Deferrable Server Algorithm for Enhanced Aperiodic Responsiveness in Hard Real-Time System

Jay K. Strosnider, John P. Lehoczky
And Lui Sha
Contents

1. Introduction
2. Polling server Algorithm
3. Deferrable Server Algorithm
Introduction

- Scheduling Algorithm feasible for Hard Real-time Systems – Periodic or Aperiodic
- Integrated occurrence of both Hard-Real time periodic Task and Hard & soft Aperiodic task

- Aperiodic task
  - Scheduled in the periodic server’s time slots – built upon Rate monotonic Scheduling algorithm
  - Soft Deadline - Average response time requirement
  - Hard Deadline – guaranteed response time as periodic task

- Requirements of Joint scheduling of periodic & Aperiodic Task
  - Periodic task – meet the hard deadline of the task
  - Aperiodic task – meet or minimize the average response time of the task
Background Scheduling

- Aperiodic task are executed when there is no periodic task to execute
- Task - no critical time response time
- Simple, no guarantee on aperiodic schedulability

Priority: $\tau_1 > \tau_2 > \text{ap}$
Polling or TDM Server Algorithm

- A Periodic task - Providing high priority service for Aperiodic Task
- Run at the start of the period – serving pending or in-coming aperiodic task over the period Cps
- If there are no aperiodic tasks at an invocation of the server, the server suspends itself during its current period and gets invoked again at its next period
- Loses any unused execution time
- Poor response time for aperiodic task

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

<table>
<thead>
<tr>
<th></th>
<th>( C_s )</th>
<th>( T_s )</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

Priority: \( \tau_1 > PS > \tau_2 \)
Deferrable Server Algorithm

- New scheduling algorithm approach – Fast response time for Aperiodic task
- Provide High priority service to Aperiodic Task
- Execution time($C_{DS}$) is allocated at the start of period
- When an aperiodic task arrives, the server is invoked to execute aperiodic tasks and maintains its priority
- When the server is invoked with no outstanding aperiodic tasks, the server does not execute but defers its assigned time slot

| $T_{DS}$ – Task Period  
  | $C_{DS}$ – Capacity or execution time |

- Period, $T_{DS} < T_{S}$ (Shortest Periodic Task Period)
Deferrable Server Algorithm (2)

<table>
<thead>
<tr>
<th></th>
<th>$C_i$</th>
<th>$T_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$C_s$</th>
<th>$T_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

$C_1 = 2$

$C_2 = 1$

Response Time = 4
Deferrable Server Algorithm (3)

- At intermediate priority level – DS Less capable of providing responsive aperiodic service
- DS Capacity can be lost, of continuous High priority periodic task
- Under overload – DS task may defer its execution time
- Contradict to Liu & Layland Assumption

- A DS with \((T_s=4, C_s=2)\) is NOT equivalent to a periodic task with \((T_i=4, C_i=2)\)
Schedulability Conditions(1)

- Schedulability of DS Task, $\tau_0$ at high priority with ‘m’ periodic task
- $T_1, T_2, \ldots, T_m$, periodic task with priority given by rate monotonic algorithm
- Period = $T_i$, Execution Time = $C_i$,
- Assume $T_0 \leq T_1 \leq T_2 \leq \ldots \leq T_m$ and Deadline, $D_i = T_i$

- DS Task, $\tau_0$ execution time, $C_0$ starting at $I_0 + (k-1)T_0$ available till $[I_0 + (k-1)T_0, I_0 + kT_0)$
- Unused capacity is lost at $I_0 + kT_0$
Schedulability Conditions(2)

- Condition to meet deadline of periodic task
  - A task $\tau_i$ has its longest response time when it arrives at a critical instant. A critical instant occurs at $t = 0$ when $I_i = 0, 1 \leq I \leq m$
  - All task deadlines will be met using the rate monotonic scheduling algorithm if the first request for each task meets its deadline under critical instant phasing, $I_i = 0, 1 \leq i \leq m$
  - All periodic task deadlines are guaranteed by the rate monotonic algorithm under all task phasing if and only if
    \[
    \max_{1 \leq i \leq m} \min_{0 \leq t \leq T_i} \sum_{j=1}^{t} \frac{C_j}{t} \left[ \frac{t}{T_j} \right] \leq 1.
    \]

- We generalize to provide DS task, $\tau_0$ high priority level
Longest Response Time

- Level-i Busy period
  - A time interval \([s,t]\) satisfying the following three conditions
    1. All requests of priority \(i\) or higher made before ‘\(s\)’ are completed by ‘\(s\)’
    2. All requests of priority \(i\) or higher made before ‘\(t\)’ are completed by \(t\)
    3. For every \(U \in (s,t)\), there exists at least one request of priority \(i\) or higher that arrived before ‘\(u\)’ and is not completed by ‘\(u\)’

- Let’s understand with example:
  - Two tasks, \(T1\) & \(T2\)
    - Level-1 busy Period, \([0,4], [10,14], [20,24]\)
    - Level-2 busy period, \([0,10], [10,20], [20,24]\). Task \(T2\) continuously busy \([0,24]\).
    - If \(D_i \leq T_i\), all deadlines are met & level-i busy period contain a request, \(T_i\). Busy period will contain only one request and will be completed at end of request time
    - Level-i concept used to characterize the longest response time of a request \(T_i\)
Lemma:

Suppose a periodic task set $T_1, T_2, \ldots, T_n$ is schedulable using a fixed priority scheduling algorithm where $T_i$ is assigned priority ‘$i$’. A necessary condition for an execution request of $T_i$ to have its longest response time is for the request of $T_i$ to initiate a level-$i$ busy period.

- If task, $T_i$ initiated at time $s$, then also it would have been completed at time $t$, increasing the response time to $(s-t)$
- Thus response time of task, $T_i$ is maximized only if it is requested at level-$i$ busy period.
Worst case phasing

- Phasing that maximizes the response time of task
- **Theorem:** The task set phasing which causes the longest response time for any periodic task occurs when all task periods of equal or higher priority are requested simultaneously, and the highest priority Deferrable Server task demands $Co$ units at this instant and is reinitiated $Co$ units later.

- To check schedulability using fixed Priority algorithm with Deferrable Server task having highest priority, we will check whether the first execution request of each task meets its deadline under the worst case phasing.

- Cumulative demand for processor made by Periodic Task, $\tau_j$ in $[0,t]$ is

$$C_j \left[ \frac{\text{max}(0, t - I_j)}{T_j} \right]$$

- When maximized for all $t >= 0$, setting $I_j = 0$

- For Deferrable sever Task, $\tau_0$ can request $C_0$ units at any time in $T_0$ units

- Largest demand for processor time will be at $I_0 = C_0$, execution request $[0, C_0], [C_0, 2C_0], [C_0+T_0, 2C_0+T_0]$ etc.
Schedulability Test with DS Task

- Task $\tau_k$, $1 \leq k \leq m$ meet all its deadline, if first execution meets its deadline under worst case phasing.

- Under worst case phasing, the total demand for processor time at $t \geq T_0$ is,

$$C_0 + C_0 \left[ \frac{t-T_0}{T_0} \right] + \sum_{j=1}^{k} C_j \left[ \frac{t}{T_j} \right]$$

- First 2 terms represent maximum execution time of $\tau_0$.

- Condition for a task, $\tau_k$ to meet its deadline.

For all task set to be schedulable, i.e. for each $k$, $1 \leq k \leq m$, condition is

$$\min_{T_0 \leq t \leq T_k} \left\{ C_0 \left( 1 + \left[ \frac{t-T_0}{T_0} \right] \right) + \sum_{j=1}^{k} C_j \left[ \frac{t}{T_j} \right] \right\} \leq t$$

- Evaluating condition as expression of discontinuous function. Determine schedulability of task set at sequence of time points $\{ S_l, \ l=0,1,2.. \}$

$$S_0 = 2C_0 + \sum_{j=1}^{k} C_j \left[ \frac{T_0}{T_j} \right], \ \text{and} \ \ S_{\ell+1} = C_0 \left( 1 + \left[ \frac{S_\ell-T_0}{T_0} \right] \right) + \sum_{j=1}^{k} C_j \left[ \frac{S_\ell}{T_j} \right]$$
Least upper bound on Schedulability

✓ For \( l \geq 1, \ S_l = S_{l+1} \leq T_k, \) Task \( \tau_k \) is schedulable
✓ If \( S_l > T_k, \) Task \( \tau_k \) is not schedulable

☑ The least upper bounds provide sufficient conditions for task set schedulability in the sense that if the task set utilization lies below the bound, all periodic task deadlines will be met.
☑ Least upper bound condition, with utilization of PS or DS task, \( U_0 = C_0/T_0 \) as fixed is \( PS_n(U_0) \) and \( DS_n(U_0) \)
☑ Periodic task utilization \( U_{per} \leq PS_n(U_0) \) or \( DS_n(U_0) \) is schedulable with high priority task with utilization \( U_0 \)
☑ \( U_{per} > PS_n(U_0) \) or \( DS_n(U_0) \), there exists a non-schedulable task
Least upper bound for two task set

- **Condition 1**: No DS task and two hard deadline periodic task $\tau_0$, $\tau_1$ and highest periodic task is polling server task and it is fixed.

- Assume that $Co$ and $To$ and $T1$ are given. Let $R1 = T1/C1$. How large $C1$ can be and still have all task deadlines satisfied under all phasings of $\tau_0$, $\tau_1$.

- Assuming the worst case phasing, the maximum utilization for $U1 = C1/T1 = C1/R1$ is given by

$$
PS_{U1}(U_0, R_1) = \begin{cases} 
\frac{k(1 - U_0)}{R_1} & \text{if } k \leq R_1 \leq k + U_0 \\
1 - \frac{(k + 1)U_0}{R_1} & \text{if } k + U_0 \leq R_1 \leq k + 1
\end{cases}
$$

- Least upper bound on $U1$, $(1-U_0)/(1+U_0)$, $0 \leq U_0 \leq 1$.

- Total schedulable utilization is minimized to $2(\sqrt{2} - 1) = 0.828$.

- Least upper bound on schedule utilization, $PS_{tot}(U_0)$ is,

$$
PS_{tot}(U_0) = (1 + U_0^2)/(1 + U_0), \quad 0 \leq U_0 \leq 1.
$$
Least upper bound

- Least upper bound for DS task, $\tau_0$ with high priority and one periodic task, $\tau_1$ is given as

$$DS_{U_1}(U_0, R_1) = \begin{cases} 
1 - \frac{2}{R_1} U_0 & \text{if } 1 \leq R_1 \leq 1 + U_0 \\
\frac{k(1-U_0)}{R_1} & \text{if } k + U_0 \leq R_1 \leq k + 2U_0, \quad k \geq 1 \\
1 - \frac{k+2U_0}{R_1} & \text{if } k + 2U_0 \leq R_1 \leq k + 1 + U_0, \quad k \geq 1.
\end{cases}$$

- Minimizing value of $U_1$ over $R_1$, total utilization of two DS-task is given as,

$$DS_{tot}(U_0) = \begin{cases} 
\min(1 - U_0, (1 + 2U_0^2)/(1 + 2U_0)) & \text{if } 0 \leq U_0 \leq 0.5 \\
U_0 & \text{if } 0.5 \leq U_0 \leq 1
\end{cases}$$

- Total schedulable utilization decrease, when DS task with utilization $U_0$ replaces an ordinary periodic task with utilization $U_0$ for $0 \leq U_0 \leq \frac{1}{4}$ is,

$$\frac{1-U_0}{1+U_0} - \frac{1-U_0}{1+2U_0} = \frac{(1-U_0)U_0}{(1+U_0)(1+2U_0)}$$

- Is an increasing function of $U_0$ and is $0$ at $U_0=0$ and increases to $1$ at $U_0=1/4$
Least upper bound for ‘n’ PS task set

- When generalized over ‘n’ task, \( \tau_0, \tau_1 \ldots \ldots \tau_n \) where \( \tau_0 \) is Polling task with highest priority and utilization \( U_0 \)
- Least upper schedulability bound with fixed PS task utilization, \( U_0 \) is,

\[
PS_{tot,n}(U_0) = n \left( \left( \frac{2}{1 + U_0} \right)^{1/n} - 1 \right) + U_0
\]

- When \( n \rightarrow \infty \), least upper bound is,

\[
PS_{tot,\infty}(U_0) = \ell n \left( \frac{2}{1 + U_0} \right) + U_0
\]

- It states, if one has polling task with utilization \( U_0 \), then ‘n’ periodic task with total utilization no greater than \( n \left( \left( \frac{2}{1 + U_0} \right)^{1/n} - 1 \right) \) is schedulable.
Consider 3 cases,
1. \( T_1 \) and \( T_n \) are both smaller than \( T_0 + C_0 \)
2. \( T_1 \) and \( T_n \) are both larger than
3. \( T_1 \) is less than \( T_0 + C_0 \), but \( T_n \) is larger than \( T_0 + 2C_0 \)

Case: Least bound on total schedulable utilization with \( R_1 = T_1/T_0 \) and \( U_0 = C_0/T_0 \), \( 0 \leq U_0 \leq 1/2 \), and

Case 1: If \( T_n < T_0 + C_0 \) is given as,

\[
DS_{tot,\infty}(U_0, R_1) = \begin{cases} 
U_0 + \ln \left( \frac{1 + U_0}{R_1} \right) + \frac{2R_1 - 1 - 3U_0}{1 + U_0} & 0 \leq U_0 \leq \frac{2R_1 - 1}{3} \\
U_0 + \ln \left( 2 \left( 1 - \frac{U_0}{R_1} \right) \right) & \frac{2R_1 - 1}{3} \leq U_0 \leq \frac{1}{2}
\end{cases}
\]

Case 2 & 3: If \( T_0 + 2C_0 \leq T_n \) is given as,

\[
DS_{tot,\infty}(U_0, R_1) = \begin{cases} 
U_0 + \ln \left( \frac{1 + U_0}{1 + 2U_0} \left( \frac{2 - U_0}{R_1} \right) \right) & 1 \leq R_1 \leq 1 + U_0 \\
U_0 + \ln \left( \frac{2 + U_0}{1 + 2U_0} \right) & 1 + U_0 \leq R_1 \leq 1 + 2U_0 \\
U_0 + \ln \left( \frac{2 + U_0}{R_1} \right) + \frac{2[R_1 - (1 + 2U_0)]}{2 + U_0} & 1 + 2U_0 \leq R_1 \leq 2
\end{cases}
\]
Least upper bound for ‘n’ DS task set

- On minimizing over R1, total utilization for high priority DS task and periodic task is

\[
DS_{tot,\infty}(U_0) = \begin{cases} 
U_0 + \ln \left( \frac{1 + U_0}{1 + 2U_0} (2 - U_0) \right) & 0 \leq U_0 \leq \frac{1}{3} \\
U_0 + \ln(2(1 - U_0)) & \frac{1}{3} \leq U_0 \leq \frac{1}{2}
\end{cases}
\]

- **Note:** Least upper bound is a combination of case 1 & case 3 providing bound \(1/3 \leq U_0 \leq 1/2\) & \(0 \leq U_0 < 1/3\) respectively

- Case 2 is bound \(0 \leq U_0 \leq 1/2\) is greater than case 3, does not contribute to least upper scheduling bound.
Deferrable Server Design

- Two design Factors,
  1. Determining whether to use the DS Approach
  2. Selecting the period and capacity for the task
- For a task to be served immediately upon arrival,
  1. DS Task must run at highest priority, ie. \( T_0 \leq T_1 \)
  2. The capacity of the DS in each period to must be sufficiently large to serve a busy period of the aperiodic tasks arriving during a single DS period

Case 2:
Example: Two DS Task, DS1 & DS2 with \( C_{01}=1, T_{01}=10 \) & \( C_{02}=2, T_{02}=20 \)
  - DS1 & DS2 Provides 2 units of capacity at time \([0,20]\)
  - In DS1, 1 unit of service capacity will be lost at 10, if not used in \([0,10]\), Thus DS1 can only service task whose service requirement is no greater than 1
  - Whereas DS2 can service task where the service requirement is no greater than 2
  - Thus aperiodic task with high priority & longer period can serve the task to minimize the response time and save high aperiodic priority service capacity
Thank you