Resource Reservation Servers

Jan Reineke

Saarland University

July 18, 2013

With thanks to Jian-Jia Chen!
Scheduling Theory in Real-Time Systems

Uniprocessor Systems

Task Models and Scheduling
Schedulability Analysis
Resource Sharing and Servers

Multi-processor Systems

Partitioned Scheduling
Semi-Partitioned Scheduling
Global Scheduling

Resource Sharing

Global Scheduling

Resource Reservation Servers

July 18, 2013
Outline

1 Resource Reservation Servers
   ■ Servers for fixed-priority systems
   ■ Servers for dynamic-priority systems
Aperiodic Tasks

So far we have dealt with *periodic* tasks. What about non-periodic tasks?

- **Sporadic** (aperiodic, but with minimum interarrival time):
  - Worst case: all sporadic tasks arrive with minimum interarrival time
  - Can assume sporadic tasks as periodic in schedulability test

- **Aperiodic** (no limitations on arrival times):
  - Need for a new mechanism to protect periodic/sporadic tasks: *resource reservation servers.*
Outline

1 Resource Reservation Servers
   - Servers for fixed-priority systems
   - Servers for dynamic-priority systems
Well-Known Resource Reservation Servers for Fixed-Priority Systems

- Background Scheduling: schedule aperiodic tasks whenever no periodic task is active.
- Polling Server (PS): provide a fixed execution budget that is only available at pre-defined times.
- Deferrable Server (DS): provide a fixed budget, in which the budget replenishment is done periodically.
- Sporadic Server (SS): provide a fixed budget, in which the budget replenishment is performed only if it was consumed.
- Others (not included): priority exchange (PE) server, slack stealer, etc.
Background Scheduling

Execute aperiodic tasks when no periodic tasks are active:

- No disturbance of periodic tasks (and their feasibility).
- Simple runtime mechanism.
- Possibly poor response time for aperiodic tasks. No guarantees.
Polling Server (PS)

- Behavior: periodic task with the specified priority
  - period: $T_{S_i}$
  - capacity (computation time): $C_{S_i}$
- Consumption rule:
  - upon activation of the task, executing events until either no request in the ready queue for the server or the capacity $C_{S_i}$ is exhausted.
An Example of PS

\( \tau_1 = (1, 4, 4), \tau_2 = (2, 6, 6). \)
Polling server: \( C_S = 2 \) and \( T_S = 5. \)
Priority: \( \tau_1 > PS > \tau_2. \)
An Example of PS

$\tau_1 = (1, 4, 4)$, $\tau_2 = (2, 6, 6)$.

Polling server: $C_S = 2$ and $T_S = 5$.

Priority: $\tau_1 > PS > \tau_2$. 

![Graphical representation of PS example]
An Example of PS

\( \tau_1 = (1, 4, 4), \ \tau_2 = (2, 6, 6). \)

Polling server: \( C_S = 2 \) and \( T_S = 5. \)

Priority: \( \tau_1 > PS > \tau_2. \)
An Example of PS

\[ \tau_1 = (1, 4, 4), \; \tau_2 = (2, 6, 6). \]
Polling server: \( C_S = 2 \) and \( T_S = 5. \)
Priority: \( \tau_1 > PS > \tau_2. \)
An Example of PS

\[ \tau_1 = (1, 4, 4), \; \tau_2 = (2, 6, 6). \]
Polling server: \( C_S = 2 \) and \( T_S = 5 \).
Priority: \( \tau_1 > PS > \tau_2 \).
Properties of PS

- **Schedulability Guarantee:**
  - Suppose that there are $n$ periodic tasks and a polling server with utilization $U_S = \frac{C_s}{T_s}$ and the RM scheduling algorithm is adopted.
  - The schedulability of the periodic task set is guaranteed if
    \[
    U_S + \sum_{i=1}^{n} \frac{C_i}{T_i} \leq U_{\text{lub}}(RM, n + 1) = (n + 1)(2^{\frac{1}{n+1}} - 1).
    \]
  - The proof can be done by imagining that a periodic task represents the polling server, which is executed for at most $C_i$ time units after it is granted for execution.
  - Since the capacity is greedily set to 0 if there is no request for the polling server to execute, the periodic task, that represents the polling server, can be imagined as early completion, instead of task suspension, of the task.
Deferrable Server (DS)

- Behavior: periodic task
  - period: $T_{S_i}$
  - capacity (computation time): $C_{S_i}$

- Replenishment rule:
  - periodic replenishment at the multiple of $T_{S_i}$

- Consumption rule:
  - aperiodic requests are served when the server still has capacity
  - capacity is lost at the end of the period
An Example of DS

\(\tau_1 = (1, 4, 4), \tau_2 = (2, 6, 6)\).

Deferrable server: \(C_S = 2\) and \(T_S = 5\).

Priority: \(\tau_1 > DS > \tau_2\).
An Example of DS

$\tau_1 = (1, 4, 4), \tau_2 = (2, 6, 6)$.
Deferrable server: $C_S = 2$ and $T_S = 5$.
Priority: $\tau_1 > DS > \tau_2$. 

- $\tau_1 = (1, 4, 4)$
- $\tau_2 = (2, 6, 6)$
- $C_S = 2$ and $T_S = 5$
- Priority: $\tau_1 > DS > \tau_2$
An Example of DS

\[ \tau_1 = (1, 4, 4), \quad \tau_2 = (2, 6, 6). \]

Deferrable server: \( C_S = 2 \) and \( T_S = 5 \).
Priority: \( \tau_1 > DS > \tau_2 \).
An Example of DS

$$\tau_1 = (1, 4, 4), \ \tau_2 = (2, 6, 6).$$
Deferrable server: $$C_S = 2$$ and $$T_S = 5.$$ Priority: $$\tau_1 > DS > \tau_2.$$
An Example of DS

\( \tau_1 = (1, 4, 4), \quad \tau_2 = (2, 6, 6). \)

Deferrable server: \( C_S = 2 \) and \( T_S = 5. \)

Priority: \( \tau_1 > DS > \tau_2. \)
Suppose that there are \( n \) periodic tasks and a deferrable server with utilization \( U_S = \frac{C_S}{T_S} \) and the RM scheduling algorithm is adopted.

RM analysis is incorrect for such a case, since the deferrable server does not act like a periodic task.

We will not go through the derivation of the utilization bound.
Suppose that there are \( n \) periodic tasks and a deferrable server with utilization \( U_S = \frac{C_S}{T_S} \) and the RM scheduling algorithm is adopted. RM analysis is incorrect for such a case, since the deferrable server does not act like a periodic task.

We will *not* go through the derivation of the utilization bound.
Theorem

A set of \( n \) independent, preemptable sporadic tasks with relative deadlines equal to their respective periods together with a deferrable server with utilization \( U_S = C_S / T_S \) can be scheduled on a processor according to the RM algorithm if its total utilization \( U \) is at most:

\[
UB(RM, n) = U_S + n \left( \left( \frac{U_S + 2}{2U_S + 1} \right)^{\frac{1}{n}} - 1 \right).
\]

Taking \( n \to \infty \), we have

\[
\lim_{n \to \infty} UB(RM, n) = U_S + \ln \frac{U_S + 2}{2U_S + 1}
\]
in which $UB(RM)$ is minimized ($UB^*(RM) \approx 0.652$) when $U_S = \frac{\sqrt{33} - 5}{4} \approx 0.186$. 
Sporadic Server (SS)

- Behavior: sporadic task with a specified priority
  - period: $T_{S_i}$
  - capacity (computation time): $C_{S_i}$

- Rules:
  - Let $\pi_{\text{exe}}(t)$ be the priority level that is executing at time $t$
  - The server is Active when the $\pi_{\text{exe}}(t)$ has no lower priority than SS.
  - The server is Idle when the $\pi_{\text{exe}}(t)$ has lower priority than SS.
  - Initially, the server is Idle and its budget is $C_{S_i}$. When the server becomes Active at time $t_1$, the replenishment time is set to $t_1 + T_{S_i}$
  - When the server becomes Idle at time $t_2$, the (next) replenishment amount is set to the amount of capacity consumed in time interval between the last replenishment time and $t_2$
An Example of SS

\(\tau_1 = (1, 5, 5), \; \tau_2 = (4, 15, 15)\).

Sporadic server: \(C_S = 5\) and \(T_S = 10\).
Priority: \(\tau_1 > SS > \tau_2\).
An Example of SS

\( \tau_1 = (1, 5, 5), \ \tau_2 = (4, 15, 15). \)

Sporadic server: \( C_S = 5 \) and \( T_S = 10. \)

Priority: \( \tau_1 > SS > \tau_2. \)
An Example of SS

\( \tau_1 = (1, 5, 5), \ \tau_2 = (4, 15, 15). \)

Sporadic server: \( C_S = 5 \) and \( T_S = 10. \)

Priority: \( \tau_1 > SS > \tau_2. \)
An Example of SS

\( \tau_1 = (1, 5, 5), \ \tau_2 = (4, 15, 15). \)

Sporadic server: \( C_S = 5 \) and \( T_S = 10. \)

Priority: \( \tau_1 > SS > \tau_2. \)
Schedulability Guarantees of SS

Theorem

If a periodic task set is schedulable, replacing a task \( \tau_i \) by a sporadic server \( SS_i \) with the same period and execution time is still schedulable.

Case 1: Similar to a periodic task that is delayed to arrive at time \( t_2 \).

Case 2: Like the behavior of a periodic real-time task.

Case 3: Like multiple tasks with the same period but with different arrival times, and the sum of their execution times is the same as that of \( \tau_i \).
## Overview of PS, DS, and SS

<table>
<thead>
<tr>
<th></th>
<th>Performance</th>
<th>computation</th>
<th>memory</th>
<th>implementation complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS</td>
<td>poor</td>
<td>excellent</td>
<td>excellent</td>
<td>excellent</td>
</tr>
<tr>
<td>DS</td>
<td>good</td>
<td>excellent</td>
<td>excellent</td>
<td>excellent</td>
</tr>
<tr>
<td>SS</td>
<td>excellent</td>
<td>good</td>
<td>good</td>
<td>good</td>
</tr>
</tbody>
</table>
Outline

1 Resource Reservation Servers
   - Servers for fixed-priority systems
   - Servers for dynamic-priority systems
Resource Reservation Servers for Dynamic-Priority Systems

- Total Bandwidth Server (TBS): provide a fixed utilization for executing jobs, in which the deadline for execution is dependent on the execution time of jobs.
- Constant Bandwidth Server (CBS): provide a fixed utilization for executing jobs, in which the deadline for execution is independent on the execution time of jobs.
- Others (not included): dynamic priority exchange (DPE) server, dynamic slack stealer, dynamic sporadic server, etc.
Behavior: assign the absolute deadline to an incoming job such that the utilization for this server is at most $U_{S_i}$, which is the only parameter required for a TBS server $S_i$.

Initialization: assign server deadline $D_{S_i}$ to $-\infty$.

Deadline assignment rule: when a job arrives at time $t$

- The absolute deadline of this job is set to

$$\max\{t, D_{S_i}\} + \frac{C_j}{U_{S_i}},$$

where $C_j$ is the required (worst) computation time of the job and $D_{S_i}$ is the server deadline.

- The server deadline $D_{S_i}$ is also updated to the above absolute deadline.
An Example of TBS

$\tau_1 = (3, 6, 6), \ \tau_2 = (2, 8, 8)$.

TBS: $U_S = 0.25$

At time $t = 0$, $D_{S_i} = -\infty$

At time $t = 3$, $D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 3 + 4 = 7$

At time $t = 9$, $D_{S_i} = \max\{t, D_{S_i}\} + \frac{2}{U_S} = 9 + 8 = 17$

At time $t = 14$, $D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 17 + 4 = 21$
An Example of TBS

\( \tau_1 = (3, 6, 6), \tau_2 = (2, 8, 8). \)

TBS: \( U_S = 0.25 \)

At time \( t = 0 \), \( D_{Si} = -\infty \)

At time \( t = 3 \), \( D_{Si} = \max\{t, D_{Si}\} + \frac{1}{U_S} = 3 + 4 = 7 \)

At time \( t = 9 \), \( D_{Si} = \max\{t, D_{Si}\} + \frac{2}{U_S} = 9 + 8 = 17 \)

At time \( t = 14 \), \( D_{Si} = \max\{t, D_{Si}\} + \frac{1}{U_S} = 17 + 4 = 21 \)
An Example of TBS

\( \tau_1 = (3, 6, 6), \ \tau_2 = (2, 8, 8). \)

TBS: \( U_S = 0.25 \)

At time \( t = 0, \ D_{S_i} = -\infty \)

At time \( t = 3, \ D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 3 + 4 = 7 \)

At time \( t = 9, \ D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 3 + 4 = 7 \)

At time \( t = 14, \ D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 3 + 4 = 7 \)

Resource Reservation Servers

July 18, 2013
An Example of TBS

\( \tau_1 = (3, 6, 6), \tau_2 = (2, 8, 8). \)

TBS: \( U_S = 0.25 \)

\[ \begin{align*}
\text{At time } t = 0, & \quad D_{S_i} = -\infty \\
\text{At time } t = 3, & \quad D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 3 + 4 = 7 \\
\text{At time } t = 9, & \quad D_{S_i} = \max\{t, D_{S_i}\} + \frac{2}{U_S} = 9 + 8 = 17 \\
\text{At time } t = 14, & \quad D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 17 + 4 = 21
\end{align*} \]
An Example of TBS

\( \tau_1 = (3, 6, 6), \tau_2 = (2, 8, 8). \)

TBS: \( U_S = 0.25 \)

At time \( t = 0 \), \( D_{Si} = -\infty \)
At time \( t = 3 \), \( D_{Si} = \max\{t, D_{Si}\} + \frac{1}{U_S} = 3 + 4 = 7 \)
At time \( t = 9 \), \( D_{Si} = \max\{t, D_{Si}\} + \frac{2}{U_S} = 9 + 8 = 17 \)
At time \( t = 14 \), \( D_{Si} = \max\{t, D_{Si}\} + \frac{1}{U_S} = 17 + 4 = 21 \)
An Example of TBS

\[ \tau_1 = (3, 6, 6), \tau_2 = (2, 8, 8). \]

TBS: \[ U_S = 0.25 \]

At time \( t = 0 \), \[ D_{S_i} = -\infty \]

At time \( t = 3 \), \[ D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 3 + 4 = 7 \]

At time \( t = 9 \), \[ D_{S_i} = \max\{t, D_{S_i}\} + \frac{2}{U_S} = 9 + 8 = 17 \]

At time \( t = 14 \), \[ D_{S_i} = \max\{t, D_{S_i}\} + \frac{1}{U_S} = 17 + 4 = 21 \]
Properties of TBS

- Schedulability Guarantee:
  - Suppose that there are $n$ periodic tasks and a total bandwidth server with utilization $U_S$.
  - The schedulability of whole task set is guaranteed if
    \[
    U_S + \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1.
    \]

Again, prove by contradiction.

The key point is that the total execution time demanded by aperiodic requests arrived at time $t_1$ or later and served with deadline less than or equal to $t_2$ is no more than $(t_2 - t_1)U_S$. 
Constant Bandwidth Server (CBS)

- **Behavior**: assign the absolute deadline to an incoming job such that the utilization for CBS server $S_i$ is at most $U_{S_i} = C_{S_i} / T_{S_i}$.
  - Capacity: $C_{S_i}$
  - Period: $T_{S_i}$

- **Initialization**: assign server deadline $D_{S_i}$ to $-\infty$ and budget $b_{S_i}$ to 0.

- **Definition**: The server is **active** at time $t$ if there are pending jobs; otherwise it is **idle**.

- **Deadline assignment rule**:
  - When the server is idle at time $t$ and a job arrives, if $t < D_{S_i}$ and $b_{S_i} / (D_{S_i} - t) < C_{S_i} / T_{S_i}$, the server becomes active with the same budget and server deadline; otherwise, $D_{S_i}$ is set to $t + T_{S_i}$ and $b_{S_i}$ is set to $C_{S_i}$.
  - The first job in the ready queue of server $S_i$ is assigned the current server deadline $D_{S_i}$.
  - The budget $b_{S_i}$ is decreased by the served execution (computation) time while the server is executing jobs.
  - When $b_{S_i}$ reaches 0, the new server deadline $D_{S_i}$ becomes $D_{S_i} + T_{S_i}$ and $b_{S_i}$ is replenished to $C_{S_i}$ immediately.
An Example of CBS

τ₁ = (4, 7, 7)

CBS: Cₛ = 3, Tₛ = 8

At time t = 0, Dₛᵢ = −∞. At time t = 3, Dₛᵢ = 11 and bₛᵢ = 3
At time t = 7, bₛᵢ becomes 0 and immediately is replenished to 3 by setting Dₛᵢ = 19.
At time t = 13, since \( \frac{bₛᵢ}{19-13} < \frac{3}{8} \), server becomes active with the same deadline/budget.
At time t = 15, bₛᵢ becomes 0 and immediately is replenished to 3 by setting Dₛᵢ = 27.
An Example of CBS

\( \tau_1 = (4, 7, 7) \)

CBS: \( C_S = 3, \ T_S = 8 \)

At time \( t = 0, \ D_{S_i} = -\infty \). At time \( t = 3, \ D_{S_i} = 11 \) and \( b_{S_i} = 3 \).
At time \( t = 7, \ b_{S_i} \) becomes 0 and immediately is replenished to 3 by setting \( D_{S_i} = 19 \).
At time \( t = 13, \) since \( \frac{b_{S_i}}{19-13} < \frac{3}{8} \), server becomes active with the same deadline/budget.
At time \( t = 15, \ b_{S_i} \) becomes 0 and immediately is replenished to 3 by setting \( D_{S_i} = 27 \).
An Example of CBS

\[ \tau_1 = (4, 7, 7) \]

CBS: \( C_S = 3 \), \( T_S = 8 \)

At time \( t = 0 \), \( D_{S_i} = -\infty \). At time \( t = 3 \), \( D_{S_i} = 11 \) and \( b_{S_i} = 3 \).
At time \( t = 7 \), \( b_{S_i} \) becomes 0 and immediately is replenished to 3 by setting \( D_{S_i} = 19 \).
At time \( t = 13 \), since \( \frac{b_{S_i}}{19-13} < \frac{3}{8} \), server becomes active with the same deadline/budget.
At time \( t = 15 \), \( b_{S_i} \) becomes 0 and immediately is replenished to 3 by setting \( D_{S_i} = 27 \).
An Example of CBS

\( \tau_1 = (4, 7, 7) \)

CBS: \( C_S = 3, \ T_S = 8 \)

At time \( t = 0 \), \( D_{Si} = -\infty \). At time \( t = 3 \), \( D_{Si} = 11 \) and \( b_{Si} = 3 \).

At time \( t = 7 \), \( b_{Si} \) becomes 0 and immediately is replenished to 3 by setting \( D_{Si} = 19 \).

At time \( t = 13 \), since \( \frac{b_{Si}}{19-13} < \frac{3}{8} \), server becomes active with the same deadline/budget.

At time \( t = 15 \), \( b_{Si} \) becomes 0 and immediately is replenished to 3 by setting \( D_{Si} = 27 \).
Another Example of CBS

\( \tau_1 = (8, 14, 14) \)
CBS: \( C_S = 3, \ T_S = 8 \)

At time \( t = 0 \), \( D_{Si} = -\infty \). At time \( t = 3 \), \( D_{Si} = 11 \) and \( b_{Si} = 3 \).
At time \( t = 6 \), \( b_{Si} \) becomes 0 and immediately is replenished to 3 by setting \( D_{Si} = 19 \).
At time \( t = 16 \), since \( \frac{b_{Si}}{19-16} > \frac{3}{8} \), server becomes active by setting \( b_{Si} = 3 \) and \( D_{Si} = 16 + 8 = 24 \).
Another Example of CBS

$\tau_1 = (8, 14, 14)$

CBS: $C_S = 3$, $T_S = 8$

At time $t = 0$, $D_{Si} = -\infty$. At time $t = 3$, $D_{Si} = 11$ and $b_{Si} = 3$
At time $t = 6$, $b_{Si}$ becomes 0 and immediately is replenished to 3 by setting $D_{Si} = 19$.
At time $t = 16$, since $\frac{b_{Si}}{19-16} > \frac{3}{8}$, server becomes active by setting $b_{Si} = 3$ and $D_{Si} = 16 + 8 = 24$. 
Another Example of CBS

\[ \tau_1 = (8, 14, 14) \]

CBS: \( C_S = 3, \ T_S = 8 \)

At time \( t = 0 \), \( D_S_i = -\infty \). At time \( t = 3 \), \( D_S_i = 11 \) and \( b_{S_i} = 3 \).
At time \( t = 6 \), \( b_{S_i} \) becomes 0 and immediately is replenished to 3 by setting \( D_S_i = 19 \).
At time \( t = 16 \), since \( \frac{b_{S_i}}{19-16} > \frac{3}{8} \), server becomes active by setting \( b_{S_i} = 3 \) and \( D_S_i = 16 + 8 = 24 \).
Another Example of CBS

\[ \tau_1 = (8, 14, 14) \]

CBS: \( C_S = 3, \ T_S = 8 \)

At time \( t = 0 \), \( D_{S_i} = -\infty \). At time \( t = 3 \), \( D_{S_i} = 11 \) and \( b_{S_i} = 3 \)

At time \( t = 6 \), \( b_{S_i} \) becomes 0 and immediately is replenished to 3 by setting \( D_{S_i} = 19 \).

At time \( t = 16 \), since \( \frac{b_{S_i}}{19-16} > \frac{3}{8} \), server becomes active by setting \( b_{S_i} = 3 \) and \( D_{S_i} = 16 + 8 = 24 \).
Properties of CBS

- Schedulability Guarantee:
  ▶ Suppose that there are $n$ periodic tasks and a constant bandwidth server with utilization $U_S$.
  ▶ The schedulability of whole task set is guaranteed if

$$U_S + \sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1.$$ 

The key point is the isolation property, in which

- the CPU utilization of a CBS server is $U_S$, independently from the computation times and the arrival pattern of the served jobs.
  \[\rightarrow\] No need to know WCETs of aperiodic tasks.
Resource reservation servers are used to safely schedule *aperiodic tasks* while maintaining schedulability of periodic tasks.

Trade-off between implementation complexity, utilization bounds, and guarantees for aperiodic tasks, both for fixed- and dynamic-priority systems.

- Sporadic Server: same utilization bound of pure RM scheduling
- Constant Bandwidth Server: same utilization bound as EDF, isolation property, no need to know WCETs of aperiodic tasks

Next week: taking into account preemption costs