XVISOR

Open-source, Lightweight, Extensible Hypervisor

ANUP PATEL <ANUP@BRAINFAULT.ORG>

Little About Me

- Hypervisor and Linux kernel developer with 12+ years of industry experience
- Post-graduated (Masters) in 2009 from IIT Bombay, India
- Work full-time for Qualcomm as Server virtualization expert
- Maintain Xvisor as hobby project in personal time (since 2010)
- Open source contributions:
  - 3300+ patches in Xvisor (http://xhypervisor.org/)
  - 100+ patches in Linux ARM/ARM64/RISC-V (https://www.kernel.org/)
  - 24+ patches in Linux KVM ARM64 (https://www.kernel.org/)
  - 16+ patches in Atomthreads RTOS (https://atomthreads.com/)
  - Few patches in Xen ARM, QEMU, KVMTOOL, etc

Agenda

- **Overview**
- **Virtualization Infrastructure**
- **Domain Isolation**
- **Device Virtualization**
- **Domain Messaging**
- **Footprint**
- **Xvisor for Automotive**
- **References**

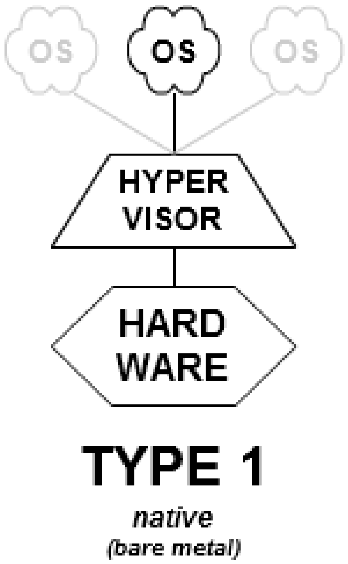
NOTE: Domains in Automotive world are referred to as **Guests** in Xvisor

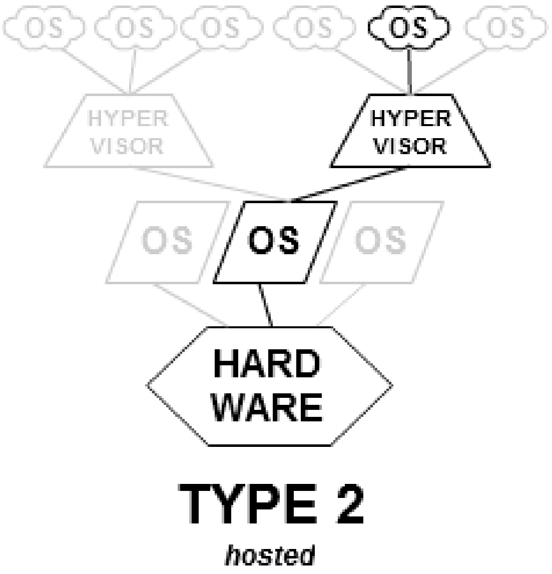
Overview

What is Xvisor?

- **XVISOR** = e**X**tensible **V**ersatile hyperv**ISOR**
- Xvisor is an open-source GPLv2 Type-1 monolithic (i.e. Pure Type-1) hypervisor
- Community driven open source project (http://xhypervisor.org, xvisor-devel@googlegroups.com)
- 8+ years of development and hardening (since 2010)
- Supports variety of architectures: ARMv5, ARMv6, ARMv7, ARMv7ve, ARMv8, x86_64, and RISC-V (work-in-progress)
- First paper in IEEE PDP 2015 titled "***Embedded Hypervisor Xvisor: A comparative analysis***"

Hypervisor Classification - Traditional

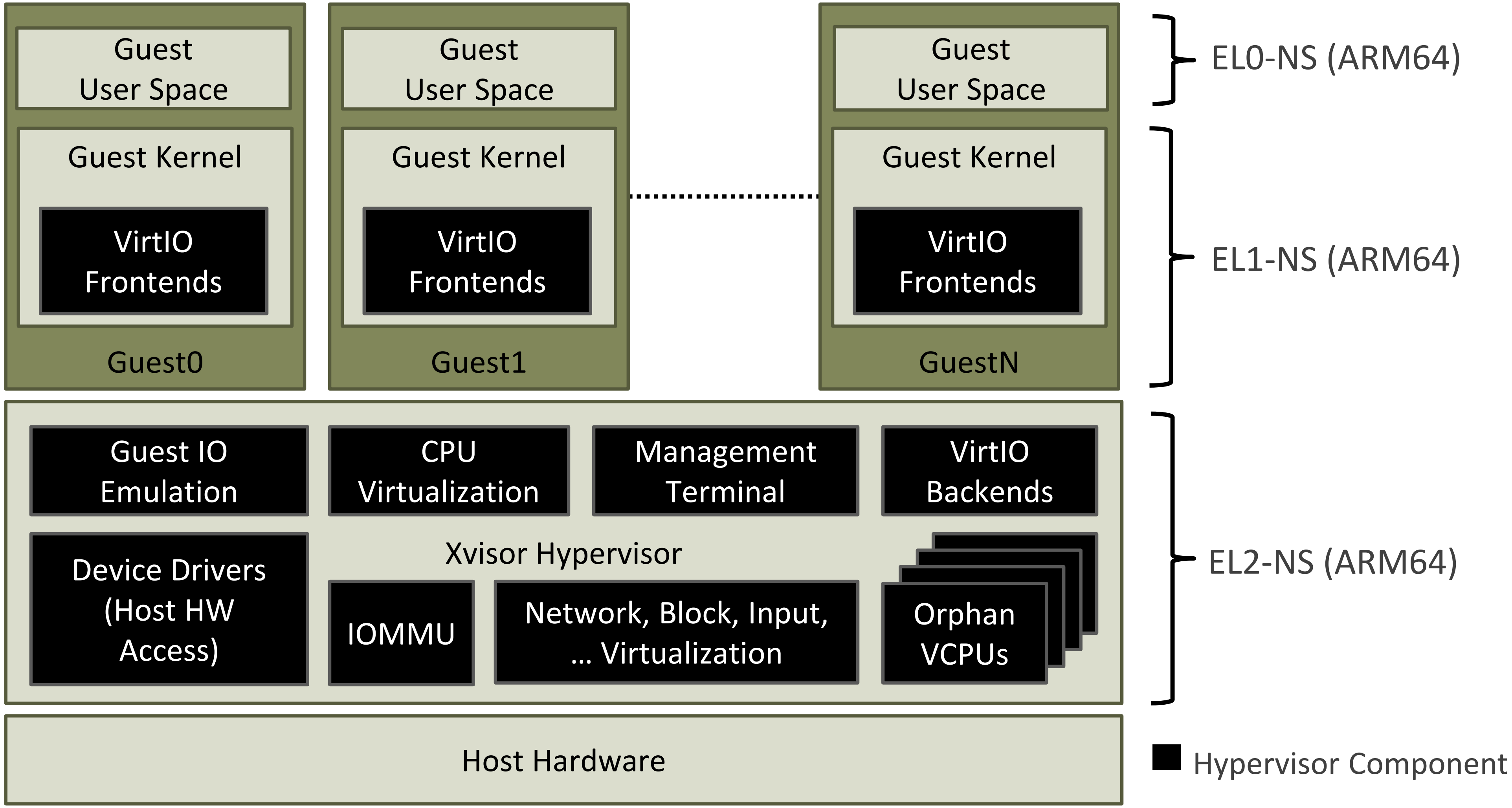




Type1 Examples: Xvisor, Xen, VMWare ESX Server, Microsoft HyperV, OKL4 Microvisor, etc

Type2 Examples: Linux KVM, FreeBSD Bhyve, VMWare Workstation, Oracle VirtualBox, etc

Xvisor - Complete Monolithic - Type1



Lots of features

- **Virtualization Infrastructure:**
  - **Device tree based configuration**
  - **Soft real-time pluggable scheduler**
  - **Hugepages for Guest and Host**
  - Tickless and high-resolution timekeeping
  - Host device driver framework
  - Threading framework
  - Runtime loadable modules
  - Management terminal
  - Light-weight filesystem
  - White-box testing
  - … Many More …

Lots of features (Contd.)

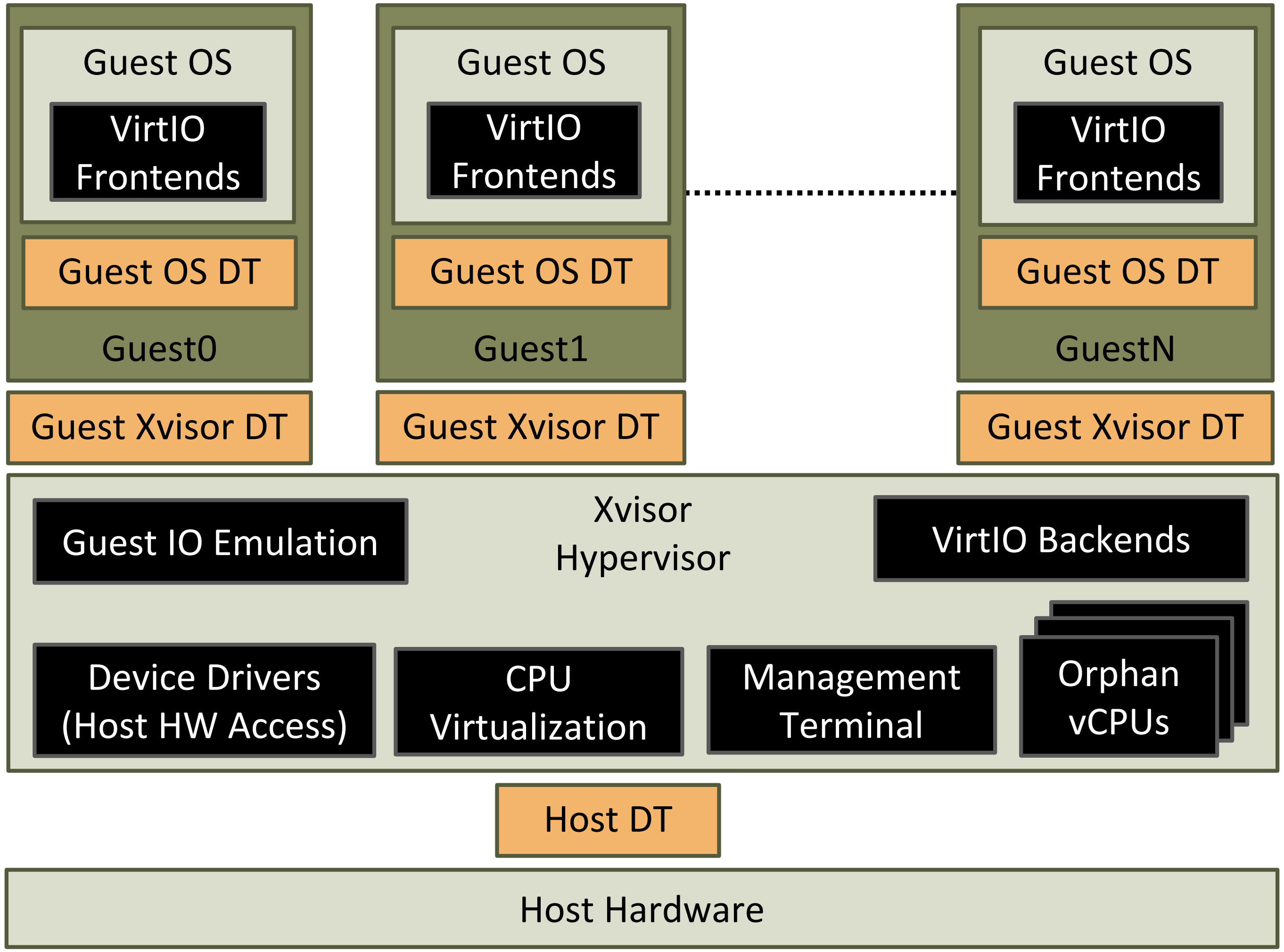
- **Domain Isolation:**
  - **VCPU and Host Interrupt Affinity**
  - **Spatial and Temporal Memory Isolation**
- **Device Virtualization:**
  - **Pass-through device support**
  - **Block device virtualization**
  - **Network device virtualization**
  - Input device virtualization
  - Display device virtualization
  - VirtIO v0.9.5 for Para-virtualization
- **Domain Messaging:**
  - **Sharing On-chip Coprocessor**
  - **Zero-copy Inter-Guest Communication**

Virtualization Infrastructure

Device Tree Based Configuration

Three types of device tree (DT):

1. **Host DT:**
   - Device tree which describes underlying host HW to Xvisor
   - Used by Xvisor at boot-time
2. **Guest Xvisor DT:**
   - Device tree which describes Guest virtual HW to Xvisor
   - Used by Xvisor to create Guest
3. **Guest OS DT:**
   - Device tree which describes Guest virtual HW to Guest OS
   - Used by Guest OS at boot-time



Soft Real-time Pluggable Scheduler

- Scheduling entity is a VCPU
- Two types of VCPUs:
  1. **Normal VCPU:** A VCPU belonging to Guest/VM
  2. **Orphan VCPU:** A VCPU belonging to Hypervisor for background processing
- Orphan VCPUs are very light-weight compared to Normal VCPUs
- Scheduler supports **pluggable scheduling policy**, available policies:
  - Fixed priority round-robin
  - Fixed priority rate monotonic
- Scheduling policies are soft real-time
- Scheduler supports multi-processors (or SMP Host)

Hugepages for Guest and Host

- Xvisor uses Stage1 (regular) page table for “Hypervisor virtual address” to “Host physical address” mappings
- Host hugepages are bigger mappings in Stage1 (regular) page table
- **Host hugepages make Xvisor memory accesses faster**
- Xvisor uses Stage2 (nested) page table for “Guest physical address” to “Host physical address” mappings
- Guest hugepages are bigger mappings in Stage2 (nested) page table
- **Guest hugepages make Guest OS memory accesses faster**
- For ARM64 and x86_64, hugepage sizes are 2M and 1G

Domain Isolation

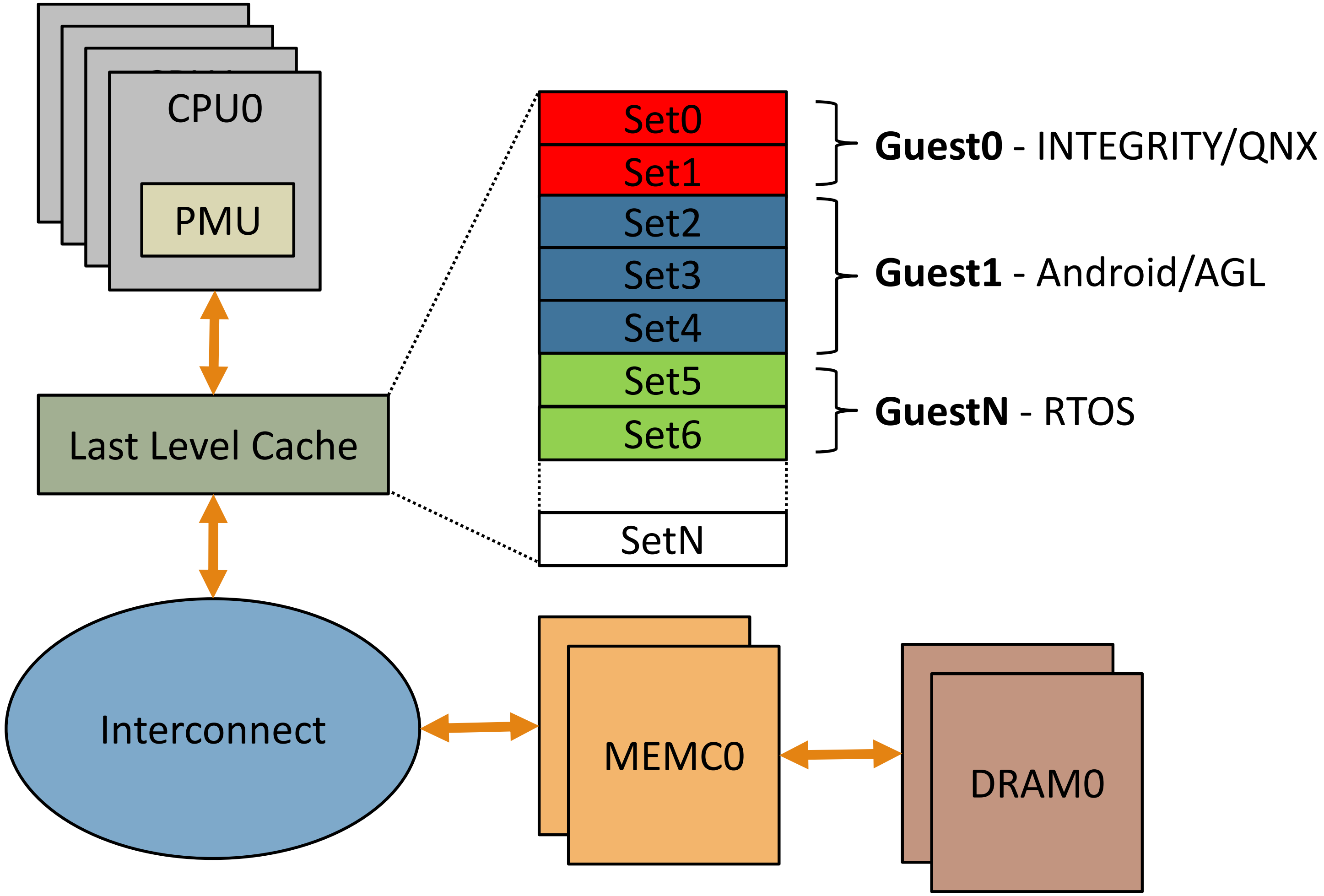
VCPU and Host Interrupt Affinity

- **VCPU affinity is an attribute of VCPU** specifying Host CPUs on which it is allowed to run
- Using VCPU affinity, we can assign particular Host CPUs for:
  - Guest VCPUs (Normal VCPUs)
  - Xvisor background threads (Orphan VCPUs)
- **Host interrupt affinity is an attribute of Host interrupt** specifying Host CPUs on which it can be processed
- Using Host interrupt affinity, we can assign particular Host CPUs for Host interrupts of a Guest pass-through device
- Host interrupt affinity of per-CPU Host interrupts (such as IPIs) cannot be changed

Spatial and Temporal Memory Isolation

- **Spatial memory isolation** achieved using cache-coloring on last level cache for Guest RAM
- **Temporal memory isolation** achieved using CPU performance monitoring unit (PMU) to control memory access rate by Guest

IEEE ICIT 2018 paper: "Supporting Temporal and Spatial Isolation in a Hypervisor for ARM Multicore Platforms"



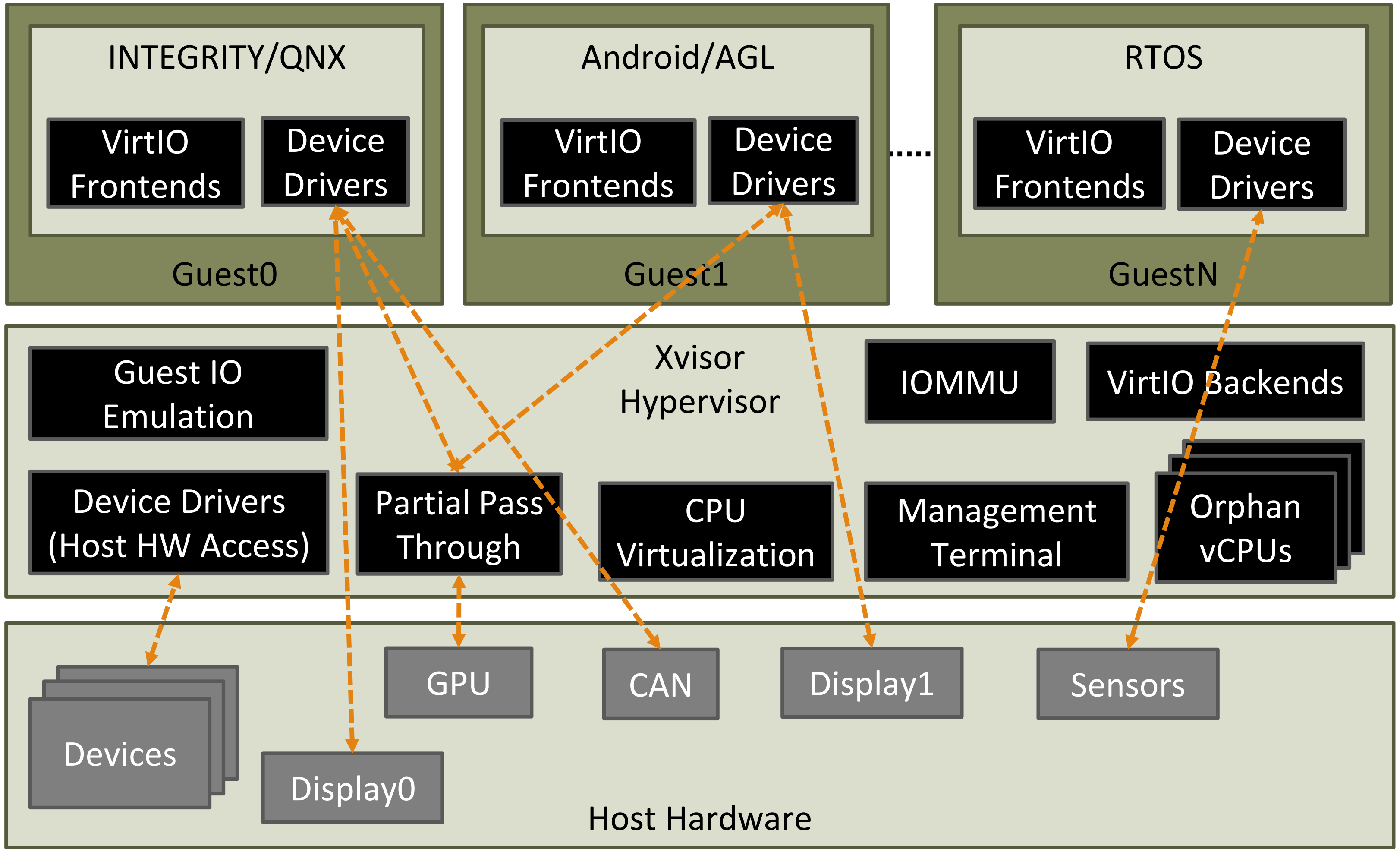
Device Virtualization

Device Virtualization Types

- Types of Virtual Devices:
  1. **Software Emulated Device:** Real-world device emulated by hypervisor. **Examples,** Emulated UART 8250, etc.
  2. **Paravirtual Device:** Pseudo-devices emulated by hypervisor which are designed to minimize register programming. **Examples,** VirtIO Net, VirtIO Block, VirtIO Console etc.
  3. **Pass-through Device:** Direct access of host device from Guest/VM. This requires IOMMU support in Host. **Examples,** PCI e1000 network adapter accessed by Guest/VM, SATA AHCI controller accessed by Guest/VM, etc.
  4. **Partial Pass-through Devices:** Access part of a host device from Guest/VM. This requires IOMMU support in Host and Host device should have special support. **Examples,** SRIOV based PCI Network Adapter, GPU with multiple channels, etc.
- **All types of virtual devices supported by Xvisor**

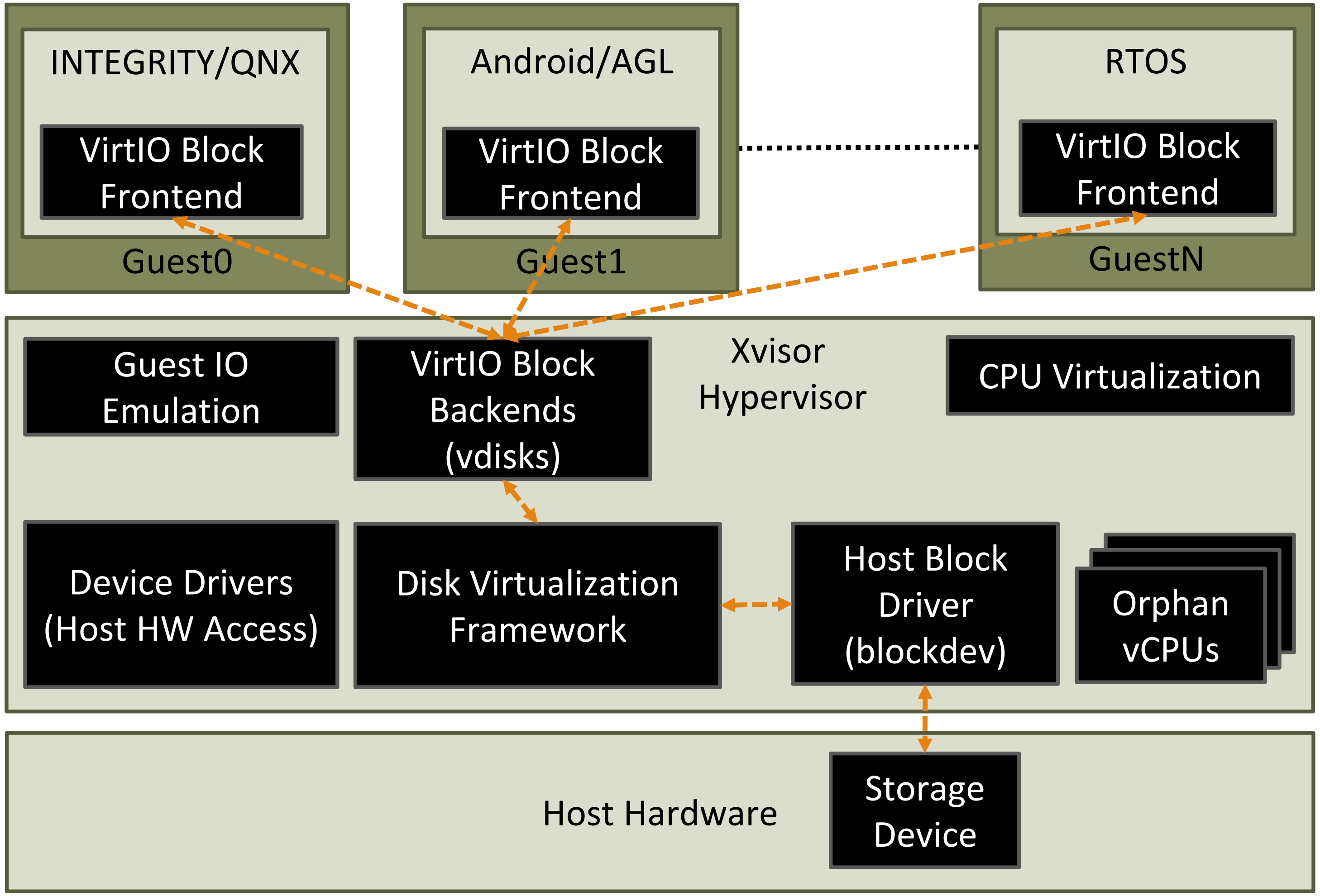
Pass-through Device Support

- Linux compatibility headers **for drivers running in Xvisor**
- IOMMU and Interrupt controller virtualization for **drivers running in Guest OS**
- Access part of device from Guest OS using **partial pass-through:**
  - Custom driver
  - Custom emulator



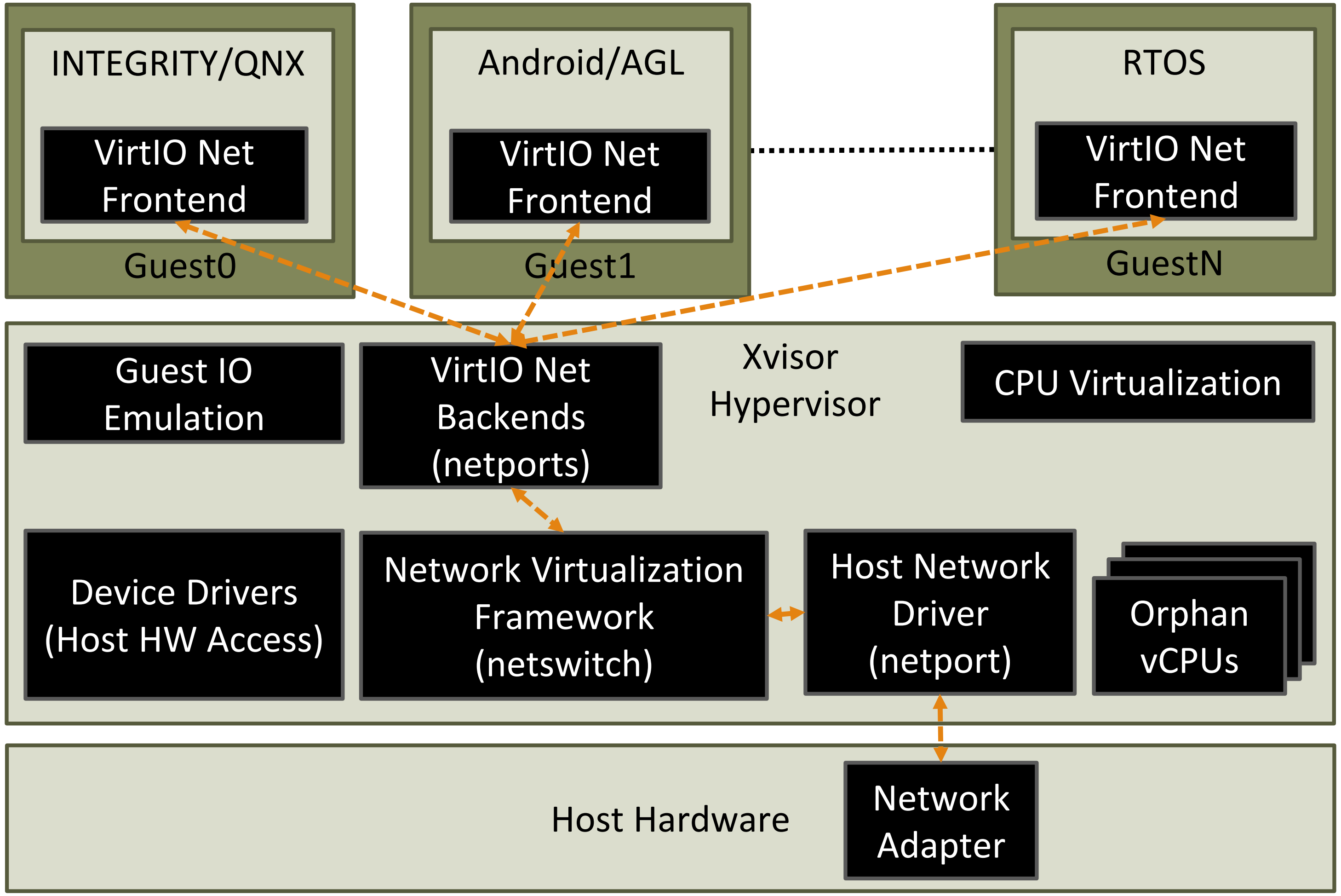
Block Device Virtualization

- Consist of:
  1. **vdisk:** Logical entity which gets block read/write requests from Guests. Examples, Storage device emulators, and VirtIO Block backends.
  2. **blockdev:** Logical entity which represents host storage device or a partition on host storage device. Examples, MMC, NAND, SATA, etc.
- **We can attach a blockdev to a vdisk**



Network Device Virtualization

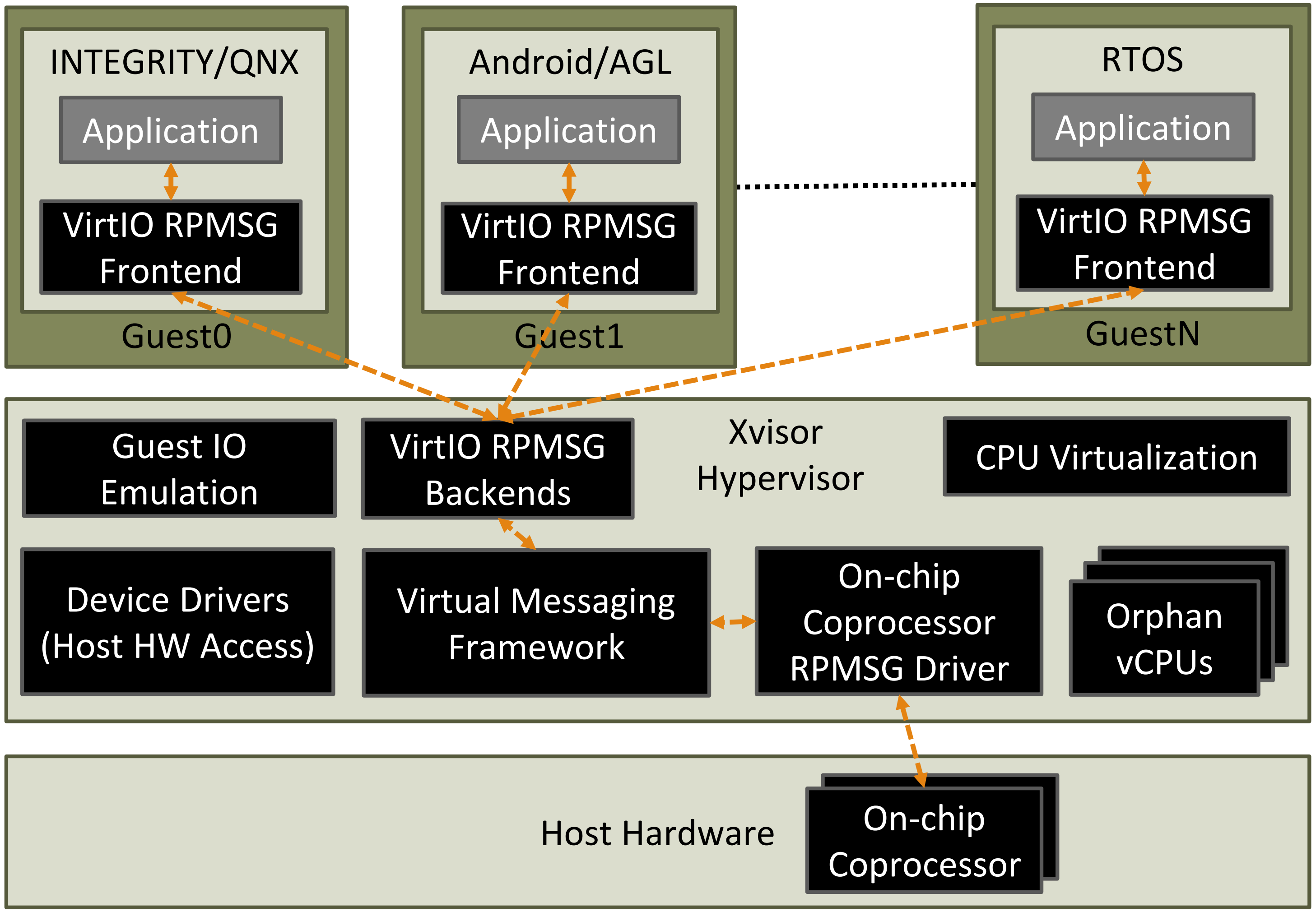
- Consist of:
  1. **netport:** Logical entity capable of consuming and generating packets. Examples host network drivers, NIC emulators, and VirtIO Net backends.
  2. **netswitch:** Logical entity which does packet routing between netports. Various routing policy available: hub, bridge, vlan, etc.



Domain Messaging

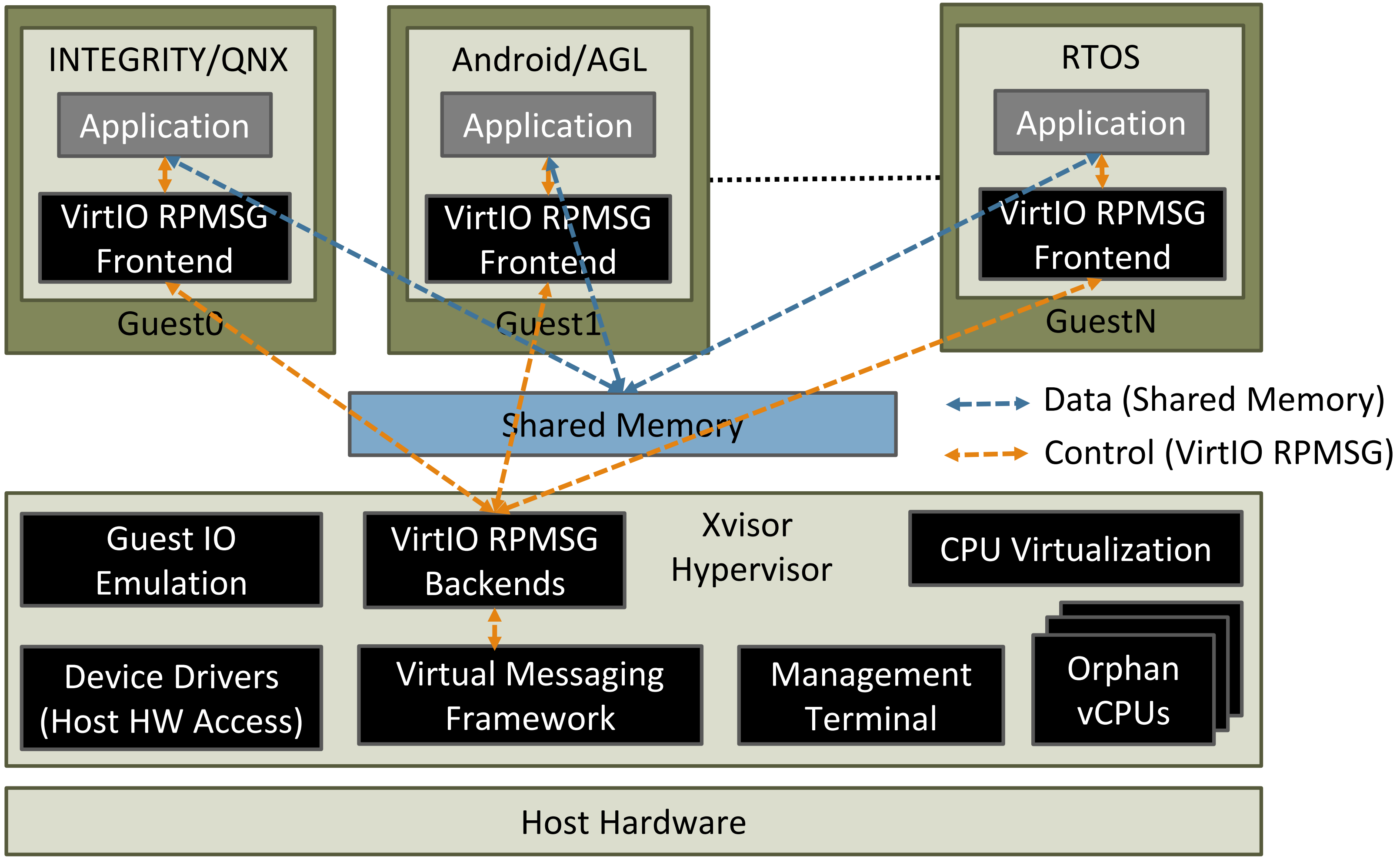
Sharing On-chip Coprocessor

- SOCs can have on-chip coprocessors for secured processing
- Linux applications can communicate with on-chip coprocessor using **RPMSG character device**
- **Virtual messaging domain** to define a set Guests allowed to communicate with on-chip coprocessor



Zero-copy Inter-Guest Communication

- Achieved using:
  1. **VirtIO RPMSP:**
     - Used for control messages
     - Name-service notifications
  2. **Shared Memory:**
     - Used for actual data transfers
     - Very fast zero-copy
- Linux applications can communicate across Guests using **RPMSG character device**
- **Virtual messaging domain** to define a set of Guests allowed to communicate



Footprint

Code Size and Memory Footprint

Lines of Code	Comments	Code
arch/arm/	7143	20614
core/	8974	35419
commands/	1025	10145
daemons/	147	526
drivers/	9427*	43922*
emulators/	3217*	24963*
libs/	5933	17445
TOTAL	**35866**	**153034**

***** Can be further decreased or increased based on compile-time configuration

BLOB	Size
.text	969 KB*
.data	129 KB
.rodata	329 KB*
.bss	202 KB
vmm.bin	**1445 KB**

Runtime Memory	Size
Text memory freed at boot-time	112 KB
Typical memory usage	21 MB*
Max VAPOOL limit	32 MB*

NOTE: Stats gathered from Xvisor-next on 22nd September 2018 for ARM64

Xvisor for Automotive

Why Xvisor is ideal for Automotive?

- No dependency on any Guest OS for running management tools
- Single software providing complete virtualization solution
- Guest types described using device tree instead of fixed Guest types
- Para-virtualization complying open-standards (such as VirtIO)
- Pass-through (or direct access) device support
- Zero-copy inter-guest communication
- Spatial and temporal memory isolation between Guests
- Low memory footprint with reasonable code size
- Playground for academic research

On-going Work in Xvisor

- Guest image authentication
- VirtIO input
- VirtIO GPU
- Netport Rx/Tx throttling
- Vdisk IO request rate-limiting
- Fixed priority deadline scheduler
- RISC-V support
- Upgrade to VirtIO 1.0 support
- … And other stuff …

References

References

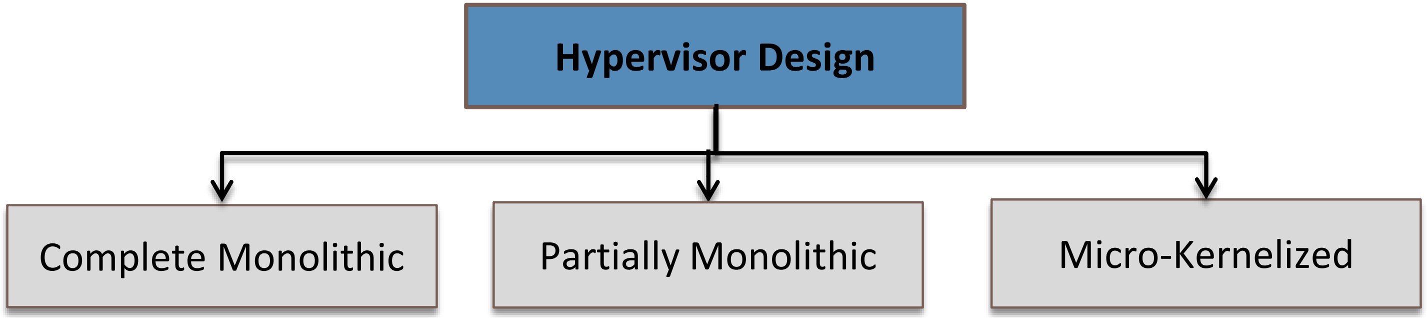
- *Embedded Hypervisor Xvisor: A comparative analysis (IEEE PDP 2015)* (http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=7092793)
- *Xvisor: An open-source, lightweight, embedded hypervisor for your car (FOSDEM 2015)* (https://archive.fosdem.org/2015/schedule/event/car_hypervisor/)
- *Xvisor VirtIO CAN: Fast virtualized CAN (ERTS 2016)* (http://xhypervisor.org/pdf/Xvisor_VirtIO_CAN_Fast_virtualized_CAN.pdf)
- *Supporting Temporal and Spatial Isolation in a Hypervisor for ARM Multicore Platforms (IEEE ICIT 2018)* (https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8342303)
- *Reconciling Security with Virtualization: A Dual-Hypervisor Design for ARM TrustZone (IEEE ICIT 2018)* (https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=8342303)

Thank You !!!

Backup

Hypervisor Classification - New

- Design of a hypervisor can be further classified based on three aspects:
  1. **CPU virtualization**
     - What part of hypervisor virtualize CPU registers and MMU ?
  2. **Host hardware access**
     - What part of hypervisor access host devices (i.e. Host device drivers) ?
  3. **Guest IO emulation**
     - What part of hypervisor virtualize peripherals (i.e. Guest I/O devices) ?

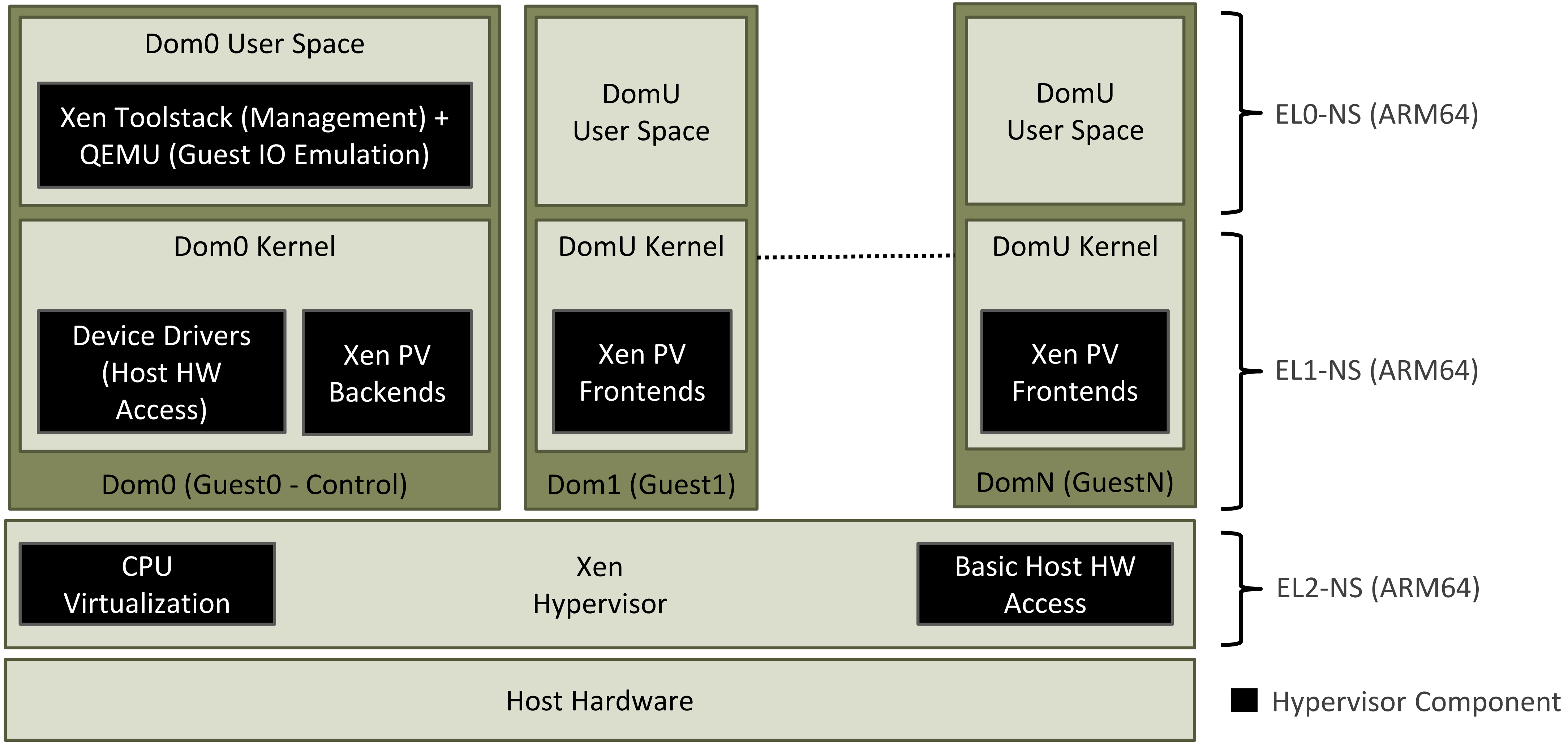


IEEE PDP 2015 paper: "Embedded Hypervisor Xvisor: A comparative analysis"

Hypervisor Classification - New (Contd.)

Complete Monolithic	Partially Monolithic	Micro-Kernelized
Single software layer	Extends an existing OS kernel	Micro-kernel providing virtualization
Main Hypervisor: • Host hardware access • CPU virtualization • Guest IO emulation **Remaining Stuff:** • Optional host hardware access from virtual machine(s)	**Main Hypervisor:** • Host hardware access • CPU virtualization in host OS **Remaining Stuff:** • Optional host hardware access from virtual machine(s) • Guest IO emulation from user-space software	**Main Hypervisor:** • Basic host hardware access • CPU virtualization in hypervisor micro-kernel **Remaining Stuff:** • Complete host hardware access in management virtual machine(s) • Guest IO emulation in management virtual machine(s)
Type-1	**Type-2**	**Type-1**
Examples: Xvisor, VMware ESX server	**Examples: Linux KVM,** FreeBSD Bhyve, VMware Workstation, Oracle VirtualBox	**Examples: Xen,** Microsoft HyperV, OKL4 Microvisor

Xen - Micro-kernelized - Type1



KVM - Partially Monolithic - Type2