School of Computer Science & Engineering

Trustworthy Systems Group

LionsOS

Towards a truly dependable operating system

Gernot Heiser

gernot@unsw.edu.au
@gernot@discuss.systems
https://microkerneldude.org/

Cyberattacks Are Everywhere

Report Sho… Cyber
Attacks o…
Have Doub…

News / World

'Most serio…
of thousan…

AP By Associated…

RAND / Research & Commentary / Blog /

Threats to America's Critical Infrastructure Are Now a Terrifying Reality

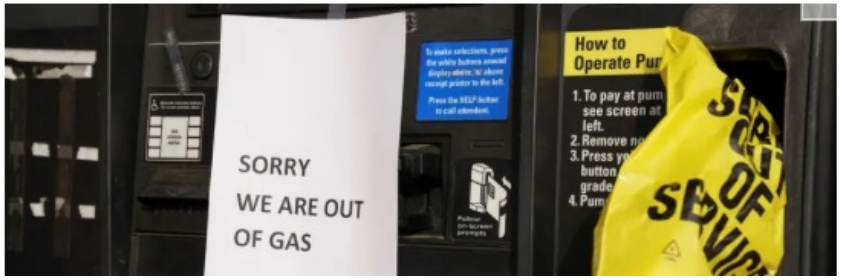
COMMENTARY — Feb 12, 2024

Cyberattacks on Automated Vehicles Rise by 99%: Report

By CISOMAG - June 9, 2020

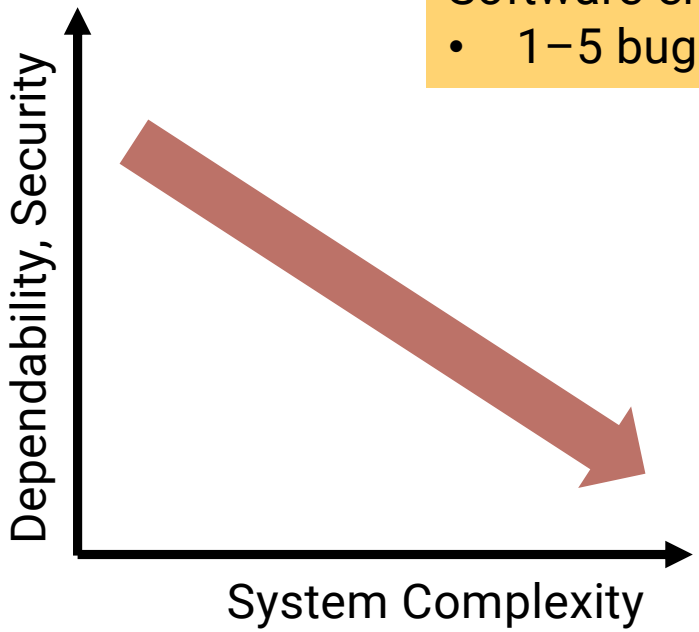
…et Electrical
…What

Increasingly used by
- organised crime
- state actors



…causes delay at Zurich Airport

Core Problem: Complexity



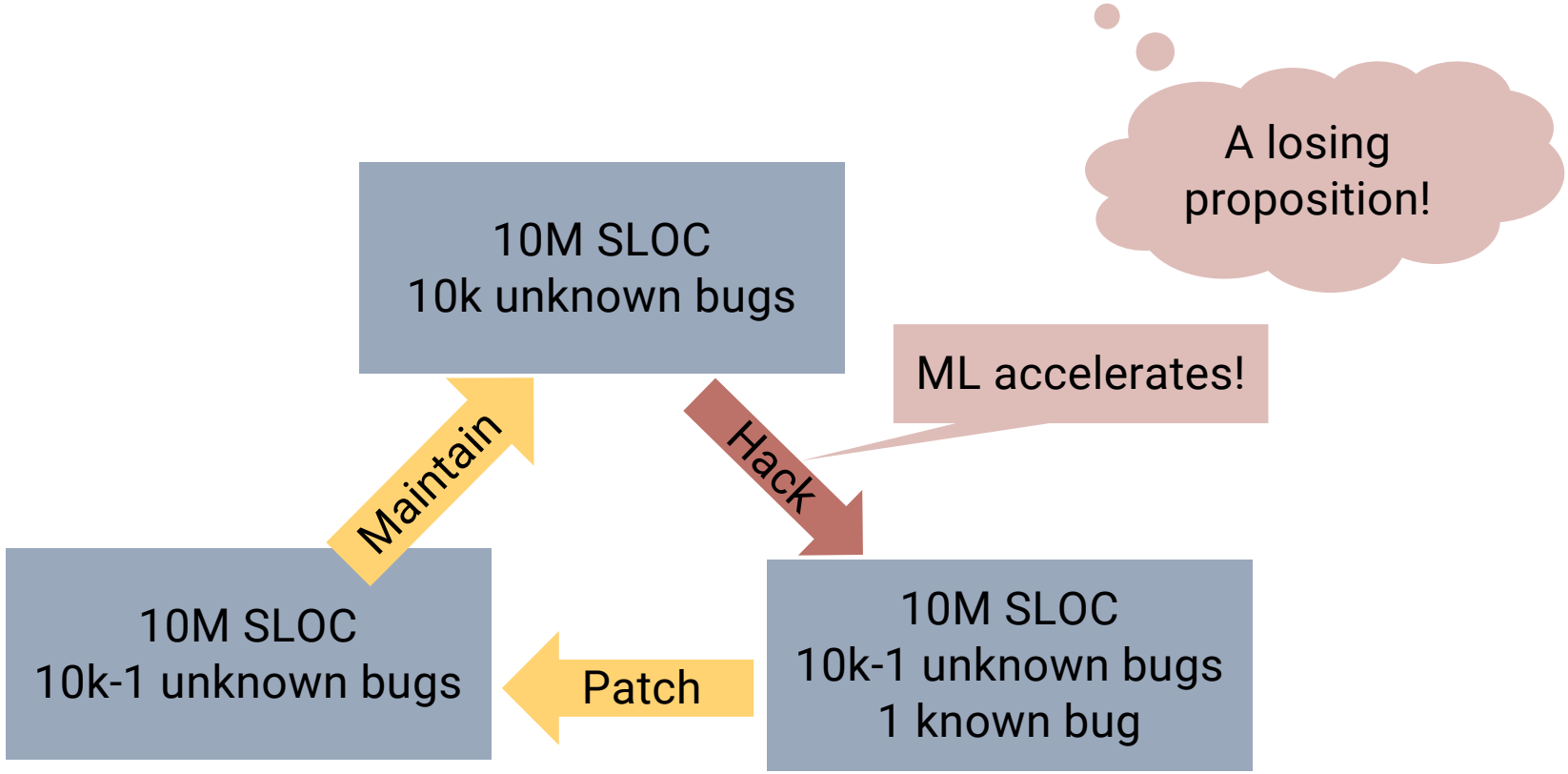
Software-engineering rule of thumb:

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

Bluetooth protocol stack:
100s kSLOC

Linux/Windows kernel:
10s MSLOC

Standard Approach: Patch-and-Pray

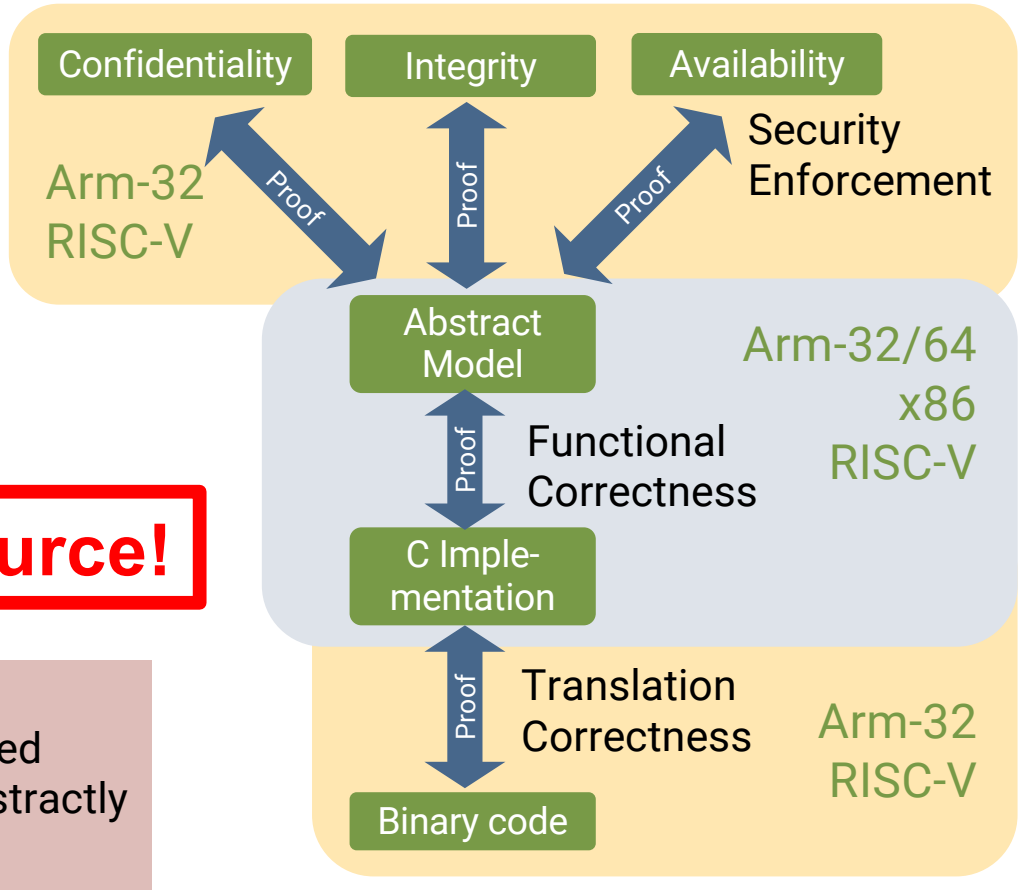


We Need To Do Better!

We need principled solutions based on solid foundations!

Mathematically Proved Secure

- Comprehensive formal verification
- Only verified OS with fine-grained protection (capabilities)
- Only protected-mode RTOS with sound and compete WCET analysis
- World's fastest microkernel

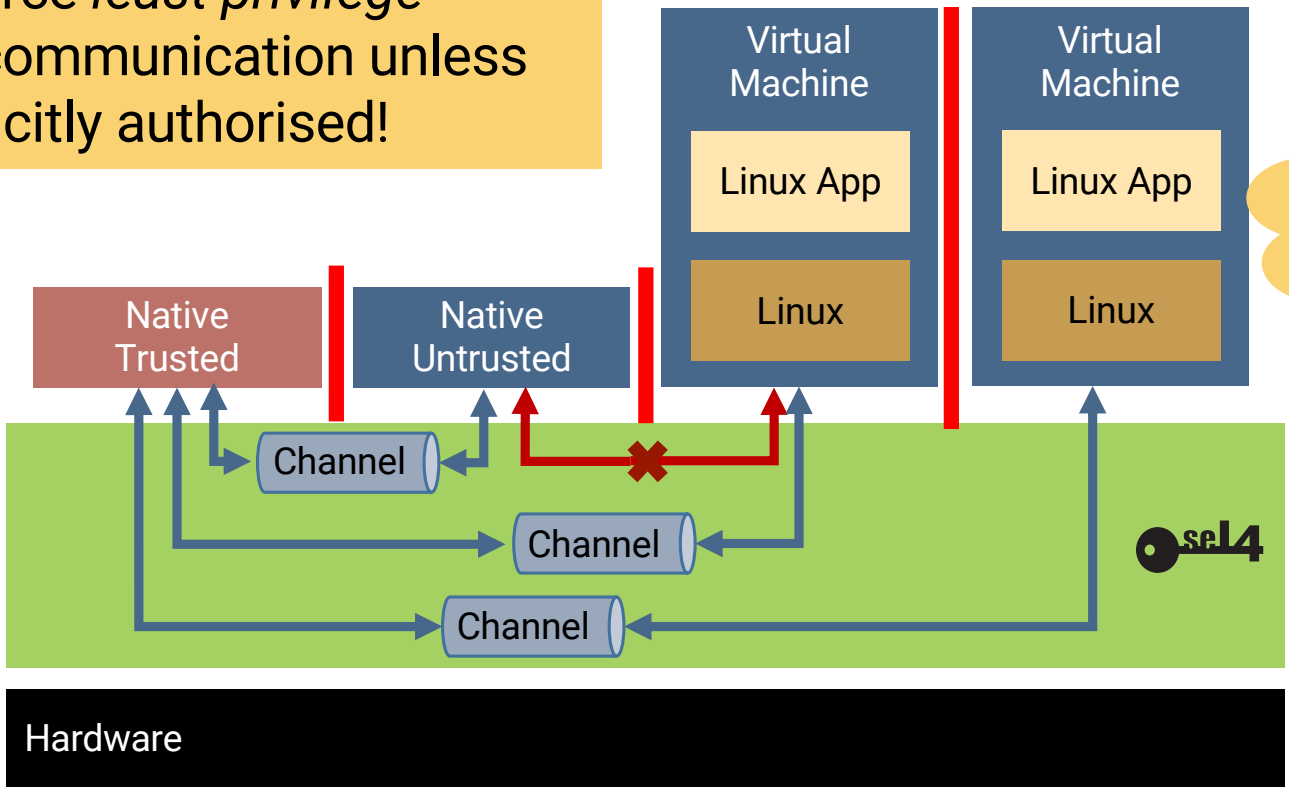


Open Source!

Present limitations
- initialisation code not verified
- MMU, caches modelled abstractly
- Multicore not yet verified

Capabilities: Fine-Grained Protection

- Enforce *least privilege*
- No communication unless explicitly authorised!



No capabilities? You're not serious about security!

The Benchmark for Performance

Round-trip cross-address-space IPC on 64-bit Intel Skylake

	seL4	Fiasco.OC aka L4Re	Google Zircon
Latency (cycles)	986	2717	8157
Mandatory HW cost* (cycles)	790	790	790
Overhead absolute (cycles)	196	1972	7367
Overhead relative	25%	240%	930%

Smaller is better

World's fastest microkernel!

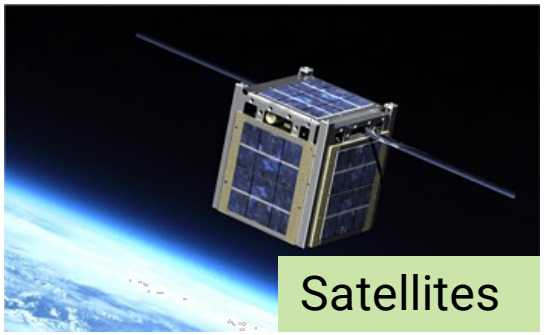
*: The Cost of `SYCALL` + 2 × `SWAPGS` + `SYSRET` = 395 cycles, times 2 for round-trip

Source:

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

Used in Real-World Systems

Autonomous vehicles



Satellites

Critical infrastructure protection

Secure communication device
In use in multiple defense forces

Cars

“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

Why Aren’t We Done Yet?

seL4 Timeline

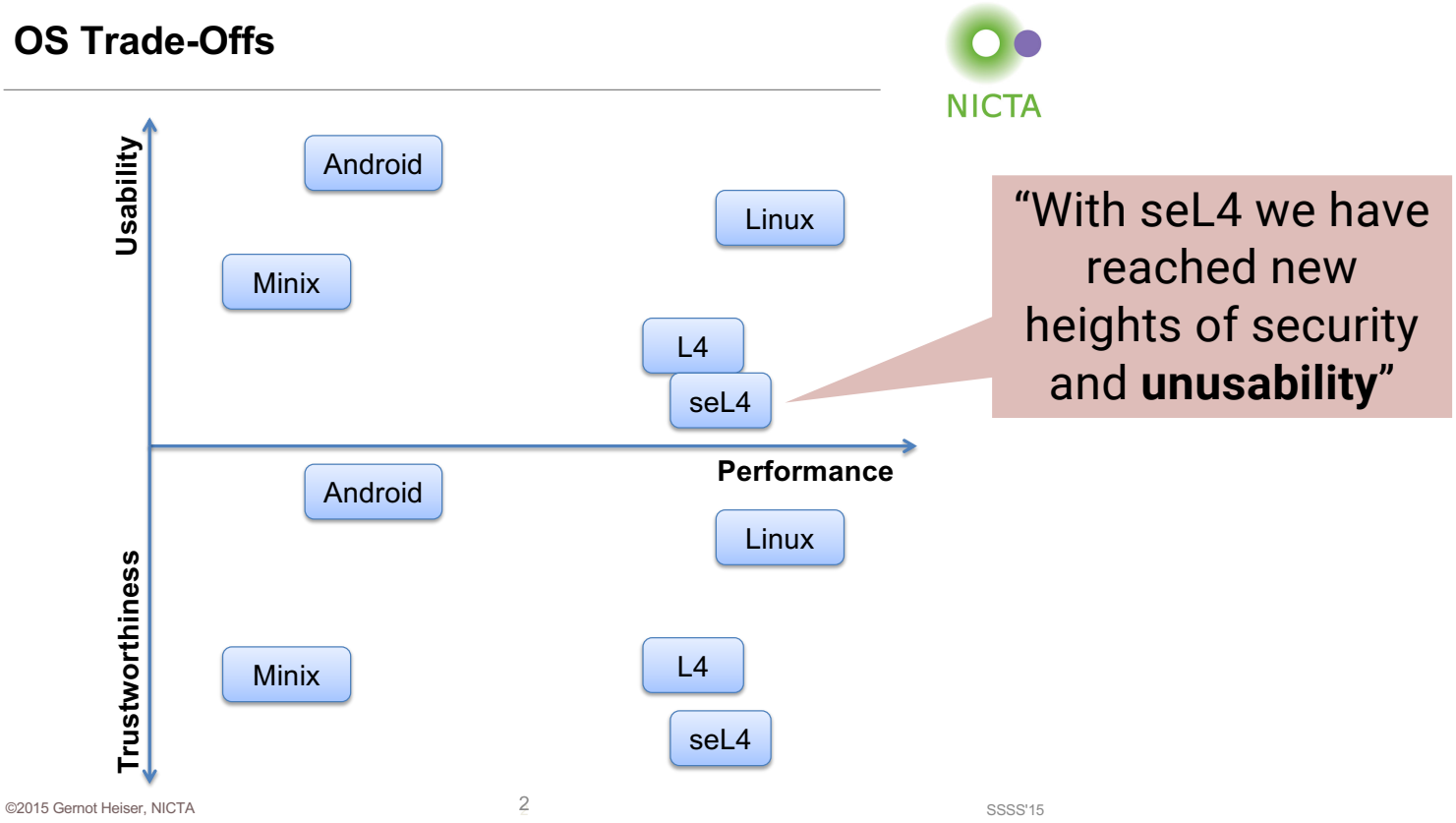
- July'09: Proof of implementation correctness (Armv7)
- Aug'11: Proof of integrity enforcement
- Nov'11: Sound worst-case execution-time analysis
- May'13: Proof of confidentiality enforcement
- Jun'13: Proof of compilation correctness
- Jul'14: **seL4 open-sourced (GPL)**
- 2012–17: DARPA HACMS: seL4 in real-world systems
- 2018: x86 verification
- Jun'20: RISC-V verification
- Mar'24: AArch64 verification
- Oct'24: Commercial car

A Microkernel

Microkernel:
- OS code that must execute in privileged mode
- Everything else belongs in user mode servers
- Servers are subject to the microkernel's security enforcement!

Consequence:
- Small: 10 kLOC
- Only fundamental, policy-free mechanisms
- No application-oriented services/abstractions
- **BYO file system, memory manager, device drivers**

A Slide from Feb'15



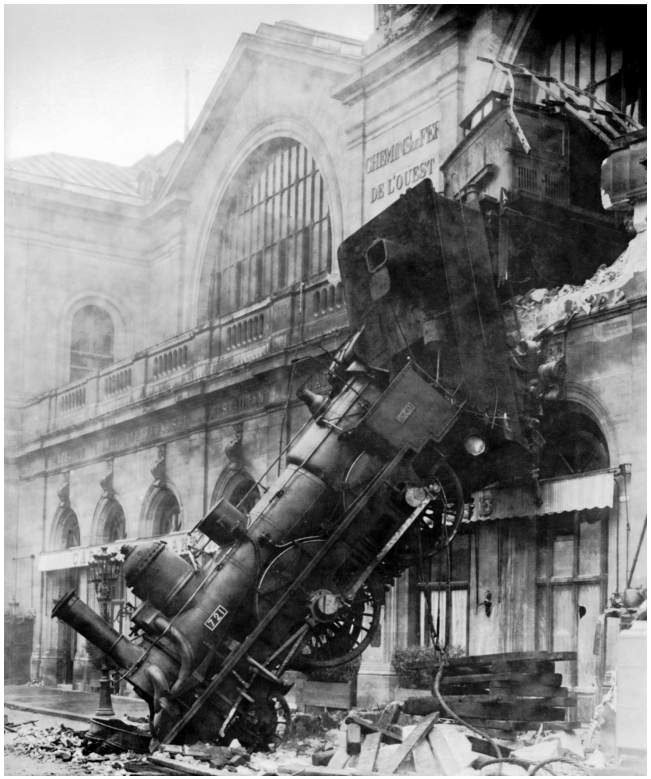
seL4 Experience of the First 10+ Years

seL4's assurance and power is unrivalled, but:

- good design of seL4-based systems requires deep expertise
- a secure microkernel doesn't guarantee a secure system

seL4 needs an OS that:

- provides "usual" OS services
- is easy to use
- is performant
- **is secure**



Enter LionsOS

Stop The Train Wrecks!

Lions OS: Secure, Fast, Adaptable

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

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

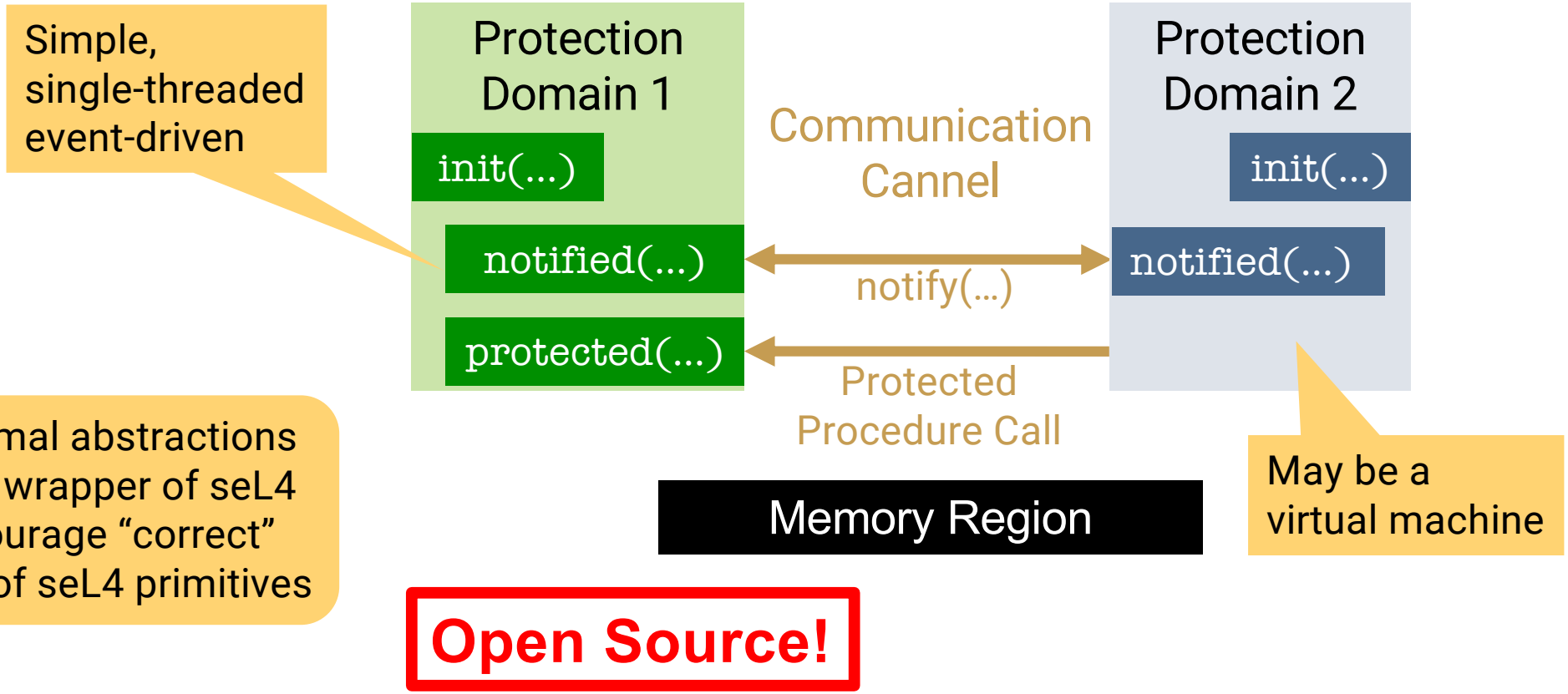
Aim 3: *Provably secure* OS

First Step: The seL4 Microkit

Implications:
- Only static architectures, i.e. all components known at build time
- Matches embedded/ cyberphysical systems!

Benefit:
- easy to use
- efficient

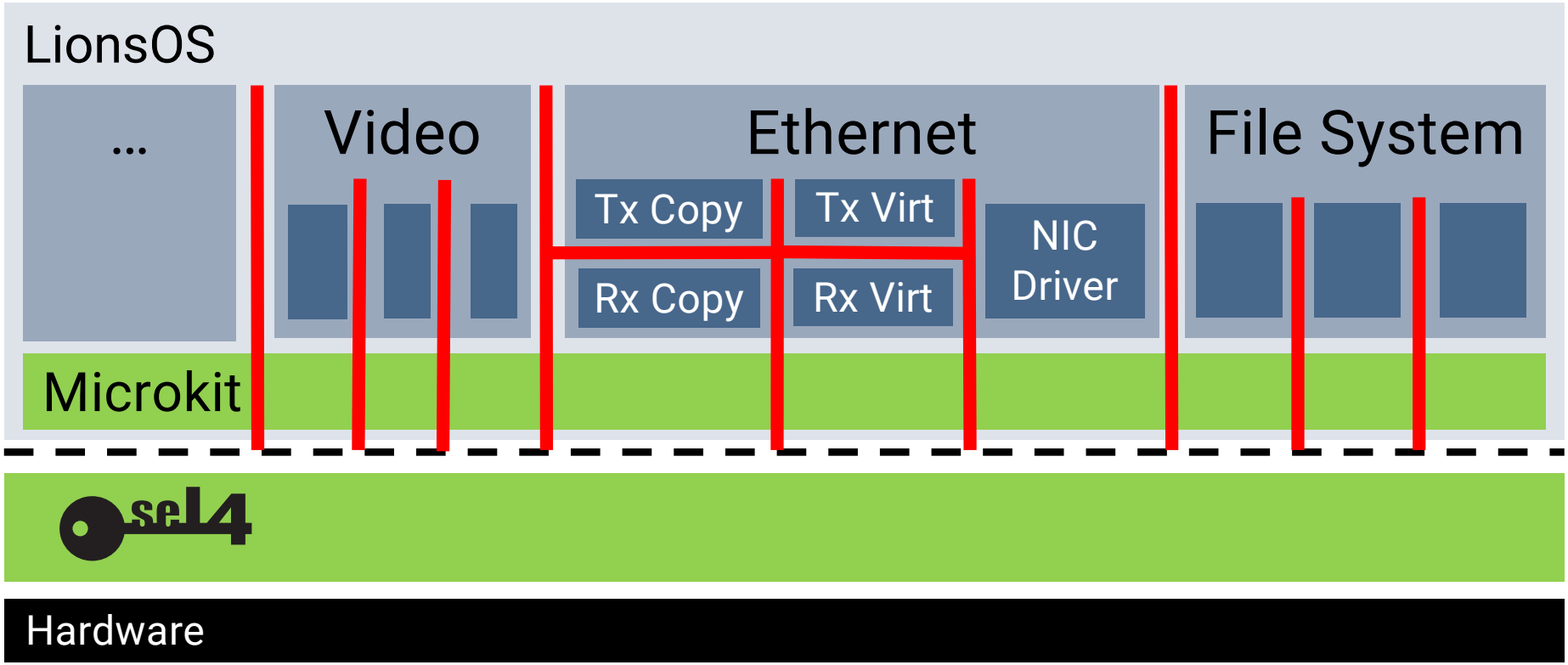
Microkit Abstractions (Yes, that's it!)



- Minimal abstractions
- Thin wrapper of seL4
- Encourage "correct" use of seL4 primitives

Open Source!

LionsOS: Modular System on Microkit



LionsOS Design Principle: KISS

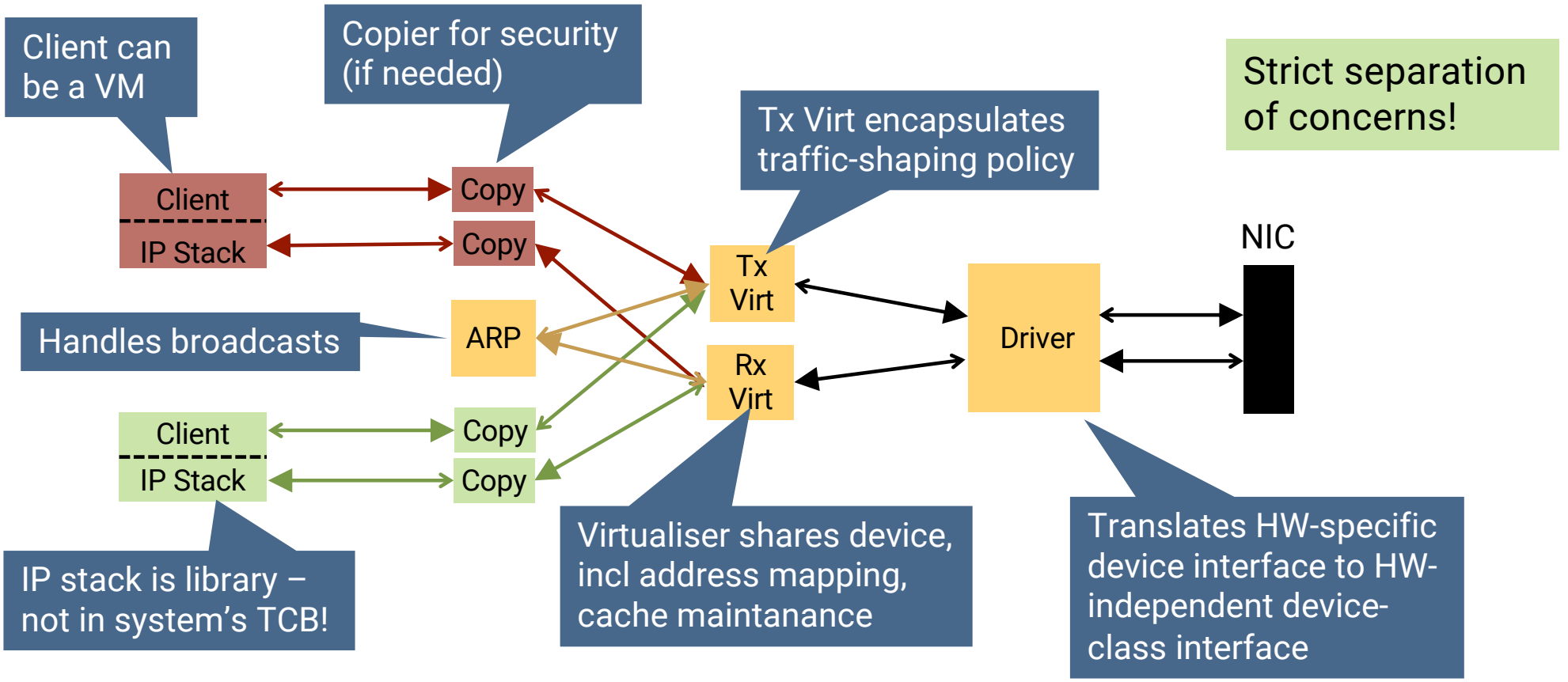
Keep it simple, stupid!

- Strong separation of concerns
- Simplest implementation possible
- Least privilege

Implications:

- Use-case-specific instead of “universal” policies
- Use-case diversity by swapping policy modules!

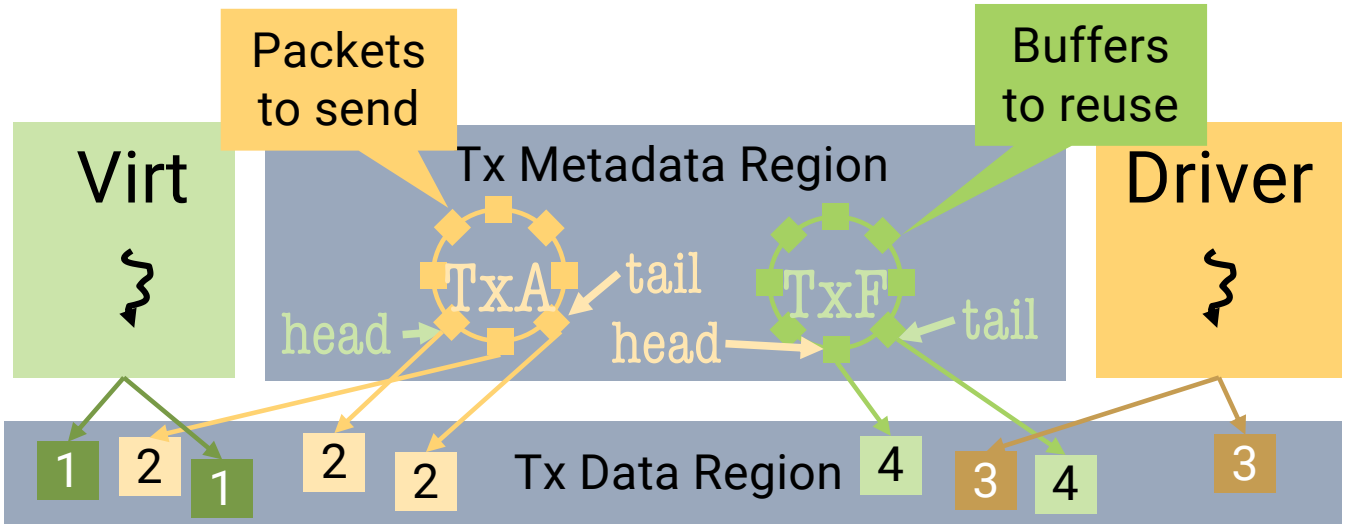
Example: Networking Subsystem



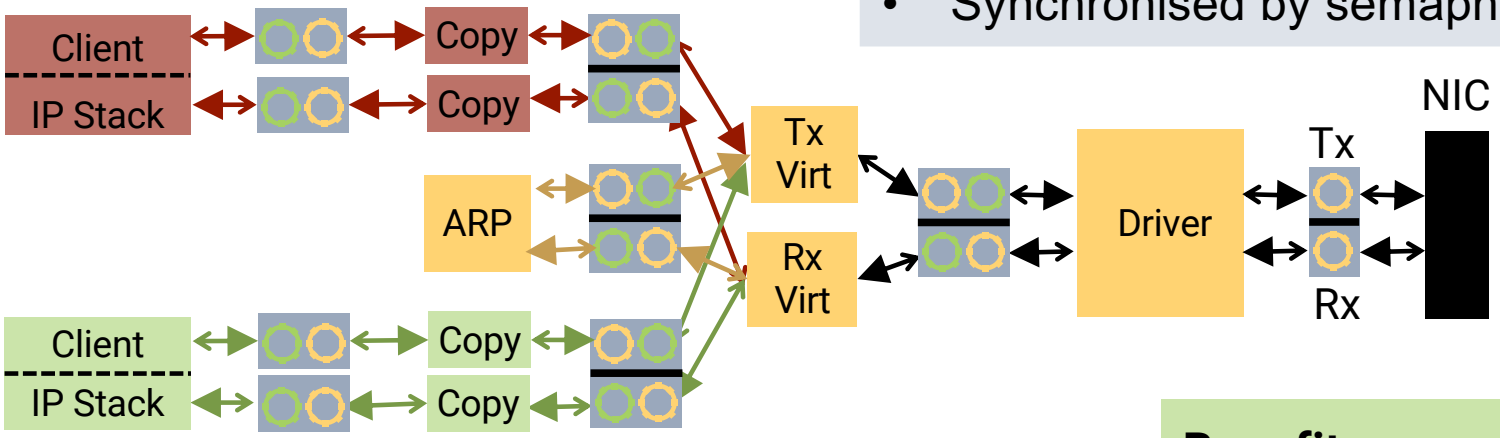
Zero-copy Data Transfer

“Tamed” concurrency!

- Lock-free bounded queues
- Single producer, single consumer
- Similar to ring buffers used by NICs
- Synchronised by semaphores



Networking Detail



Zero-copy communication:

- Lock-free, single-producer, single-consumer, bounded queues
- Synchronised by semaphores

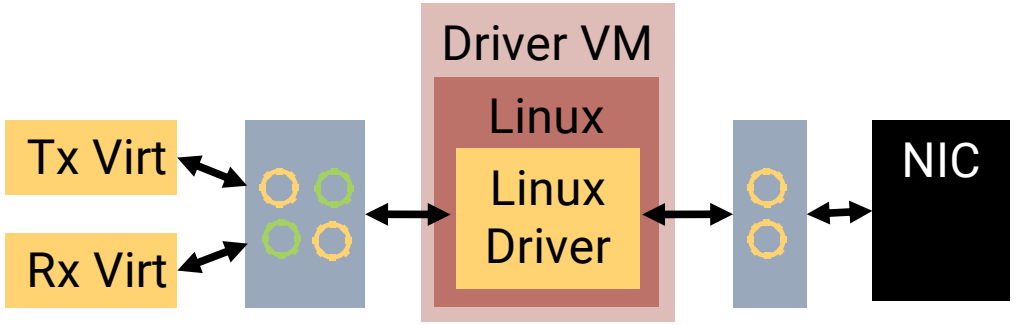
Benefits:

- simple components
- location transparency
- suitable for verification

Legacy Re-use: Driver VMs

Can re-use unmodified Linux drivers:

- Transparently use driver VM instead of native driver
- develop LionsOS components on Linux



Comparison to Linux on i.MX8M (Armv8)

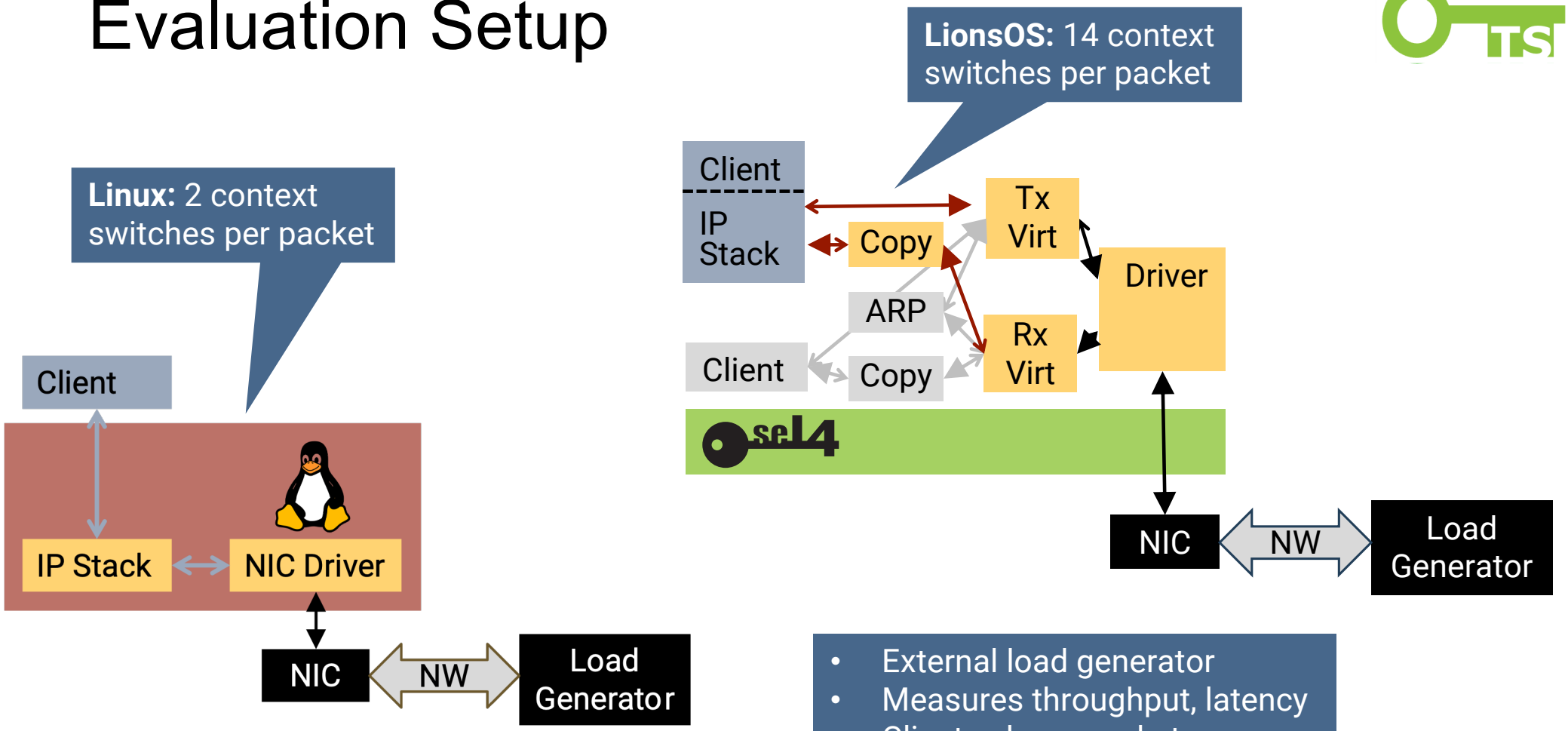
Linux:
- NW driver: 4k lines
- NW system total: 1M lines

Performance?

Written by second-year student!

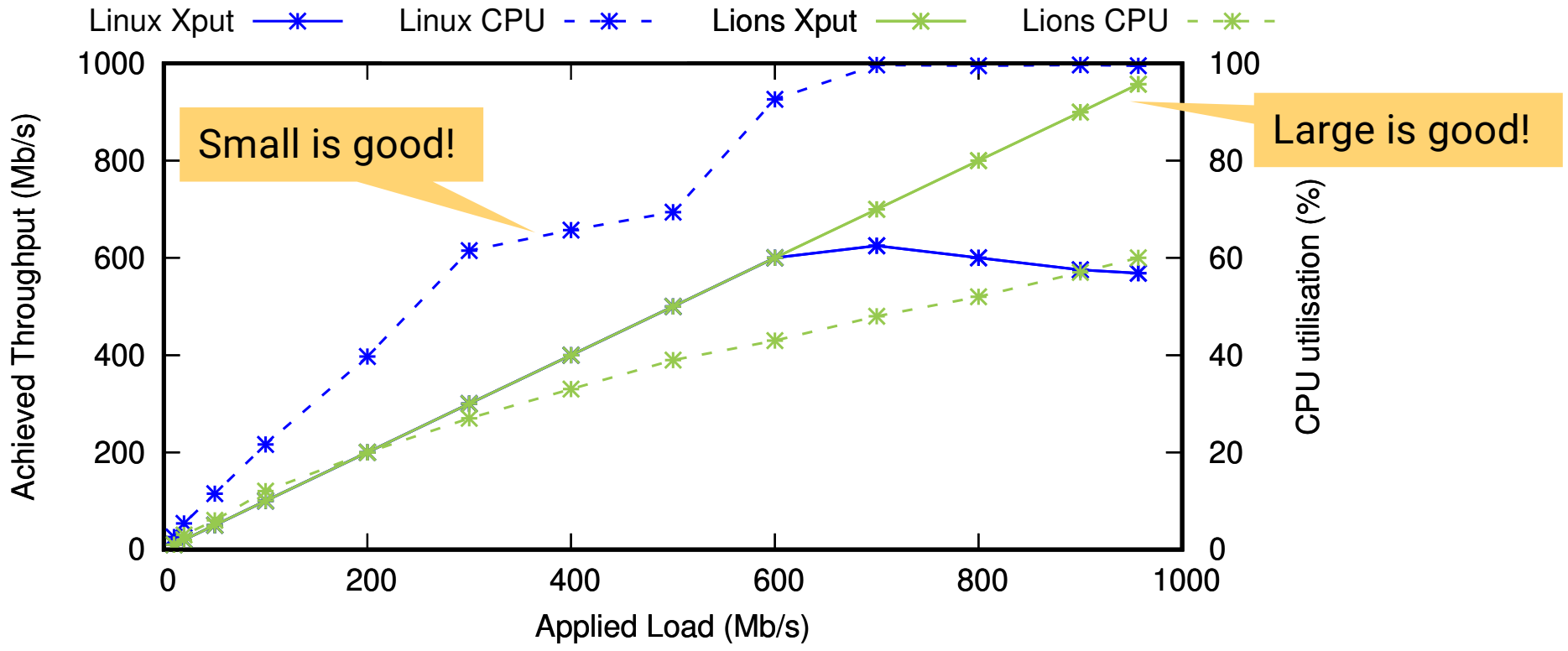
LionsOS:
- NW driver: 700 lines
- MUX: 400 lines
- Copier: 200 lines
- IP stack: much simpler, client library
- shared NW system total: < 2,000 lines

Evaluation Setup



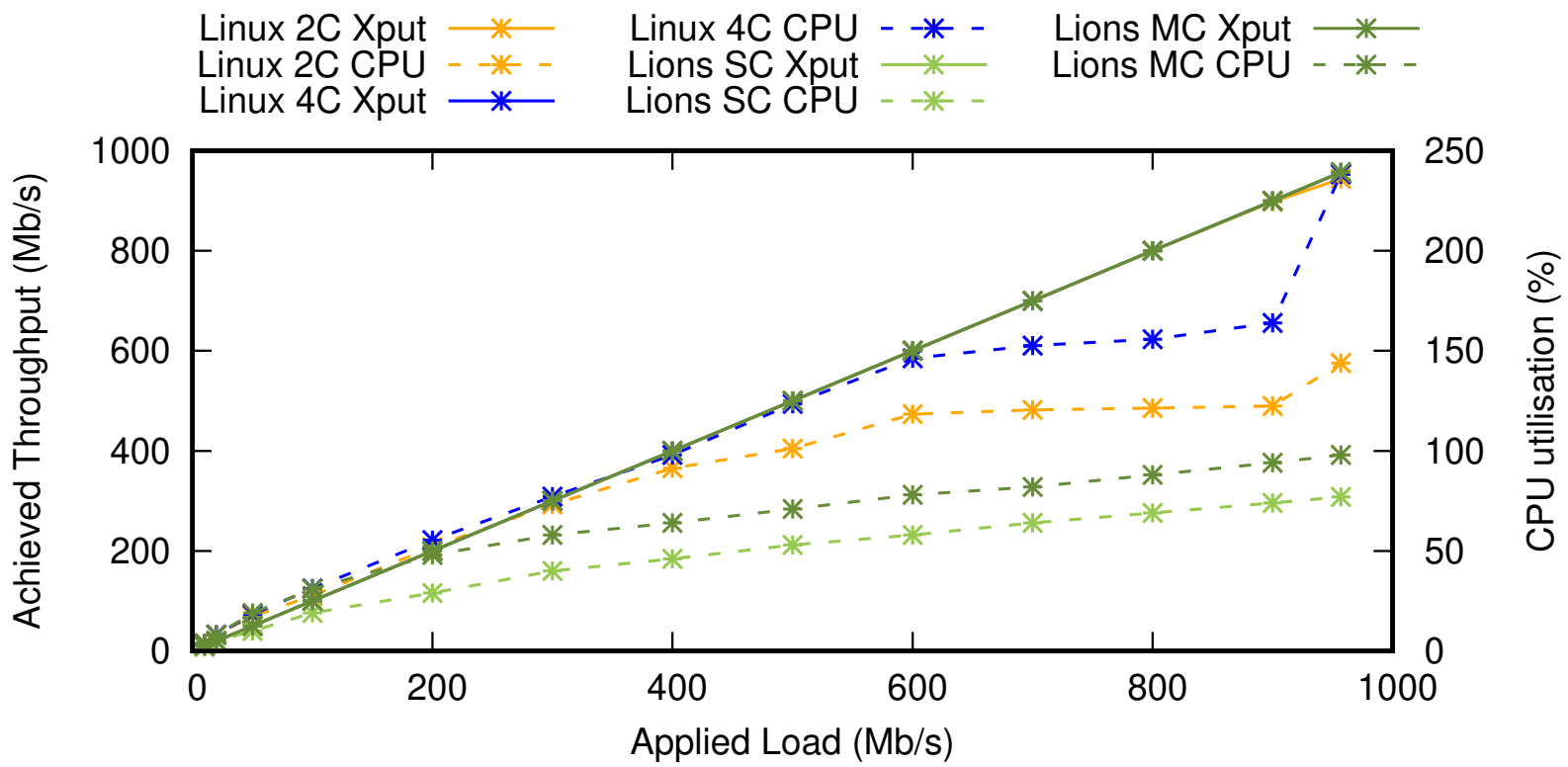
- External load generator
- Measures throughput, latency
- Client echoes packets

Performance: i.MX8M, 1Gb/s E/N, UDP



Single-core configuration

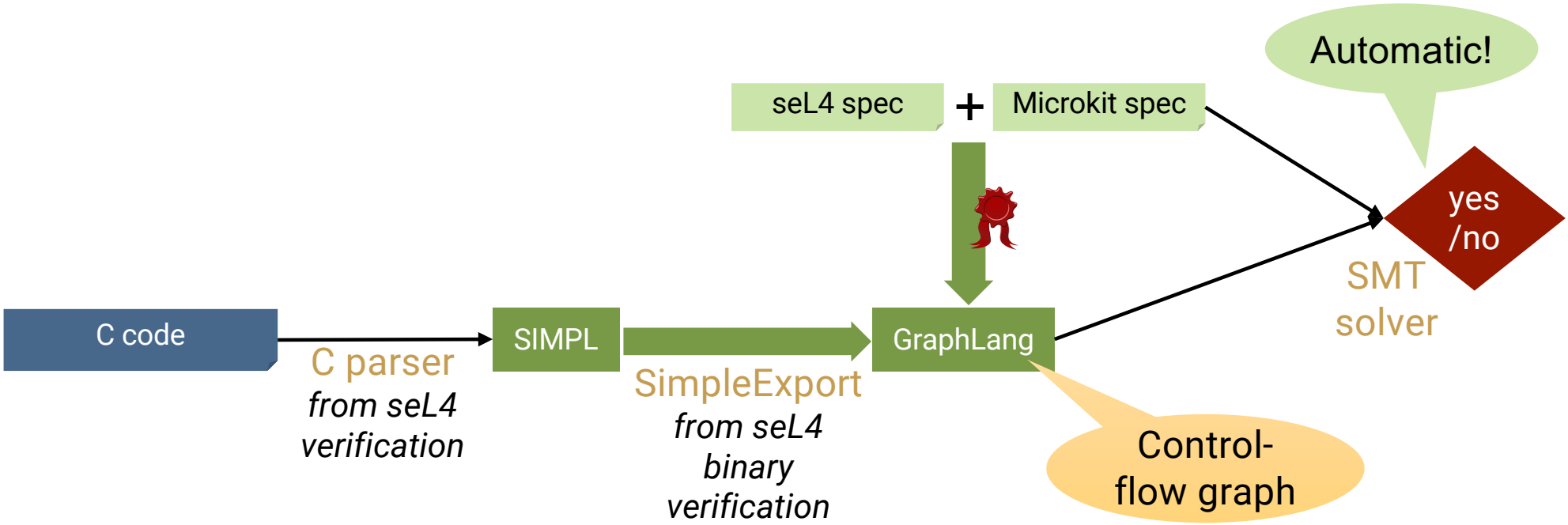
Performance: i.MX8M, 1Gb/s E/N, UDP



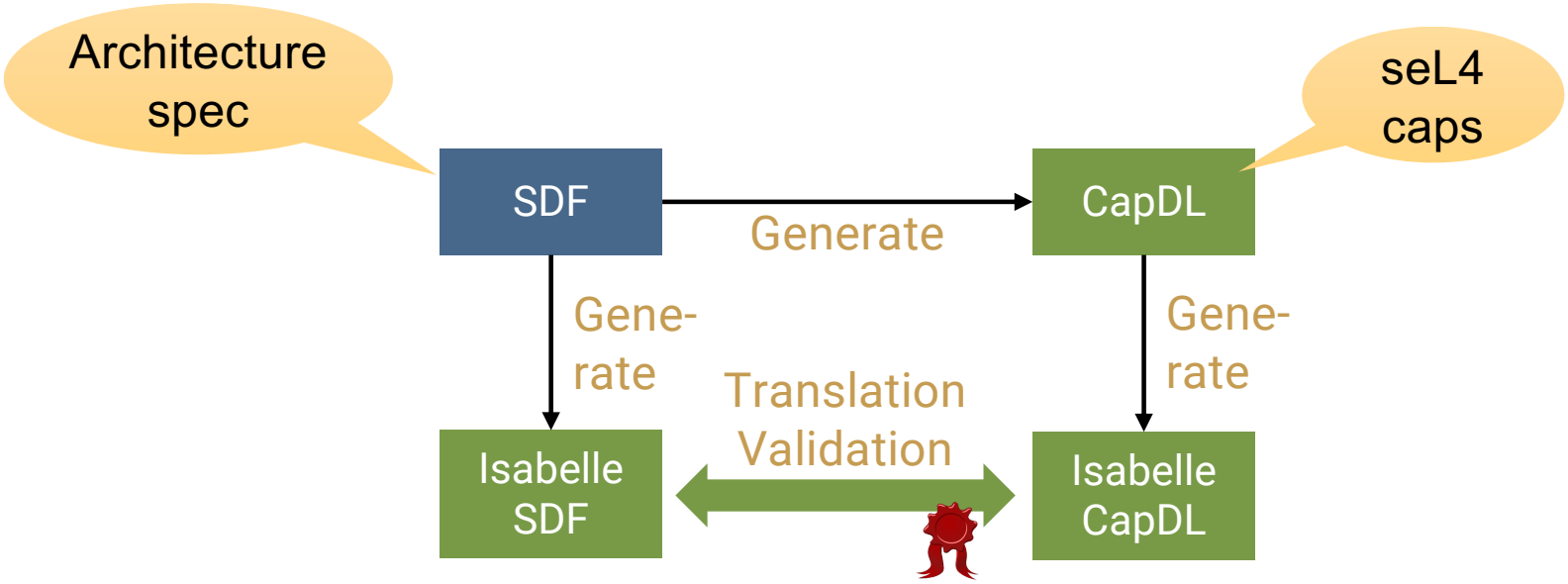
Multicore configuration

How About Verification?

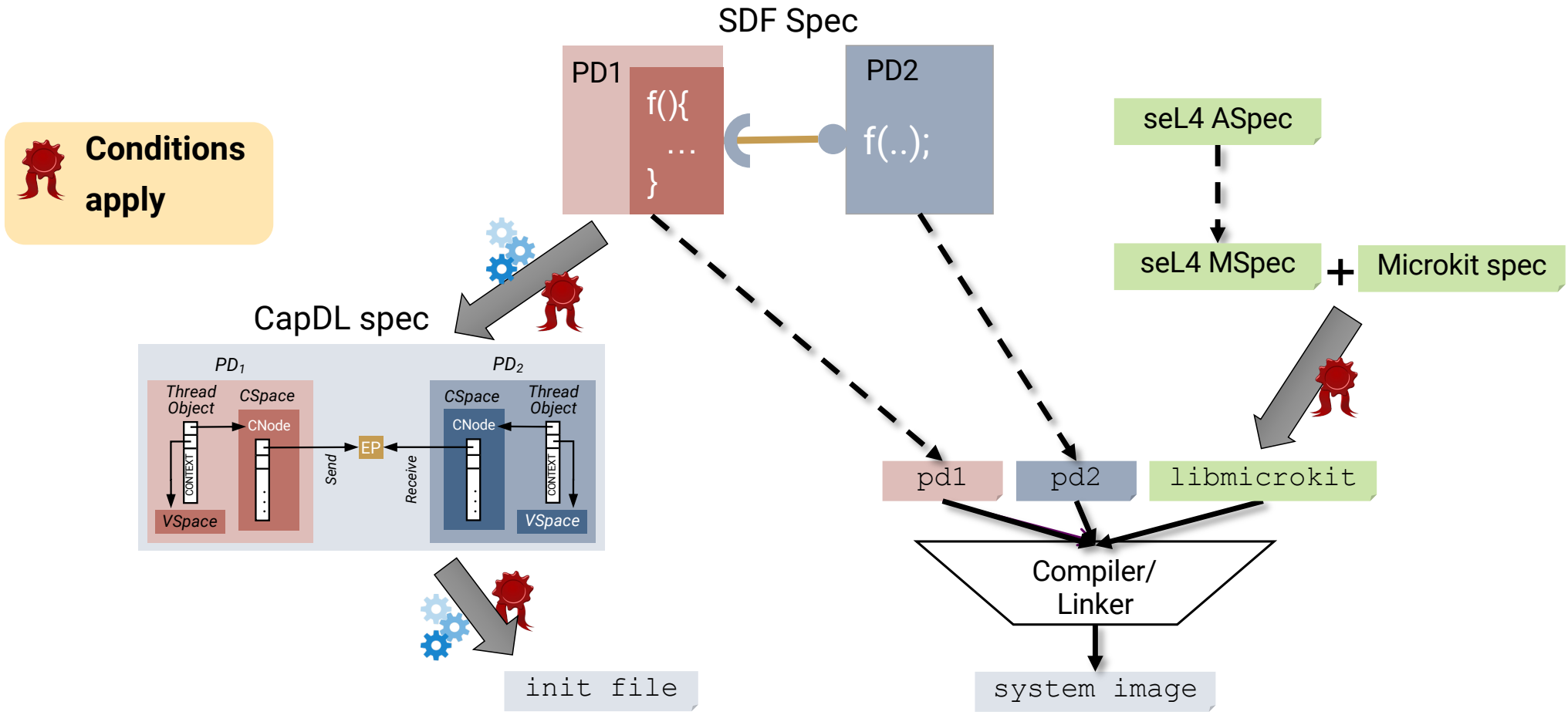
Verifying the Microkit: libmicrokit



Verifying the Microkit: System Initialisation



Microkit Verification in Context



Verifying LionsOS

- Microkit programming model:
  - simple event handlers
  - strictly sequential code

Very little time spent on debugging component logic

Suitable for SMT solvers

- Fine-grained modularity:
  - concurrency by distribution, “tamed” concurrency
  - complex signalling protocols

Protocol bugs are mostly performance problems

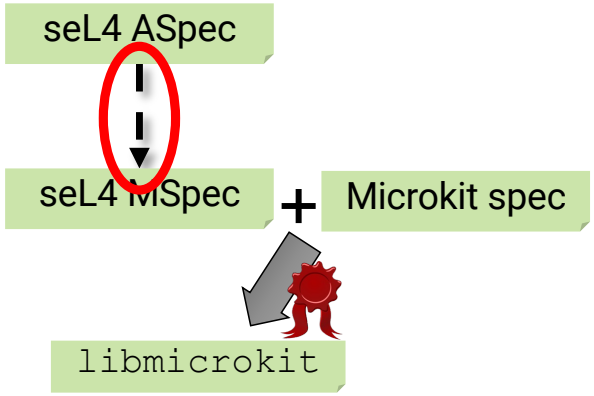
Ideal for model checking!

Automatic proofs!

LionsOS Verification Exploration

- Network-layer protocols *automatically* proved deadlock-free
  - eliminated multiple performance bugs
  - verification supports aggressive optimisation!
- One component (copier) *automatically* verified with SMT solver
  - functional correctness (subject to correctness of neightbours)
    - … but need to improve scalability
  - confident can prove global properties
- Exploring refinement proof of MSpec

Challenge:
- Scalability –many modules to verify
- Proving global security properties!



A Little Helper: Pancake

A language for verifiable system code

Verifying Systems Code

Problem:
- C semantics is complex and ambiguous
- Verifying C code is expensive

How about Rust?
- Strong type safety helps avoiding bugs

But:
- No agreed language semantics
- Huge trusted computing base
  - compiler
  - run time
- Interfacing to hardware needs “unsafe” escapes

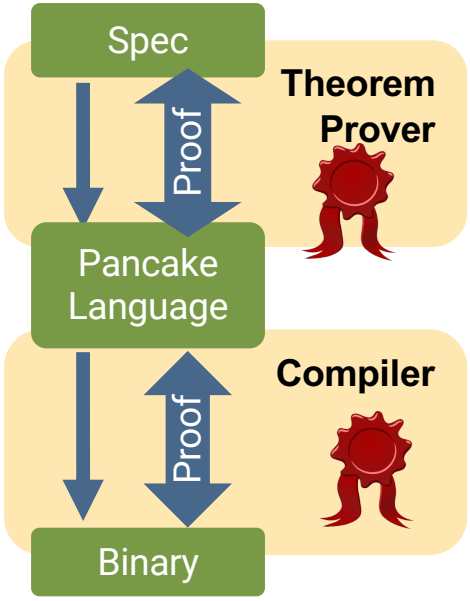
Rust is no help in achieving end-to-end verification!

New Language: Pancake

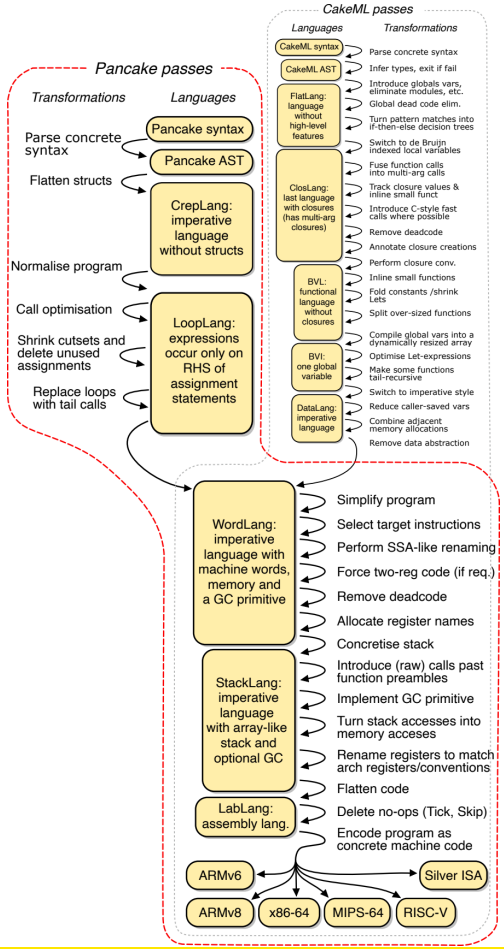
PAN CAKE
A Language for
Verified Systems Programming

TS

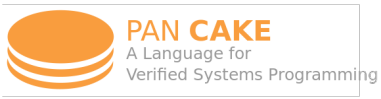
Pancake:
- Programmer-friendly through C-like syntax
- Verification-friendly through simple, well-defined semantics



Verified compiler!
- … leveraging CakeML compiler stack

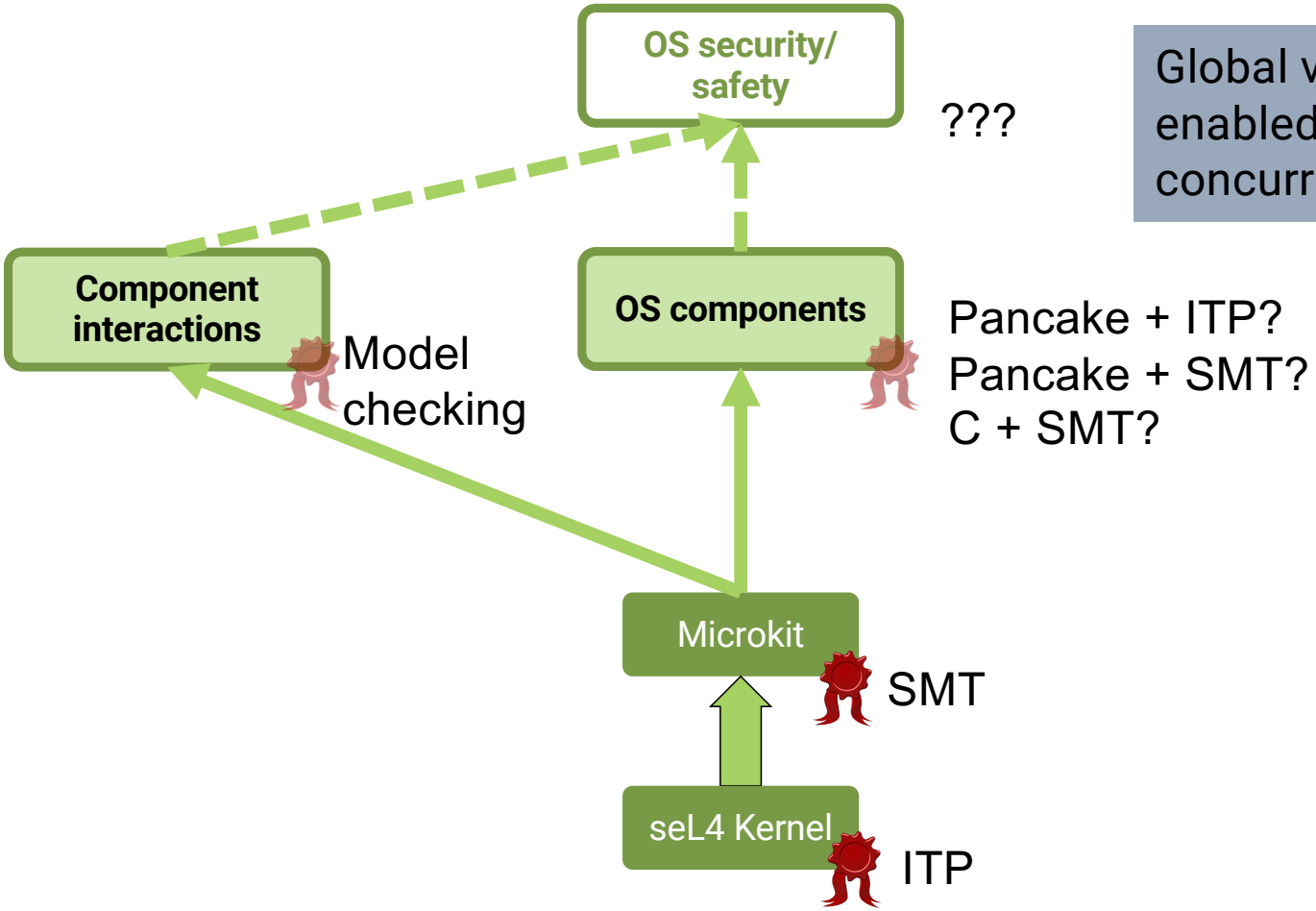


Pancake Status



- Usable language sufficient for implementing Ethernet driver
- Supports shared memory (incl memory-mapped I/O)
    - no FFI escapes needed for accessing device hardware
- Verified compiler!
- *Verification case study*: manual verification of Rx virtualisers
    - Supports liveness reasoning (unlike traditional approaches)
    - Approachable: done by an undergraduate intern
    - 7:1 lines-of-proof:lines-of-code ratio – promising but not yet ideal
- Performance evaluation in progress

Aim: Verified LionsOS

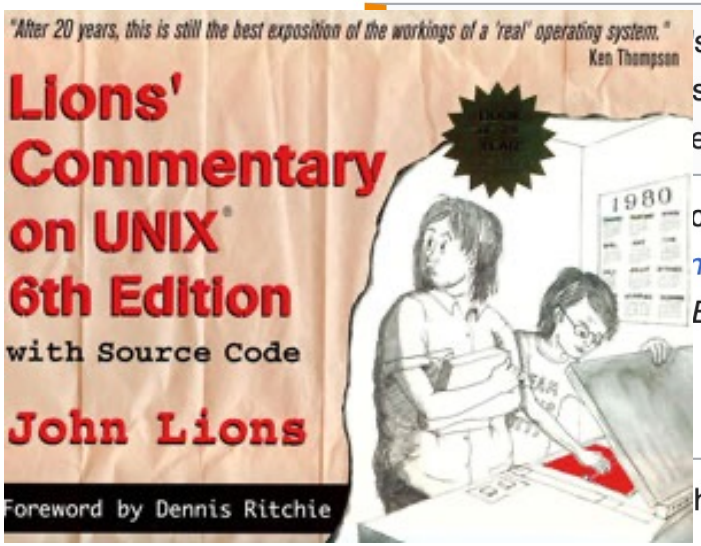


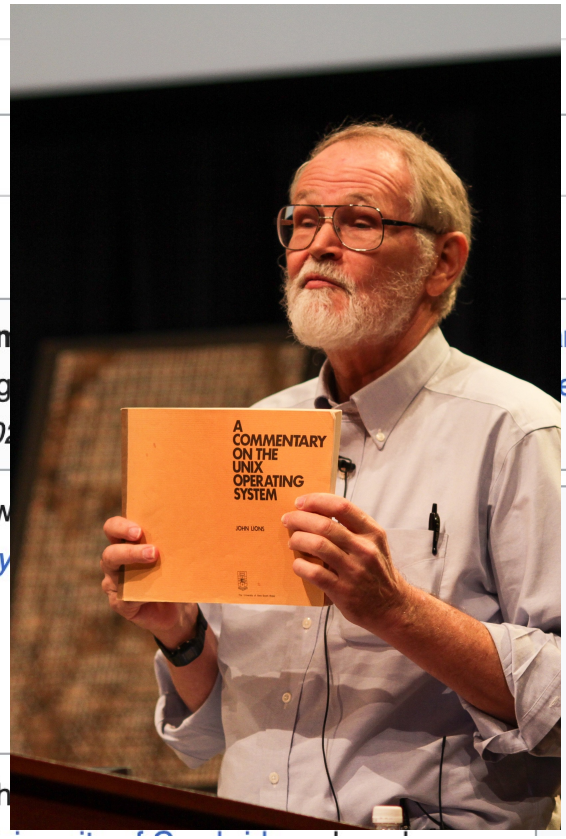
Global verification enabled by “tamed” concurrency

Summary: LionsOS

What's In a Name?

- https://en.wikipedia.org/wiki/John_Lions





John Lions

Article Talk

Read Edit View history

From Wikipedia, the free encyclopedia

4 languag

's lead section n
sider expanding
e. (February 202
cember 1998) w
ns' Commentary
Book.
honours from th

rize the key points.
ew of all important aspects

John Lions

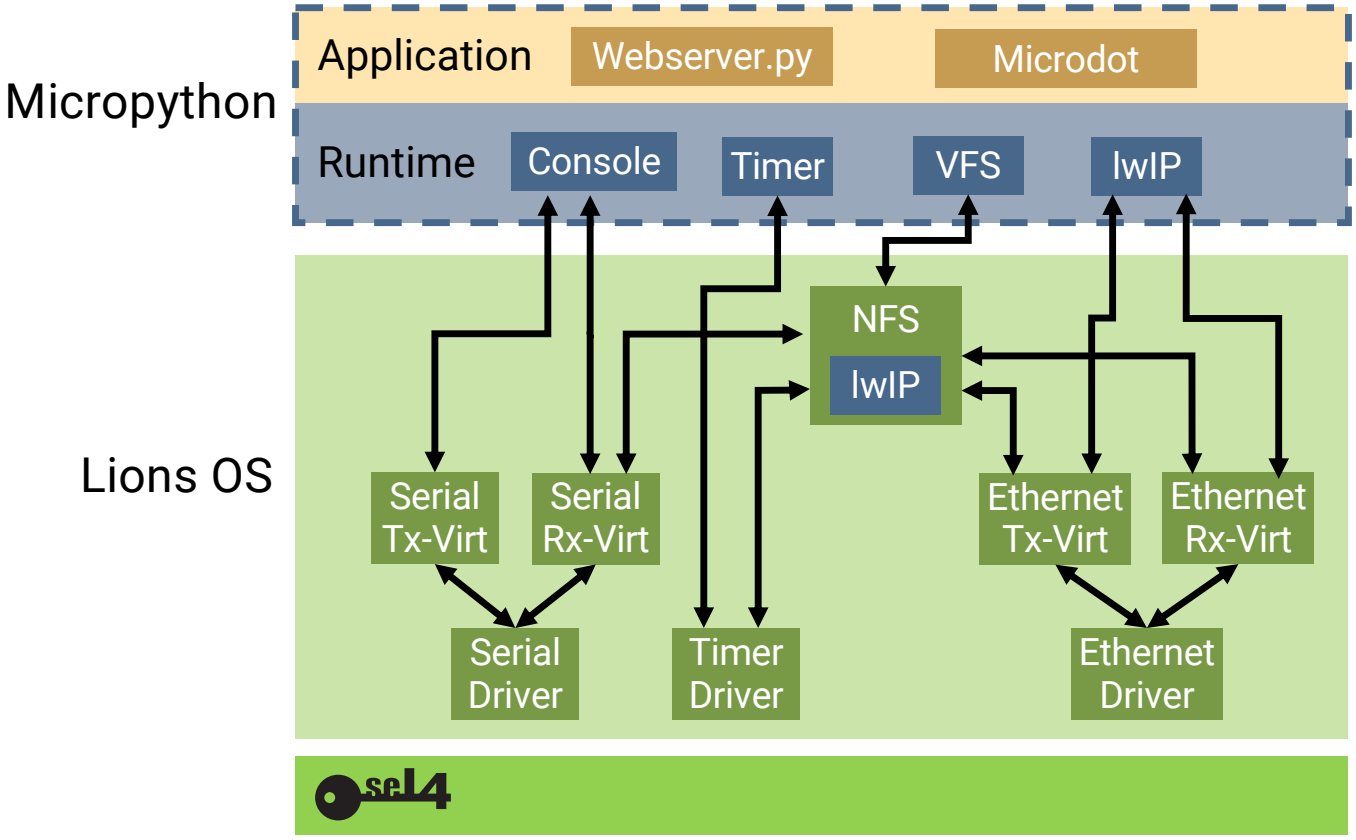
LionsOS Release 0.1 (March'24)

- Native serial, Ethernet, I2C drivers
- Native NFS client, Python interpreter (MicroPython)
- Native components in Rust supported on seL4, Microkit in progress
- Native web server (in Python)
- Driver VMs: graphics, touch screen, audio

Overview: https://trustworthy.systems/projects/LionsOS/
Docs: https://lionsos.org/
Source: https://github.com/au-ts/lionsos/
License: 2-clause BSD

Open Source!

Web Server Based on LionsOS



https://beta.sel4.systems/

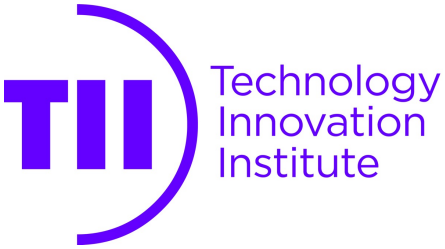
LionsOS Support

- NIO America

- DARPA PROVERS program collaborating with Collins Aerospace

Foundations (Microkit, device driver framework) supported by:

in association with
National Cyber Security Centre



Security is no excuse for bad performance!

https://trustworthy.systems