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

RTAS'13
April 10, 2013

Max
Planck
Institute
for
Software Systems

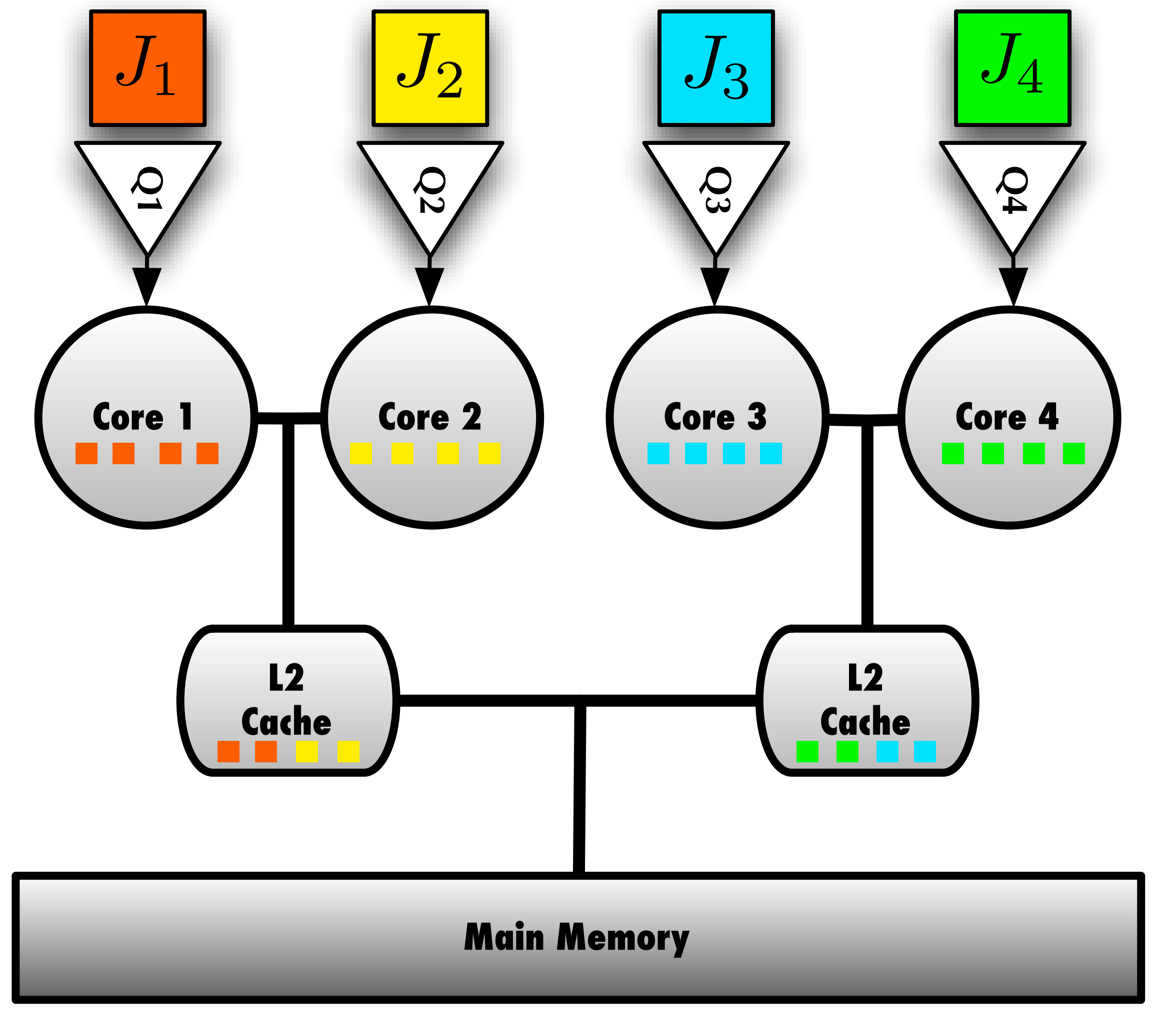
Björn B. Brandenburg
bbb@mpi-sws.org

Semaphores + P-FP Scheduling

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

Binary Semaphores in POSIX

```
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?

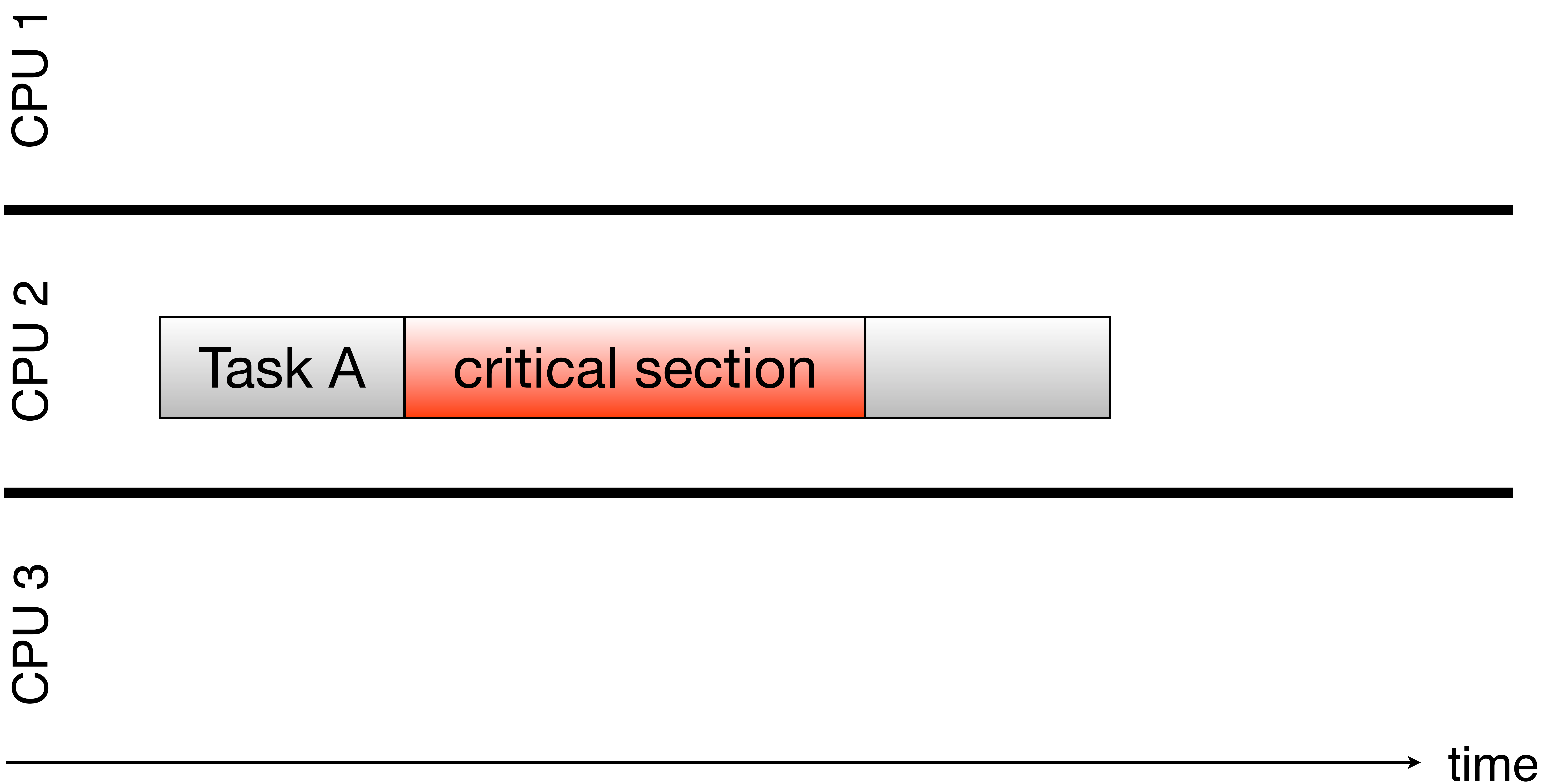
*How to **order** conflicting critical sections?*

→ **FIFO** vs. **Priority** Queues

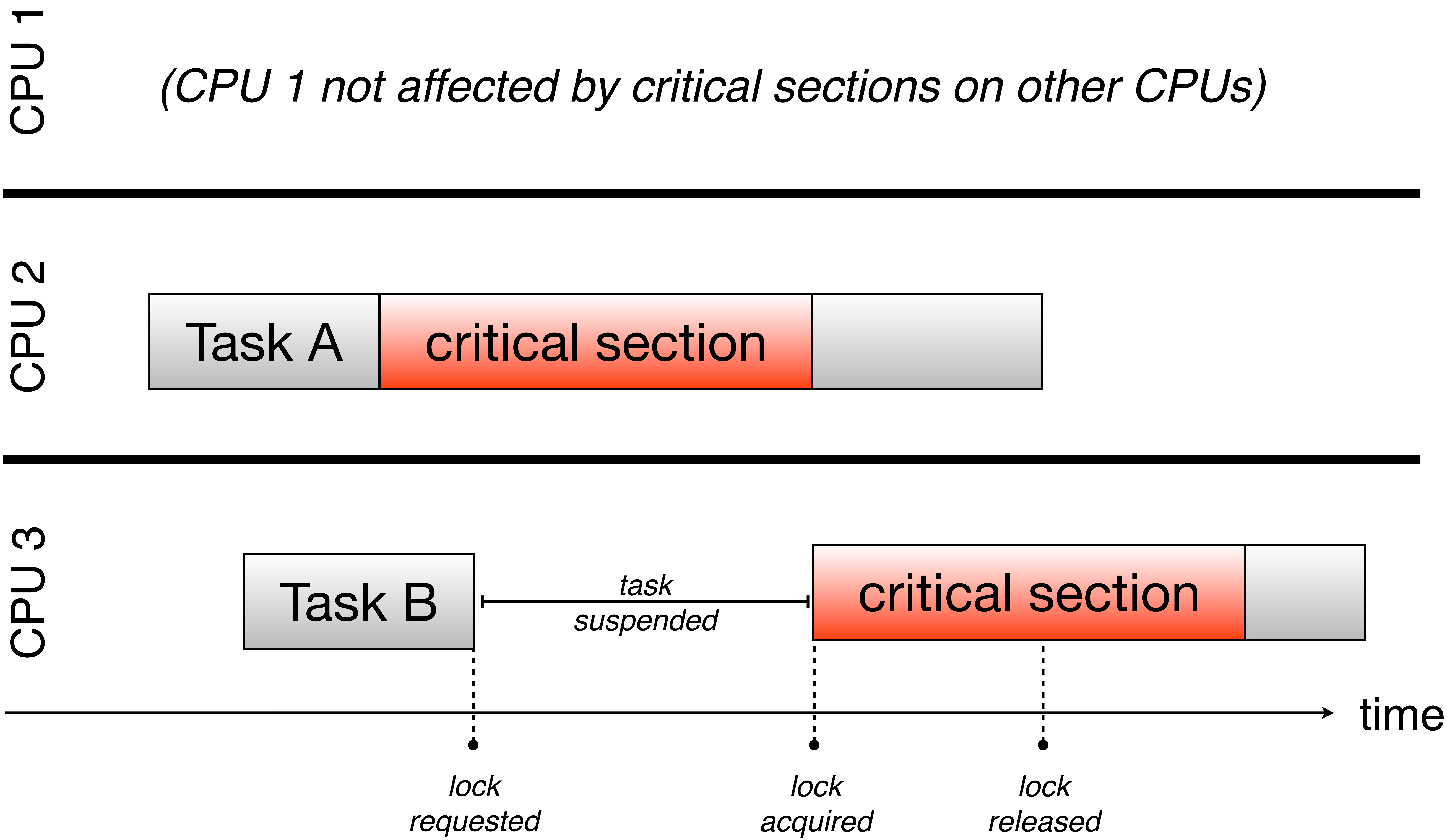
***Where** to execute critical sections?*

→ **Shared-Memory** vs. **Distributed** Locking Protocols

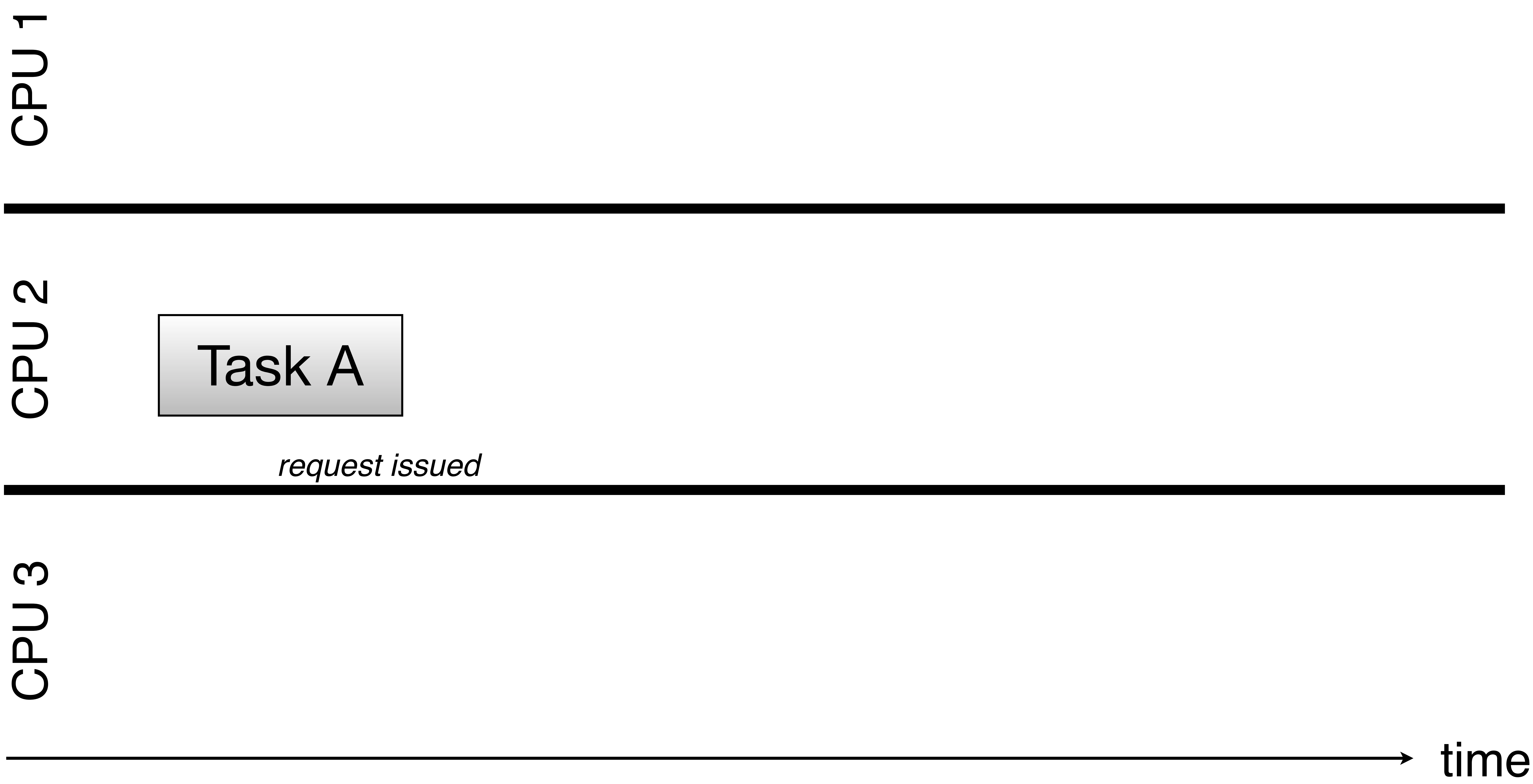
Example: Shared-Memory Protocol



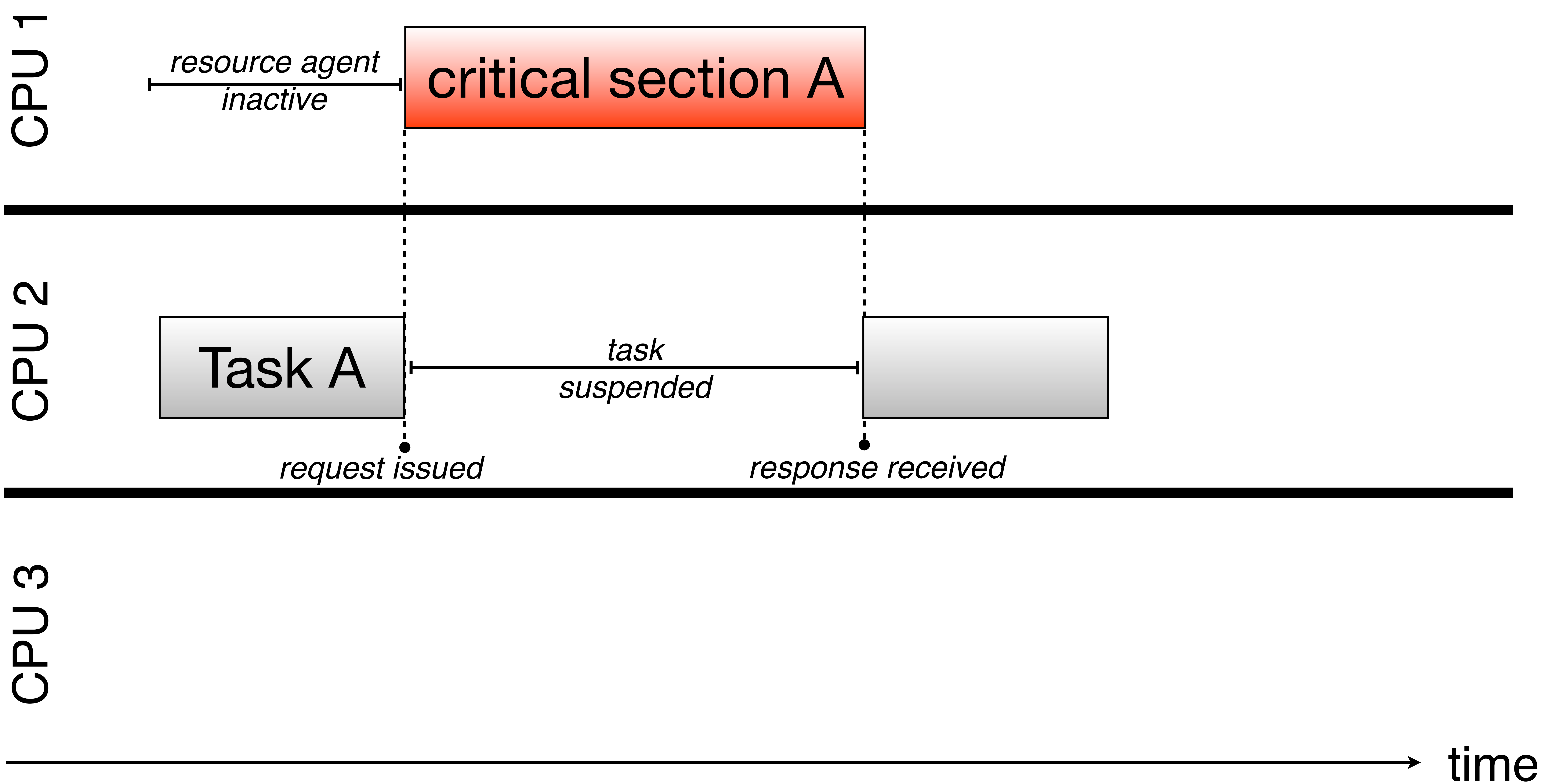
Example: Shared-Memory Protocol



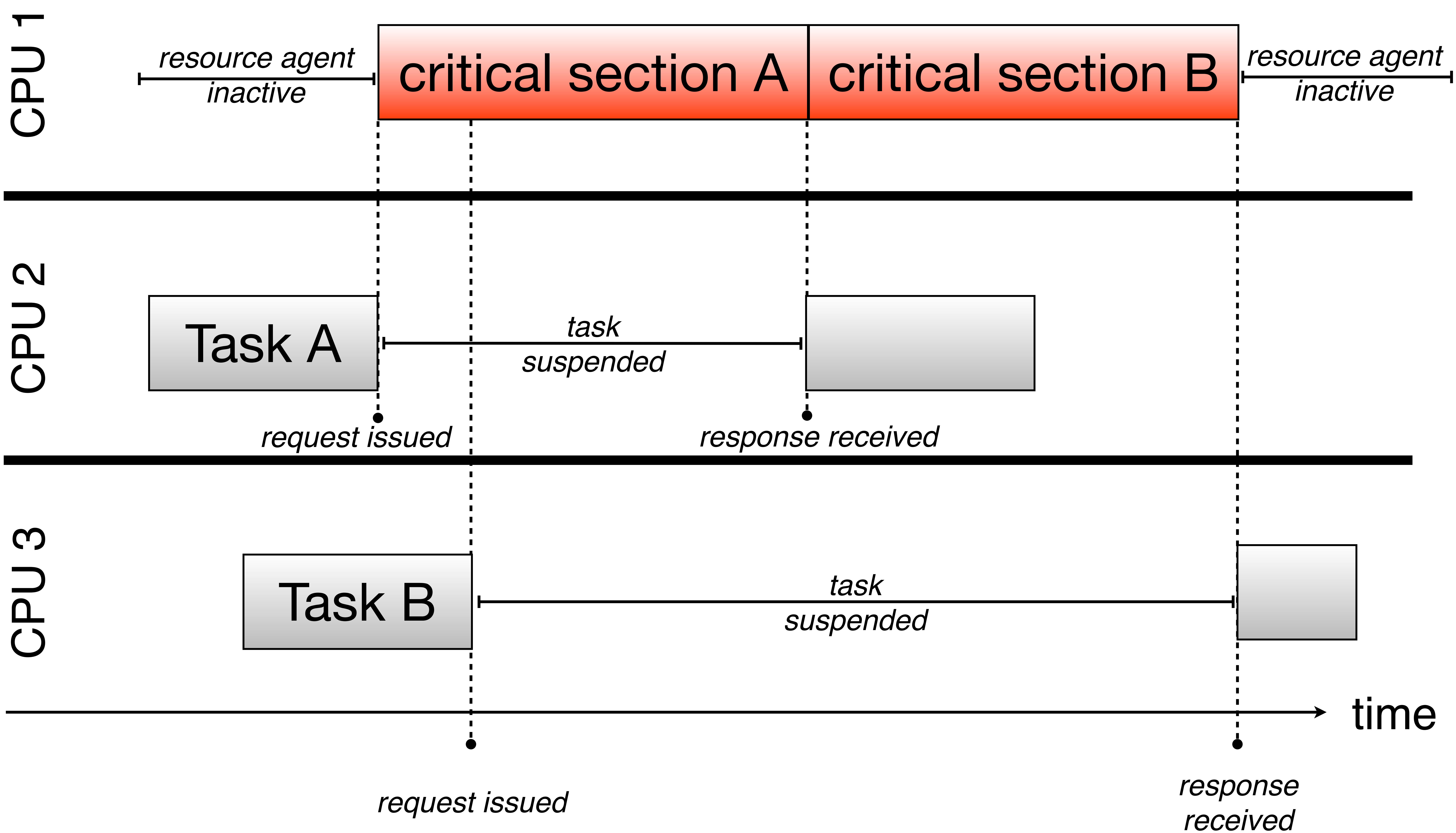
Example: Distributed Protocol



Example: Distributed Protocol



Example: Distributed Protocol



Semaphore Protocol Choices

*How to **order** conflicting critical sections?*

***Where** to execute critical sections?*

Wait Queue	**priority** queue	**FIFO** queue	**priority** queue	**FIFO** queue
Protocol Type	**shared-memory**	**shared-memory**	**distributed**	**distributed**

Multiprocessor Priority Ceiling Protocol

col Choices

	MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)			
Wait Queue	priority queue	FIFO queue	priority queue	FIFO queue
Protocol Type	shared-memory	shared-memory	distributed	distributed

FIFO Multiprocessor Locking Protocol

	MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)	**FMLP+** (Brandenburg, 2011)		
Wait Queue	priority queue	FIFO queue	priority queue	FIFO queue
Protocol Type	shared-memory	shared-memory	distributed	distributed

Semaphore Protocol Choices

	MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)	**FMLP+** (Brandenburg, 2011)		
Wait Queue	**priority** queue	**FIFO** queue	**priority** queue	**FIFO** queue
Protocol Type	**shared-memory**	**shared-memory**	**distributed**	**distributed**
Asymptotically optimal? *(w.r.t. maximum blocking)*	**NO**	**YES**		

Semaphore

Distributed Priority Ceiling Protocol

	MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)	**FMLP+** (Brandenburg, 2011)	**DPCP** (Rajkumar et al., 1988)	
Wait Queue	**priority** queue	**FIFO** queue	**priority** queue	**FIFO** queue
Protocol Type	**shared-memory**	**shared-memory**	**distributed**	**distributed**
Asymptotically optimal? *(w.r.t. maximum blocking)*	**NO**	**YES**	**NO**	

Semapho

Distributed FIFO Locking Protocol

	MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)	**FMLP+** (Brandenburg, 2011)	**DPCP** (Rajkumar et al., 1988)	**DFLP** (Brandenburg, 2012)
Wait Queue	**priority** queue	**FIFO** queue	**priority** queue	**FIFO** queue
Protocol Type	**shared-memory**	**shared-memory**	**distributed**	**distributed**
Asymptotically optimal? *(w.r.t. maximum blocking)*	**NO**	**YES**	**NO**	**YES**

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

But: constants matter…

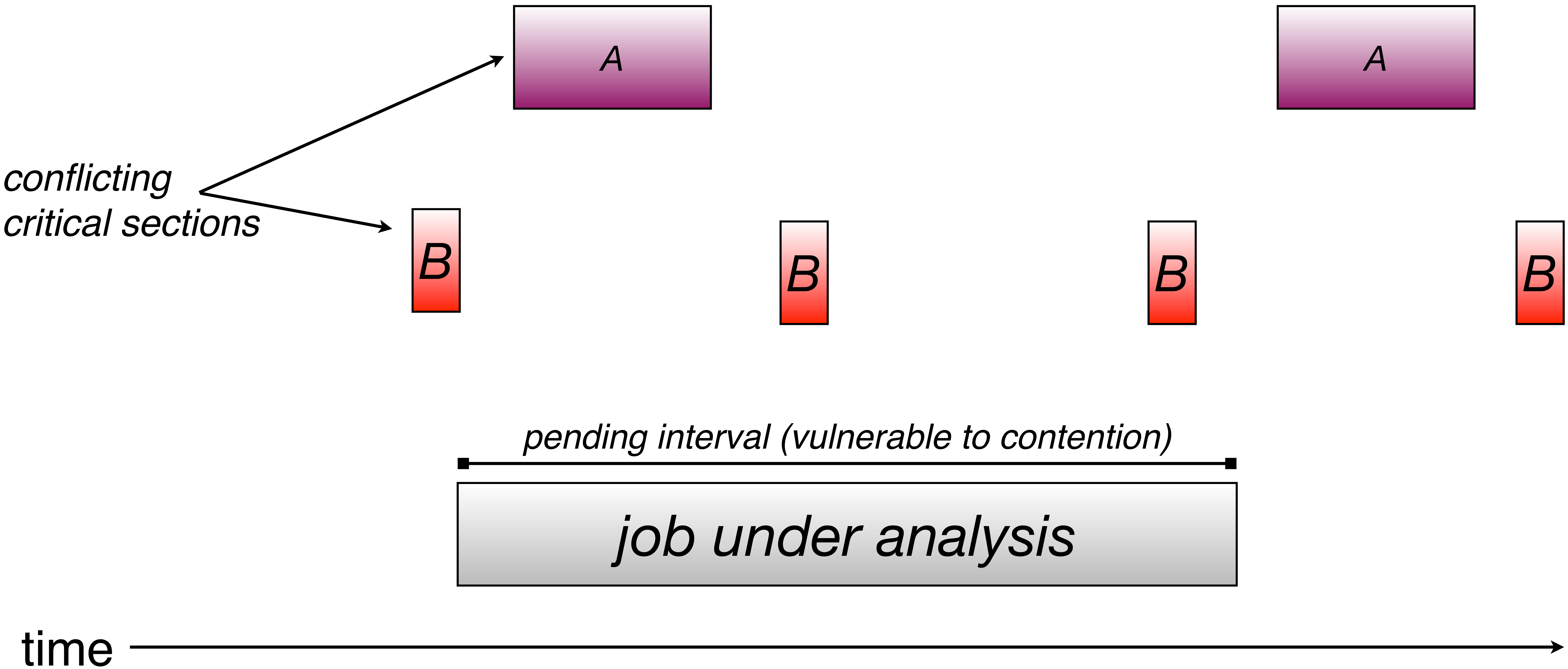
	MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)	**FMLP+** (Brandenburg, 2011)	**DPCP** (Rajkumar et al., 1988)	**DFLP** (Brandenburg, 2012)
Wait Queue	**priority** queue	**FIFO** queue	**priority** queue	**FIFO** queue
Protocol Type	**shared-memory**	**shared-memory**	**distributed**	**distributed**
Asymptotically optimal? *(w.r.t. maximum blocking)*	**NO**	**YES**	**NO**	**YES**

Part 1

Improved Analysis

Non-Asymptotic, Fine-Grained Analysis

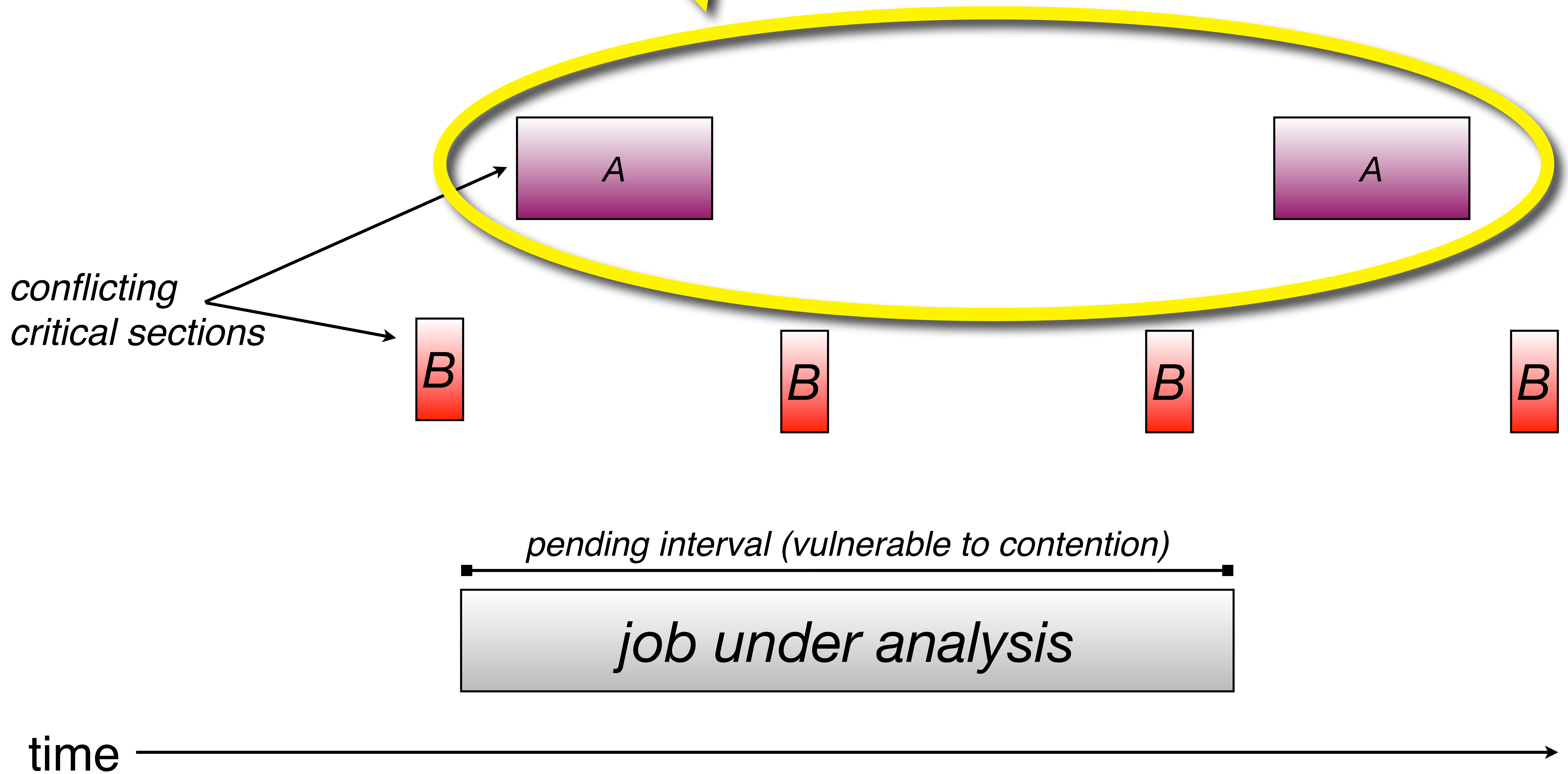
Derive tightest possible bound reflecting all constant factors.



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

Non-Asymptotic Fine-Grained Analysis

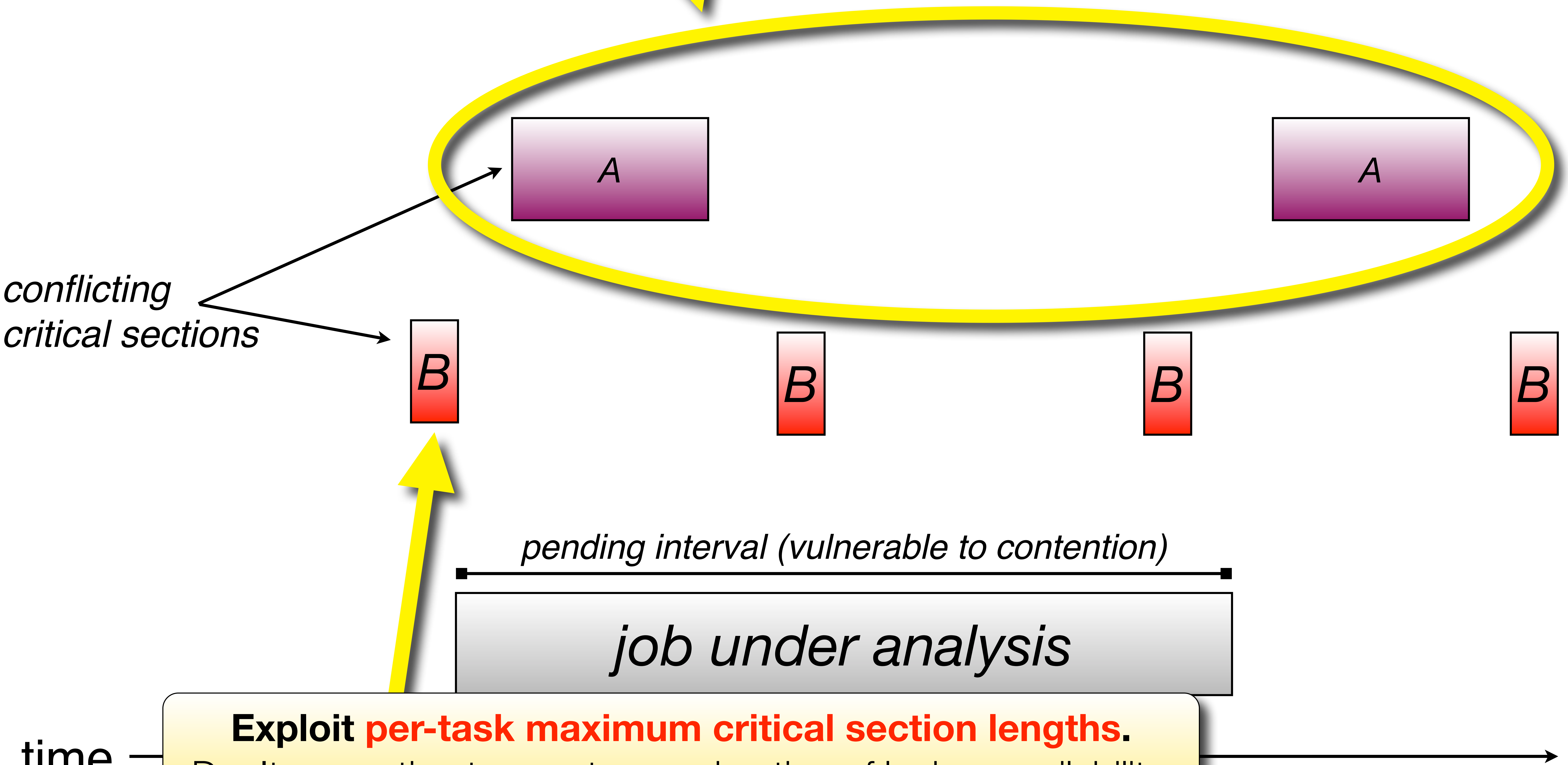
Derive tightest possible bound reflecting all constant factors.



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

Non-Asymptotic Fine-Grained Analysis

Derive tightest possible bound reflecting all constant factors.

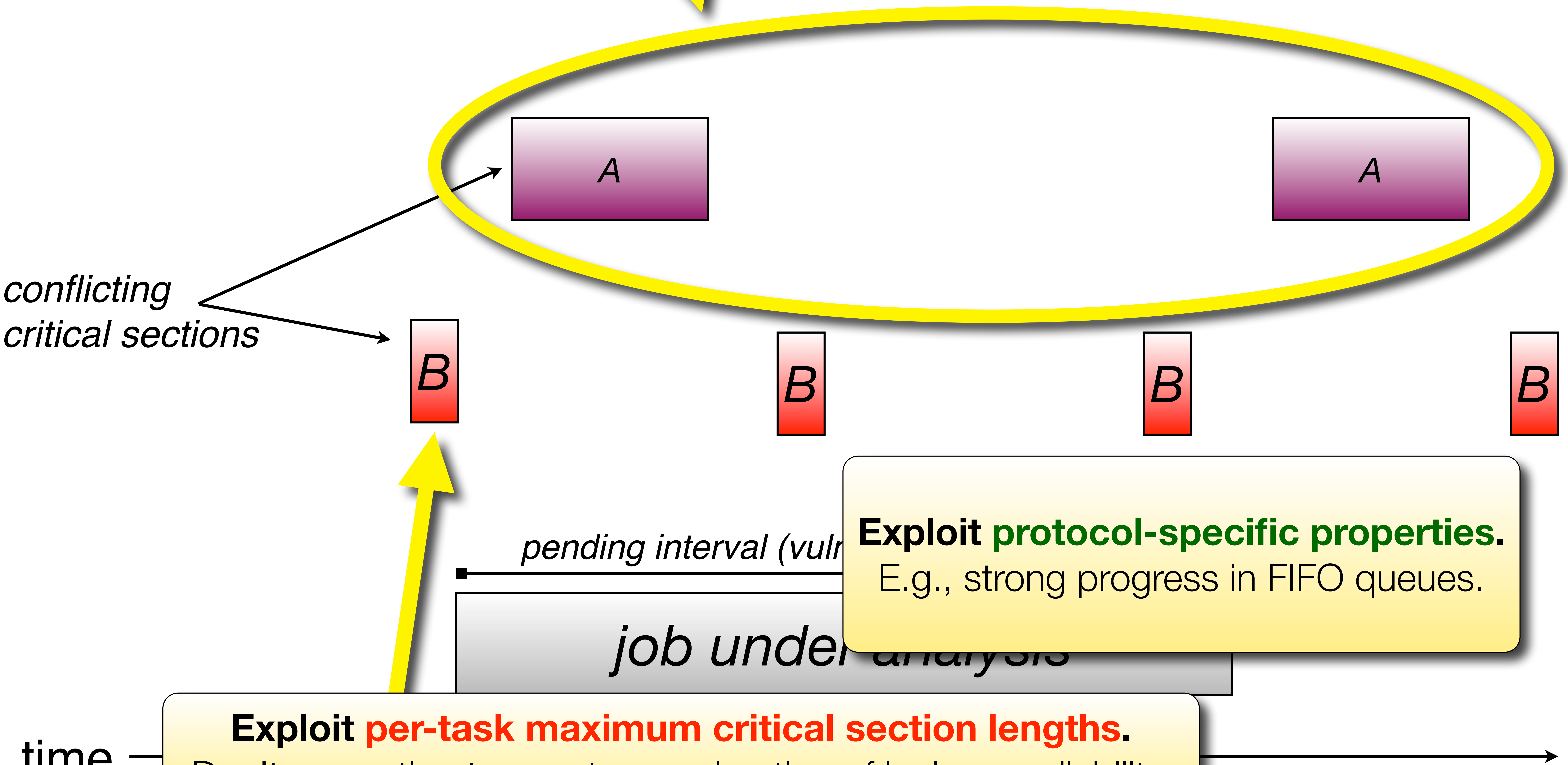


Exploit per-task maximum critical section lengths.
Don’t overestimate worst-case duration of lock unavailability.

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

Non-Asymptotic Fine-Grained Analysis

Derive tightest possible bound reflecting all constant factors.



Exploit protocol-specific properties.
E.g., strong progress in FIFO queues.

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

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)

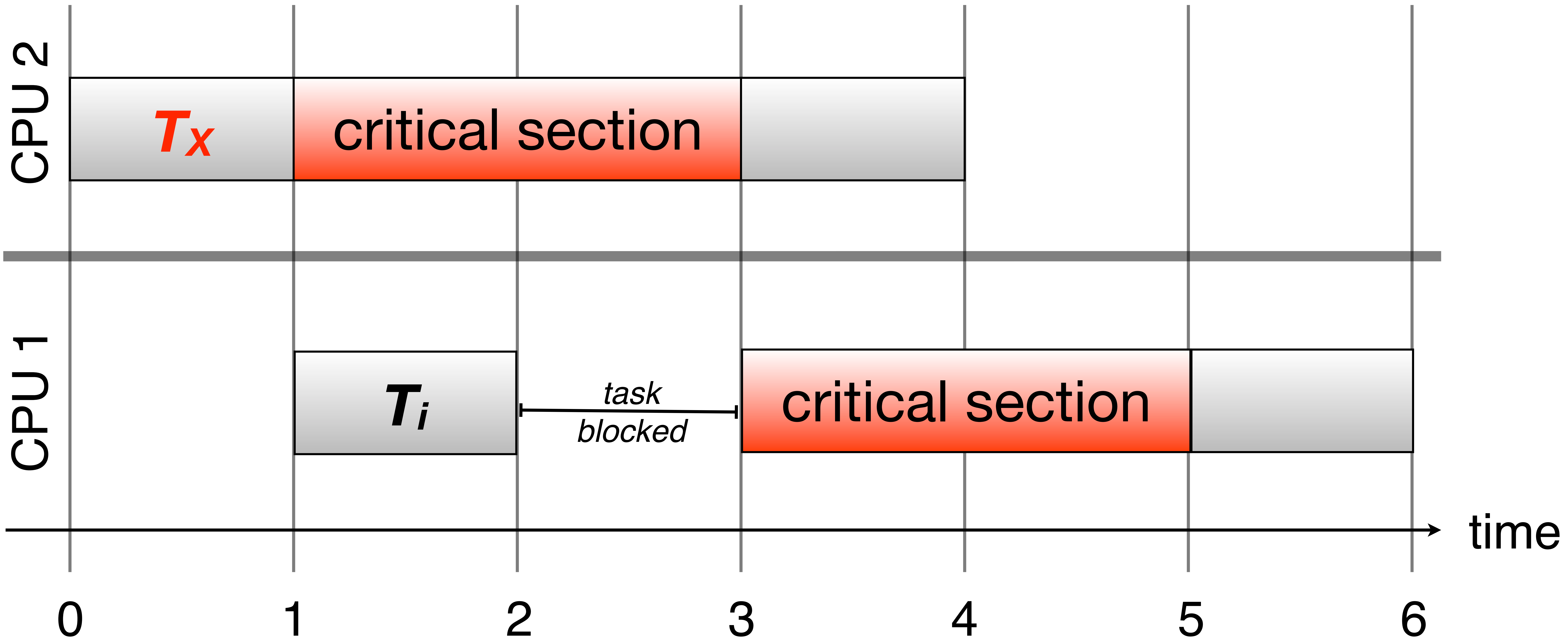
For the v^{th} concurrent critical section
of conflicting task T_x w.r.t. resource q:

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

$$0 \leq X_{x,q,v} \leq 1$$

Blocking Fraction — Example

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



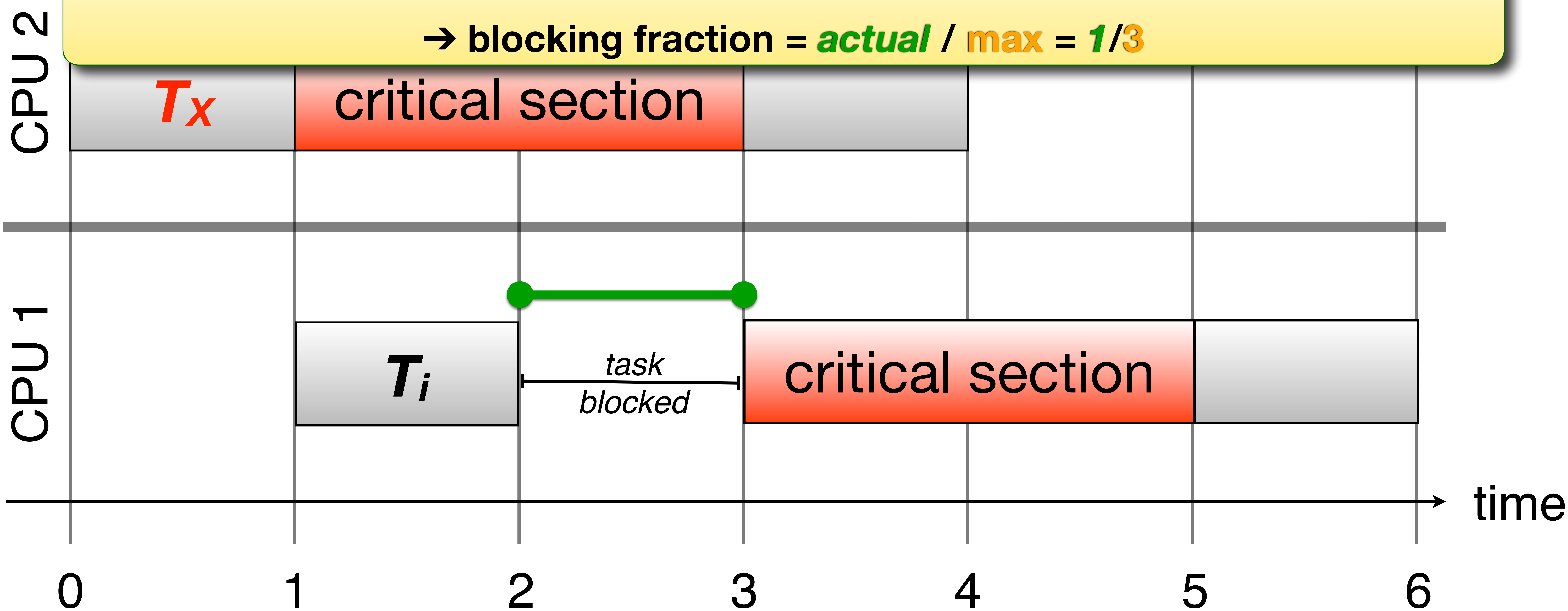
Blocking Fraction — Example

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

In this particular schedule

actual blocking **=** ***1 time unit***

→ blocking fraction = ***actual*** **/** **max** **=** ***1*****/3**



Total Blocking in a Fixed Schedule

total blocking incurred by one job =

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

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

Total B… le

Actual amount of blocking incurred (may be zero).

total blocking incurred by one job =

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

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 B e

Linear combination of ***all*** blocking fractions!
Use this as the **objective function**
of a **linear program (LP)**.

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

maximize

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

subject to

\$WORKLOAD-CONSTRAINTS

\$PROTOCOL-CONSTRAINTS

From a Fixed to All Possible Schedules

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

maximize

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

Rule out impossible schedules.

subject to

$WORKLOAD-CONSTRAINTS

$PROTOCOL-CONSTRAINTS

Example FIFO Constraint

rule out impossible schedules not compliant with FIFO ordering

Constraint 12. *In any schedule of τ under the FMLP$^+$:*

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

Example FIFO Constraint

with FIFO ordering

For **each resource *q*** and **each conflicting task *T_x***…

Constraint 12. *In any schedule of τ 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

… with FIFO ordering

For **each resource *q*** and **each conflicting task *T*$_x$**…

…all **possibly concurrent** critical sections ***v***…

Constraint [illegible]2. *In any schedule of τ under the FMLP$^+$:*

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

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 τ under the FMLP$^+$:*

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

…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 τ under the FMLP$^+$:*

$$\forall \ell_q : \boxed{\forall T_x} \quad : \boxed{\sum_{v=1}^{N_{x,q}^{i}} X_{x,q,v}} \leq \boxed{N_{i,q}.}$$

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 τ under the FMLP$^+$:*

$$\forall \ell_q : \forall T_x \in \tau^i : \sum_{v=1}^{N^i_{x,q}} X^D_{x,q,v} \leq N_{i,q}.$$

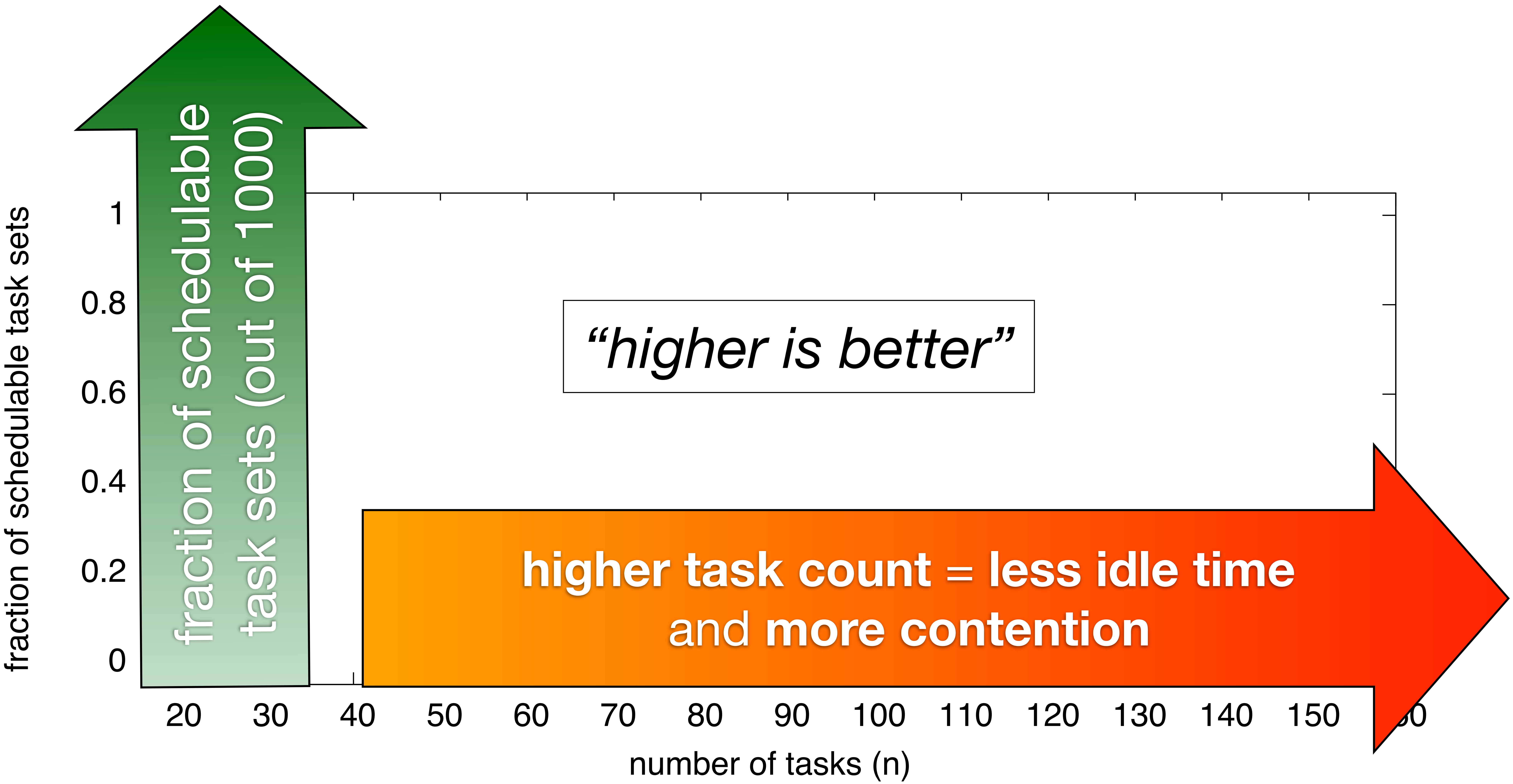
Powerful analysis technique

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

Accuracy

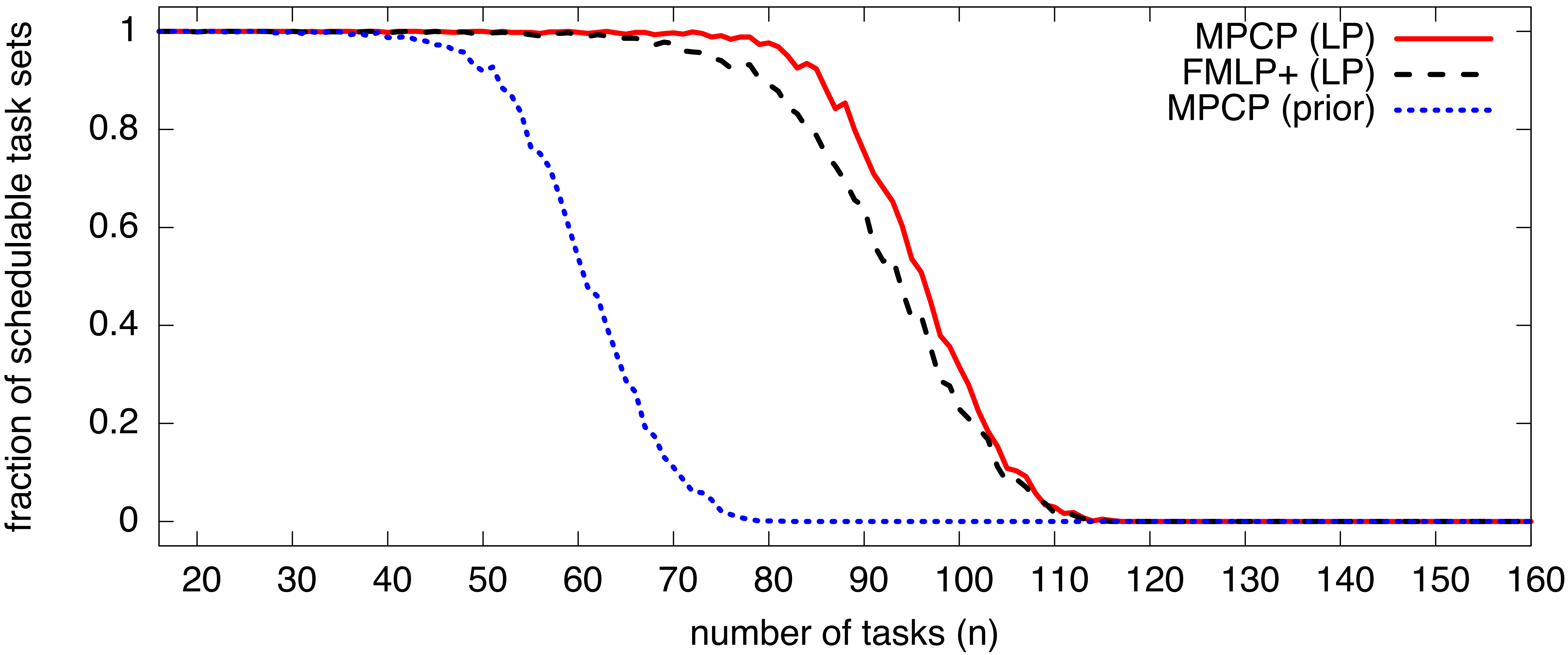
- ➡ **Never counts a request twice**.
- ➡ Much less pessimistic…

Improved Analysis Accuracy



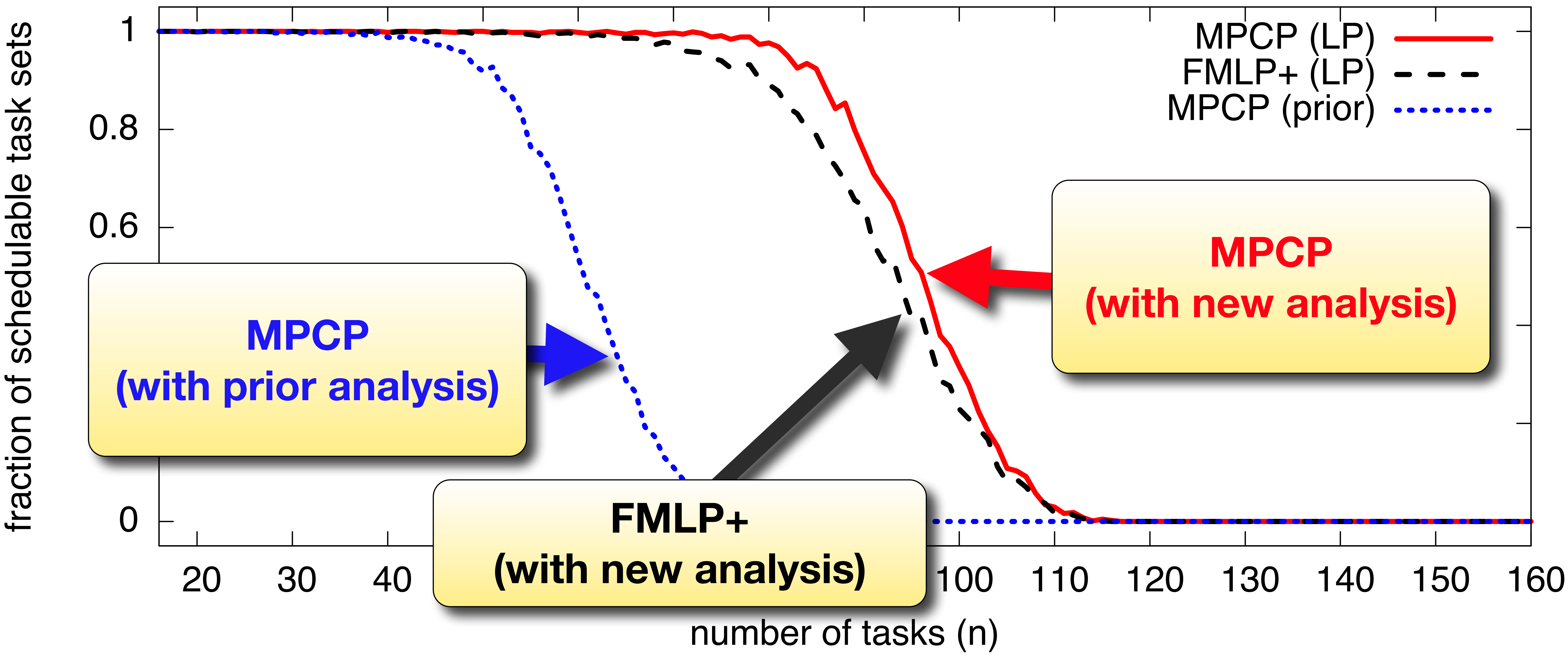
Improved Analysis Accuracy

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



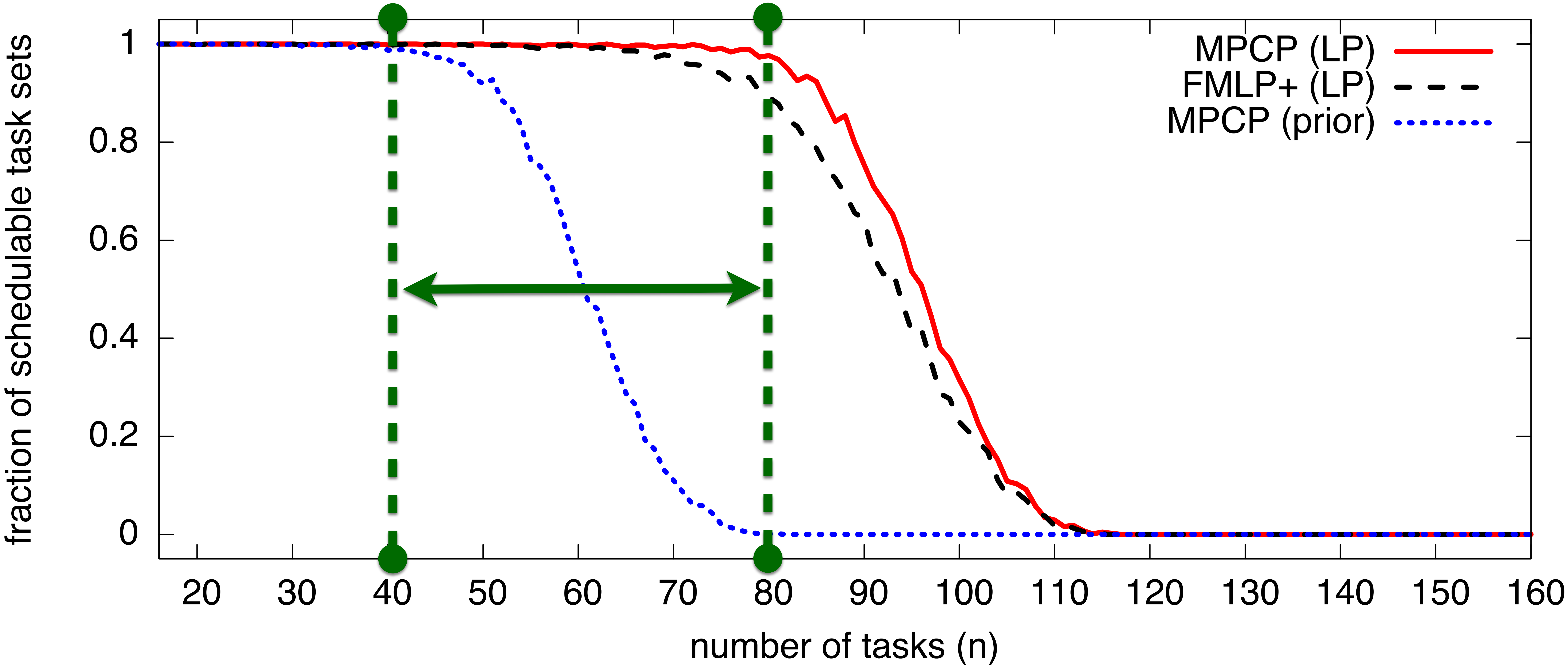
Improved Analysis Accuracy

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



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

	MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)	**FMLP+** (Brandenburg, 2011)	**DPCP** (Rajkumar et al., 1988)	**DFLP** (Brandenburg, 2012)
Wait Queue	**priority** queue	**FIFO** queue	**priority** queue	**FIFO** queue
Protocol Type	**shared-memory**	**shared-memory**	**distributed**	**distributed**
Asymptotically optimal? *(w.r.t. maximum blocking)*	**NO**	**YES**	**NO**	**YES**

Distributed locking protocols require **cross-core** interaction.

Overheads matter…

	MPCP (Rajkumar, 1990; Lakshmanan et al., 2009)	**FMLP+** (Brandenburg, 2011)	**DPCP** (Rajkumar et al., 1988)	**DFLP** (Brandenburg, 2012)
Wait Queue	**priority** queue	**FIFO** queue	**priority** queue	**FIFO** queue
Protocol Type	**shared-memory**	**shared-memory**	**distributed**	**distributed**
Asymptotically optimal? *(w.r.t. maximum blocking)*	**NO**	**YES**	**NO**	**YES**

Taking Overheads Into Account

Platform

LITMUS^RT

Linux Testbed for Multiprocessor Scheduling in Real-Time Systems

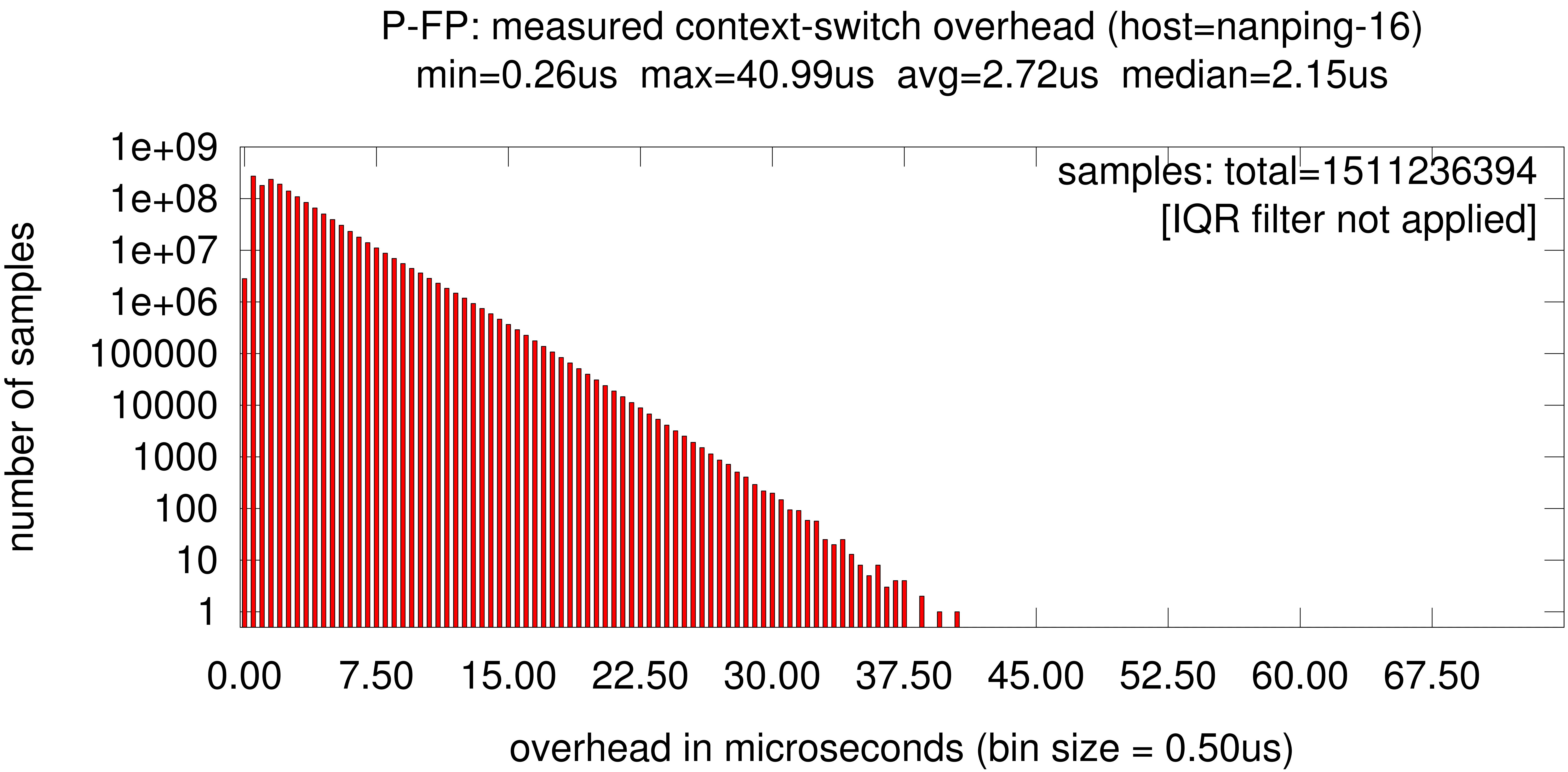
www.litmus-rt.org

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

Overhead Tracing

- ➡ > 20h of real-time exeuction
- ➡ > 400 GB of trace data
- ➡ > 15 billion valid samples
- ➡ Statistics and details in online appendix.

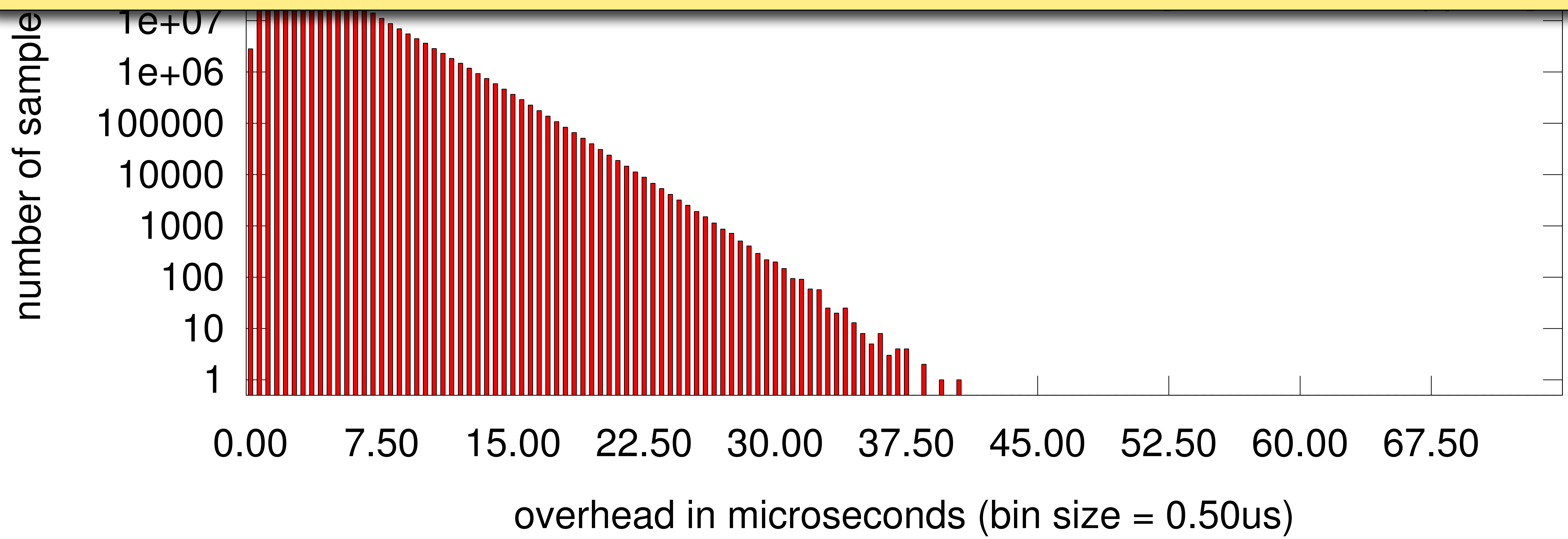
Example: Context Switch Overhead



max = 40.99μs avg = 2.72μs med = 2.15μs

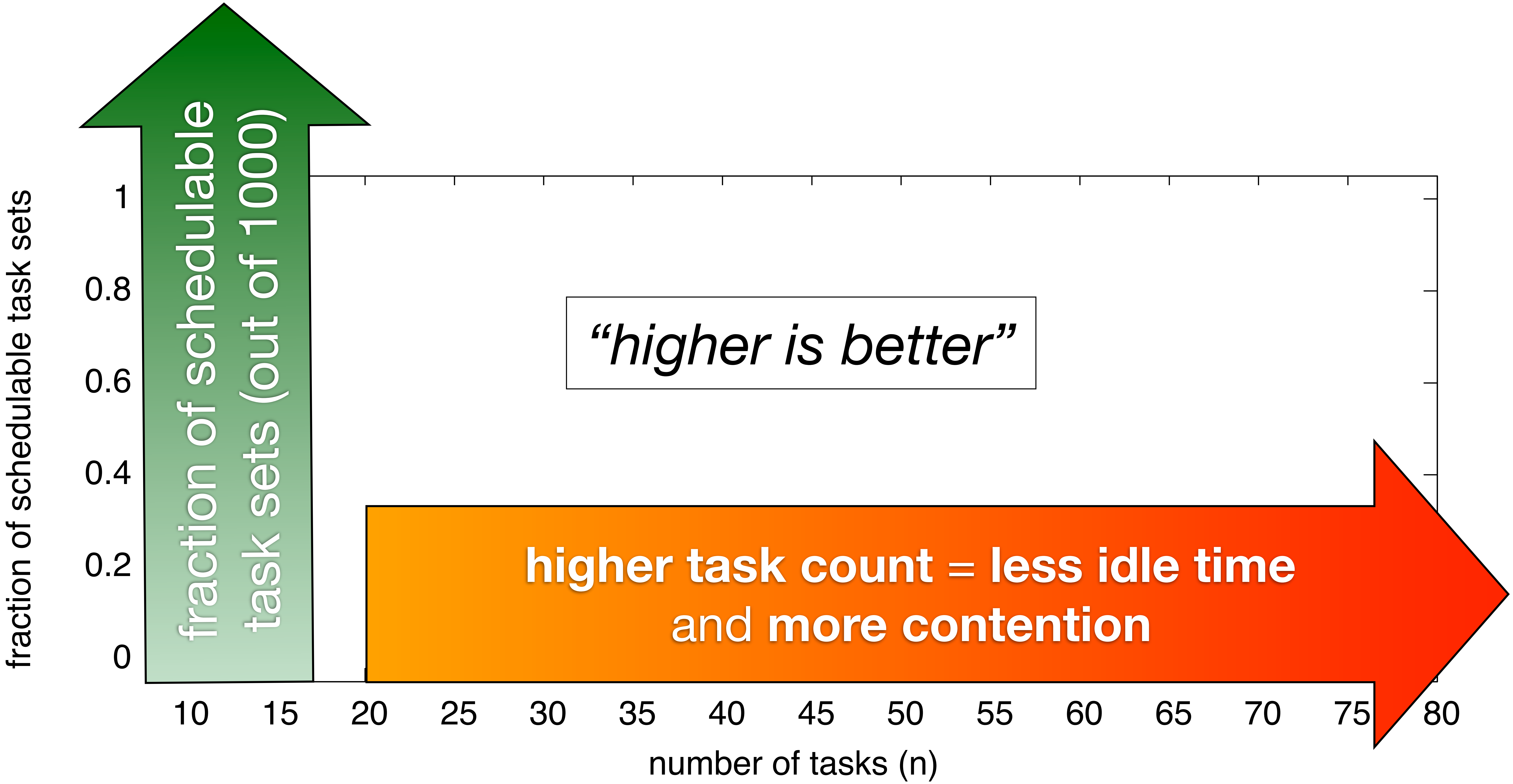
Overhead Experiments:

No statistical outlier filtering was applied.



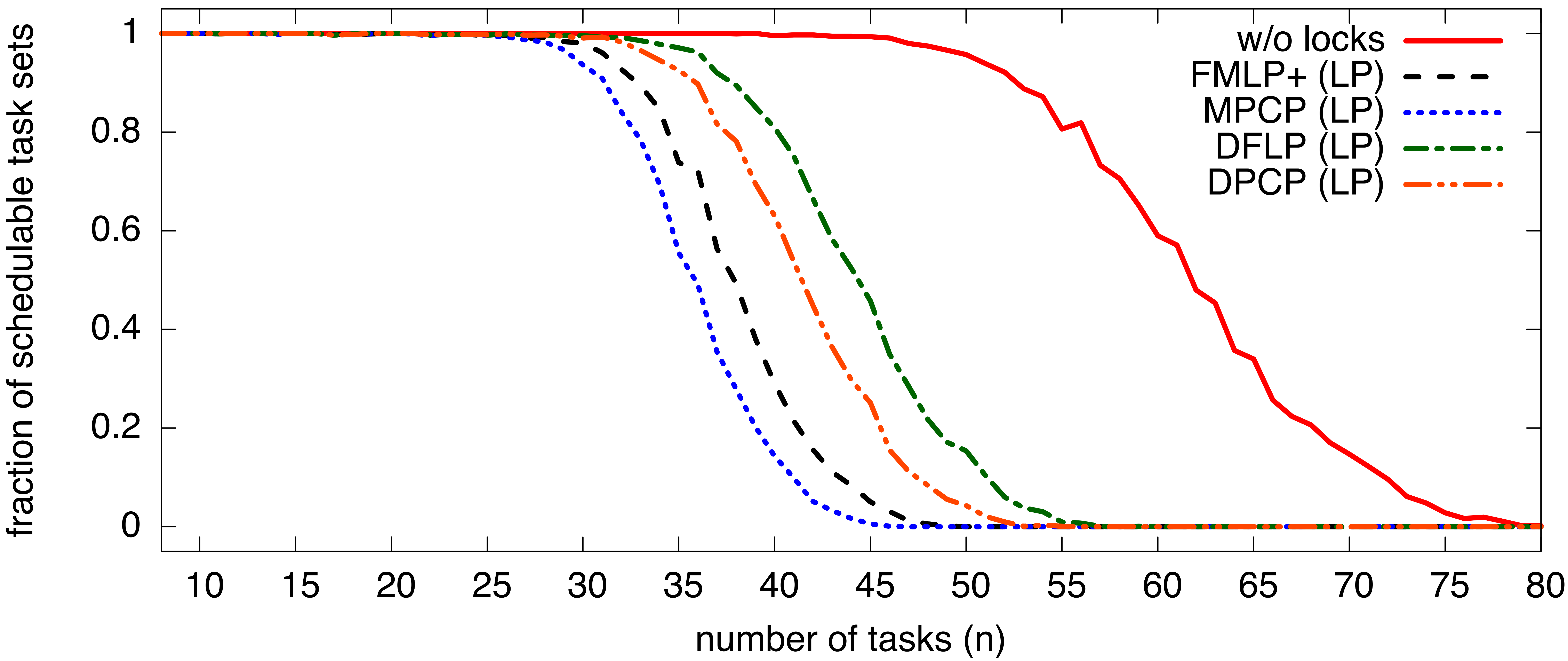
max = 40.99μs avg = 2.72μs med = 2.15μs

Overhead-Aware Schedulability



Overhead-Aware Schedulability

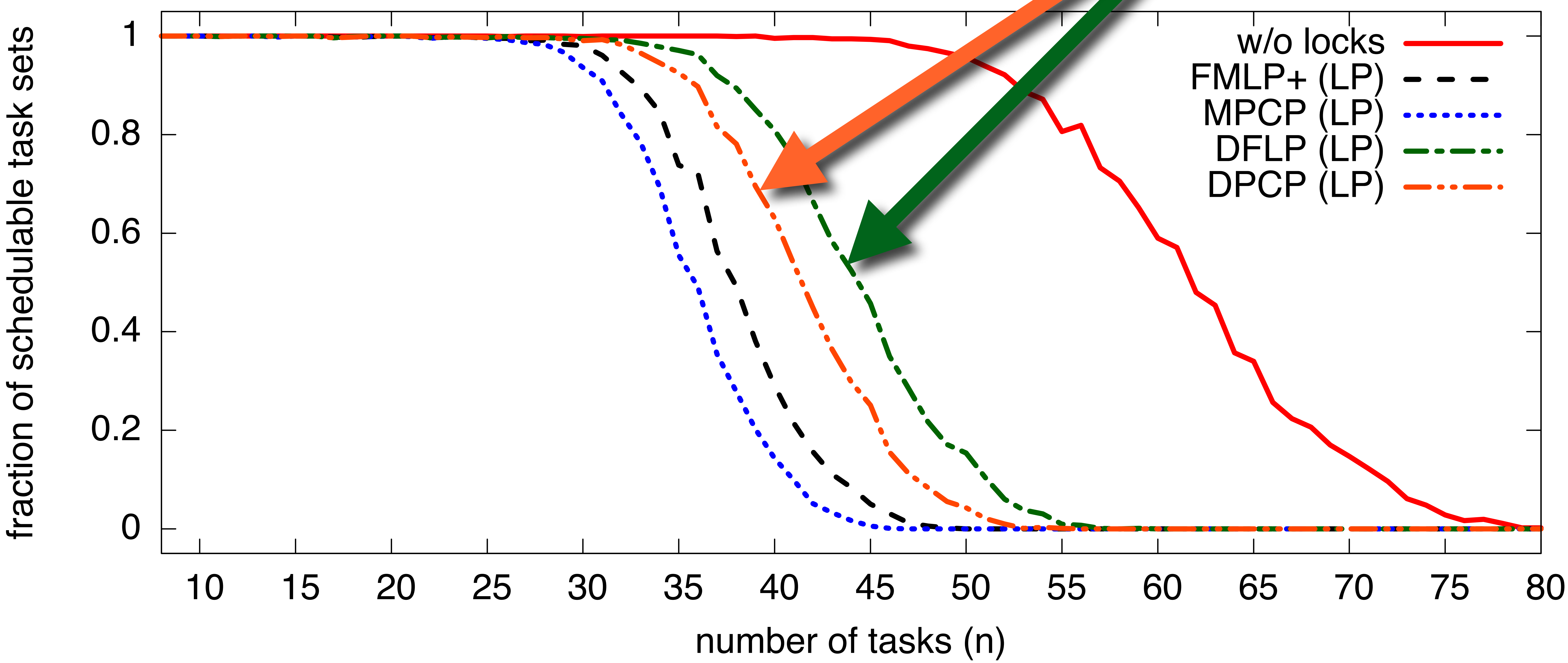
8 cores, 8 shared resources, max. 5 critical sections per task and resource, 10μs-50μs CS length, each task accesses a given resource with probability 0.3



Distributed Protocols Perform Well

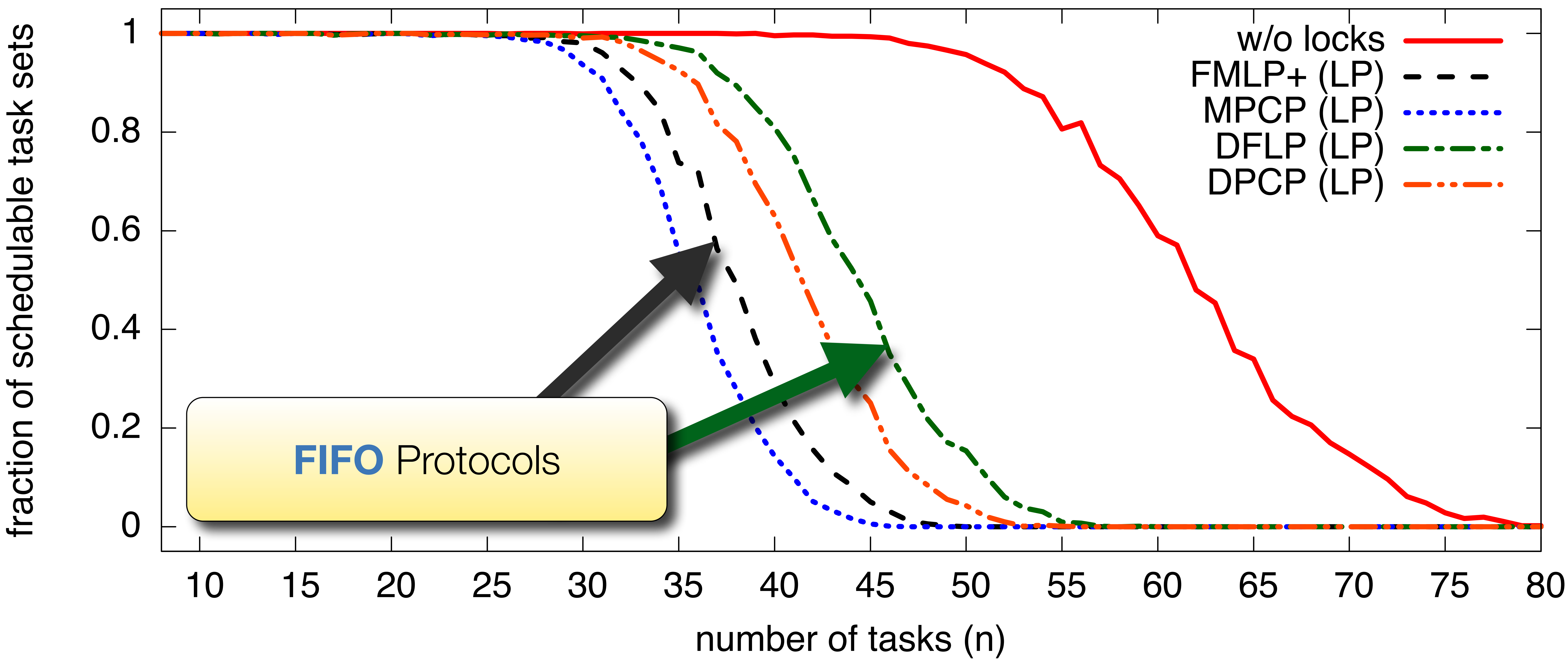
Distributed Protocols

8 cores, 8 shared resources, max. 5 critica esource,
10μs-50μs CS length, each task accesses a given re urce with probability 0.3



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

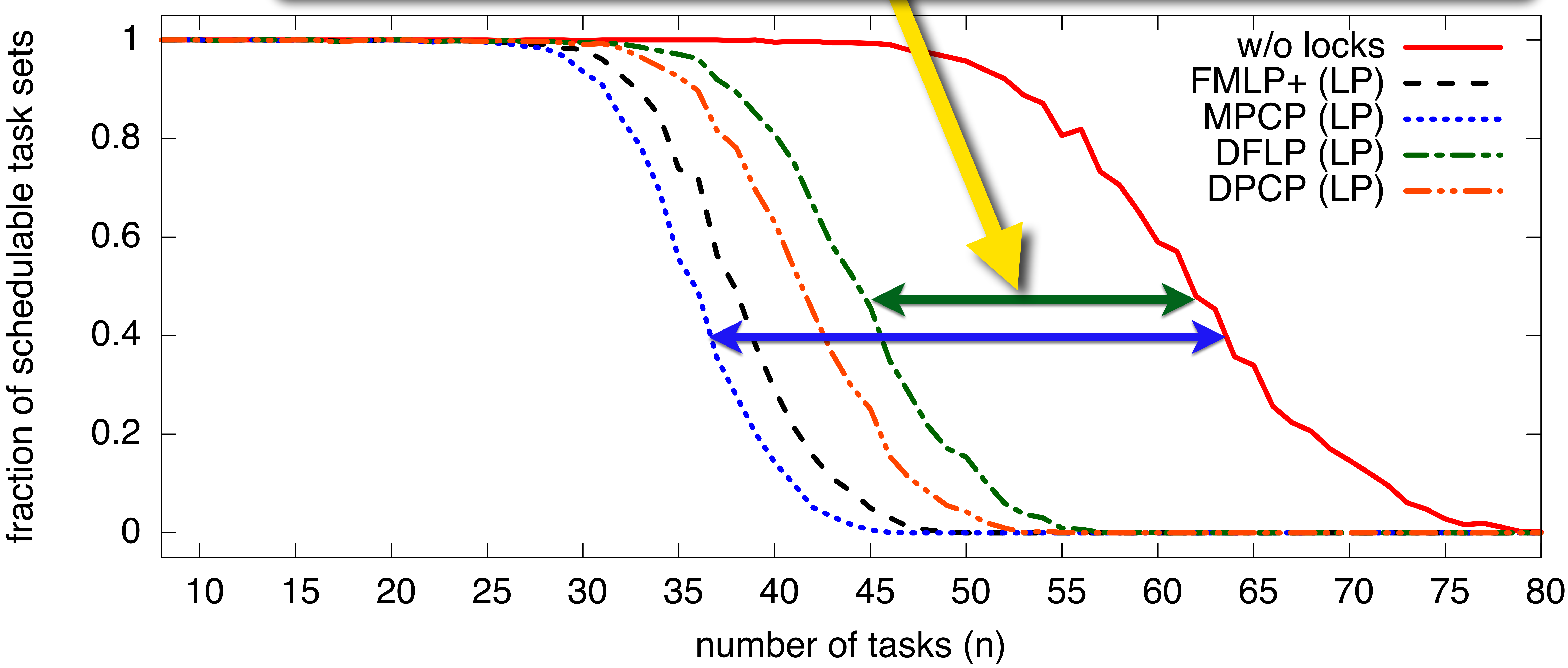


Choice of Protocol Matters

Blocking is a significant bottleneck. w.r.t. schedulability.

In this example: **DFLP** can host **>20%** more tasks than **MPCP**.

8 cores ... ce,
10μs-50μ ... 0.3



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!

LITMUS RT

Linux Testbed for Multiprocessor Scheduling in Real-Time Systems

www.litmus-rt.org

SchedCAT

Schedulability test **C**ollection **A**nd **T**oolkit

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