Circuit Switched VM Networks for Zero-Copy IO

Johannes Krude, Mirko Stoffers, Klaus Wehrle

https://comsys.rwth-aachen.de/

KBNets18, 2018-08-20

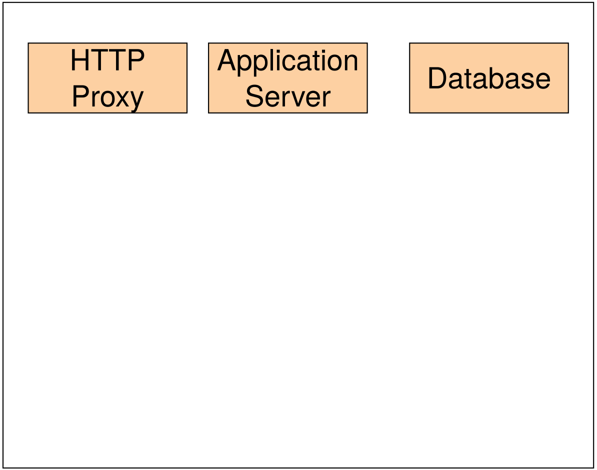
COM SYS | RWTH AACHEN UNIVERSITY

VM Networks

- **VMs are used for Isolation**
  - Multiple Tenants on the same Host
  - Compartmentalization
  - Fault Isolation

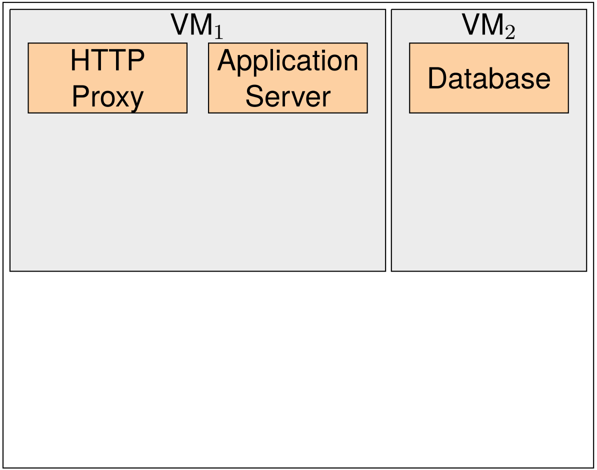
VM Networks

- **VMs are used for Isolation**
  - Multiple Tenants on the same Host
  - Compartmentalization
  - Fault Isolation



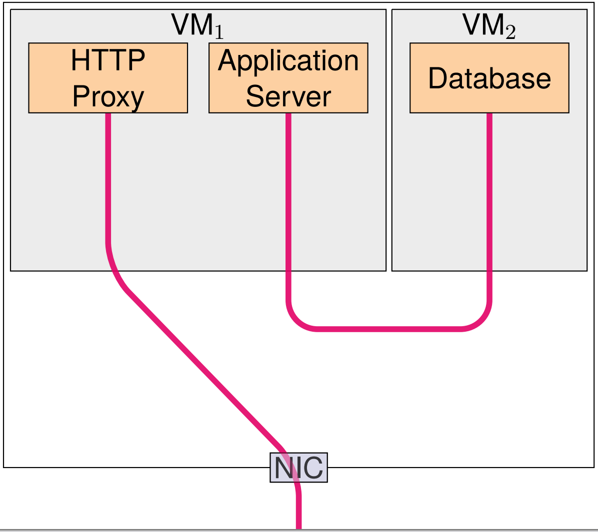
VM Networks

- **VMs are used for Isolation**
  - Multiple Tenants on the same Host
  - Compartmentalization
  - Fault Isolation

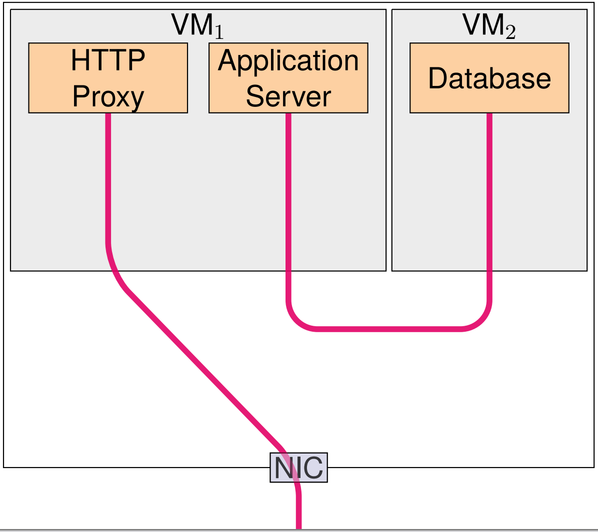


VM Networks

- **VMs are used for Isolation**
  - Multiple Tenants on the same Host
  - Compartmentalization
  - Fault Isolation
- **Isolation complicates Communication**



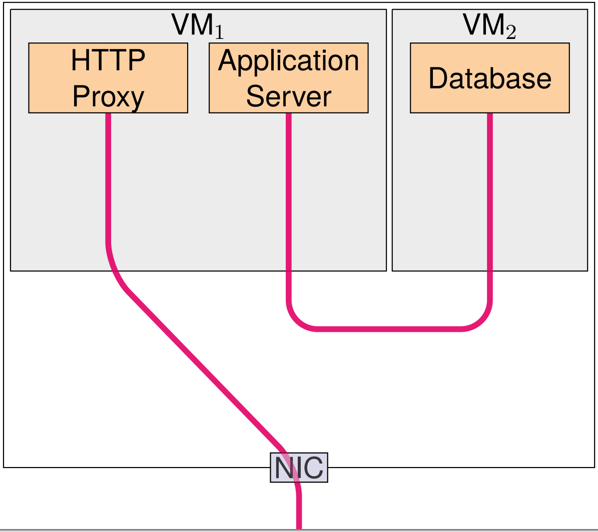
- **VMs are used for Isolation**
  - Multiple Tenants on the same Host
  - Compartmentalization
  - Fault Isolation
- **Isolation complicates Communication**
- **Until now: Performance and Isolation are mutually exclusive**



VM Networks

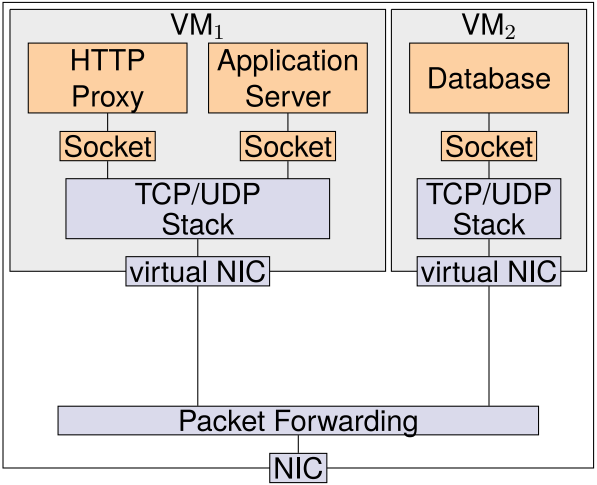
- **VMs are used for Isolation**
  - Multiple Tenants on the same Host
  - Compartmentalization
  - Fault Isolation
- **Isolation complicates Communication**
- **Until now: Performance and Isolation are mutually exclusive**

Circuit Switched VM Networks
enable
Zero-Copy IO with Isolation



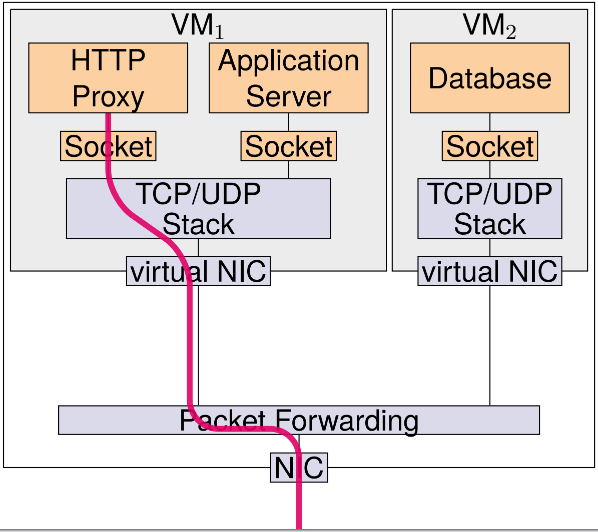
VM Packet Processing

- **Problem: Packet Switching**



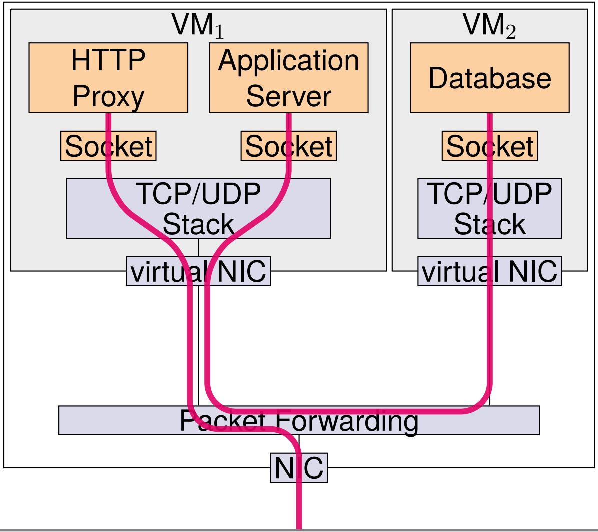
VM Packet Processing

- **Problem: Packet Switching**
- **Unnecessary Overhead**



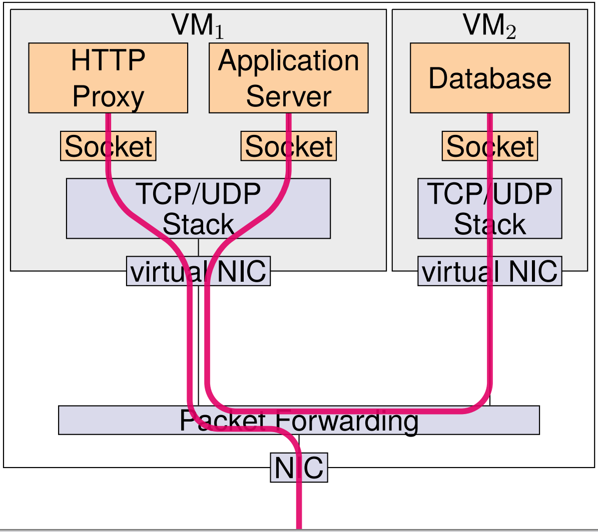
VM Packet Processing

- **Problem: Packet Switching**
- **Unnecessary Overhead**
  - ▶ Multiplexing
  - ▶ Packetization



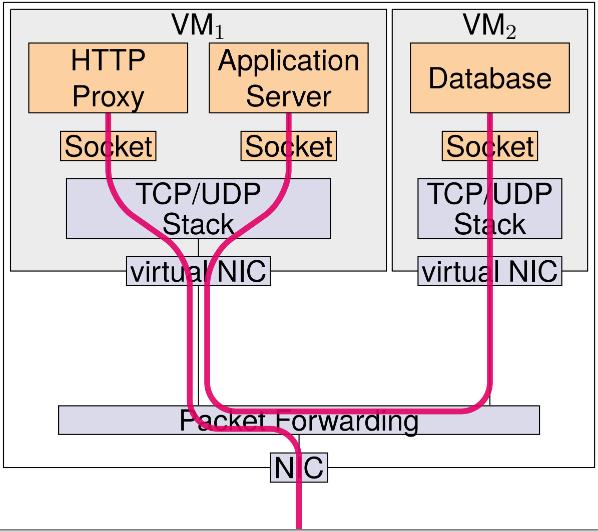
VM Packet Processing

- **Problem: Packet Switching**
- **Unnecessary Overhead**
  - Multiplexing
  - Packetization
  - Congestion Control
  - Retransmissions
  - Reordering



VM Packet Processing

- **Problem: Packet Switching**
- **Unnecessary Overhead**
  - Multiplexing
  - Packetization
  - Congestion Control
  - Retransmissions
  - Reordering
  - (Copying)

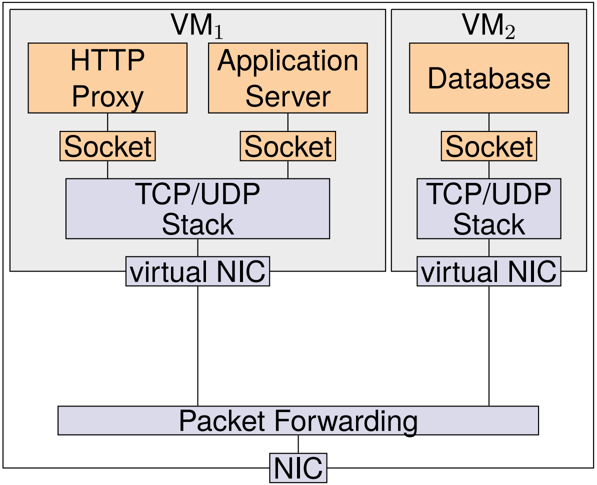


VM Packet Processing

- **Problem: Packet Switching**
- **Unnecessary Overhead**
  - Multiplexing
  - Packetization
  - Congestion Control
  - Retransmissions
  - Reordering
  - (Copying)

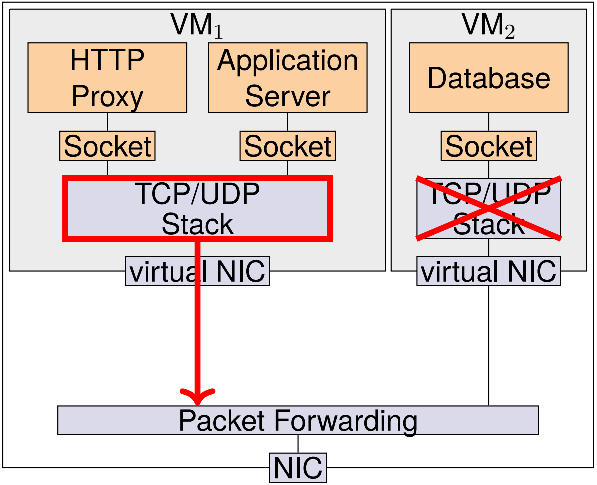
Goals

- Remove Overhead
- Keep Application Compatibility
- Keep Network Isolation



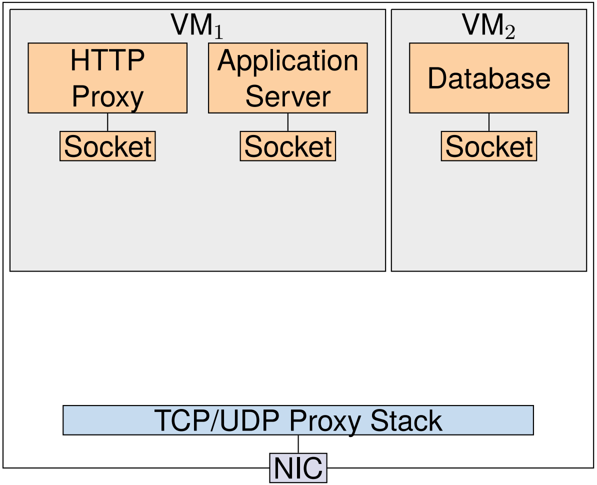
Removing Overhead

- **No Packet Processing in VM Kernels**
  - Move to Host if Still Needed
  - Remove if Possible



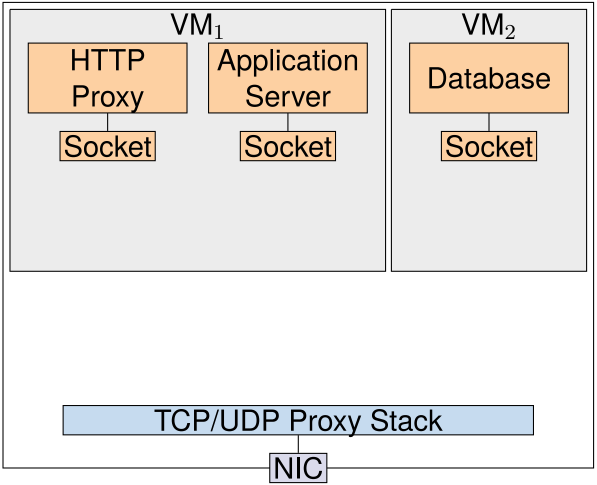
Removing Overhead

- **No Packet Processing in VM Kernels**
  - Move to Host if Still Needed
  - Remove if Possible



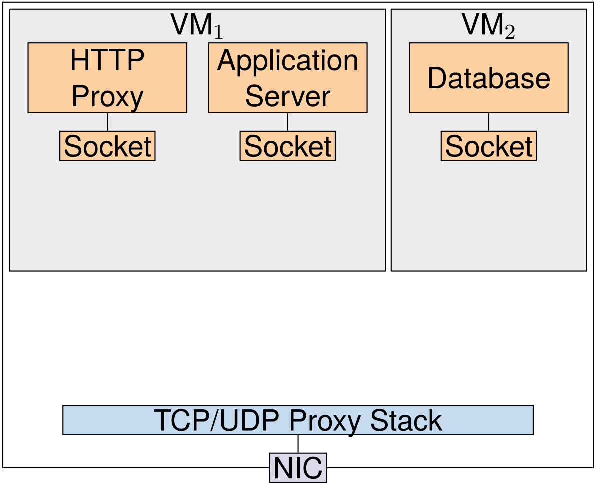
Removing Overhead

- **No Packet Processing in VM Kernels**
  - Move to Host if Still Needed
  - Remove if Possible
- **Keep Socket API**
  - Provides Access to Streams & Datagrams
  - Required to Support Legacy Applications
  - Provides Isolation between Applications

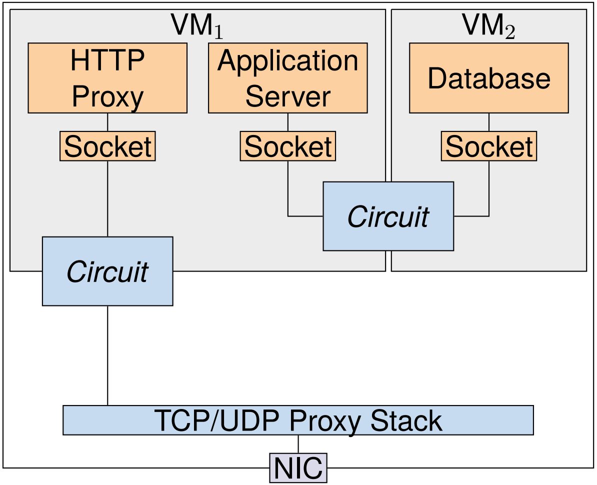


Removing Overhead

- **No Packet Processing in VM Kernels**
  - Move to Host if Still Needed
  - Remove if Possible
- **Keep Socket API**
  - Provides Access to Streams & Datagrams
  - Required to Support Legacy Applications
  - Provides Isolation between Applications
- **Provide Zero-Copy API**
  - As Optional Extension to Socket API

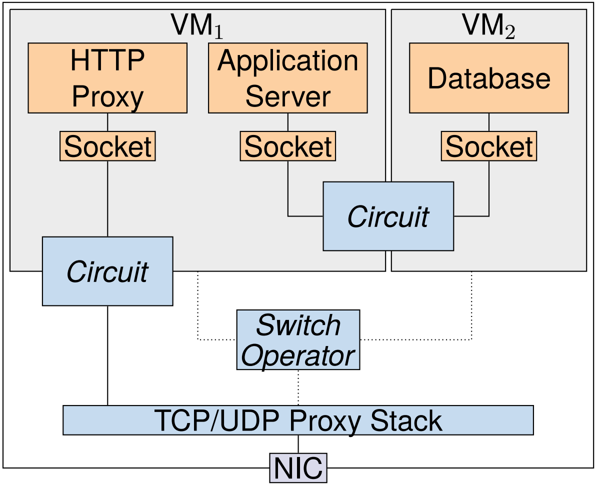


- **Separate Shared-Memory based Circuit for each Connection**
  - from VM to Proxy Stack
  - or Direct from VM to VM

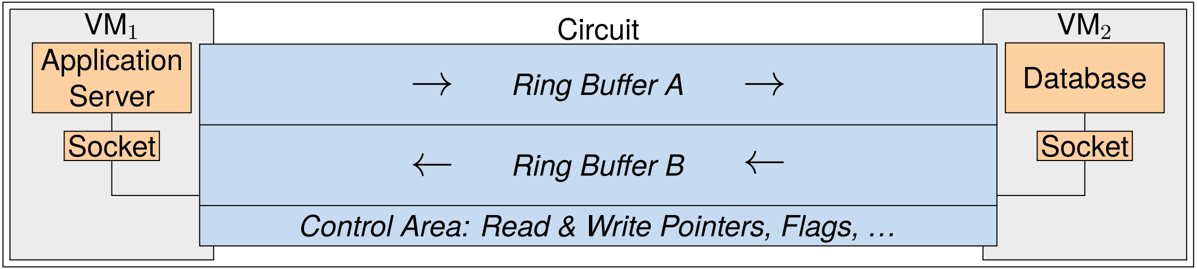


Circuit Switched VM Networks

- **Separate Shared-Memory based Circuit for each Connection**
  - from VM to Proxy Stack
  - or Direct from VM to VM
- **Switch Operator**
  - Mediates Connection Establishment
  - Enforces Connection Policies

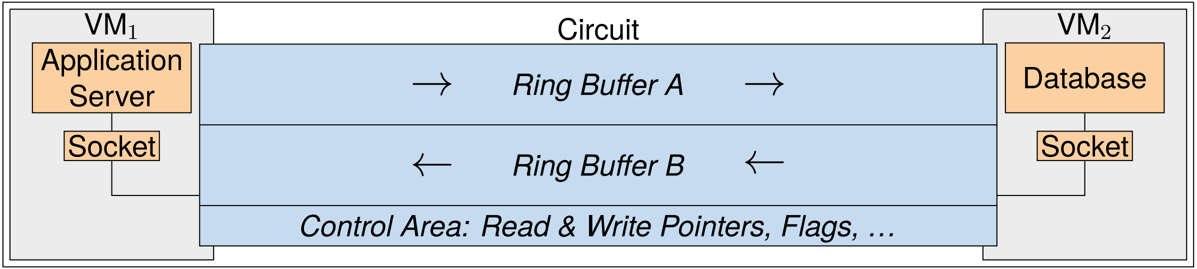


Circuits



- **Protocol Features**
  - TCP Flow Control: Ring Buffers
  - UDP Datagrams: Prepend some kind of Header

Circuits



- **Protocol Features**
  - TCP Flow Control: Ring Buffers
  - UDP Datagrams: Prepend some kind of Header
- **Zero-Copy Circuit**
  - Map Circuit Memory into Application
  - Optional ⇒ Compatible with Legacy Applications

Network Isolation

- **No Access to Communication of other Applications**
  - ▶ Keeps Socket Isolation
  - ▶ Even when doing Zero-Copy IO

Network Isolation

- **No Access to Communication of other Applications**
  - Keeps Socket Isolation
  - Even when doing Zero-Copy IO
- **Connection Policies enforced on Connection Setup**
  - No Inspection of Individual Packets needed
  - No Redundant State for Stateful Firewalls

Network Isolation

- **No Access to Communication of other Applications**
  - Keeps Socket Isolation
  - Even when doing Zero-Copy IO
- **Connection Policies enforced on Connection Setup**
  - No Inspection of Individual Packets needed
  - No Redundant State for Stateful Firewalls
- **Denying Raw Packet Access**
  - Same Level of Access as Containers
  - No Crafting of Malicious Packet Headers
  - No Unfair Congestion Control Algorithms

Implementation & Evaluation

- **Xen Hypervisor**
  - Allows for Shared-Memory between any consenting VM
- **Linux VM Kernel & Linux Host OS**
  - No VM User-Space Modifications Required
  - Use Regular Linux Sockets for Proxy Stack

Implementation & Evaluation

- **Xen Hypervisor**
  - Allows for Shared-Memory between any consenting VM
- **Linux VM Kernel & Linux Host OS**
  - No VM User-Space Modifications Required
  - Use Regular Linux Sockets for Proxy Stack

- **Works for Real-World Applications**
  - NGINX, BIND, Tor, Firefox, Transmission, Quake 3, Mutt, openssh, git, aptitude, wget, …

Implementation & Evaluation

- **Xen Hypervisor**
  - Allows for Shared-Memory between any consenting VM
- **Linux VM Kernel & Linux Host OS**
  - No VM User-Space Modifications Required
  - Use Regular Linux Sockets for Proxy Stack
- **Works for Real-World Applications**
  - NGINX, BIND, Tor, Firefox, Transmission, Quake 3, Mutt, openssh, git, aptitude, wget, …
- **Reduced VM Size**
  - Minimum Linux VM: 17 % Memory Reduction, 48 MiB to 40 MiB
  - Especially Relevant for Unikernels in high density Deployments

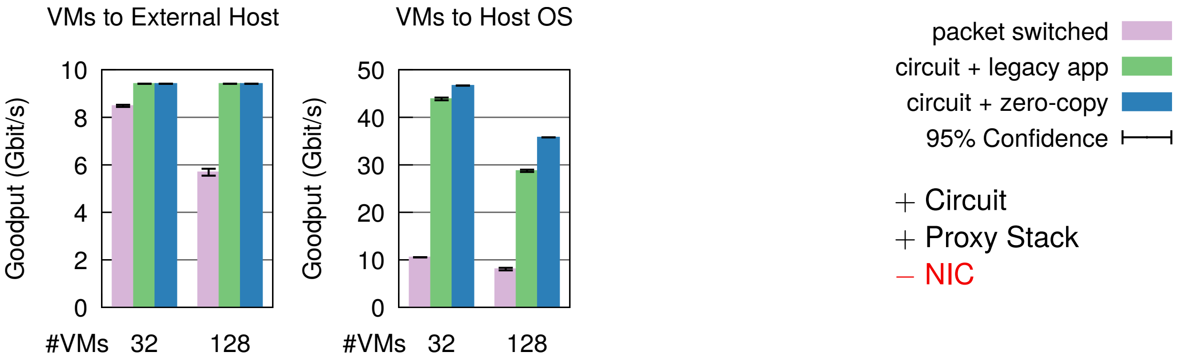
Implementation & Evaluation

- **Xen Hypervisor**
  - Allows for Shared-Memory between any consenting VM
- **Linux VM Kernel & Linux Host OS**
  - No VM User-Space Modifications Required
  - Use Regular Linux Sockets for Proxy Stack
- **Works for Real-World Applications**
  - NGINX, BIND, Tor, Firefox, Transmission, Quake 3, Mutt, openssh, git, aptitude, wget, …
- **Reduced VM Size**
  - Minimum Linux VM: 17 % Memory Reduction, 48 MiB to 40 MiB
  - Especially Relevant for Unikernels in high density Deployments
- **Measured Goodput & Response Times**
  - Hardware: Xeon E5-4610 v4 (10 Cores), Intel X710-T4 (10 Gbit)

Stream Goodput

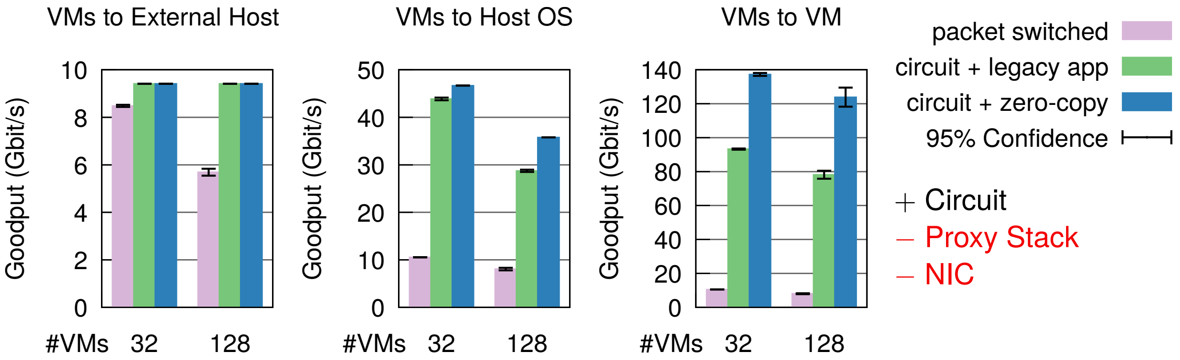
+ Circuit
+ Proxy Stack
+ NIC

Stream Goodput



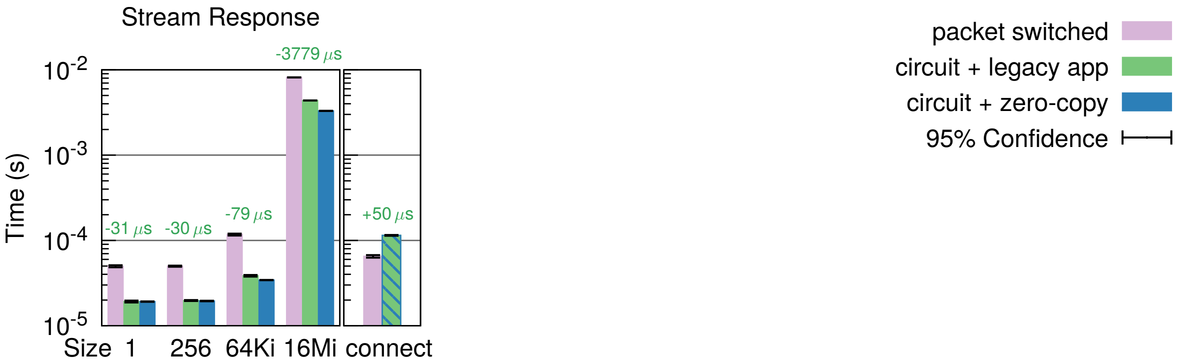
- suitable beyond 10 GBit NICs

Stream Goodput

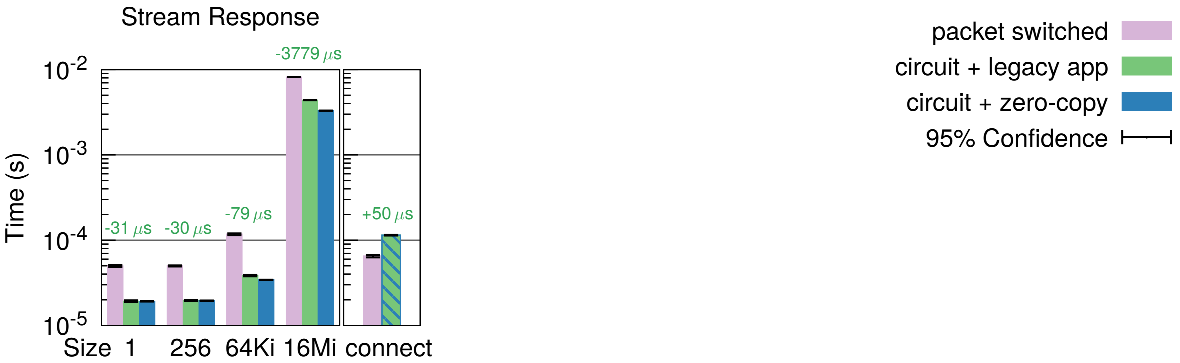


- suitable beyond 10 GBit NICs
- up to 137.2 Gbit/s with an Improvement of up to 15.4 ×

Response Times

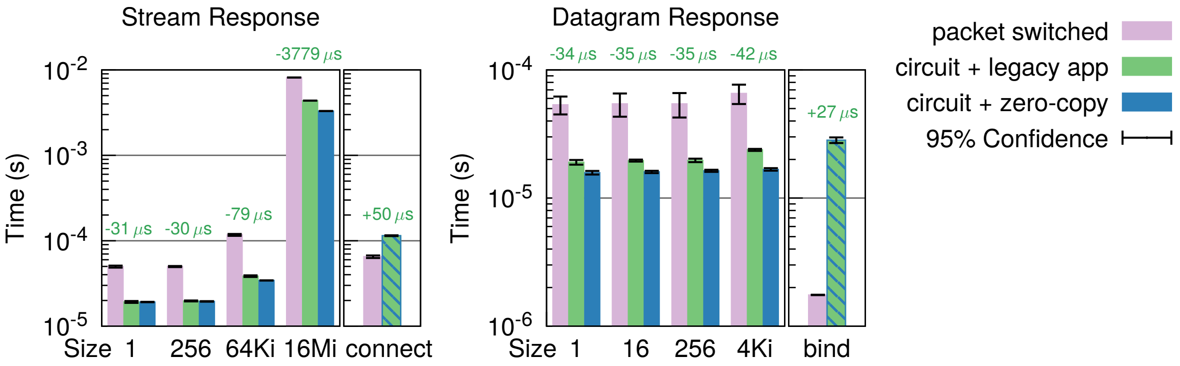


Response Times



- faster for Streams after 1-2 Rounds

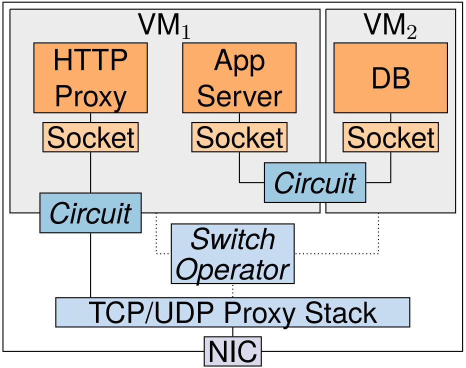
Response Times



- faster for Streams after 1-2 Rounds
- faster for Datagrams after 1 Round

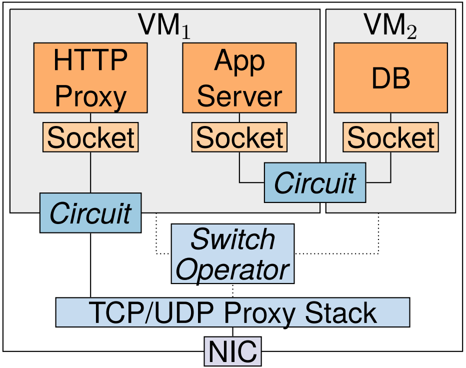
Conclusion

- **Remove Packet Processing from VM Kernels**
- **Circuit Switched VM Networks with Zero-Copy IO**
- **Network Isolation & Performance**
- **up to 137.2 Gbit/s with up to 15.4 $\times$ Improvement**



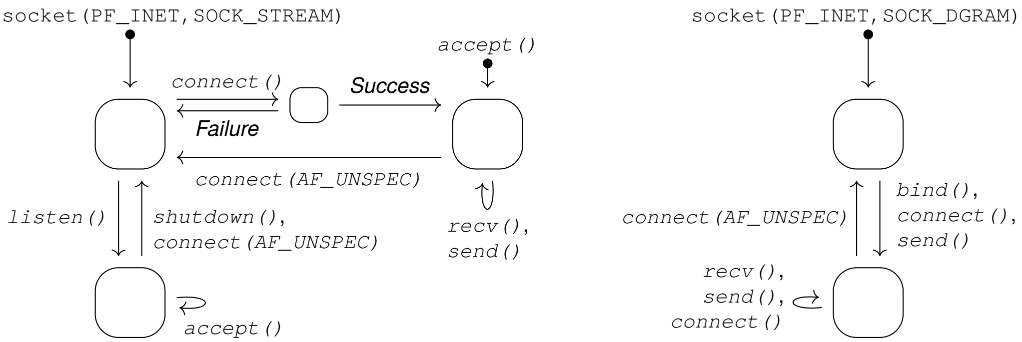
Conclusion

- Remove Packet Processing from VM Kernels
- Circuit Switched VM Networks with Zero-Copy IO
- Network Isolation & Performance
- up to 137.2 Gbit/s with up to 15.4 $\times$ Improvement



Thank you for Listening!

Socket API


socket(PF_INET,SOCK_STREAM)
accept()
connect()
Success
Failure
connect(AF_UNSPEC)
listen()
shutdown(),
connect(AF_UNSPEC)
recv(),
send()
accept()
socket(PF_INET,SOCK_DGRAM)
bind(),
connect(),
send()
connect(AF_UNSPEC)
recv(),
send(),
connect()