School of Computer Science & Engineering

Trustworthy Systems Group

Trustworthy Systems R&D Update

Gernot Heiser

UNSW and seL4 Foundation

gernot@unsw.edu.au

Agenda

- Lions OS
- Push-button verification of OS components
- Pancake: systems language with verified compiler
- Other developments

Lions Operating System

seL4 Principles

Result: High barrier to uptake!

Security:
- Suitable base for security-critical systems
- Provably correct and secure

Proper microkernel:
- Minimal
- Provides policy-free mechanisms only
- Single access-control mechanism: Capabilities

Performance:
- Security is no excuse for poor performance!
- Don't pay for what you don't use

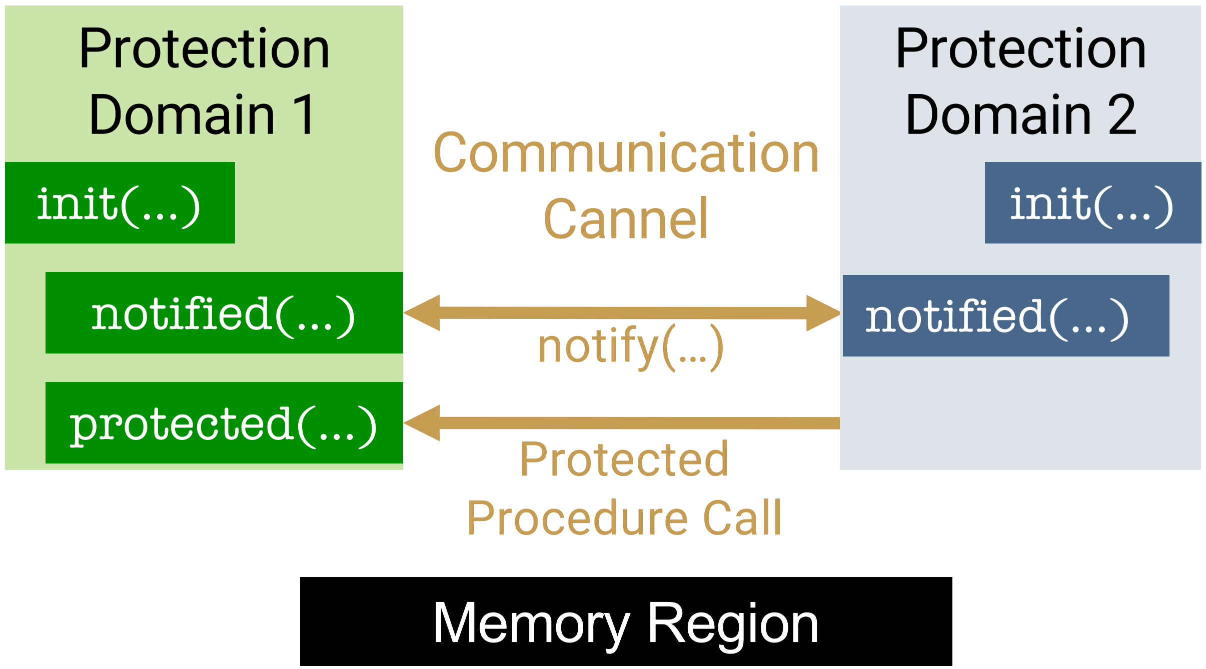
Anti-Principles:
- Hardware abstraction
- Prevent foot guns
- Usability

User-level issue!

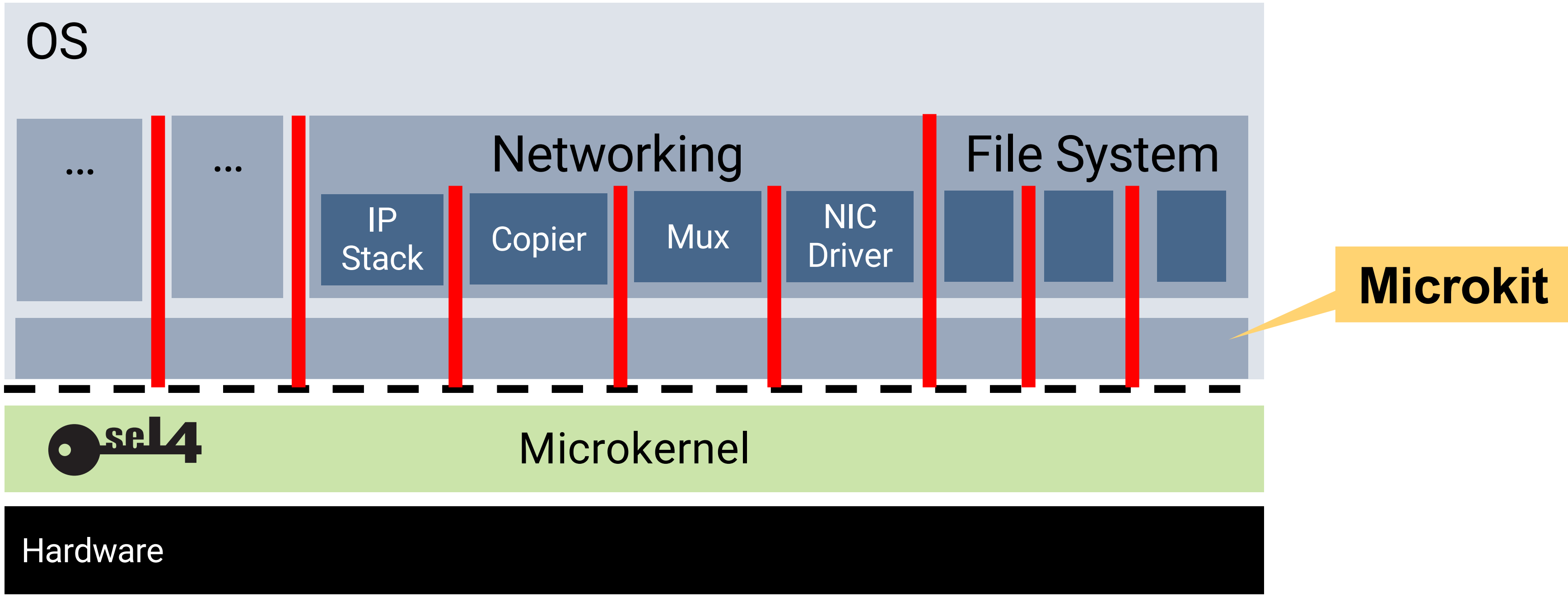
The microkernel is the assembly language of operating systems!

Taming seL4: The Microkit

- Minimal abstractions
- Thin wrapper of seL4
- Encourage “correct” use of seL4 primitives
- For IoT/cyberphysical
- More in Ivan’s talk!



Lions OS: Highly Modular OS on Microkit



Lions OS: Aims

Fast:
Best-performing microkernel-based OS ever

Secure:
Most secure real-world OS ever

Adaptable:
Suitable for a wide range of cyberphysical / IoT / embedded systems

Lions OS: Principles

Least Privilege

Strict separation of concerns

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

Radical simplicity

Use-case–specific policies

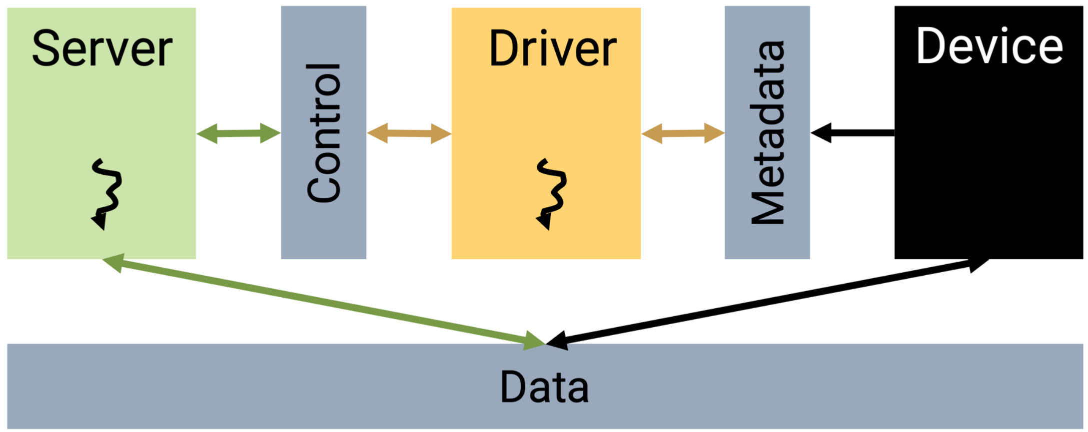
Design for verification

Least Privilege

Time-honoured security principle
[Saltzer & Schroeder, 1975]

sDDF example

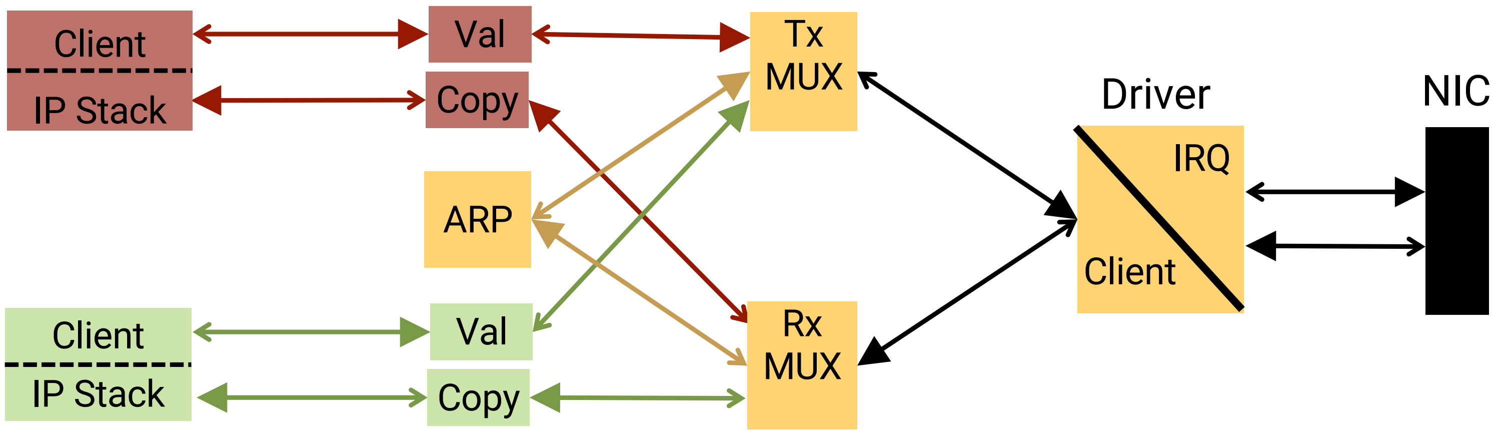
- Driver model uses 3 different memory regions
- Notifications signal updates to these regions



Future:
- **mandatory policy:** static architecture
- **discretionary policy:** per-component rights, subject to mandatory policy

Strict Separation of Concerns

Each component has one and only one job!



Radical Simplicity™

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

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

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

Design for Verification

Verification enabled by:

- modularity
- radical simplicity

Lions OS: Status

- have partial funding, looking for more
- networking layer: mature design, fine-tuning implementation
- prototype SDcard storage, NFS client, touch screen
- working on virtualised graphics

Lions OS: Timeline

- Q4’23: First release of OS prototype
  - minimal OS services, some using preliminary design
  - with point-of-sale reference system
  - debugging & profiling support inherited from Microkit [Ivan’s talk]
- Q1’24: Release of matured, documented PoS system
  - initial experience verifying components
- Q4’24: Verification of key components of OS
- Q3’25: Complete & mature OS [subject to funding!]

Push-Button Verification

What is Push-Button Verification?

Interactive Theorem Proving (ITP):

- manually writing proofs in proof assistant (Isabelle)
- powerful but labour-intensive
- used for seL4 functional correctness proof

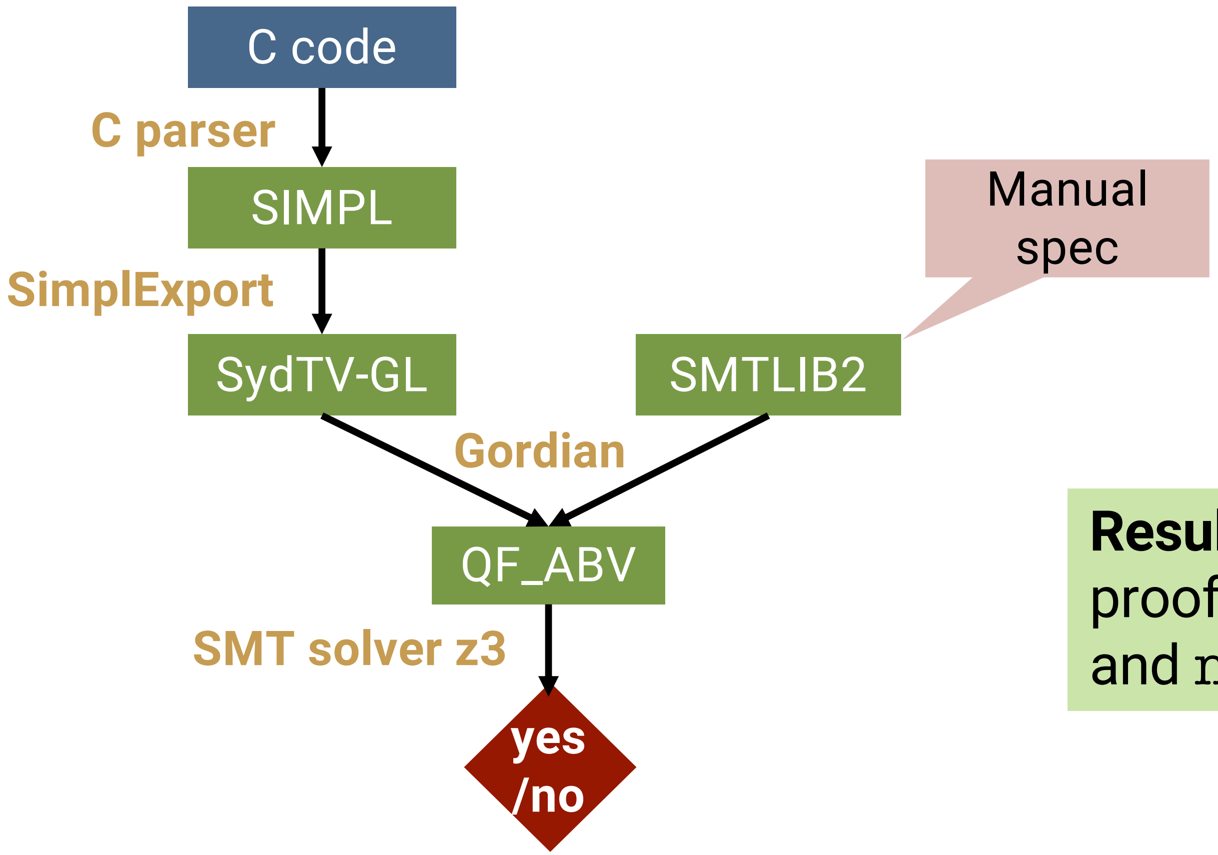
Push-Button (SMT) Techniques:

- specify correctness condition
- state-exploration tool (SMT solver)
    - proves correctness, or
    - finds counter-example, or
    - times out
- fiddly but fast (when it works)
- can only prove simple properties
- used for seL4 translation-correctness proof

Model checking:

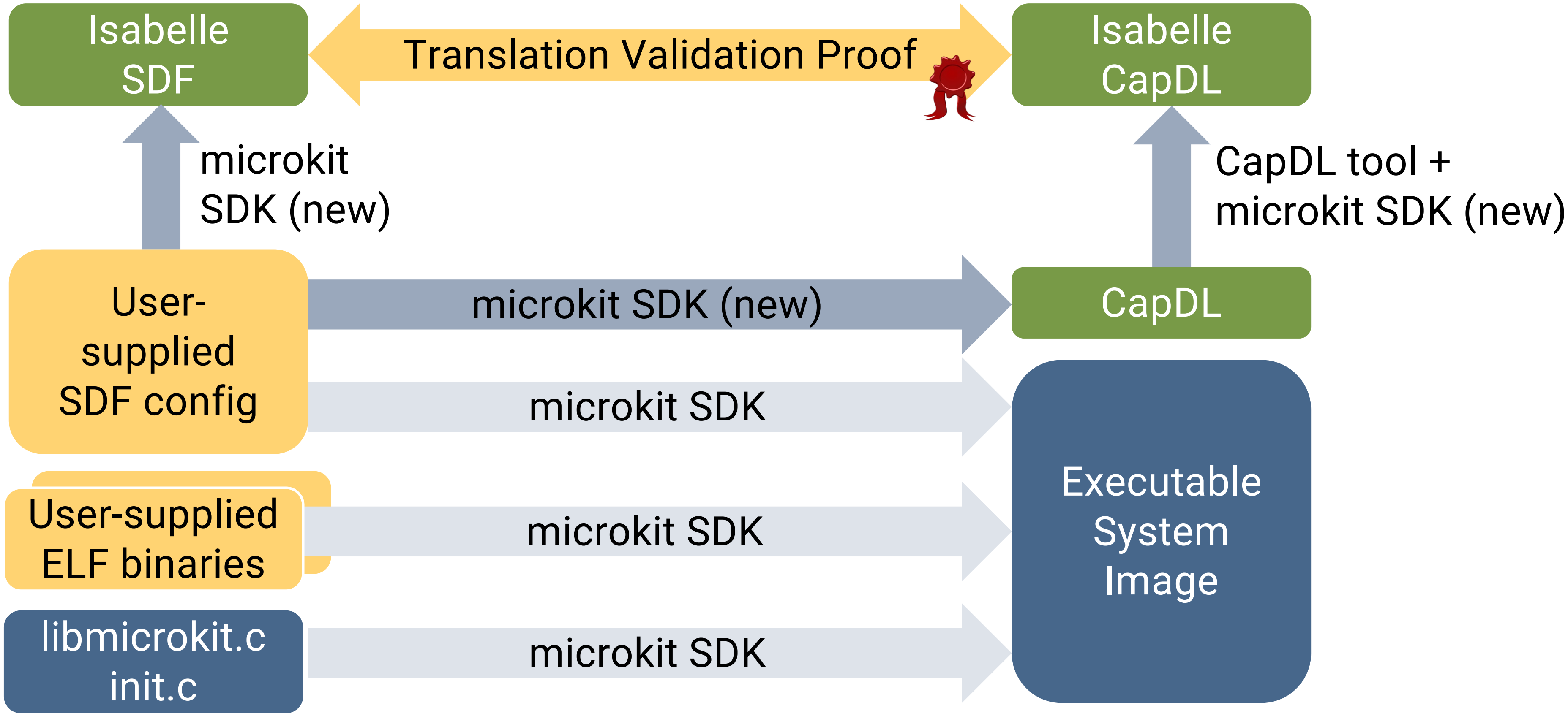
- specify (simple!) model of system
- exhaustively search state space
- more limited than SMT
- used for proving liveness of protocols etc

Verifying the Microkit Implementation

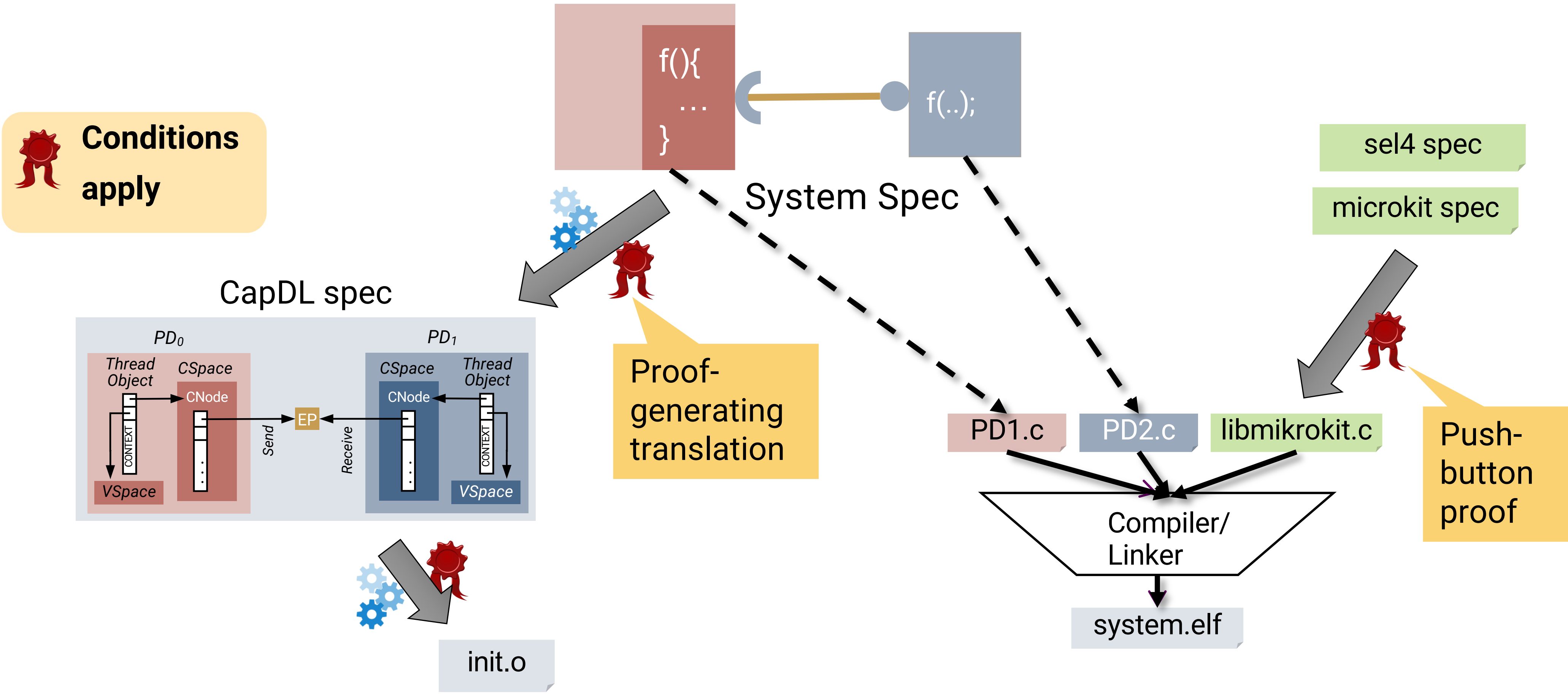


Result: Functional correctness proof of (original) `libmicrokit` and `monitor` PD

Verifying Microkit Initialisation



Microkit Verification



Present Microkit Verification Gaps

Functional correctness:

- Presently for original (fully static) version

in progress

System initialisation:

- CapDL mismatch (64 vs 32-bit, MCS vs non-MCS)

needs AArch64, MCS kernel verification

Spec gap:

- SMT seL4 spec is an unverified “projection” of seL4 Abstract Spec

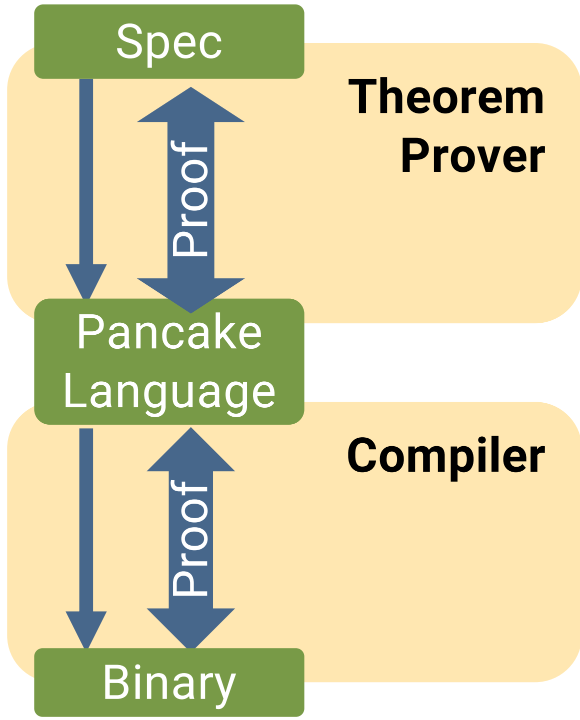
exploring

Main take-away:
Approach will work for OS components too!

Pancake

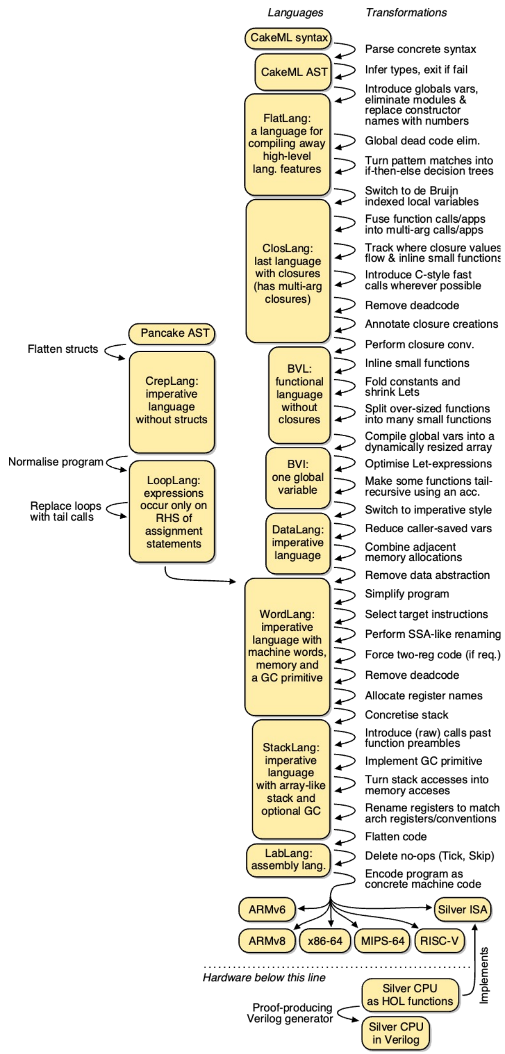
Pancake: Language for Verified Systems

Idea:
- Systems language
- C-like but safer
- No fancy type system!
- Verified compiler
- Clean semantics
- Ease verification of logic



Verified Pancake Compiler

Pancake compiler is written in CakeML
⇒ can use CakeML compiler to produce verified Pancake compiler binary!



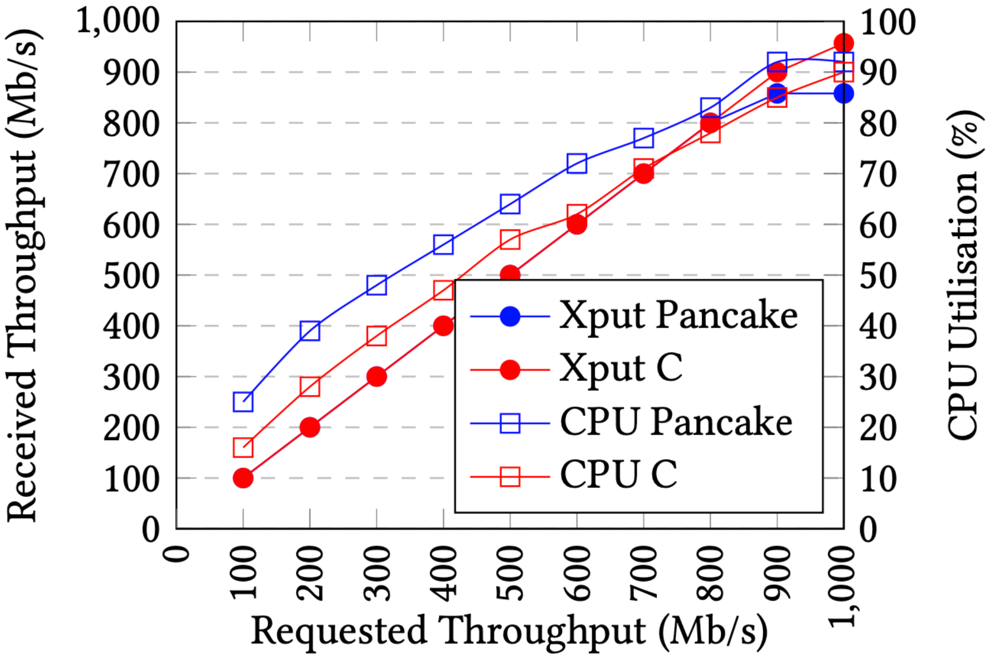
Using Pancake: Ethernet sDDF Mux

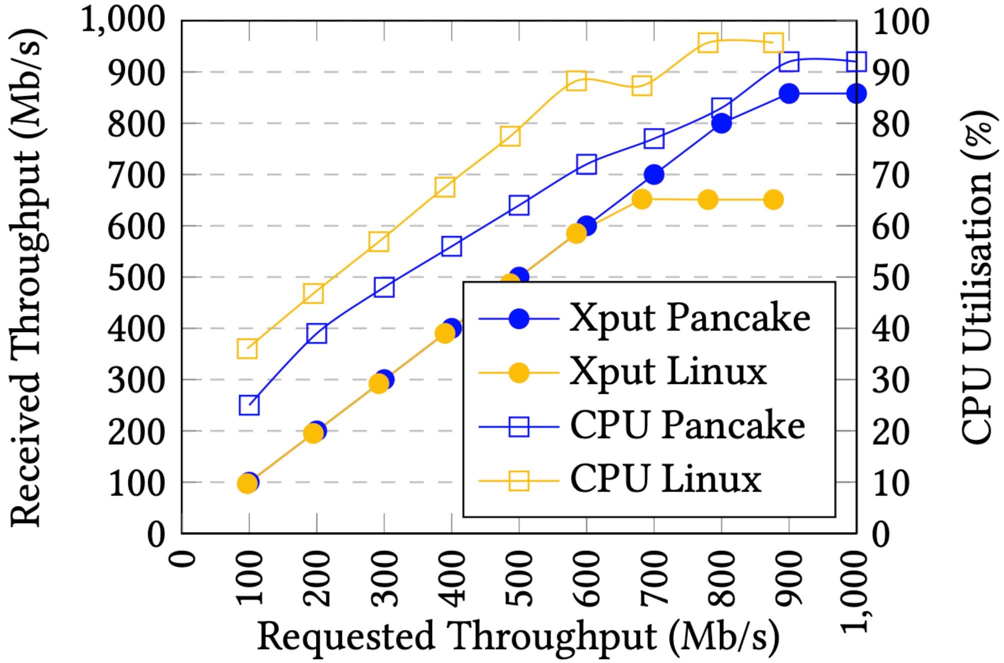
Component	Pancake Version			C Version
	Pancake	C	Total	C
Tx Mux	81	206	287	85
Rx Mux	179	314	493	222

Present Pancake limitations (WiP):

- Need C stubs for interfacing with
  - device registers
  - shared-memory data structures
- Pancake programs are standalone, not functions
  - invocation executes redundant init code

Pancake Performance





sDDF setup with Pancake Muxes

Performance degradation for well-understood reasons

Other Developments

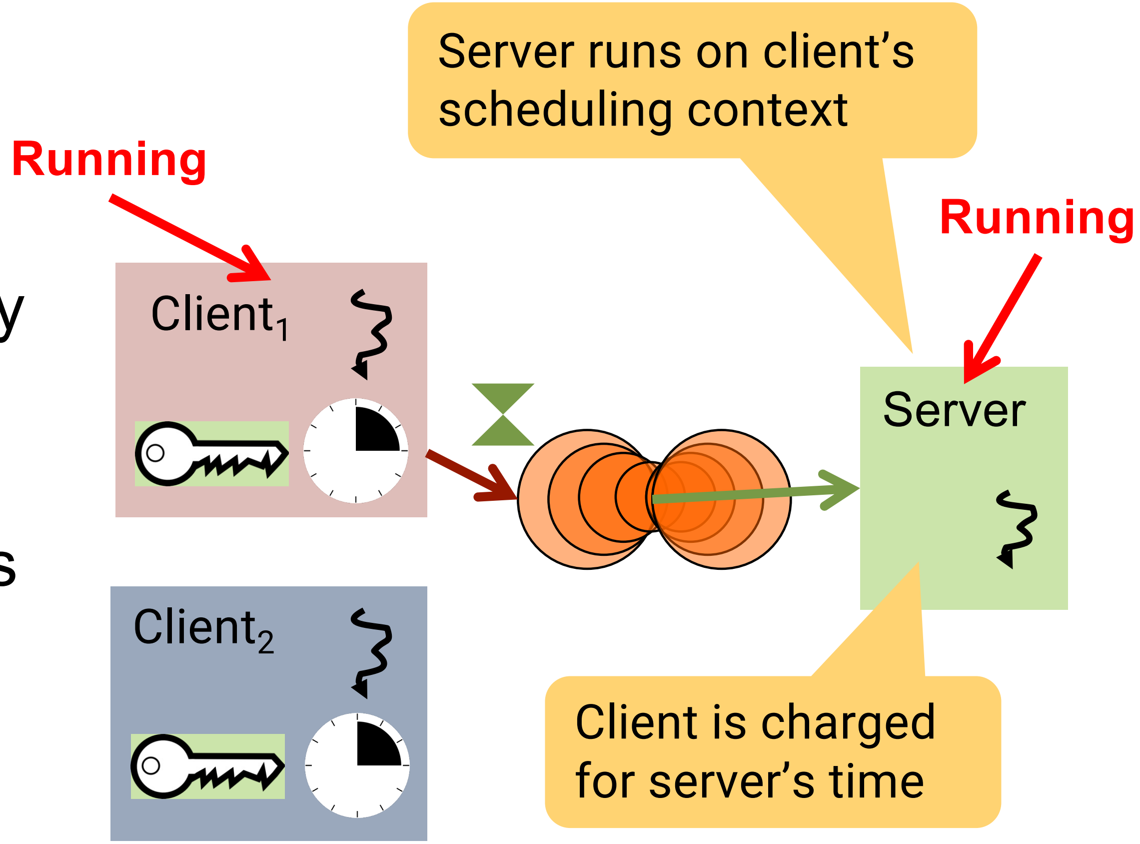
RFC 14: Budget Limit Thresholds

Problem:

- Budget may expire while server executes
- Requires complex & costly recovery

Solution:

- Only allow call if client has sufficient budget
- Timeout becomes a true error condition (misconfigured server)

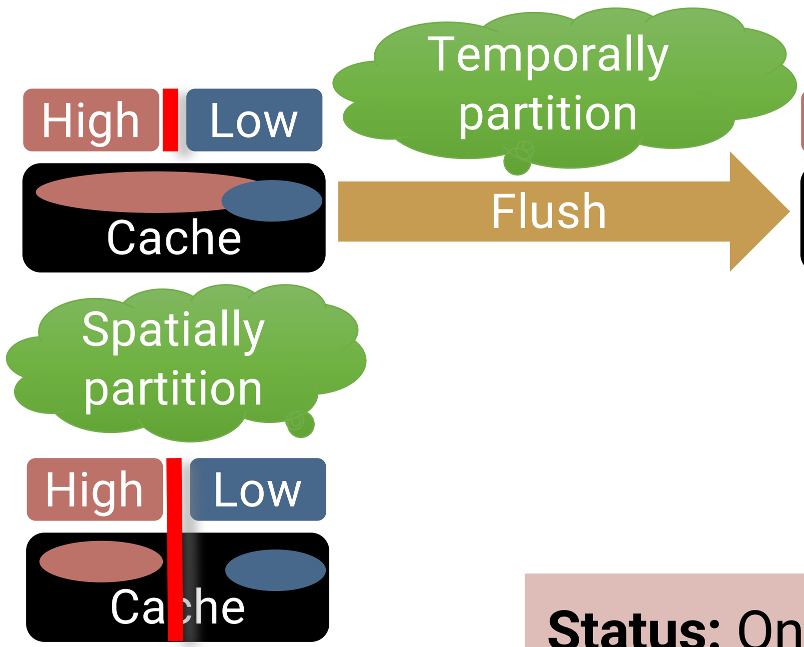


RFC 12: seL4 Device Driver Framework

As per Lucy's update:

- Networking design very advanced
- Other devices at an early state
- Will not advance this until more devices covered
- Should happen within a year

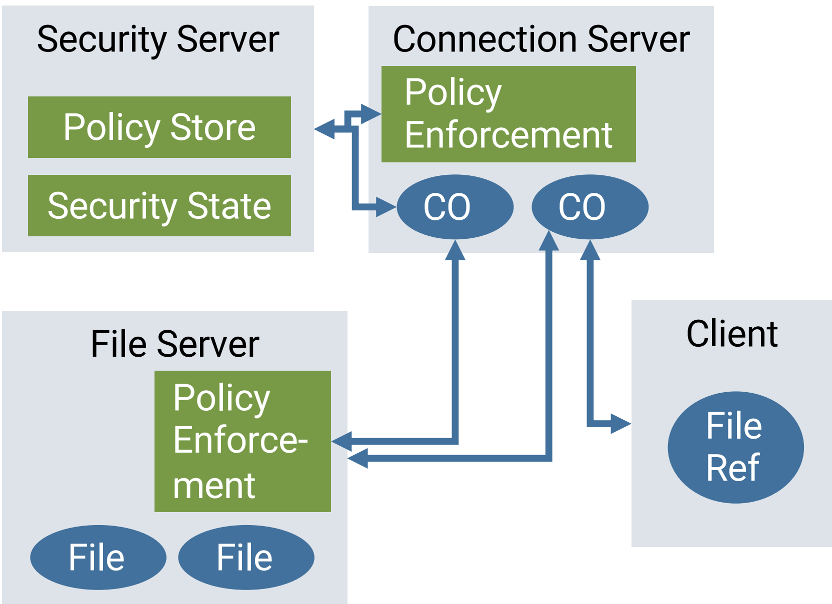
Verified Time Protection



Aim: Provably prevent information flow through micro-architectural timing channels

Status: On hold, waiting for more funding

Provably Secure General-Purpose OS



Aim:
- GP-OS with security policy diversity
- fully dynamic
- Proof that policy is enforced
- Performance

Status:
- developing & evaluating server protocols
- formalising security
- slow progress due to lack of funding

Summary

- **Main learning form past year: Simplicity wins**
  - KISS Principle can deliver high performance
  - Resulting code is tractable by push-button techniques
- Result: Lions OS!
- Pancake is promising but jury is still out

Security is no excuse for bad performance!

https://trustworthy.systems