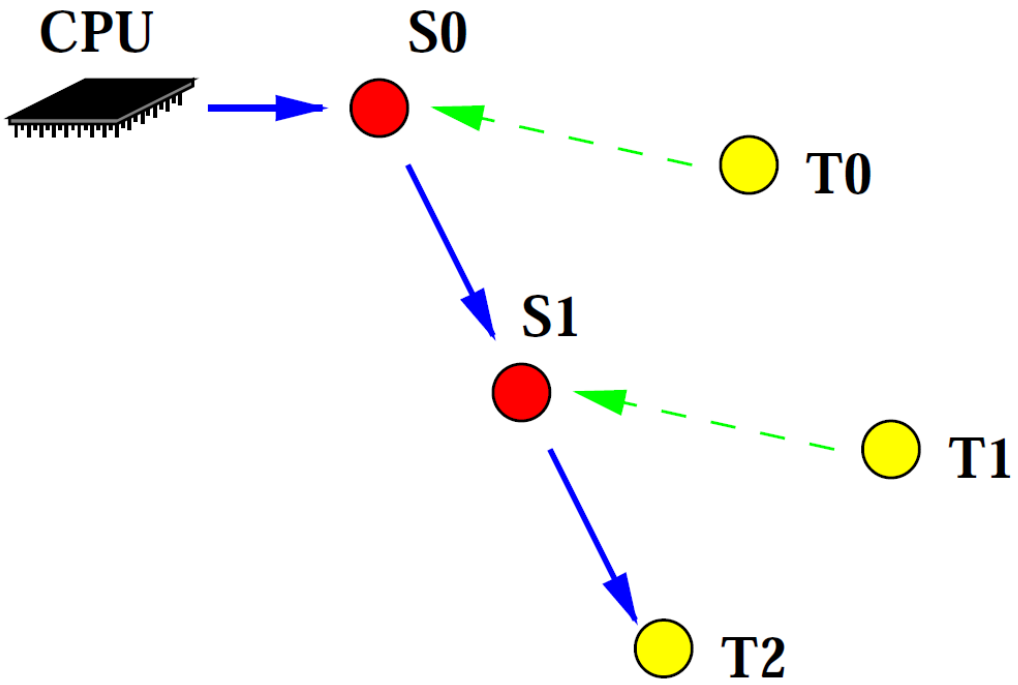
User-Level CPU Inheritance Scheduling

Marcel Kneib – marcel.kneib@posteo.de – Hochschule RheinMain

Jonas Reininger – jonas.reininger@gmail.com – Hochschule RheinMain

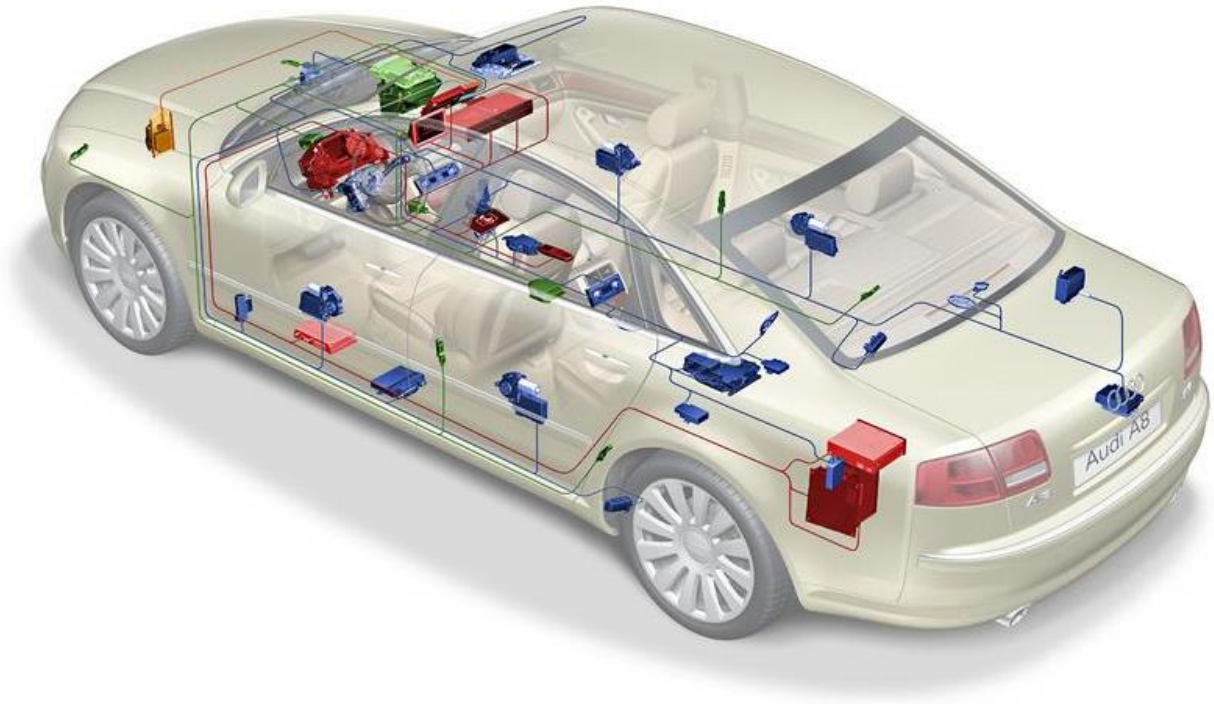
Introduction

- Scheduling in user-level
- Hierarchy of schedulers
- Threads can act as schedulers for other threads

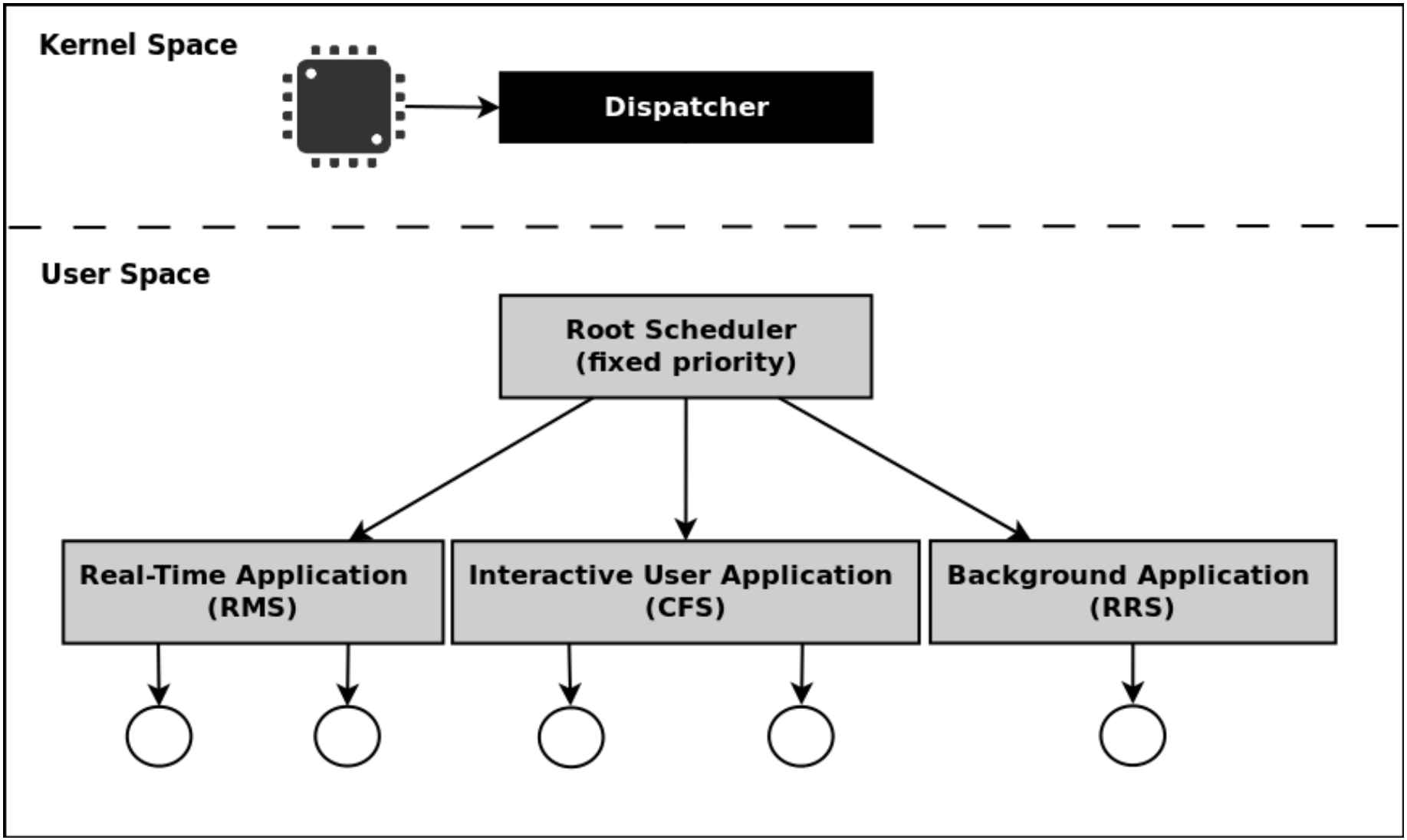


Benefits

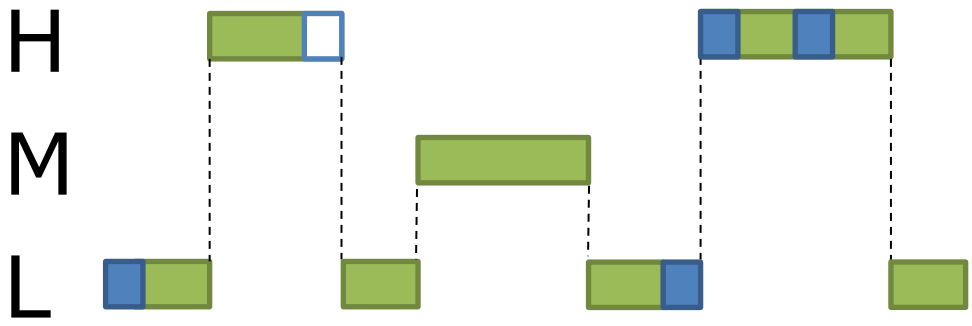
- More flexibility
- Smaller trusted code base
- Meeting different scheduling requirements
- Reduce costs



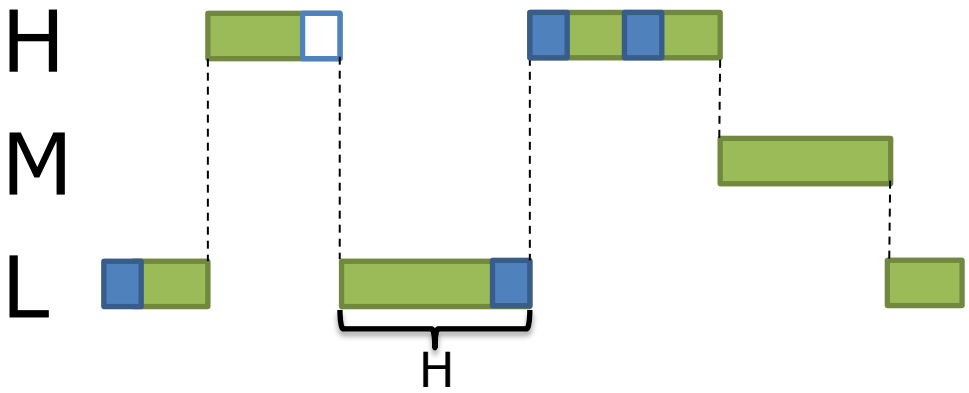
Concept


Kernel Space
Dispatcher
User Space
Root Scheduler
(fixed priority)
Real-Time Application
(RMS)
Interactive User Application
(CFS)
Background Application
(RRS)

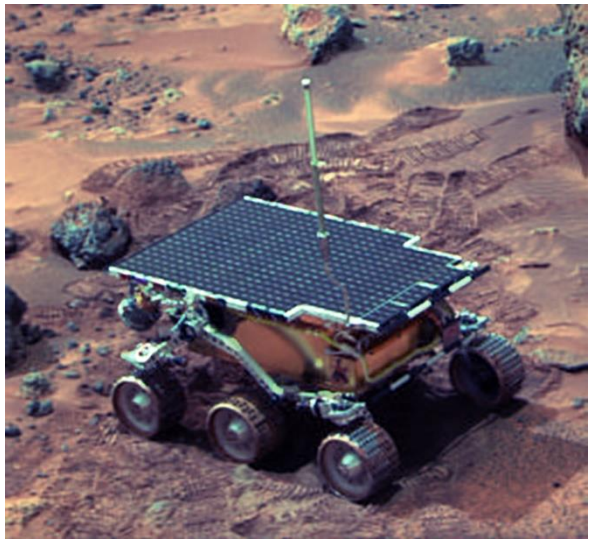
Priority Inheritance



Priority inversion

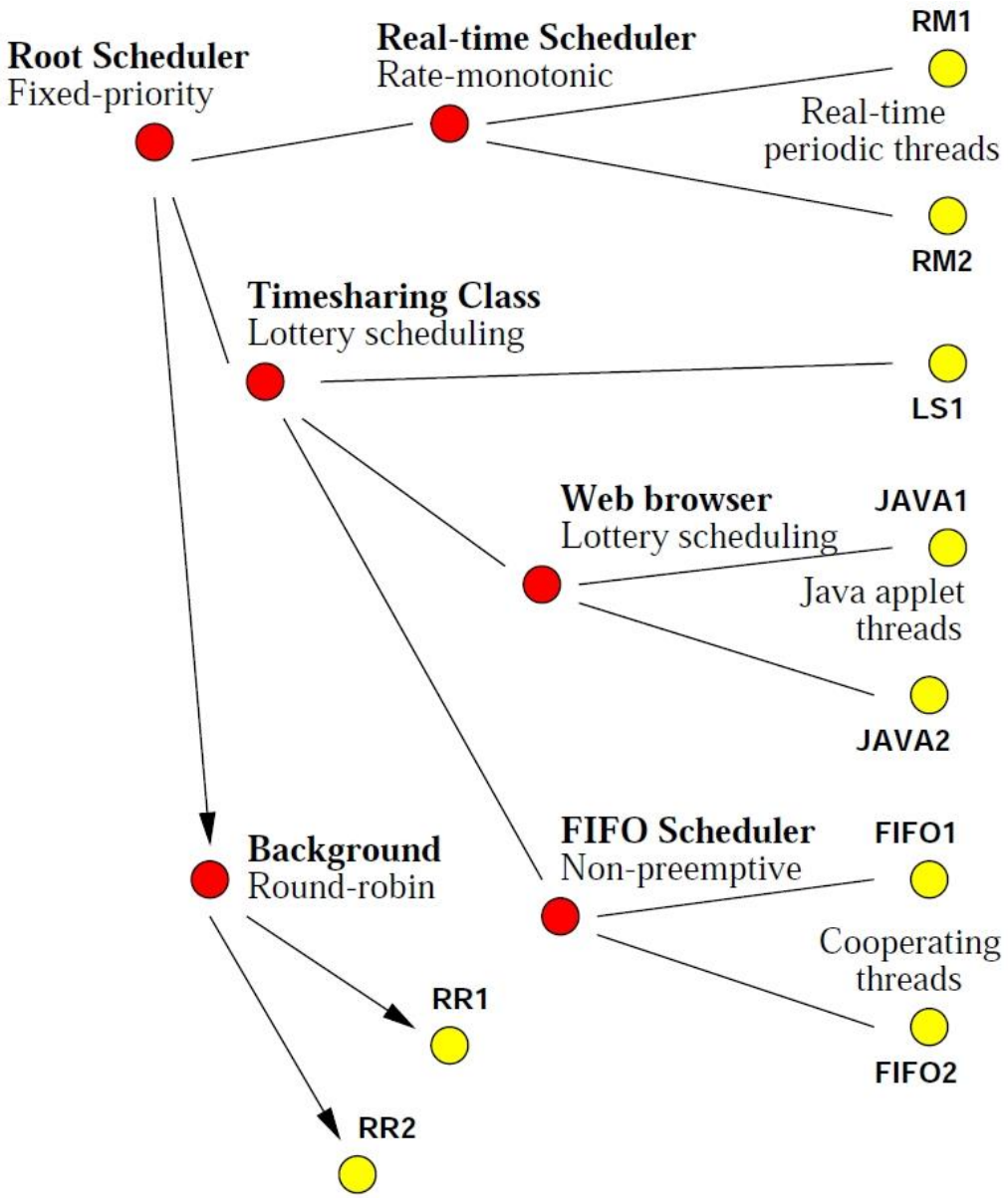


Priority inheritance



Mars Pathfinder, 1997

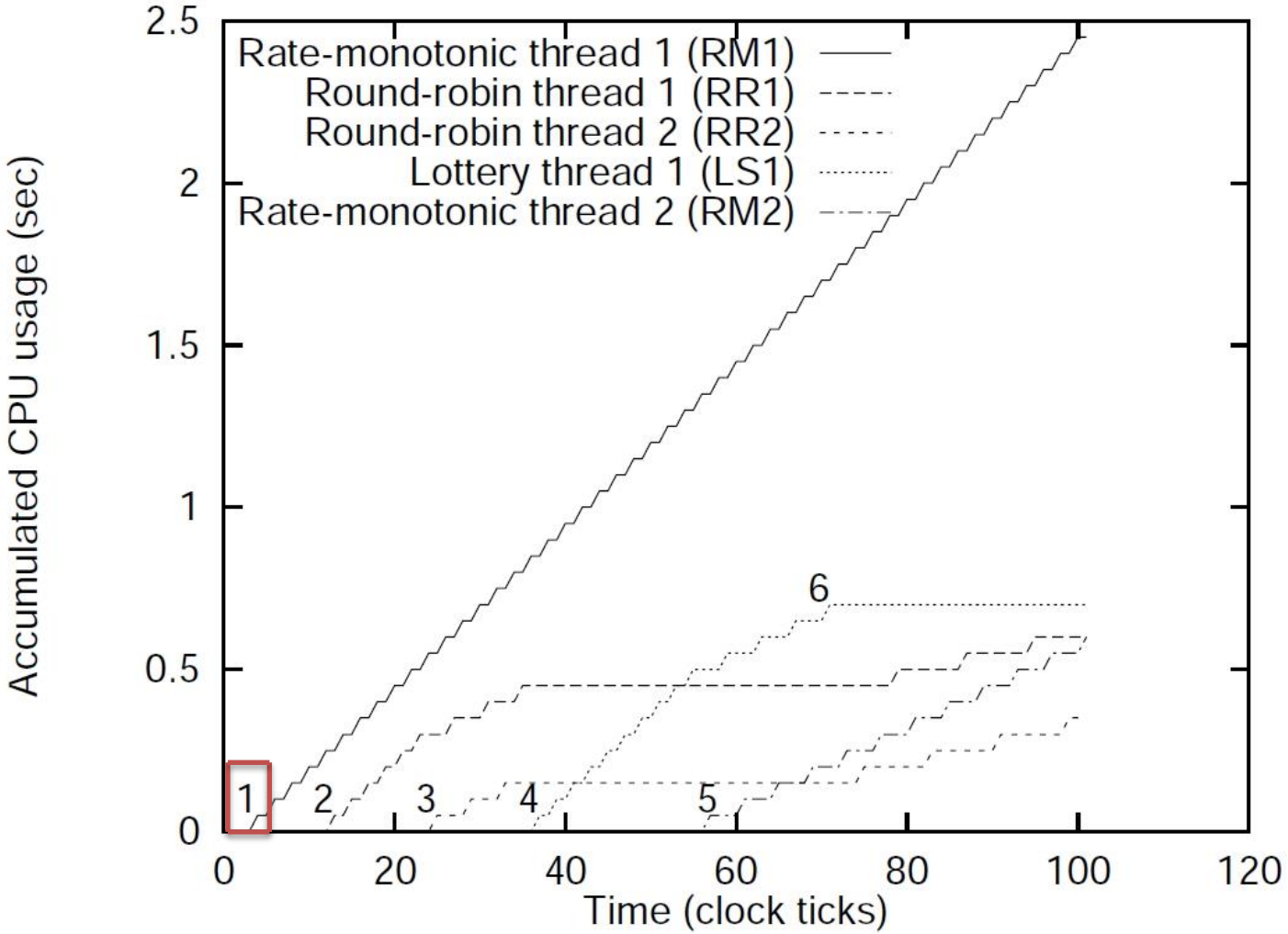
Test Environment



Root Scheduler:

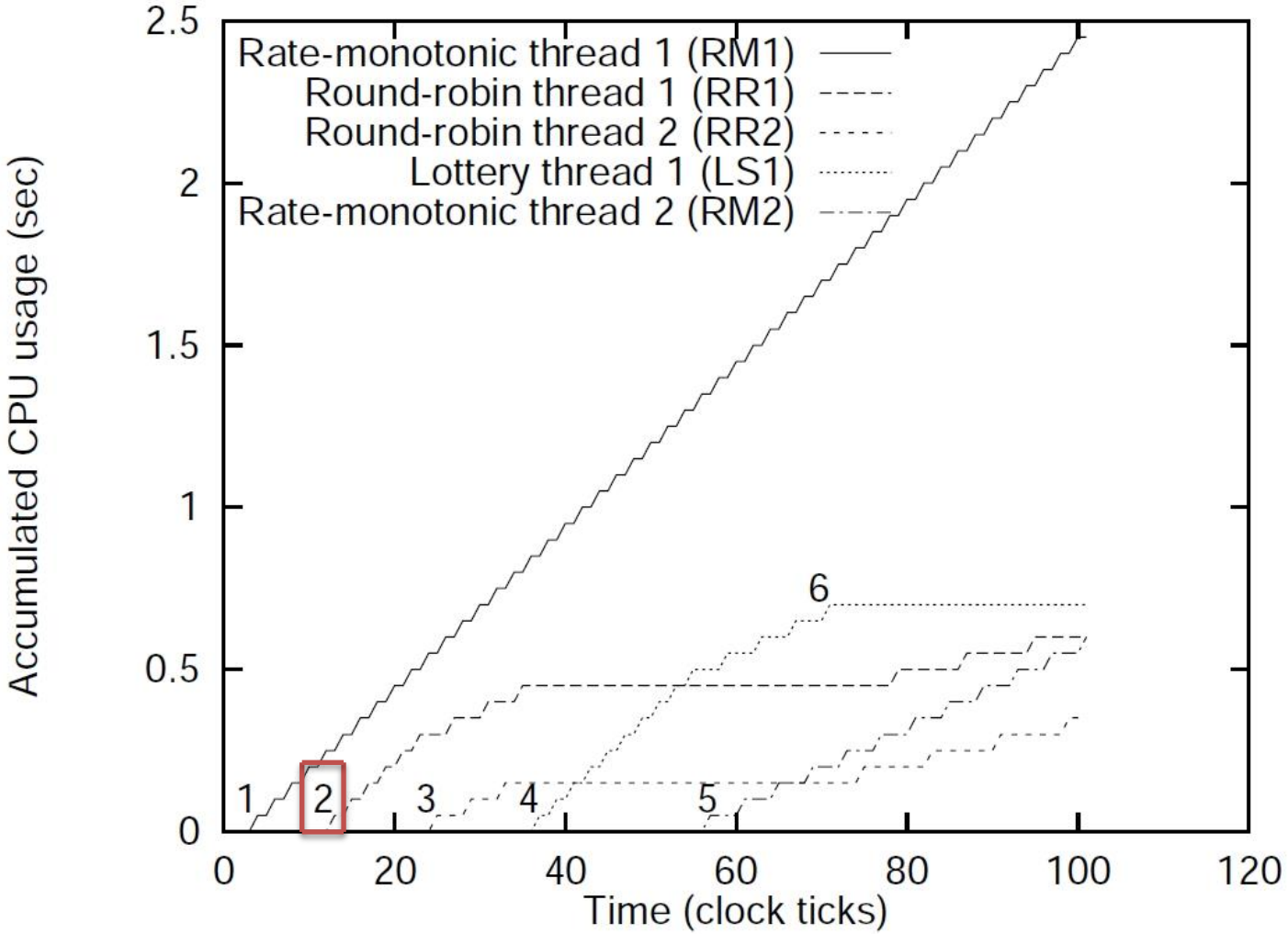
- H: Real-time
  - RM1: 50% CPU time
  - RM2: 25% CPU time
- M: Timesharing
  - LS1: available CPU time
- L: Background
  - RR1: available CPU time
  - RR2: available CPU time

Test



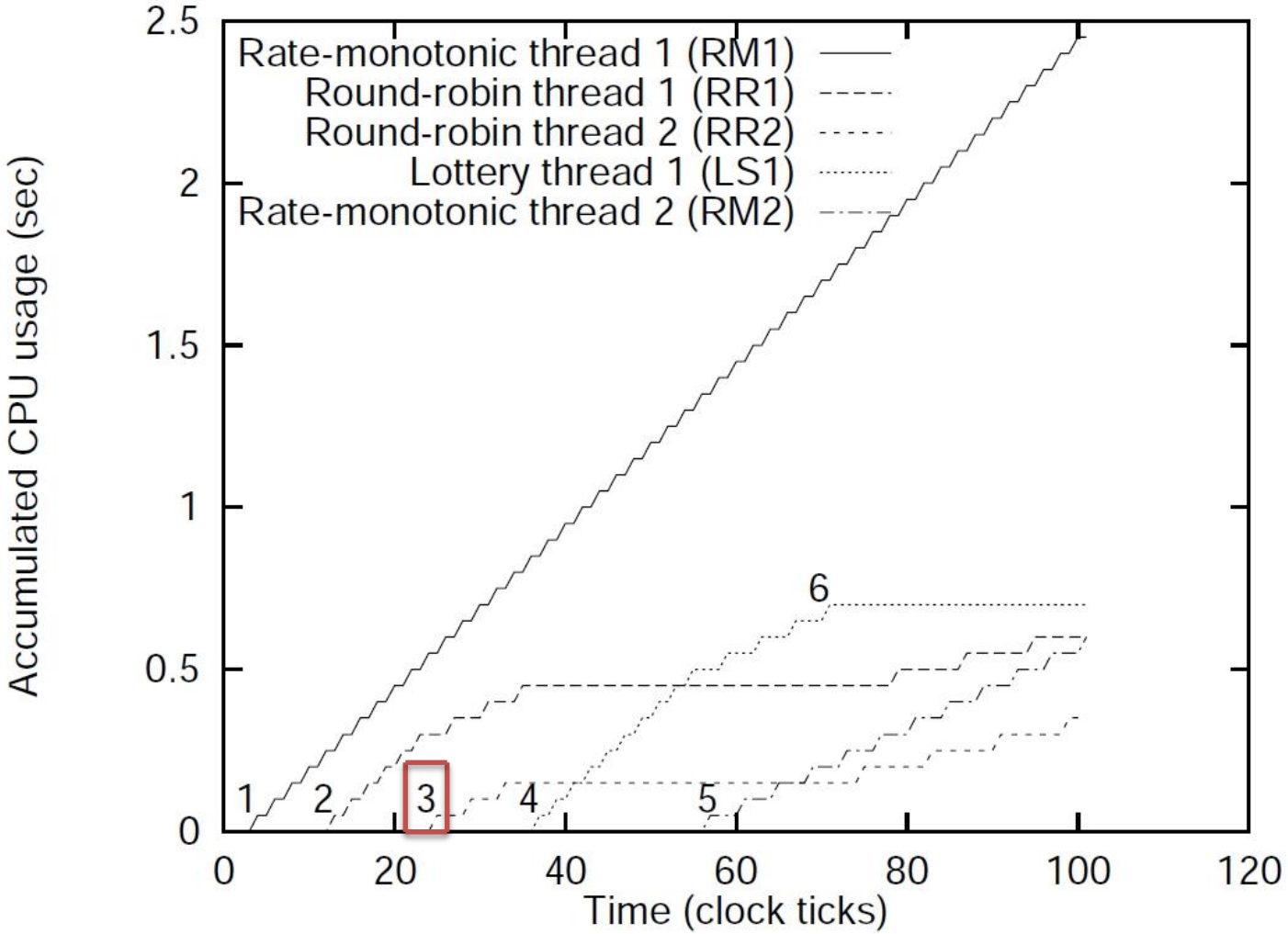
1. RM1 starts consuming 50% CPU time periodically

Test



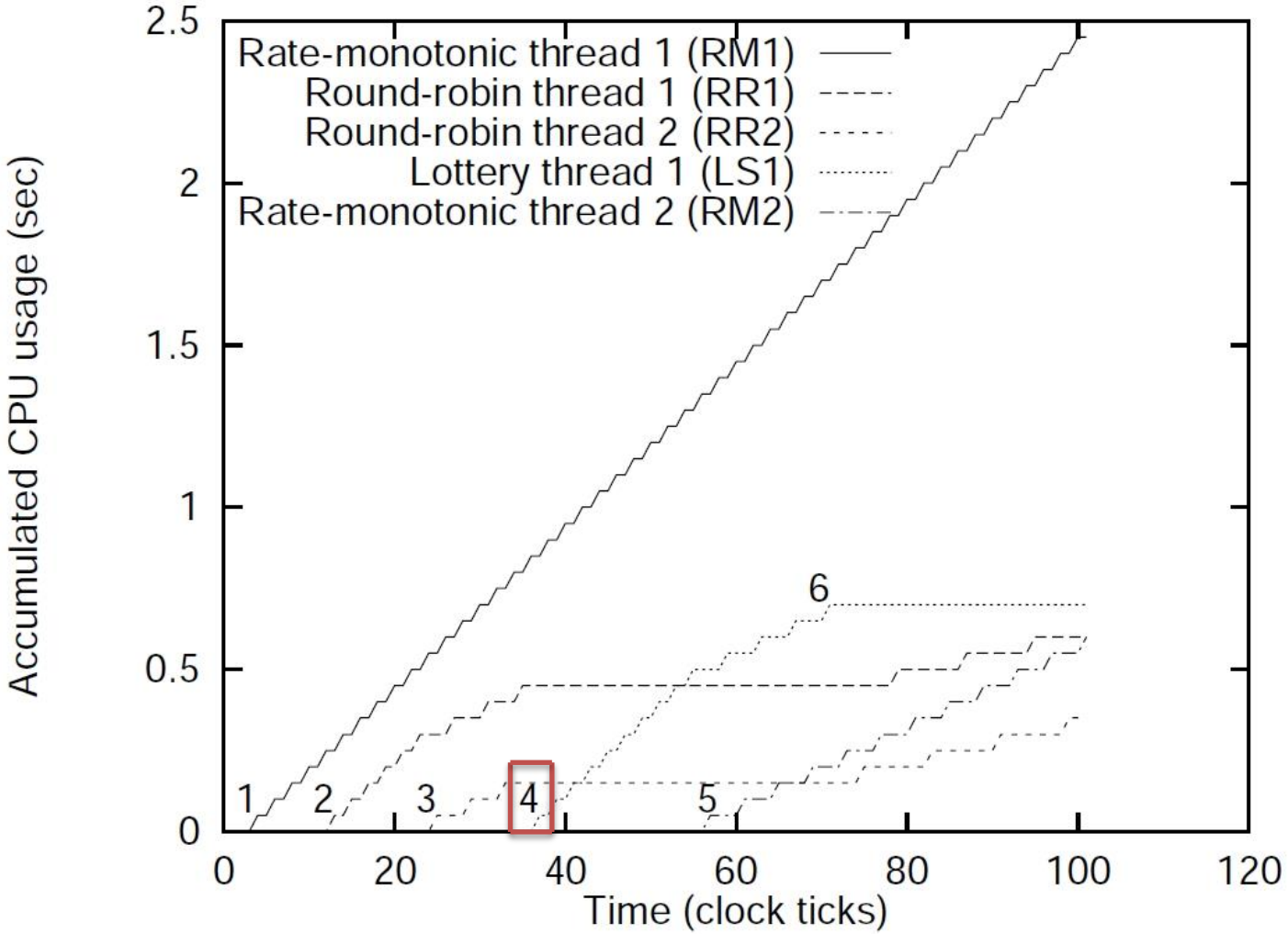
2. RR1 begins to consume alle the CPU time it can get

Test



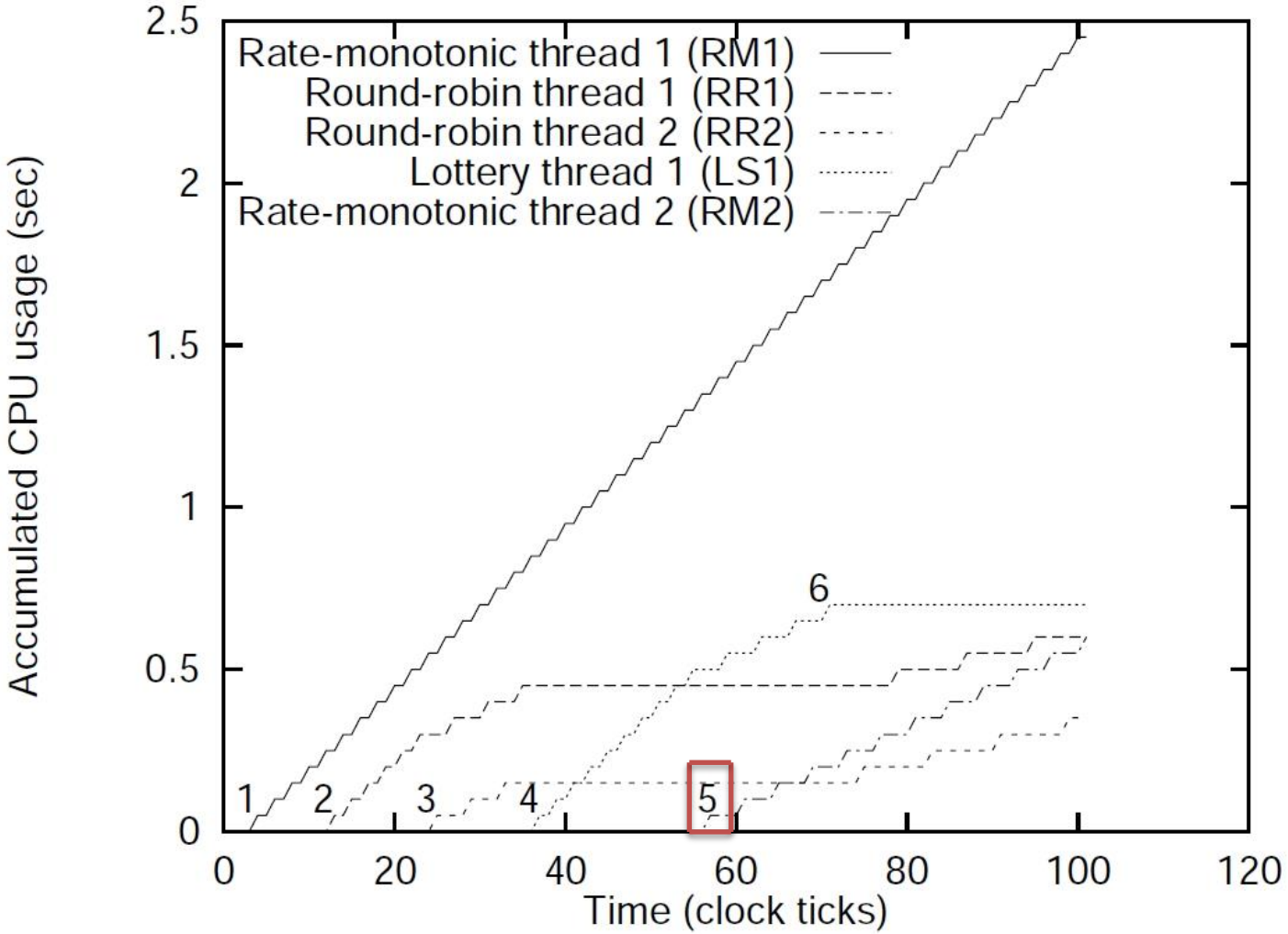
3. RR2 begins with the same priority as RR1

Test



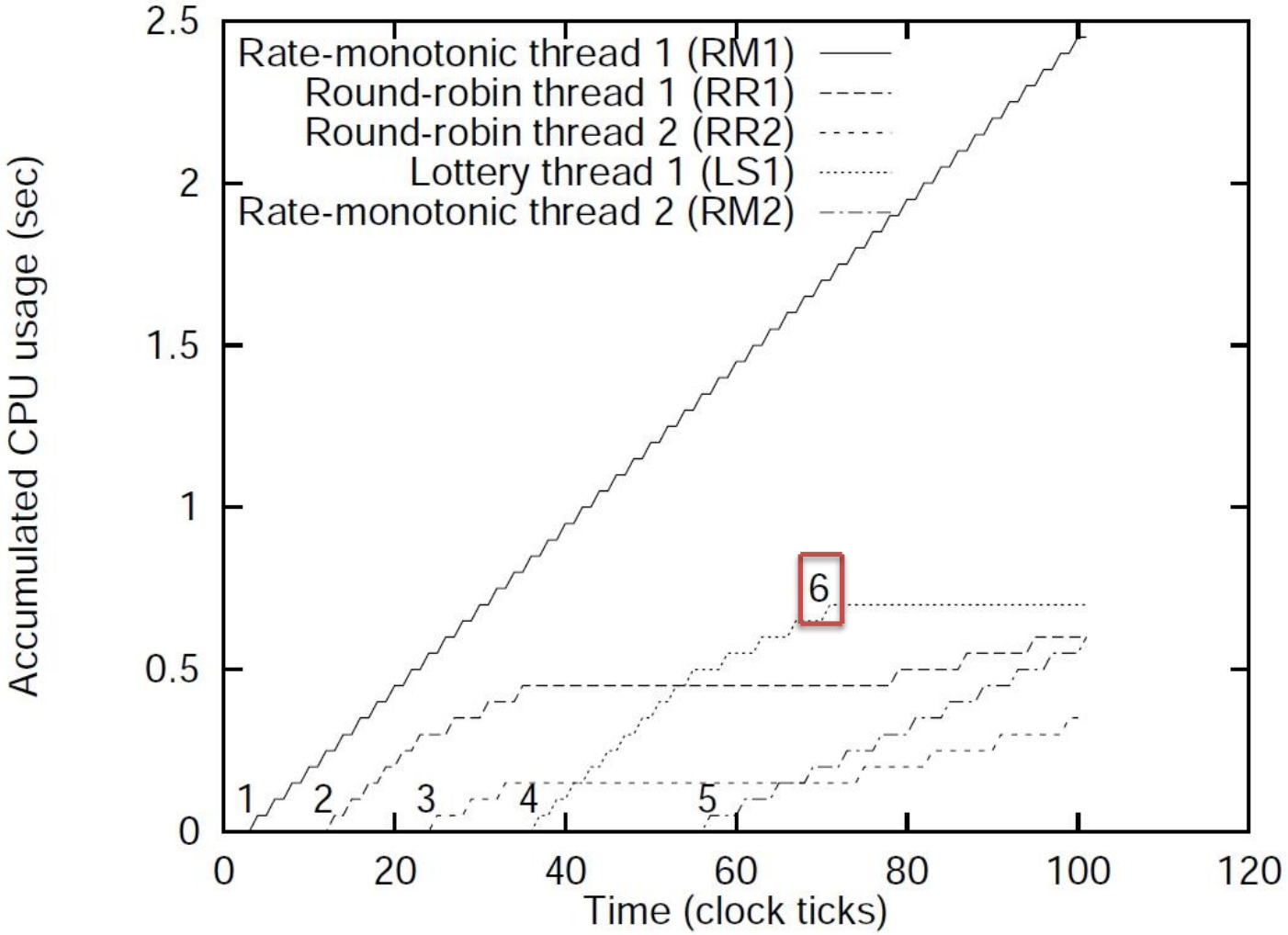
4. LS1 begins to steal all available CPU time

Test



5. RM2 starts consuming 25% of the CPU time periodically

Test

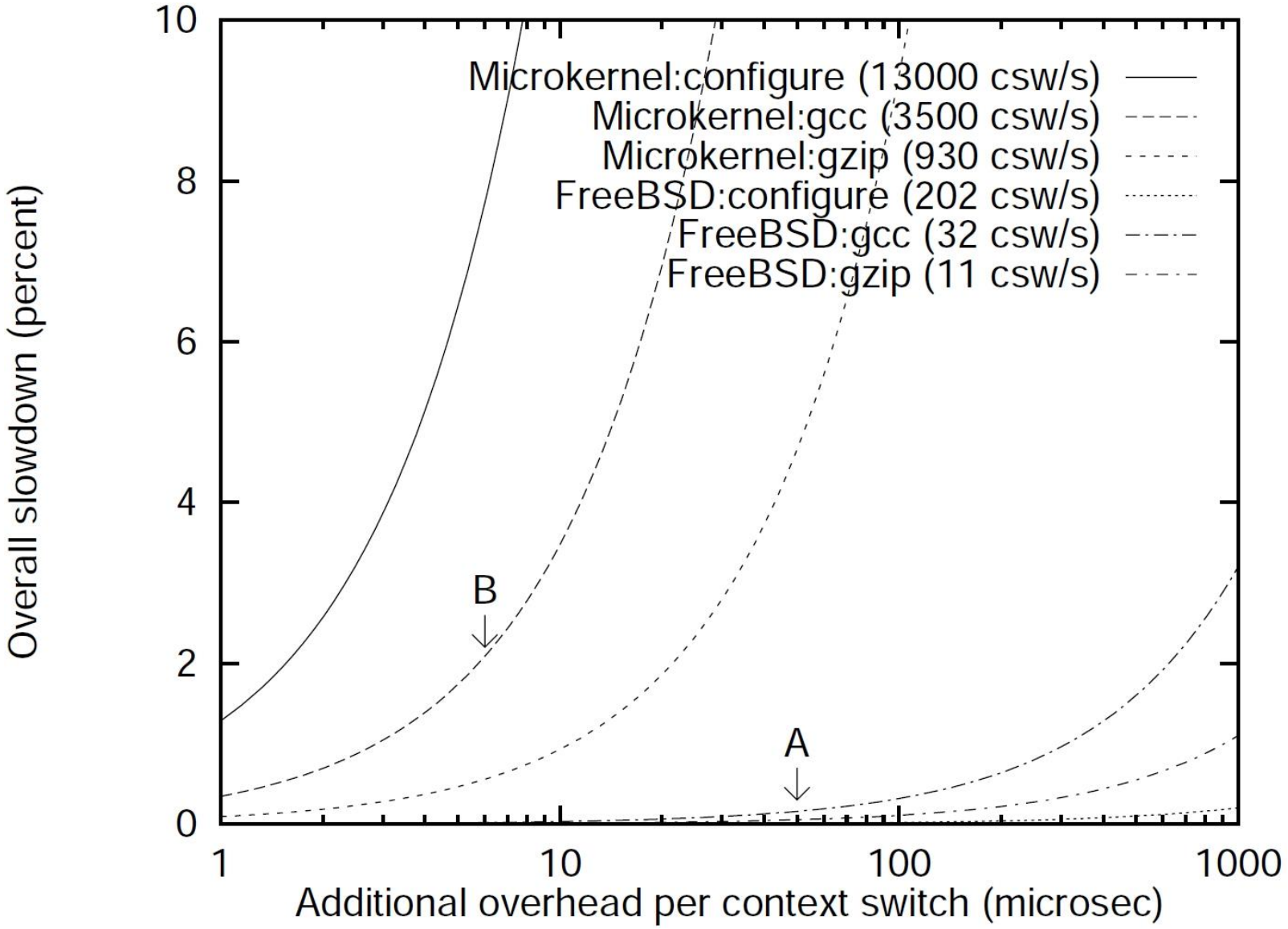


6. LS1 finishes

Problems

- Dispatcher costs
  - iteration through list or tree
- Context switch costs
  - approx. one additional context switch
- Threads running in one single address space
  - no memory protection
  - not isolated

Problems


Microkernel:configure (13000 csw/s)
Microkernel:gcc (3500 csw/s)
Microkernel:gzip (930 csw/s)
FreeBSD:configure (202 csw/s)
FreeBSD:gcc (32 csw/s)
FreeBSD:gzip (11 csw/s)
B
A
Overall slowdown (percent)
0
2
4
6
8
10
1
10
100
1000
Additional overhead per context switch (microsec)

Conclusion

- More flexibility with negligible overhead (in Test Environment)
- If per-context-switch-costs are low, CPU Inheritance Scheduling makes sense even in microkernels
- Scheduling hierarchy depth should be limited
- Sample implementation is based on user-thread library

Questions?

Marcel.Kneib@posteo.de

Jonas.Reininger@gmail.com