UNIVERSITY OF AMSTERDAM

FACULTY OF SCIENCE (FNWI)
MSc System and Network Engineering

Research Report

Using EVPN to minimize ARP traffic in an IXP environment

Stefan Plug <stefan.plug@os3.nl>
Lutz Engels <lutz.engels@os3.nl>

August 13, 2014

Supervisors
Kay Rechthien <kre@ecix.net>
Thorben Krüger <tkr@ecix.net>

University of Amsterdam
Faculty of Science (FNWI)
Science Park 904
1098XH Amsterdam

Abstract

The goal of this research project is to investigate whether *Ethernet Virtual Private Network* (EVPN), a promising *Internet Engineering Task Force* (IETF) draft, is a viable solution to the *Address Resolution Protocol* (ARP) problem as experienced by *Internet eXchange Points* (IXP) using a *Multi Protocol Label Switching* (MPLS) and *Virtual Private LAN Service* (VPLS) based peering network.

To accomplish this we designed a small scale IXP-like network integrating an Alcatel-Lucent *SROS-VM based Route Reflector* (vRR) *Virtual Machine* (VM) image, which contains an early implementation of EVPN. In three defined scenarios (*Scenario 1: VPLS, no ARP Sponge*, *Scenario 2: VPLS, ARP Sponge* and *Scenario 3: EVPN ARP proxy enabled*) we gathered data to provide the base for comparing the impact of EVPN's ARP proxy to that of a typical current IXP environment.

Due to the early stage of EVPN's implementation it was not yet possible to add MAC to IP information. This functionality is needed to enable *Provider Edge* (PE) devices to start ARP proxying. However, we did succeed in updating MAC to IP information by crafting our own *Multi Protocol BGP* (MP-BGP) update containing the required information.

As described in Chapter 4.4, EVPN can be used to eliminate ARP broadcasts on a IXP peering network using the fact that an IXP peering network only contains devices known to the administrator(s). This knowledge can be used to statically add MAC to IP information to the PE devices.

Future research is needed on the performance hit of EVPN ARP proxy on a PE device. The outcome of that research can tell us more about whether a PE device should ignore or respond in the case that it receives an ARP request for an unreachable known *Customer Edge* (CE) device, and an unknown CE device.

Contents

Glossary

brctl ethernet bridge administration tool. 11

collectd System statistics collection daemon. 16

Ostinato network packet crafter/traffic generator and analyzer. 15, 16, 22, 30

vanilla meaning uncustomized or unaltered. 5

vconfig VLAN (802.1q) configuration program. 11, 12

xte program that generates fake input. 16

Acronyms

AC Attachment Circuit. 4

AFI Address Family Identifier. 7

AMS-IX Amsterdam Internet eXchange. 1, 3, 28

ARP Address Resolution Protocol. i, 1, 3, 4, 6, 7, 9, 10, 15, 20, 21, 24–28, 30

ASIC Application-Specific Integrated Circuit. 4, 9

BIRD BIRD Internet Routing Daemon. 10

CE Customer Edge. i, 1, 3–10, 15, 20, 21, 23, 25–27

CLI Command Line Interface. 28

CPM Control Processor Module. 9

CPU Central Processing Unit. 1, 4, 9, 10, 16, 19–21, 27

CSV Comma-separated values. 16

ECIX European Commercial Internet eXchange. 1, 10, 28

ESI Ethernet Segment Identifier. 6, 7

ET Ethernet Tag. 6, 8

EVI EVPN Instance. 6–8

EVPN Ethernet Virtual Private Network. i, 1, 2, 4–10, 21, 23–28, 30

FCS Frame Check Sequence. 3, 5

GUI Graphical User Interface. 15, 16

ICMP Internet Control Message Protocol. 23

IEEE Institute of Electrical and Electronics Engineers. 5, 6, 8

IETF Internet Engineering Task Force. i, 1, 5

IP Internet Protocol. i, 1, 2, 4, 6–8, 10, 15, 21, 22, 24–27

ISP Internet Service Provider. 3

IXP Internet eXchange Point. i, 1–4, 6, 7, 9, 19, 24, 26–28

KVM Kernel-based Virtual Machine. 11, 12, 14

LDP Label Distribution Protocol. 3

LSP Label Switched Path. 3

MAC Media Access Control. i, 1, 3–8, 15, 21–27

MAC-VRF MAC Virtual Routing and Forwarding table. 6

MP-BGP Multi Protocol BGP. i, 1, 4–8, 22, 23, 25

MP_UNREACH_NLRI MP-BGP_Unreachability_Network Layer Reachability Information. 23, 25

MPLS Multi Protocol Label Switching. i, 1, 3, 5, 6, 8, 10, 20, 27

NDP Neighbour Discovery Protocol. 2

NLRI Network Layer Reachability Information. 7

P Provider. 10, 12

PC Provider Core. 10, 22

PE Provider Edge. i, 1, 3–7, 9, 10, 15, 19–27

PW PseudoWire. 3, 5

PXC Photonic Cross Connect. 27

RD Route Distinguisher. 7

RFC Request For Comments. 1, 3–5, 28

RS Route Server. 4

SAFI Subsequent Address Family Identifier. 7

TG Traffic Generator. 9, 12, 15, 19, 21

UDP User Datagram Protocol. 8, 10

VLAN Virtual LAN. 8, 10, 12

VM Virtual Machine. i, 8–10, 12, 14, 16, 19, 20, 24, 27

VNI VXLAN Network Identifier. 8

VPLS Virtual Private LAN Service. i, 1–5, 8, 10, 20, 26, 27

VPN Virtual Private Network. 3, 7

vRR SROS-VM based Route Reflector. i, 9–12, 14, 15, 17, 19–21, 26–28

VXLAN Virtual eXtensible LAN. 8–10, 23, 27

1 Introduction

1.1 Introduction

A typical *Internet eXchange Point* (IXP) provides access to a peering network. This peering network is usually set up in such a way as to be transparent and act as a single switch to the *Customer Edge* (CE) devices, although in reality the peering network can span multiple devices and can be distributed throughout multiple geographic locations.

Currently IXPs, such as the *European Commercial Internet eXchange* (ECIX) and the *Amsterdam Internet eXchange* (AMS-IX), implement a *Multi Protocol Label Switching* (MPLS) and *Virtual Private LAN Service* (VPLS) based peering network to provide their members with a single broadcast domain. VPLS succeeds in creating a transparent overlay broadcast domain, which also inherits some problems encountered in broadcast domains, such as indiscriminate *Address Resolution Protocol* (ARP) broadcasting. In larger peering networks ARP traffic has led to serious performance problems to CE devices [14], as ARP messages typically need to be processed by a device's CPU rather than being dealt with on a lower level. An OpenFlow based solution [1] has been proposed, as well as an ARP Sponge, which has been implemented in the AMS-IX network, where it reduced ARP traffic nearly tenfold [14].

Recently the *Internet Engineering Task Force* (IETF) published RFC 7209 [11] which describes the requirements for *Ethernet Virtual Private Network* (EVPN), which aims to address the "*new set of requirements that are not readily addressable by the current Virtual Private LAN Service (VPLS) solution*". Three interesting requirements in this document are described in Chapter 9 (Flood Suppression), these are:

1. "*(R11a) The solution SHOULD allow the network operator to choose whether unknown unicast frames are to be dropped or to be flooded. This attribute needs to be configurable on a per-service-instance basis.*"
2. "*(R11b) In addition, for the case where the solution is used for data-center interconnect, the solution SHOULD minimize the flooding of broadcast frames outside the confines of a given site. Of particular interest is periodic Address Resolution Protocol (ARP) traffic.*"
3. "*(R11c) Furthermore, the solution SHOULD eliminate any unnecessary flooding of unicast traffic upon topology changes, especially in the case of a multi-homed site where the PEs have a priori knowledge of the backup paths for a given MAC address.*"

In particular item 2, (R11b), specifically mentions that an EVPN solution should minimize ARP traffic.

At the time of writing this report the most recent IETF draft, which tries to meet the EVPN requirements, is draft-ietf-l2vpn-evpn-07 [10], which, among other things, proposes that the *Provider Edge* (PE) devices use *Multi Protocol BGP* (MP-BGP) to distribute IP to MAC information between each other and then act as ARP proxies to their CE devices.

The goal of this research project is to investigate whether EVPN is a viable solution to the ARP problem as experienced by IXPs. To limit the scope of this project we will

not look at how EVPN handles IPv6's *Neighbour Discovery Protocol* (NDP).

1.2 Research question

To help us research EVPN as stated in the previous section, we will endeavour to answer the following research question:

> *"How can EVPN supersede classical VPLS-based techniques in a typical IXP environment, particularly in terms of handling large broadcast domains?"*

2 Background

2.1 Currently used technologies

As already stated in the introduction (Chapter 1) an *Internet eXchange Point* (IXP) provides access to a peering network, which ideally acts as a single broadcast domain to the connected parties. Popular protocols among IXPs to achieve this goal are *Multi Protocol Label Switching* (MPLS) [8] combined with *Virtual Private LAN Service* (VPLS) [4]. To combat the large volume of *Address Resolution Protocol* (ARP) traffic, which was causing problems on the *Customer Edge* (CE) devices, the *Amsterdam Internet eXchange* (AMS-IX) developed and still uses an ARP Sponge [13].

2.1.1 MPLS

MPLS [8] is used to simplify the routing process of packets through a network. When a packet enters an MPLS network, only the ingress and egress devices will analyse the traditional layer 3 header to determine the packet's next hop. The intermediate MPLS devices will instead determine the packet's next hop through the MPLS network base on an MPLS label. This 20-bit MPLS label resides in a 32-bit MPLS header in between the layer 2 and layer 3 headers and is inserted here by the MPLS ingress device.

Multiple *Label Switched Paths* (LSP) can exist between the same MPLS ingress and egress devices, making it possible to load balance traffic or to do more sophisticated traffic engineering schemes, such as creating layer 3 MPLS *Virtual Private Networks* (VPN), which has made MPLS popular in modern *Internet Service Provider* (ISP) networks.

2.1.2 Pseudo Wires

An IXP peering network should ideally act as if it were just a single switch to its CE devices, as e.g. the CE devices should all belong to the same broadcast domain. The use of *PseudoWires* (PW) over MPLS was proposed in RFC3985 [9], RFC4447 [7] and RFC4448 [6] to create a point to point layer 2 VPN over an MPLS network. A PW creates a layer 2 VPN by encapsulating the original layer 2 frame, optionally without the original *Frame Check Sequence* (FCS) to save bandwidth, with an MPLS label. Note that this is different from vanilla MPLS described in RFC3031 [8] which instead encapsulates the layer 3 packet. When a traversing frame arrives at the end of the PW then the MPLS tag is stripped and a new FCS is generated for, and appended to, the original layer 2 frame, which is then sent on its way.

2.1.3 VPLS

RFC4762 [4] "*describes a Virtual Private LAN Service (VPLS) solution using pseudowires*". VPLS uses the *Label Distribution Protocol* (LDP) to create a full mesh of PWs between *Provider Edge* (PE) devices for each 'virtual' broadcast domain. Whenever a PE device receives a frame with an unknown source MAC address it will associate the new MAC address with the PW it received the frame on.

2.1.4 ARP flooding

An interesting behaviour in regards to our research is how VPLS behaves in the case of flooding. "*To achieve flooding within the service provider network, all unknown unicast, broadcast and multicast frames are flooded over the corresponding PWs to all PE nodes participating in the VPLS, as well as to all ACs [Attachment Circuits].*" [4]. This is good switching practise as just like in a regular broadcast domain this enables both the PE and CE devices to discover the broadcast domain's members.

As a broadcast domain becomes larger the amount of ARP traffic increases. Take for example an IXP peering network with 100 members where each member either has a direct *Multi Protocol BGP* (MP-BGP) peering with Member-1 or at least uses Member-1 as a next hop in one of their routes in the case that a *Route Server* (RS) is used. When Member-1 fails then the remaining 99 members will eventually timeout their ARP entry for Member-1 and will need to send a new ARP request. When a member gets no reply to its ARP request then it will periodically send a new request. Eventually there will be 99 members each sending out ARP request broadcasts to find Member-1. Because of the fact that member devices do not process ARP traffic in specialised *Application-Specific Integrated Circuits* (ASIC) as is usually the case for normal traffic, each member has to spend costly CPU cycles to process each ARP request messages, even if they are not intended for it. This can lead to serious performance problems in large broadcast domains [14].

2.1.5 ARP Sponge

An ARP Sponge listens for ARP request and keeps counters for all requested addresses. If an adjustable threshold of unanswered requests is registered, the ARP Sponge will generate a gratuitous ARP request using the requested host's IP address and its own MAC address. The requesting hosts will be satisfied and will stop sending ARP-requests. By doing this, the amount of unwanted ARP traffic has shown be to reduced tenfold [14].

2.2 EVPN

At the time of writing there are no ratified RFCs, which describe an *Ethernet Virtual Private Network* (EVPN) specification. However, RFC7209 [11] was recently ratified, which describes what requirements a future EVPN protocol should adhere to.

2.2.1 EVPN requirements

According to RFC7209 an EVPN solution should address some shortcomings in the current VPLS solution. Three shortcomings, which are specifically mentioned, are:

1. "*multihoming with all-active forwarding is not supported*"
2. "*no existing solution to leverage Multipoint-to-Multipoint (MP2MP) Label Switched Paths (LSPs) for optimizing the delivery of multi-destination frames*"
3. "*the provisioning of VPLS, even in the context of BGP-based auto-discovery, requires network operators to specify various network parameters on top of the access configuration*"

Although the promise for multihoming with all-active forwarding which would allow a CE device to load balance traffic between two or more PE devices might be interesting to IXPs, this research, however, will focus on the flood suppression requirements, namely:

1. "*(R11a) The solution SHOULD allow the network operator to choose whether unknown unicast frames are to be dropped or to be flooded. This attribute needs to be configurable on a per-service-instance basis.*"
2. "*(R11b) In addition, for the case where the solution is used for data-center interconnect, the solution SHOULD minimize the flooding of broadcast frames*

outside the confines of a given site. Of particular interest is periodic Address Resolution Protocol (ARP) traffic."

3. "*(R11c) Furthermore, the solution SHOULD eliminate any unnecessary flooding of unicast traffic upon topology changes, especially in the case of a multihomed site where the PEs have a priori knowledge of the backup paths for a given MAC address.*"

Several drafts have been proposed based on RFC7209. It is important to note that an *Internet Engineering Task Force* (IETF) draft is still work in progress. This means that this draft has not yet been ratified by the IETF [2]. For this research it is therefore important to be aware of the referenced draft's specific revision number. This also means that research, like ours, might help to improve future revisions.

2.2.2 BGP MPLS based EVPN

Draft-ietf-l2vpn-evpn-07 [10] was submitted on May 7th 2014 and describes a possible EVPN solution using MP-BGP and MPLS

Layer 2 tunnel

In our opinion the current draft is not clear enough on how the actual layer 2 frames, which are sent by a CE device, are transported by the PE devices. The main confusion is whether the MPLS labels are to be inserted before the layer 3 packet, or before the layer 2 frame. This confusion arises mainly due to the fact, that the draft uses the terms 'frame' and 'packet' indiscriminately, although both are data units associated with different layers. An example of this:

> [p.11] "*Ethernet frames transported over IP/MPLS*"
>
> [p.29] "*The advertising PE uses this label when it receives an MPLS-encapsulated packet to perform forwarding based on the destination MAC address toward the CE*"

This combined with the fact that vanilla MPLS [8] only mentions the encapsulation of layer 3 packets, and that the draft does not specifically mention that it differs from vanilla MPLS, makes this point rather ambiguous which could, and should, be avoided in a future IETF RFC.

We were in contact with with the authors of this draft and they told us that the layer 2 frame is indeed to be encapsulated. We are not sure whether, like in PWs, the original FCS is may be stripped before MPLS encapsulation.

In response to our questions in the IETF's Layer 2 VPN working group, we were assured that these matters would be cleared up in the next revision. However, the new revision will not be ready in time to be referred to in this report.

Local and remote CE MAC learning

EVPN distinguishes between how MAC addresses are learned depending on whether a CE device is either directly connected to a PE device, or whether the CE device is behind another PE device. When a CE device is directly connected to a PE, this is called *local learning* and can be accomplished using traditional *Institute of Electrical and Electronics Engineers* (IEEE) MAC learning techniques on the data plane. Alternatively PE devices may also use the control plane or management plane to learn the MAC addresses of its directly connected CE device.

When a CE device is connected to a remote PE device the original PE device needs to learn the CE device's MAC address via the remote PE device. This is called *remote learning*.

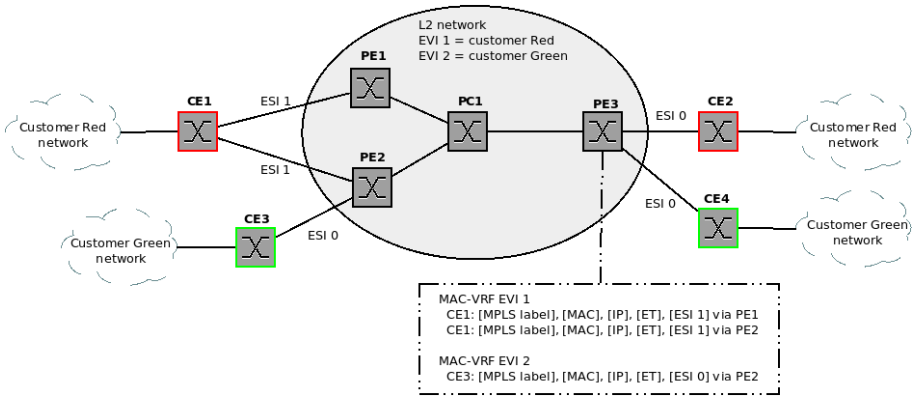
Contrary to standard IEEE Ethernet learning, which is done in normal switching as well as in VPLS, in EVPN the remote CE MAC addresses are not learned in the data plane. They are not learned through the data frames themselves, but rather in the control plane using MP-BGP. This allows network administrators to have greater control

over which MAC addresses a PE learns and can also lead to interesting situations such as the one described in Chapter 4.2.3.

EVI, MAC-VRF, ET, and ESI

A PE device can run multiple *EVPN Instances* (EVI) and an EVI can span multiple PE devices. Each PE device keeps a separate *MAC Virtual Routing and Forwarding table* (MAC-VRF) for each EVI. Each EVI consists of one or more broadcast domains, which are identified by an *Ethernet Tag* (ET). An *Ethernet Segment Identifier* (ESI) is used when a CE device is multi-homed to multiple PE devices and is used identify the set of Ethernet links used by a CE device. ESI 0 means that the CE device is single homed. Figure 2.1 shows how these terms look like in a hypothetical network.

Figure 2.1: EVPN terminology



An IXP would only use 1 EVI using only 1 ET on their PE devices if their goal is to create 1 unified peering network.

EVPN MPLS labels

When a MAC address is advertised between PE devices then the advertising, or 'downstream', PE will also send an MPLS label in the update, which will be associated to the advertised MAC address. The MPLS label to MAC address association can happen in one of the following three ways.

Per EVI label assignment Each MAC address in the same EVI uses the same MPLS label. This requires the PE device to do an additional look up to find the egress port for this particular MAC address. However, it does save on the amount of required labels.

Per <ESI, ET> label assignment Each ESI/ET couple receives a unique EVPN label.

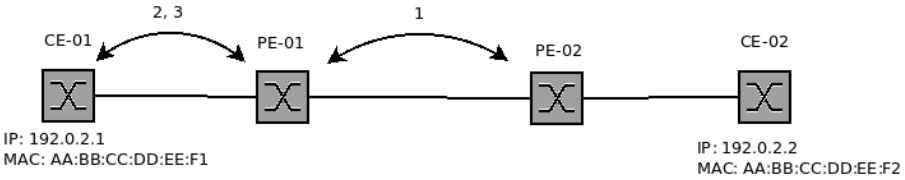
Per MAC label assignment Each MAC address is associated with a unique MPLS label. When a PE device receives a frame it only needs to look up the label to know which egress port to use.

ARP proxy

EVPN's key feature with respect to ARP flood protection is, that not only can the PE devices advertise CE MAC addresses, but optionally they can also send along the IP address which belongs to that MAC address. This information allows a PE device to respond to a CE device's ARP discovery and prevent the broadcast to spread itself on the network. In figure 2.2 we can see this scenario in three steps.

1. The PE devices use MP-BGP to advertise to each other the MAC and IP addresses of the CE devices attached to them. The PE devices can learn this information using standard IEEE Ethernet learning procedures, or alternatively through the management plane.
2. CE-01 sends out an ARP discovery broadcast for 192.0.2.2.

Figure 2.2: EVPN ARP Proxy



3. PE-01 knows the MAC address for 192.0.2.2 and responds to CE-01 without forwarding the original broadcast.

When in figure 2.2 CE-02 would fail, then PE-02 would eventually notice and send an update to PE-01 notifying it that CE-02 is no longer reachable. When CE-01 now sends an ARP message for CE-02 then PE-01 cannot answer, as it no longer has a MAC address for CE-02. In this case the ARP request would be flooded to all CE devices. Regarding our goal, namely to reduce ARP flooding in an IXP peering network, this is clearly unwanted behavior. Our recommendation for IXPs on the usage of the ARP proxy functionality can be found in Chapter 4.4.

EVPN MP-BGP MAC/IP advertisement

In regards to ARP EVPN's new MAC advertisement route the *Network Layer Reachability Information* (NLRI) is most interesting as this update can contain the relation between MAC and IP address needed for the PE devices to act as ARP proxies.

An EVPN MP-BGP reachability update uses *Address Family Identifier* (AFI) 25 (Layer-2 VPN), and *Subsequent Address Family Identifier* (SAFI) 70 (EVPN). An EVPN NLRI can contain one or more EVPN MP-BGP MAC/IP advertisement routes which are build up follows:

- Route Type (1 octet)
- Length (1 octet)
- RD (8 octets)
- Ethernet Segment Identifier (10 octets)
- Ethernet Tag ID (4 octets)
- MAC Address Length (1 octet)
- MAC Address (6 octets)
- IP Address Length (1 octet)
- IP Address (0 or 4 or 16 octets)
- MPLS Label1 (3 octets)
- MPLS Label2 (0 or 3 octets)

Route Type (1 octet) The following route type, in the case of a MAC/IP this field would contain 2.

All possible route types defined are:

1. Ethernet Auto-Discovery (A-D) route
2. MAC/IP advertisement route
3. Inclusive Multicast Ethernet Tag Route
4. Ethernet Segment Route

Length (1 octet) The length of the following update in octets.

RD (8 octets) The *Route Distinguisher* (RD) identifies the specific EVI, which this update is meant for.

Ethernet Segment Identifier (10 octets) The ESI is used in multi-homing CE devices to identify a set of Ethernet links. For our research we will only use ESI 0

which denotes a single-homed CE.

Ethernet Tag ID (4 octets) The ET identifier contains a 12-bit or 24-bit broadcast domain identifier e.g. a *Virtual LAN* (VLAN) tag.

MAC Address Length (1 octet) The subsequent MAC address length in bits. This draft tells us that this field is *always* set to 48.

MAC Address (6 octets) The MAC address length and type are hard coded in the draft for now, but it is obvious that this structure allows for easy updates to include other layer 2 address types in the future.

IP Address Length (1 octet) Subsequent IP address length in bits. Please note that when we look at the octet sizes for the IP address field there are just 3 choices which means that this field can only contain 0 for none, 32 for IPv4 or 128 for IPv6.

IP Address (0, 4 or 16 octets) An IPv4 or IPv6 address. Again note that this field has three pre-defined sizes, but that this can easily be updated for other layer 3 address types in the future.

MPLS Label1 (3 octets) The MPLS label which should be used by the receiver as discussed in Chapter 2.2.2.

MPLS Label2 (0 or 3 octets) Subsequent MPLS labels are optional.

2.3 EVPN over VXLAN

As explained in Chapter 3.2 we were provided by Alcatel-Lucent *Virtual Machines* (VM) which implement EVPN over *Virtual eXtensible LAN* (VXLAN). VXLAN uses a VXLAN header and *User Datagram Protocol* (UDP) to create, much like VPLS, an overlay network. According to the most recent draft [5] "*It [VXLAN] runs over the existing networking infrastructure and provides a means to 'stretch' a Layer 2 network. In short, VXLAN is a Layer 2 overlay scheme over a Layer 3 network.*".

VXLAN achieves this by encapsulating a frame in transit with a VXLAN, UDP, IP, and Ethernet header. The VXLAN header contains a 24-bit *VXLAN Network Identifier* (VNI), which acts much like an IEEE 802.1q [3] VLAN tag to create potentially 2^{24} separated Layer 2 networks over a Layer 3 network.

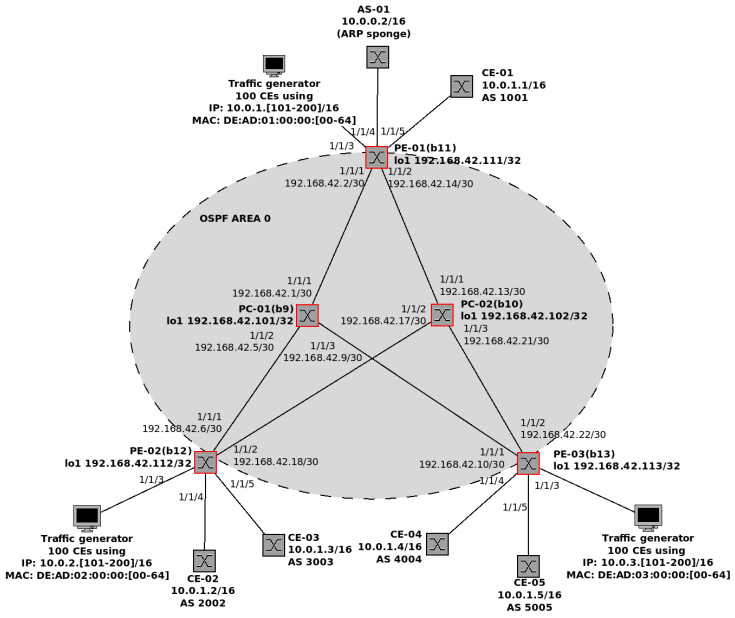
When VXLAN is used as an overlay for EVPN then the VXLAN VNI directly maps to the EVPN EVI [12]. The MP-BGP MAC update will still contain an 'MPLS Label1' field, but this will field will be used to carry the VNI instead.

3 Test Setup

3.1 Logical network design

To help us test *Ethernet Virtual Private Network* (EVPN)'s functionality we designed a small-scale *Internet eXchange Point* (IXP)-like network (see Figure 3.1) consisting of a two-node core (PC-01 and PC-02) and three *Provider Edge* (PE) devices. PE-01 has one *Customer Edge* (CE) device, a device running an *Address Resolution Protocol* (ARP) Sponge, and a *Traffic Generator* (TG) connected to it. PE-02 and PE-03 each have two CE devices and a TG attached to them.

Figure 3.1: Logical network design



3.2 Router VMs

Alcatel-Lucent kindly provided us with a *Virtual Machine* (VM)-image of the of the software (i386 version) running on their model 7750 Service Router *Control Processor Module* (CPM) and 5 matching license-keys. The internal-only pre-release version provided was: TiMOS-B-12.0.R2. These VMs contain a *Virtual eXtensible LAN* (VXLAN)-based [5] EVPN-implementation [12]. We will henceforth refer to the router VMs as *SROS-VM based Route Reflectors* (vRR).

3.2.1 Performance considerations

Virtual Machines

Router VMs process traffic solely in their virtual CPUs. Hardware routers on the other hand might utilize *Application-Specific Integrated Circuits* (ASIC) for forwarding, which enables them to switch at line speed, which does not show in CPU usage. A

direct relation of performance between the two is therefore not possible. However, comparing *Virtual Private LAN Service* (VPLS) and the EVPN ARP proxy within the same VM might show interesting differences in CPU usage. We assume that in hardware implementations ARP proxying will be done in the CPU, as the concerned device needs to analyse incoming ARP messages and process them accordingly. We asked Alcatel-Lucent for an official statement regarding the performance and use case of VMs. Their statement can be found in Chapter 4.3.

VXLAN

Initially we assumed to be working with a *Multi Protocol Label Switching* (MPLS)-based implementation, but during the course of the project we learned that Alcatel-Lucent's implementation is VXLAN-based. As discussed in Section 2.3 VXLAN encapsulates every frame with *User Datagram Protocol* (UDP), IP and Ethernet headers. This, as compared to VPLS, where only the Layer 3 packet and its data is encapsulated, might result in higher CPU usage, because of the extra overhead. We think that we can still meaningfully compare EVPN's impact on the PE devices during an ARP storm as we expect the ARP messages not to traverse the network. This is because the EVPN ARP proxy feature should ensure that the ARP messages are handled locally by the ingress PE devices. Therefore the ARP message will not be encapsulated by VXLAN.

3.3 Physical network design

The fact that we are working with VMs and not real hardware made it more complicated to set up the actual network then Figure 3.1 would suggest.

The physical network, which also contains the two physical servers, which act as VM hosts (sros1 and sros2), is shown in Figure 3.2. The physical design was built around the idea that the VMs should all be agnostic to the physical network between the two VM hosts. This is accomplished by connecting the CE devices to the PE devices through separate *Virtual LANs* (VLAN). The VMs themselves are not aware of the VLANs. The VM-hosts physical interconnection carries VLAN-tagged traffic. However the VLAN-IDs are popped by the VM hosts ensuring that the VMs themselves receive untagged traffic. The connection from the PE devices to the core is established by means of using bridges (e.g. br-PE01-PC01) that have only two interfaces, one facing the PE, the other facing the *Provider Core* (PC)-node.

All *Provider* (P) devices in Figure 3.2 are running Alcatel-Lucent's vRR. The CEs are running Ubuntu 14.04 server with running *BIRD Internet Routing Daemon* (BIRD) instances who have BGP neighbourships between each other.

3.3.1 Hardware

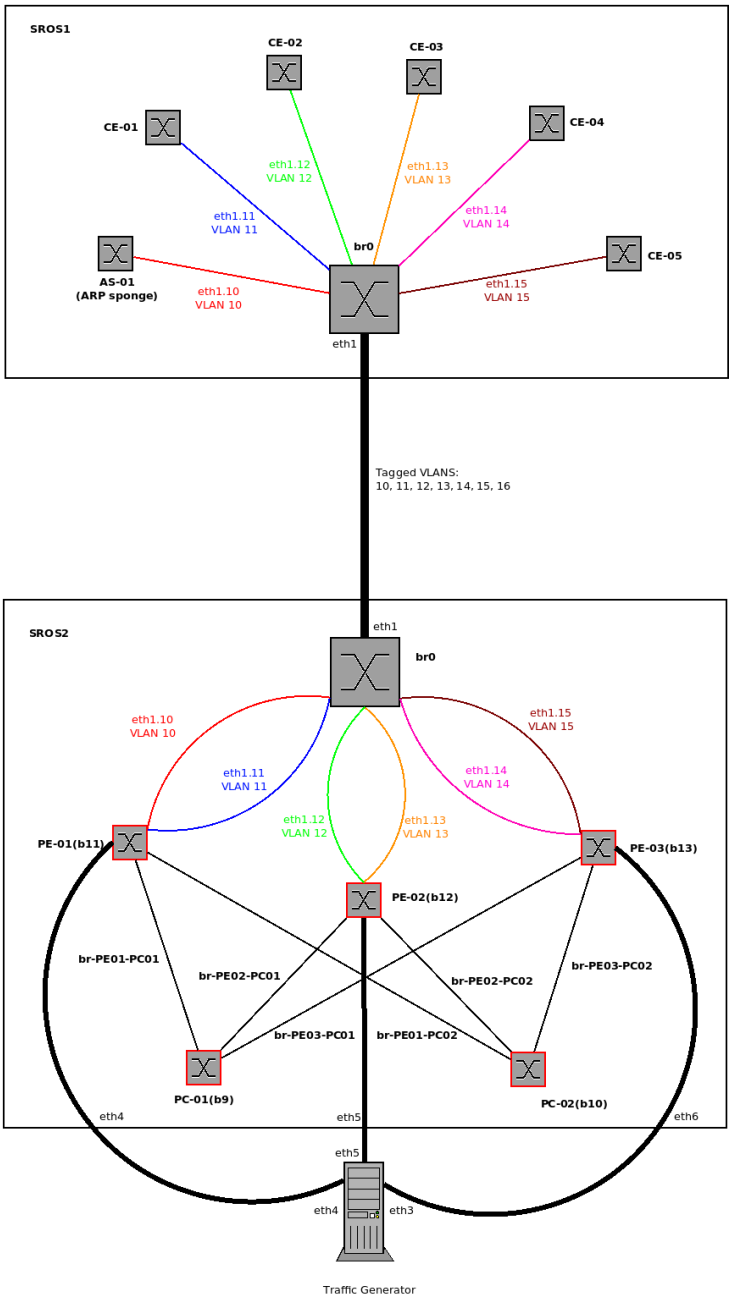
The *European Commercial Internet eXchange* (ECIX) provided two hardware servers with each two six-core Intel(R) Xeon(R) CPU E5-2640 CPUs for the experiments. The first one (*sros1*), with 64 GiB of RAM, hosts the CEs. The second one (*sros2*), with 128 GiB of RAM, hosts the P devices.

Listing 3.1: sros 1 processor and memory information

```
$ egrep "(model name|physical id)" /proc/cpuinfo | sort | uniq -c
     24 model name  : Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz
     12 physical id : 0
     12 physical id : 1

$ head -1 /proc/meminfo
MemTotal:       65946304 kB
```

Figure 3.2: Physical network design



Listing 3.2: sros 2 processor and memory information

```
$ egrep "(model name|physical id)" /proc/cpuinfo | sort | uniq -c
     24 model name  : Intel(R) Xeon(R) CPU E5-2640 0 @ 2.50GHz
     12 physical id : 0
     12 physical id : 1

$ head -1 /proc/meminfo
MemTotal:       132133512 kB
```

3.3.2 Software

For the vRRs to run as expected Alcatel-Lucent requires a CentOS 6.4, *Kernel-based Virtual Machine* (KVM) based setup. As the physical servers used have public facing interfaces, for security considerations an all up-to-date CentOS 6.5 installation with the KVM-package in the latest available version was chosen.

The network interconnections, as displayed in the layouts in section 3.1, are realized by the use of vconfig and brctl.

Interface eth1 on both physical servers was subdivided into 6 VLAN-carrying sub-

interfaces with vconfig (see Listing 3.3). In Figure 3.3 and Figure 3.4 the VLAN-configuration is indicated in the rectangles labeled *vconfig*. The traffic is tagged and untagged at the sub-interfaces. As a result the VMs will not receive tagged traffic.

As depicted in Figure 3.3, each VM (kvm rectangle) on *sros1* has two interfaces. All VM eth0 interfaces are connected through a *macvtap* interface to the aforementioned VLAN sub-interfaces. This is how KVM 'directly' attaches VM interfaces to already existing host interfaces. The VM's eth1 interfaces are connected through virtual interfaces, that are created on-the-fly upon boot of the respective VM, to the management bridge *virbr0*. Listing 3.4 shows the (virtual) interfaces attached to *virbr0*.

The interconnections of *sros2*, as shown in Figure 3.4 and Listing 3.5, are slightly more complex, as the vRR based VMs have four to six interfaces. Their *A/1* interface represent their management interface, which is connected through an on-the-fly interface to *br-managment*. This bridge in turn is connected to the host's eth7, which is cable connected to eth7 on *sros1*. The license key files necessary to boot the vRR based VMs are provided by ftp from there. The other VM interfaces are either connected through a bridge the other P devices, e.g. PE-01 and PC-01 are connected through br-PE01-PC01, to the VLAN sub-interfaces (marked in blue) or to the host interfaces connecting to the TG (marked in red).

Traffic forwarding needed to specifically be allowed in the firewall configuration. An example of the rules to accomplish this for the bridge *br-management* is shown in Listing 3.6. The rules allow all traffic that both comes from and goes to the bridge interface, but rejects traffic entering or leaving the bridge.

Listing 3.3: VLAN subinterfaces on sros1 and sros2

```
$ sudo cat /proc/net/vlan/config
VLAN Dev name  | VLAN ID
Name-Type: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
eth1.10        | 10  | eth1
eth1.11        | 11  | eth1
eth1.12        | 12  | eth1
eth1.13        | 13  | eth1
eth1.14        | 14  | eth1
eth1.15        | 15  | eth1
```

Listing 3.4: sros1 bridge configuration

```
$ brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.525400a1df19       no              as-01_eth1
                                                        ce-01_eth1
                                                        ce-02_eth1
                                                        ce-03_eth1
                                                        ce-04_eth1
                                                        ce-05_eth1
```

Listing 3.5: sros2 bridge configuration

```
$ brctl show
bridge name     bridge id               STP enabled     interfaces
br-PE01-PC01    8000.525400a1df17       no              pc-01_111
                                                        pe-01_111
br-PE01-PC02    8000.525400a1df1a       no              pc-02_111
                                                        pe-01_112
br-PE02-PC01    8000.525400a1df18       no              pc-01_112
                                                        pe-02_111
br-PE02-PC02    8000.525400a1df1c       no              pc-02_112
                                                        pe-02_112
br-PE03-PC01    8000.525400a1df19       no              pc-01_113
                                                        pe-03_111
br-PE03-PC02    8000.525400a1df1b       no              pc-02_113
                                                        pe-03_112
br-management   8000.40f2e90d2b97       no              eth7
                                                        pc-01_a1
                                                        pc-02_a1
                                                        pe-01_a1
                                                        pe-02_a1
                                                        pe-03_a1
```

Figure 3.3: The (virtual) interface interconnections of sros1

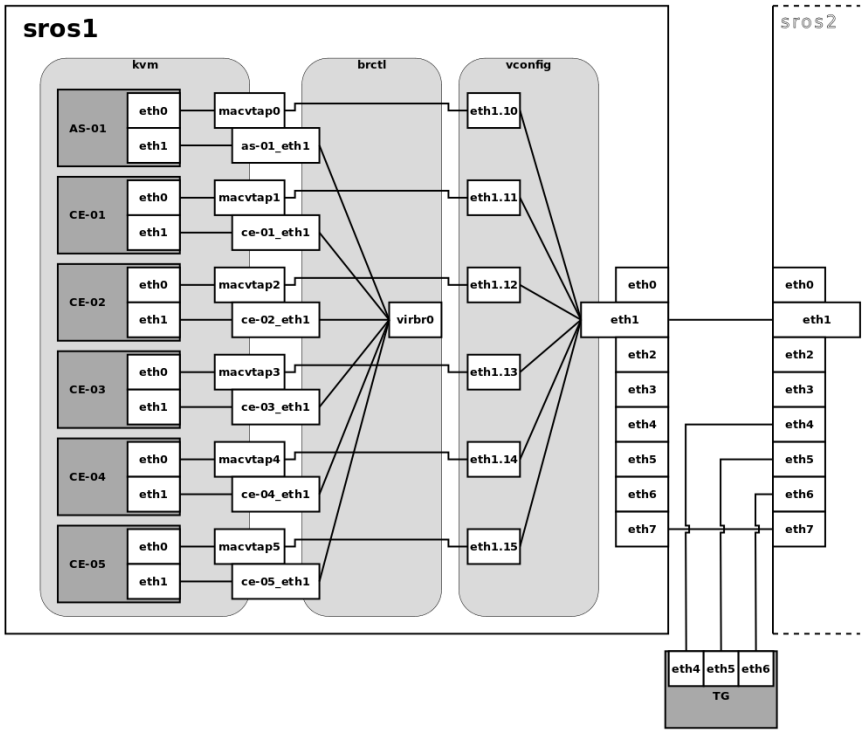
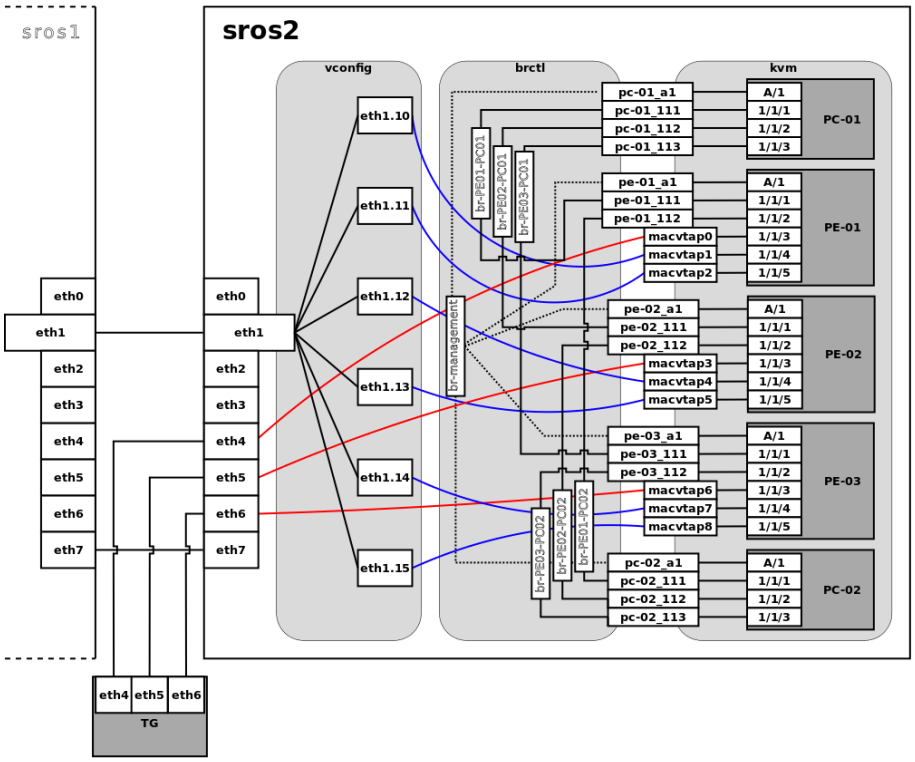


Figure 3.4: The (virtual) interface interconnections of sros2



Listing 3.6: Example iptables rules to allow traffic going back and forth through a bridge.

```
$ sudo grep "br-management" /etc/sysconfig/iptables
-A FORWARD -i br-management -o br-management -j ACCEPT
-A FORWARD -o br-management -j REJECT --reject-with icmp-port-unreachable
-A FORWARD -i br-management -j REJECT --reject-with icmp-port-unreachable
```

VM Snapshots

A set of scripts was developed to take snapshots of VMs and their configuration and to restore them. We refrained from using KVM's built-in snapshot feature, as especially during the setup-phase of the vRR VMs XML configuration file parameters needed to be changed one by one. We wanted to make sure that a particular setup could be saved and recreated at any time.

control_all_vrrs.sh (see Listing 3.7) Start, stop or view the current status of all vRRs. This script is also called upon by the the other two scripts.

snapshot.sh (see Listing 3.8) Takes a snapshot of all vRRs by copying all virtual disk images and XML config files to a specified, timestamped folder.

restore_latest.sh (see Listing 3.9) Restores the latest snapshot of all vRRs by taking a snapshot of the current situation, stopping all vRRs, copying back the disk images and XML configuration files from the directory with the latest timestamp and finally starting all vRRs.

Listing 3.7: control_all_vrrs.sh on sros1: Start/stop all vRR at the same time

```
#!/bin/bash
# Control all vRRs at the same time. As of now only destroy and (re)-create
# author: Lutz Engels

# Error handling
error_exit() {
  echo "ERROR: $1"
  exit 1
}

# Variables
CONFDIR="/etc/libvirt/qemu"
CONFIGS="$(sudo sh -c "ls $CONFDIR/P*.xml")"
SLEEP="sleep 0.3"

case "$1" in
    start)
  for i in $CONFIGS
        do
           sudo virsh create $i
           $SLEEP
        done
        ;;

    stop)
  for i in $CONFIGS
  do
            instance="$(basename $i)"
            sudo virsh destroy ${instance%.*};
            $SLEEP
        done
        ;;

    status)
        sudo virsh list
      exit 0
        ;;

    *)
    error_exit "Useage: $0 {start|stop|status}"
esac

echo "Done."
```

Listing 3.8: snapshot.sh on sros1: Takes a snapshot of all vRRs

```
#!/bin/bash
# Take snapshot of PE, PC domains
# author: Lutz Engels

# Error handling
error_exit() {
  echo "ERROR: $1"
  exit 1
}

# Variables
NOW=$(date +"%d_%m_%Y_%H_%M")
IMAGES="/var/lib/libvirt/images"
CONFIGS="/etc/libvirt/qemu"
SNAPDIR="$IMAGES/snapshot_$1_$NOW"

# only one parameter
[[ $# -ne 1 ]]                  && error_exit "usage: $0 <name-of-snapshot>"
[[ -d $SNAPDIR ]]   && error_exit "$SNAPDIR exist. Please choose different name,
     e.g. $1_1."

echo "Creating $SNAPDIR ..."
sudo mkdir $SNAPDIR
echo "Copying VM-disks to $SNAPDIR ..."
sudo cp -a $IMAGES/*.qcow2 $SNAPDIR
echo "Copying .XML config files to $SNAPDIR ..."
# Expanding wildcard
sudo cp -a $(sudo sh -c "ls $CONFIGS/P*.xml" ) $SNAPDIR
echo
echo "ls -als $SNAPDIR"
ls -als $SNAPDIR
echo
echo "Done."
```

3.4 Generating ARP request

A TG, as indicated at the bottom of Figure 3.3 and 3.4, is used to provide the ARP traffic that would appear in a real IXP network, when a PE device goes down. As can be seen in Figure 3.4 we connected interfaces eth4, eth5, and eth6 to PE-01, PE-02, and PE-03 respectively.

We found that a Ubuntu 14.04 laptop running the GUI tool Ostinato was able to perform this task.

Ostinato enabled us to craft our own ARP requests with custom source MAC addresses asking for specific IP addresses.

3.4.1 Scenarios

Making a reasoned statement about how EVPN reduces the ARP traffic in the environment described in the previous sections, a means of comparison is needed. We decided to simulate a broadcast domain with 400 CE devices where 100 CE devices fail. This would result in 300 CE devices which each send out 100 ARP requests per second.

Because we could not let the TG stop sending out ARP requests when it received an ARP reply we had to create separate tests for the three scenarios described below. Each test has a set of Ostinato streams which can all be found in `http://rp.delaat.net/2013-2014/p31/measurements.zip` → *./data/ostinato/*.

Scenario 1: VPLS, no ARP Sponge In this scenario we simulate what would happen when no measures are taken to reduce unwanted ARP traffic. Here we let Ostinato send a set of ARP requests after which it waits for 0.5 seconds. This set is then looped indefinitely. An example of this test is → *./data/ostinato/DEAD_source_PE1_100*.

Scenario 2: VPLS, ARP Sponge As explained in Section 3.4, we could not script against Ostinato. Ostinato therefore can only send out traffic, but not react to the ARP-sponge sending out gratuitous ARP requests. We still wanted to do this test to verify if and how the ARP Sponge would react. This scenario loops

through the set of rules exactly 3 times to simulate the thresholds of the ARP-sponge (900 unique ARP-requests) being reached and the simulated querying hosts backing off. An example of this test can be found in → *./data/ostinato/DEAD_source_PE1_100_SPONS*.

Scenario 3: EVPN ARP proxy enabled The EVPN ARP proxy feature is enabled. Ostinato runs through the same rules as in scenario 1, but stops after the first run. An example of this test is → *./data/ostinato/DEAD_source_PE1_100_EVPN*.

Unfortunately it is not possible to script the creation of Ostinato rules, which means that every rule needed to be inserted through the GUI manually. We could partly automate the process by the use of xte.

3.5 Data collection

Collectd with the libvirt-plugin was instrumented to measure the performance of all VMs. Both sros1 and sros2 run an instance of collectd that collects disk, interface and cpu statistics per second and stores them in *Comma-separated values* (CSV) files.

The configuration of collectd is shown in Listing 3.10 and that of the libvirt-plugin on sros1 and sros2 in Listing 3.11 and Listing 3.12 respectively. When data was collected we used GNUplot to generate graphs.

Next to the VM CPU and memory statistics, the actual data sent over the network is very important to record. We used TCPdump to capture all packets on all interfaces on all devices for all our tests.

Because of the large number of actions to take for each test we decided to automate the process. The Bash script we built to start a test is shown in Listing 3.13. The results off all our tests are published at `http://rp.delaat.net/2013-2014/p31/measurements.zip`.

Listing 3.9: restore_latest.sh on sros1: Restores the latest snapshot of all vRRs

```
#!/bin/bash
# Restore the latest snapshot of PE, PC domains
# author: Lutz Engels

# Error handling
error_exit() {
  echo "ERROR: $1"
  exit 1
}

# only one parameter
[[ $# -ne 1 ]]                  && error_exit "usage: $0 <name-of-snapshot>"

# Variables
NOW=$(date +"%d_%m_%Y_%H_%M")
IMAGES="/var/lib/libvirt/images"
BACKUP_IMAGES="$IMAGES/backup_current_$NOW"
CONFIGS="/etc/libvirt/qemu"
BACKUP_CONFIGS="$CONFIGS/backup_current_$NOW"
LATEST="$(ls -d1 $IMAGES/snapshot_$1_* 2>/dev/null | tail -1)"

# Check if there is something to restore
[[ -z $LATEST ]]    && error_exit "There is no snapshot containing \"$1\".
     Please check content of $IMAGES manually."

echo "Replace current setup with $LATEST? (yes/no)"
read yesorno
[[ $yesorno == "yes" ]]   || error_exit "Aborted."
yesorno=""

echo
echo "Taking snapshot of current situation."
echo "-------------------------------------"
./snapshot.sh current
sudo sh -c "echo \"This snapshot was taken before restoring $LATEST.\" > $(ls -
     d1 $IMAGES/snapshot_current_* 2>/dev/null | tail -1)/README.txt"
echo "-------------------------------------"

echo
echo "Restoring $LATEST ..."
echo "-------------------------------------"
echo "Restoring VM-disks ..."
sudo cp -af $LATEST/*.qcow2 $IMAGES
echo "Restoring XML configs ..."
sudo cp -af $LATEST/P*.xml $CONFIGS
echo
echo "Done."
echo "-------------------------------------"

echo "Restart all VMs with restored settings? (yes/no)"
read yesorno
echo "-------------------------------------"
if [ $yesorno == "yes" ]
then
    ./control_all_vrrs.sh stop
    echo
    ./control_all_vrrs.sh start
else
    echo "VMs not restarted."
fi
echo "-------------------------------------"
echo
echo Done.
```

Listing 3.10: main collectd configuration file /etc/collectd.conf

```
$ sudo egrep "(^[^#]|^\s)" /etc/collectd.conf
Interval      1
LoadPlugin logfile
<Plugin logfile>
        LogLevel debug
        Timestamp true
</Plugin>
LoadPlugin csv
Include "/etc/collectd.d"
```

Listing 3.11: Configuration file for the libvirt-plugin of collectd on sros1

```
$ sudo egrep -v "^\s+#" /etc/collectd.d/libvirt.conf
LoadPlugin libvirt
<Plugin "libvirt">
  RefreshInterval 1
  Domain "AS-01"
  Domain "CE-01"
  Domain "CE-02"
  Domain "CE-03"
  Domain "CE-04"
  Domain "CE-05"
  HostnameFormat "name"
</Plugin>
```

Listing 3.12: Configuration file for the libvirt-plugin of collectd on sros2

```
$ sudo egrep -v "^\s+#" /etc/collectd.d/libvirt.conf
LoadPlugin libvirt
<Plugin "libvirt">
  RefreshInterval 0.1
  Domain "PE-01"
  Domain "PE-02"
  Domain "PE-03"
  Domain "PC-01"
  Domain "PC-02"
  HostnameFormat "name"
</Plugin>
```

Listing 3.13: STARTTEST.sh on sros1 which starts the ARPsponge, TCPdumps, collectd, sleeps 16 seconds, and finally generates the graphs

```
#!/bin/bash
# author Stefan Plug

ssh AS-01 'nice --10 arpsponge --daemon --statusfile=/tmp/sponge.out --
     gratuitous --queuedepth 600 10.0.4.100/30 dev eth0'
sleep 6

ssh sros2 'service collectd start; mkdir -p data/data; nohup tcpdump -w data/
     data/tg-01_pe-01.pcap -i eth4 -n </dev/null >nohup.out 2>&1 & nohup tcpdump
      -w data/data/tg-02_pe-02.pcap -i eth5 -n </dev/null >nohup.out 2>&1 &
     nohup tcpdump -w data/data/tg-03_pe-03.pcap -i eth3 -n </dev/null >nohup.
     out 2>&1 & nohup tcpdump -w data/data/pe-01_pc-01.pcap -i br-PE01-PC01 -n
     </dev/null >nohup.out 2>&1 & nohup tcpdump -w data/data/pe-02_pc-01.pcap -i
      br-PE02-PC01 -n </dev/null >nohup.out 2>&1 & nohup tcpdump -w data/data/pe
     -03_pc-01.pcap -i br-PE03-PC01 -n </dev/null >nohup.out 2>&1 & nohup
     tcpdump -w data/data/pe-01_pc-02.pcap -i br-PE01-PC02 -n </dev/null >nohup.
     out 2>&1 & nohup tcpdump -w data/data/pe-02_pc-02.pcap -i br-PE02-PC02 -n
     </dev/null >nohup.out 2>&1 & nohup tcpdump -w data/data/pe-03_pc-02.pcap -i
      br-PE03-PC02 -n </dev/null >nohup.out 2>&1 &'


service collectd start
mkdir -p data
tcpdump -w data/as-01_eth0.pcap -i eth1.10 -n &
tcpdump -w data/ce-01_eth0.pcap -i eth1.11 -n &
tcpdump -w data/ce-02_eth0.pcap -i eth1.12 -n &
tcpdump -w data/ce-03_eth0.pcap -i eth1.13 -n &
tcpdump -w data/ce-04_eth0.pcap -i eth1.14 -n &
tcpdump -w data/ce-05_eth0.pcap -i eth1.15 -n &


sleep 16

ssh sros2 'service collectd stop; pkill tcpdump'
service collectd stop
pkill tcpdump

ssh AS-01 'pkill -USR1 arpsponge'
sleep 5

./virt_analyse_data.sh && ssh sros2 'cd data; ./virt_analyse_data.sh'
scp -r sros2:data/data/* data/
ssh sros2 'cd data; rm -fr data'

# Get ARP-sponge data
ssh AS-01 'service arpsponge stop'
scp root@AS-01:/tmp/sponge.out data/

now=$(date +"%d_%m_%Y_%H_%M")
mv /root/data/data /root/data/data_$now
```

4 Results

4.1 License problems

During preliminary testing the *Traffic Generator* (TG) was directly attached to two *Provider Edge* (PE) device interfaces, basically testing the *SROS-VM based Route Reflectors* (vRR) switching capabilities. While doing so we detected a high frame drop rate at the PE device. Figure 4.1 shows that at 10 Mbps, which is not a high rate considering an *Internet eXchange Point* (IXP) network, only 800 out of 10000 frames are received. For comparison this test was repeated with an Ubuntu server based *Virtual Machine* (VM), where all 10000 frames were received correctly.

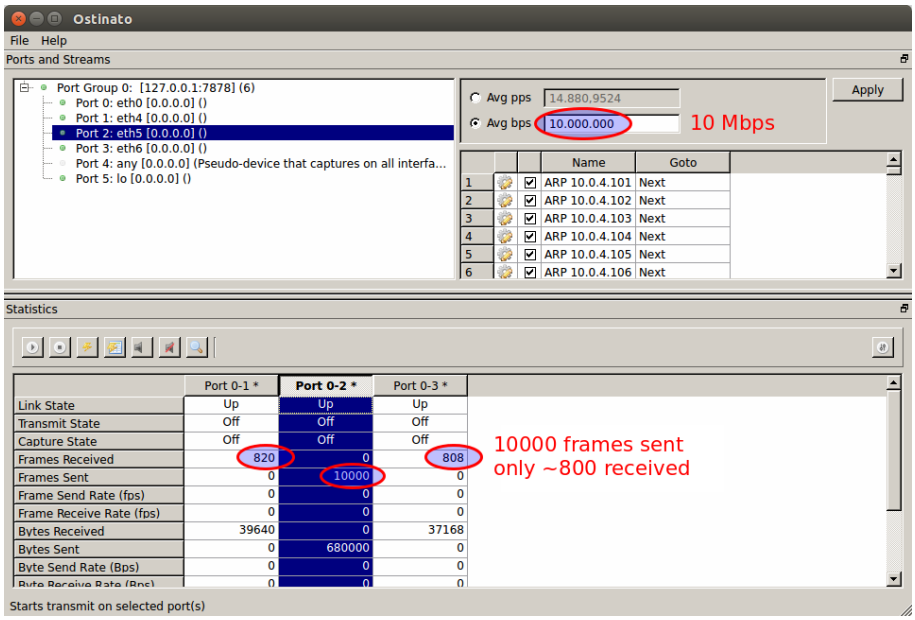


Figure 4.1: High frame drop rate at 10 Mbps

Inquiring with Alcatel-Lucent gained the following statement:

> *"[My colleague] was so kind to check with our R&D. It is in fact a result of you using a genuine vSIM license, which is intentionally limited to low data plane throughput [...] and is primarily targeted at simple lab and (self-)educational use."*

To take this fact into account and still get meaningful data, the TG rate was lowered until no more frame drop occurred. This happened to be 100 kbps.

Initially all VMs were configured to use multiple (virtualized) CPUs to meet Alcatel-Lucent's requirements. However, as we want to mainly compare the CPU usage, on such a low traffic rate measures needed to be taken to make the CPU usage visible at all. Therefore all VMs were restricted to *one* (virtualized) CPU (see Listing 4.1).

Listing 4.1: The VMs are restricted to one VCPU

```
$ sudo grep cpuset /etc/libvirt/qemu/P*.xml
/etc/libvirt/qemu/PC-01.xml:  <vcpu current='1' cpuset='0'>1</vcpu>
/etc/libvirt/qemu/PC-02.xml:  <vcpu current='1' cpuset='6'>1</vcpu>
/etc/libvirt/qemu/PE-01.xml:  <vcpu current='1' cpuset='12'>1</vcpu>
/etc/libvirt/qemu/PE-02.xml:  <vcpu current='1' cpuset='18'>1</vcpu>
/etc/libvirt/qemu/PE-03.xml:  <vcpu current='1' cpuset='23'>1</vcpu>

$ sudo grep cpuset /etc/libvirt/qemu/*-0*.xml
/etc/libvirt/qemu/AS-01.xml:  <vcpu current='1' cpuset='11'>1</vcpu>
/etc/libvirt/qemu/CE-01.xml:  <vcpu current='1' cpuset='0'>1</vcpu>
/etc/libvirt/qemu/CE-02.xml:  <vcpu current='1' cpuset='6'>1</vcpu>
/etc/libvirt/qemu/CE-03.xml:  <vcpu current='1' cpuset='12'>1</vcpu>
/etc/libvirt/qemu/CE-04.xml:  <vcpu current='1' cpuset='18'>1</vcpu>
/etc/libvirt/qemu/CE-05.xml:  <vcpu current='1' cpuset='23'>1</vcpu>
```

4.2 Scenarios

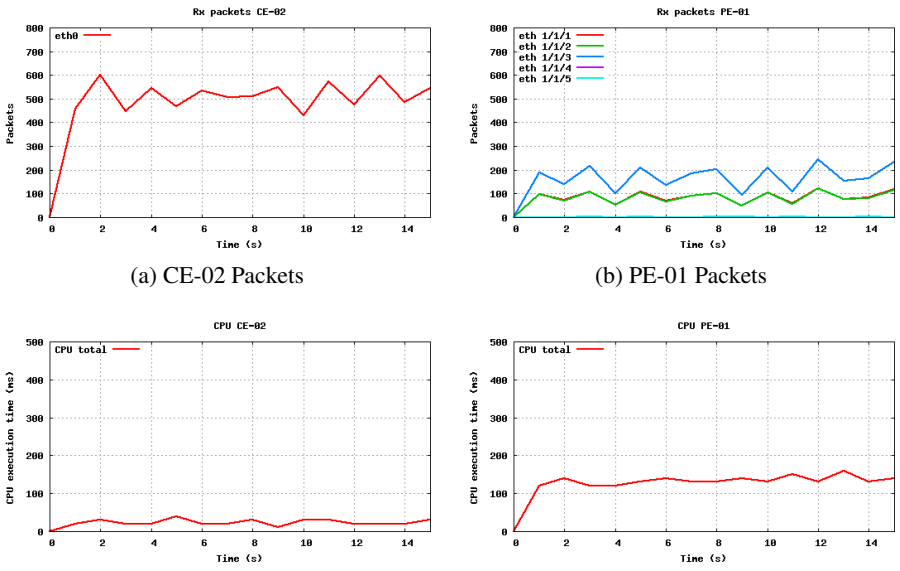
Once the variables were tuned according to the gained knowledge on the vRR limitations, the test scenarios could successively be performed.

4.2.1 Scenario 1: VPLS, no ARP Sponge

As mentioned in Section 3.4.1 a base-line measurement was taken to see how the setup behaves in case of the usual *Multi Protocol Label Switching* (MPLS)/*Virtual Private LAN Service* (VPLS) combination, with no means of reducing unwanted *Address Resolution Protocol* (ARP) traffic. Figure 4.2 compares the CPU usage and received packets of *Customer Edge* (CE)-02 and PE-01. In Figure 4.2a we see between 400 to 600 packets/s being received by CE-02. PE-01 at first sight appears to receive less packets. However the sum of traffic on all its interfaces is about the same rate of packets/s. Looking at the CPU usage CE-02 (Figure 4.2c) has less work to do as compared to PE-01 (Figure 4.2d). This is might be explained by the fact that PE-01 has to process three streams of ARP requests. However, comparing the CPU usage of an Ubuntu server based VM (CE-01) to a vRR based VM (PE-01), is like comparing apples and oranges. *As mentioned in Section 3.2.1 the CPU usage should only be compared per individual VM per scenario.*

The spiky features of the graphs are caused by the 0.5 second pause in between every set of ARP requests, as explained in Section 3.4.1. They appear one second apart, as the measurements took place once per second.

Figure 4.2: VPLS, no ARP Sponge. CE-02 and PE-01.



(a) CE-02 Packets

(b) PE-01 Packets

(c) CE-02 CPU

(d) PE-01 CPU

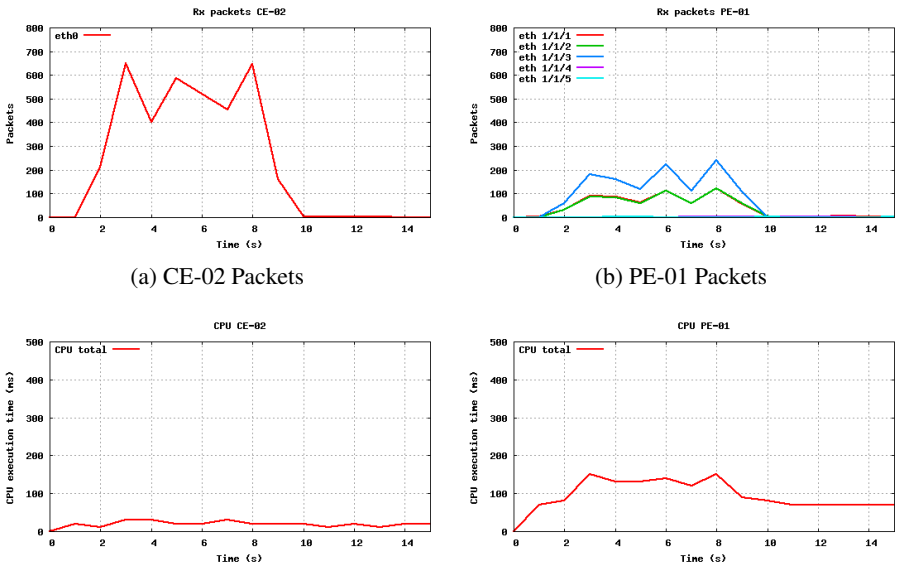
The graphs used in Figure 4.2 and their raw data can be found in `http://rp.delaat.net/2013-2014/p31/measurements.zip` → *./data/data_vpls_5/*.

4.2.2 Scenario 2: VPLS, ARP Sponge

In the second scenario, as stated in Section 3.4.1, the ARP sponge was enabled and its threshold set to 900. The TG sent out three series of ARP storms. In Figure 4.3 we clearly see the 3 peaks, this is due to the TG's triple loop of the ruleset. The graphs also show, that ideally the number of received packet/s drops to next to zero, and the CPU load lowers significantly, as the ARP Sponge provided the requesting devices with its IP address.

Although data was recorded at one measurement per second the values stored are the average over that second. Depending on when exactly the scenario was started this can lead, for example to the spikes in Figure 4.3a and 4.3b seeming apart at a varying rate.

Figure 4.3: VPLS, ARP Sponge. CE-02 and PE-01.



(a) CE-02 Packets

(b) PE-01 Packets

(c) CE-02 CPU

(d) PE-01 CPU

The graphs used in Figure 4.3 and their raw data can be found in `http://rp.delaat.net/2013-2014/p31/measurements.zip` → *./data/data_sponge_5/*

4.2.3 Scenario 3: EVPN ARP proxy enabled

When it came to the third scenario, we detected that *Ethernet Virtual Private Network* (EVPN)'s ARP proxy feature was not functioning. A capture (see Listing 4.2) showed that the IP field is empty, meaning that the MAC/IP information is not correctly combined.

Inquiry with Alcatel-Lucent brought forward the following explanation.

> "*I cross-checked with my PLM folks, and up to now, we indeed don't do ARP snooping yet (R13.0 feature), but given we also can't add static entries as of now, I am a bit unsure what the "official" way to get combined MAC/IP routes into an EVPN instance is. I was suggested some workarounds, but at least I couldn't get the easier one of both working.*"

As can be read in this statement the problem is that a PE device is not able to learn the IP address of its locally attached CE devices yet. In Chapter 2.2.2 this is called local learning. To make matters worse, the current vRR images are also not able to have the IP to MAC address information to be added statically.

Listing 4.2: Capture of an MP-BGP update showing an empty IP address field

```
25  7.903084  192.168.42.113  192.168.42.111  BGP 338 UPDATE Message, UPDATE
    Message
        Type Code: MP_REACH_NLRI (14)
        Length: 114
        Address family: Layer-2 VPN (25)
        Subsequent address family identifier: EVPN (70)
        Next hop network address (4 bytes)
        Subnetwork points of attachment: 0
            AFI: MAC Advertisement Route (2)
            Length: 33
            Route Distinguisher: 0000fde8000000c8 (65000:200)
            ESI: 00000000000000000000
            Ethernet Tag ID: 200
            MAC Address Length: 48
            MAC Address: DavicomS_78:18:2b (00:60:6e:78:18:2b)
            IP Address Length:  0
            IP Address:  NOT INCLUDED
            MPLS Label Stack: 0 (bottom)
```

Successful injection

Since a way of manually adding static MAC/IP entries is missing, we crafted our own *Multi Protocol BGP* (MP-BGP) update packet using Ostinato and injected this update on bridge br-PE01-PC01 which is the link on which PE-01 connects to *Provider Core* (PC)-01.

The ostinato stream can be found at `http://rp.delaat.net/2013-2014/p31/measurements.zip` → *./data/ostinato/EVPN_UPDATE_octetlen*.
The packet capture file can be found at `http://rp.delaat.net/2013-2014/p31/measurements.zip` → *./data/data_EVPN_SPOOF_SUCCESS_2/pe-01_pc-01.pcap*.
Listing 4.3 shows a capture of the injected packet.

Listing 4.3: Capture of the injected MP-BGP UPDATE packet

```
9 4.674060  192.168.42.113  192.168.42.111  BGP 260 UPDATE Message, UPDATE
    Message
        Type Code: MP_REACH_NLRI (14)
        Length: 48
        Address family: Layer-2 VPN (25)
        Subsequent address family identifier: EVPN (70)
        Next hop network address (4 bytes)
        Subnetwork points of attachment: 0
            AFI: MAC Advertisement Route (2)
            Length: 37
            Route Distinguisher: 0000fde8000000c8 (65000:200)
            ESI: 00000000000000000000
            Ethernet Tag ID: 200
            MAC Address Length: 48
            MAC Address: RealtekU_ce:05:01 (52:54:00:ce:05:01)
            IP Address Length:  32
            IPv4 address:  10.0.1.5 (10.0.1.5)
          MPLS Label Stack: 0 (bottom)
```

Our contact at Alcatel-Lucent at the same time tried, and also succeeded, to update the IP to MAC table statically.

> *I quickly built a setup on pe-02 that does inject a proper EVPN route, and this then also shows up in the proxy-arp table at PE-01. The way to go there is to make VPLS 200 accept an L3-Interface Binding first (allow this, and assign a service-name that is later referenced in the L3 instance)*
>
> *[..]*
>
> *I then build a local "L3-VPN" Instance - in our terminology: Virtual Private Routed Network or "VPRN" (other vendors might call this "vrf-lite"), and instead of attaching a regular SAP to the logical interface within that service, I attach the VPLS to it*

The PE-01 inserted the update, as provided by our crafted UPDATE packet, into their routes and forwarding tables, as can be seen in Listing 4.4 and 4.5 respectively.

What also can be seen is the date, which sadly shows that we had no more time to properly resume testing this scenario.

Listing 4.4: The EVPN routes of PE-01 after the succesfull injection

```
A:PE-01# show router bgp routes evpn mac
===============================================================================
 BGP Router ID:192.168.42.111   AS:65000       Local AS:65000
===============================================================================
 Legend -
 Status codes  : u - used, s - suppressed, h - history, d - decayed, * - valid
 Origin codes  : i - IGP, e - EGP, ? - incomplete, > - best, b - backup

===============================================================================
BGP EVPN Mac Routes
===============================================================================
Flag  Route Dist.         ESI                     Tag        MacAddr
                          NextHop                            IpAddr
                                                             Mac Mobility
-------------------------------------------------------------------------------
u*>i  65000:200           0:0:0:0:0:0:0:0:0       200        00:77:77:77:77:77
                          192.168.42.112                     77.77.77.77
                                                             Static

u*>i  65000:200           0:0:0:0:0:0:0:0:0       200        52:54:00:ce:05:01
                          192.168.42.113                     10.0.1.5
                                                             Seq:0

-------------------------------------------------------------------------------
Routes : 2
===============================================================================
```

Listing 4.5: The forwarding database of PE-01 after the successful injection

```
A:PE-01# show service fdb-mac

===============================================================================
Service Forwarding Database
===============================================================================
ServId    MAC               Source-Identifier        Type     Last Change
                                                     Age
-------------------------------------------------------------------------------
200       00:77:77:77:77:77 vxlan:                   EvpnS    07/01/14 21:58:05
                            192.168.42.112:200
200       52:54:00:ce:01:01 sap:1/1/5:0              L/0      07/01/14 23:17:00
200       52:54:00:ce:05:01 vxlan:                   Evpn     07/01/14 23:16:54
                            192.168.42.113:200
-------------------------------------------------------------------------------
No. of Entries: 3
-------------------------------------------------------------------------------
Legend:  L=Learned O=Oam P=Protected-MAC C=Conditional S=Static
===============================================================================
```

Although Scenario 3 did not provide data that is suitable for comparison with the other scenarios, we kept the data of the original EVPN test, as can be found in `http://rp.delaat.net/2013-2014/p31/measurements.zip` → *./data/data_evpn_arp*.

Unreach update

While testing the EVPN scenario we sometimes encountered some rather strange behaviour which can be seen in `http://rp.delaat.net/2013-2014/p31/measurements.zip` → *./data/data_EVPN_CE01_to_CE02_blocked/*

An MP-BGP UPDATE packet (Listing 4.6) was captured which PE-01 sent to PE-02 that contained an *MP-BGP_Unreachability_Network Layer Reachability Information* (MP_UNREACH_NLRI) message concerning node CE-01. As a result the route for CE-01 was withdrawn on PE-02, even though CE-01 was still up.

When we sent out an *Internet Control Message Protocol* (ICMP) request from CE-01 to CE-02 this went one way, namely from CE-01 → PE-01 → PE-02 → CE-02, but on the way back the ICMP reply was dropped at PE-02, as it did no longer have a route to CE-01.

Although we do not know why the MP_UNREACH_NLRI was sent in the first place this does proof that MAC addresses are not learned from the data frames themselves, but only from the MP-BGP UPDATES. We know this because PE-02 did not update its MAC Service Forwarding Database while it did see a *Virtual eXtensible LAN* (VXLAN) encapsulated frame with CE-01's source MAC address.

Listing 4.6: Packet capture of an EVPN UNREACH NLRI

```
1 0.000000  192.168.42.111  192.168.42.112  BGP 131 UPDATE Message
        Type Code:  MP_UNREACH_NLRI (15)
        Length: 38
        Address family: Layer-2 VPN (25)
        Subsequent address family identifier: EVPN (70)
            AFI: MAC Advertisement Route (2)
            Length: 33
            Route Distinguisher: 0000fde8000000c8 (65000:200)
            ESI: 00000000000000000000
            Ethernet Tag ID: 200
            MAC Address Length: 48
            MAC Address:  RealtekU_ce:01:01 (52:54:00:ce:01:01)
            IP Address Length: 0
            IP Address: NOT INCLUDED
            MPLS Label Stack: 0 (bottom)
```

4.3 Vendor statement

Towards the end of the project Alcatel-Lucent provided the following statement about the state and the performance of the pre-release version we were working with. Herein Alcatel-Lucent states that performance testing is not endorsed. However, as discussed in Section 3.2.1, we did not plan on comparing specialized hardware with VMs, but we planned on comparing the performance of the same VM in different scenarios. In addition the resulting data would have been used as an insight as to how the workload on a device changes depending on scenario.

> *"Alcatel-Lucent is very happy to support ECIX's and OS3's evaluation of EVPN for its suitability and proof of concept as an IXP peering fabric architecture. At the time of the testing, the virtualized SR OS simulator software was an internal-only pre-release version that was made available for testing under a special agreement. Some of the the functionality that is required to fully deploy EVPN is still under development, and will become available when the software is officially released and generally available.*
>
> *We do not recommend or endorse any control or data plane performance testing on the virtualized SR OS simulators. The purpose of the simulators is to offer an easy and cost-efficient way of building proof of concept networks that support full SR OS control plane functionality. The emulated performance is well below what custom hardware and ASICs deliver. In this context, monitoring the CPU load of a virtualized SR OS simulator does not correlate to how control plane protocols will affect the CPU performance of an actual hardware-based router."*

4.4 EVPN ARP proxy usage analysis for IXPs

Although EVPN is described thoroughly in its draft we were glad to have a working implementation to work with. The fact that we could see EVPN in practice in our setup gave us vital insights into how EVPN could be implemented by IXPs.

In a typical IXP peering network all members are known to the peering network administrator(s). Because of this it should be possible to statically add all member MAC to IP information to the PE devices. Sadly, we were unable to verify this as we were unable to statically add IP to MAC information to PE devices as described in Chapter 4.2.3.

When static IP to MAC information can be added to PE devices then it should be possible to eliminate all ARP request broadcasts on the network. How this can be achieved can be explained by looking at the different types of ARP requests and how they should be handled. based on the requested IP address, an incoming ARP request can be in one of three categories

1. **IP address is known and the MAC address has a route:**

A PE device should return the MAC address of the target.

2. **IP address is known and the MAC address has no route:**
   Although the PE device knows what MAC address belongs to the ARP request, it does not actually have a route to it. This scenario can occur when the PE device has neither learned the MAC address locally nor remotely as described in Chapter 2.2.2, or it might have received an MP_UNREACH_NLRI update such as described in Chapter 4.2.3. When this happens the CE can respond in the following ways:

   (a) Ignore and forward
   As all CE devices are known and added statically by the administrator(s) this option should not be used.

   (b) Ignore and drop
   The CE device will continue to send ARP request which the PE device has to continue to process and drop

   (c) Reply with the correct associated MAC address
   The CE device will stop sending ARP requests, but the data it sends to the requested CE device will be dropped by the PE device as it has no route.

3. **IP address is unknown:**
   This scenario can occur when, for example, an invalid route is advertised which uses a unknown next hop address, or when a ping sweep is performed on the peering network. In this case the CE device can respond in the following ways:

   (a) Ignore and forward
   As all CE devices are known and added statically by the administrator(s) this option should not be used.

   (b) Ignore and drop
   The CE device will continue to send ARP request which the PE device has to continue to process and drop

   (c) Reply with a fake MAC address
   The CE device will stop sending ARP requests, but the data it sends to the requested CE device will be either dropped by the PE device, or in case an existing MAC address was returned it can be sent to a specific location where the data can be analysed.

When we look at the items above we see that for situations 2 and 3 there are two viable options, either to drop the ARP request in the knowledge that the CE will continue to resend its ARP requests, or to 'trick' the CE by sending a reply in the knowledge that although the CE will stop sending ARP request, the data the CE device does send will never arrive to its actual destination.

The decision whether to ignore or respond depends on the performance impact of ARP requests on the PE device. When a PE can handle a large amount of ARP requests without any noticeable negative effects, then it might be better to ignore the ARP requests as then no user data would be lost. If, however a noticeable impact is detected then it is better to reply to the ARP request and drop the user data in the hope that an upper layer protocol such as MP-BGP would detect the fact that the CE is not actually reachable, for example using a handshake, or timeout mechanism.

As described in Chapter 4.2.3 we were unable to get a meaningful indication of the performance impact of EVPN in our setup. Therefore further research is needed to determine which option in both situation 2 and 3 would be preferable.

In the case that IP to MAC address information changes, for example, in the case that a member replaces its CE device, then the other CE devices need to first time out their ARP entry before they will send a new ARP request. It might be wise to implement a rule on the PE devices that whenever an IP to MAC entry changes, a gratuitous ARP request is sent to all locally attached CE devices so that they can update their ARP tables.

5 Conclusion

5.1 Scenarios

Although our testing setup was well prepared, we could only gather meaningful data for **Scenario 1: VPLS, no ARP Sponge** and **Scenario 2: VPLS, ARP Sponge**. Due to the partly non-functional *Ethernet Virtual Private Network* (EVPN) *Address Resolution Protocol* (ARP) Proxy feature the data collected for **Scenario 3: EVPN ARP proxy enabled** is not comparable to the first two scenarios. However, by further analysing the abnormalities we came across, we were able to trace their cause and to confirm our assumptions on them with Alcatel-Lucent.

1. In the current version of *SROS-VM based Route Reflector* (vRR) EVPN ARP snooping or a means of manually adding MAC to IP information entries not being implemented yet. (Empty IP information field)
2. The EVPN ARP proxy functionality itself is working, we verified this by injecting a hand-crafted MP_REACH_NLRI
3. By analysing the unexpected MP_UNREACH_NLRI, we could verify that the *Provider Edge* (PE) devices indeed learn MAC addresses from MP-BGP UPDATES only.

All gathered data is available at `http://rp.delaat.net/2013-2014/p31/measurements.zip`

5.2 Research question

To conclude this report we return to the research question posed in chapter 1.2.

> *"How can EVPN supersede classical* Virtual Private LAN Service *(VPLS)-based techniques in a typical* Internet eXchange Point *(IXP) environment, particularly in terms of handling large broadcast domains?"*

EVPN is a promising new technology which, as described in Chapter 4.4, can theoretically be used to eliminate ARP broadcasts on a IXP peering network using the fact that an IXP peering network only contains to the administrator(s) known devices. This knowledge can be used to statically add MAC to IP information to the PE devices.

Future research is needed on the performance hit of EVPN ARP proxy on a PE device. The outcome of that research can tell us more about if a PE device should ignore or respond in the case that it receives a ARP request for a unreachable known *Customer Edge* (CE) device, and an unknown CE device.

In the case that IP to MAC address information changes, for example, in the case that a member replaces its CE device, then the other CE devices need to first time out their ARP entry before they will send a new ARP request. It might be wise to implement a rule on the PE devices that whenever an IP to MAC entry changes, a gratuitous ARP request is sent to all locally attached CE devices so that they can update their ARP tables.

5.3 Future research

There are still many interesting research questions about how EVPN might be implemented by IXPs. We believe that the following items would provide interesting research projects relevant to the IXP community.

PE ARP proxy on VM testing Although we collected a lot of data on *Virtual Machine* (VM) PE CPU usage while using VPLS, we were not able to collect relevant data using our EVPN scenario to compare it with. As explained in Chapter 4.2.3 this was due to the fact that static MAC to IP bindings were not yet implemented by Alcatel-Lucent in the provided vRR image.

PE ARP proxy on hardware testing As described in Chapter 3.2.1 and 4.3 a VM will probably not be a good analogue to hardware performance. We would like to perform the same tests as described in this research project, but using specialized hardware.

MPLS based EVPN Because *Multi Protocol Label Switching* (MPLS) is heavily used throughout IXP networks, we would like to see an EVPN implementation running over MPLS instead of *Virtual eXtensible LAN* (VXLAN).

Multihoming Multihoming with all active forwarding could change the way a CE connects to an IXP. Could this replace a *Photonic Cross Connect* (PXC) by not only providing load balancing, but also redundancy in the case of a PE device failure?

Migrating from VPLS to EVPN Because of the fact that VPLS has been around for some time, and that it has been implemented often in IXP peering networks, we would like to know how an eventual migration from VPLS to EVPN could be achieved. Also we would like to know if it is possible to create a migration period by running both VPLS and EVPN at the same time.

EVPN in an IXP Finally, we would like to see an actual IXP peering network implementation of EVPN using the knowledge gained from the above research projects, as well as the ARP proxy functionality as described in Chapter 4.4.

6 Acknowledgments

First of all we would like to thank all the authors of the EVPN requirements RFC 7209, and the drafts based upon that RFC.

Secondly we would like to thank our supervisors Kay Rechthien and Thorben Krüger, and Stefan Wahl from the *European Commercial Internet eXchange* (ECIX) for providing us physical and intellectual room to conduct this project.

We would also like to thank Oliver Knapp and Greg Hankins from Alcatel-Lucent for their valuable support on setting up the vRRs, the *Command Line Interface* (CLI) and the EVPN implementation.

A big thank you to Steven Bakker from the *Amsterdam Internet eXchange* (AMS-IX) for updating and uploading the newest version of the ARP Sponge just for us!

Another big thank you to Martin Pels and Ariën Vijn from AMS-IX for their valuable input on how EVPN should work in an IXP environment.

Finally, we want to really thank the entire OS3 group! In particular for teaching us that classical encryption methods, such as the Vigenère cipher, should not be used without at least one extra layer of obfuscation.

Bibliography

[1] Victor Boteanu and Hanieh Bagheri. Minimizing ARP traffic in the AMS-IX switching platform using OpenFlow, 2013. URL http://delaat.net/rp/2012-2013/p57/report.pdf.

[2] T. Hardie. Alternative Decision Making Processes for Consensus-Blocked Decisions in the IETF. RFC 3929 (Experimental), October 2004. URL http://tools.ietf.org/pdf/rfc3929.pdf.

[3] IEEE. Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks, 2011. URL http://standards.ieee.org/getieee802/download/802.1Q-2011.pdf.

[4] V. Kompella M. Lasserre. Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling. RFC 4762 (Standards Track), 2007. URL http://tools.ietf.org/pdf/rfc4762.pdf.

[5] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T.Sridhar, M. Bursell, and C. Wright. VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. INTERNET-DRAFT, 2014. URL http://tools.ietf.org/pdf/draft-mahalingam-dutt-dcops-vxlan-09.pdf.

[6] L. Martini, E. Rosen, N. El-Aawar, and G. Heron. Encapsulation Methods for Transport of Ethernet over MPLS Networks. RFC 4448 (Standards Track), 2006. URL http://tools.ietf.org/pdf/rfc4448.pdf.

[7] L. Martini, E. Rosen, N. El-Aawar, T. Smith, and G. Heron. Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP). RFC 4447 (Standards Track), 2006. URL http://tools.ietf.org/pdf/rfc4447.pdf.

[8] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Standards Track), 2001. URL http://tools.ietf.org/pdf/rfc3031.pdf.

[9] P. Pate S. Bryant. Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture. RFC 3985 (Informational), 2005. URL http://tools.ietf.org/pdf/rfc3985.pdf.

[10] A. Sajassi, R. Aggarwal, N. Bitar, A. Isaac, and J. Uttaro. BGP MPLS Based Ethernet VPN. INTERNET-DRAFT, May 2014. URL http://tools.ietf.org/pdf/draft-ietf-l2vpn-evpn-07.pdf.

[11] A. Sajassi, R. Aggarwal, J. Uttaro, N. Bitar, W. Henderickx, and A. Isaac. Requirements for Ethernet VPN (EVPN). RFC 7209 (Informational), May 2014. URL http://tools.ietf.org/pdf/rfc7209.pdf.

[12] A. Sajassi, J. Drake, Nabil Bitar, Aldrin Isaac, James Uttaro, W. Henderickx, Y. Rekhter, R. Shekhar, B. Schliesser, S. Salam, K. Patel, D. Rao, S. Thoria, L. Yong, D. Cai, and S. Sinha. A Network Virtualization Overlay Solution using EVPN. INTERNET-DRAFT, 2014. URL http://tools.ietf.org/pdf/draft-sd-l2vpn-evpn-overlay-03.pdf.

[13] Ariën Vijn and Steven Bakker. ARP-sponge, 2014. URL https://ams-ix.net/downloads/arpsponge/3.15.1/arpsponge-3.15.1/README.

[14] Marco Wessel and Niels Sijm. Effects of IPv4 and IPv6 address resolution on AMS-IX and the ARP Sponge, 2009. URL http://delaat.net/rp/2008-2009/p23/report.pdf.

A Unrelated findings

During our project we also found the following bugs in software unrelated to EVPN itself.

Ostinato bug

While creating ARP streams in Ostinato we noticed that is is impossible to pause properly. We had to create a workaround where only 1 empty packet was sent during 0.5 seconds. This empty packet can be seen in the ARP test packet captures. We reported this bug to Ostinato.

ShareLaTeX bug

This report is unmistakeably written in LaTeX. For collaboration we chose to make use of `http://www.sharelatex.com`. Amongst others we made use of the *glossary* package. As the name suggests, the package provides support for creating glossaries, but also acronyms and lists thereof. While on a local installation we succeeded in having the package print *only* a list of acronyms, this seemed to be impossible on ShareLaTeX. With some mail exchange with the ShareLaTeX team however, we were able to pinpoint the problem and find a workaround together. (Thanks for the quick response time!)

The workaround consists of *not* putting the external file containing the acronym and glossary items into the root of the project. Also instead of printing the list of acronyms with the official `\printglossary[type=\acronymtype]`, but with `\printglossary`. In our case this would work, as - at that time - we did not have any glossary items defined.

B PE and PC configuration

PE-01

```
# TiMOS-B-12.0.R2 both/i386 ALCATEL SR 7750 Copyright (c) 2000-2014 Alcatel-Lucent.
# All rights reserved. All use subject to applicable license agreements.
# Built on Thu Apr 3 17:34:33 PDT 2014 by builder in /rel12.0/b1/R2/panos/main

# Generated WED JUN 25 13:12:34 2014 UTC

exit all
configure
#------------------------------------------------
echo "System Configuration"
#------------------------------------------------
    system
        name "PE-01"
        chassis-mode d
        dns
        exit
        snmp
            shutdown
        exit
        time
            sntp
                shutdown
            exit
            zone UTC
        exit
        thresholds
            rmon
            exit
        exit
    exit
#------------------------------------------------
echo "System Security Configuration"
#------------------------------------------------
    system
        security
            no per-peer-queuing
            cpu-protection
                link-specific-rate max
                policy 254 create
                exit
                policy 255 create
                exit
                port-overall-rate 15000
            exit
        exit
    exit
#------------------------------------------------
echo "System Login Control Configuration"
#------------------------------------------------
    system
        login-control
            idle-timeout disable
        exit
    exit
#------------------------------------------------
echo "Log Configuration"
#------------------------------------------------
    log
    exit
#------------------------------------------------
echo "System Security Cpm Hw Filters and PKI Configuration"
#------------------------------------------------
    system
        security
        exit
    exit
#------------------------------------------------
echo "QoS Policy Configuration"
#------------------------------------------------
    qos
    exit
#------------------------------------------------
echo "Card Configuration"
#------------------------------------------------
    card 1
        card-type iom3-xp-b
        mda 1
            mda-type m5-1gb-sfp-b
            no shutdown
        exit
        no shutdown
    exit
#------------------------------------------------
echo "Port Configuration"
#------------------------------------------------
    port 1/1/1
        ethernet
        exit
        no shutdown
    exit
```

```
    port 1/1/2
        ethernet
        exit
        no shutdown
    exit
    port 1/1/3
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
    exit
    port 1/1/4
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
    exit
    port 1/1/5
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
    exit
#--------------------------------------------------
echo "System Sync-If-Timing Configuration"
#--------------------------------------------------
    system
        sync-if-timing
            begin
            commit
        exit
    exit
#--------------------------------------------------
echo "Management Router Configuration"
#--------------------------------------------------
    router management
    exit

#--------------------------------------------------
echo "Router (Network Side) Configuration"
#--------------------------------------------------
    router
        interface "br-PE01-PC01"
            address 192.168.42.2/30
            port 1/1/1
            no shutdown
        exit
        interface "br-PE01-PC02"
            address 192.168.42.14/30
            port 1/1/2
            no shutdown
        exit
        interface "system"
            address 192.168.42.111/32
            no shutdown
        exit
        autonomous-system 65000
        ecmp 2
#--------------------------------------------------
echo "OSPFv2 Configuration"
#--------------------------------------------------
        ospf
            area 0.0.0.0
                interface "system"
                    no shutdown
                exit
                interface "br-PE01-PC01"
                    interface-type point-to-point
                    no shutdown
                exit
                interface "br-PE01-PC02"
                    interface-type point-to-point
                    no shutdown
                exit
            exit
        exit
#--------------------------------------------------
echo "LDP Configuration"
#--------------------------------------------------
        ldp
            interface-parameters
                interface "br-PE01-PC01"
                exit
                interface "br-PE01-PC02"
                exit
            exit
            targeted-session
            exit
            no shutdown
        exit
#--------------------------------------------------
echo "Web Portal Protocol Configuration"
#--------------------------------------------------
    exit

#--------------------------------------------------
echo "Service Configuration"
#--------------------------------------------------
    service
        sdp 12 mpls create
            far-end 192.168.42.112
            ldp
            keep-alive
                shutdown
            exit
            no shutdown
        exit
        sdp 13 mpls create
            far-end 192.168.42.113
            ldp
            keep-alive
                shutdown
            exit
            no shutdown
        exit
        customer 1 create
```

```
            description "Default customer"
        exit
        vpls 100 customer 1 create
            shutdown
            description "Regular VPLS"
            stp
                shutdown
            exit
            mesh-sdp 12:100 create
                no shutdown
            exit
            mesh-sdp 13:100 create
                no shutdown
            exit
        exit
        vpls 200 customer 1 create
            description "EVPN Service"
            vxlan vni 200 create
            exit
            bgp
                route-distinguisher 65000:200
                route-target export target:65000:200 import target:65000:200
            exit
            bgp-evpn
                vxlan
                    no shutdown
                exit
            exit
            stp
                shutdown
            exit
            sap 1/1/3:0 create
            exit
            sap 1/1/4:0 create
            exit
            sap 1/1/5:0 create
            exit
            no shutdown
        exit
    exit
#--------------------------------------------------
echo "Router (Service Side) Configuration"
#--------------------------------------------------
    router
#--------------------------------------------------
echo "OSPFv2 Configuration"
#--------------------------------------------------
        ospf
        exit
#--------------------------------------------------
echo "BGP Configuration"
#--------------------------------------------------
        bgp
            family evpn
            group "internal"
                peer-as 65000
                neighbor 192.168.42.112
                exit
                neighbor 192.168.42.113
                exit
            exit
            no shutdown
        exit
    exit


exit all

# Finished WED JUN 25 13:12:34 2014 UTC
```

PE-02

```
# TiMOS-B-12.0.R2 both/i386 ALCATEL SR 7750 Copyright (c) 2000-2014 Alcatel-Lucent.
# All rights reserved. All use subject to applicable license agreements.
# Built on Thu Apr 3 17:34:33 PDT 2014 by builder in /rel12.0/b1/R2/panos/main

# Generated WED JUN 25 13:13:20 2014 UTC

exit all
configure
#--------------------------------------------------
echo "System Configuration"
#--------------------------------------------------
    system
        name "PE-02"
        chassis-mode d
        dns
        exit
        snmp
        exit
        time
            sntp
                shutdown
            exit
            zone UTC
        exit
        thresholds
            rmon
            exit
        exit
    exit
#--------------------------------------------------
echo "System Security Configuration"
#--------------------------------------------------
    system
        security
            no per-peer-queuing
            cpu-protection
                link-specific-rate max
                policy 254 create
                exit
                policy 255 create
```

```
                exit
                port-overall-rate 15000
            exit
        exit
    exit
#--------------------------------------------------
echo "System Login Control Configuration"
#--------------------------------------------------
    system
        login-control
            idle-timeout disable
        exit
    exit
#--------------------------------------------------
echo "Log Configuration"
#--------------------------------------------------
    log
    exit
#--------------------------------------------------
echo "System Security Cpm Hw Filters and PKI Configuration"
#--------------------------------------------------
    system
        security
        exit
    exit
#--------------------------------------------------
echo "QoS Policy Configuration"
#--------------------------------------------------
    qos
    exit
#--------------------------------------------------
echo "Card Configuration"
#--------------------------------------------------
    card 1
        card-type iom3-xp-b
        mda 1
            mda-type m5-1gb-sfp-b
            no shutdown
        exit
        no shutdown
    exit
#--------------------------------------------------
echo "Port Configuration"
#--------------------------------------------------
    port 1/1/1
        ethernet
        exit
        no shutdown
    exit
    port 1/1/2
        ethernet
        exit
        no shutdown
    exit
    port 1/1/3
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
    exit
    port 1/1/4
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
    exit
    port 1/1/5
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
    exit
#--------------------------------------------------
echo "System Sync-If-Timing Configuration"
#--------------------------------------------------
    system
        sync-if-timing
            begin
            commit
        exit
    exit
#--------------------------------------------------
echo "Management Router Configuration"
#--------------------------------------------------
    router management
    exit

#--------------------------------------------------
echo "Router (Network Side) Configuration"
#--------------------------------------------------
    router
        interface "br-PE02-PC01"
            address 192.168.42.6/30
            port 1/1/1
            no shutdown
        exit
        interface "br-PE02-PC02"
            address 192.168.42.18/30
            port 1/1/2
            no shutdown
        exit
        interface "system"
            address 192.168.42.112/32
            no shutdown
        exit
        autonomous-system 65000
        ecmp 2
#--------------------------------------------------
echo "OSPFv2 Configuration"
#--------------------------------------------------
        ospf
            area 0.0.0.0
                interface "system"
                    no shutdown
                exit
                interface "br-PE02-PC01"
```

```
                    interface-type point-to-point
                    no shutdown
                exit
                interface "br-PE02-PC02"
                    interface-type point-to-point
                    no shutdown
                exit
            exit
        exit
#------------------------------------------------
echo "LDP Configuration"
#------------------------------------------------
        ldp
            interface-parameters
                interface "br-PE02-PC01"
                exit
                interface "br-PE02-PC02"
                exit
            exit
            targeted-session
            exit
            no shutdown
        exit
#------------------------------------------------
echo "Web Portal Protocol Configuration"
#------------------------------------------------
    exit

#------------------------------------------------
echo "Service Configuration"
#------------------------------------------------
    service
        sdp 21 mpls create
            far-end 192.168.42.111
            ldp
            keep-alive
                shutdown
            exit
            no shutdown
        exit
        sdp 23 mpls create
            far-end 192.168.42.113
            ldp
            keep-alive
                shutdown
            exit
            no shutdown
        exit
        customer 1 create
            description "Default customer"
        exit
        vpls 100 customer 1 create
            shutdown
            description "Regular VPLS"
            stp
                shutdown
            exit
            mesh-sdp 21:100 create
                no shutdown
            exit
            mesh-sdp 23:100 create
                no shutdown
            exit
        exit
        vpls 200 customer 1 create
            description "EVPN Service"
            vxlan vni 200 create
            exit
            bgp
                route-distinguisher 65000:200
                route-target export target:65000:200 import target:65000:200
            exit
            bgp-evpn
                vxlan
                    no shutdown
                exit
            exit
            stp
                shutdown
            exit
            sap 1/1/3:0 create
            exit
            sap 1/1/4:0 create
            exit
            sap 1/1/5:0 create
            exit
            no shutdown
        exit
    exit
#------------------------------------------------
echo "Router (Service Side) Configuration"
#------------------------------------------------
    router
#------------------------------------------------
echo "OSPFv2 Configuration"
#------------------------------------------------
        ospf
        exit
#------------------------------------------------
echo "BGP Configuration"
#------------------------------------------------
        bgp
            family evpn
            group "internal"
                peer-as 65000
                neighbor 192.168.42.111
                exit
                neighbor 192.168.42.113
                exit
            exit
            no shutdown
        exit
    exit


exit all

# Finished WED JUN 25 13:13:20 2014 UTC
```

PE-03

```
# TiMOS-B-12.0.R2 both/i386 ALCATEL SR 7750 Copyright (c) 2000-2014 Alcatel-Lucent.
# All rights reserved. All use subject to applicable license agreements.
# Built on Thu Apr 3 17:34:33 PDT 2014 by builder in /rel12.0/b1/R2/panos/main

# Generated WED JUN 25 13:13:35 2014 UTC

exit all
configure
#------------------------------------------------
echo "System Configuration"
#------------------------------------------------
    system
        name "PE-03"
        chassis-mode d
        dns
        exit
        snmp
        exit
        time
            sntp
                shutdown
            exit
            zone UTC
        exit
        thresholds
            rmon
            exit
        exit
    exit
#------------------------------------------------
echo "System Security Configuration"
#------------------------------------------------
    system
        security
            no per-peer-queuing
            cpu-protection
                link-specific-rate max
                policy 254 create
                exit
                policy 255 create
                exit
                port-overall-rate 15000
            exit
        exit
    exit
#------------------------------------------------
echo "System Login Control Configuration"
#------------------------------------------------
    system
        login-control
            idle-timeout disable
        exit
    exit
#------------------------------------------------
echo "Log Configuration"
#------------------------------------------------
    log
    exit
#------------------------------------------------
echo "System Security Cpm Hw Filters and PKI Configuration"
#------------------------------------------------
    system
        security
        exit
    exit
#------------------------------------------------
echo "QoS Policy Configuration"
#------------------------------------------------
    qos
    exit
#------------------------------------------------
echo "Card Configuration"
#------------------------------------------------
    card 1
        card-type iom3-xp-b
        mda 1
            mda-type m5-1gb-sfp-b
            no shutdown
        exit
        no shutdown
    exit
#------------------------------------------------
echo "Port Configuration"
#------------------------------------------------
    port 1/1/1
        ethernet
        exit
        no shutdown
    exit
    port 1/1/2
        ethernet
        exit
        no shutdown
    exit
    port 1/1/3
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
    exit
    port 1/1/4
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
    exit
    port 1/1/5
        ethernet
            mode access
            encap-type dot1q
        exit
        no shutdown
```

```
    exit
#------------------------------------------------------------------
echo "System Sync-If-Timing Configuration"
#------------------------------------------------------------------
    system
        sync-if-timing
            begin
            commit
        exit
    exit
#------------------------------------------------------------------
echo "Management Router Configuration"
#------------------------------------------------------------------
    router management
    exit

#------------------------------------------------------------------
echo "Router (Network Side) Configuration"
#------------------------------------------------------------------
    router
        interface "br-PE03-PC01"
            address 192.168.42.10/30
            port 1/1/1
            no shutdown
        exit
        interface "br-PE03-PC02"
            address 192.168.42.22/30
            port 1/1/2
            no shutdown
        exit
        interface "system"
            address 192.168.42.113/32
            no shutdown
        exit
        autonomous-system 65000
        ecmp 2
#------------------------------------------------------------------
echo "OSPFv2 Configuration"
#------------------------------------------------------------------
        ospf
            area 0.0.0.0
                interface "system"
                    no shutdown
                exit
                interface "br-PE03-PC01"
                    interface-type point-to-point
                    no shutdown
                exit
                interface "br-PE03-PC02"
                    interface-type point-to-point
                    no shutdown
                exit
            exit
        exit
#------------------------------------------------------------------
echo "LDP Configuration"
#------------------------------------------------------------------
        ldp
            interface-parameters
                interface "br-PE03-PC01"
                exit
                interface "br-PE03-PC02"
                exit
            exit
            targeted-session
            exit
            no shutdown
        exit
#------------------------------------------------------------------
echo "Web Portal Protocol Configuration"
#------------------------------------------------------------------
    exit

#------------------------------------------------------------------
echo "Service Configuration"
#------------------------------------------------------------------
    service
        sdp 31 mpls create
            far-end 192.168.42.111
            ldp
            keep-alive
                shutdown
            exit
            no shutdown
        exit
        sdp 32 mpls create
            far-end 192.168.42.112
            ldp
            keep-alive
                shutdown
            exit
            no shutdown
        exit
        customer 1 create
            description "Default customer"
        exit
        vpls 100 customer 1 create
            shutdown
            description "Regular VPLS"
            stp
                shutdown
            exit
            mesh-sdp 31:100 create
                no shutdown
            exit
            mesh-sdp 32:100 create
                no shutdown
            exit
        exit
        vpls 200 customer 1 create
            description "EVPN Service"
            vxlan vni 200 create
            exit
            bgp
                route-distinguisher 65000:200
                route-target export target:65000:200 import target:65000:200
            exit
            bgp-evpn
                vxlan
```

```
                    no shutdown
                exit
            exit
            stp
                shutdown
            exit
            sap 1/1/3:0 create
            exit
            sap 1/1/4:0 create
            exit
            sap 1/1/5:0 create
            exit
            no shutdown
        exit
    exit
#--------------------------------------------------
echo "Router (Service Side) Configuration"
#--------------------------------------------------
    router
#--------------------------------------------------
echo "OSPFv2 Configuration"
#--------------------------------------------------
        ospf
        exit
#--------------------------------------------------
echo "BGP Configuration"
#--------------------------------------------------
        bgp
            family evpn
            group "internal"
                peer-as 65000
                neighbor 192.168.42.111
                exit
                neighbor 192.168.42.112
                exit
            exit
            no shutdown
        exit
    exit


exit all

# Finished WED JUN 25 13:13:35 2014 UTC
```

PC-01

```
# TiMOS-B-12.0.R2 both/i386 ALCATEL SR 7750 Copyright (c) 2000-2014 Alcatel-Lucent.
# All rights reserved. All use subject to applicable license agreements.
# Built on Thu Apr 3 17:34:33 PDT 2014 by builder in /rel12.0/b1/R2/panos/main

# Generated WED JUN 25 13:14:02 2014 UTC

exit all
configure
#--------------------------------------------------
echo "System Configuration"
#--------------------------------------------------
    system
        name "PC-01"
        dns
        exit
        snmp
        exit
        time
            sntp
                shutdown
            exit
            zone UTC
        exit
        thresholds
            rmon
            exit
        exit
    exit
#--------------------------------------------------
echo "System Security Configuration"
#--------------------------------------------------
    system
        security
            no per-peer-queuing
            cpu-protection
                link-specific-rate max
                policy 254 create
                exit
                policy 255 create
                exit
                port-overall-rate 15000
            exit
        exit
    exit
#--------------------------------------------------
echo "System Login Control Configuration"
#--------------------------------------------------
    system
        login-control
            idle-timeout disable
        exit
    exit
#--------------------------------------------------
echo "Log Configuration"
#--------------------------------------------------
    log
    exit
#--------------------------------------------------
echo "System Security Cpm Hw Filters and PKI Configuration"
#--------------------------------------------------
    system
        security
        exit
    exit
#--------------------------------------------------
```

```
echo "QoS Policy Configuration"
#------------------------------------------------------------
    qos
    exit
#------------------------------------------------------------
echo "Card Configuration"
#------------------------------------------------------------
    card 1
        card-type iom3-xp-b
        mda 1
            mda-type m5-lgb-sfp-b
            no shutdown
        exit
        no shutdown
    exit
#------------------------------------------------------------
echo "Port Configuration"
#------------------------------------------------------------
    port 1/1/1
        ethernet
        exit
        no shutdown
    exit
    port 1/1/2
        ethernet
        exit
        no shutdown
    exit
    port 1/1/3
        ethernet
        exit
        no shutdown
    exit
    port 1/1/4
        shutdown
        ethernet
        exit
    exit
    port 1/1/5
        shutdown
        ethernet
        exit
    exit
#------------------------------------------------------------
echo "System Sync-If-Timing Configuration"
#------------------------------------------------------------
    system
        sync-if-timing
            begin
            commit
        exit
    exit
#------------------------------------------------------------
echo "Management Router Configuration"
#------------------------------------------------------------
    router management
    exit

#------------------------------------------------------------
echo "Router (Network Side) Configuration"
#------------------------------------------------------------
    router
        interface "br-PE01-PC01"
            address 192.168.42.1/30
            port 1/1/1
            no shutdown
        exit
        interface "br-PE02-PC01"
            address 192.168.42.5/30
            port 1/1/2
            no shutdown
        exit
        interface "br-PE03-PC01"
            address 192.168.42.9/30
            port 1/1/3
            no shutdown
        exit
        interface "system"
            address 192.168.42.101/32
            no shutdown
        exit
#------------------------------------------------------------
echo "OSPFv2 Configuration"
#------------------------------------------------------------
        ospf
            area 0.0.0.0
                interface "system"
                    no shutdown
                exit
                interface "br-PE01-PC01"
                    interface-type point-to-point
                    no shutdown
                exit
                interface "br-PE02-PC01"
                    interface-type point-to-point
                    no shutdown
                exit
                interface "br-PE03-PC01"
                    interface-type point-to-point
                    no shutdown
                exit
            exit
        exit
#------------------------------------------------------------
echo "LDP Configuration"
#------------------------------------------------------------
        ldp
            interface-parameters
                interface "br-PE01-PC01"
                exit
                interface "br-PE02-PC01"
                exit
                interface "br-PE03-PC01"
                exit
            exit
            targeted-session
            exit
            no shutdown
        exit
```

```
#------------------------------------------------------------
echo "Web Portal Protocol Configuration"
#------------------------------------------------------------
    exit

#------------------------------------------------------------
echo "Service Configuration"
#------------------------------------------------------------
    service
        customer 1 create
            description "Default customer"
        exit
    exit
#------------------------------------------------------------
echo "Router (Service Side) Configuration"
#------------------------------------------------------------
    router
#------------------------------------------------------------
echo "OSPFv2 Configuration"
#------------------------------------------------------------
        ospf
        exit
#------------------------------------------------------------
echo "WLAN Gateway Configuration"
#------------------------------------------------------------
    exit


exit all

# Finished WED JUN 25 13:14:02 2014 UTC
```

PC-02

```
# TiMOS-B-12.0.R2 both/i386 ALCATEL SR 7750 Copyright (c) 2000-2014 Alcatel-Lucent.
# All rights reserved. All use subject to applicable license agreements.
# Built on Thu Apr 3 17:34:33 PDT 2014 by builder in /rel12.0/b1/R2/panos/main

# Generated WED JUN 25 13:15:00 2014 UTC

exit all
configure
#------------------------------------------------------------
echo "System Configuration"
#------------------------------------------------------------
    system
        name "PC-02"
        dns
        exit
        snmp
        exit
        time
            sntp
                shutdown
            exit
            zone UTC
        exit
        thresholds
            rmon
            exit
        exit
    exit
#------------------------------------------------------------
echo "System Security Configuration"
#------------------------------------------------------------
    system
        security
            no per-peer-queuing
            cpu-protection
                link-specific-rate max
                policy 254 create
                exit
                policy 255 create
                exit
                port-overall-rate 15000
            exit
        exit
    exit
#------------------------------------------------------------
echo "System Login Control Configuration"
#------------------------------------------------------------
    system
        login-control
            idle-timeout disable
        exit
    exit
#------------------------------------------------------------
echo "Log Configuration"
#------------------------------------------------------------
    log
    exit
#------------------------------------------------------------
echo "System Security Cpm Hw Filters and PKI Configuration"
#------------------------------------------------------------
    system
        security
        exit
    exit
#------------------------------------------------------------
echo "QoS Policy Configuration"
#------------------------------------------------------------
    qos
    exit
#------------------------------------------------------------
echo "Card Configuration"
#------------------------------------------------------------
    card 1
        card-type iom3-xp-b
        mda 1
            mda-type m5-1gb-sfp-b
            no shutdown
        exit
```

```
        no shutdown
    exit
#--------------------------------------------------
echo "Port Configuration"
#--------------------------------------------------
    port 1/1/1
        ethernet
        exit
        no shutdown
    exit
    port 1/1/2
        ethernet
        exit
        no shutdown
    exit
    port 1/1/3
        ethernet
        exit
        no shutdown
    exit
    port 1/1/4
        shutdown
        ethernet
        exit
    exit
    port 1/1/5
        shutdown
        ethernet
        exit
    exit
#--------------------------------------------------
echo "System Sync-If-Timing Configuration"
#--------------------------------------------------
    system
        sync-if-timing
            begin
            commit
        exit
    exit
#--------------------------------------------------
echo "Management Router Configuration"
#--------------------------------------------------
    router management
    exit

#--------------------------------------------------
echo "Router (Network Side) Configuration"
#--------------------------------------------------
    router
        interface "br-PE01-PC02"
            address 192.168.42.13/30
            port 1/1/1
            no shutdown
        exit
        interface "br-PE02-PC02"
            address 192.168.42.17/30
            port 1/1/2
            no shutdown
        exit
        interface "br-PE03-PC02"
            address 192.168.42.21/30
            port 1/1/3
            no shutdown
        exit
        interface "system"
            address 192.168.42.102/32
            no shutdown
        exit
#--------------------------------------------------
echo "OSPFv2 Configuration"
#--------------------------------------------------
        ospf
            area 0.0.0.0
                interface "system"
                    no shutdown
                exit
                interface "br-PE01-PC02"
                    interface-type point-to-point
                    no shutdown
                exit
                interface "br-PE02-PC02"
                    interface-type point-to-point
                    no shutdown
                exit
                interface "br-PE03-PC02"
                    interface-type point-to-point
                    no shutdown
                exit
            exit
        exit
#--------------------------------------------------
echo "LDP Configuration"
#--------------------------------------------------
        ldp
            interface-parameters
                interface "br-PE01-PC02"
                exit
                interface "br-PE02-PC02"
                exit
                interface "br-PE03-PC02"
                exit
            exit
            targeted-session
            exit
            no shutdown
        exit
#--------------------------------------------------
echo "Web Portal Protocol Configuration"
#--------------------------------------------------
    exit

#--------------------------------------------------
echo "Service Configuration"
#--------------------------------------------------
    service
        customer 1 create
            description "Default customer"
        exit
    exit
```

```
#------------------------------------------------------------------
echo "Router (Service Side) Configuration"
#------------------------------------------------------------------
    router
#------------------------------------------------------------------
echo "OSPFv2 Configuration"
#------------------------------------------------------------------
        ospf
        exit
#------------------------------------------------------------------
echo "WLAN Gateway Configuration"
#------------------------------------------------------------------
    exit


exit all

# Finished WED JUN 25 13:15:00 2014 UTC
```