Open Kernel Labs™

Be open. Be safe.

Next-Generation Embedded Operating Systems

Gernot Heiser

Founder and CTO, *Open Kernel Labs*

Professor of Operating Systems, *UNSW*

Program Leader, *NICTA*

14 May 2007

Embedded Systems are Everywhere

Let's think about the implications...

Lessons from Desktop Systems

Desktop Computers Suck

- They crash
- They get cracked
- They get infected

How about embedded systems?

Wireless Everywhere!

- Bank accounts
  - Is someone monitoring your financial transactions?
  - Is someone taking money out of your account?
- Automobiles
  - Is someone changing your engine settings?
  - Is someone manipulating your breaks?
- Health cards
  - Is someone accessing your medical history?
  - Is someone changing your medication?
- Your home
  - Is someone watching you at home?
  - Is someone entering while you are away?

Computer Unreliability — Why?

- Complexity is the arch-enemy of reliability
  - Complex systems are impossible to understand completely
  - Complex systems are faulty
- Software systems are incredibly complex
  - Smartphones have 5-7 M lines of code (LOC)
  - Cars contain Gigabytes of software
  - Future systems will be even more complex
- Software is buggy
  - Good-quality software has about 1 bug per 1,000 LOC
  - Bug count grows super-linearly in code size
  - Systems have thousands and thousands of bugs

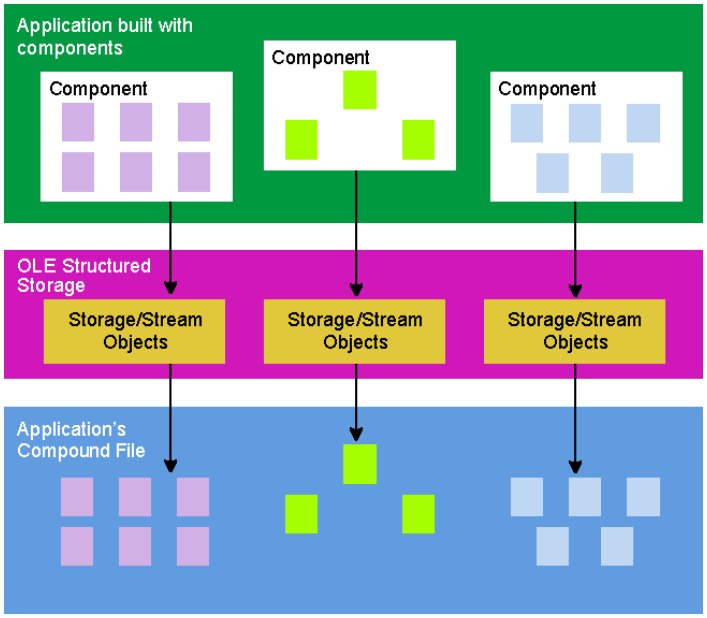
Open Kernel Labs™
Be open. Be safe.

Reliability, Security, Safety, Trustworthiness...

- Reliability is key!
  - unreliable systems are most likely not secure
  - unreliable systems are most likely not safe
  - unreliable systems are most *definitely* not trustworthy!
- Reliability is a *system* challenge
  - permeates all layers of a system
  - requires support from reliable/trustworthy mechanisms
- Reliability is an *operating-system* challenge
  - If the OS isn't reliable, the rest of the system cannot be
  - The OS must provide the mechanisms for desig for reliability
- The OS must be designed for providing and supporting reliability

What Can We Do To Combat Unreliability?

- Componentised software
  - Break system into *components*
  - *Encapsulate* implementation
  - Communication via *interfaces*
  - Can achieve *fault containment*
- Requires *reliable base*
  - Ensures encapsulation
  - Guarantees interfaces
  - Provides communication



- This is the *trusted computing base* (TCB)
  - *Def: Part of the system that can circumvent security*
- Reliability requires the TCB to be *trustworthy*
  - How can we ensure its correctness?

Open Kernel Labs™

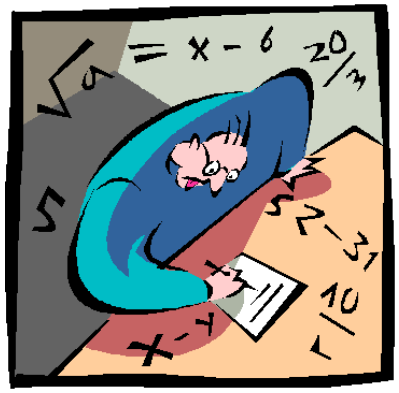
Be open. Be safe.

Open Kernel Labs™

Be open. Be safe.

TCB Reliability — Size is Key!

- Without a *trustworthy* TCB we cannot have a reliable system
- ***TCB must be correct!***
- How can the TCB be made correct?
- ***how can any software be made correct?***
- Testing
  - exhaustive testing only scales to 100s LOC
  - non-exhaustive testing can show the *presence*, not the *absence* of bugs
- Formal methods (mathematical proof)
  - ultimate guarantee
  - scales only to 1000s LOC
- Modularity is key
  - partition problem into mangeable bits



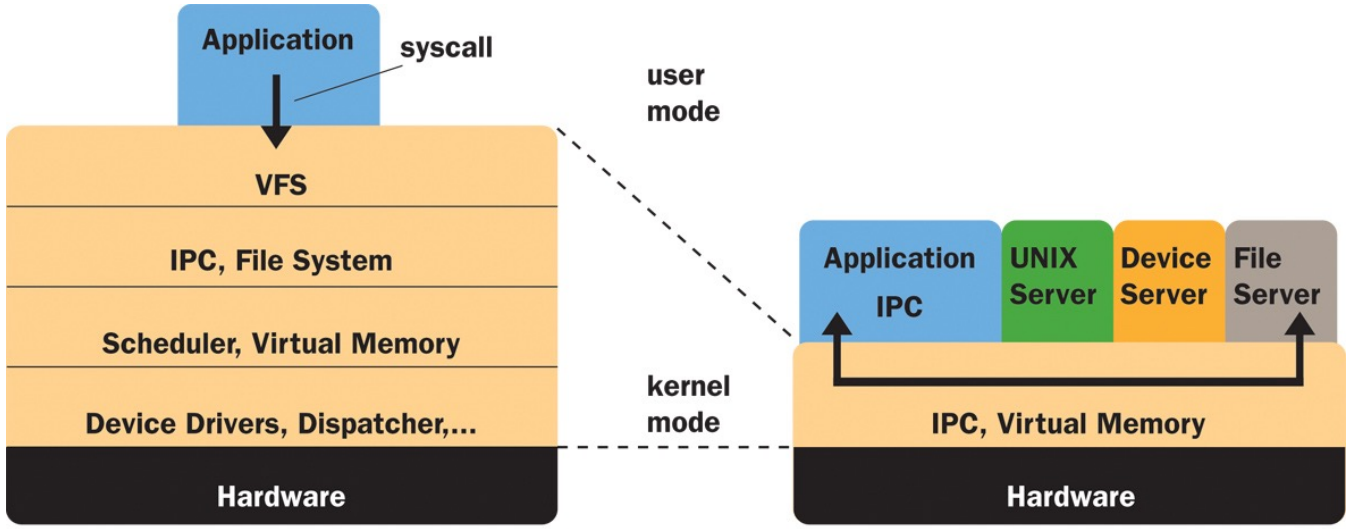
Kernel Size is Key!

- Kernel: Code that executes in privileged mode
  - always part of the TCB
- Kernel verification cannot be subdivided
  - *all kernel code is privileged*
  - there is no protection against misbehaving kernel code
- Kernel must be very small
  - small enough to be tractable by formal methods
  - must have *absolutely minimal functionality*
- Kernel must be a microkernel!
  - only contain code that must execute in privileged mode
  - everything else at user level

What Is a Microkernel?

Open Kernel Labs
Be open. Be safe.

- Small kernel providing core functionality
  - → no other code running in privileged mode
  - → provide mechanisms for building arbitrary systems on top
- OS services provided by user-level servers
- Applications communicate with servers by message-passing IPC



Monolithic OS

Microkernel OS

Open Kernel Labs™

Be open. Be safe.

Open Kernel Labs™

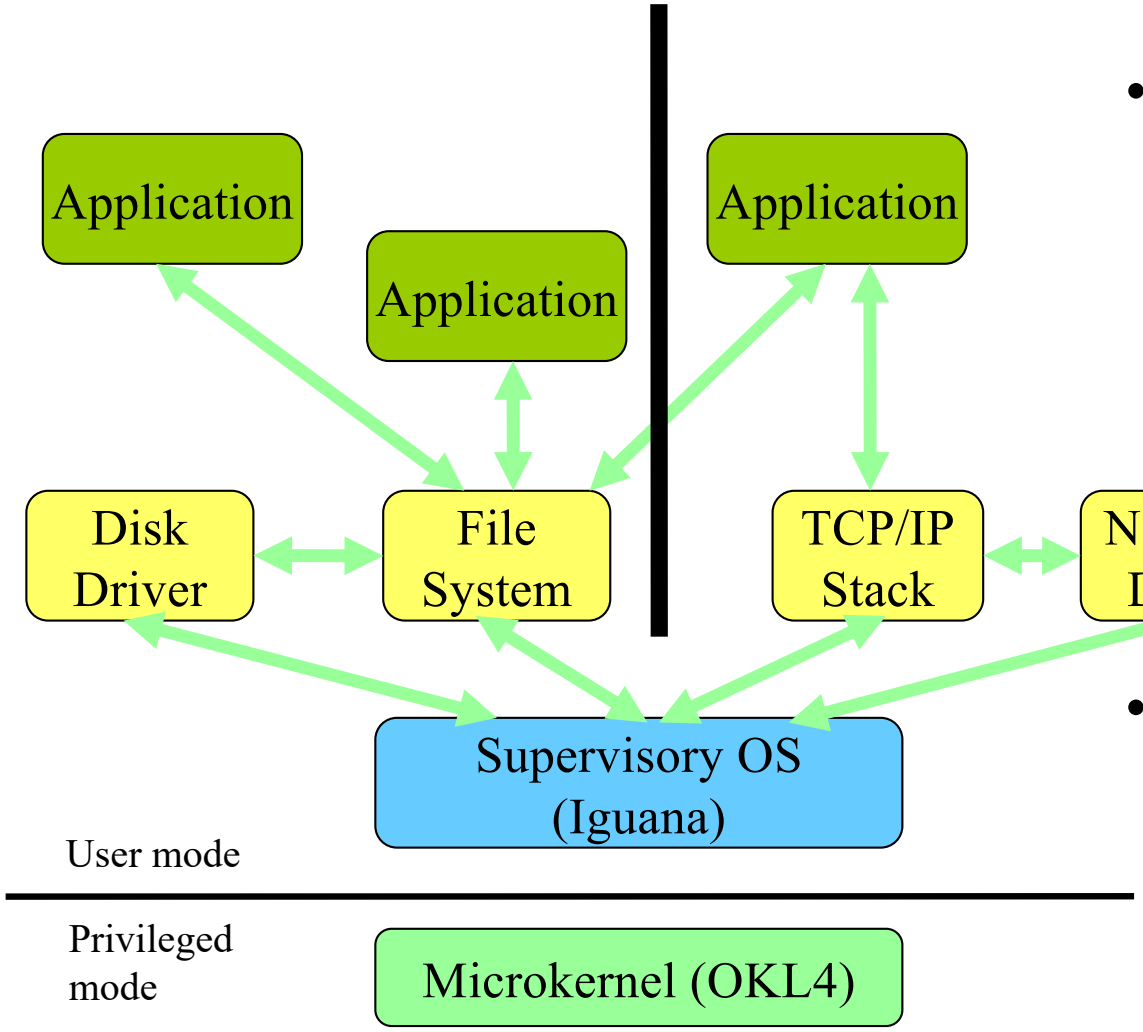
Be open. Be safe.

Reliable Systems — How?

- Need a high-performance microkernel
  - → This exists: OKL4
- Need proof it provides *right mechanisms*
  - → can support secure systems (encapsulation etc)
  - → NICTA project seL4
- Need proof its *implementation is correct*
  - → implementation matches specification
  - → NICTA project L4.verified
- Need *credible timing model*
  - → actual worst-case latencies, based on sound methodology
  - → NICTA project Potoroo
  - → Need *software-engineering infrastructure*
    - support for building large systems on microkernel
    - NICTA project CAmkES

seL4 Project: High-Security Microkernel API

Open Kernel Labs
Be open. Be safe.



- Need flexible and efficient mechanism to authorise communication between processes
- Similarly, need full control over system resources

seL4 Project

- Aims:
  - → API suitable for highly secure systems (military, banking etc)
  - → Complete control over communication and system resources
  - → Proofs of security properties (Common Criteria)
  - → Suitable for formal verification of implementation
- Status:
  - → Semi-formal specification in Haskell
  - → “Executable spec”: Haskell implementation plus ISA simulator
  - → Can port application code before kernel implementation available
  - → C kernel prototype under evaluation
  - → Initial proofs of security properties
  - → To be completed by the end of the year

Open Kernel Labs™

Be open. Be safe.

L4.Verified Project

Open Kernel Labs
Be open. Be safe.

- Aims:
  - → API suitable for highly secure systems (military, banking etc)
  - → Complete control over communication and system resources
  - → Proofs of security properties (Common Criteria)
  - → Suitable for formal verification of implementation
- Status:
  - → Semi-formal specification in Haskell
  - → “Executable spec”: Haskell implementation plus ISA simulator
  - → Can port application code before kernel implementation available
  - → C kernel prototype under evaluation
  - → Initial proofs of security properties
  - → To be completed by the end of the year

Open Kernel Labs™

Be open. Be safe.

Open Kernel Labs™

Be open. Be safe.

Potoroo Project

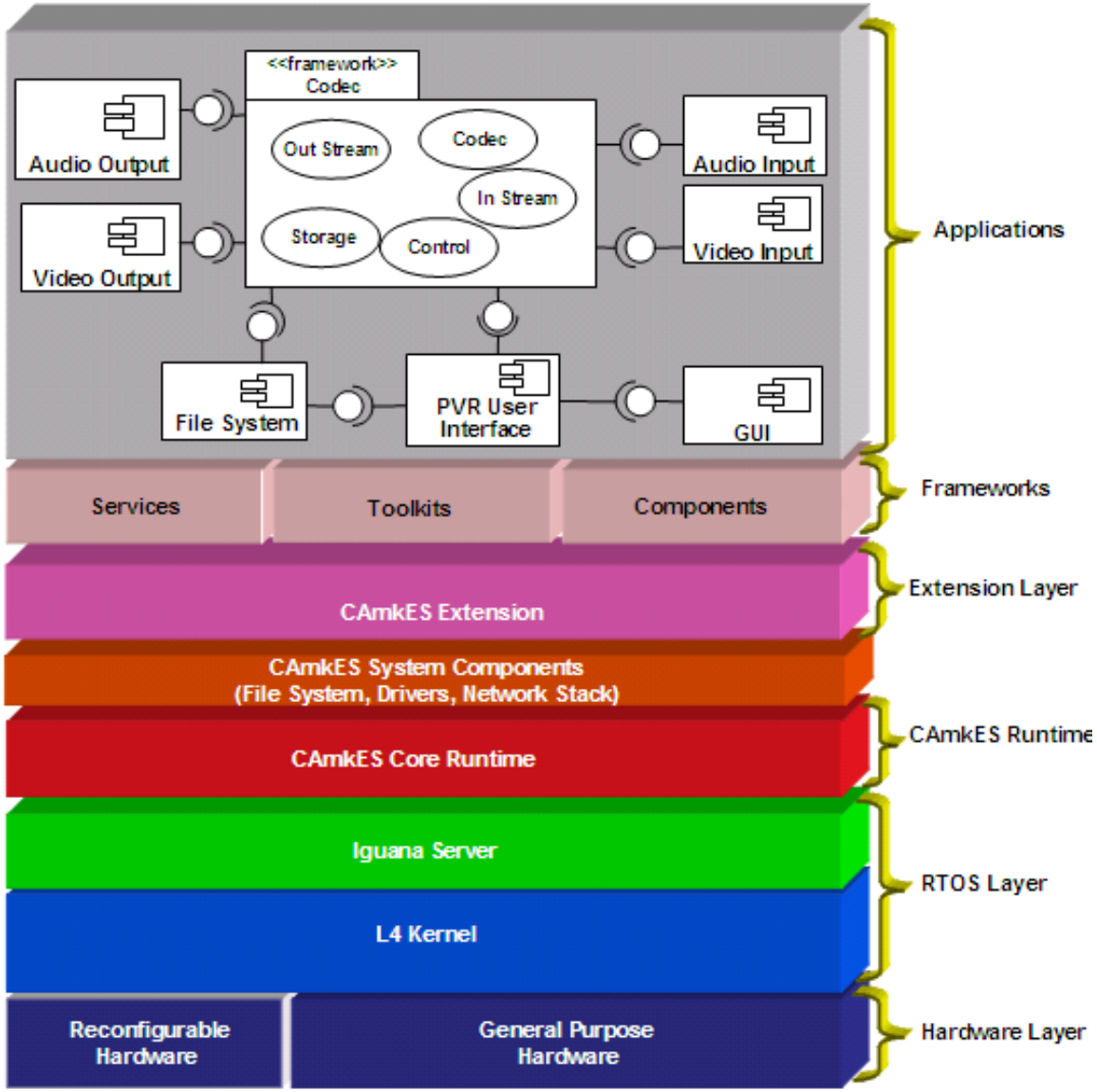
Open Kernel Labs
Be open. Be safe.

- Methodology:
  - → Real measurements of execution times at basic-block level
    - →need not rely on accurate timing models of processors
  - → Static analysis to determine whether worst case was observed
    - → also reduces pessimism (exclude impossible combinations)
- Status:
  - → Commenced April 2004
  - → Prototype tools analyse actual kernel code
  - → Static analysis in progress
  - → To be completed by July 2008

CamkES Project: Component Architecture

Open Kernel Labs
Be open. Be safe.

- Software-Engineering framework for L4
- ***highly modular systems***
- ***components encapsulated by kernel***
- Designed for embedded systems
- ***very lightweight***
- ***no overhead for unused features (dynamic comp.)***



CamkES Project Status

Open Kernel Labs
Be open. Be safe.

- Core system exists
  - → static components, configured at system build time
  - → connectors as first-class objects
  - → architecture definition language and tools
  - → being introduced to production L4 environment
- Dynamic system under development
  - → run-time loading, linking, unloading of components
  - → to be completed by September 2007
- Next phase in planning
  - → model-driven development
  - → non-functional requirements (real-time, power)

Next-Generation Embedded Operating Systems

Open Kernel Labs™
Be open. Be safe.

- Need to be ultra-reliable
  - → based on microkernels
  - → provably-secure mechanisms
  - → provably-correct implementation
  - → credible timing models
- Need to be highly componentised
  - → components protected by microkernel address spaces
  - → can isolate faults, support run-time upgrades
  - → can prove correctness of components, or at least confinement of faults
- NICTA/OK Partnership will deliver this
  - → core technology OKL4 already on market and deployed on products
  - → research agenda for next generation completed next year
  - → commercial availability within 2-3 years

Open Kernel Labs™

Be open. Be safe.

Gernot Heiser
Founder and CTO

Open Kernel Labs
t +61 28306 0550
gernot@ok-labs.com