The Road To Trustworthy Systems

Gernot Heiser
NICTA and University of New South Wales, Sydney

Australian Government
Department of Broadband, Communications and the Digital Economy
Australian Research Council

NICTA Funding and Supporting Members and Partners

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

Trust Without Trustworthiness

What’s Next?

Our Vision: Trustworthy Systems

NICTA

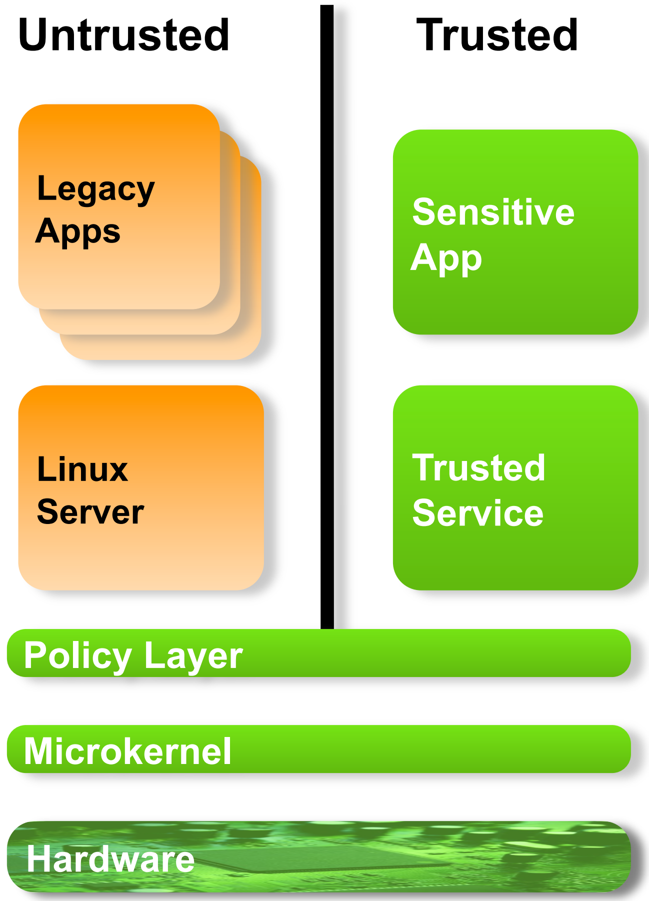
- We will change industry's approach to the design and implementation
- of embedded software, resulting in systems which are *trustworthy*.

Trustworthy means *truly dependable* in that there are *hard guarantees* about the *security*, *safety* or *reliability* of the software.

Approach: Microkernel Technology

NICTA

- Protect critical components by sandboxing complex components
- Provide tightly-controlled communication channels
- *Trustworthy microkernel* enforces security/safety policies
- Microkernel becomes core of trusted computing base
  - System trustworthiness only as good as microkernel



Trustworthy Systems Agenda

NICTA

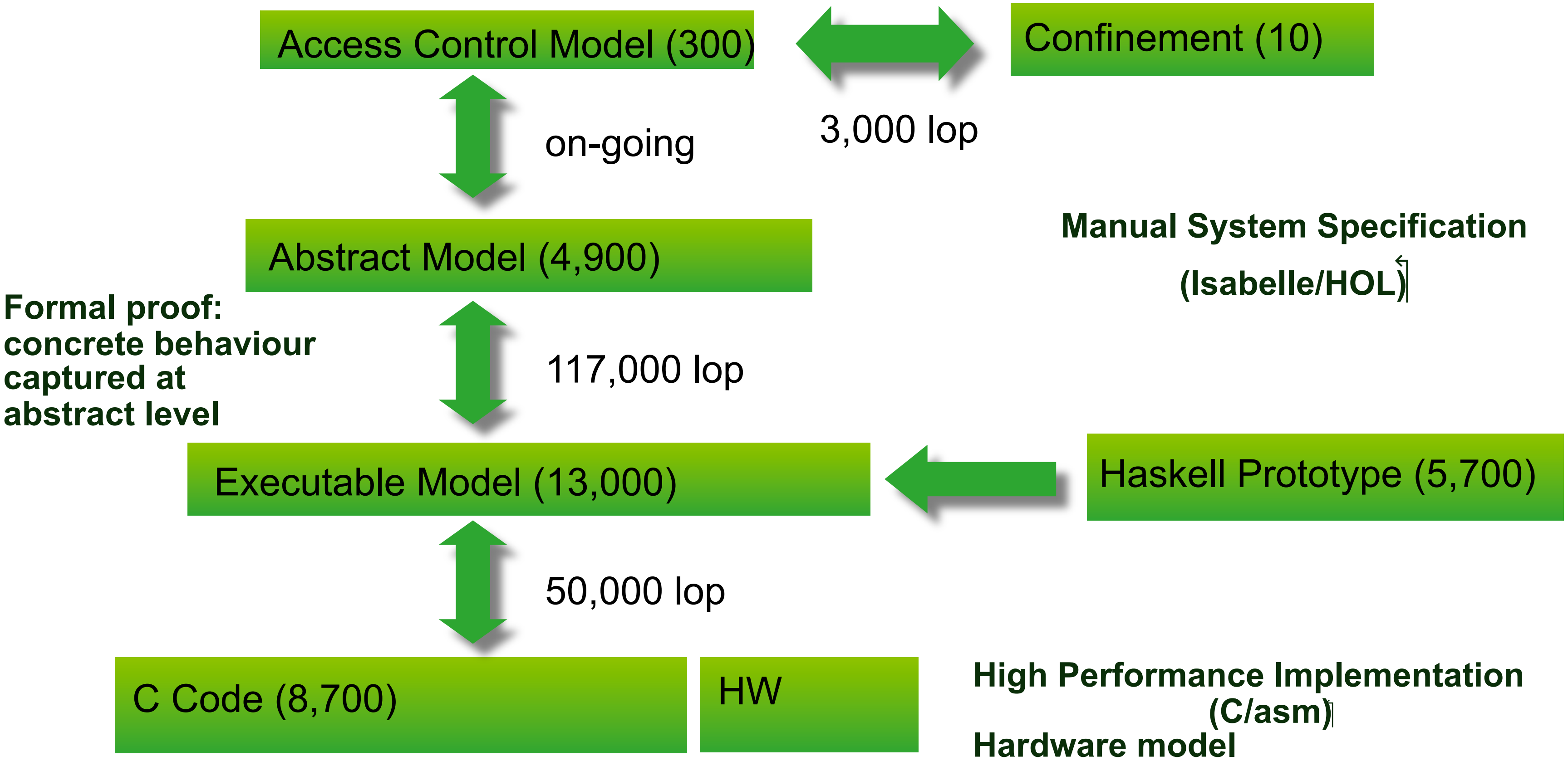
1. **Ensure microkernel trustworthiness (seL4)**
   - Proof of functional correctness
   - Proof of safety/security properties
2. **Lift microkernel guarantees to whole system**
   - Use safety/security guarantees
   - Prove correctness of balance of trusted computing base
   - Prove safety/security of complete system

Ingredients:

- Functional correctness
- Isolation / non-interference / information flow
- Timeliness / worst-case latency guarantees
- Energy management

Proof of Functional Correctness

NICTA



Correctness

NICTA

```
datatype
  rights = Read
         | Write
         | Grant
         | Create

record cap =
  entity :: entity_id
  ri

reco

type
```

```
lemma isolation:
  "⟦sane s;
    s' ∈ execute cmds s;
    isEntityOf s eS;
    sEntityOf s e;
    ntity c = e;
    :> subSysCaps s eS⟧
    ⟹ c :> subSysCaps s' eS"
```

```
constdefs
  schedule :: "unit s_monad"
  "schedule ≡ do
```

```
schedule :: Kernel ()
schedule = do
```

```
d
Thread
```

```
void
setPriority(tcb_t *tptr, prio_t prio) {
    prio_t oldprio;

    if(thread_state_get_tcbQueued(tptr->tcbState)) {
        oldprio = tptr->tcbPriority;
        ksReadyQueues[oldprio] = tcbSchedDequeue(tptr, ksReadyQueues[
        if(isRunnable(tptr)) {
            ksReadyQueues[prio] = tcbSchedEnqueue(tptr, ksReadyQueues
        }
        else {
            thread_state_ptr_set_tcbQueued(&tptr->tcbState, false);
        }
    }

    tptr->tcbPriority = prio;
}

void
yieldTo(tcb_t *target) {
    target->tcbTimeSlice += ksCurThread->tcbTimeSlice;
```

ecification

L)

e (5,700)

mance Implementation
(C/asm)
odel

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

Verification vs Certification

NICTA

Common Criteria: Military-Strength Security

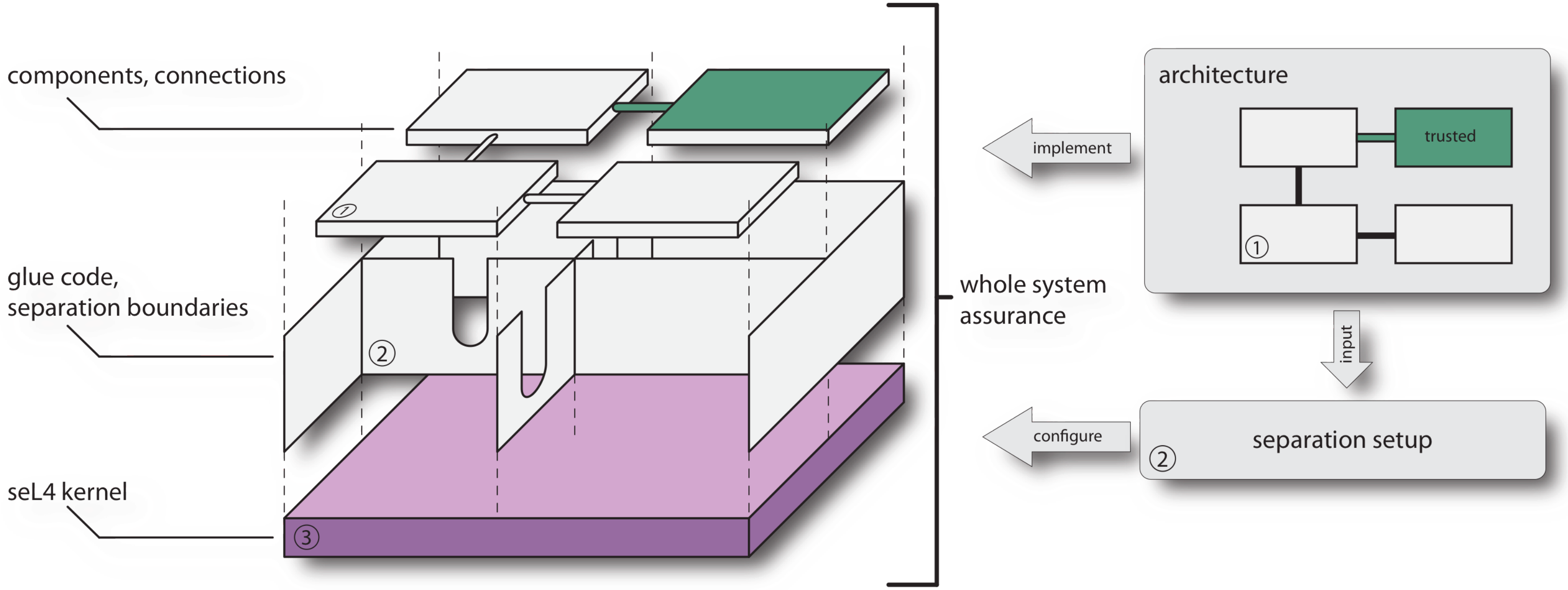
Evaluation Level	Requirements	Functional Specification	Top Down Design	Imple-mentation	Cost
EAL1		Informal			
EAL2		Informal	Informal		
EAL3		Informal	Informal		
EAL4		Informal	Informal	Informal	
EAL5		Semi-formal	Semi-formal	Informal	
EAL6	Formal	Semi-formal	Semi-formal	Informal	1K/LoC
EAL7	Formal	Formal	Formal	Informal	
seL4	Formal	Formal	Formal	Formal	0.6K/LoC

Phase Two: Full-System Guarantees

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

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

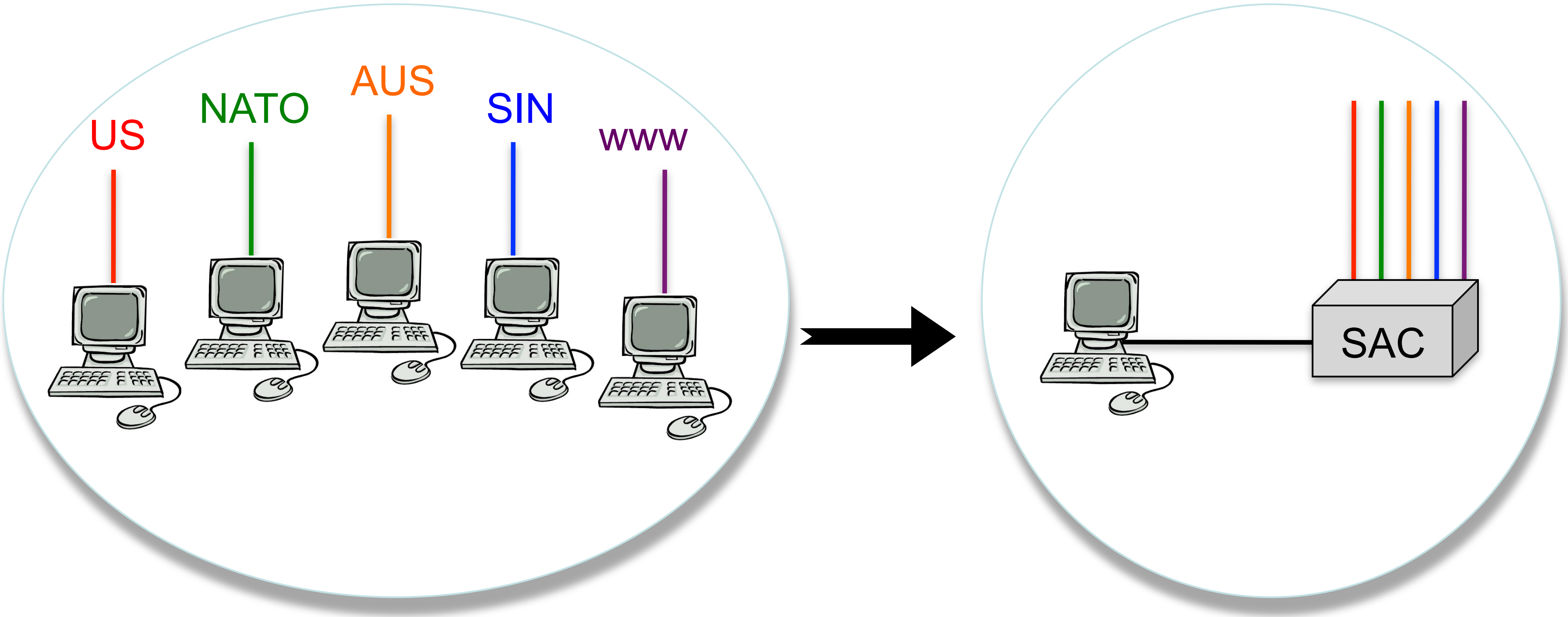
Overview of Approach



- 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

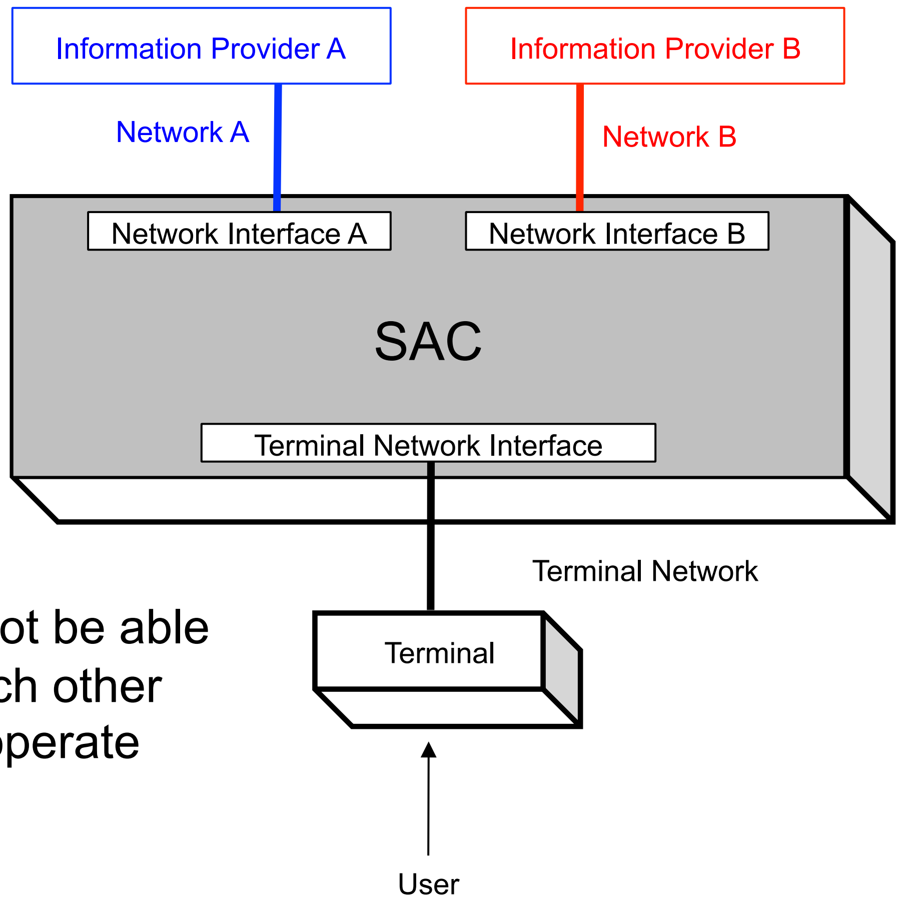
Proof of Concept: Secure Access Controller

NICTA



SAC Aim

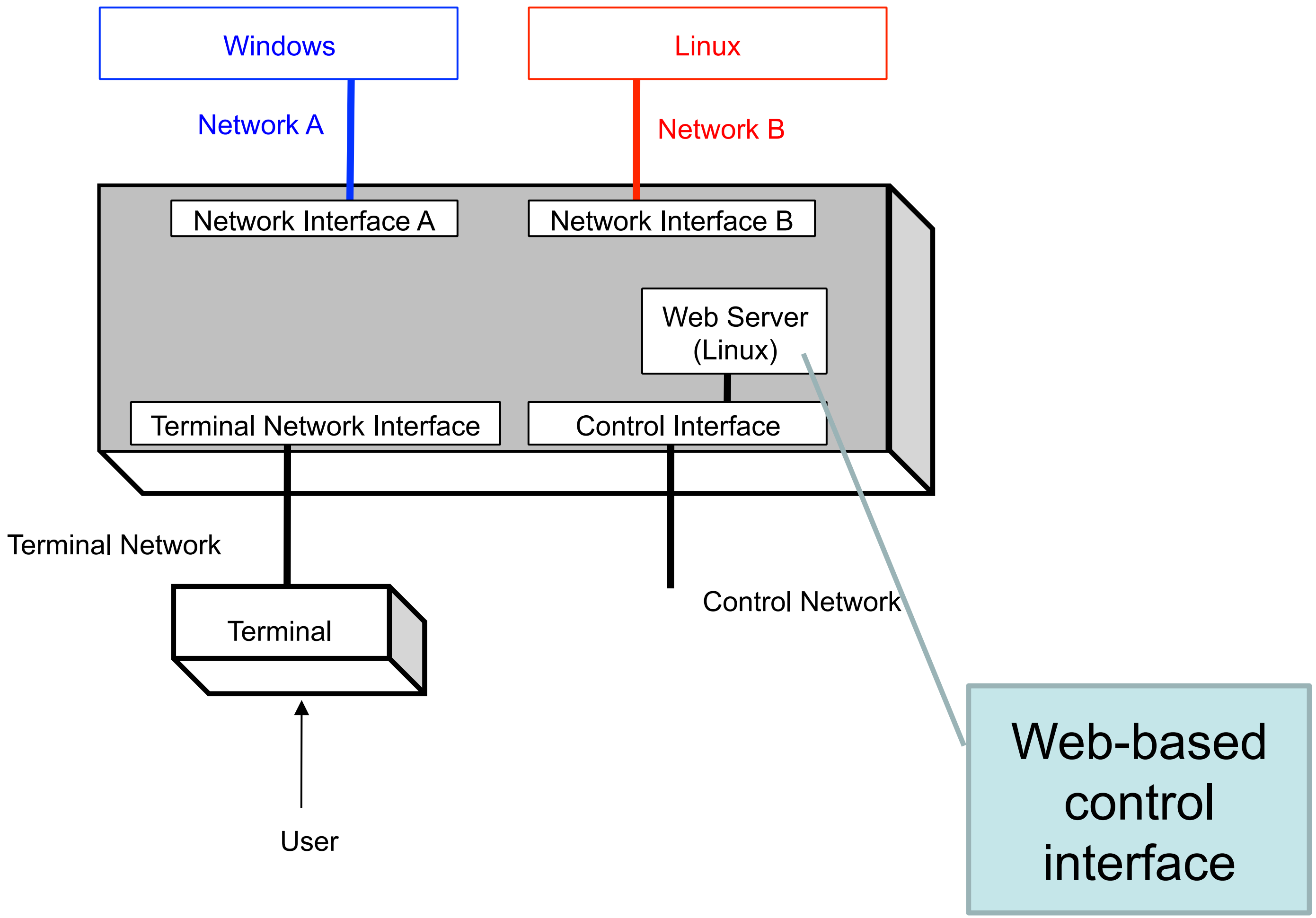
NICTA



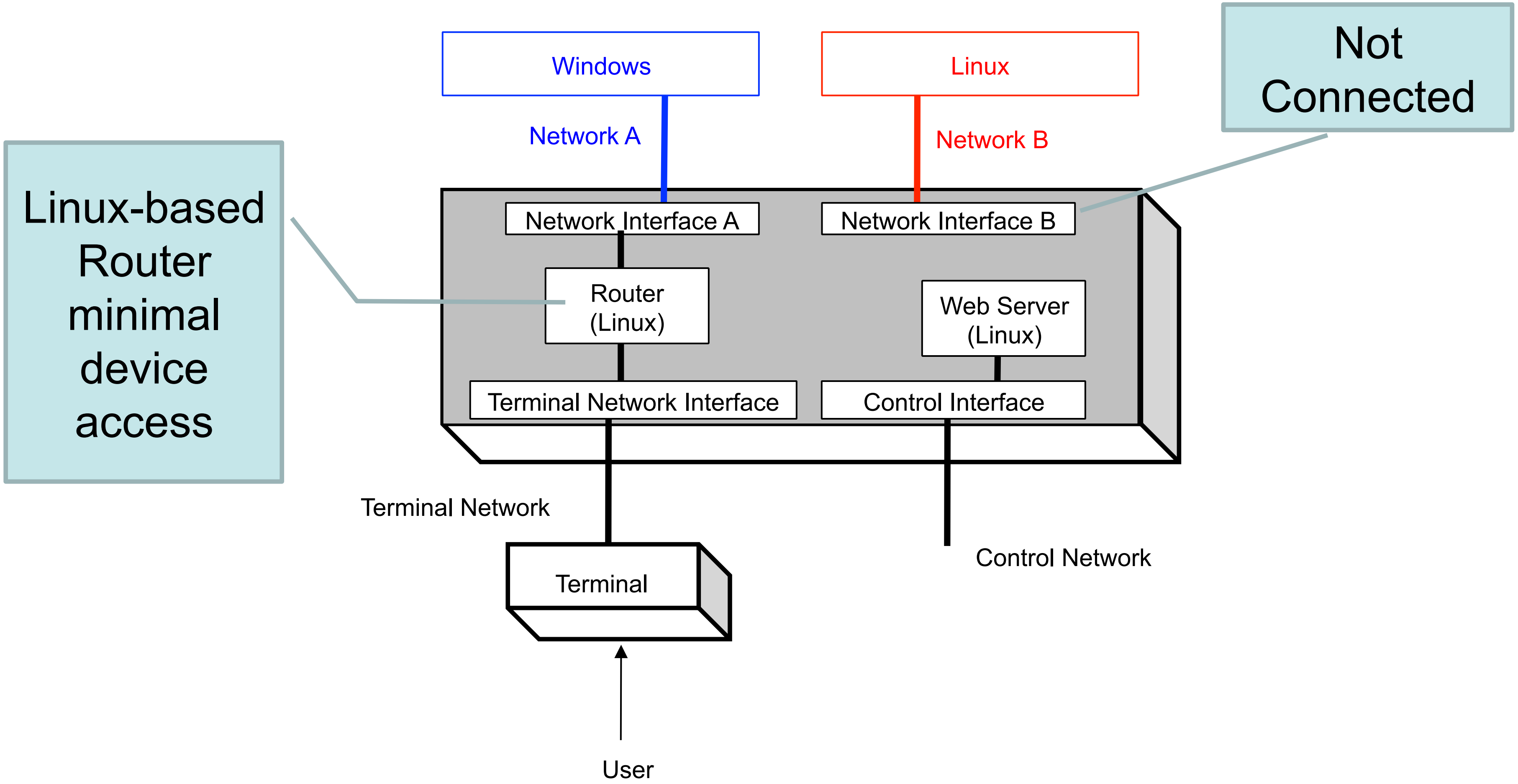
Provider A & B should not be able to leak info between each other even if they actively cooperate

Our Solution Overview

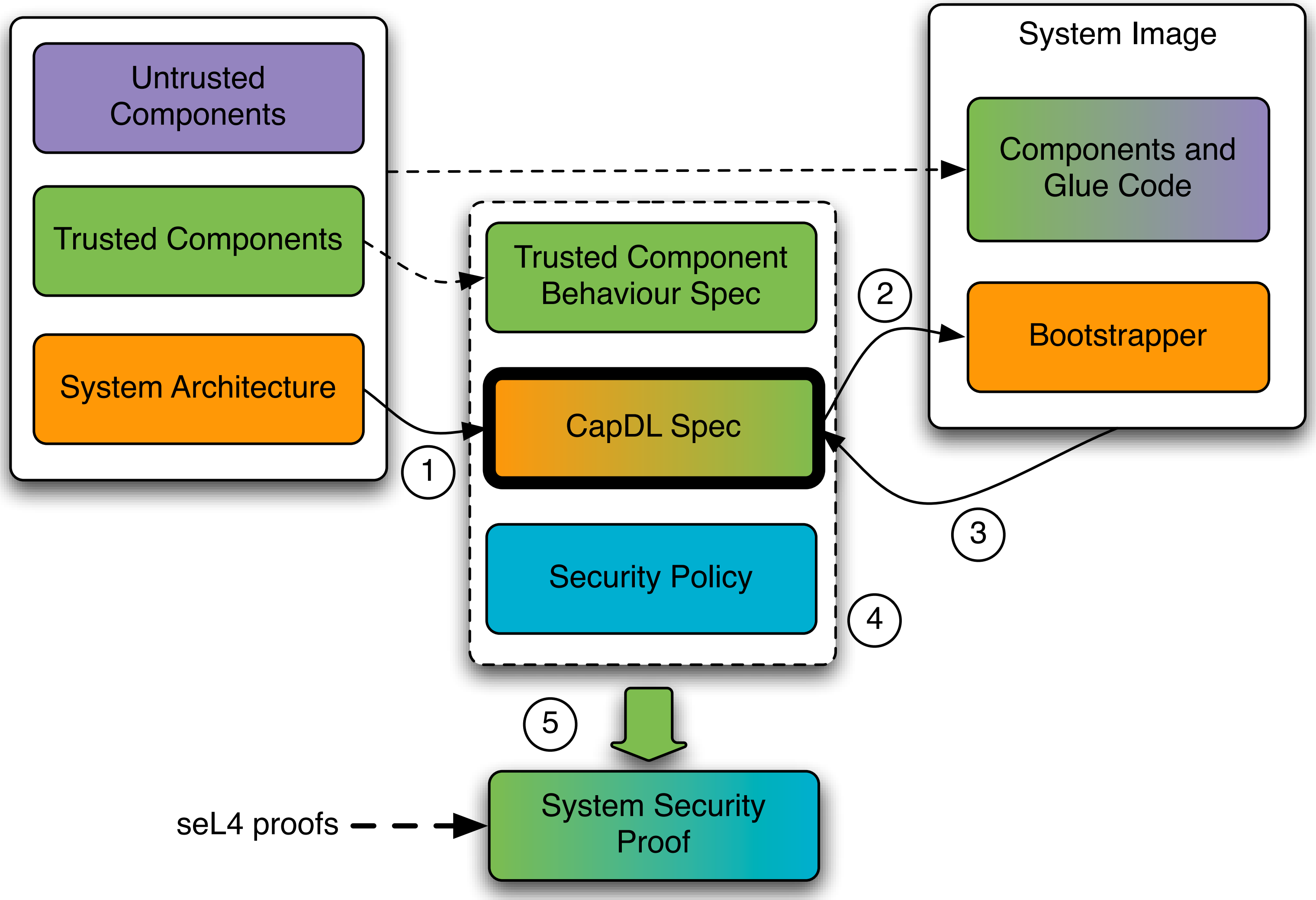
NICTA



Solution Overview

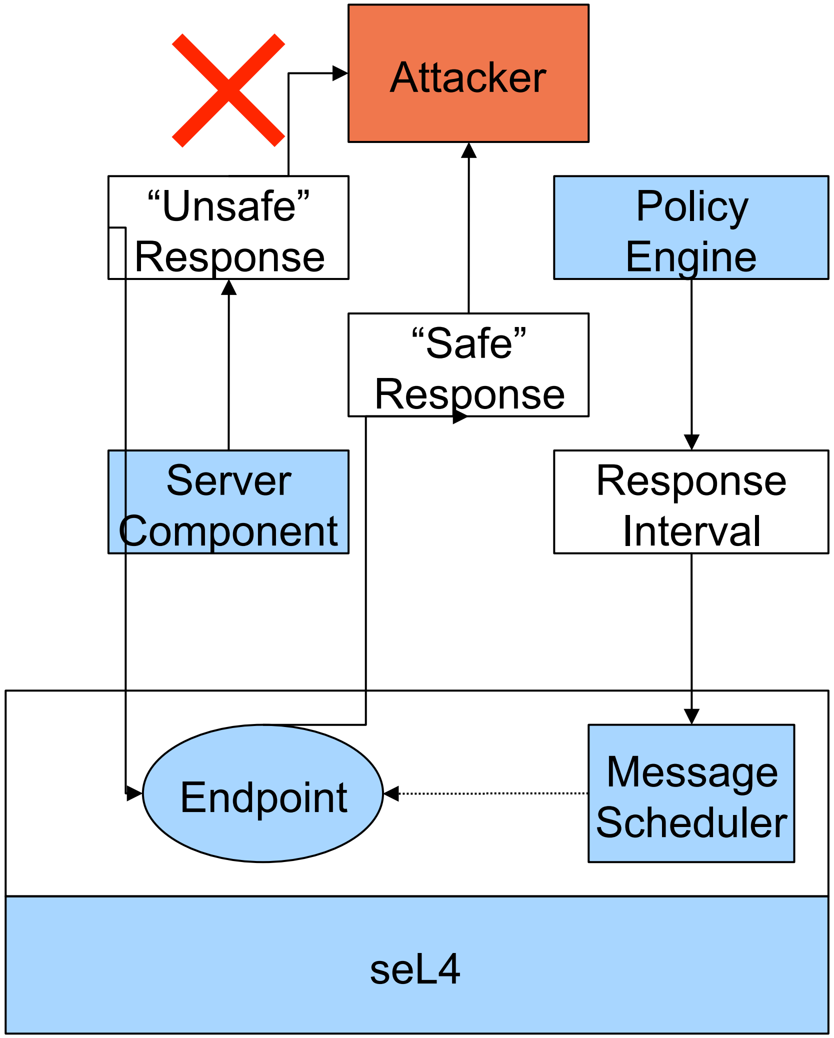


Specifying Security Architecture



Component Side-Channel Mitigation

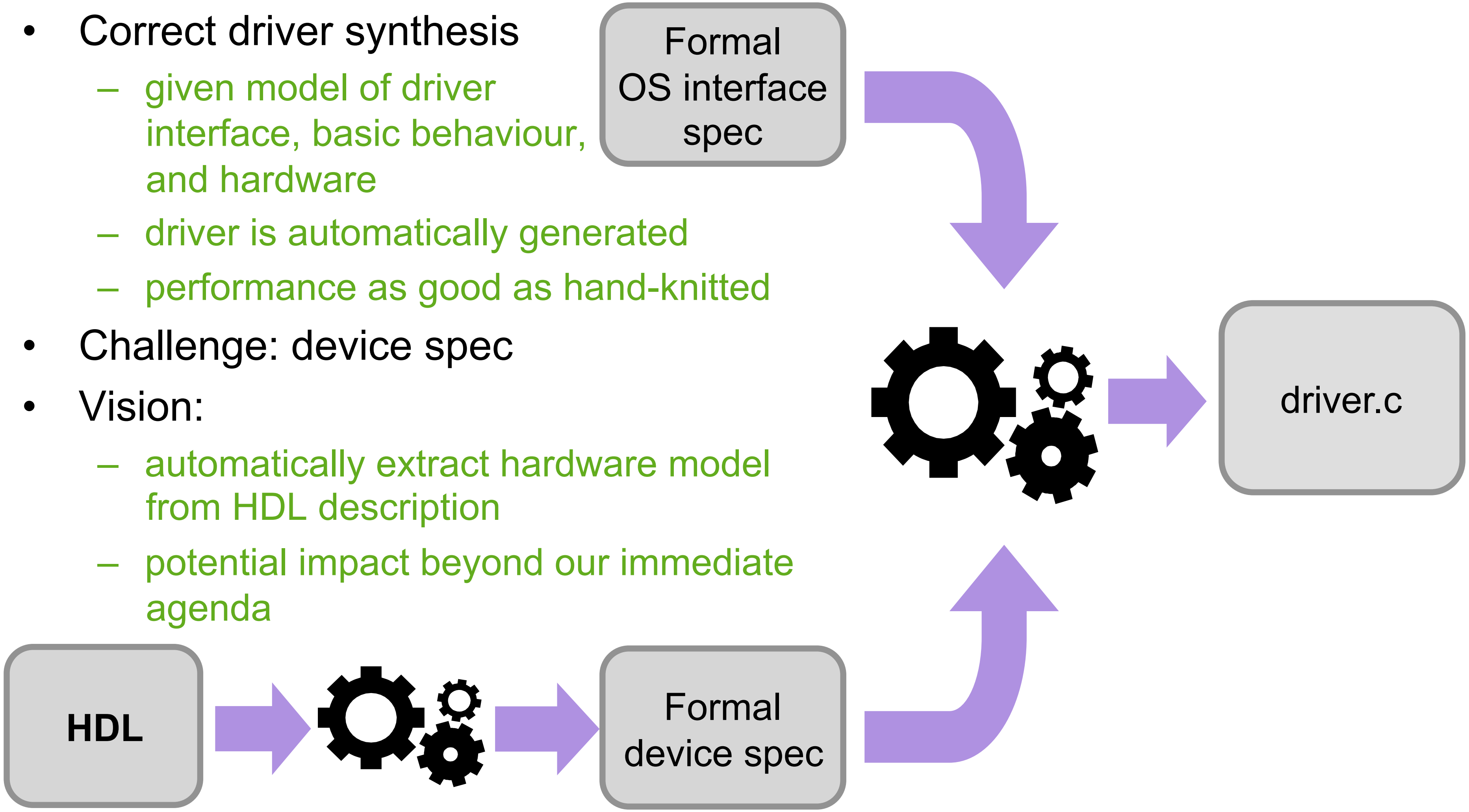
NICTA



- Component response time variability exposes secrets.
- Add a response time control policy.
- Real-time scheduling gives a mechanism.
- seL4 endpoints give a framework.

Trusted Synthesized Drivers

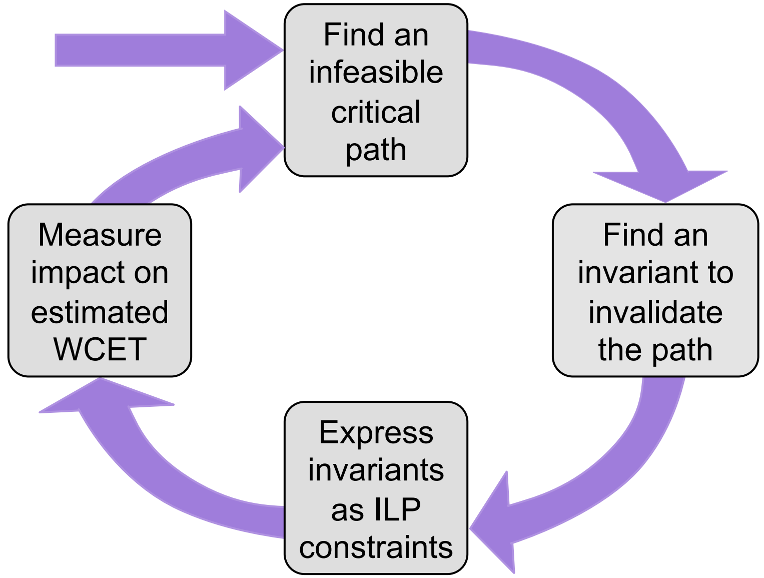
- Correct driver synthesis
  - given model of driver interface, basic behaviour, and hardware
  - driver is automatically generated
  - performance as good as hand-knitted
- Challenge: device spec
- Vision:
  - automatically extract hardware model from HDL description
  - potential impact beyond our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



Kernel Worst-Case Execution Time

NICTA

- seL4 is small as an OS kernel (9 kLOC)
  - … but large as an object for WCET analysis
  - … and full of performance optimisations
- However, we know a lot about it (in a very formal way!)
  - Plenty of invariants proved during verification
  - E.g. loop iteration counts, non-interference
- Can make use of this for WCET analysis
  - Collaboration with WCET experts at NUS (Abhik Roychoudhury)



Trustworthy Systems Are Possible!

NICTA

- World's first functional correctness proof of an OS kernel (seL4)
  - 100 citations in 12 months [SOSP 2009]
- Demonstrated secure virtualization using seL4 microkernel
  - Multi-level secure device for national-security use
  - Small and verifiable trusted computing base
  - Untrusted Linux system
- Demonstrated synthesis of high-performance device-drivers
  - Path to eliminating dominant source of OS crashes

mailto:gernot@nicta.com.au

Google: "ertos"