Virtual Machine Security

CSE443 - Spring 2012

Introduction to Computer and Network Security

Professor Jaeger

www.cse.psu.edu/~tjaeger/cse443-s12/

Operating System Quandary

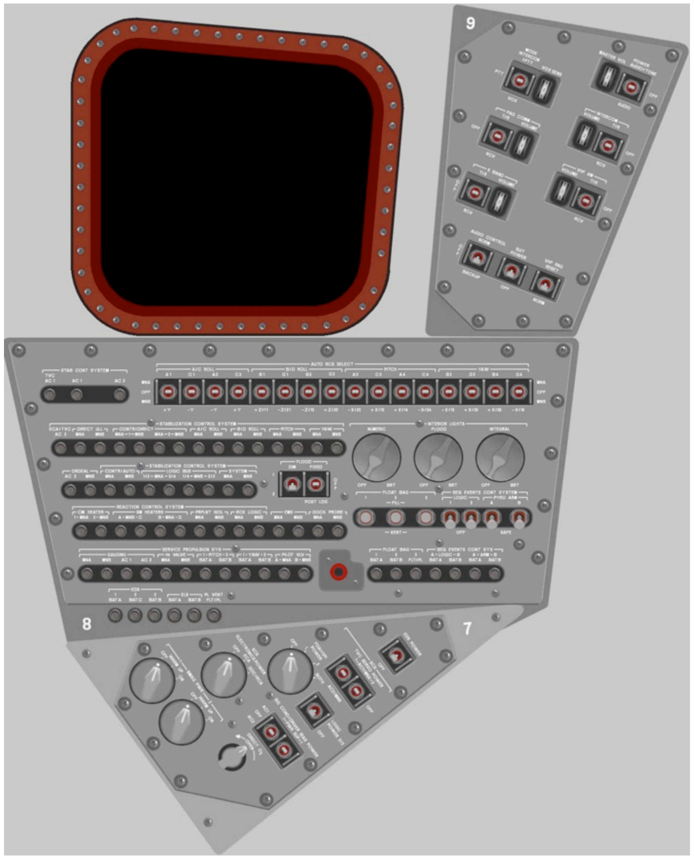
- Q: What is the primary goal of system security?
  - OS enables multiple users/programs to share resources on a physical device
- Q: What happens when we try to enforce Mandatory Access Control policies on UNIX systems
  - Think SELinux policies
- What can we to do to simplify?

Virtual Machines

- Instead of using system software to enable sharing, use system software to enable ***isolation***

- Virtualization
  - “a technique for hiding the physical characteristics of computing resources from the way in which others systems, applications, and end users interact with those resources”

- Virtual Machines
  - Single physical resource can appear as multiple logical resources



Virtual Machine Architectures

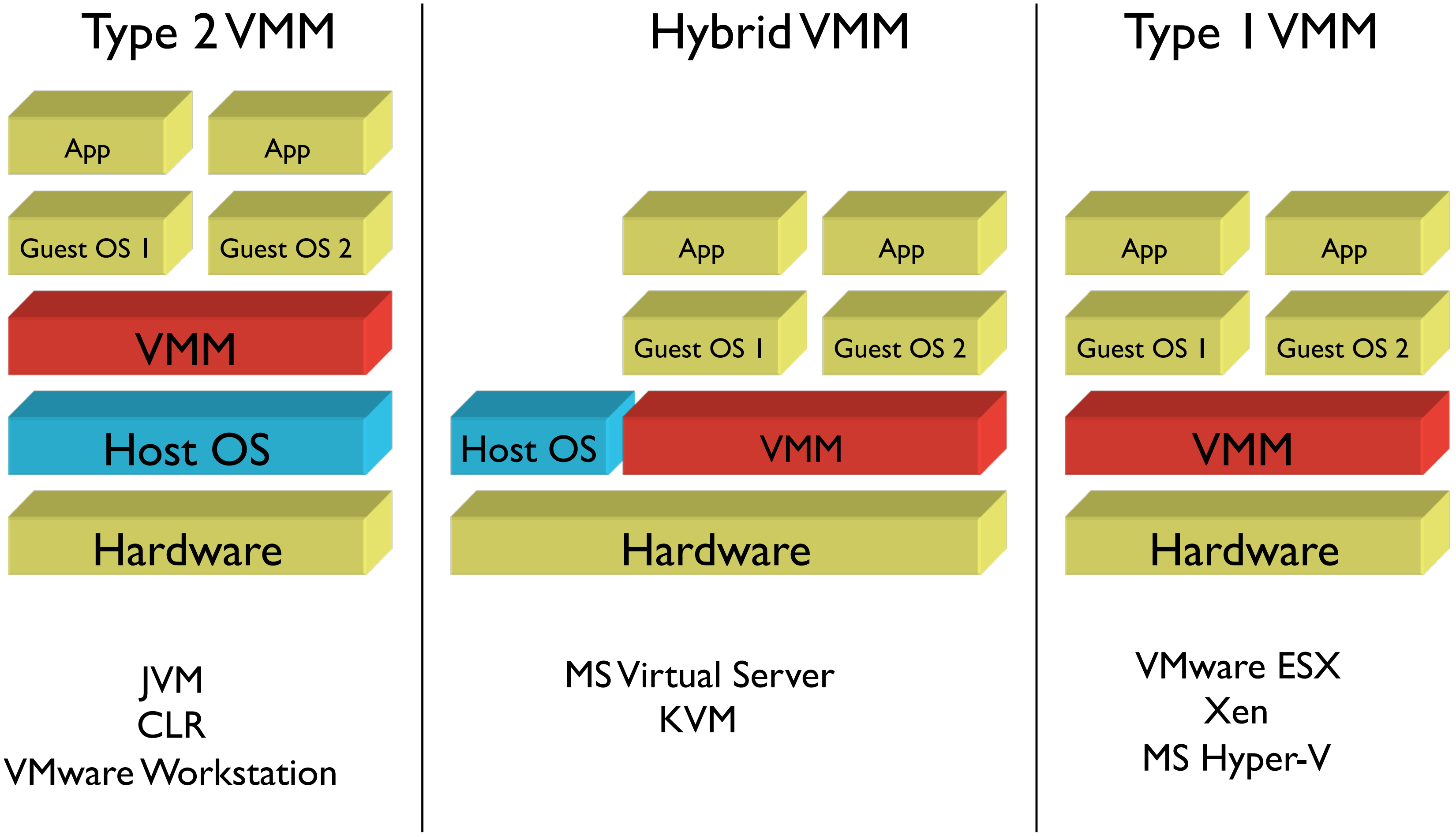
- ***Full system simulation***
  - CPU can be simulated
- ***Paravirtualization (Xen)***
  - VM has a special API
  - Requires OS changes
- ***Native virtualization (VMWare)***
  - Simulate enough HW to run OS
  - OS is for same CPU
- ***Application virtualization (JVM)***
  - Application API

Virtual Machine Types

- ***Type I***
  - Lowest layer of software is VMM
  - E.g., Xen, VAX VMM, etc.
- ***Type II***
  - Runs on a host operating system
  - E.g., VMWare, JVM, etc.

- Q: What are the trust model issues with Type II compared to Type I?

Virtual Machine Types



Type 2 VMM

JVM
CLR
VMware Workstation

Hybrid VMM

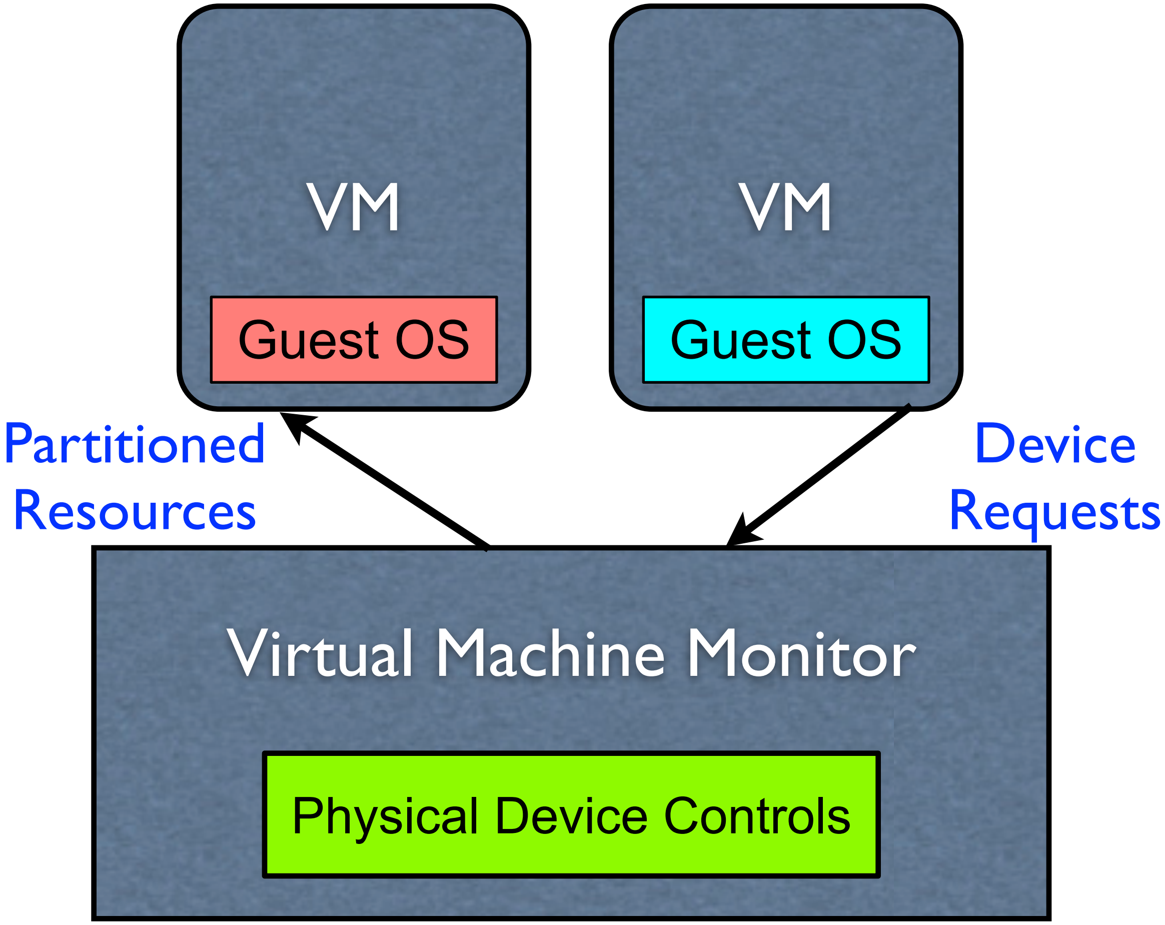
MS Virtual Server
KVM

Type 1 VMM

VMware ESX
Xen
MS Hyper-V

VM Security

- Isolation of VM computing
- Like a separate machine



VAX VMM System

- First system design to examine virtualization in the context of information flow security
  - Virtualization mechanisms necessary to implement a *reference validation mechanism* that satisfies the *reference monitor concept*
  - Assure system design and implementation to the highest level – *A1 level per the Orange Book*
  - Control all system information flows according to *MLS* and *Biba integrity policies* (modulo exceptions in “privileges”)
  - Also, *covert channel countermeasures* were produced, approximating noninterference
- System was piloted, but not released commercially

VAX VMM System

- Key design tasks of secure VMM
  - Virtualize processor
    - All security-sensitive instructions must be mediated by VMM
  - VMM protection ring
    - VMM must be deployed in a more privileged protection ring than the VMs
  - I/O emulation
    - Privileged I/O tasks must be executed in VMM or trusted VM
  - Self-virtualizable
    - OS must not detect when running on a VMM (or VMMs)

Virtualizing Instructions

- Security-Sensitive Instructions
  - *Instructions that read or modify privileged system state*
- Privileged Instructions
  - *Instructions that cause a trap when executed in a non-privileged ring*
- All security-sensitive instructions must be privileged to enable the VMM to manage privileged system state (rather than individual VMs)
- This requirement was not met by VAX hardware nor x86 originally

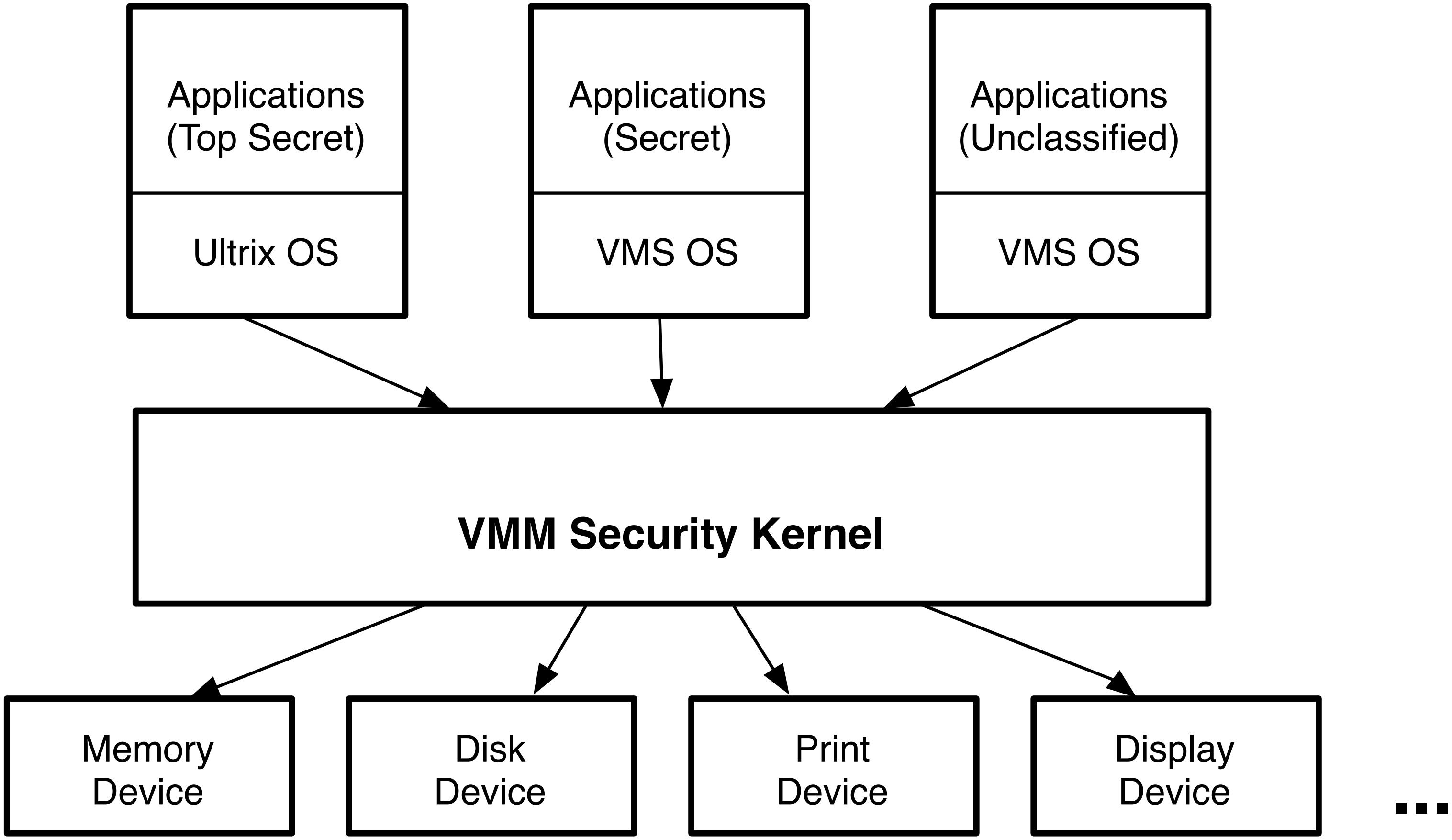
I/O Emulation

- Access to devices is expected by each operating system, but this access is security-sensitive
  - Thus, devices are virtualized
- Access to devices must be directed to the party with physical device access
  - Memory-mapped I/O uses unprivileged instructions
- VAX VMM adds a layer of indirection
  - I/O interface that causes a trap
  - OS must be modified to use that interface (paravirtualize)

Other Issues

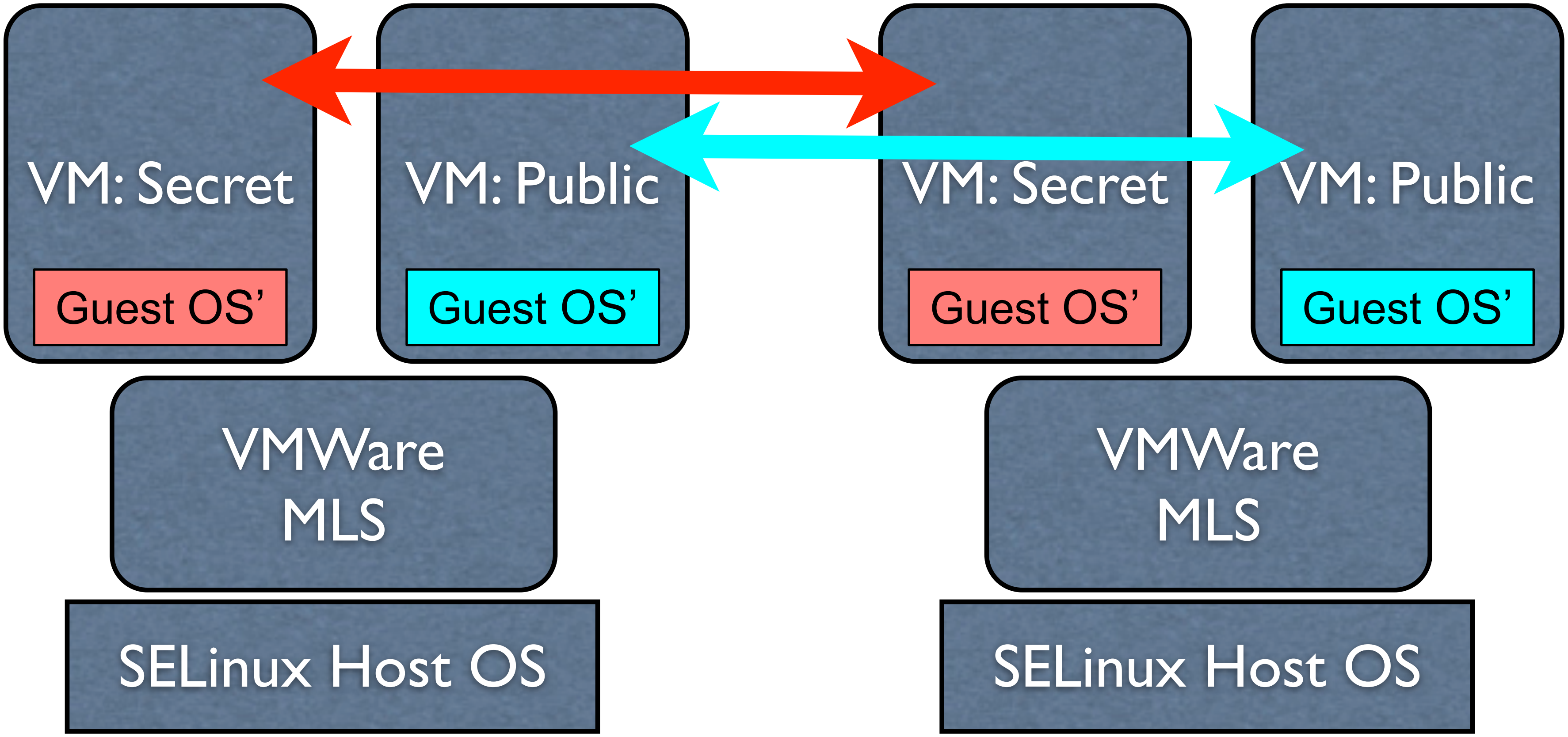
- Driver management
  - In VAX VMM, all drivers were in the VMM kernel
  - This was for assurance, but added code to VMM
    - Drivers are outside the VMM in most systems
- DMA
  - Devices can use this mechanism to write to physical memory, but under guidance of untrusted VMs
    - VAX VMM trusted drivers, but not practical today
- Performance – E.g., page table lookups

VAX VMM System



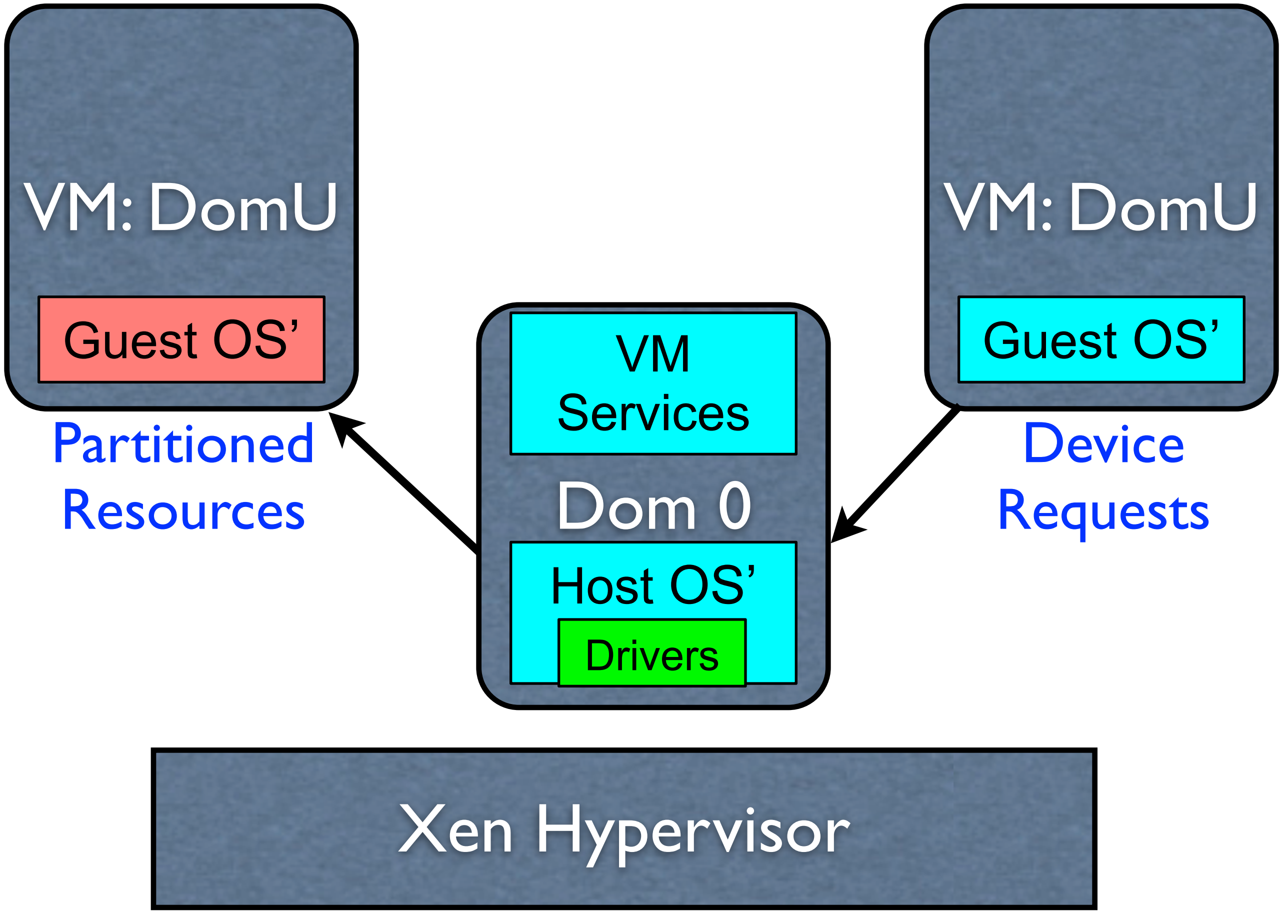
NetTop

- Isolated networks of VMs
- Alternative to “air gap” security



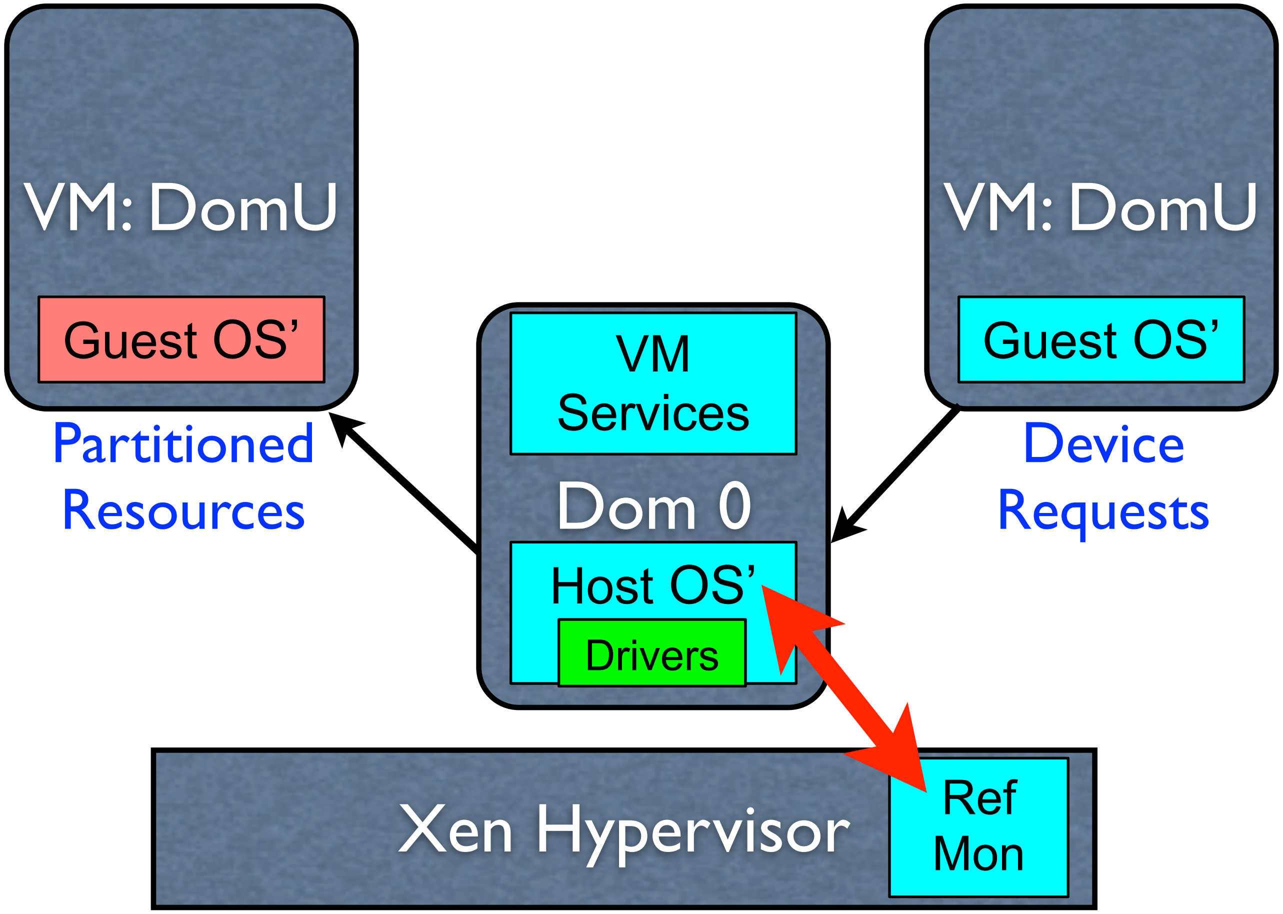
Xen

- Privileged VM



Xen sHype

- Controlled information flows among VMs



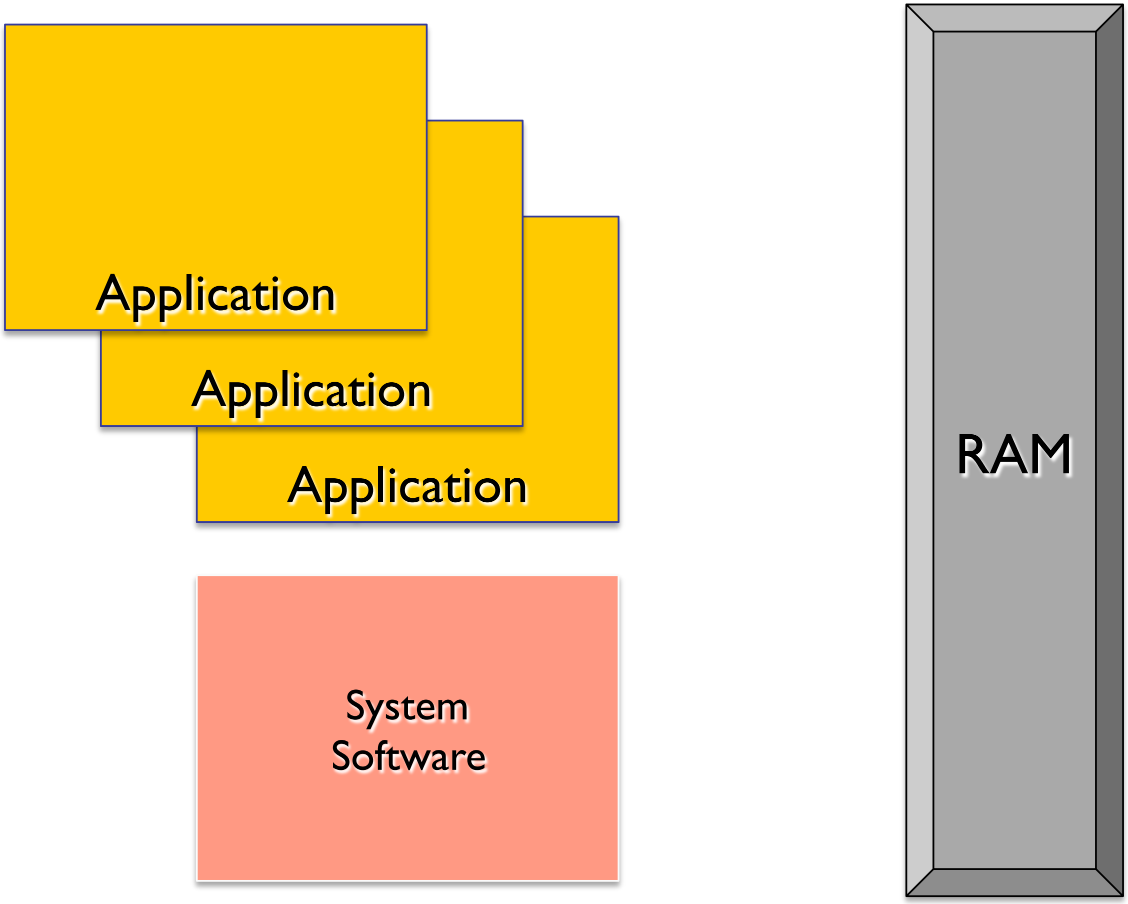
Xen sHype Policies

- Type Enforcement over VM communications
  - VM labels are subjects
  - VM labels are objects
- How do VMs communicate in Xen?
  - Grant tables: pass pages between VMs
  - Event channels: notifications (e.g., when to pass pages)
- sHype controls these
- Q: What about VM communication across systems?

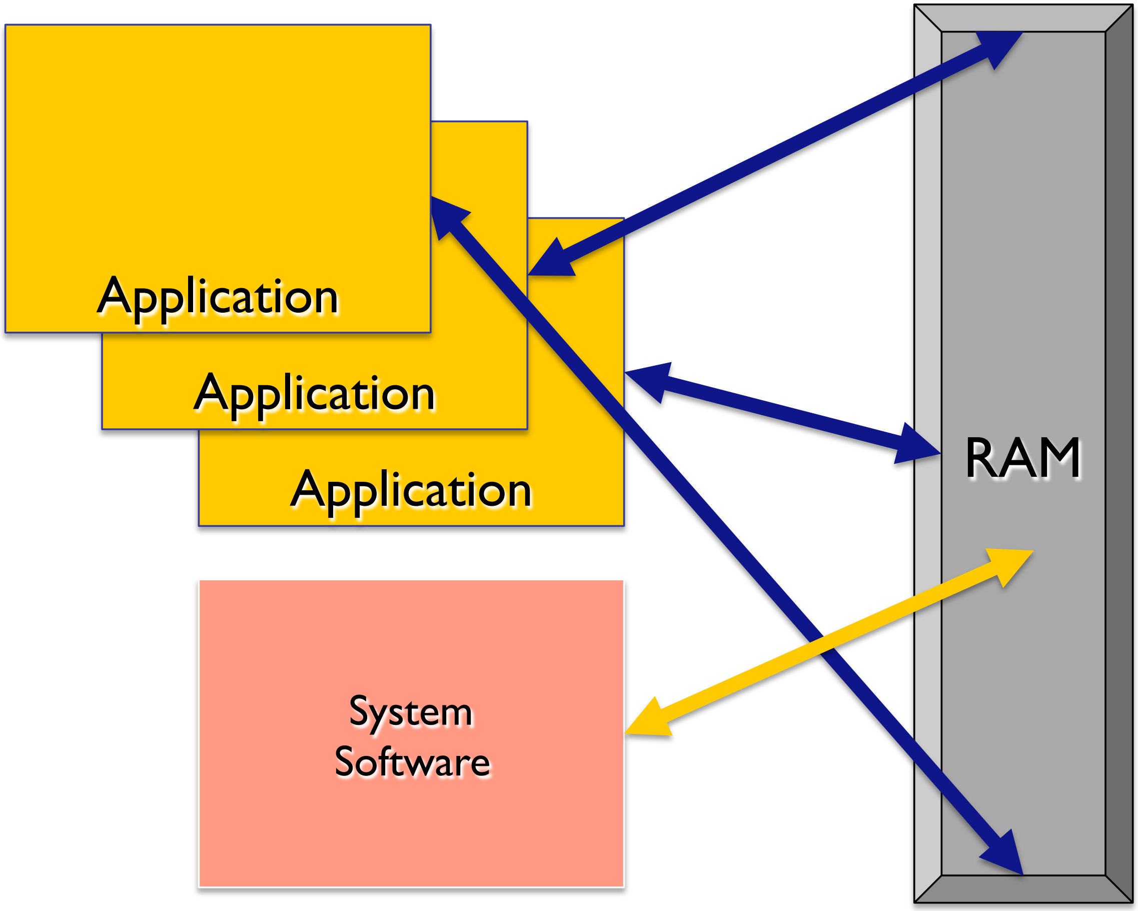
Xen Security Modules

- Comprehensive Reference Monitor interface for Xen
  - Based on LSM ideas
- Includes about 57 “hooks” (more expected)
  - Supports sHype hooks
  - Plus, hooks for VM management, resource partitioning
- Another aim: Decompose domain 0
  - Specialize kernel for privileged operations
  - E.g., Remove drivers

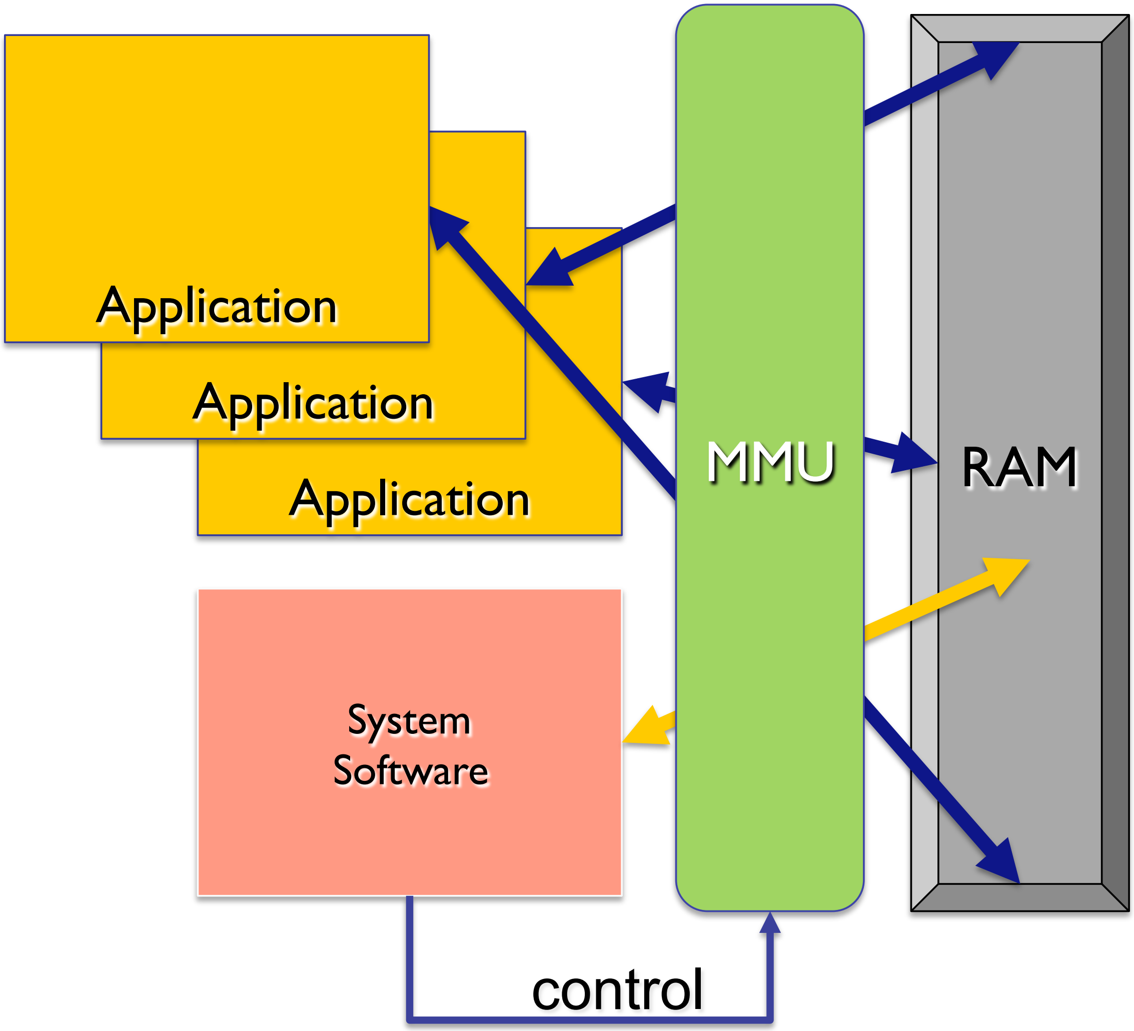
IOMMU Role in the System



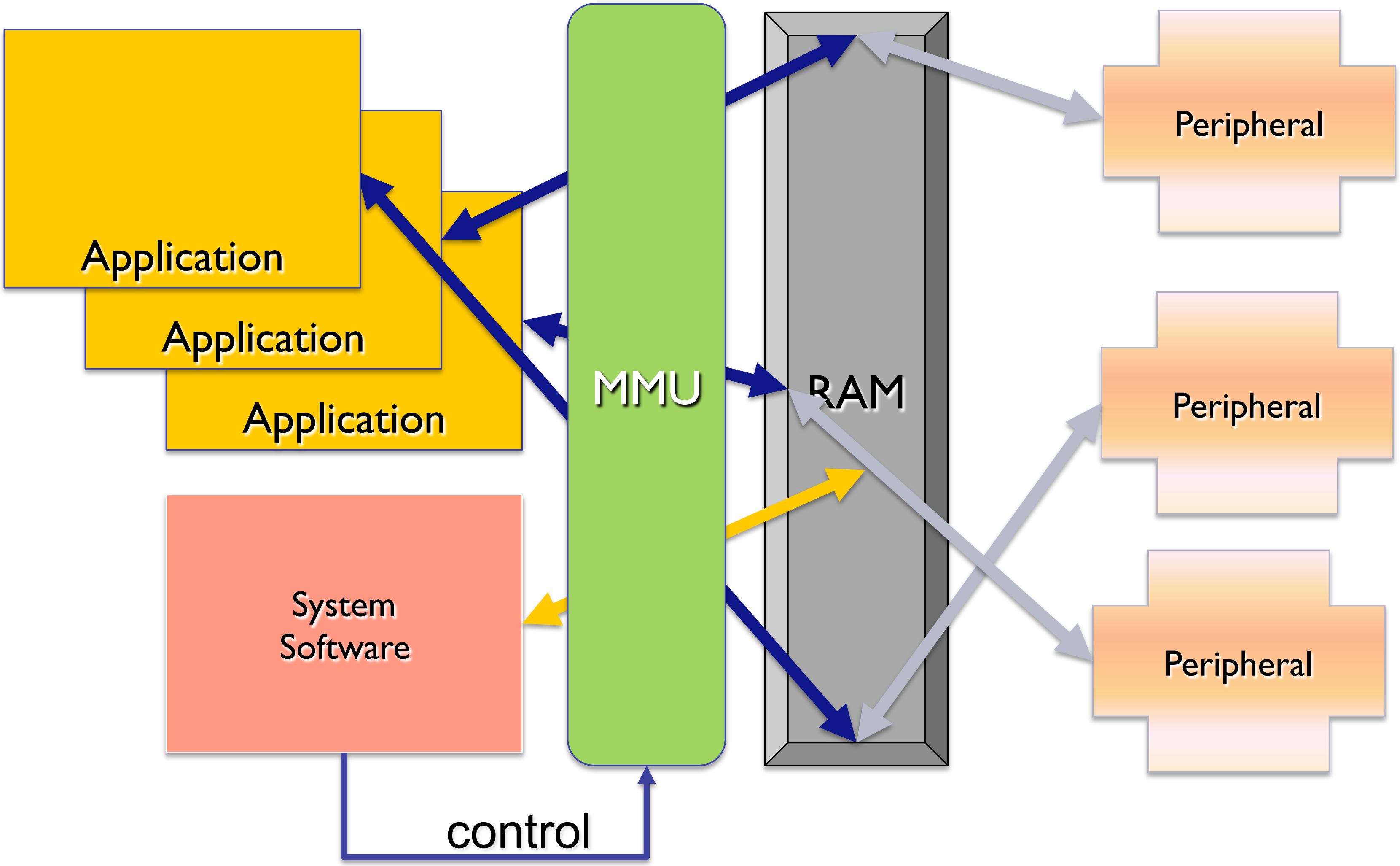
IOMMU Role in the System



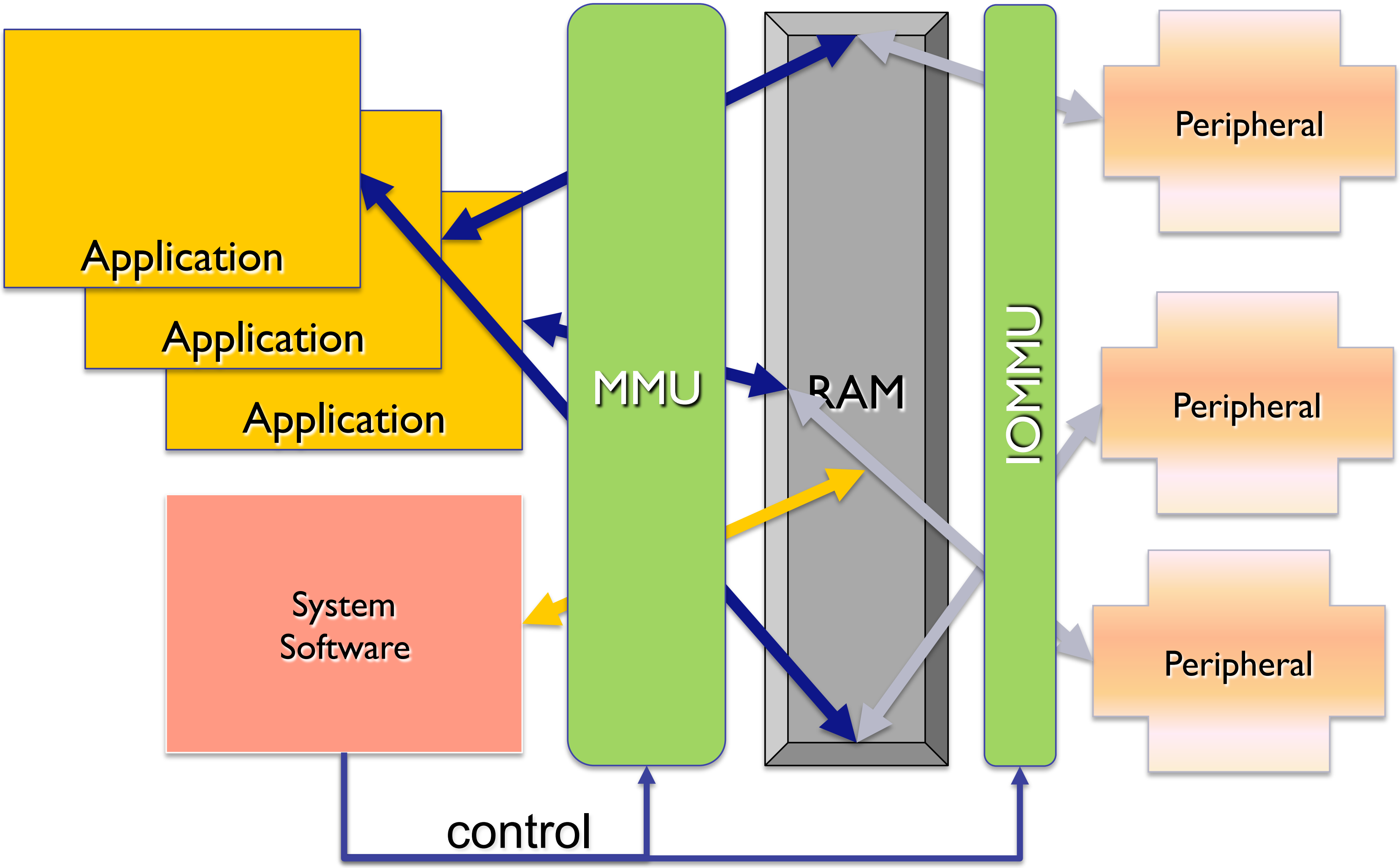
IOMMU Role in the System



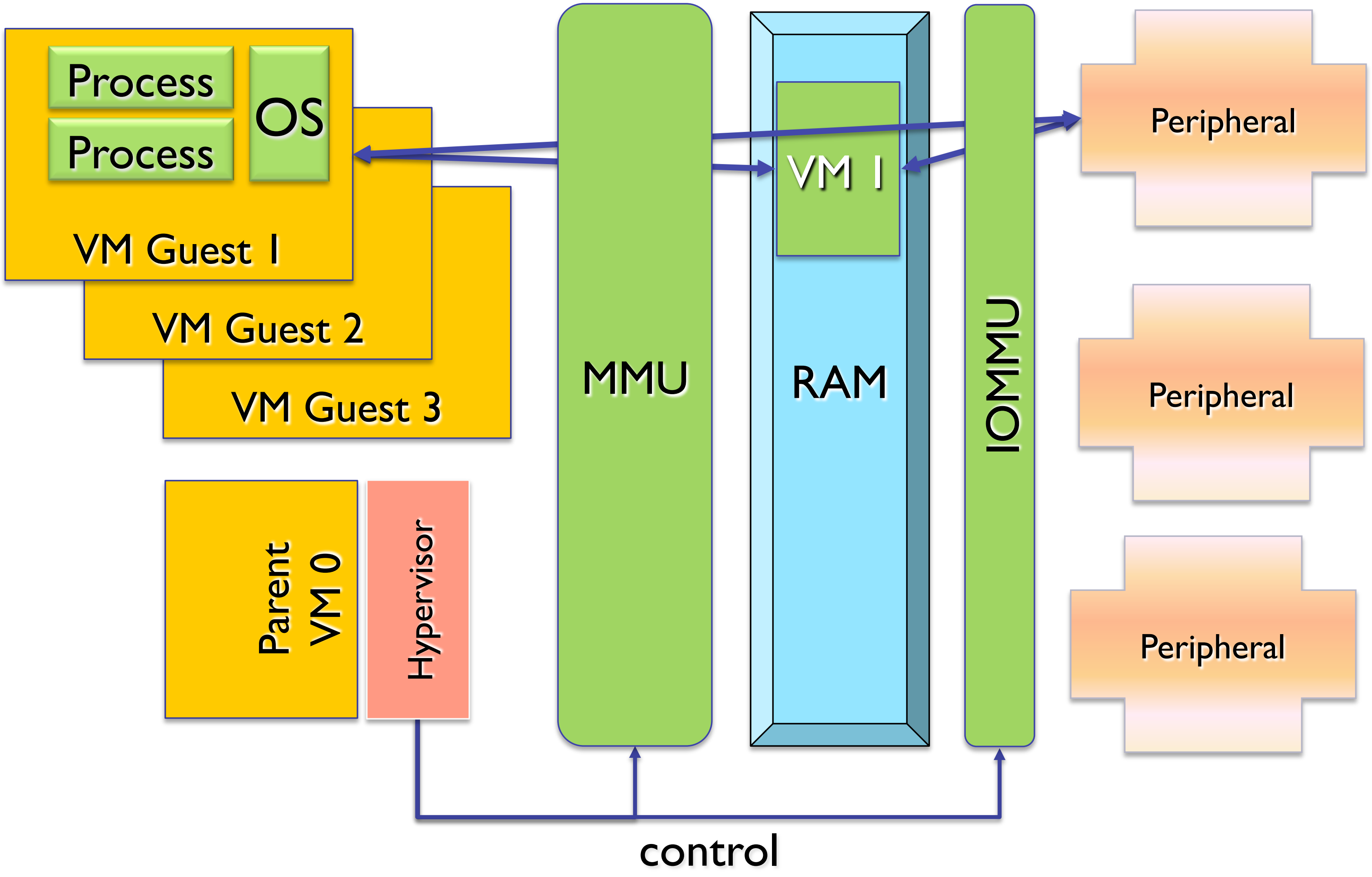
IOMMU Role in the System



IOMMU Role in the System



I/O Device Assignment



VM Security Status

- Aim is simplicity
  - Are we achieving this?
- Do we care what happens in the VMs?
  - When might we care?
- Trusted computing base
  - How does this compare to traditional OS?

Virtual Machine Threats

- How does the insertion of a virtual machine layer change the threats against the system?

Virtual Machine Rootkit

- Rootkit
  - Malicious software installed by an attacker on a system
  - Enable it to run on each boot
- OS Rootkits
  - Kernel module, signal handler, ...
  - When the kernel is booted, the module is installed and intercepts user process requests, interrupts, etc.
  - E.g., keylogger
- VM Rootkit
  - Research project from Michigan and Microsoft
  - If security service runs in VM, then a rootkit in VMM can evade security
  - E.g., Can continue to run even if the system appears to be off

Take Away

- VM systems focus on isolation
  - Enable reuse, but limited by security requirements
- Enable limited communication
  - The policies are not trivial, but refer to coarser-grained objects