



School of Computer Science & Engineering  
**Trustworthy Systems Group**

# Lions OS

## Secure – Fast – Adaptable

**Gernot Heiser**

[gernot@unsw.edu.au](mailto:gernot@unsw.edu.au)

[@gernot@discuss.systems](mailto:@gernot@discuss.systems)

<https://microkerneldude.org/>



# August 2009



A NICTA bejelentette a világ első, formális módszerekkel igazolt,



▶ [Stories](#) [Recent](#) [Popular](#) Search

Slashdot is powered by [your submissions](#)

+ - Technology: **World's First**

Posted by [Soulskill](#) on Thursday Aug 27, 2009 10:00 AM  
from the wait-for-it dept.

An anonymous reader writes

"Operating systems usually have bugs and so forth are known by almost everyone. It's hard to [prove that a particular OS kernel is](#) formally verified, and as such it's not surprising that researchers used an executable model checker, the Isabelle theorem prover to generate a formal proof that matches the executable and the kernel code.

Does it run Linux? ["We're pleased to say that it does."](#)



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

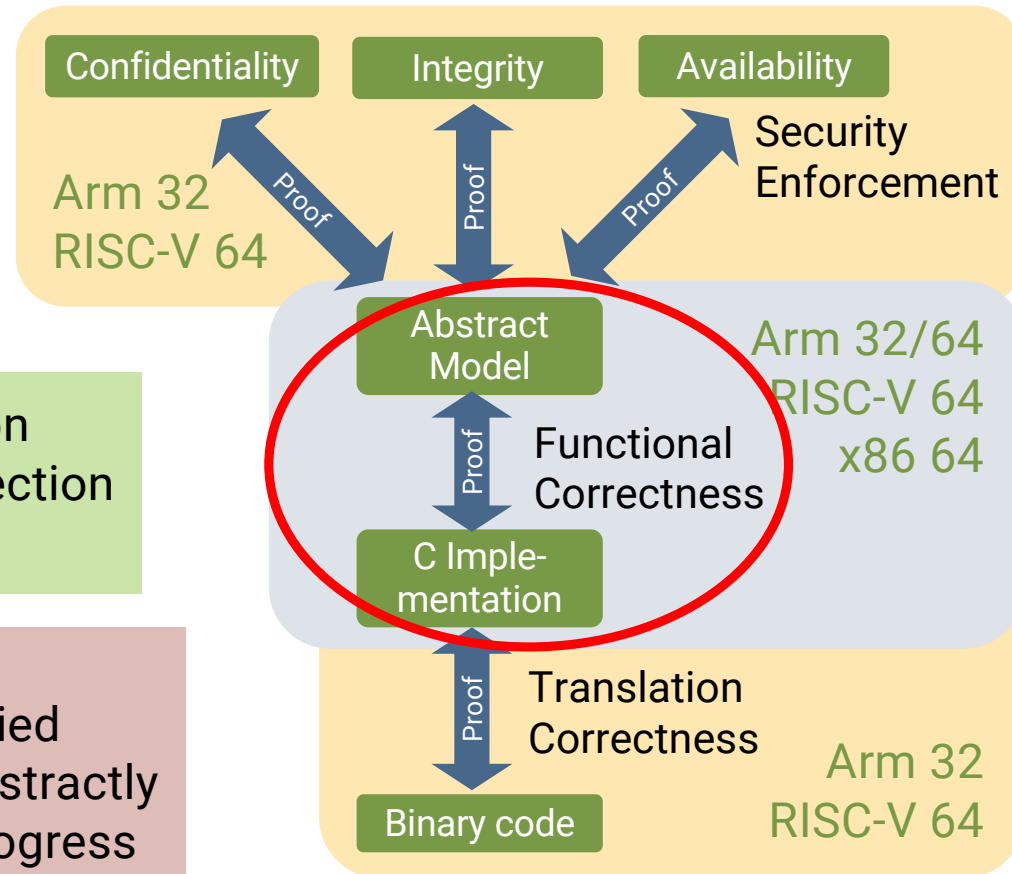
**FLAWS** in the code, or "kernel", that sits at the heart of modern computers leave them prone to occasional malfunction and vulnerable to attack by worms and viruses. So the development of a secure general-purpose microkernel could pave the

way to a more secure system, 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

eredemenyekeppen pedig egy olyan megbízhatóságot kapnak a szoftvertől, amely e

# seL4 What Is This About?



- Comprehensive formal verification
- Capabilities for fine-grained protection
- World's fastest microkernel

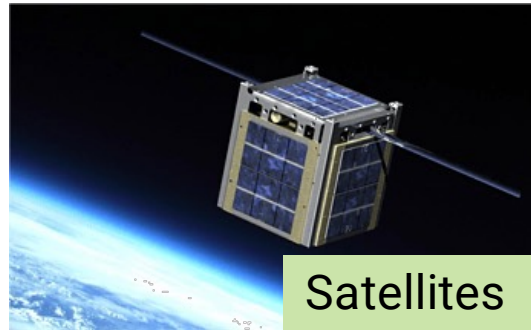
## Present limitations

- initialisation code not verified
- MMU, caches modelled abstractly
- Multicore verification in progress

# se14 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



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 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
- ~Sep'24: Commercial car

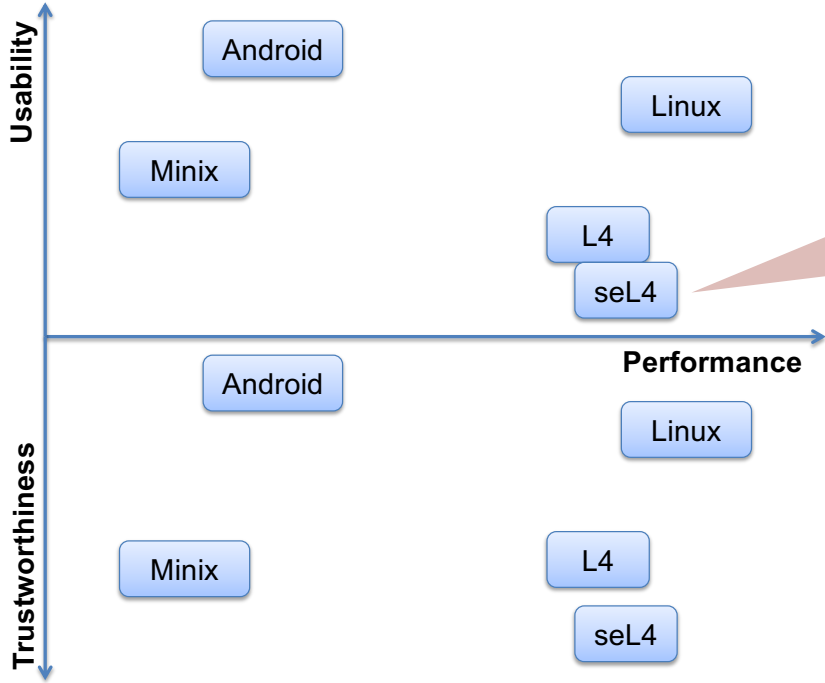


# seL4 is cool!

... but why hasn't it taken over the world?

# A Slide from Feb'15

## OS Trade-Offs



“With seL4 we have reached new heights of security and **unusability**”

©2015 Gernot Heiser, NICTA



# Is This a Problem?




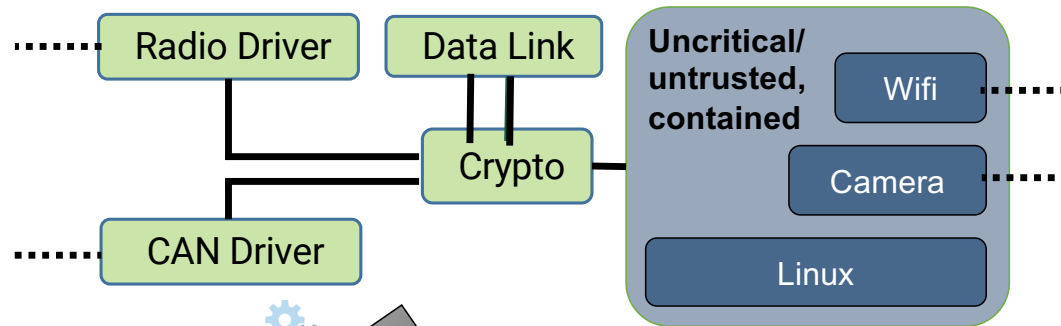
Microkernel's job is secure HW multiplexing

- Minimal but general mechanisms
- Policy freedom
- Usability is for user-level frameworks!

# User-level Framework: CAmkES

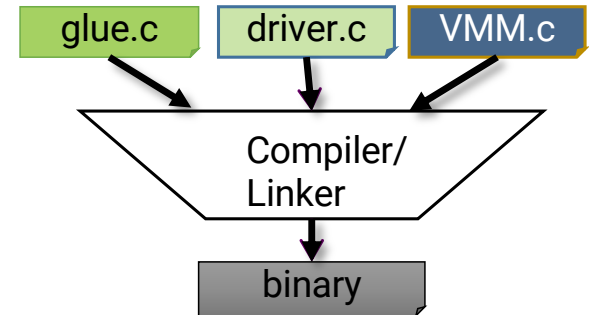
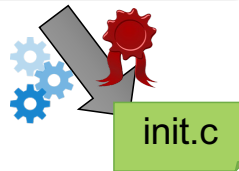
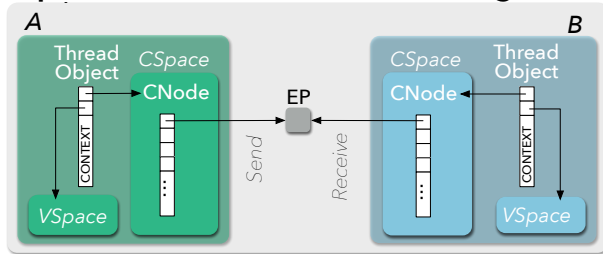


 **Conditions apply**



Architecture specification

CapDL: Low-level access rights





# CAmkES Critique

- ✓ simple, intuitive model
- 👎 ... but encourages sub-optimal use of seL4 mechanisms
- ✓ partial verification story (verified initialiser generated from CapDL)
- 👎 forces use of (very complex) kernel build system – hard to use!
- 👎 rigid, totally static, hard to extend
- 👎 high overheads

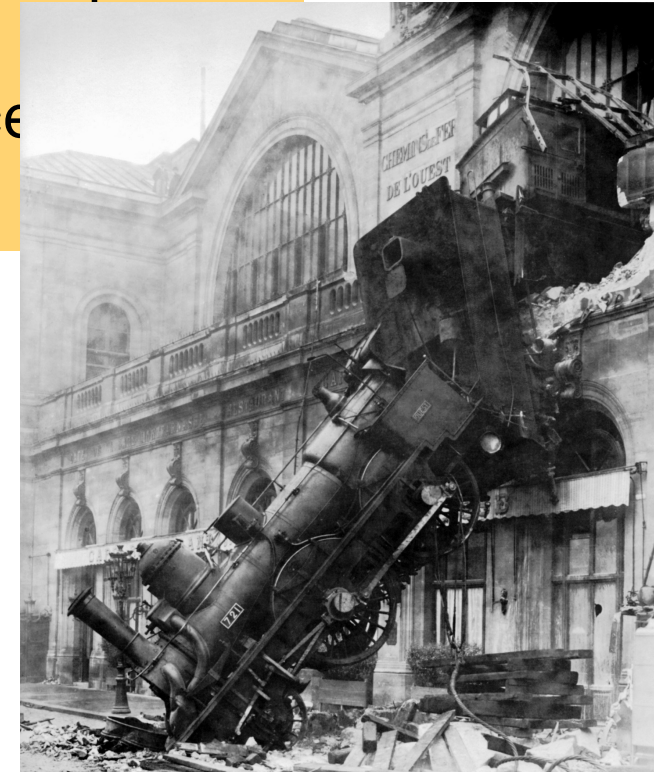
# seL4 Experience of the First 10+ Years



seL4's assurance and power still unrivalled, but...

- Designing seL4-based systems well requires deep expertise
- CAmkES didn't help, plenty of bad designs
- CAmkES overheads overpower kernel performance
- Kernel build system is unsuitable for most users

- Need an SDK
- Need an OS!



# Enter Lions OS

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:** *Most secure* OS ever

# Really – an OS Built from Scratch?



**Yes – if we strictly observe the KISS principle: Keep it simple, stupid!**

- Fine-grained modularity, strong *separation of concerns*
- Least privilege
- *Radical Simplicity*<sup>™</sup>: provide *only* the features needed
- *Use-case specific policy* (rather than universal policy)

Software engineering 101

Reason about security

KISS extreme

“Universal” policies are complex, always have pathological cases

Also limiting scope allows use of static architecture

Use-case diversity by swapping (policy) modules!

# First Step: The seL4 Microkit

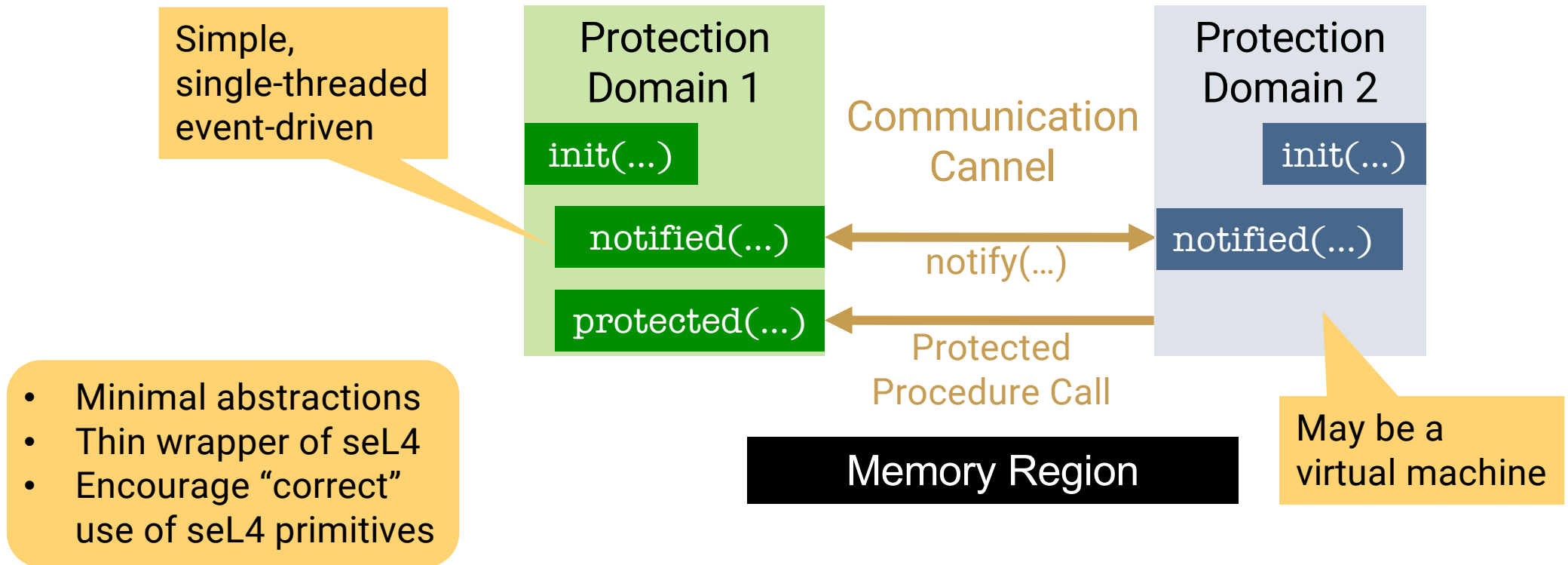


## Minimal base for IoT, cyberphysical, other embedded use:

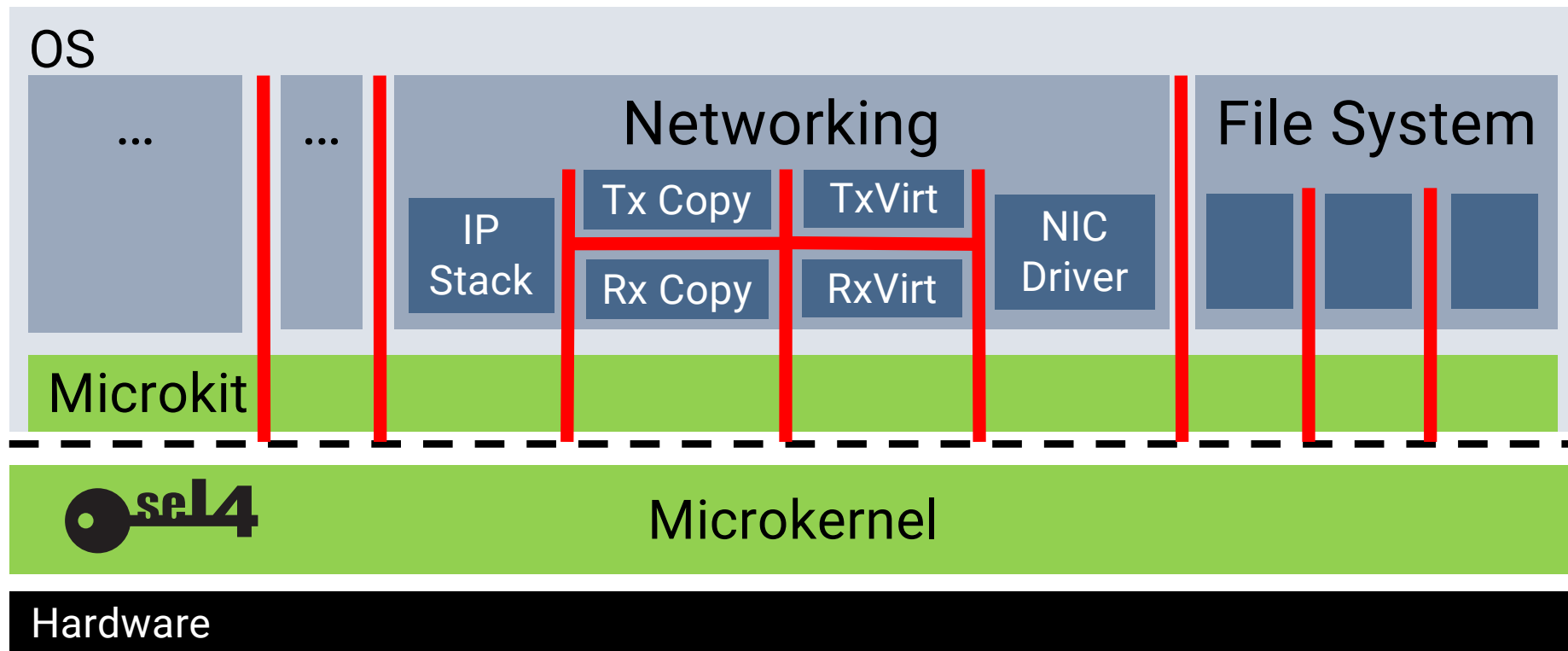
- Restrict to static *architectures* (more general than CAmkES)
  - i.e. components & communication channels defined at build time
  - ... but can stop/restart/reload/late-load components
- Ease development and deployment
  - SDK, integrate with build system of your choice
- Retain minimal trusted computing base (TCB)
  - *TCB suitable for formal verification*
- Retain seL4's superior performance



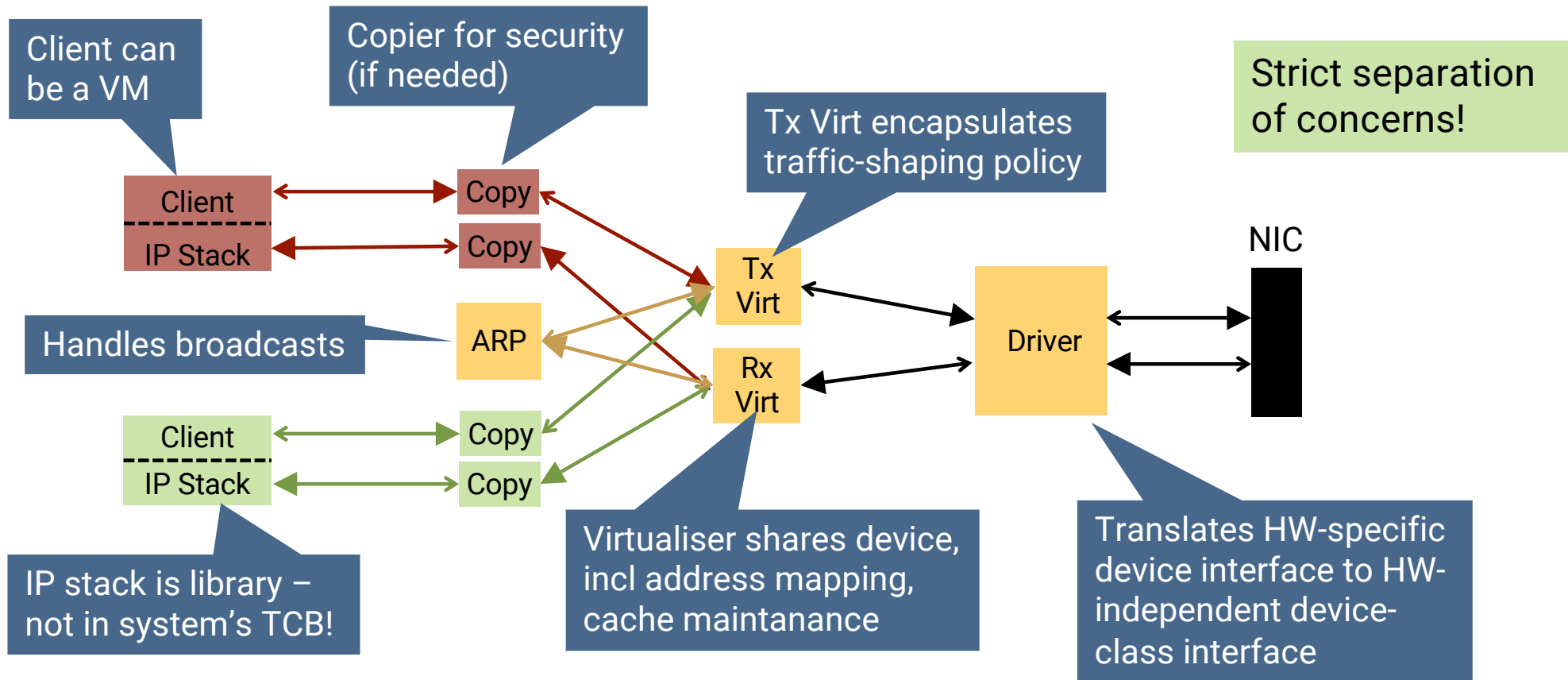
# Microkit Abstractions (Yes, that's all!)



# Lions OS: Modular System on Microkit

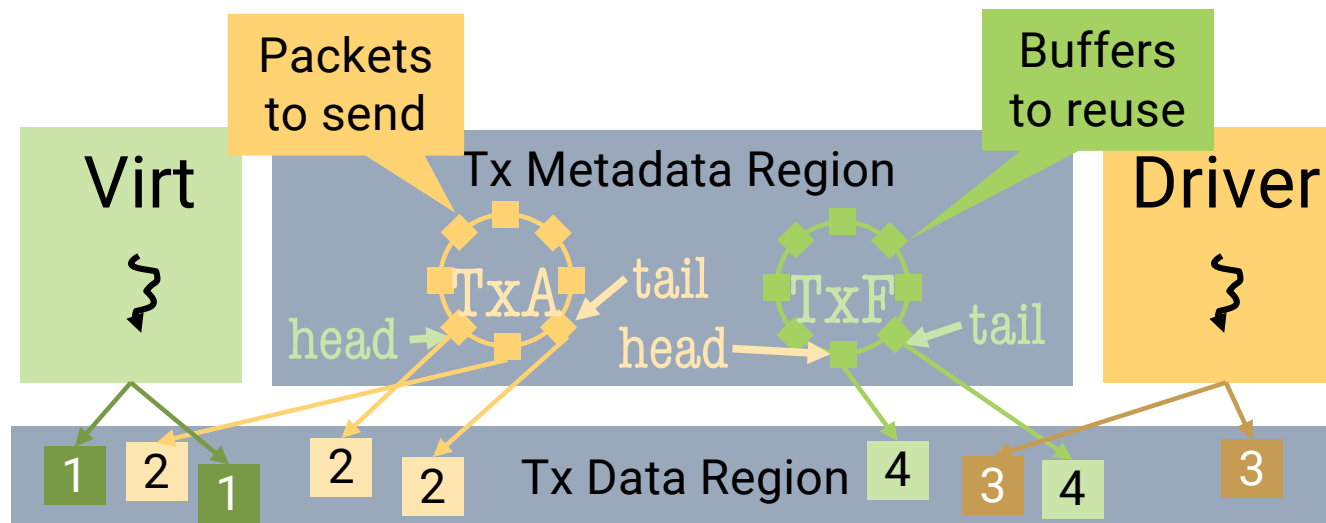


# Mature Part: Networking Subsystem

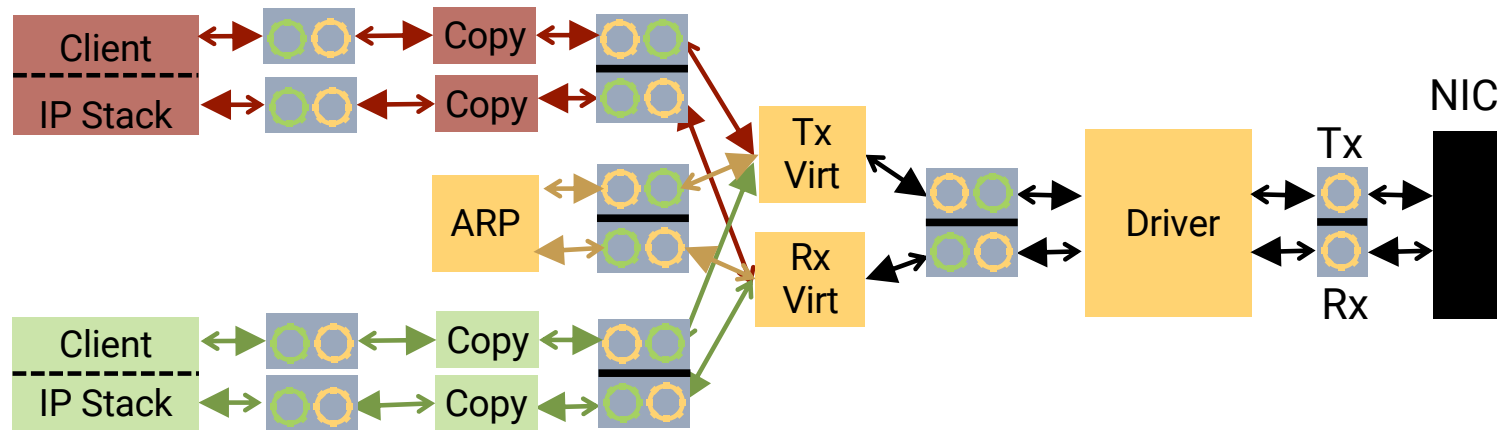


# Zero-copy Data Transfer

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



# Networking Detail

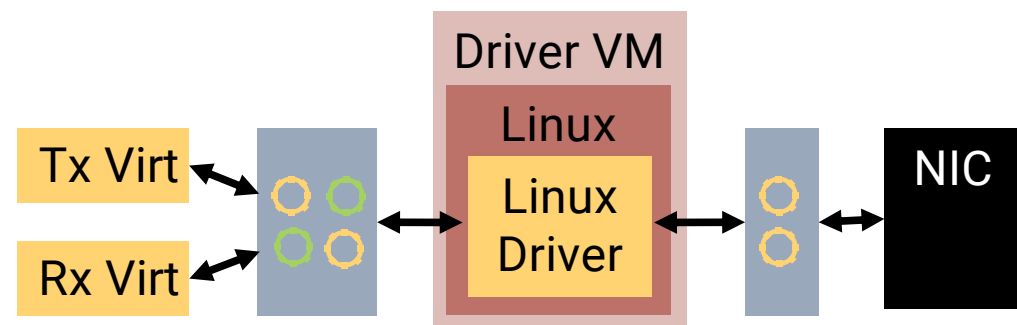


# Legacy Re-use: Driver VMs



Use Linux UIO for:

- allowing Linux user-mode component to forward requests to in-kernel driver
- develop Lions-OS components on Linux



# Comparison to Linux (i.MX8M)



## Linux:

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

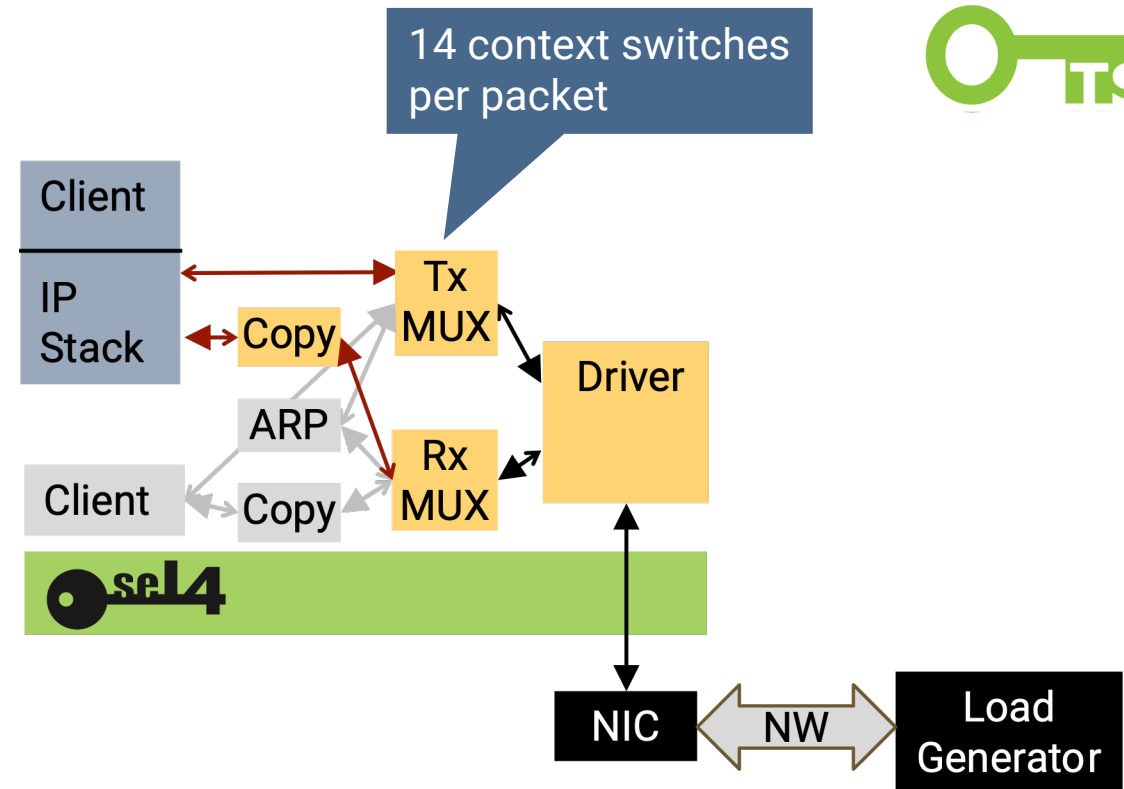
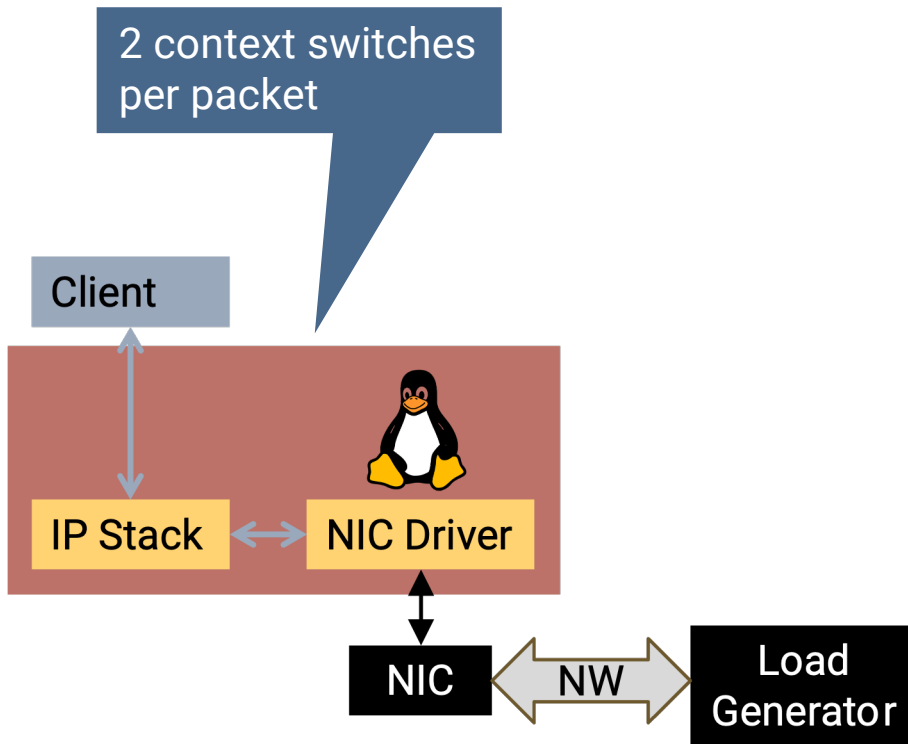
Performance?

## Lions OS:

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

Written by second-year student!

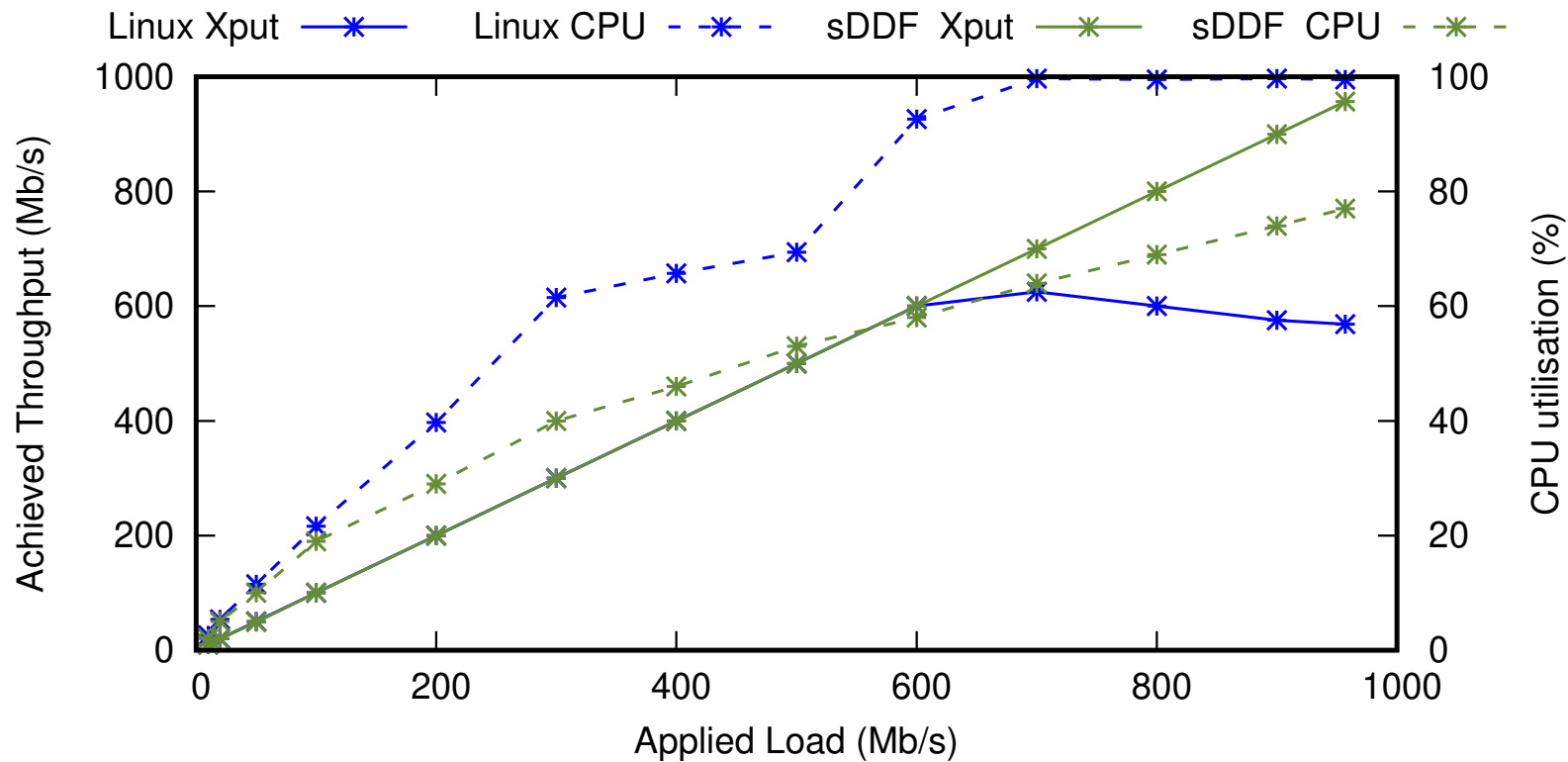
# Evaluation Setup



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

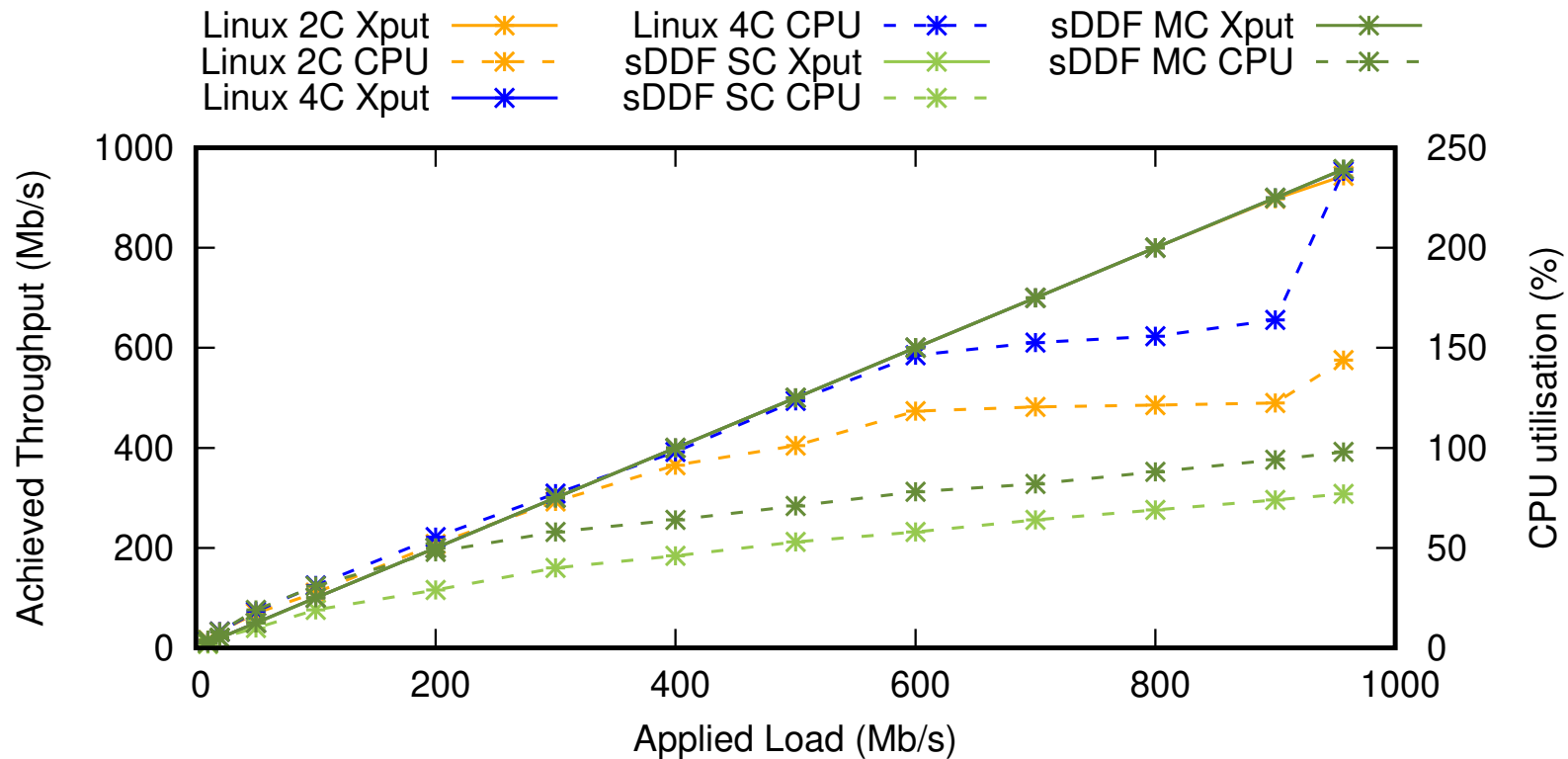


# Performance: i.MX8M, 1Gb/s Ethernet



Single-core configuration

# Performance: i.MX8M, 1Gb/s Ethernet

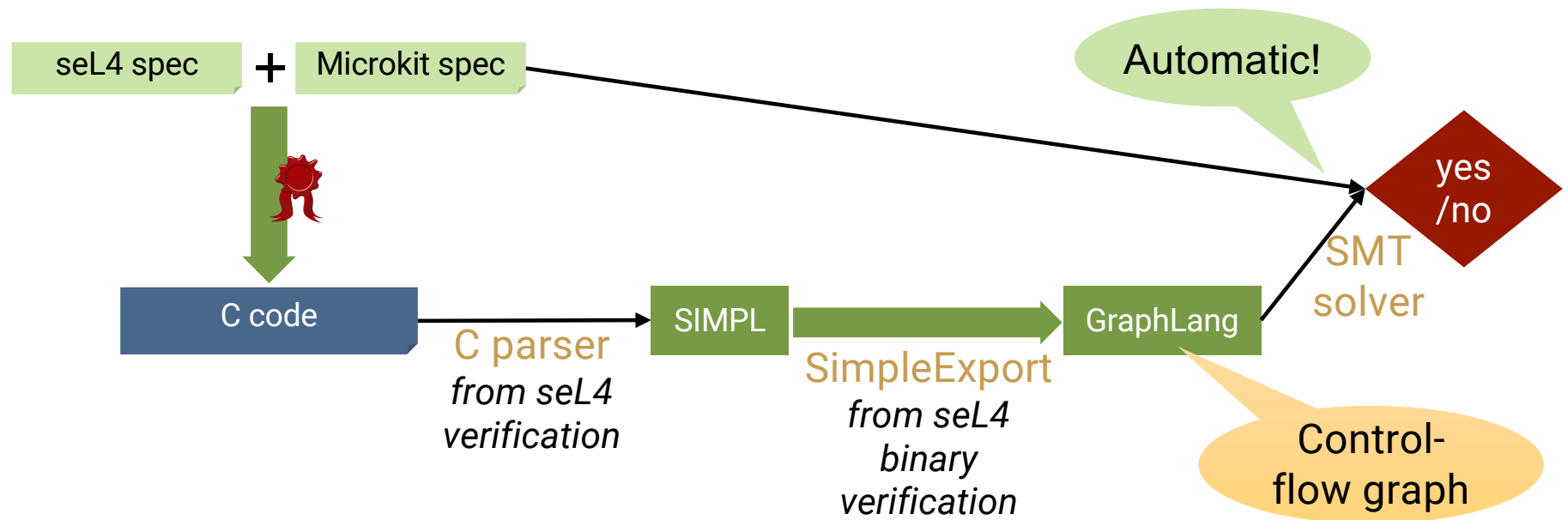


Multicore configuration

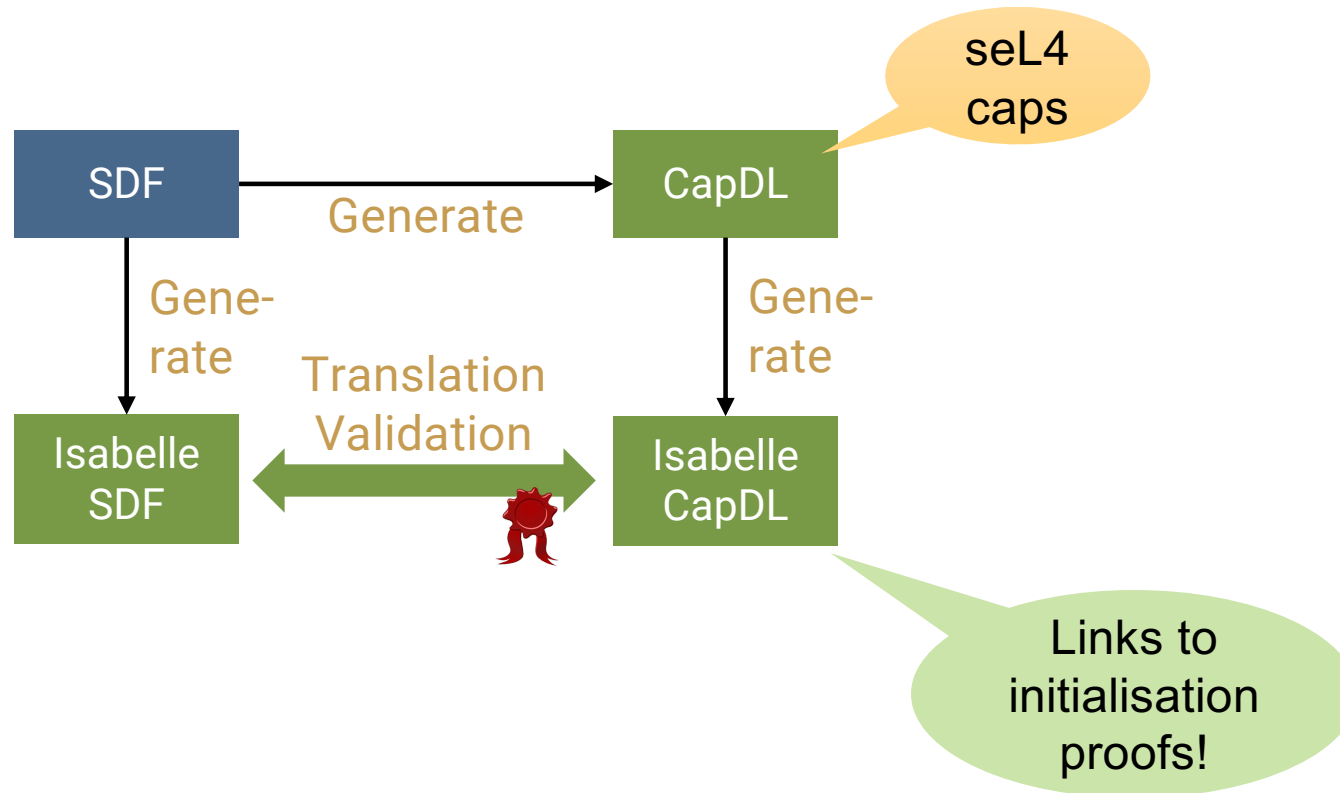


# How About Verification?

# Verifying the Microkit: libmicrokit




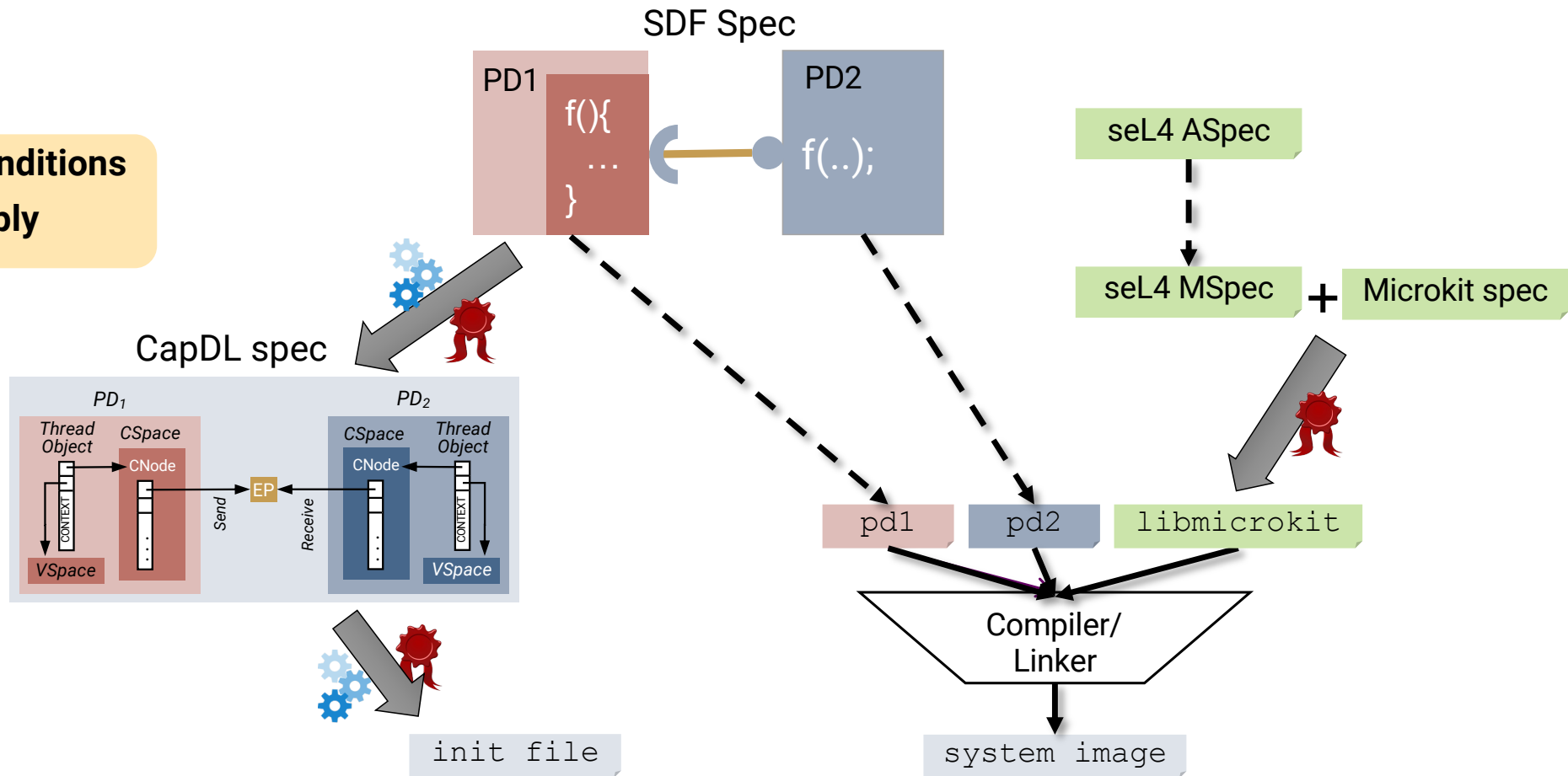
# Verifying the Microkit: System Initialisation



# Microkit Verification in Context



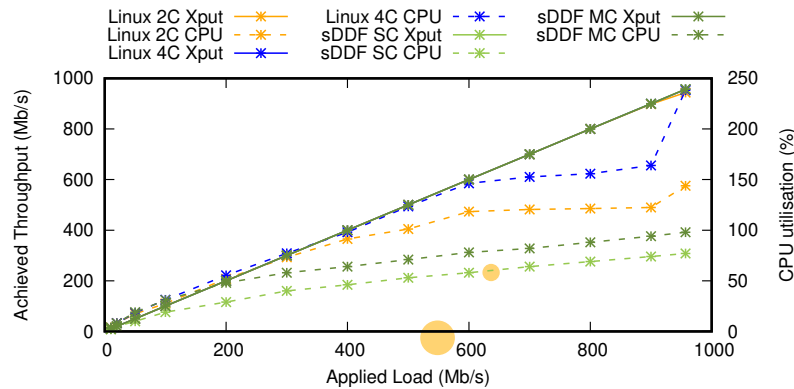
 **Conditions apply**





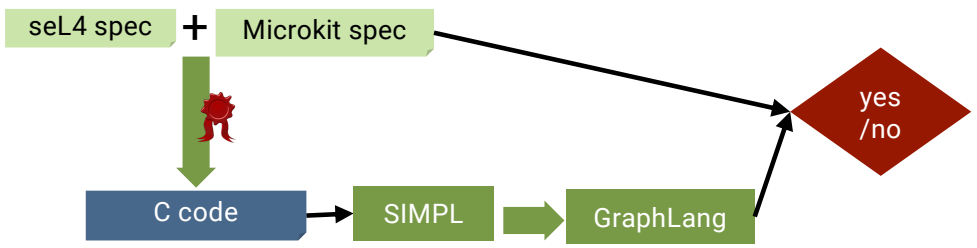
# Stepping Back

# What Does This Mean For Lions OS?



We can **verify** a highly modular, OS!

We can **build** a highly modular, yet performant OS!





# Verifying Lions OS

- 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
  - complex signalling protocols

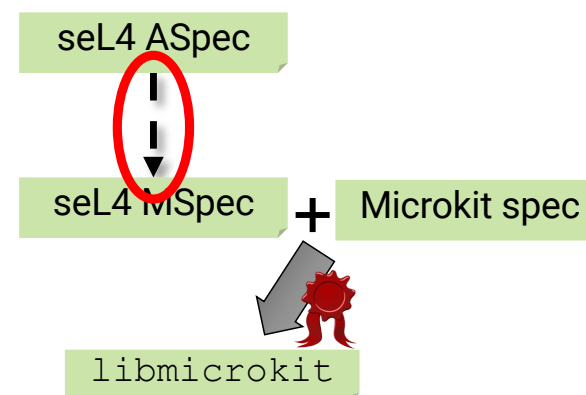
Protocol bugs are mostly performance problems

Ideal for model checking!

Automatic proofs!

# Lions OS Verification Status

- 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 neighbours)
  - confident can prove global properties
- Exploring refinement proof of MSpec



# Lions OS Support



- UK National Cyber Security Centre (NCSC)



in association with  
National Cyber  
Security Centre

- NIO America



- DARPA PROVERS program  
collaborating with Collins Aerospace



- *more in pipeline*

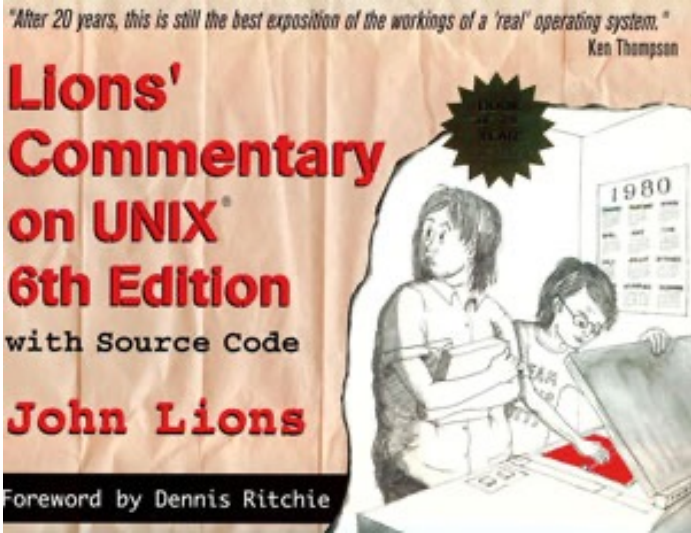
# What's In a Name?

- [https://en.wikipedia.org/wiki/John\\_Lions](https://en.wikipedia.org/wiki/John_Lions)

## John Lions

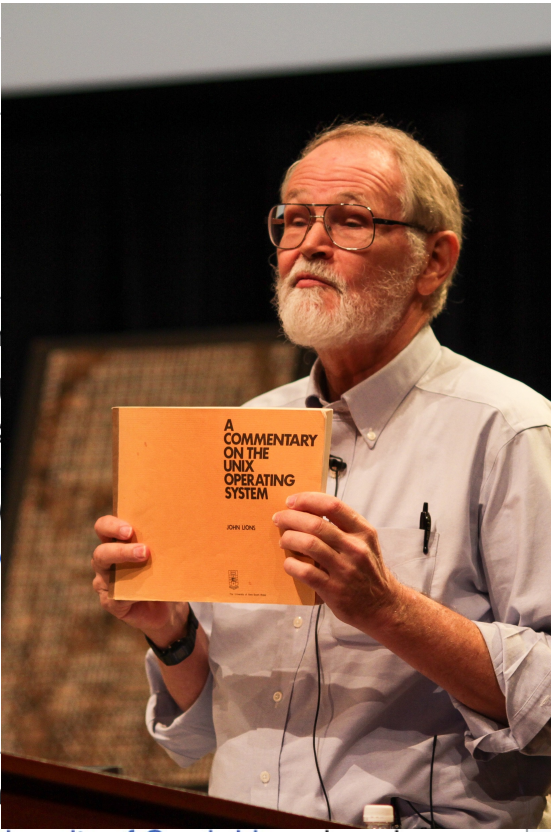
[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia



...s lead section m  
sider expanding  
e. (February 20  
cember 1998) w  
ns' Commentary  
Book.

honours from th



 4 languages

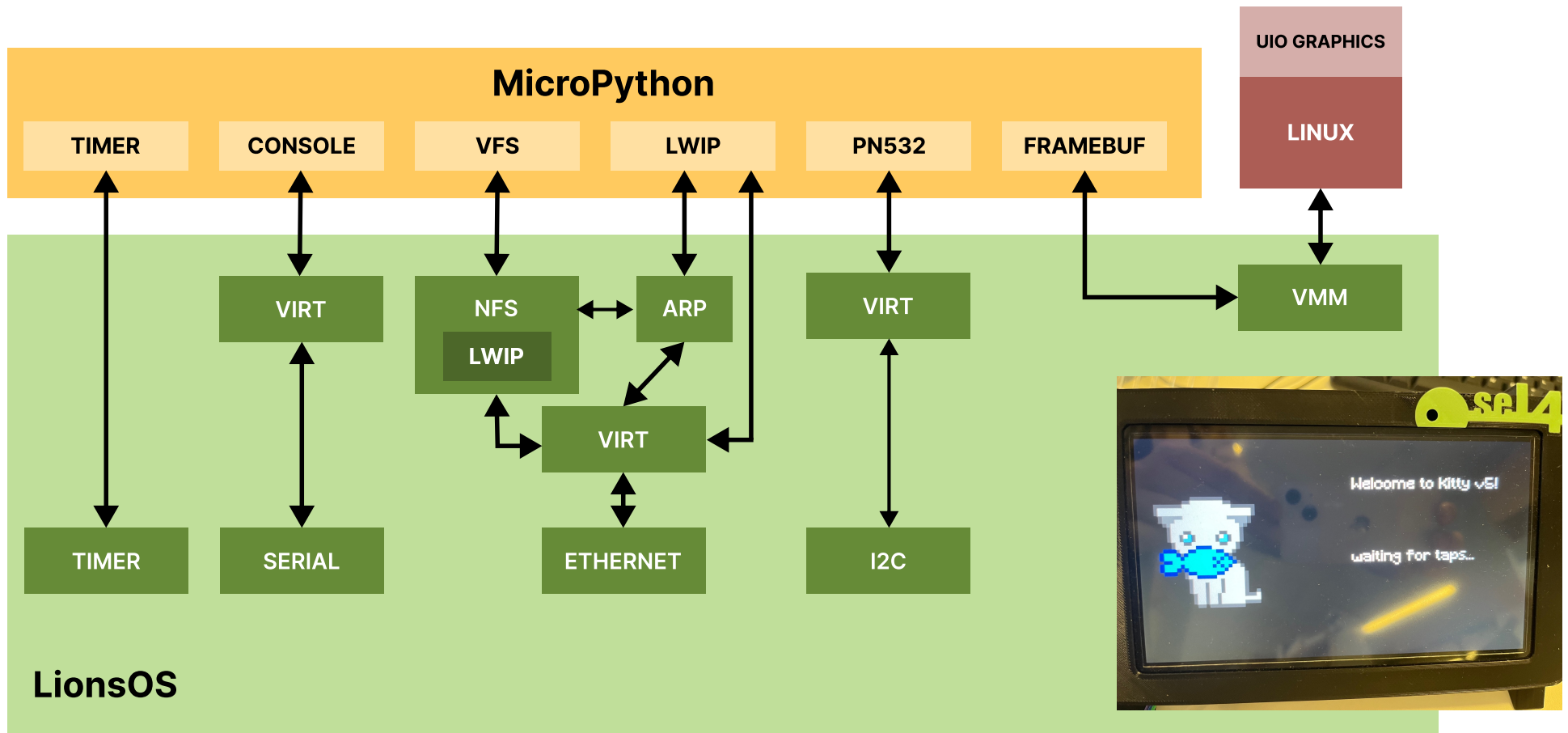
[Read](#) [Edit](#) [View history](#) [Tools](#)

...rize the key points.  
...ew of all important aspects

**John Lions**



# Lions OS Reference System: TS “Kitty”





# Announcing: Lions OS Release 0.1

- 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



Security is no excuse  
for bad performance!



<https://trustworthy.systems>