A Stack-Based Resource Allocation Policy for Realtime Processes

T.P. Baker
Department of Computer Science
Florida State University
Tallahassee, FL 32304-4019

Presented by Tsai Lin-Jiun
Outline

- Abstract & Introduction
- Definitions
- Stack Resource Policy (SRP)
- Schedulability
- Relation to Priority Ceiling Protocol (PCP)
- Implementation Consideration, Conclusions, and Further Research
Abstract & Introduction

- share a single runtime stack
- if Job is preempted it can’t resume until all the jobs that occupy stack space above it have completed
- refinement of the Priority Ceiling Protocol
SRP offers improvements over the PCP.
- unifying the treatment of stack, reader-writer, and multiunit resources, and binary semaphores
- applying directly to some dynamic scheduling policies, including EDF, as well as to static priority policies
- with EDF scheduling, supporting a stronger schedulability test
- reducing the maximum number of context switches by a factor of two
Outline

- Abstract & Introduction
- Definitions
- Stack Resource Policy (SRP)
- Schedulability
- Relation to Priority Ceiling Protocol (PCP)
- Implementation Consideration, Conclusions, and Further Research
Definitions

- **Jobs**
  - a finite sequence of instructions to be executed on a single processor
  - pending requests are classified as *waiting*, meaning the job hasn’t yet started
  - *active*, meaning the job has started to execute
  - process $P_i$ is an (infinite) sequence of job execution requests $J_{i,1}, J_{i,2}, J_{i,3} ...$
Definitions (contd.)

- Resources
  - assume there is a single processor, which is preemptable, and a finite set of nonpreemptable resources \( R_i, \ldots, R_m \)
  - \((J,R,m)\): a job, \( R \): a nonpreemptable resource, \( m \): a mode (read = 1, write = \( N_R \) (total # of R))
  - while a job holds an allocation, says outstanding
  - LIFO request order, overlap if properly nested
Definitions (contd.)

Stack Space

- Shared runtime stack space is a nonpreemptable resource
- 1. request at least 1 cell before execution, can’t relinquish until completes execution, entire execution of each job is a critical section
- 2. it must continue to hold its stack resources while it is blocked for some request
- 3. request can be granted iff is not yet holding any space or holding the top of the stack
- 4. only the job at the top may execute (grow up)
Definitions (contd.)

- **Direct blocking**
  - \((J, R, m)\) is blocked directly iff \(V_R < m\)
  - identifiable set of other jobs that are blocking \(J\)
  - job \(J\) is directly blocked iff there’s another job \(J'\) holding the space immediately above the space occupied by \(J\) on the stack
Definitions (contd.)

- Priorities
  - \( J \) has higher priority than \( J' \) iff \( p(J) > p(J') \)
  - larger values indicate greater urgency
  - preemptable according to the priorities of requests and FIFO among jobs of equal priority
Definitions (contd.)

- Preemption levels $\pi(J)$
  - statically assigned to jobs
  - $J'$ isn’t allowed to preempt another job $J$ unless $\pi(J') > \pi(J)$
  - enable static analysis of potential resource conflicts, even for dynamic priority scheme
  - $p(J) < p(J')$ iff $t' + D' < t + D$ (by EDF)
  - $\pi(J) < \pi(J')$ iff $D(J') < D(J)$
  - $J'$ can never be preempted by $J$, but this doesn’t mean that $J'$ always have higher priority than $J$
Definitions (contd.)

- Preemption levels (contd.)
  - $\pi(J) < \pi(J')$: 
  - $p(J) > p(J')$ or $p(J') > p(J)$ can preempt $J$
Outline

- Abstract & Introduction
- Definitions
- Stack Resource Policy (SRP)
- Schedulability
- Relation to Priority Ceiling Protocol (PCP)
- Implementation Consideration, Conclusions, and Further Research
Stack Resource Policy (SRP)

- Unify and extend definition of priority ceiling
  - Priorities are replaced by preemption levels. This allows EDF priorities to be handled without requiring to recompute ceilings at run time.
  - Ceilings are defined for multiunit resources, subsuming both binary semaphores and r/w locks.

- Abstract ceilings
  - If $J$ is currently executing or can preempt the currently executing job, and may request an allocation of $R$ that would be blocked directly by the outstanding allocation of $R$, then $\lceil R \rceil \geq \pi(J)$
Stack Resource Policy (SRP) (contd.)

- **Specific ceilings**
  - \( \lceil R \rceil_{VR} = \max(\{0\} \cup \{\pi(J) \mid V_R < \mu_R(J)\}) \)
  - \( V_R \) units of \( R \) available
  - \( \mu_R(J) \) is the maximum number of units of \( R \) that job \( J \) may need to hold at any one time
Stack Resource Policy (SRP) (contd.)

- Specific ceilings (contd.)

```
<table>
<thead>
<tr>
<th>R</th>
<th>N_R</th>
<th>μ_1</th>
<th>μ_2</th>
<th>μ_3</th>
<th>\hat{R}_0</th>
<th>\hat{R}_1</th>
<th>\hat{R}_2</th>
<th>\hat{R}_3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>R_2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>R_3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Stack Resource Policy (SRP) (contd.)

- **Current ceiling**
  - $\pi' = \max(\lceil R_i \rceil | i = 1,\ldots,m) \cup \{\pi(J_c)\}$
  - if there’re no jobs currently execute, $\pi' = 0$

- **the SRP**
  - requires that a job execution request $J$ be blocked from starting execution until $\pi' < \pi(J)$
  - once $J$ has started execution, all subsequent resource request by $J$ are granted immediately
  - doesn’t restrict the resource acquiring order, and allocate only when requests.
Stack Resource Policy (SRP) (contd.)

- The SRP (contd.)
  - Release resources when they are not needed.
  - $J_H$ is free to preempt until $J$ actually requests enough of $R$ to block $J_H$ (without being blocked).

- Examples
  - Solid horizontal lines indicate job executions.
  - Barred lines indicate $\pi'$.
  - Ex1: Since $\lceil R_2 \rceil_0 = 2$, $J_2$ is unable to preempt $J_1$ after it acquire $R_2$. $J_3$ preempts $J_1$ as soon as $J_1$ release $R_1$. 
Stack Resource Policy (SRP) (contd.)

- the SRP (contd.)

\[ J_1 \text{ acquires } R_2 \]
\[ J_2 \text{ arrives} \]
\[ J_1 \text{ acquires } R_1 \]
\[ J_3 \text{ arrives} \]
\[ J_1 \text{ releases } R_1 \]
\[ J_3 \text{ acquires } R_3 \]
\[ J_3 \text{ releases } R_3 \]
\[ J_3 \text{ completes} \]
\[ J_1 \text{ releases } R_2 \]
Stack Resource Policy (SRP) (contd.)

- the SRP (contd.)
Stack Resource Policy (SRP) (contd.)

- Blocking properties of the SRP

**Theorem 1**

- If no job $J$ is permitted to start until $\pi' < \pi(J)$ =>
  - (a) No job can be blocked after it starts
  - (b) There can be no transitive blocking or deadlock
  - (c) If the oldest highest-priority job is blocked, it will become unblocked no later than the first instant that the currently executing job isn’t holding any nonpreemptable resources.
Outline

- Abstract & Introduction
- Definitions
- Stack Resource Policy (SRP)
- Schedulability
- Relation to Priority Ceiling Protocol (PCP)
- Implementation Consideration, Conclusions, and Further Research
Schedulability

Theorem 2

A set of \( n \) (periodic and aperiodic) jobs is schedulable by EDF scheduling if

\[
\forall k \quad \left( \sum_{i=1}^{k} \frac{C_i}{D_i} \right) + \frac{B_k}{D_k} \leq 1.
\]

\( B_i \): the execution time of the longest critical section of any job \( J_k \) such that \( D_i \leq D_k \) and \( i \neq k \)

\( C_i \): max execution time

\( D_i \): relative deadline
Outline

- Abstract & Introduction
- Definitions
- Stack Resource Policy (SRP)
- Schedulability
- Relation to Priority Ceiling Protocol (PCP)
- Implementation Consideration, Conclusions, and Further Research
Relation to PCP

- Ceiling are defined in terms of preemption levels, instead of priorities, so that the SRP applies directly to EDF scheduling (without dynamic recomputation of ceilings)
- Ceilings are defined for multiunit resources.
- Stack sharing is supported
- The blocking test is only applied when a job tries to start execution
Relation to PCP (contd.)

- Resources requests never block, and hence can’t require extra context switches (at most TWO!)
- Because there is no blocking after a job starts executing, a stronger EDF schedulability result can be obtained than with dynamic priority ceilings
- Different jobs of a process may have different priorities
Relation to PCP (contd.)

- Theorem 3
  - The maximum priority-inversion time of any job under the SRP is no longer than under the PCP

- Theorem 4
  - The SRP requires at most two context switches per job execution request

- Theorem 5
  - The PCP, like any other policy that waits to block a job until it makes a resource request, may require four context switches per job execution request, for any job that shares a semaphore with a lower priority job
Outline

- Abstract & Introduction
- Definitions
- Stack Resource Policy (SRP)
- Schedulability
- Relation to Priority Ceiling Protocol (PCP)
- Implementation Consideration, Conclusions, and Further Research
Implementation Consideration, Conclusions, and Further Research

- Simple and efficiently, similar to that of PCP, but simpler blocking operation.
- Ceilings $\lceil R \rceil_n$ are static in table
- $\pi' = \lceil R \rceil_{vr}$ iff $\pi' < \lceil R \rceil_{vr}$ when $V_R$ is updated, and the old $\pi'$ and $V_R$ are pushed on the stack (be restored later and check whether waiting jobs to preempt)
- refinement version of SRP, the Minimal SRP (MSRP) is developed
Thank you