

Mizar

Current state and work in progress

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

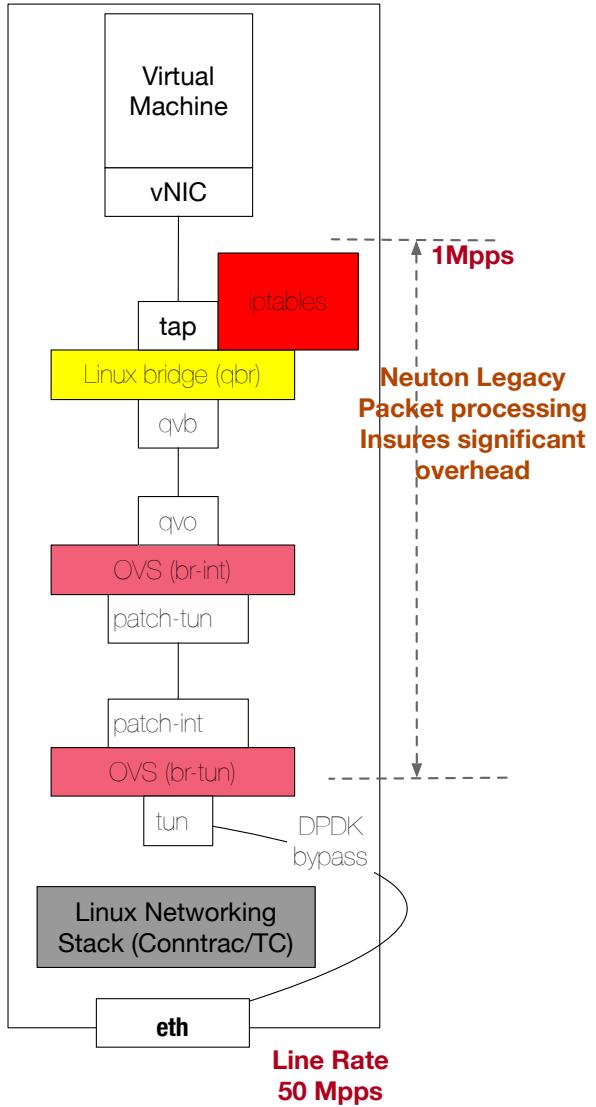
The Problem We are Trying to Solve

- Support provisioning and management of large number endpoints (300K hosts, 10M endpoints)
- Accelerate network resource provisioning for dynamic cloud environments
- Achieve high network throughput and low latency
- Create an extensible cloud-network of pluggable network functions
- Unify the network data-plane for containers, serverless functions, virtual machines, etc

Problems with Current solutions

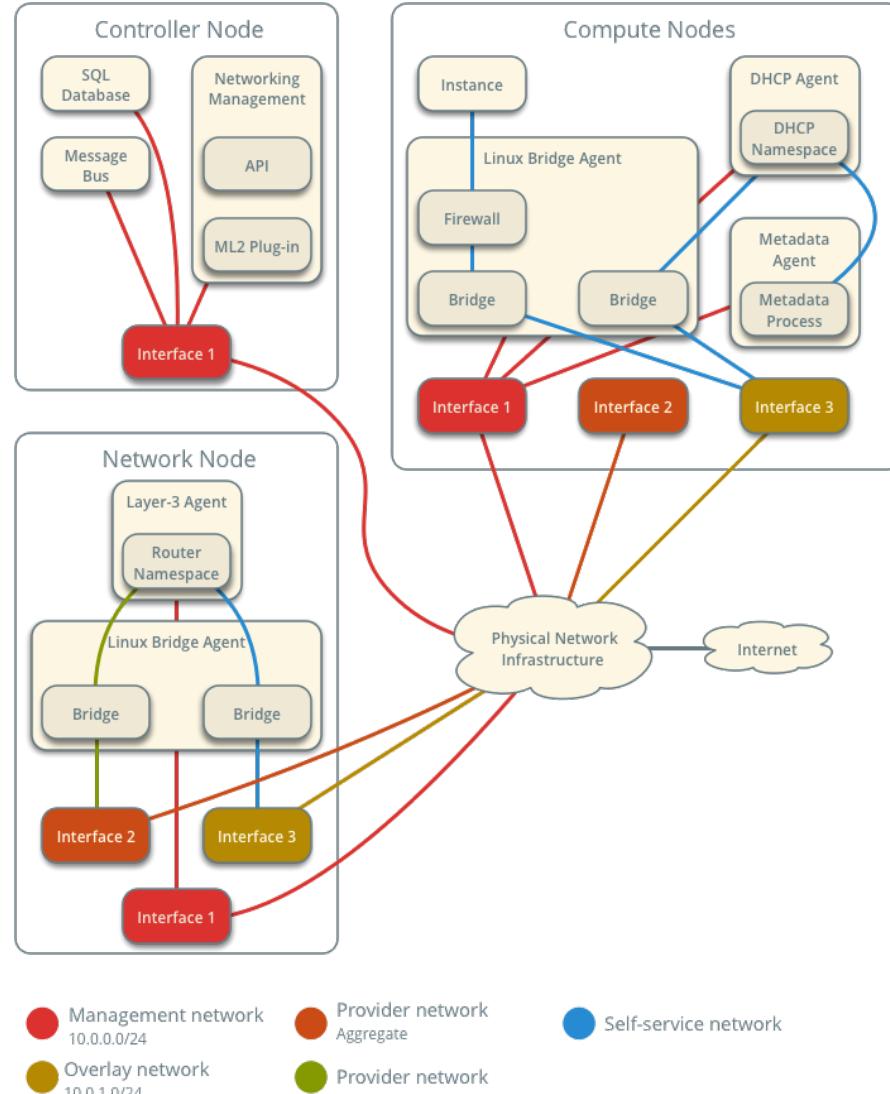
- Program every host every time a user provision an endpoint:
- Approaching cloud-networking with a conventional programming model and network devices
 - e.g. OpenFlow programming in OVS
 - Virtual Switches and Routes are essentially softwareization of hardware switches and routers, but not necessarily programmable to support rapid network changes.
- Existing solution bring up software network devices, that are primarily created for Teleco, ISP, or datacenter networking and run them in virtual machines to support cloud networking.
- Packets traverses multiple network stacks on the same host
- Packets traverses multiple network devices (as if we are operating a data-center), while these functions could be consolidated during design.

Problems with Current solutions (e.g. Neutron)



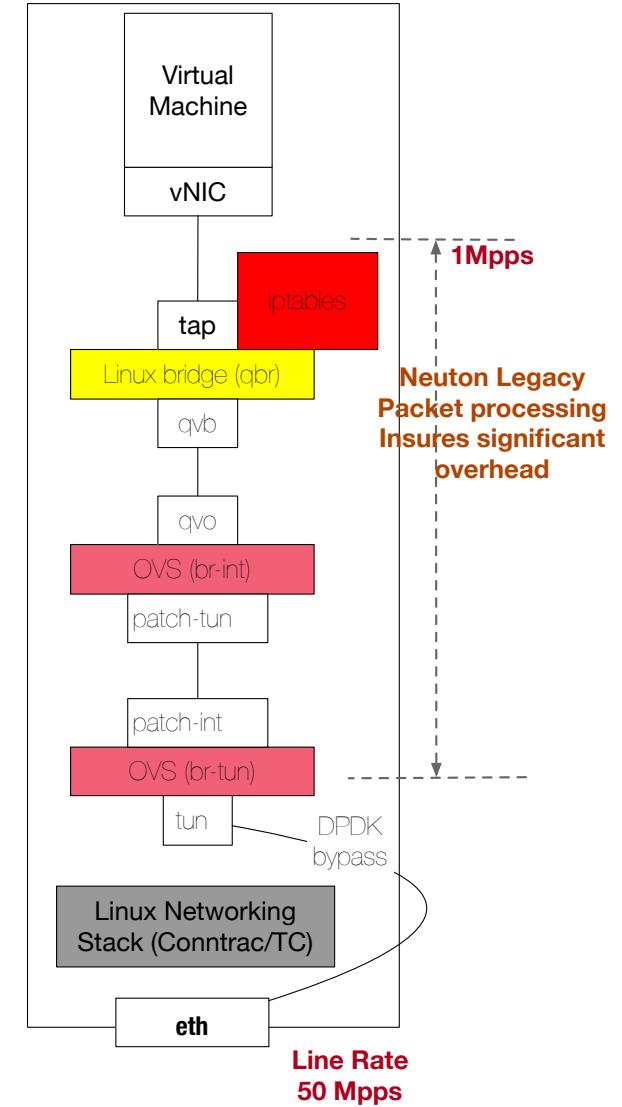
Linux Bridge - Self-service Networks

Overview



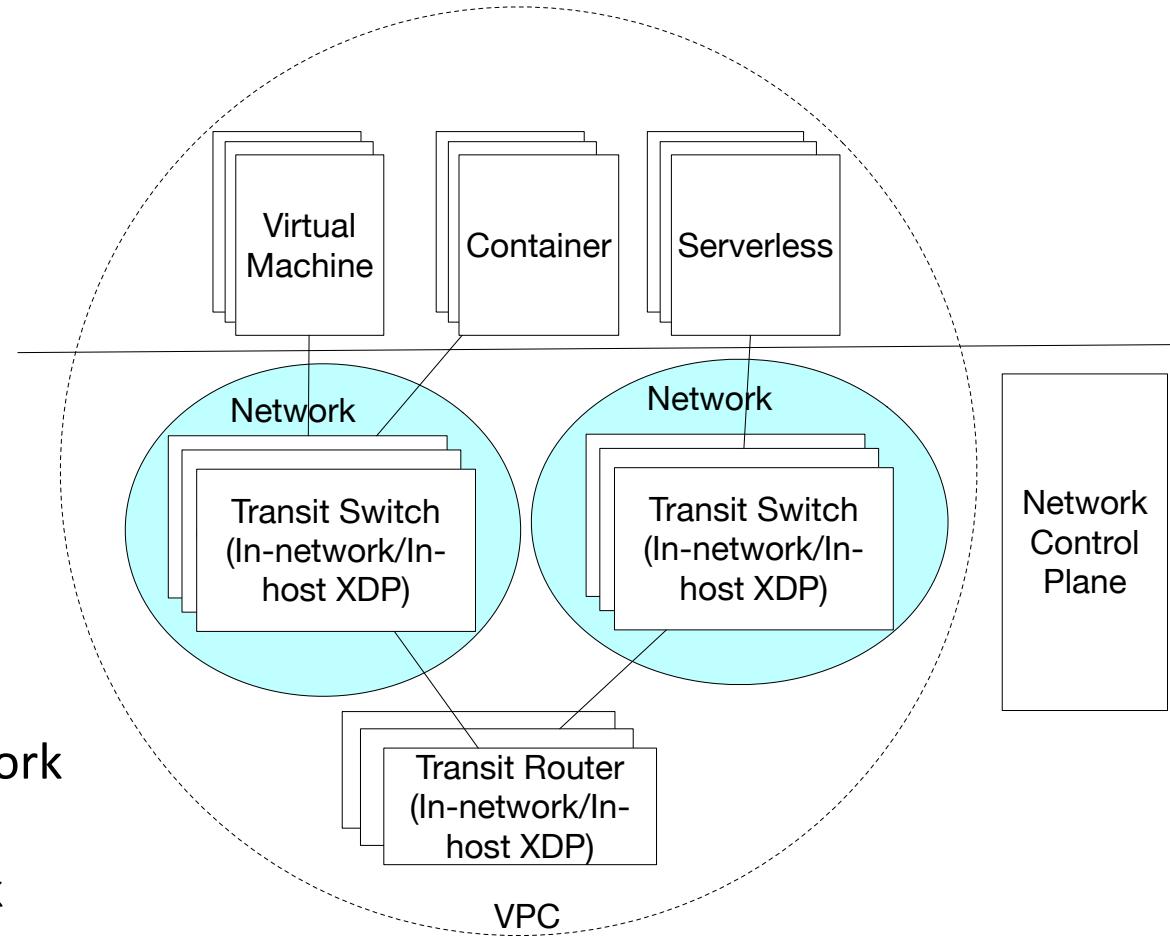
Observation (not really a new one):

- 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



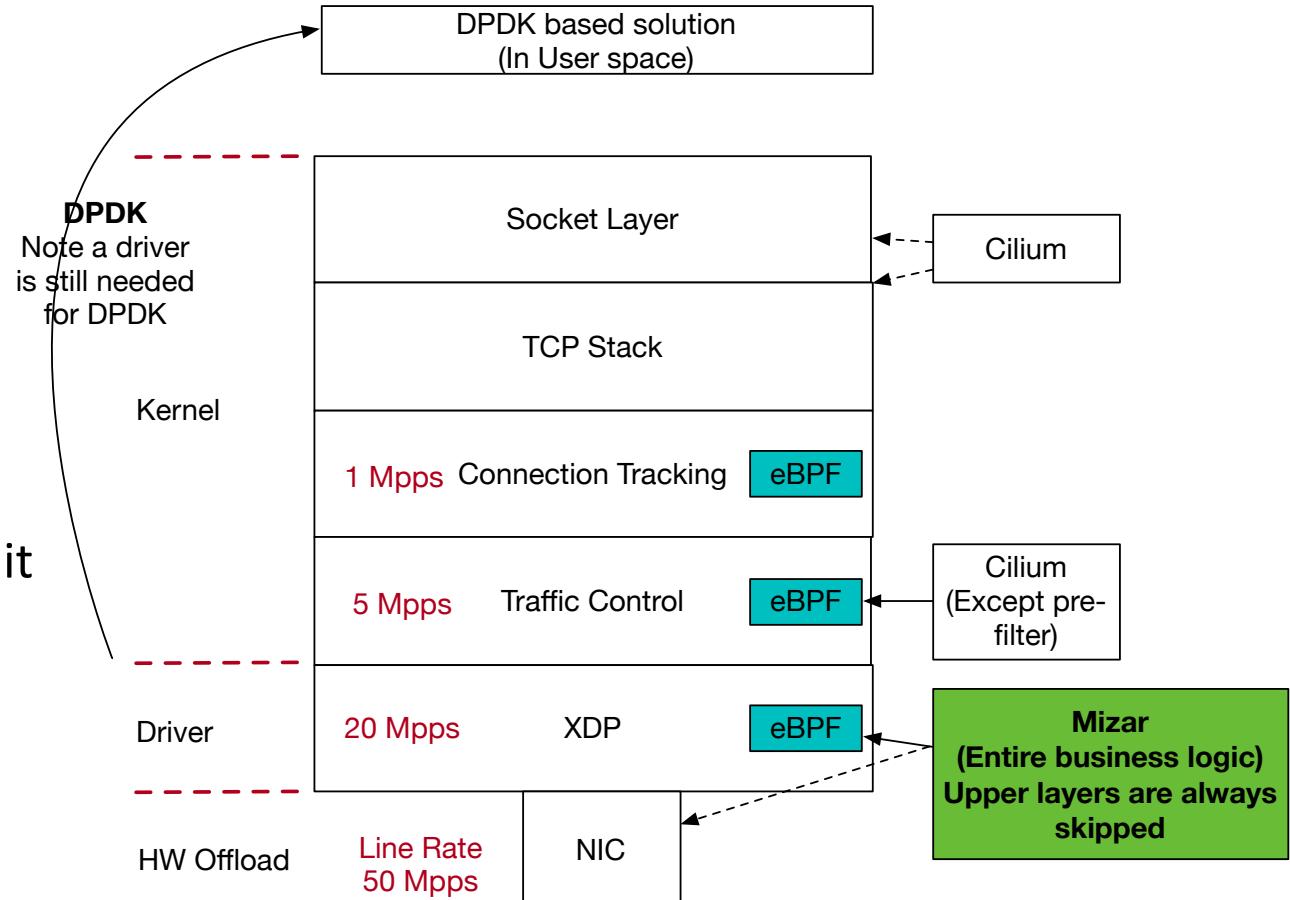
Mizar Overall Architecture!

- Natural Partitioning domains of Cloud Network
 - Virtual Private Cloud VPC
 - Networks within a VPC
 - Endpoints within a network
- Goal: Constant time provisioning of endpoints
- Transit Switches
 - In-network hash tables
 - Holds the configuration of endpoints within a network
 - Determines an endpoint host
 - Implements all Network functions within a network
- Transit Router
 - In-network hash tables
 - Holds the configuration of networks within a VPC
 - Determines a transit switch of an endpoint
 - Implements all functions within a VPC



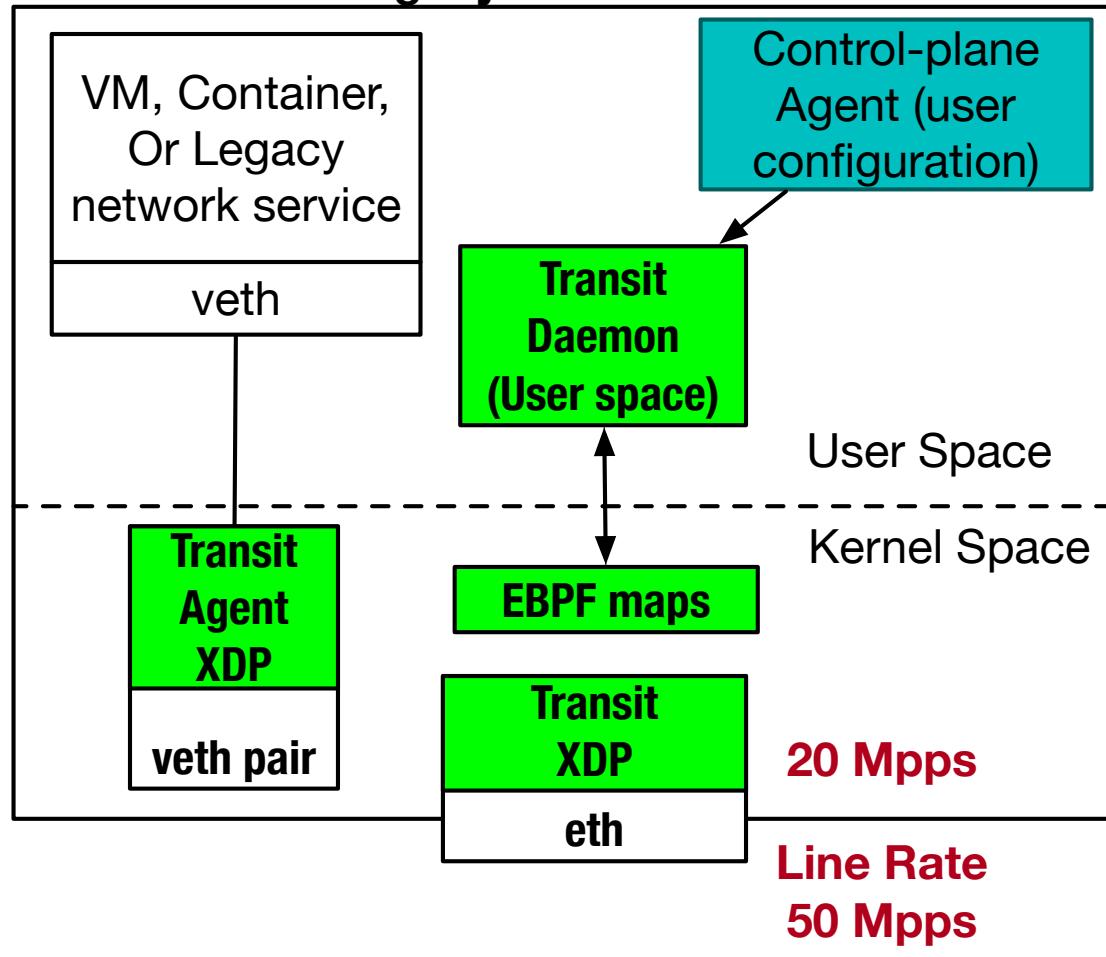
Background XDP: Simplified and Extensible Packet Processing Near Line Rate

- Packet processing is entirely in-kernel.
- Makes the best use of kernel packet processing constructs without being locked-in to a specific processor architecture.
- Skip unnecessary stages of network stack whenever possible and transit packet processing it to smart NICs.
- Very small programs < 4KB

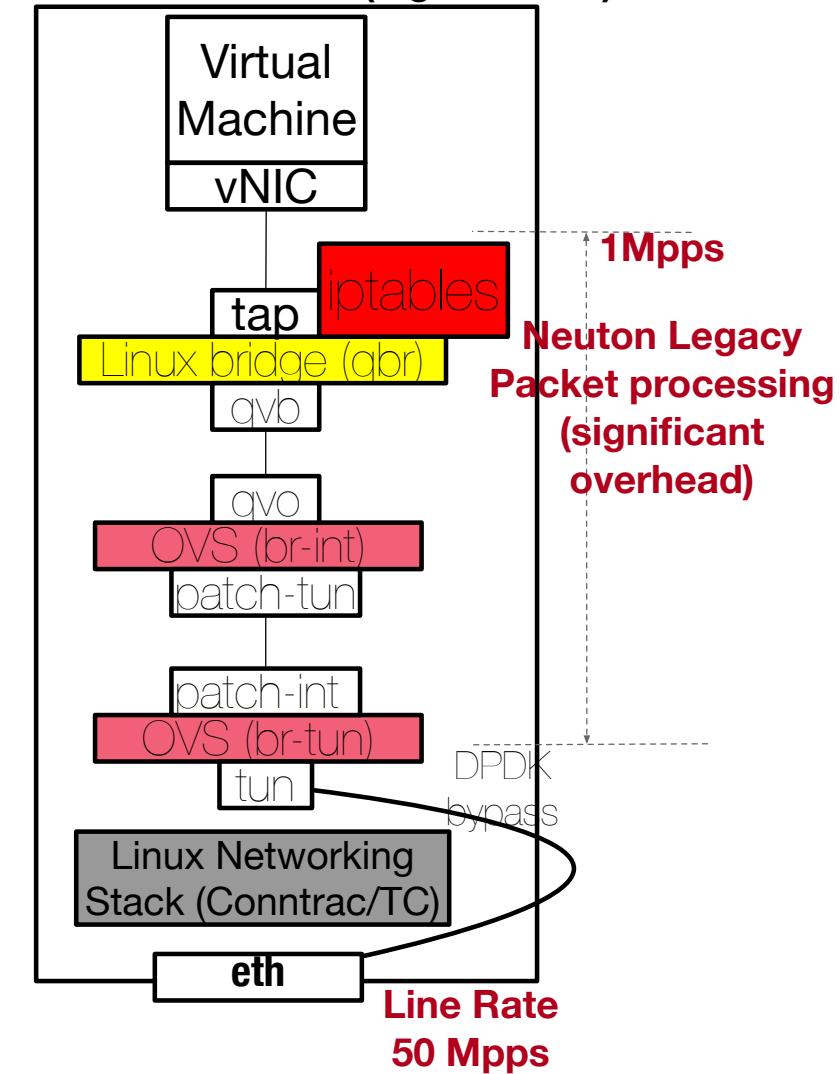


Inside a Mizar host

Mizar Simplified Node design for VMs, Containers, and legacy network services

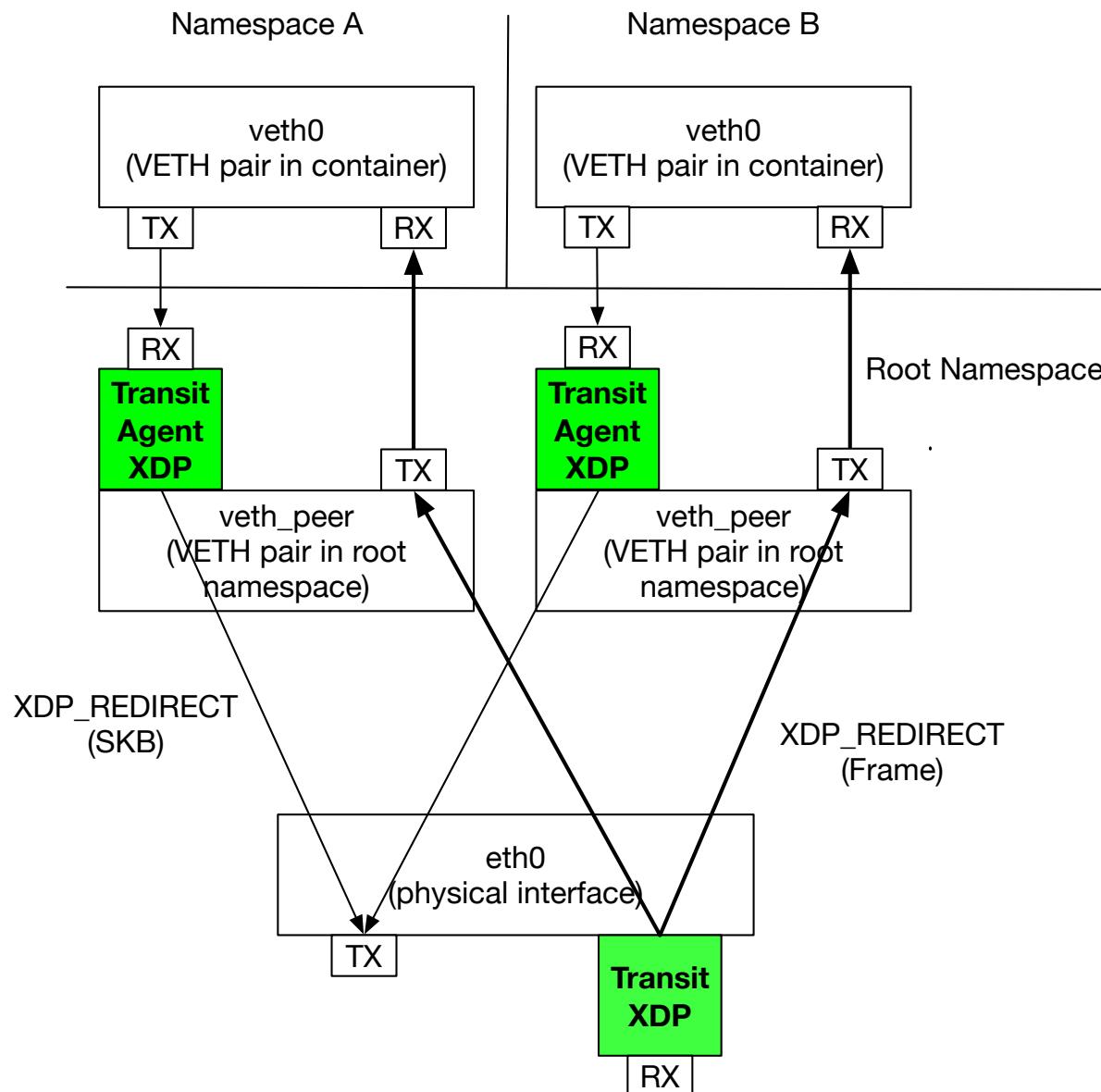


OVS Based Solutions (e.g. Neutron)



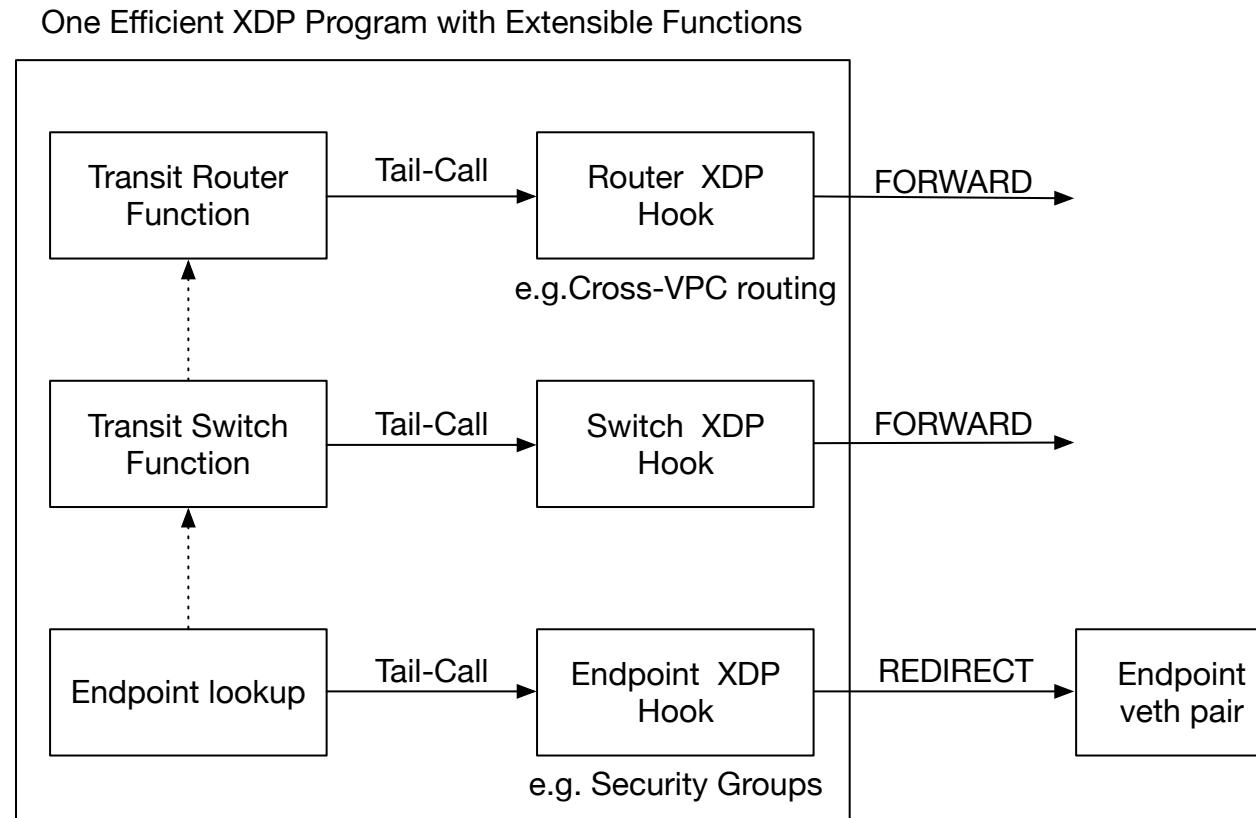
In-host packet flow: Bypass network stack

- Packets traverses only the container stack
- On egress packets are redirected (SKB) to the main interface after tunneling
- On ingress packets are redirected directly to the container veth peer in the root namespace.

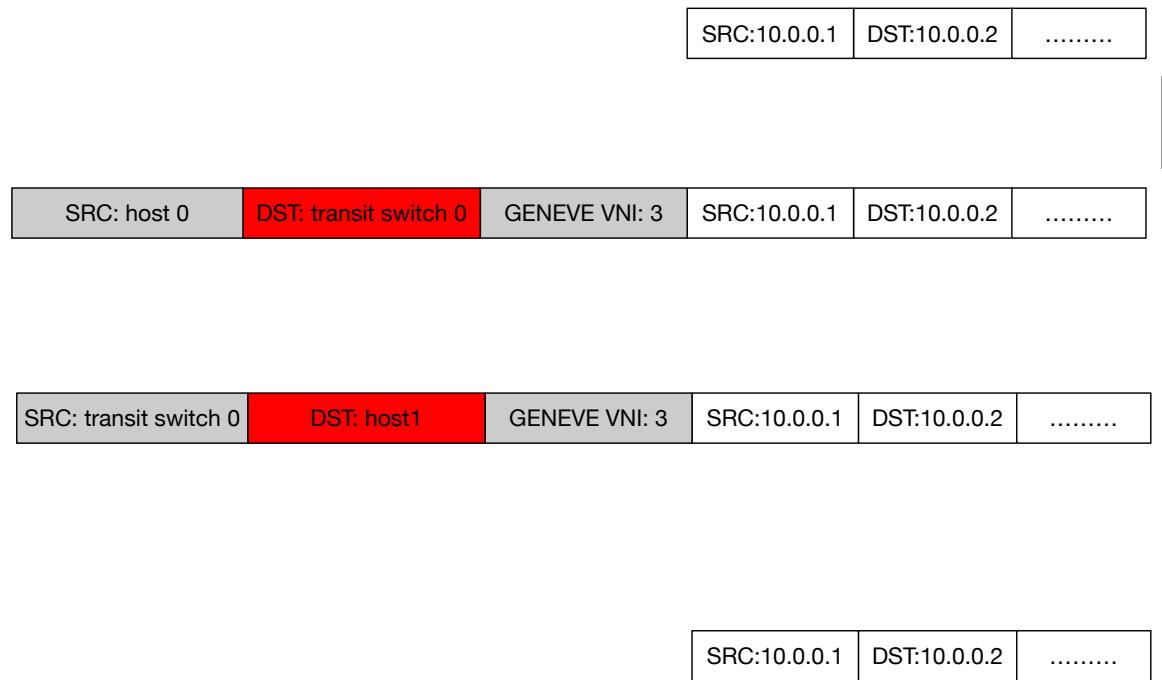


Extensible Packet Processing inside the main XDP program!

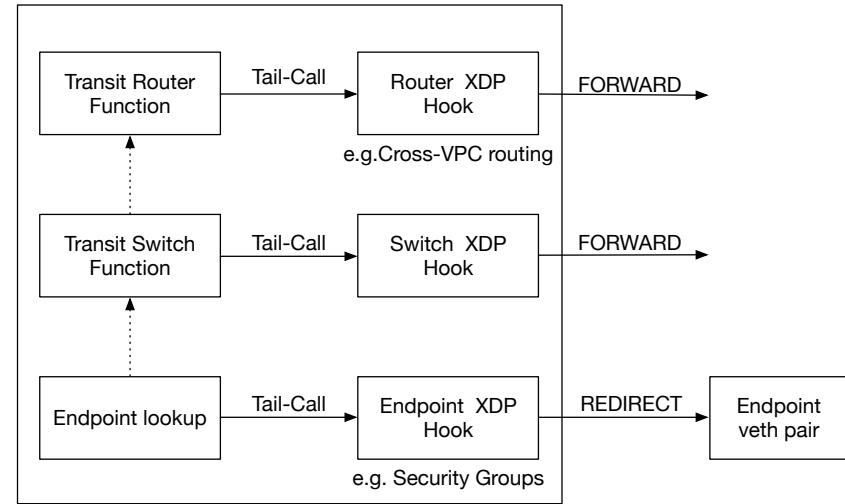
- Implements essential logical networking function within the same XDP program that provides multi-tenant cloud networking solutions through **new** transit switch and transit routers concepts
- Smart Control Plan **will** allow Mizar to Autonomously adapts to various traffic demands in immense scale 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, Load-balancing, Connectivity, Traffic Shaping Control



Example packet within a network



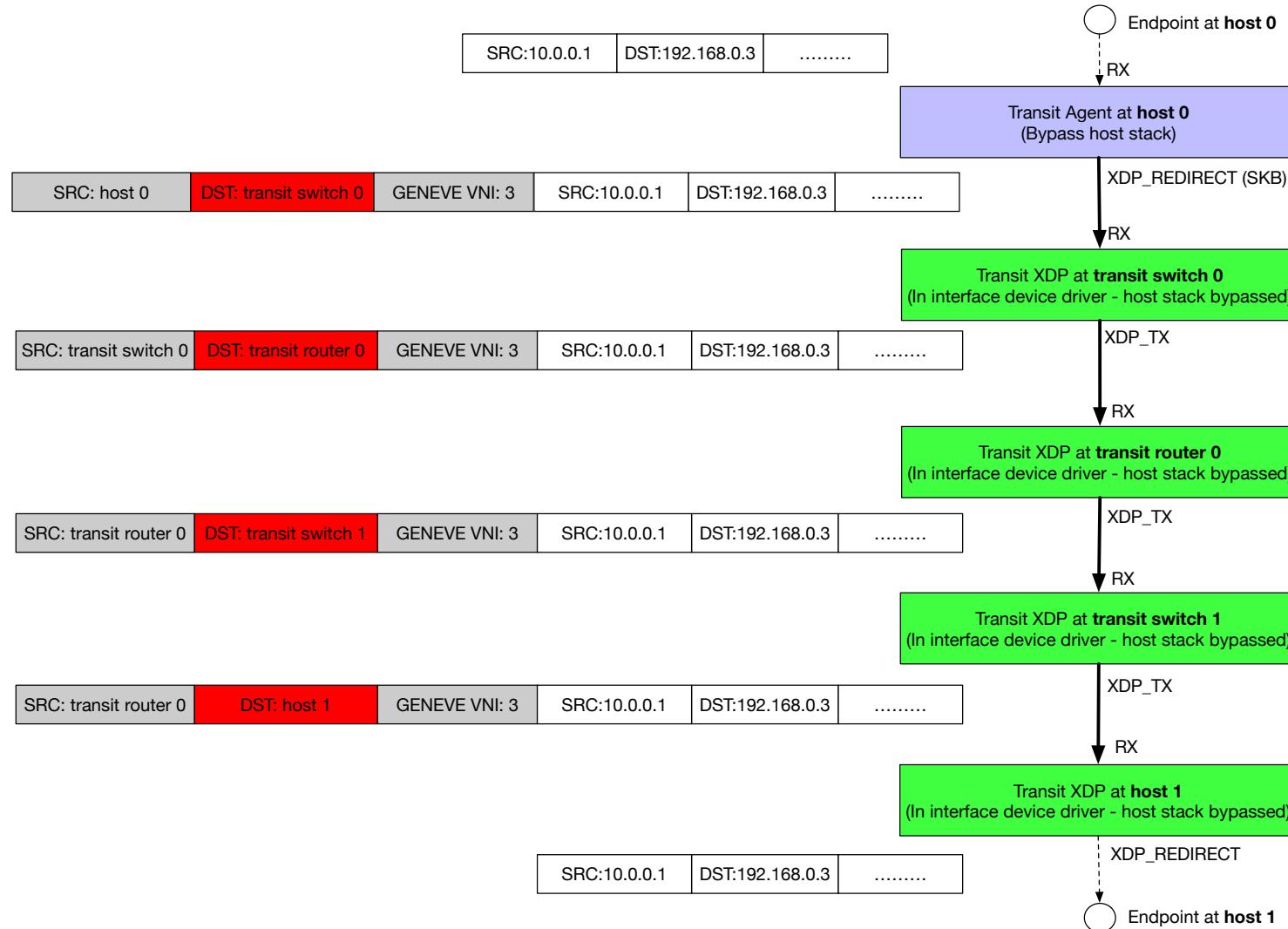
One Efficient XDP Program with Extensible Functions



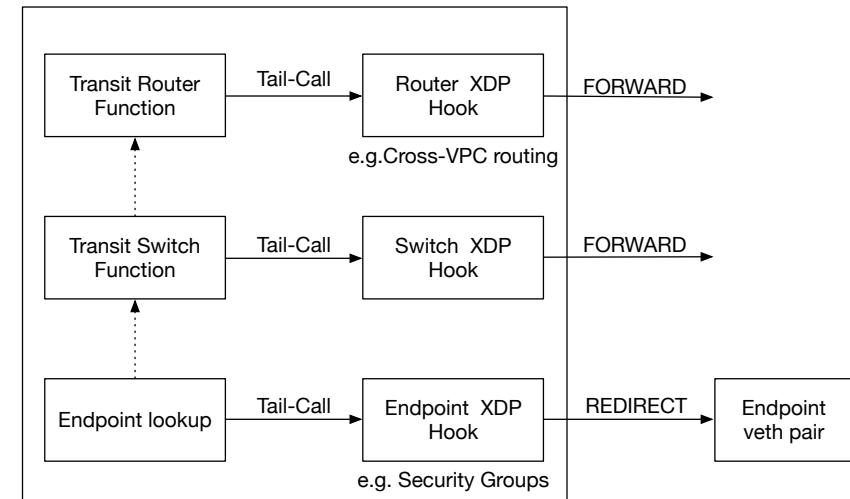
Three steps to provision an endpoint

1. Add the endpoint to N transit switch table
2. Provision the endpoint on the host
3. Configure the host transit agent to tunnel the endpoint traffic to the transit switch

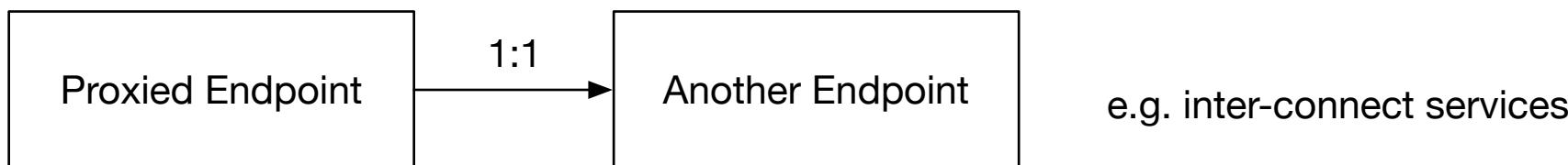
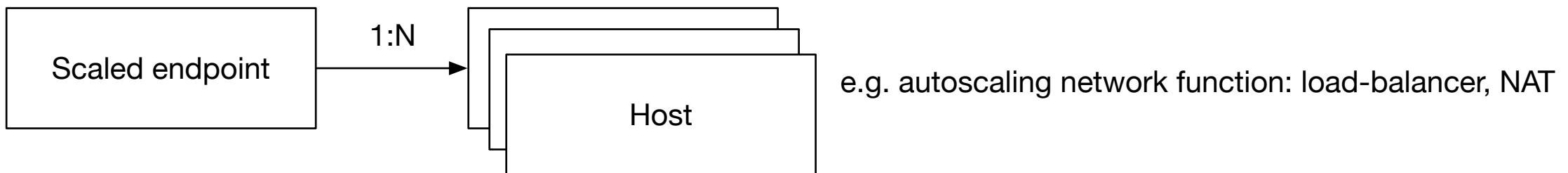
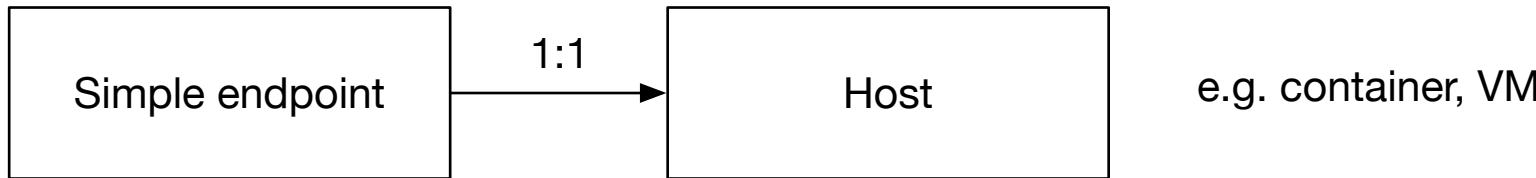
Example packet cross networks



One Efficient XDP Program with Extensible Functions



New endpoint types



Problems we are working on

What's next?

New Problems: Self-optimizing data-plane

- **Smart Placement of transit switches and routers:**
 - Can we solve it in multi-agent formulation?
 - Does it fit a game-theoretic framework?
 - Reinforcement learning?
- **Auto scaling the transit switches and transit routers:**
 - Can transit switches clone themselves?
 - How to disseminate information about the new transit switches?
 - What metrics shall we use?
 - How to ensure stable decisions for scaling and placement?
- **Can scaling and placement ensure SLAs?**
- **Implementation for a self-contained data-plane (no dependency on another layer of management)?**

New Problems: Constant Time Distributed Data-plane

- **Minimize Hops:**
 - Can we minimize the number of hops?
 - Scaling on number of connections not number of packets
 - What if hosts fail?
- **Distributed Flow Tables:**
 - Can we bring down provisioning time to $O(1)$?
 - Persist flow decisions for scaled endpoints?
- **Example services:**
 - Load-balancer
 - NAT
 - Cross-VPC routing

New Problems: Packet Forwarding optimization

- **Improvements to the veth device driver:**
 - Native Geneve support in the same device driver
 - How to get a better packet processing performance in generic XDP mode?
- **Per-hop congestion control:**
 - Most packets are not subject to congestion-control. Are we in a trouble?
 - Applicability of per-hop congestion control.
- **Ongoing: What Linux stack functions shall be reused and what to avoid?**

New Problems: Application Centric Data-plane

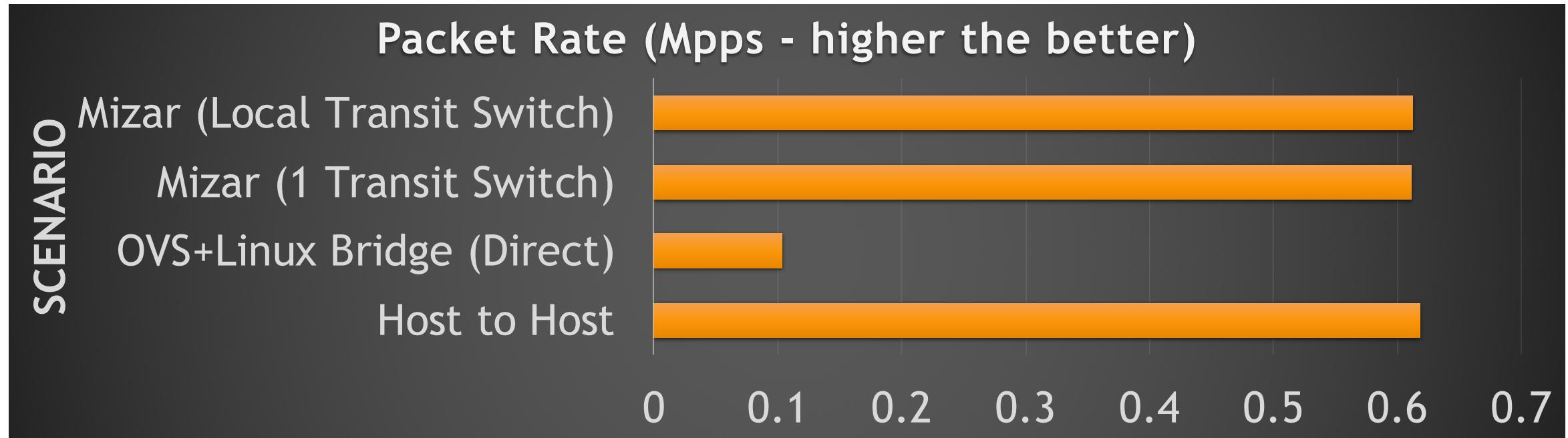
- **Coworking with TCP as a service.**
- **What can we learn about the application and inject as Geneve options? And what to do about?**
- **How to support a network as a group of applications (not a conventional subnet)?**
- **New types of endpoints?**

Results we have so-far...

Notes

- All the following tests are done in SKB mode (XDP Generic), which has a performance disadvantage
- We wanted to test Mizar's XDP program in driver mode, but for now we don't have the needed hardware

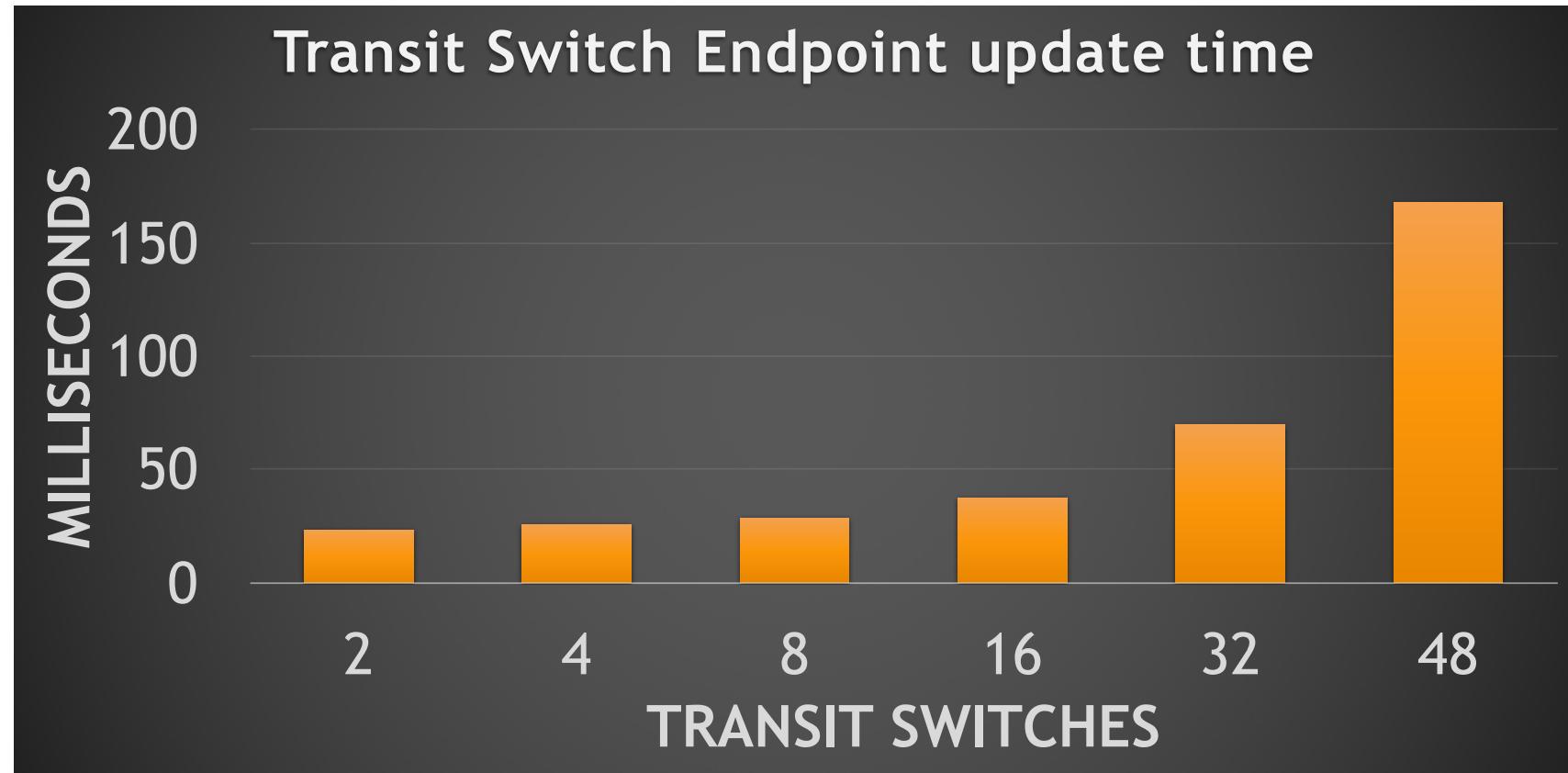
Packet Rate (non TCP) – Scaling Network Services



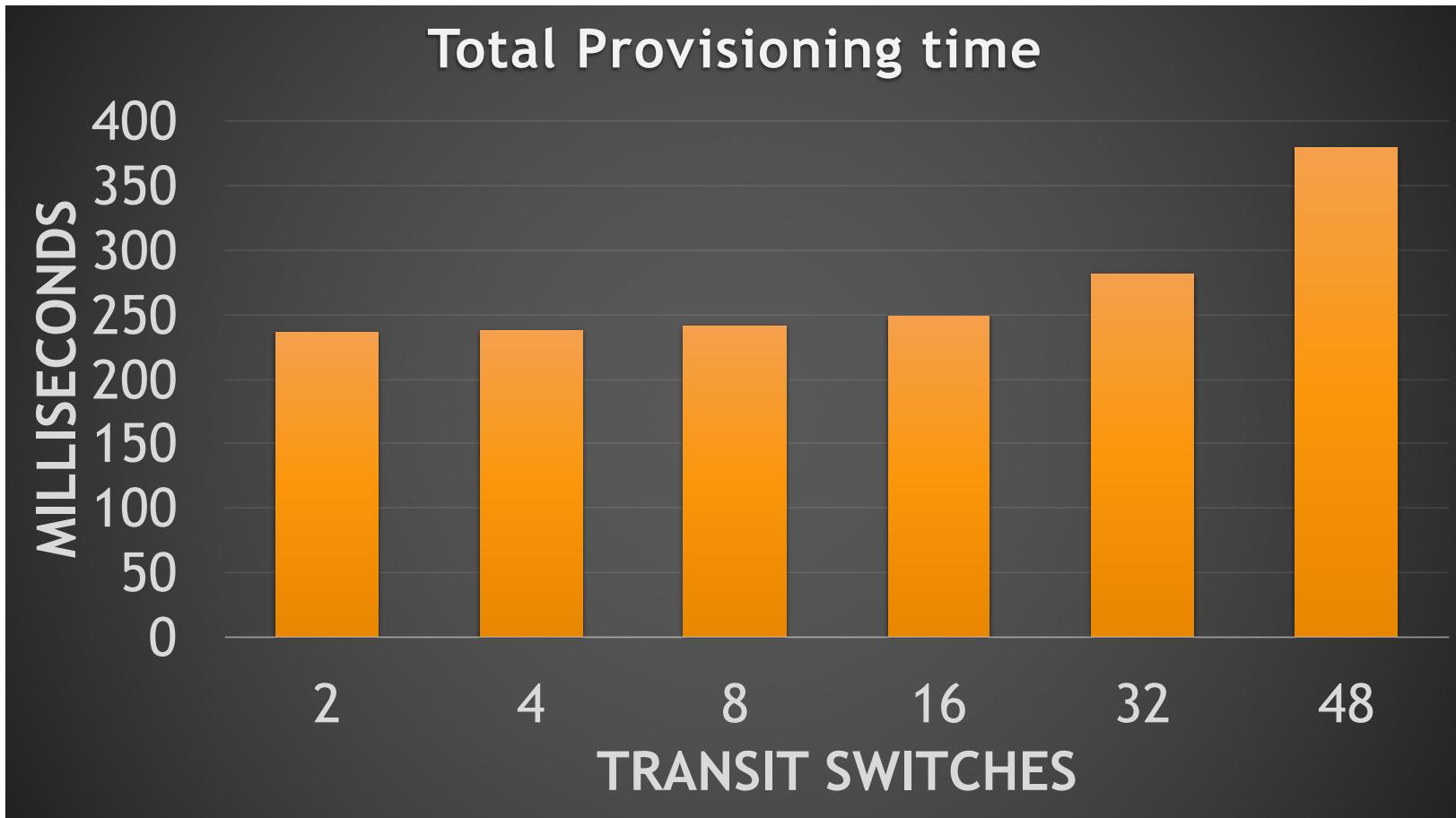
- **HIT:** Near line rate packet per second

Endpoint Update Time with multiple Transit Switches

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

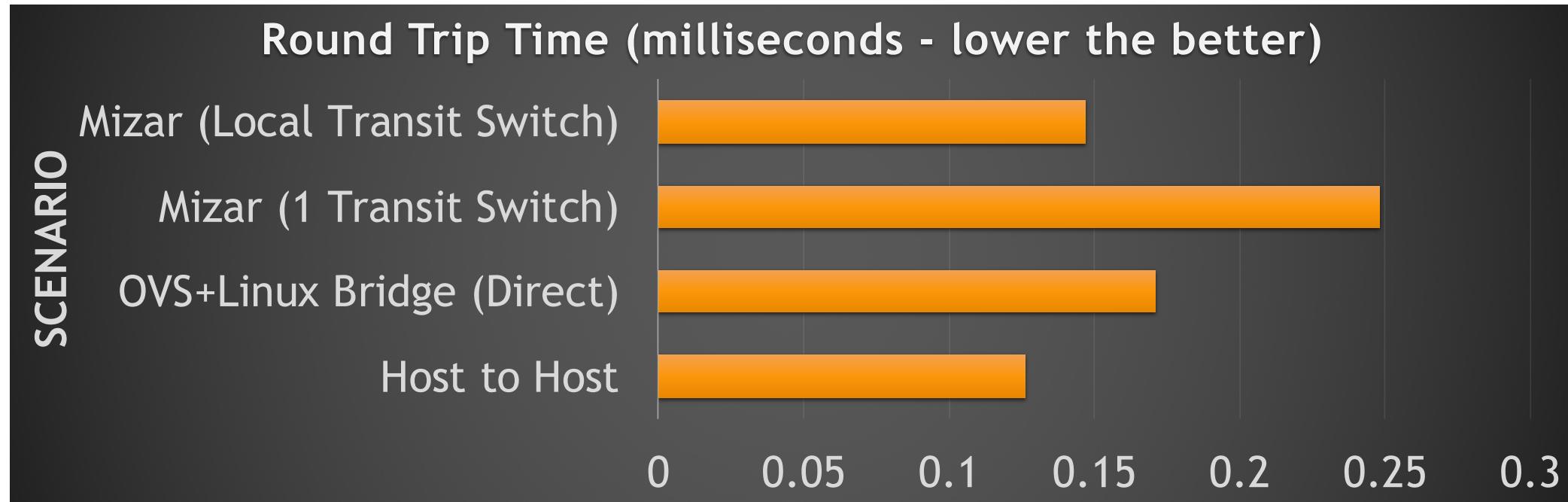


Endpoint E2E provisioning time multiple Transit Switches



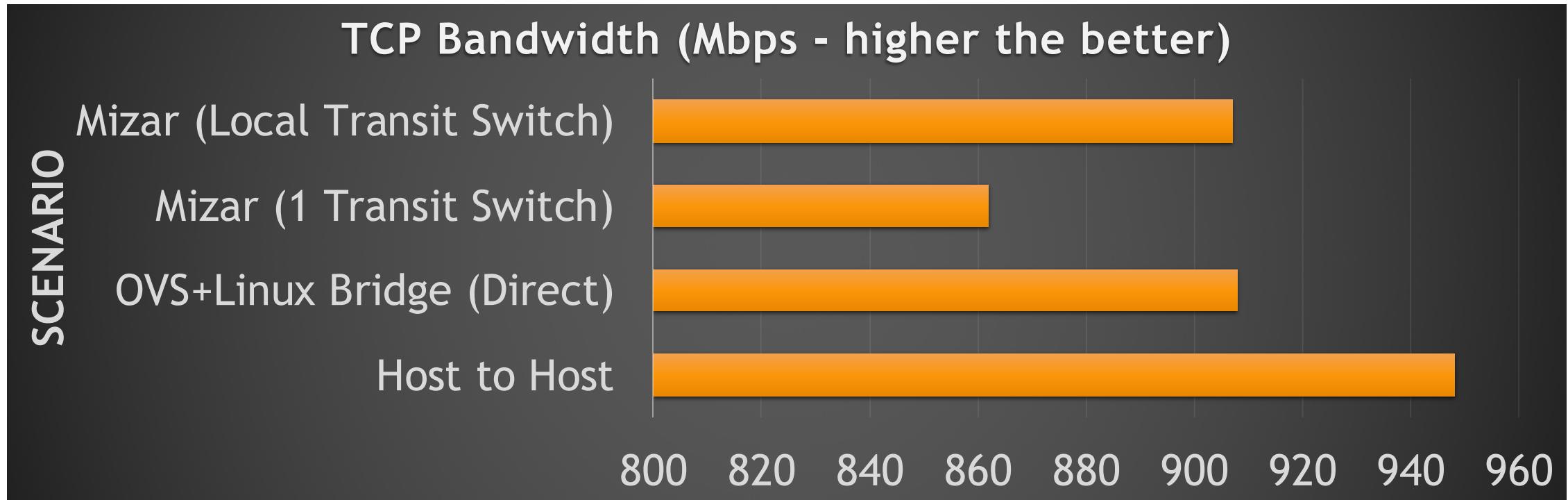
- **HIT:** Scale remains constant (until hitting test controller machine limits)
- Primarily overhead on the host from creating the virtual interfaces by executing shell command (~250 ms).
- **IMPROVEMENT:** Expected to improve with production ready control-plane as it makes use of netlink.

Round Trip Time Effect on End-user



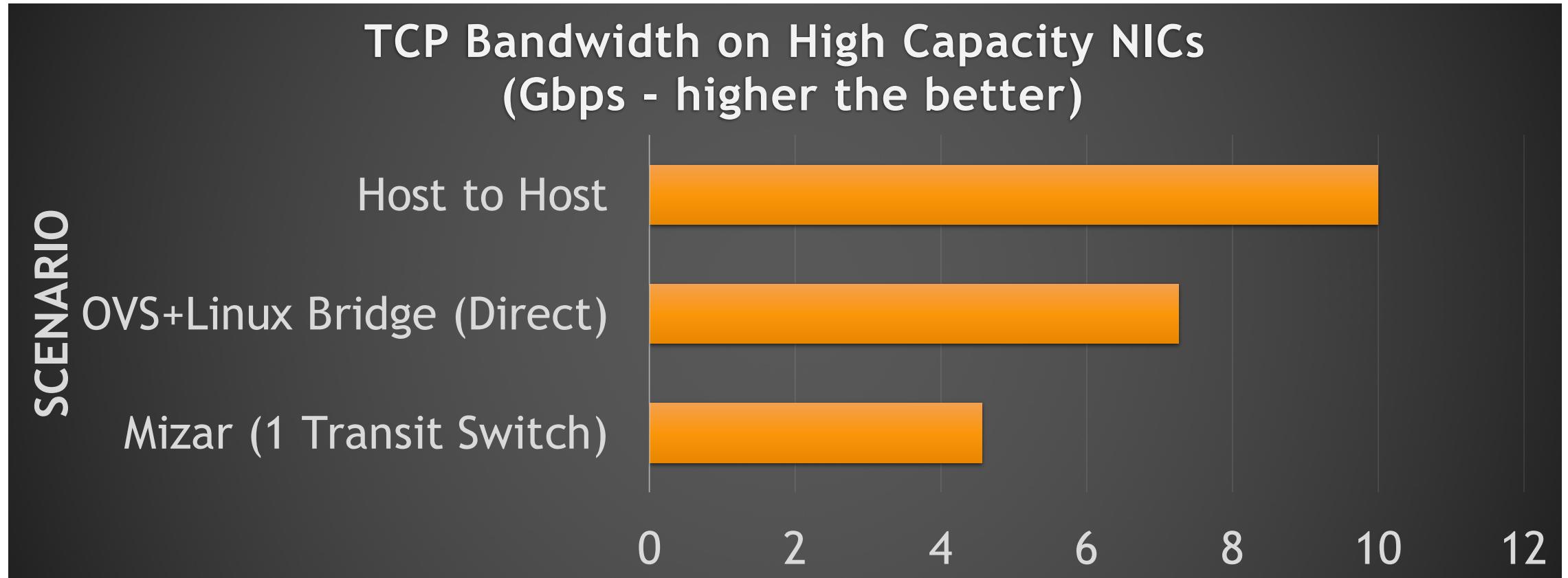
- **HIT:** Mizar direct path is faster than OVS+Linux Birdge. Though, Still has minimal impact on PPS and TCP BW
- **HIT:** Even with an increased latency due to the extra hop, the packet per second processed by endpoints remains close to line rate
- Primarily benefit of fast-path is latency sensitive applications.

TCP Bandwidth (On a slow NIC 1Gbps)



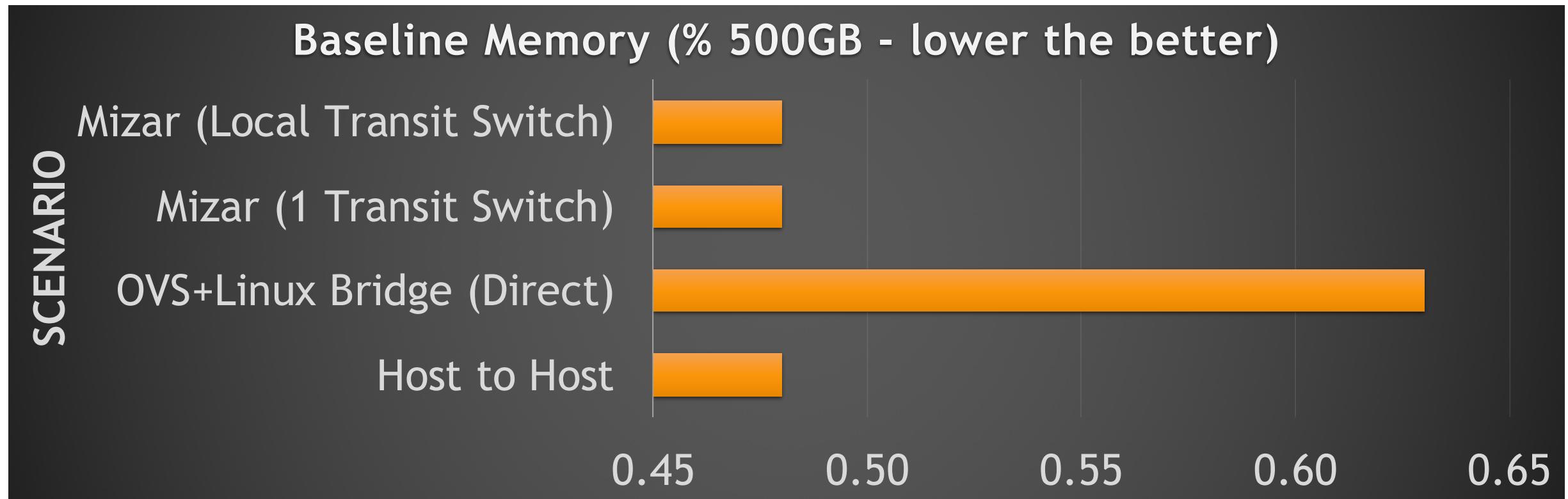
- **HIT:** Comparable throughput to OVS+Bridge (even though we don't use XDP driver mode). *This is applicable for NICs < 4Gbps*
- The transit switch hop accounts only for 5% less TCP throughput, which shall be negligible for very high bandwidth NICs. This is despite that RTT of the extra hop accounts for 45% more latency.

TCP Bandwidth (On a faster NIC 10 Gbps)

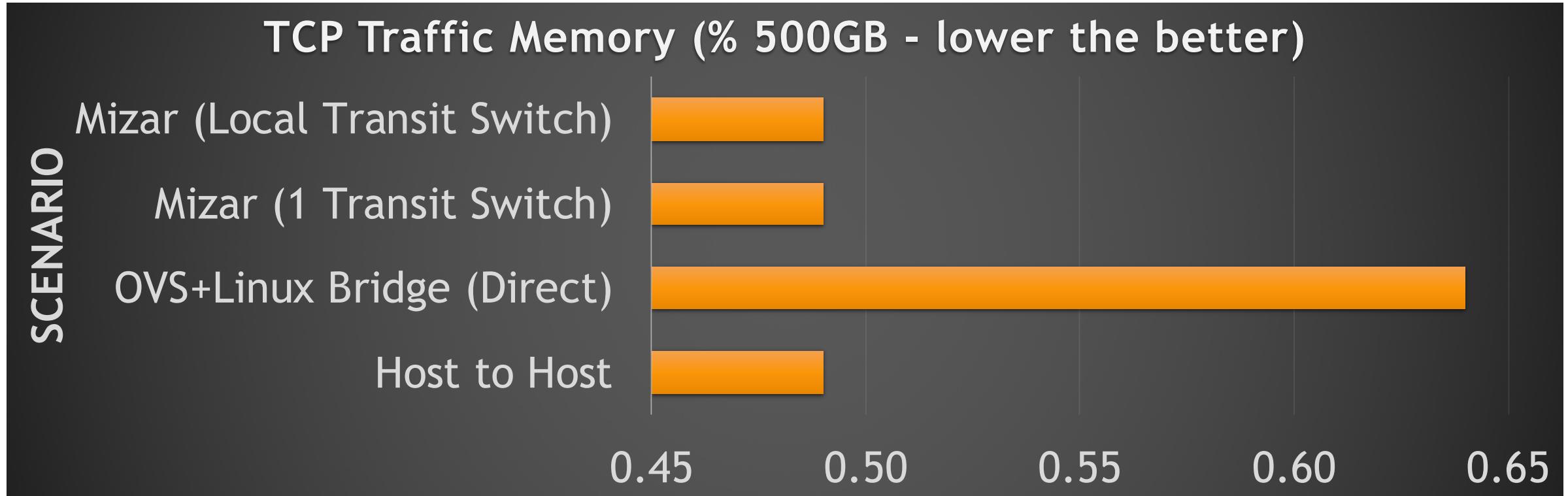


- **MISS:** The TCP bandwidth caps at around 4Gbps.
- **IMPROVEMENT:** Change to Driver mode (require support in NIC)
- **IMPROVEMENT:** Change on-host wiring architecture and reduce reliance on Transit Agent
- **IMPROVEMENT:** Improved device driver for veth

Memory Idle case

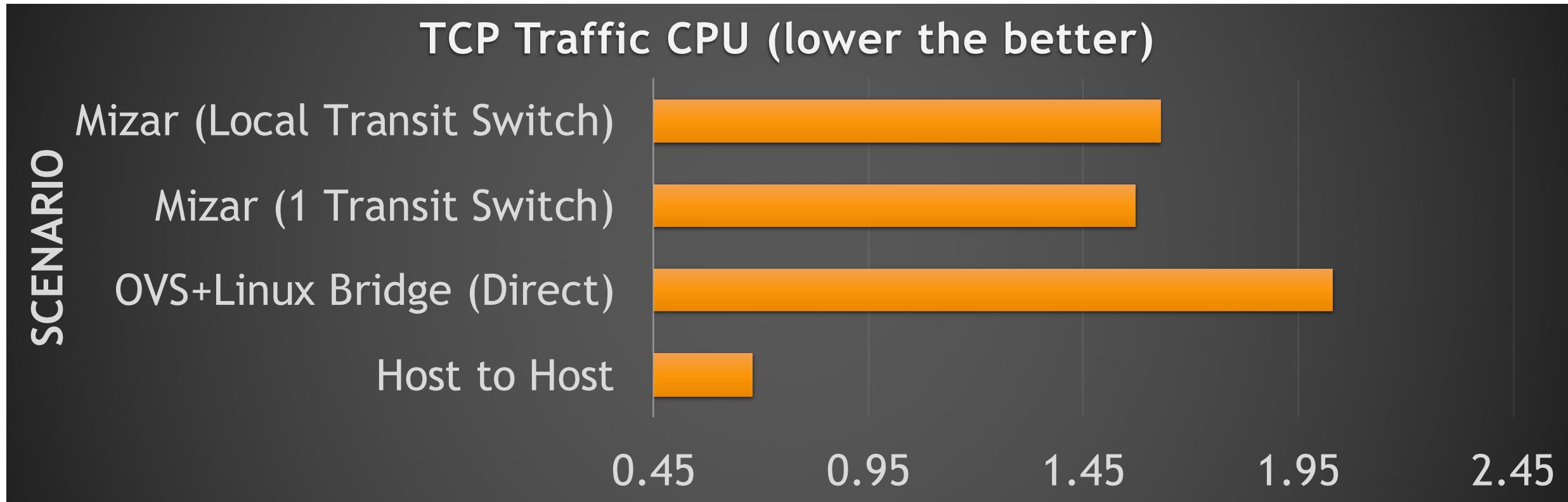


Memory During TCP Performance Tests



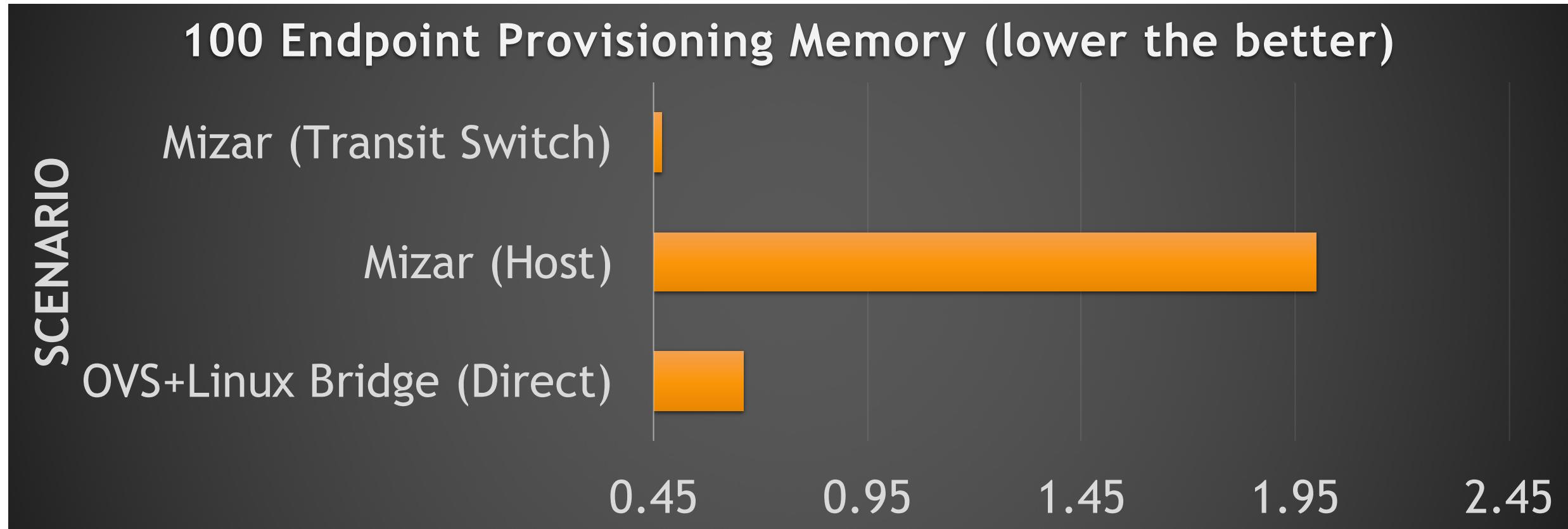
- **HIT:** Negligible Memory overhead very close to an idle host without networking constructs event with Traffic processing

CPU TCP Performance Tests



- **HIT:** CPU Overhead is much better than OVS + Linux bridge scenario

Memory Idle case (100 Endpoints per host)

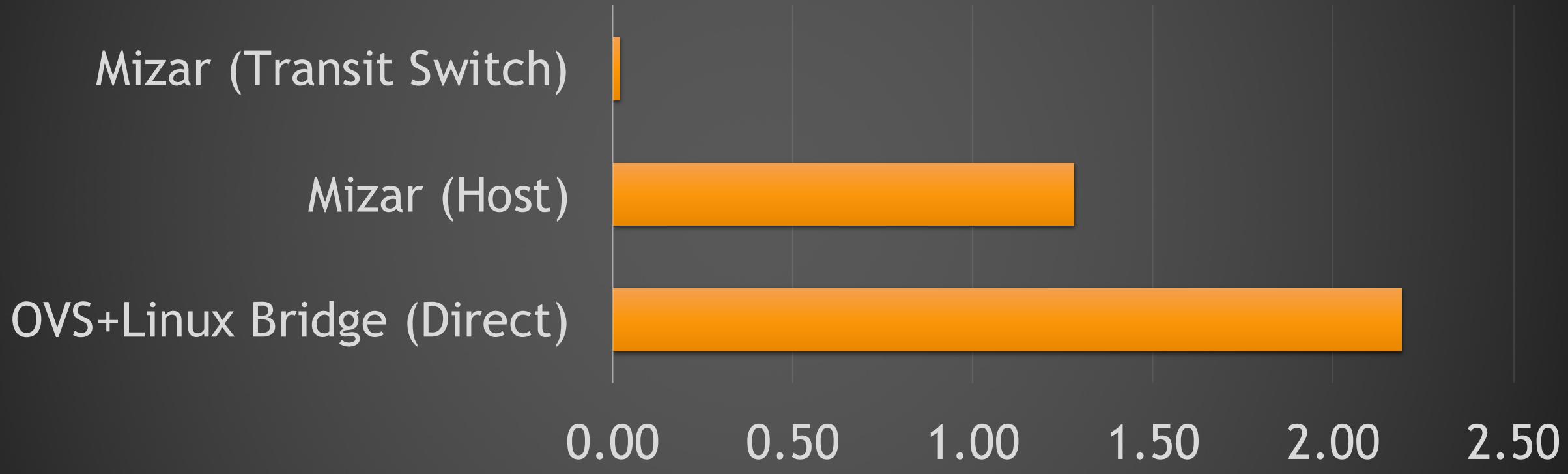


- **HIT:** Memory overhead on Transit Switch remain at baseline level
- **MISS:** On Host memory increases as we provision more endpoints
- **IMPROVEMENT:** Share one transit agent across multiple endpoints

CPU During TCP Performance Tests

100 Endpoint Provisioning CPU (lower the better)

SCENARIO



- **HIT:** Significantly less CPU overhead during provisioning on both transit switch and host