Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

RTAS’13
April 10, 2013

Björn B. Brandenburg
bbb@mpi-sws.org
Semaphores + P-FP Scheduling

Used in practice: VxWorks, QNX, ThreadX, Real-Time Linux variants, …

Binary Semaphores in POSIX

```c
pthread_mutex_lock(…)
// critical section
pthread_mutex_unlock(…)
```

**binary semaphore**
A blocked task suspends & yields processor.

**Partitioned Fixed-Priority (P-FP) scheduling**
Tasks statically assigned to cores.
How to Implement Semaphores?

How to order conflicting critical sections?

→ FIFO vs. Priority Queues
How to Implement Semaphores?

*How to order conflicting critical sections?*

→ **FIFO vs. Priority** Queues

*Where to execute critical sections?*

→ **Shared-Memory vs. Distributed** Locking Protocols
Example: Shared-Memory Protocol

CPU 1

CPU 2

Task A critical section

CPU 3

→ time
Example: **Shared-Memory Protocol**

**(CPU 1 not affected by critical sections on other CPUs)**

- **CPU 1**
  - Task A
  - Critical section

- **CPU 2**
  - Task A
  - Critical section

- **CPU 3**
  - Task B
  - Task suspended
  - Critical section

Events:
- Lock requested
- Lock acquired
- Lock released
Example: Distributed Protocol
Example: Distributed Protocol

CPU 1

resource agent inactive

critical section A

CPU 2

Task A

request issued

task suspended

CPU 3

response received

time
Example: **Distributed Protocol**

- **CPU 1**
  - Critical section A
  - Critical section B
  - Resource agent inactive

- **CPU 2**
  - Task A
  - Request issued
  - Task suspended
  - Response received

- **CPU 3**
  - Task B
  - Request issued
  - Task suspended
  - Response received

Time line:
- Request issued
- Response received
# Semaphore Protocol Choices

How to **order** conflicting critical sections?

Where **to execute** critical sections?

<table>
<thead>
<tr>
<th>Wait Queue</th>
<th>Protocol Type</th>
<th>priority</th>
<th>FIFO</th>
<th>priority</th>
<th>FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority queue</td>
<td>shared-memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIFO queue</td>
<td>shared-memory</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>priority queue</td>
<td>distributed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIFO queue</td>
<td>distributed</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Protocol Choices

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Wait Queue</th>
<th>Protocol Type</th>
<th>Wait Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>priority queue</td>
<td></td>
<td>priority queue</td>
</tr>
<tr>
<td></td>
<td>FIFO queue</td>
<td></td>
<td>FIFO queue</td>
</tr>
<tr>
<td>MPCP (^{(1)})</td>
<td></td>
<td>FMLP+ (^{(2)})</td>
<td></td>
</tr>
<tr>
<td>(^{(1)})Rajkumar, 1990; Lakshmanan et al., 2009| (^{(2)})Brandenburg, 2011| (^{(3)})Rajkumar et al., 1988| (^{(4)})Brandenburg, 2012|</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multiprocessor Priority Ceiling Protocol</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>shared-memory</th>
<th>shared-memory</th>
<th>distributed</th>
<th>distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^{(1)}\) Rajkumar, 1990; Lakshmanan et al., 2009
\(^{(2)}\) Brandenburg, 2011
\(^{(3)}\) Rajkumar et al., 1988
\(^{(4)}\) Brandenburg, 2012
## Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

### Semaphore Choices

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)</th>
<th>FMLP+ (Brandenburg, 2011)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait Queue</td>
<td>priority queue</td>
<td>FIFO queue</td>
</tr>
<tr>
<td>Protocol Type</td>
<td>shared-memory</td>
<td>shared-memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>distributed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>distributed</td>
</tr>
</tbody>
</table>

**FIFO Multiprocessor Locking Protocol**

- **Asymptotically optimal** (w.r.t. maximum blocking)
## Semaphore Protocol Choices

<table>
<thead>
<tr>
<th></th>
<th><strong>MPCP</strong></th>
<th></th>
<th><strong>FMLP+</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Rajkumar, 1990; Lakshmanan et al., 2009)</td>
<td></td>
<td>(Brandenburg, 2011)</td>
<td></td>
</tr>
<tr>
<td>Wait Queue</td>
<td><strong>priority</strong> queue</td>
<td><strong>FIFO</strong> queue</td>
<td><strong>priority</strong> queue</td>
<td><strong>FIFO</strong> queue</td>
</tr>
<tr>
<td>Protocol Type</td>
<td><strong>shared-memory</strong></td>
<td><strong>shared-memory</strong></td>
<td><strong>distributed</strong></td>
<td><strong>distributed</strong></td>
</tr>
<tr>
<td>Asymptotically optimal? (w.r.t. maximum blocking)</td>
<td>NO</td>
<td>YES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

### Semaphore Protocol Choices

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Wait Queue</th>
<th>Protocol Type</th>
<th>Wait Queue</th>
<th>Asymptotically optimal? (w.r.t. maximum blocking)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MPCP</strong> (Rajkumar, 1990; Lakshmanan et al., 2009)</td>
<td>priority queue</td>
<td><strong>FMLP+</strong> (Brandenburg, 2011)</td>
<td>FIFO queue</td>
<td>NO</td>
</tr>
<tr>
<td><strong>FMLP+</strong> (Brandenburg, 2011)</td>
<td>FIFO queue</td>
<td><strong>DPCP</strong> (Rajkumar et al., 1988)</td>
<td>priority queue</td>
<td>YES</td>
</tr>
<tr>
<td><strong>DPCP</strong> (Rajkumar et al., 1988)</td>
<td>FIFO queue</td>
<td></td>
<td></td>
<td>NO</td>
</tr>
</tbody>
</table>

### Protocol Types

- **MPCP**: Multi-Process Ceiling Protocol
- **FMLP+**: Fair Memory Limited Protocol +
- **DPCP**: Distributed Priority Ceiling Protocol
### Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

#### Distributed FIFO Locking Protocol

<table>
<thead>
<tr>
<th>Wait Queue</th>
<th>Protocol Type</th>
<th>Asymptotically optimal? (w.r.t. maximum blocking)</th>
</tr>
</thead>
<tbody>
<tr>
<td>priority queue</td>
<td>shared-memory</td>
<td>NO</td>
</tr>
<tr>
<td>FIFO queue</td>
<td>shared-memory</td>
<td>YES</td>
</tr>
<tr>
<td>priority queue</td>
<td>distributed</td>
<td>NO</td>
</tr>
<tr>
<td>FIFO queue</td>
<td>distributed</td>
<td>YES</td>
</tr>
</tbody>
</table>

**Semaphore Protocol Choices**

- **MPCP** (Rajkumar, 1990; Lakshmanan et al., 2009)
- **FMLP** (Brandenburg, 2011)
- **DPCP** (Rajkumar et al., 1988)
- **DFLP** (Brandenburg, 2012)
### Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

**Semaphore Protocol Choices**

- **MPCP** *(Rajkumar, 1990; Lakshmanan et al., 2009)*
- **FMLP+** *(Brandenburg, 2011)*
- **DPCP** *(Rajkumar et al., 1988)*
- **DFLP** *(Brandenburg, 2012)*

<table>
<thead>
<tr>
<th>Wait Queue</th>
<th>Protocol Type</th>
<th>Asymptotically optimal? (w.r.t. maximum blocking)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>priority queue</td>
<td><strong>NO</strong></td>
</tr>
<tr>
<td></td>
<td>FIFO queue</td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td></td>
<td>priority queue</td>
<td><strong>NO</strong></td>
</tr>
<tr>
<td></td>
<td>FIFO queue</td>
<td><strong>YES</strong></td>
</tr>
</tbody>
</table>

**Asymptotically** FIFO queues offer lower maximum blocking.

**But:** constants matter…
Part 1

Improved Analysis
Non-Asymptotic, Fine-Grained Analysis

Derive tightest possible bound reflecting all constant factors.

conflicting critical sections

pending interval (vulnerable to contention)

job under analysis
Non-Asymptotic Fine-Grained Analysis

Derive tightest possible bound reflecting all constant factors.

Exploit activation frequency.
Don’t overestimate worst-case contention.

pending interval (vulnerable to contention)

job under analysis

conflicting critical sections
Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

Non-Asymptotic, Fine-Grained Analysis

Derive tightest possible bound reflecting all constant factors.

Conflicting critical sections

Exploit activation frequency. Don’t overestimate worst-case contention.

Exploit per-task maximum critical section lengths. Don’t overestimate worst-case duration of lock unavailability.

Pending interval (vulnerable to contention)

Job under analysis

Time
Non-Asymptotic Fine-Grained Analysis

Derive tightest possible bound reflecting all constant factors.

- Exploit activation frequency. Don’t overestimate worst-case contention.
- Exploit per-task maximum critical section lengths. Don’t overestimate worst-case duration of lock unavailability.
- Exploit protocol-specific properties. E.g., strong progress in FIFO queues.

Conflicting critical sections

Pending interval (vulnerable to contention)

Job under analysis

Time
Non-Asymptotic, Fine-Grained Analysis

Derive tightest possible bound reflecting all constant factors.

The key problem:

Ease of exposition vs. pessimism!
Declarative Fine-Grained Analysis

Concise.

Easy to write, easy to read, easy to check.

Not inherently pessimistic.

**Compositional**: analyst should not have to reason about protocol as a whole.

Easy to implement.

**Sound by construction**.

Not based on ad-hoc formalism.
Declarative Fine-Grained Analysis

Approach

Use linear programming to derive task-set-specific blocking bounds.

(not integer linear programming!)
Blocking Fractions

with regard to a fixed schedule (i.e., one particular execution)

For the \( v^{th} \) concurrent critical section of conflicting task \( T_x \) w.r.t. resource \( q \):

\[
X_{x,q,v} = \frac{\text{actual amount of blocking caused}}{\text{maximum critical section length w.r.t. } q}
\]

\( 0 \leq X_{x,q,v} \leq 1 \)
Blocking Fraction — Example

suppose maximum critical section length of task $T_x$ is 3 time units

Suppose maximum critical section length of task $T_x$ is 3 time units.

The diagram illustrates the execution of tasks $T_i$ and $T_x$ on CPUs 1 and 2, respectively. Task $T_x$ enters its critical section from time 2 to time 5, while task $T_i$ is blocked from time 2 to time 3 due to the critical section of $T_x$.
Blocking Fraction — Example

suppose *maximum* critical section length of task $T_x$ is *3 time units*

In this particular schedule

*actual blocking* = 1 time unit

→ blocking fraction = \( \text{actual} / \text{max} = 1/3 \)

**suppose maximum critical section length of task $T_x$ is 3 time units**

**suppose maximum critical section length of task $T_x$ is 3 time units**

In this particular schedule

*actual blocking* = 1 time unit

→ blocking fraction = \( \text{actual} / \text{max} = 1/3 \)
Total Blocking in a Fixed Schedule

\[ \sum_{\text{each resource } q} \sum_{\text{each task } T_x} \sum_{\text{each CS } v} X_{x,q,v} \cdot \text{maximum critical section length w.r.t. } q \]

\[ X_{x,q,v} = \frac{\text{actual amount of blocking caused}}{\text{maximum critical section length w.r.t. } q} \]
Total Blocking in a Fixed Schedule

total blocking incurred by one job =

\[ \sum_{\text{each resource } q} \sum_{\text{each task } T_x} \sum_{\text{each CS } v} X_{x,q,v} \cdot \text{maximum critical section length w.r.t. } q \]

Actual amount of blocking incurred (may be zero).

All potentially concurrent critical sections.
(No cleverness and hence no errors involved!)

\[ X_{x,q,v} = \frac{\text{actual amount of blocking caused}}{\text{maximum critical section length w.r.t. } q} \]
Total Blocking Incurred by One Job

\[
\sum_{\text{each resource } q} \sum_{\text{each task } T_x} \sum_{\text{each CS } v} X_{x,q,v} \cdot \text{maximum critical section length w.r.t. } q
\]

\[X_{x,q,v} = \frac{\text{actual amount of blocking caused}}{\text{maximum critical section length w.r.t. } q}\]
From a Fixed to All Possible Schedules

\[
\text{maximize} \quad \sum_{\text{each resource } q} \sum_{\text{each task } T_x} \sum_{\text{each CS } v} X_{x,q,v} \cdot \text{maximum critical section length w.r.t. } q
\]

subject to

$\text{WORKLOAD-CONSTRAINTS}$

$\text{PROTOCOL-CONSTRAINTS}$
**From a Fixed to All Possible Schedules**

\[ \sum_{\text{each resource } q} \sum_{\text{each task } T_x} \sum_{\text{each CS } v} x_{x,q,v} \cdot \text{maximum critical section length w.r.t. } q \]

**maximize**

Find **worst-case blocking** across **all possible schedules**.

**subject to**

\[ \text{WORKLOAD-CONSTRAINTS} \]

\[ \text{PROTOCOL-CONSTRAINTS} \]

Rule out **impossible schedules**.
Example FIFO Constraint

rule out impossible schedules not compliant with FIFO ordering

Constraint 12. In any schedule of $\tau$ under the FMLP$^+$:

$$\forall \ell_q : \forall T_x : \sum_{v=1}^{N^i_{x,q}} X_{x,q,v} \leq N_{i,q}.$$
Example FIFO Constraint

For each resource $q$ and each conflicting task $T_x$...

Constraint 12. In any schedule of $\tau$ under the FMLP$^+$:

$$\forall \ell_q : \forall T_x : \sum_{v=1}^{N^i_{x,q}} X_{x,q,v} \leq N_{i,q}.$$
Example FIFO Constraint

For each resource \( q \) and each conflicting task \( T_x \)...

...all possibly concurrent critical sections \( v \)...

Constraint 13. In any schedule of \( \tau \) under the FMLP\(^+\):

\[
\forall \ell_q : \forall T_x \sum_{v=1}^{N_i,q} X_{x,q,v} \leq N_{i,q}.
\]
Thus, at most \( \min \) conflicting requests for tasks on processor \( x \) of critical sections of any delays one of another task most once per request, either directly or indirectly, but not both.

Analysis pertaining to the DFLP applies analogously to the contention (among priority-boosted jobs). In fact, most of the job ordering, with regard to both lock contention and processor the FMLP and indirect request delay under the FMLP to delay resumes, each other local, lower-priority task a resource),

\[ \sum_{u=1}^{N_i} X_{x,q,v} \leq N_{i,q}. \]

Example FIFO Constraint with FIFO ordering

For each resource \( q \) and each conflicting task \( T_x \)...

...all possibly concurrent critical sections \( v \)...

Constraint 12. In any schedule of \( \tau \) under the FMLP\(^+\):

...the sum of all blocking fractions...

...cannot exceed the number of requests for the resource issued by task \( T_i \) (the task under analysis).
Example FIFO Constraint

rule out impossible schedules not compliant with FIFO ordering

Constraint 12. In any schedule of $\tau$ under the FMLP$^+$:

$$\forall \ell_q : \forall T_x$$

$$\sum_{v=1}^{N^i_{x,q}} X_{x,q,v} \leq N_{i,q}.$$

Suppose not: then there exists a schedule in which the sum of the blocking fractions of one task $T_x$ exceeds the number of requests issued.

Thus one request of $T_i$ must have been blocked by at least two requests of $T_x$. This is impossible in a FIFO queue.
Advantages

Constraint 12. In any schedule of $\tau$ under the FMLP$^+$:

$$\forall \ell_q : \forall T_x \in \tau^i : \sum_{v=1}^{N_i^x,q} X_{x,q,v}^D \leq N_{i,q}.$$ 

Powerful analysis technique

- **compositional**: LP solver combines constraints
- **flexible**: can handle many protocols
- **declarative**: much easier to read and check

Accuracy

- **Never counts a request twice.**
- Much less pessimistic…
Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

**Improved Analysis Accuracy**

- **Fraction of schedulable task sets (out of 1000)**
  - **“higher is better”**
  - **Higher task count = less idle time and more contention**

**Plot Parameters**
- **x-axis:** Number of tasks (n)
- **y-axis:** Fraction of schedulable task sets

---

**Legend**
- Green arrow indicates improved analysis accuracy.
- Orange arrow signifies worsening accuracy due to increased task count.
Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

Improved Analysis Accuracy

16 cores, 16 shared resources, max. 5 critical sections per task and resource, 10µs-50µs CS length, each task accesses a given resource with probability 0.1

- MPCP (LP)
- FMLP+ (LP)
- MPCP (prior)

![Graph showing schedulability for different protocols](image-url)

The graph illustrates the fraction of schedulable task sets over the number of tasks for different protocols. It highlights the improved analysis accuracy compared to previous versions.

Mathematical notation and details are omitted for brevity, focusing on the overall trend and comparison among protocols.
Improved Analysis and Evaluation of Real-Time Semaphore Protocols for P-FP Scheduling

Improved Analysis Accuracy

16 cores, 16 shared resources, max. 5 critical sections per task and resource, 10µs-50µs CS length, each task accesses a given resource with probability 0.1

fraction of schedulable task sets

0 0.2 0.4 0.6 0.8 1

number of tasks (n)

20 30 40 50 60 70 80 90 100 110 120 130 140 150 160

MPCP (LP)
FMLP+ (LP)
MPCP (prior)

MPCP (with prior analysis)
FMLP+ (with new analysis)

MPCP (with new analysis)
Improved Analysis Accuracy

16 cores, 16 shared resources, max. 5 critical sections per task and resource, 10µs-50µs CS length, each task accesses a given resource with probability 0.1

New analysis roughly **doubled number of supported tasks**!

*And: offers new observations on MPCP / FMLP*⁺ relative performance.*
Part 2

Improved Evaluation
## Evaluated Protocols

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Asymptotically optimal? (w.r.t. maximum blocking)</th>
<th>MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)</th>
<th>FMLP+ (Brandenburg, 2011)</th>
<th>DPCP (Rajkumar et al., 1988)</th>
<th>DFLP (Brandenburg, 2012)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait Queue</td>
<td>priority queue</td>
<td>FIFO queue</td>
<td>priority queue</td>
<td>FIFO queue</td>
<td></td>
</tr>
<tr>
<td>Protocol Type</td>
<td>shared-memory</td>
<td>shared-memory</td>
<td>distributed</td>
<td>distributed</td>
<td></td>
</tr>
<tr>
<td>Asymptotically optimal?</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>
Distributed locking protocols require cross-core interaction.

*Overheads matter…*

<table>
<thead>
<tr>
<th>Protocol Type</th>
<th>Wait Queue</th>
<th>Asymptotically optimal? (w.r.t. maximum blocking)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MPCP</strong> (Rajkumar, 1990; Lakshmanan et al., 2009)</td>
<td>priority queue</td>
<td>NO</td>
</tr>
<tr>
<td><strong>FMLP+</strong> (Brandenburg, 2011)</td>
<td>FIFO queue</td>
<td>YES</td>
</tr>
<tr>
<td><strong>DPCP</strong> (Rajkumar et al., 1988)</td>
<td>priority queue</td>
<td>NO</td>
</tr>
<tr>
<td><strong>DFLP</strong> (Brandenburg, 2012)</td>
<td>FIFO queue</td>
<td>YES</td>
</tr>
</tbody>
</table>
Taking Overheads Into Account

Platform

<table>
<thead>
<tr>
<th>LITMUSRT</th>
<th>Linux Testbed for Multiprocessor Scheduling in Real-Time Systems</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.litmus-rt.org">www.litmus-rt.org</a></td>
<td></td>
</tr>
</tbody>
</table>

| 8-core / 16-core |
| 2.0 GHz Intel Xeon X7550 |

Overhead Tracing
- > 20h of real-time execution
- > 400 GB of trace data
- > 15 billion valid samples
- Statistics and details in online appendix.
Example: Context Switch Overhead

P-FP: measured context-switch overhead (host=nanping-16)
min=0.26us max=40.99us avg=2.72us median=2.15us

max = 40.99µs  avg = 2.72µs  med = 2.15µs
Overhead Experiments:

No statistical outlier filtering was applied.

max = 40.99µs    avg = 2.72µs    med = 2.15µs
Overhead-Aware Schedulability

“higher is better”

higher task count = less idle time and more contention
Overhead-Aware Schedulability

8 cores, 8 shared resources, max. 5 critical sections per task and resource, 10µs-50µs CS length, each task accesses a given resource with probability 0.3
Distributed Protocols Perform Well

8 cores, 8 shared resources, max. 5 critical sections per task and resource, 10μs-50μs CS length, each task accesses a given resource with probability 0.3

![Graph showing performance of different semaphore protocols](image)

- Distributed Protocols
FIFO Protocols Perform Well

8 cores, 8 shared resources, max. 5 critical sections per task and resource, 10µs-50µs CS length, each task accesses a given resource with probability 0.3
Choice of Protocol Matters

Blocking is a significant bottleneck w.r.t. schedulability.
In this example: DFLP can host >20% more tasks than MPCP.
What is the “best” protocol?

All results available online (>6,000 plots).
What is the “best” protocol?

There is no single “best” protocol!
(w.r.t. schedulability)

Results are highly workload-dependent!
What is the “best” protocol?

*How to order conflicting critical sections?*

➔ FIFO works well, but *priority* queues needed for *highly heterogenous* timing parameters.

*Where to execute critical sections?*

➔ *Distributed* protocols very competitive for many resources with high contention; *shared-memory* protocols better for few resources.
Summary & Outlook

Contributions

- There is no single best protocol yet.
- Distributed protocols perform surprisingly well.
- Use linear programs to analyze blocking.

Future Work

- Generalized protocol that always works.
- Support clustered scheduling.
- Analyze spin locks with LPs.
- Extend LPs to handle nesting.
Thanks!

[Image of LitmusRT logo]

www.litmus-rt.org

[SchedCAT logo]

Schedulability test Collection And Toolkit

www.mpi-sws.org/~bbb/projects/schedcat