Problem

- In many applications, there are as well aperiodic as periodic tasks.
- **Periodic tasks**: time-driven, execute critical control activities with hard timing constraints aimed at guaranteeing regular activation rates.
- **Aperiodic tasks**: event-driven, may have hard, soft, non-real-time requirements dependent on the specific application.
- **Sporadic tasks**: Offline guarantee of event-driven aperiodic tasks with critical timing constraints can be done only by making proper assumptions on the environment; that is by assuming a maximum arrival rate for each critical event. Aperiodic tasks characterized by a minimum interarrival time are called sporadic.
- **Firm tasks**: Aperiodic tasks requiring on-line guarantee on individual instances are called firm. Whenever a firm aperiodic request enters the system, an acceptance test can be executed by the kernel to verify whether the request can be served within its deadline. If this test fails, the request is rejected.

Problem

- **Problem**: Present a number of scheduling algorithms for handling hybrid task sets consisting of a subset of hard periodic tasks and a subset of soft aperiodic tasks.
- **Rate monotonic server**:
  - Periodic tasks are scheduled based on a fixed priority assignment, namely RM.
  - All periodic tasks start simultaneously at t=0 and their relative deadlines are equal to their periods.
  - Arrival times of aperiodic requests are unknown.
  - If not specified otherwise, the minimum interarrival time of a sporadic task is assumed to be equal to its deadline.
- **Algorithms**: polling server, deferrable server, priority exchange, sporadic server, slack stealing, … .

**EDF Server**

- Periodic tasks are scheduled based on EDF.
- All periodic tasks are simultaneously activated at t=0 and have relative deadlines equal to their periods.
- All aperiodic tasks do not have deadlines.
- Each aperiodic request has a known computation time but an unknown arrival time.
- **Algorithms**: priority exchange, sporadic server, total bandwidth server, earliest deadline late server, …
**Background scheduling**

- **Simple solution** for RM and EDF scheduling of periodic tasks:
  - Processing of aperiodic tasks in the background, i.e. if there are no periodic request.
  - Periodic tasks are not affected.
  - Response of aperiodic tasks may be prohibitively long and there is no possibility to assign a higher priority to them.

---

**Example (rate monotonic periodic schedule):**

![Example of rate monotonic periodic schedule](image)

---

**Overview RM server**

<table>
<thead>
<tr>
<th></th>
<th>excellent</th>
<th>good</th>
<th>poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>performance</td>
<td><img src="image" alt="Performance" /></td>
<td><img src="image" alt="Performance" /></td>
<td><img src="image" alt="Performance" /></td>
</tr>
<tr>
<td>computational complexity</td>
<td><img src="image" alt="Complexity" /></td>
<td><img src="image" alt="Complexity" /></td>
<td><img src="image" alt="Complexity" /></td>
</tr>
<tr>
<td>memory requirement</td>
<td><img src="image" alt="Memory" /></td>
<td><img src="image" alt="Memory" /></td>
<td><img src="image" alt="Memory" /></td>
</tr>
<tr>
<td>implementation complexity</td>
<td><img src="image" alt="Implementation" /></td>
<td><img src="image" alt="Implementation" /></td>
<td><img src="image" alt="Implementation" /></td>
</tr>
</tbody>
</table>

![Comparison table](image)

---

**RM - polling server**

- **Idea**: Introduce an artificial periodic task whose purpose is to service aperiodic requests as soon as possible (therefore, “server”).
  - Like any periodic task, a server is characterized by a period $T_s$ and a computation time $C_s$.
  - The server is scheduled with the same algorithm used for the periodic tasks and, once active, it serves the aperiodic requests within the limit of its server capacity.
  - Its priority (period!) can be chosen to match the response time requirement for the aperiodic tasks.
RM - polling server

- **Function of polling server (PS)**
  - At regular intervals equal to $T_s$, PS becomes active and serves any pending aperiodic requests within the limit of its capacity $C_s$.
  - If no aperiodic requests are pending, PS suspends itself until the beginning of the next period and the time originally allocated for aperiodic service is not preserved for aperiodic execution.

- **Disadvantage:** If an aperiodic request arrives just after the server has suspended, it must wait until the beginning of the next polling period.

RM - polling server

- **Schedulability analysis of periodic tasks**
  - As in the case of RM as the interference by a server task is the same as the one introduced by an equivalent periodic task.
  - A set of periodic tasks and a server task can be executed within their deadlines if

\[
\frac{C_s}{T_s} + \sum_{i=1}^{n} \frac{C_i}{T_i} \leq (n+1)(2^{1/(n+1)} - 1)
\]

- Again, this test is sufficient but not necessary.

RM - polling server

- **Aperiodic guarantee of firm aperiodic activities**
  - **Assumption:** An aperiodic task is finished before a new aperiodic request arrives.
  - Computation time $C_a$, deadline $D_a$.
  - Sufficient schedulability test:

\[
(1 + \left[\frac{C_a}{C_s}\right])T_s \leq D_a
\]

If the server task has the highest priority there is a necessary test also.

The aperiodic task arrives shortly after the activation of the server task.

Maximal number of necessary server periods.
RM - deferrable server (DS)

- **Polling Server**: Does not preserve the capacity of the server until it is needed.
- **Deferrable Server**:
  - DS preserves its capacity if no requests are pending upon the invocation of the server.
  - The capacity is maintained until the end of the period, so that aperiodic requests can be serviced at the same server’s priority at any time, as long as the capacity has not been exhausted.
  - At the beginning of any server period, the capacity is replenished at its full value.
  - Shorter response times for aperiodic requests.

RM - deferrable server (DS)

- Periodic tasks have worse guarantees as the server tasks may move its execution to a later time instant (contrary to the RM assumption!).
- **Periodic guarantee** (derivation somewhat complex):

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

where \( U_s = \frac{C_s}{T_s} \) and \( U_p = \sum_{i=1}^{n} \frac{C_i}{T_i} \)

RM - deferrable server

- **Example**

<table>
<thead>
<tr>
<th>( C_i )</th>
<th>( T_i )</th>
<th>( C_s ) = 2</th>
<th>( T_s ) = 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

RM - deferrable server

- **Example**

<table>
<thead>
<tr>
<th>( C_i )</th>
<th>( T_i )</th>
<th>( C_s ) = 2</th>
<th>( T_s ) = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>3</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

aperiodic requests

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
<th>18</th>
<th>20</th>
<th>22</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

aperiodic requests

<table>
<thead>
<tr>
<th>0</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>14</th>
<th>16</th>
<th>18</th>
<th>20</th>
<th>22</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**EDF - server**

- **EDF Server**: With respect to fixed-priority assignments, dynamic scheduling algorithms are characterized by higher schedulability bounds, which allow the processor to be better utilized, increase the size of aperiodic servers, and enhance aperiodic responsiveness.
- Similar concepts as in the case of RM servers can be used.
- But there is another possibility:
  - Assign a deadline to each aperiodic request.
  - The overall processor utilization of the aperiodic load never exceeds a specified maximum value $U_s$.

**EDF – total bandwidth server**

- **Total Bandwidth Server**:
  - When the $k$th aperiodic request arrives at time $t = r_k$, it receives a deadline
    \[ d_k = \max(r_k, d_{k-1}) + \frac{C_k}{U_s} \]
    where $C_k$ is the execution time of the request and $U_s$ is the server utilization factor (that is, its bandwidth). By definition, $d_0 = 0$.
  - Once a deadline is assigned, the request is inserted into the ready queue of the system as any other periodic instance.

**EDF - total bandwidth server**

- **Schedulability test**:
  Given a set of $n$ periodic tasks with processor utilization $U_p$ and a total bandwidth server with utilization $U_s$, the whole set is schedulable by EDF if and only if
  \[ U_p + U_s \leq 1 \]

- **Proof**:
  - In each interval of time $[t_1, t_2]$, if $C_{apc}$ is the total execution time demanded by aperiodic requests arrived at $t_1$ or later and served with deadlines less or equal to $t_2$, then
    \[ C_{apc} \leq (t_2 - t_1) U_s \]
EDF - total bandwidth server

Example: \( U_p = 0.75, \ U_s = 0.25, \ U_p + U_s = 1 \)