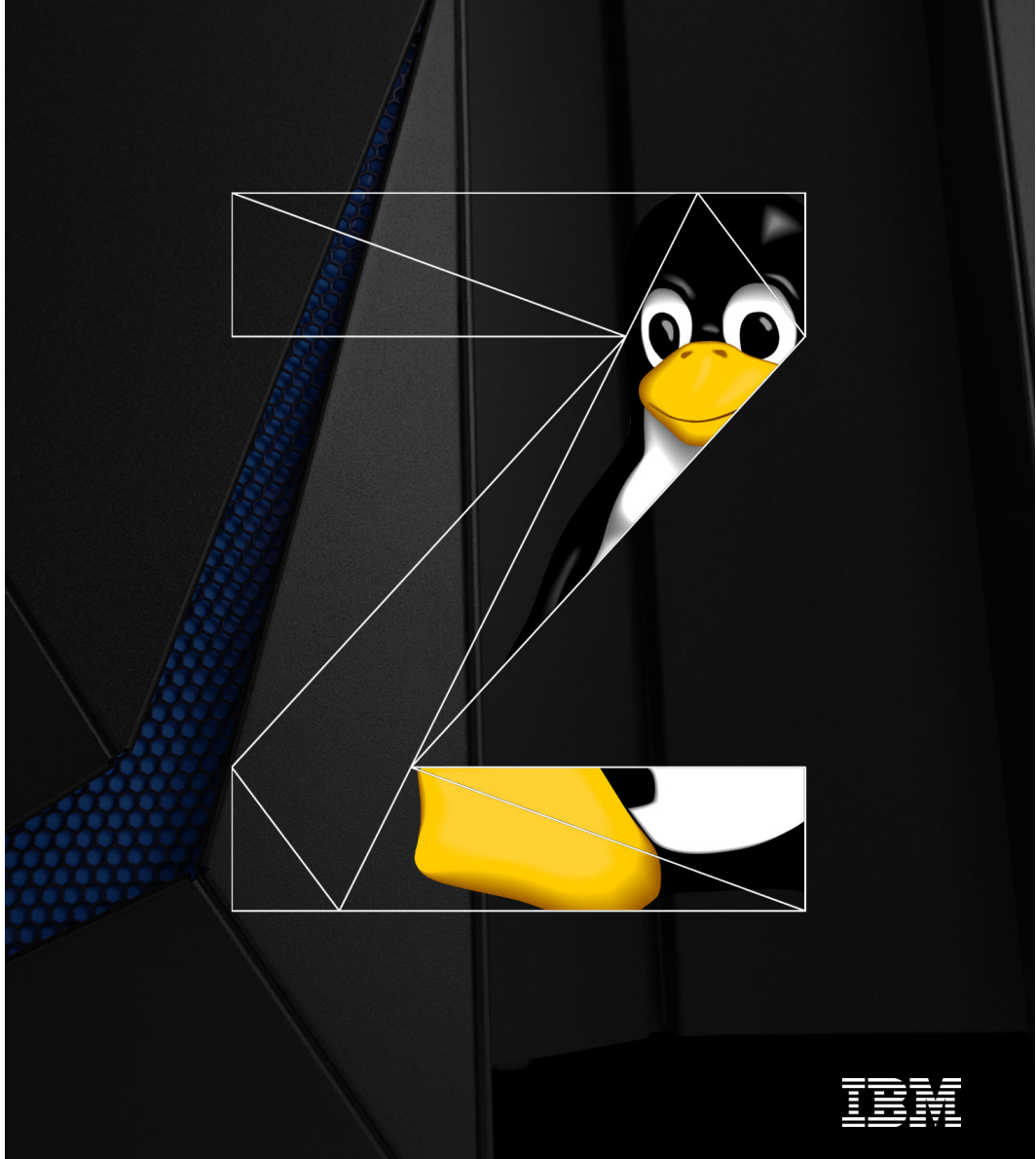
Linux on IBM z Systems (IBM Z)

KVM Network Performance

Exploitation of IP Routing

Dr. Juergen Doelle



Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

- A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/us/en/copytrade.shtml

The following are trademarks or registered trademarks of other companies.

- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- SUSE is a registered trademark of Novell, Inc. in the United States and other countries.
- Red Hat, Red Hat Enterprise Linux, the Shadowman logo and JBoss are registered trademarks of Red Hat, Inc. in the U.S. and other countries.
- Oracle and Java are registered trademarks of Oracle and/or its affiliates in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

Introduction

- **KVM provides a virtualization infrastructure that run separate and distinct operating systems in a virtual machine**
  - The Linux kernel becomes a Hypervisor
  - IBM Z platforms support the use of KVM to run multiple virtual machines or guests in each LPAR.

- **KVM virtualization on IBM Z and IBM LinuxONE**
  - Provides the performance and flexibility to address the requirements of multiple, differing Linux workloads.
  - KVM's open source virtualization on IBM Z and LinuxONE allow businesses to reduce costs by deploying fewer systems to run more workloads, sharing resources, and improving service levels to meet demand

- **HiperSockets is a IBM Z specific high performance network connection with a very short latency**
  - Provide fast LPAR to LPAR communication within the CEC
  - Often used to connect the middle-ware layer on IBM Z to the database layer on IBM Z

- **Exploiting HiperSockets in a KVM environment**
  - The KVM guest requires a layer 2 connection (Open Systems Interconnection layer)
  - For guest to guest communication HiperSockets can be used running both ends in layer 2 mode
  - They can not be used to connect a guest via layer 2 directly with a back- end on z/OS requiring layer 3.

- **Exploiting IP routing**
  - The KVM host act as gateway
  - Transparent and seamless communication across network interfaces using different OSI network models (Layer 2 and Layer 3).
  - Removes the limitations imposed when a KVM guest interface is bound to host network interface
  - Enables software defined networking

Overview – KVM Guest Connectivity

- **KVM guest connectivity on IBM Z**
  - Standard variants are
    - MacVTap and
    - Open vSwitch (OVS) with physical network cards

Pro and Con

- **MacVTap**
  - Pro:
    - Fastest interface
    - The driver manages the MAC address registration for the OSA card
  - Con:
    - Each external interface needs one guest interface
    - Changes in the host interfaces need to be applied to all guest
- **OVS with a network card attached**
  - Pro:
    - Many additional features
    - Single point of management for host network changes
  - Con:
    - For each external interface one OVS with a separate guest interface is needed
    - Requires promiscuous mode of the OSA card
      - only available for one LPAR on one OSA card (many KVM hosts need many OSA cards)
      - or manual MAC address registration and management is required

The Alternative: Routing!

- **Use the host as gateway**
- **Setup the OVS with an virtual network interface (no physical network card)**
  - All guests are connected to the OVS
  - The OVS has only one virtual network interface into the host
  - The host routes the requests to the appropriate interfaces

- **Con:**
  - Performance Overhead due to additional software layers
  - Planning effort for the subnets for the KVM guests
- **Pro:**
  - Guest network can be managed independently from the host physical network
  - Very flexible network design
    - Simplifies the guest management
    - Guest can easily and transparent access multiple host networks
    - Different MTU sizes between guest and the host networks are possible → managed by host IP stack
    - Supports any mix of layer 2 or 3 interfaces → managed by host IP stack

- **The feature allows software defined networks!**
  - Just the routing rules define who can access what
  - Changes in routing rules take immediately effect

- **But there are also severe pitfalls :-(**

OSI Network Layers and IBM Z - Some Backgrounds

- **Open Systems Interconnection (OSI) network models**
  - Layer 1 physical layer
  - Layer 2 data link (MAC address based, frames)
  - Layer 3 network layer (IP address based, packets)
  - Layer 4 transport layer (TCP, UDP)

- **Traditional IBM Z systems often run on Layer 3 (for example z/OS, z/VSE, and z/VM)**
  - KVM guests run on Layer 2 only

- **Layer 2 and Layer 3 network packets look different!**
  - OSA cards can interpret both
  - The Linux TCP/IP stack can handle both formats

- **Communication between a KVM guest (Layer 2) and a Layer 3 partner over HiperSockets**
  - Is not possible directly, because HiperSockets require that both ends are configured for the same type of packets!
  - Options are
    - A Linux bridge with NAT mode or
    - An Open vSwitch and routing over the IP Stack from the KVM host

The Idea:

I want a network connectivity with a KVM guest, which supports

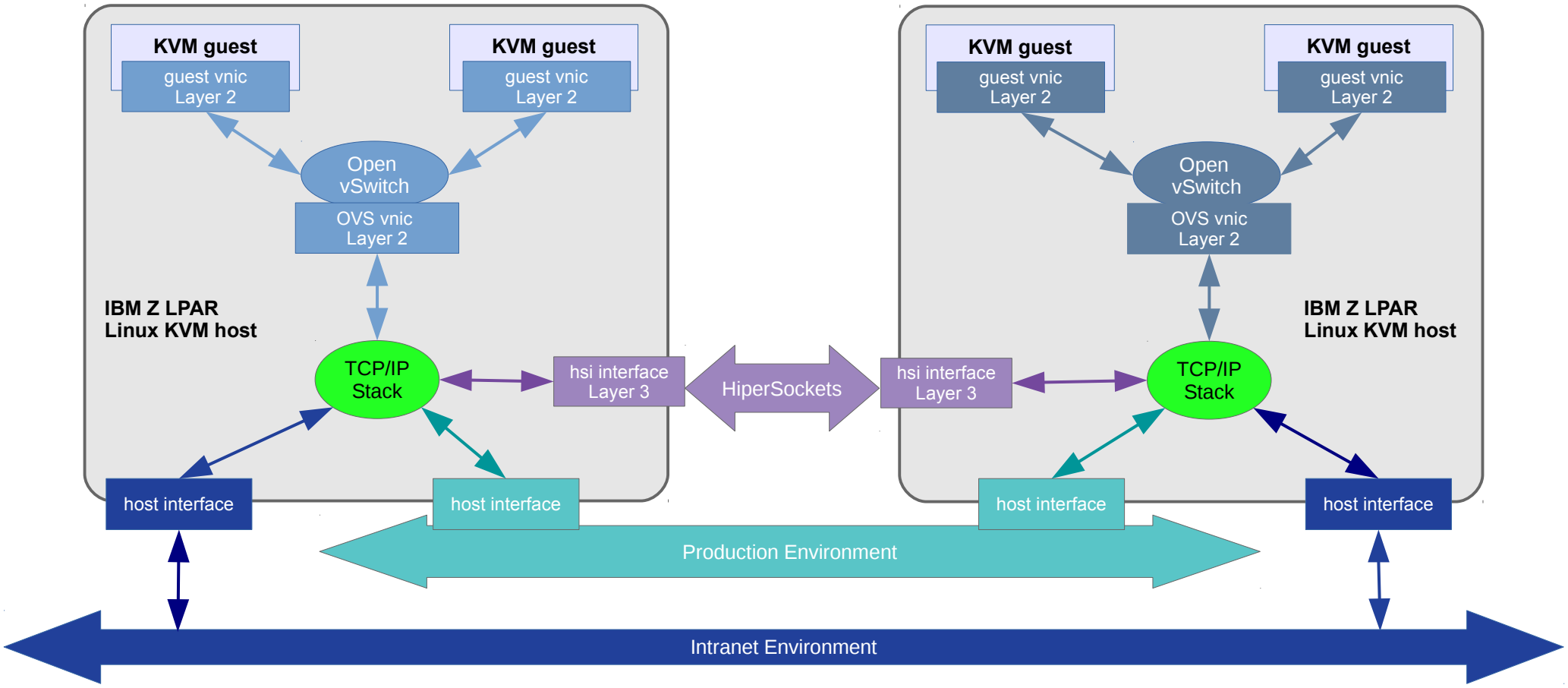
- A simple guest setup
- The guest should access a complex network with more than one subnet
- Just via host routing
- Without changing the global company routing rules and switches

Network Access from Guests on KVM on IBM Z

- **Guest to guest communication**
  - On the same KVM host on one LPAR
    - pure virtual network (Scenario I) for example via Open vSwitch
    - everything runs Layer 2
  - On KVM hosts on different LPARs on the same CEC
    - HiperSockets (Scenario IIIa), target runs in Layer 2

- **Guest to any server communication**
  - Within a distinct LAN (same network e.g. 10.x), for example production environment
    - via OSA cards (Scenario II) to the same subnet as the host
    - **all ends are under admin control** (routing, default gateway, name resolution)
  - To different LPARs on the same CECs (e.g. z/OS)
    - via HiperSockets (Scenario IIIb)
    - guests run Layer 2 and back-ends might be require Layer 3
    - **all ends are under admin control**
  - To resources in an intranet, for example to access update servers
    - via OSA cards (Scenario IV) to a different subnet
    - **only the local end is under admin control!**

Final KVM Network Configuration – Each Guest is Connected to Four Subnets



- **Each color represents a separate subnet!**

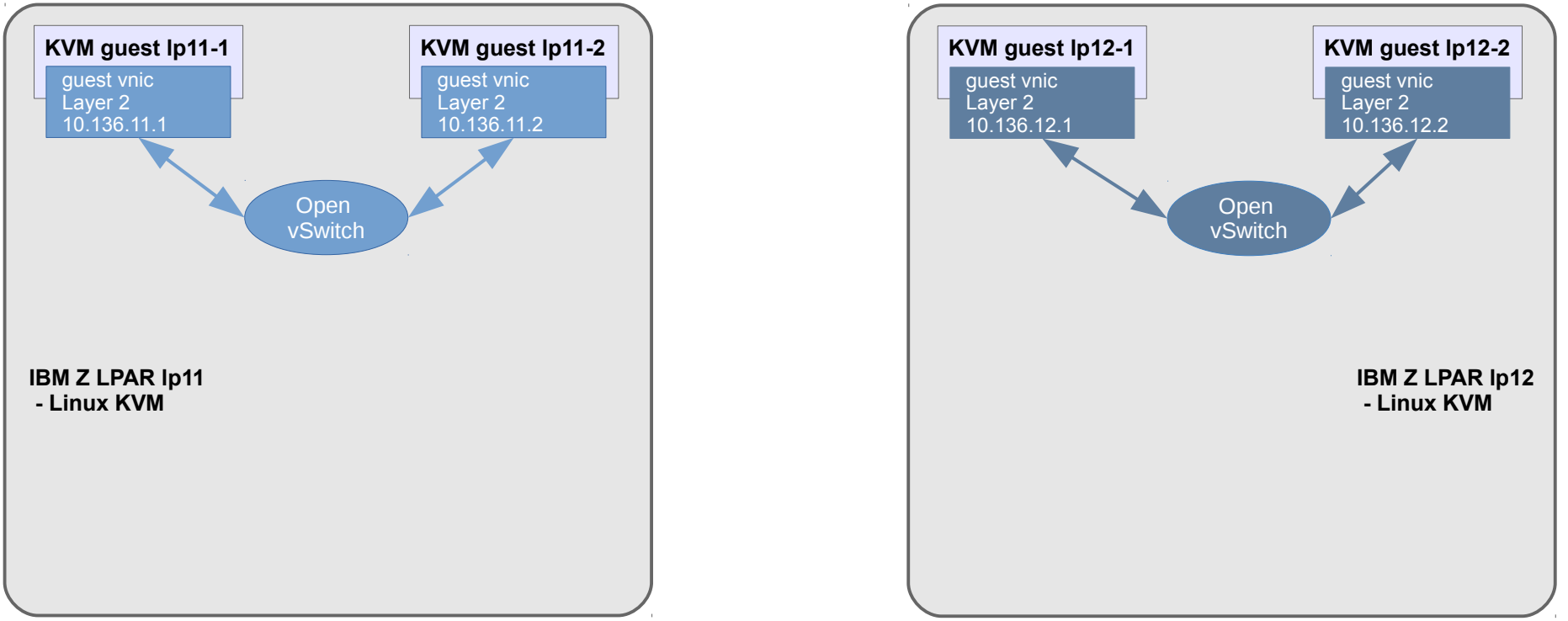
Setup - Howto

- **All that can be established with routing via the IP stack from the KVM host**
- **We have 4 scenarios**
  - Scenario I: Pure virtual network
  - Scenario II: OSA card to the same subnet as the KVM host
  - Scenario III: HiperSockets connecting Layer 2 (IIIa) and Layer 3 resources (IIIb)
  - Scenario IV: OSA card and to a different subnet

- **On the following charts we are setting up that environment in 4 steps**
  1. Start with guests connected to an Open vSwitch
  2. Continue with connecting the guests with routing to the subnet of the KVM host
     - members of this net can reach the host directly
  3. Add a HiperSockets connection to another LPAR
  4. Finally open access for the guest to a different subnet
     - members of this net can reach a host interface, but have no access to the guests

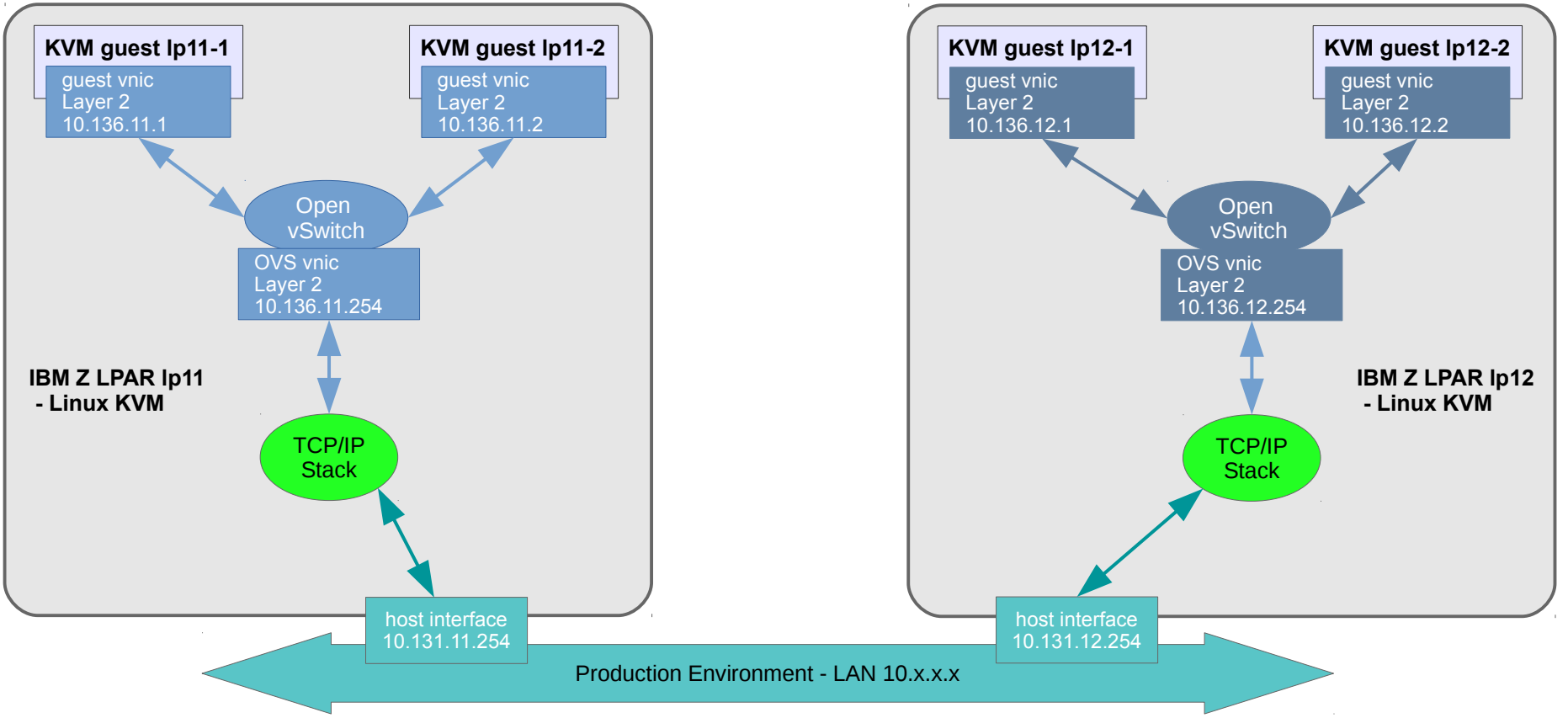
- **It is all about arp and routes**

Scenario I: Pure Virtual Network within one LPAR



- **The guest routes 10.0.0.0/8 via its vnic**
  - Guest to guest communication is provided by default from Open vSwitch, all guests are in the same subnet
  - Just add an interface to the OVS via guest domain.xml
  - Netmask 255.0.0.0 is specified in the guest interface definition
  - Required routing rules are created automatically when the guest network comes up
  - At the moment this is an **isolated** network! Such a setup would require additional network interfaces at least in one guest

Scenario II: OSA Card to the same Subnet as the KVM Host



- **To connect the OVS to a network a virtual network interface (vnic) is added to the OVS**
  - This interface is now used as the default gateway for the guests on this OVS!

Scenario II: OSA Card to the same Subnet as the KVM Host – Host Setup

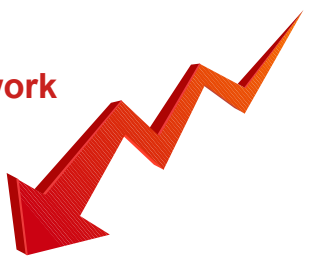
KVM host setup

- **Use the internal port of the OVS as virtual interface**
  - The internal port has the same name as the OVS
    - *ip addr change 10.136.11.254/24 dev <OVS-interface>*
      *ip link set <OVS-interface> mtu **57344** txqueuelen 2500 **up***

- **Enable ip forwarding**
  - *sysctl -w net.ipv4.conf.<OVS-interface>.forwarding=1*
    sysctl -w net.ipv4.conf.<10.x interface>.forwarding=1
    for the host interface into the 10.x net and the OVS interface
  - Use that parameter very restrictive for security reasons!

- **Enable proxy_arp!**
  - *sysctl -w net.ipv4.conf.<OVS-interface>.proxy_arp=1*
    sysctl -w net.ipv4.conf.<10.x interface>.proxy_arp=1
    for the involved host interface into the 10.x net and the OVS interface
  - Use that parameter very restrictive and **only** under control of **arptables** → continue reading!
  - **Note:** proxy_arp feature only works with layer2 interfaces!

- **The proxy_arp feature enables an interface to reply for arp requests to systems from another network**
  - The interface replies then to arp request for everything it thinks it can reach internally based on the routing rules
  - But the 'everything' part is normally not intended!
  - Control this behavior with arptables (also for security reasons) → continue reading, see also chart 'Pitfalls'



Scenario II: OSA Card to the same Subnet as the KVM Host – Host Setup arptables

KVM host setup

- **arptables**
  - With arptables the reply behavior of a network interface to arp requests can be controlled (highly recommended when proxy_arp is enabled)

- **Example: the system should reply for local guest addresses (10.136.11.xx) and for the 10.x host interface only**
  - default INPUT policy is ACCEPT, set to DROP for security reasons
    - `arptables -P INPUT DROP`
    - This restricts to reply only to request pattern which are explicitly allowed
  - Allow arp requests for guest subnet and host interface
    - `arptables -I INPUT -d 10.131.11.0/24 -i <host interface to 10.x net> -j ACCEPT`
      `arptables -I INPUT -d 10.136.11.0/24 -i <host interface to 10.x net> -j ACCEPT`
  - Allow all arp requests from guests on the OVS interface !
    - `arptables -I INPUT -i <OVS-interface> -j ACCEPT`
  - The order is important, especially when having accept and drop rules. The first hit is been used
    - -I <chain> adds at the beginning, -A <chain> at the end
    - Check results with `arptables -L -vn`

Scenario II: OSA Card to the same Subnet as the KVM Host – Host Setup iptables

KVM host setup

- **Firewall (iptables)**
  - default FORWARD policy is ACCEPT, change to DROP for security reasons
    - `iptables -P FORWARD DROP`
    - This restricts forwarding only to request pattern which are explicitly allowed
  - Allow ip forwarding via the card for input and output
    - `iptables -t filter -I FORWARD 1 -i <host interface to 10.x net> -j ACCEPT`
      `iptables -t filter -I FORWARD 1 -o <host interface to 10.x net> -j ACCEPT`
  - Allow ip forwarding via the OVS interface for input and output

    `iptables -t filter -I FORWARD 1 -i <ovs interface> -j ACCEPT`
    `iptables -t filter -I FORWARD 1 -o <ovs interface> -j ACCEPT`
  - Order of rules is important, see arptables slide
  - Check with `iptables -L -vn`

Scenario II: OSA Card to the same Subnet as the KVM Host – Host Setup Routing

- **The routes for the OVS are setup automatically from the host operating system**
  - Manual setup: *ip route add 10.136.11.0/24 dev <OVS-interface>*

- **Sample host routing table**

```
Kernel IP routing table
Destination  Gateway     Genmask         Flags   …     Iface           Comment
10.0.0.0     0.0.0.0     255.0.0.0       U       …     o5s_10g_1       net route to production net
10.136.11.0  0.0.0.0     255.255.255.0   U       …     br-ex           net route to the guests
…
```

  - Generated automatically means based on the device definition in the ifcfg files

- **MTU size**
  - Recommendation is to configure the OVS with the very large MTU size 57344
    - Advantage for guest to guest communication and when HiperSockets are involved (more details later)
    - Define guest and OVS device MTU size consistent (always)!

Scenario II: OSA Card to the same Subnet as the KVM Host – Guest Setup

- **KVM guest setup**
  - One virtio network interface connected to the OVS in the domain.xml
- **Routing**
  - Use the OVS-interface as gateway for the guest (independent from the host interfaces)
  - This must be setup manually
    - *ip route add default via 10.136.11.254 dev <guest interface>*
    - or for example in /etc/sysconfig/network/routes: *default 10.136.11.254 - <guest interface>*
- **Sample guest routing table**

```
Kernel IP routing table
Destination  Gateway        Genmask          Flags  …    Iface       # Comment
0.0.0.0      10.136.11.254  0.0.0.0          UG     …    vic_10g_2   # OVS vnic as default gateway
10.136.11.0  0.0.0.0        255.255.255.0    U      …    vic_10g_2   # route 10.x net over this interface
```

- **MTU size**
  - Recommendation is to configure the guest and the OVS with the very large MTU size 57344
    - Advantage for guest to guest communication and if HiperSockets are involved
    - Define guest and OVS device MTU size consistent (always)!

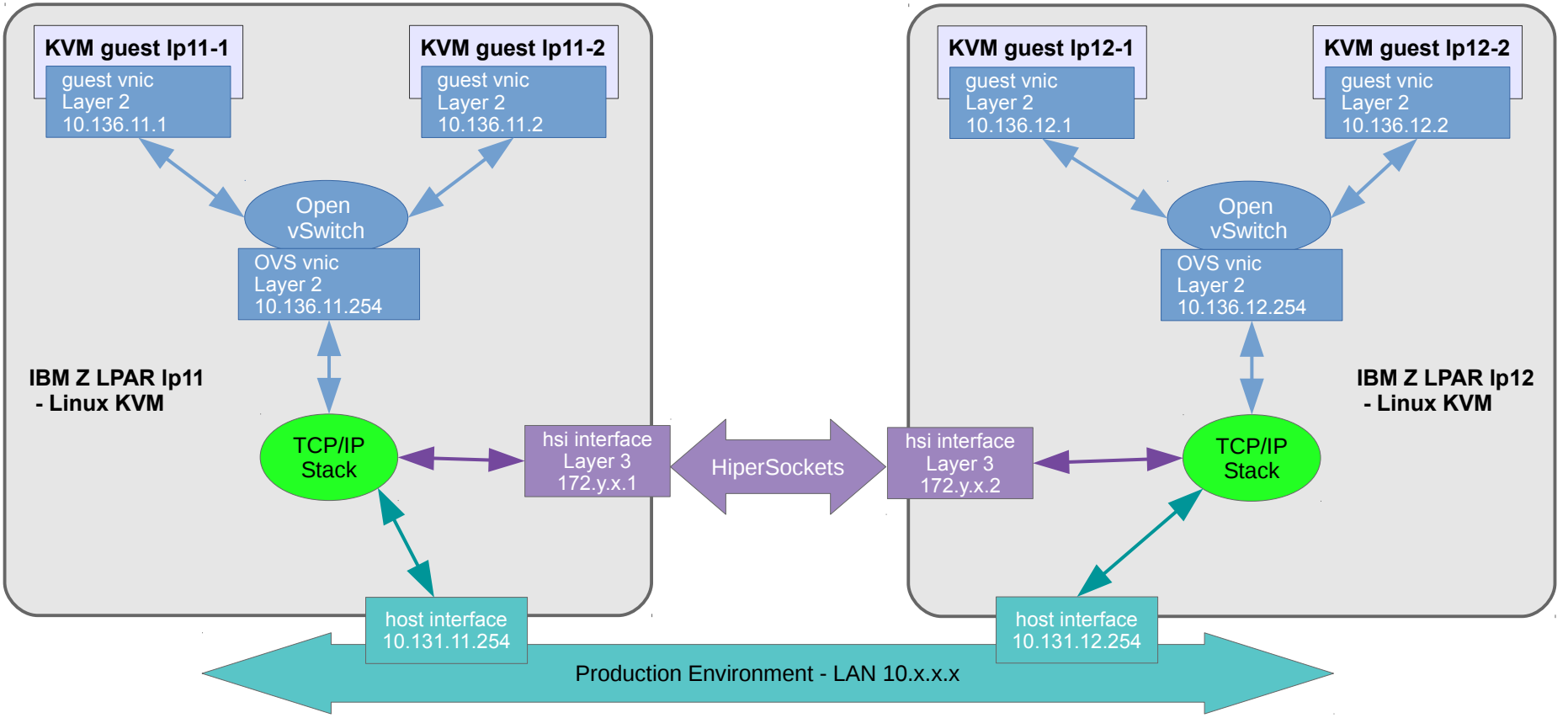
Scenario II: OSA Card to the same Subnet as the KVM Host – Pitfalls

- **There are some very dangerous pitfalls!**

- **Multiple host network interfaces with its own IP addresses in the same subnet and proxy_arp enabled**
  - Symptom: Makes (all) hosts in the 10.x network unreachable
  - With proxy_arp enabled an interface replies to all arp requests which can be reached via the routing rules of the system, except himself
  - Two network interfaces into the same subnet leads to the effect that one interface replies for arp requests for all IP addresses in this net
    - but it can not route the packages to the target, because it is the same subnet
    - and the net is dead!

- **MTU size with attached network devices (bridges, MacVTap)**
  - Symptom: network I/O errors on the receiver side
  - Define the MTU size for all interfaces **attached** on the OVS consistently!
  - Network cards have typically an MTU size of 1492 or 8192
  - Set guests and vnic from OVS to the sameMTU (e.g. 57344)
    - With **routing** the packets are fragmented by the IP stack as appropriate for the target device.
  - When guests have MTU size 57344 and a network card is **attached** to the OVS packet then packets going over the network card are just get **cut off** when they are larger than the MTU size
    - That means as long as the packet size is smaller than the smallest MTU size everything works fine
    - But larger packets result in I/O errors on the receiver site, because the checksum fails

Scenario III: HiperSockets



- **Challenge**
  - The guests should use HiperSockets only for communication with the other LPAR
  - External communication from 10.x should not be routed over the HiperSockets connection

Scenario III: HiperSockets – Host Setup

KVM host setup

- **Add a HiperSockets interface to both LPARs with a different subnet**
  - They are not visible outside the LPAR when routing is setup correctly
  - MTU size 56K, netmask 255.255.255.0 (/24)

- **enable ip forwarding for the HiperSockets interfaces**
  - *sysctl -w net.ipv4.conf.<hsi-interface>.forwarding=1*

- **enable proxy_arp for the HiperSockets interfaces**
  - Not required for HiperSockets running layer 3

- **arptables**
  - Not required when running the HiperSockets in layer 3 without proxy_arp

- **Firewall (iptables)**
  - Allow ip forwarding via the for the HiperSockets interfaces for input and output
  - *iptables -t filter -I FORWARD 1* ***-i*** *<HiperSockets interface> -j ACCEPT*
    iptables -t filter -I FORWARD 1 ***-o*** *<HiperSockets interface> -j ACCEPT*

Scenario III: HiperSockets – Host Setup

Routing

- Add a net routing rule (the first self defined rule)
  - for the guest subnet of the **other** LPAR with netmask /24
  - with the HiperSockets interface from the **other** LPAR as gateway!
  - via the **local** HiperSockets interface

- Example /etc/sysconfig/network/routes:
  - on LPAR11: `10.136.12.0  172.31.56.2  255.255.255.0 <HiperSockets interface>`
    on LPAR12: `10.136.11.0  172.31.56.1  255.255.255.0 <HiperSockets interface>`

- Or using the ip command
  - on LPAR11: `ip route add 10.136.12.0/24 via 172.31.56.2 dev <HiperSockets interface>`

Host routing table

```
Destination     Gateway         Genmask         Flags   …   Iface      # Comment
10.136.12.0     172.31.56.2     255.255.255.0   UG      …  hs56k_0     # the new rule defined above
172.31.56.0     0.0.0.0         255.255.255.0   U       …  hs56k_0     # that is generated automatically

…                                                                      # that is what we had so far:
10.0.0.0        0.0.0.0         255.0.0.0       U       …   o5s_10g_1  # all generated automatically
10.136.11.0     0.0.0.0         255.255.255.0   U       …   br-ex      #
```

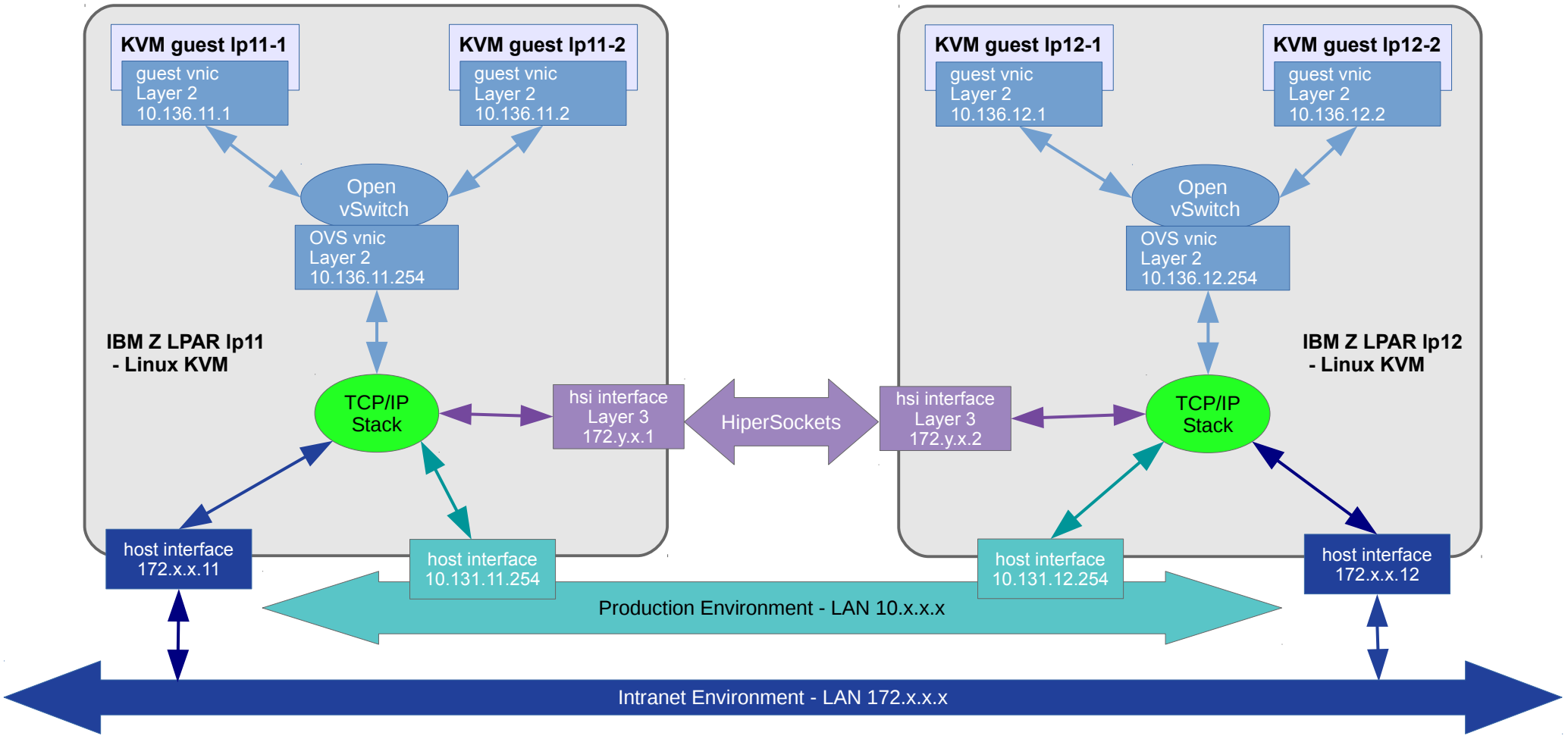
- Generated automatically means based on the device definition in the ifcfg files

Scenario III: HiperSockets

- **Guest Setup: no changes required :-)**

- **Pitfalls**
  - The challenge here is to avoid having two paths to the same guest
    - for example via HiperSockets from the other LPAR
    - and via the local 10.x net interface
  - Solution:
    - Prevent that the KVM host replies to arp requests for systems not hosted locally
      - For example due the default Input policy drop in arptables without accept rule for this interface
    - Ensure that the routing setup is symmetric on both KVM hosts

Scenario IV: OSA Card to a different Subnet



Scenario IV: OSA Card to a different Subnet - Challenges

- **Often enterprise services are provided in a special subnet**
  - For example update servers in the intranet
  - The access is only from the guest to these services
  - IP addresses and policies are managed on company level

- **The challenge for that setup**
  - The host has an interface into that net
  - But the guest interfaces can not be reached from there
    - We could route guest requests to the 172.x host, but the packages do not find the way back
    - Additional routing rules on the gateways might help, but that cause additional admin and maintenance effort and might conflict with policies

- **The solution is to use the Linux NAT feature with iptables**
  - The packets from the guest appear with the IP address of the OSA card from the KVM host in the 172.x net
  - The guest can access systems in the 172.x net
  - But they can not be accessed from there
    - And that might be intended for isolation reasons
  - All required changes are transparent for the guest

Scenario IV: OSA Card to a different Subnet – Host Setup

KVM host setup

- **Add a host network interface to the target network 172.x**

- **sysctl settings**
  - ip forwarding - not required
  - proxy_arp - not required
- **arptables - not required**

- **NAT setup is created using iptables**

```
iptables -t nat -A POSTROUTING -o <host interface>_1 -j MASQUERADE
iptables -A FORWARD -i <host interface 172.x> -o <ovs interface> -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -i <ovs interface>        -o <host interface 172.x>                              -j ACCEPT
```

Scenario IV: OSA Card to a different Subnet – Host Setup

KVM host setup

- **Routing**
  - In our case the default gateway is already in the 172.x net,
    - /etc/sysconfig/network/routes: `default 172.18.0.1 - -`
    - or `ip route add default via 172.18.0.1 dev <host interface>`
  - In case the default gateway is in the 10.x net
    - Add a gateway statement for the 172.x net, similar to the Scenario III: HiperSockets
    - The net route statement is created automatically based on the ifcfg file for that interface

- **Host routing table**

```
Destination     Gateway         Genmask          Flags   …   Iface      # Comment
172.18.0.0      0.0.0.0         255.254.0.0      U       …   o5s_1g_1   # the new 172.x net route

…                                                                       # that is what we had so far:
0.0.0.0         172.18.0.1      0.0.0.0          UG      …   o5s_1g_1   # system wide default gateway
10.0.0.0        0.0.0.0         255.0.0.0        U       …   o5s_10g_1  # production net
10.136.11.0     0.0.0.0         255.255.255.0    U       …   br-ex      # guest subnet
10.136.12.0     172.31.56.2     255.255.255.0    UG      …   hs56k_0    # HiperSockets
172.31.56.0     0.0.0.0         255.255.255.0    U       …   hs56k_0    # gateway for HiperSockets
```

Scenario IV: OSA Card to a different Subnet – Guest Setup

- **KVM guest Setup**
  - Add a net routing rule (the second self defined rule)
    - /etc/sysconfig/network/routes *172.18.0.0/15 – dev <guest interface>*
    - or *ip route add 172.18.0.0/15 dev vic_10g_2*

- **Sample guest routing table**

```
Kernel IP routing table
Kernel IP routing table
Destination  Gateway        Genmask          Flags  …    Iface       # Comment
172.18.0.0   0.0.0.0        255.254.0.0      U      …    vic_10g_2   # the new 172.x net route

0.0.0.0      10.136.11.254  0.0.0.0          UG     …    vic_10g_2   # OVS vnic as default gateway
10.136.11.0  0.0.0.0        255.255.255.0    U      …    vic_10g_2   # route 10.x net over this interface
```

The next step:

- **OK, now it works :-)**
  - What is the pay-off?

Performance Considerations

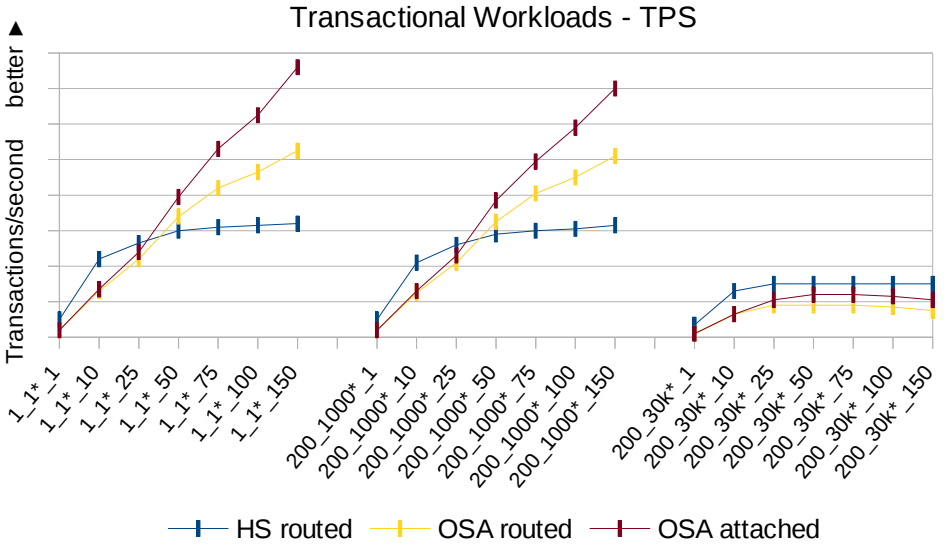
- **The routing increases the pathlength → How big is the Overhead?**

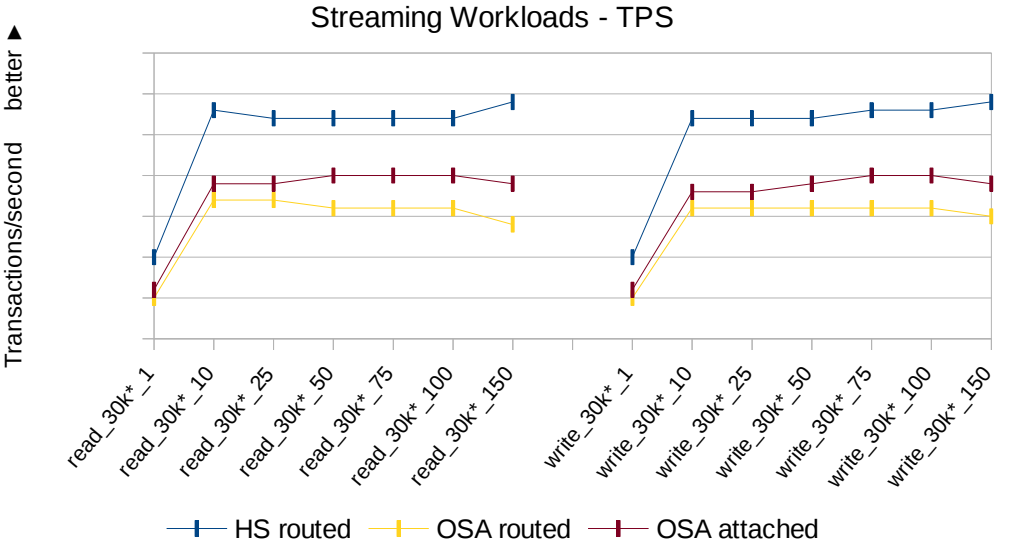
- **uperf (www.uperf.org) with two types of workload**
  - Transactional workload (RR) - a (send) request followed by (receiving) a response.
    - Typical for web servers as users interact with websites using web browsers.
  - Streaming workloads (STR) - Considered uni-directional
    - The Request-and-Response ratio can be well over 1:1,000,000 or higher.
    - Network servers experience when supporting operations such as backup/restore, large file transfers and other content delivery services.

- **Note:**
  - One uperf connection represents a permanently firing workload, corresponds to much higher number of real world connections!

- **Naming convention: {category}{1c}-{requestsize}x{responsesize}--{users}**
  - Example: rr1c-200x1000--10 describes a Request-and-Response test
    - sending a 200 byte request
    - receiving a 1000 byte response
    - being generated each by 10 concurrent users (connections)
  - Example: str-readx30k--50
    - describes a streaming test read of 30KB datagrams
    - being generated each by 50 concurrent users (connections)

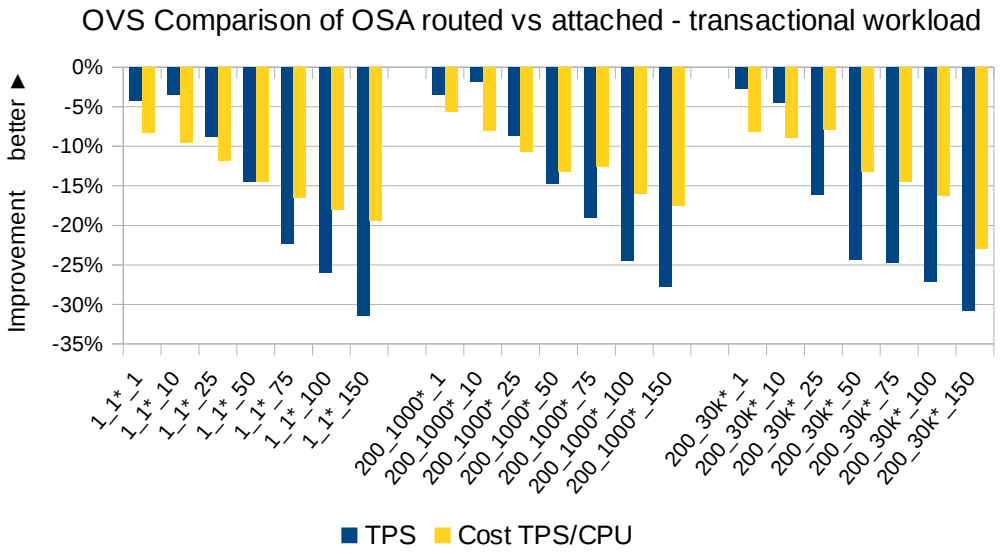
Network Connectivity for Open vSwitch: Transactional and Streaming Workloads





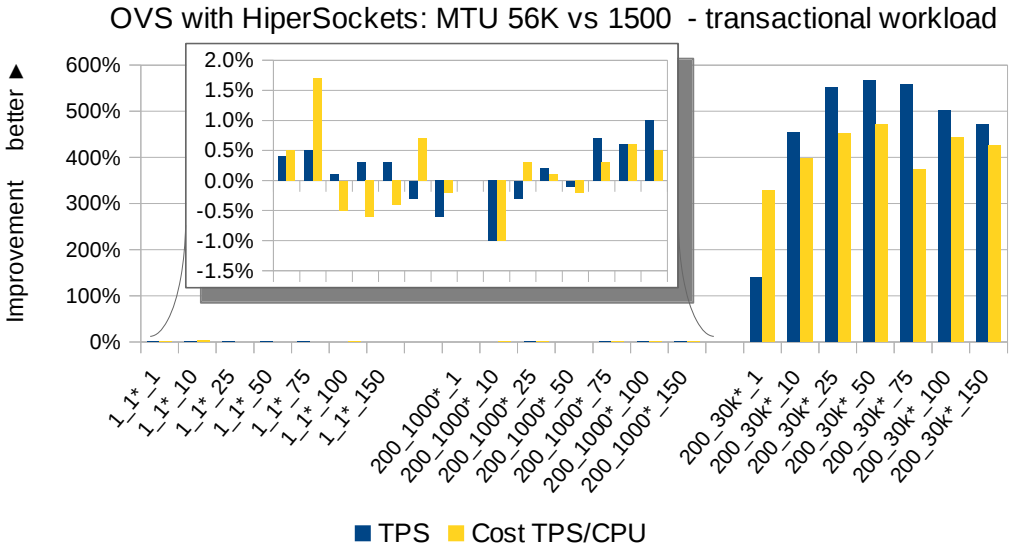
- **The performance result depends on workload type and packages size**
  - **HiperSockets routed:** outperforming setup for
    - Up to a medium amount (1-25) of parallel connections with transactional workloads, throughput 2.7x - +20% compared to attached OSA
    - For larger packages when the environment runs with a MTU size of 56k, especially streaming workloads (>+50% compared to attached OSA)
  - **OSA routed :** a good setup for up to a medium amount of parallel connections with a request size around 1000 bytes and smaller
    - With 50 and more uperf connections the attached OSA card performs better
  - **OSA attached:** best setup for high numbers of parallel connections with a request size around 1000 bytes and smaller

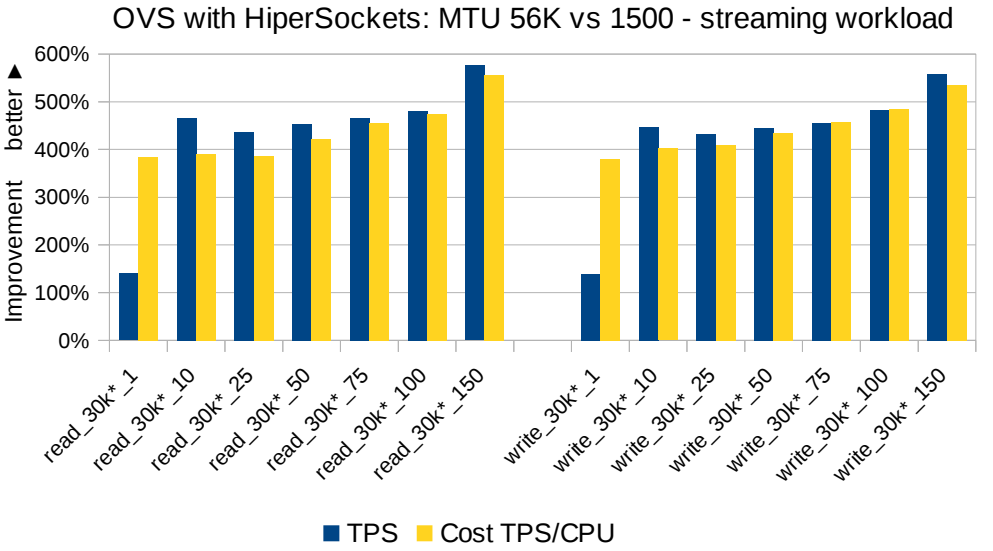
Network Connectivity for Open vSwitch: *OSA Routed vs OSA Attached*



- **Comparing the routed OSA card environment with the attached OSA card in detail.**
- **Up to 25 parallel uperf connections and a package size of 1500 and smaller**
  - Throughput degradation is around 5-10% which is moderate
  - Overhead in cost (throughput per CPU) is below 10%
- **Streaming workloads behave similar**

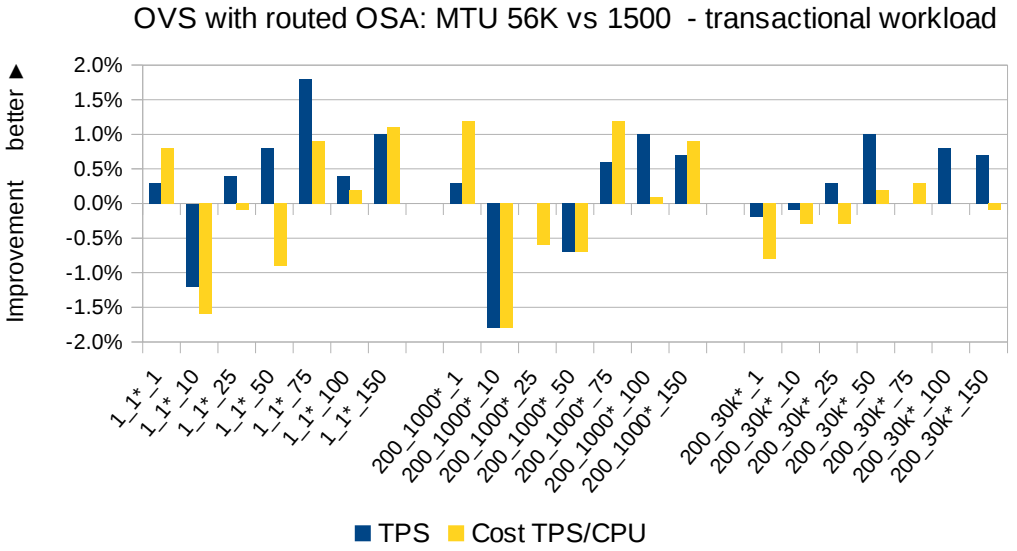
Guest to Guest Communication via OVS and *Routed HiperSockets*: Scaling MTU size 56k vs 1500

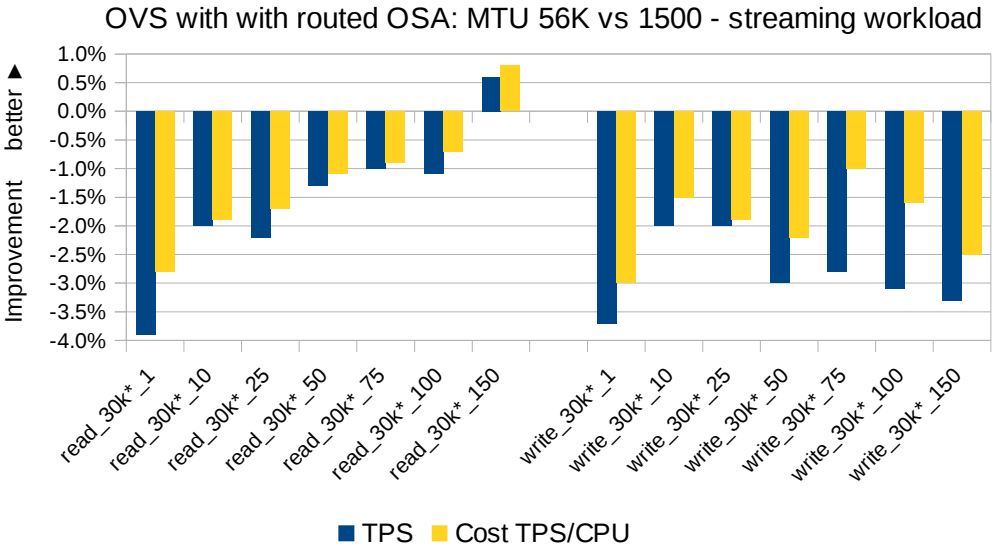




- **Has the MTU size of the guest and OVS environment an impact when routing via HiperSockets?**
  - The setup uses HiperSockets mit MTU size of 57344 (56K)
  - MTU size for guest and OVS vnic is 1500 or 56k
- **Results**
  - As long as the network package size is below the MTU size 1500 is the difference to the large MTU size is very small (mostly < 1%)
  - For large packages the MTU size of 56k provide a huge advantage, throughput (4x - 5x) at reduced cost (3x -4x)
- **Conclusion:**
  - For the usage with HiperSockets a large MTU size for guest and OVS is highly recommended

Guest to Guests Communication via OVS and *Routed OSA Card*: Scaling MTU size 56k vs 1500





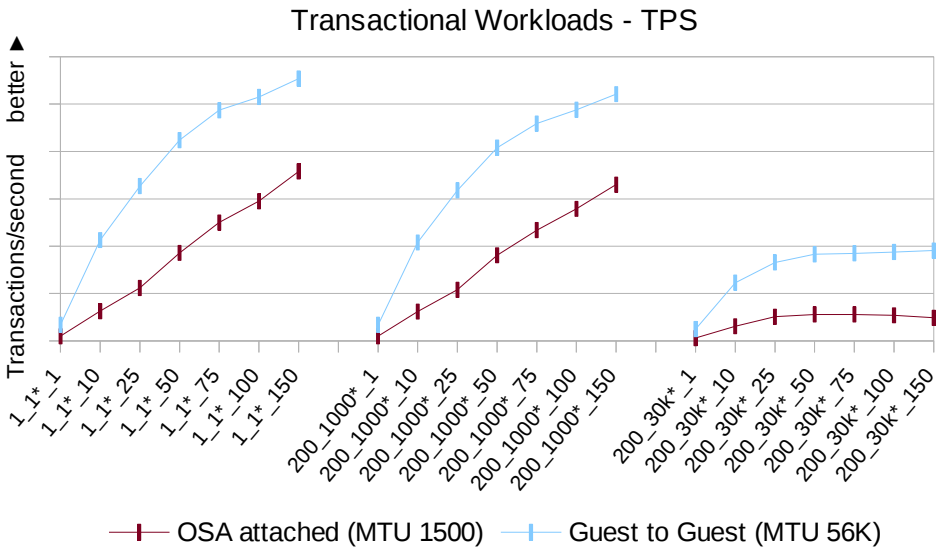
- **Has the MTU size of the guest and OVS environment an impact when routing via OSA card?**
  - the OSA card runs with a MTU size of 1500
  - MTU size for guest and OVS vnic is 1500 or 56k
- **Results**
  - For transactional workloads the impact is very small
  - For streaming workloads just the single connection has a little bit higher impact (throughput -4%, cost -3%)
    - the other scenarios are around -1.5%
- **Conclusion:**
  - There is nearly no overhead related when the guest and OVS always runs with the large MTU size
  - With a routed connection the IP stack from the KVM host handles the frame fragmentation if required

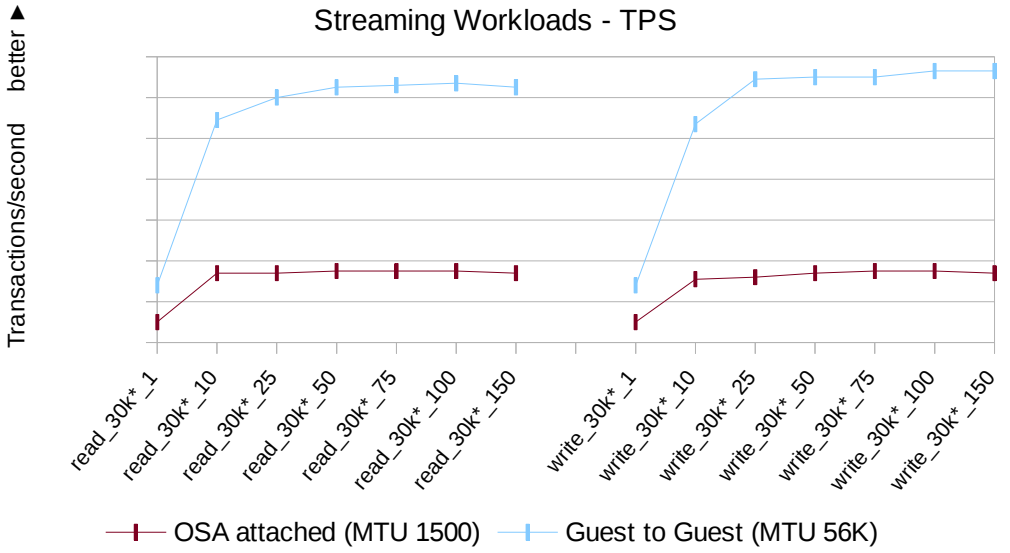
Guest to Guests Communication via OVS and *Attached OSA Card*: Scaling MTU size 56k vs 1500

- **Has the MTU size of the guest and OVS environment an impact with an attached OSA card?**
  - Yes, and a very bad one!
  - When the guest has a larger MTU size than the card, I/O errors will happen on the receiver side
    - When packages are larger than the MTU size from the OSA card!

- **Conclusion: When attaching an OSA card to the Open vSwitch**
  - Ensure that the MTU size of the guest is not larger that the MTU size of the OSA card!
    - That applies to a MacVTap connection also
  - When the OSA card runs with MTU size 8192 the guest could use the same size or MTU size 1500

Guest to Guests Communication local via OVS vs remote via OVS and Attached OSA Card





- **The connection between guest to guest via the OVS outperforms and the OVS with attached OSA card**
  - Shortest latency
  - Lower CPU cost
  - Applies to all workload pattern

Summary (1/2)

Recommendations

- The fastest connectivity tested here is guest to guest on the same host via OVS
  - Shortest pathlength, lowest interrupt overhead
- For external connections and performance and latency is the most important requirement use MacVTap
- For guests with high numbers of parallel connections is the Open vSwitch with attached OSA card recommended
  - Management of guest networking depends on the host network setup
- For the remaining scenarios is the Open vSwitch with routing a good approach
  - Easy to manage
    - This approach requires no external routing rules, but planning of the guest subnets
    - Management of guest networking is independent from the host network
  - But caution is needed
    - Use proxy_arp only in combination with arptables
  - Allows very flexible and complex network configuration
    - with OSA and HiperSockets
    - mixing of layer 2 and layer 3 devices
  - Overhead is moderate

Summary (2/2)

- **MTUs size for guest and Open vSwitch**
  - For the routing setup is the recommendation to use large MTU sizes
    - For routed OSA it doesn't matter
    - The large MTU size can become a large benefit
      - for HiperSockets
      - for local guest to guest communication
  - Ensure to have the MTU size from guest and the Open vSwitch interface consistent!

- **To make it even more powerful**
  - Using different subnets for the guests allows different routes for groups of guests
  - Transparent for the guests
  - All is managed by the host

- **The setup with KVM host routing is an enhancement to the global routing rules**
  - Subnet management is still required
  - The global routing rules still determine the overall routing behavior
  - Locally limited guest communication can be easily established without changing the global routing rules
    - The same effect can also be reached with global rules routing a certain guest subnet to a certain host
  - Usage is recommended for private IP address ranges which are isolated by switches/gateways

Questions?

- **Further information**
  - ProxyARP Subnetting HOWTO
    https://www.tldp.org/HOWTO/text/Proxy-ARP-Subnet
  - Proxy-ARP HOWTO
    http://leaf.sourceforge.net/doc/bk10pt01ar07.html
  - KVM Network Performance - Best Practices and Tuning Recommendations
    https://www.ibm.com/support/knowledgecenter/linuxonibm/liaag/wkvm/l0wkvm00_2016.htm
  - Network Storage Protocols in a KVM Environment - NFS/SMB/iSCSI Report
    https://www.ibm.com/support/knowledgecenter/linuxonibm/liaag/wnsp/l0wnsp00_2017.htm
  - Exploiting HiperSockets in a KVM Environment Using IP Routing with Linux on Z - Results and Findings
    https://www.ibm.com/support/knowledgecenter/linuxonibm/liaag/wehs/l0wehs00_2018.htm

- **Thanks for listening**

Dr. Juergen Doelle

Linux on System z
System Software
Performance Analyst

IBM Deutschland Research & Development
Schoenaicher Strasse 220
71032 Boeblingen, Germany

Backup

Used Schema for IP Addresses

- **Note: we use a schema to generate subnets and IP addresses for avoiding conflicts**

- **A LPAR has**
  - one OSA interface in an IP adress in the 10.x production net `10.131.<LPAR ID>.254`
    - /24 net with the LPAR ID, allows 253 additional interfaces (e.g. MacVTAP for guest starting with x.x.x.1)
    - /16 net for the CEC, all 10.131.x.x interfaces are host interfaces on that CEC
  - one OSA interface in the Intranet containing the LPAR ID `172.x.x.<LPAR ID>`
  - one HiperSockets interface (local addresses) `172.x.y.<1 or 2>`

- **The virtual OVS interface (vnic)**
  - the OVS interface is in the 10.x production network `10.136.<LPAR ID>.254`
    - /24 net with 253 addresses for guest interfaces
      - where .254 is always the OVS vnic
      - the guests start with x.1

- **The guests**
  - have one interface in the same /16 net as the OVS `10.136.<LPAR ID>.<1-253>`
    - that way we can immediately address 253 guests (.254 is used for the vnic of the OVS)