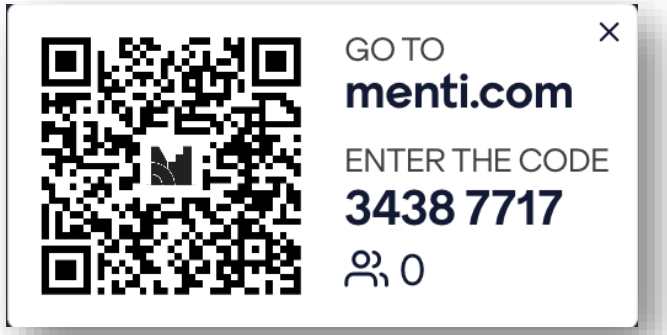
Optimal Multiprocessor Locking Protocols Under FIFO Scheduling

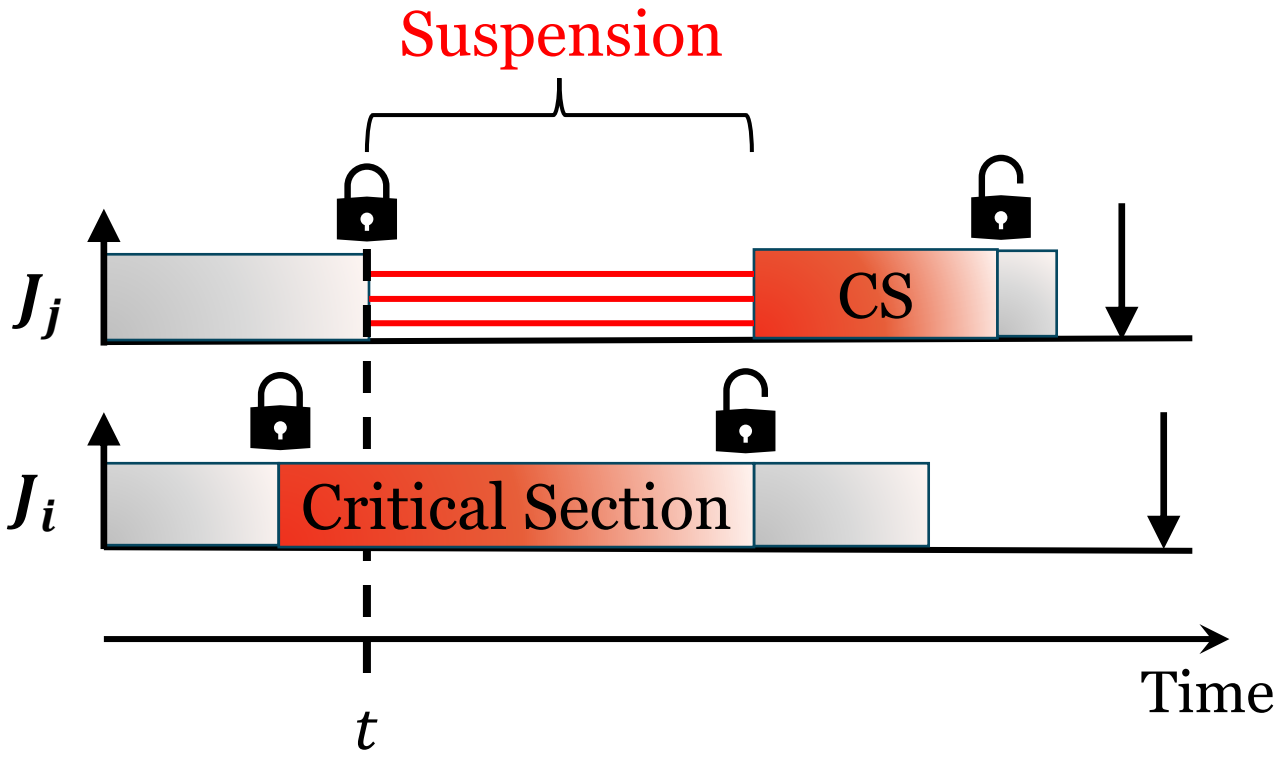
Shareef Ahmed and Jim Anderson



Background

Mutex locks

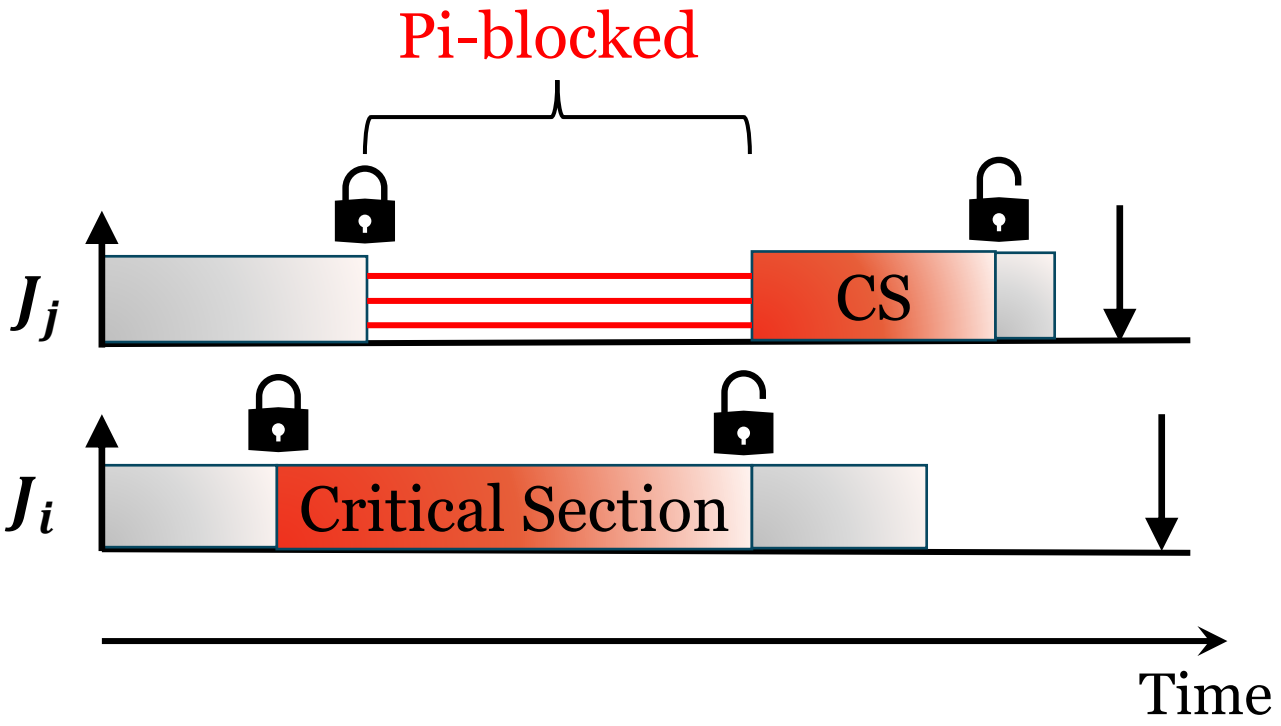
```
pthread_mutex_lock(…)
//critical section
Pthread_mutex_unlock(…)
```



Background

Pi-blocking

A job is NOT scheduled when it should be

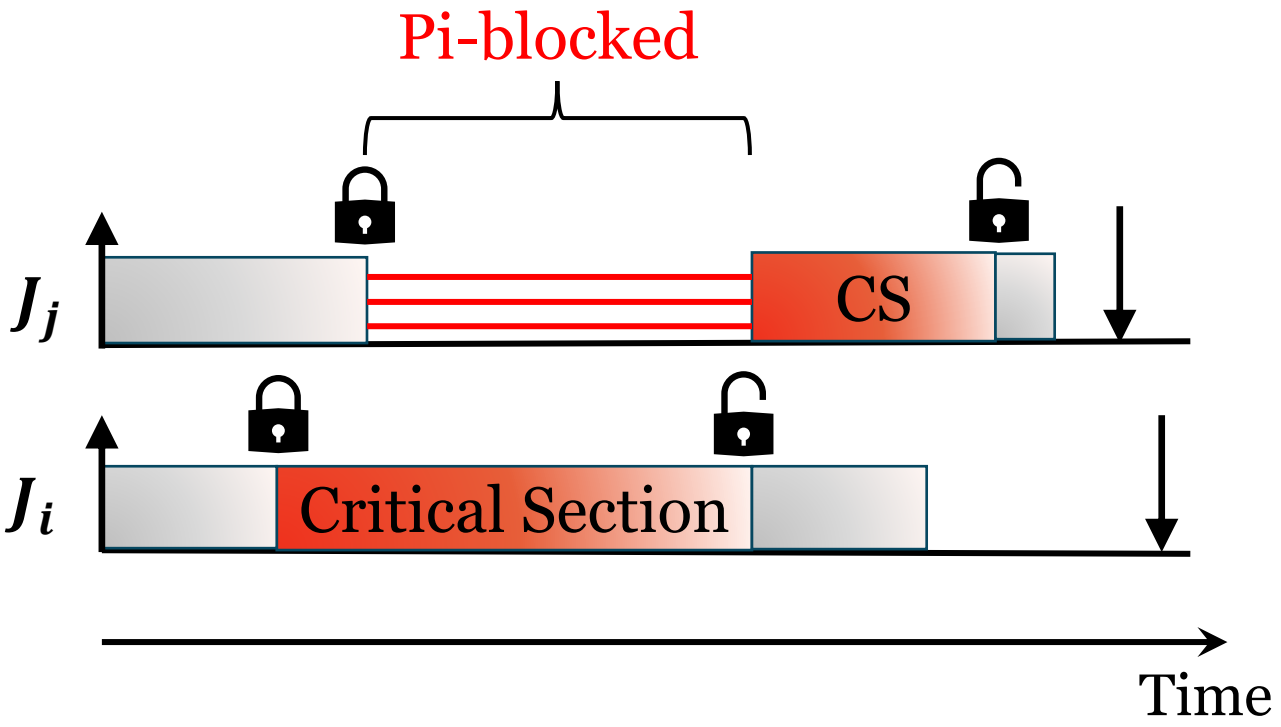


J_j is pi-blocked assuming it has high-enough priority to be scheduled

Background

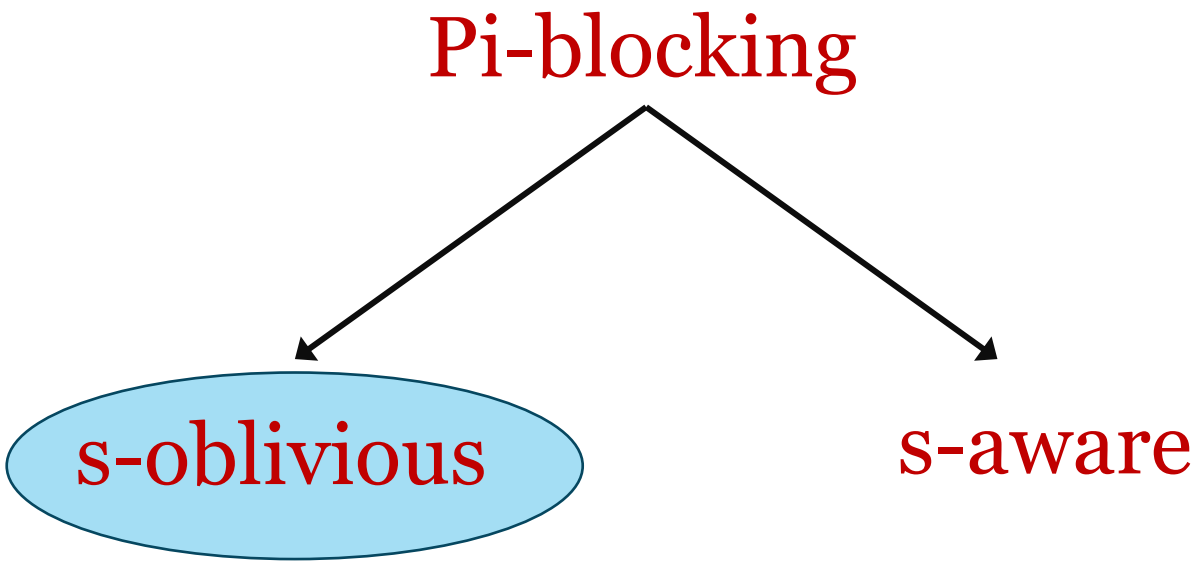
Pi-blocking

A job is NOT scheduled when it should be



Goal: Bounded maximum pi-blocking time in terms of CS lengths

Background

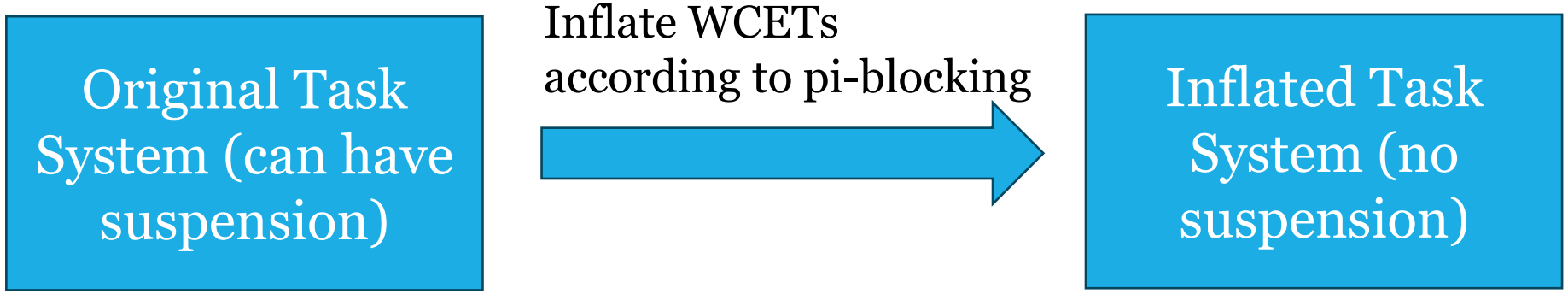


Focus of this work

Background

S-oblivious analysis:

- Schedulability analysis assumes that there is NO suspension



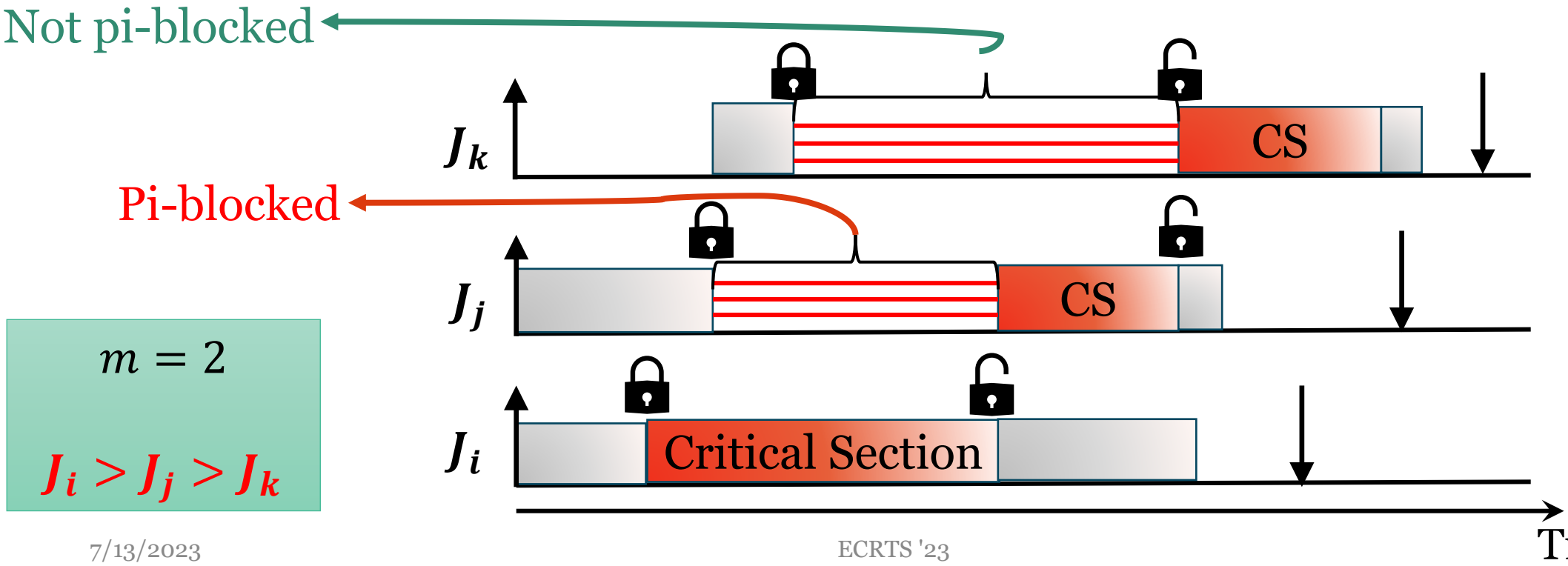
This talk:

- Global scheduling
- One shared resource and one CS per task

Background

S-oblivious pi-blocking:

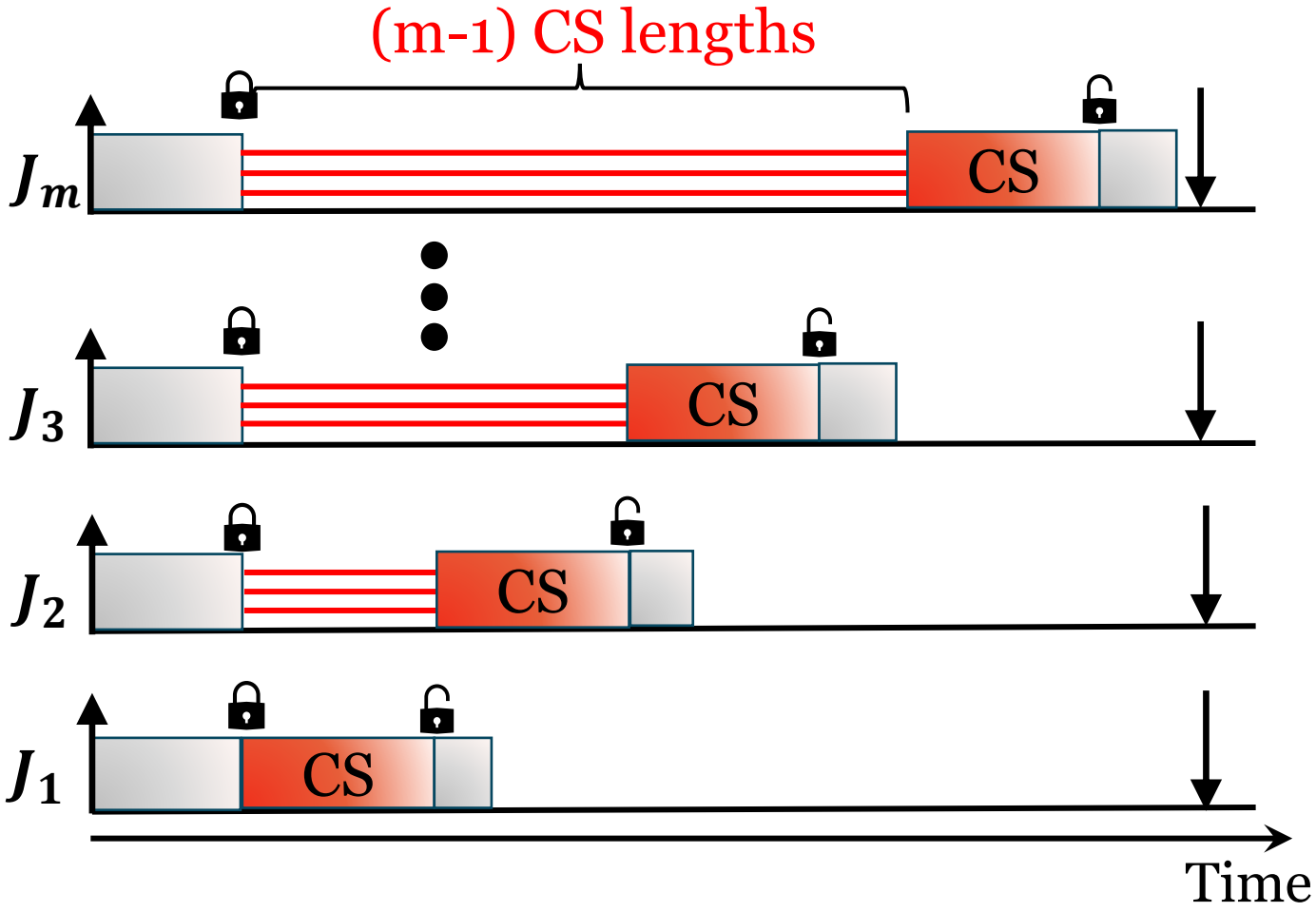
A job is pi-blocked if it is one of the top-m-priority pending jobs, but NOT scheduled [Brandenburg & Anderson, 2010]



Background

S-oblivious pi-blocking lower bound:

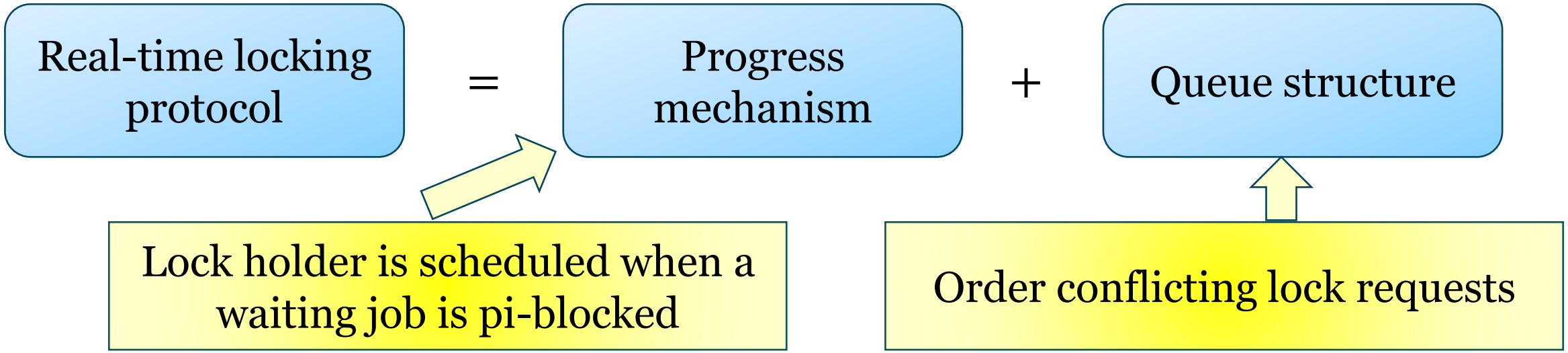
- Worst-case per-task pi-blocking ≥ (m-1) CS lengths [Brandenburg & Anderson, 2010]



Prior Work

Asymptotically optimal locking protocols:

- Ensures pi-blocking upper bound of O(m) CS lengths



Prior Work

L_{max} = max(CS length)

Global-OMLP [Brandenburg & Anderson, 2010]

- Works under **ANY** job-level fixed-priority global scheduler

Real-time locking protocol = Progress mechanism + Queue structure

G-OMLP

Priority inheritance

Hybrid FIFO and Priority queue

Known lower bound is $(m-1)$ CS lengths

Per-request pi-blocking $\leq (2m-1)L_{max}$

Prior Work

$L_{max} = \max(\text{CS length})$

Clustered-OMLP [Brandenburg & Anderson, 2011]

- Works under any job-level fixed-priority clustered scheduler

Real-time locking protocol = Progress mechanism + Queue structure

C-OMLP

Priority donation

FIFO queue

Per-request pi-blocking $\leq (m-1)L_{max}$

Prior Work

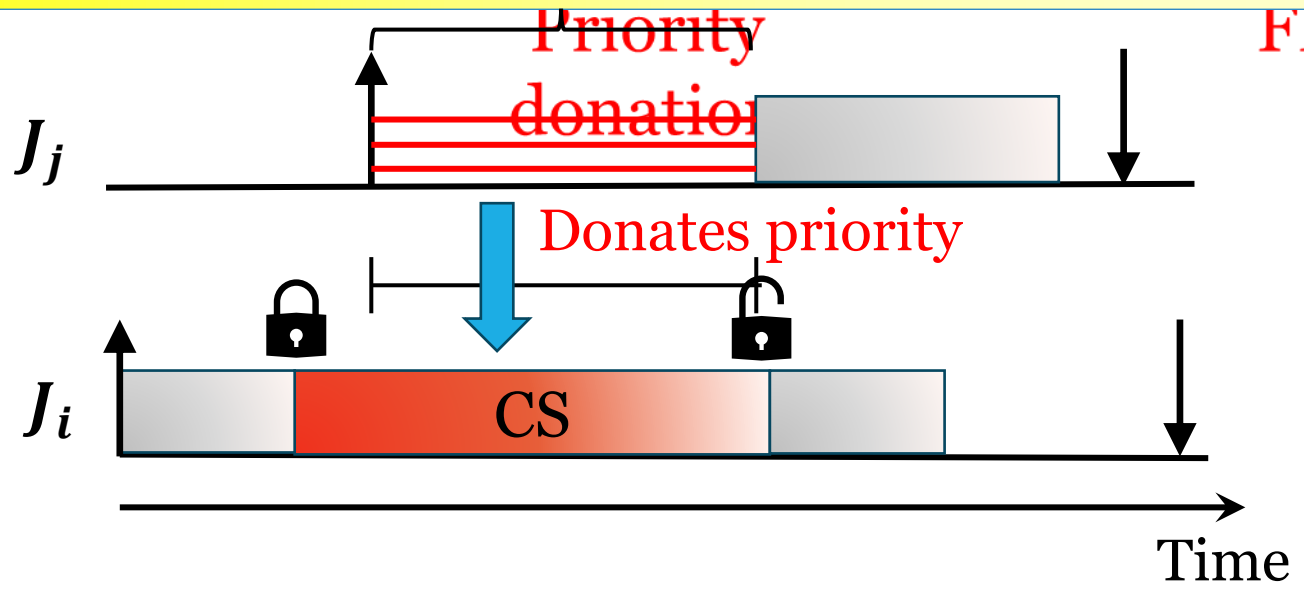
Clustered-OMLP:

Real-time locking prot = Progress + Queue structure

Pi-blocking at job release (**Release Blocking**) ≤ mL_{max}

C-OMLP

FIFO queue



Prior Work

L_{max} = max(CS length)

Clustered-OMLP:

Real-time locking protocol = Progress mechanism + Queue structure

C-OMLP = Priority donation + FIFO queue

Pi-blocking at job release $\leq mL_{max}$

Per-request pi-blocking $\leq (m-1)L_{max}$

Total pi-blocking $\leq (2m-1)L_{max}$

Prior Work

Protocol	Scheduling	Progress Mechanism	Release blocking	Request blocking	Total pi-blocking
G-OMLP	Global JLFP	Prio. Inheritance	0	$(2m-1)L_{max}$	$(2m-1)L_{max}$
C-OMLP	Clustered JLFP	Prio. Donation	mL_{max}	$(m-1)L_{max}$	$(2m-1)L_{max}$
OMIP	Clustered JLFP	Migratory Prio.	0	$(2m-1)L_{max}$	$(2m-1)L_{max}$

Question: Is pi-blocking of $(2m-1)$ CS lengths fundamental under **any JLFP scheduling**?

This Paper

Question: Is pi-blocking of $(2m - 1)$ CS lengths fundamental under **any JLFP scheduling**?

Answer: NO!

Pi-blocking of at most $(m - 1)$ CS lengths is possible under **FIFO** scheduling

This Paper

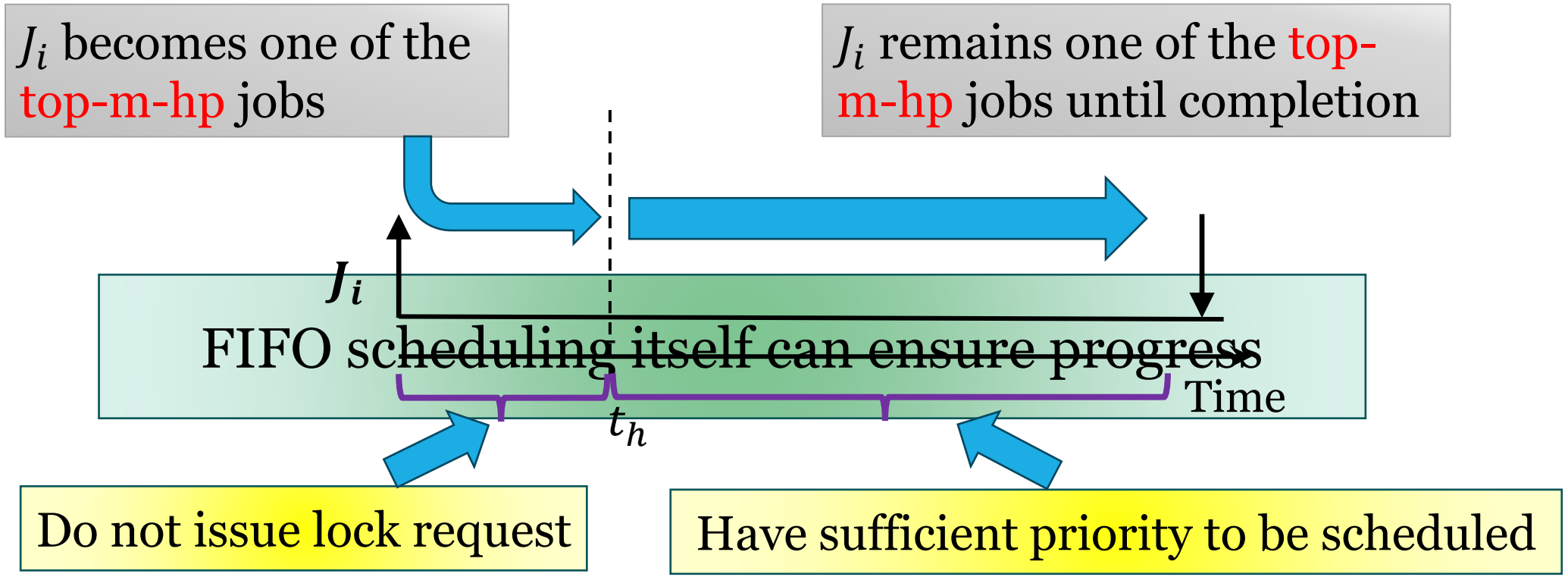
What's special in FIFO scheduling?

No preemption ⟹ No release blocking

FIFO scheduling itself can ensure progress.

NO explicit progress mechanism needed!

Progress Under FIFO



OLP-F: Optimal Locking Protocol under FIFO

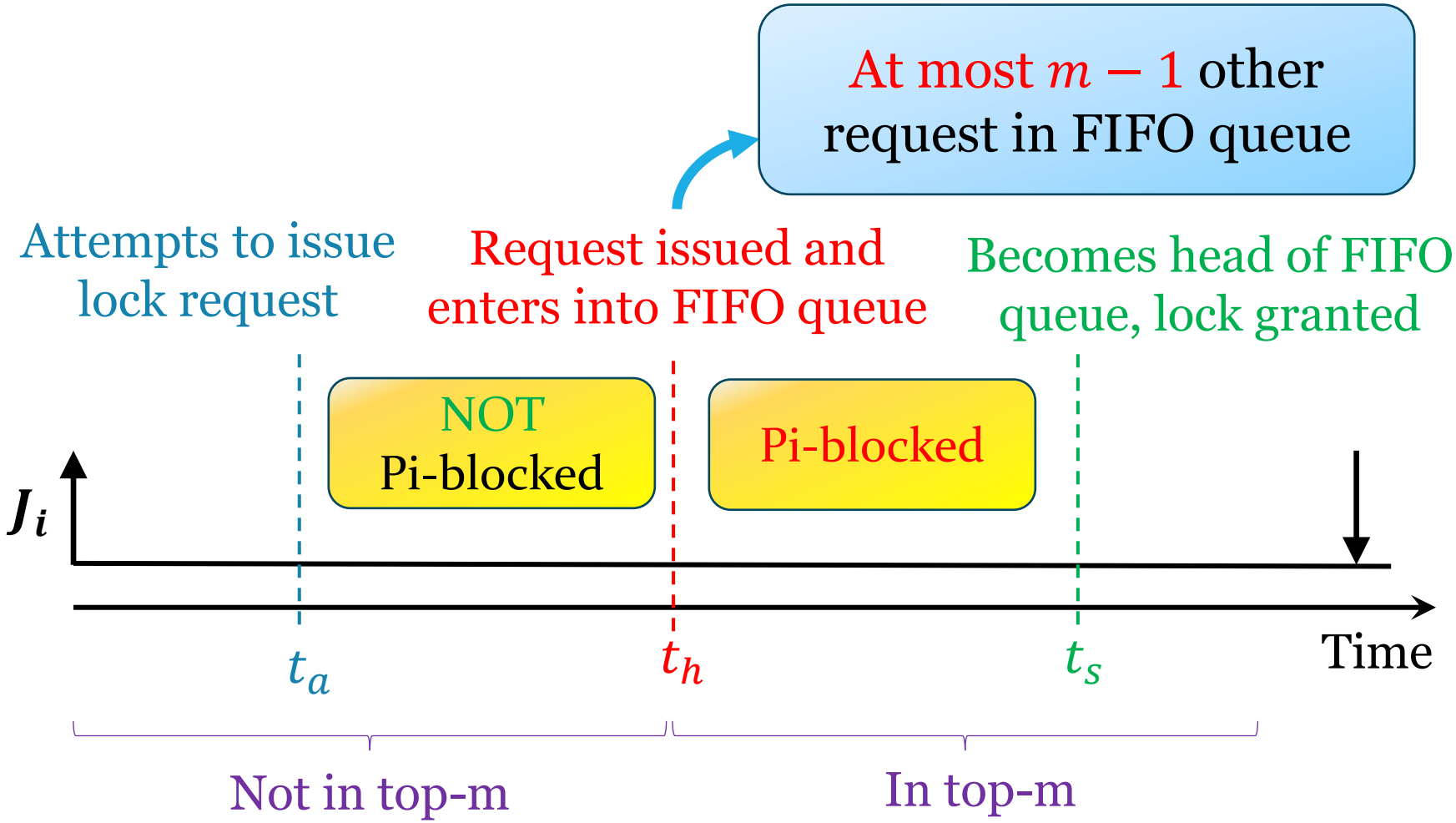
Real-time locking protocol = Progress mechanism + Queue structure

OLP-F

FIFO scheduling itself

FIFO queue

OLP-F: Optimal Locking Protocol under FIFO



OLP-F: Optimal Locking Protocol under FIFO

Real-time locking protocol = Progress mechanism + Queue structure

OLP-F

FIFO scheduling itself

FIFO queue

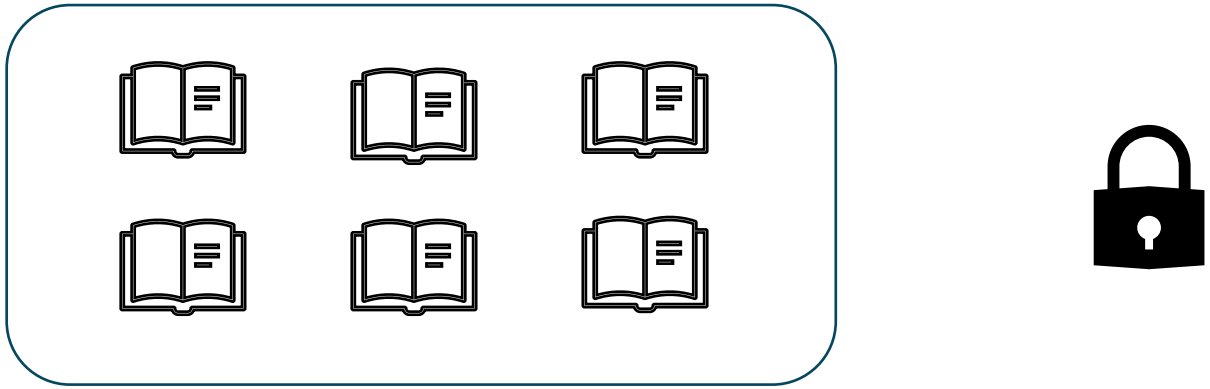
No Release blocking

Per-request pi-blocking $\leq (m-1)L_{max}$

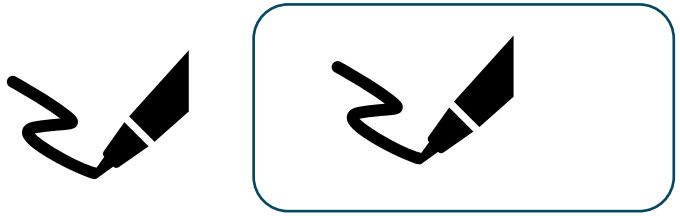
Total pi-blocking $\leq (m-1)L_{max}$

Reader Writer Lock

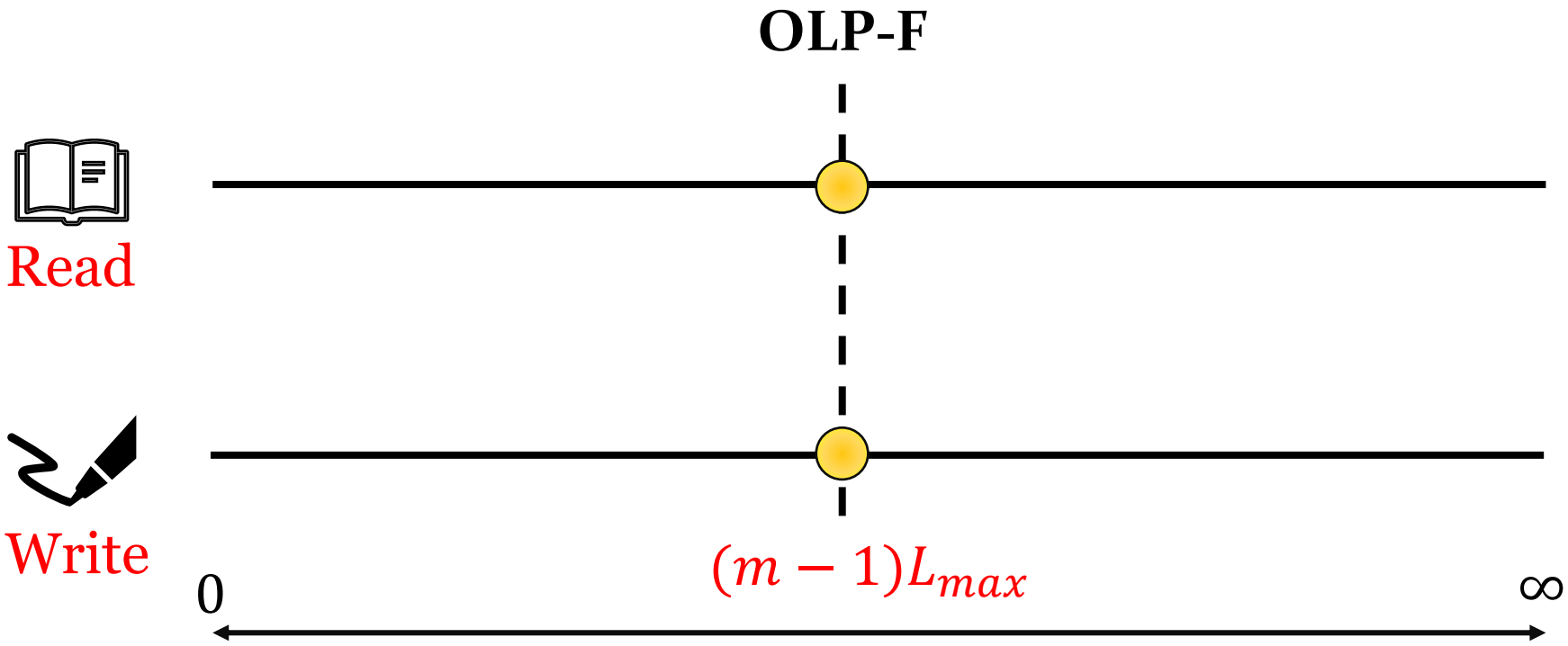
- Multiple read requests can be satisfied simultaneously



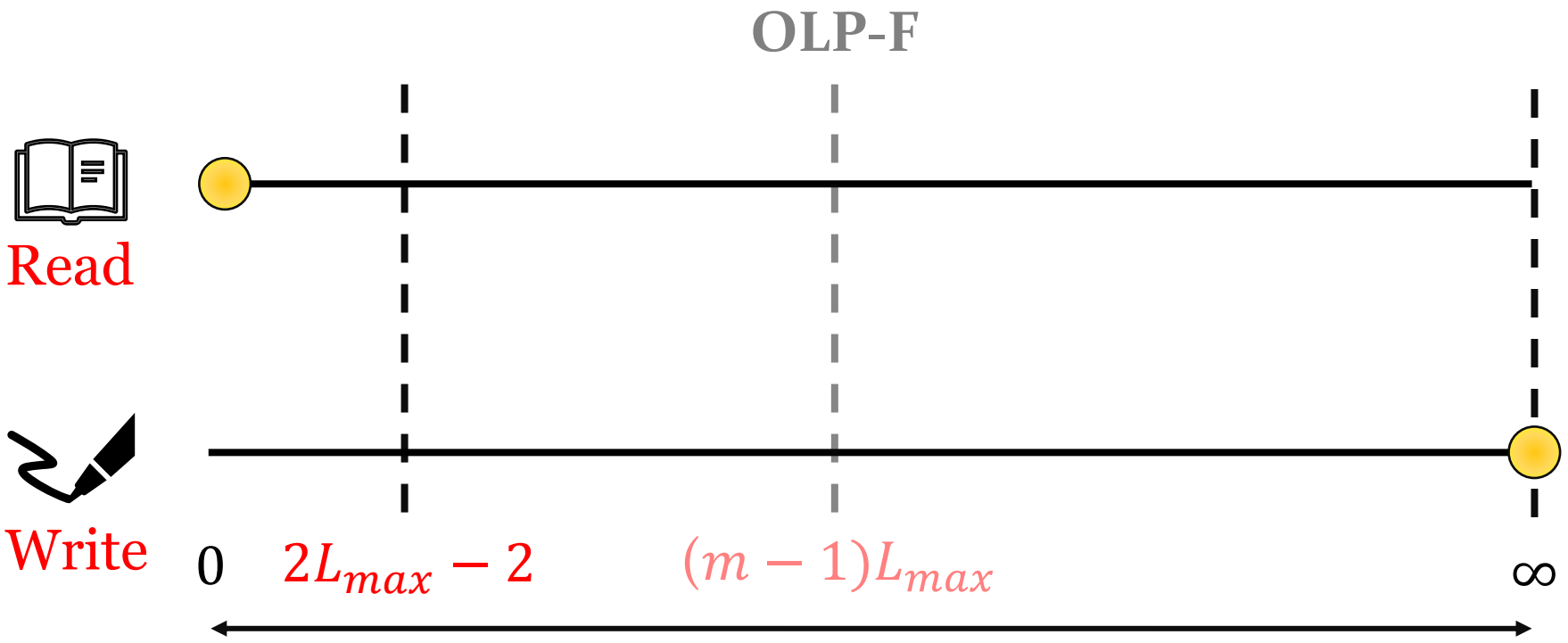
- Only one write requests can be satisfied at any time



Reader Writer Lock

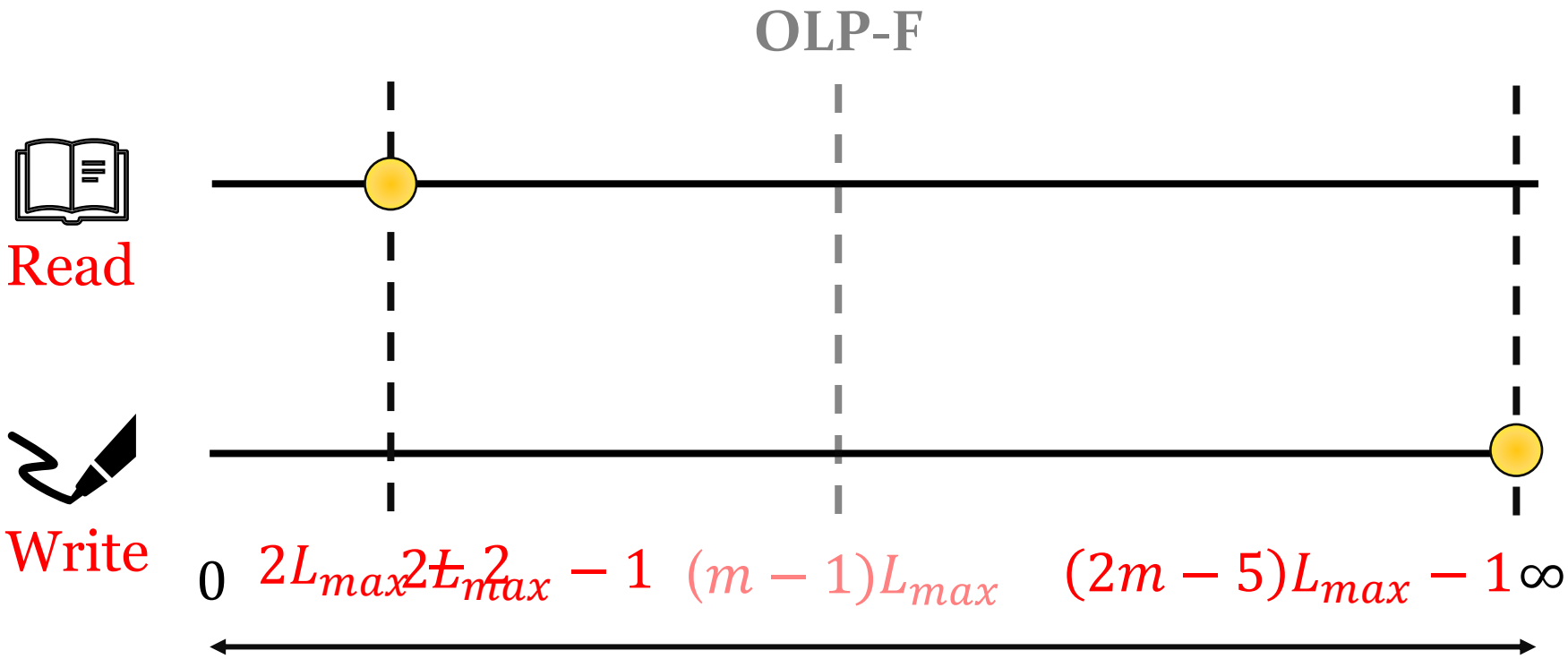


Reader Writer Lock



"$\leq 2L_{max} - 2$"read blocking can cause write starvation

Reader Writer Lock



"$\leq 2L_{max} - 1$" read blocking can cause "$\geq (2m-5)L_{max} - 1$" write blocking

RW-OLP-F

Optimal locking protocol for read requests under FIFO scheduling

Protocol	Release blocking	Read blocking	Write blocking
RW-OLP-F	0	$2L_{max} - 1$	$(2m-3)L_{max}$

Optimal

$\sim 2L_{max}$ larger

"$\leq 2L_{max} - 1$" read blocking can cause "$\geq (2m-5)L_{max} - 1$" write blocking

RW-OLP-F

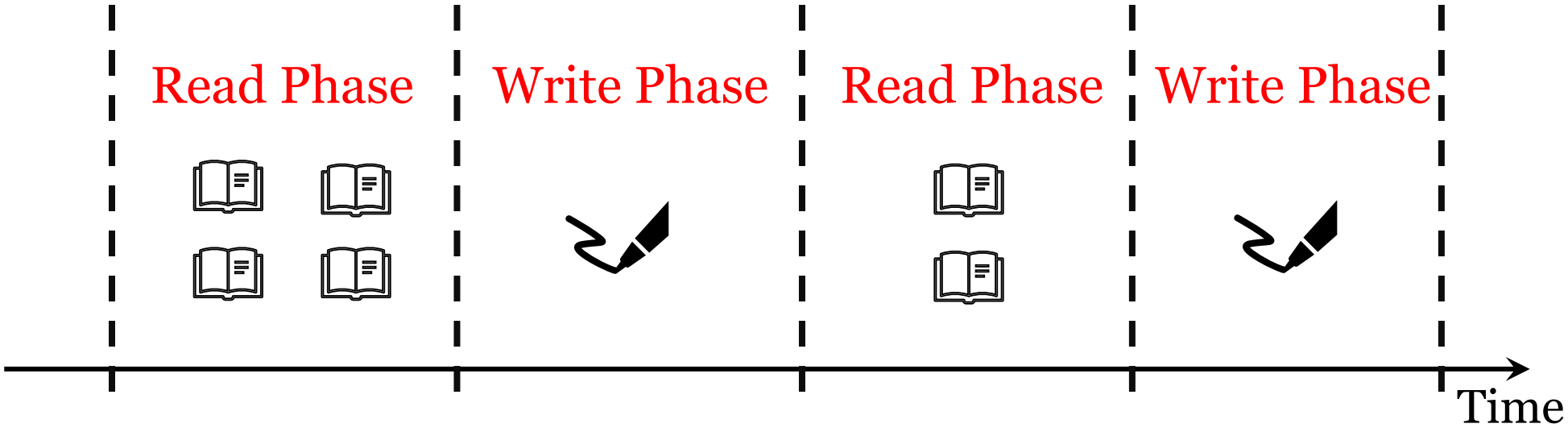
Optimal locking protocol for read requests under FIFO scheduling

Protocol	Release blocking	Read blocking	Write blocking
RW-OLP-F	0	$2L_{max} - 1$	$(2m-3)L_{max}$
CRW-OMLP	$2mL_{max}$	$2L_{max}$	$(2m-1)L_{max}$

"$\leq 2L_{max} - 1$" read blocking can cause "$\geq (2m-5)L_{max} - 1$" write blocking

RW-OLP-F

A phase fair RW lock

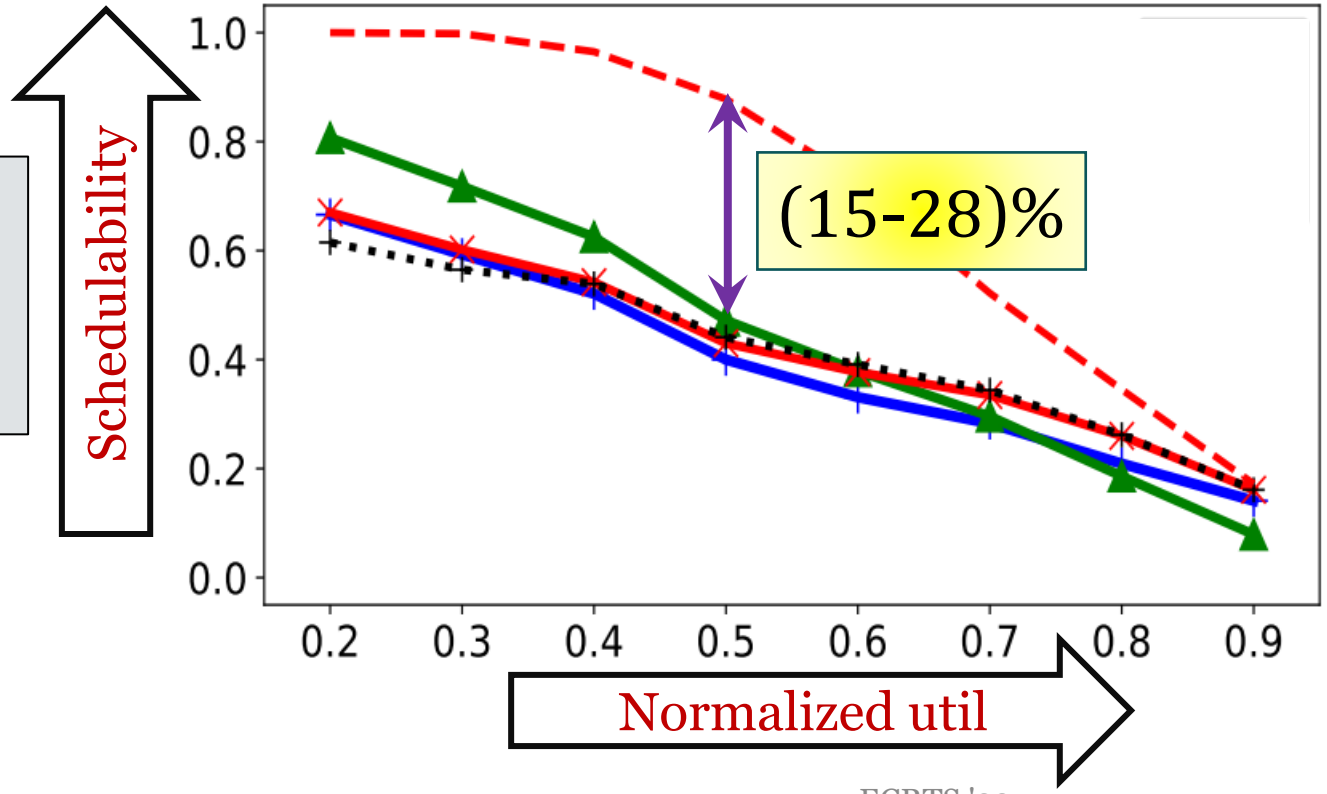


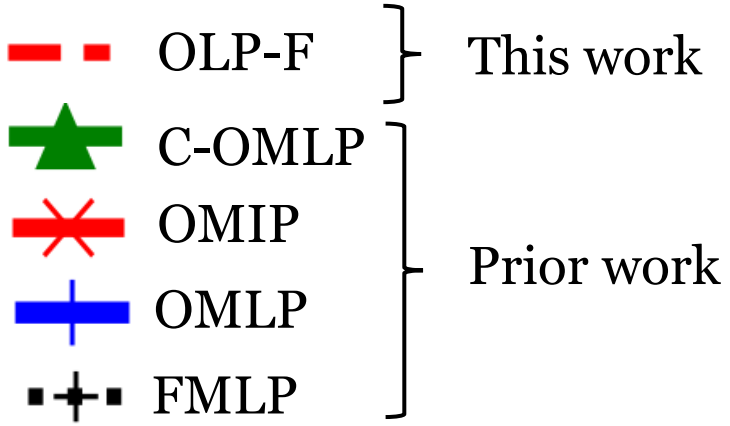
Experiments: Mutex Locks

Soft real-time schedulability test: schedulable iff bounded response time

#processor = 8, #resources = 2, periods ∈ [10,100]ms, CS length ∈ [1,100]μs

Higher area under the curve, the better



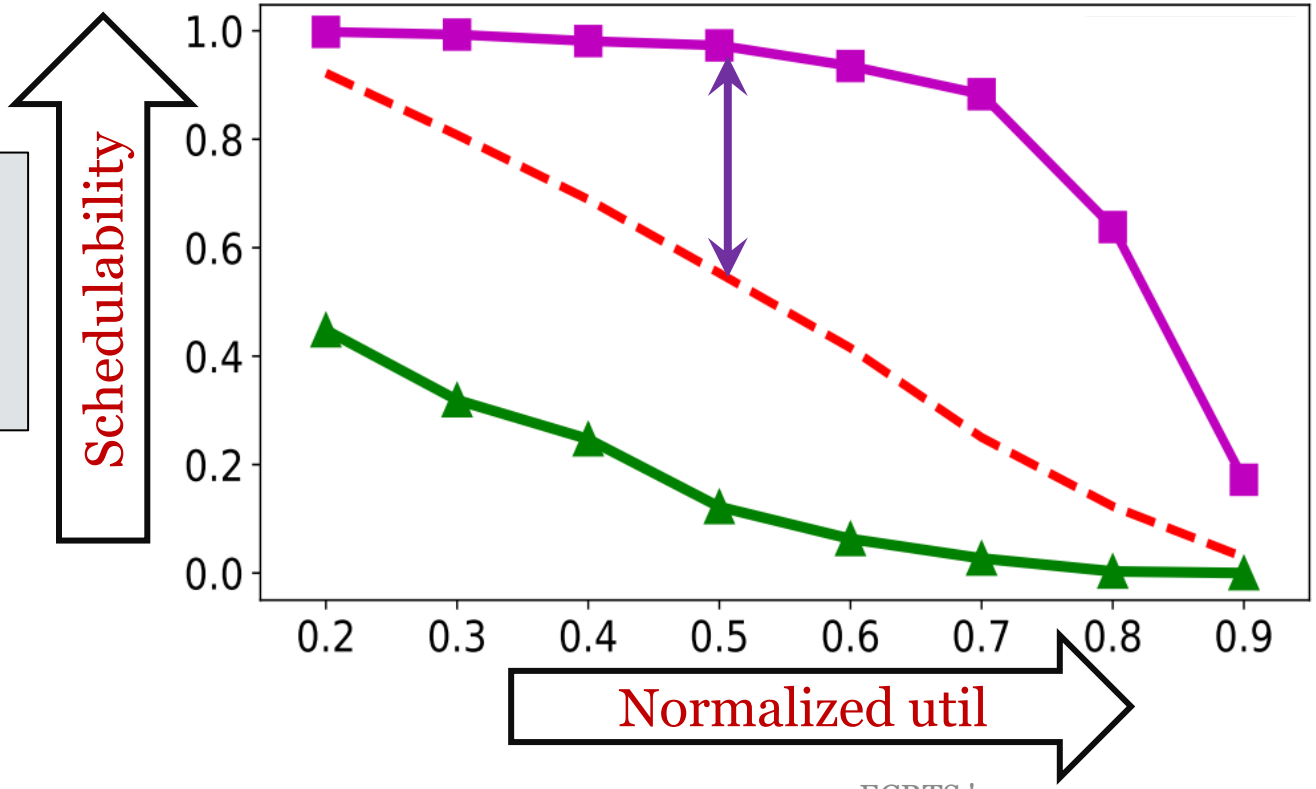


Experiments: RW Locks

Soft real-time schedulability test: schedulable iff bounded response time

#processor = 8, #resources = 32, periods ∈ [50,500]ms, CS length ∈ [1,100]μs, write~30%

Higher area under the curve, the better



RW-OLPF
OLP-F
} This work

CRW-OMLP
} Prior work

Conclusion

- The OLP-F is optimal under FIFO scheduling: pi-blocking for (m-1) request lengths
- The RW-OLP-F is read optimal under FIFO scheduling
- More on the paper: k-exclusion lock
- Similar principles applicable for spin-locks
- Future work:
  - What about other JLFP scheduling policies?

Thank You