Network Functions Virtualization on top of Xen

Joao Martins*, Mohamed Ahmed*, Felipe Huici*, Costin Raiciu §, Vladimir Olteanu §,
Michio Honda*, Roberto Bifulco*, Simon Kuenzer*

* NEC Europe Ltd., Heidelberg, Germany
§University Politehnica of Bucharest

Agenda

- Introduction & Motivation
  - Middleboxes and Network Functions Virtualization
- Introducing ClickOS
  - ClickOS
  - Click: The Click Modular Router
  - Mini-OS
  - Xen
  - ClickOS on Xen
- Optimizations
  - Network performance (to achieve 10Gb/s)
  - Click processing performance
  - Management tools (influences boot time)

Agenda

- Probing ClickOS
  - ClickOS in Comparison
  - Middlebox Application Examples and their Performance
  - Scaling out ClickOS
    - Concurrent Transmit VMs
    - Multiple NICs/VMs
- Conclusion & Future Work
- Q&A

Introduction & Motivation

The Middleboxes World

ad insertion

BRAS

WAN accelerator

carrier-grade NAT

transcoder

session border controller

IDS

load balancer

DDoS protection

firewall

QoE monitor

DPI

Middleboxes

- Middleboxes are an intrinsic and fundamental part of today‘s networks
- But come with their problems:
  - Expensive *(specialized hardware)*
  - Difficult to
    - Manage
    - Extend/Upgrade
      (Introducing new features often means deploying new hardware)
  - Cannot be scaled with demand
  - Hard for new players to enter the market

Network Function Virtualization and Middleboxes

Virtualized Software Middleboxes

- Address these issues of specialized hardware Middleboxes
- Enable a number of scenarios:
  - Replace existing expensive Middleboxes
  - Shift functionality into the cloud on demand
  - Shift functionality close to consumer *(e.g., mobile users)*
  - Scaling with demand *(e.g., instantiate and migrate with demand)*
  - Re-programm SDN‘s data plane
- Virtualization can shift software Middlebox processing to multi-tenant platforms

- Open Question:
  - Can it be built using commodity hardware while still fulfilling the high performance requirements?

Requirements of Network Function Virtualization

Requirements for such virtualized Middleboxes

- Isolation
  (e.g., multi-tenants on common hardware, feature-testing)
  - Performance
  - Security
- High Throughput, Low Delay
- Scalability
  - Quickly instantiable
    (in order to scale with demand)
- Consolidation
  - Tiny (small memory footprint)
    (in order to be able to increase number of Middleboxes running concurrently on a single server)
- Easy to deploy functionality as needed;
  → Programmable

Network Functions Virtualization on top of Xen

Introducing ClickOS

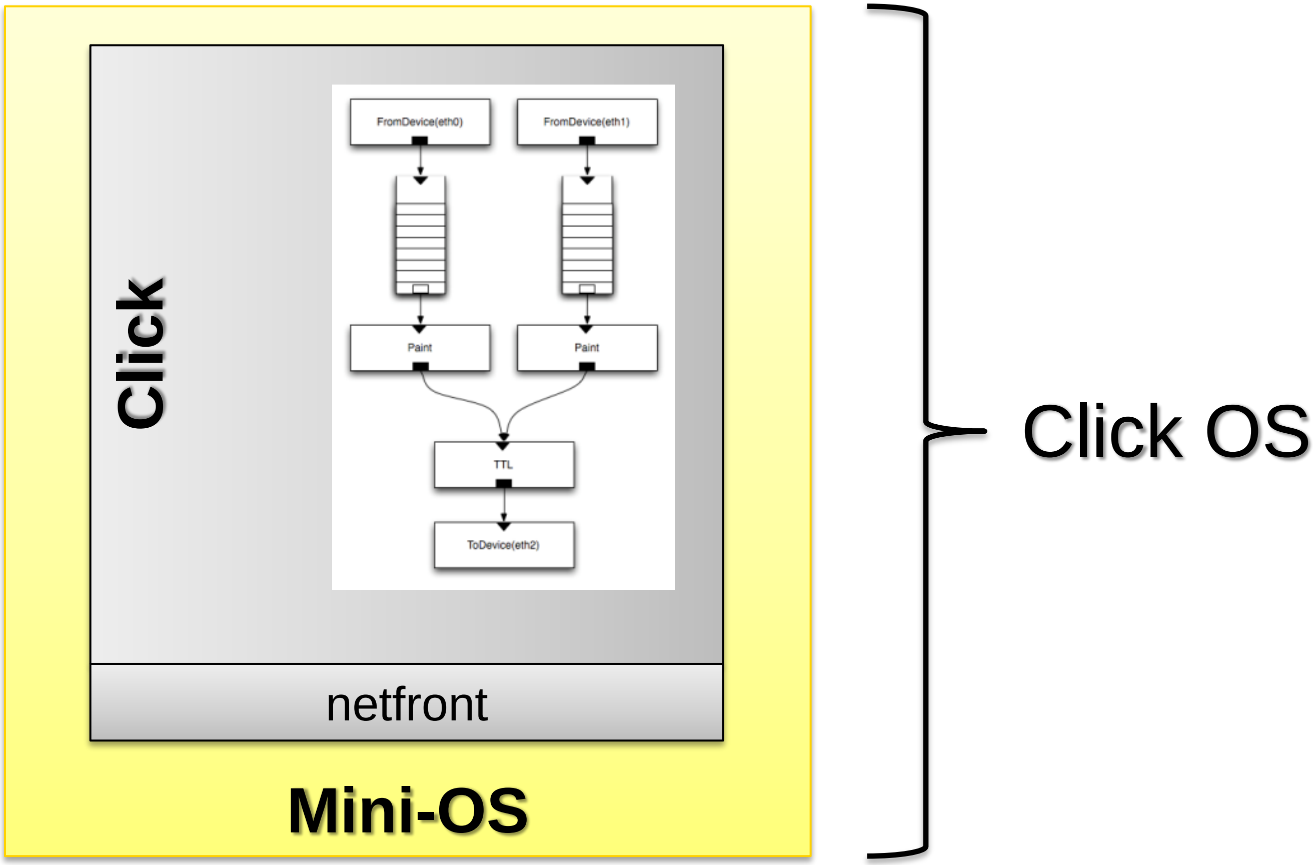
ClickOS

We created ClickOS

- A prototype for programmable middlebox VMs
- Tiny VM that runs Click configurations
  - Varied range of Middlebox configurations possible
- Can achieve 10Gb/s throughput using only a single core

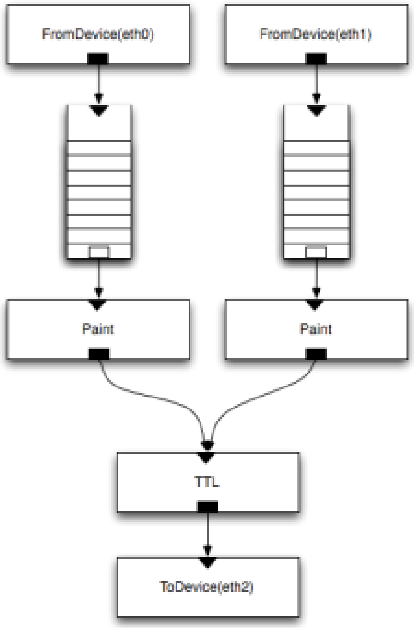
ClickOS: Click and Mini-OS

- ClickOS is a minimalistic and specialized VM that loads and runs Click configurations



Click: The Click Modular Router

- Click is a modular architecture for “horizontal” packet processing
- Concept of elements
  - An element represents basic functionality *(e.g., queue, classifier)*
  - Each element has inputs and outputs
  - Elements are configurable, parametrable
- Elements in- and outputs are connected together (graph) in order to create a network function
  - IP Router
  - NAT
  - DNS Proxy
  - IDS
  - …



Modular *Router*

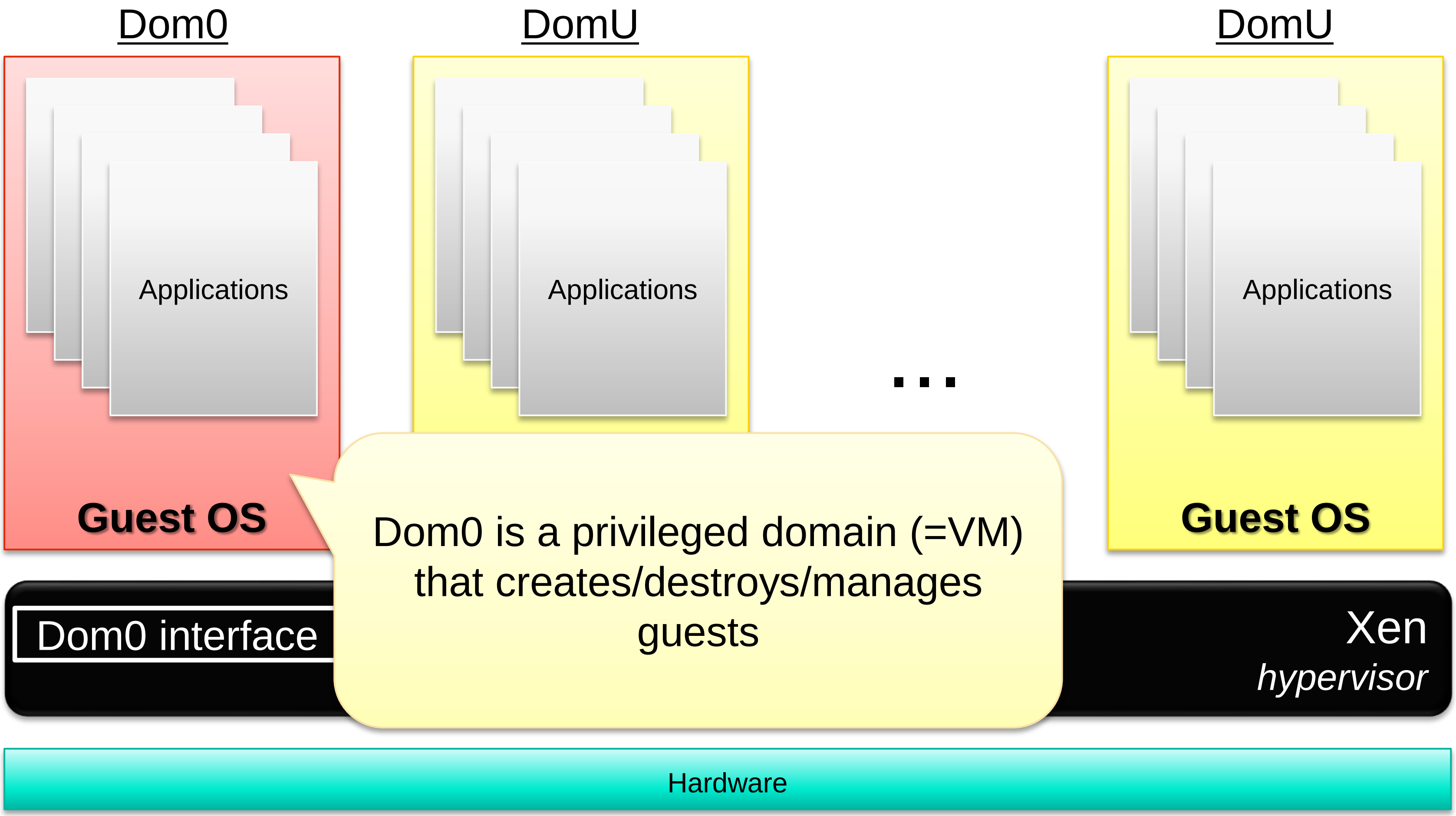
Mini-OS

- “Minimalistic Operating System”
- Provided by the Xen community
  - Originally demonstrates para-virtualizated interfaces/drivers of the Xen hypervisor
- Is executed as a para-virtualized guest domain on Xen
  - Minimalistic VM↔HV interface *(even no hw-virtualization support is required)*
- Simple monolithic OS design → low overhead
  - Single address space *(no system calls)*
  - Simple co-operative scheduler

- → We run Click directly within a Mini-OS container

Xen Primer

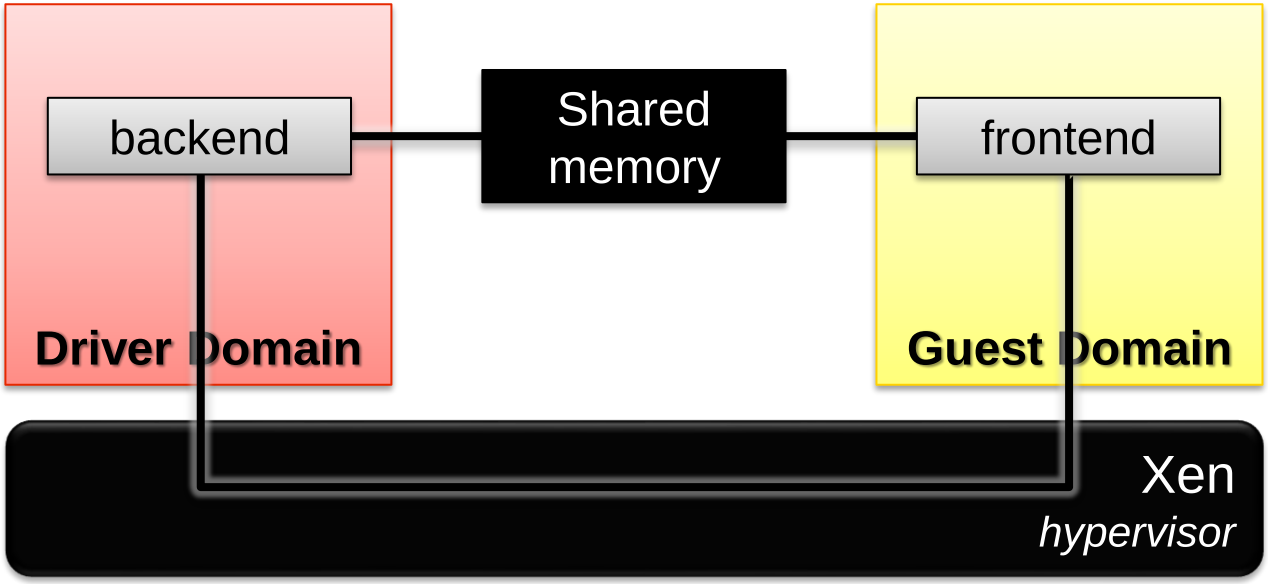
Xen is a Type-1 hypervisor



Dom0 is a privileged domain (=VM) that creates/destroys/manages guests

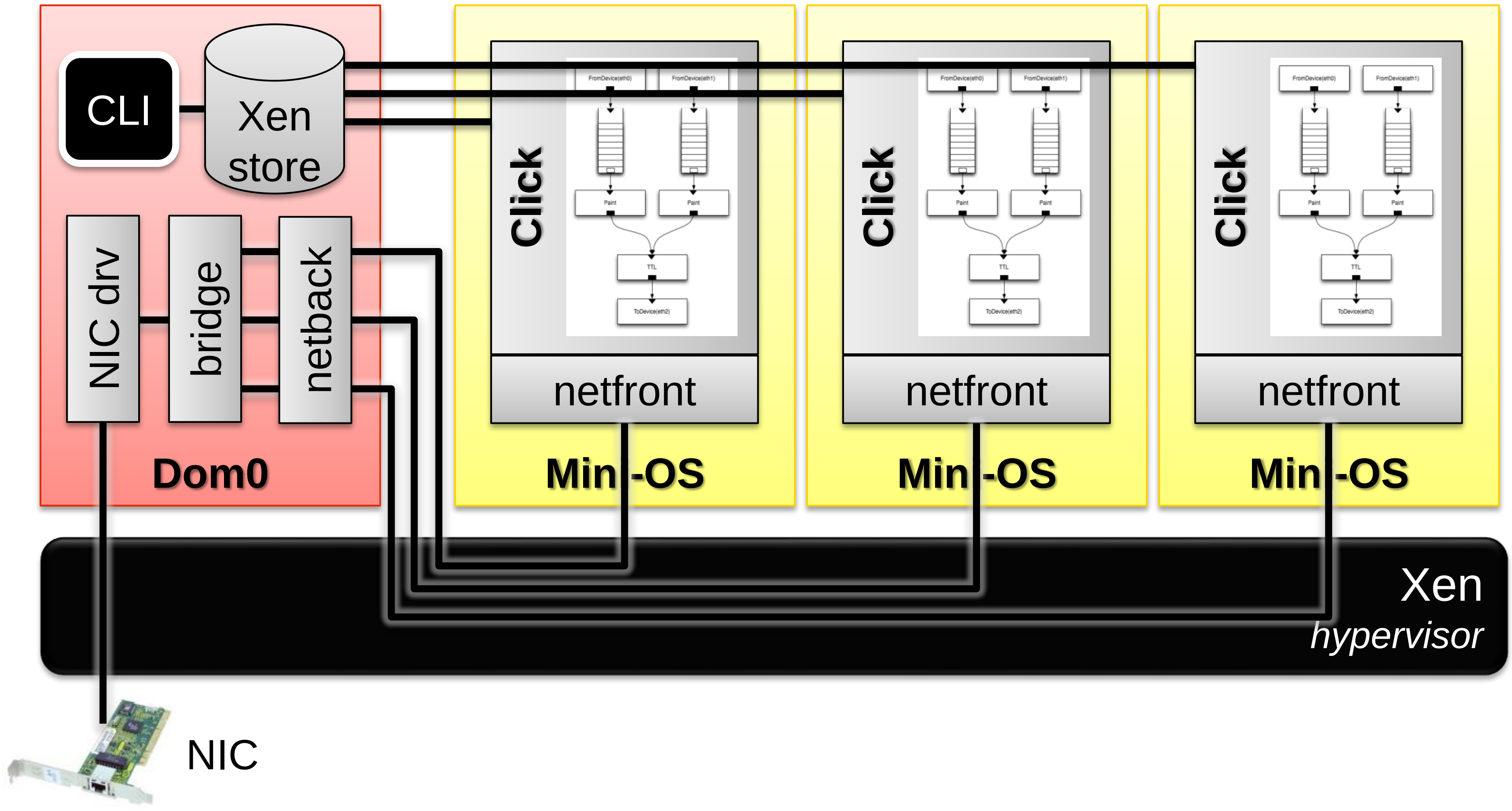
Xen: Split-driver model

- Xen implements the split-driver model
- Actually device driver is operated by a driver domain
  (this is usually Dom0)
  - Its functionality is provided to guests via a generic “backend” interface
- Guests only need to implement a generic frontend driver
  (e.g., network, block, console)



The Final Picture: ClickOS on Xen

- Example: 3 ClickOS instances



Network Functions Virtualization on top of Xen

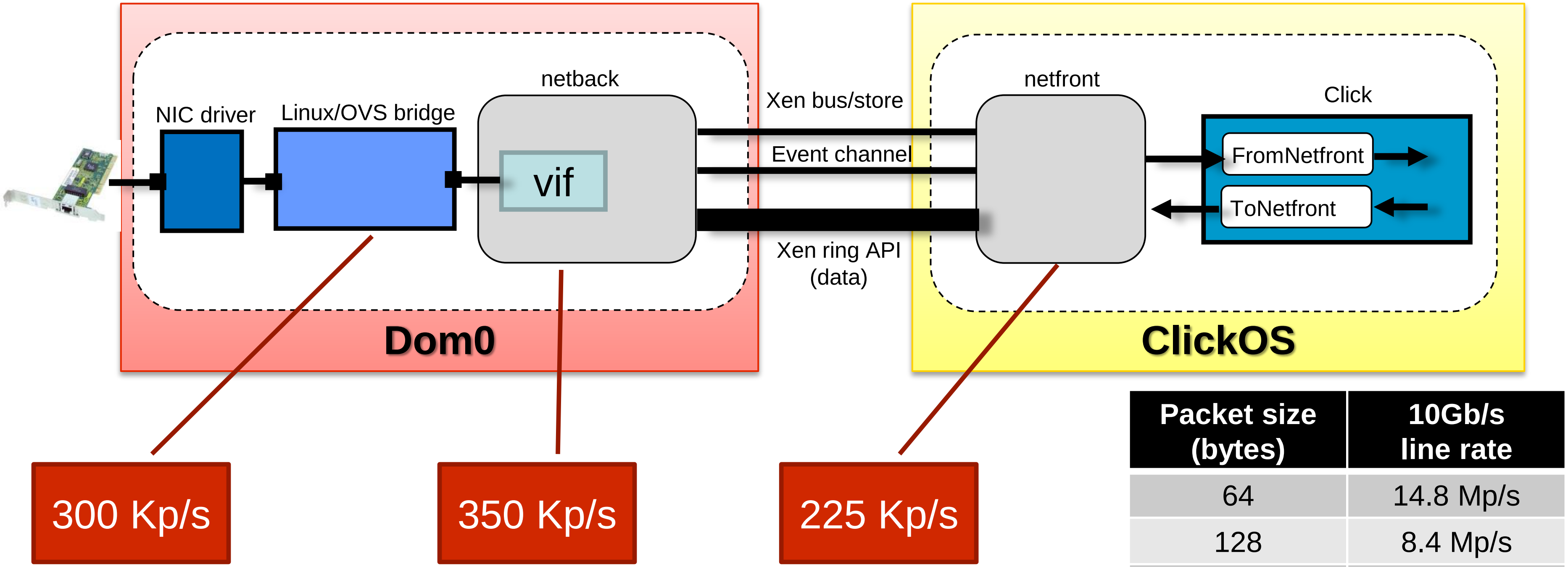
Optimizations

Reasons for Optimizations

- Mini-OS's implementation was mainly just functional
- After sticking all pieces together (Click, Mini-OS, Xen)
  - → we observed poor performance

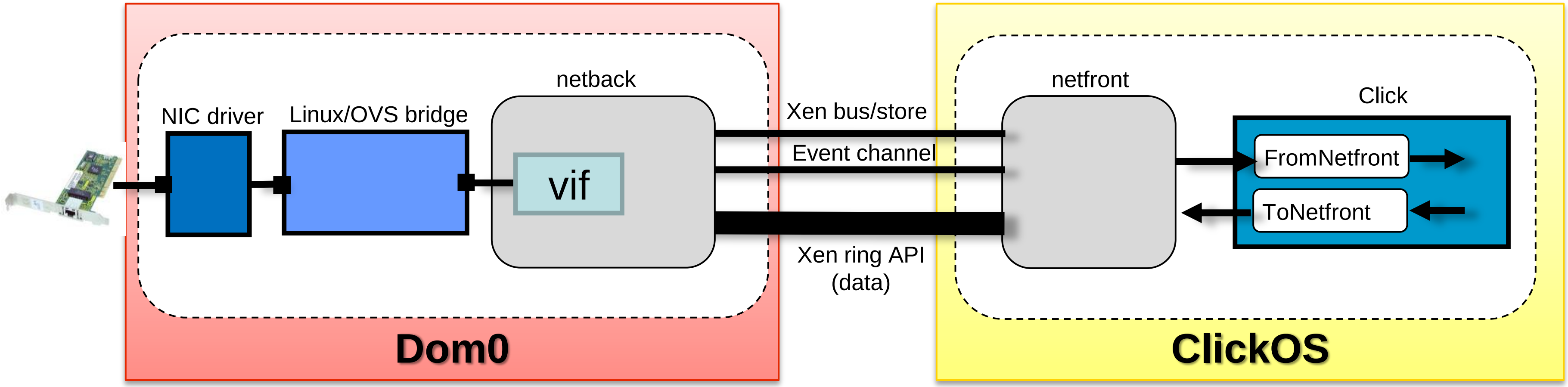
- Performance improvements on
  - Xen I/O Subsystem & Mini-OS
  - Click
  - Management Tools (introduced 'cosmos')

Bottlenecks in Xen’s I/O Subsystem

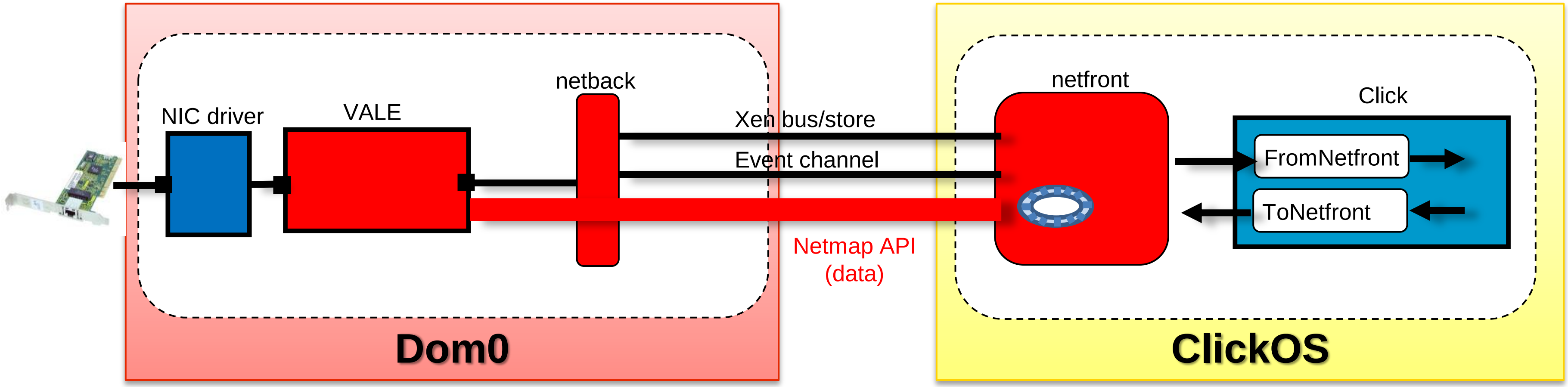


Packet size (bytes)	10Gb/s line rate
64	14.8 Mp/s
128	8.4 Mp/s
256	4.5 Mp/s
512	2.3 Mp/s
1024	1.2 Mp/s
1500	810 Kp/s

Improving Xen’s I/O Subsystem



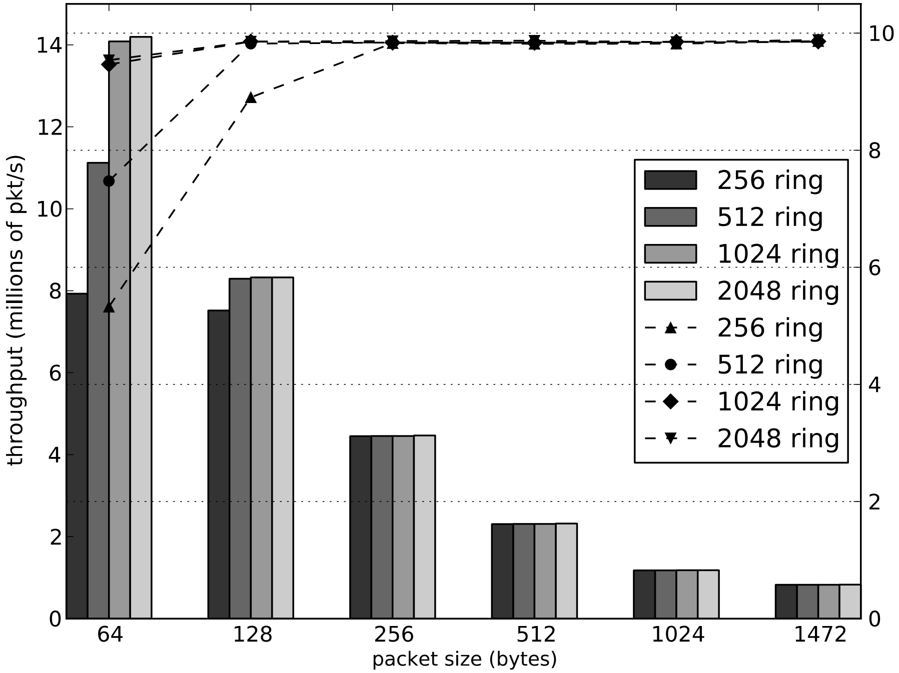
Improving Xen’s I/O Subsystem



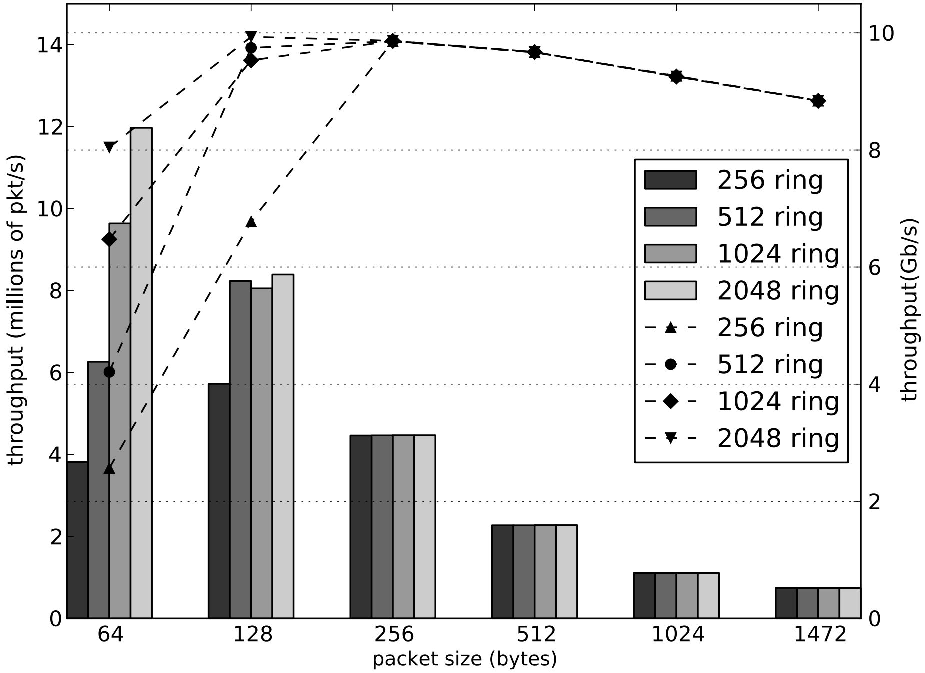
Resulting Mini-OS Base Performance

- **`pkt-gen`** as Mini-OS application
- Single-core, single VM performance

Transmitting



Receiving



Improving Click’s Performance

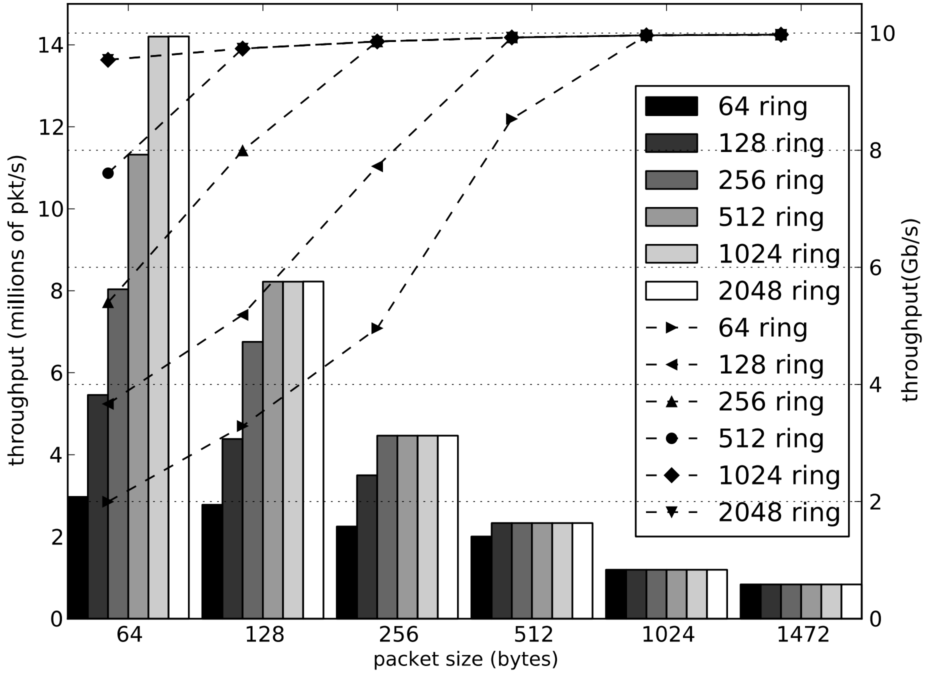
- First ClickOS implementation could only generate 560Kp/s
  - InfiniteSource → UDPIPEncap → EtherEncap
- A number of optimizations:
  1. Inherent problem: Click forces packets to be copied when not needed
     - » Solution: do not copy packet for modification when only one entity has a reference to the packet *(lazy copy)*
  2. Increate **`burst`** parameter to minimize Click scheduler overhead
  3. Allocate larger head room so that no further allocations are needed once a packet is created
- → Applying these results in 13.2 Mp/s

- → Together with our optimized “netfront” we are able to achieve 10Gb/s line rate for almost all packet sizes

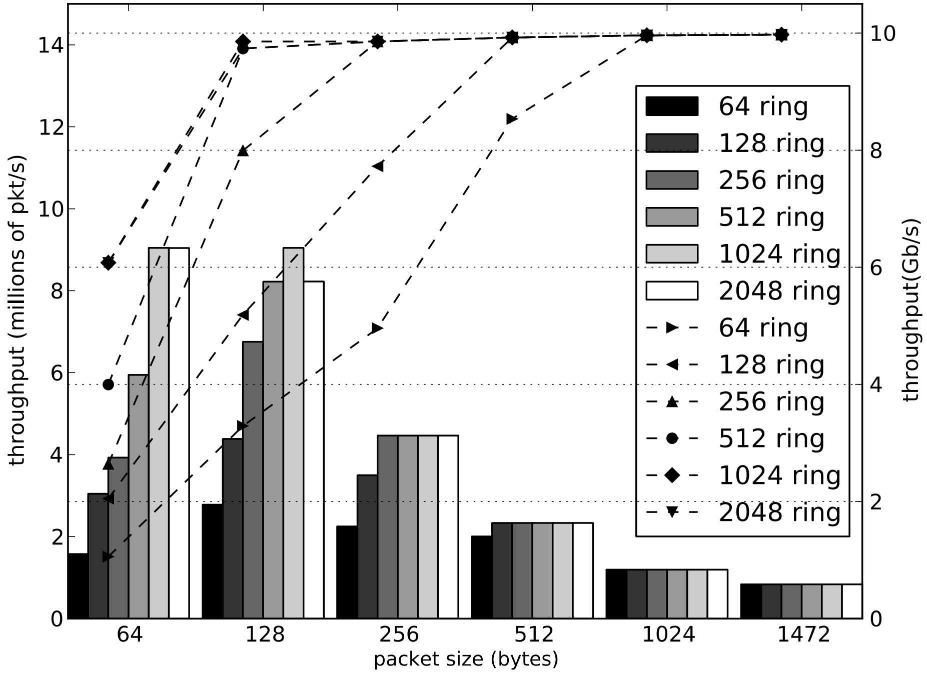
Resulting ClickOS Base Performance

- Simple Click configurations to generate/receive traffic
- Single-core, single-VM performance

Transmitting



Receiving

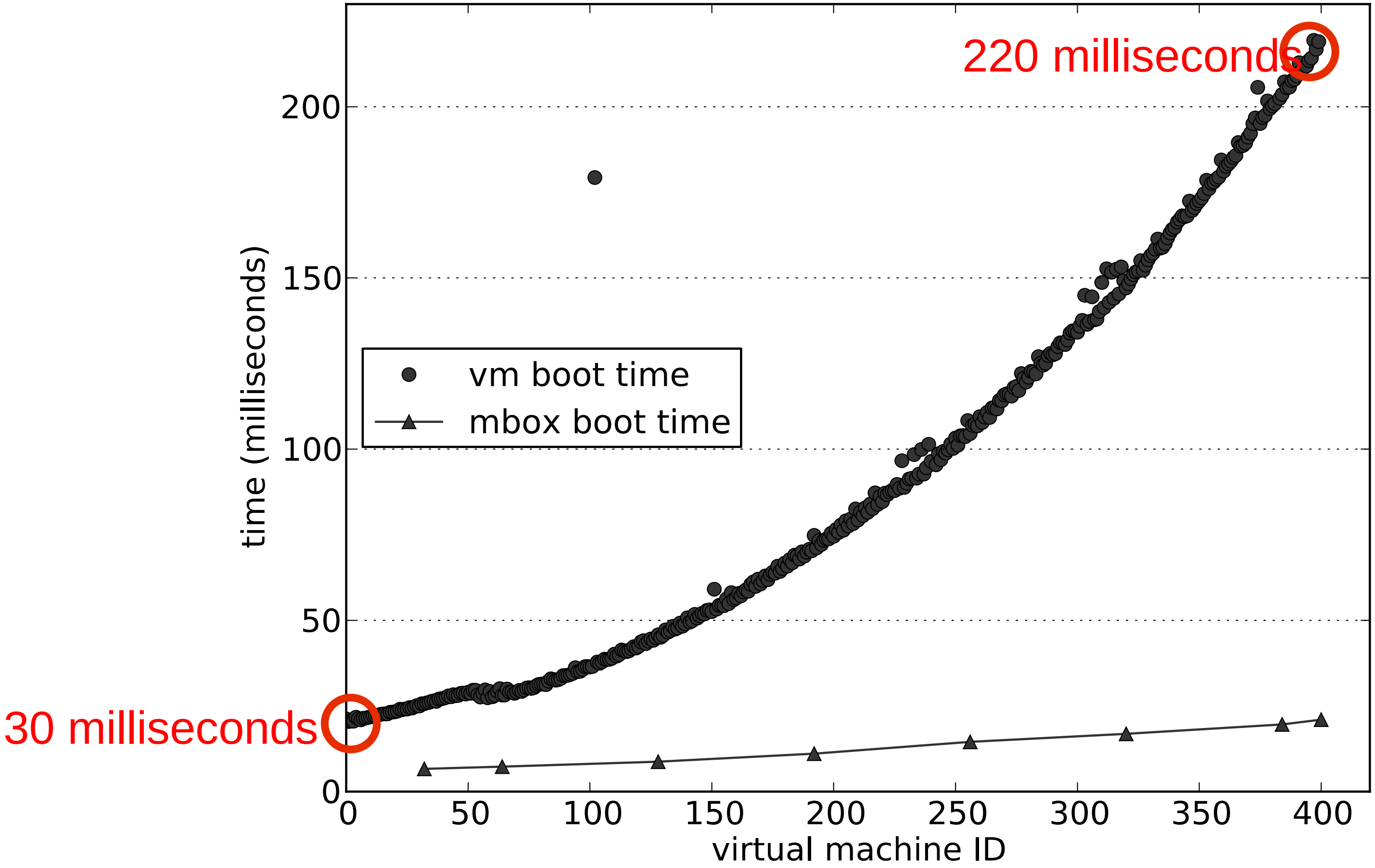


Management Tools

- First tests yielded ClickOS boot times of ~1 sec
- Carried out a number of changes
  - Using new XenStore implementation (oXenstore)
  - Using own management tool, called **`cosmos`** instead of **`xl`**/**`xm`** (together with a wrapper in python to load click configurations) to create/destroy/manage ClickOS guests

Boot time of 400 VMs

- **VM Boot time:** Time between VM creation until Click control thread runs
- **Mbox Boot Time:** Time to instantiate a click configuration within the VM

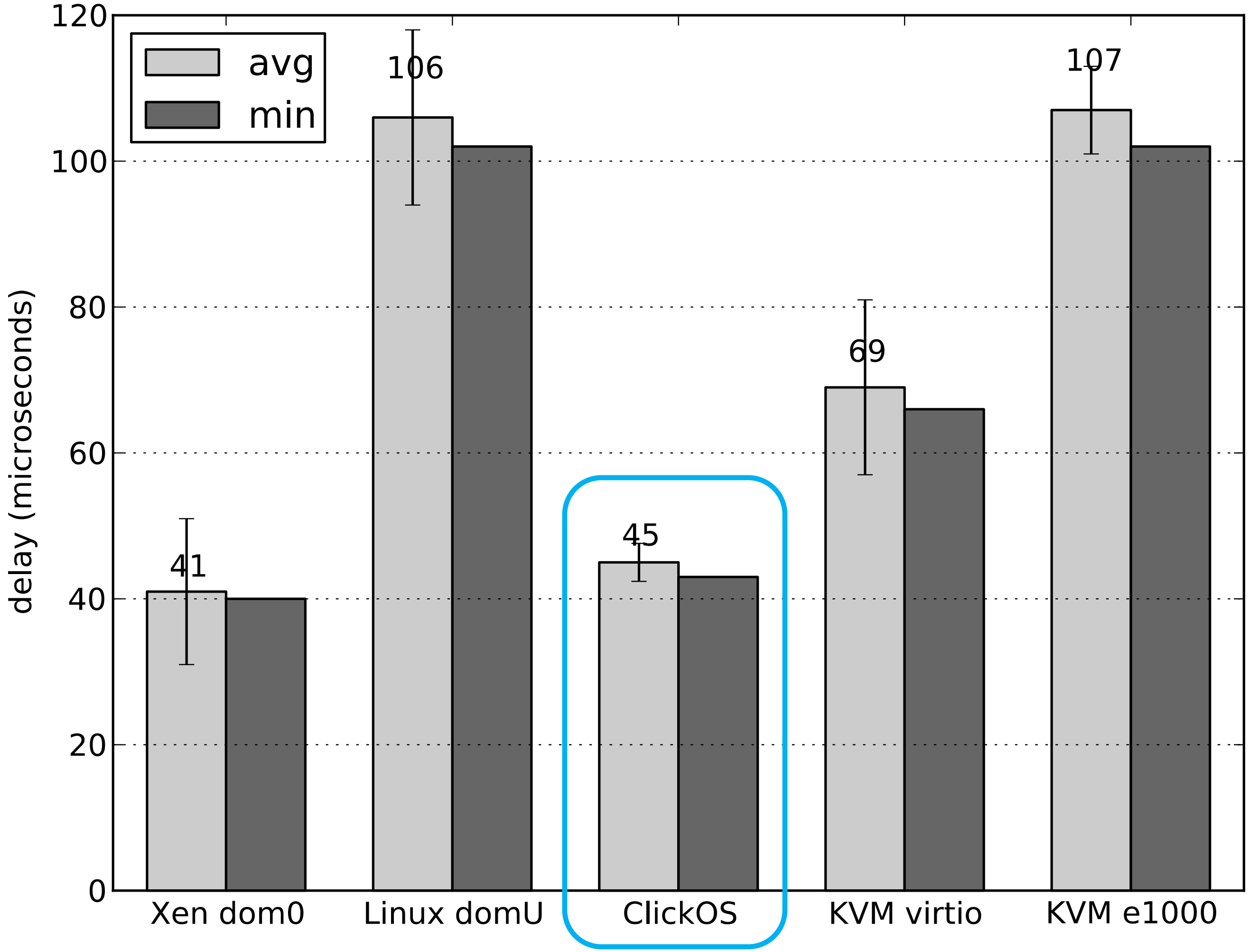


Network Functions Virtualization on top of Xen

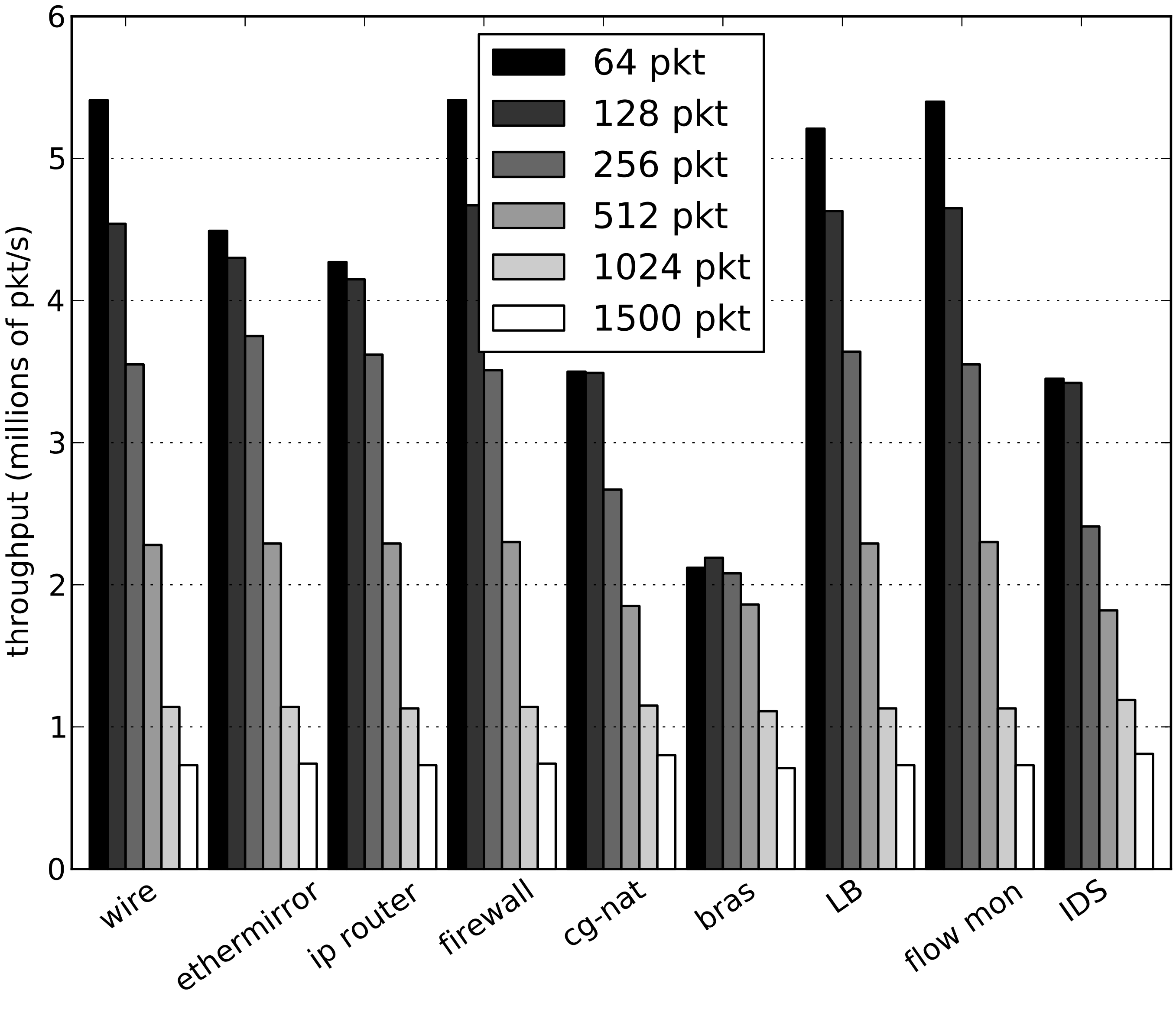
Probing ClickOS

ClickOS in Comparison

- *Here:* ICMP Ping response (RTT); measured from external Box

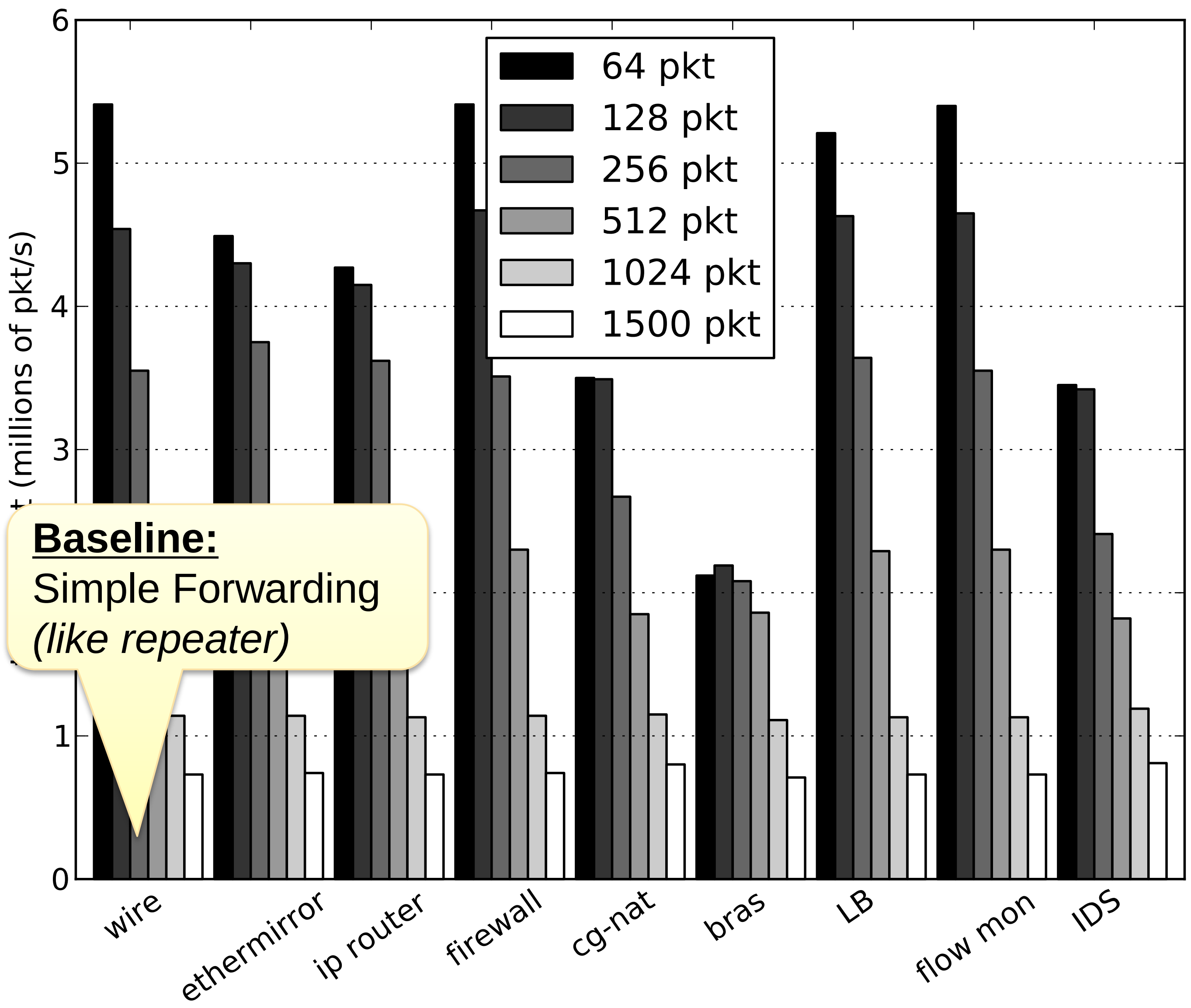


Middlebox Application Examples and their Performance



Packet size (bytes)	10Gb/s line rate
64	14.8 Mp/s
128	8.4 Mp/s
256	4.5 Mp/s
512	2.3 Mp/s
1024	1.2 Mp/s
1500	810 Kp/s

Middlebox Application Examples and their Performance

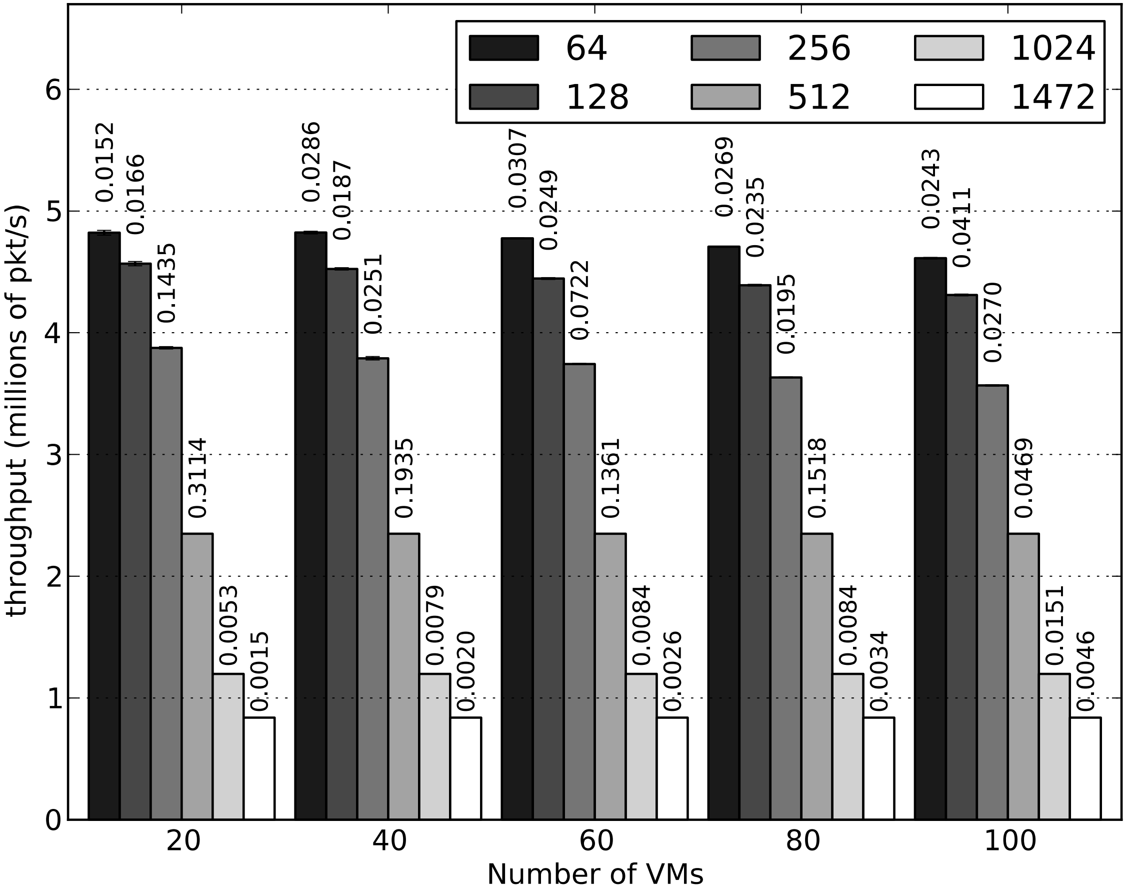


Baseline:
Simple Forwarding
(like repeater)

Packet size (bytes)	10Gb/s line rate
64	14.8 Mp/s
128	8.4 Mp/s
256	4.5 Mp/s
512	2.3 Mp/s
1024	1.2 Mp/s
1500	810 Kp/s

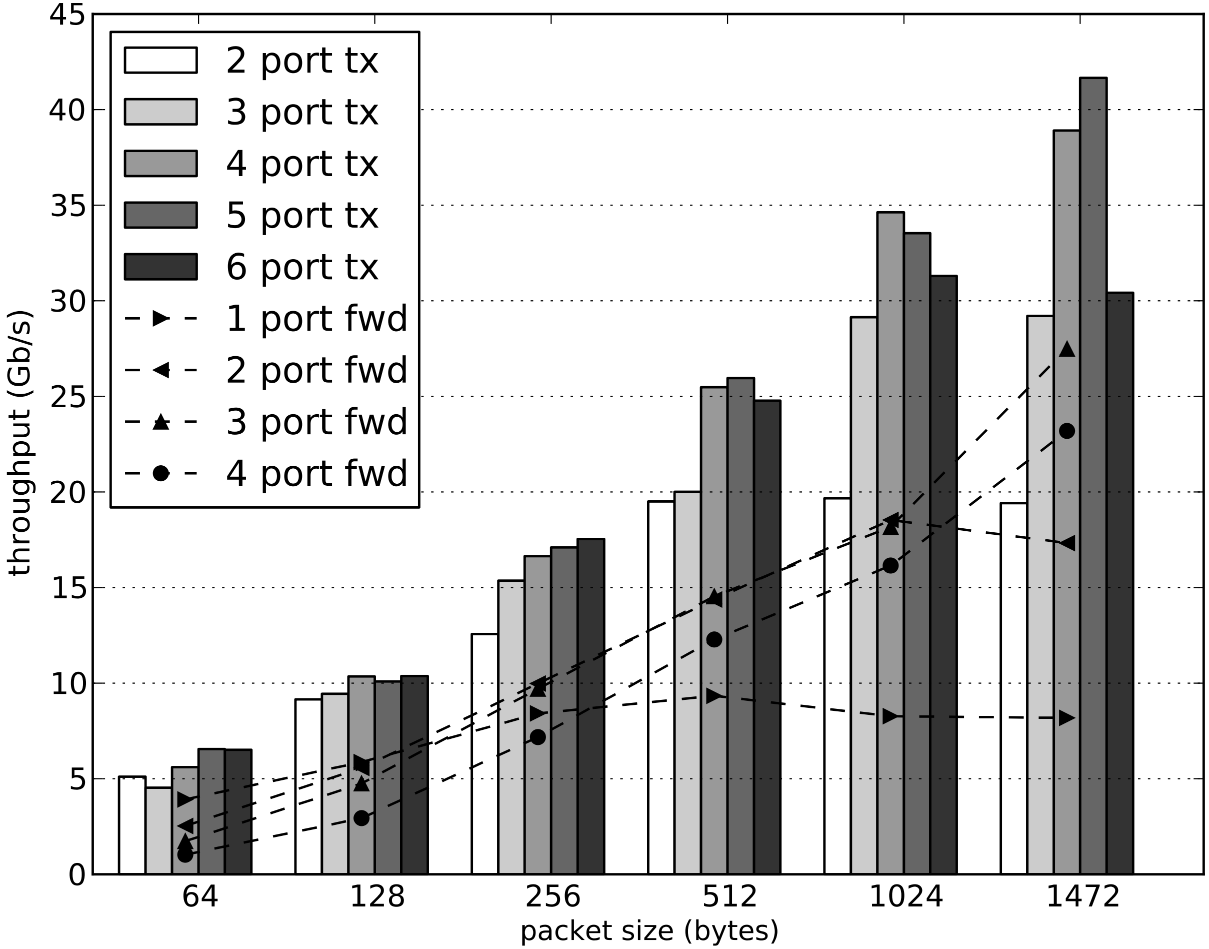
Scaling out ClickOS: Concurrent Transmit VMs

- Cumulative throughput:
  Multiple VMs generate traffic to a single NIC
- Numbers on top of bars show standard deviation between VMs



Scaling out ClickOS: Multiple NICs/VMs

- Cumulative throughput by using multiple 10 GB/s ports *(1 port per VM)*



Conclusion & Future Work

Presented ClickOS

- Tiny (5MB) Xen VM tailored at network processing
- Can be booted in 30 milliseconds
- Can run a large number of ClickOS VMs concurrently (~ 100)
- Can achieve 10Gb/s throughput using only a single core.
- Can run a varied range of Middleboxes with high throughput

Future Work

- Improving performance on NUMA systems
- High consolidation of ClickOS VMs (thousands)
- Service chaining

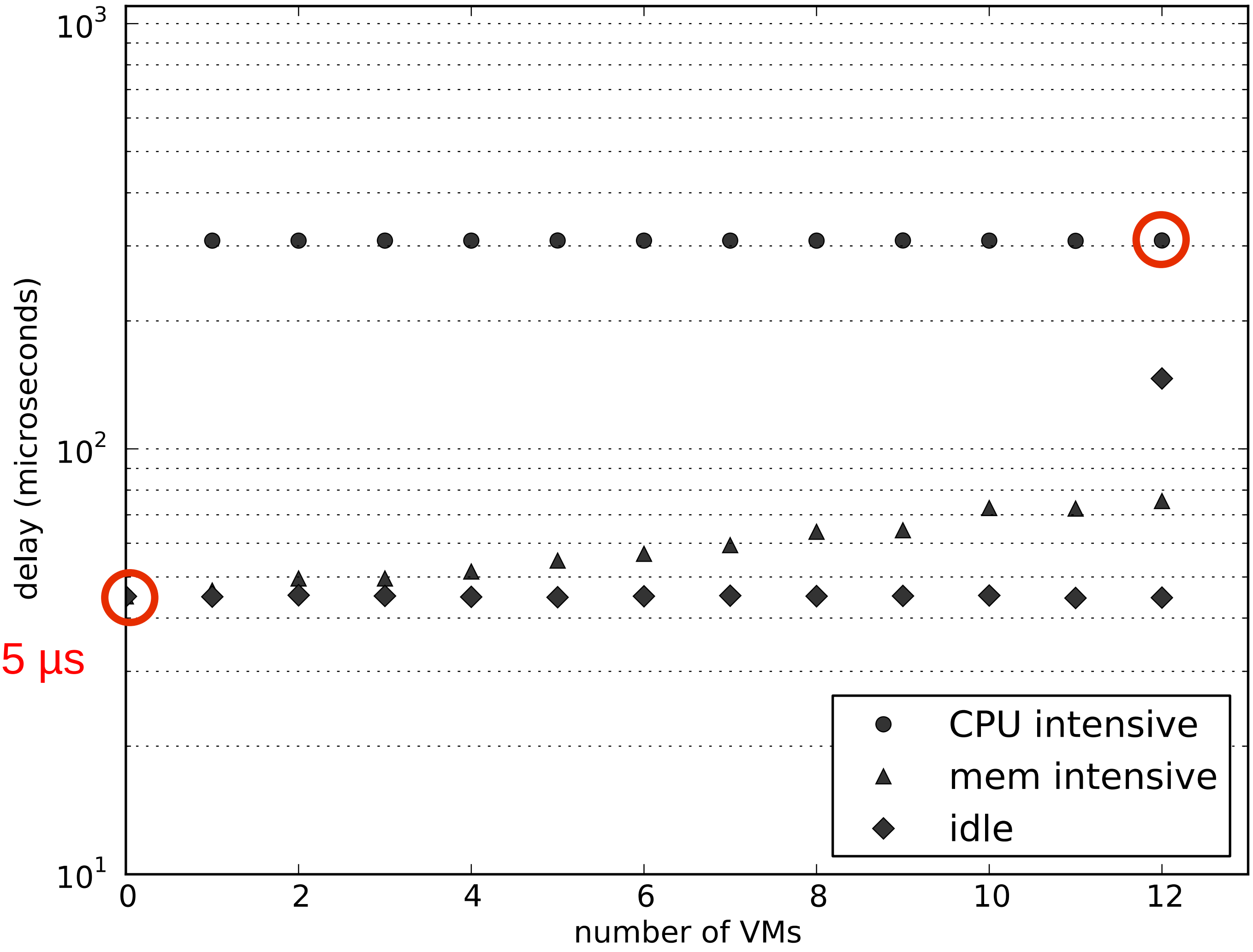
Empowered by Innovation

NEC

Q&A

ClickOS under Stress

- *Here:* ICMP Ping response (RTT); measured from external Box



~ 300 µs

~ 45 µs

Utilizing XenStore to Manage Click Configurations

Install/Uninstall Middlebox functionality

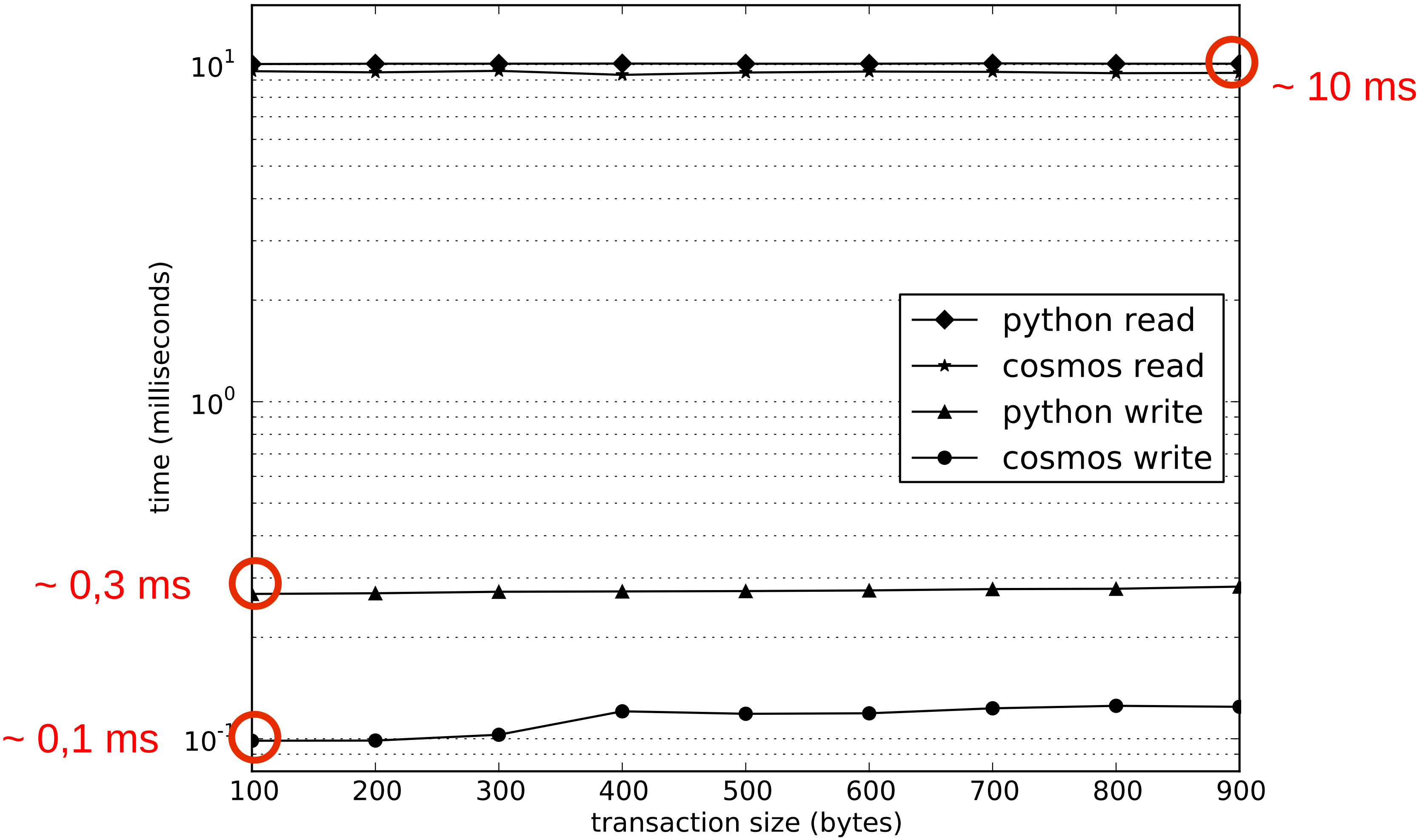
- Control thread creates a new entry in the XenStore (e.g., **`/local/domain/<domID>/clickos/config`**)
- Control thread sets up a watch on that entry
- When written to, control thread creates a new mini-OS thread to run the main Click loop (i.e., the data plane)
- Uninstall by writing empty string to Xenstore entry

Read/Write handlers

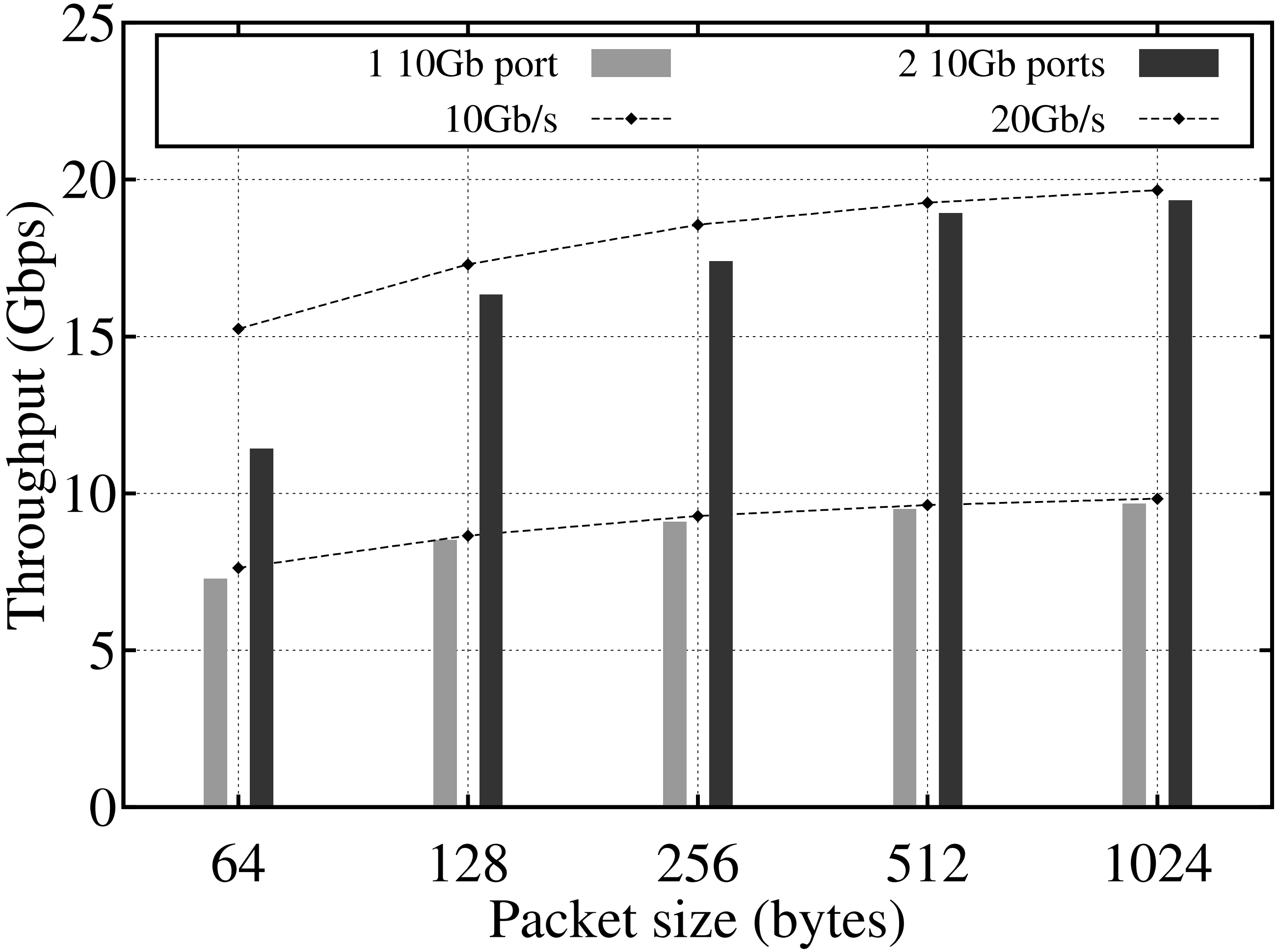
- New, transparent *ClickOSControl* element
- Intermediary between CLI/API and elements' read/write handlers

XenStore Performance

- Reduced XenStore read/write delay by using C libraries (via **`cosmos`**) instead of python bindings



Back-End Switch Performance (VALE)


1 10Gb port
2 10Gb ports
10Gb/s
20Gb/s
Throughput (Gbps)
Packet size (bytes)