IOVTEE: A Fast and Pragmatic Software-based Zero-copy/Pass-through Mechanism for NFV-nodes

Assist. Prof. Ryota Kawashima
Nagoya Institute of Technology, Japan
Softwarization for Ultimate Flexibility

Traditional Networks

- Functions
- Forwarding

Multi-Slicing

- Converged
- Converged
- Converged

Softwarized Networks

Highly flexible infrastructures are crucial for 5G/cloud services
High-Speed Communications

Core Network Traffic

Are software-based approaches viable?

Marcus K. Weldon, “The Future X Networks”
White Box Switches vs. COTS Servers

White Box Switch

White Box Switch

COTS Server

HIGH PERFORMANCE

LOW PERFORMANCE
The Reality of NFV-nodes (COTS Servers)

The cost of flexibility is too high a price!
vhost-user (de-facto)

- VM
  - VNF
  - DPDK
  - Driver virtio

- Container
  - VNF
  - DPDK
  - Driver virtio

- vhost-user

- Virtual Switch
  - DPDK
  - Driver
  - NIC

- User-space to User-space

- Packet copy (each direction)

- Tx zero-copy (optional)
Zero-copy Approaches

**NetVM***
- VM
- VNF
- NetLib
- Emulated PCI
- Packet Pool
- Packet Core Engine
- DPDK
- Driver
- NIC

**Zcopy-vhost***
- VM
- VNF
- NetLib
- Emulated PCI
- Virtual Switch
- DPDK
- Phy-Vir
- Phy-Vir


(H/W) Pass-through Approaches

**SR-IOV**

**VM**
- VNF
- DPDK
- VF Driver

**Container**
- VNF
- DPDK
- VF Driver

**Pass-through**

**Virtual Switch**

- Bypassed
- Dedicated to the physical NIC

- Hairpin routing for inter-guest comm.
## Problem Statements

<table>
<thead>
<tr>
<th>Concerns</th>
<th>Description</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Exposing the host memory</td>
<td>NetVM, IVSHMEM</td>
</tr>
<tr>
<td>Transparency</td>
<td>VNFs are aware of the host environment</td>
<td>NetVM, IVSHMEM, Zcopy-vhost, SR-IOV</td>
</tr>
<tr>
<td>Portability</td>
<td>The method broadly depends on other system components</td>
<td>Zcopy-vhost</td>
</tr>
<tr>
<td>Traceability</td>
<td>Internal behaviors are hidden</td>
<td>SR-IOV</td>
</tr>
<tr>
<td>Container</td>
<td>Container-based VNFs are not supported</td>
<td>IVSHMEM, Zcopy-vhost</td>
</tr>
</tbody>
</table>

The existing methods have pragmatic problems

A yet another practical approach is needed
Proposed Approach (IOVTEE)

- **VM**
  - VNF
  - DPDK
  - Virtio Driver
  - vhost-user

- **Container**
  - VNF
  - DPDK
  - Virtio Driver
  - vhost-user

- **DMA Rx Queue Mapping**
  - DMA-to-VNF

- **Packet Processing on the Host**

- **Vhost-user Interface**

**Proposed Approach**: (IOVTee)

- **Vhost-user Interface**
- **Packet Processing on the Host**
- **DMA Rx Queue Mapping**
- **VM**
  - VNF
  - DPDK
  - Virtio Driver
  - vhost-user

- **Container**
  - VNF
  - DPDK
  - Virtio Driver
  - vhost-user
Step-by-Step Description

1. The points are redirected (Rx Queue Mapping)
2. Packets are stored in the NIC's physical queue
3. Packets are DMAed (DMA-to-VNF)
4. MBufs are created
5. vSwitch's processing
6. vhost-user comm. (Zero-copy)
7. MBufs are created
8. VNF's processing
9. The Rx queue points to the memory buffer
10. The Rx queue points to the memory buffer
### Are Problems Resolved?

<table>
<thead>
<tr>
<th>Concerns</th>
<th>Description</th>
<th>Resolved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Host memory is NOT exposed to VNFs</td>
<td>✓</td>
</tr>
<tr>
<td>Transparency</td>
<td>IOVTee is completely hidden by the vhost-user interface</td>
<td>✓</td>
</tr>
<tr>
<td>Portability</td>
<td>IOVTee is implemented only within the host DPDK internals</td>
<td>✓</td>
</tr>
<tr>
<td>Traceability</td>
<td>IOVTee is a complete software-based approach</td>
<td>✓</td>
</tr>
<tr>
<td>Container</td>
<td>The vhost-user interface is not changed</td>
<td>✓</td>
</tr>
</tbody>
</table>

**IOVTee is a pragmatic zero-copy/pass-through mechanism**

**What about the performance?**
Contents

1. Backgrounds
2. Related Work
3. IOVTEE
4. Evaluation
5. Conclusion
Three Experiments

1. Various Rx Queue Sizes
2. Various Tx/Rx Optimizations
3. Various Packet Sizes

1. Rx Queue Size
2. Tx/Rx Optimizations
3. Packet Size
Environment

Device under Test

**Single CPU core**

**Dual CPU cores (Rx/Tx)**

TCP/IP Stack

100 GbE

**OS**
CentOS 7.5

**DPDK**
v18.02

**VNF**
Testpmd

**vCPU**
4 cores

**Memory**
8 GB

**vNIC**
virtio-net/mlx5

**OS**
CentOS 7.5

**DPDK**
v18.02

**vSwitch**
Testpmd

**VMM**
KVM (QEMU 2.11.0)

**CPU**
Intel Core i9-7900X 3.3 GHz (10 cores w/o HT)

**Memory**
32 GB (DDR4-2133)

**NIC**
Mellanox ConnectX-5 EN

**Driver**
DPDK

**NIC**
MoonGen

**Driver**
DPDK

**NIC**
vSwitch

**Driver**
virtio

**Driver**
DPDK

**VNF**
Testpmd

**Driver**
DPDK

**VNF**
Testpmd

**Driver**
DPDK

**VNF**
Testpmd

**Driver**
DPDK

**VNF**
Testpmd
Exp1: Physical/Virtual Ring Sizes

Default vhost-user

IOVTEE

Lower cache hit ratio

Higher cache hit ratio

Virtual must be greater than Physical
Exp2: Tx/Rx Zero-copy Methods

- Tx: copy
- Tx: zero-copy
- Tx: fake-zero-copy
- Tx: SR-IOV

Overhead of IOVTEE

- Unstable and poor performance
- 20% boost

Throughput [Mpps]

Rx: copy
Rx: fake-zero-copy
Rx: IOVTEE
Rx: SR-IOV

Zero-copy for Rx path is effective

Current implementation could be further optimized
Exp3: Packet Sizes

- 19 Mpps for 64-byte packets
- Worst performance for mid-size packets
- 90 Gbps for 1518-byte packets
- IOVTe is superior for any packet size

Tx: zero-copy/SR-IOV are effective for large-size packets
Baremetal vs. Virtual Machine

What causes this gap?

Zero-copy is effective, but not enough

Basic Forwarding Throughput

(64-byte packets, Single datapath)
Summary

IOVTEE: A yet another Zero-copy/Pass-through method

**Fast**
- 19 Mpps (64-byte)
- 90 Gbps (1518-byte)

**Pragmatic**
- Security
- Portability
- Transparency
- Traceability
- Container

Future Work
- Further optimizing current implementation
- Identifying actual performance bottleneck