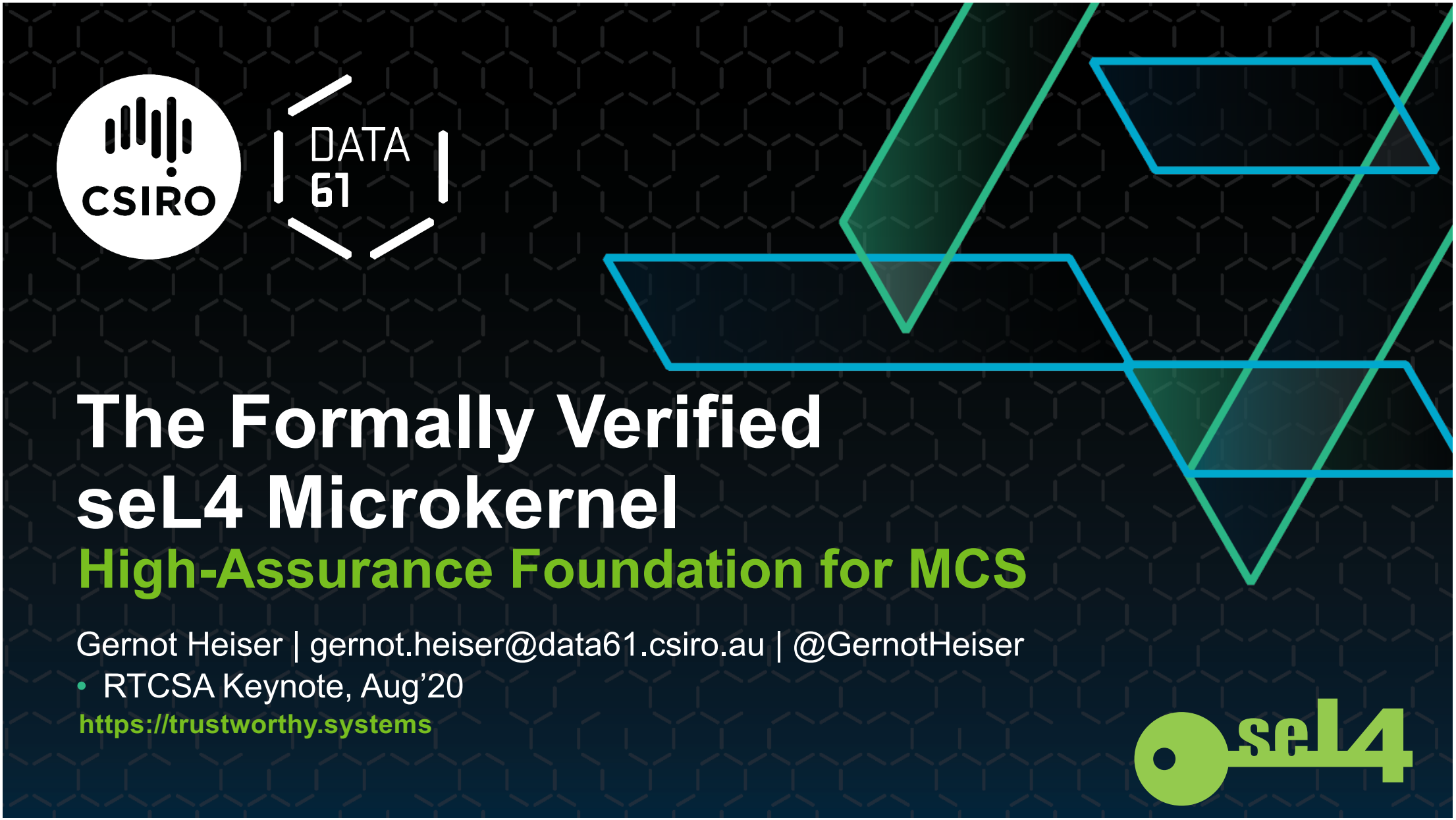



# The Formally Verified seL4 Microkernel

## High-Assurance Foundation for MCS

Gernot Heiser | gernot.heiser@data61.csiro.au | @GernotHeiser

- RTCSA Keynote, Aug'20

https://trustworthy.systems

seL4

# What Is Needed For Mixed-Criticality?

**During a review process, ca Aug'17:**

- [Gernot:] Temporal isolation is *necessary* for mixed criticality systems.
- [Reviewer:] Wrong, temporal isolation is *sufficient*.

# What Is a Mixed-Criticality System?

*"A mixed-critical system […] supports the execution of safety-critical, mission-critical, and non-critical software within a single, secure compute platform."* [Barhorst'09]

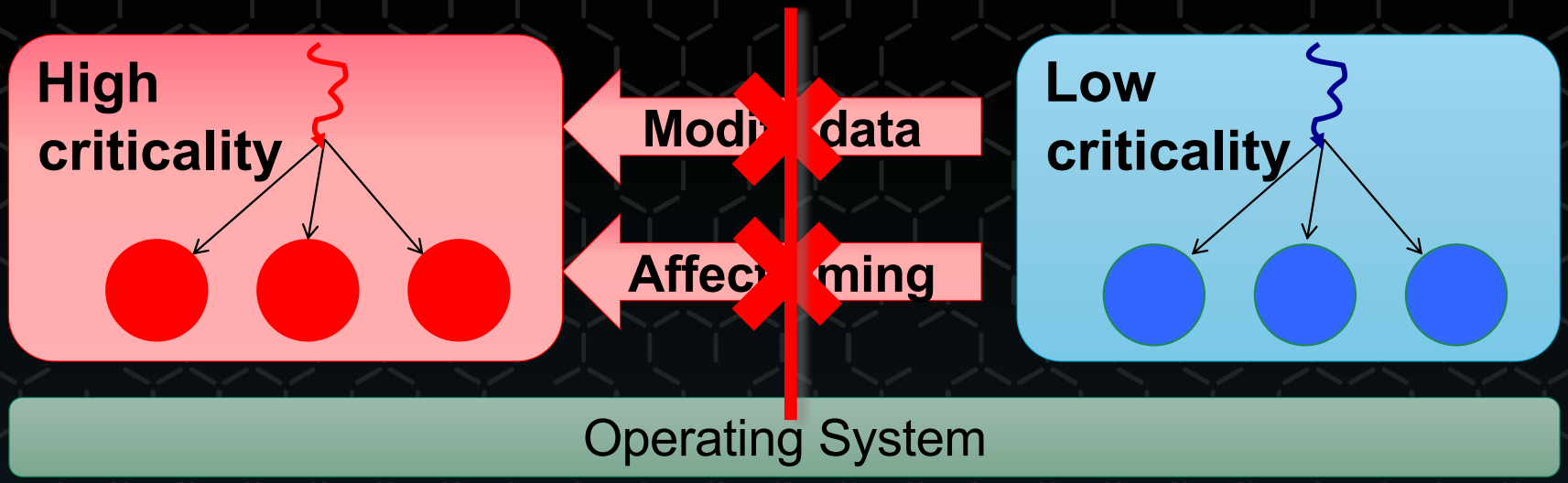
Criticality of a component is defined by the impact of failure:

- loss of life
- injury
- inconvenience

Certification of critical component must not depend on behaviour of less critical components
⇒ **must prevent *any* interference by less critical components!**

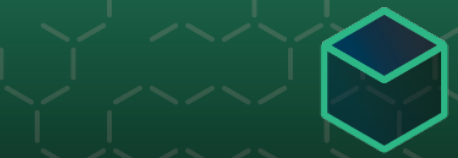

CSIRO DATA 61

# Preventing Interference – The OS's Job




We need an OS that can guarantee the absence of interference!

# seL4:
# Provable Isolation




CSIRO DATA 61

# What is seL4?


The world's **first** operating-system kernel with **provable** security enforcement

World's most advanced mixed-criticality OS

Open Source

The world's **only** protected-mode OS with complete, sound timeliness analysis

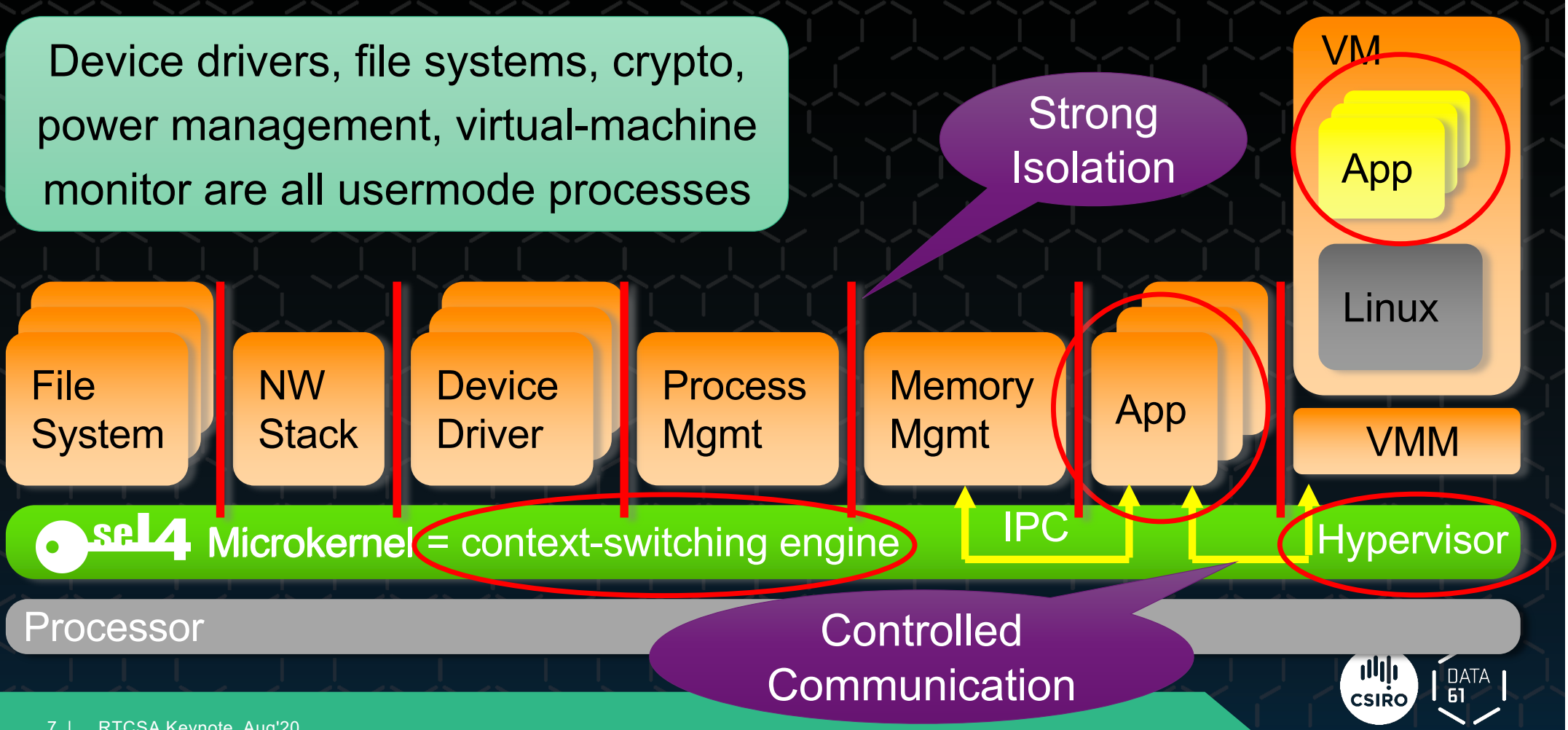
The world's **fastest** general-purpose microkernel, designed for **real-world** use



CSIRO DATA 61

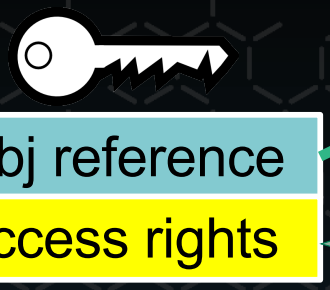
# A Microkernel is not an OS

Device drivers, file systems, crypto, power management, virtual-machine monitor are all usermode processes



# Capability-Based Access Control

**Capability = Access Token:**
Prima-facie evidence of privilege





Eg. thread, address space

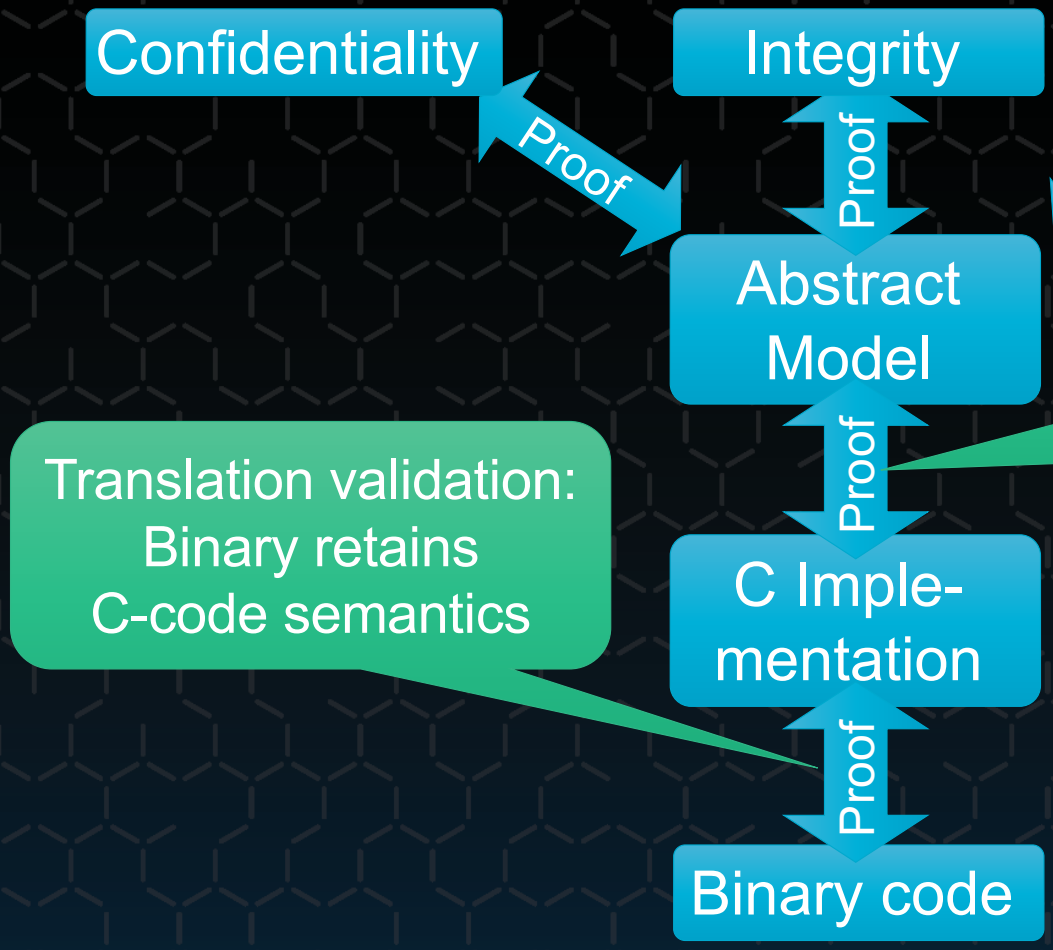
Eg. read, write, send, execute…

Capabilities provide:
- Fine-grained access control
- Reasoning about information flow

Any system call is invoking a capability:

```
err = method( cap, args );
```

# Proved Spatial Isolation




Model enforces isolation

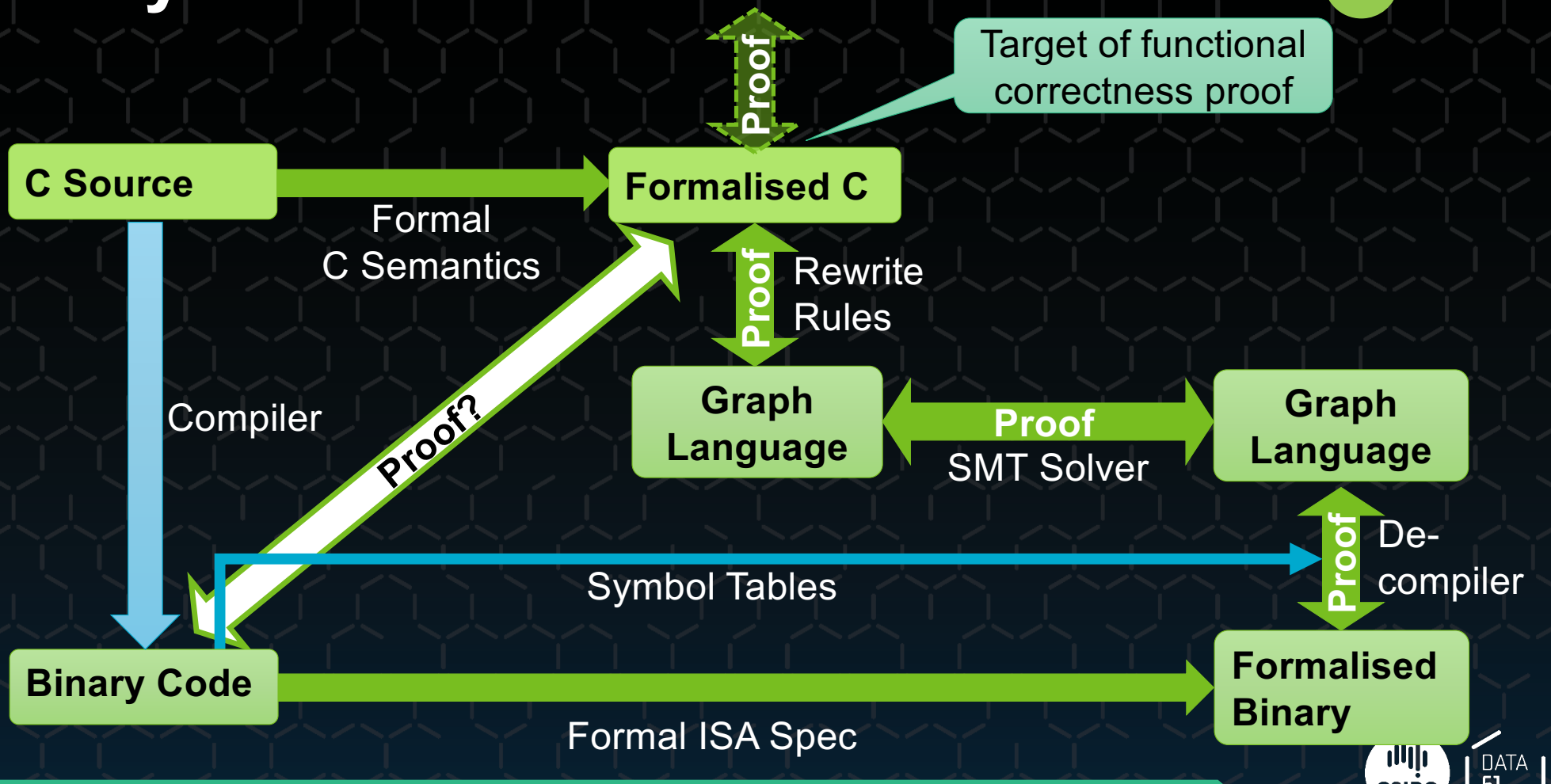
Functional correctness: C code only behaves as specified

Translation validation: Binary retains C-code semantics

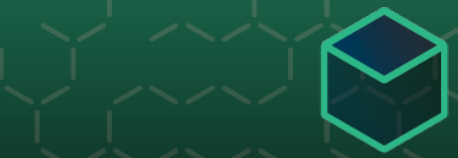
**Limitations (work in progress):**

- Kernel initialisation not yet verified
- MMU & caches modelled abstractly
- Multicore version not yet verified
- Timing channels not ruled out

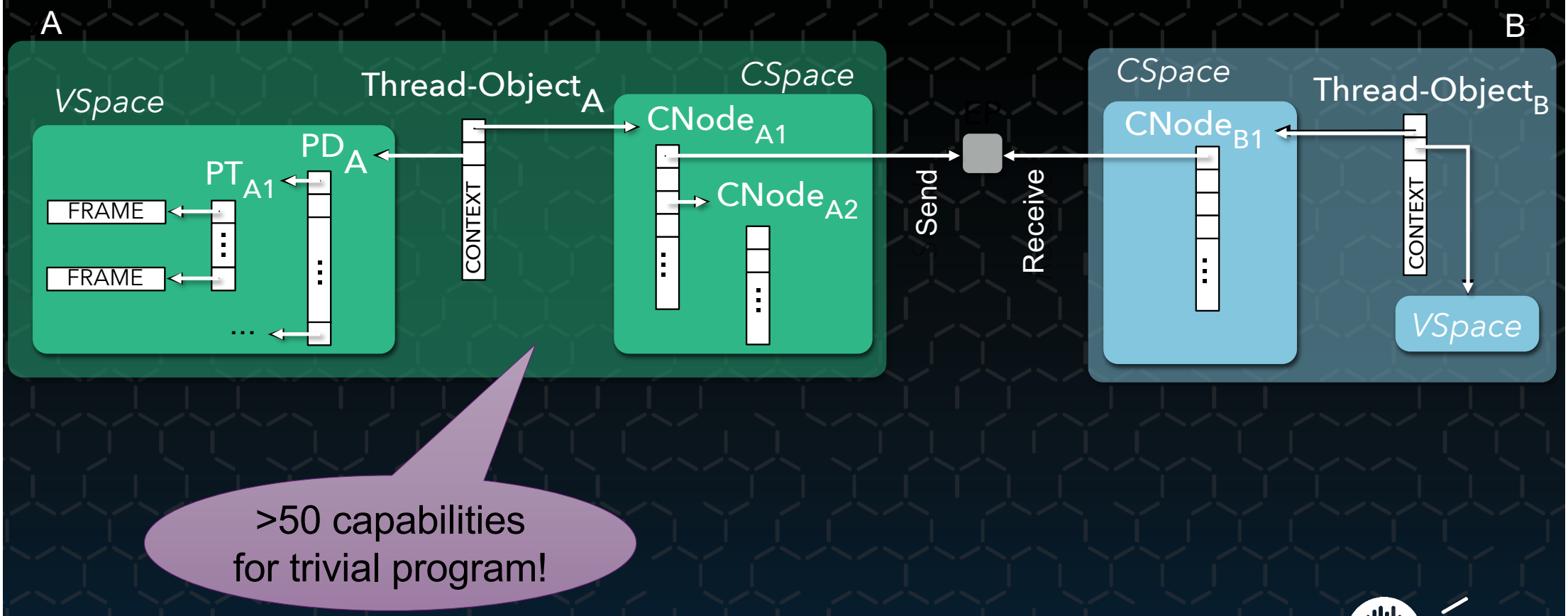
# Binary Code Verification



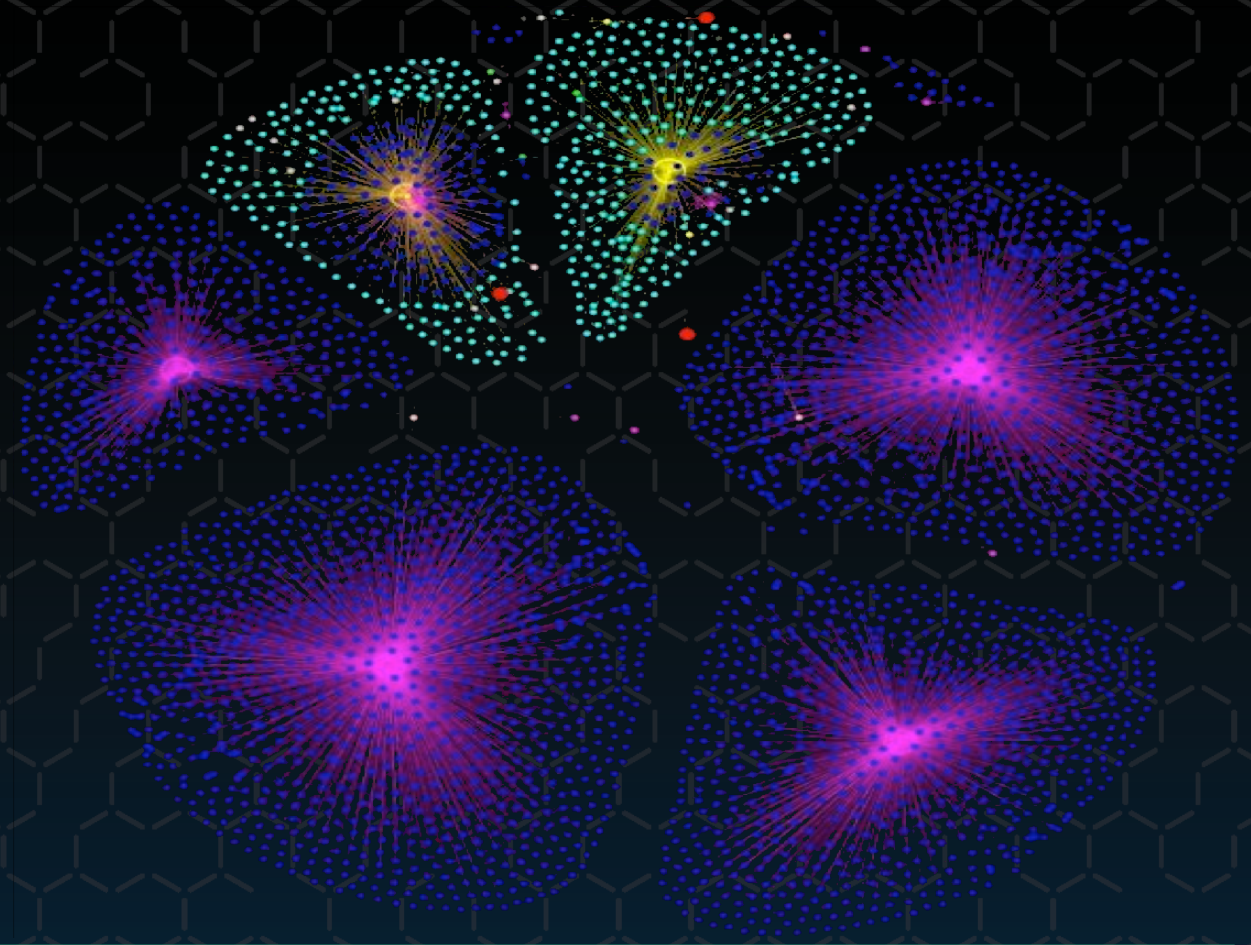
# Isolation by Architecture



# Issue: Capabilities are Low-Level

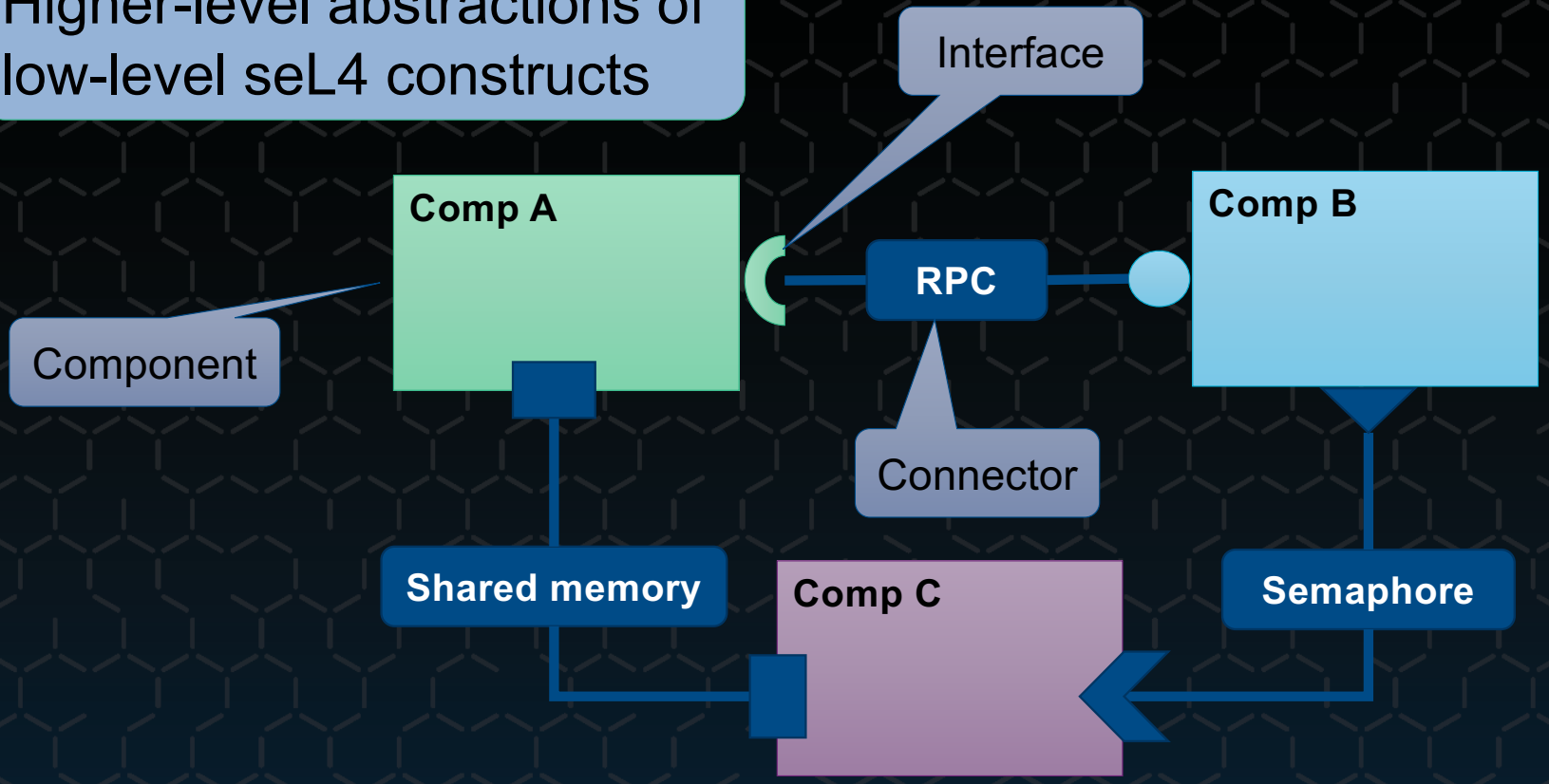


# Simple But Non-Trivial System

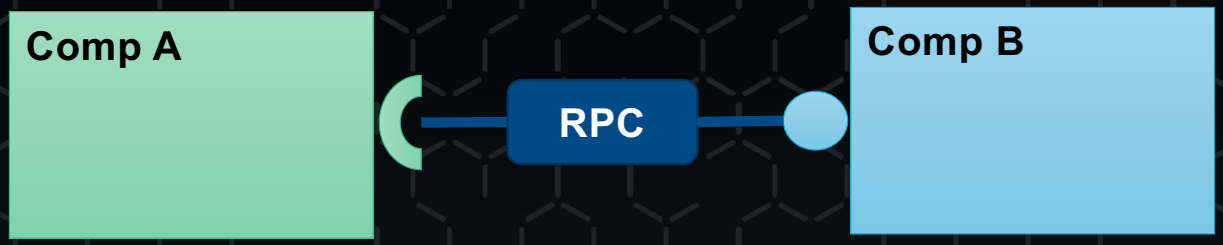


# Component Middleware: CAmkES

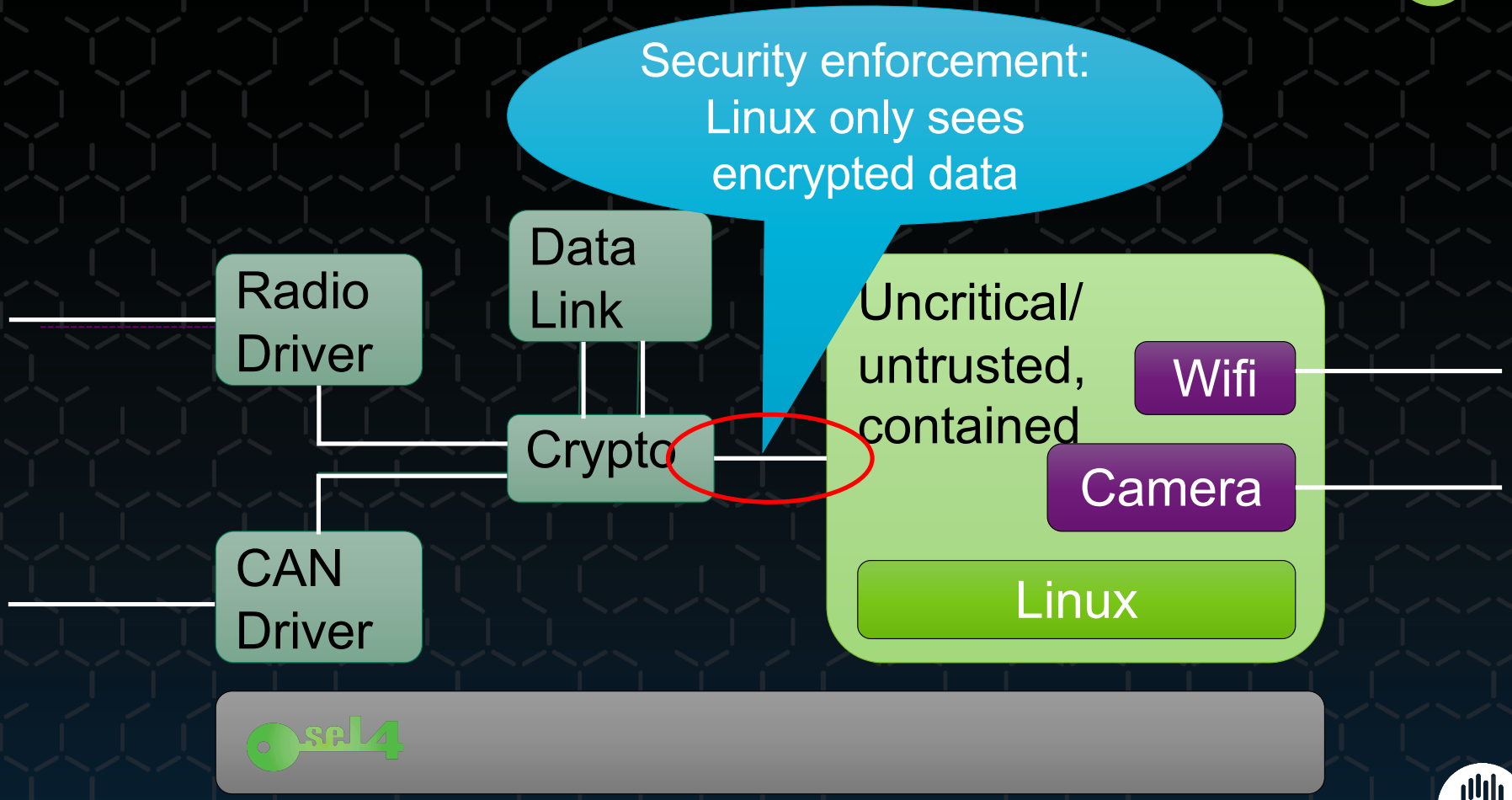
Higher-level abstractions of low-level seL4 constructs



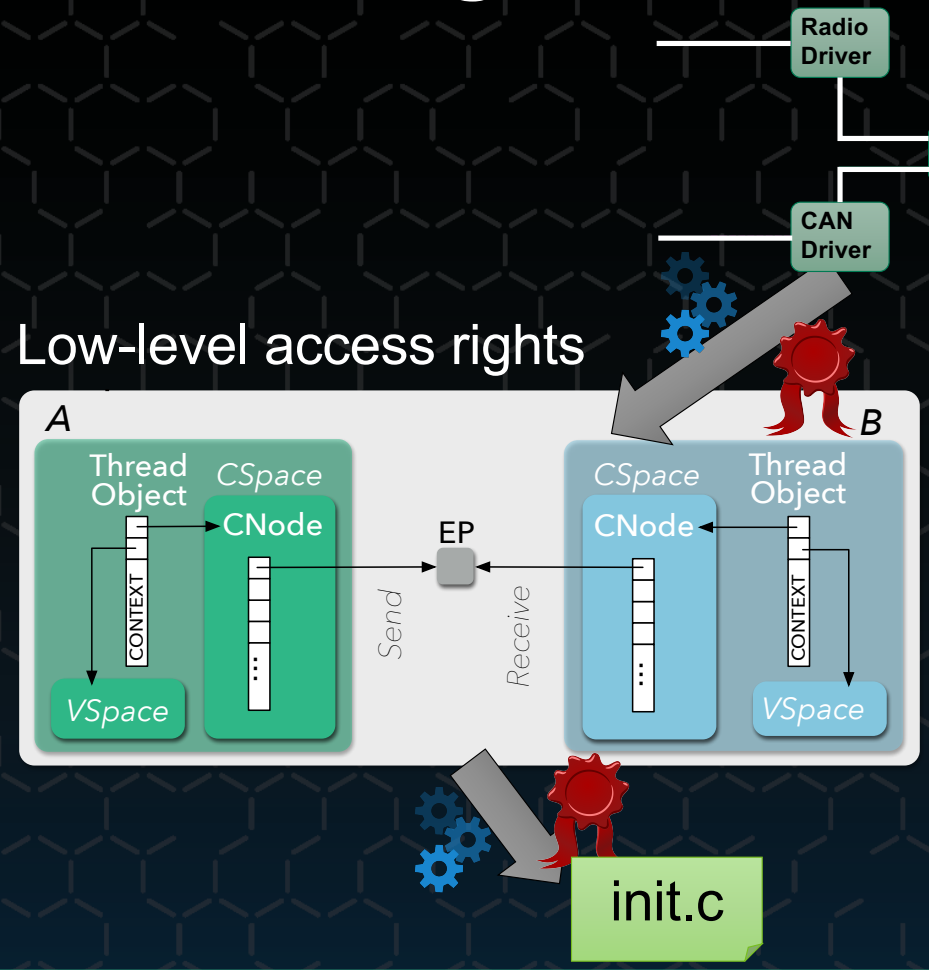
# Trivial System in CAmkES



# HACMS UAV Architecture



# Enforcing the Architecture




Architecture specification language

glue.c

driver.c

VMM.c

Compiler/ Linker

binary

**Conditions apply**

# Military-Strength Security

Unmanned Little Bird (ULB)

**DARPA HACMS: Retrofit existing system!**

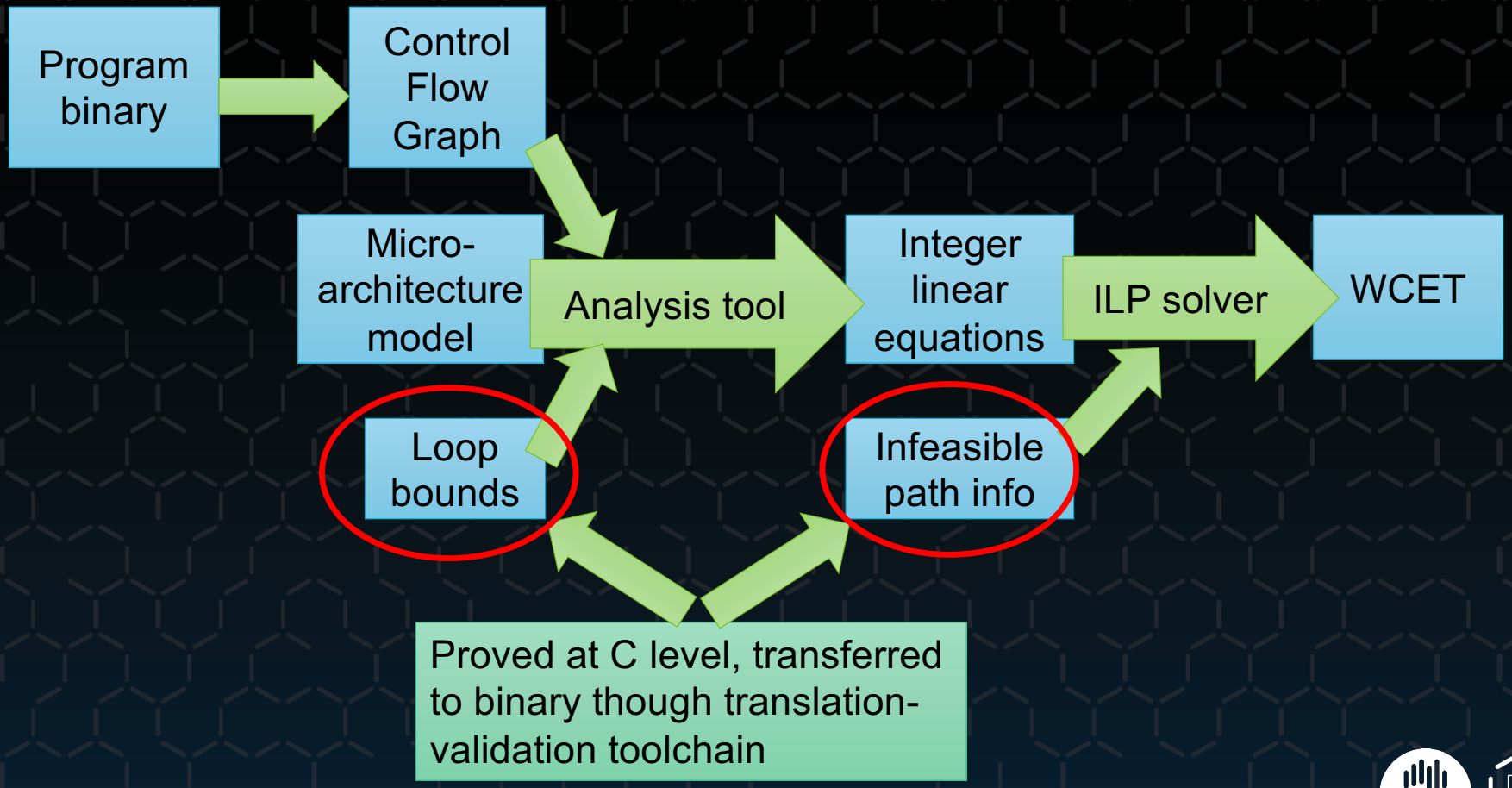
Autonomous trucks

Secure Comms Dongle


Cross-Domain Desktop Compositor

# Temporal Isolation: WCET Analysis

# High-Assurance WCET Analysis

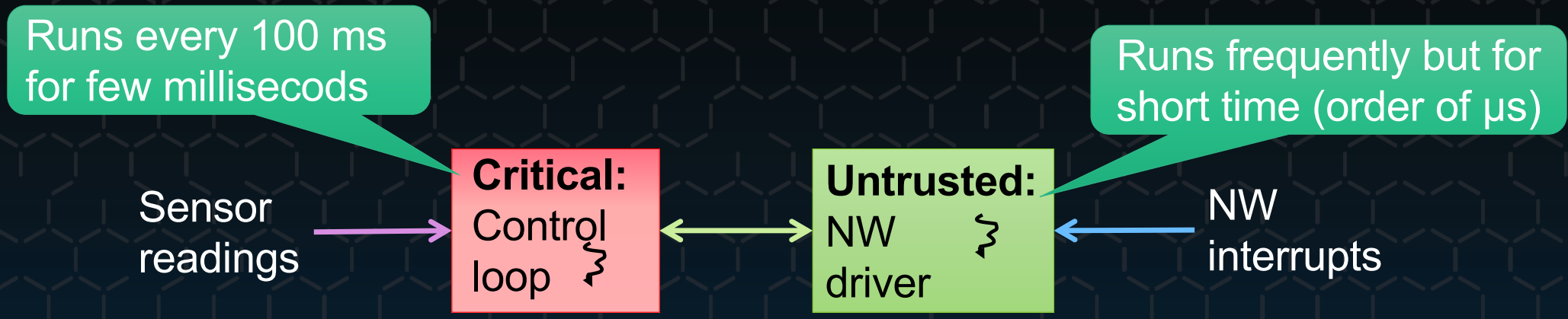


# Temporal Isolation: Controlling Time

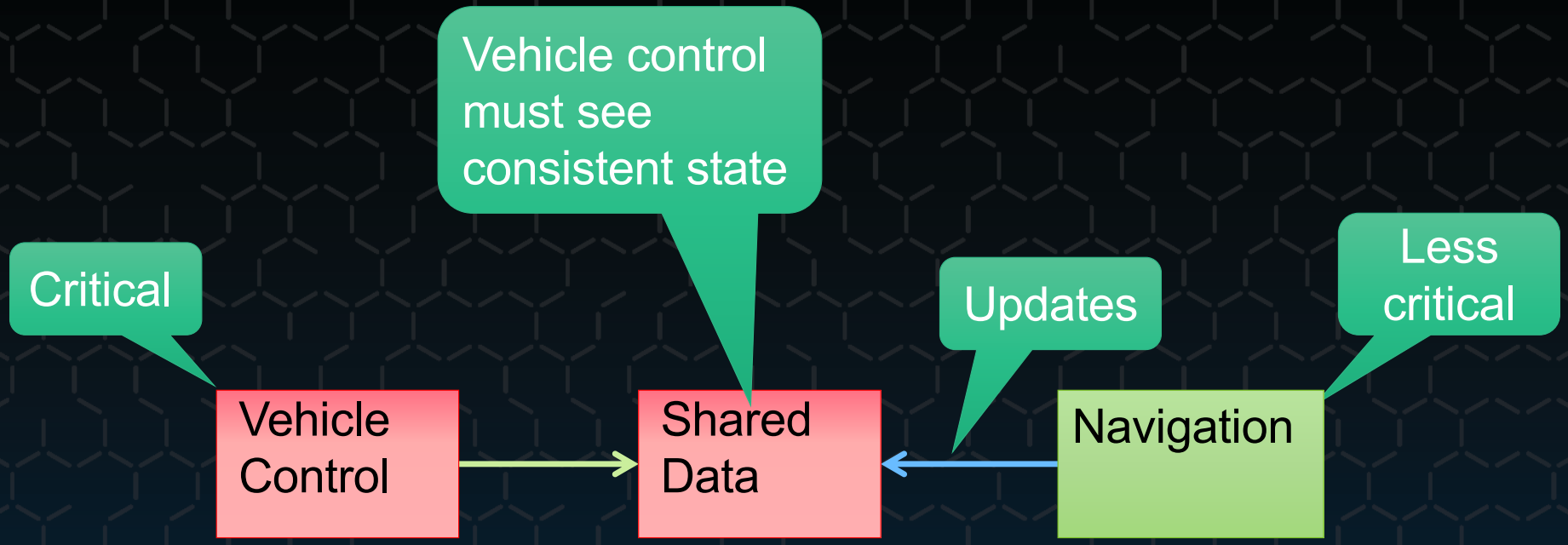
# Mixed Criticality: Critical + Untrusted


**NW driver must preempt control loop**

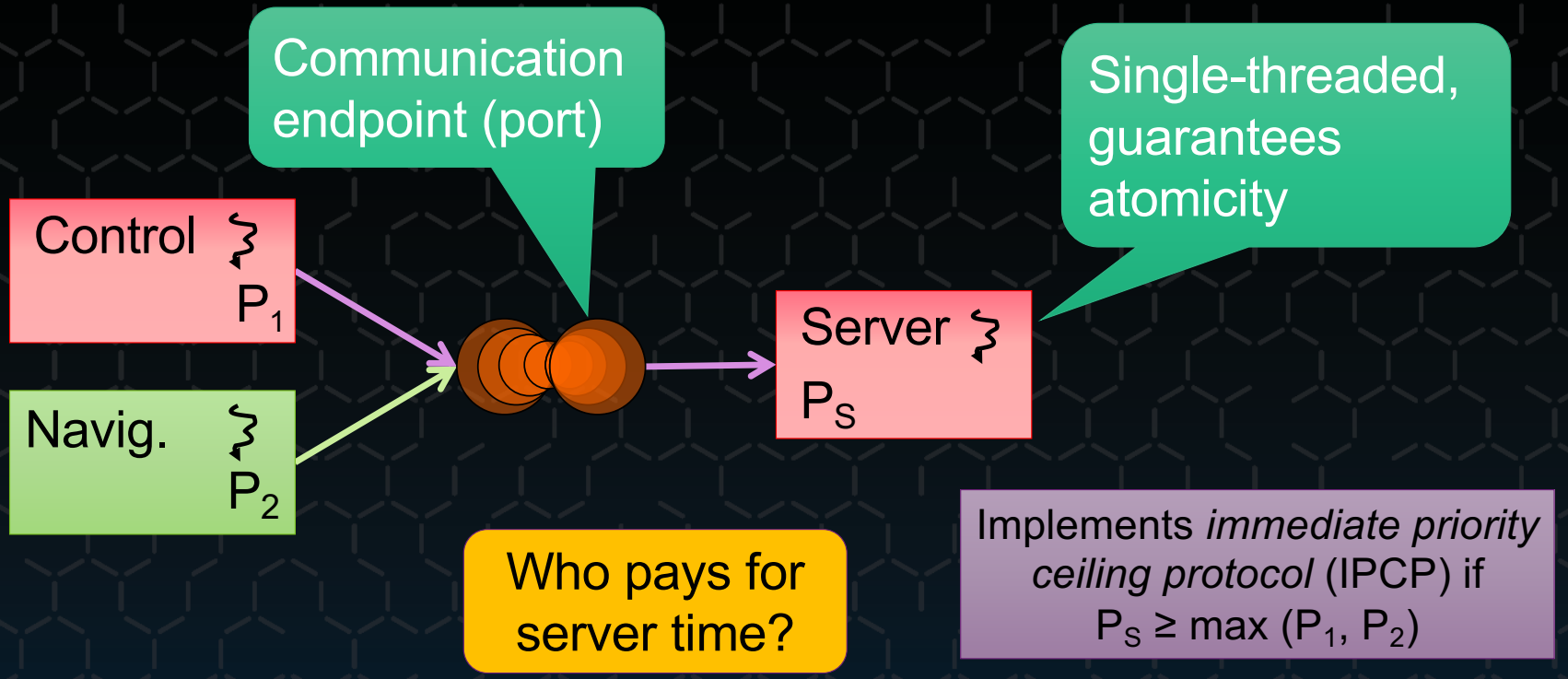
- … to avoid packet loss
- Driver must run at high prio
- Driver must be trusted not to monopolise CPU



# MCS Challenge: Sharing



# Sharing: Delegation to *Resource Server*



# Solution: Time Capabilities

## Classical thread attributes

- Priority
- Time slice

Not runnable if null

## New thread attributes

- Priority
- Scheduling context capability

Capability for time

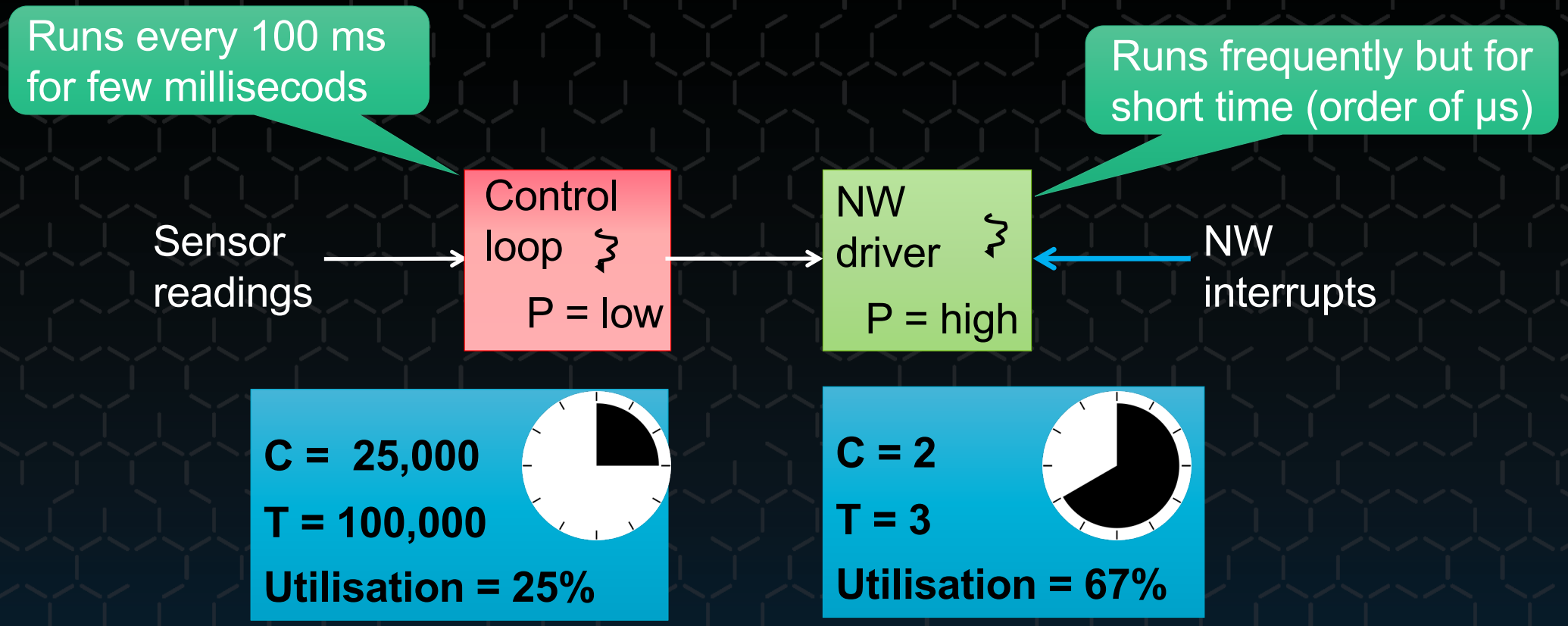
Limits CPU access – sporadic server

**Scheduling context object**
- T: period
- C: budget (≤ T)

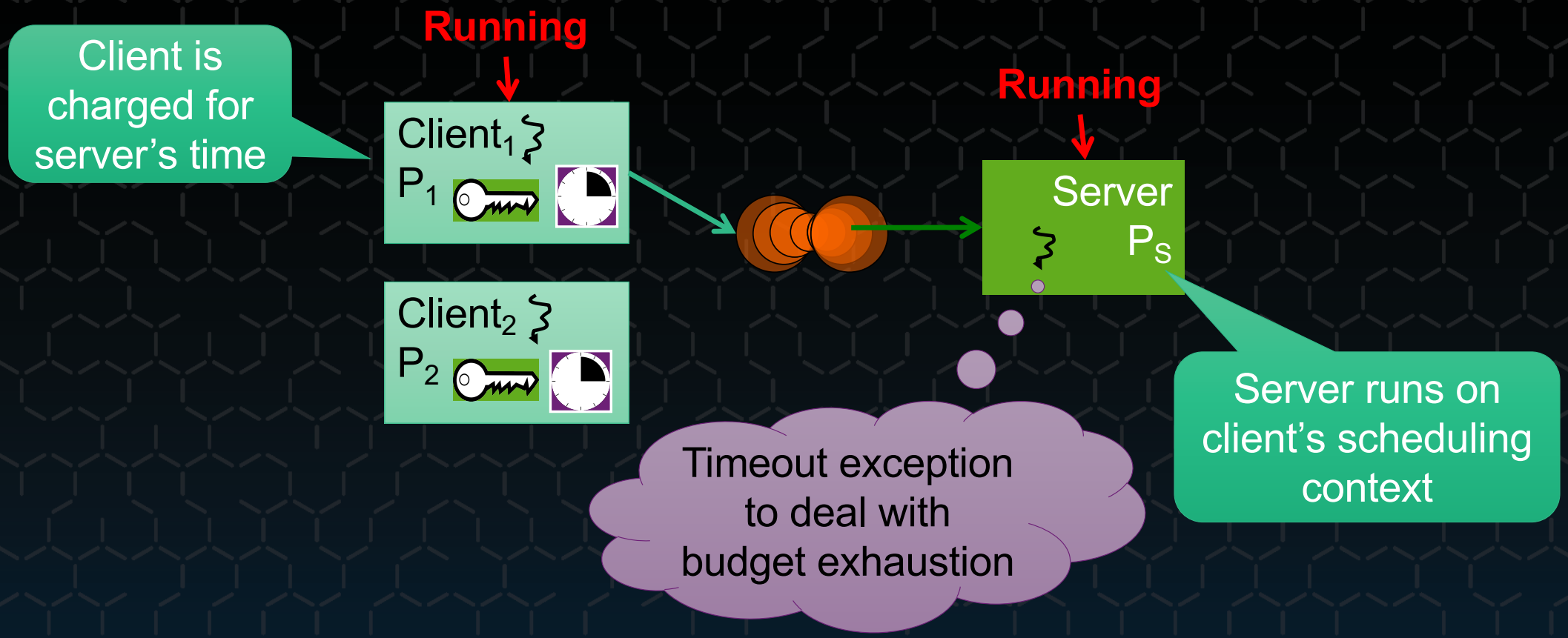
C = 2
T = 3

Enables reasoning about time and temporal isolation for mixed-criticality systems

# MCS with Scheduling Contexts



# Shared Server Time Charged to Client



# seL4 MCS Support


- **Time as a first-class resource:**
  - Enforcement of delegatable time budgets
  - Suitable for formal reasoning
  - Verification to be completed this year
- **Status:**
  - Functional correctness of MCS extensions presently being verified for Arm and RISC-V
- **To Do:**
  - Proving scheduler properties
  - Formal framework for reasoning about timeliness of applications

# Thank You!