Using EVPN to minimize ARP traffic in an IXP environment

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

University of Amsterdam
Faculty of Science (FNWI)
MSc System and Network Engineering

July 3rd, 2014
Auditorium C0.110, FNWI, Sciencepark 904, Amsterdam

Background

IXP

Internet eXchange Point (IXP)

- Provides a L2 peering network
- Usually distributed over multiple locations
- Acts as a single Ethernet switch
- Members use this L2 peering network to do BGP peering
- Examples: AMS-IX, ECIX, DECIX, LINX

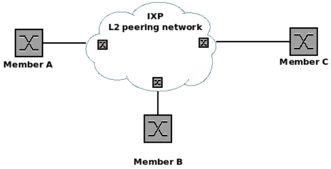


Figure : Simple L2 IXP network

How can IXPs build distributed L2 networks?

Hint: using MPLS/VPLS

MPLS (RFC 3031)

Multi Protocol Label Switching (MPLS)

- 20-bit labels create Label Switched Paths (LSP)s through the network
- MPLS ingress device determines LSP to use
- L3 packet is encapsulated with an MPLS header
- MPLS egress device 'pops' the MPLS header

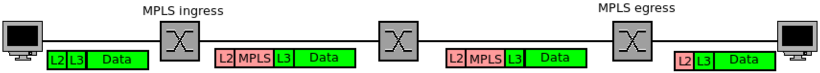


Figure : (Very) simple MPLS example

Pseudo Wires (RFCs 3985, 4447 and 4448)

Pseudo Wires (PWs)

- MPLS ingress device removes the L2 Frame Checksum Sequence (FCS)
- MPLS ingress device puts the MPLS label in front of L2 frame
- MPLS egress device re-calculates the original FCS

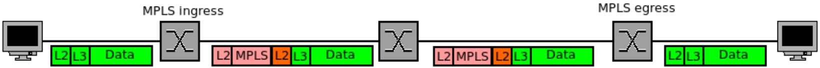


Figure : (Very) simple PW example

VPLS (RFC 4762)

Virtual Private LAN Service (VPLS)

- Creates a full mesh of PWs
- Do you remember how a normal switch learns MAC addresses?
- In VPLS Customer Edge MAC addresses are ascociated with a Pseudo Wire

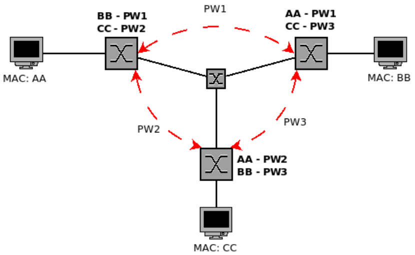


Figure : (Very) simple VPLS example

But all is not well

The ARP problem (theory)

With many members come many ARPs

- 100 members
- 1 member down
- 99 members send an ARP broadcast
- Each member has to process 98 ARP broadcasts
- When no response is received, try again!

The ARP problem (practice)

Making a Cisco Catalyst 3550 sweat

- Normal traffic == (usually) switched on hardware
- ARP traffic == processed by the CPU
- 200 members
- 100 member down
- 10000 ARPs/s

```
2w1d: %SYS-2-MALLOCFAIL: Memory allocation of 1780 bytes
failed from 0x161B38, alignment 0
Pool: I/O  Free: 9572  Cause: Memory fragmentation
Alternate Pool: None  Free: 0  Cause: No Alternate pool
-Process= "Pool Manager", ipl= 0, pid= 5
-Traceback= 1A57D0 1A6DF4 161B3C 1B2BF0 1B2E38 1C6440

CE-06#show process memory
Total: 54706596, Used: 7290848, Free: 47415748
 PID TTY  Allocated      Freed    Holding    Getbufs    Retbufs Process
   5   0 3588357308   12341112    2608820 2551100460   18951784 Pool Manager
   9   0         92  962095304       6940          0 2595909708 ARP Input

CE-06#show process cpu
CPU utilization for five seconds: 98%/14%; one minute: 47%; five minutes: 15%
 PID Runtime(ms)   Invoked      uSecs   5Sec   1Min   5Min TTY Process
   5      124152     18789       6607 24.57% 11.25%  3.73%   0 Pool Manager
   9      526356    572797        918 56.16% 26.10%  8.40%   0 ARP Input
```

Current solution

Current solution: ARP sponge

Currently used solution: ARP sponge

- Counts ARP requests to a specific IP address
- Sends out a (gratious) ARP reply when counter reaches a threshold
- Members are now satisfied and **S**top **T**he **F**rantic **U**nnesecerities
- In practice it reduced ARP traffic nearly tenfold (ask Niels)

```
<STATE>
IP           State    Queue    Rate (q/min)      Updated
10.0.4.101   DEAD     600      7755.420          2014-06-24@18:42:15
10.0.4.102   DEAD     600      10622.259         2014-06-24@18:42:14
</STATE>

1819 10.540946 RealtekU_a5:01:01 Broadcast ARP 42 Gratuitous ARP for 10.0.4.101 (Request)
                    Sender MAC address: RealtekU_a5:01:01 (52:54:00:a5:01:01)
                    Sender IP address: 10.0.4.101 (10.0.4.101)
                    Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
                    Target IP address: 10.0.4.101 (10.0.4.101)

1820 10.541152 RealtekU_a5:01:01 Broadcast ARP 42 Gratuitous ARP for 10.0.4.102 (Request)
                    Sender MAC address: RealtekU_a5:01:01 (52:54:00:a5:01:01)
                    Sender IP address: 10.0.4.102 (10.0.4.102)
                    Target MAC address: Broadcast (ff:ff:ff:ff:ff:ff)
                    Target IP address: 10.0.4.102 (10.0.4.102)
```

But what if we could prevent ARP entirely?
Introducing: EVPN

EVPN - requirements (RFC 7209)

EVPN requirements RFC7209 (May 2014)

An EVPN implementation should address the following shortcomings of VPLS:

- Multihoming with all-active forwarding (members can load balance)
- Multipoint-to-multipoint LSP support
- Simpler provisioning
- VLAN-aware bundling
- Network reconfigures time indepedant from MAC addresses learned
- **Minimizing of flooding of multi-destination frames**
- Support for flexible VPN technologies

The most Interesting specific rule in regards to ARP is:

- (R11b) "*... **the solution SHOULD minimize the flooding of broadcast frames** ...*"

EVPN (draft-ietf-l2vpn-evpn-07)

draft-ietf-l2vpn-evpn-07 (May 2014)

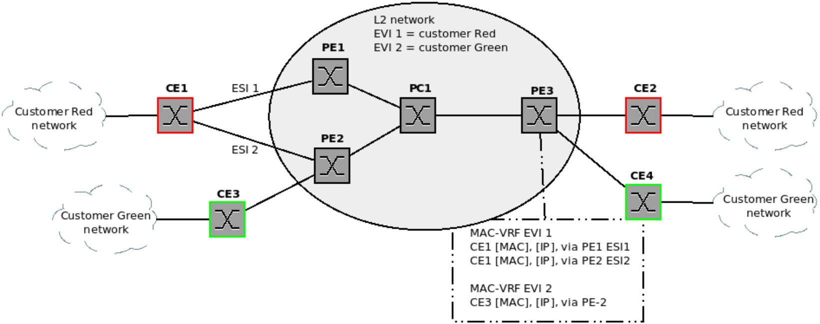
- Do NOT learn MAC address from data frames
- Use MP-BGP to learn MAC addresses
- Optionally also send the IP address!
- **Act as an ARP proxy!**
- But the workload is shifted to the EVPN edge!

Figure : EVPN ARP proxy

EVPN - Terminology

EVPN Terminology

- CE - Customer Edge device *
- PE - Provider Edge device *
- PC - Provider Core device *
- EVI - a unique EVPN instance running across the PEs
- Ethernet Tag - a VLAN tag within an EVI
- MAC-VRF - a Virtual Routing and Forwarding table for an EVI on a PE
- ESI - Ethernet Segment Identifier used for multi homing



Building the L2 tunnel
Everyone knows MPLS is on layer 1.5, right?

EVPN - building the L2 tunnel

Where to put the MPLS labels?

- The draft is not as clear as we would like
- L2 MPLS encapsulation might be common (PWs)
- It is NOT standard MPLS (RFC 3031)
- L2 MPLS encapsulation is not properly introduced
  first mention chap. 6.1 (*VLAN Based Service Interface*), page 11:
  "*[. . .] Ethernet frames transported over MPLS/IP network [. . .]*"
- Is this like a Pseudo Wire, i.e. is the FCS dropped?
- Is the entire frame encaplulated including the FCS?

EVPN - MP-BGP MAC/IP update

EVPN MP-BGP MAC/IP Update

```
+---------------------------------------+
| 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)           |
+---------------------------------------+
```

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

EVPN - MP-BGP MAC/IP update

EVPN MP-BGP MAC/IP Update

```
+---------------------------------------+
| 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 Distinguisher
- Identifies the EVI of this update

Logical setup

Logical network design

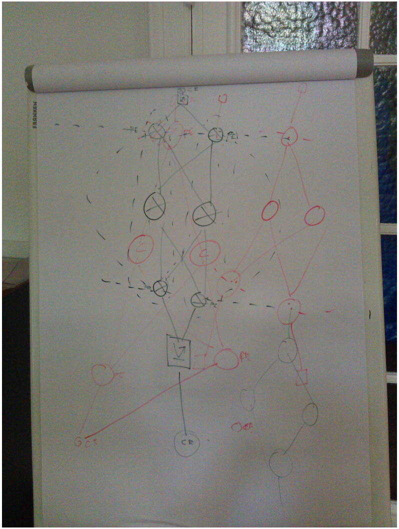


Figure : Logical network layout

Logical network design

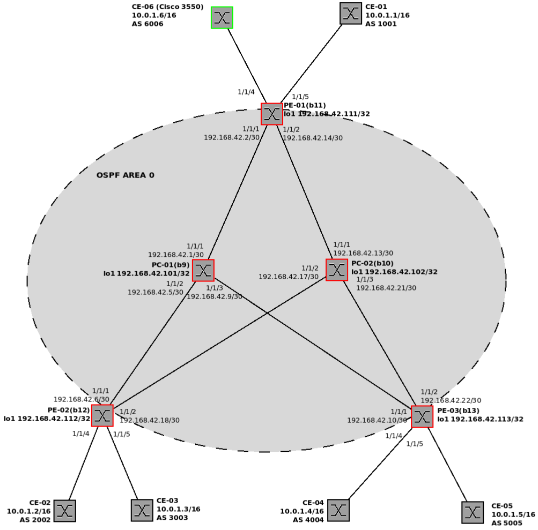


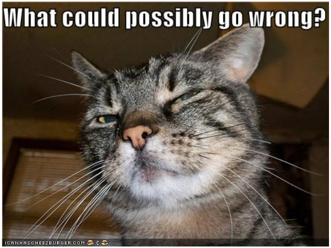
Figure : Logical network layout

Physical setup

SROS-VM based Router Reflectors (vRRs)

SROS-VM based Router Reflectors (vRRs)

- Alcatel-Lucent (ALU)
- VM of i386 hardware control processor module for 7750 Service Router
- 5 Matching licences
- Internal only pre-release
- Used with kvm/qemu



Performance of VMs

Performance of VMs

- Router VMs traffic solely processed by virtual CPU(s)
- Hardware routers utilize e.g. ASICs for the forwarding (linespeed)
- No direct relation of performance possible

. . . but

- Comparing VPLS and EVPN ARP proxy within the same VM might show interesting differences in CPU usage
- We ASSUME that ARP proxying might be done in CPU anyway

Physical network design

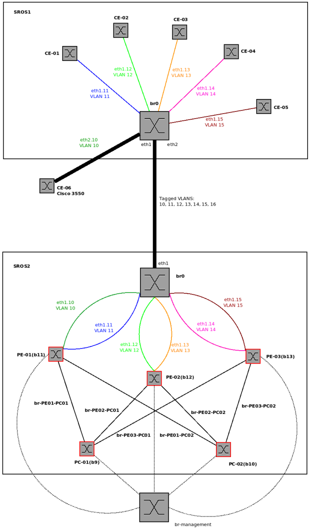


Figure : Overview of the physical network layout

Physical network design - top

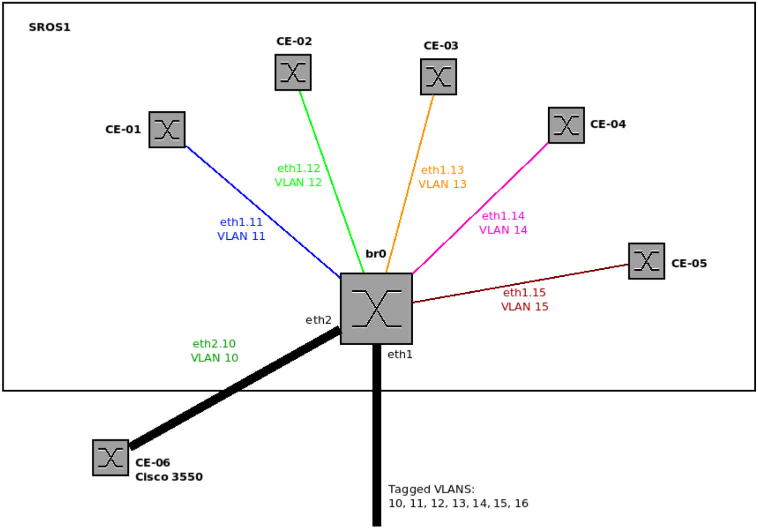


Figure : Bottom part of the physical network layout

Physical network design - bottom

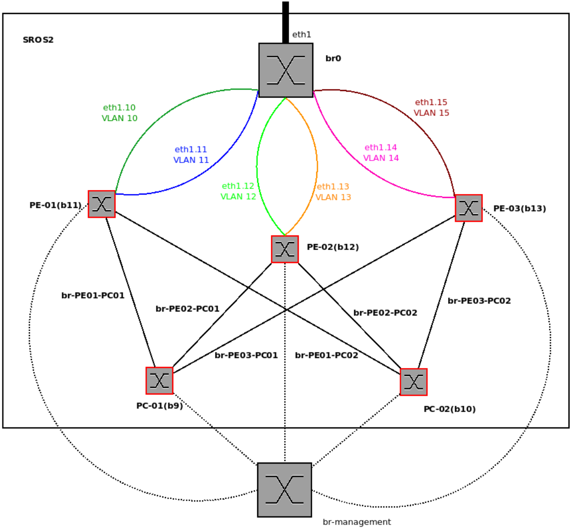


Figure : Bottom part of the physical network layout

Interconnections sros1

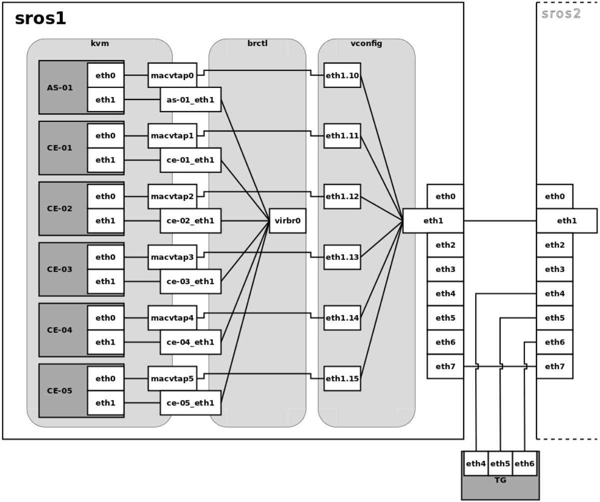


Figure : Interconnections of (virtual) interfaces on sros1

Interconnections sros2

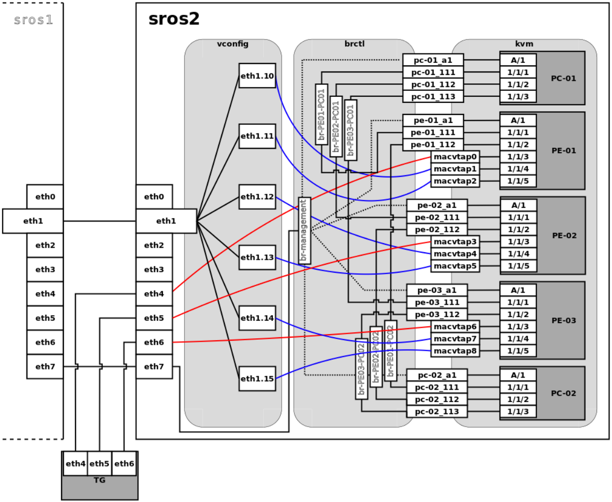


Figure : Interconnections of (virtual) interfaces on sros2

Getting the vRRs to boot

Getting the vRRs to boot

- Little to no documentation available
- Configuration files contained options to provide license file
- . . . but those were not respected.
- It took about two weeks to get the vRRs to boot in our setup

Test scenarios

Test scenarios

Simulating an IXP with 100 clients behind each PE-router, where a 4th router becomes unreachable.

1. **Pure VPLS**: Test without any measure against unwanted ARP traffic. Baseline measurement.
2. **ARP SPONGE**: Test with the ARP sponge enabled and the threshold set to 600.
3. **EVPN**: Test with EVPNs ARP proxy feature enabled.

Oh, look mommy! We got VXLAN!

VXLAN (draft-sd-l2vpn-evpn-overlay-03)

VXLAN

- We always assumed that we would be working with MPLS-based EVPN
- But we got to know that it actually is VXLAN-based
- However in regard to the ARP proxy functionality, both work the same
- e.g. the MAC/IP UPDATE looks exactly the same
- Note: performance SHOULD not be influenced (No ARP should be VXLANned)

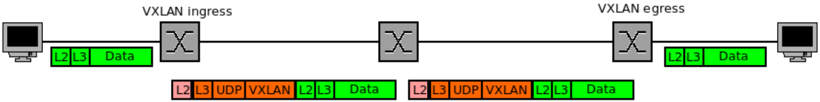


Figure : VXLAN

Results

High packet drop rate @ 10 Mbps

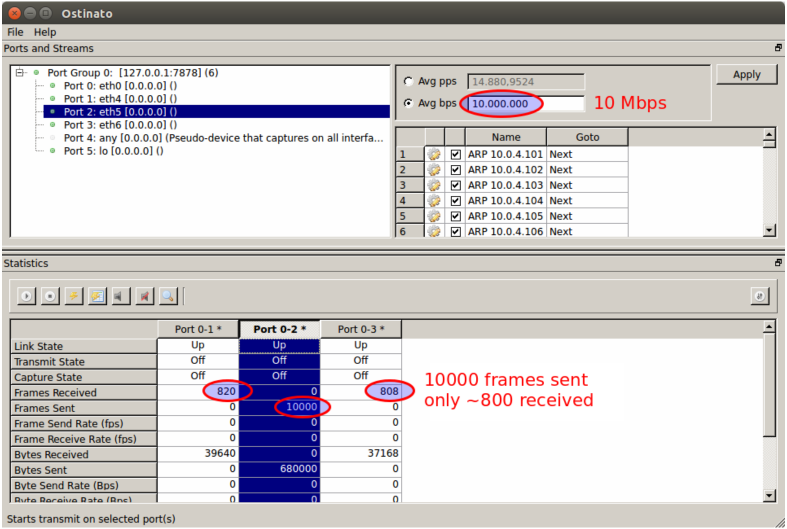


Figure : 10 Mbps - high packet droprate

Still high drop rate @ 1 Mbps

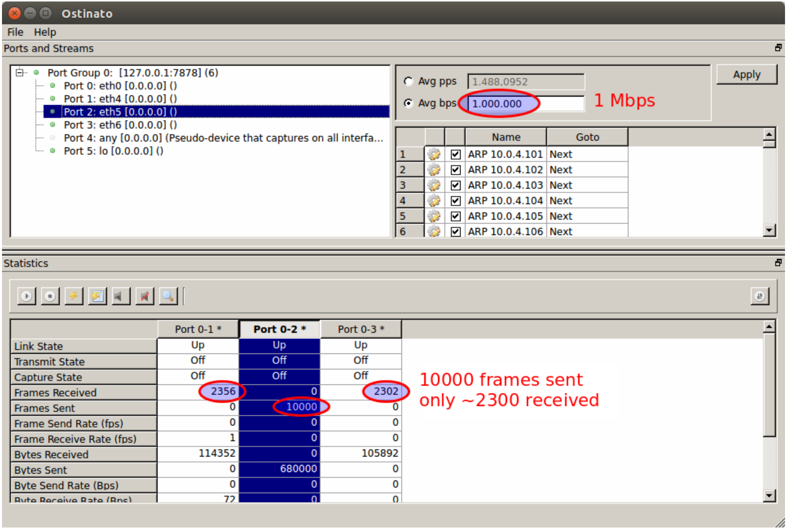


Figure : 1 Mbps - still high droprate

No drops drops @ 100 Kbps

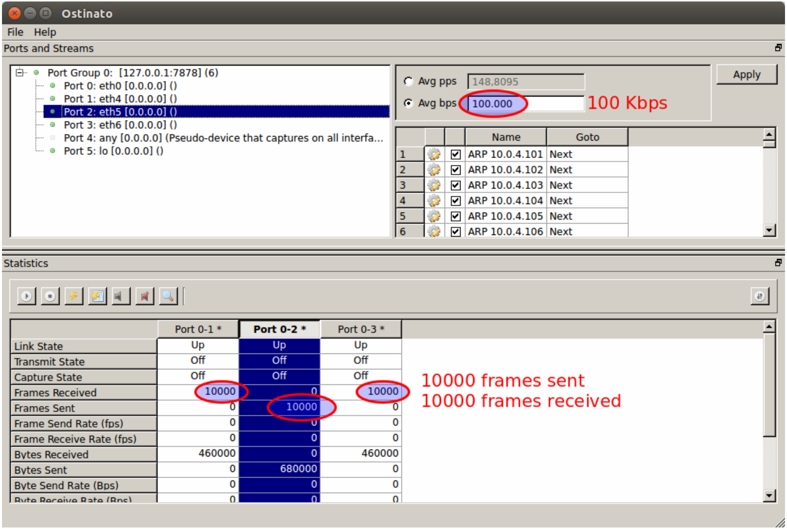


Figure : 100 Kbps - no drops!

High drop rate - confirmed

[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.[1]

[1] At this time we were offered to test on a hardware setup, but due to time restraints we had to decline.

Restricting to one CPU

Restricting to one CPU

- Low data plane throughput == low CPU usage
- Restricted the vRR VMs to one 1 VCPU each

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

~~Lutz~~ Lots-o-data

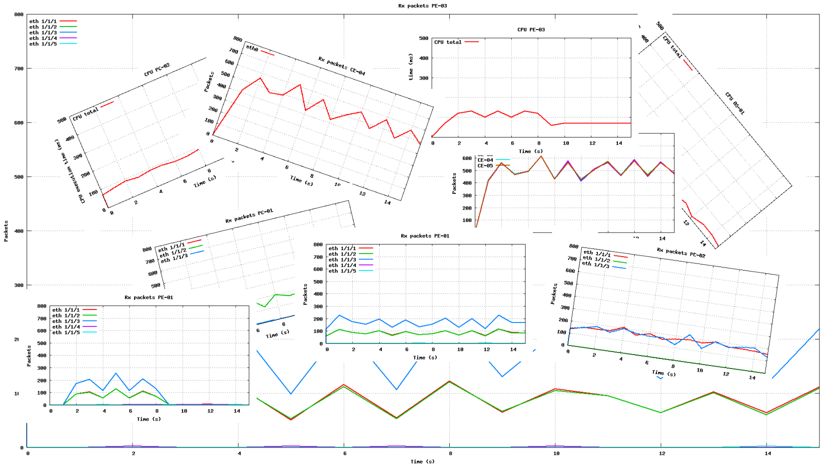


Figure : In an automated fashion data was gathered and graphed.

Performing the tests

- Measurements for VPLS scenario
- Measurements for ARP sponge scenario
- Measurements for EVPN....Wait...What?

EVPN - What is wrong?

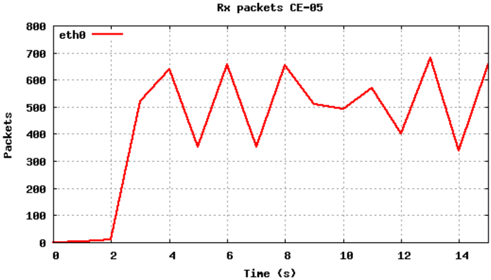


Figure : EVPN - What is wrong?

ARP proxy not working

EVPN MAC/IP UPDATE

```
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)
```

ARP proxy not working - confirmed

I cross-checked with my PLM folks, and up to now, we indeed don't do ARP snooping yet (R13.0 feature)[2], 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.

[2] We were conducting our research on R12.0

Handcrafted 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)
```

Injected Mac Route

```
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           0:0:0:0:0:0:0:0:0:0     200       00:77:77:77:77:77
                          192.168.42.112                    77.77.77.77
                                                            Static

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

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

ARP proxy working

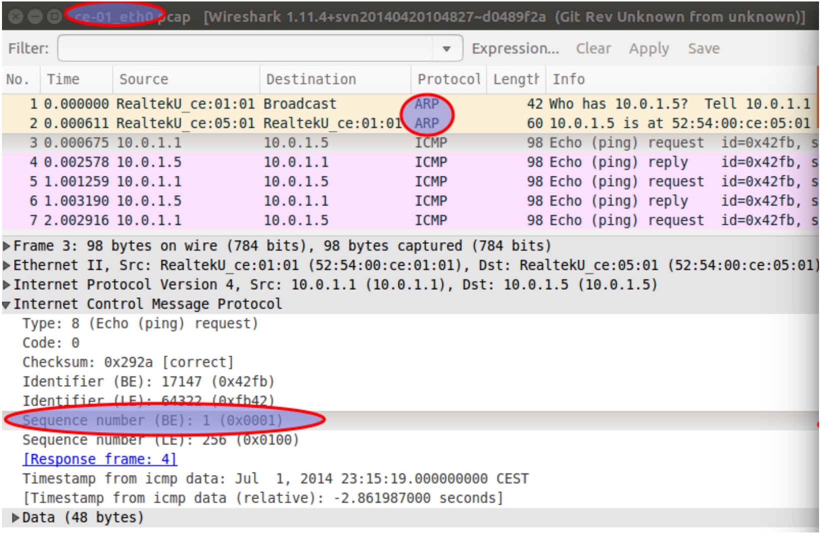


Figure : ARP proxy capture on CE-01 eth0

ARP proxy working

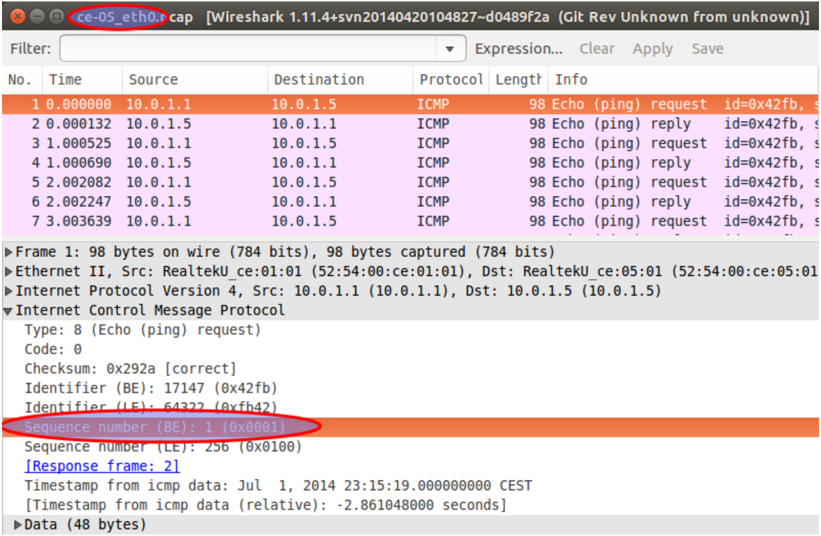


Figure : ARP proxy capture on CE-05 eth0

Conclusion

Findings

Problems identified and confirmed

- ARP Proxy not working yet, due to combination of a lack of ARP Snooping and no means to manipulate the bindings manually
- draft-ietf-l2vpn-evpn-07 not clear as to which L2 tunneling technique is used

. . . but still

- We think EVPN is a VERY interesting and promising technology for IXPs
- We would love to test this on hardware
- We would love to see a performance comparison on PEs between VPLS and EVPN with ARP Proxy
- We would love to dive into the multi-homing aspect

Abschlussprojektspräsentationsabrundungsfragenzeit!

Thank You!