

# Protected Hard Real-time: The Next Frontier



Bernard Blackham  
Yao Shi  
Gernot Heiser

Software Systems Group, NICTA  
The University of New South Wales  
Sydney, Australia



Australian Government  
Department of Broadband, Communications  
and the Digital Economy  
Australian Research Council

## NICTA Funding and Supporting Members and Partners



Australian  
National  
University

UNSW  
THE UNIVERSITY OF NEW SOUTH WALES



Griffith  
UNIVERSITY



Queensland  
Government

State Government  
Victoria

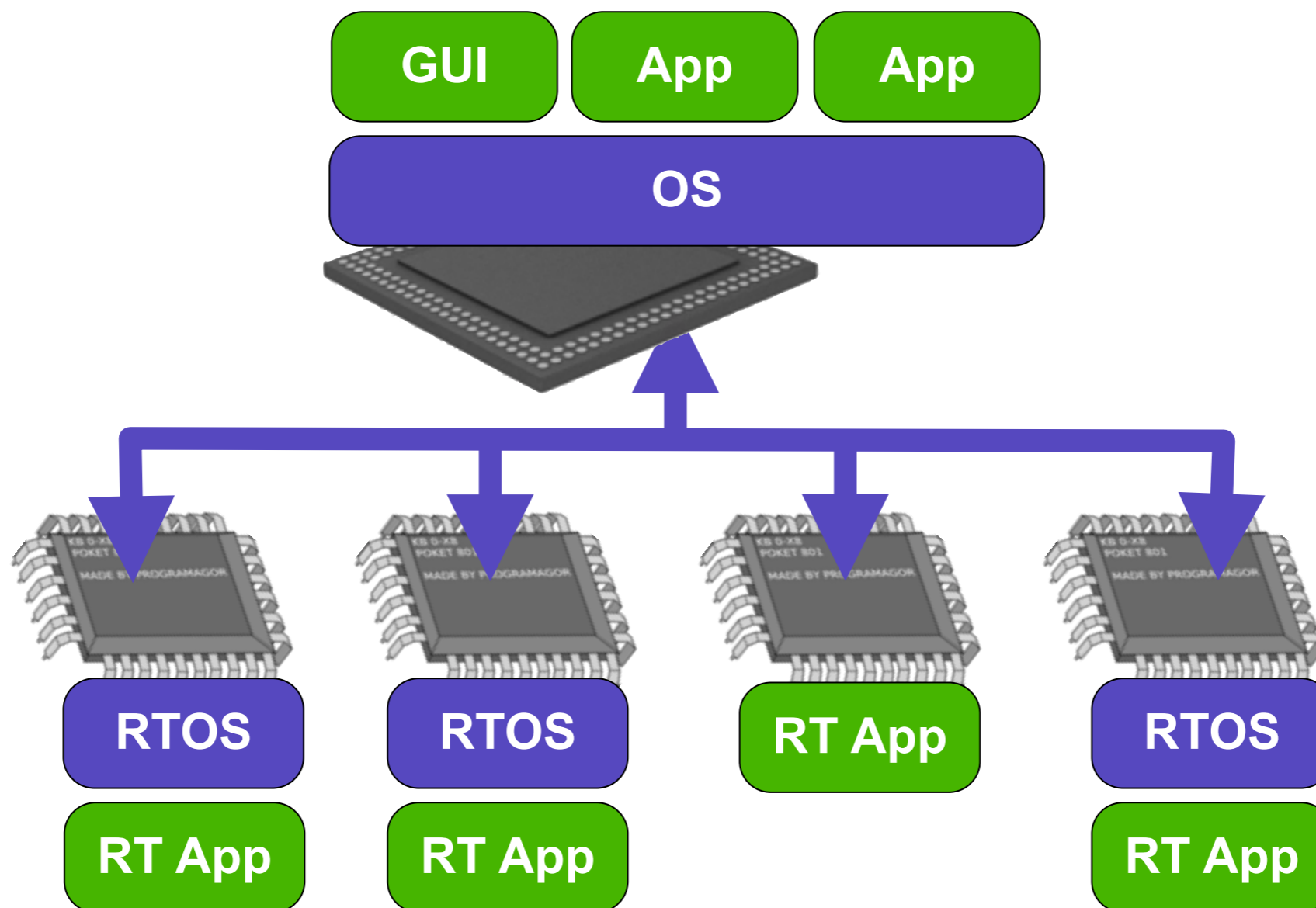
QUT  
Queensland University of Technology



THE UNIVERSITY OF  
QUEENSLAND  
AUSTRALIA

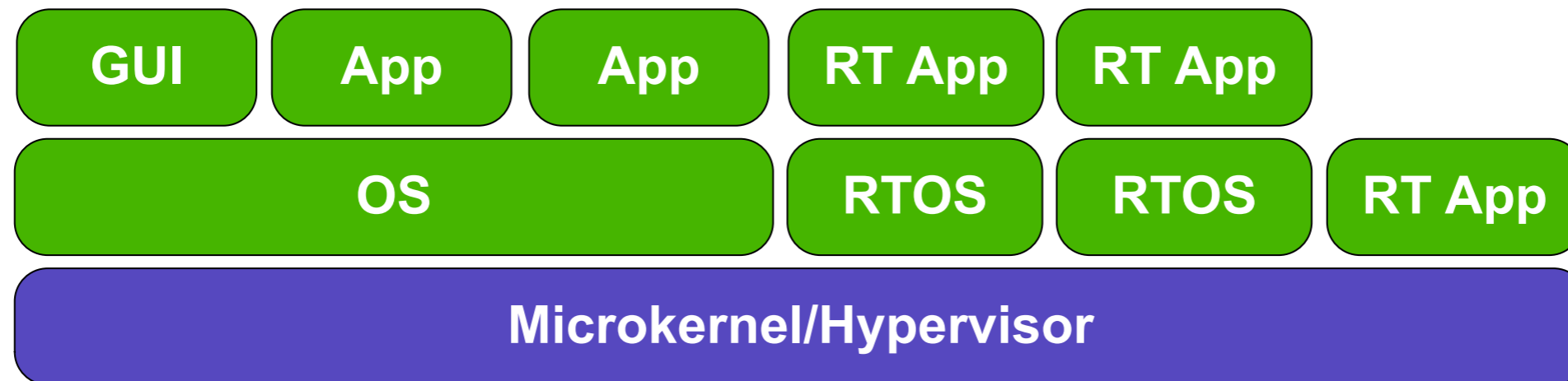
# Hard real-time, the old way

- Dedicated CPUs for real-time-critical tasks.
  - Explicit hardware isolation



# Hard real-time, consolidated!

- Merge subsystems onto one CPU
  - As suggested by [Mehnert et al, RTSS'02], and many others



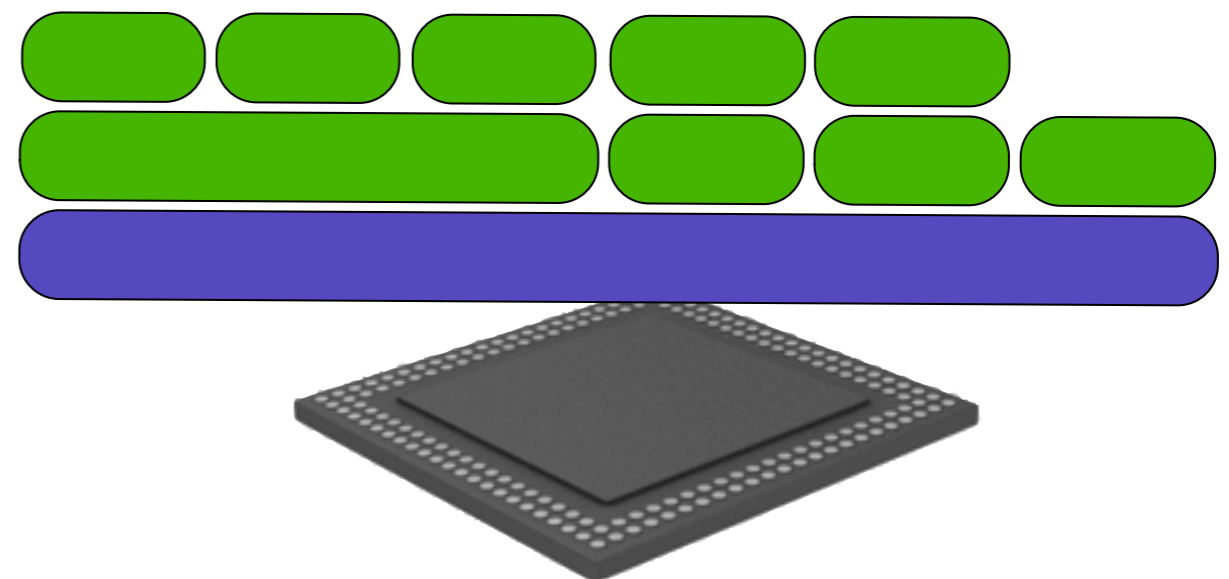
# Hard real-time, consolidated!

- Advantages:

- Reduced hardware complexity and cost
- Simpler communication between components
- Easier to debug a single entity
- Simpler firmware updates

- Disadvantages:

- Single point of failure (the trusted OS kernel)
- Less predictable timing behaviour



# Mitigating the issues

- seL4 microkernel gives *trustworthiness* using
  - MMU-based isolation
  - Small trusted code base (TCB)
  - Formal specification of functional behaviour
  - Machine-checked proof of compliance to specification  
[Klein et al, SOSp'09]
- For hard real-time systems we also want predictable temporal behaviour, i.e.,
  - Interrupt latency guarantees
  - Timing guarantees on OS requests

# Mitigating the issues

- seL4 microkernel gives *trustworthiness* using
  - MMU-based isolation
  - Small trusted code base (100k)
  - Formal specification of functional behaviour
  - Machine-checked proof of compliance to specification  
[Klein et al, SOSP'09]
- For hard real-time systems we also want **predictable temporal behaviour, i.e.,**
  - Interrupt latency guarantees
  - Timing guarantees on OS requests

# Mitigating the issues

- seL4 microkernel gives *trustworthiness* using
  - MMU-based isolation
  - Small trusted code base (100k)
  - Formal specification of functional behaviour
  - Machine-checked proof of compliance to specification  
[Klein et al, SOSP'09]
- For hard real-time systems we also want predictable temporal behaviour, i.e.,
  - Interrupt latency guarantees
  - Timing guarantees on OS requests

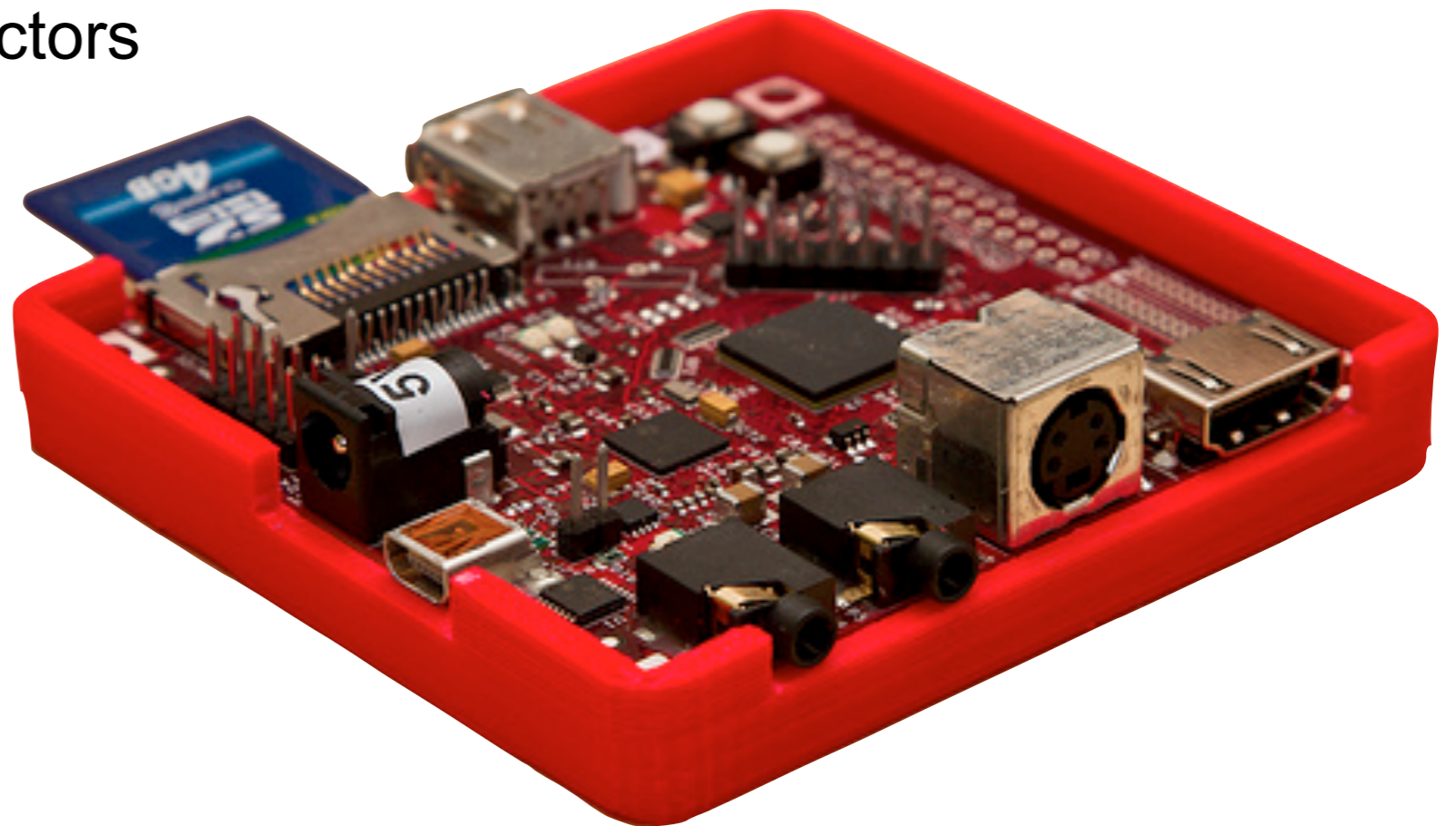
???

# Why is seL4 suited to WCET?

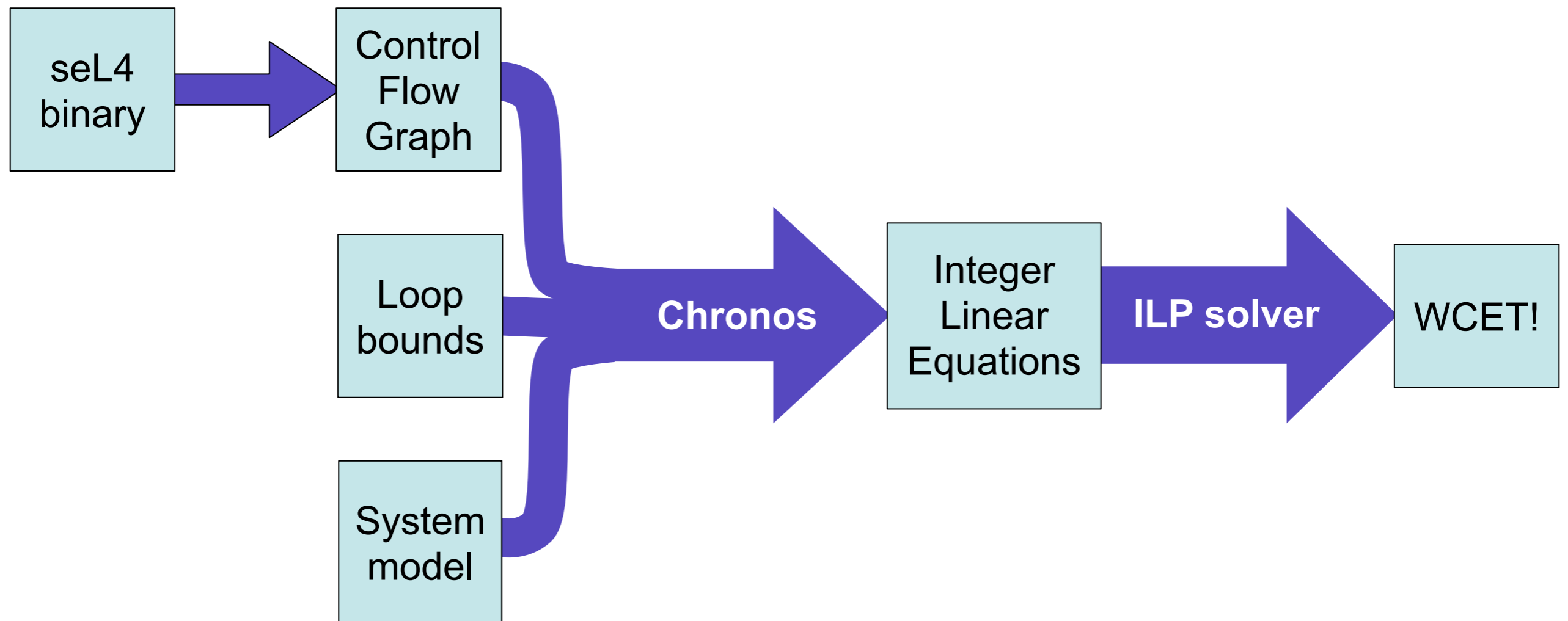
- Small code base (8,700 LoC)
- Event-based design – single kernel stack
- Explicit preemption points
- Well-structured code

# Evaluation Platform

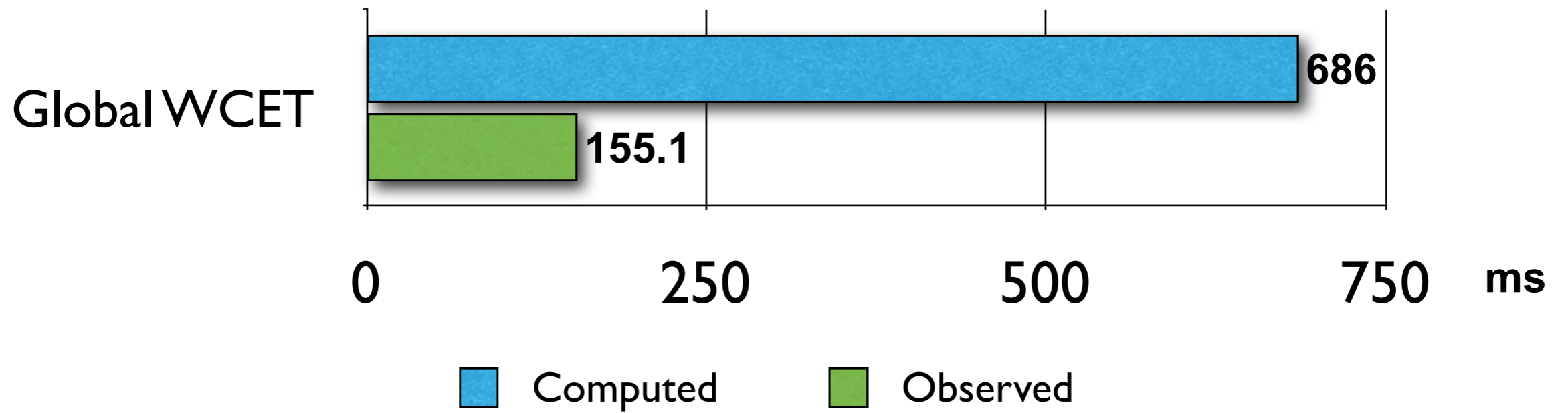
- OMAP3-based BeagleBoard-xM
  - ARM Cortex-A8 @ 800 MHz
  - 128 MB memory
  - 32 KB 4-way set-associative L1 instruction cache
  - Disabled data cache
  - Disabled branch predictors



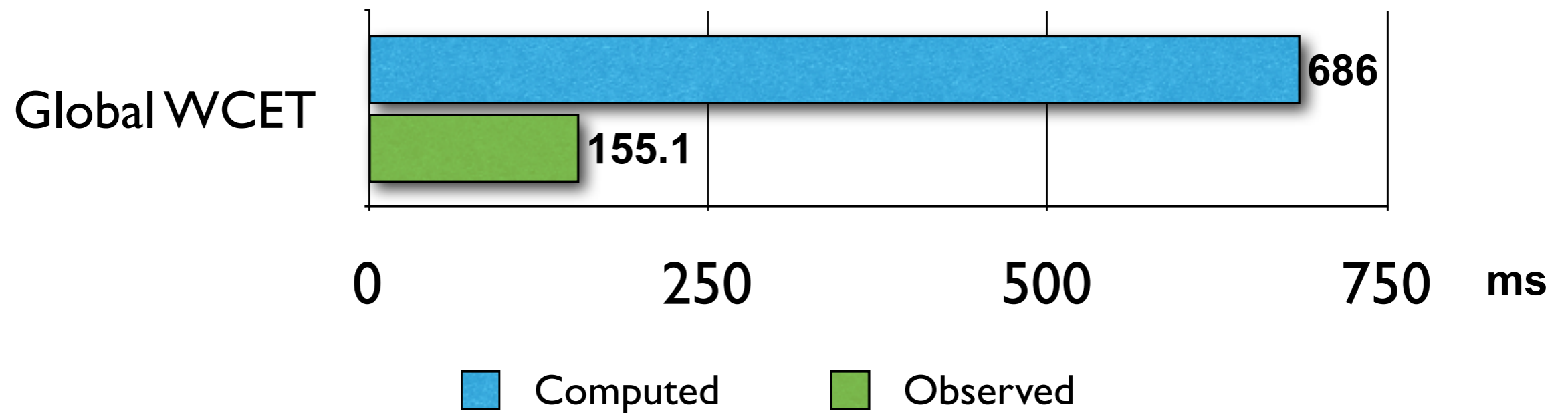
# Analysis method



# Results



# Results

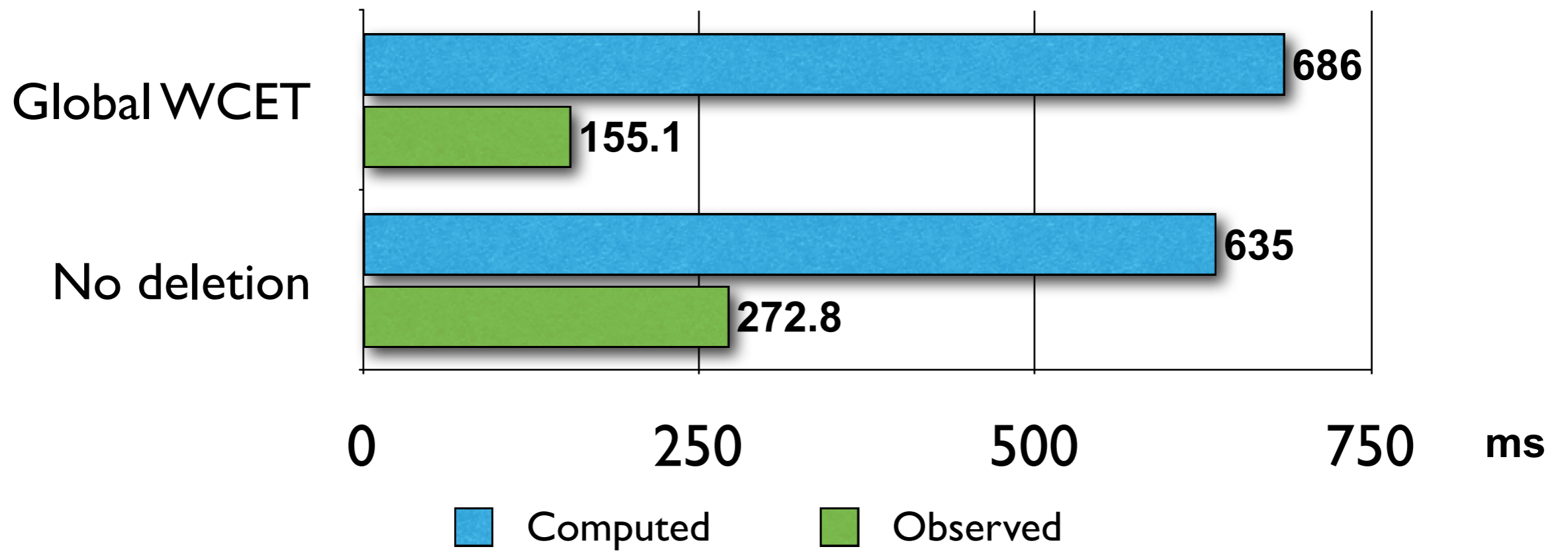


## *Endpoint deletion:*

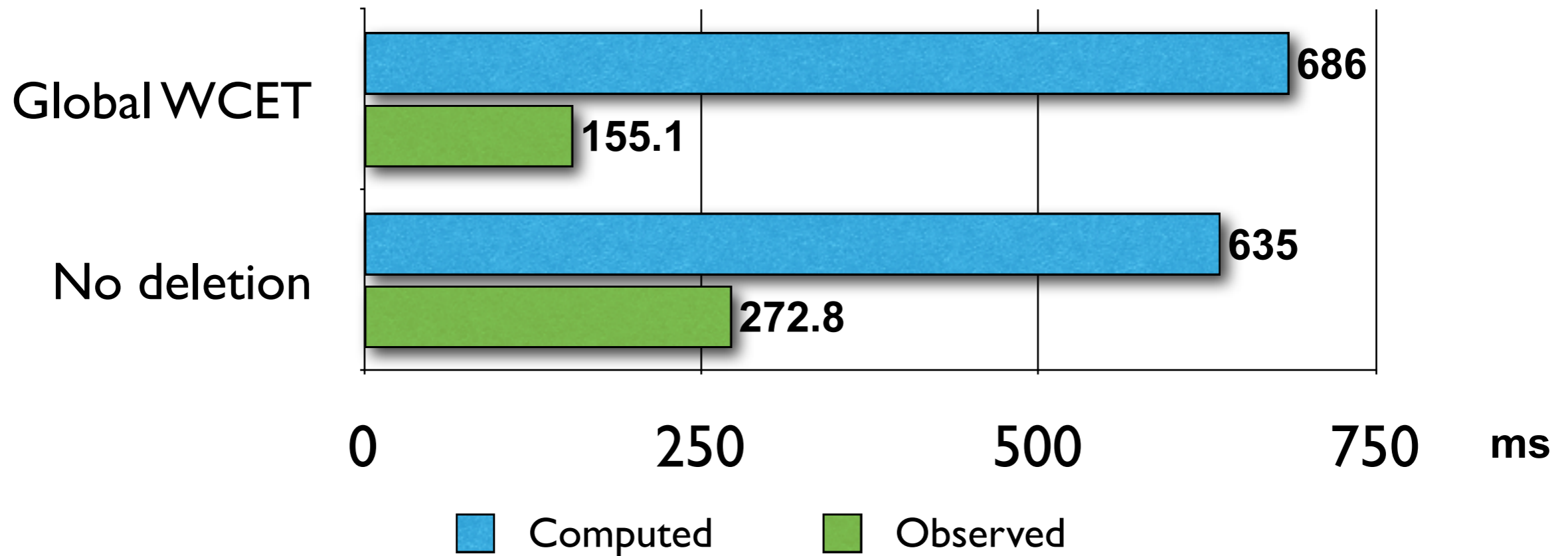
deleting an IPC object with  
many threads waiting

- Malicious entity can easily force this scenario

# Results



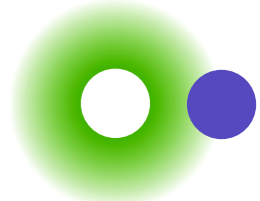
# Results



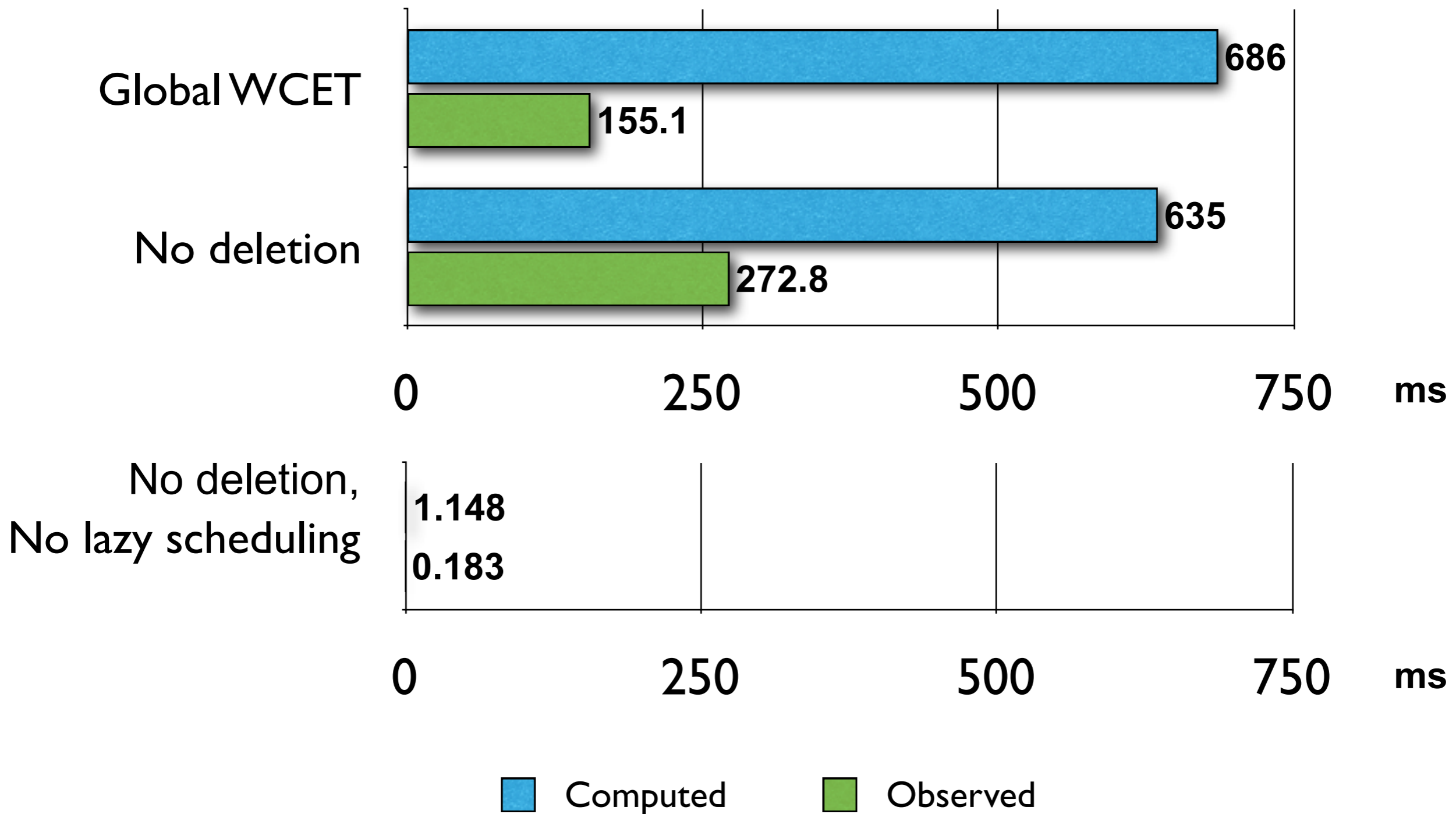
## *Lazy scheduling:*

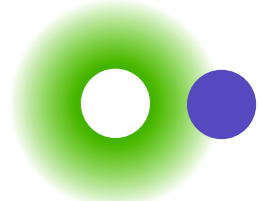
does not eagerly remove blocked threads from the run queue

- Malicious entity may pollute the run queue with blocked threads



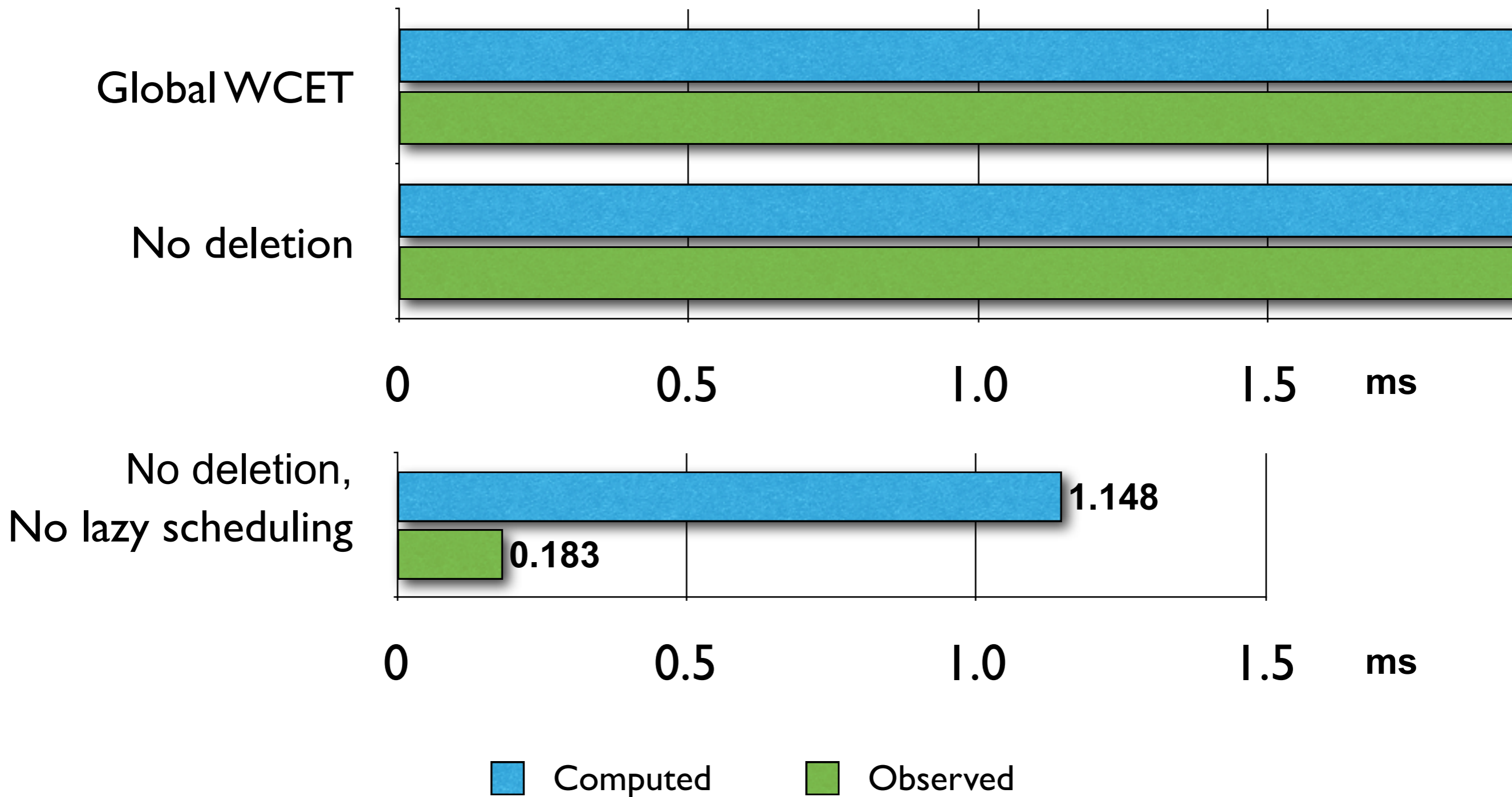
# Results





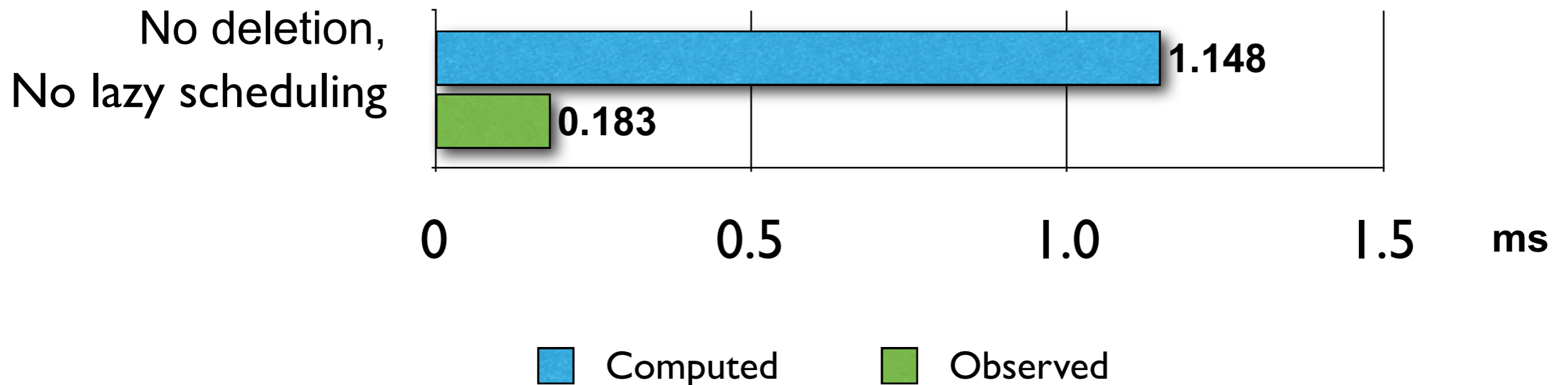
# Results

NICTA



# Results

- Closed system WCET bounded by IPC itself
  - Maximum length message transfer (120 words)
  - Transfer of capability objects



## Future work

- Support for data caches in analysis
- How can these results be improved by leveraging proof invariants?
- How can seL4 be improved for real-time applications?
  - Remove lazy scheduling
  - Fix other thread-dependent operations

# Conclusion

- Protected hard-real time is the way forward!
- It requires a *trustworthy* platform to build on, e.g. seL4
- Real-time guarantees are easier to compute with the right kernel design