NICTA

Towards Trustworthy Systems

or

The Continuing Relevance of OS Research

Gernot Heiser

NICTA and University of New South Wales

Sydney, Australia

Australian Government

Department of Broadband, Communications and the Digital Economy

Australian Research Council

NICTA Funding and Supporting Members and Partners

Australian National University

UNSW THE UNIVERSITY OF NEW SOUTH WALES

NSW GOVERNMENT

ACT GOVERNMENT

State Government Victoria

THE UNIVERSITY OF MELBOURNE

THE UNIVERSITY OF SYDNEY

Queensland Government

Griffith UNIVERSITY

QUT Queensland University of Technology

THE UNIVERSITY OF QUEENSLAND AUSTRALIA

Windows

An exception 06 has occured at 0028:C11B3ADC in VxD DiskTSD(03) + 00001660. This was called from 0028:C11B40C8 in VxD voltrack(04) + 00000000. It may be possible to continue normally.

* Press any key to attempt to continue.
* Press CTRL+ALT+RESET to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue

What's Next?

Trust Without Trustworthiness

Core Issue: Complexity

- Massive functionality ⇒ huge software stacks
  - Expensive recalls of CE devices

- Increasing usability requirements
  - Wearable or implanted medical devices
  - Patient-operated
  - GUIs next to life-critical functionality

- On-going integration of critical and entertainment functions
  - Automotive infotainment and engine control

Our Vision: Trustworthy Systems

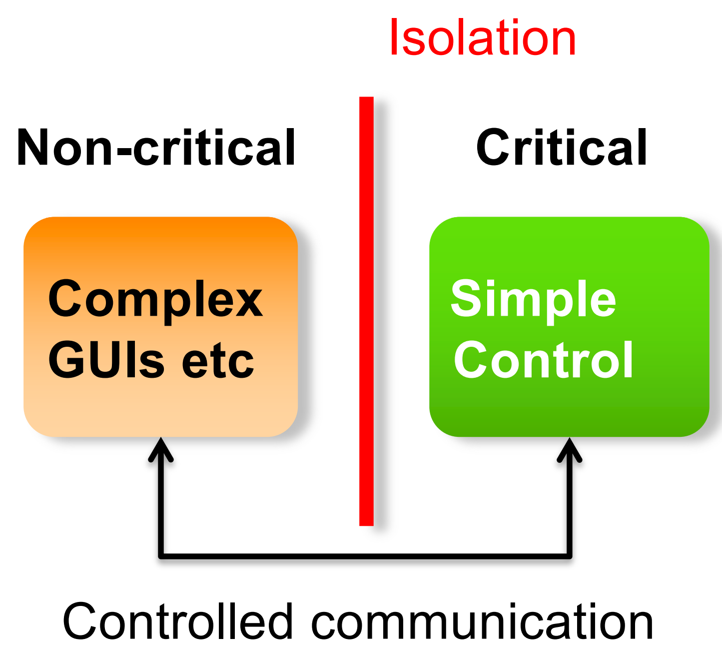
We will change industry's approach to the design and implementation of critical systems, resulting in true *trustworthiness*.

Trustworthy means *highly dependable*, with *hard guarantees* on security, safety or reliability.

Dealing With Complexity

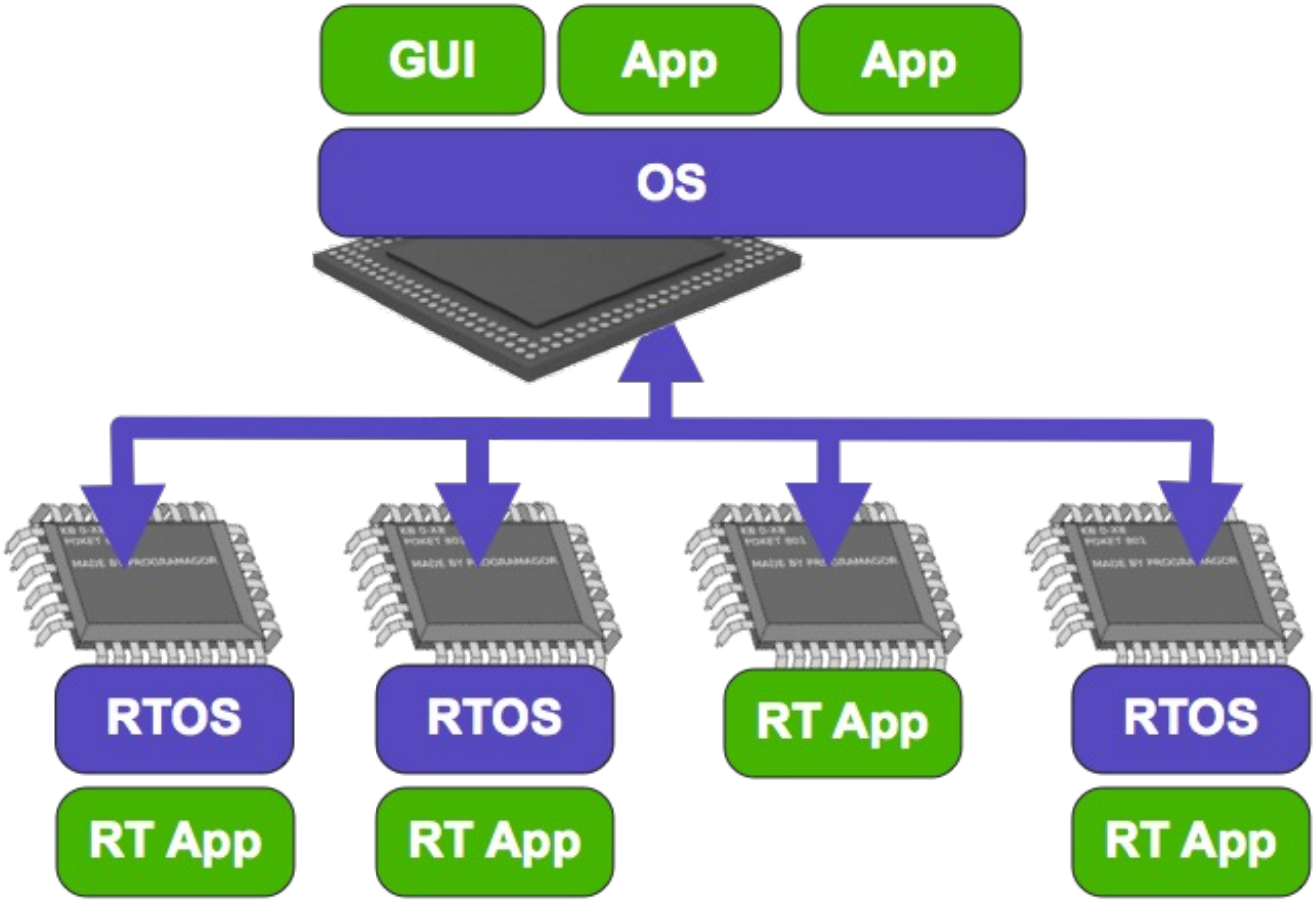
- Complexity of critical devices will continue to grow
  - Critical systems with millions of lines of code (LOC)
- We need to learn to ensure *dependability* despite complexity
  - Need to *guarantee* dependability
- Correctness guarantees for MLOCs unfeasible

- Key to solution: *isolation*
  - … with controlled communication



Isolation: Physical

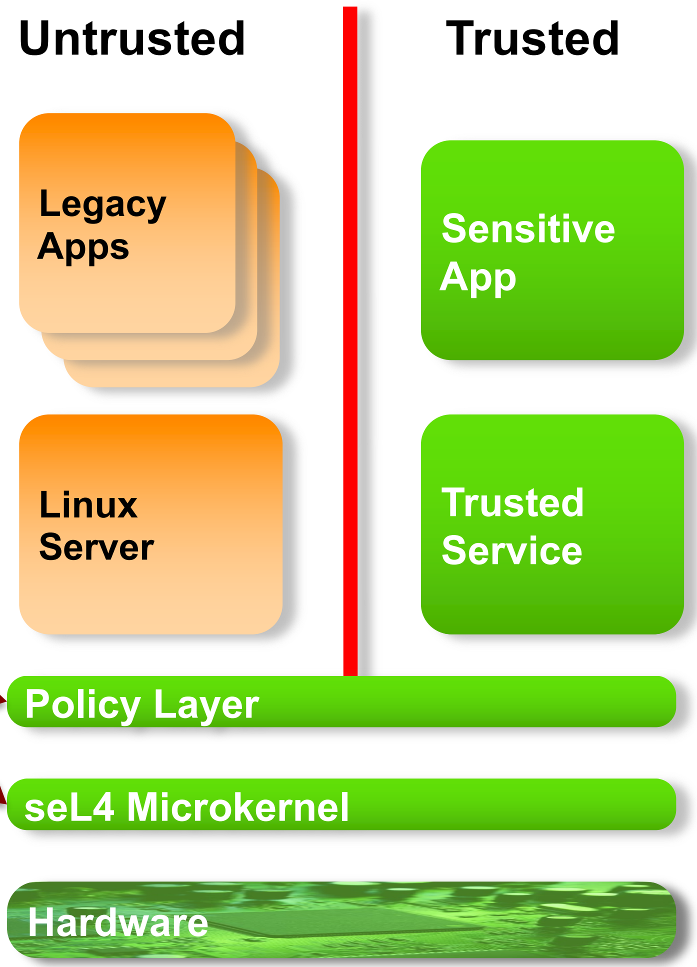
Dedicated CPUs for critical tasks



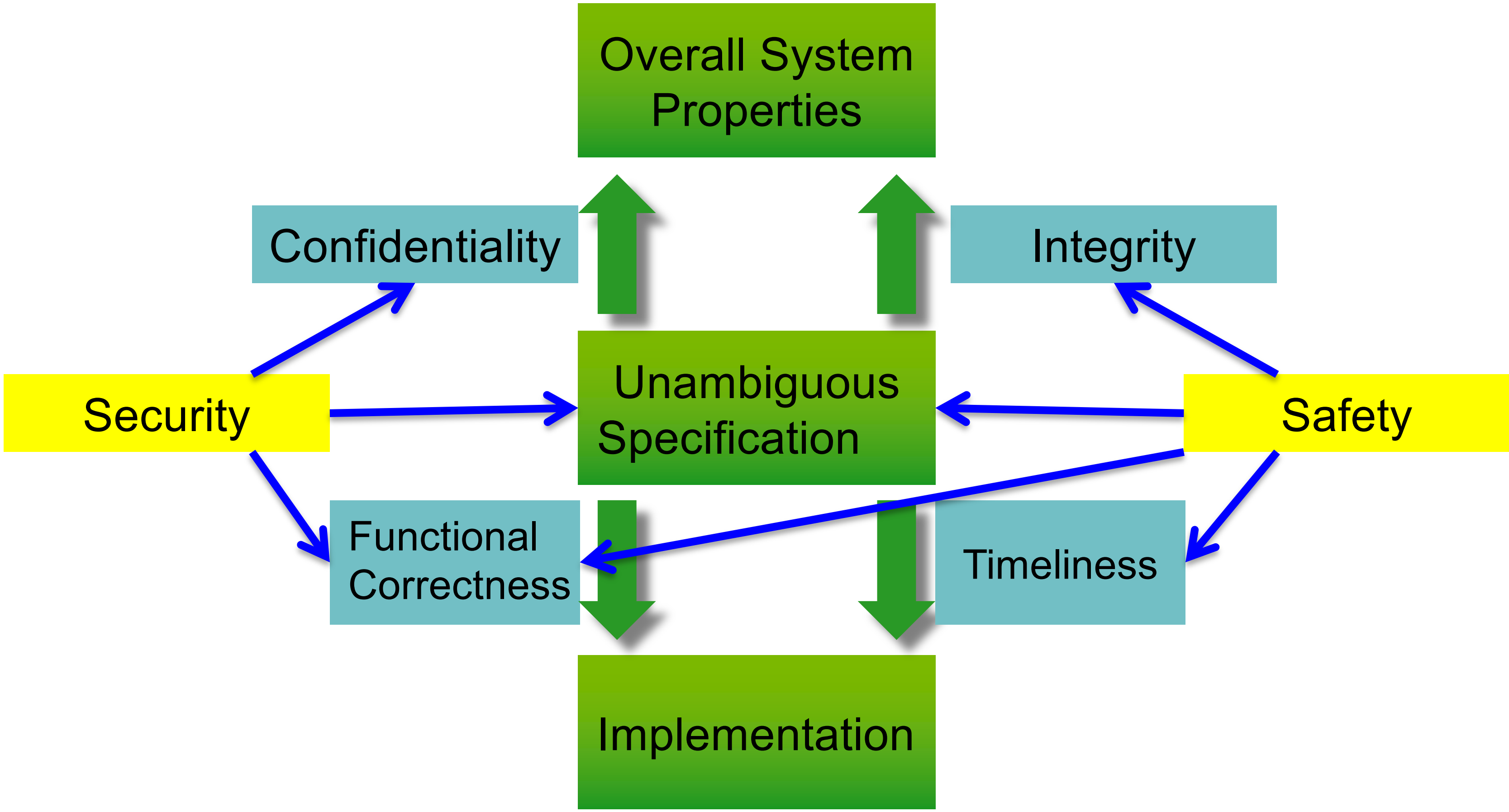
Cost: Space, costly interconnects, poor use of hardware

Isolation: Logical

- Protect critical components by sandboxing complex components
- Provide tightly-controlled communication channels
- *Trustworthy microkernel* provides general mechanisms to enforce isolation
- *Policy layer* defines access rights
- Microkernel becomes core of *trusted computing base*
  - System trustworthiness only as good as microkernel
  - **But**: small enough so that real trustworthiness may actually be achievable!



Dependability Requirements

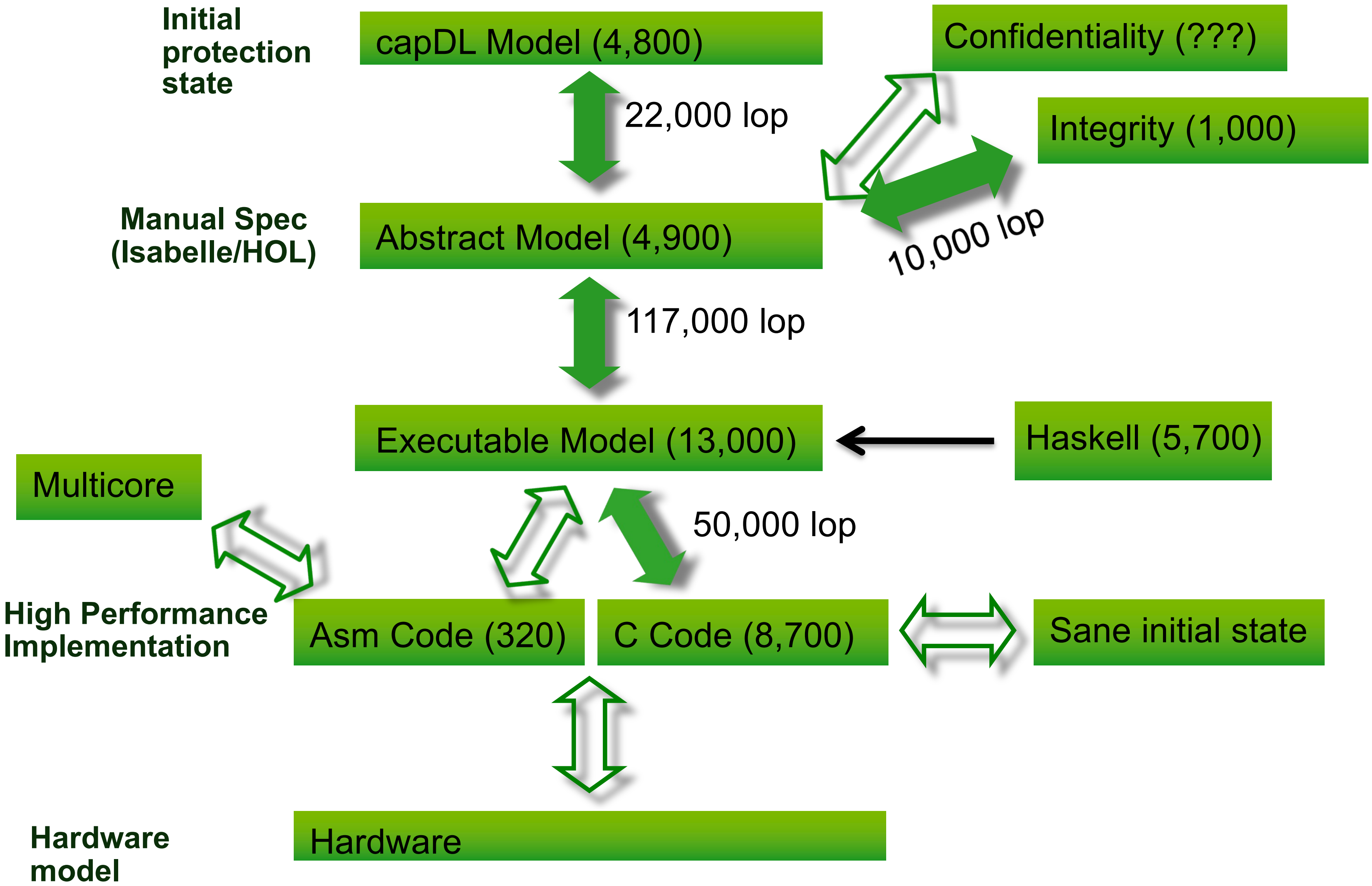


NICTA Trustworthy Systems Agenda

1. **Dependable microkernel (seL4) as a rock-solid base**
   - Formal specification of functionality
   - Proof of functional correctness of implementation
   - Proof of safety/security properties
   - Timeliness guarantees

2. **Lift microkernel guarantees to whole system**
   - Use kernel correctness and integrity to guarantee critical functionality
   - Ensure correctness of balance of trusted computing base
   - Prove dependability properties of complete system

seL4 Microkernel Formal Verification



seL4 Formal Verification Summary

Kinds of properties proved

- Behaviour of C code is fully captured by abstract model
- Behaviour of C code is fully captured by executable model
  - Can prove many interesting properties on higher-level models
- Kernel never fails, behaviour is always well-defined
  - assertions never fail
  - will never de-reference null pointer
  - cannot be subverted by misformed input
- All syscalls terminate, reclaiming memory is safe, ...
- Well typed references, aligned objects, kernel always mapped…
- Access control is decidable

Effort:

- Average 6 people over 5.5 years
- About 50–100% higher than traditional (low-assurance) projects
- Resulting kernel performs at par with best L4 microkernels

Kernel Worst-Case Execution Time

Issues for WCET analysis of seL4

- Need knowledge of worst-case interrupt-latency
  - Longest non-preemptible path + IRQ delivery cost
  - seL4 runs with interrupts disabled
    - System calls in well-designed microkernel are short!
    - Strategic preemption points in long-running operations
    - Optimal average-case performance with reasonable worst-case
- Applications also need to know cost of system calls
  - Need WCET analysis of *all* possible code paths

Kernel Worst-Case Execution Time

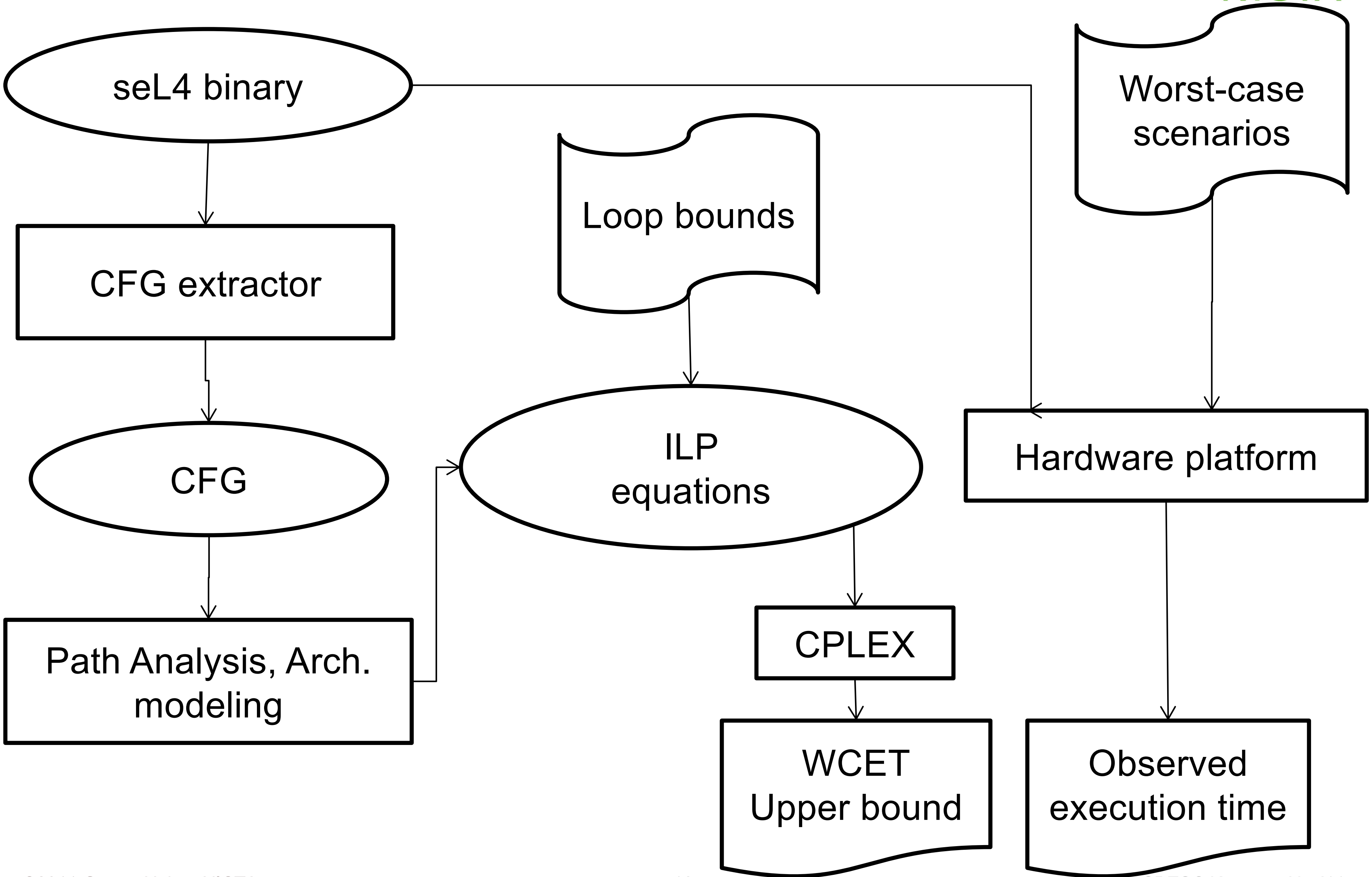
Challenges for WCET analysis of OS kernels in general:

- Kernel code notoriously unstructured
- Low-level system-specific instructions
- Context-switching
- Assembly code

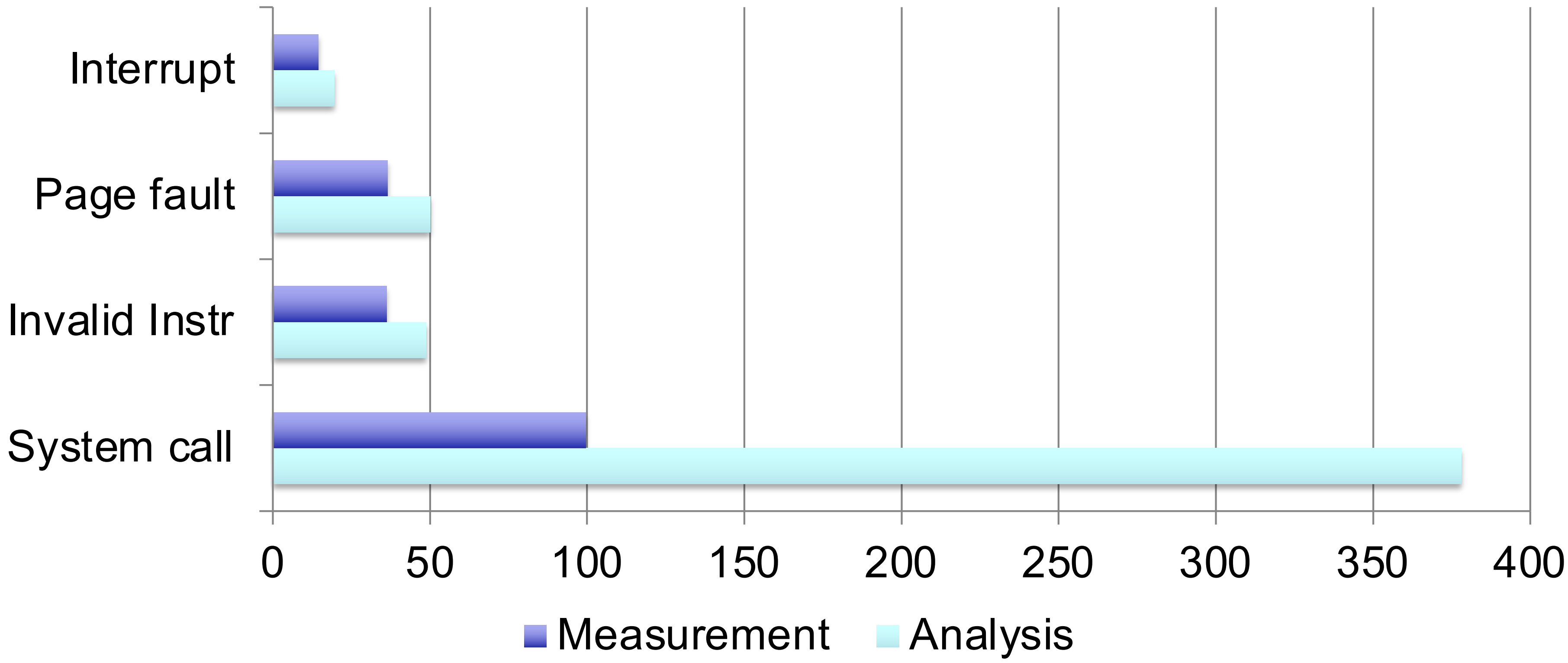
seL4-specific advantages:

- (Relatively) structured design (evolved from Haskell prototype)
- Event-based kernel (single kernel stack)
- Small (as far as operating systems go!)
- No function pointers in C
- Preemption points are explicit and preserve code structure
- Memory allocation performed in userspace

WCET analysis process



WCET Results

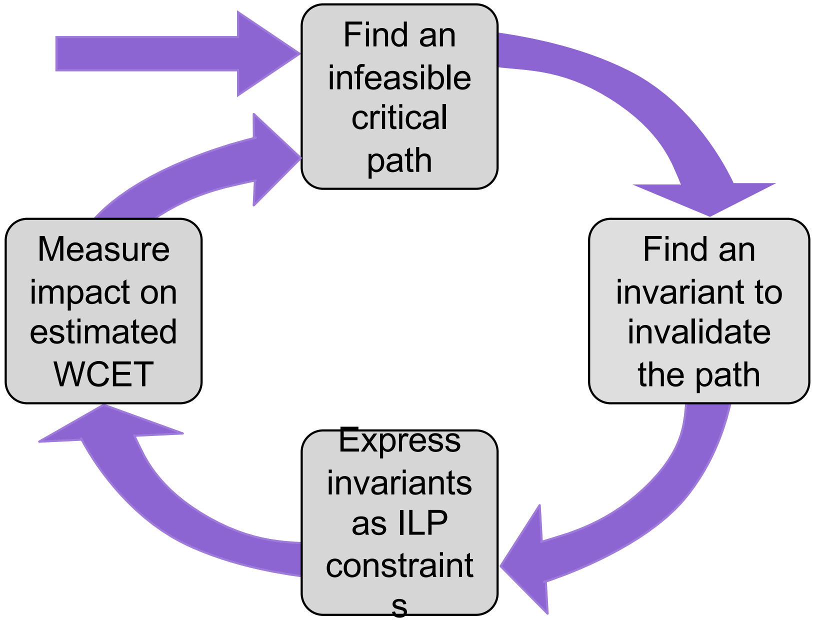


Execution times in μs on Freescale i.MX31 (ARM1136 @ 532 MHz)

- L2 cache and branch cache disabled
  - present limitation of analysis tools…

Improve WCET

- Knowledge about seL4 can eliminate many paths
  - Invariants proved during verification
  - E.g. loop iteration counts, non-interference
  - Can easily prove new invariants
  - Presently done manually (no proof)

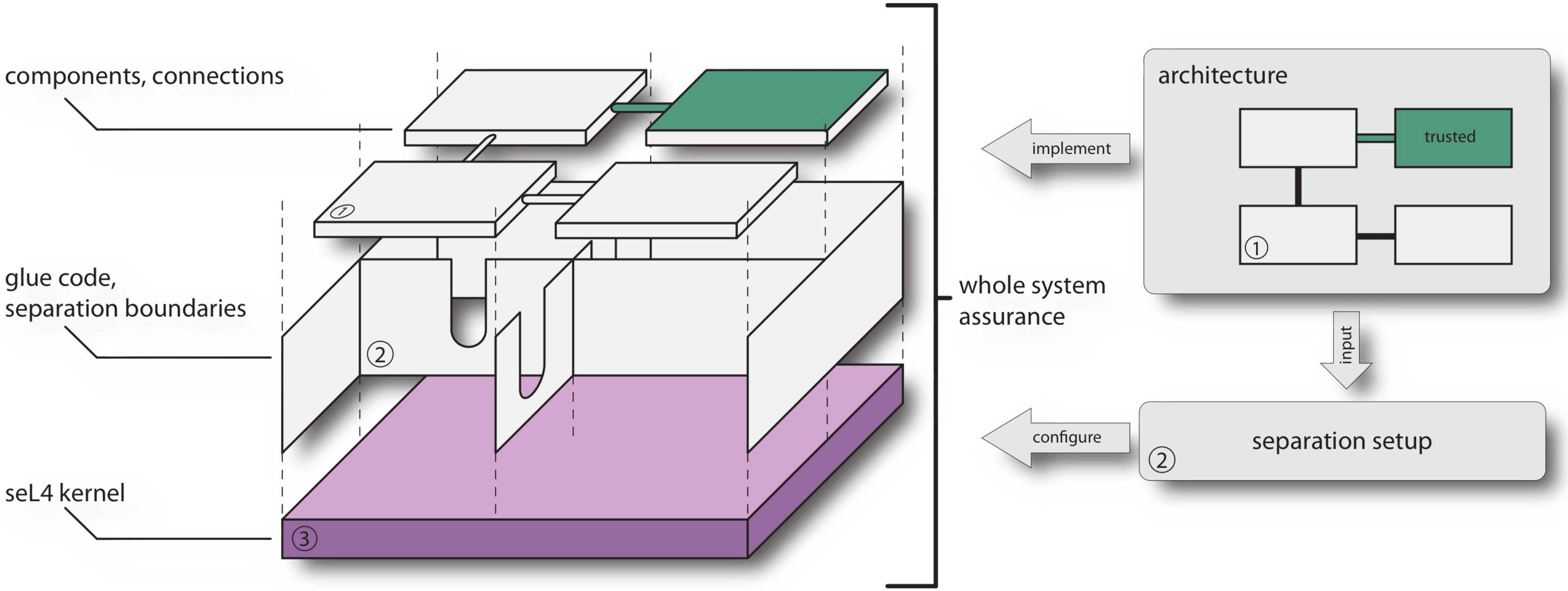


Phase Two: Full-System Guarantees

- Achieved: Verification of microkernel (8,700 LOC)

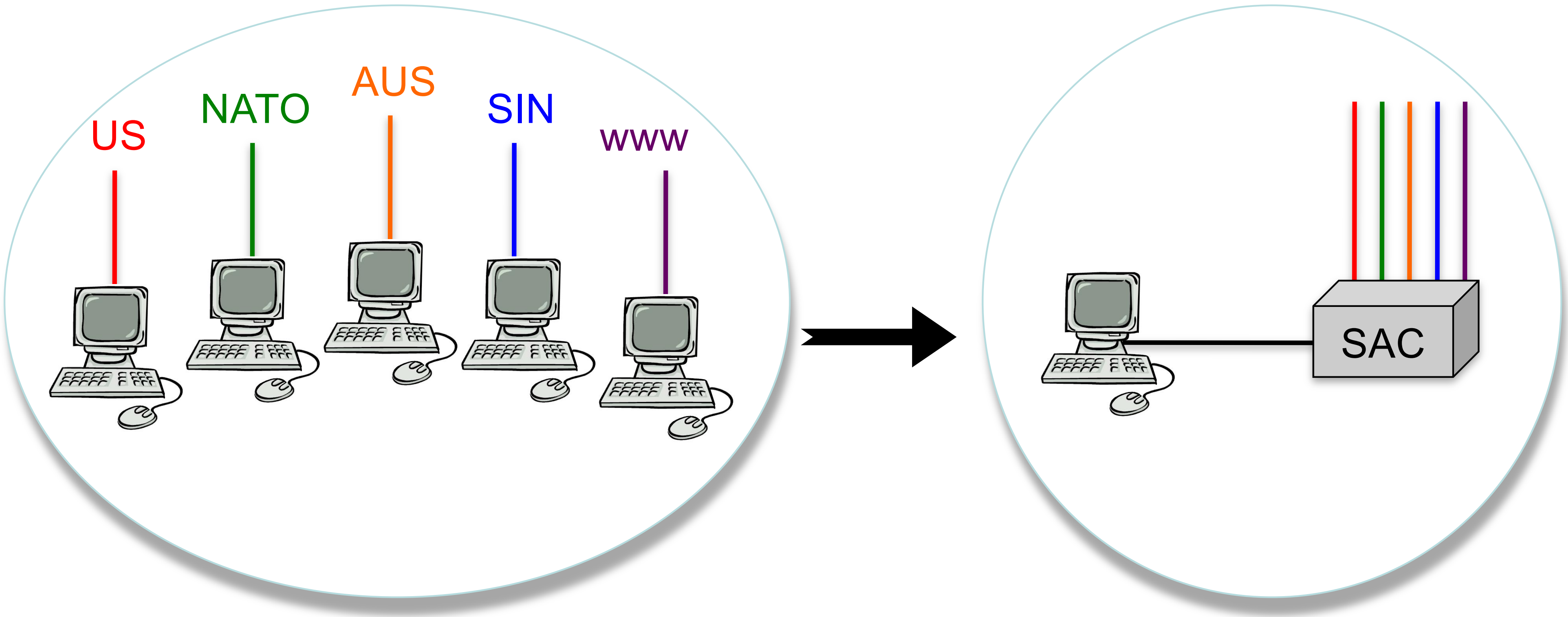
- Next step: Guarantees for real-world systems (1,000,000 LOC)

Overview of Approach

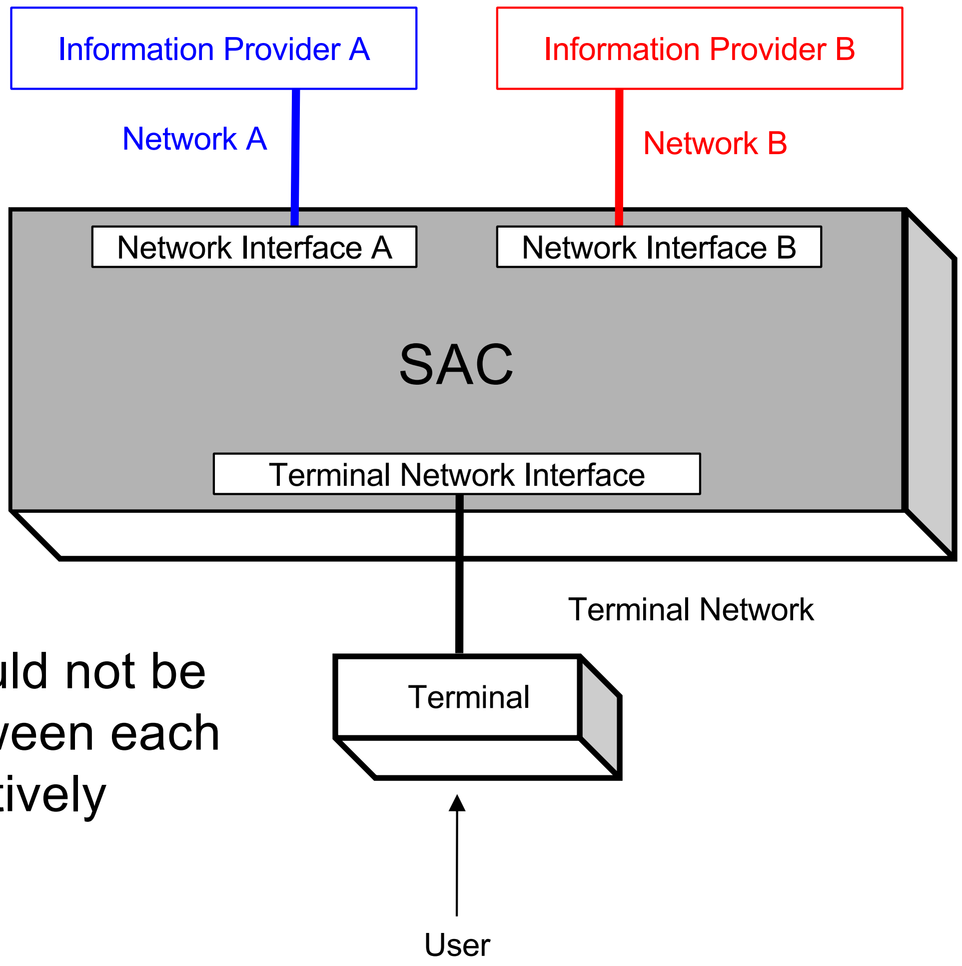


- Build system with minimal TCB
- Formalize and prove security properties about architecture
- Prove correctness of trusted components
- Prove correctness of setup
- Prove temporal properties (isolation, WCET, …)
- Maintain performance

Proof of Concept: Secure Access Controller

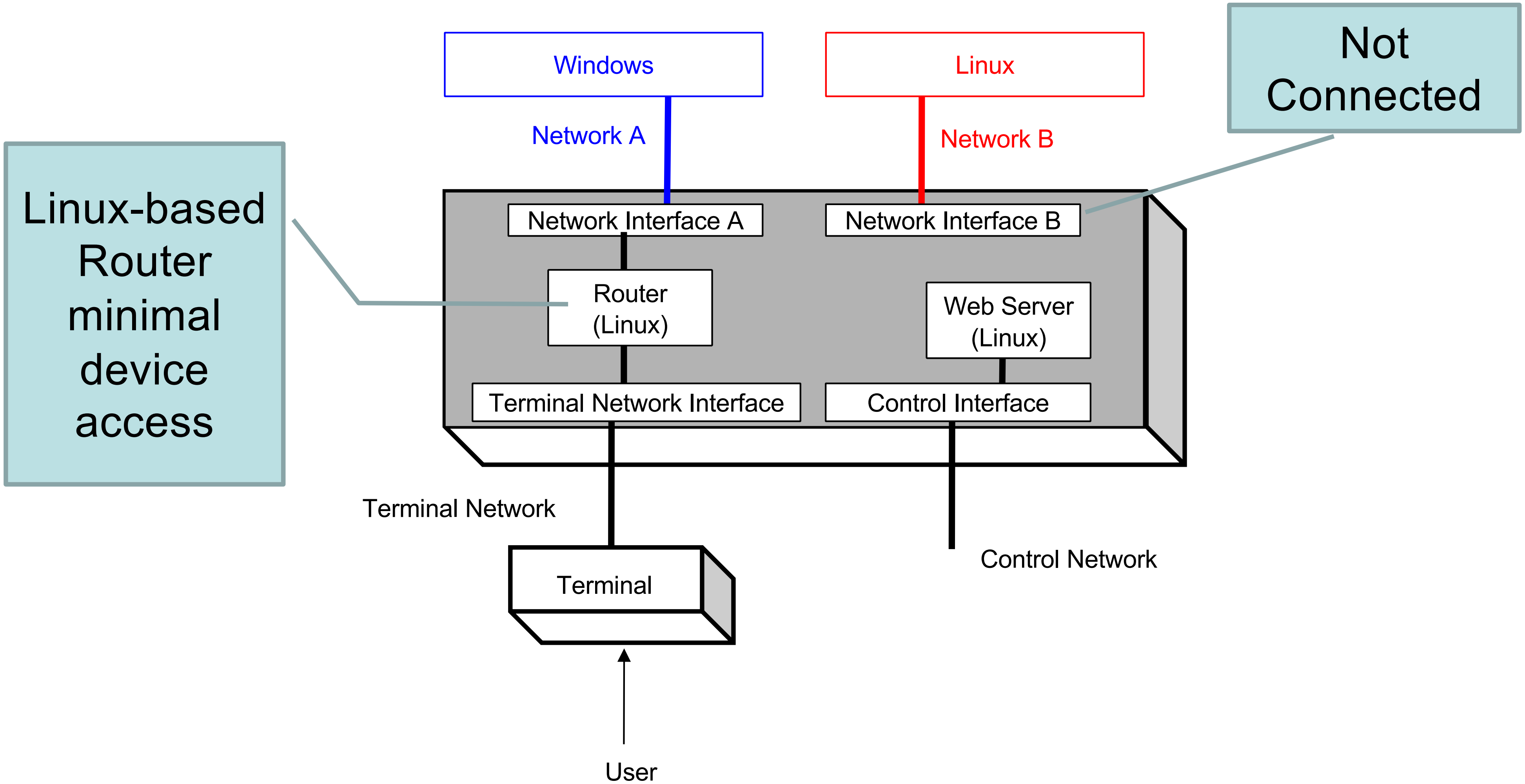


SAC Aim

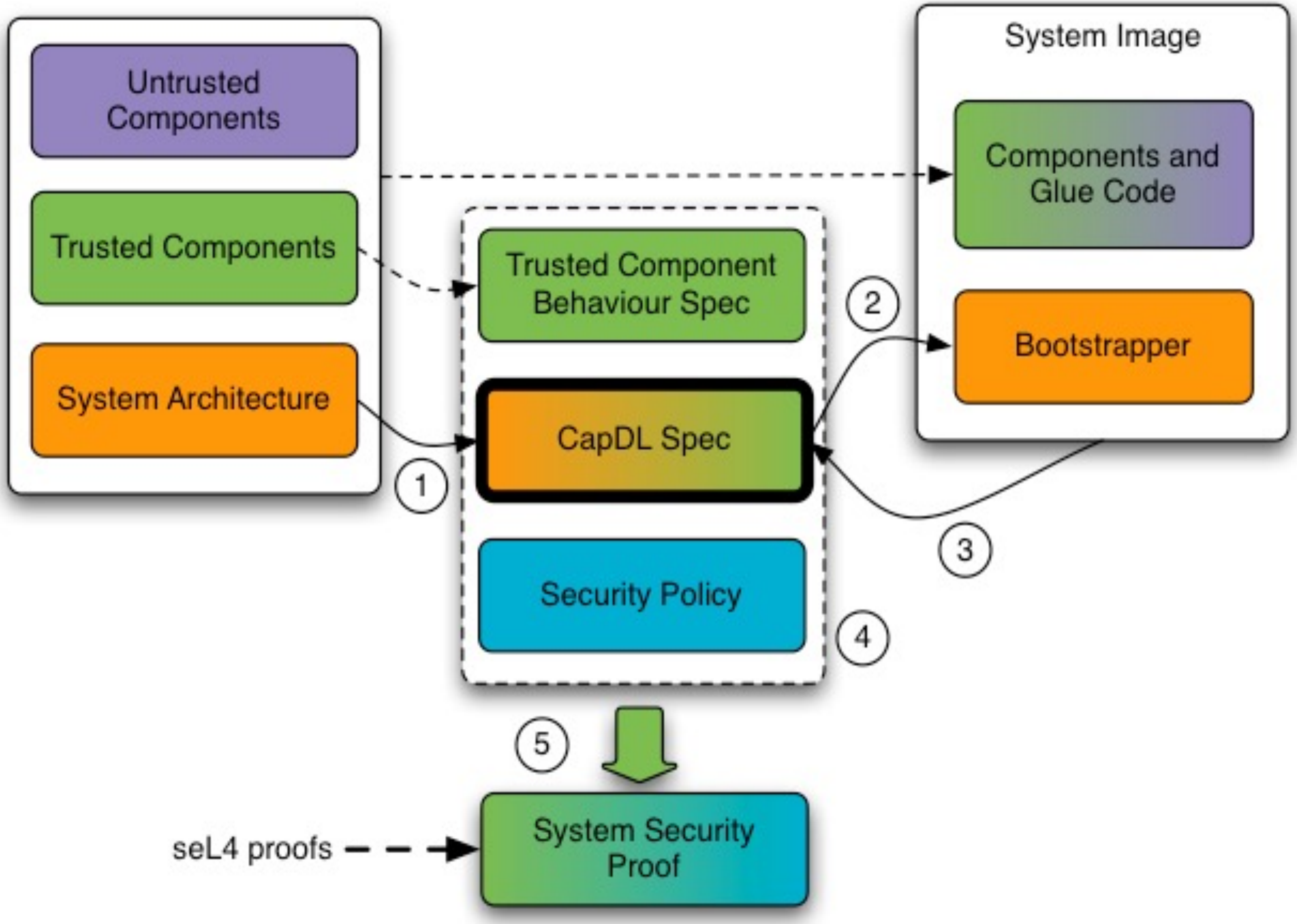


Providers A & B should not be able to leak info between each other even if they actively cooperate

Solution Overview



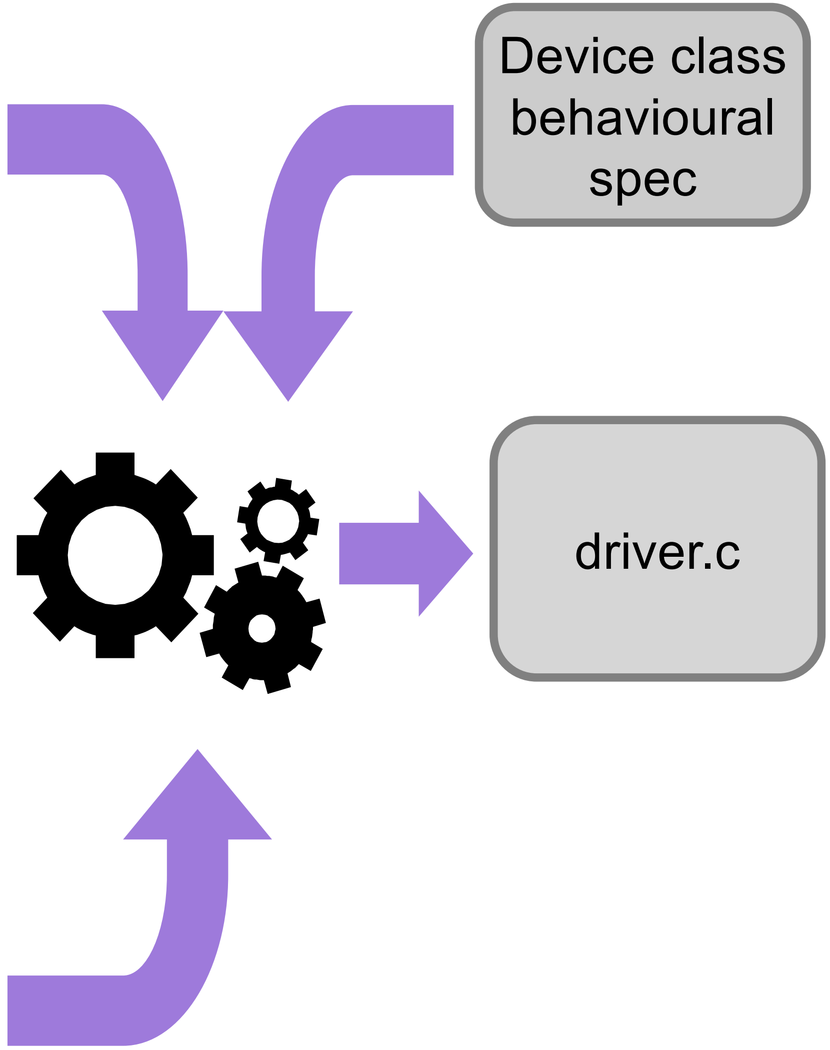
Specifying Security Architecture

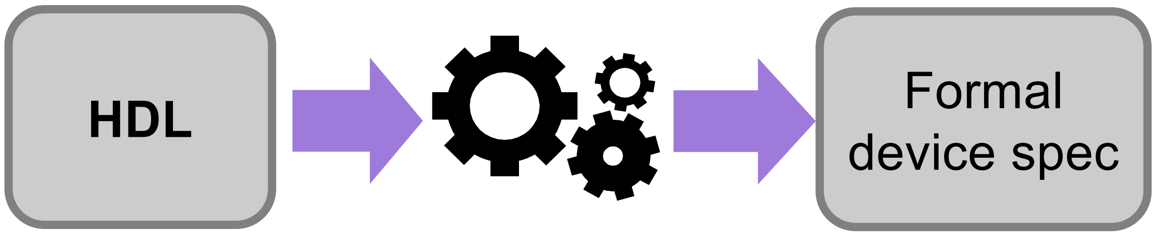


Trusted Synthesized Drivers

NICTA

- Correct driver synthesis
  - given model of driver interface, basic behaviour, and hardware
  - driver is automatically generated
  - performance as good as hand-knitted
- Challenge: device spec
- Vision:
  - automatically extract hardware model from HDL description
  - potential impact beyond our immediate agenda





Trustworthy Systems Are Possible!

- Achieved to date:
  - First general-purpose OS kernel with
    - proof of functional correctness
    - proof of integrity enforcement
    - complete and sound timing model
  - … and high performance!
  - Secure system prototype
  - Demonstration of driver synthesis feasibility
  - Framework for reasoning about system-wide access rights
- In progress:
  - Confidentiality proof
  - General real-time capabilities
  - Eliminating holes in verification
    - Compiler, asm code, multicore…

Trustworthy Systems Are Possible!

- But still lots to be done:
  - Whole-of-system security/safety proofs
  - Truly safe languages for higher-level code
    - Haskell, RT Java with verified runtime system?
  - General component synthesis…

Obrigado!

mailto:gernot@nicta.com.au
@GernotHeiser
Google: “ertos”