Trustworthy Operating Systems

For Critical Embedded / Cyber-Physical Systems

Gernot Heiser | gernot.heiser@data61.csiro.au | @GernotHeiser
Trustworthy Systems | Data61 CSIRO and UNSW Sydney

Embedded Systems Week, Seoul 2017

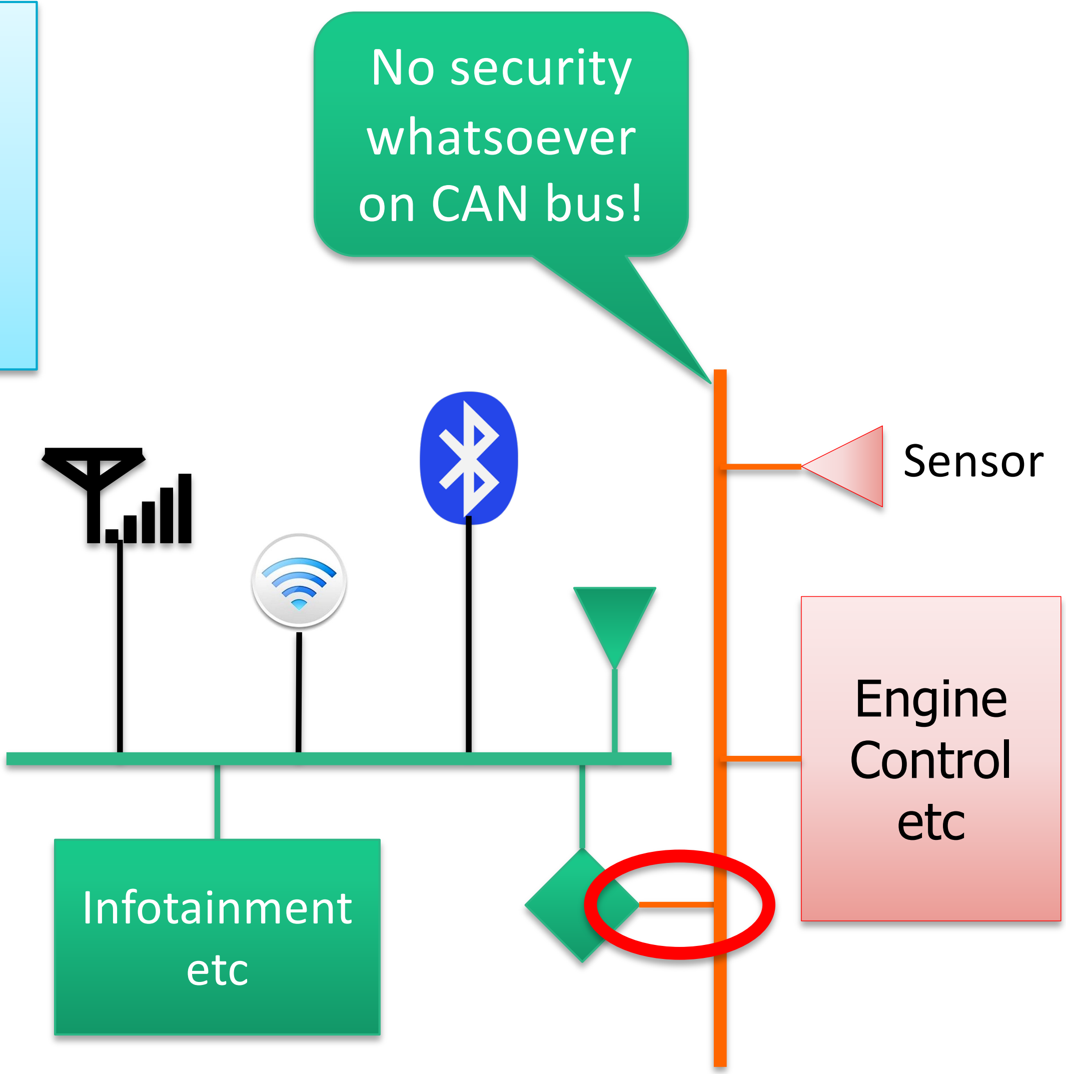
https://trustworthy.systems

Embedded Systems Security – An Oxymoron?

Car Hacking – What's Behind?

Networking for:

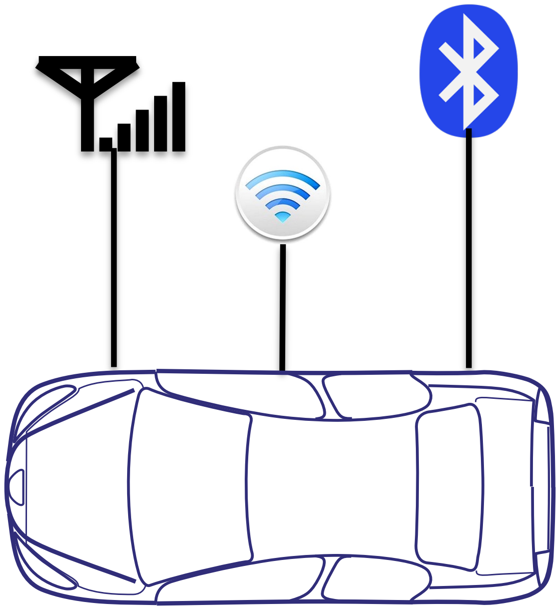
- Entertainment
- Connected car
- Safety (tire pressure…)
- Maintenance (OTA upgrades)



Challenge of Networking

Networking creates remote attack opportunities

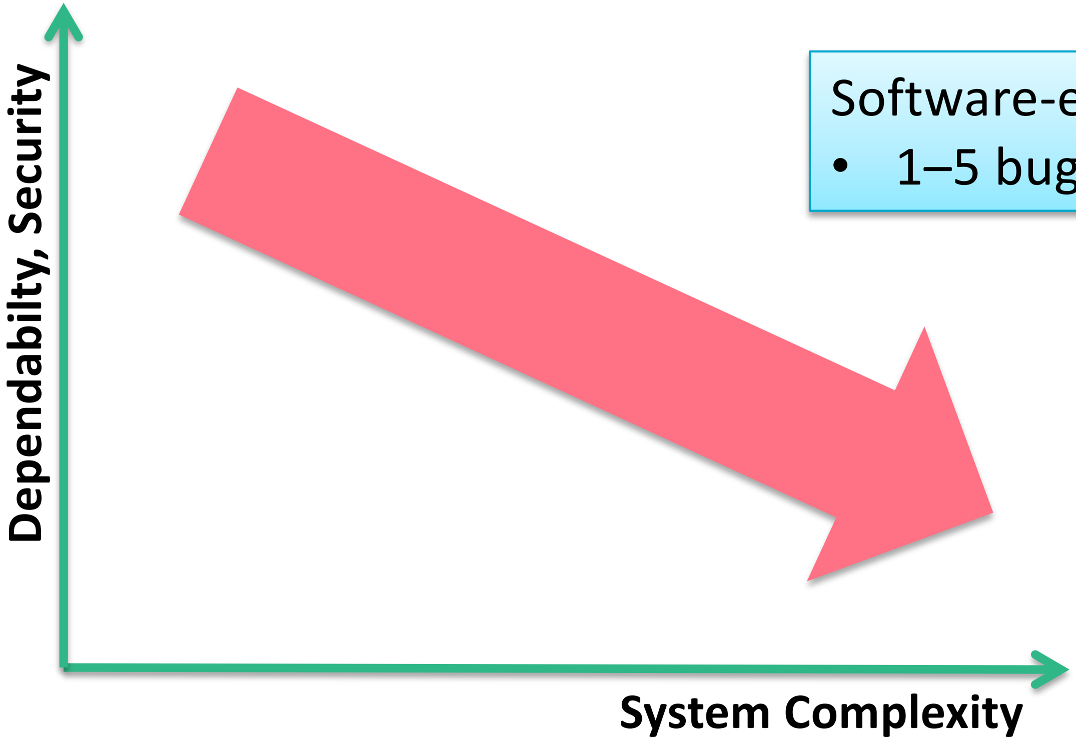
- from passengers (wifi, Bluetooth)
- from nearby cars (wifi, Bluetooth) – drive-by shooting, spread of viruses
- from anywhere (cellular)



Attack vectors:

- Insecure protocols
- Reusing crypto keys
- Software vulnerabilities

Software Vulnerabilities



Software-engineering rule of thumb:

- 1–5 bugs per 1,000 lines of ***quality*** code

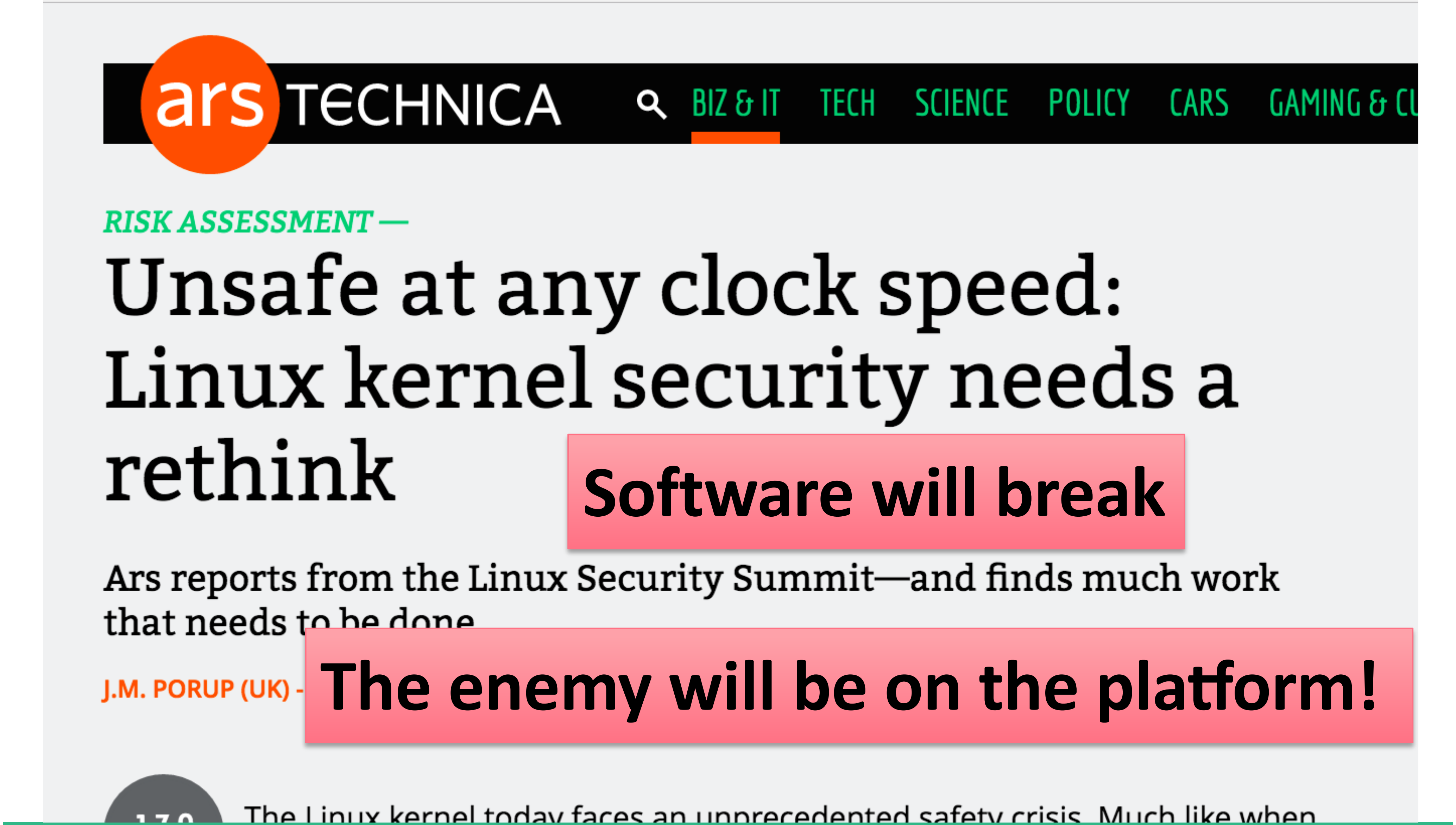
Bluetooth protocol stack:
Multiple 100,000 lines

Linux kernel:
Tens of millions lines

Complexity Drivers

- Features/functionality
- Legacy reuse

Linux “Security”



RISK ASSESSMENT —

Unsafe at any clock speed: Linux kernel security needs a rethink

Ars reports from the Linux Security Summit—and finds much work that needs to be done.

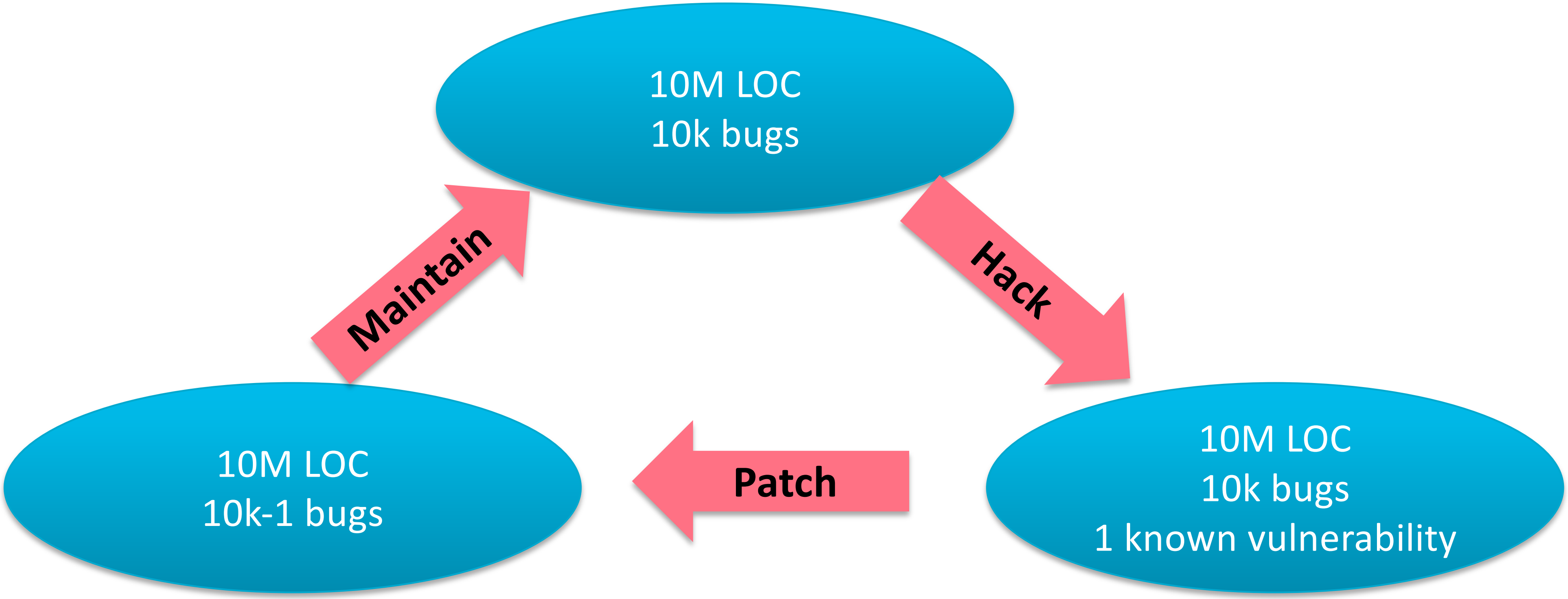
J.M. PORUP (UK) -

The Linux kernel today faces an unprecedented safety crisis. Much like when

Software will break

The enemy will be on the platform!

OK, So Let's Patch Regularly


10M LOC
10k bugs

Hack

10M LOC
10k bugs
1 known vulnerability

Patch

10M LOC
10k-1 bugs

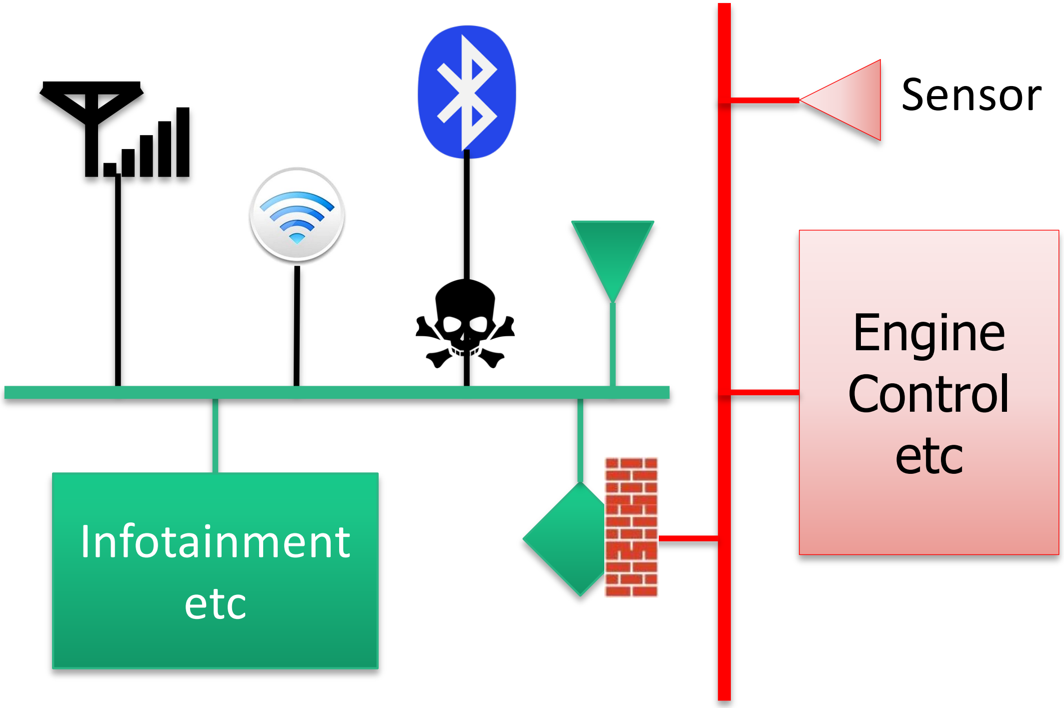
Maintain

Patch-and-Pray: A losing proposition

So, Let’s Use Firewalls!

- Imposes overhead (SWaP) or
- Runs on vulnerable OS ⇒ worthless if OS compromised
- Even more code – may *increase* attack surface
- No help for valid messages that trigger bugs in software

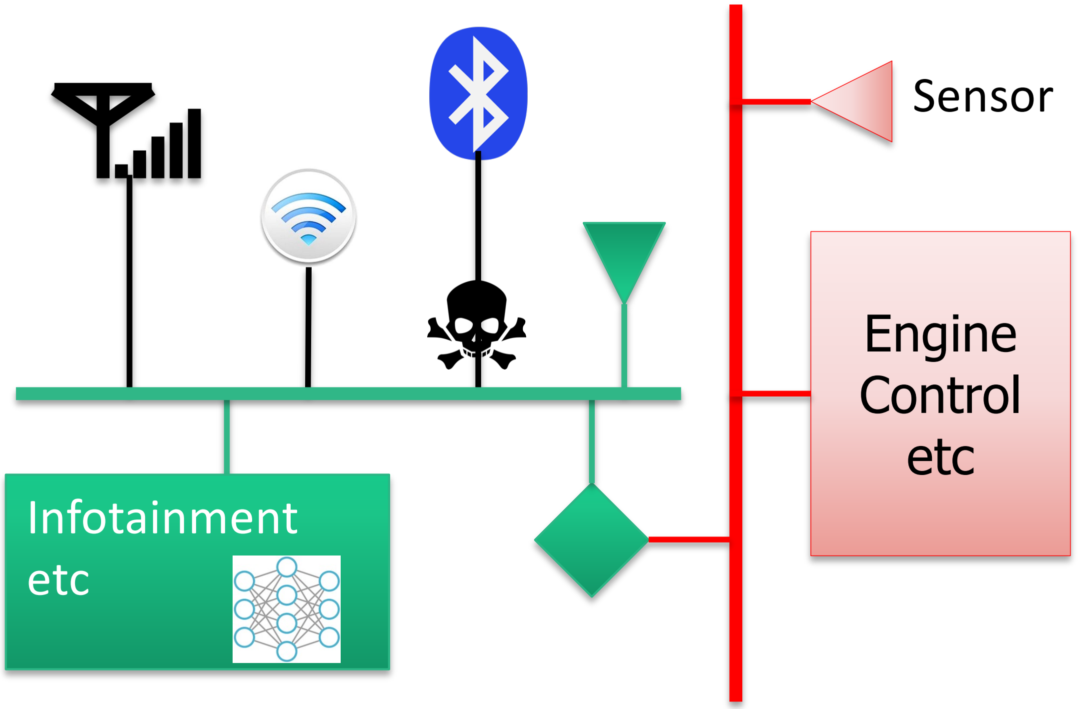
Firewalls treat symptoms, not causes of problems!



Let's Use AI to Detect Compromise!

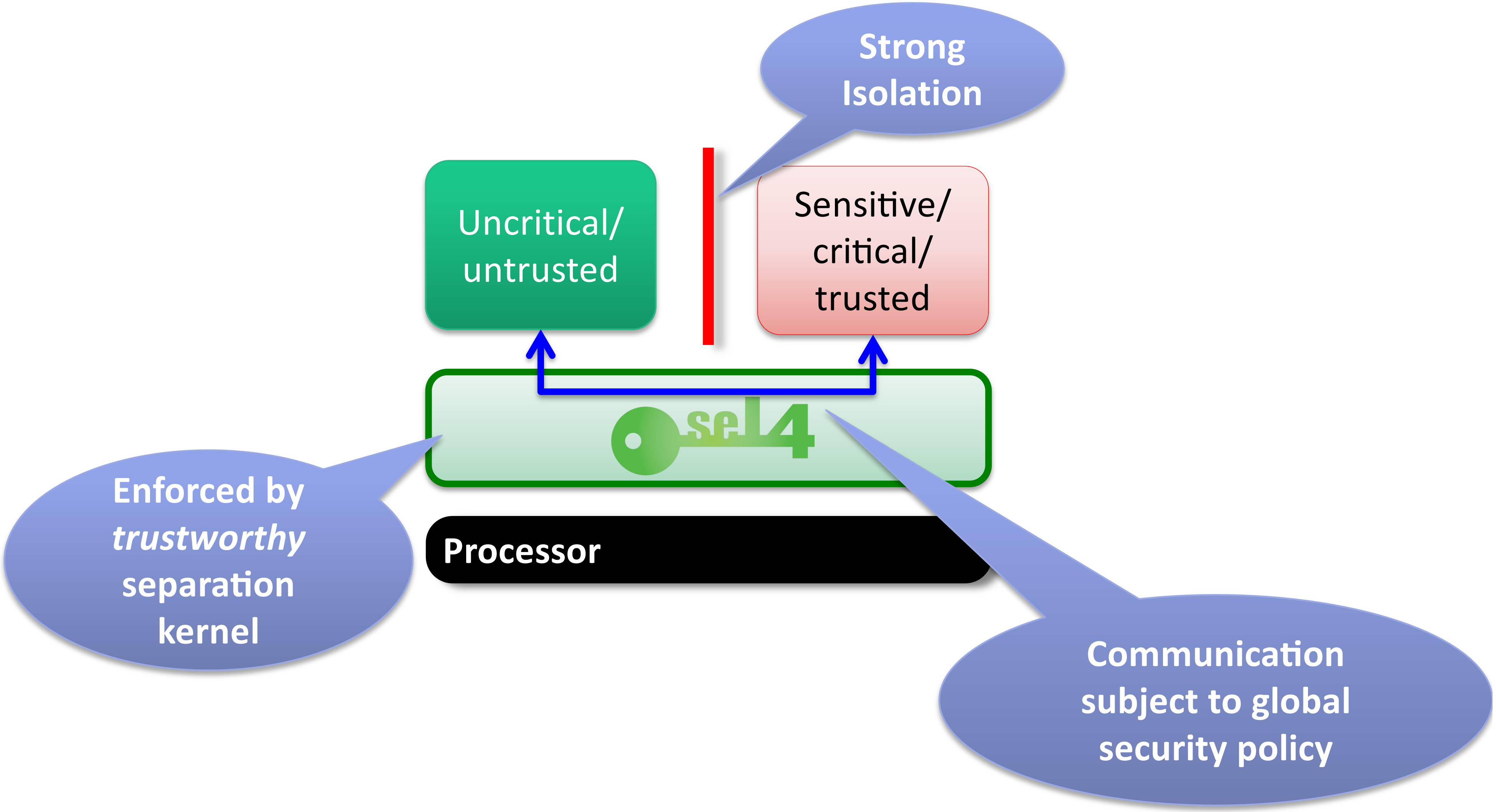
- Runs on vulnerable OS ⇒ worthless if OS compromised
- Even more code – may *increase* attack surface
- Can only detect that system is **already compromised**

Intrusion detection: admission of defeat



Trustworthy Operating Systems

Fundamental Security Requirement: Isolation


Strong Isolation
Uncritical/ untrusted
Sensitive/ critical/ trusted
seL4
Processor
Enforced by *trustworthy* separation kernel
Communication subject to global security policy

Trustworthiness: Can We Rely on Isolation?

A system is **trustworthy** if and only if:

- it behaves **exactly** as it is specified,
- in a **timely** manner,
- while ensuring **secure** execution

Claim:

A system must be considered *untrustworthy* unless *proved* otherwise!

Corollary [with apologies to Dijkstra]:

Testing, code inspection, etc. can only show *lack of trustworthiness*!

seL4 Provably Secure Operating System

~10,000 lines of code

⬇

Small attack surface,
Amenable to verification

Small,

fast,

capability-based,

OS kernel

World's fastest OS designed for security and safety

⬇

Suitable for real world

All operations explicitly authorised by an access token, i.e. capability

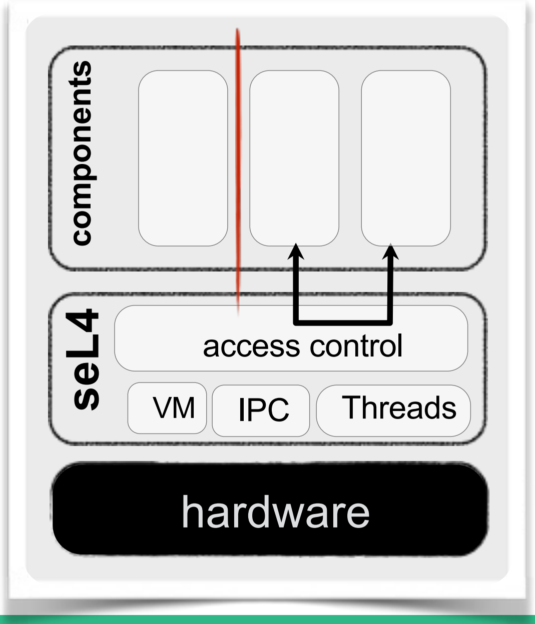
□

- **Confined damage**
- **Least privilege**

Code that runs in privileged mode of the hardware

□

Most critical part

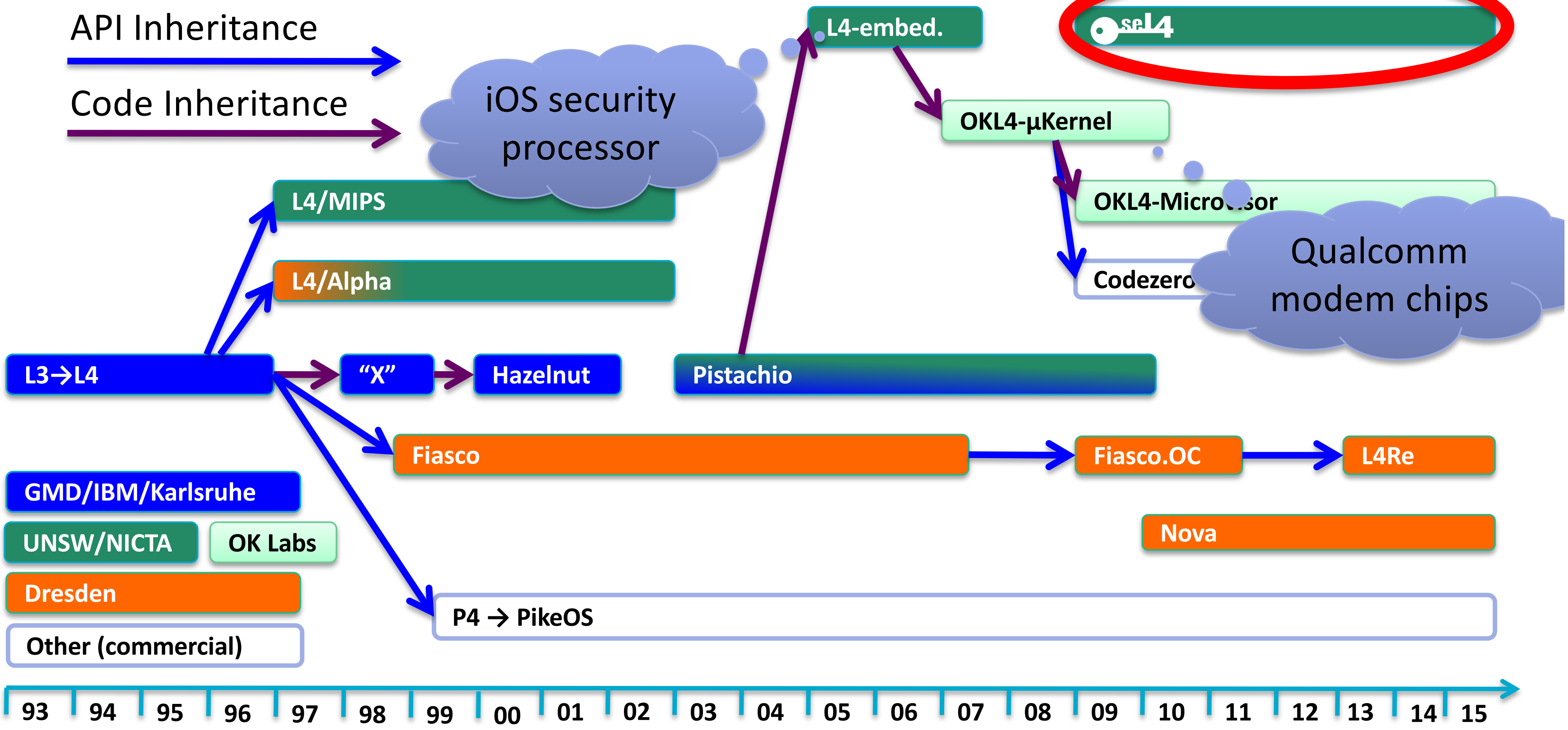


Unprivileged mode

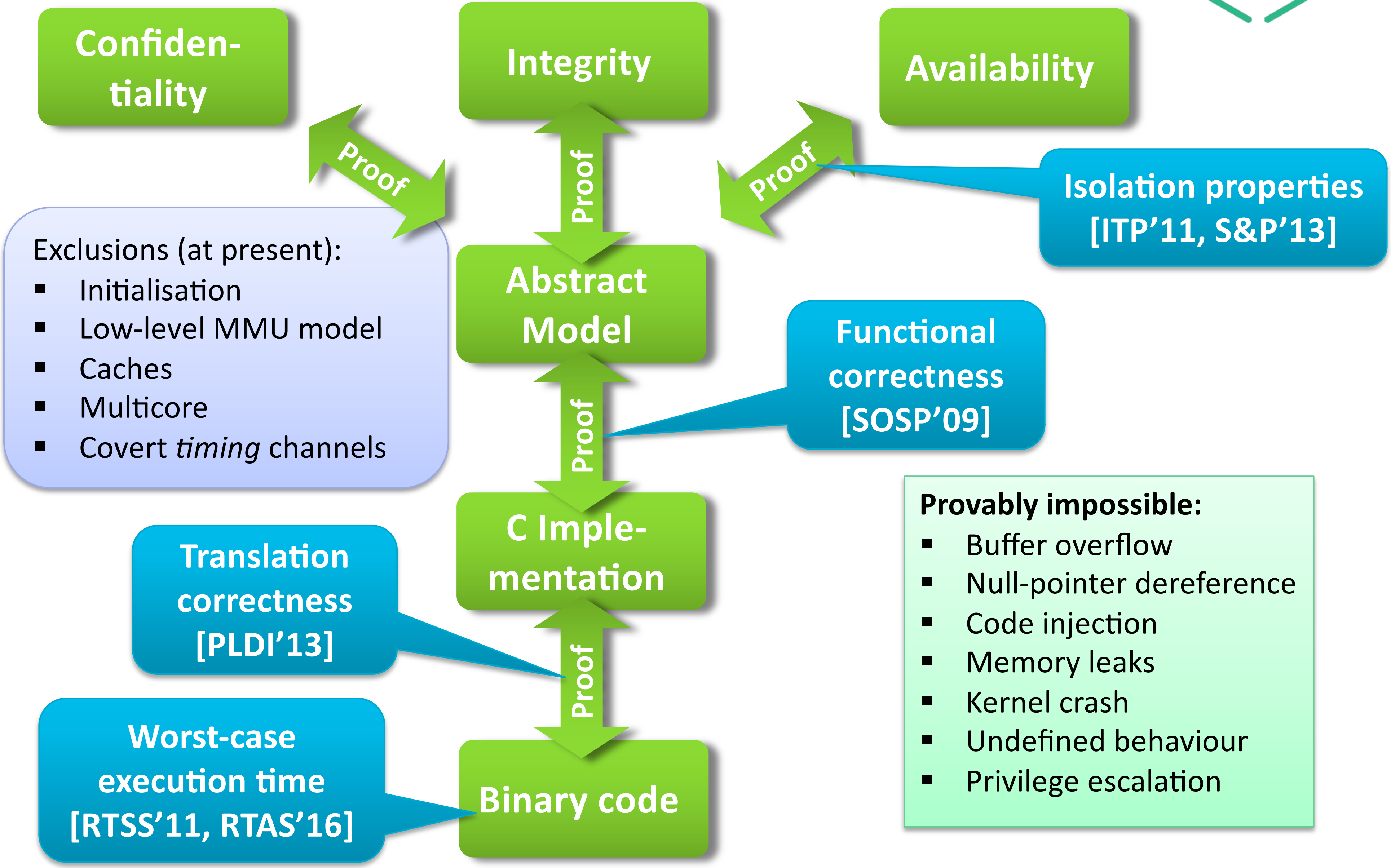
Privileged mode

20+ Years of L4 Microkernel R&D

seL4: The latest (and most advanced) member of the L4 microkernel family



Proving Trustworthiness of seL4



Confidentiality
Integrity
Availability
Proof
Proof
Proof
Isolation properties [ITP’11, S&P’13]
Abstract Model
Proof
Functional correctness [SOSP’09]
C Implementation
Proof
Translation correctness [PLDI’13]
Binary code
Worst-case execution time [RTSS’11, RTAS’16]

Exclusions (at present):

- Initialisation
- Low-level MMU model
- Caches
- Multicore
- Covert *timing* channels

Provably impossible:

- Buffer overflow
- Null-pointer dereference
- Code injection
- Memory leaks
- Kernel crash
- Undefined behaviour
- Privilege escalation

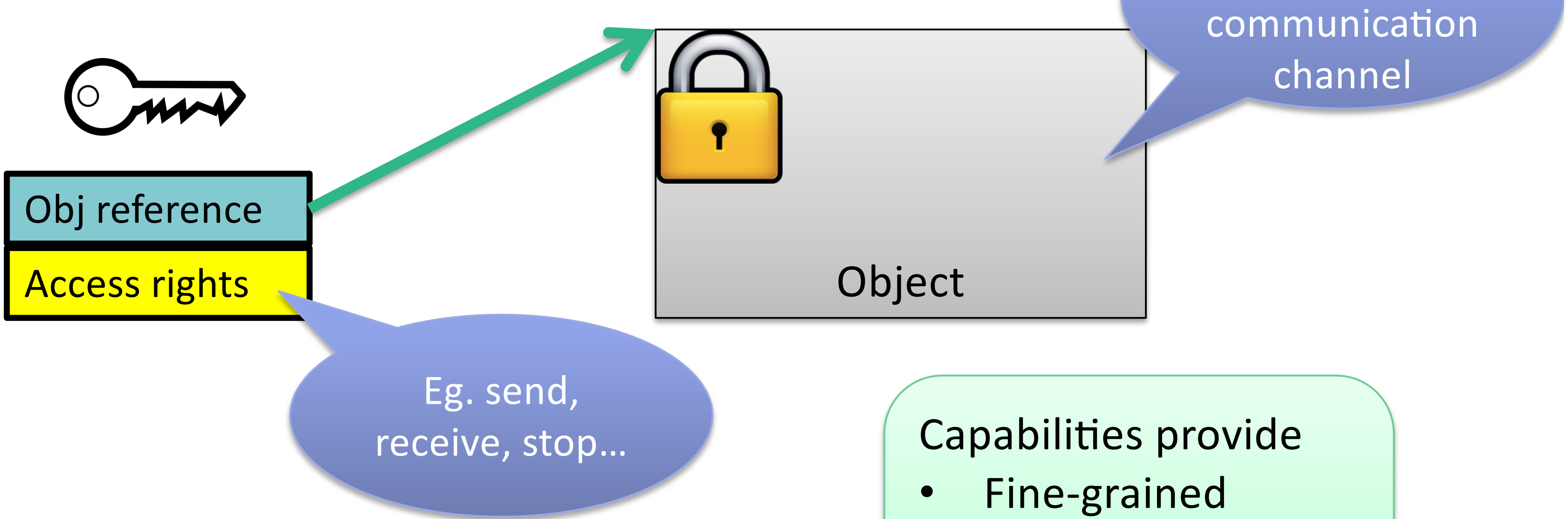
How Does seL4 Compare?

Feature	seL4	Other hypervisors, RTOSes, separation kernels
Performance	Fastest	2–10 × slower
Functional correctness	Proved	No Guarantee
Isolation	Proved	No Guarantee
Worst-case latency bounds	Sound & complete	Estimates only or no protection
Storage channel freedom	Proved	No Guarantee
Timing channel prevention	Low overhead (in progress)	None or High Overhead
Mixed-criticality support	Fully supported, high utilisation	Limited, resource-wastive

What’s Under the Hood?

Capability-Based Access Control

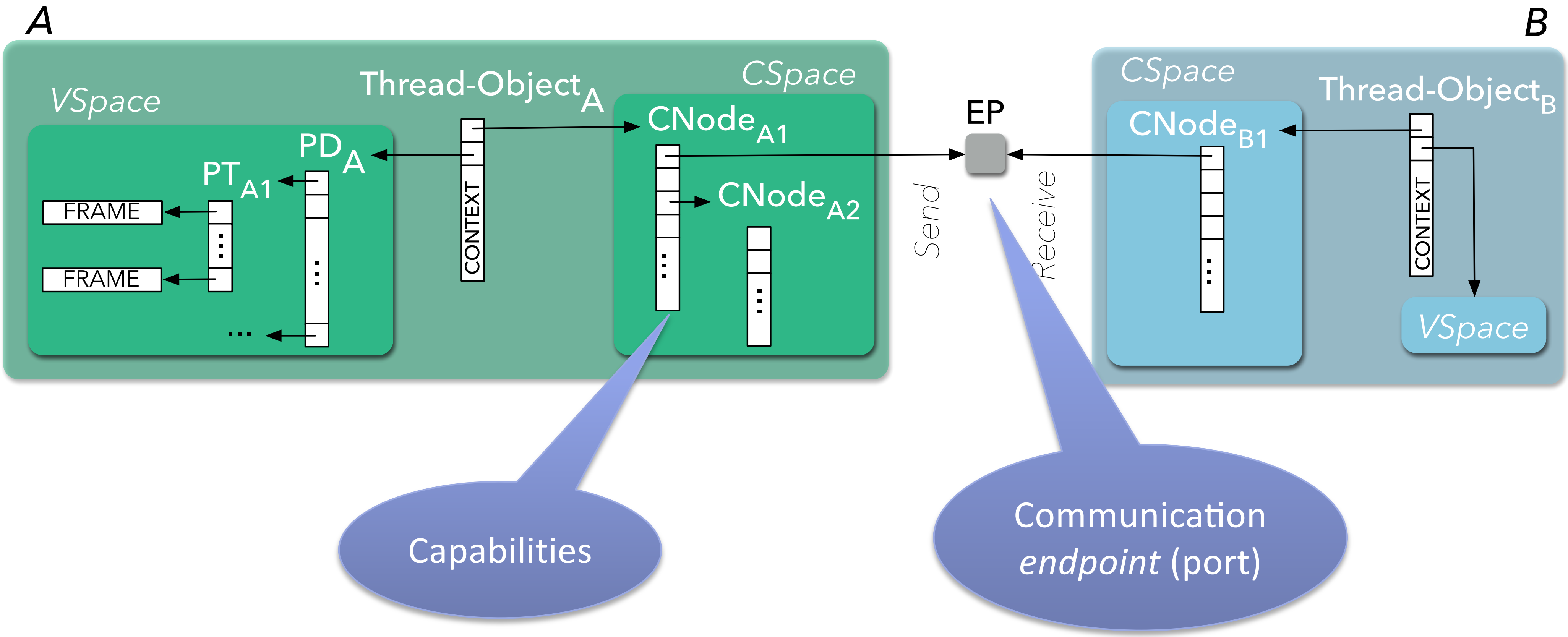
Capability = Access Token:
Prima-facie evidence of privilege



Any system call is invoking a capability:
err = method(cap, args);

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

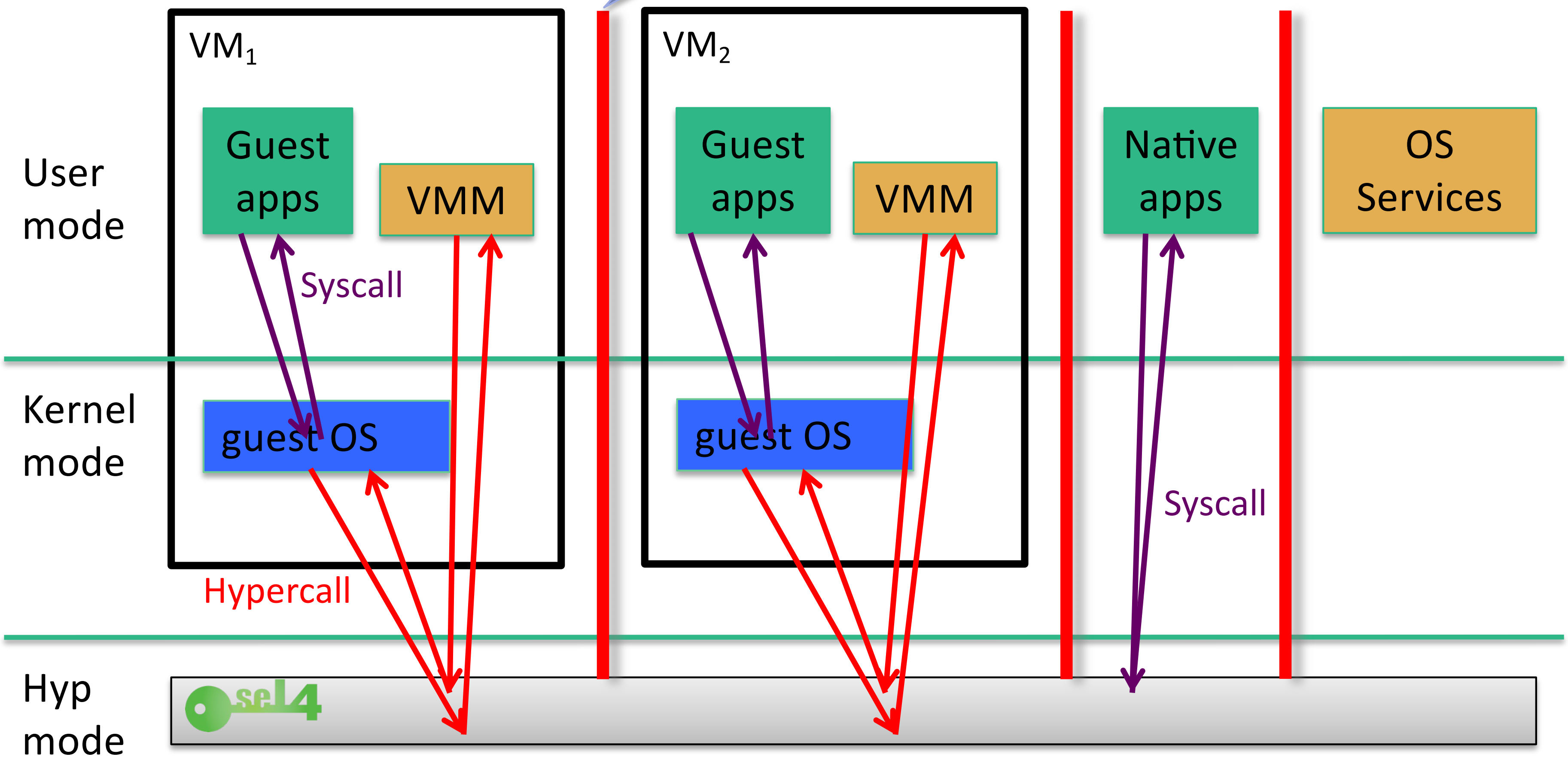
Example: Communicating Processes



Example: Virtualisation

Least privilege for all components

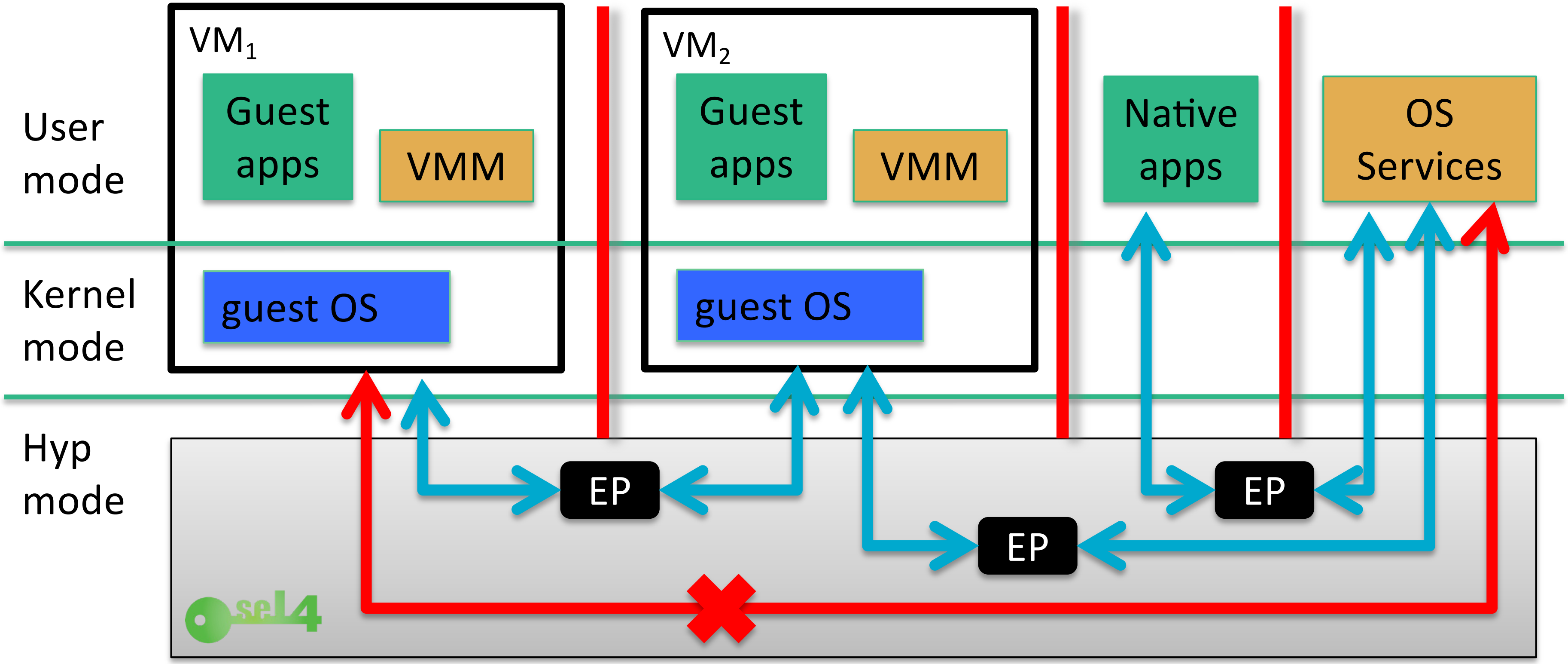
Only seL4 can bypass isolation!



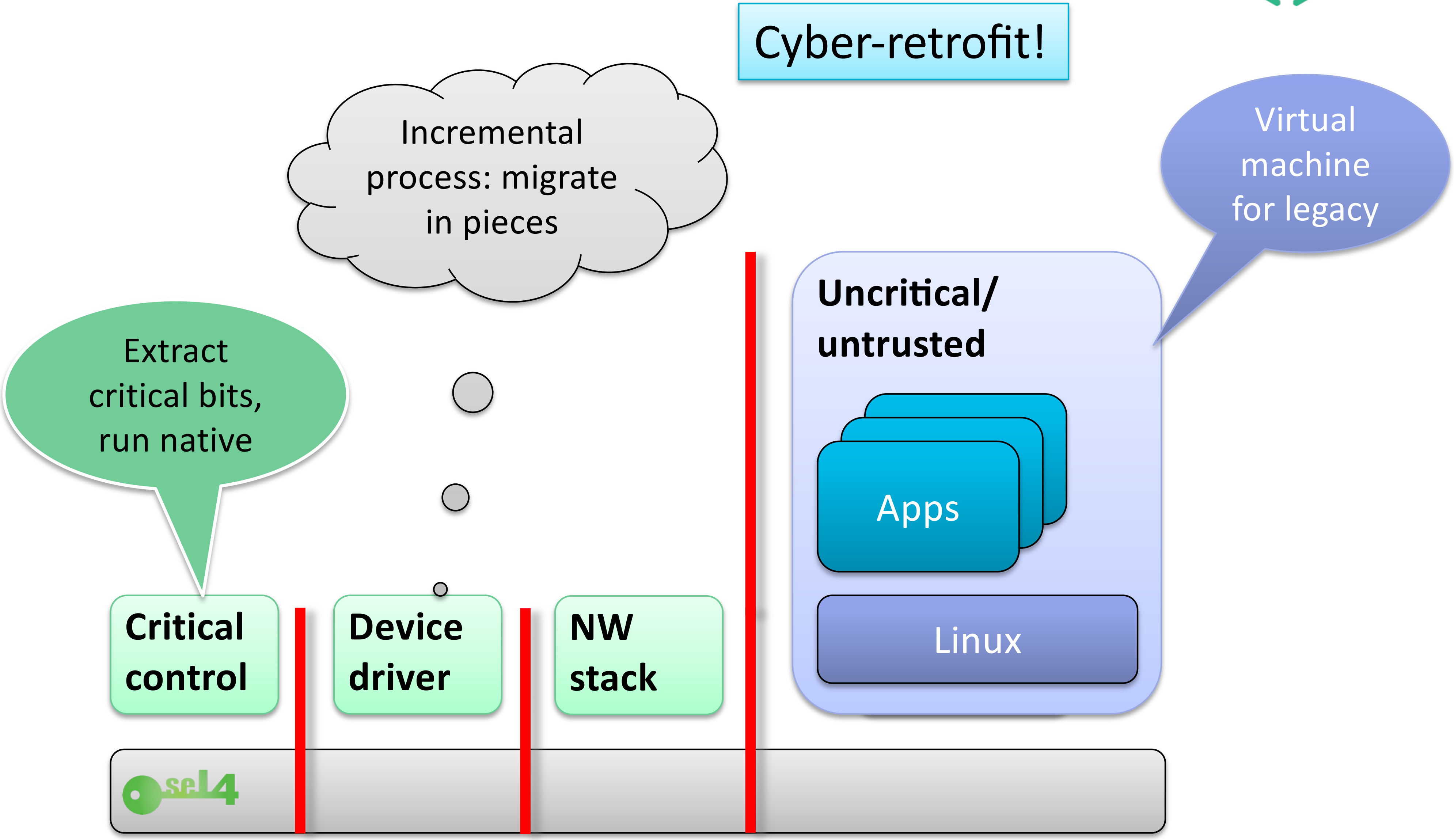
Cross-Partition Communication

No communication unless:

- **explicitly authorised**
- **via an Endpoint capability**



Result: Security by Architecture


Cyber-retrofit!
Incremental process: migrate in pieces
Virtual machine for legacy
Extract critical bits, run native
Uncritical/ untrusted
Apps
Linux
Critical control
Device driver
NW stack
seL4

Real-World Use

DARPA HACMS Program

Boeing Unmanned Little Bird

Retrofit existing system!

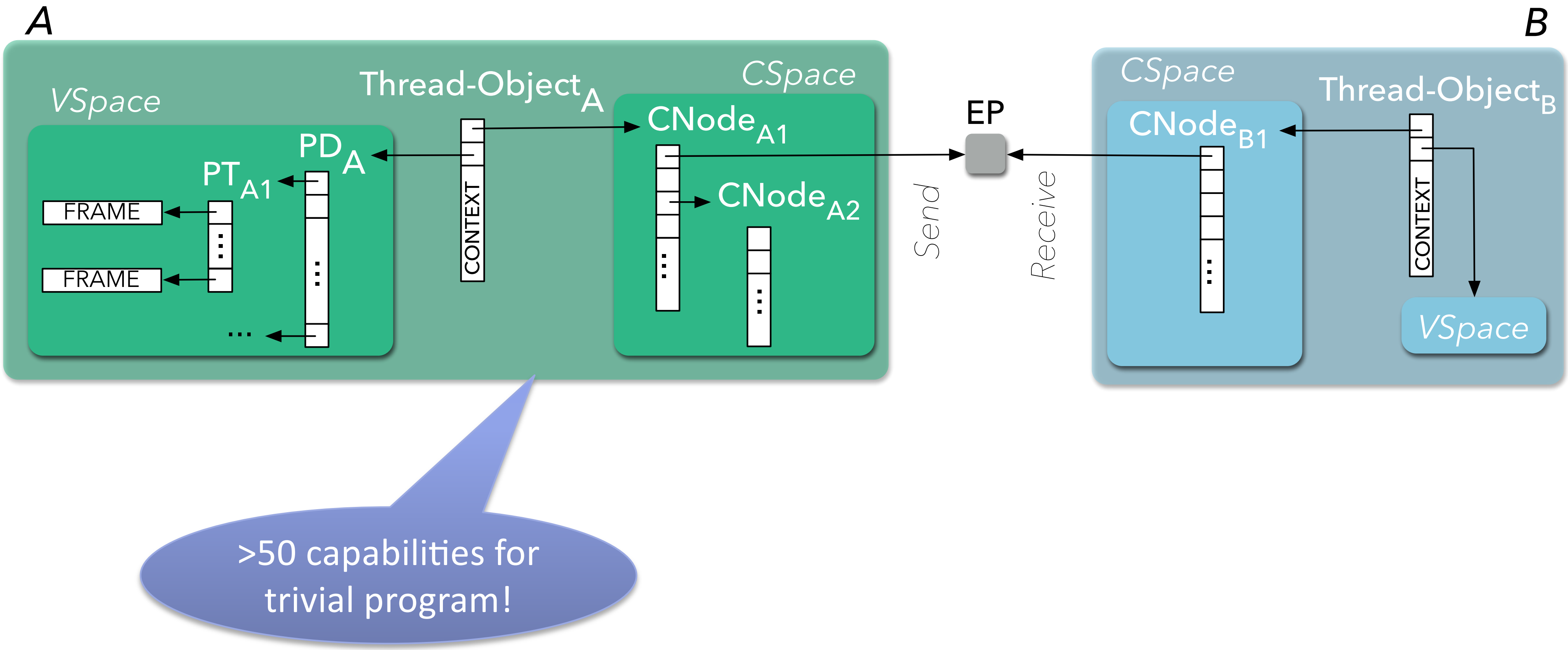
US Army Autonomous Trucks

SMACCMcopter Research Vehicle

Develop technology

TARDEC GVR-Bot

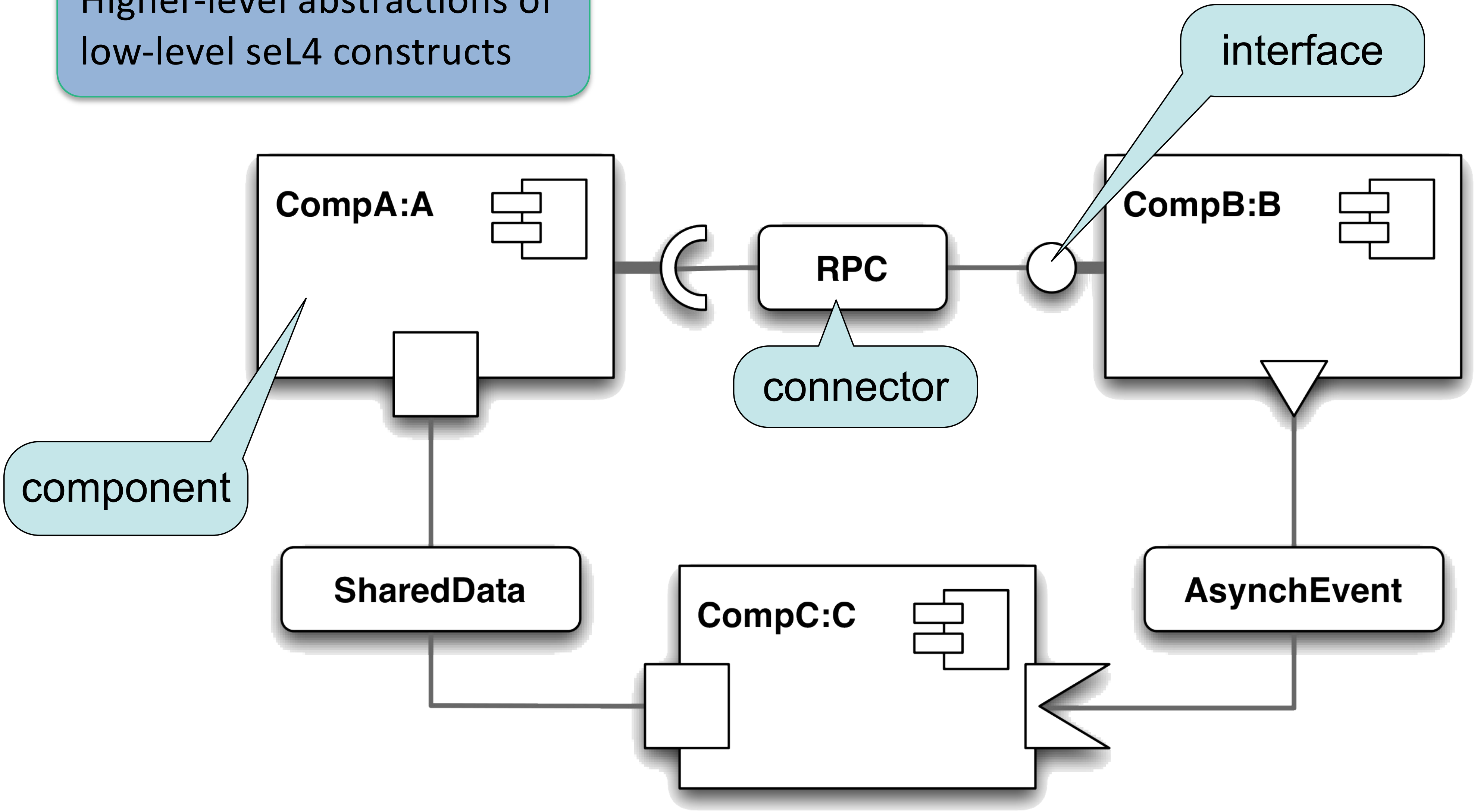
Issue: Capabilities are Low-Level



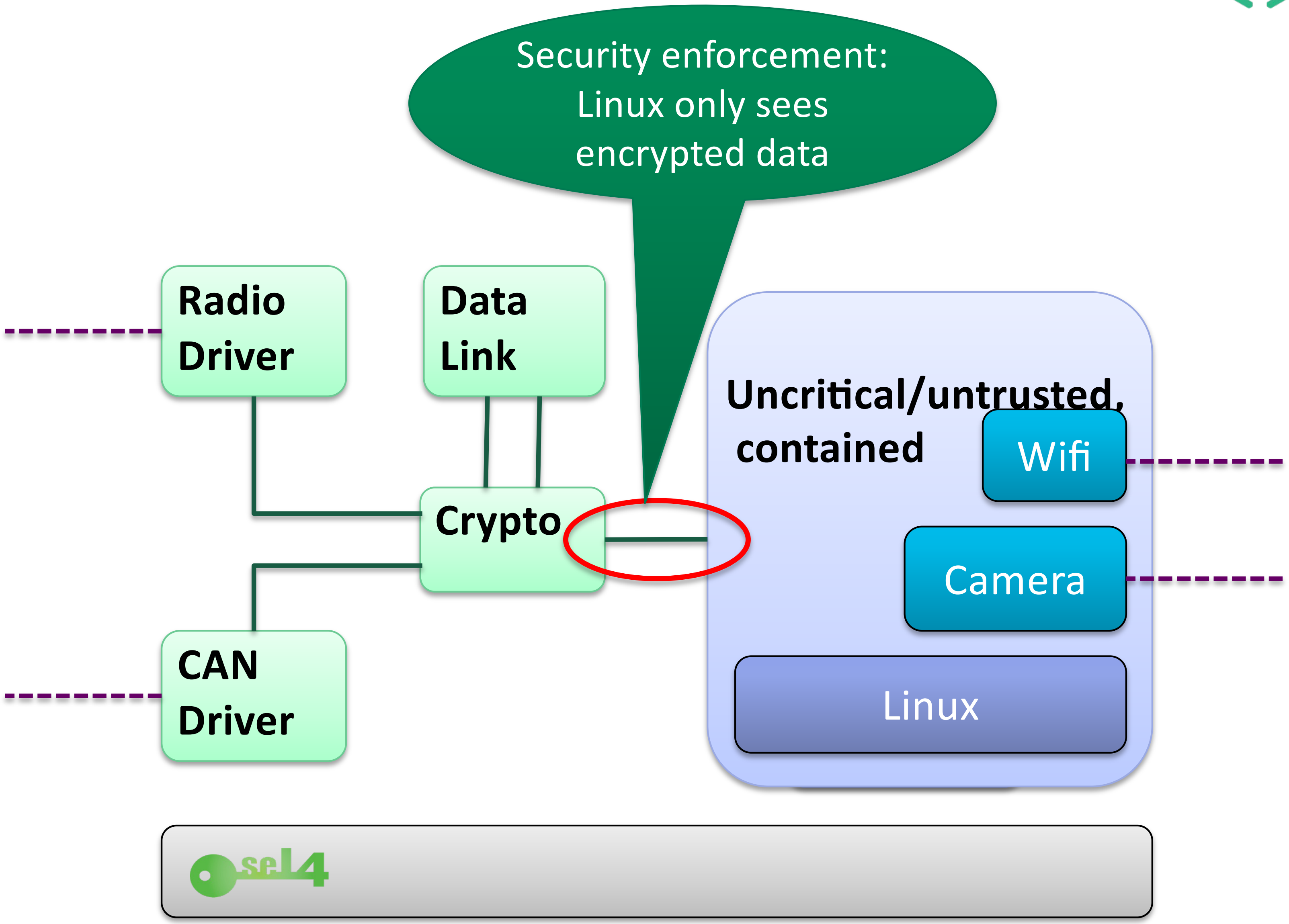
>50 capabilities for trivial program!

Component Middleware: CAmkES

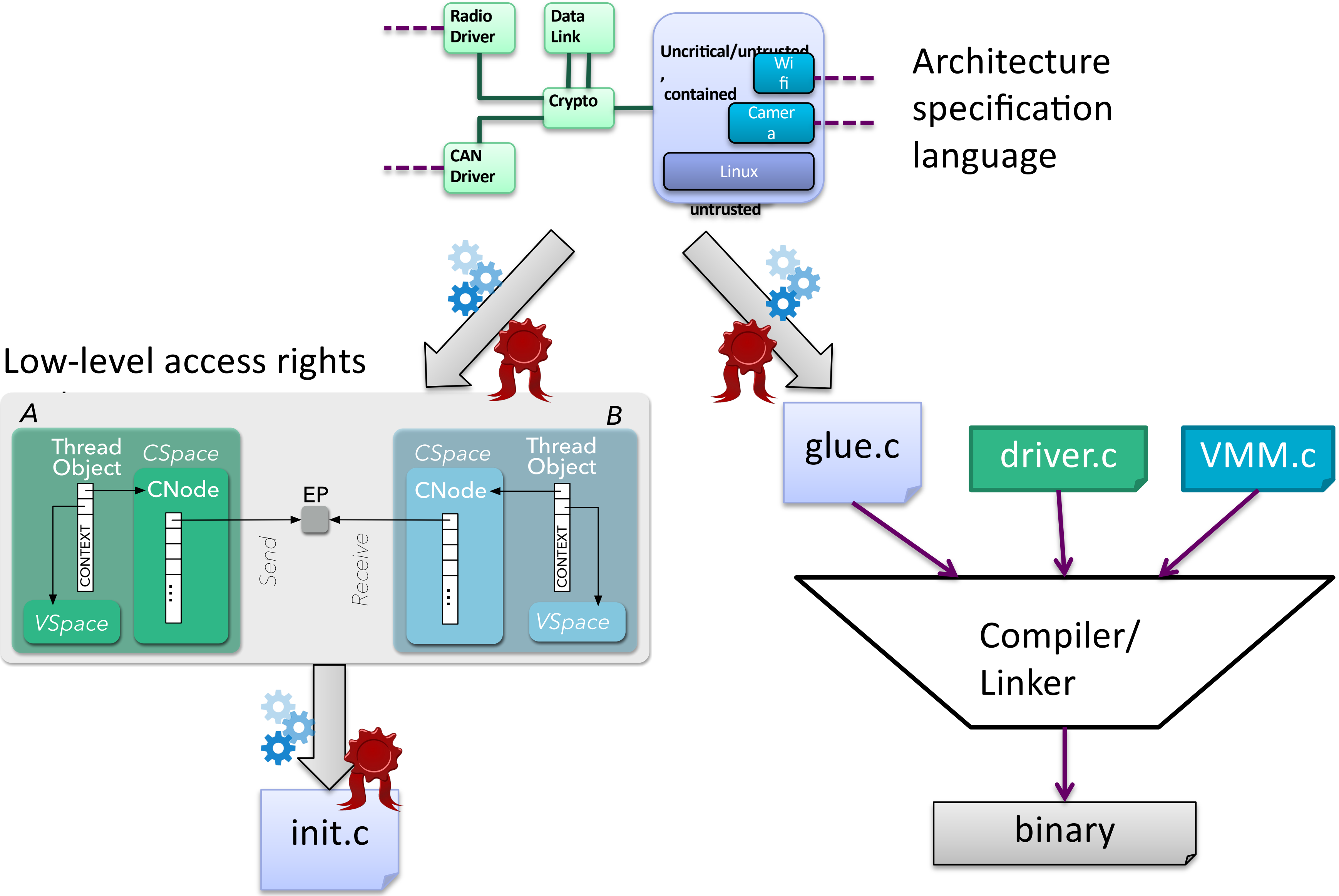
Higher-level abstractions of low-level seL4 constructs



Example: Simplified HACMS UAV



Enforcing the Architecture



Radio Driver

Data Link

Crypto

CAN Driver

Uncritical/untrusted, contained

Wi fi

Camer a

Linux

untrusted

Architecture specification language

Low-level access rights

A

Thread Object

CSpace

CNode

CONTEXT

VSpace

EP

Send

Receive

B

CSpace

CNode

Thread Object

CONTEXT

VSpace

init.c

glue.c

driver.c

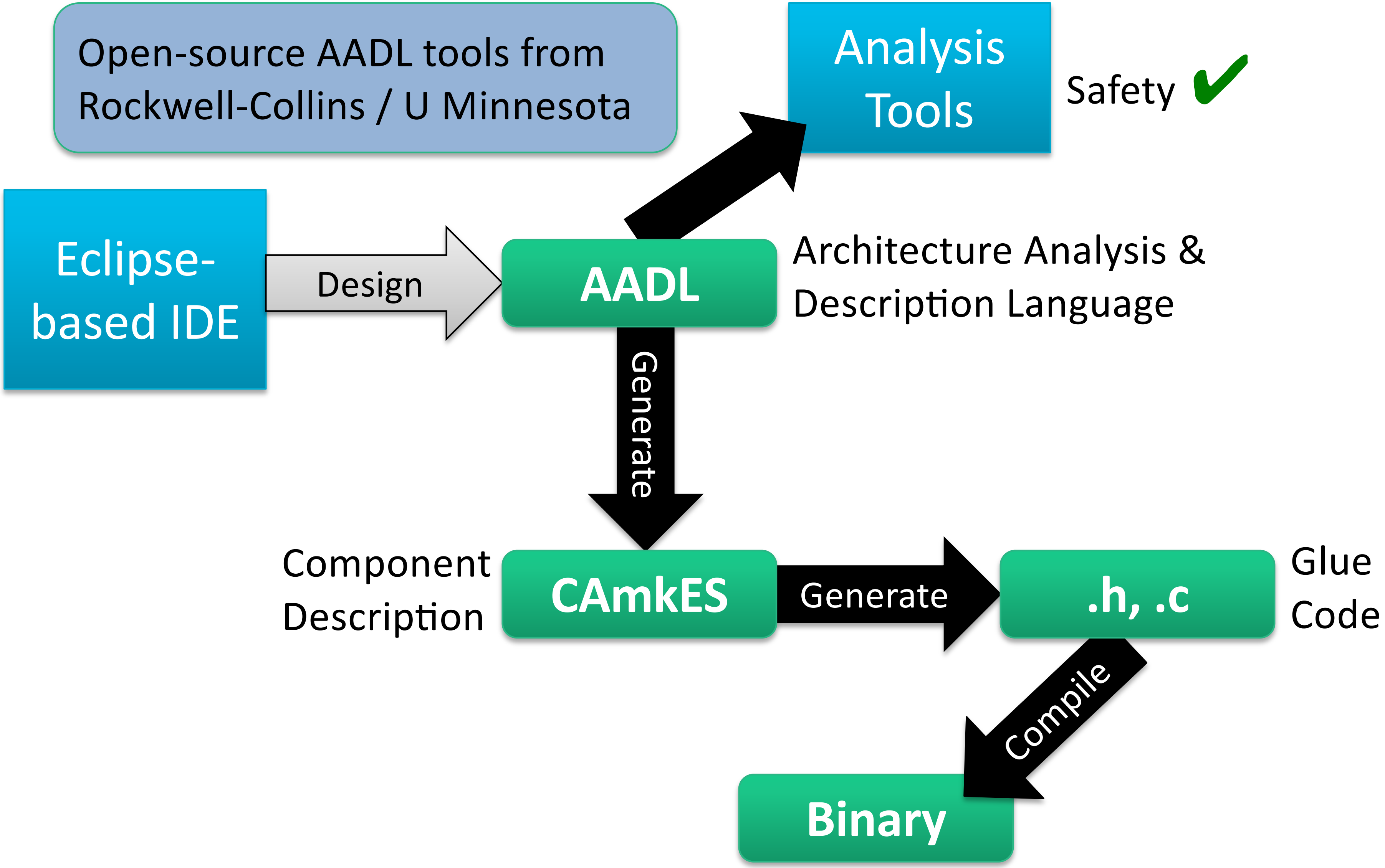
VMM.c

Compiler/ Linker

binary

Architecture Analysis

Open-source AADL tools from Rockwell-Collins / U Minnesota



Analysis Tools

Safety ✓

Eclipse-based IDE

Design

AADL

Architecture Analysis & Description Language

Generate

Component Description

CAmkES

Generate

.h, .c

Glue Code

Compile

Binary

Real-World Use

Courtesy Boeing, DARPA

seL4 Military-Grade Security

Cross-Domain Desktop Compositor

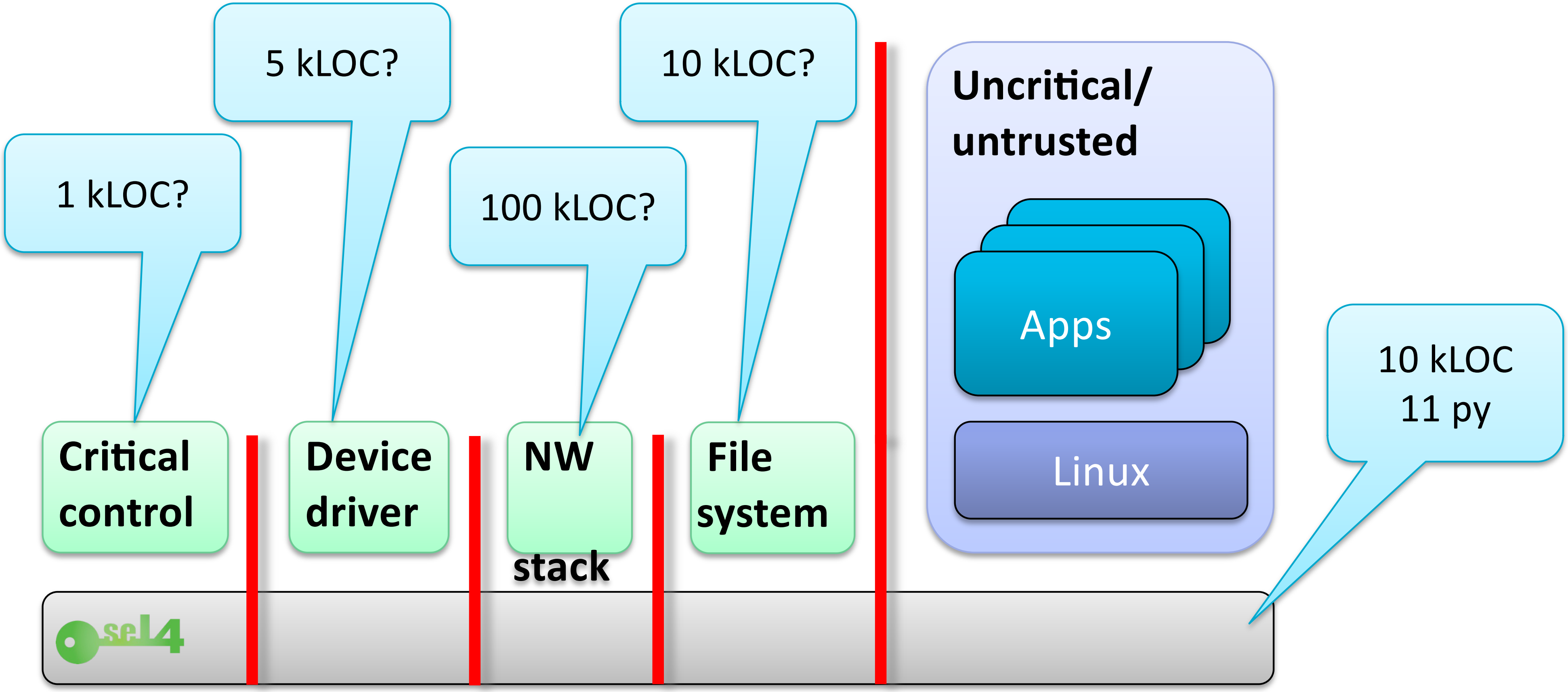
Multi-level secure terminal

- Successful defence trial in AU
- Evaluated in US, UK, CA
- Formal security evaluation soon

Pen10.com.au crypto communication device undergoing formal security evaluation in UK

Beyond the Kernel: Verifying Userland

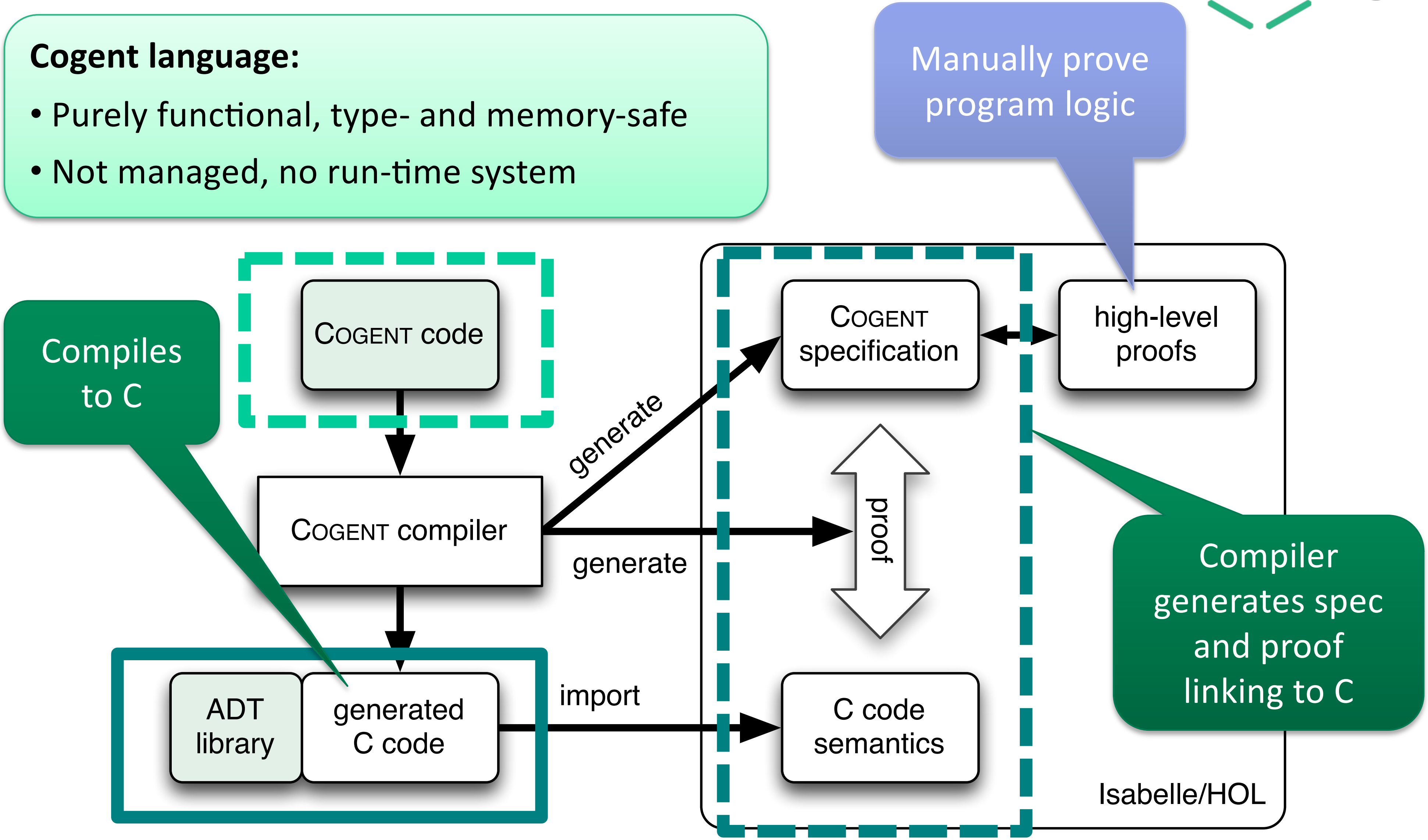
Beyond Kernel: Trustworthy Userland



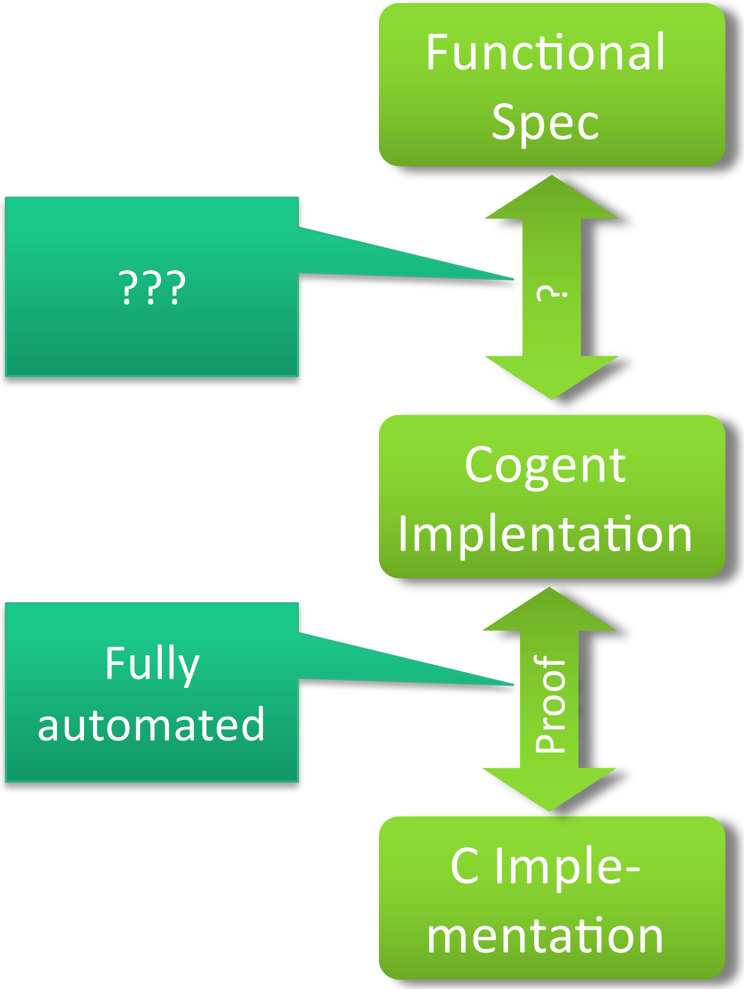
Cogent: Code + Proof Co-Generation

Cogent language:

- Purely functional, type- and memory-safe
- Not managed, no run-time system



Dependable & Affordable Systems



Functional Spec

???

?

Cogent Implentation

Fully automated

Proof

C Imple-mentation

Dependability-cost tradeoff:

- Reduced faults through safe language
- Property-based testing (QuickCheck)
- Model checking
- Full functional correctness proof

Work in progress:

- File-system case study
- Extending to network stacks and device drivers
- More domain-specific language layer

Trustworthy Systems Are Possible!

Thank you, awesome Trustworthy Systems Team!

Thank you, Audience!

Military-Grade Security for You!

Security is no excuse for poor performance!

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

Embedded Systems Week, Seoul 2017

http://sel4.systems

Temporal Isolation

Core Mechanism: Budget

Thread scheduling parameters

- P: Priority
- SC: Scheduling context capability

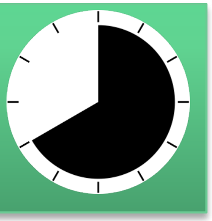
- Integrates with spatial access control
- Supports reasoning about isolation

Integrity property:

- **Observe** priorities for runnable threads
- Thread not runnable when out of budget

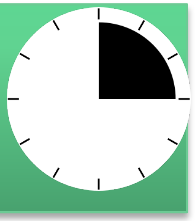
C = 2

T = 3



C = 250

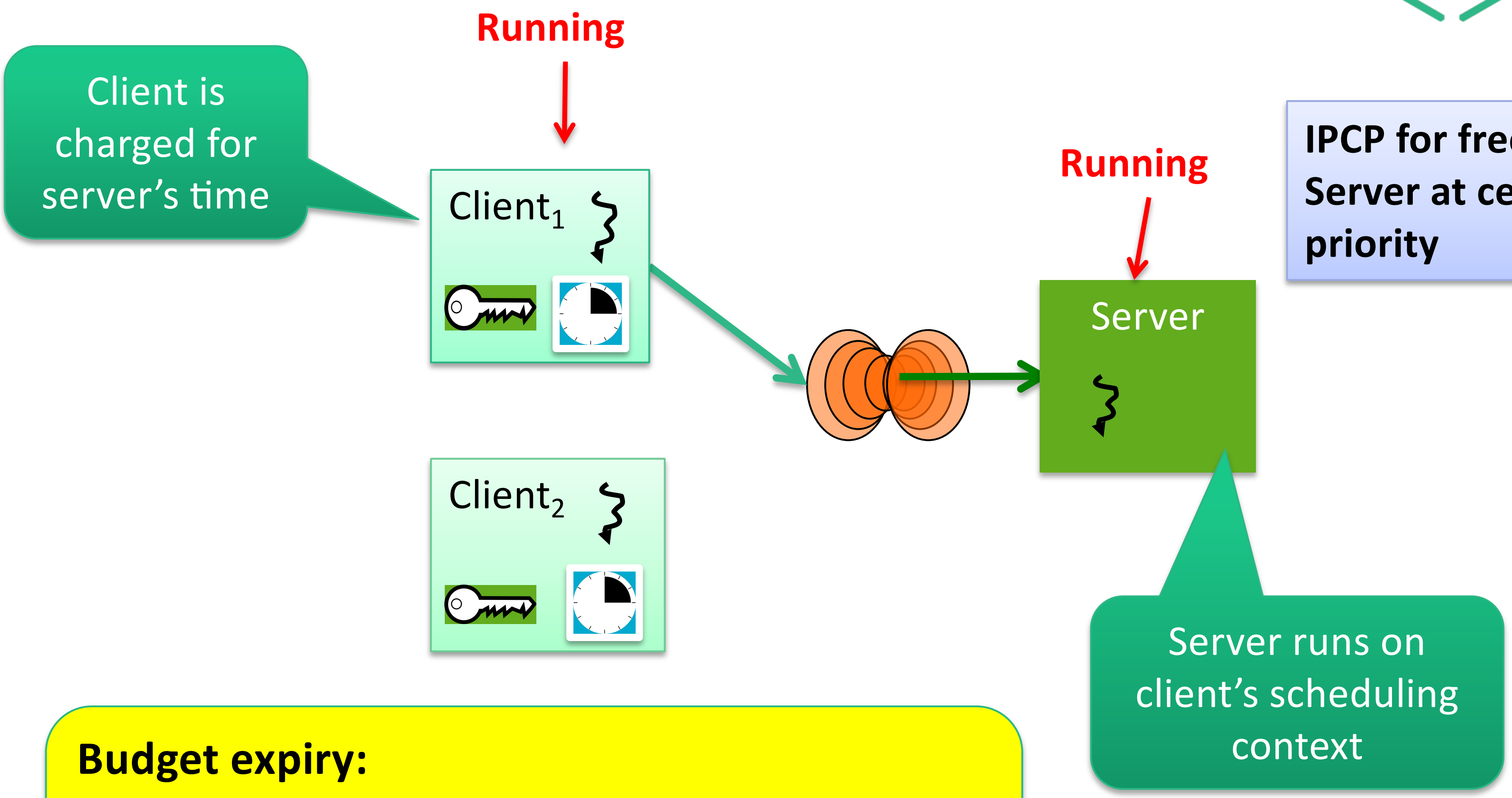
T = 1000



Scheduling context object

- T: period
- C: budget (≤ T)

Critical Sections: Resource Server



Client is charged for server's time

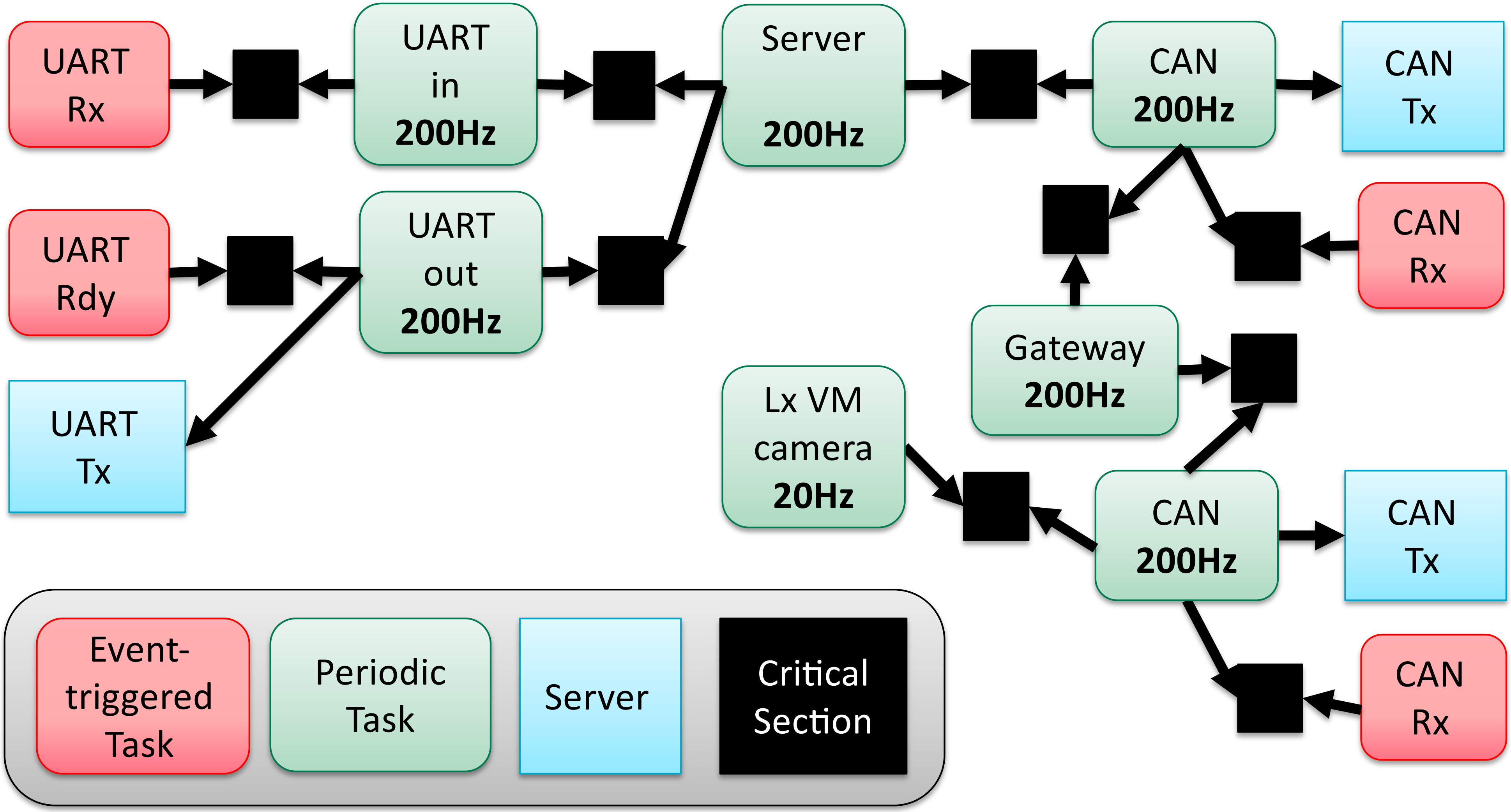
IPCP for free: Server at ceiling priority

Server runs on client's scheduling context

Budget expiry:

- Timeout-fault generates message to handler
- Can implement arbitrary policy

Example: SMACCMcopter


UART Rx
UART in 200Hz
Server 200Hz
CAN 200Hz
CAN Tx
UART Rdy
UART out 200Hz
CAN Rx
UART Tx
Gateway 200Hz
Lx VM camera 20Hz
CAN 200Hz
CAN Tx
CAN Rx
Event-triggered Task
Periodic Task
Server
Critical Section