MIZAR

Futurewei Technologies

Current state and work in progress

https://github.com/futurewei-cloud/mizar

THE PROBLEM

Support provisioning of large number endpoints (10M)

Achieve high network routing throughput and low latency

Create an extensible plugin framework for cloud network

Unify the network data-plane for VMs, Containers, Serverless and others

THE PROBLEM WITH CURRENT SOLUTIONS

Program every host every time a user provisions an endpoint:

Approaching cloud-networking with a conventional programming model and network devices

e.g. OpenFlow programming in OVS

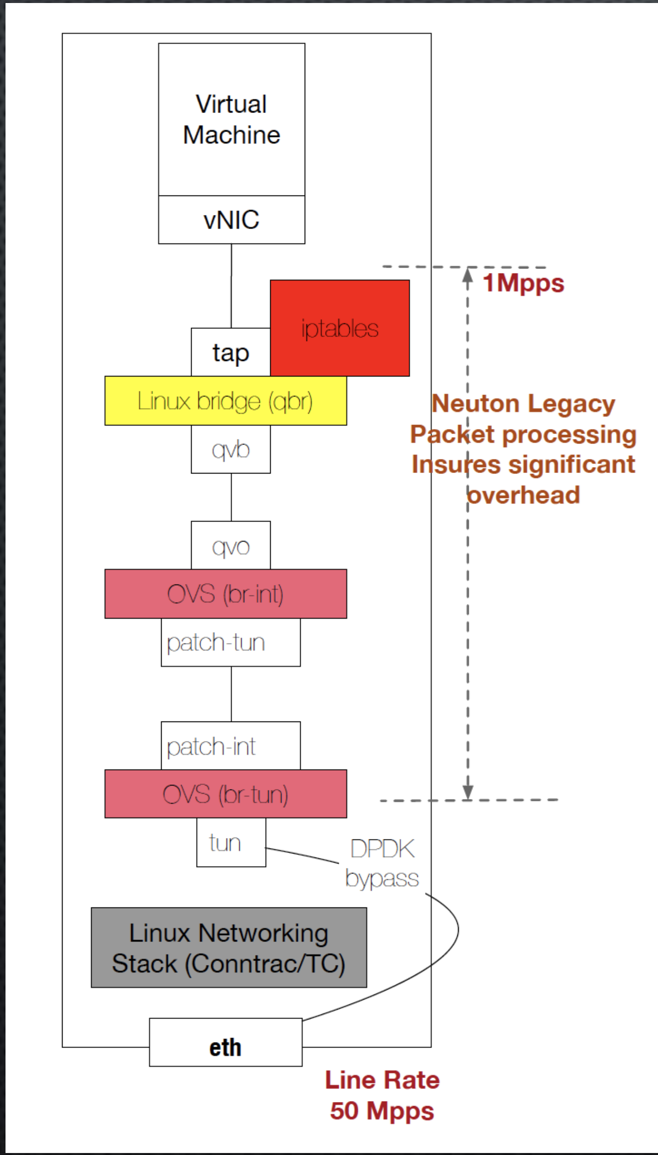
Virtual Switches and Routers are essentially softwarization of hardware switches and routers, but not necessarily programmable to support rapid network changes.

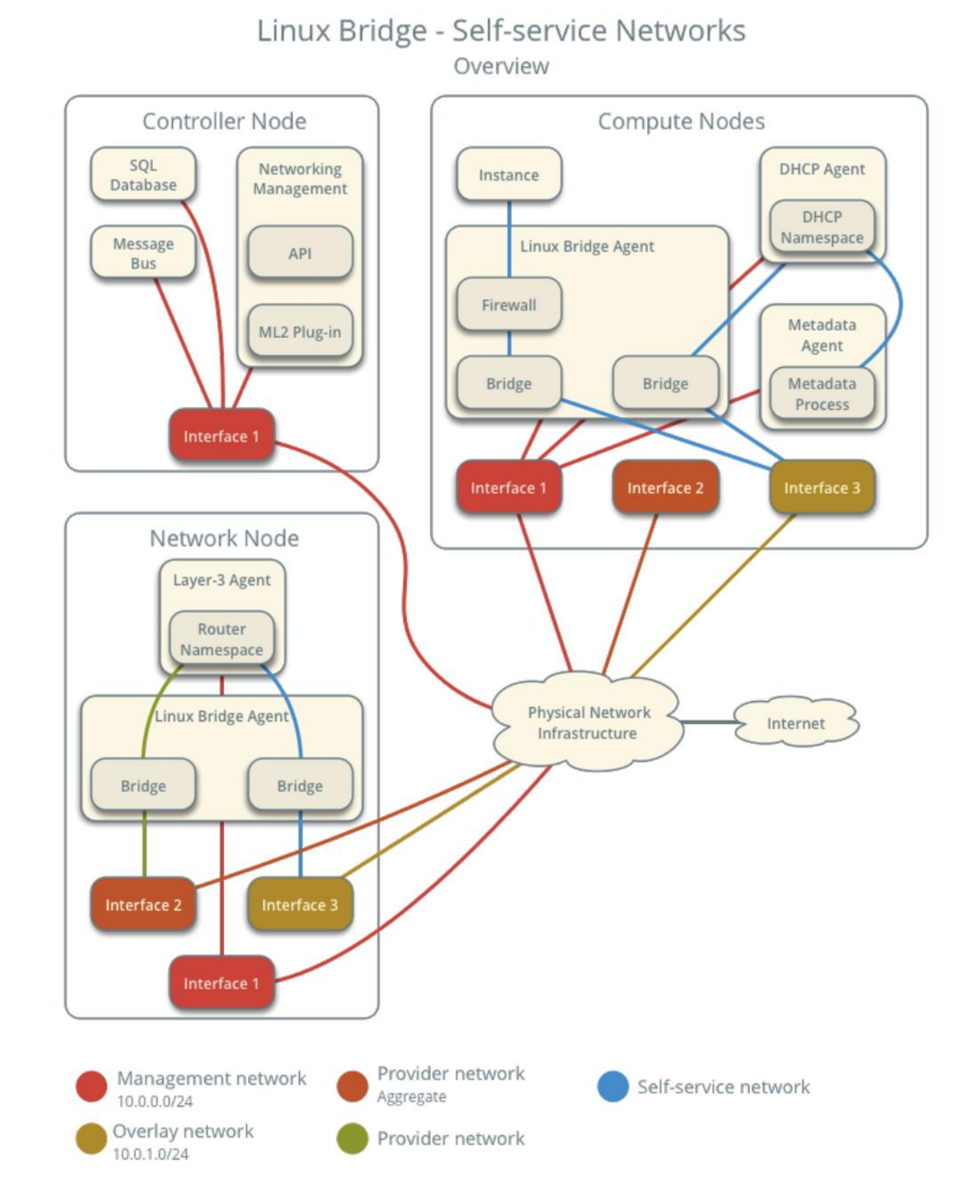
Existing solutions bring up software network devices, that are primarily created for Telecom, ISP, or datacenter networking and run them in virtual machines to support cloud networking.

Packets traverse multiple network stacks on the same host

Packets traverse multiple network devices (as if we are operating a data-center), while these functions could be consolidated during design.

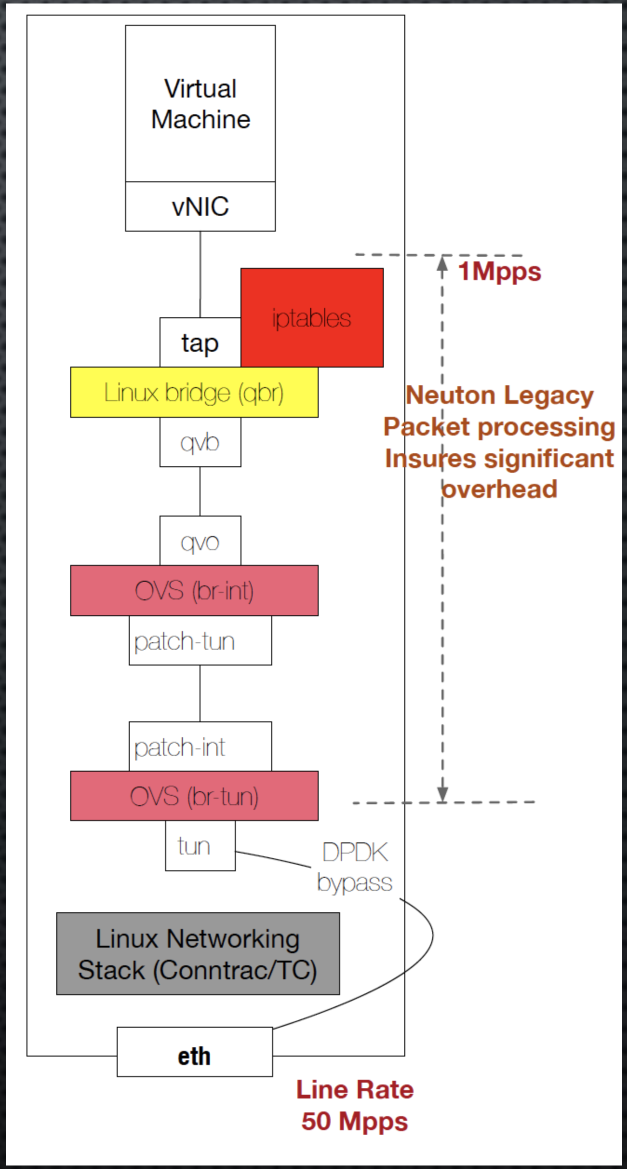
THE PROBLEM WITH CURRENT SOLUTIONS (E.G NEUTRON)





OBSERVATION: (NOT REALLY A NEW ONE)

- In a cloud network (overlay), most functions can be reduced to
  1. Encapsulate/decapsulate a packet
  2. Modify the outer packet header and forward it
  3. Modify the inner packet header and forward it
  4. Drop unwanted packets
- Several network functions can be thought of in a similar way:
  1. Responding to ARP
  2. DHCP
  3. NAT
  4. Passthrough load-balancing



MIZAR OVERALL ARCHITECTURE!

Natural Partitioning domains of Cloud Network

Virtual Private Cloud VPC

Networks within a VPC

Endpoints within a network

Bouncer

In-network hash tables

Holds the configuration of endpoints within a network

Determines flow modifications, and bounce the packet for TX

Implements all middleboxes within a network

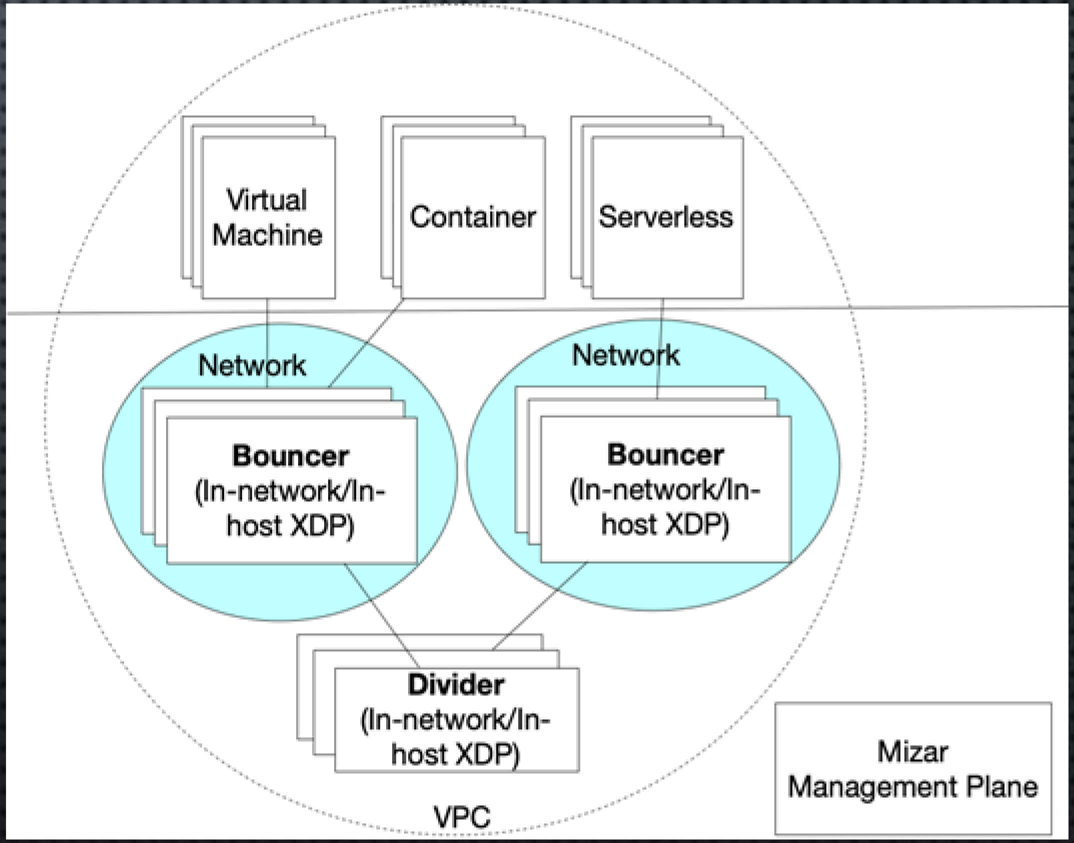
Divider

In-network hash tables

Holds the configuration of Bouncers within a VPC

Divides the VPCs endpoint's configuration into clusters of Bouncers

Implements all middleboxes at the VPC level



MIZAR OVERALL ARCHITECTURE!

Natural Partitioning domains of Cloud Network

Virtual Private Cloud VPC

Networks within a VPC

Endpoints within a network

Bouncer

In-network hash tables

Holds the configuration of endpoints within a network

Determines flow modifications, and bounce the packet for TX

Implements all middleboxes within a network

Divider

In-network hash tables

Holds the configuration of Bouncers within a VPC

Divides the VPCs endpoint's configuration into clusters of Bouncers

Implements all middleboxes at the VPC level

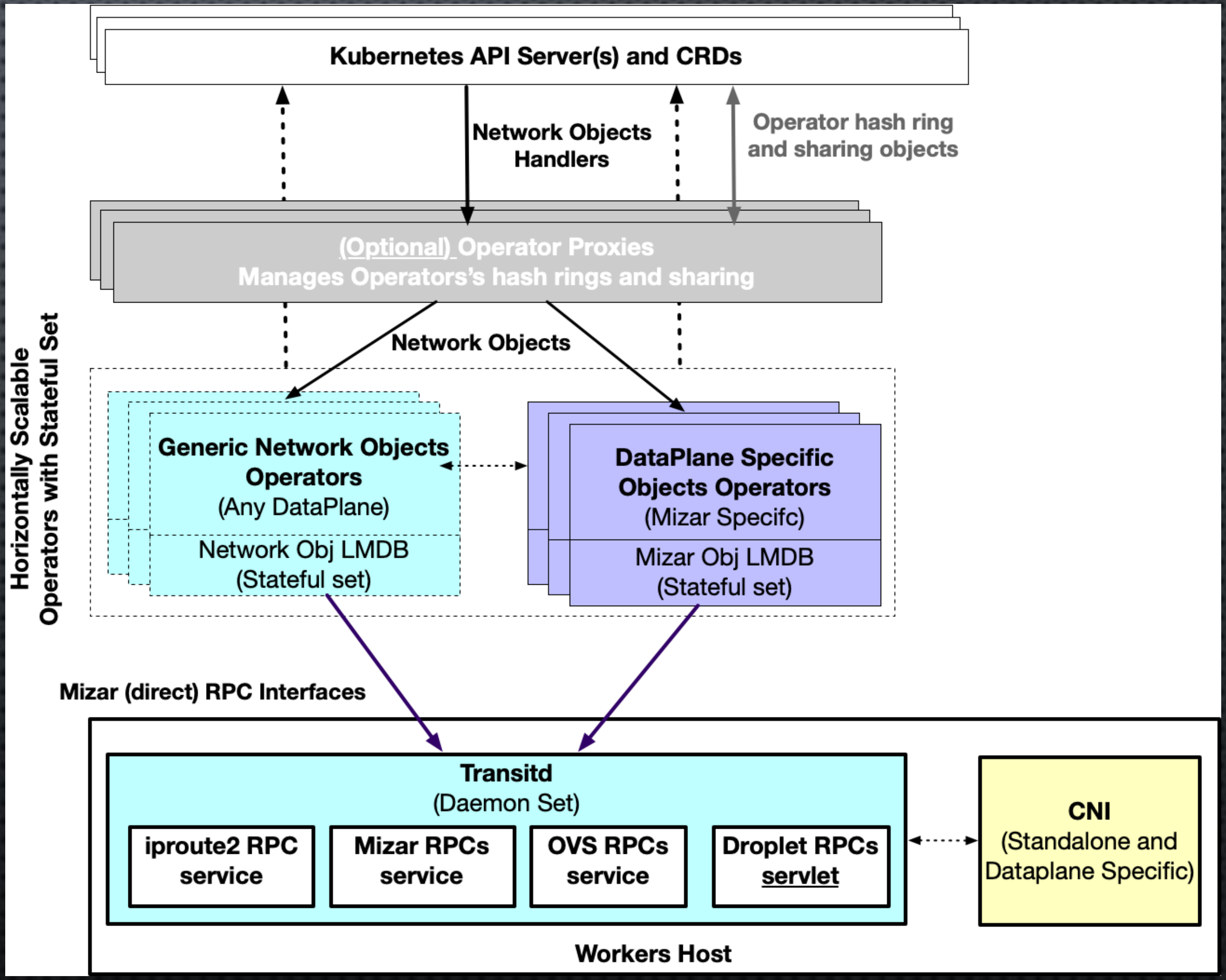
MANAGEMENT PLANE ARCHITECTURE

Kubernetes CRDS

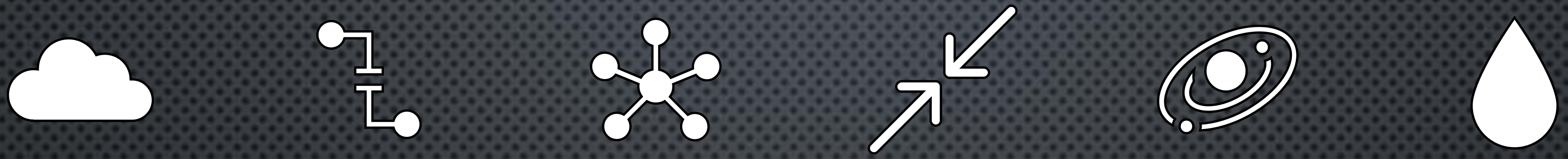
Allows us to extend the K8s API server with networking objects. Some of these objects are generic to any networking solution and some of them are specific to Mizar.

Operator Framework

Extending Kubernetes with domain-specific operators that *act as a controller* for CRDs. Operator framework allow us to write custom lightweight operators that derive the networking objects life-cycle.



OPERATORS, CRDS, AND WORKFLOWS



All objects are defined through Kubernetes CRDs under the mizar.com resource group. An operator exposes interfaces to the management-plane workflows.

The lifecycle of an object is governed by three workflows: create, update, and delete. Each of these workflows are triggered by state changes in the respective object. For example, the droplet object which represents a physical interface on a node Will trigger the management-plane delete workflow if the physical interface itself were to be removed.

THE CRDS

VPC

- Contains information about the VPC such as CIDR range, and list of dividers

Divider

- Contains information about the divider such as parent VPC, and host information (IP, MAC, etc)

Network

- Contains information about the Network such as parent VPC, CIDR range, and bouncers.

Bouncer

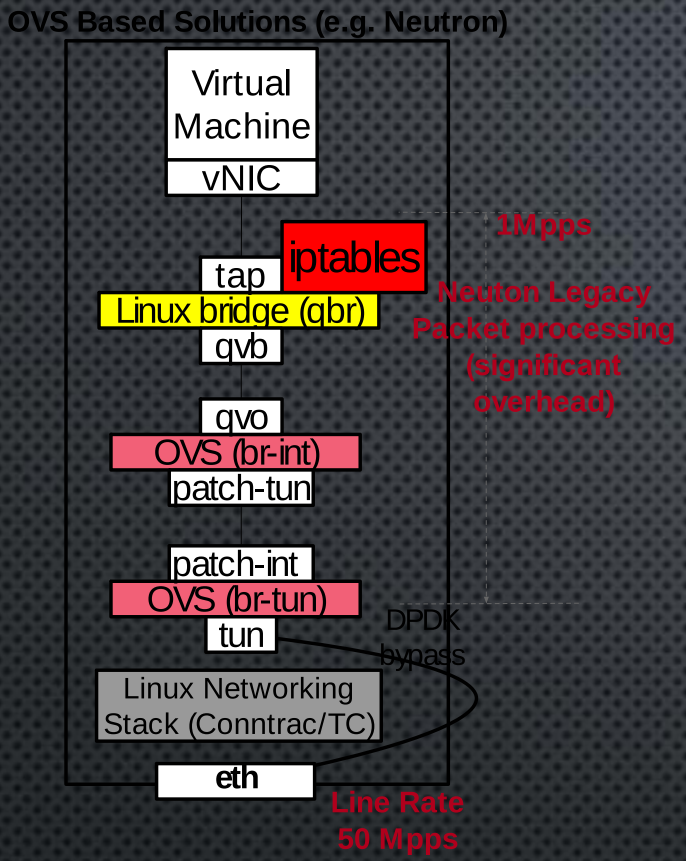
- Contains information about the bouncer such as parent Network, and host information (IP, MAC, etc)

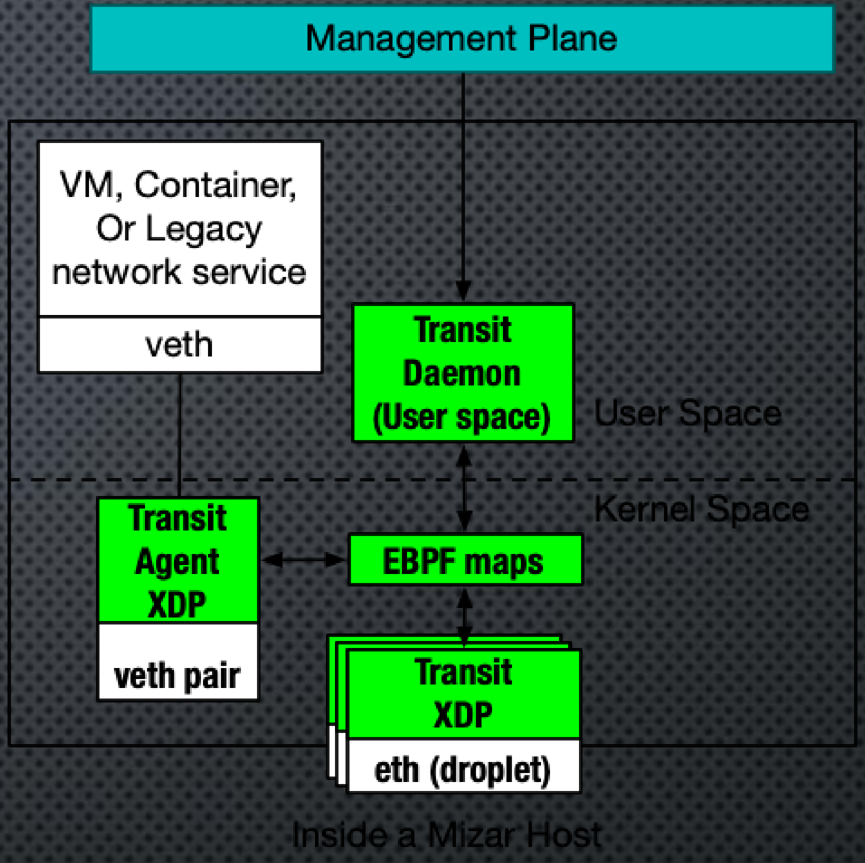
Endpoint

- Contains information about the bouncer such as parent Network, and host information (IP, MAC, etc)

Droplet

- Contains information about the current host interface (IP, MAC, etc_





INSIDE A MIZAR HOST

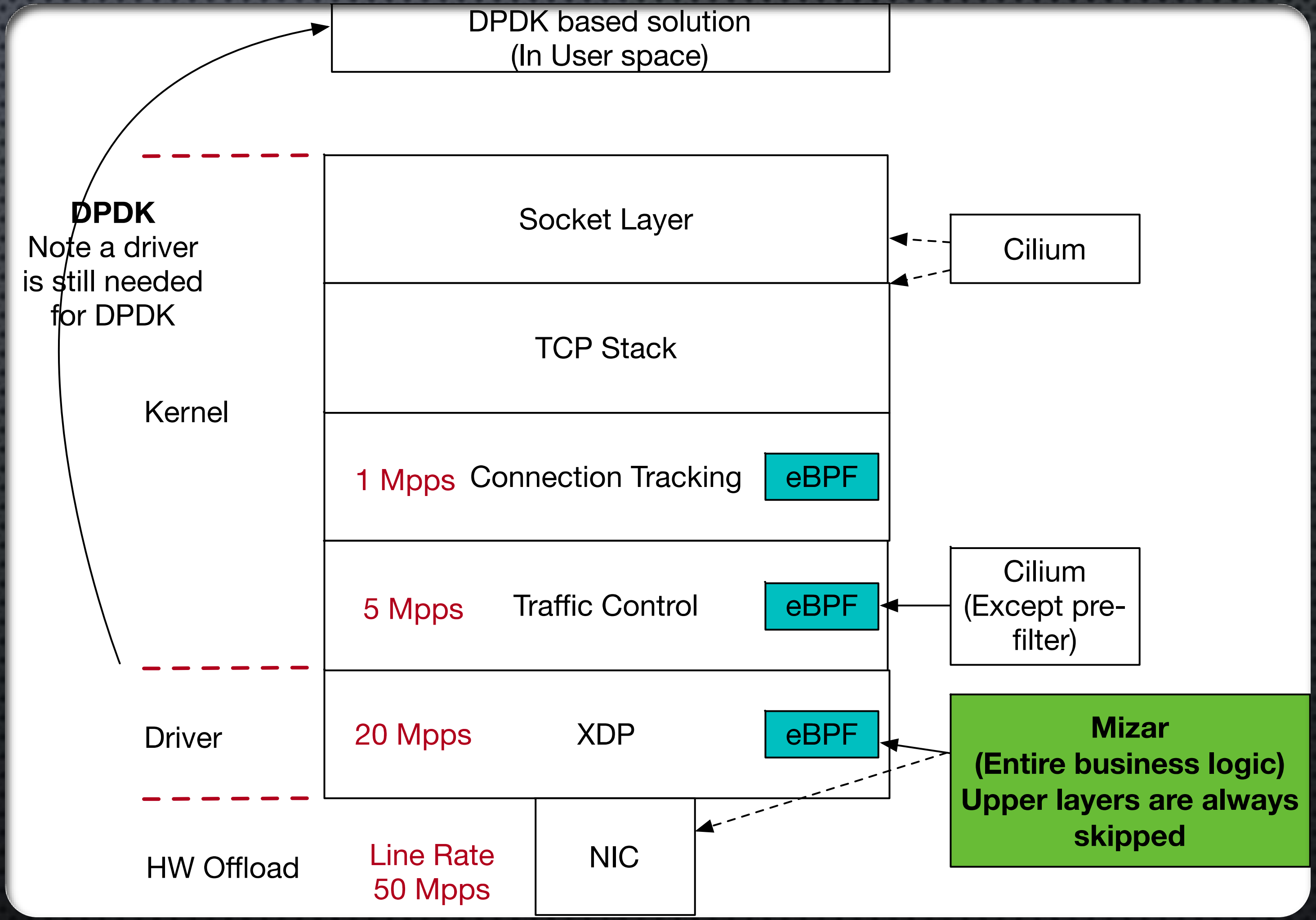
BACKGROUND XDP: SIMPLIFIED AND EXTENSIBLE PACKET PROCESSING NEAR LINE RATE

Skip unnecessary stages of network stack whenever possible and transit packet processing to smart NICs.

Makes the best use of kernel packet processing constructs without being locked in to a specific processor architecture.

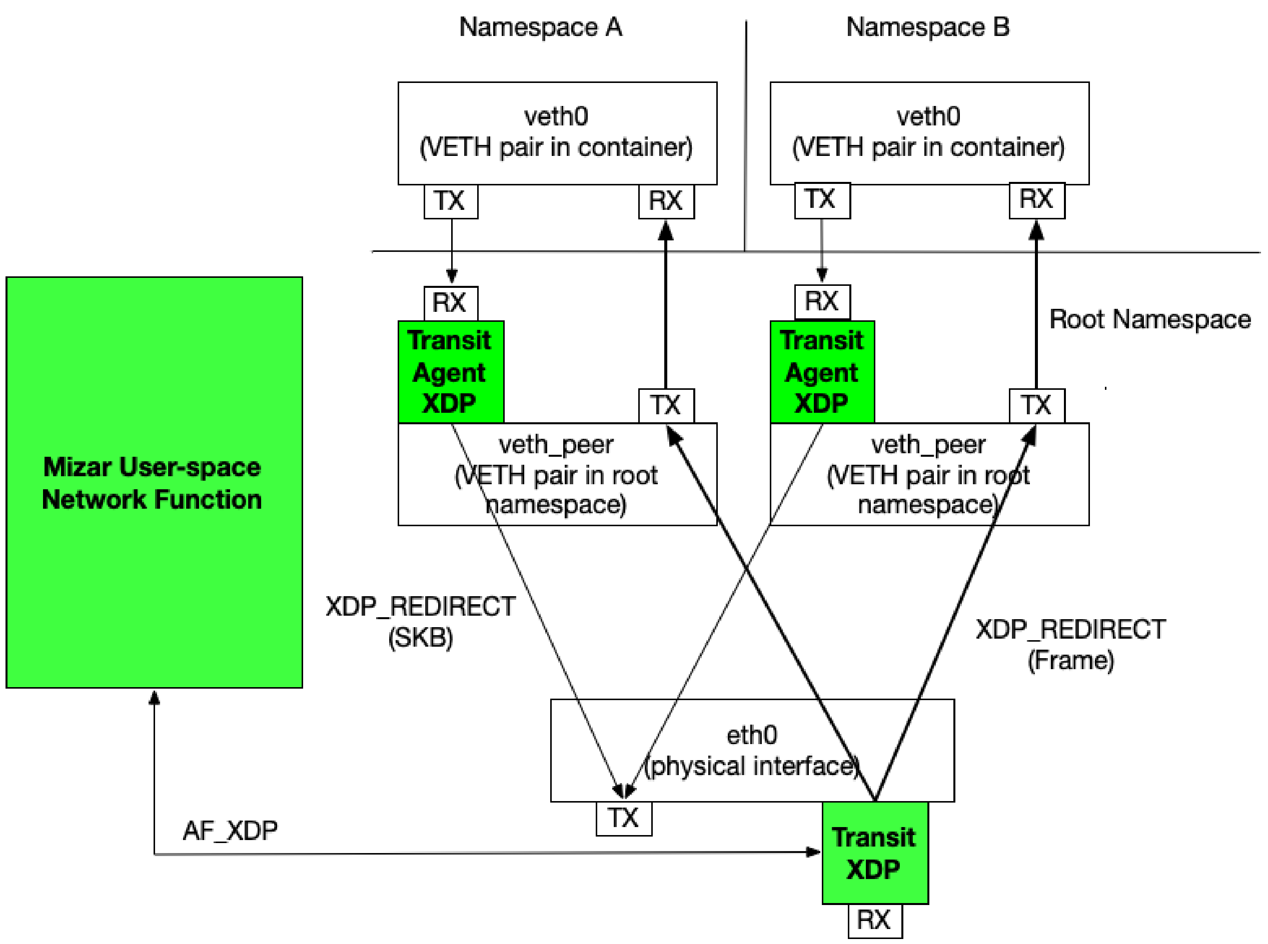
Packet processing is entirely in-kernel.

Very small programs < 4KB



IN-HOST PACKET FLOW: BYPASS NETWORK STACK

Packets traverses only the container stack

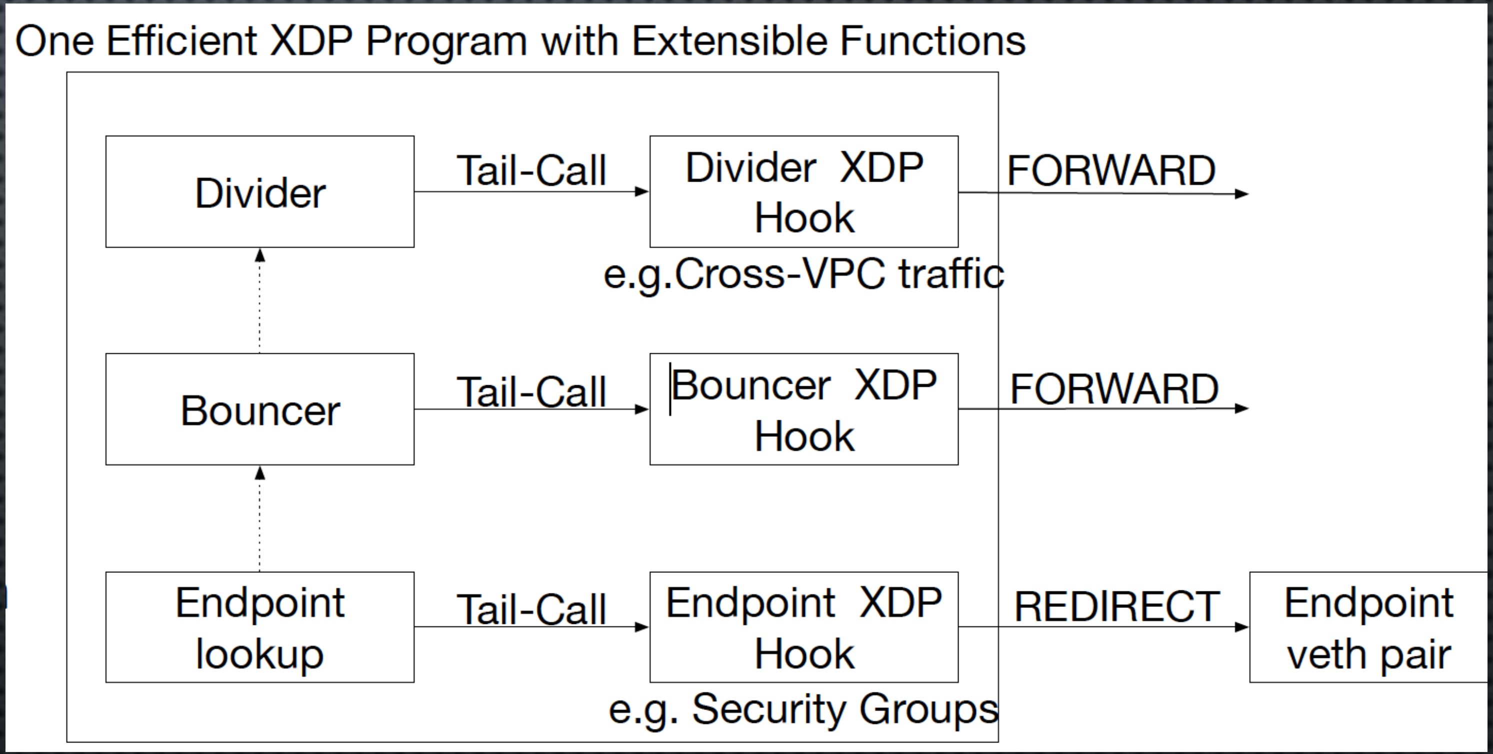


EXTENSIBLE PACKET PROCESSING INSIDE THE MAIN XDP PROGRAM

- Implements essential logical networking function within the same XDP program that provides multitenant cloud networking solutions through new Bouncer and Divider concepts

- Mizar autonomously adapts to various traffic demands in immense scale cloud environments. Allowing Mizar to serve various cloud workloads in a multi-tenant environment optimally.

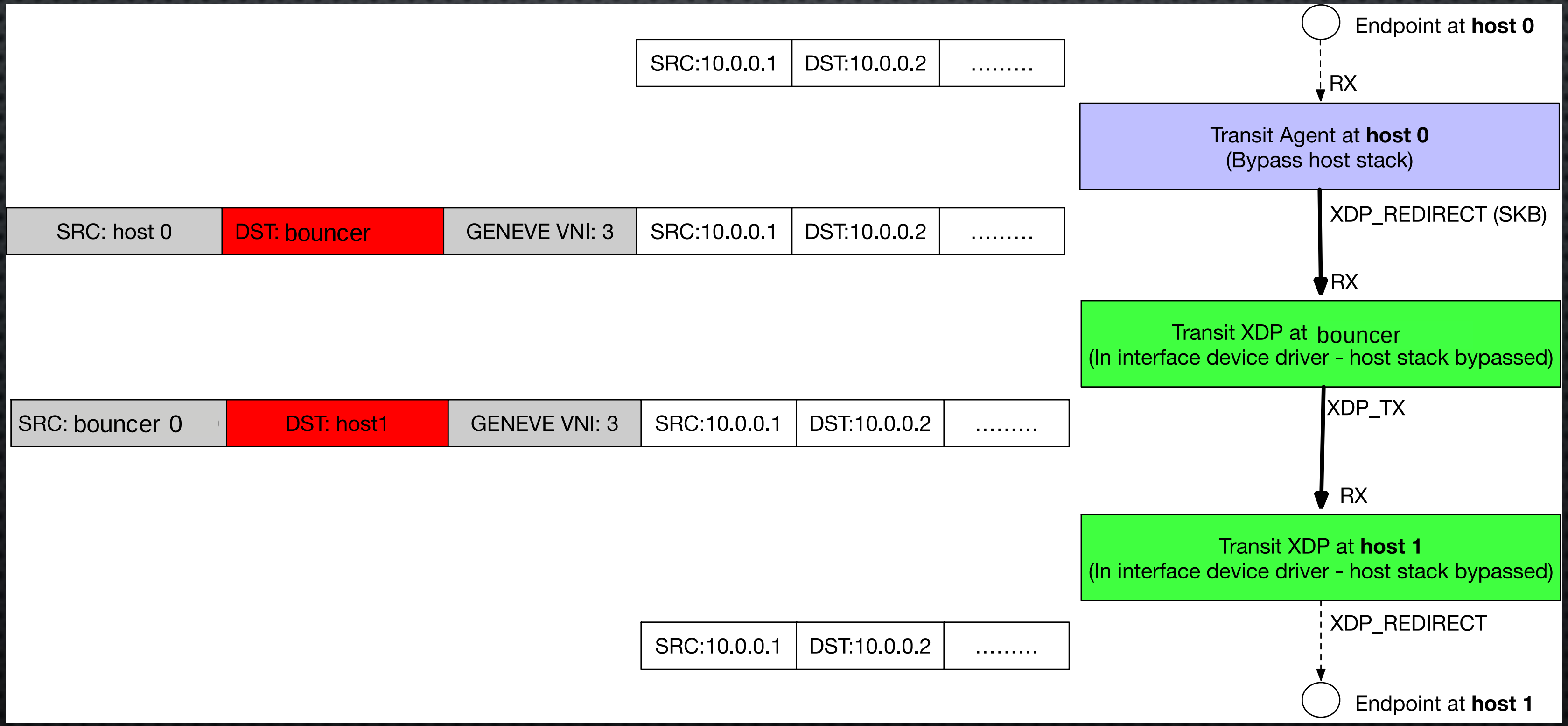
- Extensible support of native networking features through custom chains of optimized XDP programs hooks and Geneve protocol options. Future possible Features including: Security, Loadbalancing, Connectivity, Traffic Shaping Control



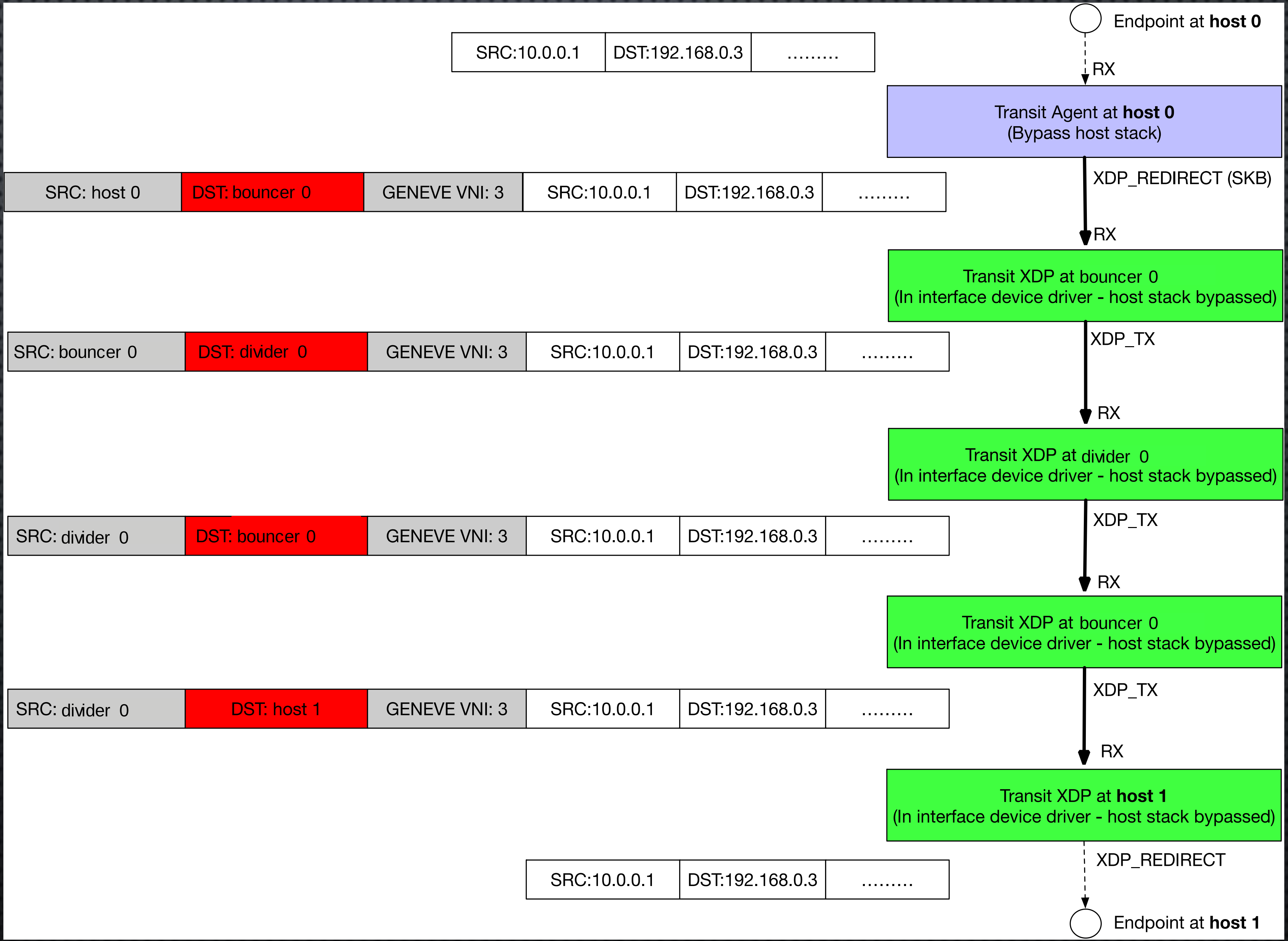
EXAMPLE PACKET WITHIN A NETWORK

Three steps to provision an endpoint

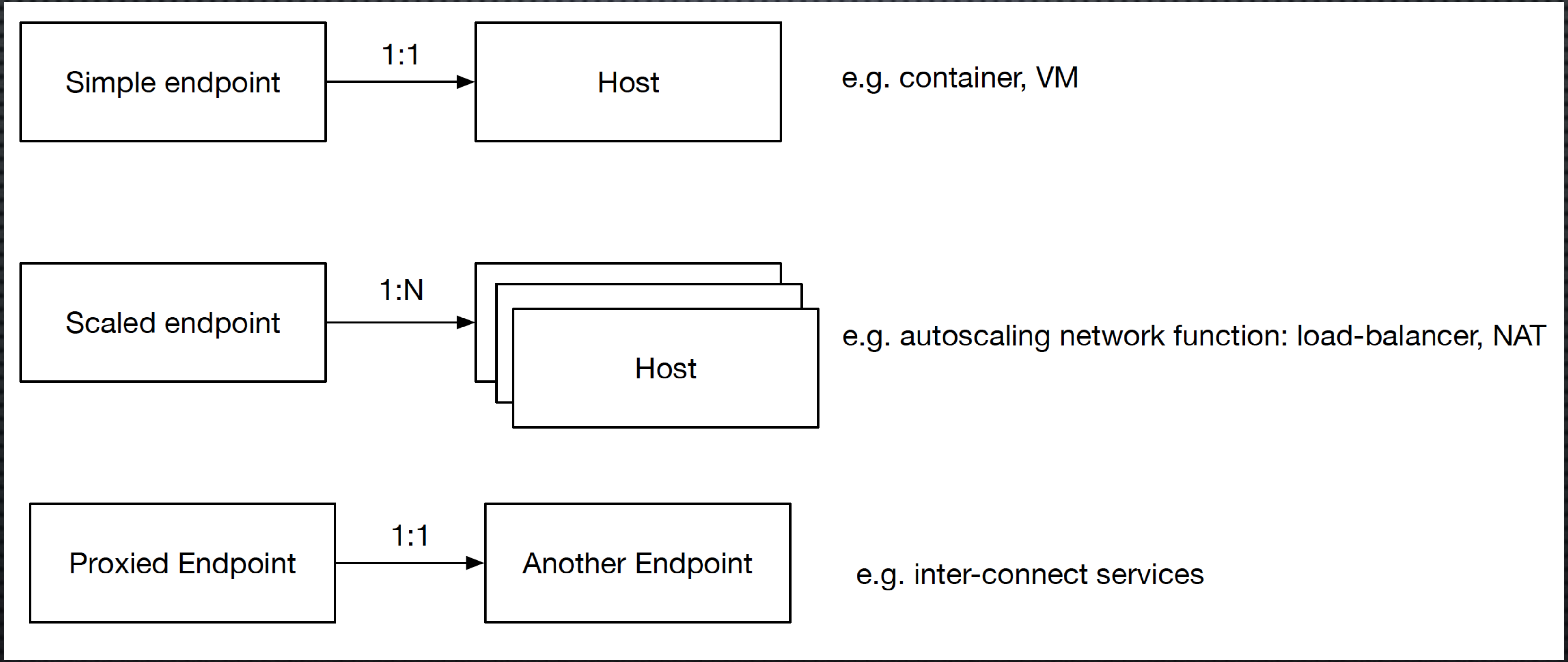
- Add the endpoint to N Bouncer table
- Provision the endpoint on the host
- Configure the host transit agent to tunnel the endpoint traffic to the Bouncer



CROSS NETWORK



NEW ENDPOINT TYPES



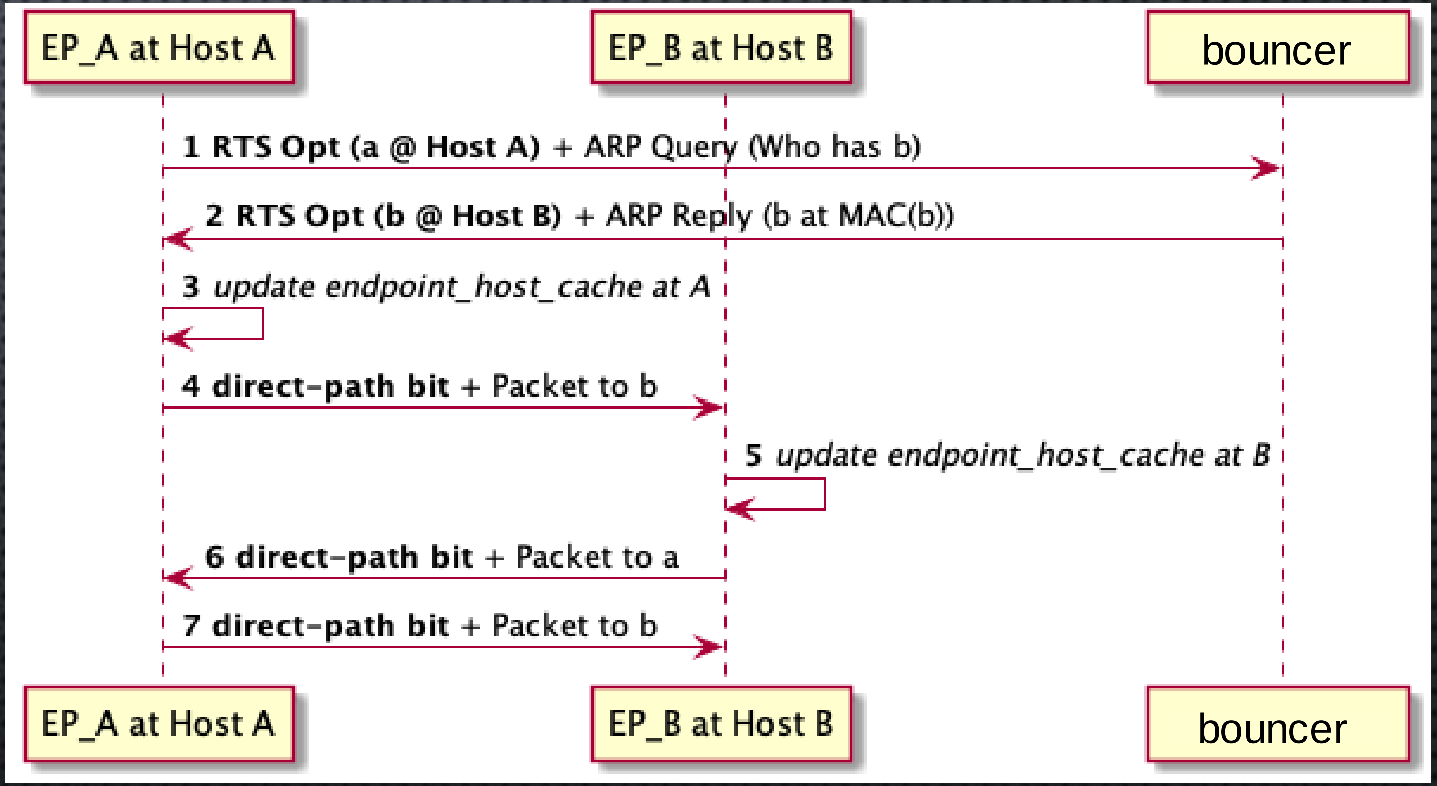
DIRECT PATH

The direct path feature allows a flexible data-plane fast path that avoids multiple hops of flows through bouncers. Endpoints shall be able to communicate directly between their hosting nodes without the need to go through multiple intermediaries.

DIRECT PATH: INTRA-NETWORK

To support the direct path feature, the transit agent may add a Return To Sender (RTS) option to the outer packet header. The option carries information about the endpoint's host IP and MAC addresses.

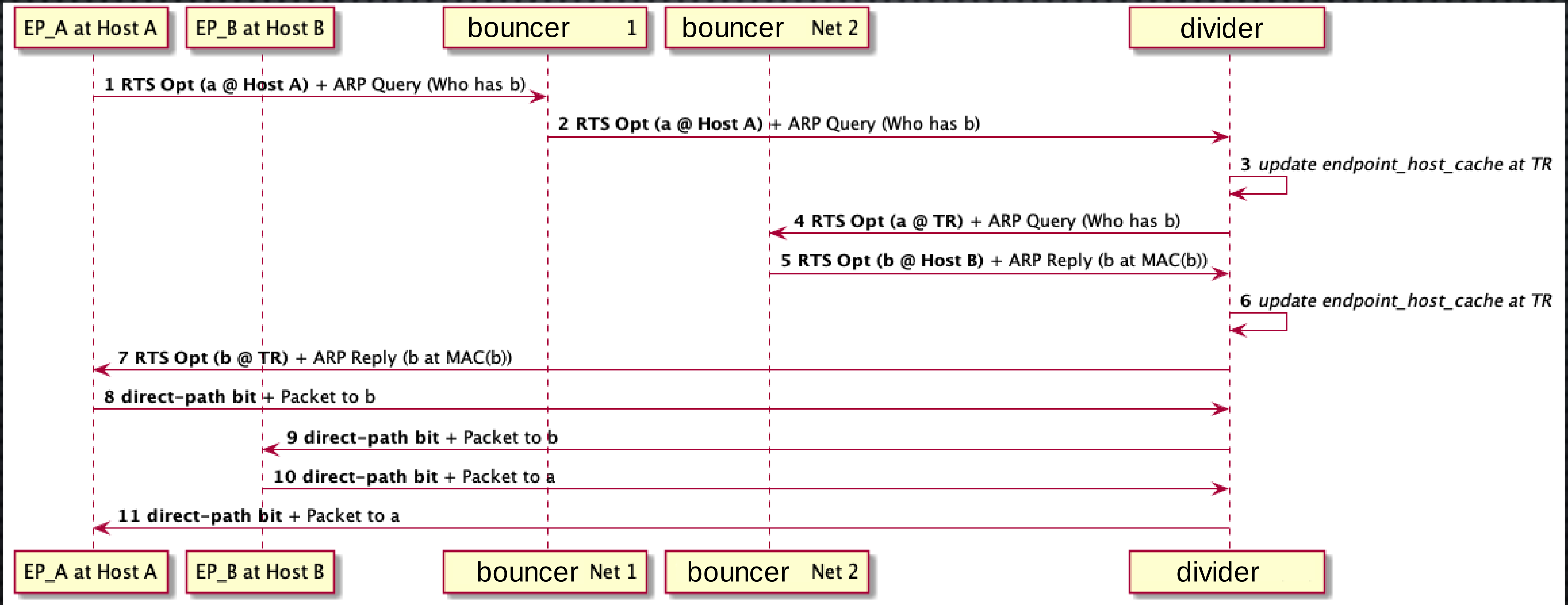
With direct path, all packets sent proceeding the first initial packet is sent directly between hosts



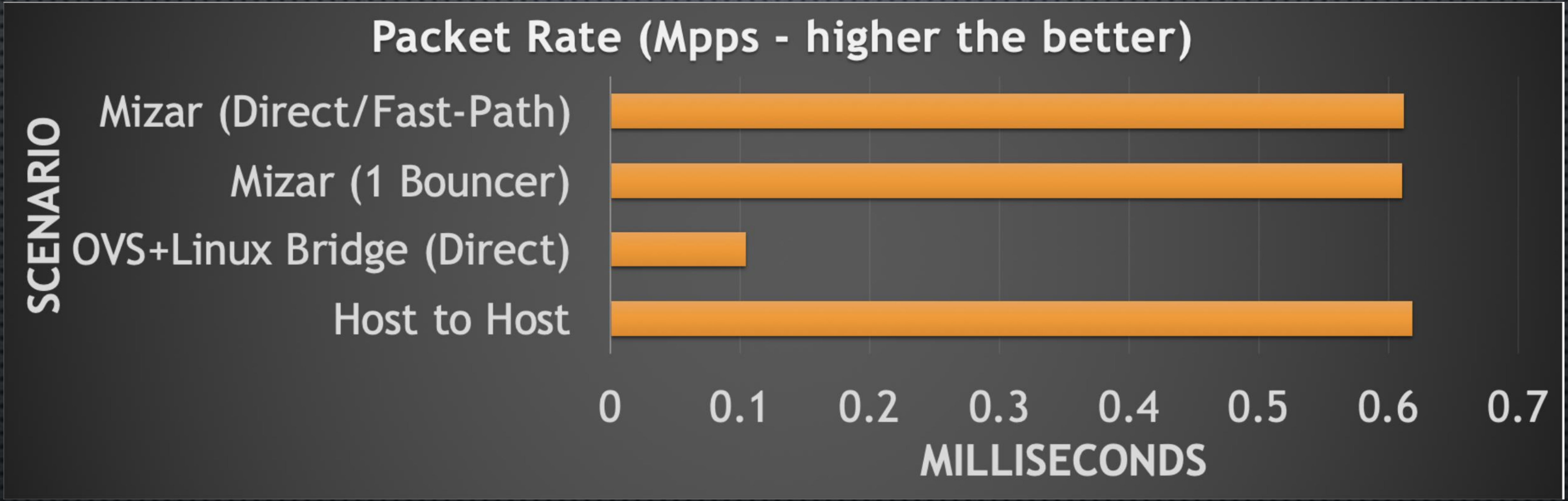
DIRECT PATH: INTER-NETWORK

As the divider will be evaluating network access control (ACL), it is important that all inter-network and inter-VPC packets be processed by the dividers to take effect.

All packets proceeding the initial packet will bypass the bouncers and be sent between the two hosts and the divider.



PACKET RATE (NON TCP) – SCALING NETWORK SERVICES



HIT

- Near line rate packet per second

ENDPOINT UPDATE TIME WITH MULTIPLE BOUNCERS

HIT

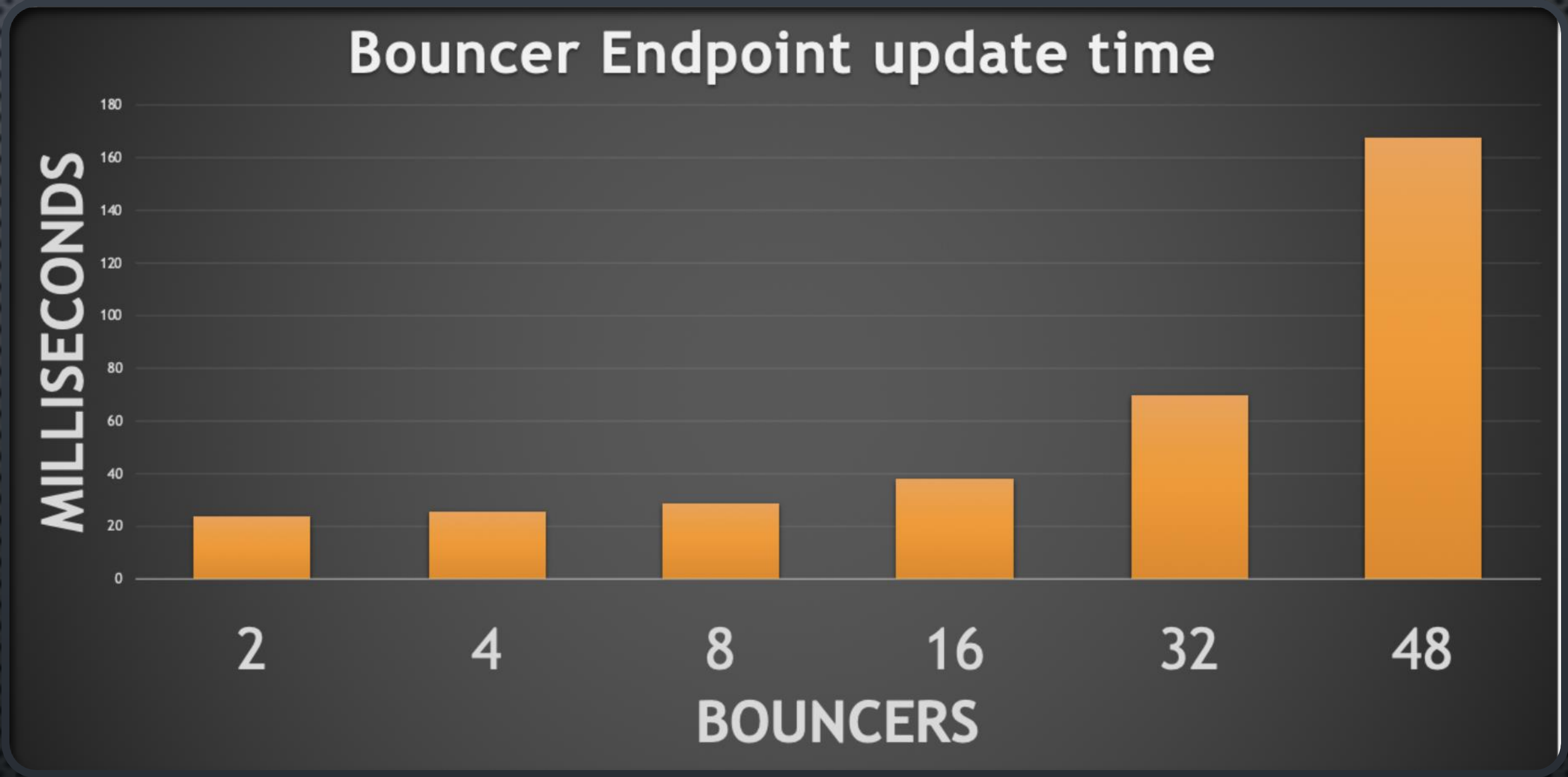
- Constant time with parallel updates (20ms) until the Test Controller starts to Hit its re

Scale

- With a scalable Control-plane (on multiple machines), we foresee maintenance of constant time scaling.

IMPROVEMENT

- Simplifications in data-plane as we introduce the scaled endpoint. One core required.



ENDPOINT E2E PROVISIONING TIME MULTIPLE BOUNCERS

HIT

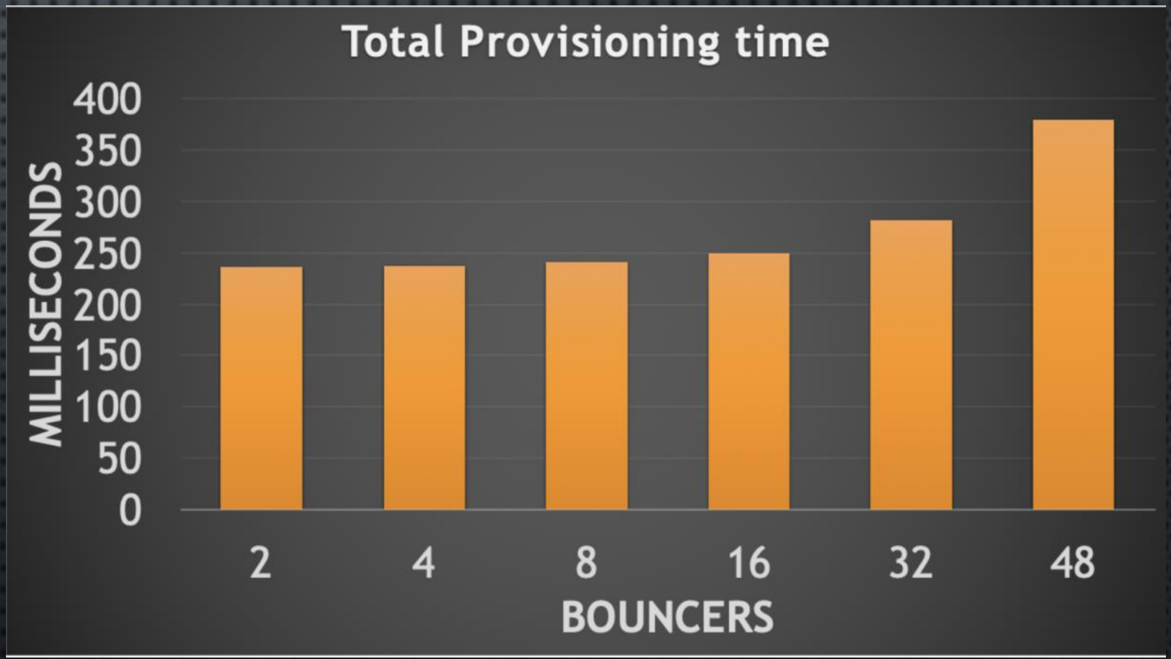
- Scale remains constant (until hitting test controller machine limits)

Overhead

- Primarily overhead on the host from creating the virtual interfaces by executing shell command (~250 ms).

IMPROVEMENT

- Expected to improve with production ready control-plane as it makes use of netlink.



ROUND TRIP TIME EFFECT ON END-USER

HIT

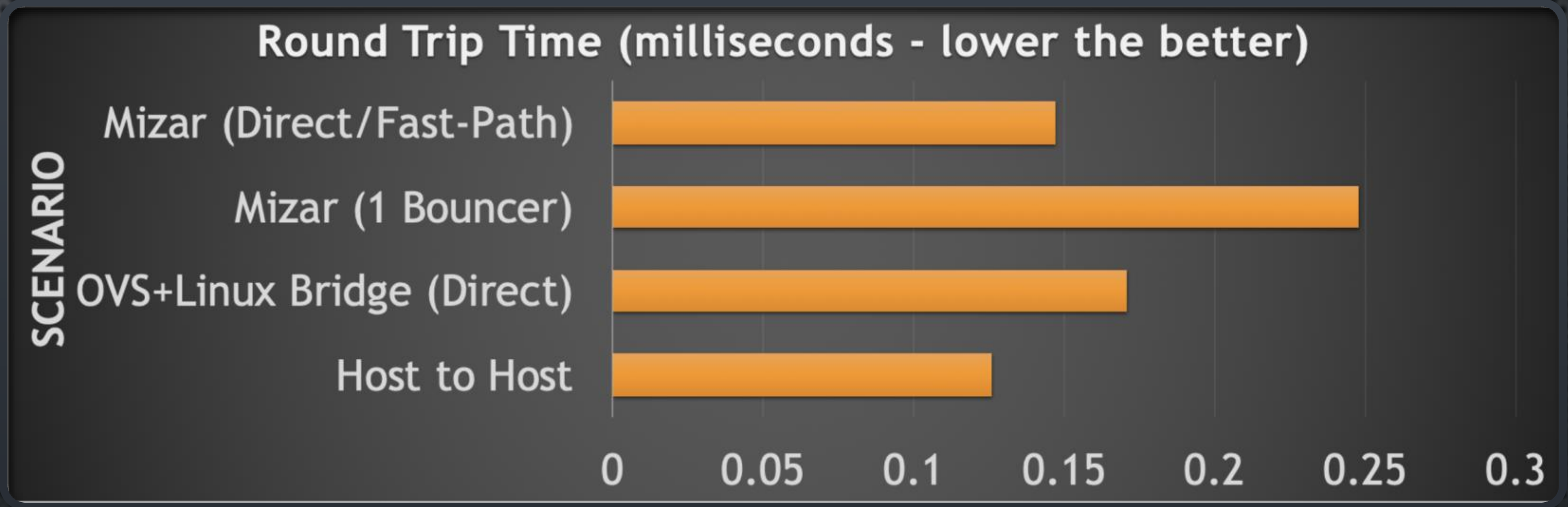
Mizar direct path is faster than OVS+Linux Bridge. Though, Still has minimal impact on PPS and TCP BW.

HIT

Even with an increased latency due to the extra hop, the packet per second processed by endpoints remains close to line rate.

Benefit

Primarily benefit of fast-path is latency sensitive applications.



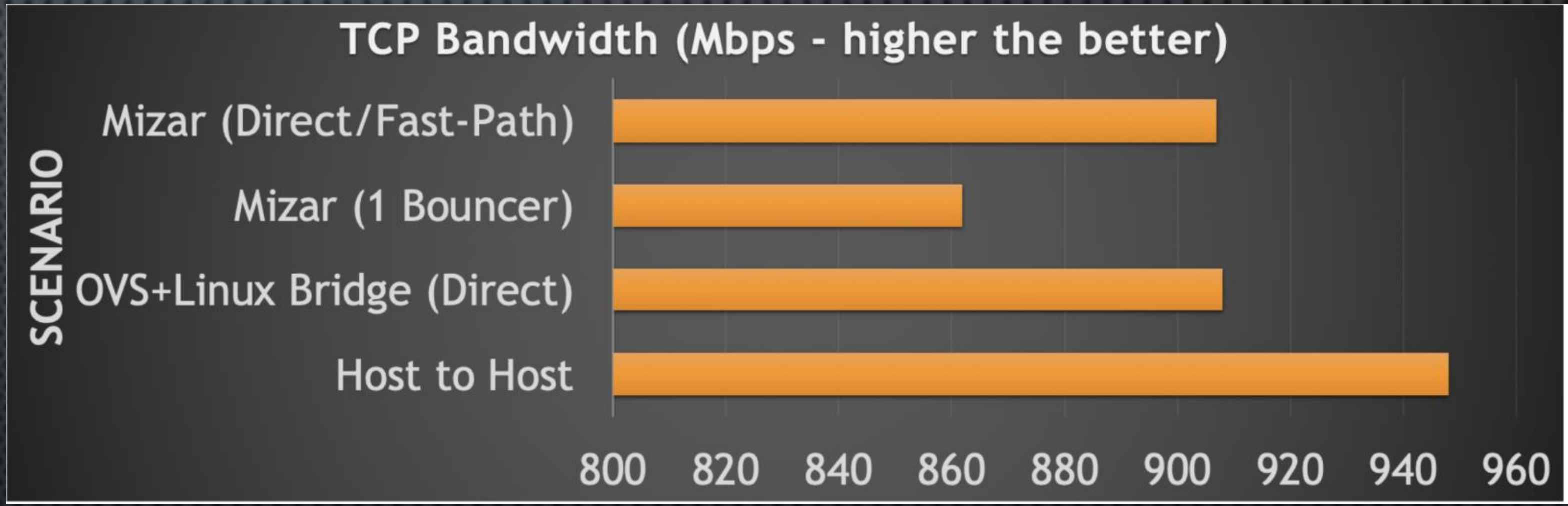
TCP BANDWIDTH (ON A SLOW NIC 1GBPS)

HIT

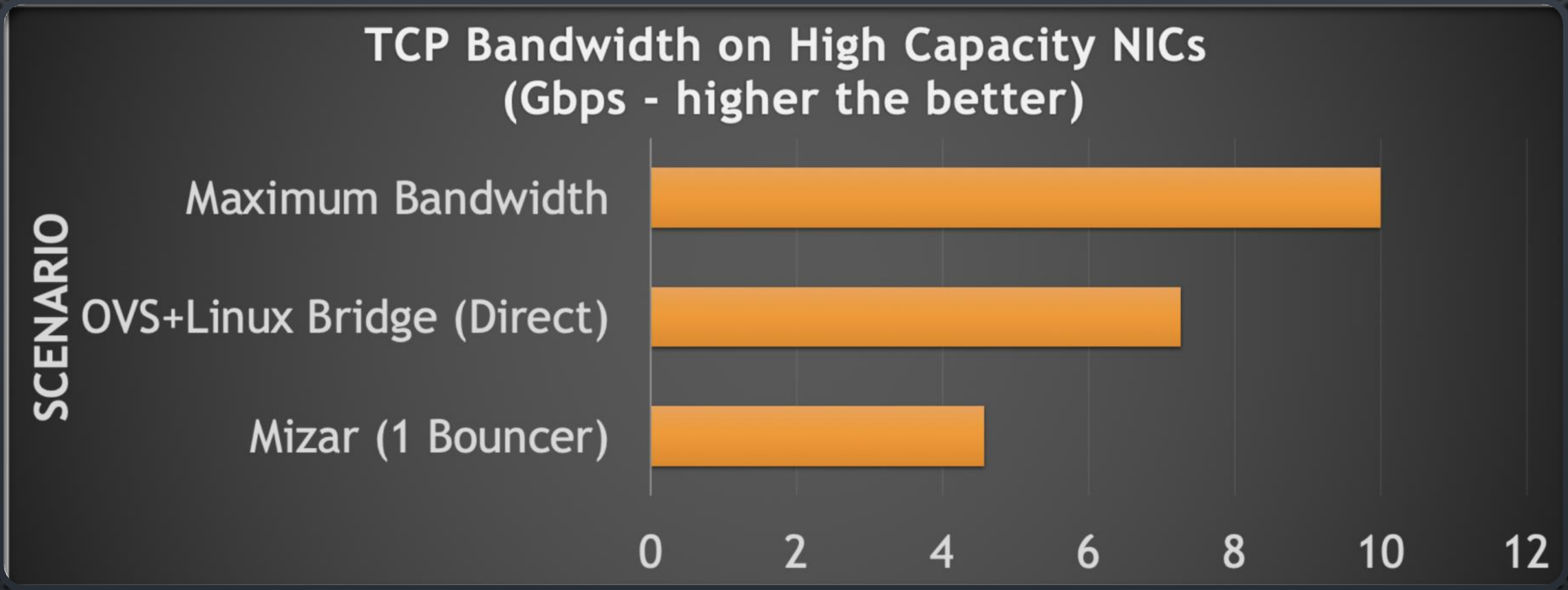
Comparable throughput to OVS+Bridge (even though we don't use XDP driver mode). *This is applicable for NICS < 4Gbps*

Hops:

The Bouncer hop accounts only for 5% less TCP throughput, which shall be negligible for very high bandwidth NICs. This is despite that RTT of the extra hop accounts for 45% more latency.



TCP BANDWIDTH (ON A FASTER NIC 10 GBPS)



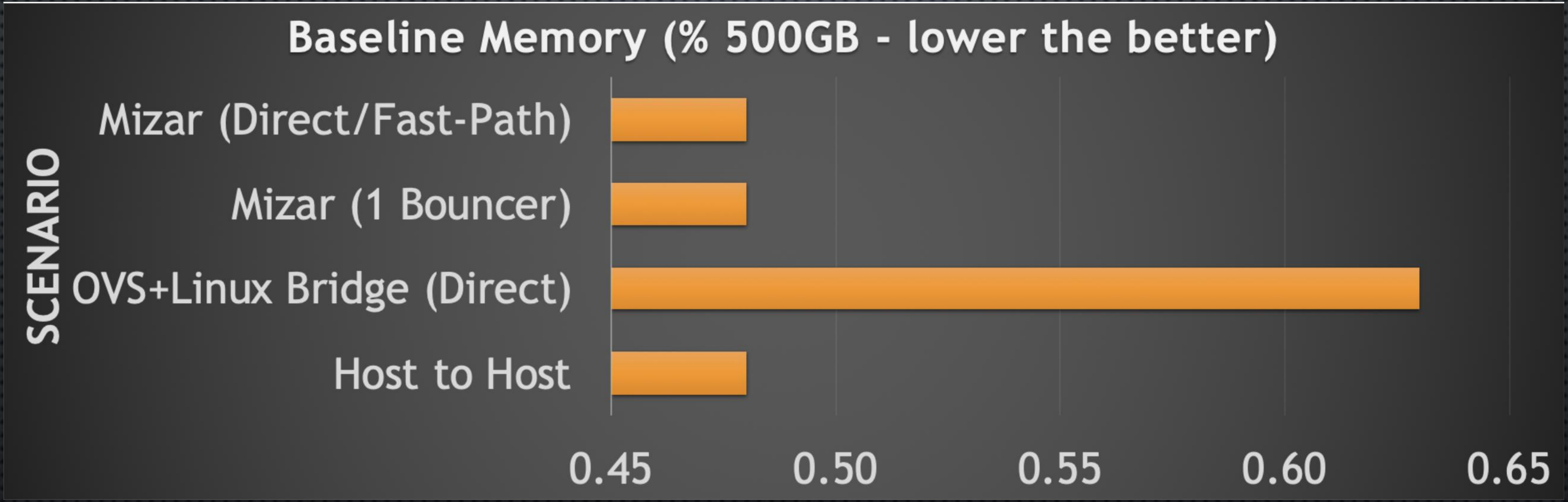
MISS:
The TCP bandwidth caps at around 4Gbps.

IMPROVEMENT:
Change to Driver mode (require support in NIC)

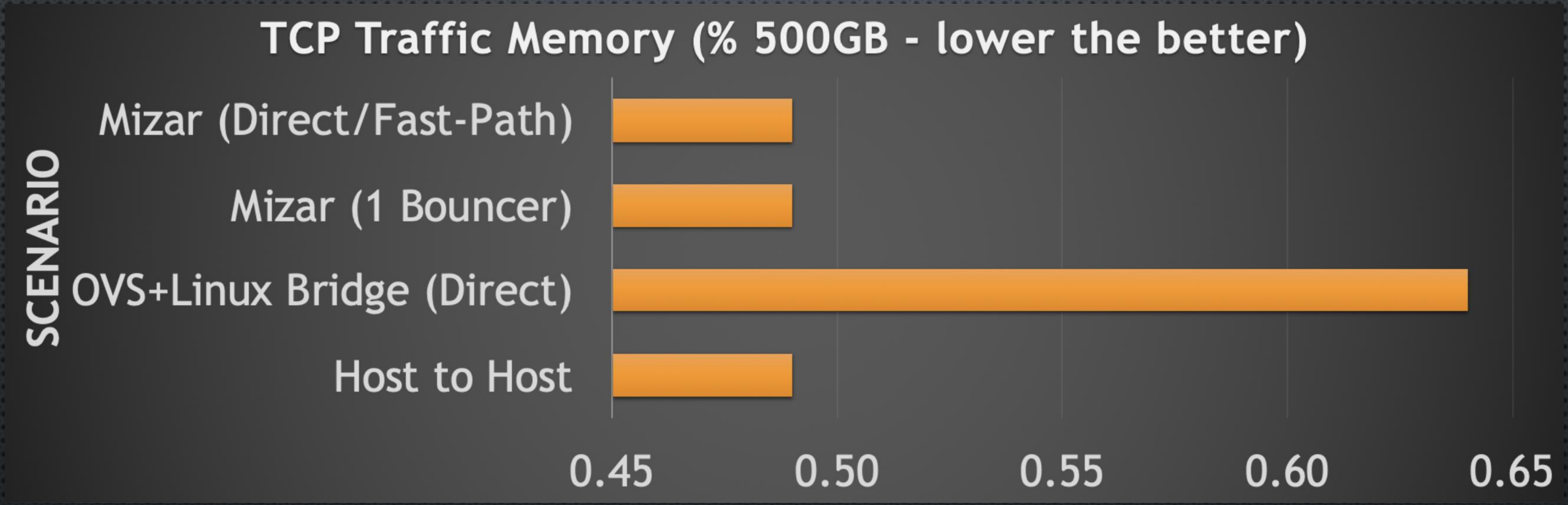
IMPROVEMENT:
Change on-host wiring architecture and reduce reliance on Transit Agent

IMPROVEMENT:
Improved device driver for veth

MEMORY IDLE CASE



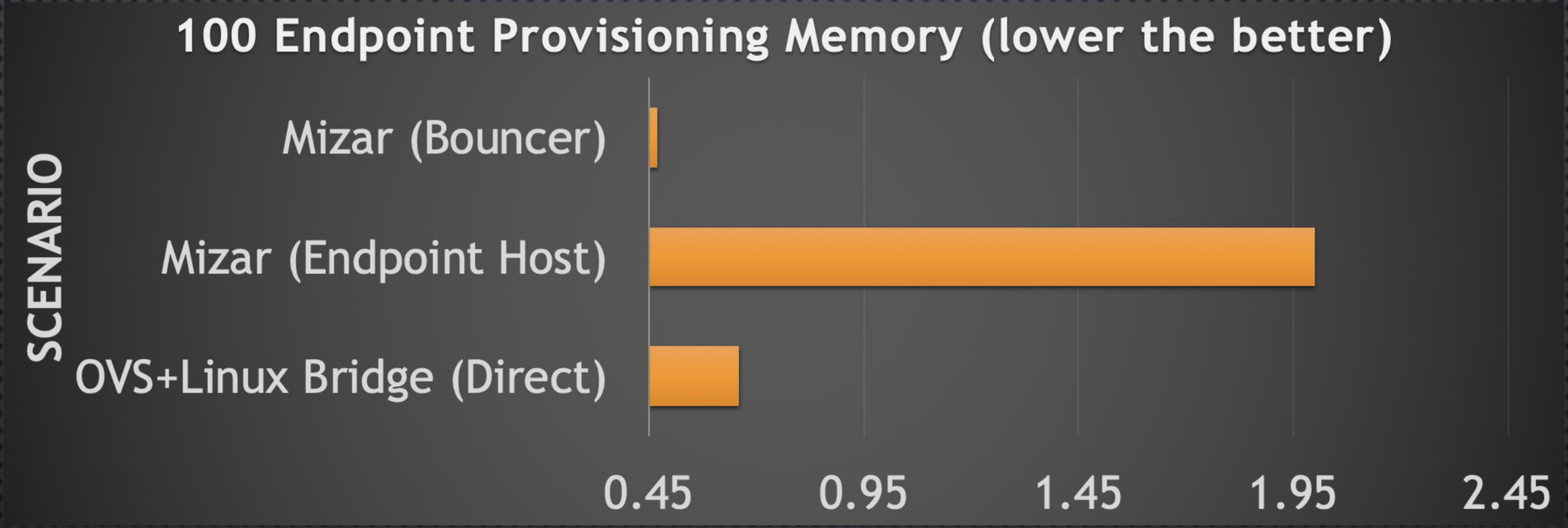
Memory During TCP Performance Tests



- Negligible Memory overhead very close to an idle host without networking constructs event with Traffic processing

HIT

MEMORY IDLE CASE (100 ENDPOINTS PER HOST)



HIT

Memory overhead on Bouncers remain at baseline level

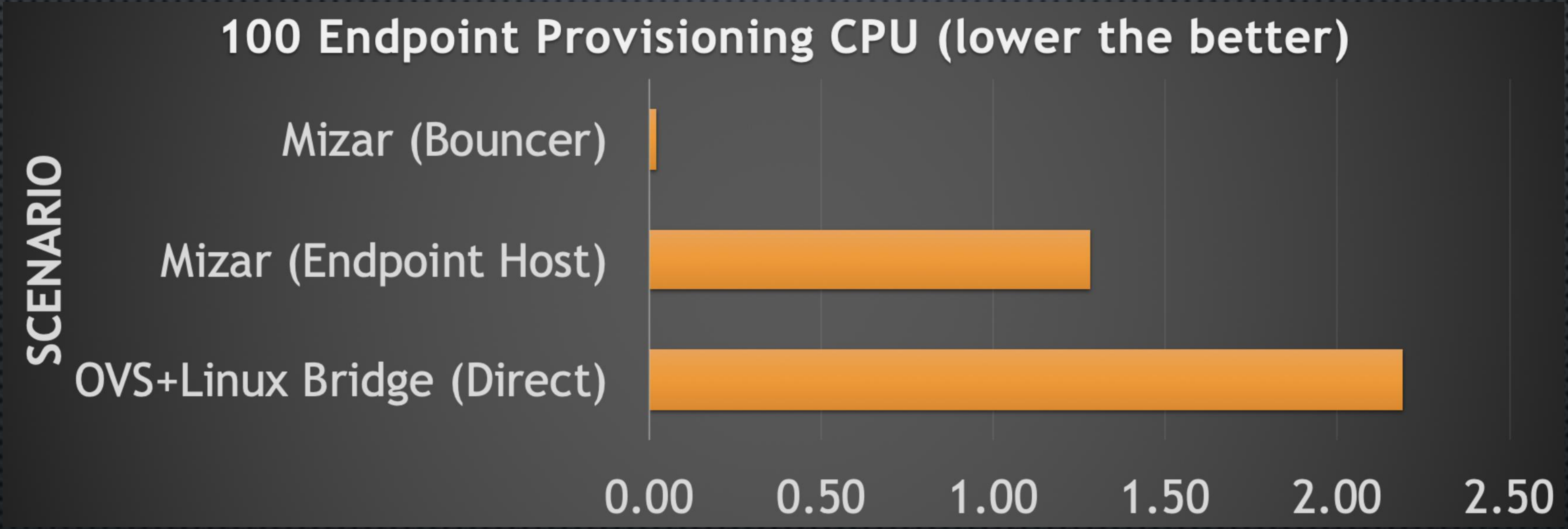
MISS

On Host memory increases as we provision more endpoints

IMPROVEMENT

Share one transit agent across multiple endpoints

CPU DURING TCP PERFORMANCE TESTS



HIT

- Significantly less CPU overhead during provisioning on both bouncer and host