Programable Data-plane for Multi-Tenant Network Services at Scale

Sherif Abdelwahab
Futurewei, Cloud-Lab

June 12th, 2020
Platform-Lab retreat
Dynamic Cloud Applications

- Short lifetime: From Days/Hours to minutes and fraction of seconds
- Large Scale:
  - 300K+ Hosts
  - 1K - 1M Interconnected Applications (Tasks)
  - Elastic Scaling of network services
  - Sub-RTT fail-over
- Multitenancy:
  - Address Separation
  - Shared network services
- Failure Resiliency
  - Why flow-programming may not be enough to scale?
  - Mizar-Project: [https://github.com/futurewei-cloud/mizar](https://github.com/futurewei-cloud/mizar)
Problems with programmers thinking in flow-rules

- Example from OVN/OVS
- 10K logical ports generates >40K and 10K port-bindings
- High CPU utilization during flow-parsing
- Time to provision ports increases significantly as the number of ports increases.
- Provisioning time of a new container, depends on the number of containers already existing in the system.

Most updated OVN Architecture for Kubernetes

- https://raw.githubusercontent.com/ovn-org/ovn-kubernetes/
Several ways to approach this problem

• Optimize SQL, databases, parallel threads, and REST APIs

• Optimize OVN-controller!
  • *Fantastic* efforts from the OVN/OVS community over the last year!
  • Adding a single port reduced from tens of seconds close to a second

• Flow monitoring and per-flow fast-path (Andromeda*):
  • A flow-rule to send all packets to the hoverboard
  • Monitor the flows, determine which ones to install flow-rules for!

---

Finding ways to reduce the number of flow-rules passed down to data-plane

• **What shall we do to update an overlay network with:**
  • One $O(1)$ RPC call to hosts
  • One single configuration (No flow-rules)

• **If we never had flow-rules, What does it take to program “the host data-plane” as a regular application?**

*Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization, Michael Dalton et al, NSDI’18*
**Background:** eXpress Data Path (XDP) – A Linux Kernel Superpower

- Safely and Dynamically modify the NIC device driver behavior without packet processing interruption
- Process Packets before delivering it to the stack • PASS, TX, REDIRECT, DROP
- API interfaces that programmers understand!
- Does not require dedicated CPUs and Off-loadable to SmartNICs
- Small programs 4K ebpf instructions!

The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel

- Toke Holiland-Jørgensen
  Karlstad University
toke@toke.dk
- John Fastabend
  Cilium.io
  john@cilium.io
- Jesper Dangaard Brouer
  Red Hat
  brouer@redhat.com
- Tom Herbert
  Quantumum Inc.
tom@herbertland.com
- Daniel Borkmann
  Cilium.io
daniel@cilium.io
- David Ahern
  Cumulus Networks
dahern@gmail.com
- David Miller
  Red Hat
drivem@redhat.com

https://www.iovisor.org/technology/xdp
**Mizar: Host Networking Architecture**

- One XDP Program attached to NIC
  - Processes all ingress packets
- One XDP program attached to the veth-pair of a container
  - Process egress packets from that container
- Expose RPC interface to the management plane
  - Load/Unload the XDP programs
  - Push any form of configuration to ebpf maps
Mizar: Host Networking Architecture
**Mizar:** We now see all the **Network Interfaces** as programmable tiny-servers
Mizar: **Label** one of the hosts as bouncer
Mizar: Two RPC Calls whenever a container is created

RPC Call
(Bouncer is Host-B)

RPC Call
(10.0.0.1 @ Host-A)

XDP Host-A

10.0.0.1
Bouncer = Host-B

Endpoints Map
Compute  Host

Endpoints Map
Compute  Host

Endpoints Map
Compute  Host

XDP Host-B
(Label: Bouncer)

XDP Host-C

10.0.0.1 Host-A

10.0.0.1
Bouncer = Host-B

10.0.0.1

RPC Call
(Bouncer is Host-B)

RPC Call
(10.0.0.1 @ Host-A)
Mizar: Two RPC Calls whenever a container is created

- SunRPC
- 20 ms To update the Bouncer
- <300 ms to provision the container
Mizar: Packet Example

SRC: 10.0.0.1 DST: 10.0.0.2

RX

Transit Agent at host A
(Bypass host stack)

XDP_REDIRECT (SKB)

Transit XDP at Host B
(In interface device driver - host stack bypassed)

XDP_TX

Transit XDP at Host C
(In interface device driver - host stack bypassed)

XDP_REDIRECT

SRC: 10.0.0.1 DST: 10.0.0.2

RX

Endpoint at host C
Mizar: Adding an ARP Responder + Direct-Path Protocol

- Modified the Transit Agent packet encapsulation to add a Geneve option
Mizar: Direct Path – ARP Packet

SRC: 10.0.0.1 | DST: 10.0.0.2 | ........

Endpoint at host A

Transit Agent at host A
(Bypass host stack)

XDP_REDIRECT (SKB)

Transit XDP at host B (Bouncer)
(In interface device driver - host stack bypassed)

XDP_TX

Transit XDP at host A
(In interface device driver - host stack bypassed)

XDP_REDIRECT

Endpoint at host A

SRC: 10.0.0.1 | DST: 10.0.0.2 | ........

SRC: host A | DST: host B | GENEVE VNI: 3 | ARP Query: Target: 10.0.0.2?

10.0.0.1 @ Host-A

SRC: host B | DST: host A | GENEVE VNI: 3 | ARP Response for 10.0.0.2

10.0.0.2 @ Host-B
Mizar: Adding an ARP Responder + Direct-Path Protocol

<table>
<thead>
<tr>
<th>Endpoints Map</th>
<th>Compute</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDP Host-A</td>
<td>10.0.0.1</td>
<td>Bouncer = Host-B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Endpoints Map</th>
<th>Compute</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDP Host-B</td>
<td>10.0.0.2</td>
<td>Host-C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Endpoints Map</th>
<th>Compute</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDP Host-C</td>
<td>10.0.0.2</td>
<td>Bouncer = Host-B</td>
</tr>
</tbody>
</table>

10.0.0.2 Host-C
10.0.0.1 Host-A
10.0.0.2 Host-C

ARP Responder
**Mizar: Direct Path – First Packet of the Flow**

1. **Endpoint at host A**
   - **RX**
   - **Transit Agent at host A** (Bypass host stack)

2. **Transit XDP at host A**
   - **In interface device driver - host stack bypassed**
   - **XDPI_REDIRECT**
   - **Endpoint at host C**

| SRC: host A | DST: host C | GENEVE VNI: 3 | DIRECT-FLAG | SRC: 10.0.0.1 | DST: 10.0.0.2 | ........ |
Mizar: Direct Path – Updated Endpoints Maps

- Provisioned the endpoint with $O(1)$ RPC calls
- All packet processing is Fast-Path, Mizar does not send packets to SDN/OpenFlow controllers
- Containers send/receive packets directly almost all the time
- There is no need for a per-flow monitoring, or flow size analysis (Like for e.g. in Andromeda*)

*Andromeda: Performance, Isolation, and Velocity at Scale in Cloud Network Virtualization, Michael Dalton et al, NSDI’18
Mizar: Host Extensibility

Shared ebpf maps

Primary-XDP
Attached to
NIC
RX
Tail-Call

Module-XDP-1
Module-XDP-2
Module-XDP-N

Updated from management
or data-plane
Userspace or Kernel-space

Input Packet

TX
Output Packet

Loaded and
Modified with RPC Calls

Userspace or Kernel-space
Mizar: Host Extensibility

IP Rewriter
XDP Program Written in Rust

Cargo-ebpf

.elf

RPC Call
Load XDP Program (IP-Rewriter)

Endpoints Map
Compute: Host-A
10.0.0.1
Bouncer = Host-B

Endpoints Map
Compute: Host-A
10.0.0.1
Host-A
10.0.0.2
Host-C
10.0.0.5
Tail-Call (fd)

Endpoints Map
Compute: Host-A
10.0.0.1
Host-A
10.0.0.2
Host-C

 ARP Responder

Packet Rewriter XDP

(RPC Call)

Load XDP Program (IP-Rewriter)

Endpoints Map
Compute: Host-B
10.0.0.1
Bouncer = Host-B
10.0.0.2
Bouncer = Host-B

Endpoints Map
Compute: Host-B
10.0.0.1
Bouncer = Host-B

Endpoints Map
Compute: Host-C
10.0.0.2
Bouncer = Host-B
Mizar: Host Extensibility – Managing flow state

RPC Call
Load XDP Program
(IP-Rewriter)

<table>
<thead>
<tr>
<th>XDP Host-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.1</td>
</tr>
<tr>
<td>Bouncer = Host-B</td>
</tr>
</tbody>
</table>

Endpoints Map

<table>
<thead>
<tr>
<th>Compute</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XDP Host-B (Label: Bouncer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow Table</td>
</tr>
<tr>
<td>Packet Rewriter XDP</td>
</tr>
</tbody>
</table>

ARP Responder

<table>
<thead>
<tr>
<th>XDP Host-C</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.2</td>
</tr>
<tr>
<td>Bouncer = Host-B</td>
</tr>
</tbody>
</table>

Endpoints Map

<table>
<thead>
<tr>
<th>Compute</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Endpoints Map

<table>
<thead>
<tr>
<th>Compute</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.1</td>
<td>Host-A</td>
</tr>
<tr>
<td>10.0.0.2</td>
<td>Host-C</td>
</tr>
<tr>
<td>10.0.0.5</td>
<td>Tail-Call (fd)</td>
</tr>
</tbody>
</table>
Mizar: Problem with stateful network services

Before Scaling
- Dest-1
- Dest-2
- Dest-3
- Middlebox-1
- Middlebox-2
- Flow-1
- Flow-2
- Flow-3

After Scaling
- Dest-1
- Dest-3
- Dest-5
- Middlebox-1
- Middlebox-2
- Middlebox-3
- Flow-1
- Flow-2
- Flow-3

Incorrect Destination After Scaling Out. Connection must be re-established.
Correct State of Flow-2
Mizar: State Replication

- **NetChain**: Scale-Free Sub-RTT Coordination, Xin Jin et al, NSDI’18
- Fault tolerance for service function chains, Milad Ghaznavi et al, preprint 2020

### Diagram

- **XDP Program**
- **No Packet buffering**

---

*NetChain: Scale-Free Sub-RTT Coordination, Xin Jin et al, NSDI’18*

- Fault tolerance for service function chains, Milad Ghaznavi et al, preprint 2020
Mizar: Generalizing state-tracking with in-network distributed flow-table

* Maglev: A Fast and Reliable Software Network Load Balancer, Daniel E. Eisenbud et al, NSDI’16
Mizar: Endpoint Update Time with multiple Bouncers

![Bar chart showing Bouncer Endpoint update time](chart.png)
**Mizar: CPU – Provisioning 100 Endpoints**

### 100 Endpoint Provisioning CPU (lower the better)

- **Mizar (Bouncer)**
- **Mizar (Endpoint Host)**
- **OVS+Linux Bridge (Direct)**

![Chart showing CPU usage for different scenarios](chart_image)
**Mizar: Memory Usage During TCP Performance Tests**

### Bar Chart: TCP Traffic Memory (% 500GB - lower the better)

- **Mizar (Direct/Fast-Path)**
- **Mizar (1 Bouncer)**
- **OVS+Linux Bridge (Direct)**
- **Host to Host**

- **HIT**: Negligible Memory overhead very close to an idle host without networking constructs event with Traffic processing
**Mizar:** CPU Utilization during TCP Performance Tests

<table>
<thead>
<tr>
<th>SCENARIO</th>
<th>TCP Traffic CPU (lower the better)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mizar (Direct/Fast-Path)</td>
<td>![Mizar (Direct/Fast-Path)]</td>
</tr>
<tr>
<td>Mizar (1 Bouncer)</td>
<td>![Mizar (1 Bouncer)]</td>
</tr>
<tr>
<td>OVS+Linux Bridge (Direct)</td>
<td>![OVS+Linux Bridge (Direct)]</td>
</tr>
<tr>
<td>Host to Host</td>
<td>![Host to Host]</td>
</tr>
</tbody>
</table>

- **HIT:** CPU Overhead is much better than OVS + Linux bridge scenario
Mizar: Packet Rate Measured from Containers

Packet Rate (Mpps - higher the better)

- Mizar (Direct/Fast-Path)
- Mizar (1 Bouncer)
- OVS+Linux Bridge (Direct)
- Host to Host

<table>
<thead>
<tr>
<th>SCENARIO</th>
<th>Packet Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mizar (Direct/Fast-Path)</td>
<td>0.6</td>
</tr>
<tr>
<td>Mizar (1 Bouncer)</td>
<td>0.6</td>
</tr>
<tr>
<td>OVS+Linux Bridge (Direct)</td>
<td>0.3</td>
</tr>
<tr>
<td>Host to Host</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Mizar: TCP Throughput (On a slow NIC 1Gbps)

**TCP Bandwidth (Mbps - higher the better)**

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mizar (Direct/Fast-Path)</td>
<td>900</td>
</tr>
<tr>
<td>Mizar (1 Bouncer)</td>
<td>860</td>
</tr>
<tr>
<td>OVS+Linux Bridge (Direct)</td>
<td>900</td>
</tr>
<tr>
<td>Host to Host</td>
<td>940</td>
</tr>
</tbody>
</table>

TCP Throughput (On a slow NIC 1Gbps)
Mizar: TCP Throughput (On a faster NIC 10Gbps) - Limitations

- Suboptimal Resource Management in XDP veth driver implementation
- No support for TSO/GSO in XDP yet
- XDP in TX path

![Graph showing TCP bandwidth comparison between Mizar (1 Bouncer), OVS+Linux Bridge (Direct), and Maximum Bandwidth. The x-axis represents Gbps with higher values being better, and the y-axis shows the scenarios.]
Mizar: Interests

- XDP Community efforts we are interested in: https://xdp-project.net
  - Multi-buffer XDP (jumbo-frames)
  - XDP in TX-path
  - XDP Redirect Optimizations
  - TSO/GSO support in TX path or Veth RX

- SmartNICs Hardware Offloading:
  - Hardware offloading (Tested Netronome, looking forward for tests on Broadcom PS225)
  - New NIC Architectures (nanoPU)?
  - Performance difference of jitting ebpf to Risc-V or ARM?

- Protocol implementations:
  - Is it possible to implement network protocols in XDP (Homa)?

- Auto-Scaling of Stateful network Services?

- Standardized API interfaces

- OVS extensions with ebpf support (There is already progress on that within the OVS Community)
The flow-programming model is great for programmable switches but not scalable for multi-tenant cloud networks

Programming the SmartNics with small, safe, and dynamically loadable programs enable the management-plane to scale overlay networks

Distributed System Techniques in the Network:
- Chain-Replication of connection states
- Consistent Hashing (Maglev-Hashing)
- In-network Key/Value stores

Extensions to Mizar to implement stateful network services: https://github.com/futurewei-cloud/mizar