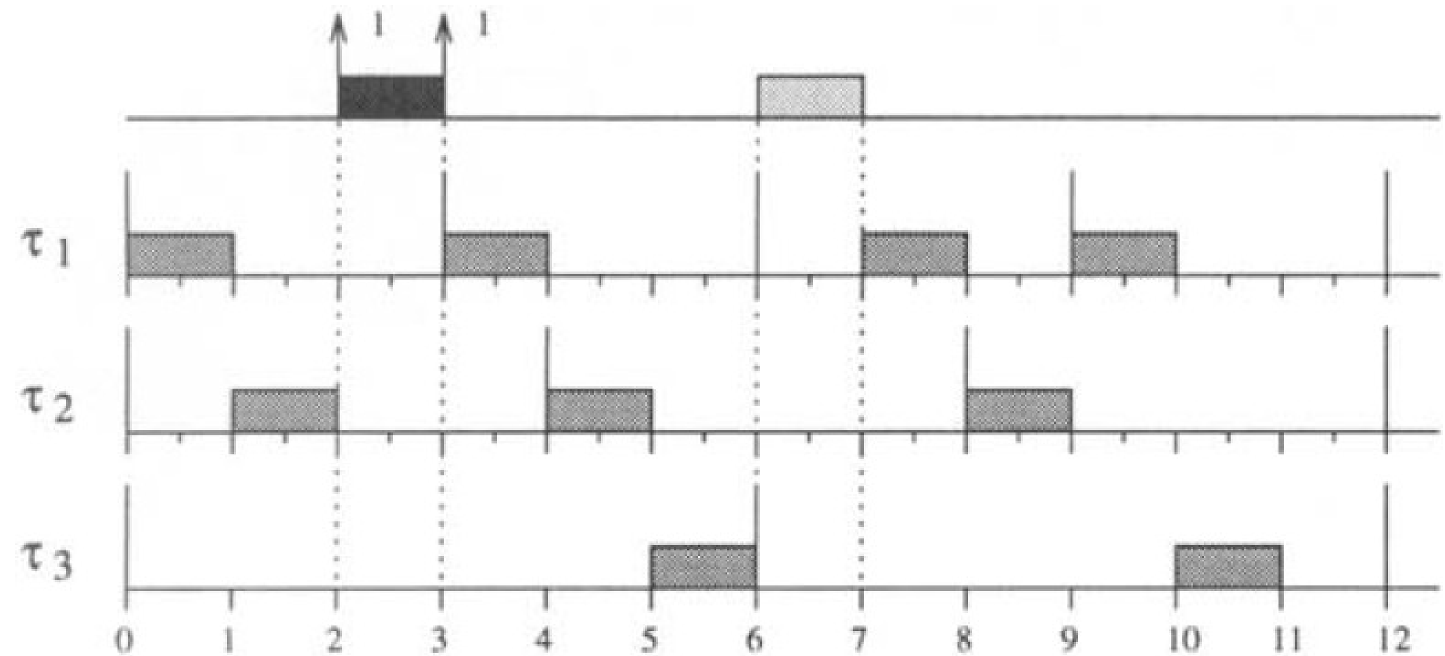


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

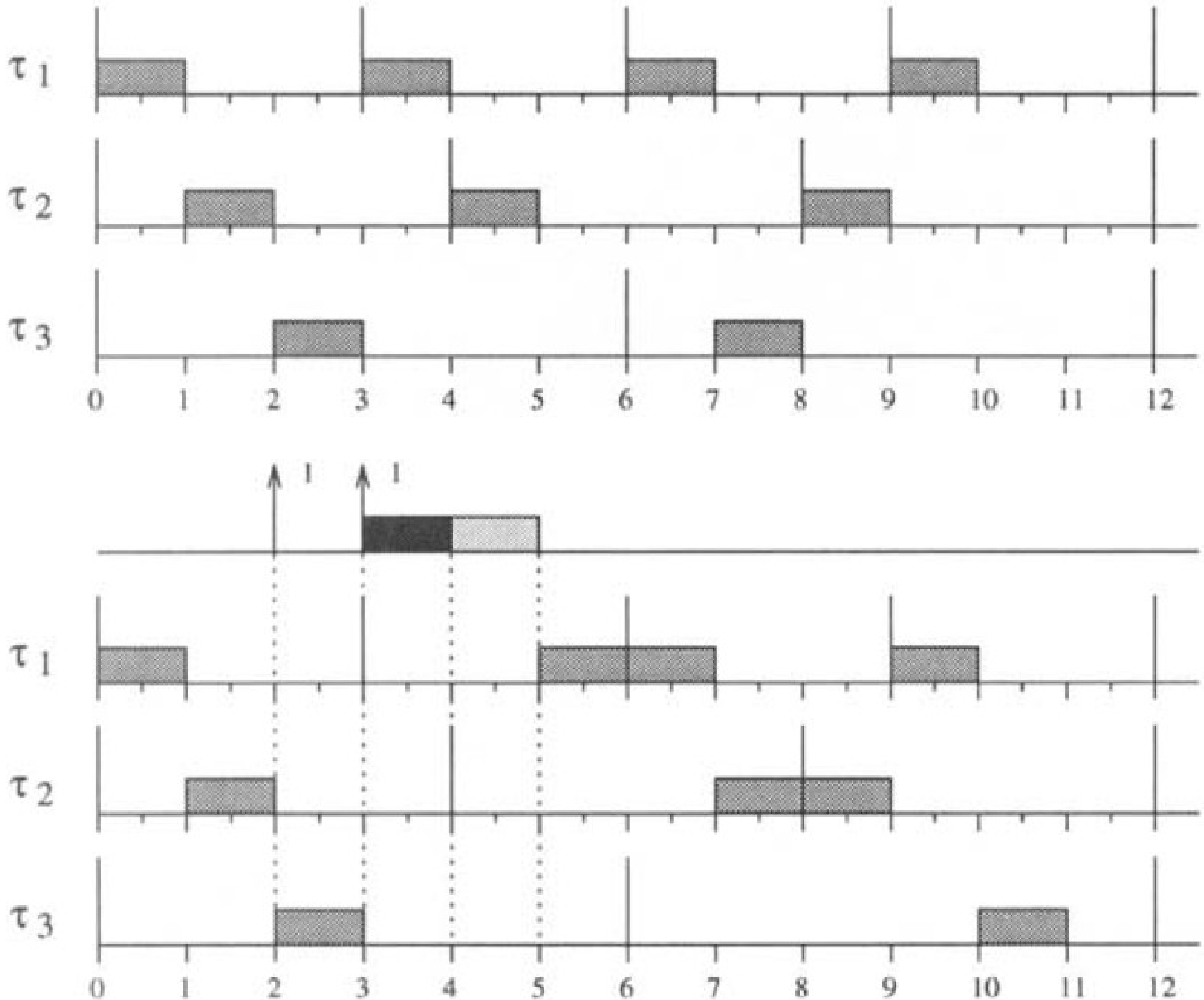


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Evaluation

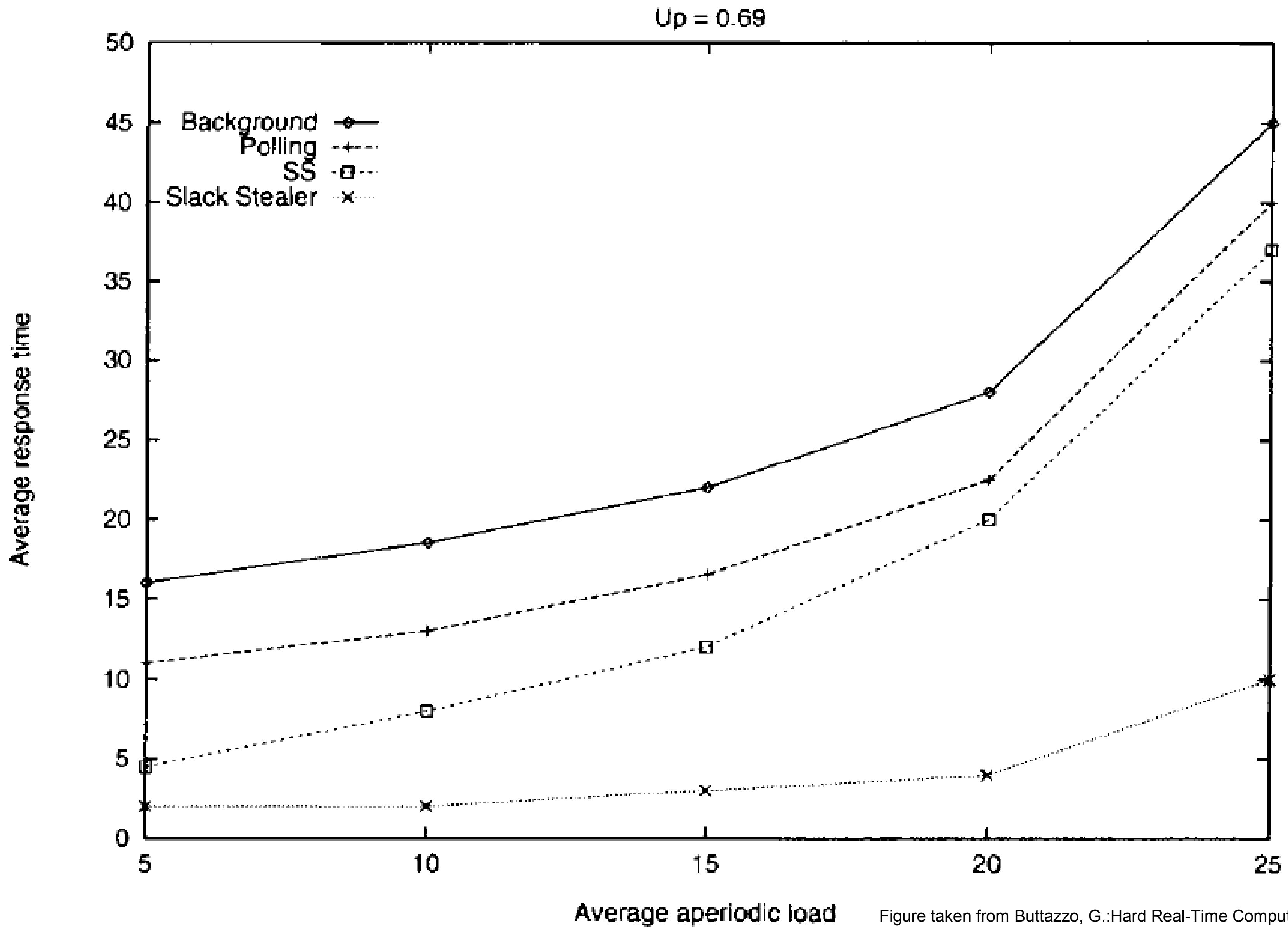


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems

Evaluation

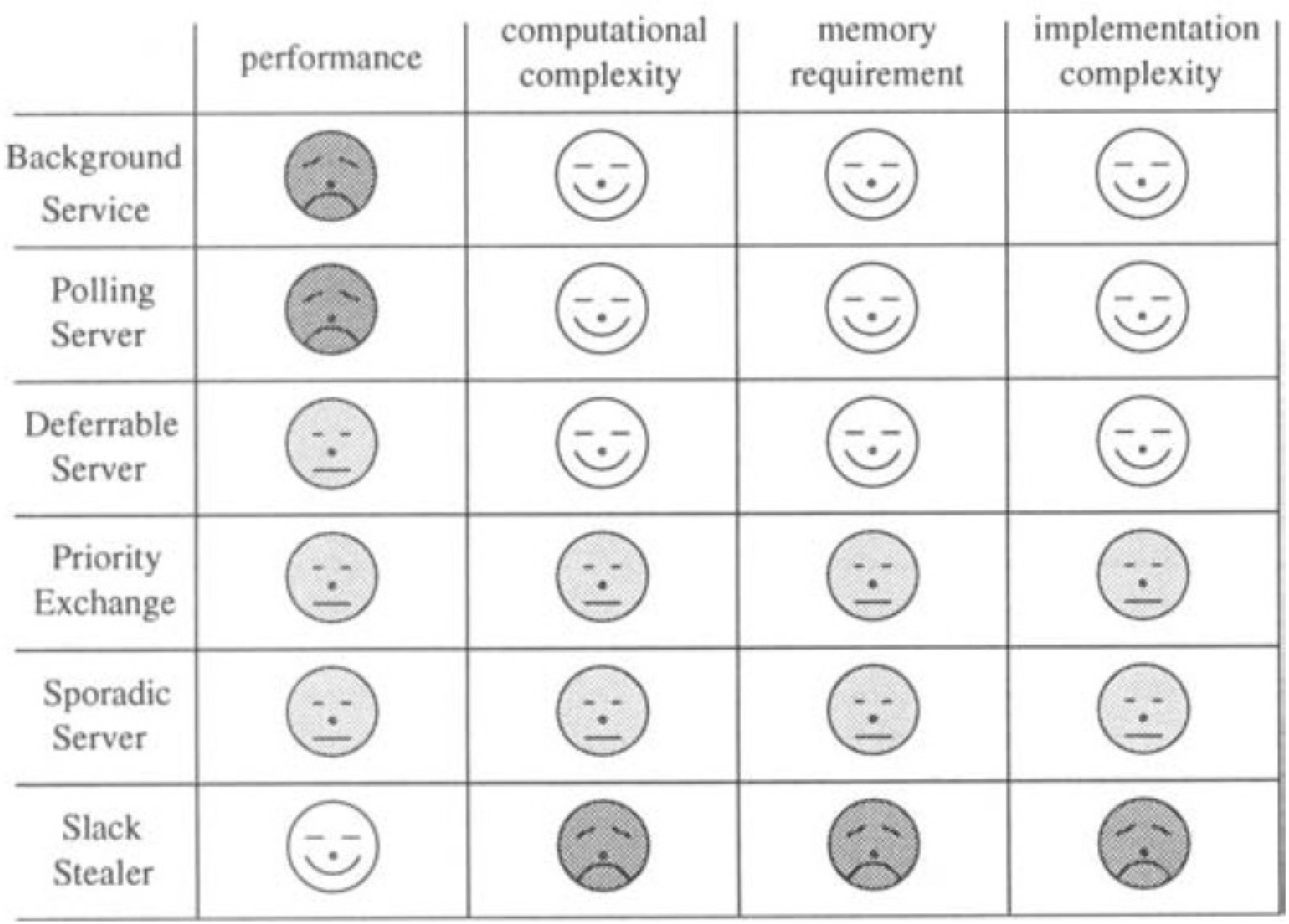


Figure taken from Buttazzo, G.:Hard Real-Time Computing Systems