Secure Architecture and Implementation of Xen on ARM for Mobile Devices

Sang-bum Suh
sbuk.suh@samsung.com
SW Laboratories
CTO, Samsung Electronics

April 17, 2007

Presented at Xen Summit Spring 2007, IBM TJ Watson

Contributor

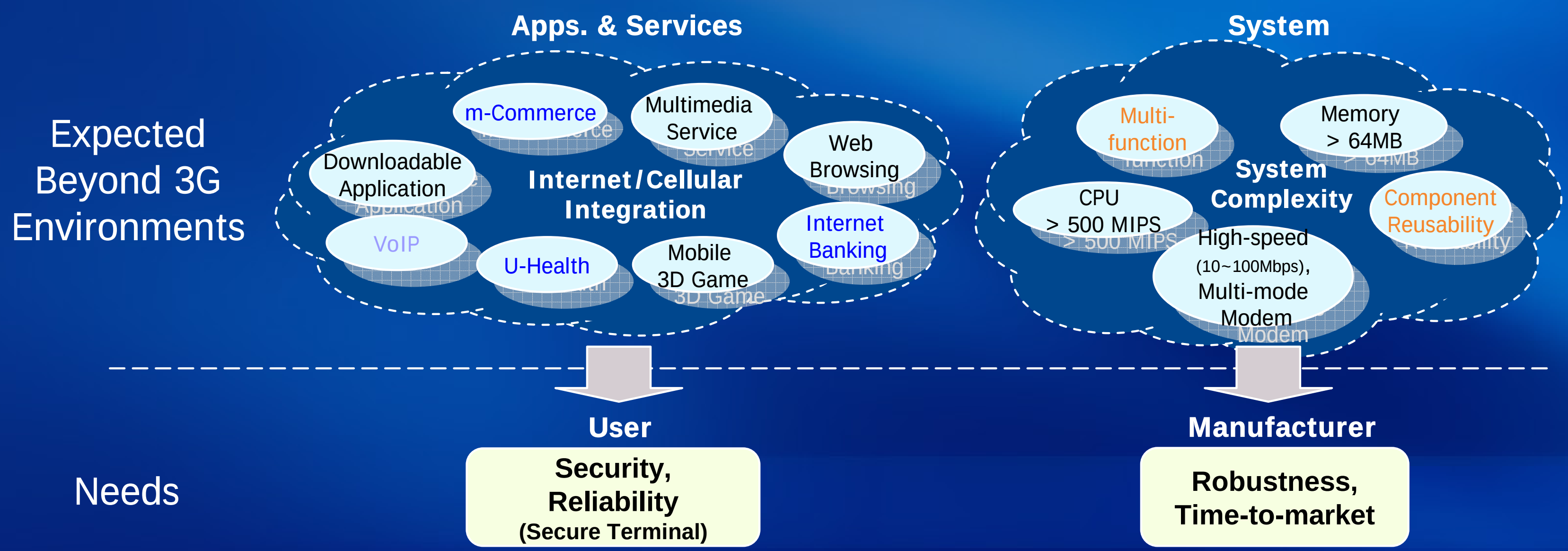
- Sang-bum Suh
- Joo-Young Hwang
- Sung-min Lee
- Sungkwan Heo
- Sangdok Mo
- ChanJu Park
- Seong-Yeol Park
- Jong-Tae Kim
- Bokdeuk Jeong
- Chul ryun Kim
- Jaemin Ryu
- Jaera Lee
- Mikhail Pozhenko

Agenda

- Requirements for Beyond 3G Mobile Device
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix

Requirements for Beyond 3G Mobile Devices

- **High-level Requirements**
  - **End user: Secure and reliable mobile terminals for mobile Internet services using WiBro**
  - **Manufacturer: Robustness though complexity of devices gets increased**
  - **Contents provider: Protection of IP rights in end-user terminals**
  - **Carrier companies: Open and Secure Mobile Platform**
    - **OSTI (Open Secure Terminal Initiative): NTT DoCoMo, Intel**



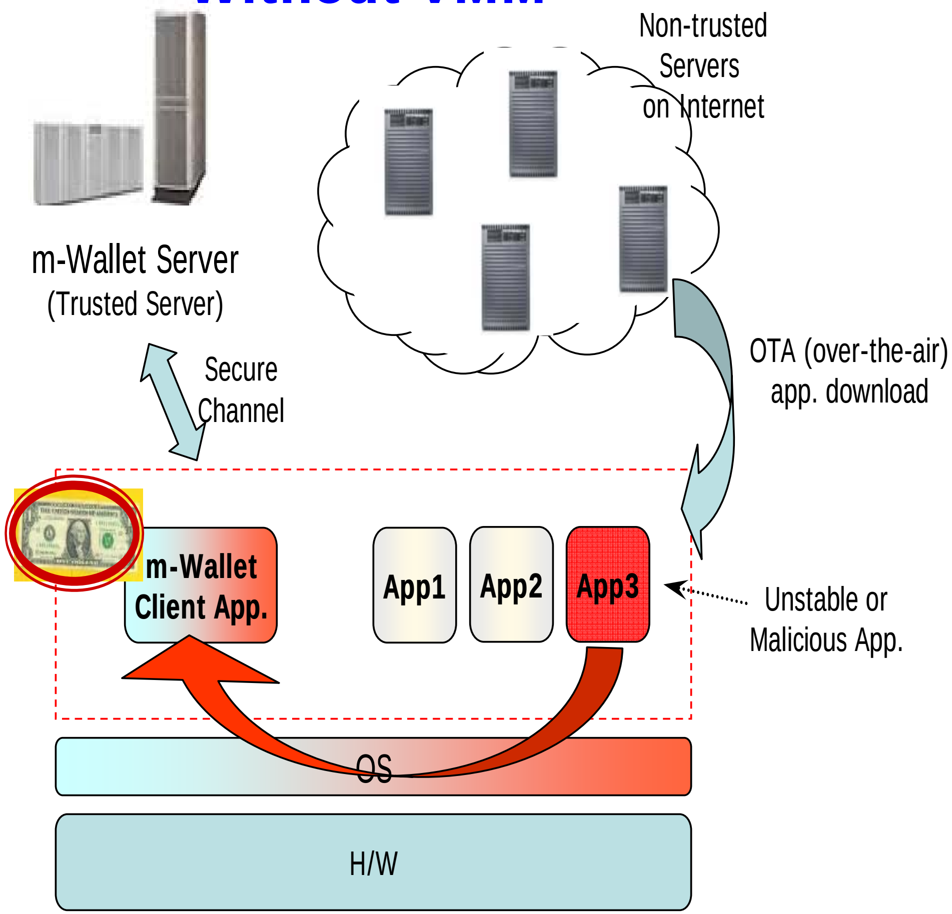
Beyond 3G environments and Needs

Threats to Mobile Devices

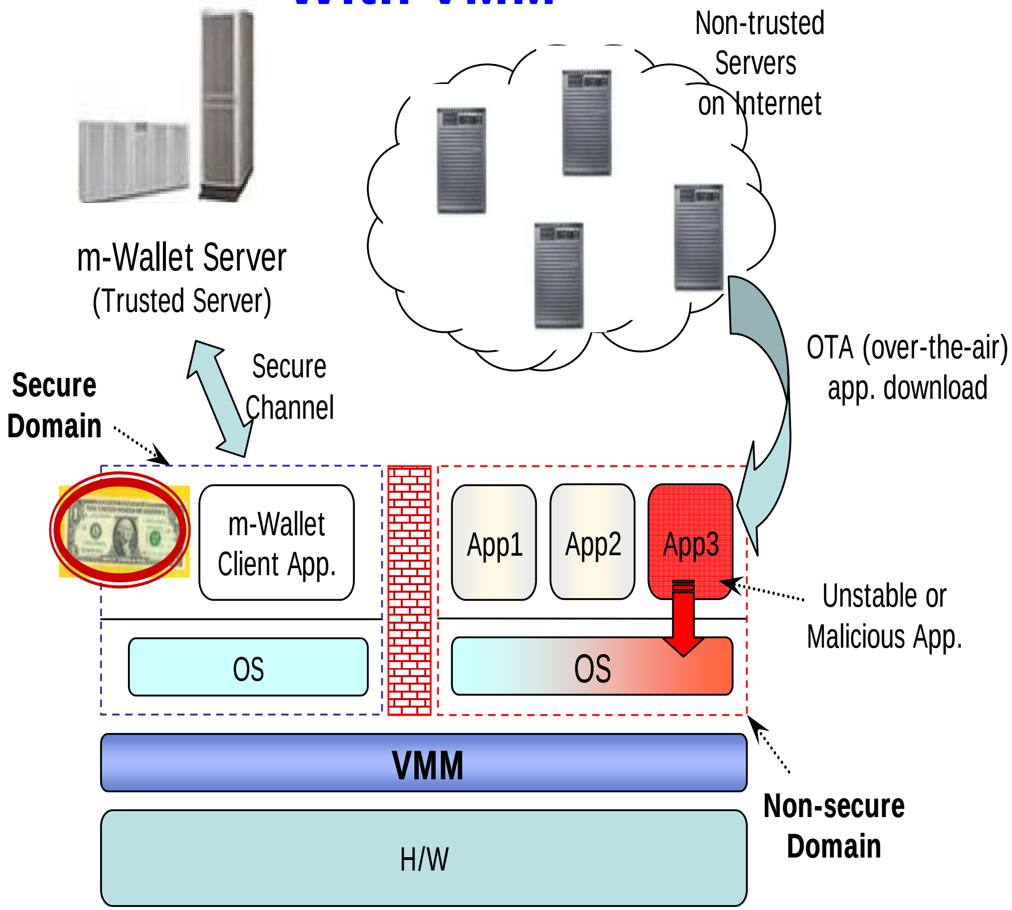
- According to McAfee, threats to mobile devices will continue to grow in 2007
  - The number of malware created for Windows CE/Mobile and Symbian was expected to reach 726 by the end of 2006, from an estimated 226 at the end of 2005 [KAW06]
- Attacks on mobile banking and trading
  - Steals financial data and sends them to a remote attacker
  - Examples [GOS06]
    - StealWar Worm (2006), Flexispy Trojan (2006), Brador Backdoor (2004)
- Denial of service (DoS) attacks
  - Inappropriate execution of instructions consuming system resources (e.g., memory, CPU, battery), resetting a system
  - Examples [GOS06]
    - Cabir Worm (2004), CommWarrior Worm (2005), Skulls Trojan (2004), Mobler.a Worm (2006), Cxoever Worm (2006)

Typical User Scenario

Without VMM



With VMM



* VMM = Virtual Machine Monitor

Features for Secure Mobile Devices

SAMSUNG

- Low-overhead system virtualization
- Separation of guest domains
- Hot plug-in/-out of guest domains
- Secure boot
- Secure storage
- Access control

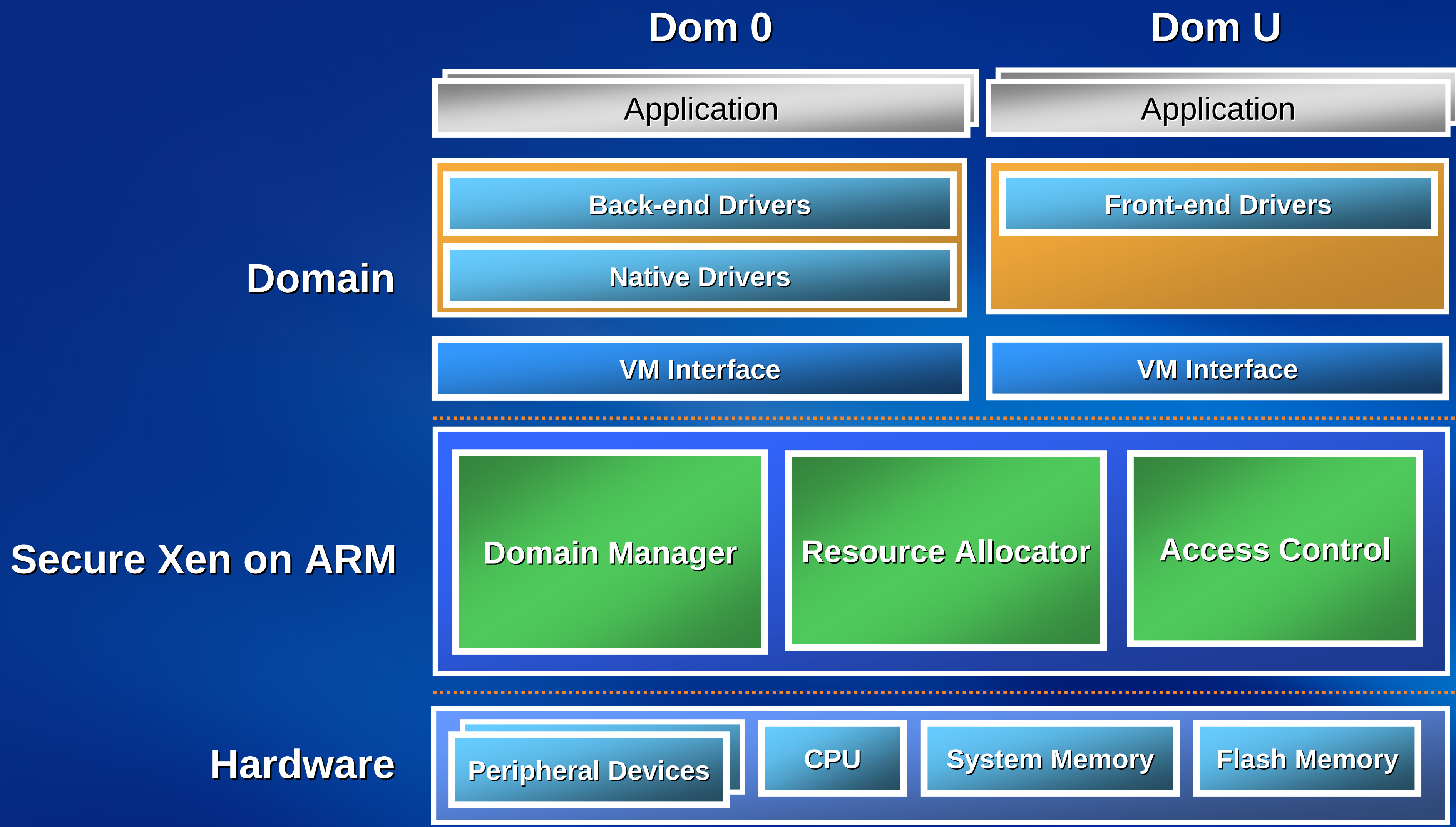
Agenda

- Requirements for Beyond 3G Mobile Device
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix

Goal and Approach

- Goal
  - Light-weight secure virtualization technology for beyond 3G mobile devices
- Approach
  - Design and implementation of
    - VMM on ARM using Xen architecture
    - Security features using Xen on ARM:
      guaranteeing confidentiality, integrity, and availability
- Deliverables
  - VMM: Secure Xen on ARM
  - Dom0, DomU: Para-virtualized ARM Linux-2.6.11 kernel/ device drivers

Architecture: Secure Xen on ARM



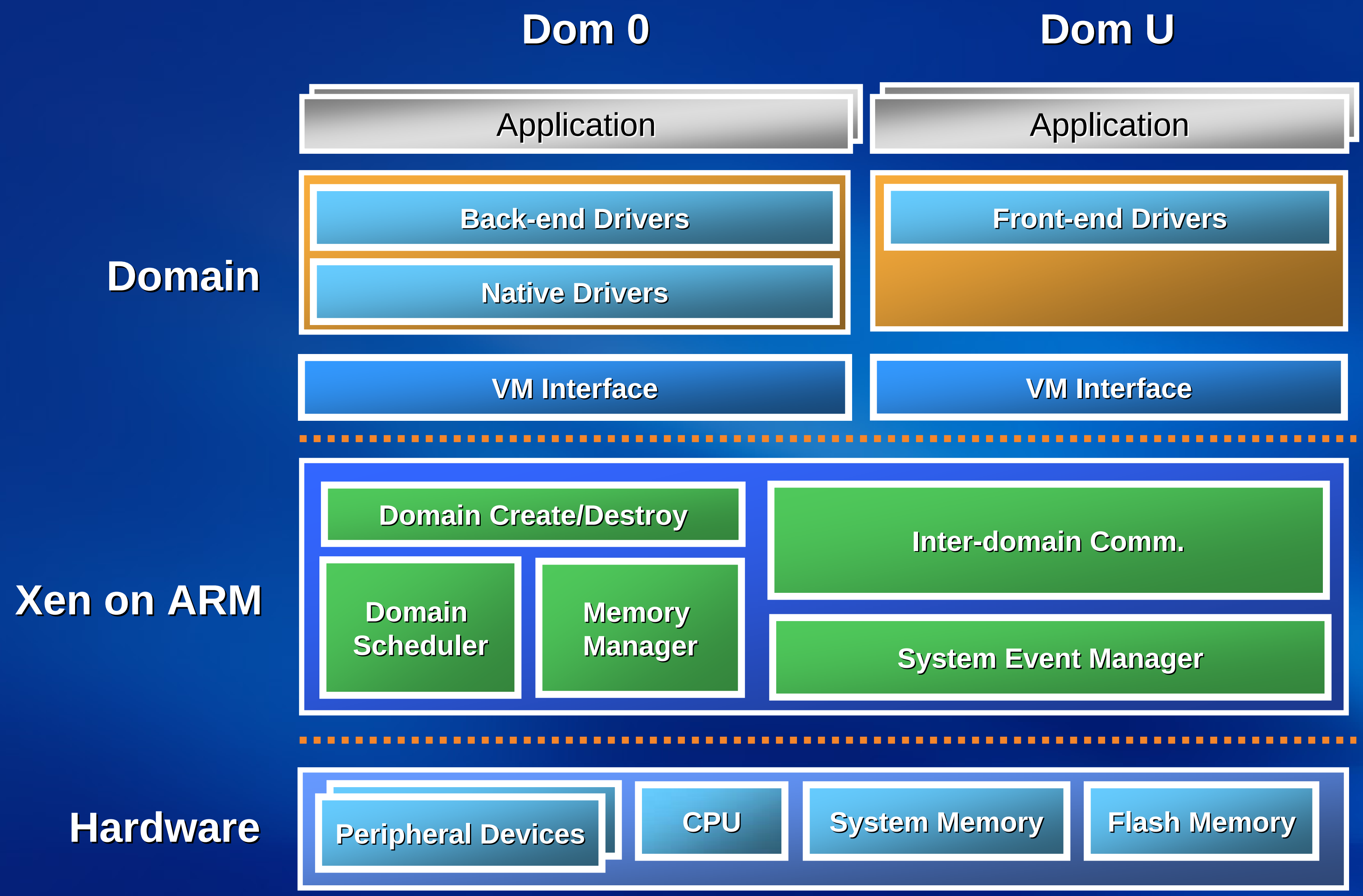
Development Environments

- HW and SW Environments
  - A Reference System for Implementation
    - SW
      - Xen : Xen-3.0.2
      - Linux : ARM Linux-2.6.11
      - GUI : Qtopia
    - HW
      - Processor : ARM-9 266Mhz (Freescale i.MX21)
      - Memory : 64MB
      - Flash : NOR 32MB / NAND 64MB
      - LCD : 3.5 inch
      - Network : CS8900A 10Base-T Ethernet Controller
  - Development Environments
    - OS : Fedora Core 6
    - Cross-compiler: Montavista ARM GCC 3.3.1
    - Debugger : Trace32 ICD (In Circuit Debugger)

Agenda

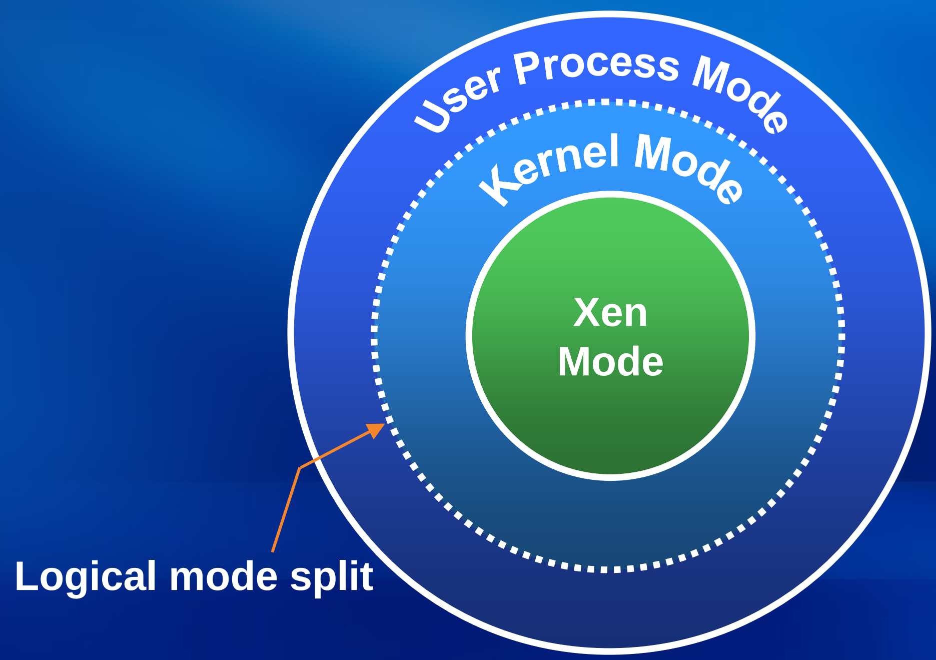
- Requirements for Beyond 3G Mobile Device
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix

Xen on ARM Architecture



CPU Virtualization (1/2)

- **Physically two privilege modes (User mode and Supervisor mode) in ARM CPU. However,**
  - **Supervisor mode is assigned to Xen mode**
  - **User mode is split into two logical modes (kernel and user process of Linux)**
  - **Address space protection between kernel mode and user process mode is guaranteed by *ARM domain access control mechanism*.**



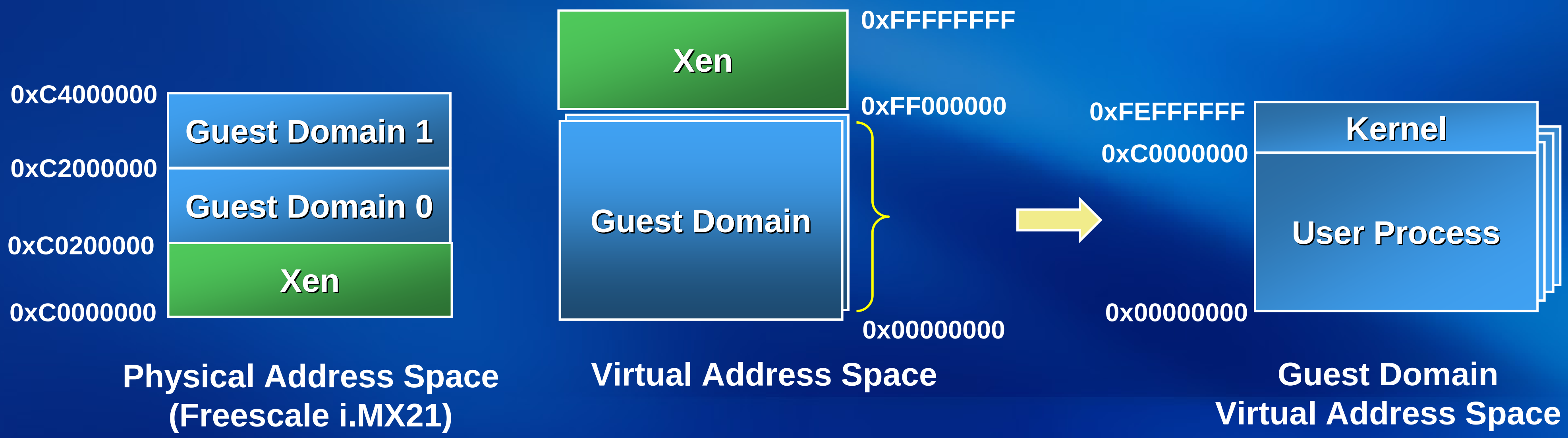
Virtualized CPU modes

CPU Virtualization (2/2)

- **Exception Handling**
  - **Para-virtualization of system calls.**
    - **System calls are implemented with software interrupt.**
    - **In Xen on ARM, system calls are interpreted by Xen**

Memory Virtualization (1/3)

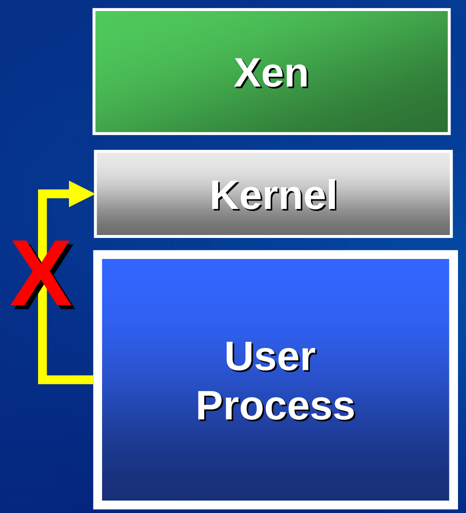
- Memory Map
  - Xen and guest domain (kernel + user process) are mapped on a same virtual address space.



Memory Virtualization (2/3)

- **Domain Access Control is used to prevent a user process from accessing to address space of kernel in ARM CPU user mode.**
  - **Kernel Mode : D0, D1, D2 enabled**
  - **User Process Mode: D0, D2 enabled, D1 disabled**

Virtual Address Space



ARM Domain (Dynamic)	Page Table Access Permission Field (static)
D0	S: RW, U: No Access
D1	S: RW, <u>U: RW</u>
D2	S: RW, <u>U: RW</u>

D3 ~ D15 : reserved for future use.

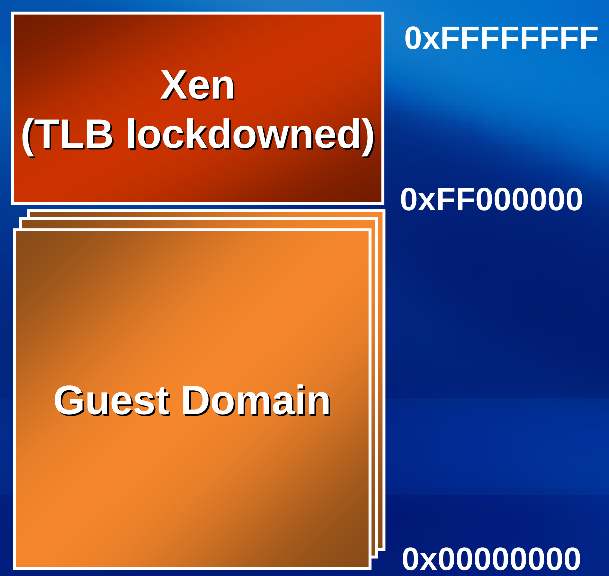
ARM Domain access bit assignments [ARM01]

Access	Bit Field	Comments
Manager	11	No access control
Reserved	10	
Client (Enabled)	01	Use page table access permission field.
No Access (Disabled)	00	

* S : ARM Supervisor mode
U : ARM User mode

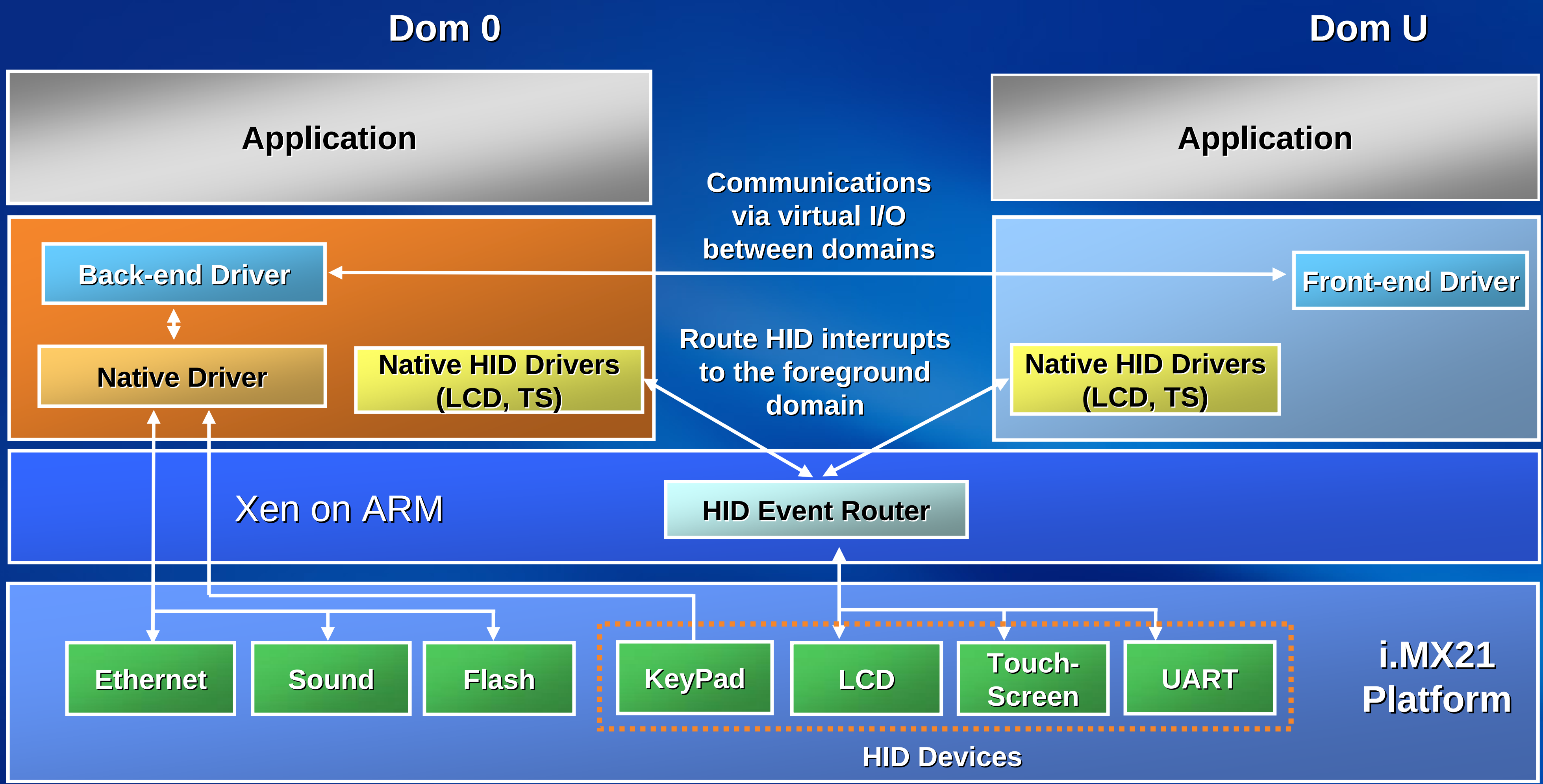
Memory Virtualization (3/3)

- Keep Xen address translation info from being flushed.
  - After page table changes (domain/process switching), TLB entries are flushed explicitly.
  - TLB lockdown mechanism provided by processor can be used to avoid TLB flushing and reloading
  - Two lockdown TLB entries used for Xen pages
    - ARM926 provides 8 lockdown TLB entries



I/O Virtualization (1/2)

- **Mixed Device Driver Architecture**
  - **Split device drivers and coordinated native device drivers**

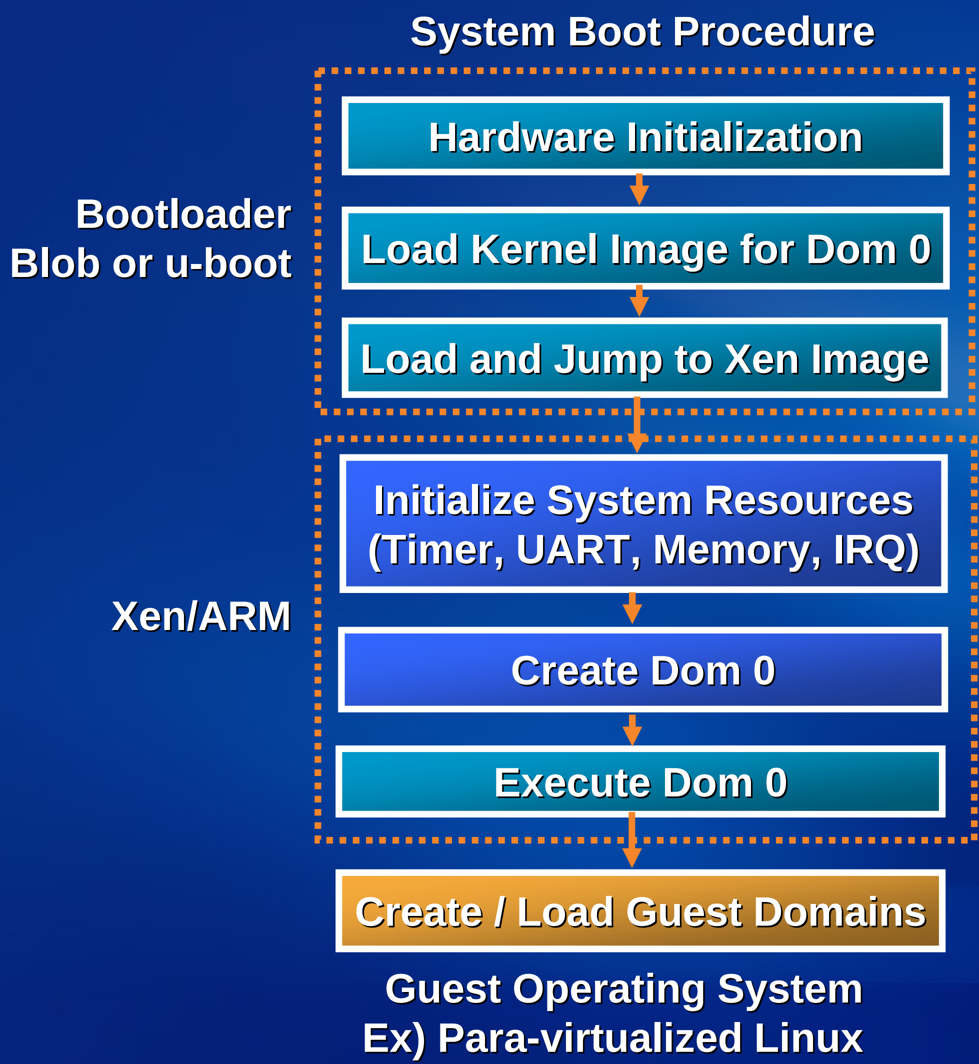


I/O Virtualization (2/2)

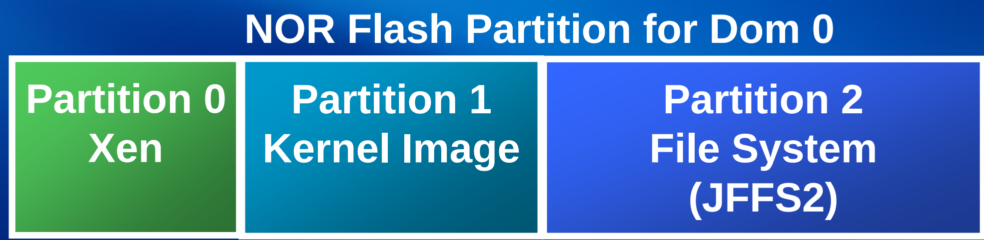
- Mixed device driver architecture for devices shared among guest domains
  - Consists of split device drivers and deterministically coordinated native device drivers
    - Split device driver model
      - Xen-compliant device driver architecture
        - E.g.: Network device, storage device, keypad device
    - Coordinated native device driver model
      - Foreground domain gets exclusive access rights to coordinated native devices
        - Coordinated native device drivers installed in each guest OS domain
        - One button in keypad is reserved to change between domains.
        - E.g.: Human Interaction Device (HID: LCD, touch screen) and UART

System Boot Procedure

- Xen and dom 0 kernel images are loaded at predefined memory location.

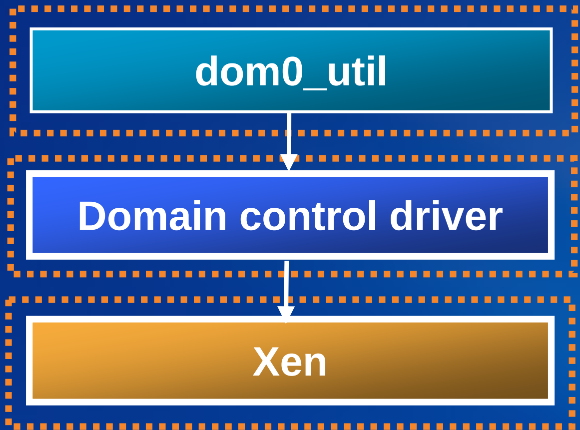


Platform	Load Address	
	Xen	Dom 0
I.MX21	0xC0008000	0xC1C00000



VM Create / Destroy

- Guest domains (dom U) are created and destroyed by a user level application, dom0_util.
  - Dom0_util supports only create and destroy functions.



Control guest domain

Request Xen to create and execute / destroy dom U kernel, where this driver loads the kernel image.

Create and execute dom U / destroy dom U

- Dom U kernel uses NAND flash memory as storage.

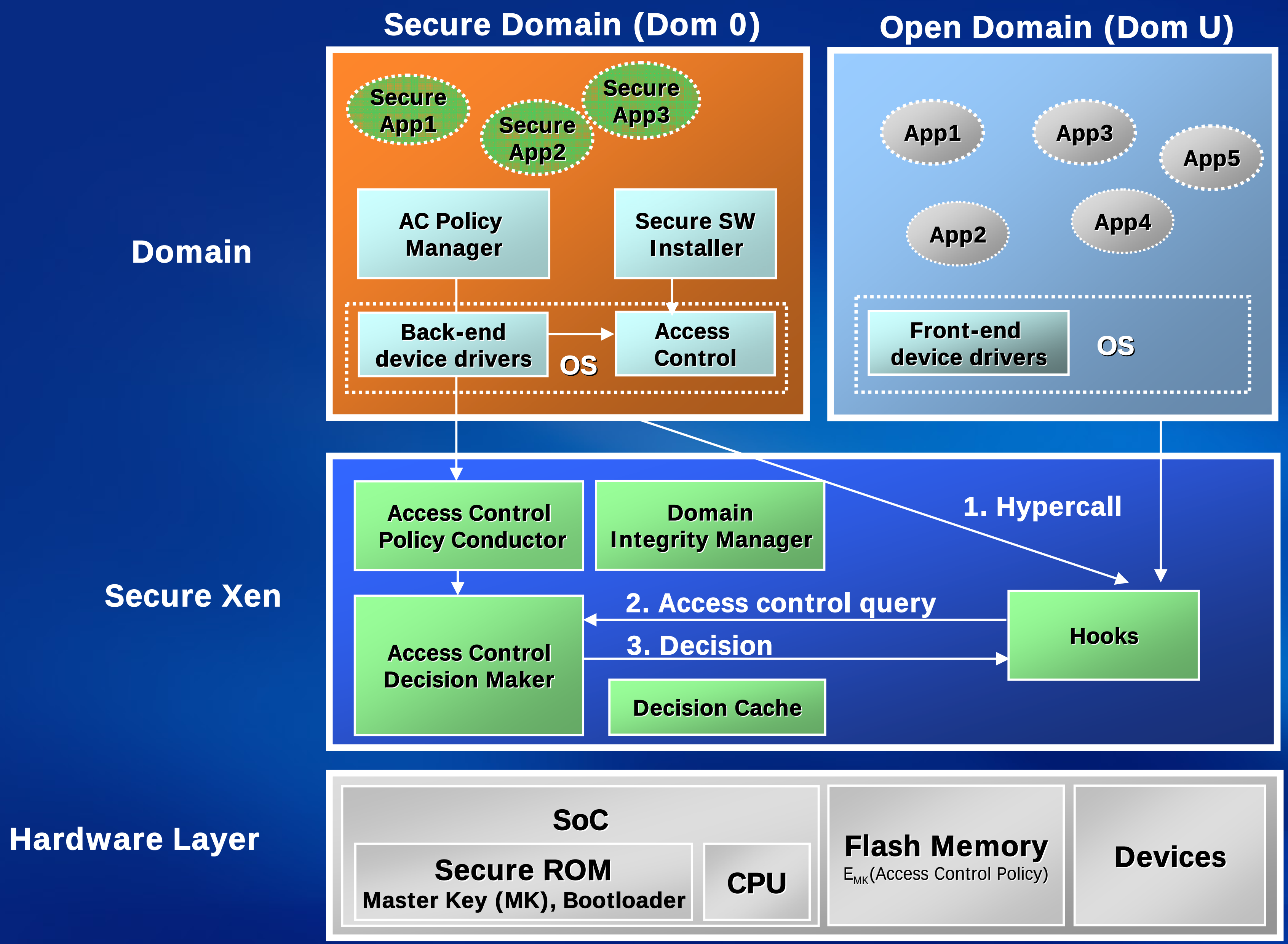
NAND Flash Partition for Dom 1

Platform	Load Address
I.MX21	0xc3c00000

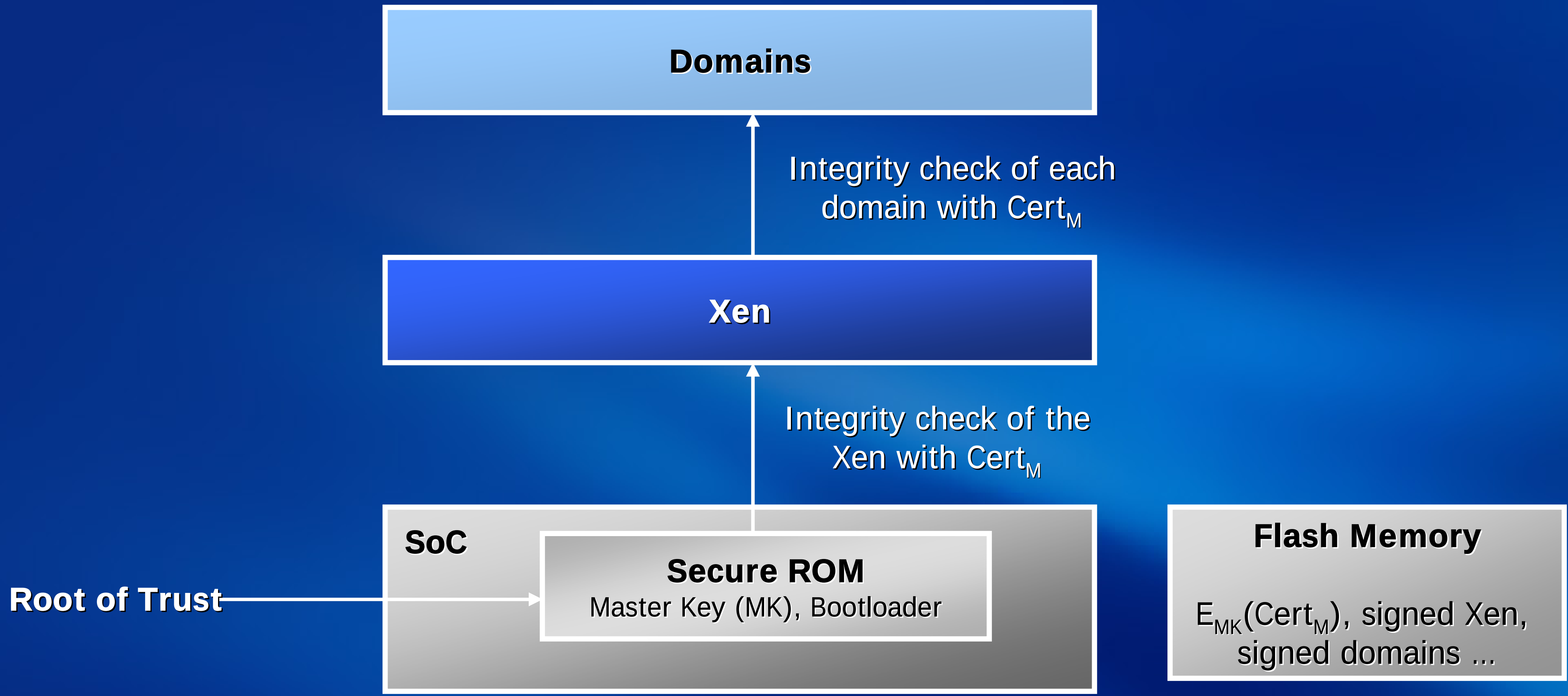
Agenda

- Requirements for Beyond 3G Mobile Device
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix

Security Architecture



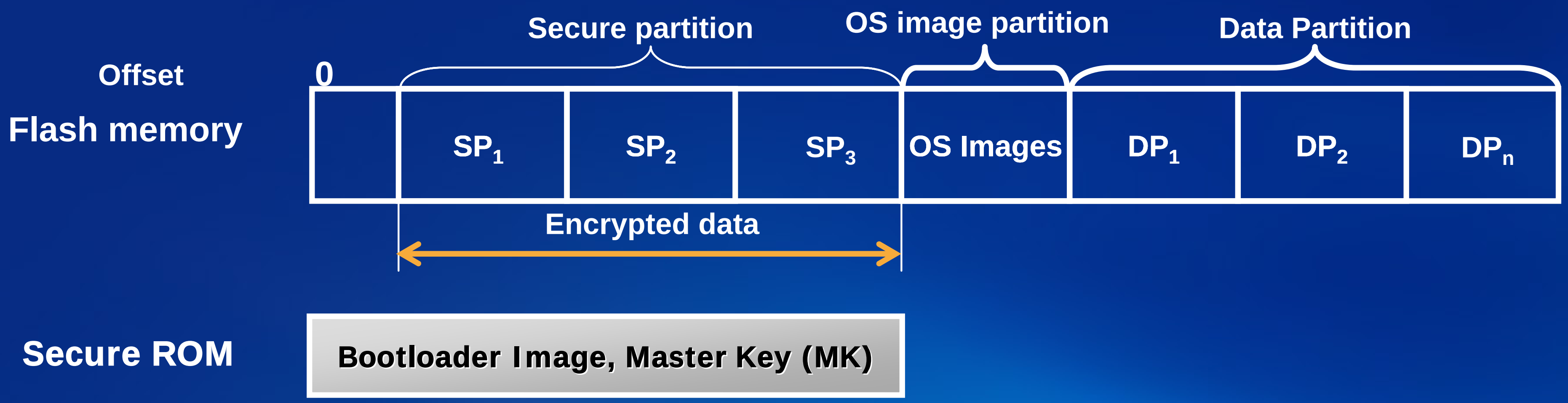
Secure Boot



E_{MK}: Encryption with the master key (MK)

Cert$_M$: Manufacturer's public key certificate

Secure Storage



Symbols	Descriptions
MK	Master key. Each mobile device has a unique MK to encrypt data stored in secure partitions (SPs).
$Cert_M$	Manufacturer's public key certificate. It is used for integrity measurement of Xen or kernel images.
SP_1	A secure partition for Xen image and data for integrity measurement during a system boot. *E_{MK}(Xen Image||Sig_M(H(Xen Image))||Sig_M(H(Secure Domain Image))||Sig_M(H(Normal Domain Image))||$Cert_M$)*
SP_2	A secure partition for access control policies. *E_{MK}(Access Control Policies)*
SP_3	A secure partition for cryptographic keys which are used by secure domain. *E_{MK}(Cryptographic keys)*
DP_n	Partitions for guest OS domains. Each OS is allowed to access its own partition.

Access Control (1/2)

- Flexible architecture based on Flask
- Objects for access control
  - Physical resources
    - Memory, CPU, IO space, IRQ, DMA
  - Virtual resources
    - Event channel, grant table
  - Domain management
    - Creation and destroy of guest domains
- Multi-layered access control not to degrade Xen performance

Access Control (2/2)

- Use case
  - Resources which are used badly due to DoS attacks are controlled by access control module (ACM) using our proprietary policy
    - Resources: CPU, memory, DMA, the number of event channel, battery
    - E.g.:
      - ACM can control CPU time allocated to a guest domain in order to keep malware on this domain from using CPU excessively
      - If battery stock is less than a threshold, ACM shuts a guest domain down

Implementation: Status (1/2)

- Access control
  - 35 access control hooks in hypercalls used for access to physical resources or virtual resources, and domain management
  - Type Enforcement (TE) policy and proprietary policy to protect a mobile device from DoS attacks
  - Performance
    - About 20 micro sec. per access control hook
- Secure boot
  - Integrity measurement of a Xen and two domains
  - Performance
    - About 75 ms for the integrity measurement (digital signature verification) during a system boot

Implementation: Status (2/2)

- Secure storage
  - Secure partitioning applied to NAND/NOR flash memory
  - Secure ROM simulated by using NOR flash memory

Agenda

- Requirements for Beyond 3G Mobile Devices
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix

Conclusions (1/2)
Xen on ARM for Mobile Devices

- Requires
  - Virtualized three CPU modes
    - Modes: Xen, kernel and user process
  - Protection of virtual address spaces for Xen, kernel and user process through domain access control
- Mixed device driver architecture for shared devices works well
  - Split device drivers and deterministically coordinated native device drivers

Conclusions (2/2) Xen Security for Mobile Devices

SAMSUNG

- Requires
  - Integrity measurement of core components
  - Multi-layered access control
    - Access control at Xen layer
      - Physical/virtual resources and domain management are enforced by ACM at Xen
    - Access control at domain layer
      - In order not to degrade Xen performance, detailed access control of the resources in each domain is individually enforced by ACM at each domain

Future Work

- Virtualization of DMA
- Merging Xenstore
- Dynamic memory allocation to guest domains
- Secure download protocol
- Study on separation of a device driver domain from guest OS kernel
- Performance analysis and optimization

Prototype Demo: Video

- HW: a smart phone development platform
  - CPU: ARM9, 266 MHz
  - System memory: 64 MB
  - HID: 3.5 inch LCD, touch screen, keypad
  - Storage: NAND/NOR flash memory
  - Network: Ethernet
- SW
  - VMM: secure Xen on ARM
  - OS: para-virtualized ARM Linux 2.6.11
  - GUI: Qtopia
- Contents: booting secure Xen and dom 0 (Linux), creating/destroying dom U (Linux), and etc.

References

- [COK06] G. Coker, "Xen Security Modules (XSM)," Xen Summit, 2006.
- [GOS06] A. Gostev, "Mobile Malware Evolution: An Overview, Part 1," 2006. http://www.viruslist.com/en/analysis?pubid=200119916
- [KAW05] D. Kawamoto, "2006: Year of the mobile malware," 2005. http://news.com.com/2006+Year+of+the+mobile+malware/2100-7349_3-6001651.html
- [SAI05] R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. van Doorn, J. L. Griffin, and S. Berger. "sHype:A secure hypervisor approach to trusted virtualized systems," IBM Research Report, 2005.
- [ARM01] Andres N.Sloss, Dominic Symes, C.Wright. "ARM System Developer's Guide", Morgan Kaufmann, 2004
- [KEV01] Kevin Lawton, "Running multiple operating systems concurrently on an IA32 PC using virtualization techniques". 2000.

Agenda

- Requirements for Beyond 3G Mobile Device
- Goal and Approach
- Xen on ARM
  - Xen on ARM Architecture
  - System Virtualization
  - System Boot Operation
- Security
  - Security Architecture and Its Components
  - Implementation: Status
- Conclusions and Future Work
- Appendix

Comparison: Xen

Feature	Xen/x86	Xen/ARM
Booting guest domain U	XM	Lightweight version of XM
Memory allocation to domain	Dynamic	Static
Virtual Device Interface / Device Configuration	Xenbus / Xenstore	Modified Xenbus* / Proprietary (Xenstore to be implemented)
Console I/O	Xenconsole daemon and xenconsole client	Deterministically coordinated HID Device Driver
Virtual Block Device Support	IDE, SCSI HDD	NAND, NOR flash

Based on current status

* Modified Xenbus to support virtual I/O setup without xenstore

Comparison: CPU

Feature	x86	ARM v4/v5
# of Privilege levels	4	2
Software Interrupt Handling	Direct execution	Indirect execution through VMM
# of sensitive instructions	Approx. 57 [KEV01]	18 [ARM01] (in case of ARM v5)
Cache Model	PIPT – No cache alias	VIVT – Cache Alias

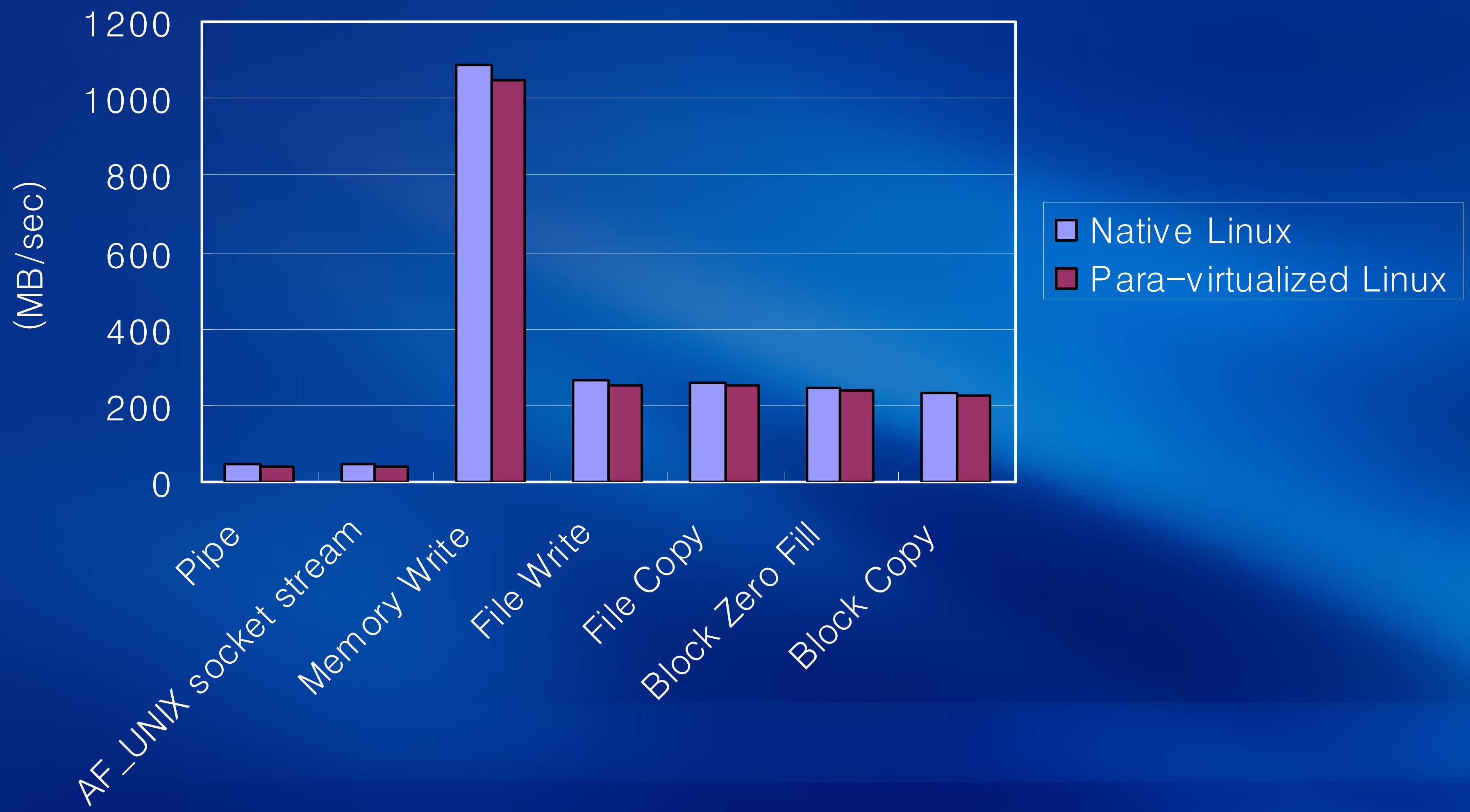
Comparison: Access Control

- sHype, XSM, and Our ACM

	sHype [SAI05]	XSM [COK06]	Our ACM
Access Control Policies	Flexible based on Flask (TE and Chinese Wall)	Flexible based on Flask (TE, Chinese Wall, RBAC, MLS, and MCS)	Flexible based on Flask (TE and proprietary policy)
Objects of access control	Virtual resources and domain management	Physical/virtual resources and domain management	Physical/virtual resources and domain management
Protection against mobile malware-based DoS attacks	N/A	N/A	Memory, battery, DMA, and event channels are controlled by ACM
Access control to objects in each guest domain	Enforced by ACM at VMM	Enforced by ACM at VMM	Enforced by ACM at each domain (for performance reason)
Etc			Xen on ARM specific hooks

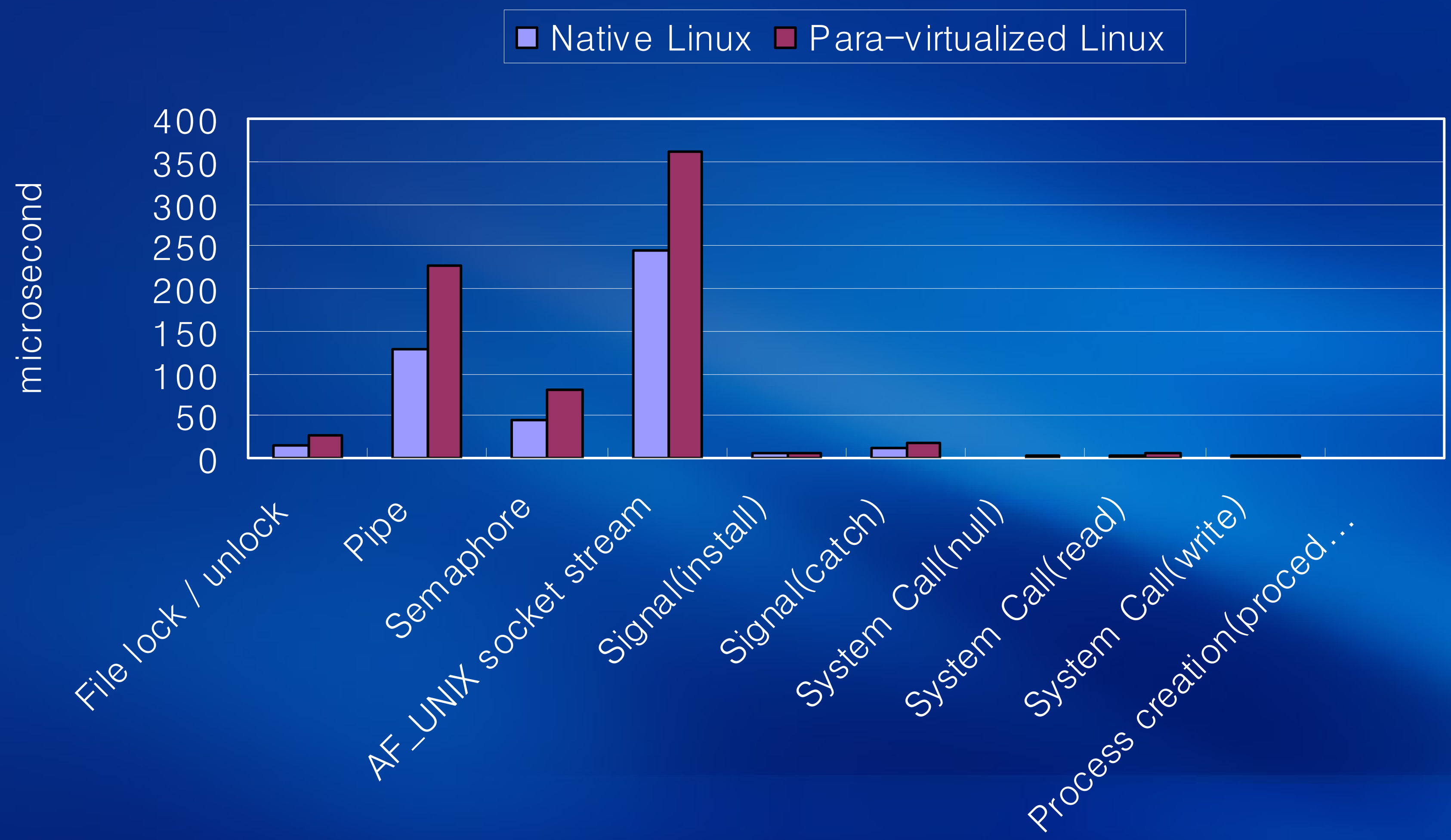
Performance (1/2)

- **Bandwidth Test (LMBench): Snapshot**



Performance (2/2)

- **Latency Test (LMBench): Snapshot**



Xen Tools

- **Xen Tools**
  - **Python packages are too big for small flash memory.**
  - **Smaller size by removing unused Python modules.**

	Full Python	Embedded Python
Total size	40MB	5.7MB
# of modules	280	40

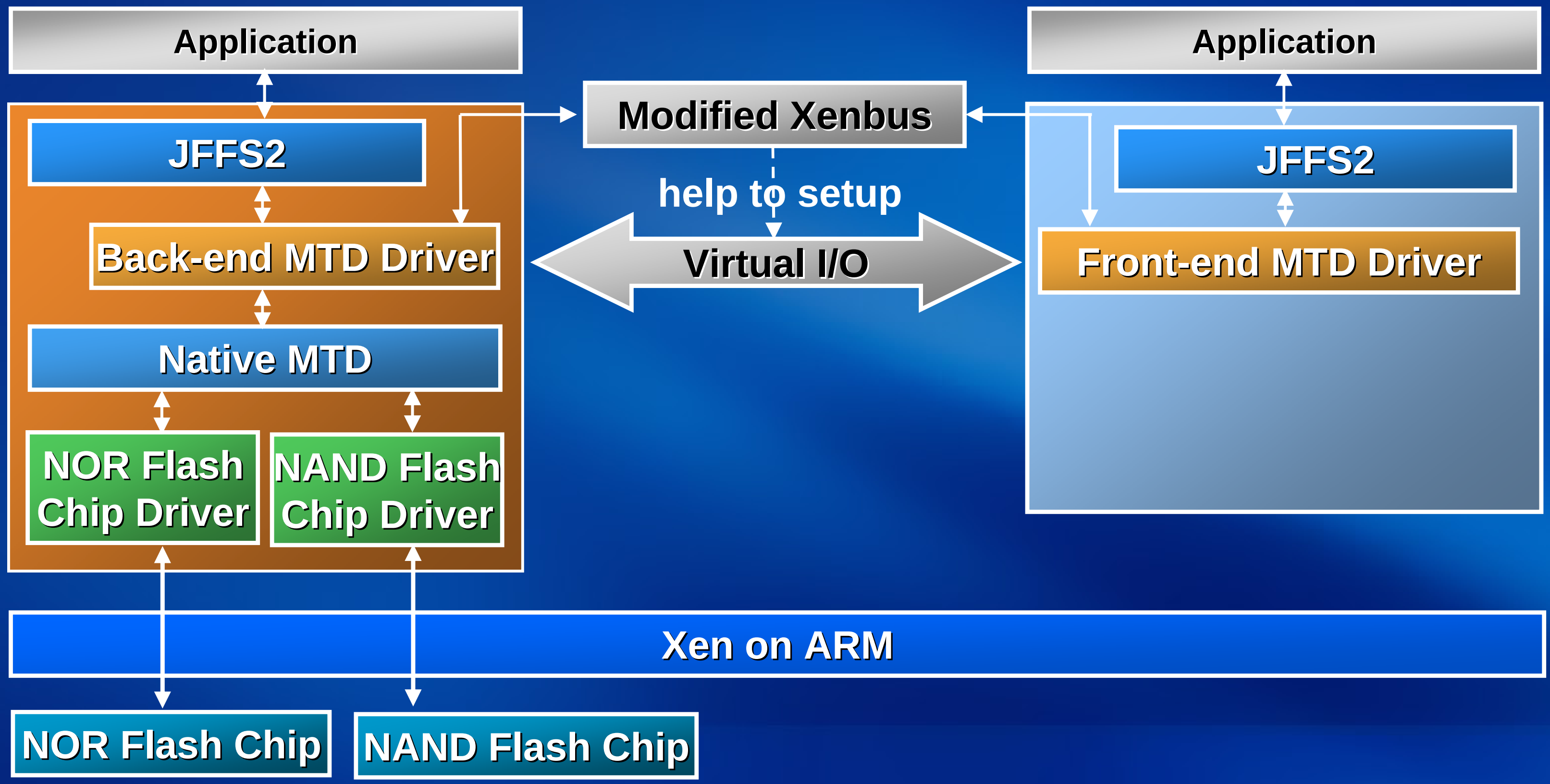
Python version : 2.4.3

I/O Virtualization: Xenbus

- Modified Xenbus
  - Modified to support virtual I/O setup without xenstore.
    - Xenstore porting is in progress.
  - All configuration data is maintained in shared configuration page.
    - E.g. :
      - Event Channel No.
      - Grant Table Ref. No.

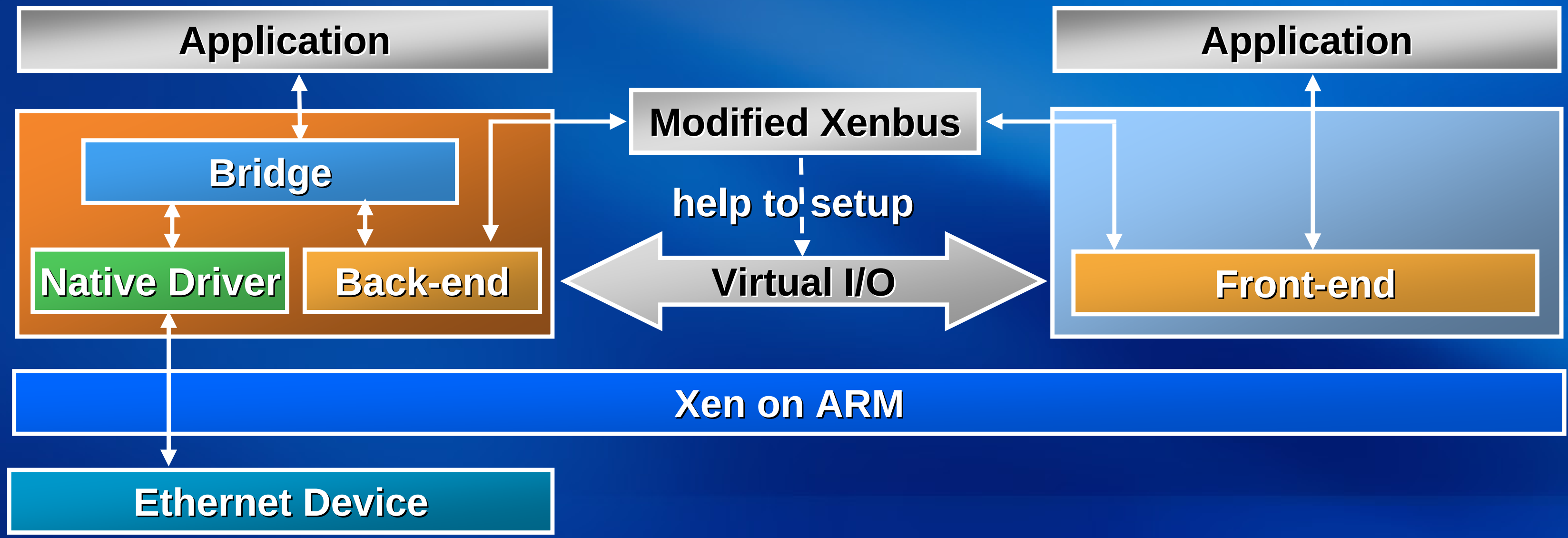
I/O Virtualization: example

- Virtual Memory Technology Device (MTD) Driver
  - To share flash memory between guest domains



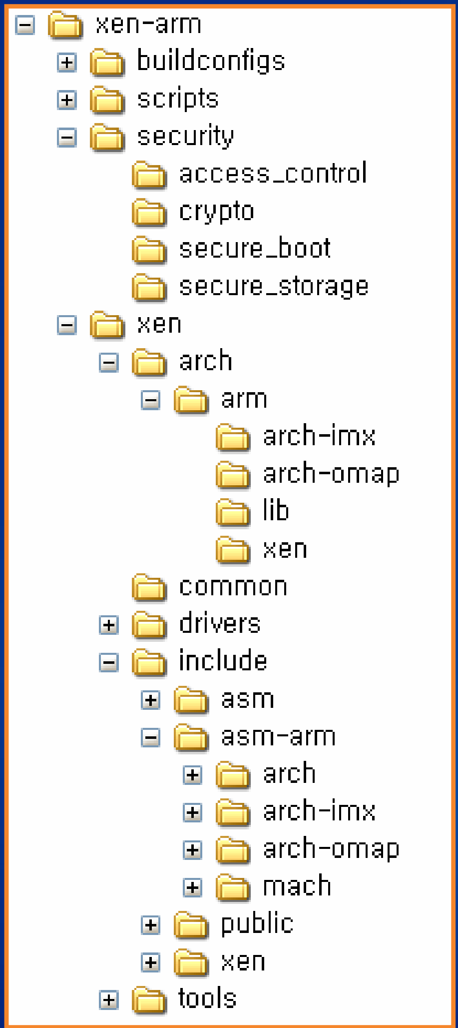
I/O Virtualization: example

- Virtual Network Driver
  - Use synchronous I/O buffer instead of asynchronous I/O ring.
  - Transmit and receive data via shared pages



Current Source Code Status (1/2)

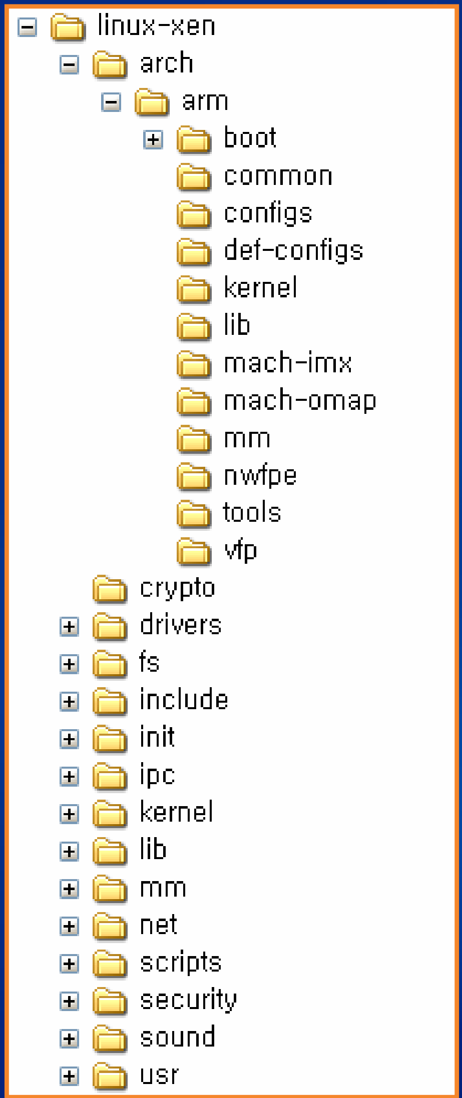
- **Xen/ARM (3.0.2)**



Directory	LOC
security/access_control	2500
security/crypto	793
security/secure_boot	1500
security/secure_storage	720
arch/arm/xen	7455
arch/arm/arch-imx	1031
arch/arm/arch-omap	1127
arch/arm/lib	2695
include/asm-arm	4953
Include/asm-arm/arch-imx	2110
Include/asm-arm/arch-omap	4030

Current Source Code Status (2/2)

- **Para-virtualized Linux Kernel (2.6.11)**



Directory	LOC
arch/arm/kernel	1134
arch/arm/mm	1730
arch/arm/mach-imx	1008
Include/asm-arm	646