Implementation of Overrun and Skipping in VxWorks

Mikael Åsberg, Moris Behnam
and Thomas Nolte
MRTC/Mälardalen University
Sweden

Reinder J. Bril
Technische Universiteit Eindhoven
The Netherlands

PROGRESS
A national Swedish Strategic Research Centre

MÄLARDALEN UNIVERSITY

MRTC
MÄLARDALEN REAL-TIME
RESEARCH CENTRE

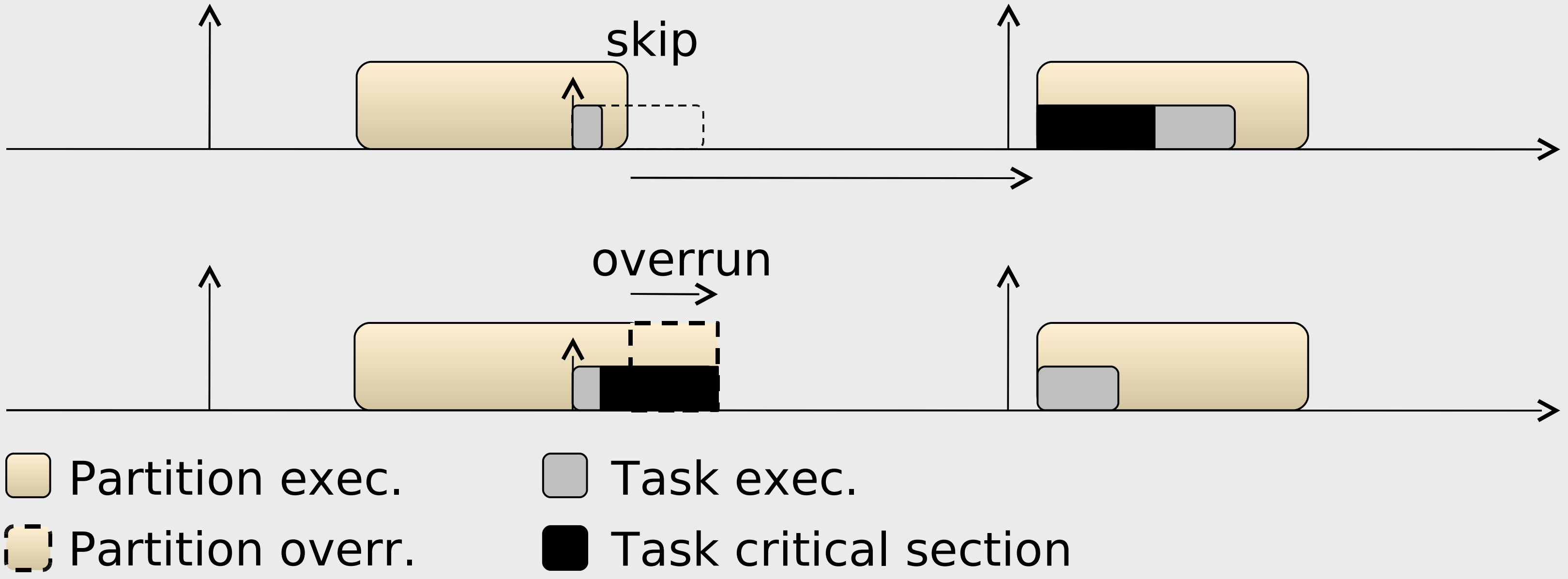
SCIENCE • TECHNOLOGY • MEDICINE
Swedish Foundation for Strategic Research

Overview

- Background (Resource sharing in the hierarchical scheduling environment)
  - Overrun
  - Skipping
- Preliminaries:
  - Stack resource policy (SRP)
  - Hierarchical scheduling framework (HSF)
- VxWorks implementation:
  - SRP
  - Overrun/Skipping
    - Common (Overrun/Skipping)
    - Overrun
    - Skipping
- Comparison (Overrun/Skipping)
- Evaluation
- Conclusion

Background

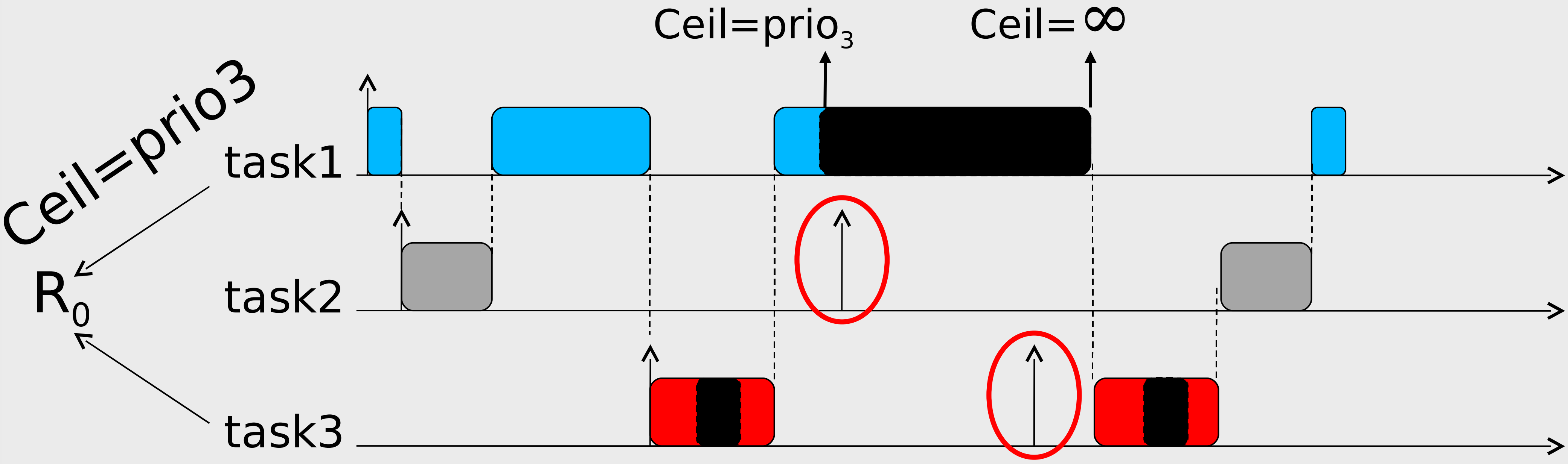
- Problem: Resource sharing across subsystems
- Solution: Protocols such as Overrun or Skipping
  - Overrun: extend budget to complete critical sec.
  - Skipping: Skip task exec. if budget is too small
- Both are based on SRP (at both subsystem and task level)



Preliminaries: SRP

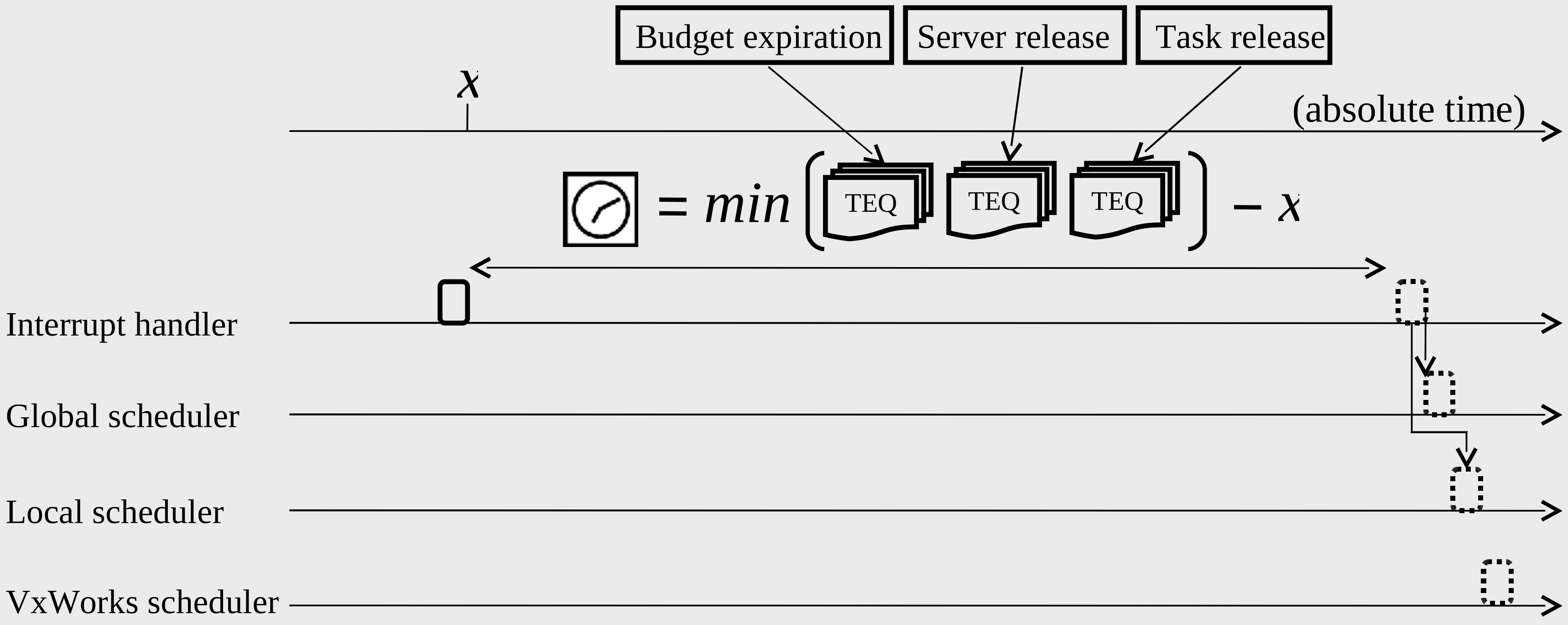
- Synchronization protocol (FPS and EDF)
- SRP notations:
  - Task priority ceiling
  - Resource ceiling
  - System ceiling
- Example: FPS (RM)

prio1 < prio2 < prio3



Preliminaries: HSF

- Two-level HSF (*OSPERT'08*)
  - Supports FPPS and EDF in both levels
  - Periodic tasks
  - Subsystems are impl. as periodic servers
  - Impl. made in VxWorks

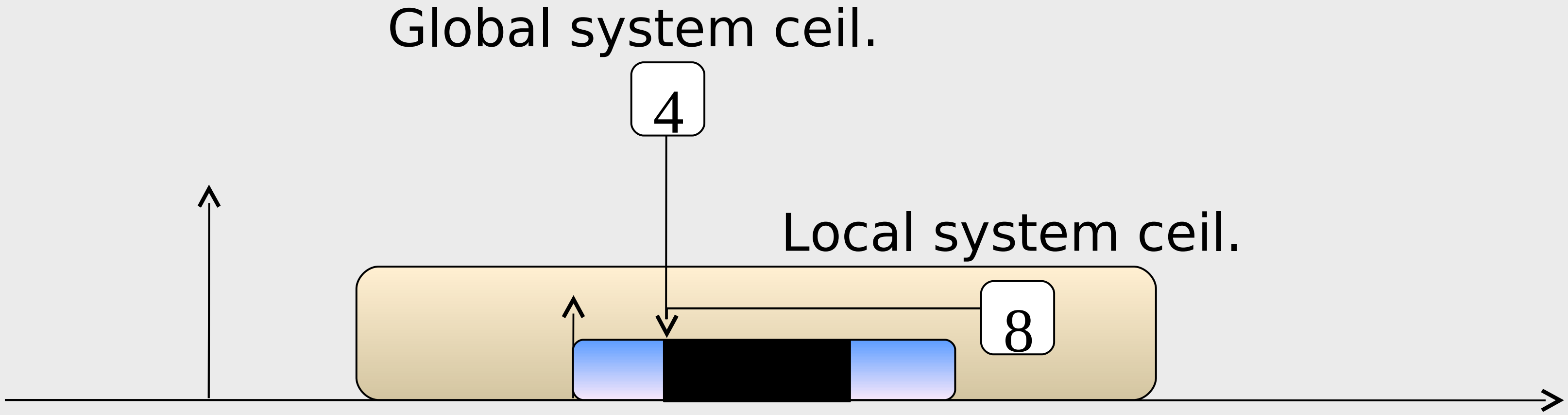


Implementation: SRP

- Data-structures
  - Resource ceiling stack (system ceiling)
  - Blocked task queue (FIFO)
- Added/modified functionality
  - Lock and Unlock
  - Modification to local scheduler

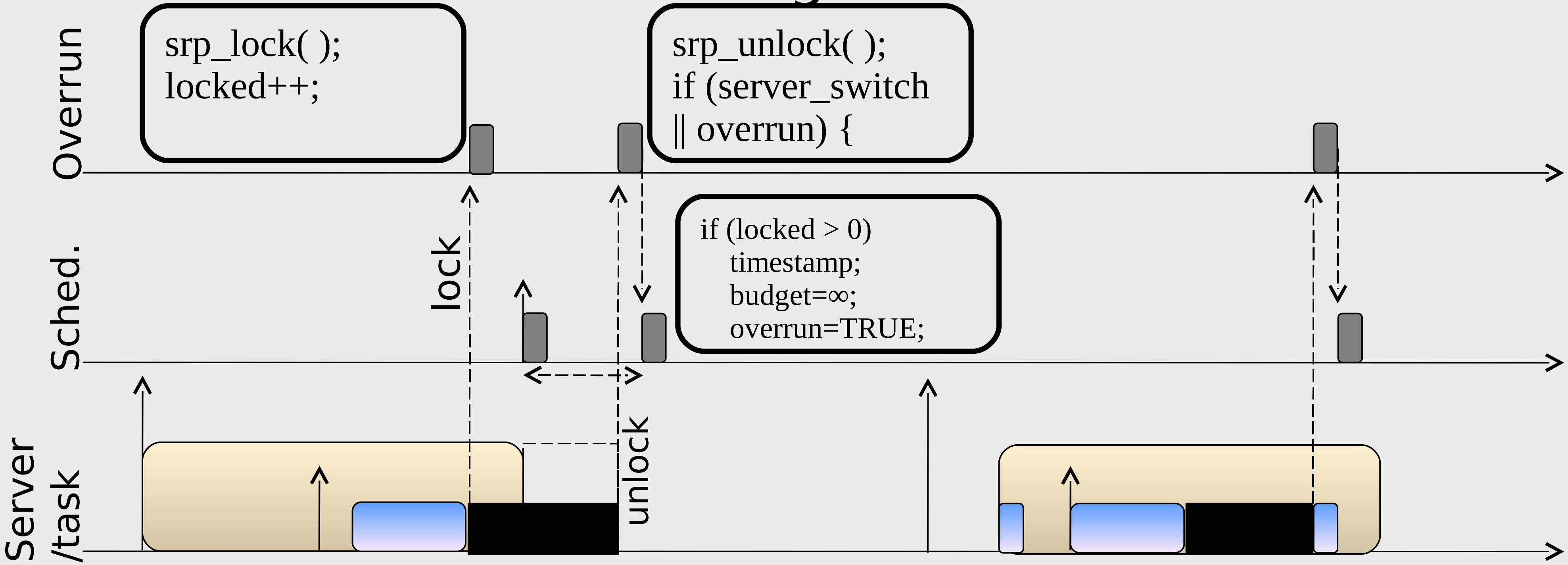
Implementation: Common

- Resources
    - Globally shared (mapped to local)
    - Locally shared
- Global/local system ceiling
- Data-structures
    - Resource ceiling stack (system ceiling)
    - Blocked server queue (FIFO)
- Check sys. ceil. at server release



Implementation: Overrun

- Data-structures
  - Overrun flag
  - Resource counter
- + Low lock overhead
- + Low amount of data-structures
- - Amount of calls to global scheduler



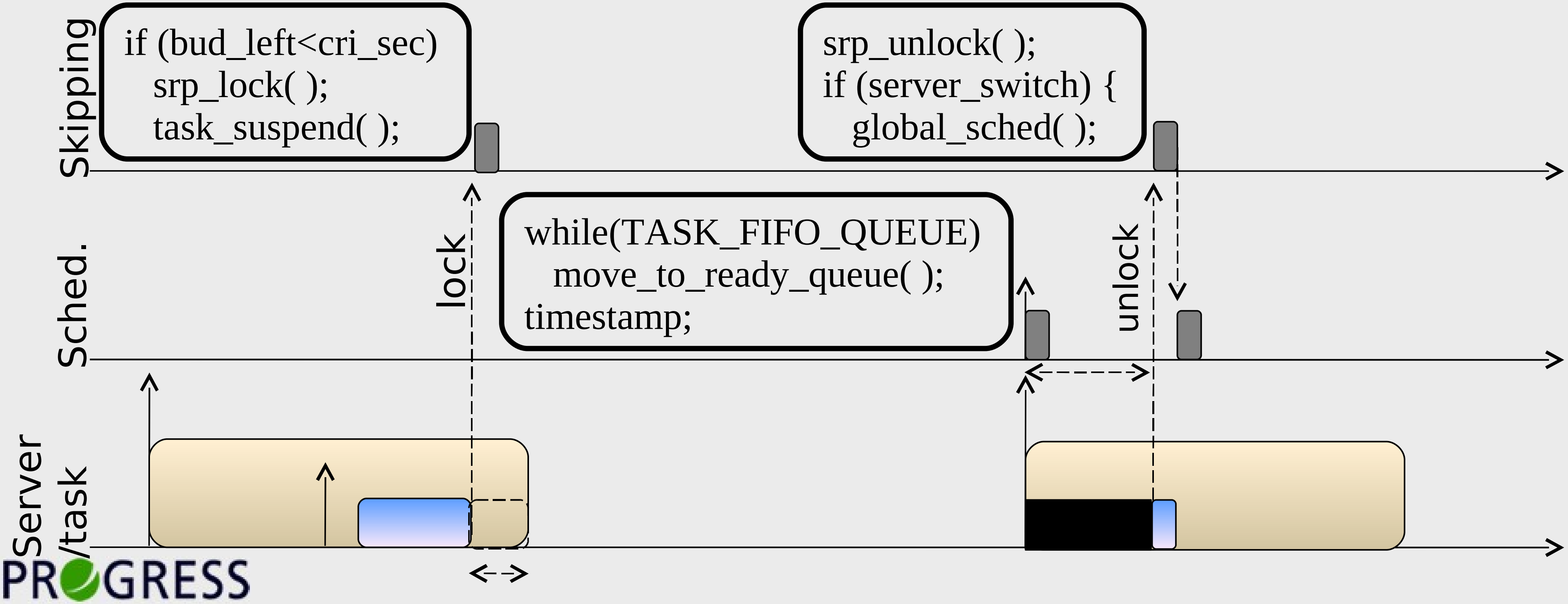
```
srp_lock( );
locked++;
```

```
srp_unlock( );
if (server_switch
|| overrun) {
```

```
if (locked > 0)
    timestamp;
    budget=∞;
    overrun=TRUE;
```

Implementation: Skipping

- Data-structures
  - Resource locking time
  - FIFO queue (for self-blocking)
- + Less calls to global scheduler
- - Overhead in lock and local scheduler
- - Need to save data for critical section length



```
if (bud_left<cri_sec)
   srp_lock( );
   task_suspend( );
```

```
srp_unlock( );
if (server_switch) {
   global_sched( );
```

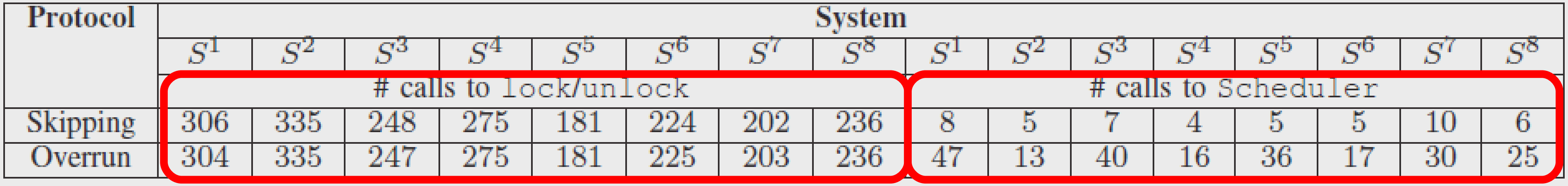
```
while(TASK_FIFO_QUEUE)
   move_to_ready_queue( );
timestamp;
```

Comparison

- Memory complexity:
    - Skipping higher than Overrun
        - Self blocking queue, resource holding time
- Skipping needs modification of local scheduler
- Both modifies the global scheduler
    - Both use server-level SRP
    - Skipping checks self-blocking at release
    - Overrun checks overrun at budget depletion
- Less time deviation for Skipping
    - Overrun calls global scheduler more often
- Overhead
    - Skipping: More overhead in Lock function
    - Overrun: More overhead in Unlock function
    - It points to that Skipping has lower than Overrun
        - (Details in next slide)

Evaluation

- Experimental setup:
  - Hardware:
    - Robotics controller (ABB)
    - Pentium Pro (200 MHz) processor
    - VxWorks 5.2
  - 8 generated systems (S1 – S8)
  - Systems were recorded 600 time units (tu)
  - Task period: 40-100 tu, server period: 5-20 tu
  - Task utilization per system: ~15%



Protocol	System															
	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8	S^1	S^2	S^3	S^4	S^5	S^6	S^7	S^8
	# calls to lock/unlock								# calls to Scheduler							
Skipping	306	335	248	275	181	224	202	236	8	5	7	4	5	5	10	6
Overrun	304	335	247	275	181	225	203	236	47	13	40	16	36	17	30	25

Conclusion

- We have implemented 2 synchronization protocols in VxWorks
  - Overrun and Skipping
- Both protocols are based on our previous work: HSF (OSPERT'08)
- Evaluation results indicate less overhead for Skipping
  - Although memory allocation grows with the nr. of un-nested global resources
- Overrun causes more time deviation
- Skipping needs modification of local scheduler
- Future work:
  - More evaluation (measure entire overhead)
  - Optimizations

The end

Thank you!

Questions?