



Open Kernel Labs™

Be open. Be safe.

How Secure is Your Embedded Design?

Abi Nourai

October 21, 2008

Agenda



*Be open.
Be safe.*

- **Security for mobile device and other embedded systems**
- Virtualization and contribution of virtualization to security and reliability
- Embedded virtualization
- Building a Smartphone using OKL4
- Open Kernel Labs

Some Mobile Device Market Requirements Requiring Security



*Be open.
Be safe.*

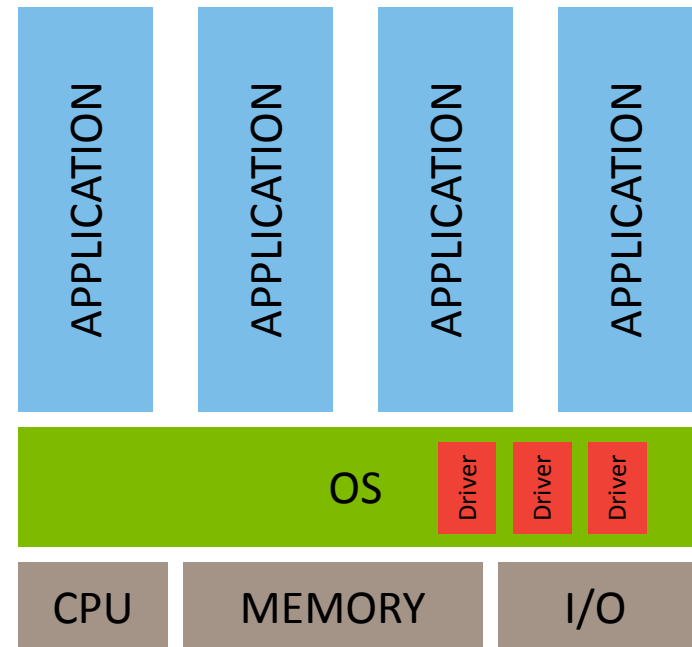
- Three classes of mobile device users
 - End users
 - High quality voice and data service
 - Safety from malicious software (similar to a laptop/desktop)
 - User installed applications and downloaded content
 - Complete internet experience
 - Network operators
 - High quality voice and data service
 - Subsidized handsets must be kept from use on “other” networks
 - Content owners
 - Enterprises require specific security guidelines be addressed for devices that are allowed to access their corporate network
 - Financial institutions require specific security guidelines be addressed for financial transactions using mobile devices
 - Media content owners expect their license terms for usage on a mobile device to be enforced

Security In Embedded Systems



Be open.
Be safe.

- Types of threat
 - Denial of service
 - Exposure of data
 - Unauthorized access
- Trusted components that are not trustworthy create vulnerability
- Designing for security
 - Isolation between trusted and untrusted components
 - Principle of least authority/privilege
 - Minimal trusted computing base and complexity for trusted components
 - Protect integrity and confidentiality against *internal* and external exploits
 - Need control over *information flow*
 - Who has access to what
 - Communication channels



Agenda

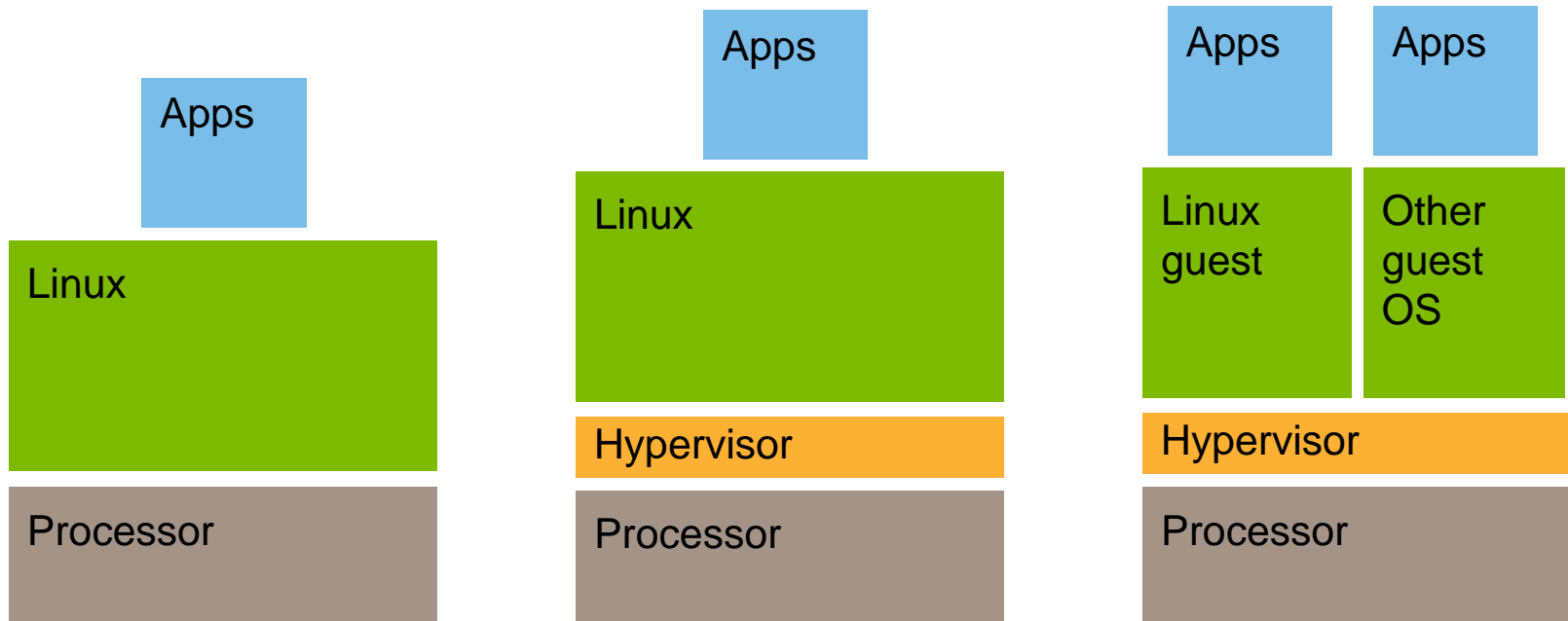


*Be open.
Be safe.*

- Security for mobile device and other embedded systems
- **Virtualization and contribution of virtualization to security and reliability**
- Embedded virtualization
- Building a Smartphone Using OKL4
- Open Kernel Labs

What are Virtual Machines?

- OS normally runs directly on hardware
- Virtualization inserts software layer between OS and hardware
 - Called *hypervisor* or *virtual-machine monitor*
 - Supports several concurrent OSes, called *guests*

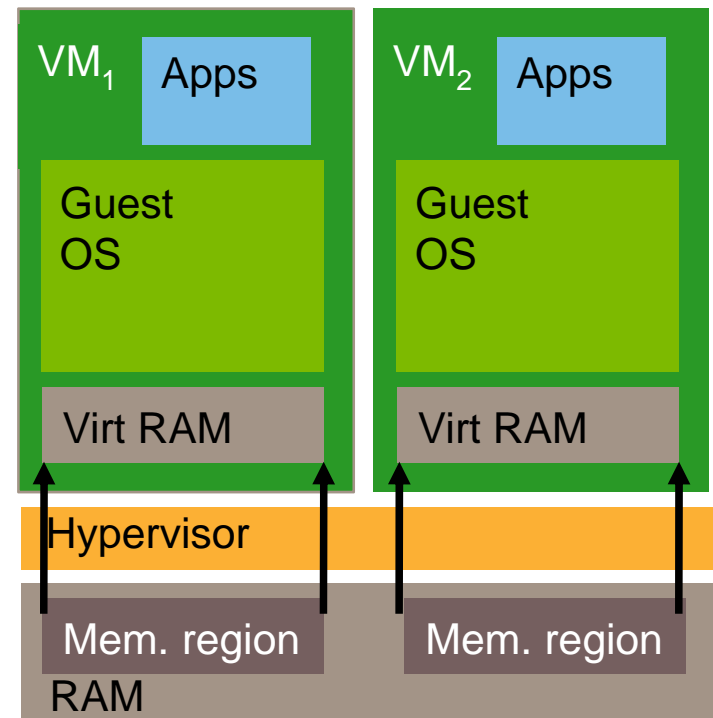


The Hypervisor



Be open.
Be safe.

- Hypervisor partitions and multiplexes hardware between guests
- Hypervisor is in complete control of all physical resources
 - Memory, devices, interrupts, CPU time
- Virtual machines access virtual resources
 - Mapped to physical resources by hypervisor
- Guest OS executes at lesser privilege
 - *Only hypervisor runs in privileged mode*
 - Essential to ensure hypervisor has control
 - Some products violate this principle and run guests in most privileged mode
 - Creates an undesirable, significant increase in the amount of privileged code
 - Watch out for this when comparing performance



Enterprise vs. Embedded Virtualization: Isolation vs. Cooperation



*Be open.
Be safe.*

Enterprise

- Independent services
- Emphasis on isolation
- Inter-VM communication is secondary
 - Performance secondary
- VMs connected to Internet (and thus to each other)
 - Can freely communicate with each other and the rest of the world

Embedded

- Integrated system
- Cooperation with protection
- Inter-VM communication is critically important
 - Performance is crucial
- VMs are subsystems accessing shared (but restricted) resources
 - Communication must be controlled

→ Need: **strong protection, yet highly-efficient (low-latency, high-bandwidth) communication**

→ This doesn't fit the virtual machine model!

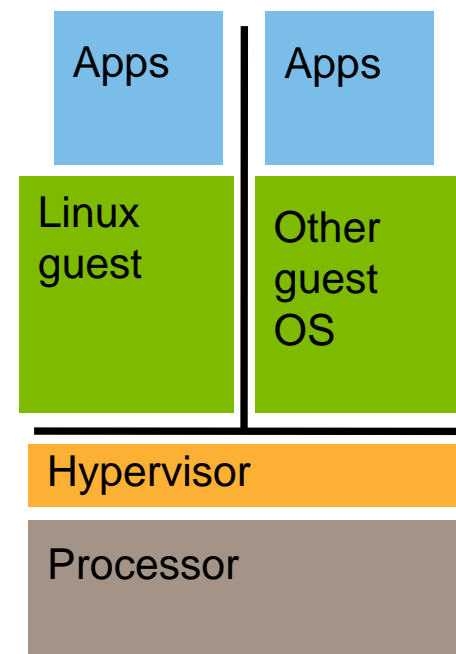
→ An embedded hypervisor must provide a more generalised solution to address the specific needs of embedded system applications

How Virtualization Contributes to Security and Reliability



*Be open.
Be safe.*

- Isolate security critical functions and data from other functions and data
 - Much harder for attacks to cross from one machine to another than from one application to another
- Isolate functions so that faults in one subsystem do not propagate to create faults in another subsystem
- You could do the same using physically separate processors BUT
 - More expensive
 - More likely to result in CPU capacity that is under-utilized
 - Reduced flexibility results from direct tie between the security architecture and the hardware design
- Virtualization means running guest OSes entirely at user level
 - If a guest OS is compromised it is isolated from the other guest OSes
 - Beware approaches that involve guest OS software running in supervisor mode



What Would Further Improve Security?



*Be open.
Be safe.*

- Finer grained partitioning of software into isolated components
- Proven trustworthy system software enforcing partitioning and governing all access to system resources (physical memory, device interrupts, etc.)
- A minimal trusted computing base for security-critical applications
- Minimizing the amount of code running in privileged mode and therefore able to circumvent system security policies
- Mechanisms allowing a configurable level of security by controlling communication between components and use of system resources by components
- Running device drivers in their own, protected domain, and at user level
- Reducing the dependency on an Application OS for security policy enforcement

Virtualization does not provide these, something more is required

Agenda



*Be open.
Be safe.*

- Security for mobile device and other embedded systems
- Virtualization and contribution of virtualization to security and reliability
- **Embedded virtualization**
- Building a Smartphone using OKL4
- Open Kernel Labs

Embedded Hypervisor: Improving on the Classical Model

- System virtualization (as provided by the classical hypervisor)
- Mechanisms for enabling high-performance interaction between VM's
 - Fast context switches
 - Shared memory
 - High-performance IPC
- Mechanisms enabling better security and reliability
 - Encapsulated device drivers
 - Highly trustworthy privileged code
 - Minimal-TCB execution environment
 - Fine-grained access/communications control
 - Ideally: Answer to future security/safety challenges

OKL4 Microkernel Technology: Ideally Addresses the Requirements of an Embedded Hypervisor



*Be open.
Be safe.*

- Small kernel providing core functionality
 - 10 kLOC, just enough for required mechanisms, no policy
 - No other code running in privileged mode, not even device drivers
 - Provide mechanisms for building arbitrary systems on top
 - Services can have 15 kLOC TCB
- Uses MMU to isolate address spaces
- Fast message-passing IPC operation
 - Performance close to the hardware limit
- Shared memory for bulk memory transfer
- 10-year track record of high-performance Linux virtualization
- Suitable as a basis for general operating systems
- Powerful and sophisticated security mechanisms
- *Open source!*

OKL4: The Embedded Hypervisor

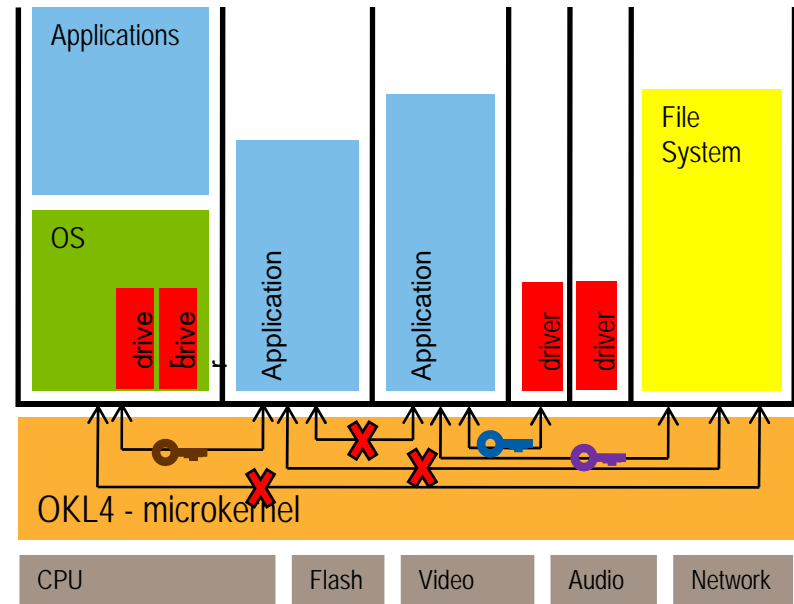


Be open.
Be safe.

Secure HyperCell™ Technology

- Goes well beyond the classical hypervisor model
- Enables virtualization and componentization
 - VM = OS plus its applications in a cell
 - Lightweight execution environments
 - Drivers
 - HW enforced isolation between cells
- Control over communication between cells
 - Required for mandatory access control
- Fast context switching and high performance inter-cell communication
- Highly trustworthy privileged code
 - Small, clean, and open source

User level "cells" contain components in HW-enforced address space isolation



Privileged mode software limited to the OKL4 microkernel

OKL4 Lightweight Execution Environments



Be open.
Be safe.

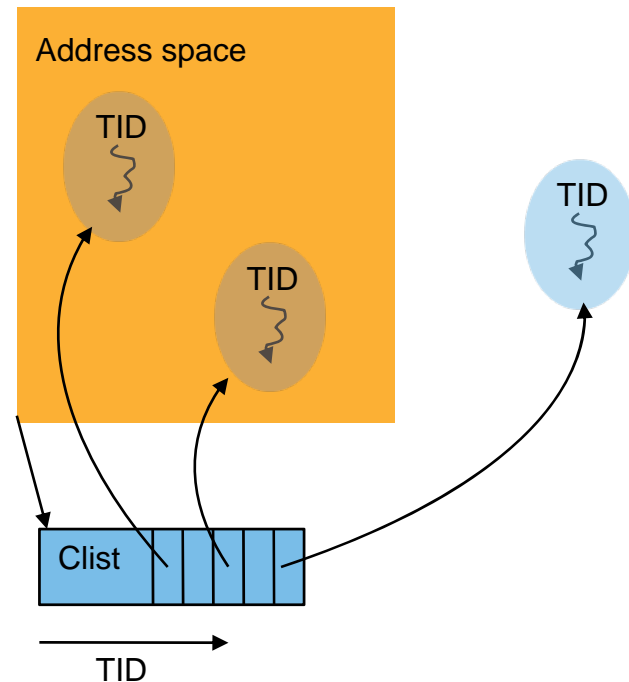
- Conventional virtualization solutions only support running complete operating systems (Linux, WinCE, ...) in a virtual machine
- OKL4 supports execution of *lightweight execution environments* (LWEE)
 - A LWEE executes an application or system service directly on OKL4 without inclusion of a complete operating system
 - Critical services software can be hosted directly on top of the OKL4 microkernel to reduce their *trusted computing base* (code they depend on for correct functionality)
 - Enables secure, fine-grained componentization of a system for increased reliability and robustness
 - Separate security-sensitive and proprietary software from an Application OS (e.g. Linux) without running a second instance of an Application OS
 - Migrate DSP software components (e.g. multimedia codecs) to host processor without sacrificing real-time guarantees or increasing power consumption

OKL4 Resource and Security Management



Be open.
Be safe.

- Resources controlled by kernel-protected *capabilities*
- Capability conveys privilege
- Efficient resource delegation by providing set of caps
- Subject to system-wide security and resource-management policies defined by system designer
- *Secure HyperCell™ technology* provides security management framework



Capabilities vs. ACLs



*Be open.
Be safe.*

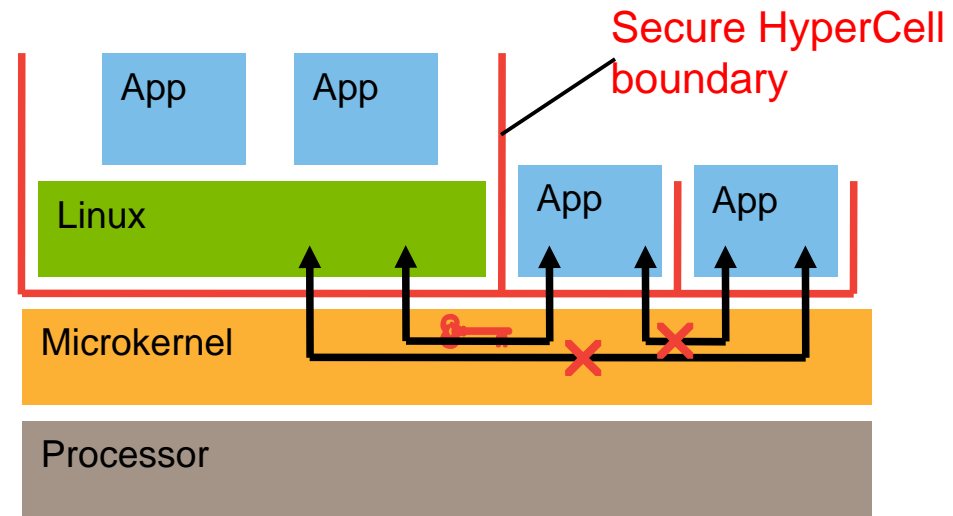
- Within computer systems, the two fundamental means of enforcing privilege separation are [access control lists](#) (ACLs) and [capabilities](#). The semantics of ACLs have been proven to be insecure in many situations (e.g., [confused deputy problem](#)). It has also been shown that ACL's promise of giving access to an object to only one person can never be guaranteed in practice. Both of these problems are resolved by capabilities. This does not mean practical flaws exist in all ACL-based systems, but only that the designers of certain utilities must take responsibility to ensure that they do not introduce flaws. ¹
- [Mandatory access control](#) can be used to ensure that privileged access is withdrawn when privileges are revoked. For example, deleting a user account should also stop any processes that are running with that user's privileges. ¹
- [Capability](#) and [access control list](#) techniques can be used to ensure privilege separation and mandatory access control. ¹

Trusted Computing Base



Be open.
Be safe.

- The **trusted computing base** (TCB) of a [computer system](#) is the set of all [hardware](#), [firmware](#), and/or [software](#) components that are critical to its [security](#), in the sense that [bugs](#) occurring inside the TCB might jeopardize the security properties of the entire system. ¹
- The careful design and implementation of a system's trusted computing base is paramount to its overall security. ¹
- Identify security-critical apps
 - Crypto services
 - DRM agents
 - Trusted binary loaders
- Run services directly on OKL4 in their own cell
- Specify authorized communication channels in accordance with security policy
- *TrustZone-like isolation*, but more general
- TCB: <15kLOC, much smaller than for a Linux application

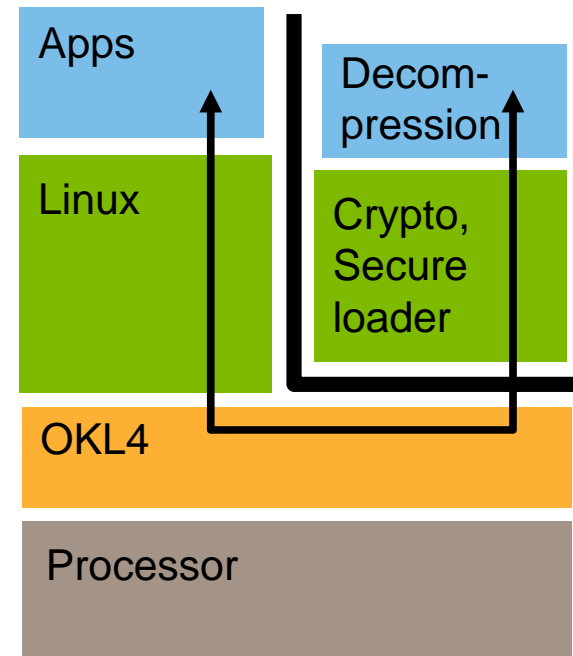


Software IP Protection



*Be open.
Be safe.*

- Protect valuable IP (software or algorithms) from theft
 - Customer example: Device runs Linux and contains high value compression algorithm
 - Protect against reverse engineering of algorithms
 - Enable secure update
 - Requires trustworthy code
 - Load from Flash to RAM
 - Decrypt code
 - Protect from Linux side access
 - Provide secure field update path

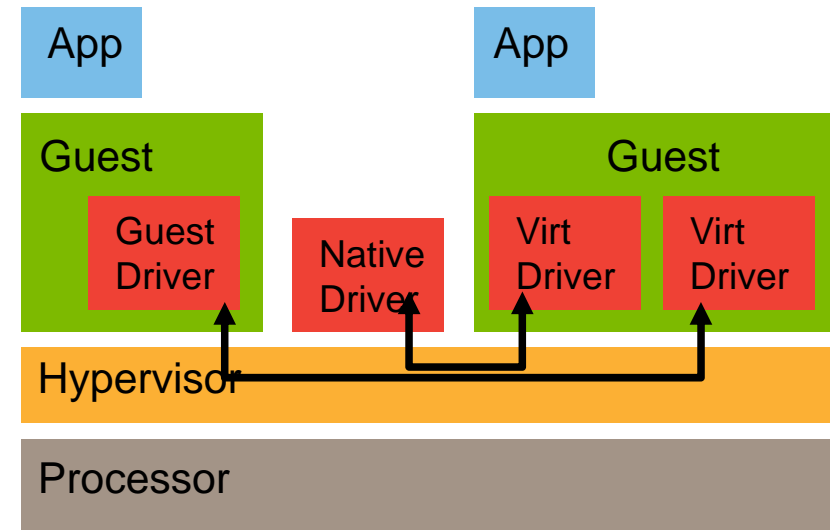


OKL4 Embedded Hypervisor Device Drivers



*Be open.
Be safe.*

- Simple embedded systems
 - Drivers owned by OS
 - Drivers privileged
- With OKL4 embedded hypervisor
 - Guest OS or OKL4 native drivers
 - User level drivers
 - Device sharing mechanisms
- Benefits of OKL4 native drivers
 - Smaller TCB
 - Fault isolation
 - Control access using caps
 - Improves separation of trusted and un-trusted subsystems



OKL4 Performance



Be open.
Be safe.

- OKL4 and OK Linux features Fast Address Space Switching (FASS) Technology
 - By taking advantage of ARM domains and FCSE extensions
 - “Faster than native” virtualized Linux context switching performance on ARMv5
- *Unmatched IPC performance* operating as the backbone for interrupt delivery
 - OKL4 IPC implemented in assembly “fastpaths” to achieve zero-copy overhead
 - Low latency interrupt handling (best-case and worst-case)
- OKL4 high-performance inter-cell communication
 - Communication channels built over shared memory and lightweight OKL4 IPC
 - Allows for highly cooperative yet modular systems without sacrificing performance
- OKL4 microkernel design provides *strong real-time guarantees*
 - By providing lightweight primitives backed by short kernel execution paths
 - Enables consolidation of real-time and non-real-time system components on a single-core solution

Other Benefits of OKL4 Embedded Hypervisor



*Be open.
Be safe.*

- Reduce deployment cost by
 - Consolidating physically separate systems onto shared processing resources to eliminate unused capacity
 - Implementing a more modular and therefore more maintainable software architecture
- Reduce time to market
 - Replace OS porting with VM integration when adding new applications
 - Reuse legacy SW components in their legacy OS environment “as-is” alongside a new application OS
- Reduce development effort and improve return on software development investment by
 - Reuse one implementation of a critical function in multiple products with different hardware and operating system components
 - Deploy the same software architecture on single and multicore systems
 - Applies either over time or across products at a given time

Agenda



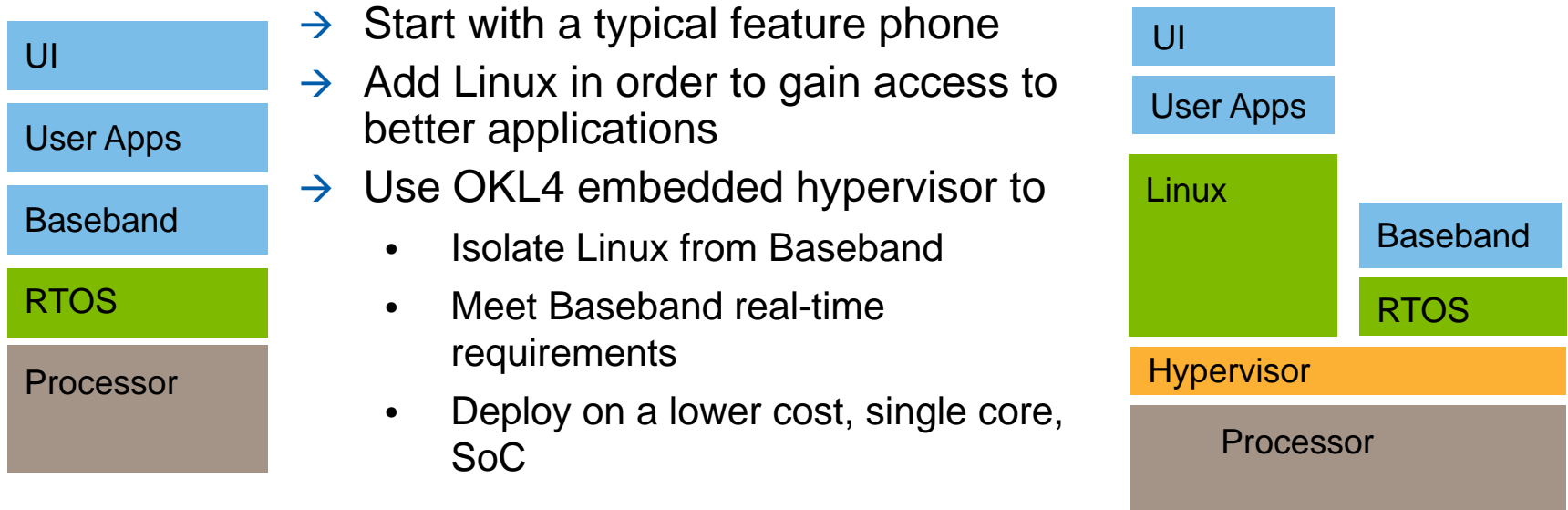
*Be open.
Be safe.*

- Security for mobile device and other embedded systems
- Virtualization and contribution of virtualization to security and reliability
- Embedded virtualization
- **Building a Smartphone using OKL4**
- Open Kernel Labs

Building a Single Core Smartphone Using OKL4



*Be open.
Be safe.*



- Start with a typical feature phone
- Add Linux in order to gain access to better applications
- Use OKL4 embedded hypervisor to
 - Isolate Linux from Baseband
 - Meet Baseband real-time requirements
 - Deploy on a lower cost, single core, SoC

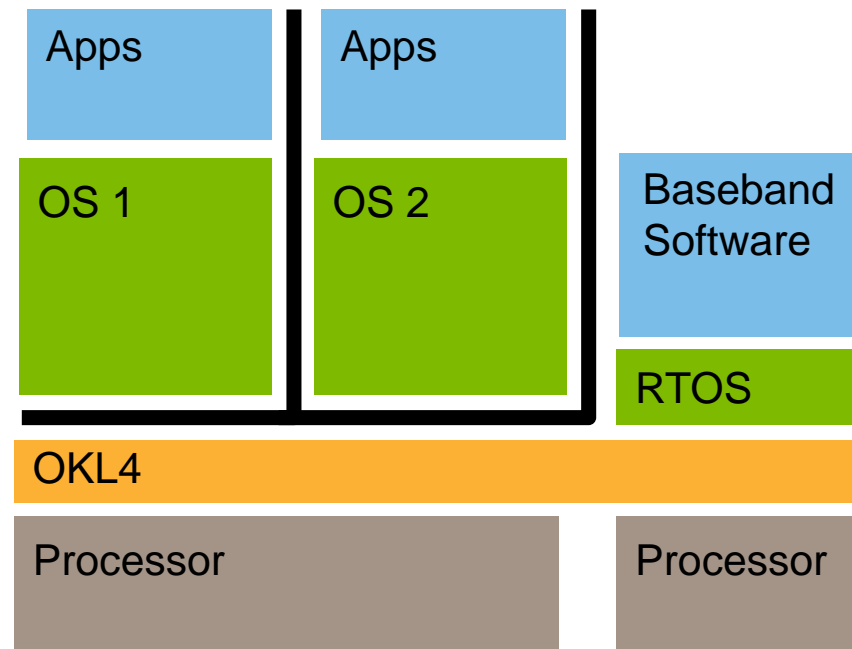
- OK Labs provides OK Linux, a para-virtualized Linux kernel for the OKL4 embedded hypervisor
 - Linux runs in user mode, securely restricted to the HW resources dedicated to it
 - OK Linux preserves complete binary compatibility with Linux applications
 - Features OK Fast Address Space Switching Technology (FASS) for ARMv5 processors
 - “Faster than native” context switching performance on the ARM926-EJS

Building a “Smarter” Phone



*Be open.
Be safe.*

- Multiple instances of the same OS, different version, different OSes
- Separate enterprise and private phone environments
- Separate open and closed application environments
- Integrate the best software from different platforms in the same product



Agenda



*Be open.
Be safe.*

- Security for mobile device and other embedded systems
- Virtualization and contribution of virtualization to security and reliability
- Embedded virtualization
- Building a Smartphone using OKL4
- **Open Kernel Labs**

Who We Are



*Be open.
Be safe.*

- Your best choice for a provider of embedded hypervisor solutions
- Founded in 2006 with headquarters in the US and engineering in Sydney, Australia
- With ownership of technology resulting from a 10+ year program of research and development on microkernel technology across 7 HW architectures
- Offering it's customers unique access to the fruits of continuing research
 - A commercialization vehicle for NICTA, Australia's Centre of Excellence in ICT research and an ongoing partnership with University of New South Wales (UNSW)
- The provider of an embedded hypervisor solution that is already deployed in 250+ million of end-user devices

The Open Kernel Advantage



Be open.
Be safe.

- *Open source*
- Mature: deployed on 100+ million devices
 - 2006- Toshiba W47T CDMA phone selling in Japan
 - 2007- 3G phones from HTC, LG shipping since July
 - 2008- Samsung SPH-m800 Instinct™ and Linux phones later this year
- No compromises: secure *and* fast
 - Encapsulated device drivers
 - Unbeaten performance — for ten years!
 - ARM9 context switches as fast as ARM11
 - Information-flow control mechanisms
- Active developer community
- Aggressive roadmap to formal correctness proofs
- Direct IP pipeline from world-class research center (NICTA)





*Be open.
Be safe.*



Open Kernel Labs™

Be open. Be safe.

Abi Nourai
Technical Director

Open Kernel Labs
anourai@ok-labs.com