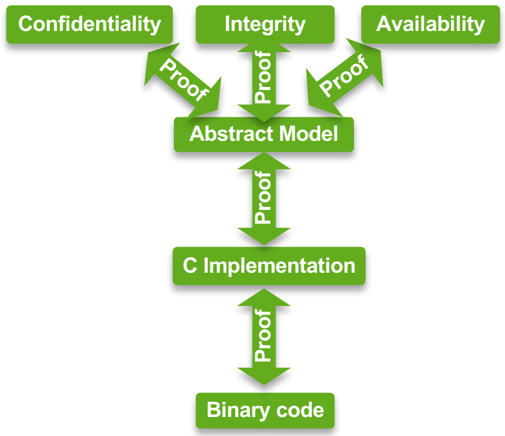
School of Computer Science & Engineering

**COMP9242 Advanced Operating Systems**

# 2025 T3 Week 10 Part 1

## seL4 and LionsOS

@GernotHeiser



# Copyright Notice

# August 2009

A NICTA bejelentette a világ első, formális módszerekkel igazolt,

Slashdot

Stories Recent Popular Searc

Slashdot is powered by your subm

Technology: World's Firs

Posted by Soulskill on Thursday Aug
from the wait-for-it dept.

An anonymous reader writes

> "Operating systems usually have
> and so forth are known by almos
> to prove that a particular OS ker
> formally verified, and as such it
> researchers used an executable
> the Isabelle theorem prover to ge
> matches the executable and the

Does it run Linux? "We're pleased to say that it does. Presently, we have a para-virtualized ver

eredményeképpen pedig egy olyan megbízhatóságot kapnak a szoftvertől, amely e

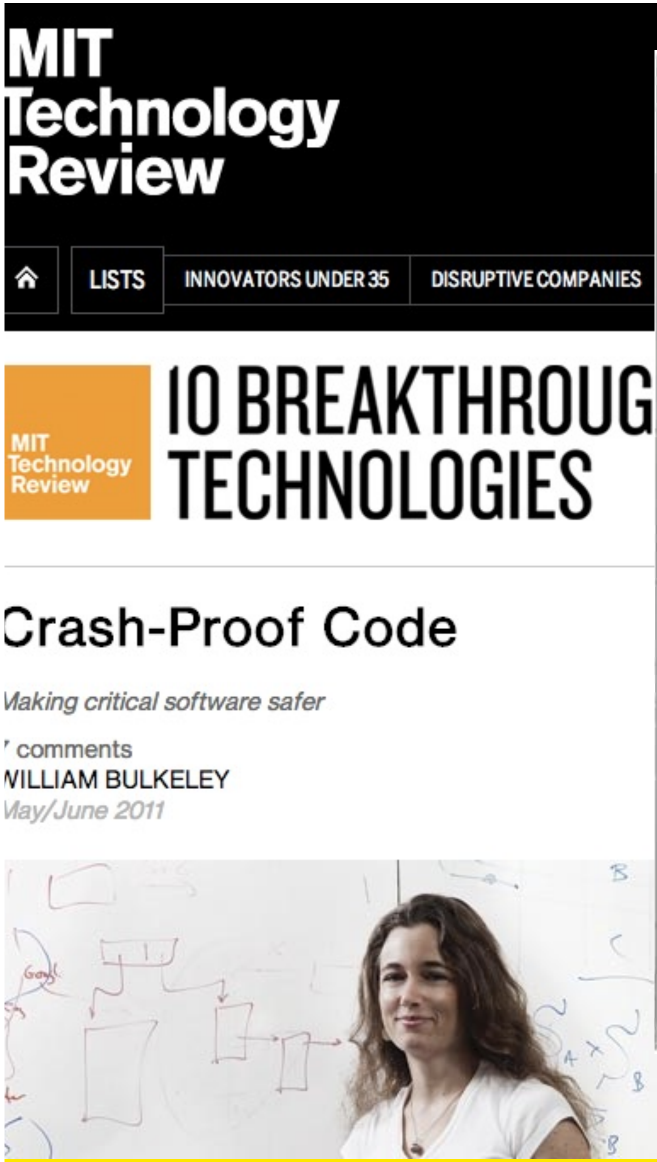
New Scientist
Saturday 29/8/2009
Page: 21
Section: General News
Region: National
Type: Magazines Science / Technology
Size: 196.31 sq.cms.
Published: -----S-

## The ultimate way to keep your computer safe from harm

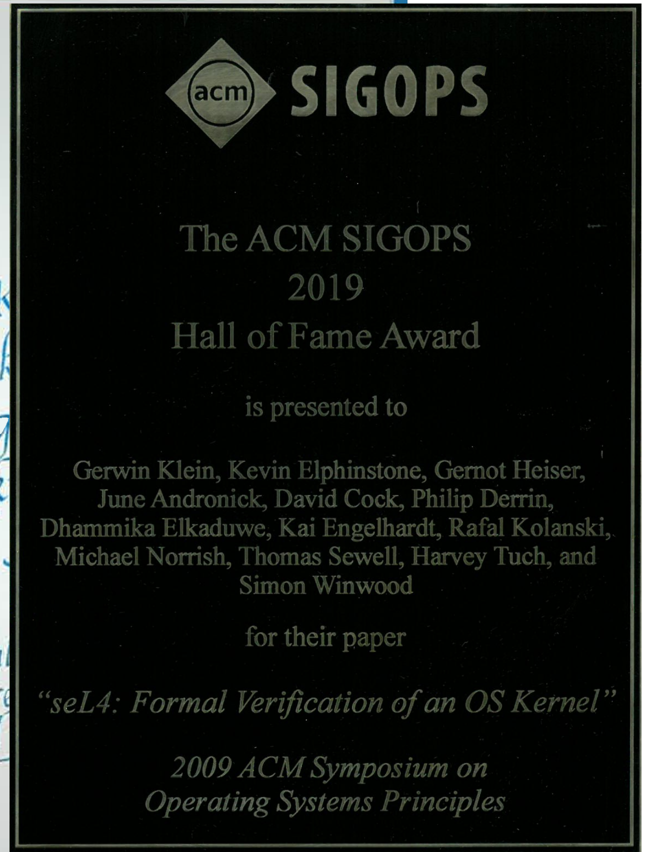
FLAWS in the code, or "kernel", that sits at the heart of modern computers leave them prone to occasional malfunction and vulnerable to attack by worms and viruses. So the development of a secure general-purpose microkernel could pave the

just mathematics, and you can reason about them mathematically," says Klein.

His team formulated a model with more than 200,000 logical steps which allowed them to prove that the program would always behave as its


MIT Technology Review
LISTS
INNOVATORS UNDER 35
DISRUPTIVE COMPANIES
MIT Technology Review
10 BREAKTHROUG
TECHNOLOGIES
Crash-Proof Code
Making critical software safer
comments
WILLIAM BULKELEY
May/June 2011
accepted


acm
ACM
Software System Award
2022
PRESENTED TO
Gernot Heiser, Gerwin Klein, K
June Andronick, David Cock
Dhammika Elkaduwe, Kai Eng
Toby Murray, Rafal Kolansk
Thomas Sewell, Harvey Tuch,
For the development of the first industrial
performance operating system kernel to have
complete, mechanically checked proof of full
ACM PRESIDENT
AWARDS COMMITTEE CO-CHAIR
AWARDS COMMITTEE CO-CHAIR


acm SIGOPS
The ACM SIGOPS
2019
Hall of Fame Award
is presented to
Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood
for their paper
"seL4: Formal Verification of an OS Kernel"
2009 ACM Symposium on Operating Systems Principles



UNSW SYDNEY

# Today's Lecture

- Assurance and verification
  - Common Criteria
  - Formal verification
- seL4
  - Design principles & verification
  - Limitations & present status
- Security impact of OS design
- seL4 strengths & weaknesses
- seL4 Microkit
- LionsOS

# Assurance and Verification

# Refresher: Assurance and Formal Verification

- **Assurance**:
  - systematic evaluation and testing
  - essentially an intensive and onerous form of quality assurance

Assurance and formal verification aim to establish correctness of
- mechanism design
- mechanism implementation

- **Formal verification**:
  - mathematical proof

- **Certification**: independent examination
  - confirming that the assurance or verification was done right

# Assurance: Substantiating Trust

- Specification
  - Unambiguous description of desired behaviour
- System design
  - Justification that it meets specification
- Implementation
  - Justification that it implements the design
- Maintenance
  - Justifies that system use meets assumptions

Informal (English) or formal (maths)

Compelling argument or formal proof

Code inspection, rigorous testing, proof

# Common Criteria

**Common Criteria for IT Security:**

- ISO standard [ISO/IEC 15408, 99], for general use
- Evaluates QA used to ensure systems meet their requirements
- Developed out of the famous US DOD "Orange Book":
  *Trusted Computer System Evaluation Criteria* [1985]

# CC: Evaluation Assurance Levels

Thoroughness, cost ↓

| Level | Requirements | Specification | Design | Implementation |
|---|---|---|---|---|
| EAL1 | not evaluated | **Informal** | not eval | not evaluated |
| EAL2 | not evaluated | **Informal** | **Informal** | not evaluated |
| EAL3 | not evaluated | **Informal** | **Informal** | not evaluated |
| EAL4 | not evaluated | **Informal** | **Informal** | not evaluated |
| EAL5 | not evaluated | **Semi-Formal** | **Semi-Formal** | **Informal** |
| EAL6 | **Formal** | **Semi-Formal** | **Semi-Formal** | **Informal** |
| EAL7 | **Formal** | **Formal** | **Formal** | **Informal** |

# COTS OS Certifications

- EAL3:
  - 2010 Mac OS X (10.6)
- EAL4:
  - 2003: Windows 2000
  - 2005: SuSE Enterprise Linux
  - 2006: Solaris 10 (EAL4+)
    - against CAPP (an EAL3 PP!)
  - 2007: Red Hat Linux (EAL4+)
- EAL6:
  - 2008: Green Hills INTEGRITY-178B (EAL6+)
    - relatively simple PPC-based hardware platform
- EAL7:
  - 2019: Prove & Run PROVENCORE
    - TEE OS for Arm TrustZone

Get regularly hacked!

# Common Criteria Limitations

Effectively dead in 5-Eyes defence

- Very expensive
  - rule of thumb: EAL6+ costs $1K/LOC [Green Hills] design-implementation-evaluation-certification
- Too much focus on development process
  - rather than the product that was delivered
  - "evaluating paperwork, not the product" [N Daughety, AFRL]
- Lower EALs of little practical use for OSes
  - c.f. COTS OS EAL4 certifications
- Commercial Evaluation Facilities licenses rarely revoked
  - Leads to potential "race to the bottom" [Anderson & Fuloria, 2009]

# Formal Verification

## Prove properties about a mathematical model of a system

**Automatic ("push-button") techniques**
- **Model checking / abstract interpretation / SMT**
- **Systematic exploration of system state space**

❑ Cannot generally prove code correct
  - Proves specific properties
  - Functional correctness in simple cases

❑ Generally have to
  - over-approximate (false positives), or
  - under-approximate (false negatives, unsou

❑ Suffers state-space explosion

✓ Can scale to large code bases

**Interactive techniques:**
- **Theorem proving**
- **Proofs about state spaces**

✓ Can deal with large (even infinite) state spaces

✓ Can prove functional correctness against a spec

❑ Very labour-intensive

Recent work automatically proved functional correctness of simple systems using SMT solvers [Hyperkernel, SOSP'17; Atmosphere, SOSP'25]

# seL4 Design Principles

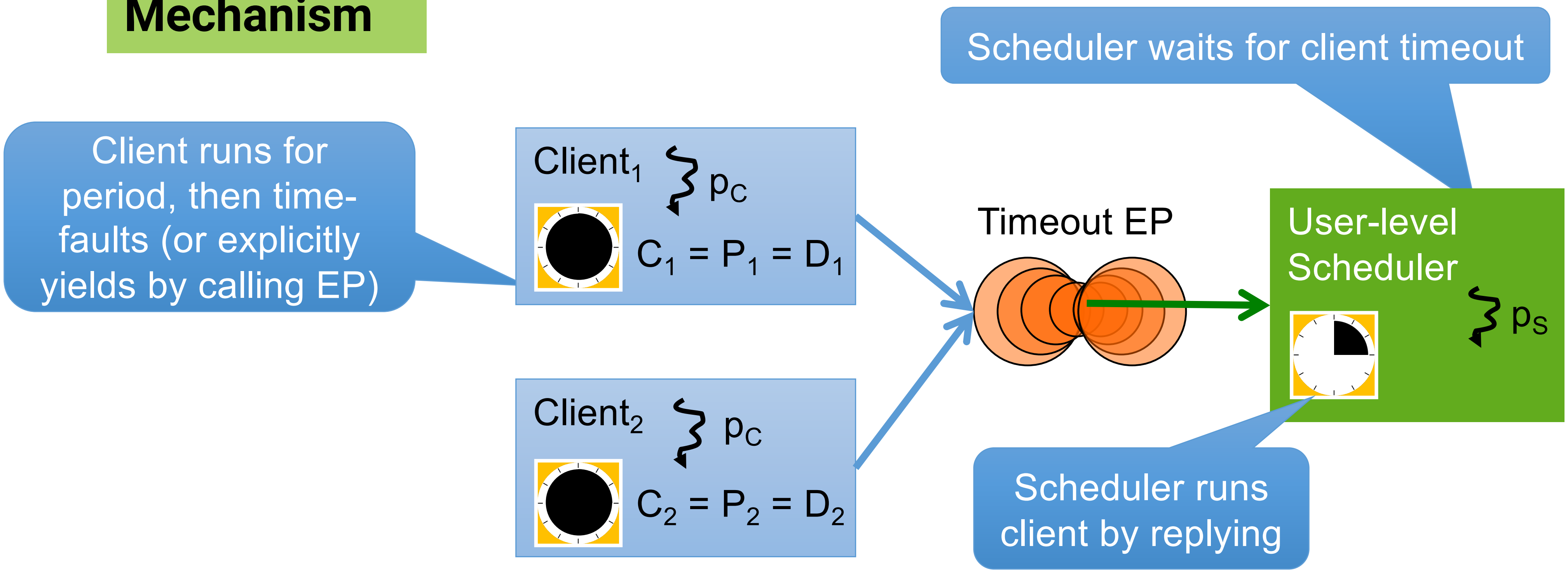
- Fully delegable access control via capabilities
- All resource management is subject to user-defined policies
  - Applies to kernel resources too!
- Performance on par with best-performing L4 kernels
  - Prerequisite for real-world deployment!
- Suitability for real-time use
  - Important for safety-critical systems
- Suitable for *formal verification*
  - Requires small size, avoid complex constructs

Largely in line with traditional L4 approach!

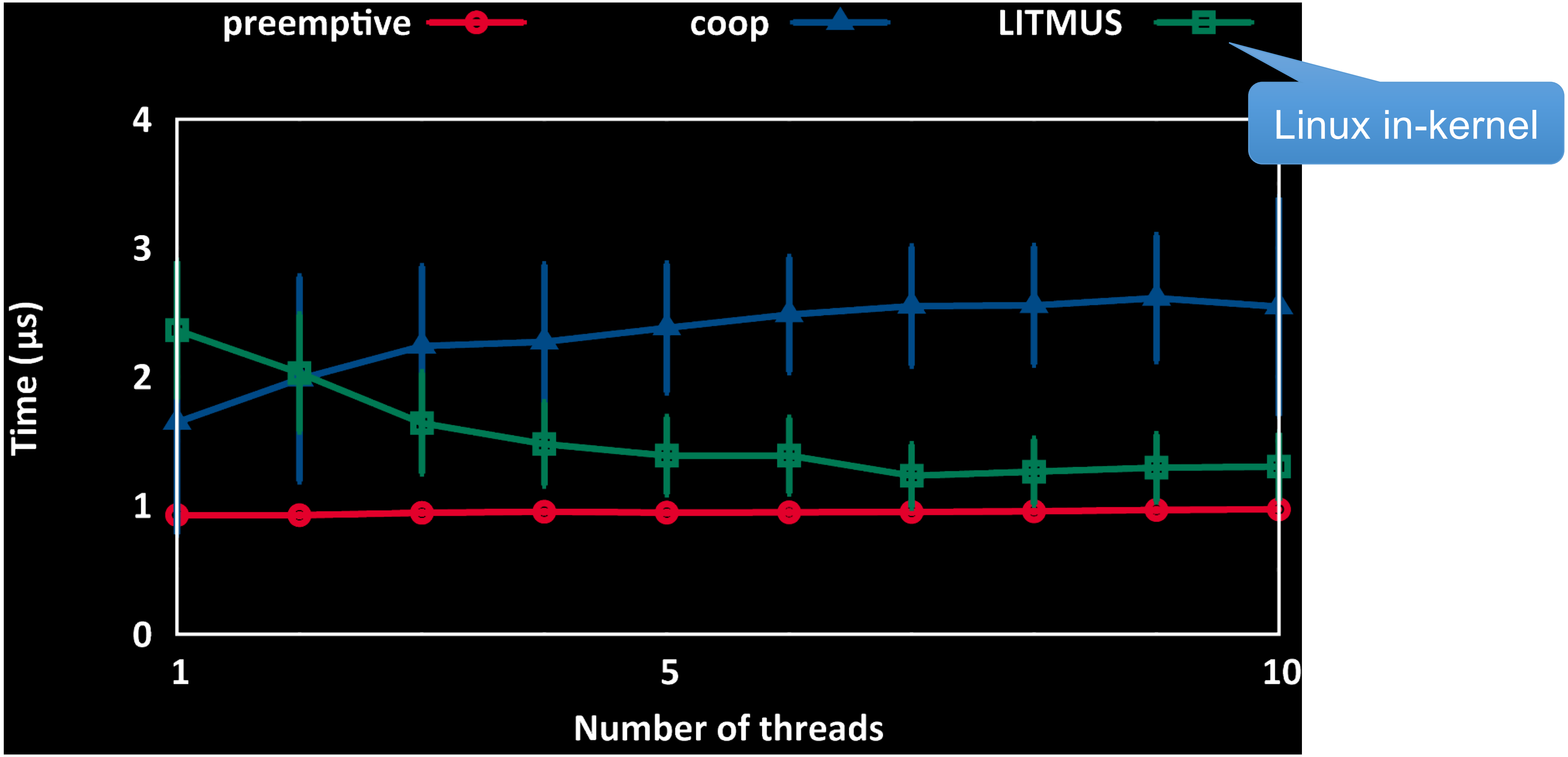
# Isn't a Fixed-Prio Scheduler Policy?

**Prios + SCs = Mechanism**

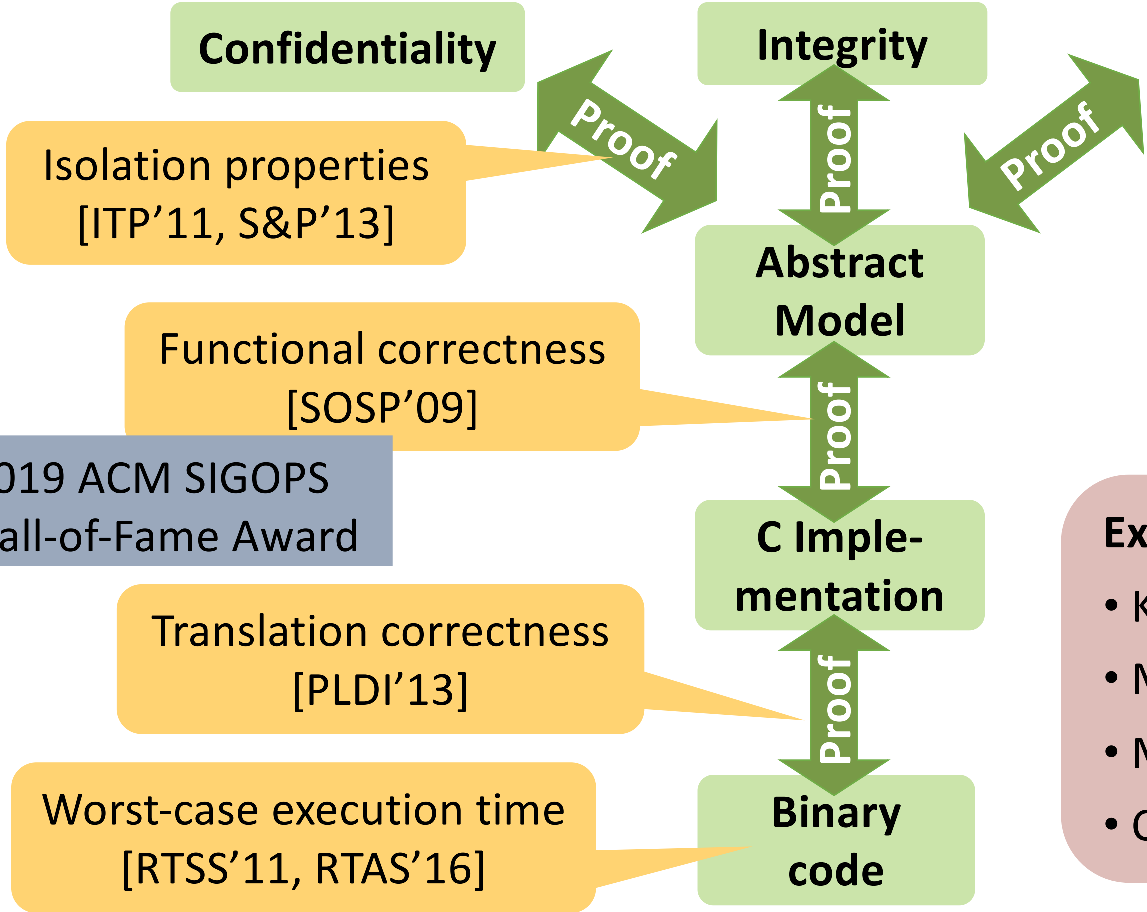
**Implementing scheduling policy at user level**



# User-Level EDF Scheduler Performance



# Proving Security and Safety (Armv6/7)




Still most compre-hensive verification

Still only verified capability-based OS

**Exclusions (at present, Armv7):**

- Kernel initialisation not yet verified
- MMU & caches modelled abstractly
- Multicore not yet verified
- Covert *timing* channels not precluded

# Security Is No Excuse For Bad Performance!

| Cost | seL4 | Fiasco.OC | Zircon |
|---|---|---|---|
| IPC RT latency (cycles) | 986 | 2717 | 8157 |
| Mand. HW cost (cycles) | 790 | 790 | 790 |
| Abs. overhead (cycles) | 196 | 1972 | 7367 |
| Rel. overhead (%) | 25 | 240 | 930 |

Hardware cost dominates

SW overheads dominate

*Round-trip, cross-address-space IPC on x64 (Intel Skylake)*

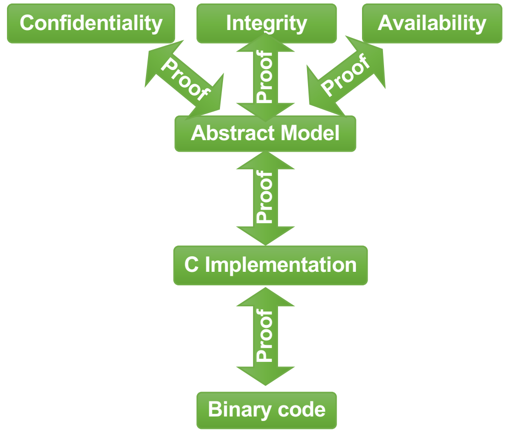
| Operation | 1-way | RT |
|---|---|---|
| SYSCALL | 82 | 164 |
| SWAPGS | 2×26 | 104 |
| Switch PT | 186 | 372 |
| SYSRET | 75 | 150 |
| **Total** | **395** | **790** |

**Source:** Zeyu Mi, Dingji Li, Zihan Yang, Xinran Wang, Haibo Chen: "SkyBridge: Fast and Secure Inter-Process Communication for Microkernels", EuroSys, April 2019

# Limitations

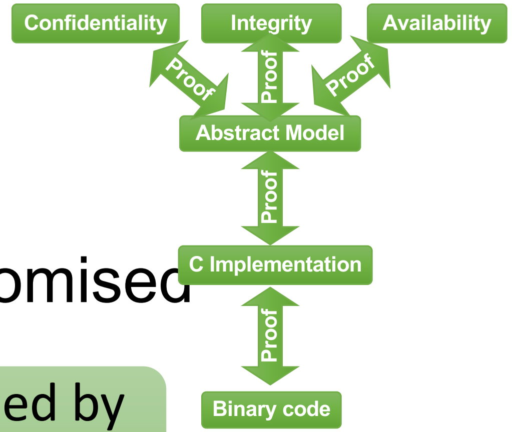
# Verification Assumptions

1. Hardware behaves as expected
   - Formalised hardware-software contract (ISA)
   - Hardware implementation free of bugs, Trojans, …
2. Spec matches expectations
   - Can only prove "security" if specify what "security" means
   - Spec may not be what we think it is
3. Proof checker is correct
   - Isabel/HOL checking core that validates proofs against logic




With binary verification do **not** need to trust C compiler!

# seL4 Present Verification Limitations



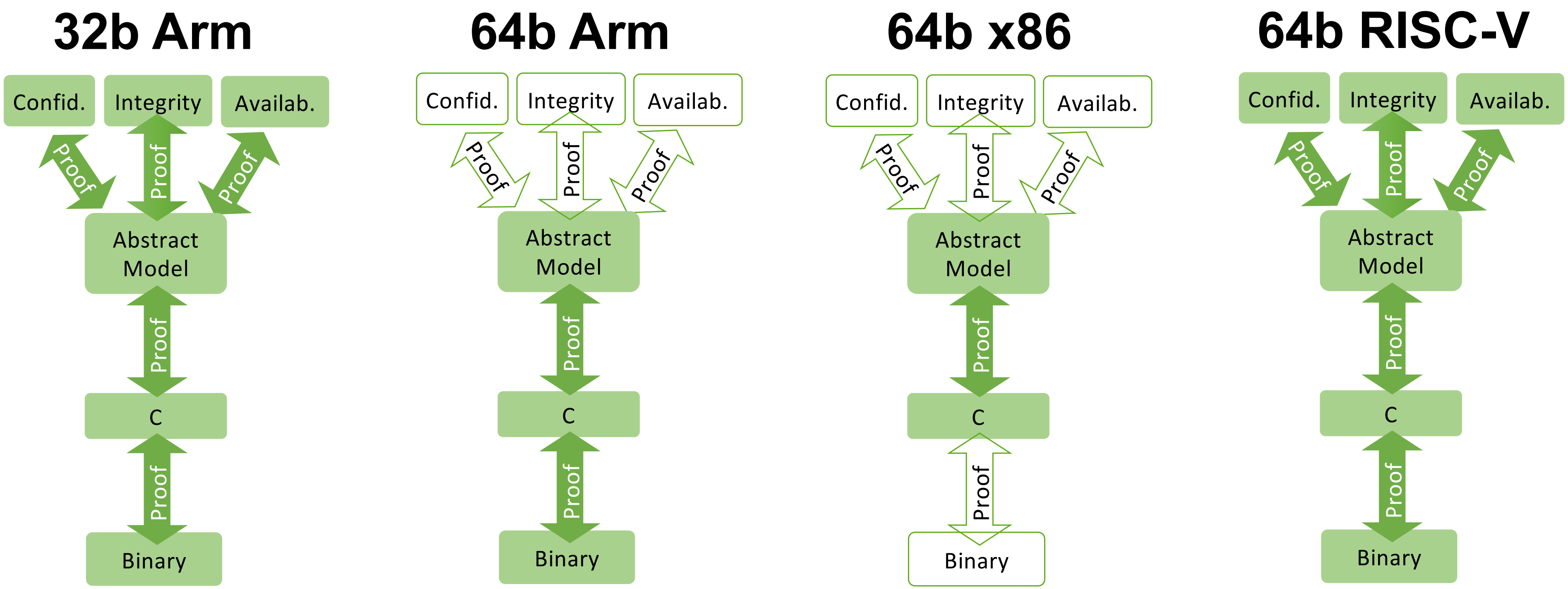

- Not verified boot code
  - **Assume** it leaves kernel in safe state
- Caches/MMU presently modeled at high level / axiomised
- SMP kernel not verified
  - … but multi-kernel is in progress
- Not proved any temporal properties
  - Presently not proved scheduler observes priorities, properties needed for RT
  - WCET analysis applies only to outdated ARM11/A8 cores
  - No proofs about timing channels (yet)

MMU model finished by PhD but not integrated

Just re-done for 64b RISC-V!

Present research!

# Present Status



# seL4 Common Criteria?

| Level | Requirements | Specification | Design | Implementation |
|---|---|---|---|---|
| EAL1 | not evaluated | **Informal** | not eval | not evaluated |
| EAL2 | not evaluated | **Informal** | **Informal** | not evaluated |
| EAL3 | not evaluated | **Informal** | **Informal** | not evaluated |
| EAL4 | not evaluated | **Informal** | **Informal** | not evaluated |
| EAL5 | not evaluated | **Semi-Formal** | **Semi-Formal** | **Informal** |
| EAL6 | **Formal** | **Semi-Formal** | **Semi-Formal** | **Informal** |
| EAL7 | **Formal** | **Formal** | **Formal** | **Informal** |
| seL4 | **Formal** | **Formal** | **Formal** | **Formal** |

# Security Impact of OS Design

# Quantifying OS-Design Security Impact

**Approach:**

- Examine all ***critical*** Linux CVEs (vulnerabilities & exploits database)

  - easy to exploit
  - high impact
  - no defence available
  - confirmed

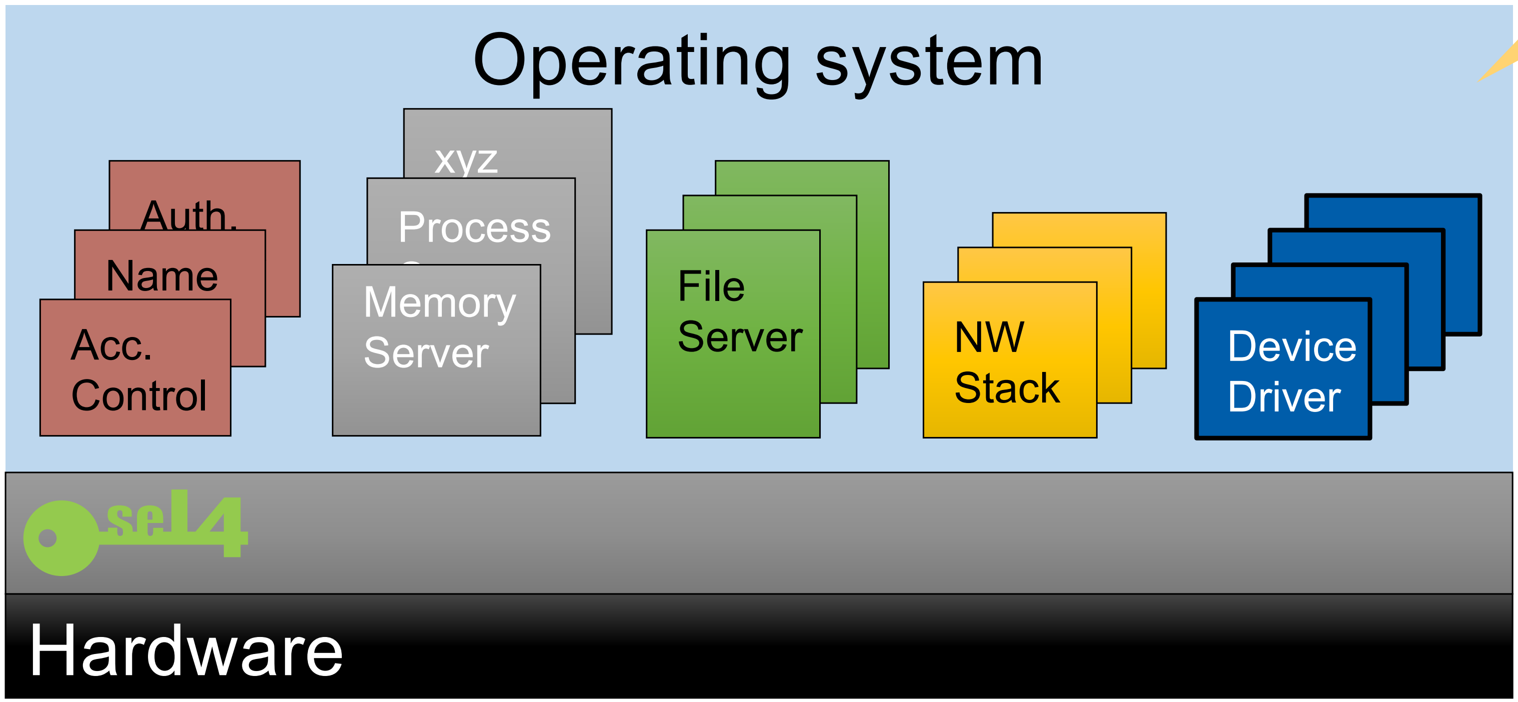
115 critical Linux CVEs to Nov'17

- For each establish how microkernel-based design would change impact

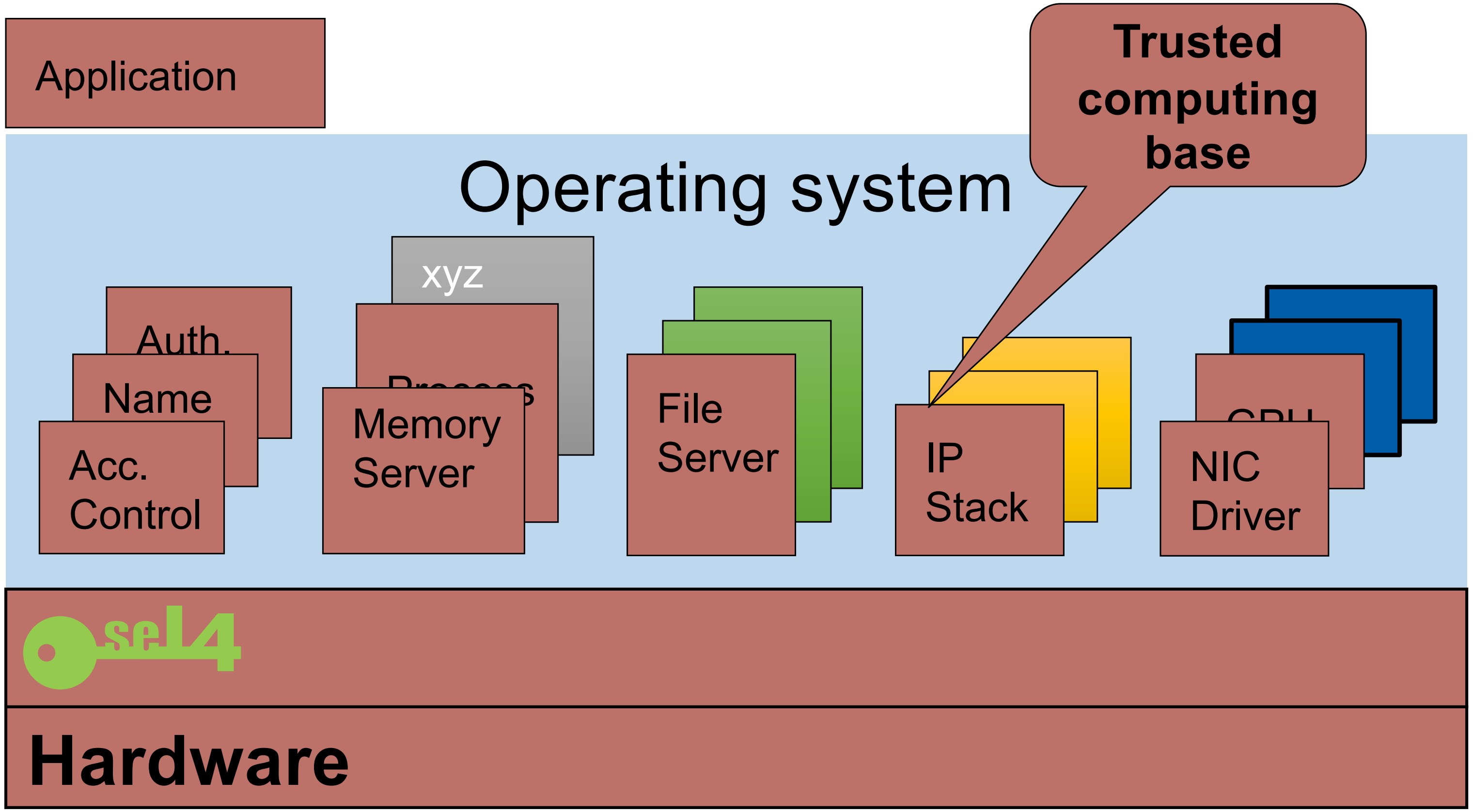
# Hypothetical seL4-based OS

OS structured in *isolated* components, minimal inter-component dependencies, *least privilege*

Functionality comparable to Linux

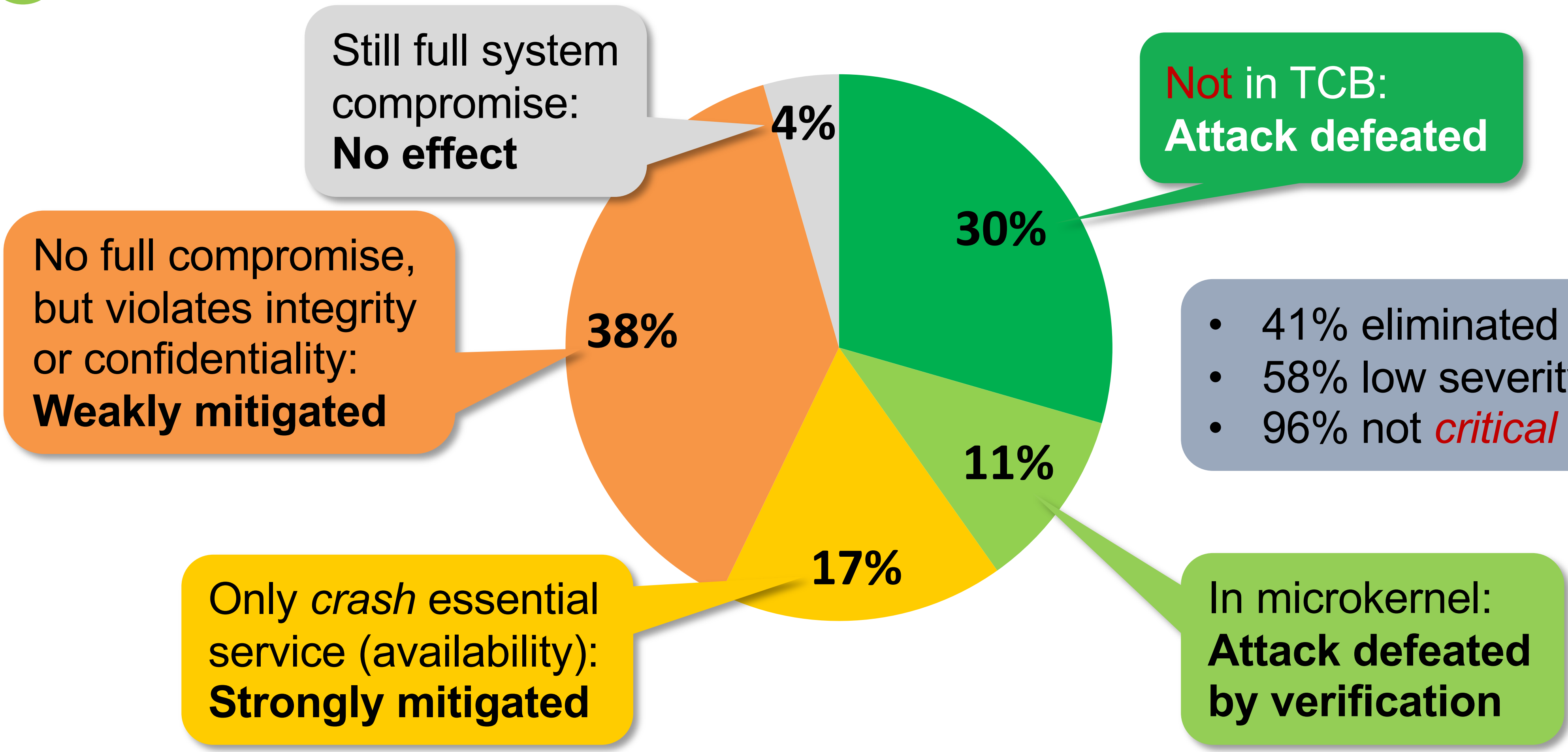


# Hypothetical Security-Critical App




App requires:
- IP networking
- File storage
- Display output

# All Critical Linux CVEs to 2017




- 41% eliminated
- 58% low severity
- 96% not *critical*

# Conclusion: OS Structure Matters

**Problem:**
Where's the
seL4 OS?

- Microkernels definitely improve security
- Microkernel verification improves further
- Monolithic OS design is *fundamentally flawed from security point of view*

[Biggs et al., APSys'18]

**Use of a monolithic OS in security- or safety-critical scenarios is professional malpractice!**

# Strengths & Weaknesses

# seL4 "World's Most Secure Drone"

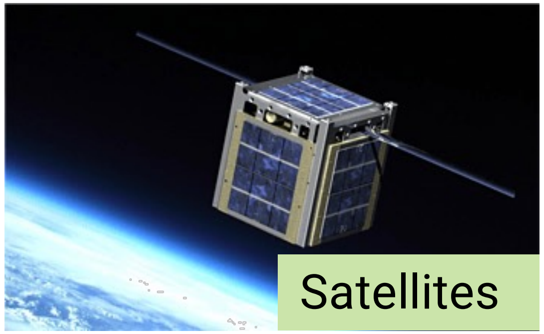
← **Tweet**

**DARPA** @DARPA

We brought a hackable quadcopter with defenses built on our HACMS program to @defcon #AerospaceVillage. As program manager @raymondrichards reports, many attempts to breakthrough were made but none were successful. Formal methods FTW!

DEFCON'22

# seL4 In Real-World Systems



Satellites

Critical infrastructure protection

Autonomous vehicles

Secure communication device
In use in multiple defense forces

Cars

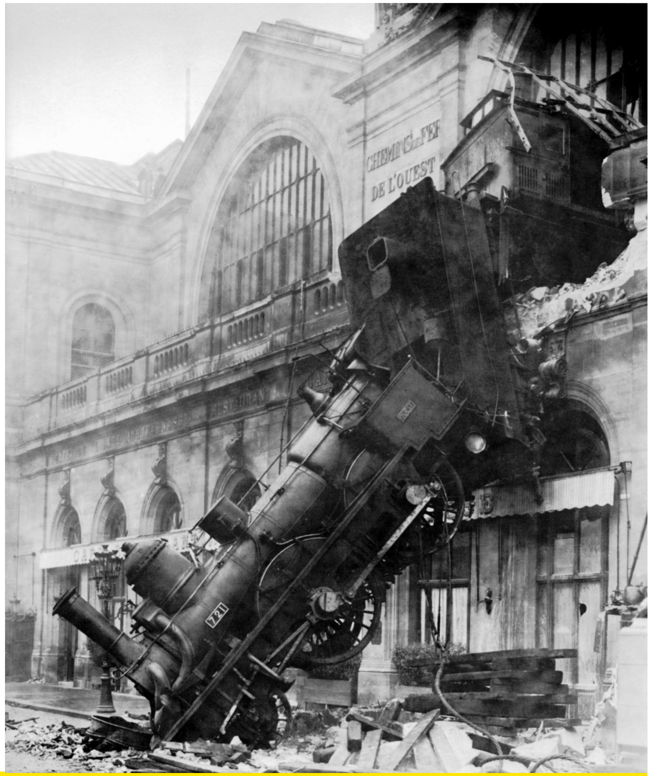
# Microkernel: Assembly Language of OS

**seL4 provides**

- threads
- scheduling contexts
- pages
- endpoints
- notifications
- …

**Programmer wants**

- Processes
- Sockets
- Files



Result: everyone builds their own

... but good design on seL4 requires deep expertise

# Enter LionsOS

Stop The Train Wrecks!

# LionsOS Aims: Fast, Secure, Adaptable

**Aim 1:** *Practical, easy-to-use, open-source* OS for wide range of *embedded/IoT/cyberphysical* use cases

Can use static architecture

Must be well designed!

**Aim 2:** *Best-performing* microkernel-based OS ever

**Aim 3:** *Most secure* OS ever

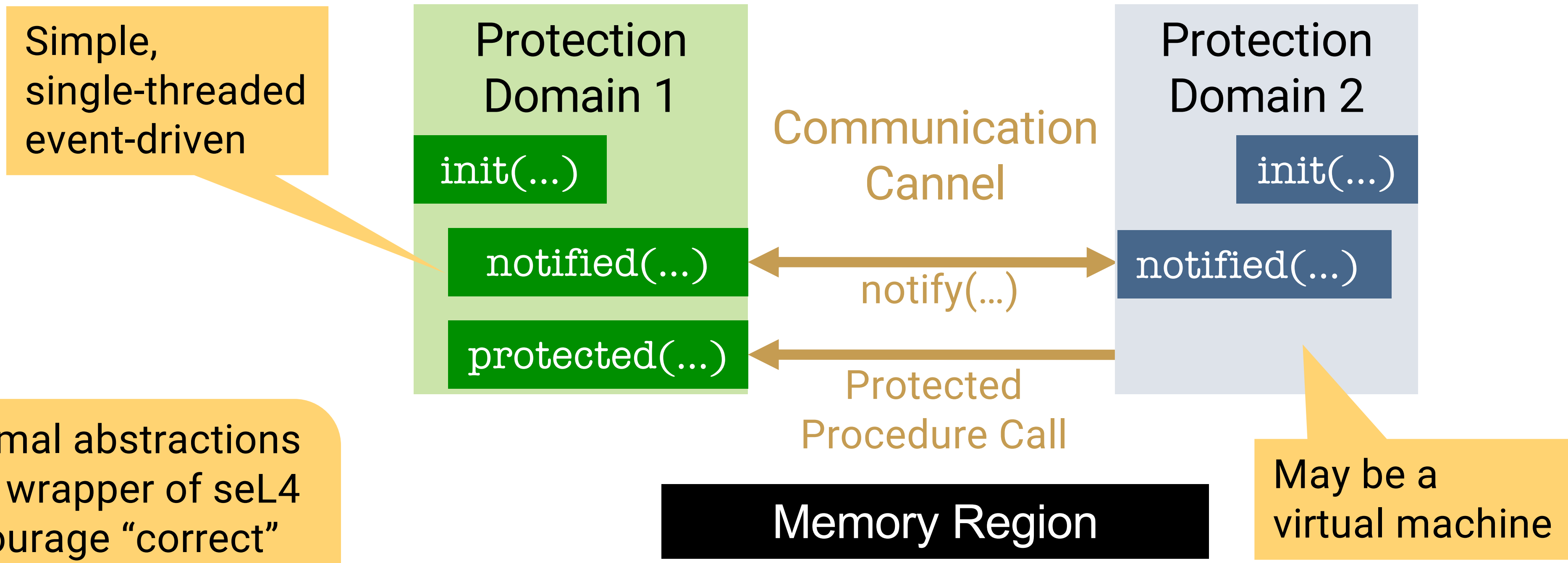
Must be verified!

# Step 1: Microkit – Simple seL4 Abstraction

**Minimal base for IoT, cyberphysical, other embedded use**

- Restrict to static architectures
  - i.e. components & communication channels defined at build time
- Ease development and deployment
  - SDK, integrate with build system of your choice
- Retain near-minimal trusted computing base (TCB)
  - TCB suitable for formal verification
- Retain seL4's superior performance

# Microkit Abstractions




- Minimal abstractions
- Thin wrapper of seL4
- Encourage “correct” use of seL4 primitives
- Static architecture

# seL4 libmicrokit: Event-handler loop

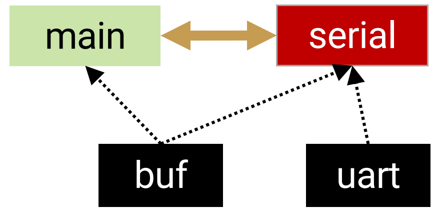
```
for (;;) {
    if (have_reply) {
        tag = seL4_ReplyRecv(INPUT_CAP, reply_tag, &badge, REPLY_CAP);
    } else if (have_signal) {
        tag = seL4_NBSendRecv(signal, signal_msg, INPUT_CAP, &badge, REPLY_CAP);
        have_signal = false;
    } else {
        tag = seL4_Recv(INPUT_CAP, &badge, REPLY_CAP);
    }
    event_handle(badge, &have_reply, &reply_tag, &notified);
}
```

# libmicrokit: Invoking user code

```
event_handle(badge, &have_reply, &reply_tag, &notified) {
    if ((have_reply) = badge >> 63) {
        reply_tag = protected(badge & 0x3f, tag);
    } else {
        unsigned int idx = 0;
        do {
            if (badge & 1) {
                notified(idx);
            }
            badge >>= 1; idx++;
        } while (badge != 0);
    }
}
```

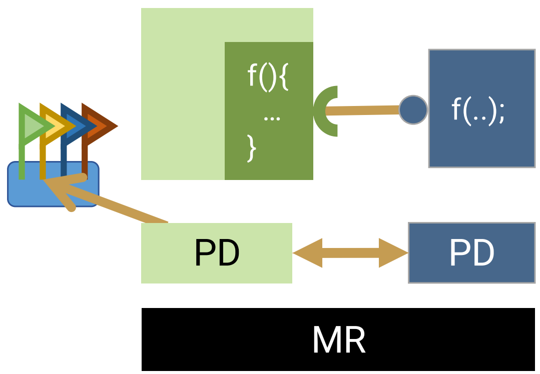
# Microkit System Description File (SDF)

```
<system>
    <memory_region name="uart" size="0x1000" phys_addr="0x9000000" />
    <memory_region name="buf" size="0x1000" />
    <protection_domain name="serial” priority="250”>
        <irq irq="33" id="0" />
        <program_image path="serial_server.elf" />
        <map mr="uart" vaddr="0x4000000" perms="rw" cached="false” … />
        <map mr="buf" vaddr="0x4001000" perms="rw" setvar_vaddr="input" />
    </protection_domain>
    <protection_domain name=”main">
        <program_image path=”main.elf" />
    </protection_domain>
    <channel>
        <end pd="serial" id="1" />
        <end pd="client" id="0" />
     </channel>
</system>
```



# Microkit Status

- Easy to use – non-experts productive within hours
- Supports AArch64, RV64 (x64 release next month)
- Verification presently for initial version & hacky, doing properly
- Limited dynamic features:
  - fault handlers
  - start/stop protection domains
  - empty protection domains (for late app loading)
- In progress:
  - re-initialise protection domains
  - “template PDs” – discretionary access
  - Core management: on-/off-lining cores



# LionsOS

Fast – secure – adaptable!

# Lions OS: Principles

**Least Privilege**

**Strict separation of concerns**

**Overarching principle: KISS**
"Keep it simple, stupid!"

**Radical simplicity**

**Use-case–specific policies**

**Design for verification**

# Radical Simplicity™

Provide **exactly** the functionality needed, not more

Simple programming model:
- strictly sequential code (Microkit)
- event-based (Microkit)
- single-producer, single-consumer queues
- location transparency
- …

Static **architecture**, mostly static resource management

# Use-Case–Specific Policies

Source of massive complexity

'80s model of computer use!

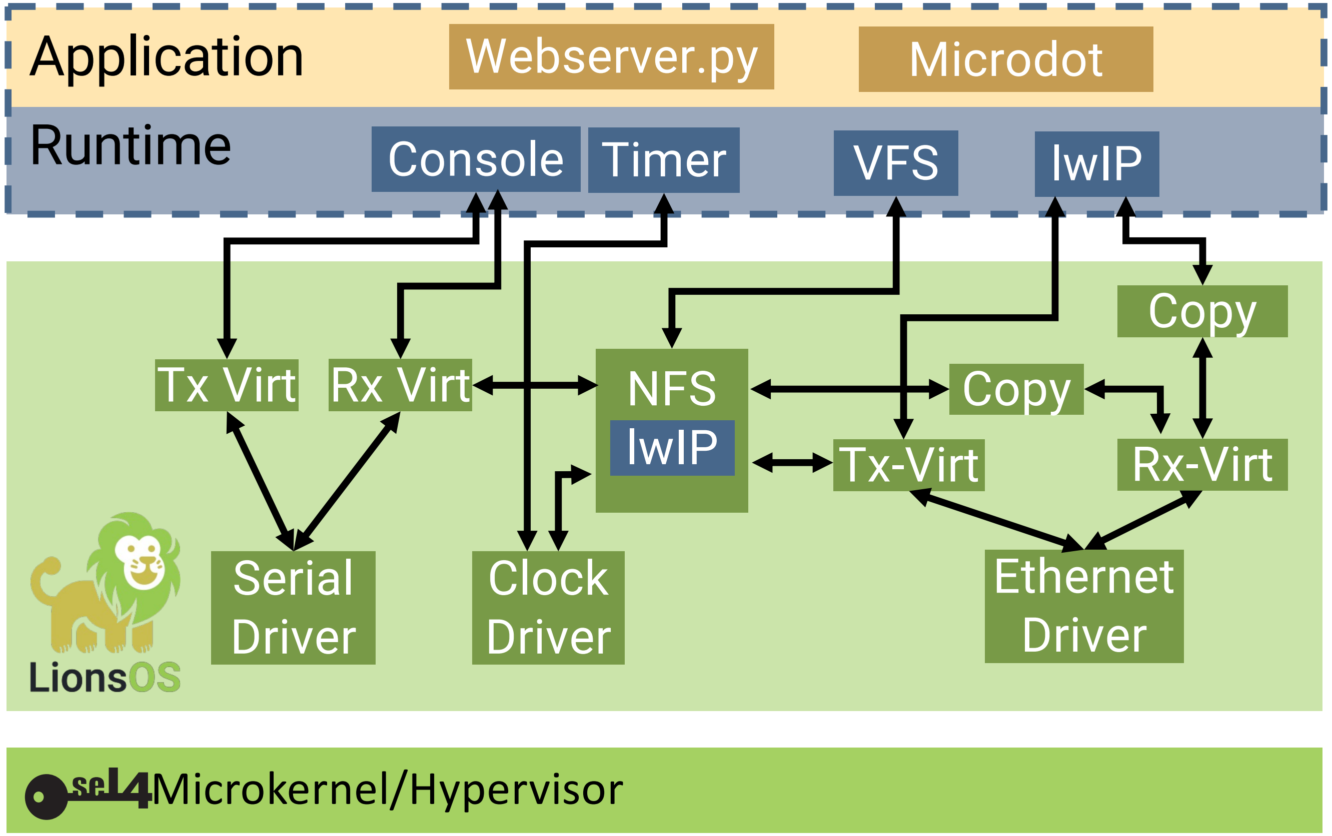
~~**Traditional OS:** achieve adaptability by universal policies~~

**Lions-OS: Use-case diversity through policies that are:**

- optimised for one specific use case
- simple, localised implementation
- easy to replace by swapping component

# Underneath https://sel4.systems/

**Web-server OS:**
- 10 modules
- 3 libraries



# Web Server Code Sizes (all C)

| Component | LoC | Library | LoC |
|---|---|---|---|
| Timer Driver | 139 | Microkit | 368 |
| Serial Driver | 231 | Serial queue | 169 |
| Serial Tx Virt | 159 | Eth queue | 140 |
| Serial Rx Virt | 109 | Filesys queue & protocol | 268 |
| Eth Driver | 397 | | |
| Eth Tx Virt | 107 | | |
| Eth Rx Virt | 151 | Coroutines | 848 |
| Eth Copier | 73 | LWIP | 16,280 |
| Monitor | 1,188 | NFS | 45,707 |
| **LionsOS trusted** | **3,545** | **Untrusted** | **62,356** |
| Web server app | 7,246 | MicroPython | 402,554 |

**Trusted:**
- 13 modules/ libraries
- Av 270 LoC

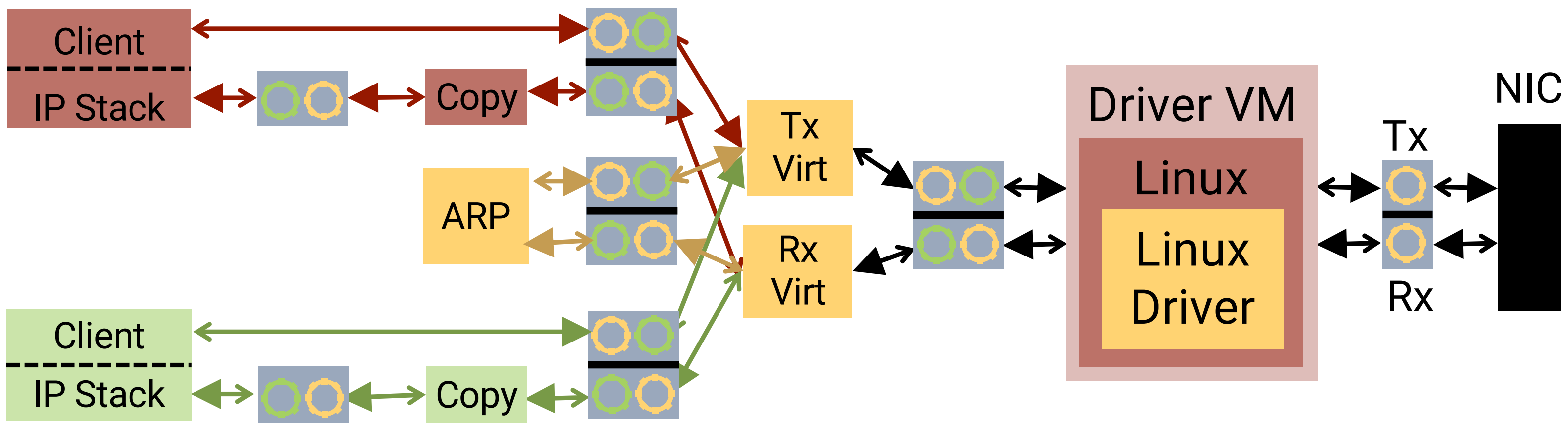
Untrusted



UNSW SYDNEY

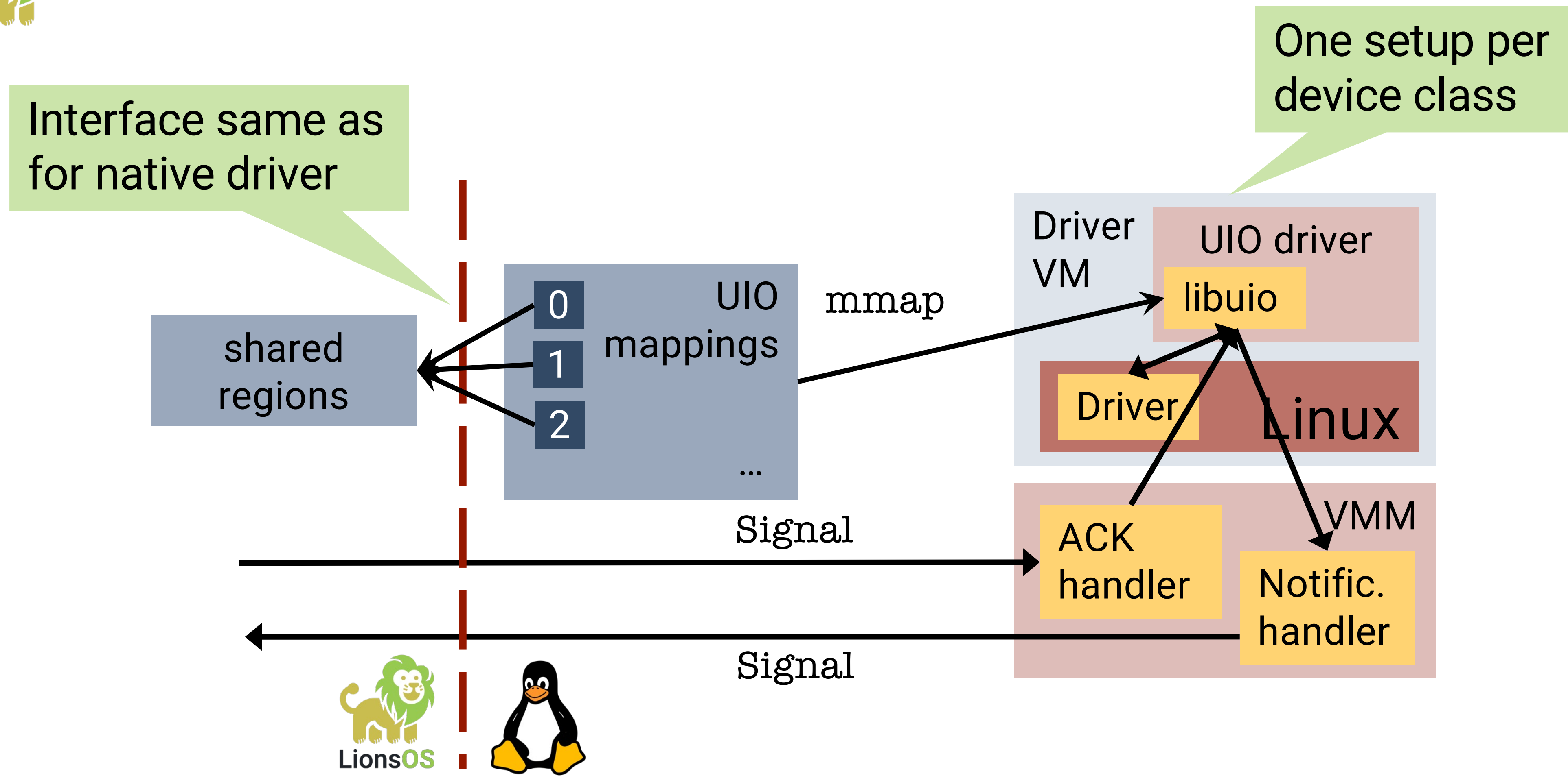
# LionsOS Driver VMs

- Transparently use per-device driver VM instead of native driver
- Re-use unmodified Linux driver




Approach pioneered by LeVasseur&Uhlig, OSDI'04

# LionsOS Driver VMs



# Driver-VM Cost

**In progress:** using same setup to develop LionsOS modules under Linux

| Driver | Kernel | RAM Disk | Runtime | Total |
|---|---|---|---|---|
| Default | 29 MiB | 6.7 MiB | 70 MiB | 106 MiB |
| Audio | 3 MiB | 2.4 MiB | 18 MiB | 23 MiB |
| Block | 3 MiB | 0.05 MiB | 12 MiB | 15 MiB |

Optimised

**Effort:**
- Few days to set up UIO driver
- Total ≈ 2 weeks / device class

# Reminders

- Please complete the myExperience Survey
- Exam preparation session: Wed 26/11 at 3pm
- Honours theses at Trustworthy Systems
  https://trustworthy.systems/students/theses
- **John Lions CS Honours Award for thesis in OS**
  https://www.scholarships.unsw.edu.au/scholarships/id/1757
  **Deadline: 5 December for T1/25!**