Virtual WiFi: Bring Virtualization from Wired to Wireless

Lei Xia, Sanjay Kumar, Xue Yang

Praveen Gopalakrishnan, York Liu,

Sebastian Schoenberg, Xingang Guo

Northwestern University, EECS

Intel Labs, Hillsboro, OR

This work was done in Intel Labs during Xia's internship

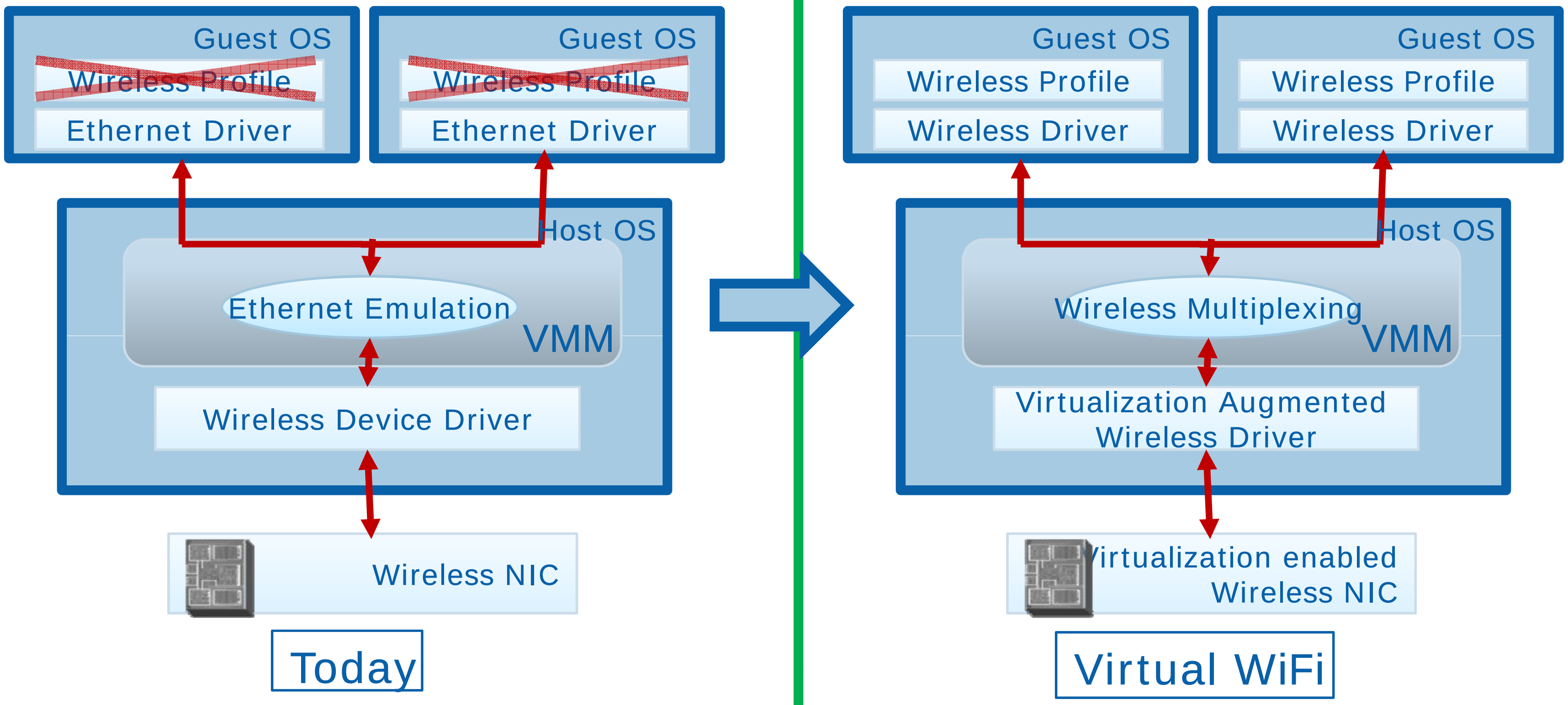
Virtual WiFi

- New virtualization approach suitable for wireless LAN virtualization
  - Full wireless LAN functionalities are supported inside VMs
  - Multiple separate wireless LAN connections are supported through one physical wireless LAN network interface

Wireless Virtualization

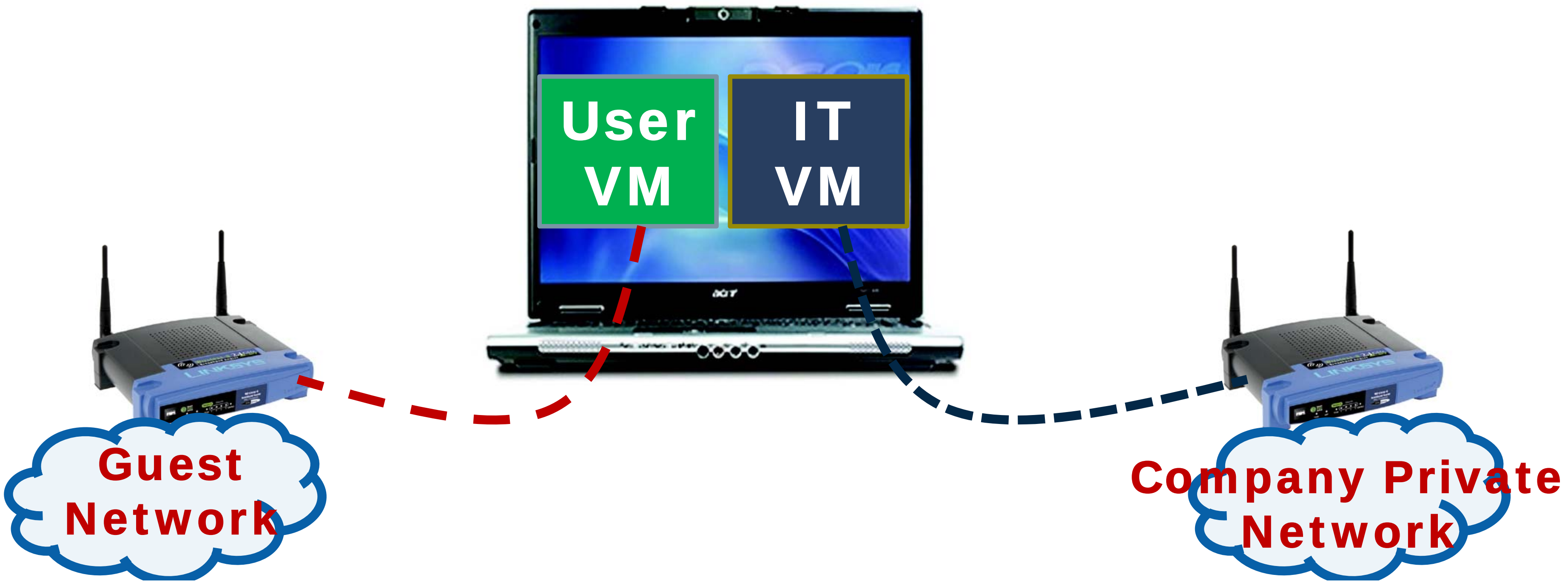
- Wireless driver stack sits in Host OS only
- VMs see only wired NIC
- Wireless functionality invisible to Guest

- Wireless driver stack runs inside the Guest as well
- Providing rich wireless functionalities to Guest



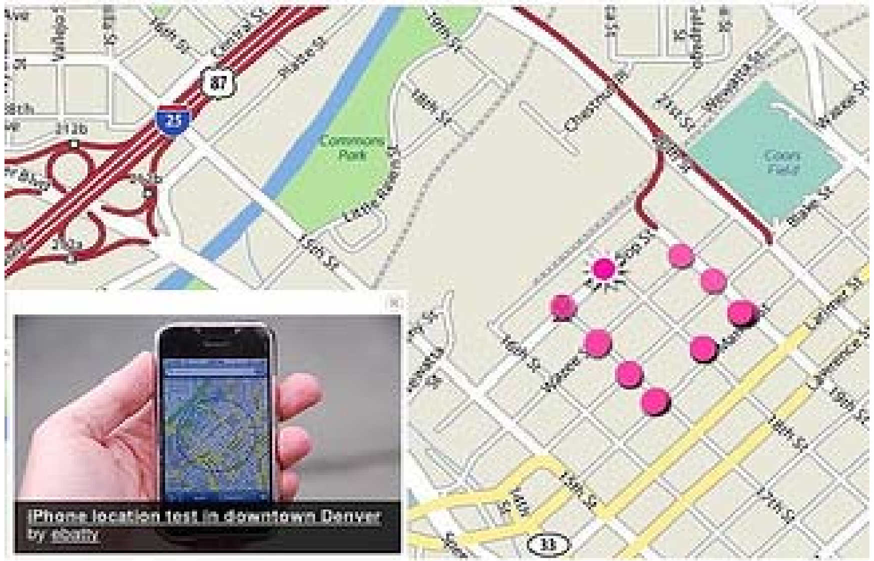
Why we need *virtual WiFi*?

- Client Virtualization
  - Enterprise IT: Separate enterprise & personal applications, data and configurations



Why we need *virtual WiFi*?

- Mobile and ultra-mobile devices
  - Separate work from play
  - User connect exclusively through wireless
  - Software tools depend on WiFi connectivity
    - **WiFi-based Location-Aware System**
      - Maps WiFi hotspots to determine location



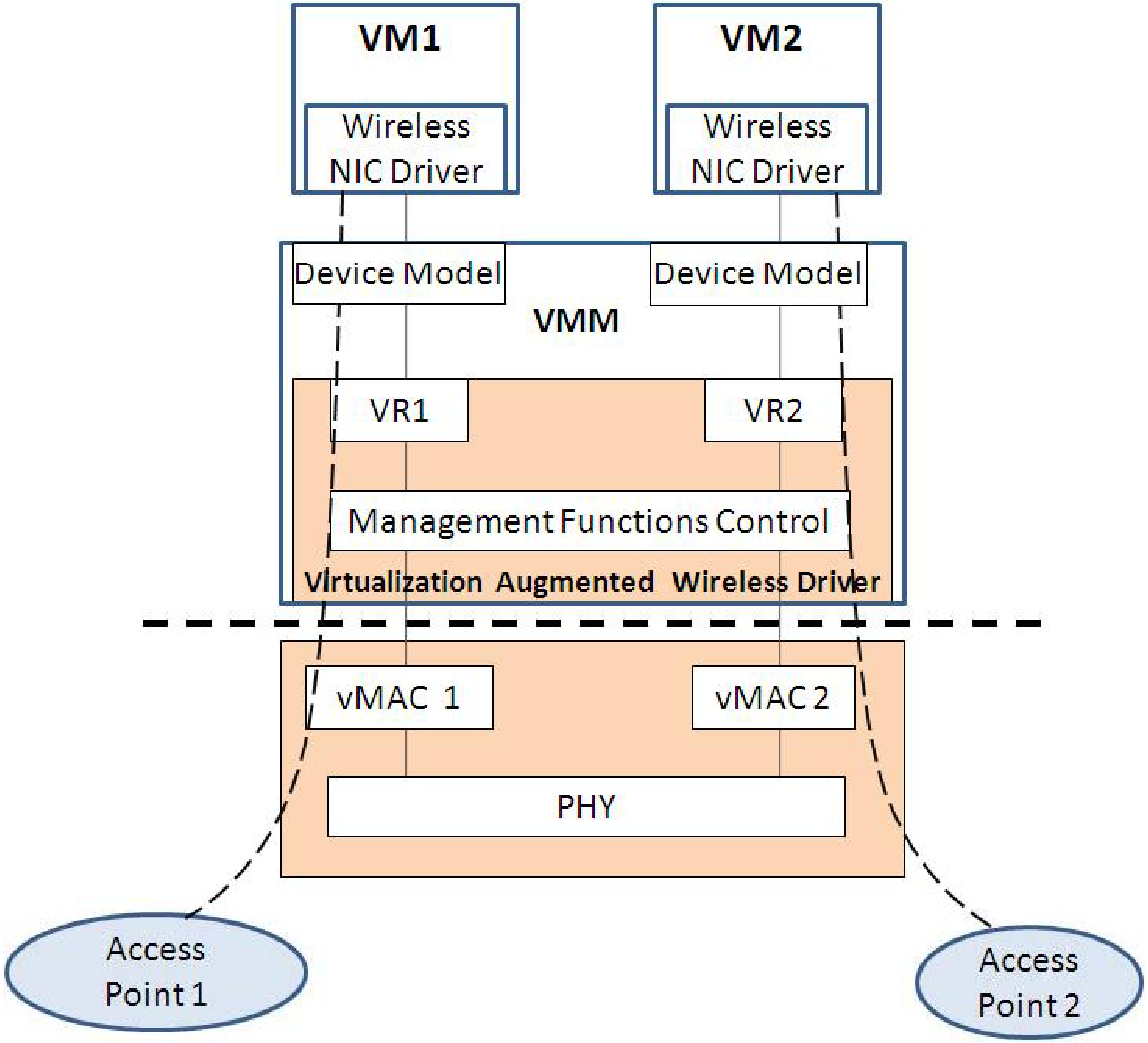
Wireless LAN Specific Features

- Complex management functions that affect the functionalities of WLAN devices.
  - Scan/Associate to specific access point
  - Rate adaption
    - Dynamic switching data rates to match the channel conditions
  - Power management
    - Device driver can control how long and how often the radio needs to be on to save battery
  - ……
- **Device driver has to be involved in many of those management decisions**

Required Hardware Support

- **Multiple virtual MACs in wireless NIC**
  - Available on most of commercial WiFi devices, For example, used in Intel MyWiFi technology
  - ***Virtual WiFi extends such technology***
    - To support wireless virtualization, multiple MAC entities maintain their independent associations with corresponding APs
    - Number of independent associations dependent on number of vMACs

Virtual WiFi: System Architecture



Virtual WiFi: Architecture

- VMs run native Intel WiFi device driver
  - Using WiFi features from device driver
  - Guest manage its own WiFi connections

Virtual WiFi: Architecture

- VMs run native Intel WiFi device driver
- Device Model exposes same virtual WiFi device to VM as physical device
  - Commands from guest can be pass to physical device without translation
  - Device model (VMM) knows few about device-specific knowledge
  - Wireless vendor minimally dependent of VMM vendor

Virtual WiFi: Architecture

- VMs run native Intel WiFi device driver
- Expose same virtual WiFi device to VM as physical device
- Virtualization *Augmented* host Wireless device driver
  - Management functions of virtualized wireless interfaces
  - Logically assigns vMAC to a VM
  - Processing commands from device model
    - Forwarding them directly or with consolidation or
    - Emulated some locally
  - Forwarding receive network packet to VMs
  - Mapping table between vMAC and VM
    - Configuration/connection status/state machine for each vMAC/VM pair

Virtual WiFi: Architecture

- Address Translation
  - Commands from guest: GPA->HPA
    - Avoids extra memory copy for TX packets
    - Either Software/Hardware IOMMU
    - Enable VT-d table to support multi-domain for single device
      - Collapse multiple page tables to single address translation table in VT-d

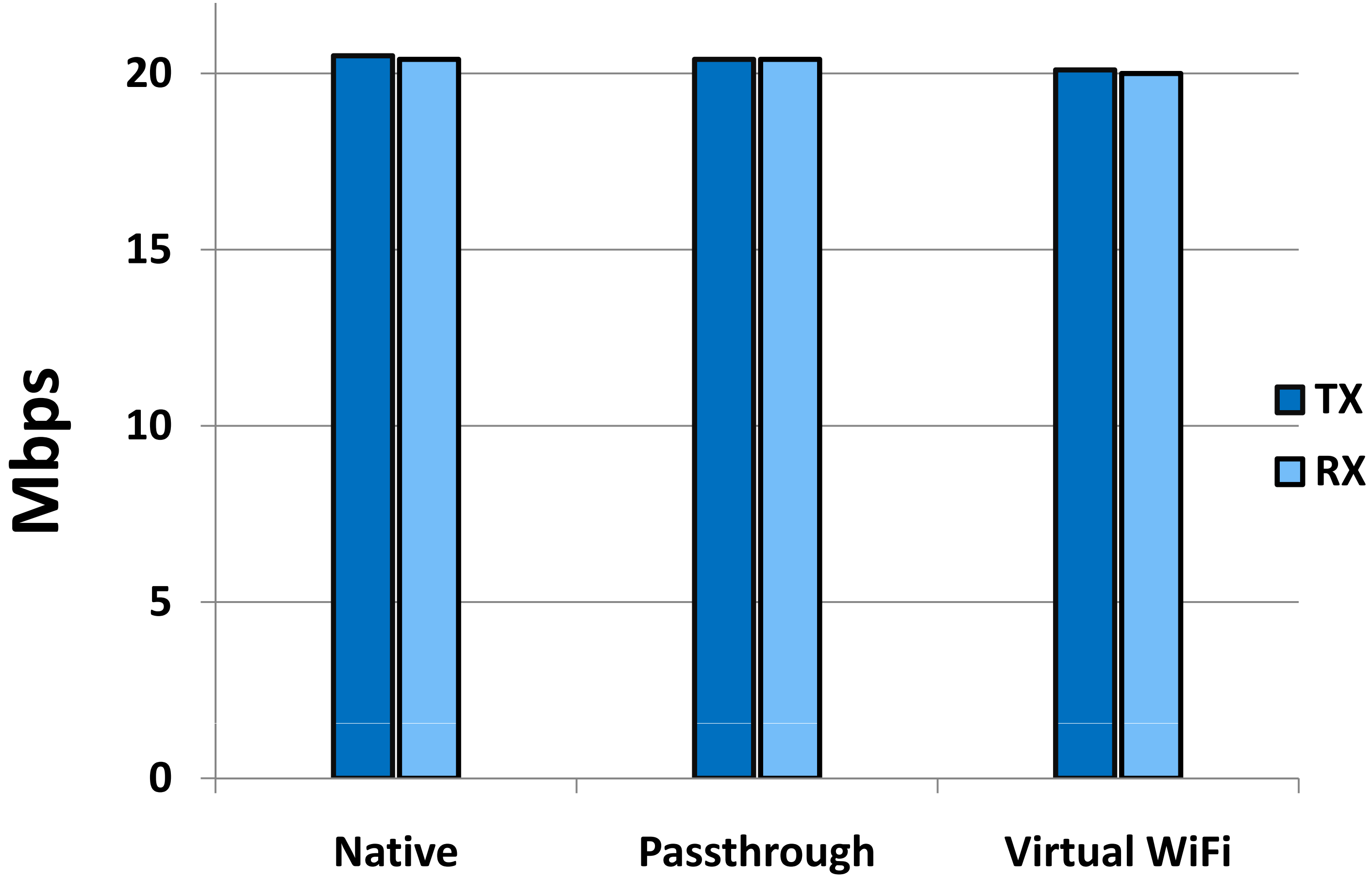
Augmented Host Device Driver Commands Handling

- TX command
  - Pass it directly to associated vMAC on WiFi NIC
- Rate Control Command
  - Only update the rate table associated with the specific VM-ID
- Device Initialization
  - Start a new vMAC, and starting state/information mapping to new vMAC
- Scan request
  - Consolidate properly of scan requests from different VMs
  - May return previous stored scan results to VMs
- And a lot more ......

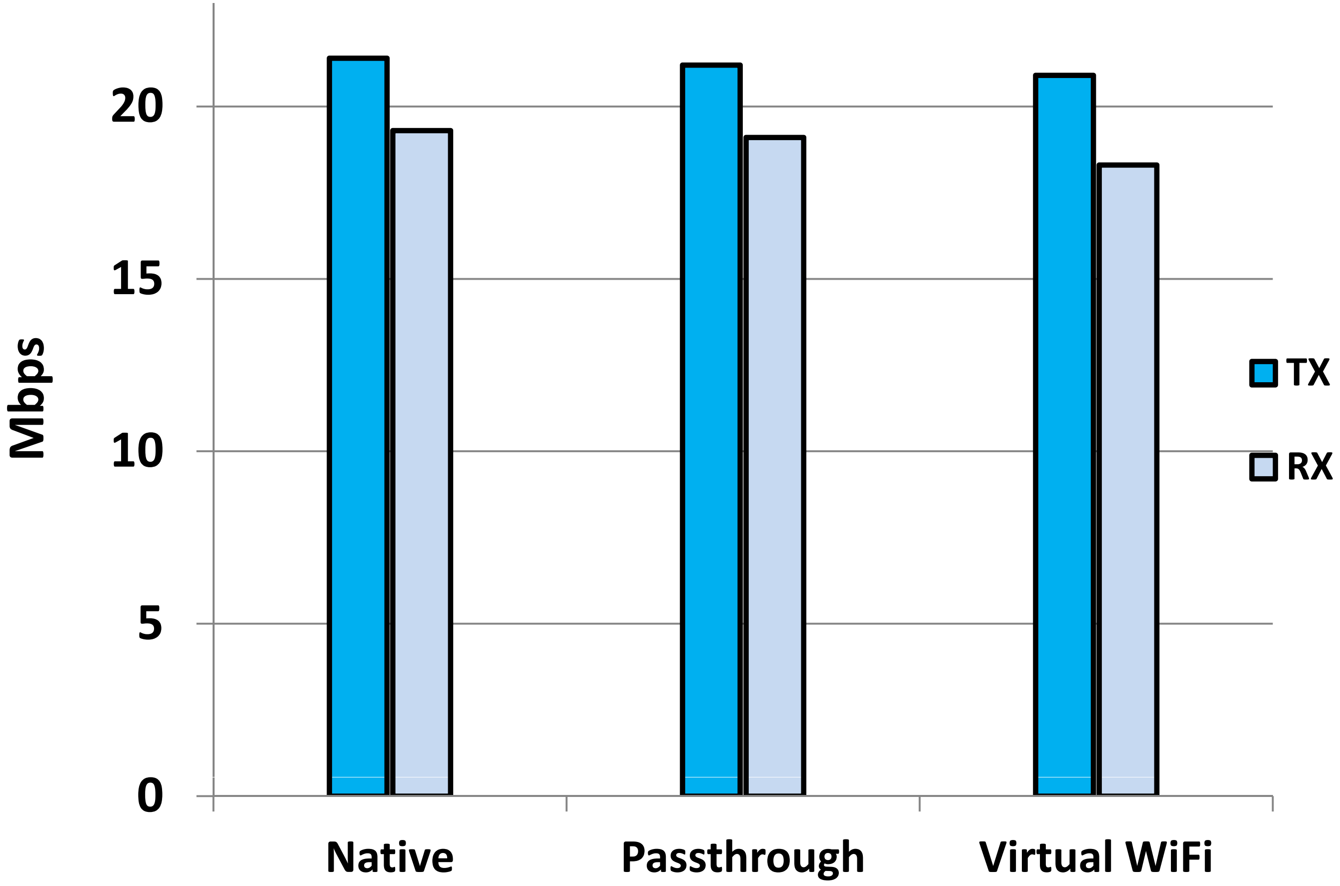
Performance

- **Benchmarks**
  - **Chariot** benchmark tool
  - *Metrics*: TCP & UDP throughputs,
  - Ping round-trip latency
- **Setup**
  - HP EliteBook 6930p Laptop with Intel Core2 Duo CPU 2.53GHz (one core used), 4GB RAM, 80GB HD
  - Intel WiFi 5300 AGN Card + Cisco WAP410N AP
  - KVM + Qemu + Linux 2.6.33.1
- **Comparing Groups**
  - *Virtual WiFi*: VM with virtual WiFi system
  - *Native*: Linux with Native WiFi driver
  - *Passthrough*: VM with direct assigned WiFi device

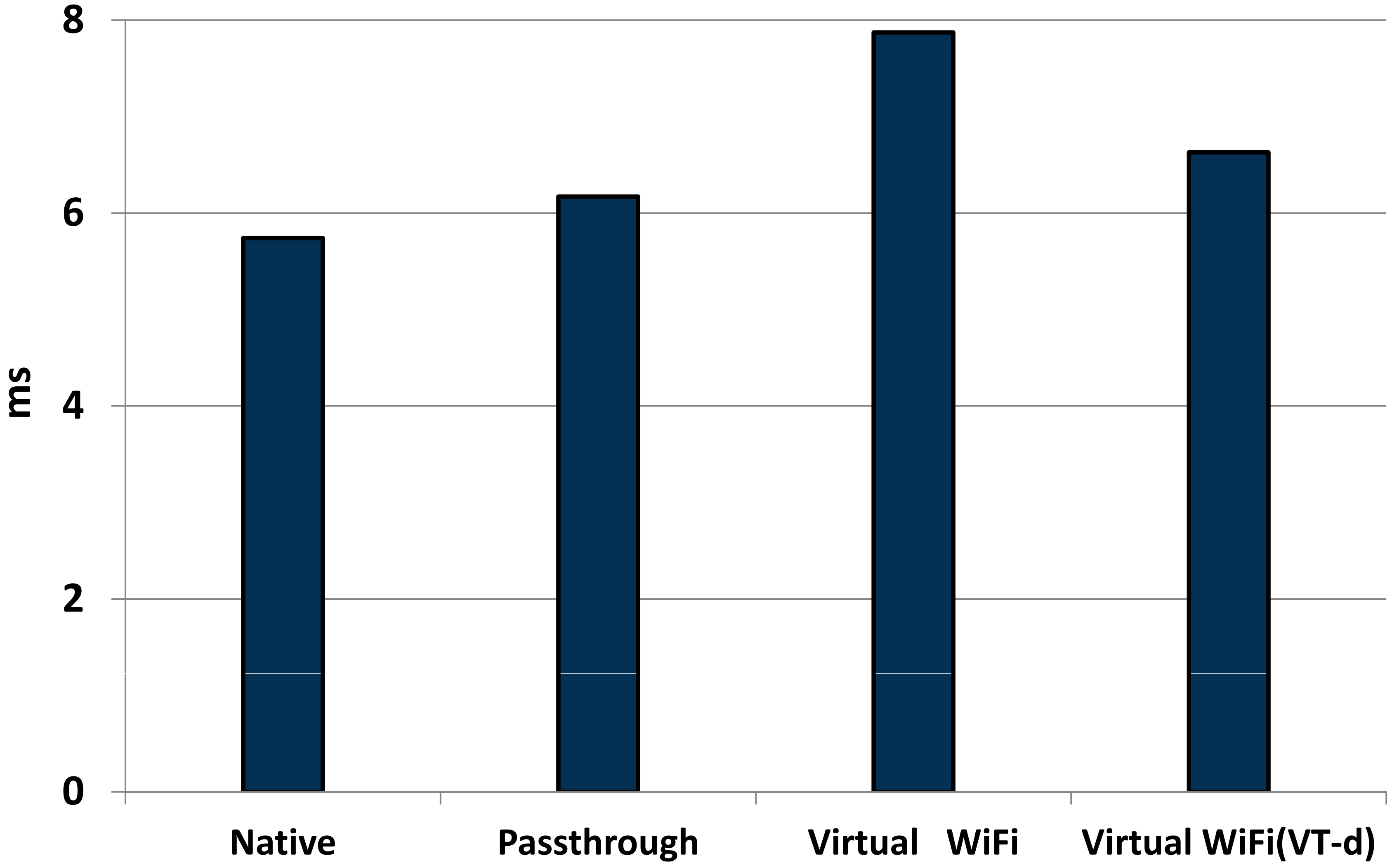
Performance – TCP Throughputs



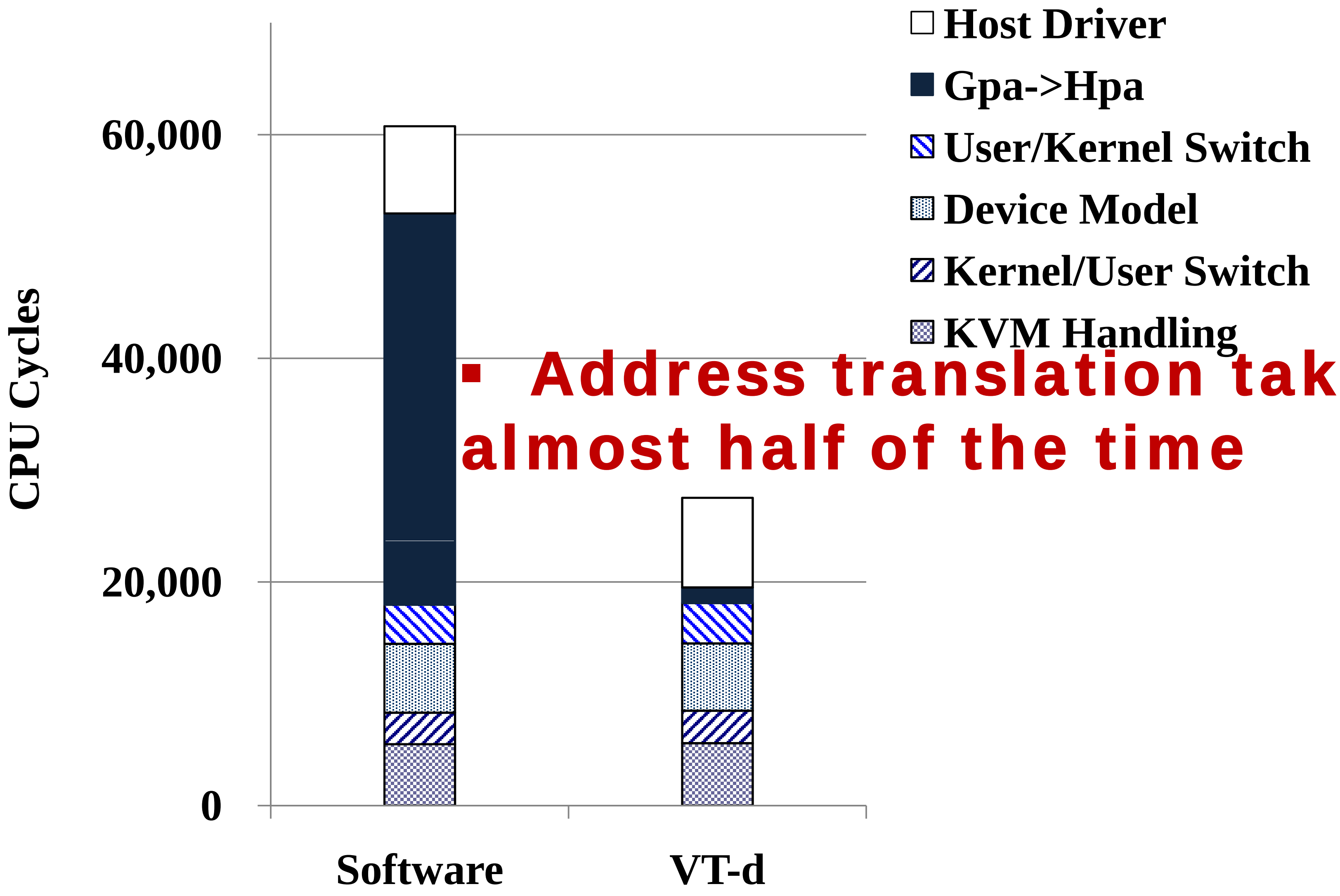
Performance - UDP Throughputs



Virtual WiFi: Performance - Latency



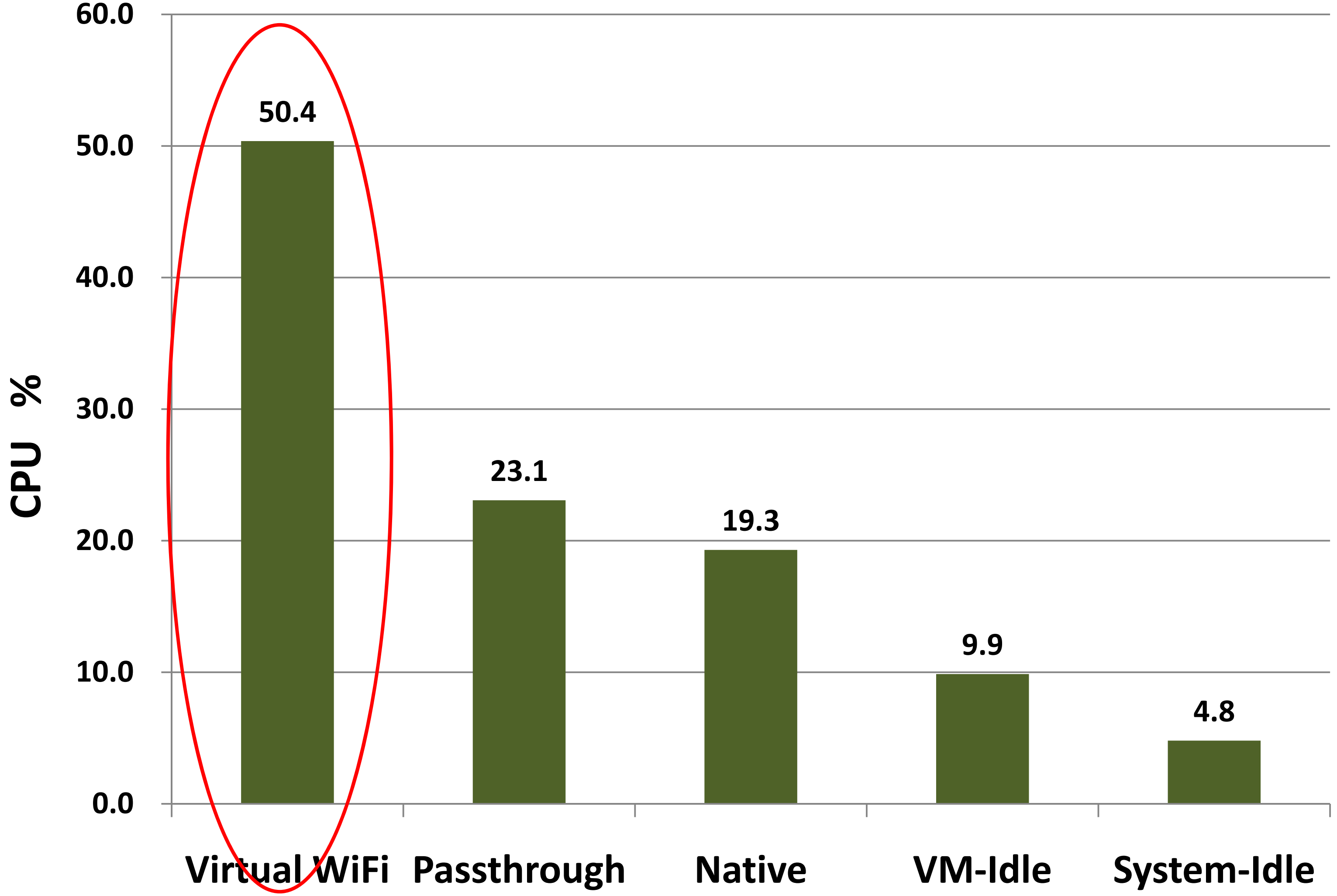
VM Additional Latency on TX-path



- Address translation takes almost half of the time

System Overall CPU Cost

(single core, 2.53GHz)



Major Virtualization Overheads

- **Address translation**
  - Solution: Hardware IOMMU
    - IOMMU hw do the address translation
    - Reduce the VM additional latency/CPU usage

Major Virtualization Overheads

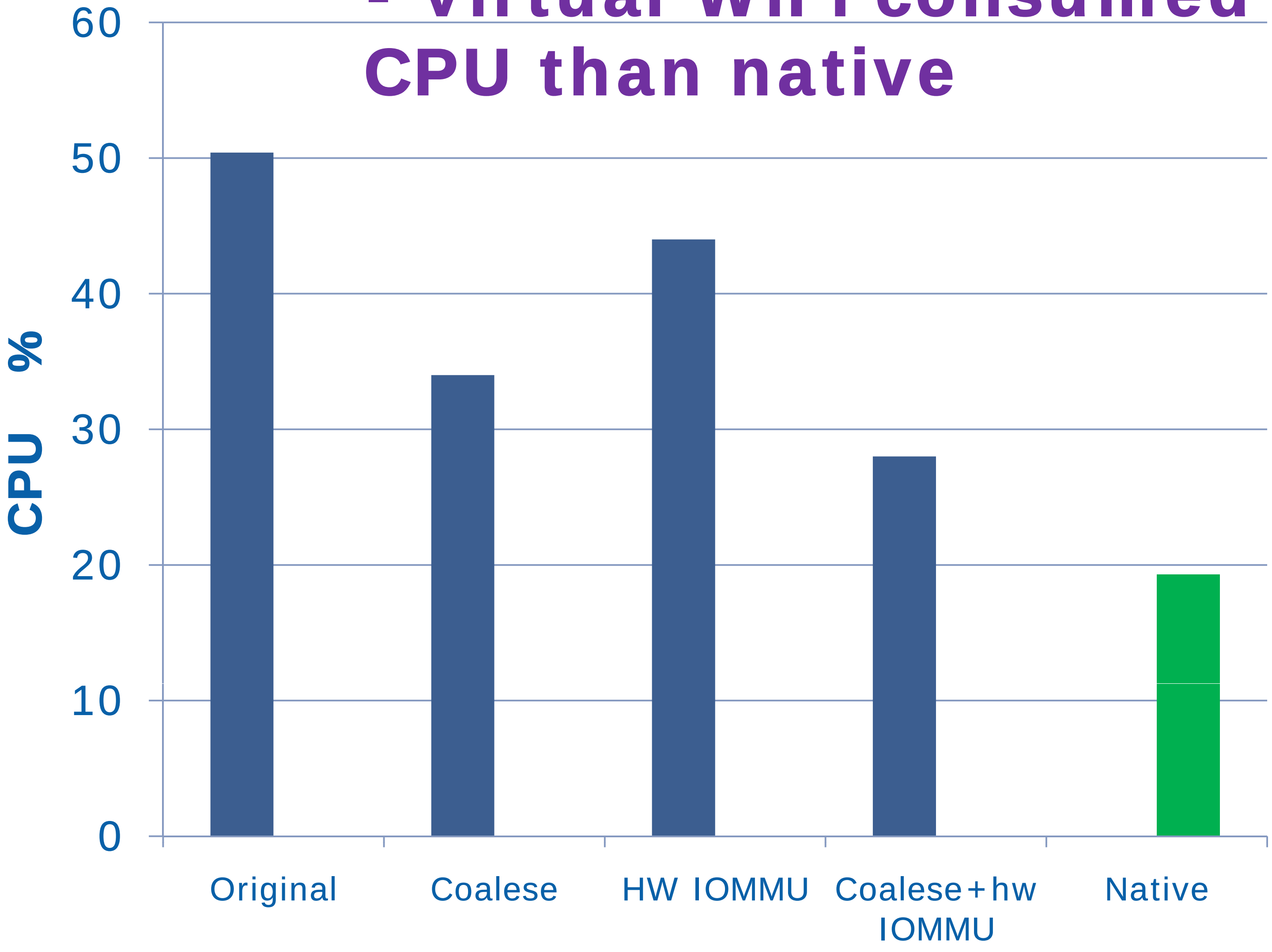
- **Address translation**
  - Solution: Hardware IOMMU
- **Interrupt Handling**
  - Coalesce interrupts disabled in host device driver
  - Each physical interrupt leads to more synchronization & signal VMs and kernel
  - Solution: Interrupt coalescing in device model

Major Virtualization Overheads

- **Address translation**
  - Solution: Hardware IOMMU
- **Interrupt Handling**
  - Solution: Interrupt coalescing in device model
- **I/O handling**
  - **MMIO handling**
    - Context switches, Threads synchronization overhead for each TX/RX packet
  - Solution: Fast data pass-through (***Future Work***)
    - Data traffic passthrough into physical device through separate queue

CPU Usage with Optimizations

- Virtual WiFi consumed 9% more CPU than native



Related Work

- **MultiNet (Microsoft vWiFi)**
    - A software layer that abstracts the wireless LAN card hardware into multiple virtual adapters
    - Continuously switch the wireless card across multiple wireless networks

- **Virtual Pass-through IO (VPIO)**
    - A modeling-based approach to high performance I/O virtualization
    - Device is directly assigned to guest
    - Most of IOs from a guest are directly applied on physical device, no VMM inventions.
    - VMM uses a behavior model to determine when IO has to be intercepted for security and device switching

Summary

- **Virtual WiFi: new virtualization approach for wireless LAN device**
  - Support fully wireless functionalities inside VMs
  - Separate wireless connections among VMs through one physical wireless interface
- **Prototype system using virtual WiFi**
  - Native throughputs with 7% extra latency
  - Less than 9% more CPU cost

Lei Xia Ph.D candidate, Northwestern University

http://www.cs.northwestern.edu/~lxi990

http://v3vee.org

Backup Slides

Backup Slides

Backup Slides

Current Wireless virtualization

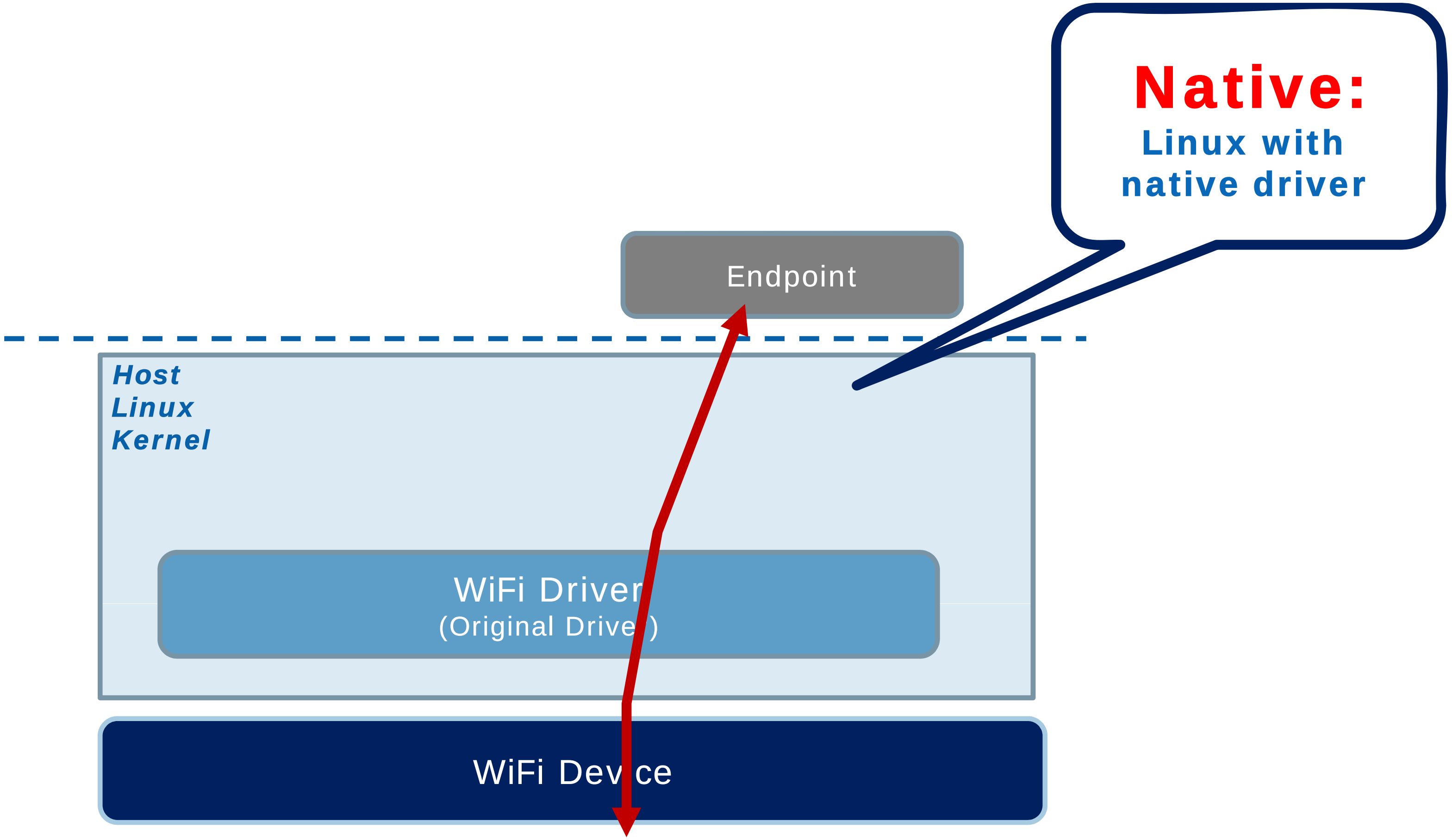
- Map network connections to virtual wired Ethernet device
    - Works well for data transfer
    - But downsides for wireless connection
        - Feature of network infrastructure can not be controlled from inside VM
        - Wireless NIC has to be configured and managed by VMM

System profiling setup

- Profiling Components for virtual WiFi system
  - Kernel: App-specific, kernel-general
  - Kernel modules: KVM, driver
  - Application-level: Endpoint, Qemu, Guest

- Presented test results based on KVM/QEMU; similar evaluations need to be performed for other VMM software, such as Xen.

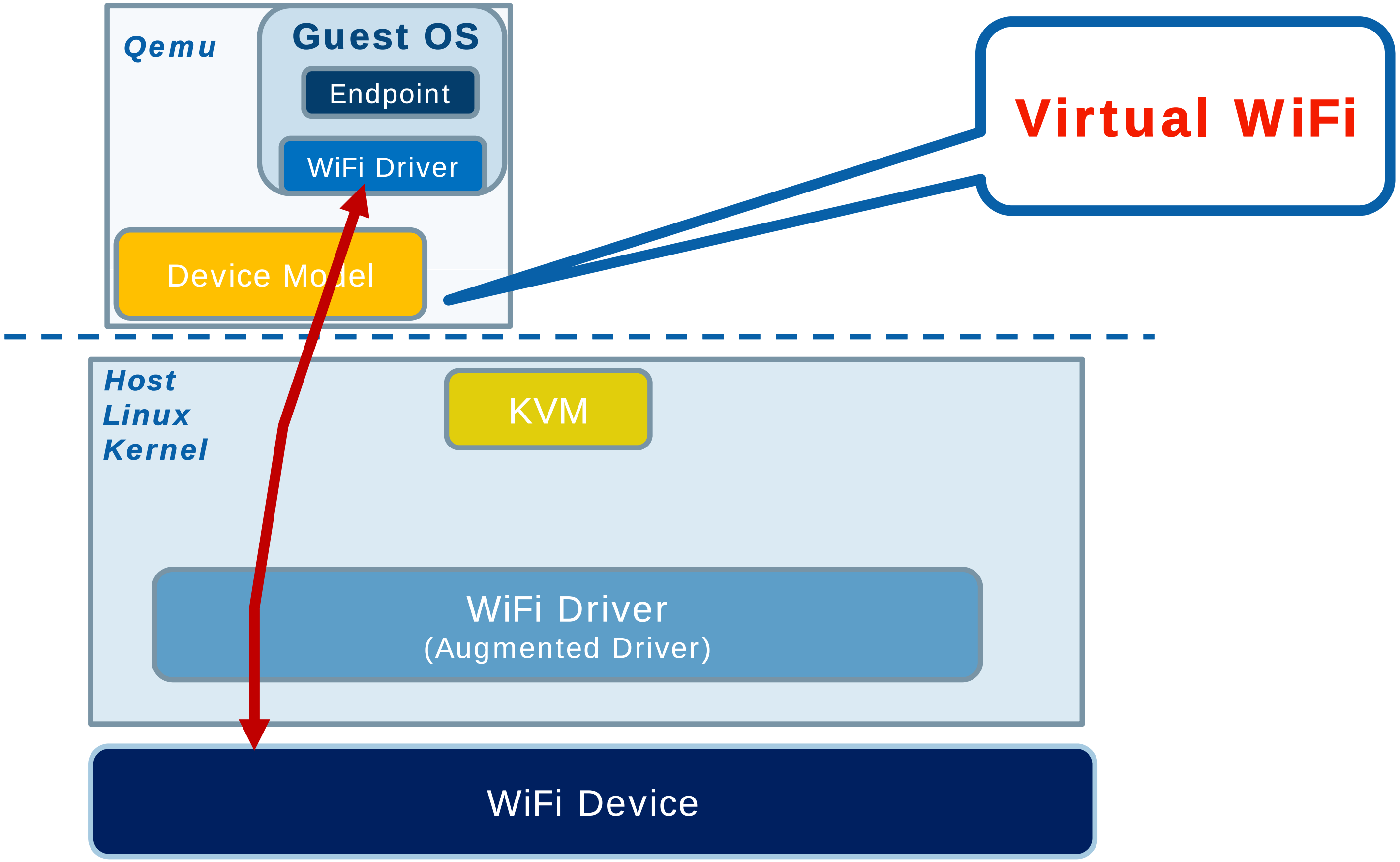
Comparing Groups



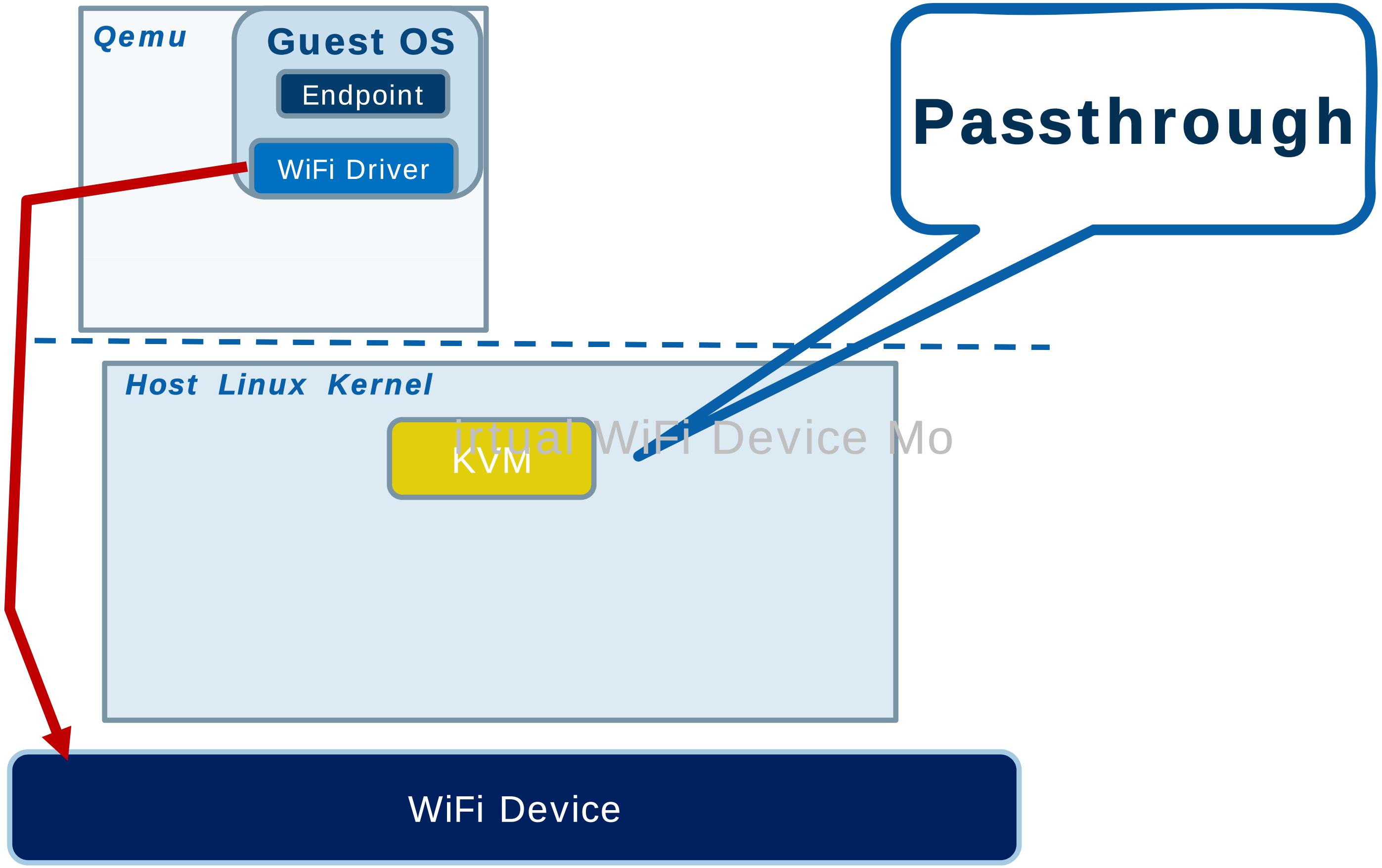
Native:

Linux with native driver

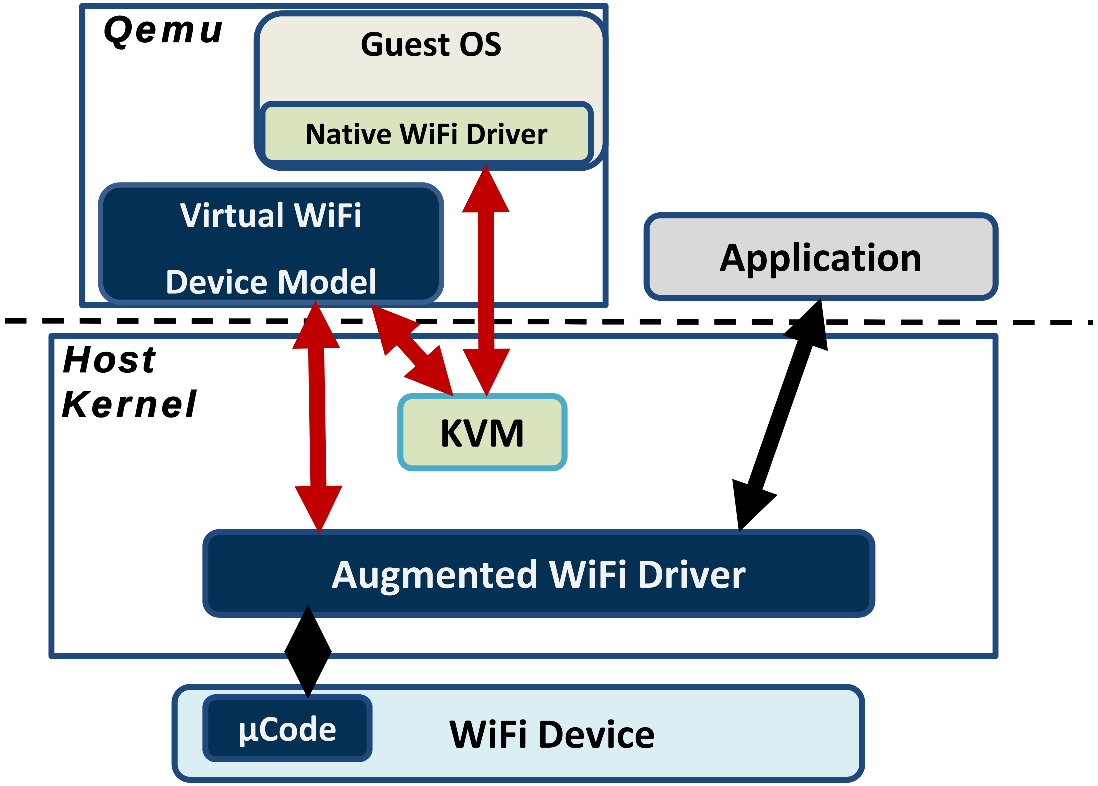
Comparing Groups



Comparing Groups



Virtual WiFi: Implementation



- Type II hosted VMM
- Can be easily ported to Type I bare metal VMM

Implementation

- Virtual WiFi Device Model
  - Expose only PCI config and MMIO mapping
  - Tag command with VM-ID, Injecting virtual interrupt to VMs.

Implementation

- Virtual WiFi Device Model
- Augmented WiFi Device driver
  - Forwarding commands directly to physical WiFi device, or Emulated some locally
  - Receive network packet from WiFi interface, Identify destination VM, signal device model

Implementation

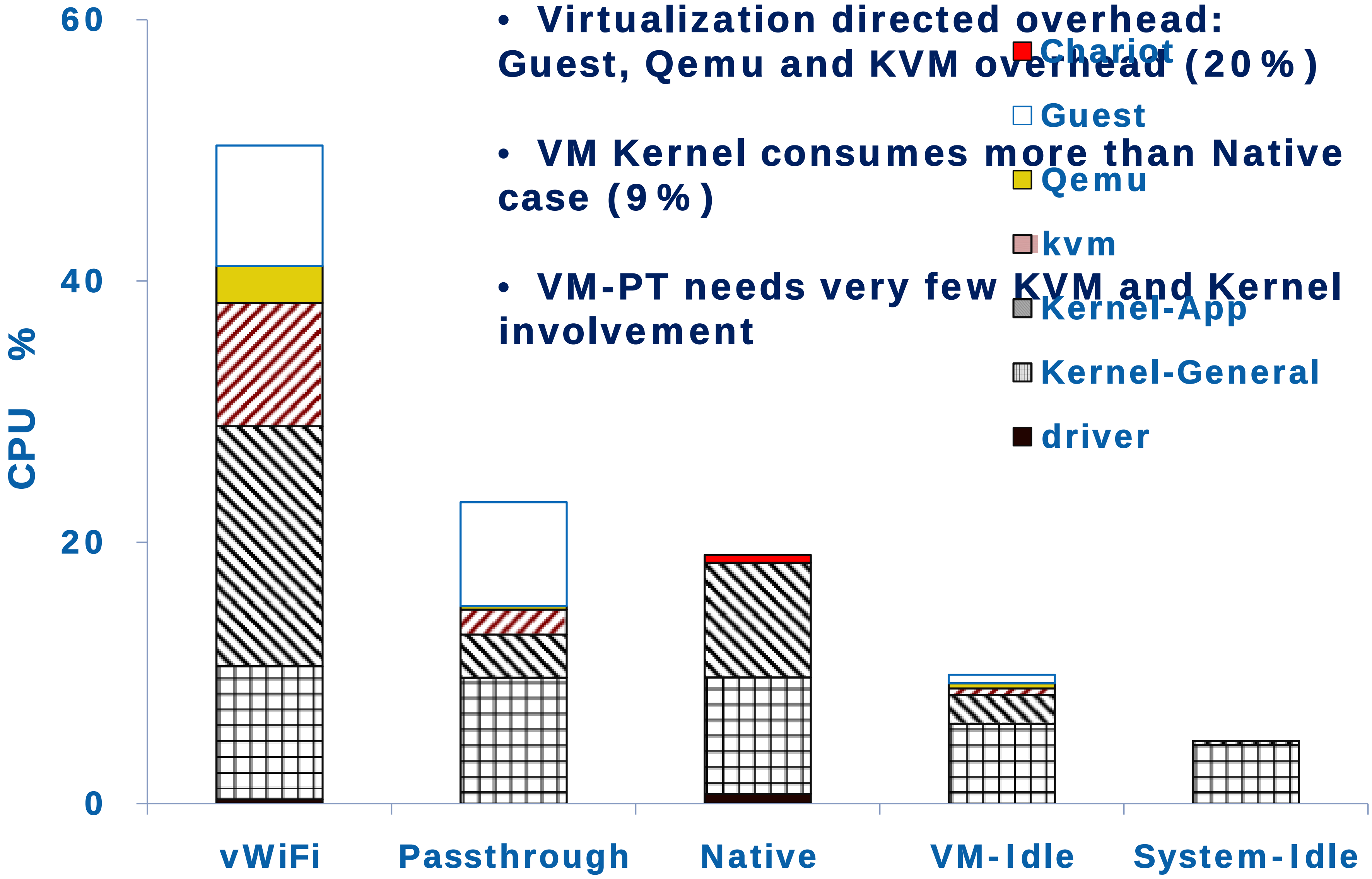
- Vdel
- Augmented WiFi Device driver
- Augmented NIC
  - Only uCode update needed
  - Virtualization extension added to uCode

The CPU Usage Matters!

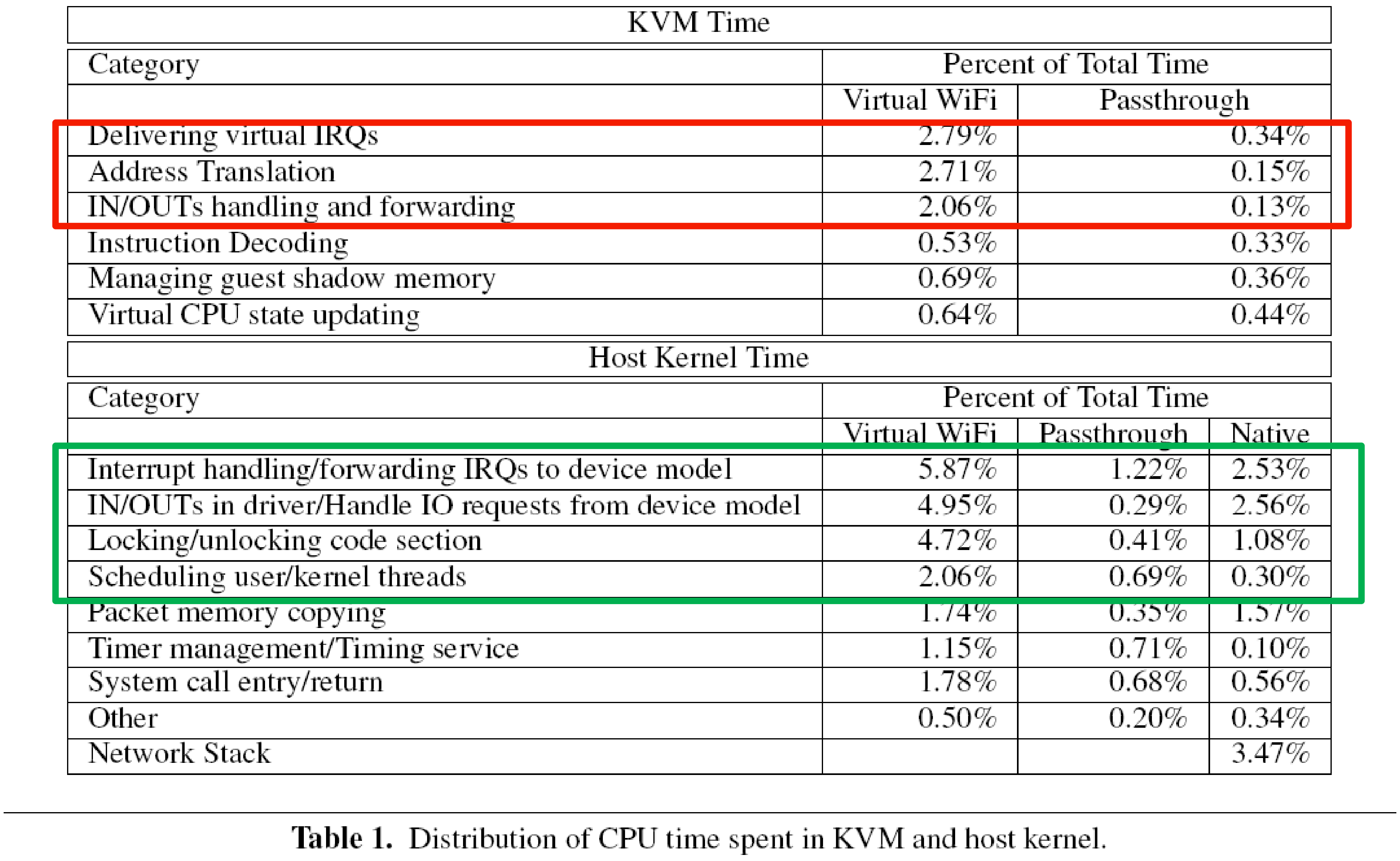
- **Scalability**
  - From 802.11g (50Mbps) to 802.11n (up to 500Mbps)
  - CPU usage grows with throughput
- **Mobile platform**
  - Limited processor resources
- **User experience**

CPU Usage breakdown (Chariot TX)

- Virtualization directed overhead: Guest, Qemu and KVM overhead (20%)
- VM Kernel consumes more than Native case (9%)
- VM-PT needs very few KVM and Kernel involvement



CPU Usage breakdown:
KVM and Host Kernel (by Oprofile)



KVM Time

Category	Percent of Total Time	
	Virtual WiFi	Passthrough
Delivering virtual IRQs	2.79%	0.34%
Address Translation	2.71%	0.15%
IN/OUTs handling and forwarding	2.06%	0.13%
Instruction Decoding	0.53%	0.33%
Managing guest shadow memory	0.69%	0.36%
Virtual CPU state updating	0.64%	0.44%

Host Kernel Time

Category	Percent of Total Time		
	Virtual WiFi	Passthrough	Native
Interrupt handling/forwarding IRQs to device model	5.87%	1.22%	2.53%
IN/OUTs in driver/Handle IO requests from device model	4.95%	0.29%	2.56%
Locking/unlocking code section	4.72%	0.41%	1.08%
Scheduling user/kernel threads	2.06%	0.69%	0.30%
Packet memory copying	1.74%	0.35%	1.57%
Timer management/Timing service	1.15%	0.71%	0.10%
System call entry/return	1.78%	0.68%	0.56%
Other	0.50%	0.20%	0.34%
Network Stack			3.47%

Table 1. Distribution of CPU time spent in KVM and host kernel.

Future Works

- **Data Pass-through**
  - Data traffic passthrough into physical device through separate queue
  - Control/management commands go through device model/augmented driver
- **Apply on next generation WiFi standards**
  - WiFi 802.11n
  - Expected throughput: ~500Mbps

Related Work

- **Full-virtualization** by emulating
  - Large performance overhead, Many development efforts, Lack of device datasheets
- **Para-virtualization**
  - Need guest modification/new para-virtualized device driver
  - WLAN device specific features are closed to VMM vendor for back-end driver
- **SR-IOV**: hardware support virtualization
  - Costly/complexity/Time line