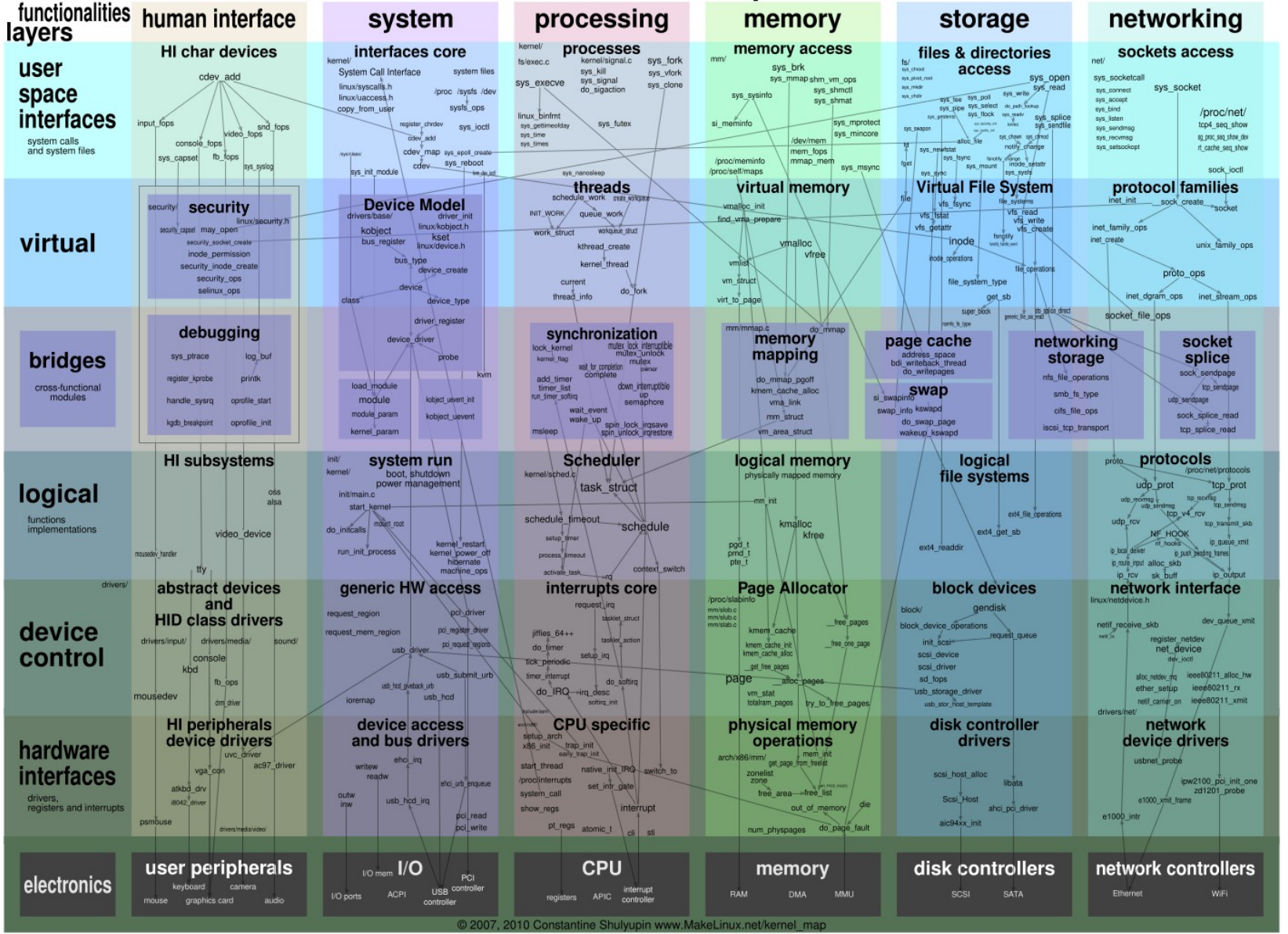
A Policy-free Kernel for Scalable Multi-core, Bounded Execution, and Verification

Gabe Parmer
Based on work by a village of strong researchers

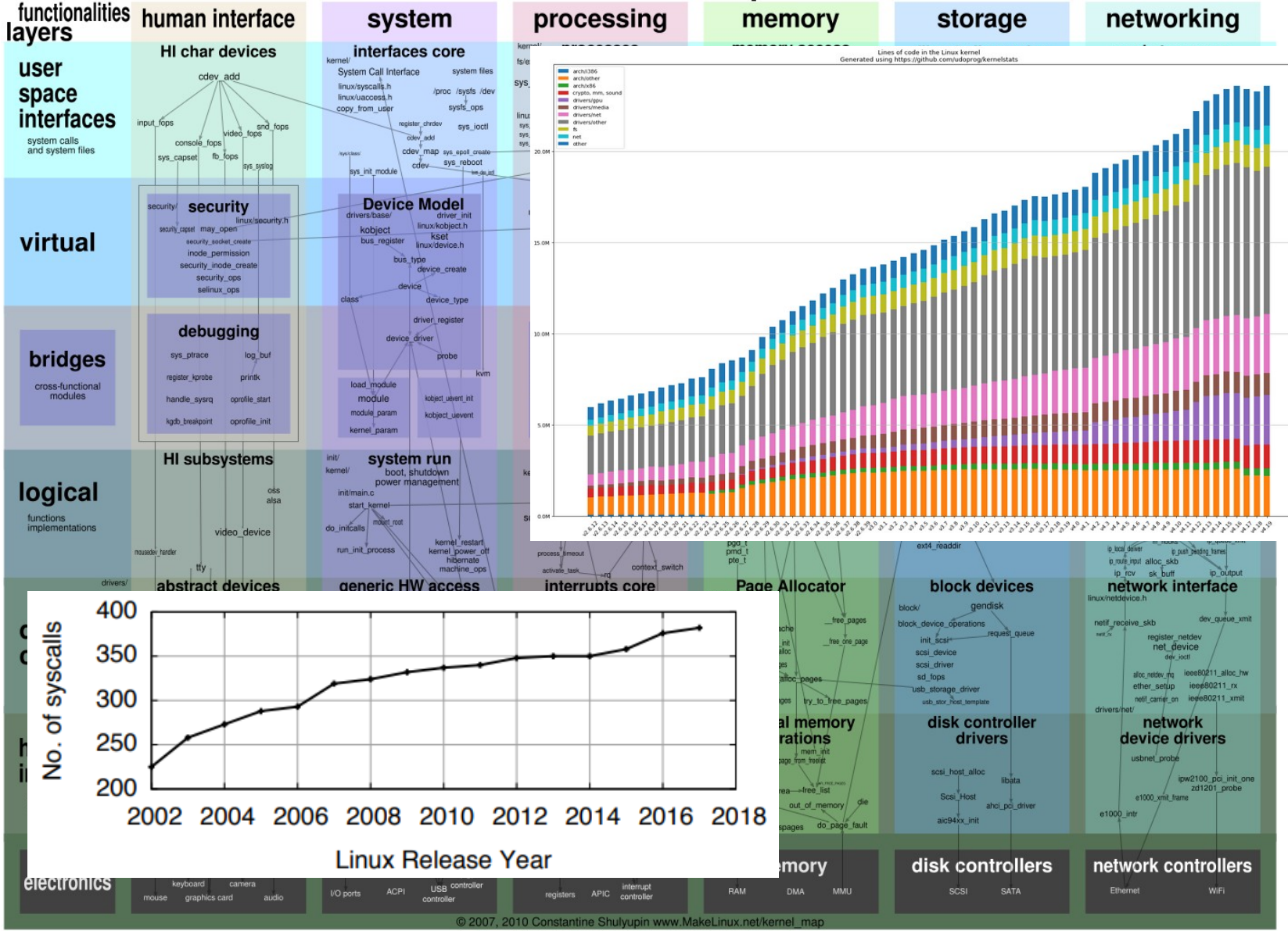
System Design to Control Interference

Gabe Parmer
Based on work by a village of strong researchers

POSIX / Monolithic OSes

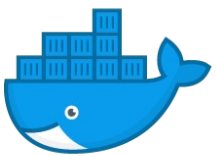


POSIX / Monolithic OSes



Choose your trade-off

ARINC 653



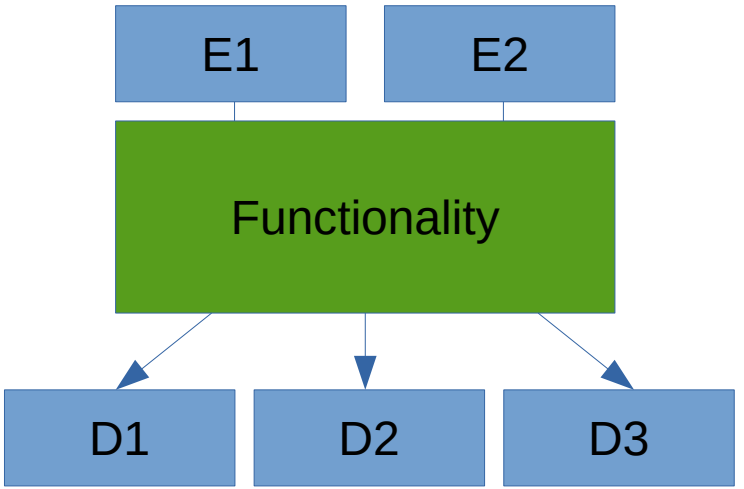
High Assurance
Predictable
Simple

Lower Assurance
Tenuous predictability
Complex

Goal:
Functionality + Dependability + Productivity

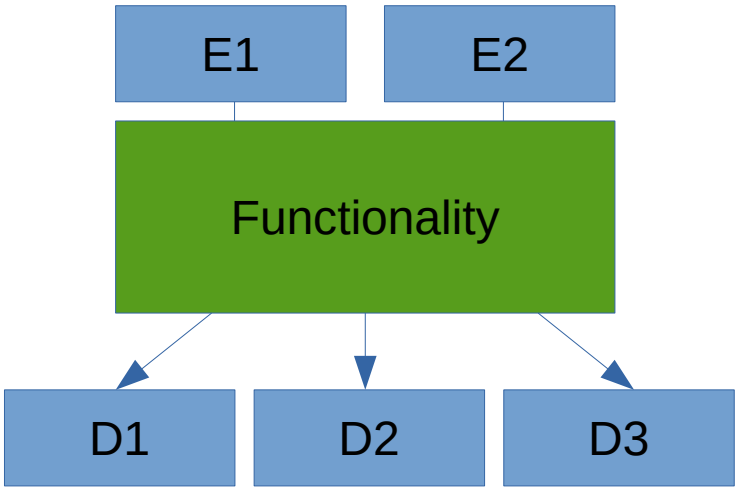
Component-based system design

- Code, data
- Export APIs (E1 = {`fn, ...`})
- Explicit dependencies (D1, ...)
- Unit of reuse & isolation



Goal:
Functionality + Dependability + Productivity

Component-based system design

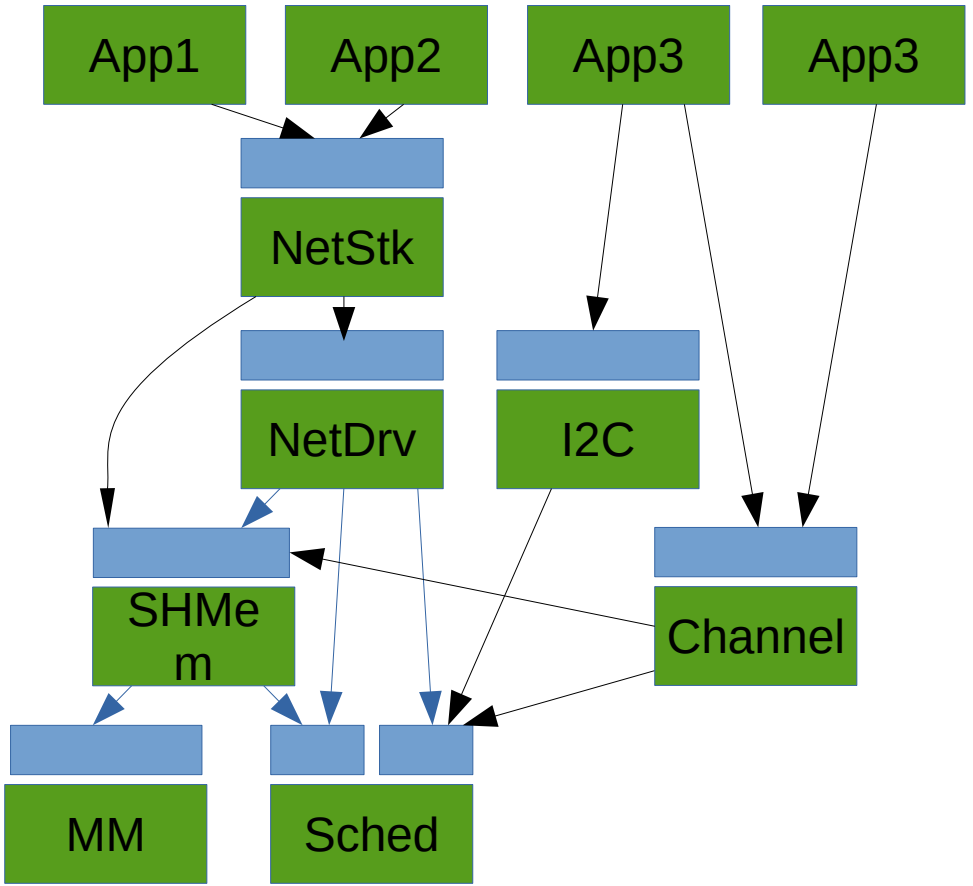


- Code, data
- Export APIs (E1 = `{fn, ...}`)
- Explicit dependencies (D1, ...)
- Unit of reuse & isolation
- *Minimize* functionality for the *necessary APIs*
- Strong, *fine-grained isolation*
- LCM & PoLP

Goal:

Functionality + Dependability + Productivity

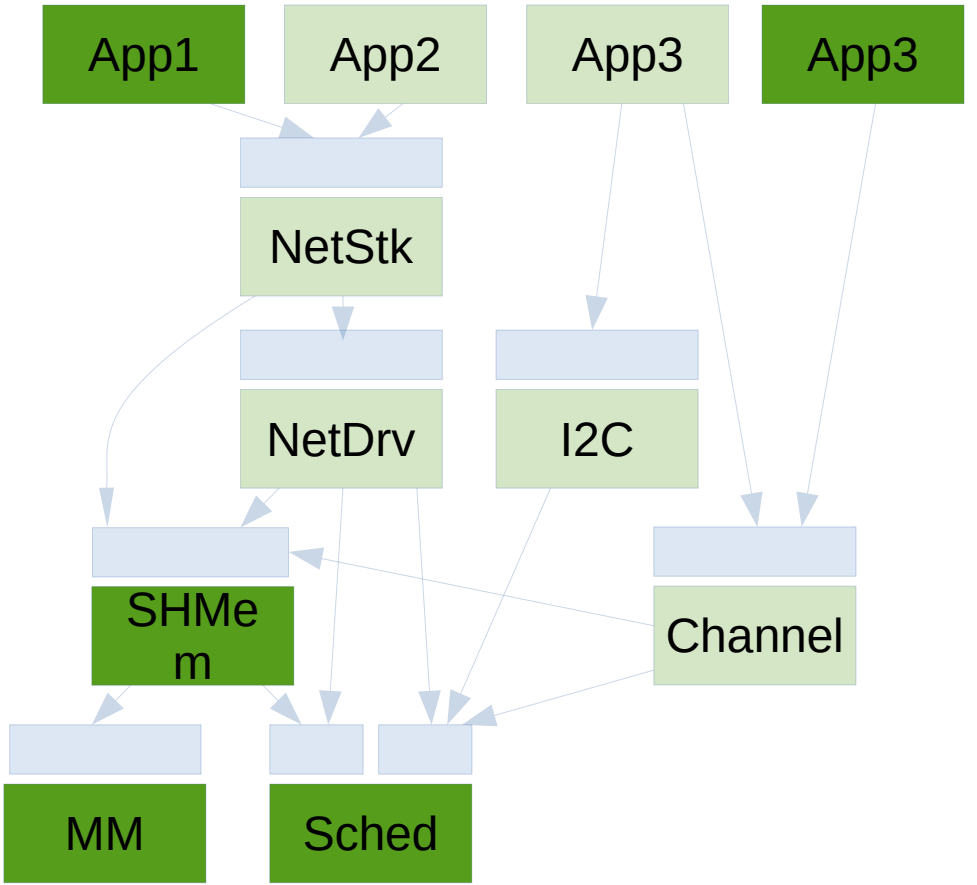
- Components *compose* system
- Limit *scope* of
  - compromise
  - fault
  - unpredictability



Goal:

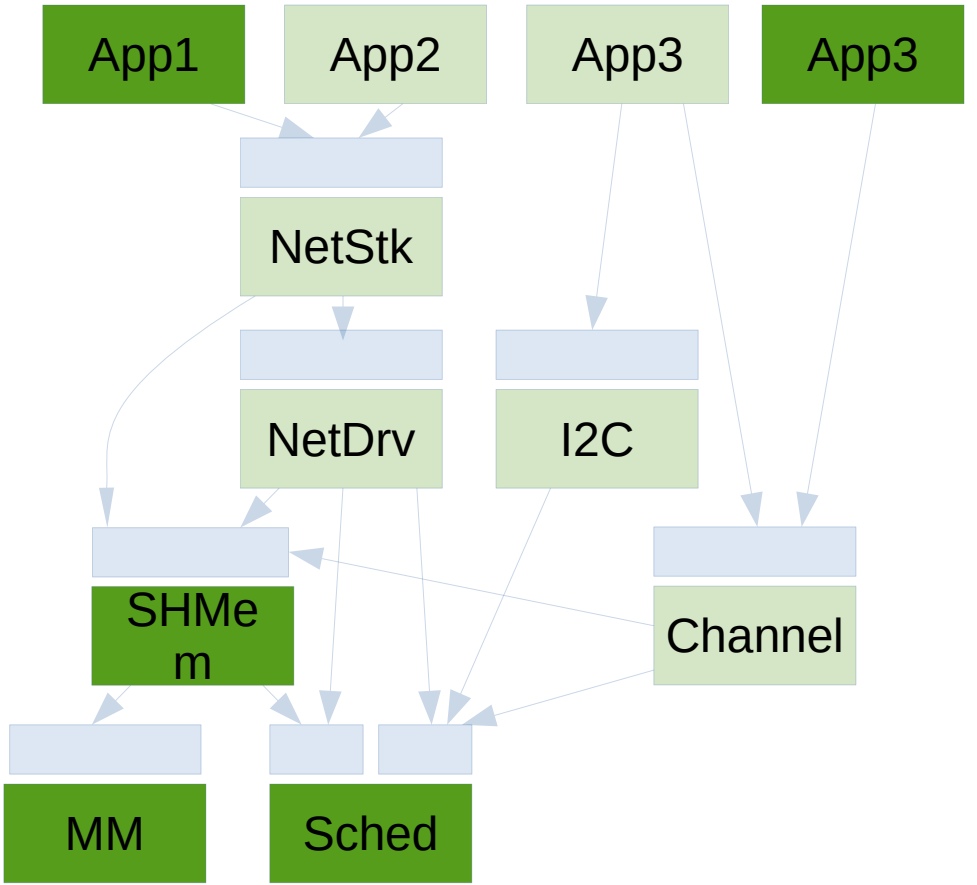
Functionality + Dependability + Productivity

- Components *compose* system
- Limit *scope* of
  - compromise
  - fault
  - unpredictability



Goal:
Functionality + Dependability + Productivity

- Non-functional challenges
  - Shared resource interf.
  - Timing channels
  - …



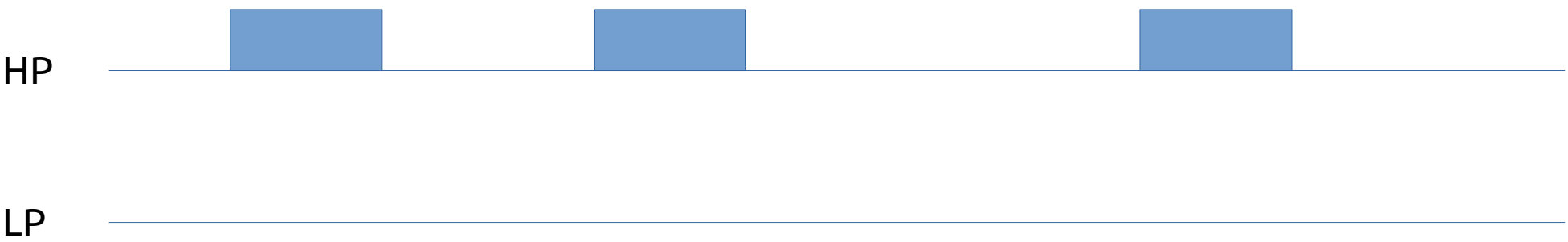
Composite: OS for System Composition

- Microkernel: small kernel, fast communication
  - User-level policies for resource mgmt
  - Page-table memory isolation between components
- Unique features – *get out of the way*!
  - User-level scheduling → configurable policy
  - Wait-free kernel → predictably scalability
  - Integration with push-button verification

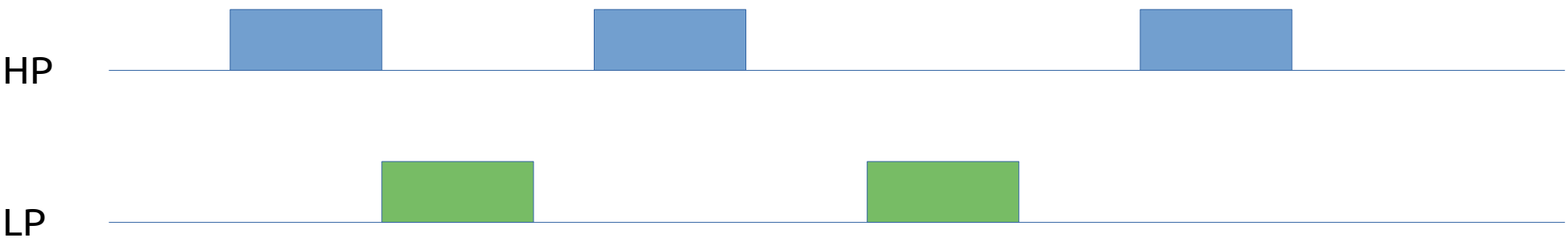
OS Building Toolkit

- Chaos – managed interference for mixed-criticality systems
- EdgeOS – *per-client*, multi-tenant isolation for edge processing
- Microsecond-scale multi-tenant infrastructure
- Multi-tenant host network functions
  - VM execution with NF interposition faster than bare-metal!

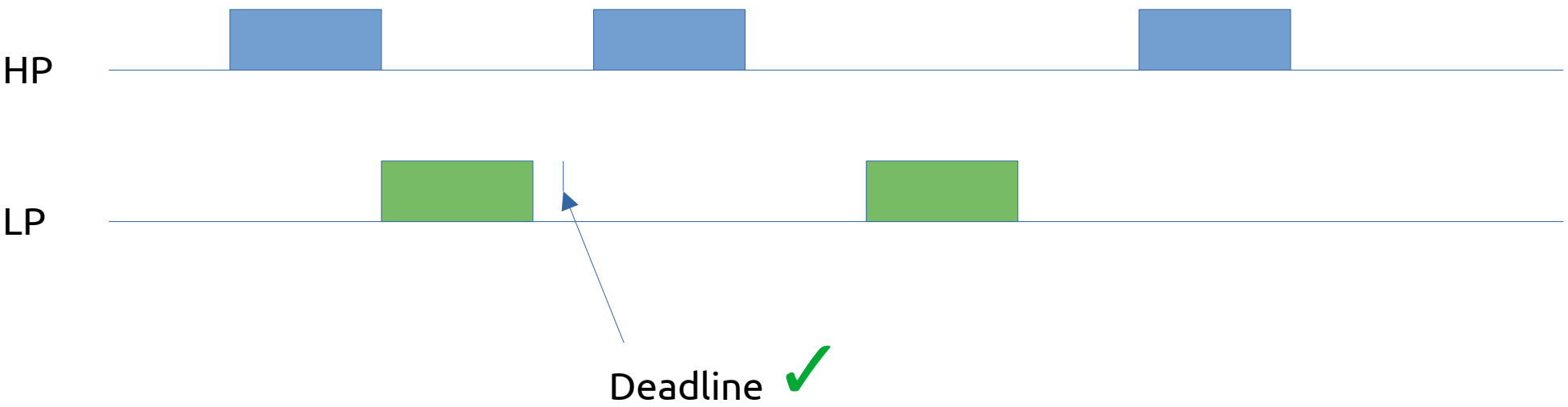
Good intentions: Reservations to prevent processing interference



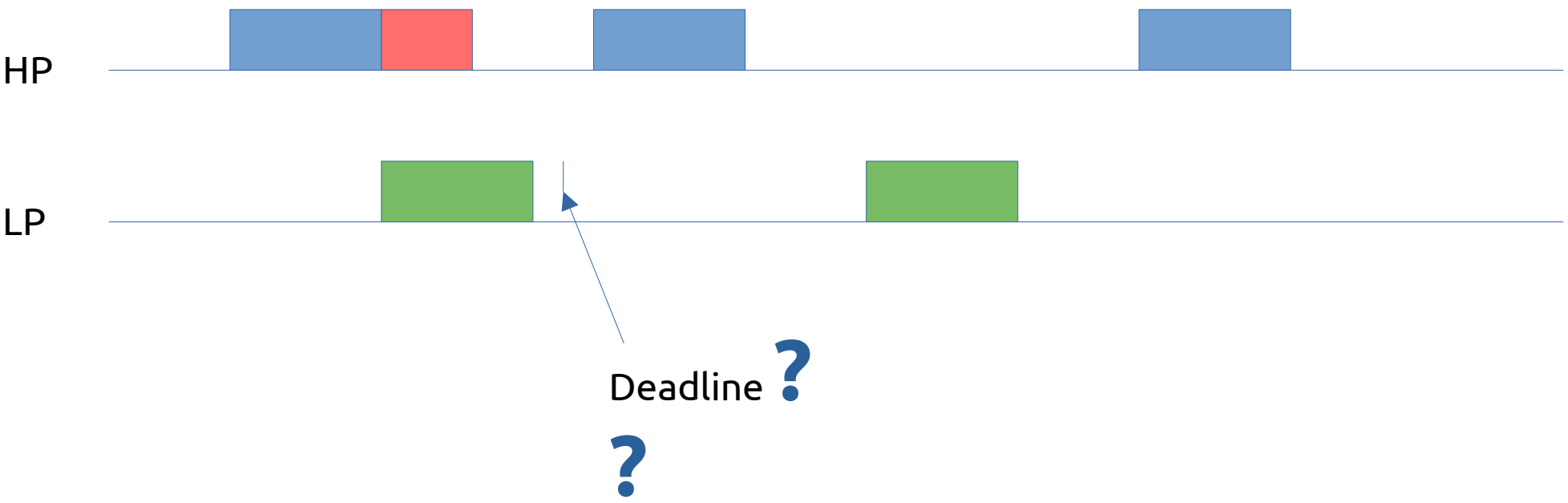
Good intentions: Reservations to prevent processing interference



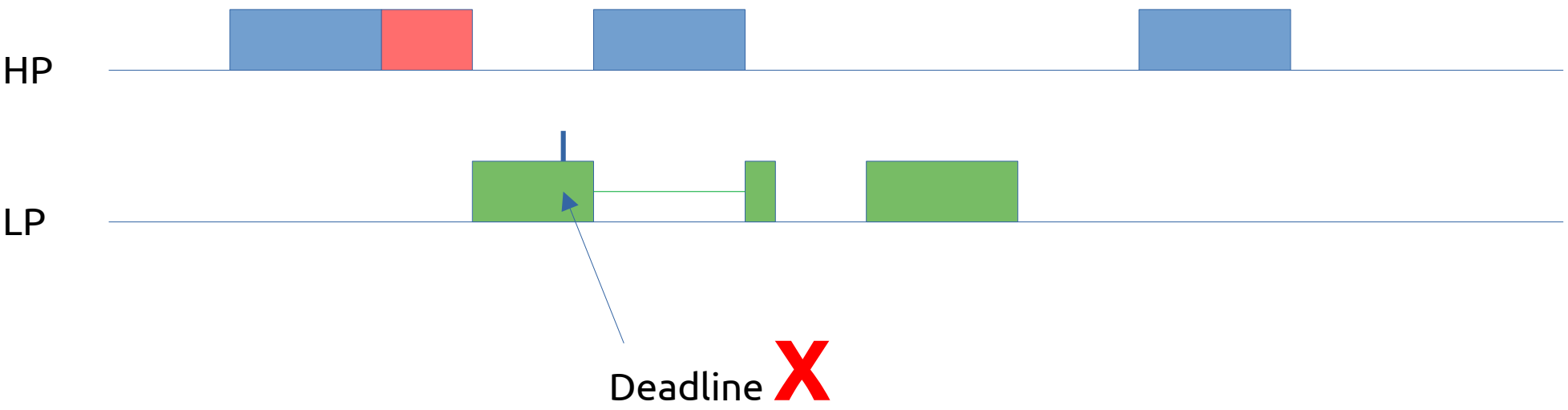
Good intentions: Reservations to prevent processing interference



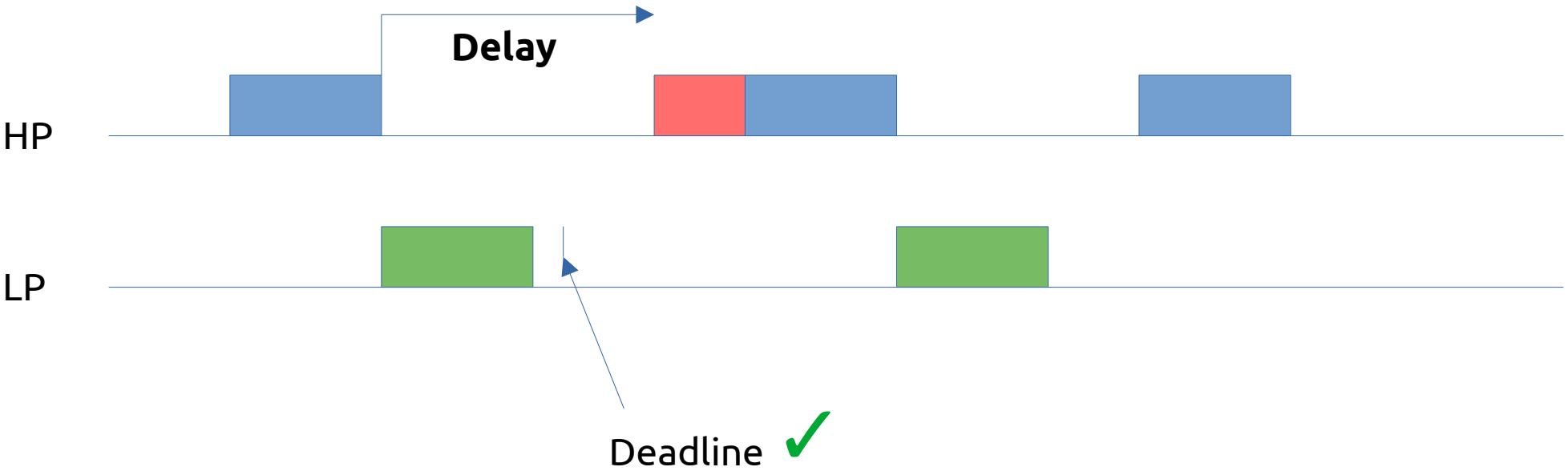
Good intentions: Reservations to prevent processing interference



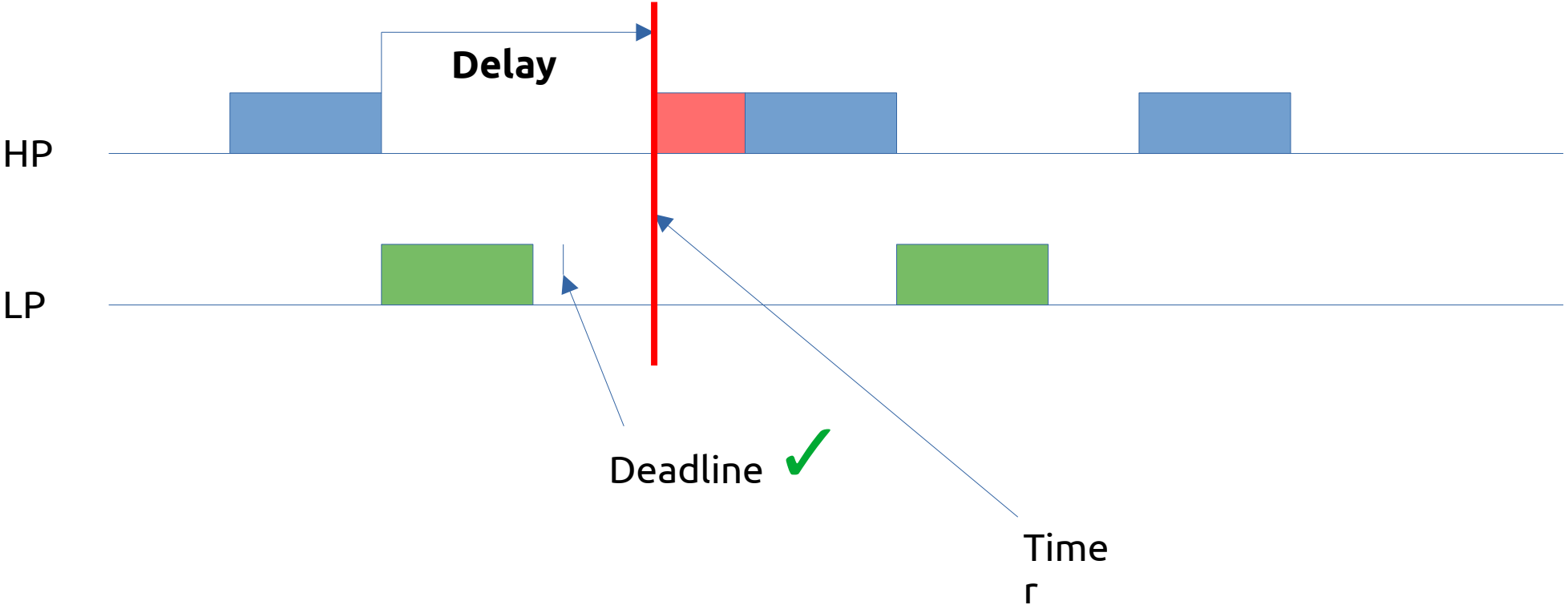
Good intentions: Reservations to prevent processing interference



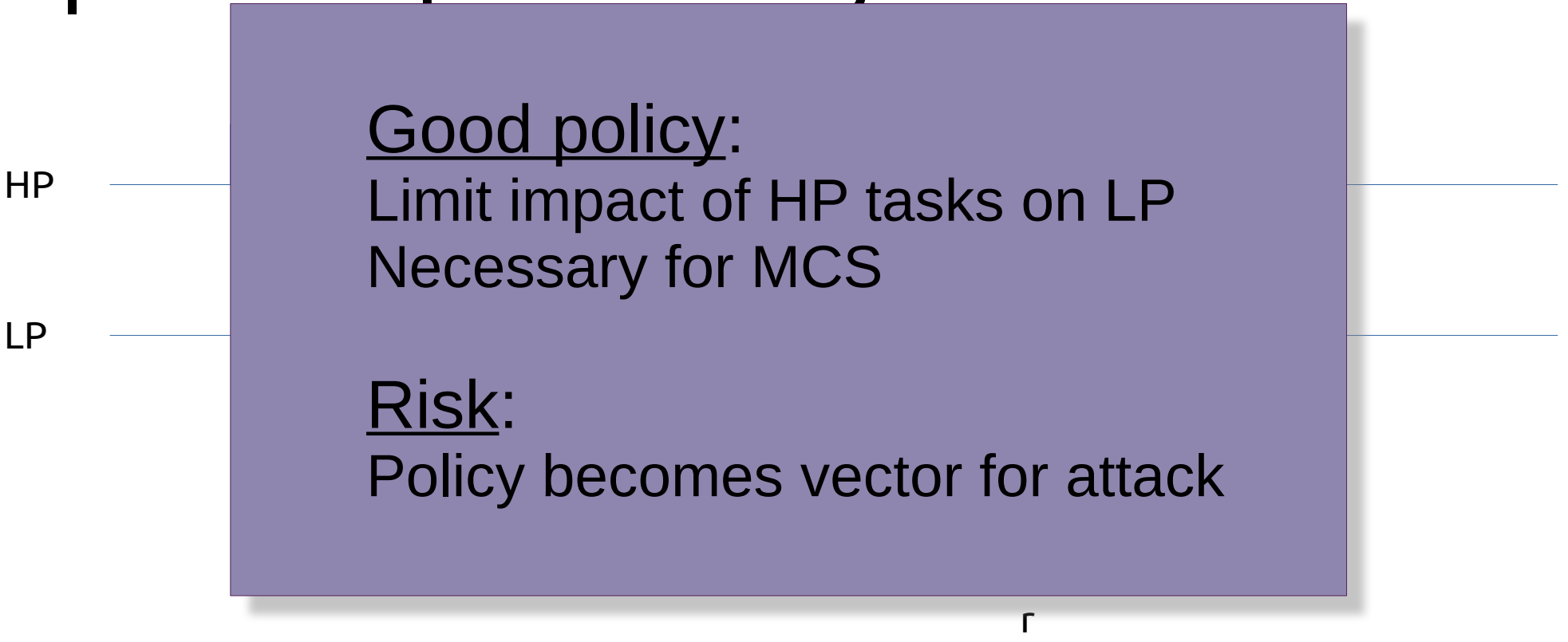
Good intentions: Reservations to prevent processing interference

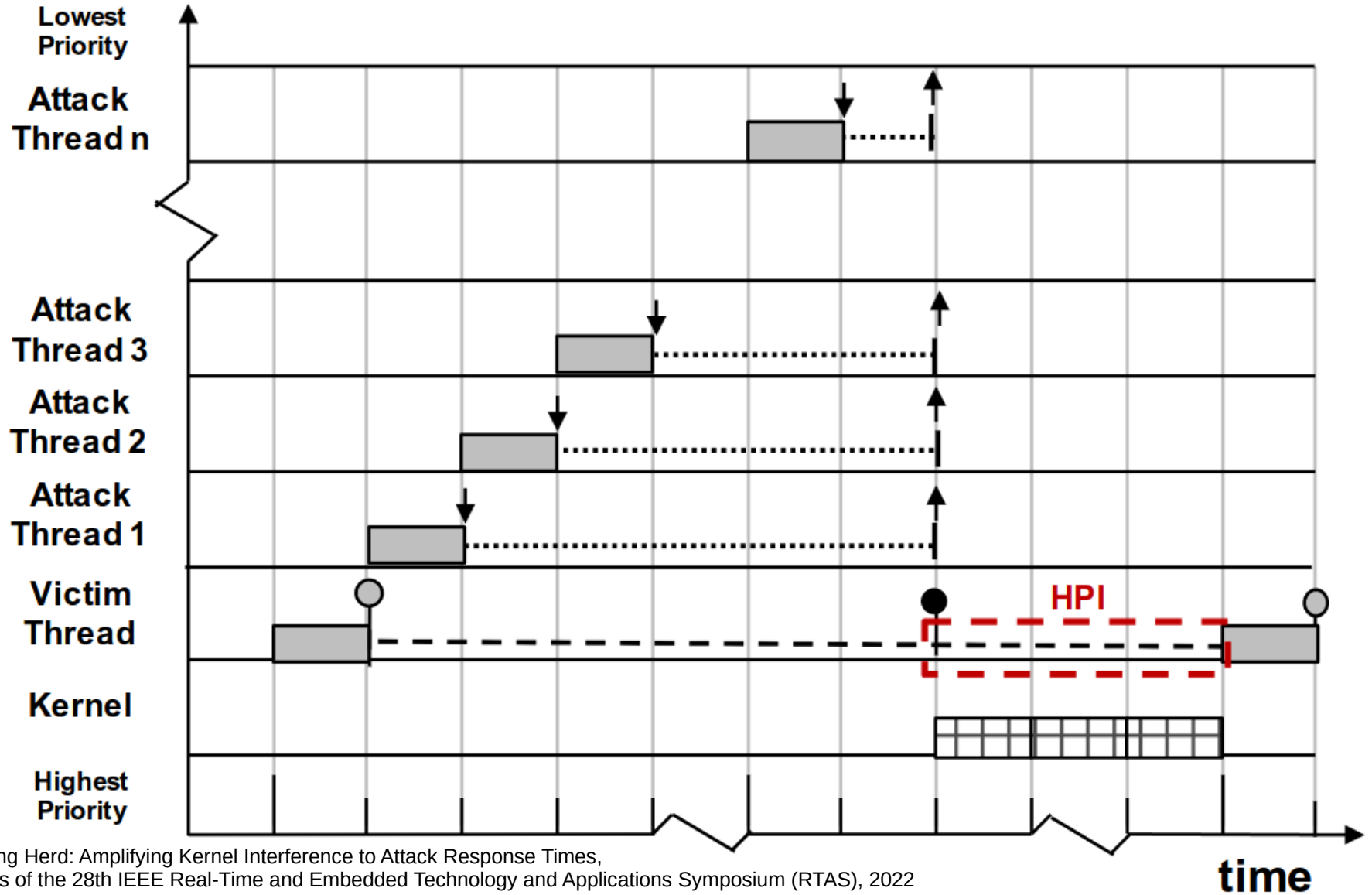


Good intentions: Reservations to prevent processing interference

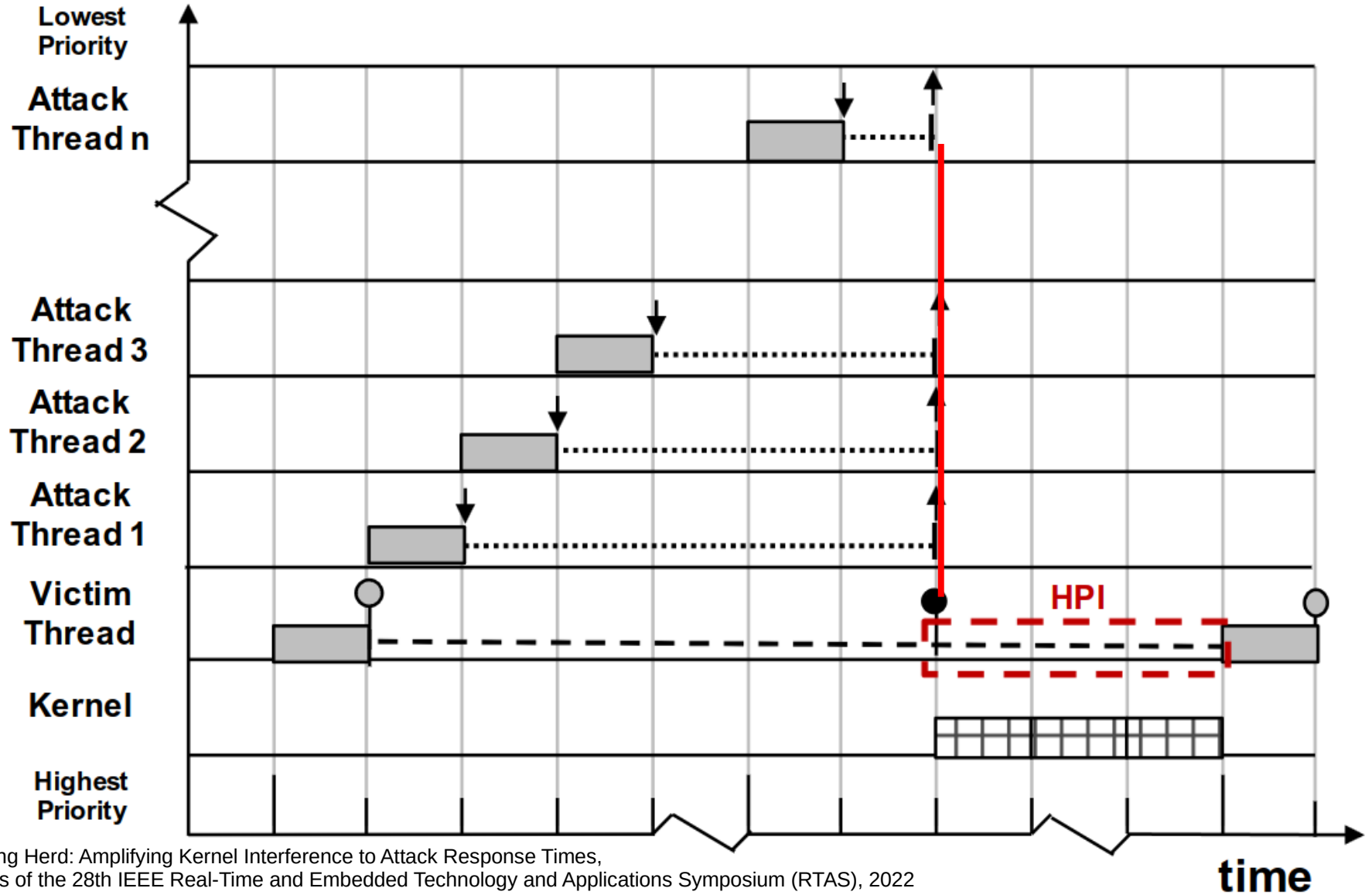


Good intentions: Reservations to prevent processing interference

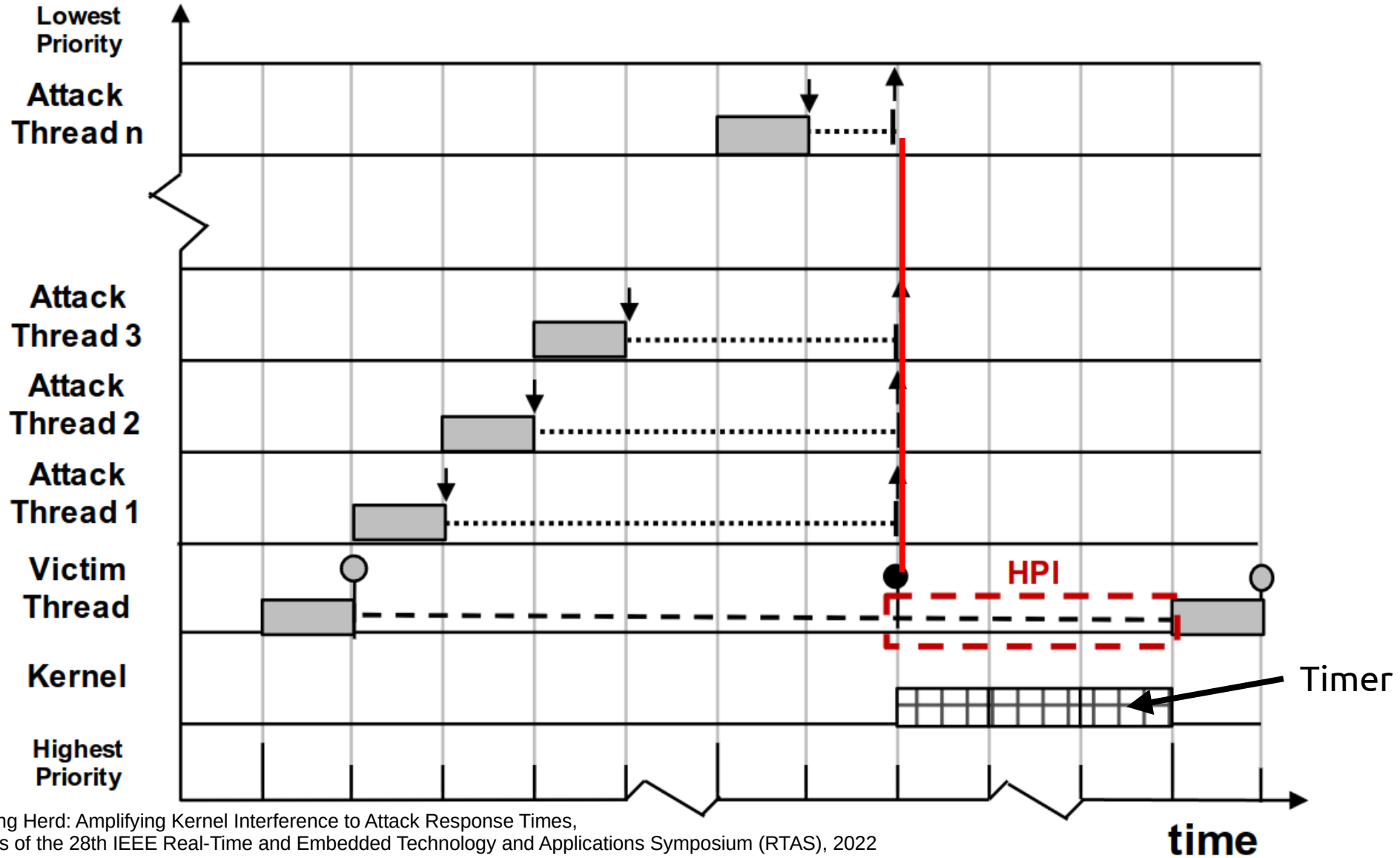




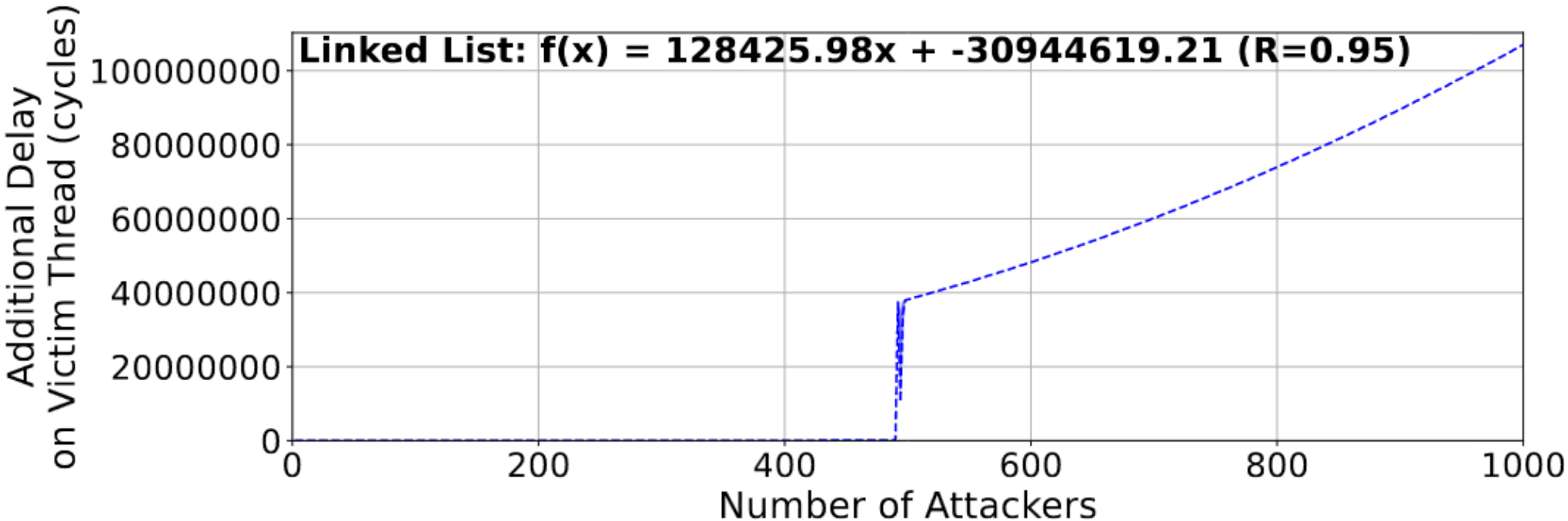
The Thundering Herd: Amplifying Kernel Interference to Attack Response Times,
in Proceedings of the 28th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2022



The Thundering Herd: Amplifying Kernel Interference to Attack Response Times,
in Proceedings of the 28th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2022



The Thundering Herd: Amplifying Kernel Interference to Attack Response Times,
in Proceedings of the 28th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2022



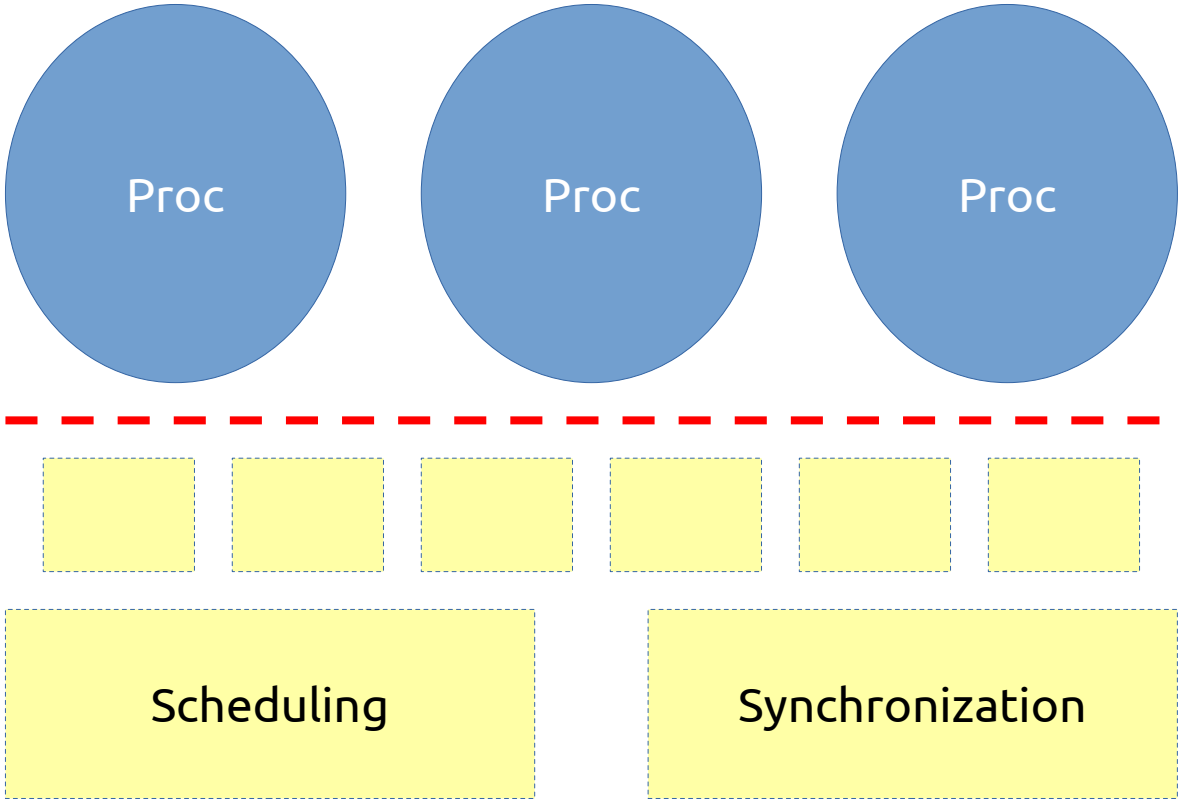
The Thundering Herd: Amplifying Kernel Interference to Attack Response Times,
in Proceedings of the 28th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2022

Policy Freedom

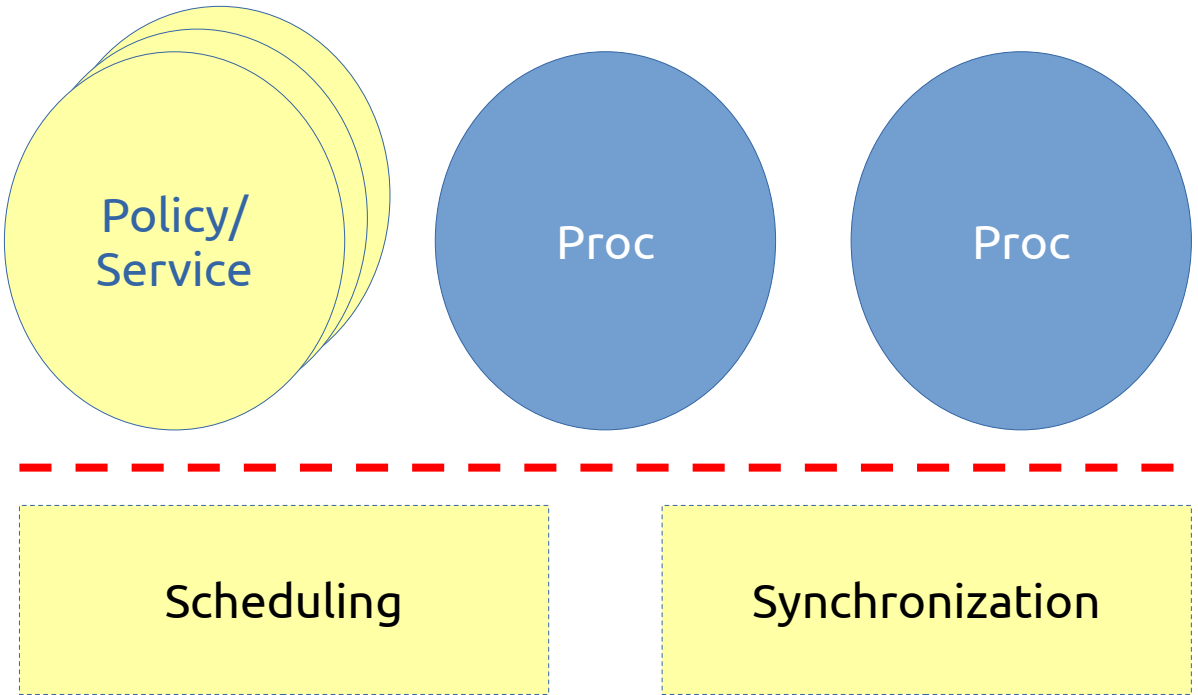
- Kernel policies are appealing
  - Simple
  - Centralized
- Policies will be attacked, when static
  - Policy impl. w/ interrupts disabled

    ...challenging

Scheduling & Synchronization

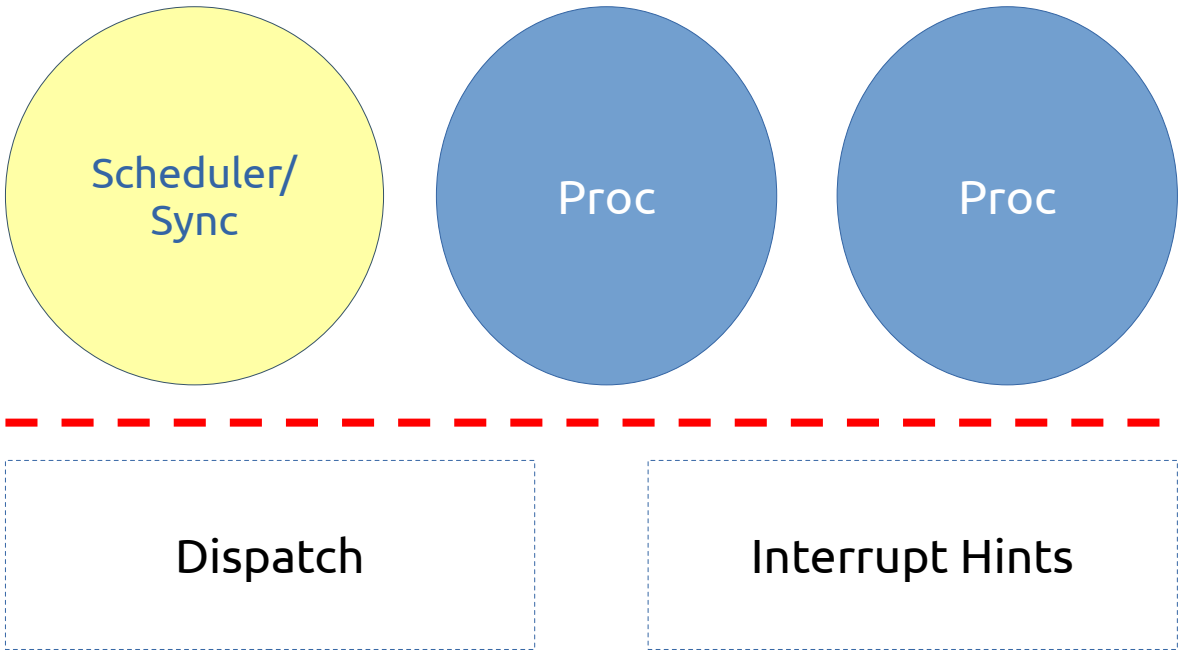


Scheduling & Synchronization

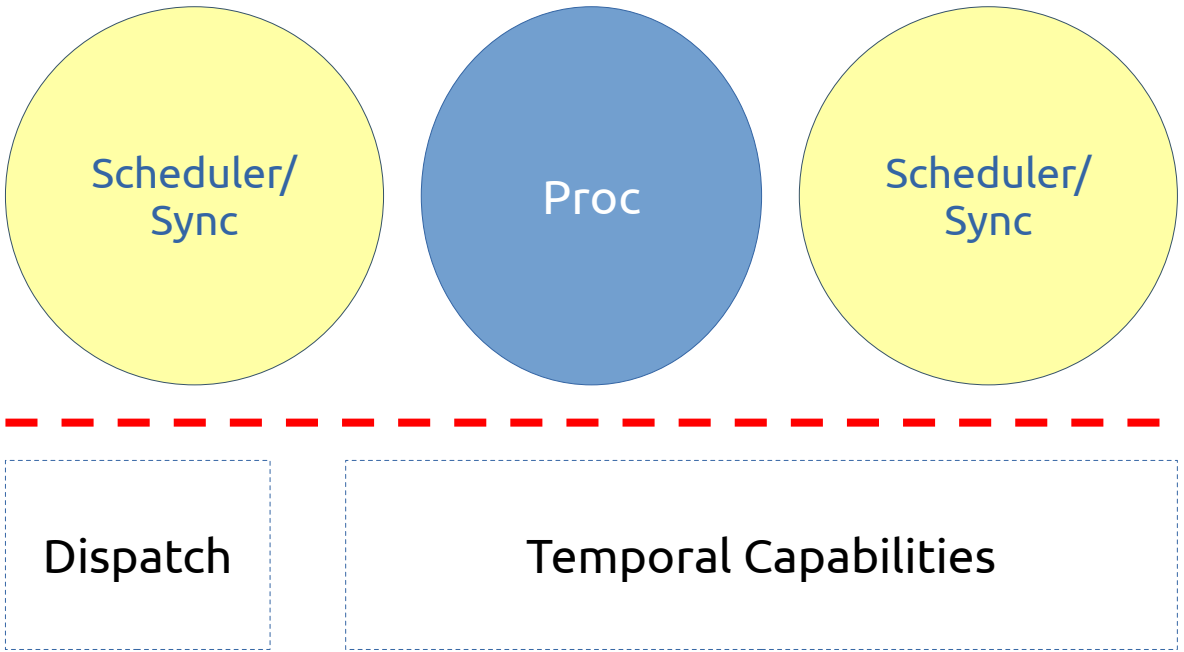


Scheduling & Synchronization

- Slite: OS Support for Near Zero-Cost, Configurable Scheduling, in Proceedings of the 26th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2020
- HiRes: a System for Predictable Hierarchical Resource Management, to appear in Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2011), Chicago, IL, April 2011
- "Predictable Interrupt Management and Scheduling in the Composite Component-based System", in Proceedings of the 29th IEEE Real-Time Systems Symposium (RTSS), Barcelona, Spain, 1-3 December 2008



Scheduling & Synchronization



Temporal Capabilities: Access Control for Time, in the Proceedings of the 38th IEEE Real-Time Systems Symposium (RTSS), 2017

Scheduling & Synchronization

Kernel:

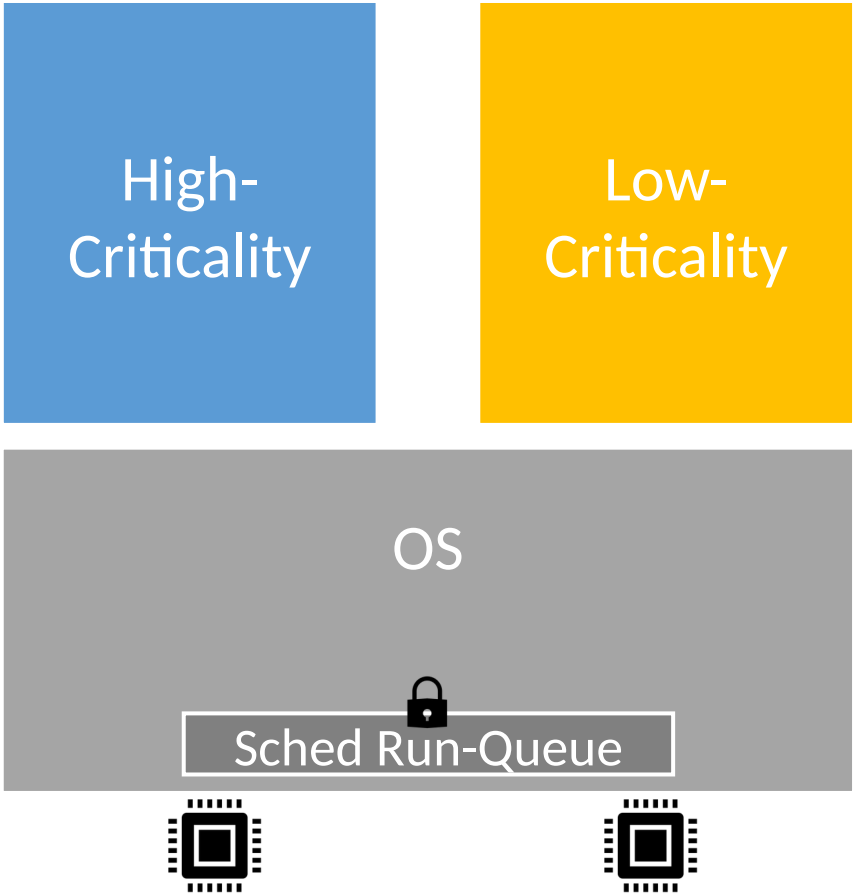
- No unbounded loops
- Actions: directly from syscall instructions
  - Capability-checked

Scheduler:

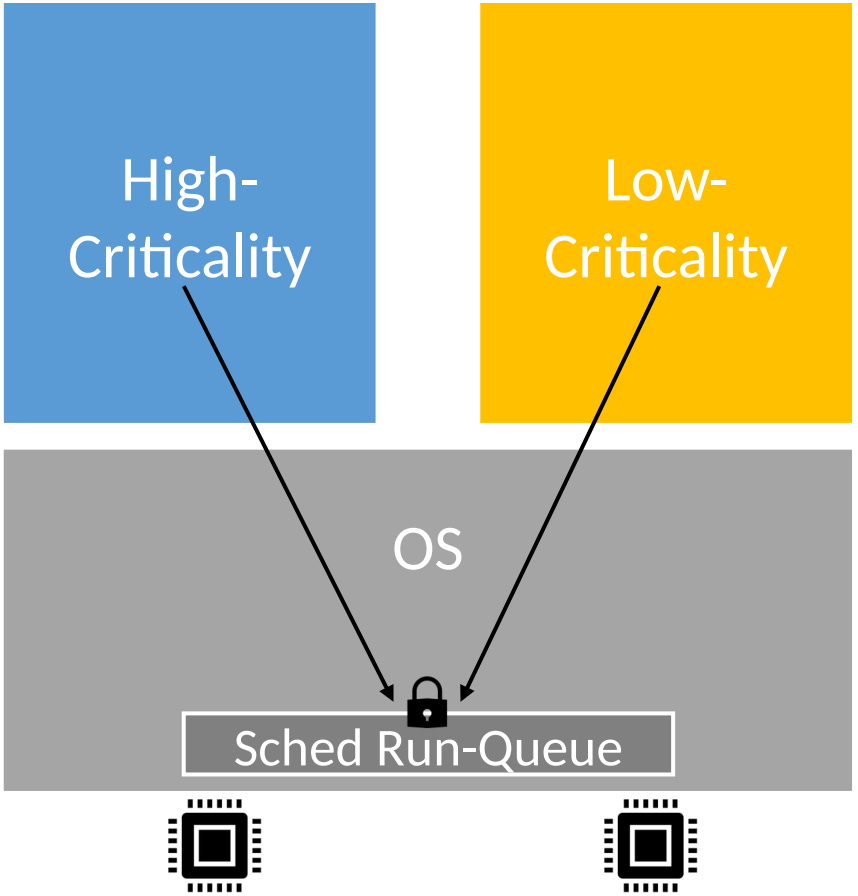
- Any policy – incl. emulating L4 IPC!
- Current work: constant-time *everything*

SPeCK: A Kernel for Scalable Predictability, in Proceedings of the 21st IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015

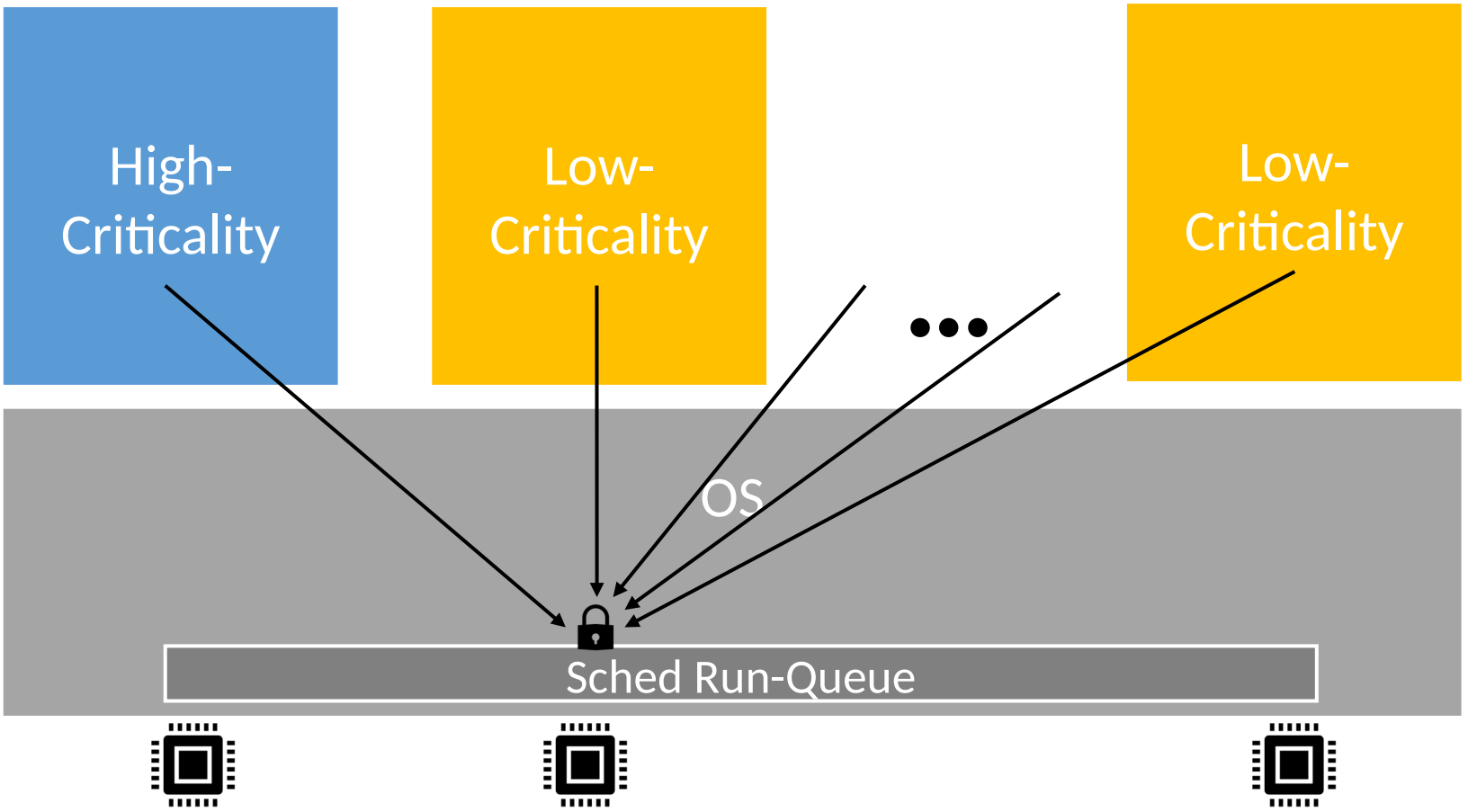
Multi-Core: **Shared-Memory** Impact



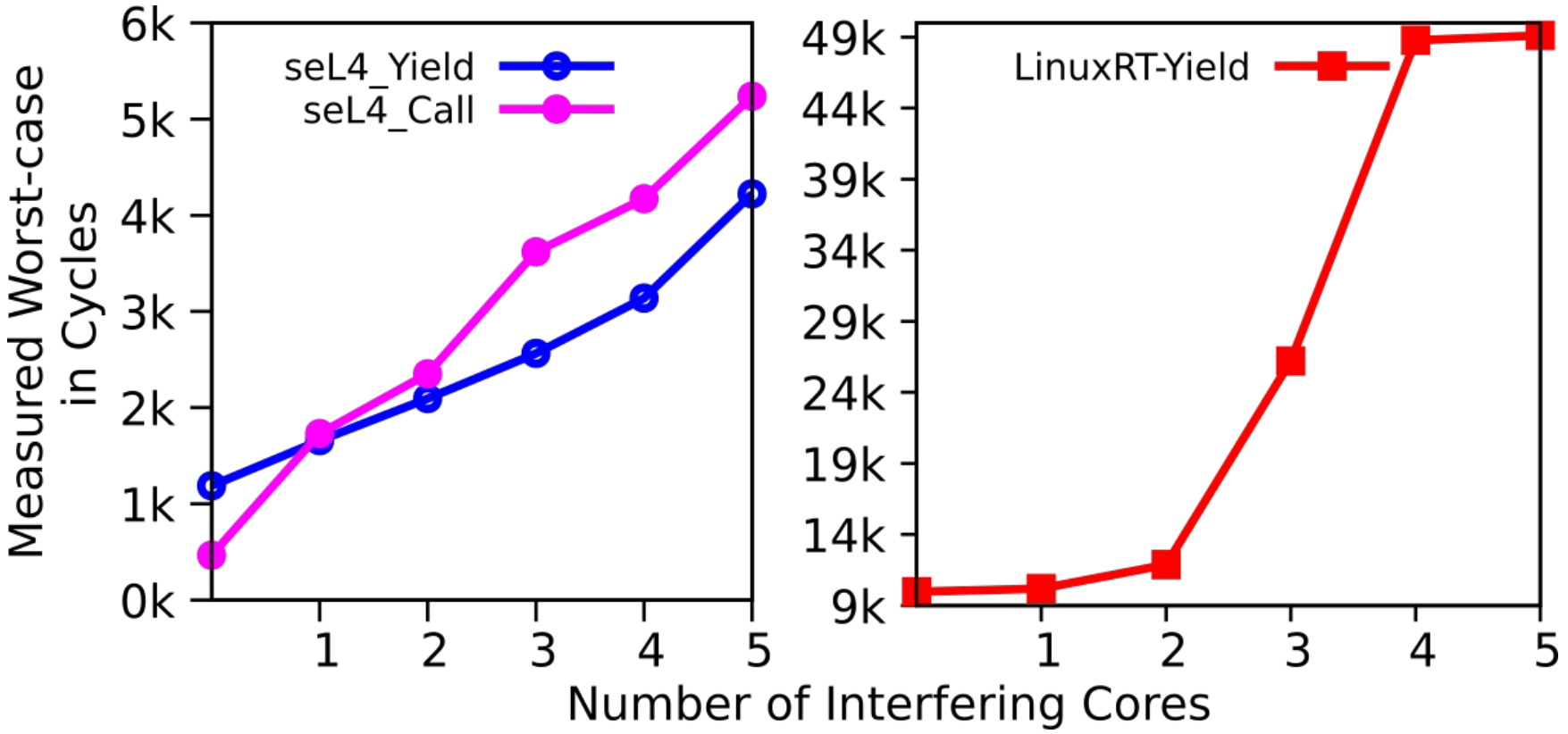
Multi-Core: **Shared-Memory** Impact



Multi-Core: **Shared-Memory** Impact



Impact of Shared-Memory



Chaos: a System for Criticality-Aware, Multi-core Coordination, in Proceedings of the 25th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2019

Composite µ-kernel

- Wait-free kernel data-structures
- No locks: all synchronization:
  - Atomic instructions
  - Without loops
  - Scalable memory reclamation using *time*
- Bounded execution!

→ parallel kernel with *no locks!*

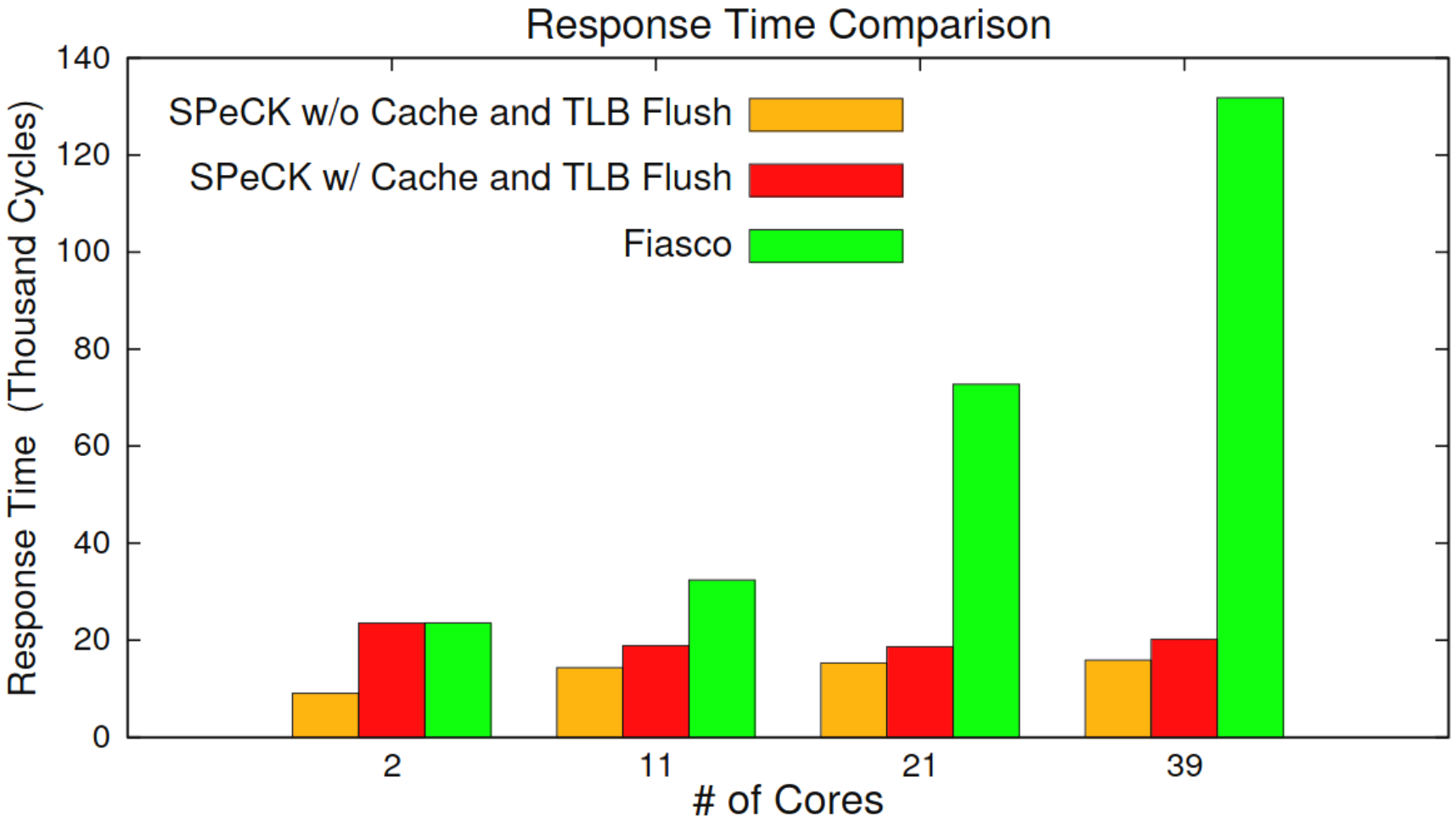
Composite µ-kernel

- Wait-f
- No lo
  - Ato
  - Wit
  - Scal
- Bounded execution!

Scalable Predictability:

Syscall worst-case on 1 core = *E*
Syscall worst-case on N cores = *cE*

→ parallel kernel with *no locks!*



SPeCK: A Kernel for Scalable Predictability, in Proceedings of the 21st IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015

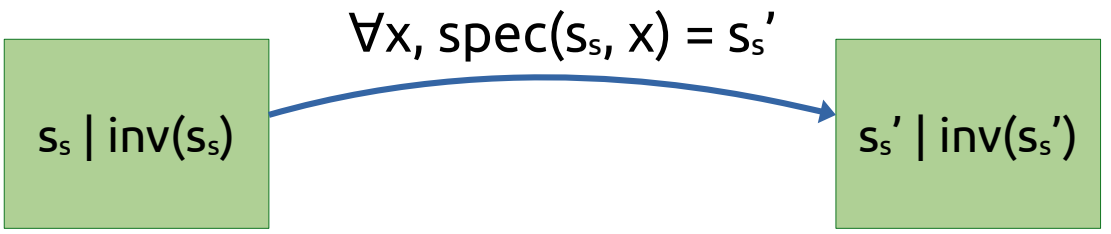
Real-time + Scalable Predictability

- Bounded execution
  - Constant-bounded loops
  - Bounded execution time
  - Bounded *path complexity*
- Wait-free
  - Syncronization: faa, non-repeating cas
  - Scalable memory reclamation
  - Bounded *path complexity*

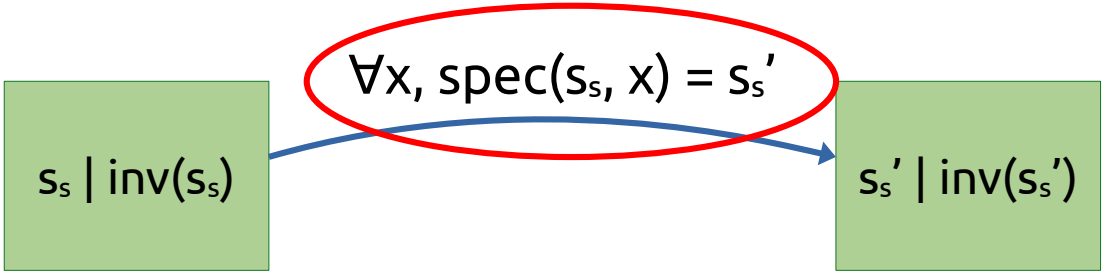
Real-time + Scalable Predictability

- Bounded path complexity
  - Good match for *push-button verification*
  - Example: *Serval* from U. Wash.
- Push-Button (Functional) Verification
  - Implementation (C) – symbolic evaluation
  - Specification – state machine transformations
  - Mapping – from Implementation to Specification
  - Proof 100% via SMT solver

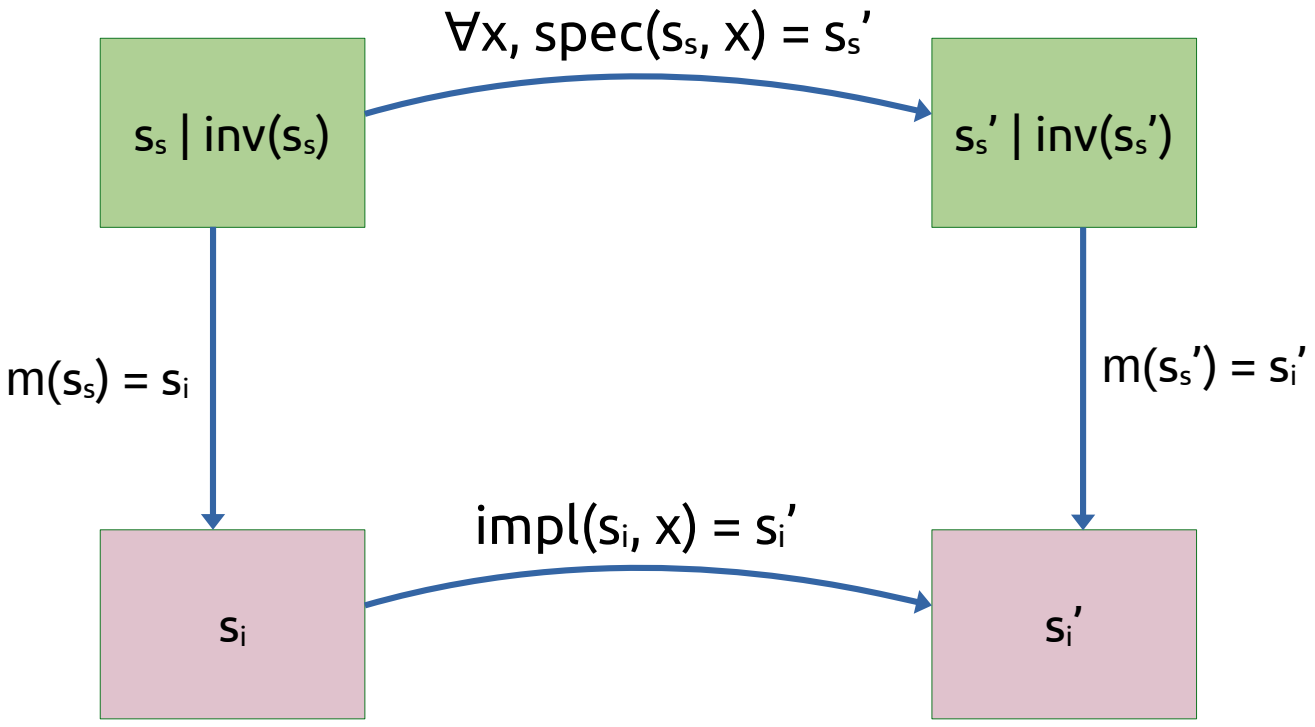
Push-button Verification



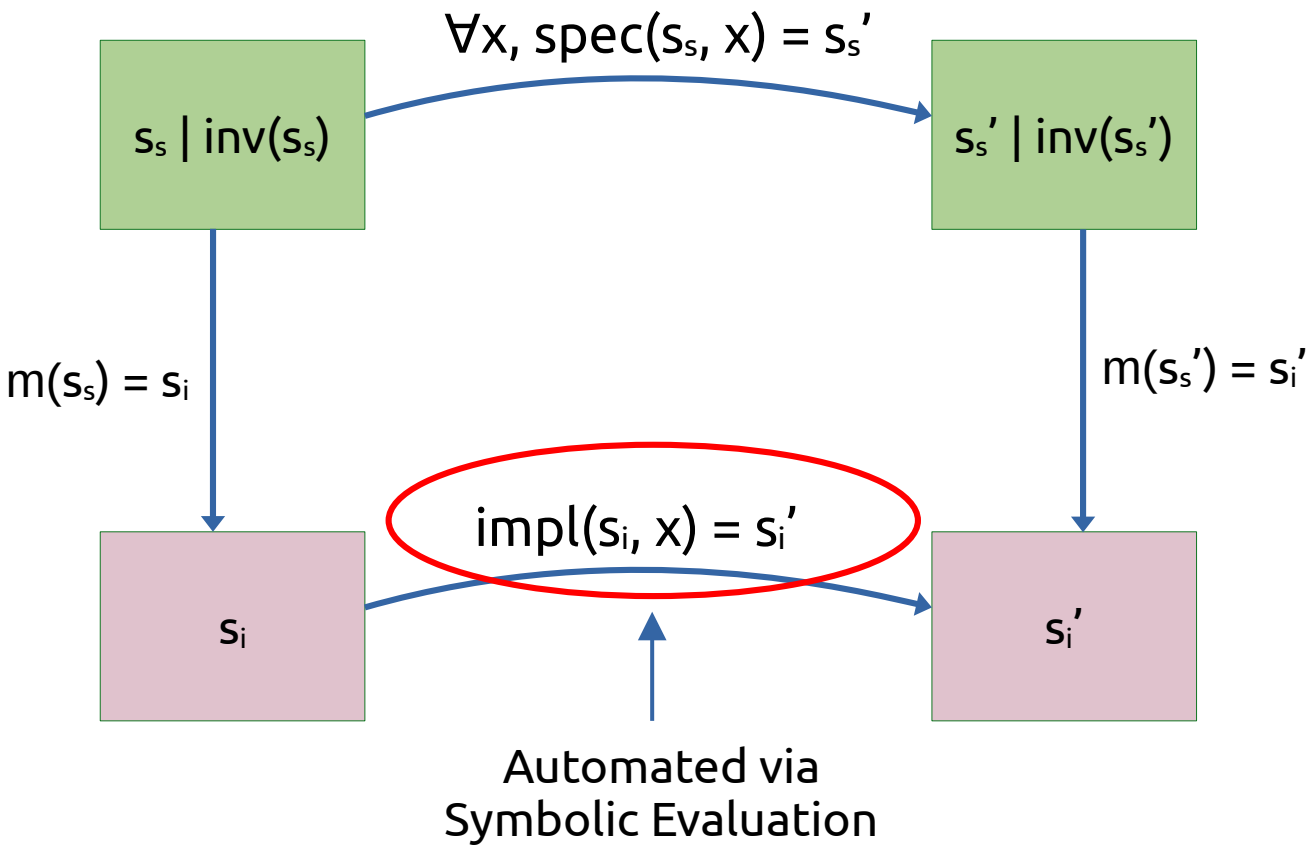
Push-button Verification



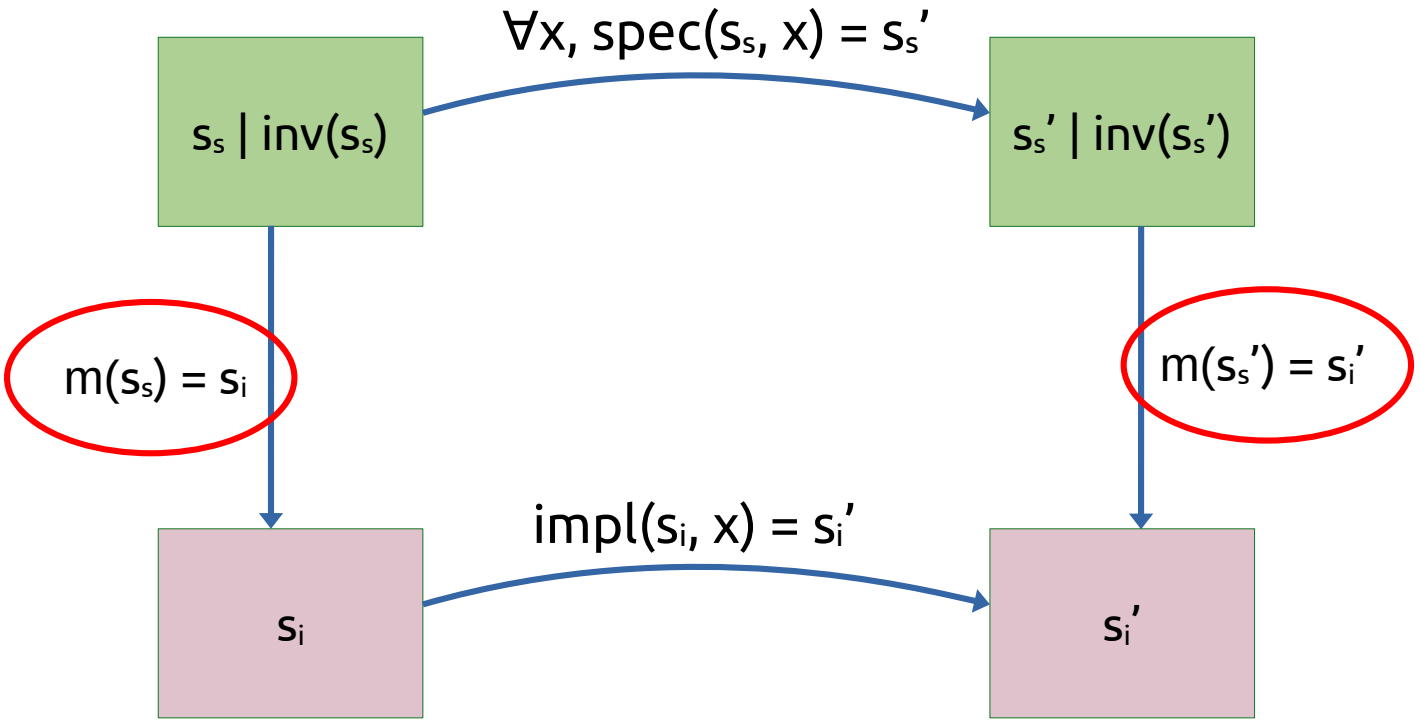
Push-button Verification



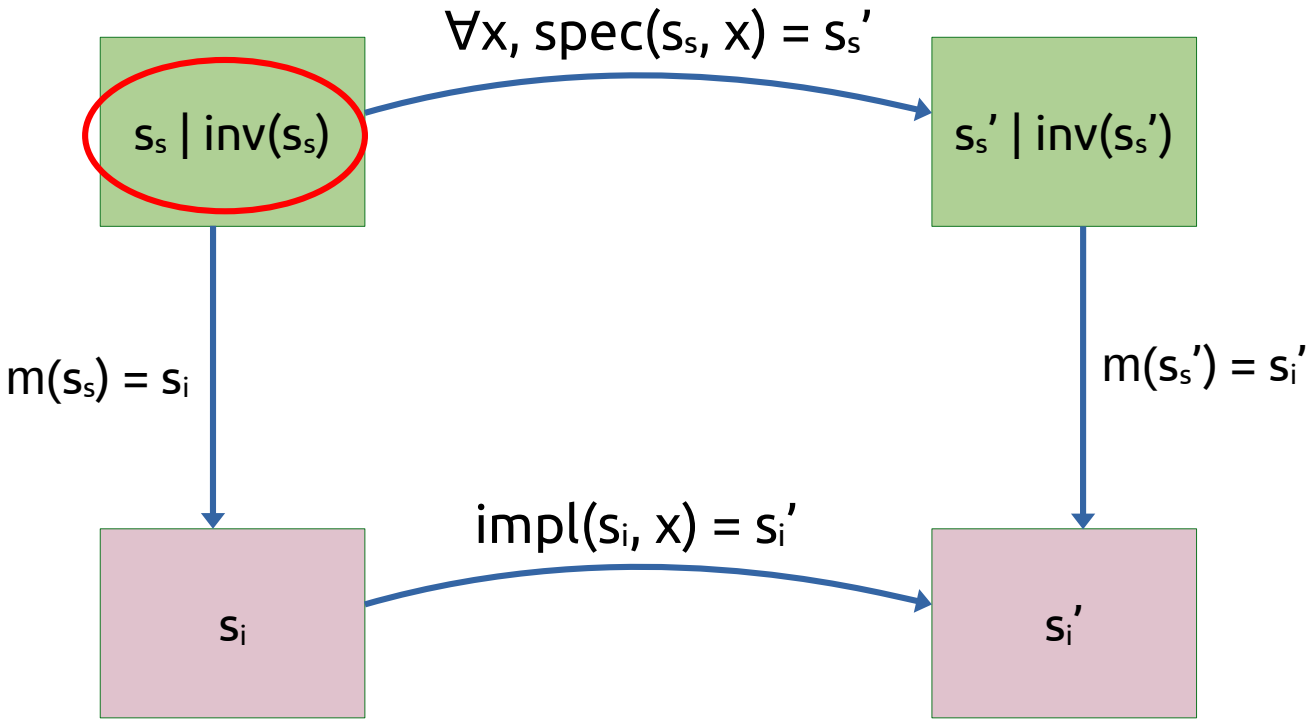
Push-button Verification



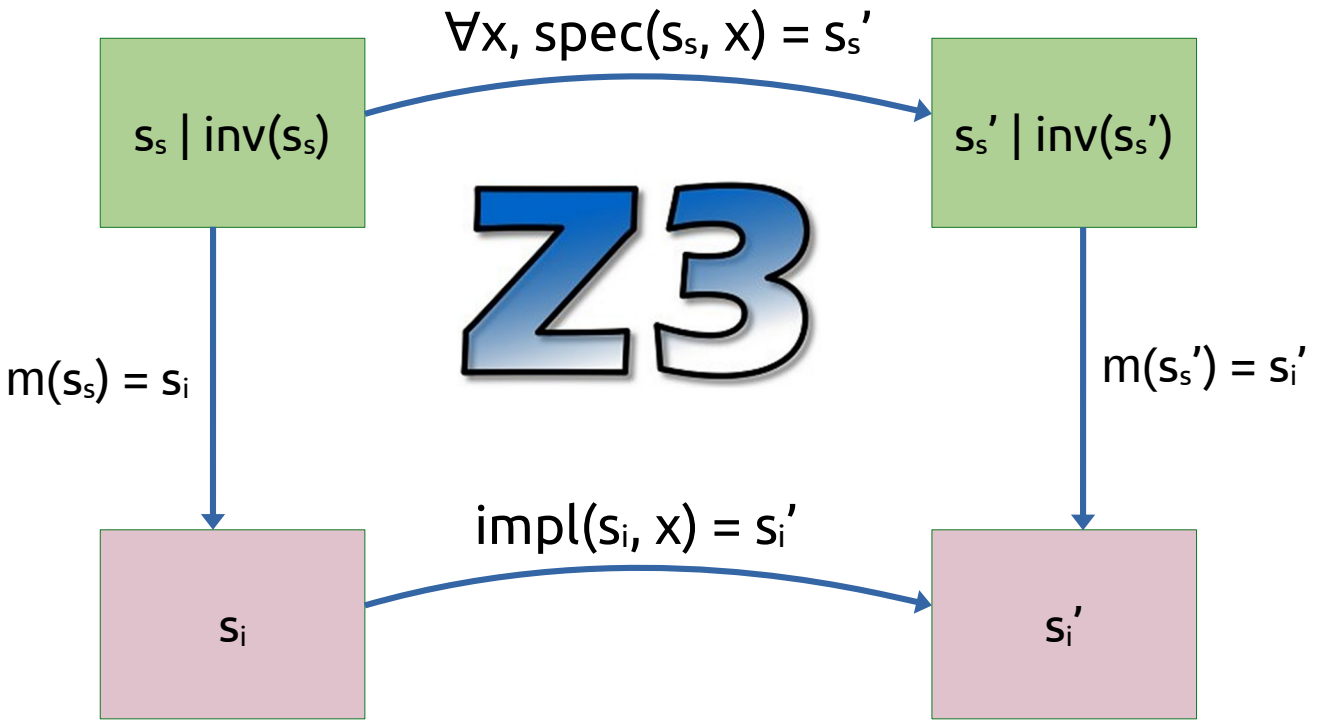
Push-button Verification



Push-button Verification

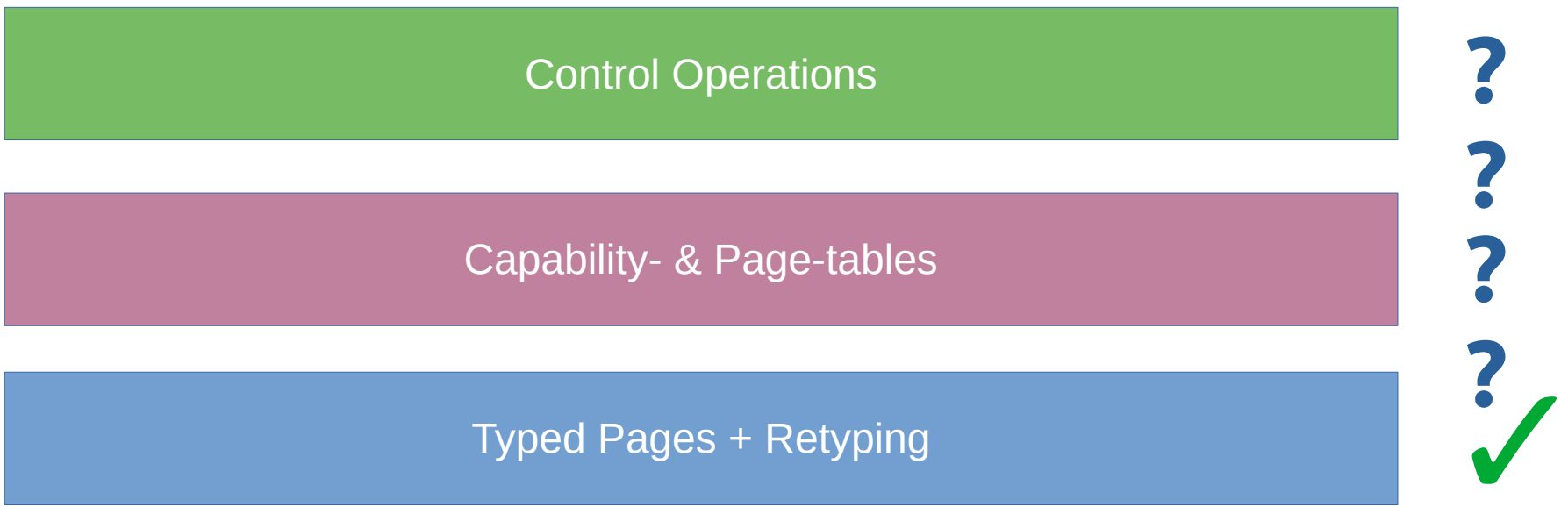


Push-button Verification



Composite Push-Button Verification

WiP: Composite Kernel, Version 4



Composite & Systems @ GWU