

# An Attack-Resistant and Rapid Recovery Desktop System

A PhD Defense by Todd Deshane  
Clarkson University  
August 25, 2010

# Outline

- Motivation
- Background
- Design
- Implementation
- Evaluation
- Conclusions and Future Work

# Motivation

# Motivation: Personal Computing

- The general-purpose computing model is broken
  - Personal computers (PCs) provide too much power to users
  - Users don't make full use of the computer, but malicious software (malware) does.
    - Malware can send a large quantity of spam
    - 90% of email is spam
- Too much of the PC's security is placed on users
  - Users are required to update the operating system, applications, and virus definitions

# Motivation: Personal Data

- Even if users update software and virus definitions, the latest (zero day) exploits can still infect them
  - System updates and anti-virus software fall short
- When infected with a virus, challenges arise
  - Existence of personal data backups
  - Re-installation of the system and applications
- The system data can be recovered, but personal data might not be recoverable
  - Re-writing a document is possible
  - Recovering digital photos from a one-time event may not be possible

# Motivation: Lack of Security

- The Internet was not designed with security in mind
  - A trusting, naive Internet design has been exploited by malware
- Explosive growth of the world wide web in the 1990s
  - Millions of users (not all users are nice)
- On the most popular operating system of today:
  - Users run with full administrative rights
  - Malware infiltrates a user account
  - Malware gains full access (system and user data)
- Security improvements on newer operating systems
  - Malware can still get access to user data

# Motivation: Malware Behavior

- Evolution of malware
  - Early motives
    - Experimentation, curiosity, prank, vandalism
  - Current motives
    - Crime and profit
- Motives have changed, but methods remain the same
- Malware follows the trends (whatever is popular)
  - Social networking (Facebook, Twitter, etc.)
- Current malware activity often botnet-based
  - Spam, distributed denial of service, drive-by-downloads

# Motivation: Business Models

- Solving this problem doesn't help anyone's business
  - Fixing the problem doesn't:
    - Sell more computers
    - Sell new versions of operating systems
    - Sell new versions of security software
- Large software companies suggest:
  - Automatic updates are the best medicine
  - Users just need to follow security best practices
  - Their latest technology will make things better
- Current methods are too reactive to malware trends

# Motivation: Usable Security

- Mandatory Access Control (MAC) could be a decent solution
  - Usability is problematic
  - Difficult to configure
  - Disabled by users or disabled by default
  - False positives
- Users make poor security decisions
  - Users are goal/task-oriented
  - User studies: SSL browser warnings (Sunshine, et al.), Windows access control (Motiee, et al.)

Background

# Background: Security

- The Principle of Least Privilege (POLP)
  - "Every program and every user of the system should operate using the least set of privileges necessary" - Saltzer and Schroeder
- Complete Isolation
  - Separation of principals
  - No flow of information
- Access Control
  - Who can access what and with what permissions

# Background: Virtualization

- Hardware virtualization (platform virtualization)
  - Virtual Machine Monitor (VMM) or hypervisor
- Run multiple operating systems on the same computer
- Operating system instances = virtual machines (VMs)
- Strong isolation between VMs
- Virtual machine appliances (virtual appliances)
  - One or more applications in a VM
- Types of platform virtualization used in our Rapid Recovery Desktop system
  - Paravirtualization: high performance, modified OS
  - Hardware-assisted full virtualization: unmodified OS

# Background: Virtualization

Hardware

# Background: Virtualization



Hypervisor

Hardware

# Background: Virtualization



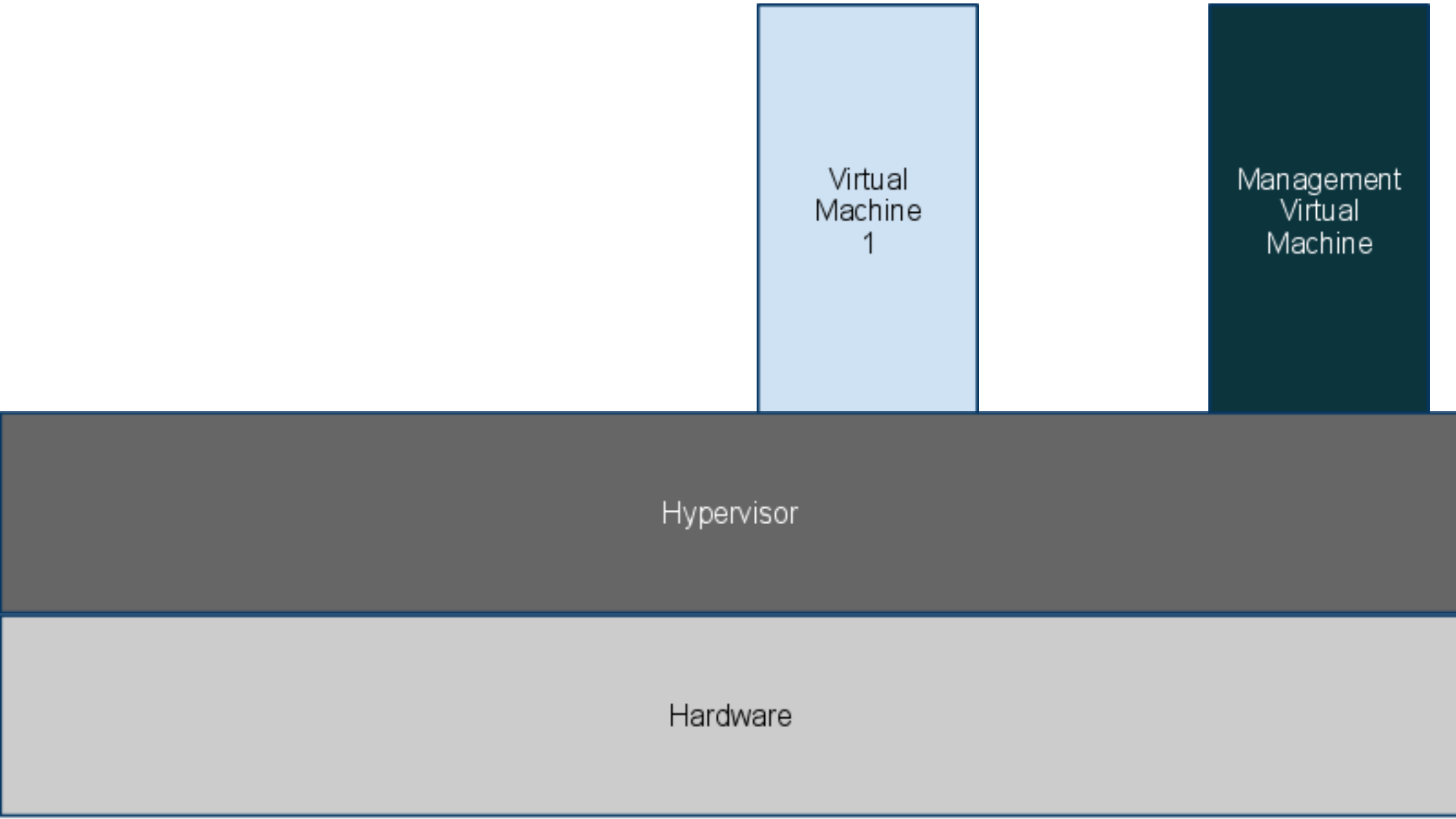
The diagram illustrates the layers of virtualization. It consists of three main horizontal layers. The bottom layer is a light gray rectangle labeled 'Hardware'. Above it is a dark gray rectangle labeled 'Hypervisor'. On top of the Hypervisor layer, on the right side, is a dark teal vertical rectangle labeled 'Management Virtual Machine'.

Management  
Virtual  
Machine

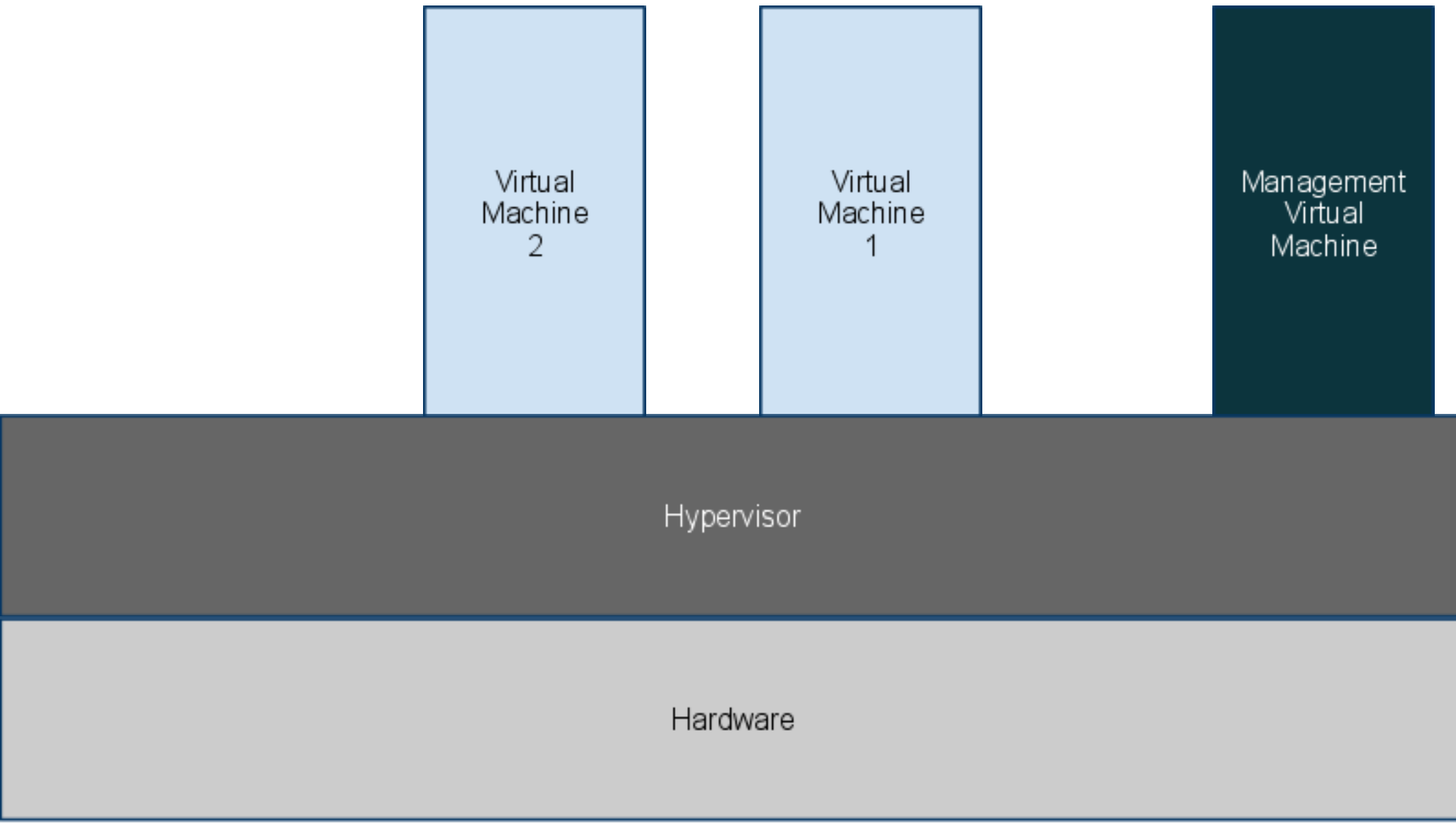
Hypervisor

Hardware

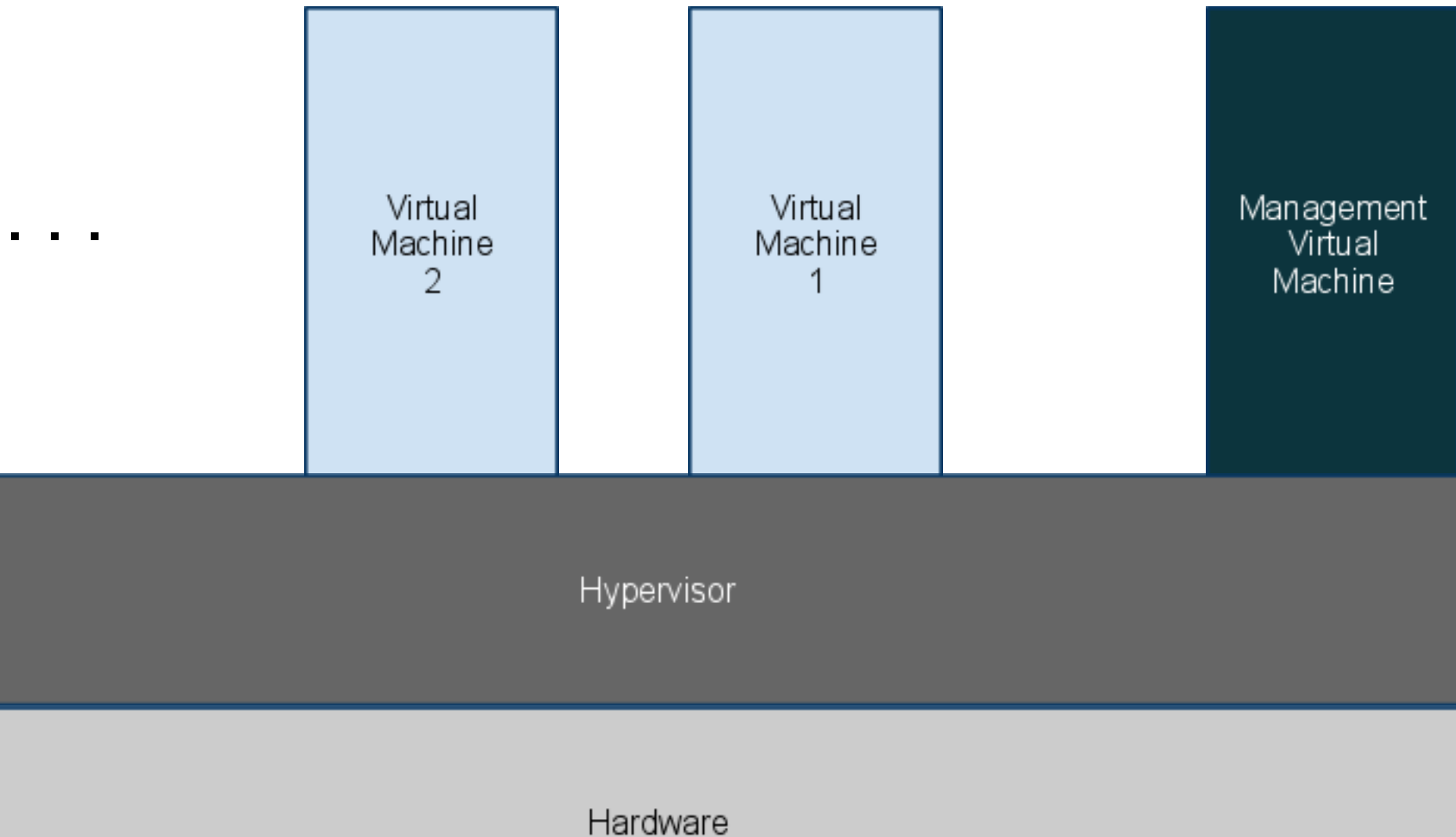
# Background: Virtualization



# Background: Virtualization



# Background: Virtualization

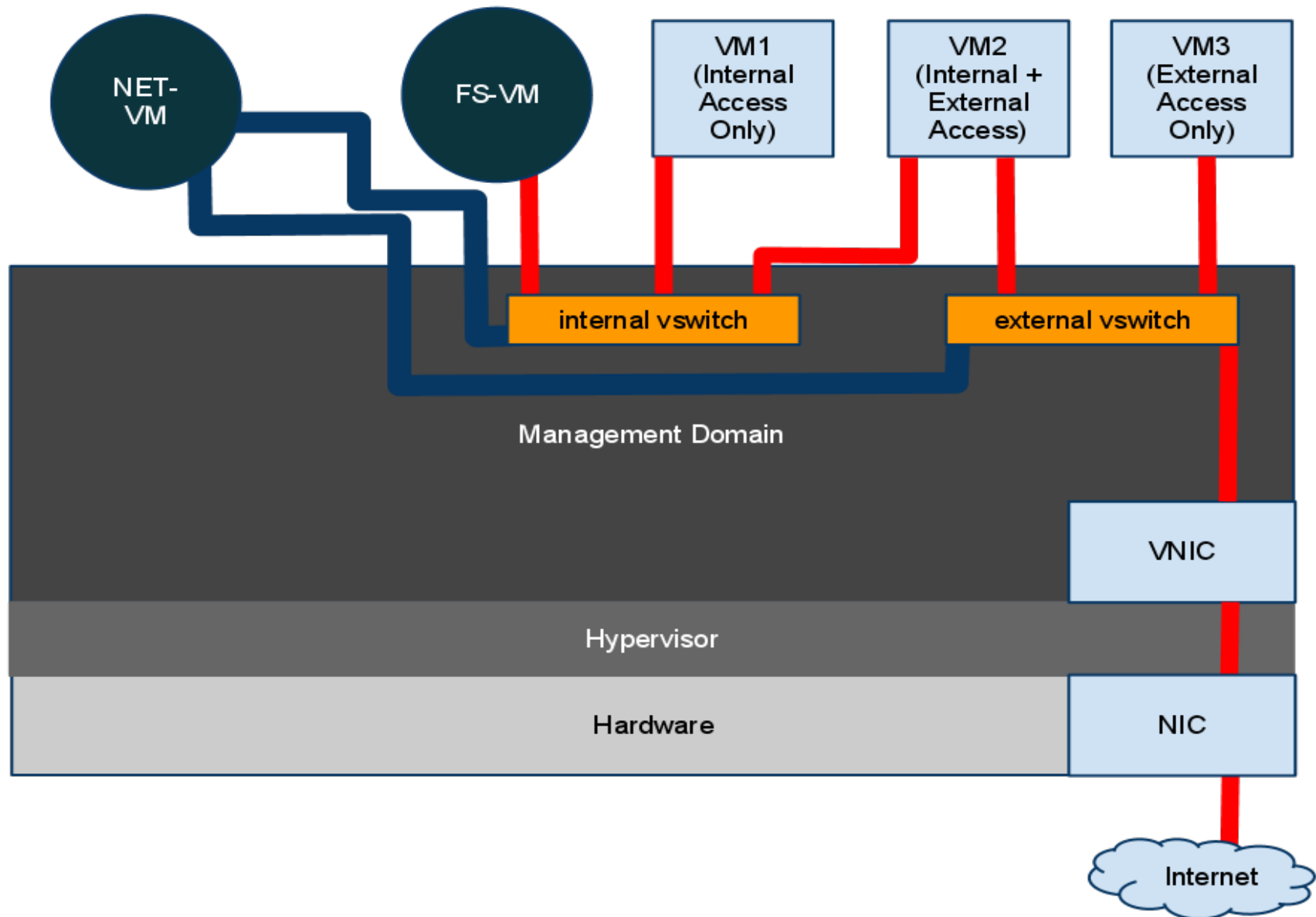


Design

# Design: Overview of Our System

- Place applications in virtual appliances
- File server virtual machine (FS-VM) stores user data
- Network virtual machine (NET-VM) protects network
- Each virtual appliance has an associated contract:
  - Specify the user data to mount from the FS-VM
  - Specify network access needs (e.g. web browsing)
- Implications of this design:
  - Restoring virtual appliance leaves user data intact
  - Malware is contained within a virtual appliance
  - Malware is restricted by the appliance's contract
  - Explicit support for POLP, isolation, access control

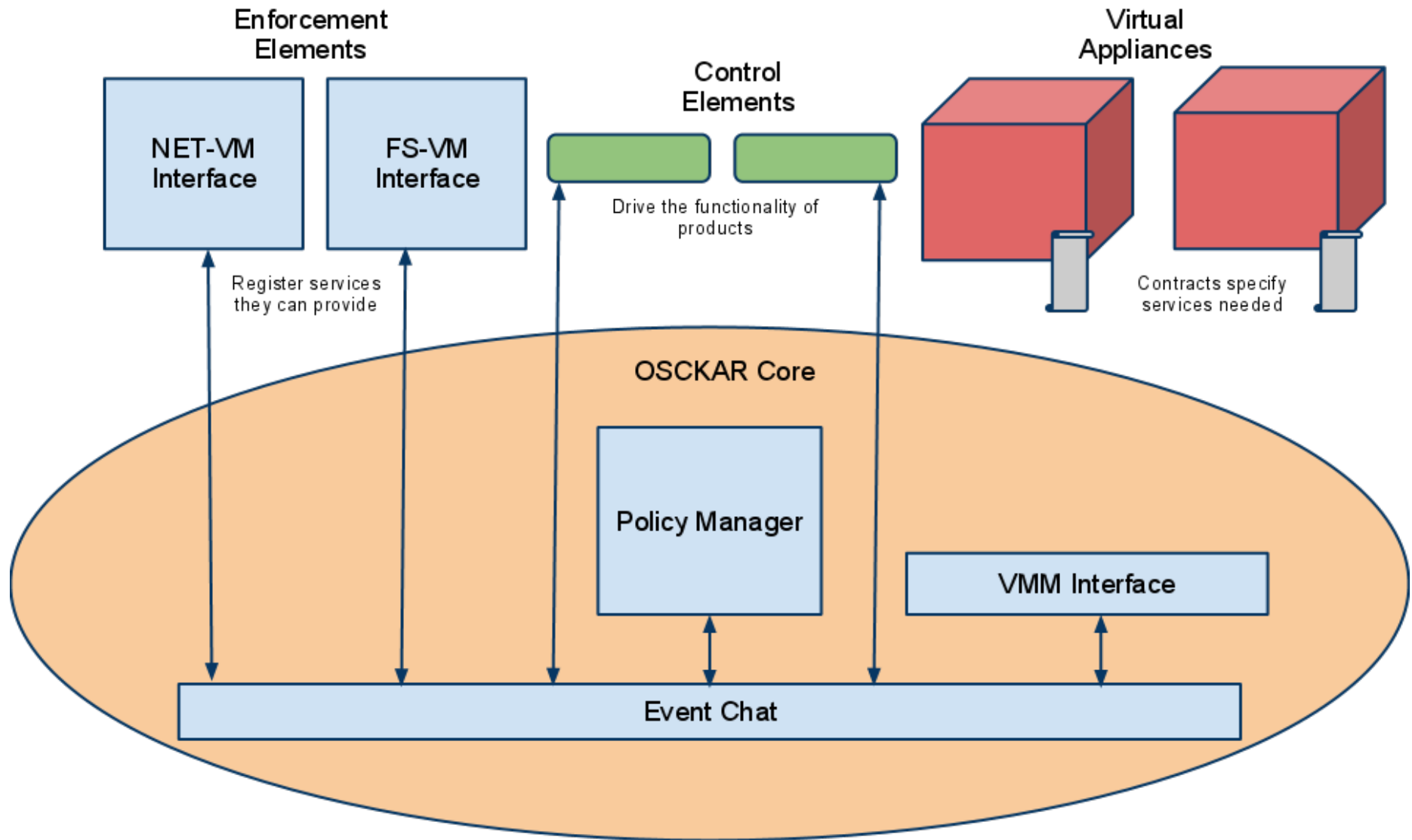
# Design: Network View



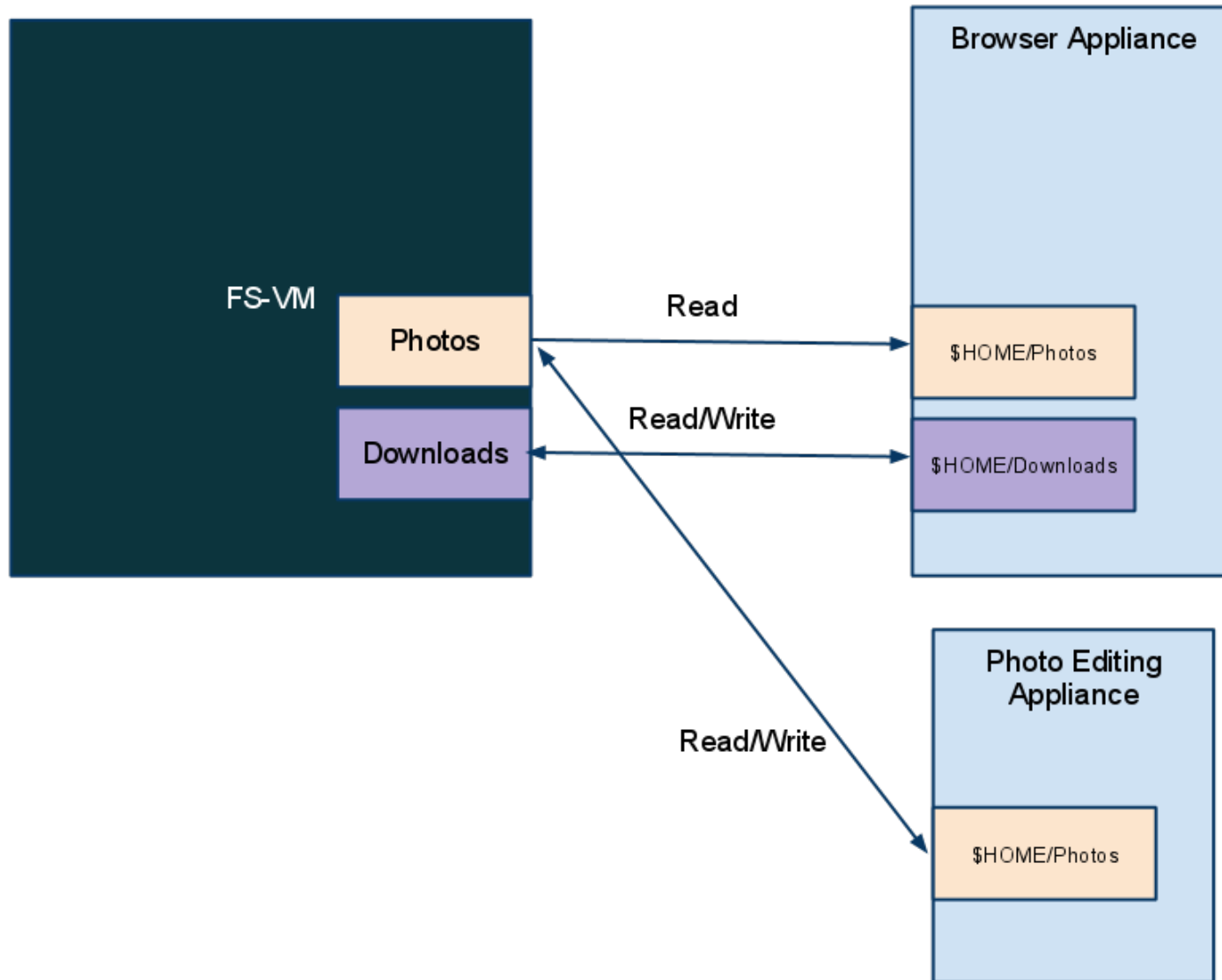
# Design: Virtualization Security Framework

- We developed OSCKAR, a virtualization security framework that includes:
  - Event Chat as a simple message bus
  - Policy Manager as a security daemon
  - VMM and builder interfaces
  - Enforcement elements (FS-VM, NET-VM)
  - Control elements (product control)
  - Virtual appliance contract handling

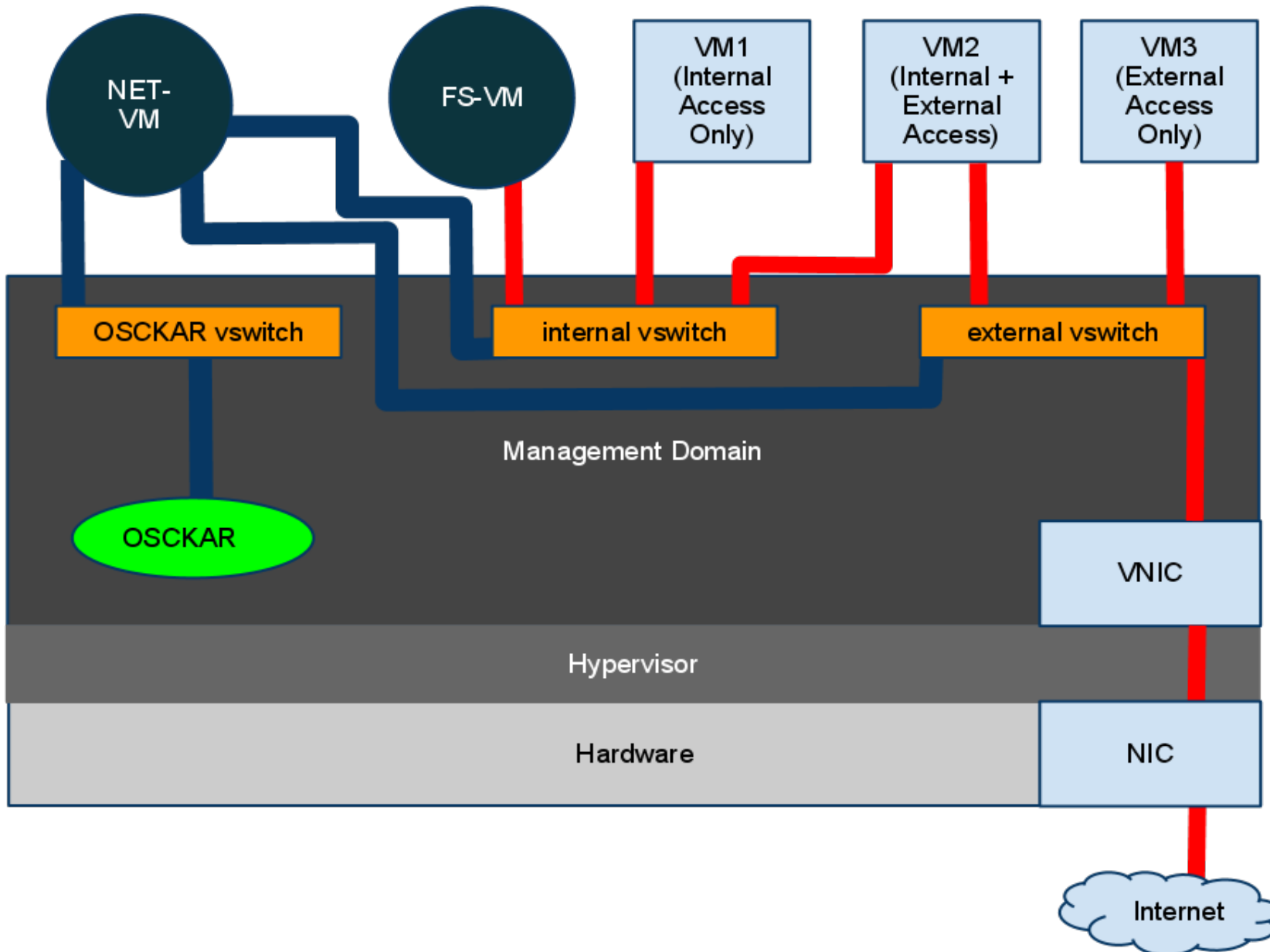
# Design: OSCKAR



# Design: File System Level



# Design: Network Level + OSCKAR

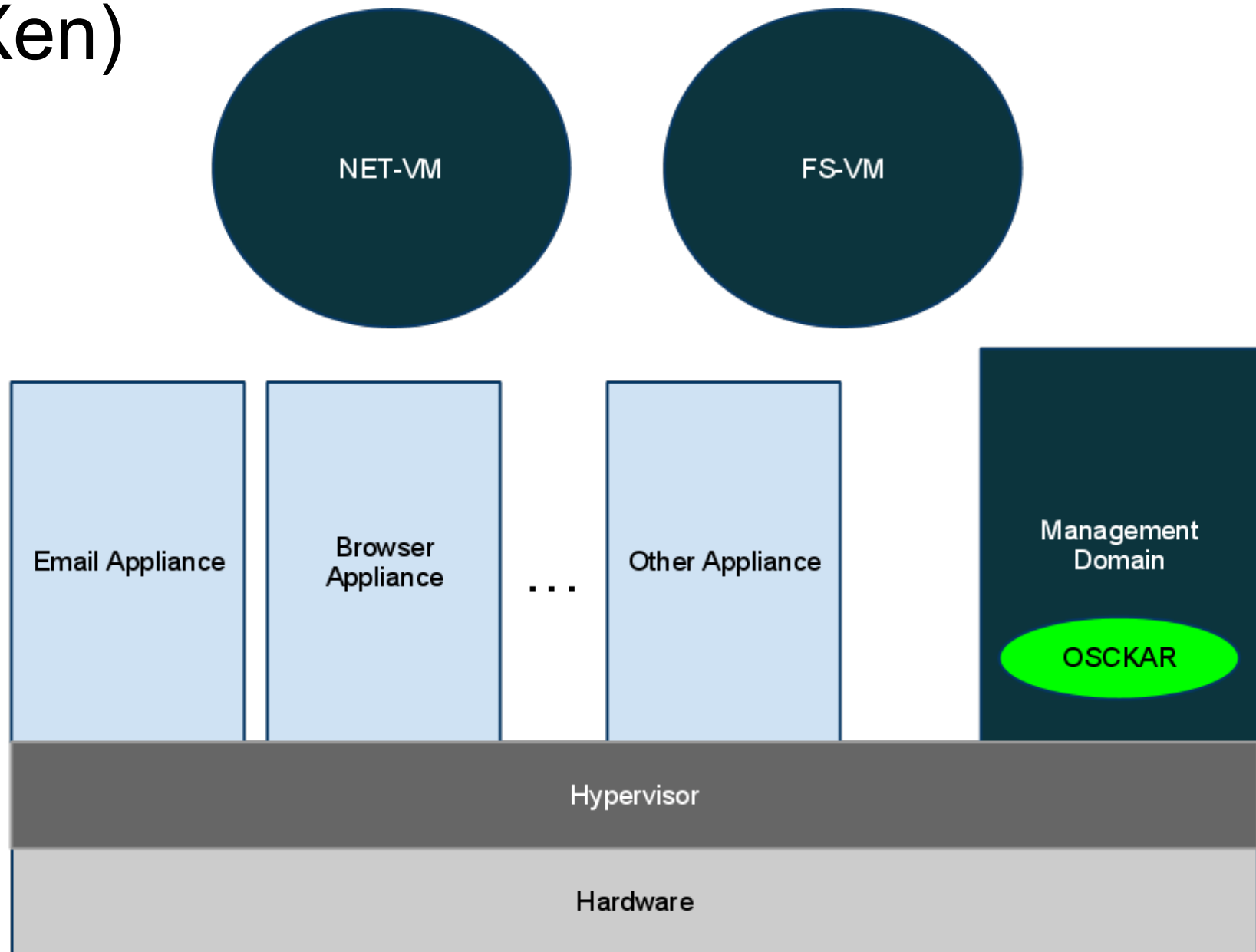


# Implementation

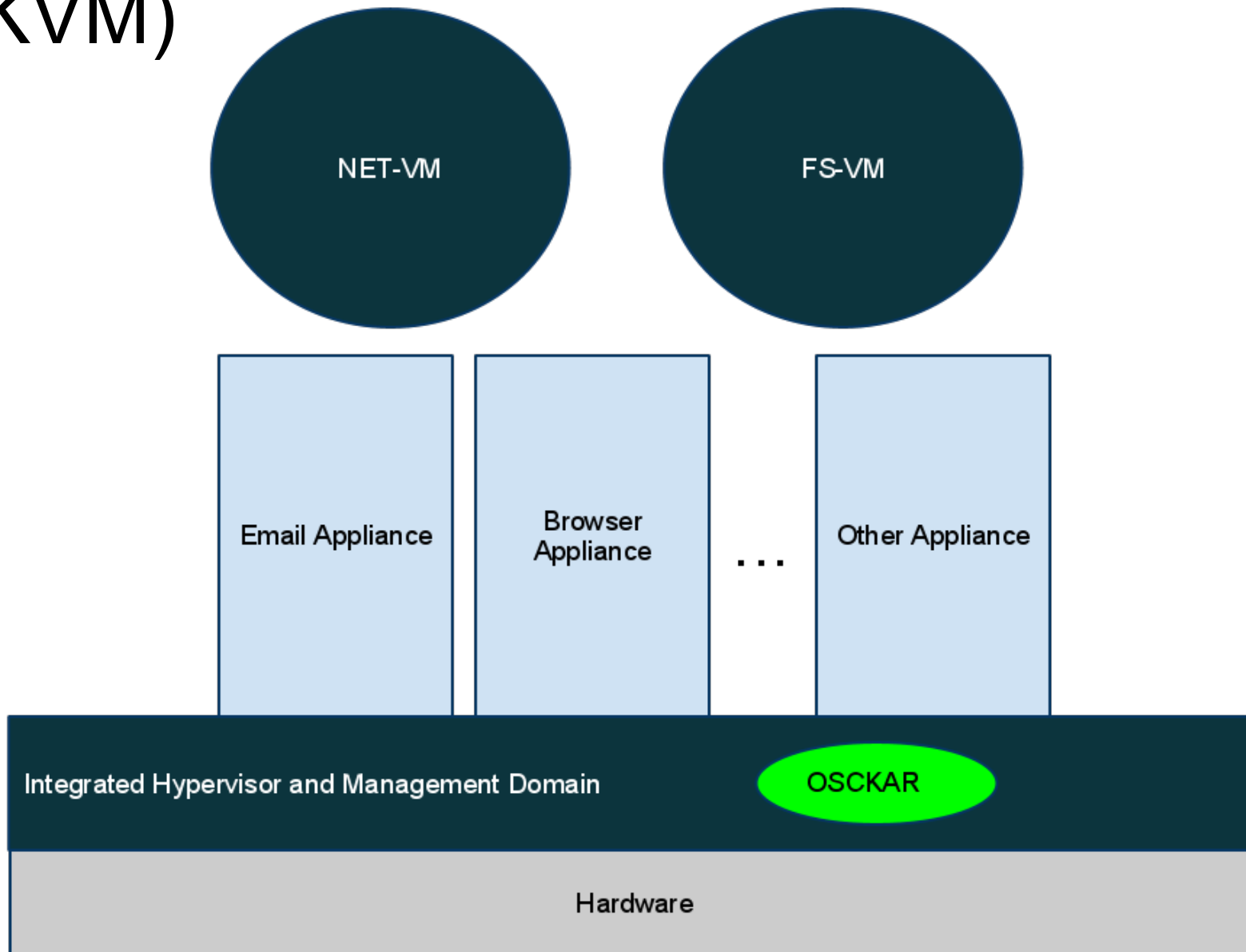
# Implementation: Virtualization

- Support for two open source hypervisors
  - Xen
    - Standalone
  - Kernel-based Virtual Machine (KVM)
    - Integrated
- Support libvirt toolkit
  - Interact with library abstraction
- Support using Xen/KVM directly
  - Interact with native Xen/KVM interfaces

# Implementation: Standalone Hypervisor (Xen)



# Implementation: Integrated Hypervisor (KVM)



# Implementation: Lines of Code

```
deshantm@xenbert: ~/src/osckar/trunk/core/usr/sbin
File Edit View Terminal Help
deshantm@xenbert:~/src/osckar/trunk/core/usr/sbin$ wc -l *
 187 eventchat
 325 osckar
   30 osckar-core-debug
 416 osckar-interface-builder
 197 osckar-interface-vmm
 131 policymanager
   13 run_builder_interface
   14 run_eventchat
   14 run_policymanager
   14 run_vmm_interface
1341 total
deshantm@xenbert:~/src/osckar/trunk/core/usr/sbin$
```

# Implementation: VM Contract (VMC)

```
<contract type="vmc" id="browser-appliance">
```

```
<ruleset type='vmm'>
```

```
.  
.
```

```
</ruleset>
```

```
<ruleset type='builder'>
```

```
.  
.
```

```
</ruleset>
```

```
<ruleset type='other-interfaces'>
```

```
.  
.
```

```
</ruleset>
```

```
<ruleset type='FS-VM'>
```

```
<mounts>
```

```
.  
.
```

```
</mounts>
```

```
<violations>
```

```
.  
.
```

```
</violations>
```

```
</ruleset>
```

```
<ruleset type='NET-VM'>
```

```
<flows>
```

```
.  
.
```

```
</flows>
```

```
<violations>
```

```
.  
.
```

```
</violations>
```

```
</ruleset>
```

```
<ruleset type='other-enforcement-elements'>
```

```
.  
.
```

```
</ruleset>
```

```
<ruleset type='PolicyManager'>
```

```
<responses event="some-event">
```

```
</responses>
```

```
<responses event="other-events">
```

```
</responses>
```

```
</ruleset>
```

```
</contract>
```

# Implementation: VMM Interface (Generic)

```
<ruleset type="vmm">
  <name>browser</name>
  <memory>
    <min>128MB</min>
    <max>1G</max>
    <default>512MB</default>
  </memory>
  <disks>
    <cowdisk type="qcow2" cowbase="browser.qcow2">browser</cowdisk>
  </disks>
  <vnics>
    <vnic type="bridge" />
  </vnics>
  <graphics>
    <graphic type="vnc" />
  </graphics>
  <displays>
    <display type='vga' />
  </displays>
  <features>
    <feature>acpi</feature>
  </features>
</ruleset>
```

# Implementation: VMM Interface (Specific)

```
<ruleset type="qemu-spice">
  <name>browser</name>
  <memory>
    <min>128MB</min>
    <max>1G</max>
    <default>512MB</default>
  </memory>
  <disks>
    <cowdisk type="qcow2" cowbase="browser.qcow2">browser</cowdisk>
  </disks>
  <vnics>
    <vnic type="bridge" />
  </vnics>
  <graphics>
    <graphic type='spice' port='5930' />
  </graphics>
  <displays>
    <display type="qxl" />
  </displays>
  <features>
    <feature>acpi</feature>
  </features>
</ruleset>
```

# Implementation: Enforcement (FS-VM)

```
<ruleset type='FS-VM'>
  <mounts>
    <mount from="$FS-VM:/downloads" write="true"
      to="$HOME/downloads" />
    <mount from="$FS-VM:/photos" read="true"
      to="$HOME/photos read-rate="5 files/min" />
  </mounts>
  <violations>
    <violation="read-rate-violation"
      raise="PolicyManager.read-rate-violation $VM" />
  </violations>
</ruleset>
```

```
<ruleset type='PolicyManager'>
  <responses event="PolicyManager.read-rate-violation">
    <response event="PolicyManager.log_violation"
      argument="$ARG" />
    <response
      event="RapidRecoveryDesktop_control.alert_user"
      argument="$ARG" />
    <response event="vmm.destroy_vm" argument="$ARG" />
    <response event="builder.restore_cow_vm"
      argument="$ARG" />
    <response event="vmm.start_vm" argument="$ARG" />
  </responses>
</ruleset>
```

# Implementation: Enforcement (NET-VM)

```
<ruleset type='NET-VM'>
<flows>
  <flow action="allow" type="web" direction="outgoing"
    rate="1Mbit/s" />
  <flow action="allow" type="dns" direction="outgoing"
    rate="10 lookups per second" />
  <flow action="allow" type="dhcp" direction="outgoing"
    rate="2 requests per day" />
</flows>
<violations>
  <violation="web-rate-violation"
    raise="PolicyManager.web-rate-violation $VM" />
  .
  .
  .
</violations>
</ruleset>
```

```
<ruleset type='PolicyManager'>
<responses event="PolicyManager.web-rate-violation">
  <response event="PolicyManager.log_violation"
    argument="$ARG" />
</responses>
.
.
.
</ruleset>
```

# Implementation: VMC (Full)

```
<contract type="vmc" id="browser-appliance">
```

```
<ruleset type='vmm'>
```

```
.  
.
```

```
</ruleset>
```

```
<ruleset type='builder'>
```

```
.  
.
```

```
</ruleset>
```

```
<ruleset type='other-interfaces'>
```

```
.  
.
```

```
</ruleset>
```

```
<ruleset type='FS-VM'>
```

```
<mounts>
```

```
.  
.
```

```
</mounts>
```

```
<violations>
```

```
.  
.
```

```
</violations>
```

```
</ruleset>
```

```
<ruleset type='NET-VM'>
```

```
<flows>
```

```
.  
.
```

```
</flows>
```

```
<violations>
```

```
.  
.
```

```
</violations>
```

```
</ruleset>
```

```
<ruleset type='other-enforcement-elements'>
```

```
.  
.
```

```
</ruleset>
```

```
<ruleset type='PolicyManager'>
```

```
<responses event="some-event">
```

```
</responses>
```

```
<responses event="other-events">
```

```
</responses>
```

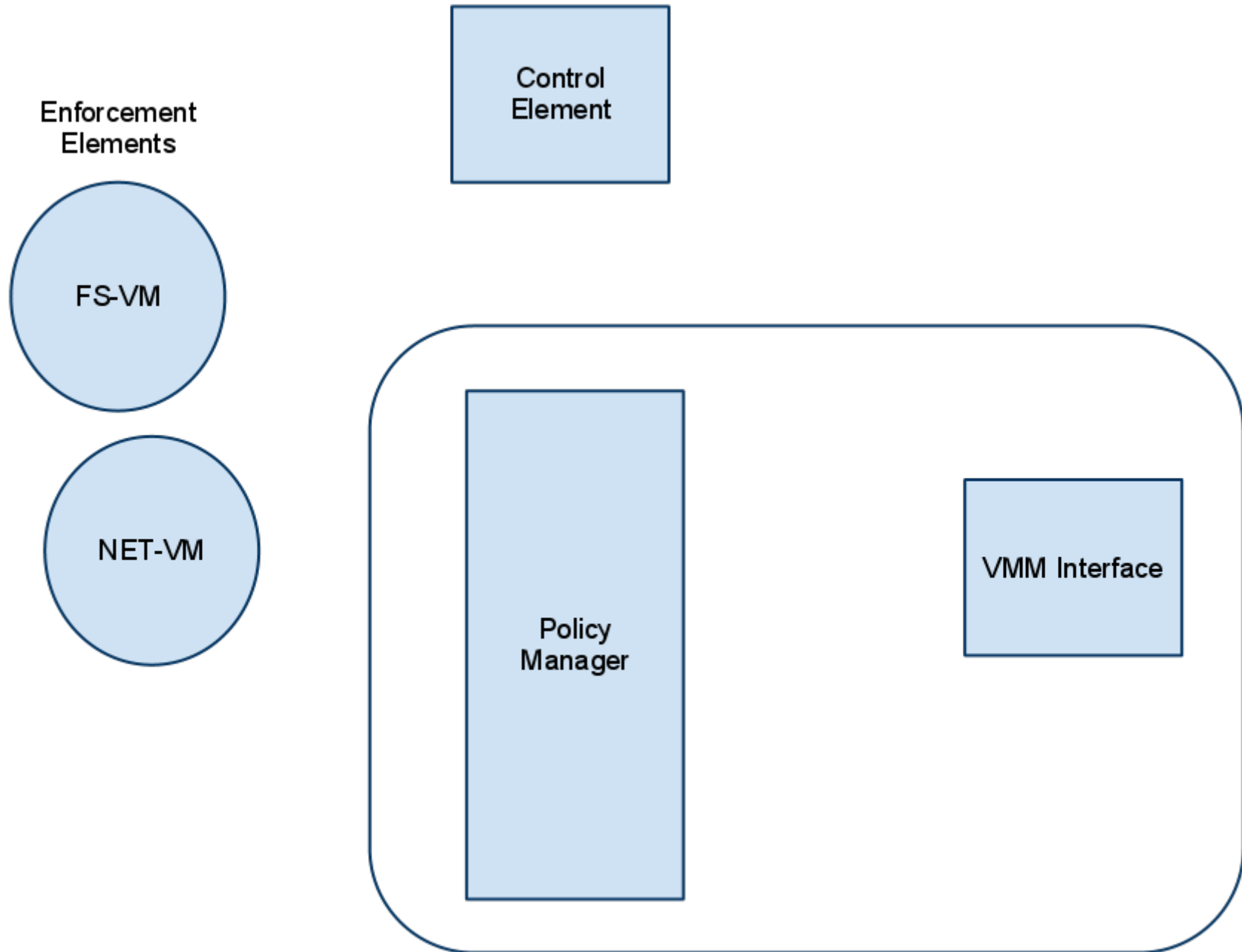
```
</ruleset>
```

```
</contract>
```

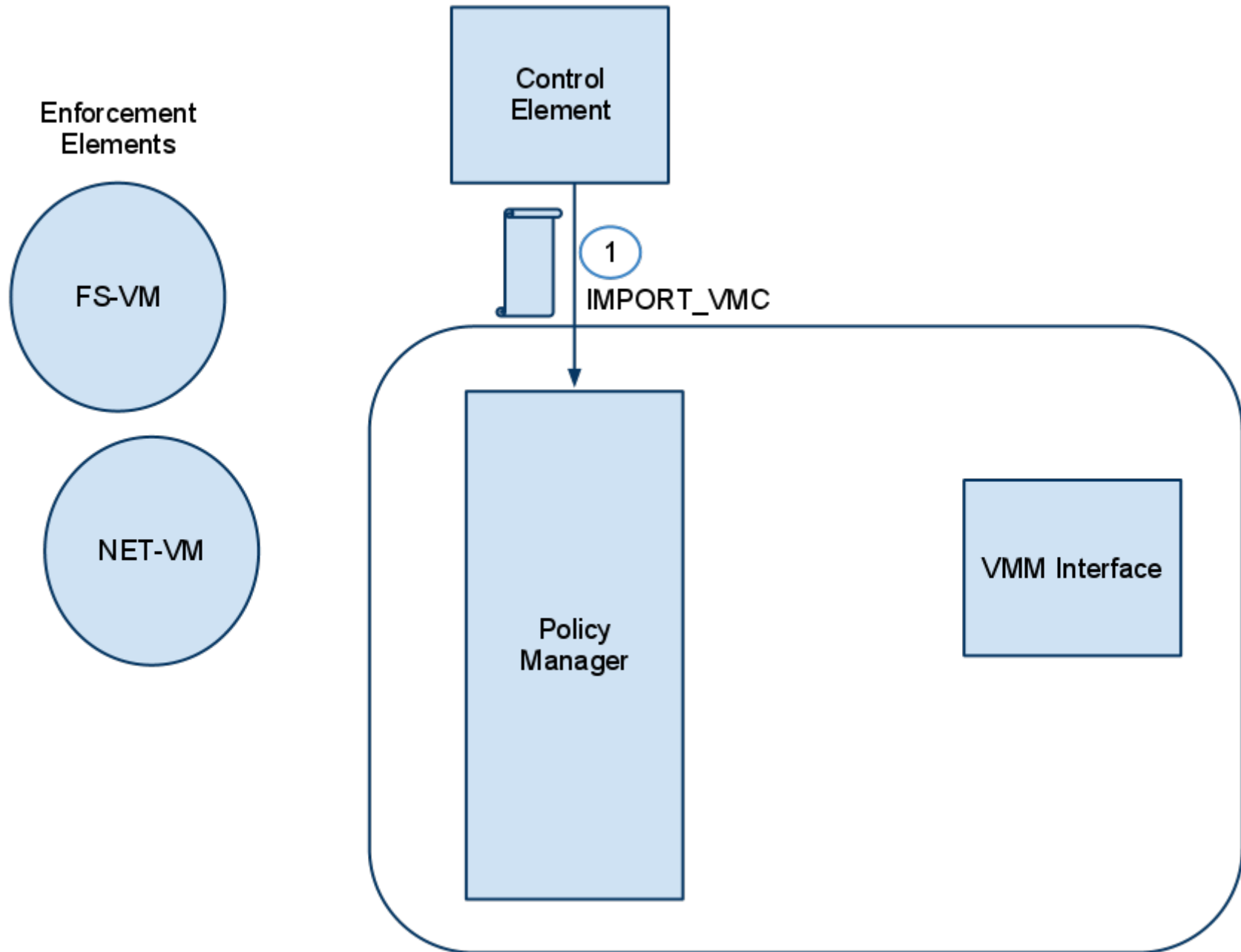
# Implementation: Policy Manager Contract (PMC)

```
<contract type="global_PolicyManager_policy">  
  <ruleset type="response">  
    <handle event="IMPORT_VMC">  
      <raise event="PolicyManager.import_contract" argument="$ARG " />  
    </handle>  
    <handle event="START_VM">  
      <raise event="PolicyManager.send_subcontracts" argument="$ARG" />  
      <raise event="vmm.start_vm" argument="$ARG" />  
    </handle>  
  </ruleset>  
</contract>
```

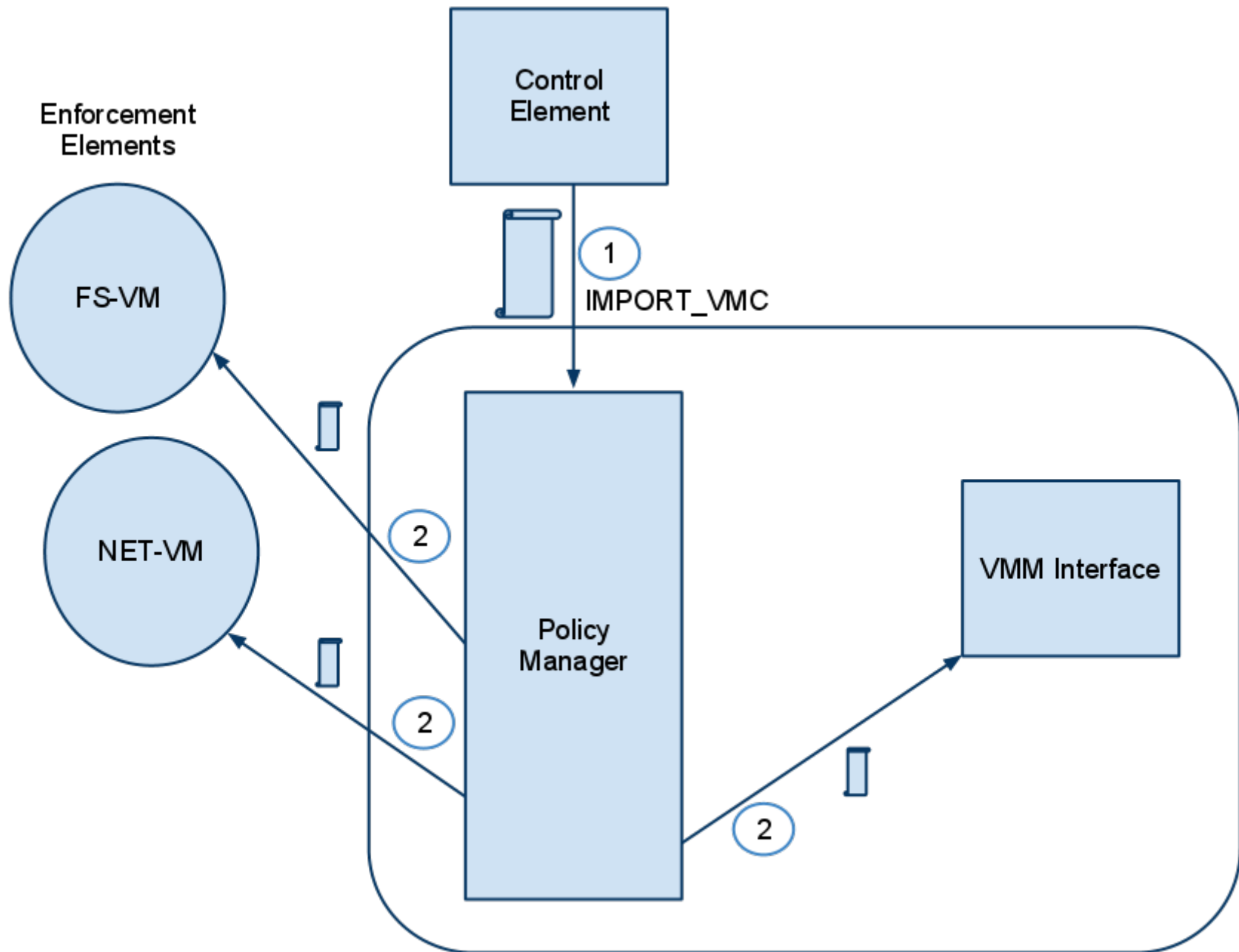
# Implementation: Contract Import



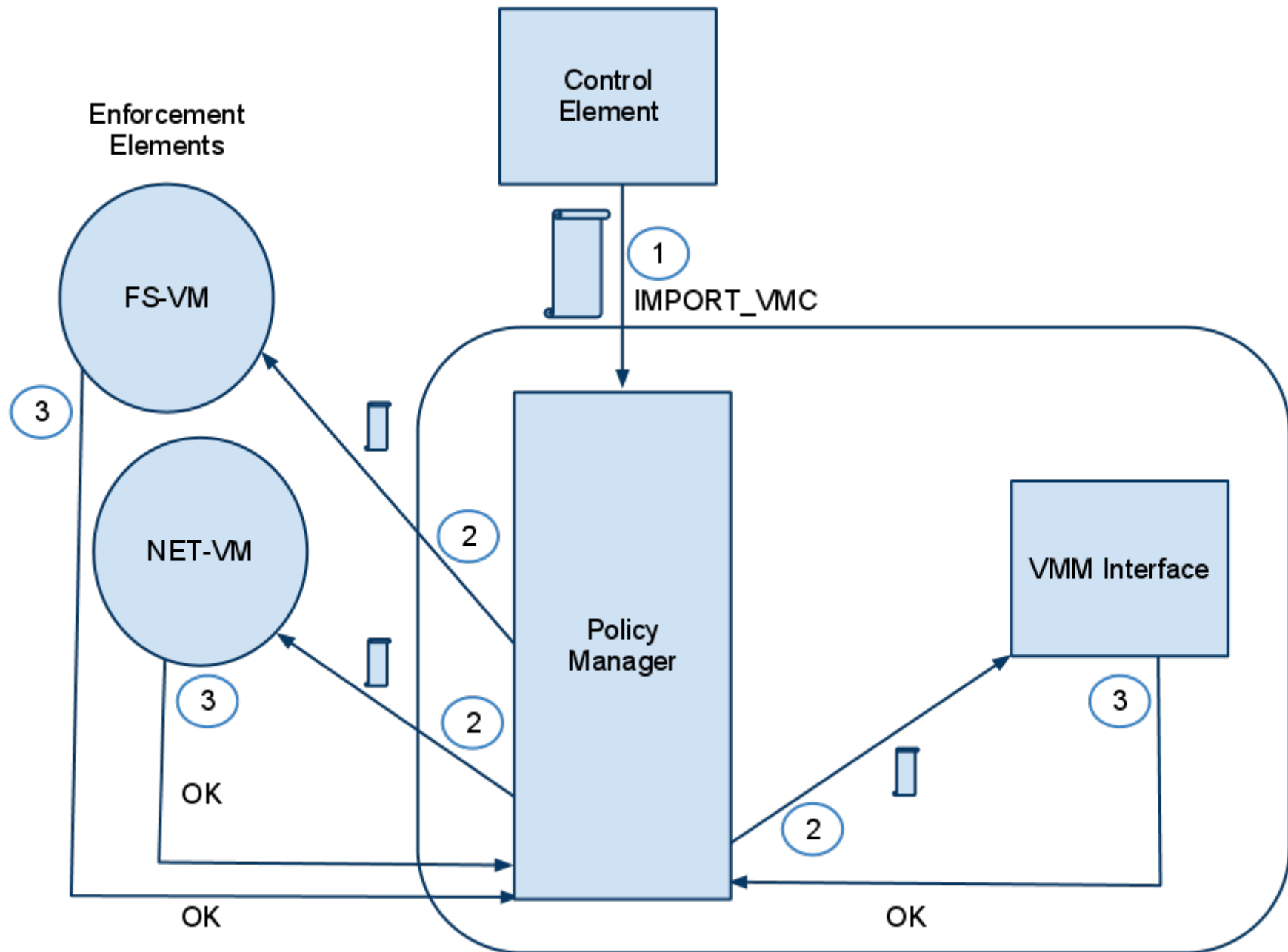
# Implementation: Contract Import



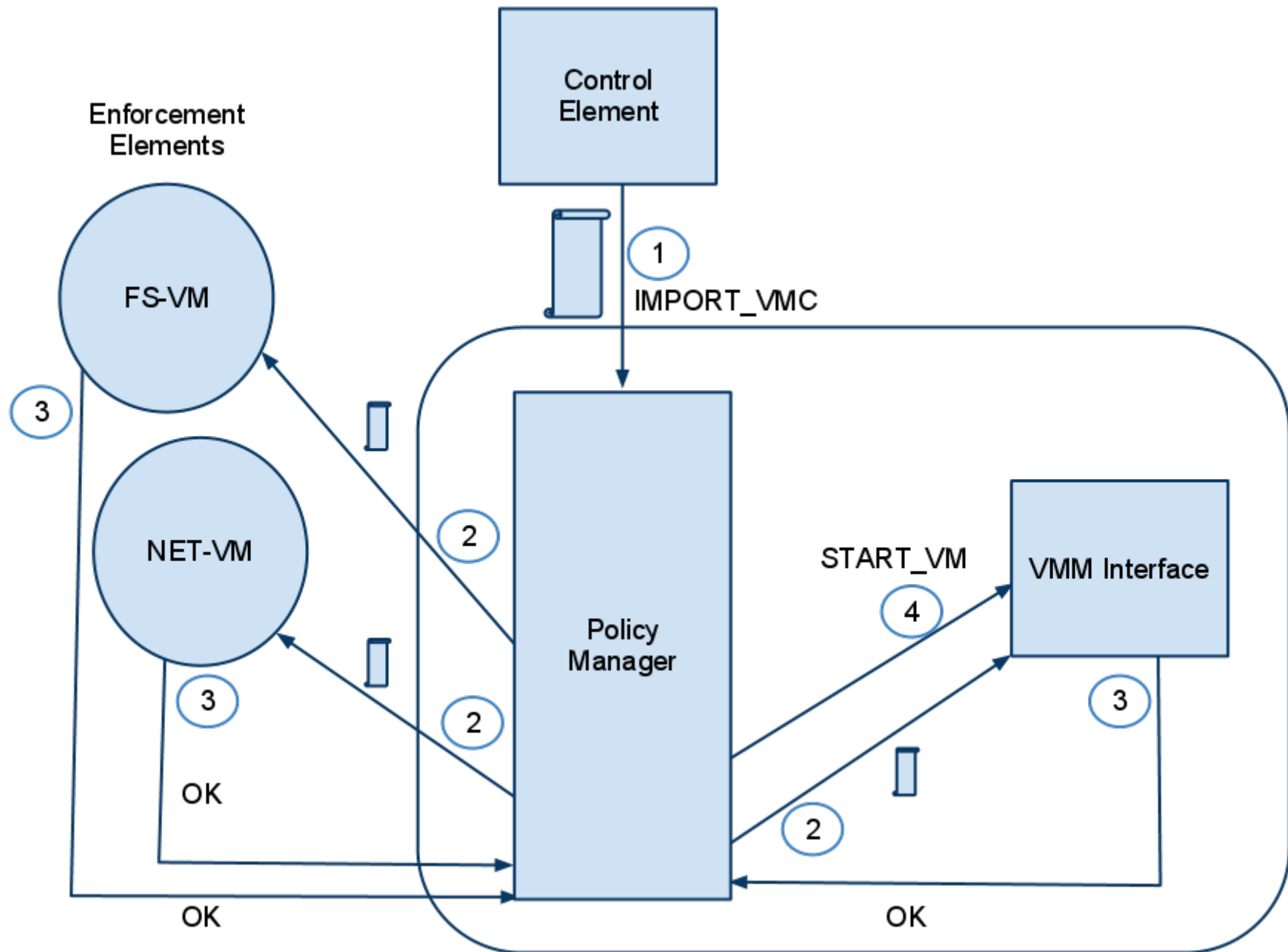
# Implementation: Contract Import



# Implementation: Contract Import



# Implementation: Contract Import



# Evaluation

# Evaluation: Overview

- Performance overhead due to virtualization
  - Disk
  - Network
  - Enforcement
    - FS-VM (Disk)
    - NET-VM (Network)
- Effectiveness
  - Protection against malware
- Recovery Properties
  - Rollback and backup behavior

# Evaluation: Performance Setup

- Hardware

- Dell OptiPlex 745
- 2.4 GHz Intel Core 2 CPU 2200
- 250 GB hard drive
- 4 GB of RAM

- Software

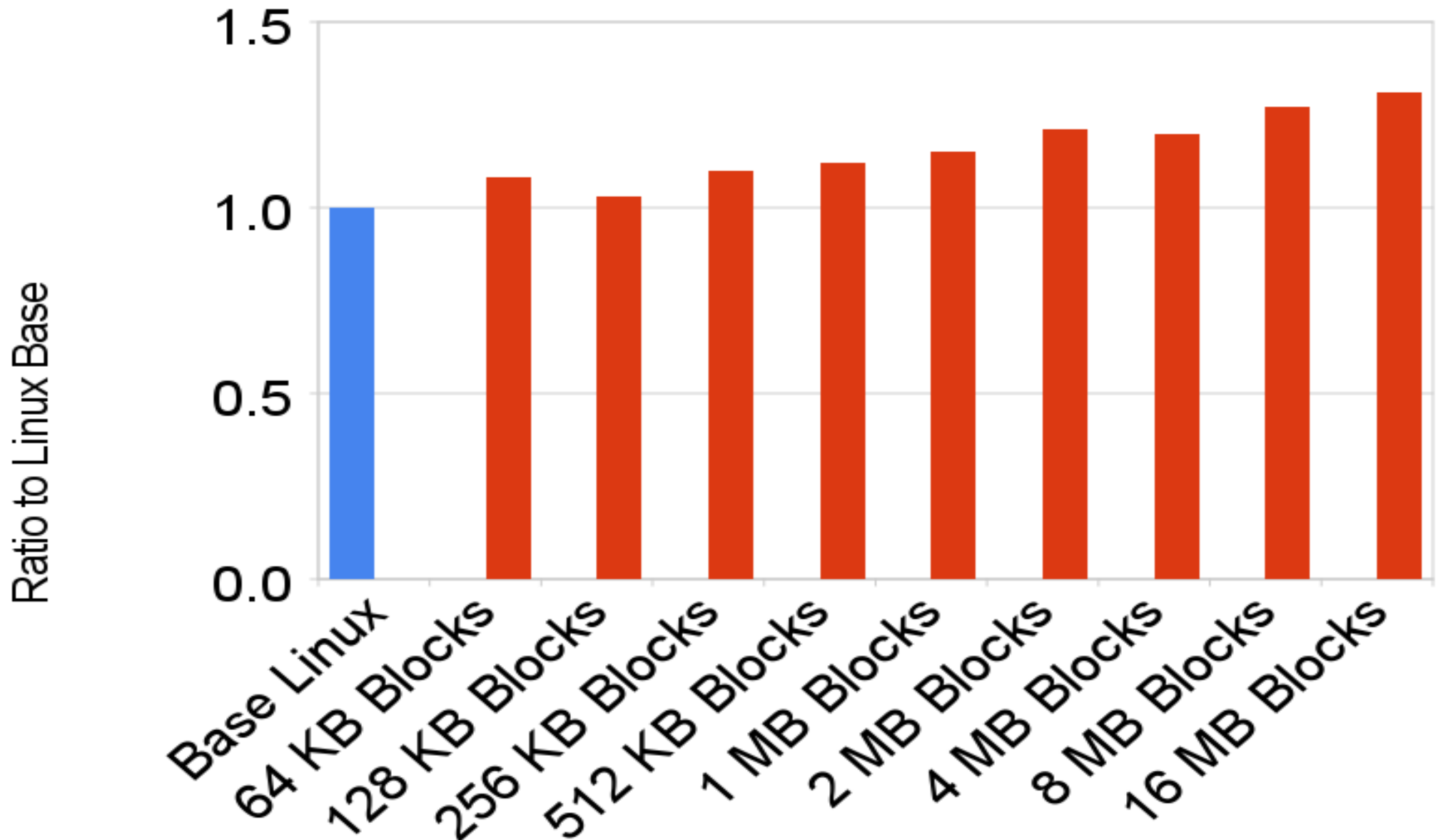
- QEMU-KVM hypervisor 0.12.3
- Fedora 13 Base
- Windows 7 and Ubuntu 10.04 Guests (both 32 bit)

# Evaluation: Disk Performance

- IOzone 3\_347
  - Compiled from source on each platform
- Tested a range of file sizes and block sizes
- Report on 8 GB file size (2x base RAM)
  - Block size 64 KB to 16 MB by powers of 2
- Base systems
  - Fedora 13, Windows 7
- Guest systems
  - Ubuntu 10.04, Windows 7

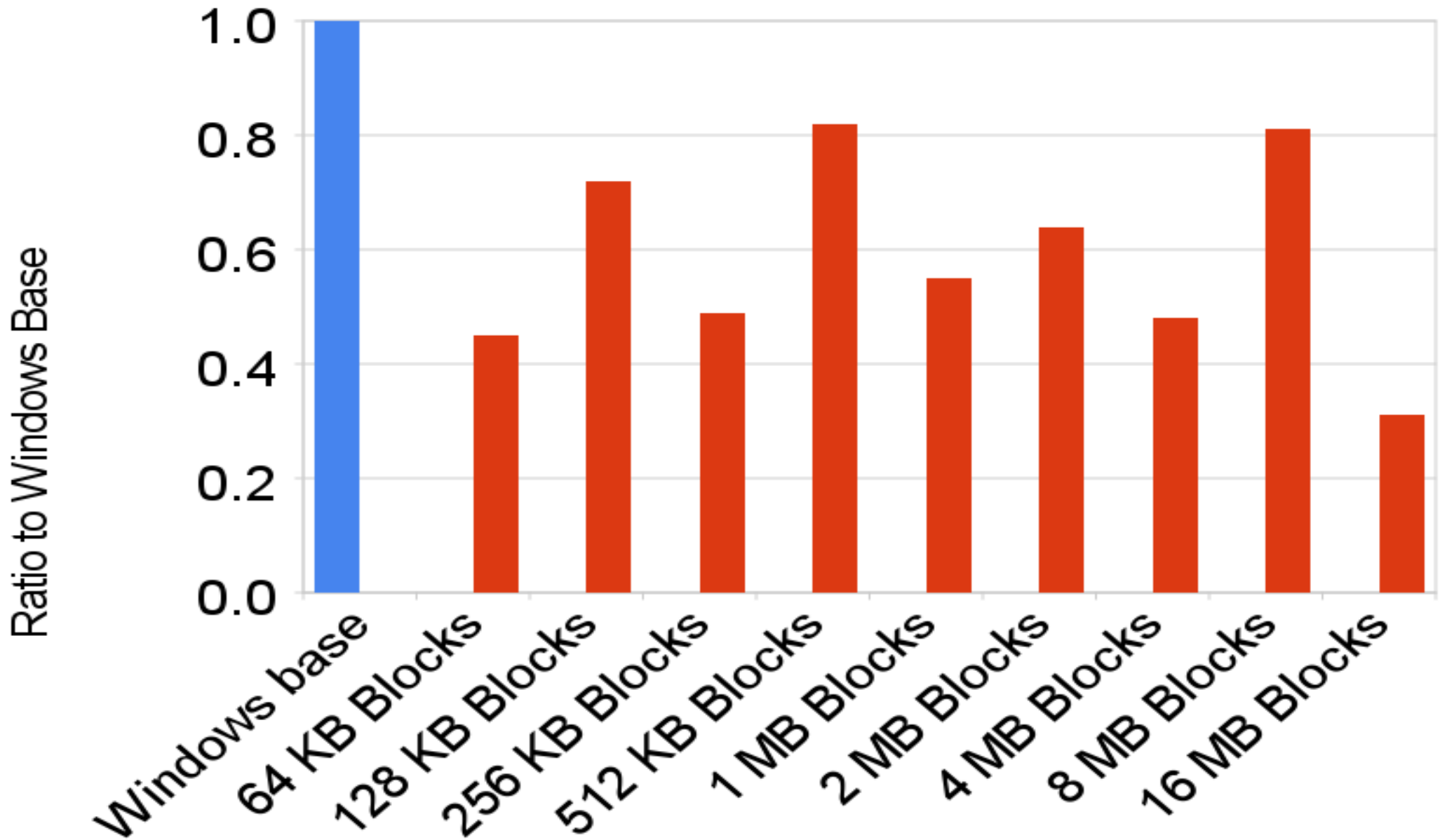
# Evaluation: Linux Guest Read

Linux Guest / Qcow2 Disk Image / Read Performance / 8 GB File



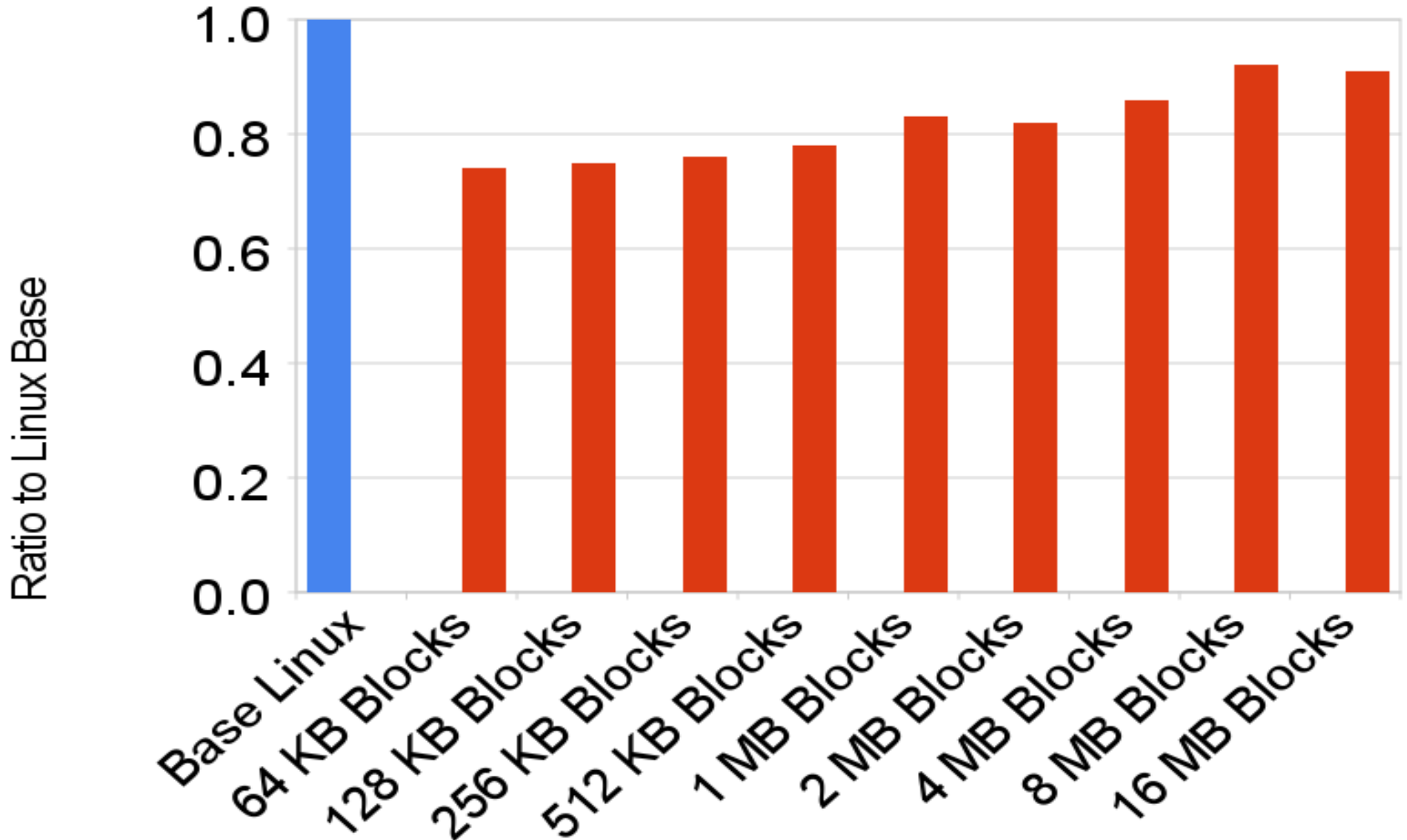
# Evaluation: Windows Guest Read

Windows Guest / Qcow2 Disk Image / Read Performance / 8 GB File



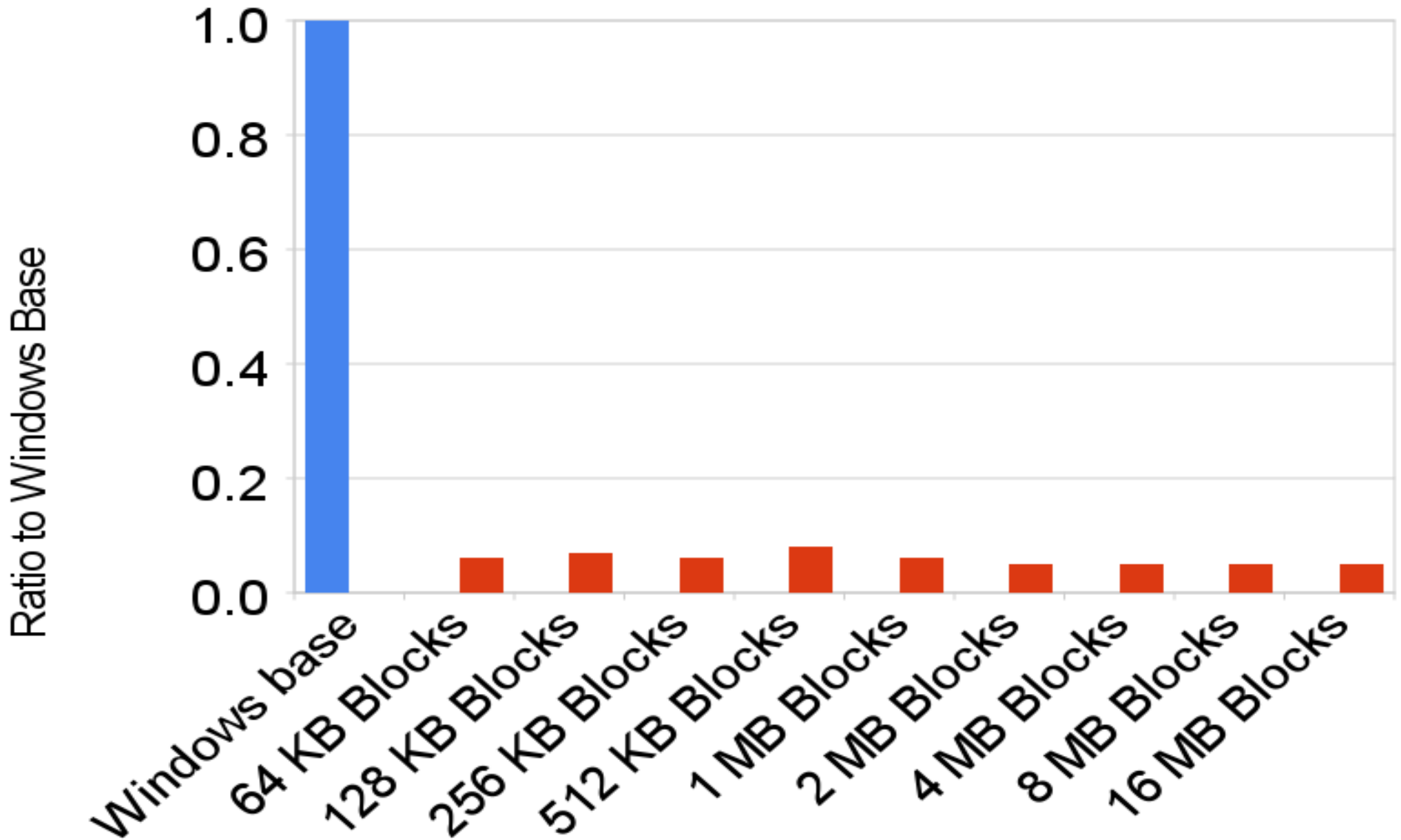
# Evaluation: Linux Guest Write

Linux Guest / Qcow2 Disk Image / Write Performance / 8 GB File



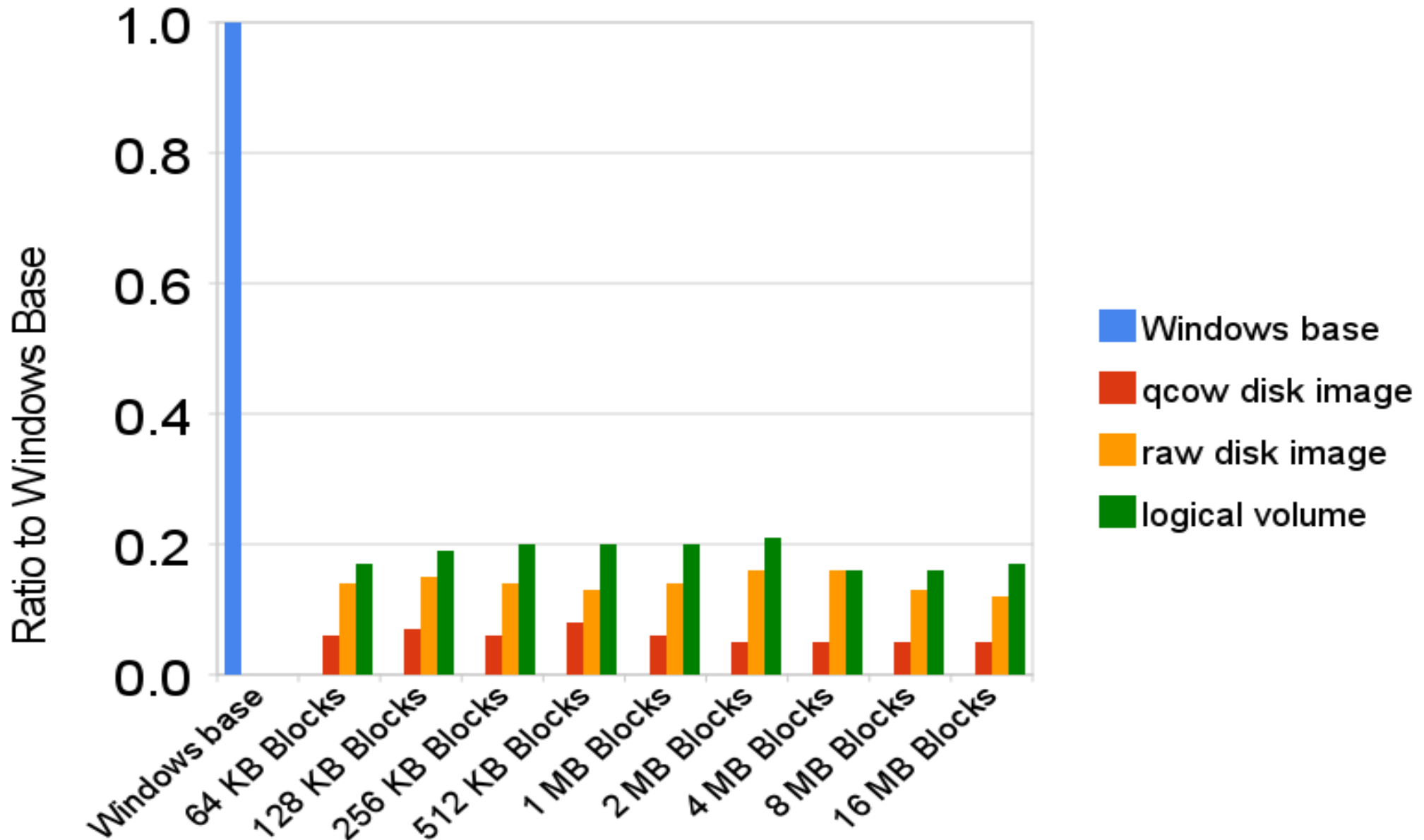
# Evaluation: Windows Guest Write

Windows Guest / Qcow2 Disk Image / Write Performance / 8 GB File



# Evaluation: Windows Guest Write

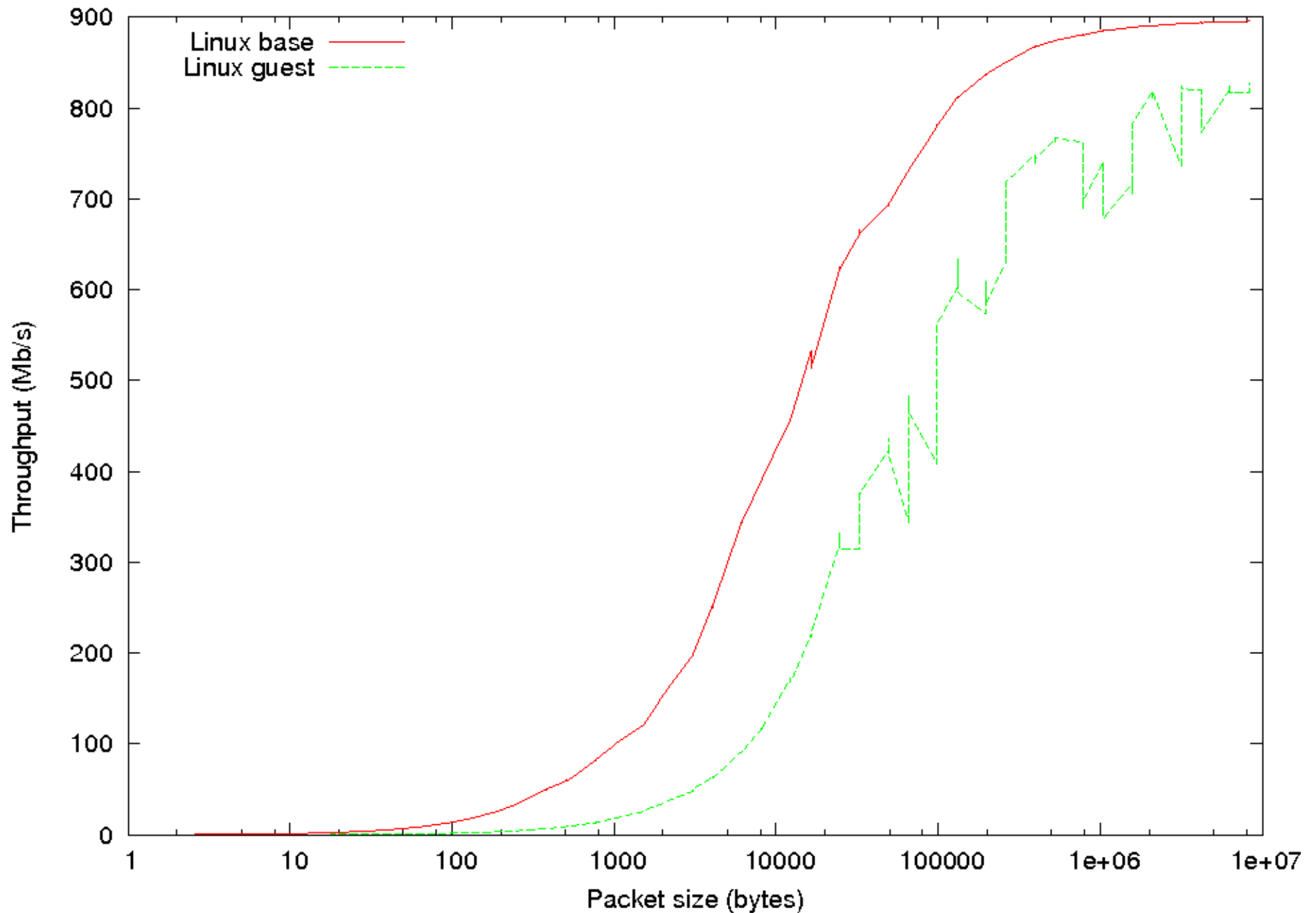
## Windows Guest Write Performance / 8GB File



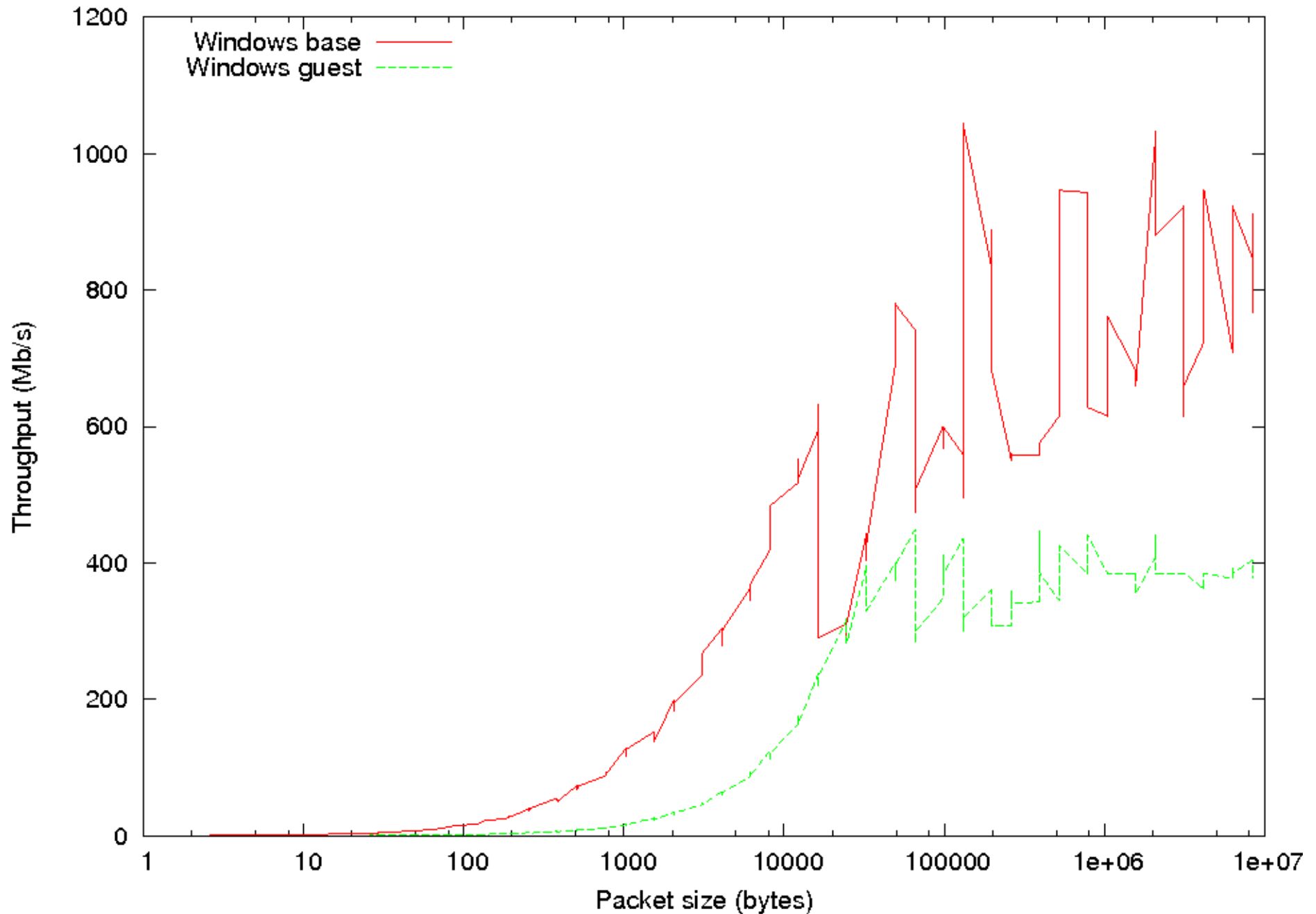
# Evaluation: Network Performance

- NetPIPE 3.7.1
  - Compiled from source on each platform
- Tested a range of packet sizes
- Report throughput
  - Packet sizes up to 10 MB
- Base systems
  - Fedora 13, Windows 7
- Guest systems
  - Ubuntu 10.04, Windows 7

# Evaluation: Linux Guest Network



# Evaluation: Windows Guest Network



# Evaluation: Performance Summary

(Disk and Network)

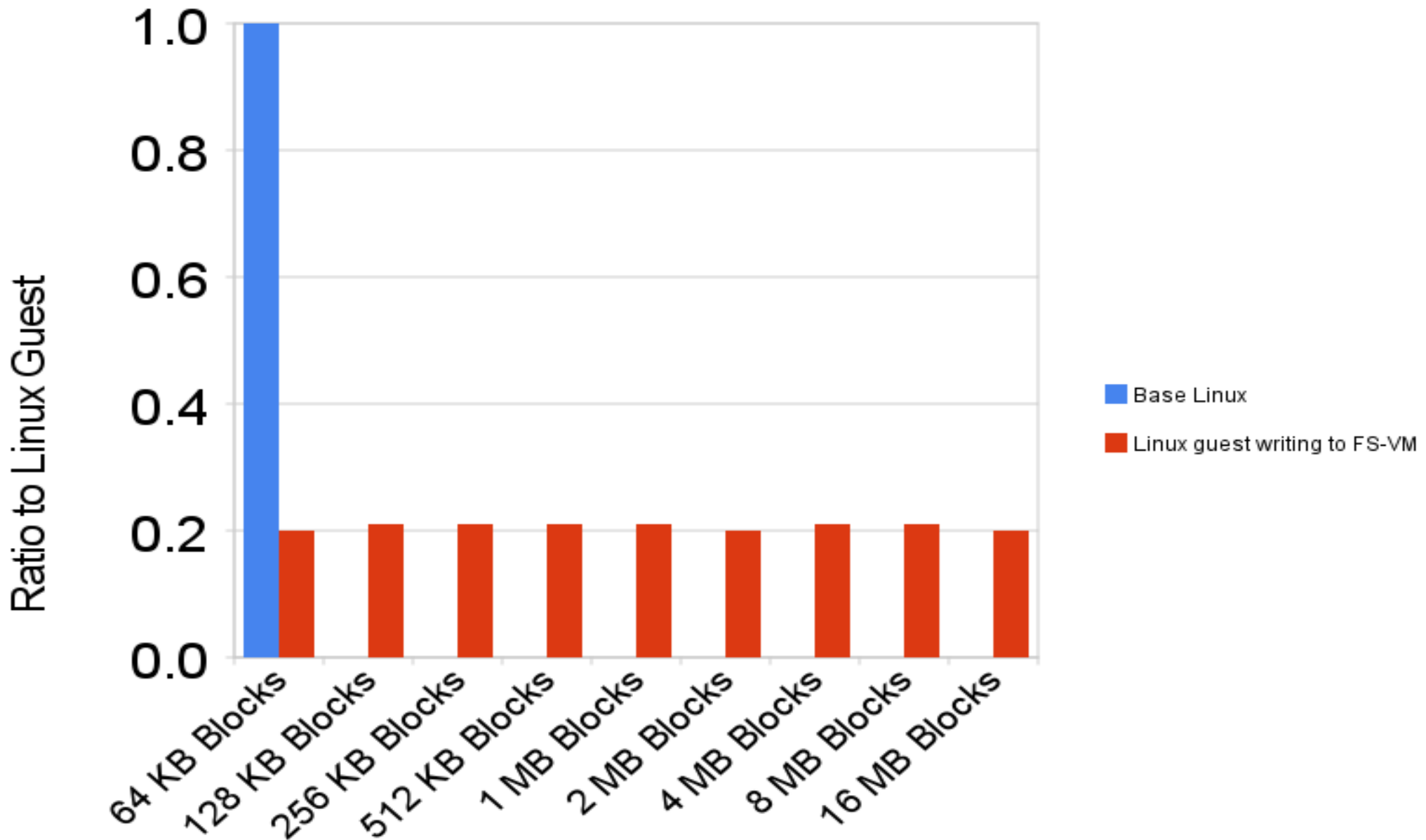
- Overhead due to virtualization is significant
  - Especially with Windows guests
  - Sensitive to guest operating system
  - Disk image backend matters
- Reasonable performance for more typical tasks
  - Web browsing
  - Watching an online video
- Performance tweaking is still possible
- Better hardware support for virtualization is available
- Xen and KVM are both improving
- Co-evolution of hardware and software

# Evaluation: Enforcement Performance

- FS-VM
  - Protects disk accesses in a file server
  - Samba version 3.4.7 on Ubuntu 10.04 Server
  - Repeat IOzone tests to FS-VM mount point
- NET-VM
  - Protects network access with flow control rules
  - Open vSwitch 1.0.1
  - Repeat NetPIPE tests with flow rules enabled

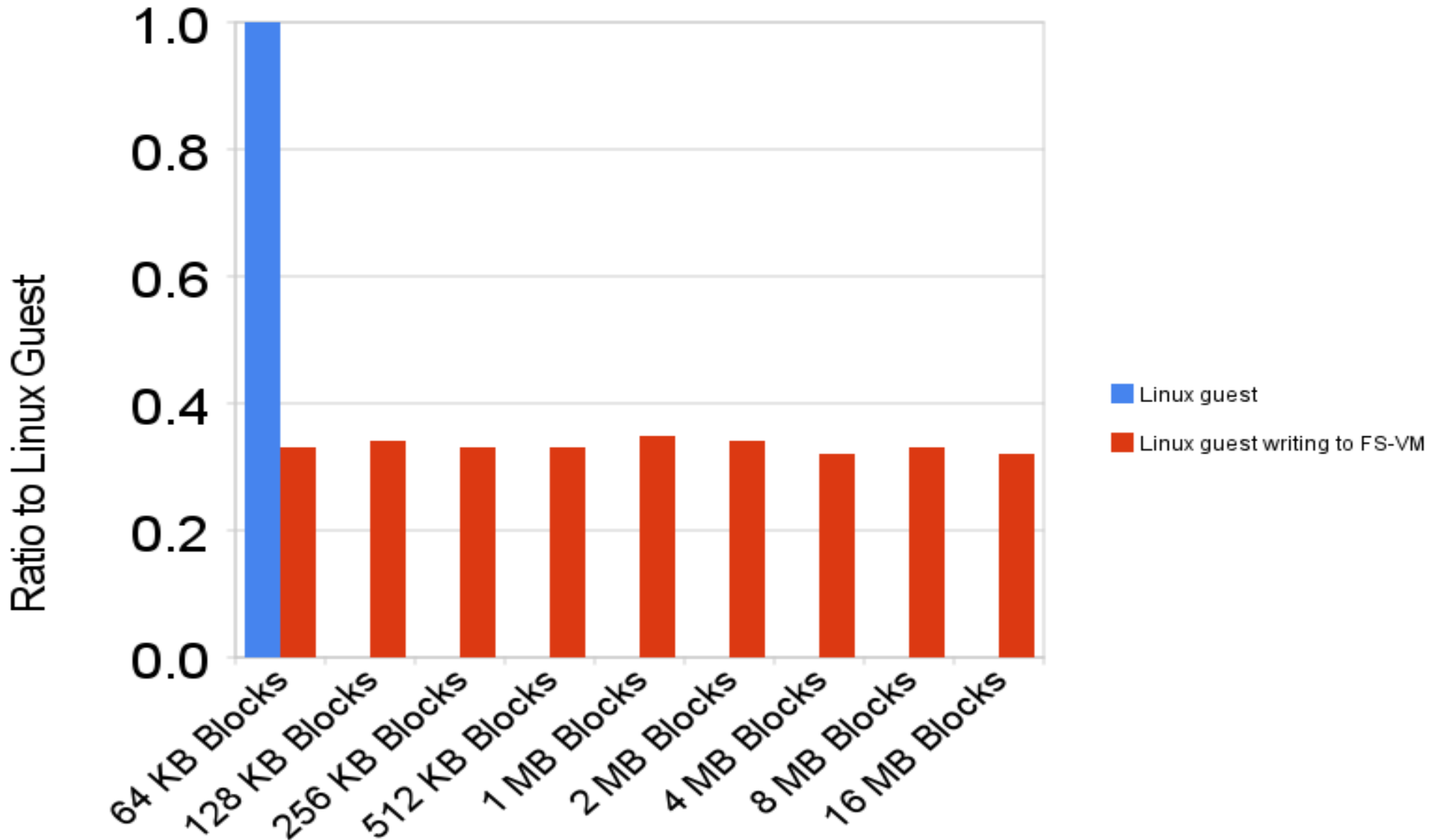
# Evaluation: FS-VM Read Enforcement

## Linux Guest Read Performance with FS-VM

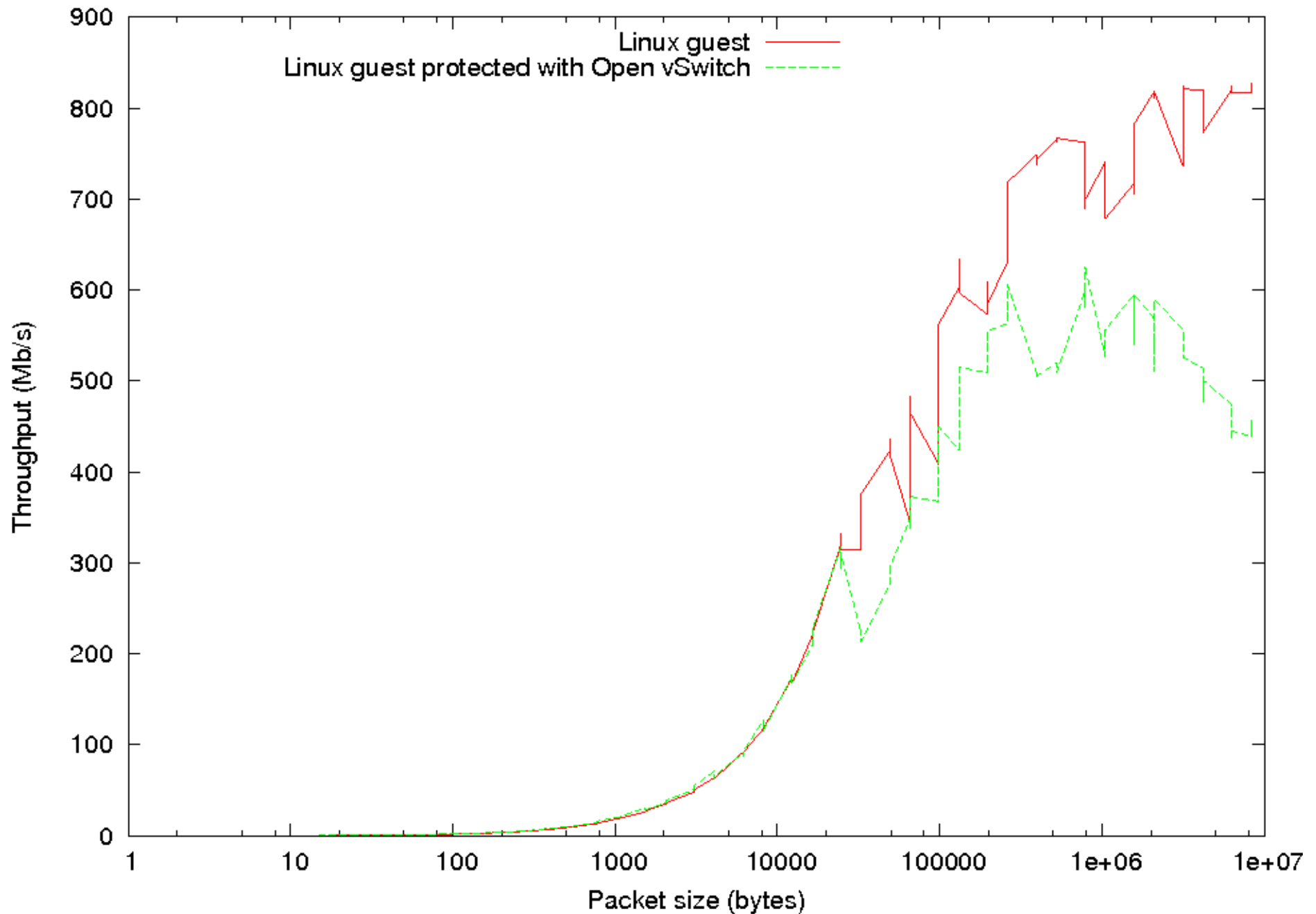


# Evaluation: FS-VM Write Enforcement

## Linux Guest Write Performance with FS-VM



# Evaluation: NET-VM Network Enforcement



# Evaluation: Performance Summary

(Enforcement)

- File system enforcement overhead is high
  - Shouldn't be a fundamental problem
    - A previous FS-VM prototype (based on NFS) performed very well (2005 Matthews, et al.)
  - Several ways to try to improve performance:
    - Change disk image backends
    - Change caching algorithms
    - Change file systems
    - Change file server
- Network enforcement overhead is reasonably low
  - Likely to improve as Open vSwitch matures

# Evaluation: Effectiveness

- Worm Taxonomy (2003 Weaver, et al.) still applies
  - Malware commonalities:
    - Target discovery
    - Distribution method
    - Activation mechanism
    - Payload

# Evaluation: Target Discovery

- Scanning
  - Port scanning blocked and limited by NET-VM
  - Incoming attacks blocked by NET-VM
- Target lists
  - Local file scans blocked and limited by FS-VM
  - External communication channels blocked and limited by NET-VM

# Evaluation: Distribution Method

- Self-carried
  - Malware could be undetected (unknown signature)
  - Malware is blocked and limited during payload
- Second channel
  - Malware is blocked and limited by NET-VM
- Embedded
  - Malware could be undetected (unknown signature)
  - Malware is blocked and limited during payload
- Manual propagation (user-triggered)
  - Malware is blocked and limited by the containing virtual appliance

# Evaluation: Activation Mechanism

- Manual human activation
  - Malware is blocked and limited by the containing virtual appliance
- Human activity-based
  - Restart and login occur less frequently
  - Use copy-on-write disk image and rollback the disk
- Self activation
  - Malware is blocked and limited by the containing virtual appliance

# Evaluation: Payload

- None/non-functional
  - Example: Morris Worm & Slammer were disruptive
  - Malware is contained by virtual appliance contract
- Internet remote control, spam relays, proxy servers
  - Malware is blocked and limited by NET-VM
- Internet denial of service
  - Malware is contained by virtual appliance contract
  - Malware is blocked and limited by NET-VM
- Data collection, data damage
  - Malware is blocked and limited by FS-VM

# Evaluation: Recovery Properties

- The Rapid Recovery Desktop system saves known good checkpoints of virtual appliances
  - User data stored on FS-VM is safe during rollback
- Automatic and user-triggered checkpoint and rollback
  - System updates can be rolled back
  - User has working system quickly
  - Compromised images can be analyzed
  - Improve contracts based on analysis
- Implementation options in our system:
  - Copy-on-write disk images
  - LVM snapshots

# Conclusions and Future Work

# Conclusions

- Contributions
  - A Rapid Recovery Desktop system
    - Design
    - Implementation
    - Based on open source technologies
  - Construction and integration of:
    - A file server virtual machine (FS-VM)
    - A network virtual machine (NET-VM)
    - A virtualization security framework (OSCKAR)
    - A virtual machine appliance contract system
  - Evaluation of the system

# Future Work

- Human Computer Interaction and Security (HCI-SEC)
  - Usable security
    - Consider user intent
    - Drag and drop and open dialogs
- Implementation-related improvements
  - FS-VM
    - More advanced file system (LFS, copy-on-write)
    - Finer-grained access tracking
    - Integration of Network Attached Storage (NAS), Mandatory Access Control (MAC), etc.

# Future Work

- More Implementation-related improvements
  - NET-VM
    - Consider in-VM policy and enforcement
    - Add dedicated OpenFlow controller VM (NOX)
    - Integrate with physical OpenFlow switches
  - OSCKAR
    - Consider RPC mechanisms, RESTful API
    - Port to more programming languages
    - Add support for more use cases

# Future Work

- Enforcement element additions:
  - Mandatory Access Control
  - Introspection
  - Host-based intrusion detection
  - Network-based intrusion detection
  - Anti-virus
  - System data-based file handling
    - Variation of FS-VM for system data
- Application to other environments
  - Data center, public/hybrid cloud, web, mobile, etc.

# Acknowledgements

- God
- My wife, Patty
- Family and friends
- Advisor, Jeanna Matthews
- The Applied Computer Science Labs at Clarkson
  - Clarkson Open Source Institute (COSI)
  - Internet Teaching Laboratory (ITL)
- Committee
- Reviewers
- Open source community

Questions?