Effects of IPv4 and IPv6 address resolution on AMS-IX and the ARP Sponge

Marco Wessel and Niels Sijm
Universiteit van Amsterdam
{mwessel, nsijm}@os3.nl

Research Project 2, System and Network Engineering.

July 4, 2009

Abstract

Heavy ARP traffic can cause routers to run out of resources. Therefore, AMS-IX developed a daemon called ARP Sponge. When the number of ARP requests for an IP address exceeds a threshold, the ARP Sponge starts sending ARP replies which keeps ARP traffic down. The ARP Sponge currently works using IPv4 only.

IPv6 uses Neighbor Discovery (ND) instead of ARP for address resolution. ND uses the *solicited-node* multicast address instead of broadcast, consuming less resources from routers. Therefore, IPv6 support should not necessary, though it still can be used to detect network problems and legacy router configurations.

We tested this on an emulation of the AMS-IX platform, both the current version and its new version, yet to be introduced, based on MPLS/VPLS and with various customer platforms and found that IPv6 support need (and maybe should) not be implemented.

Contents

Chapter 1

Introduction and backgrounds

1.1 Introduction to AMS-IX

The Amsterdam Internet Exchange[1] (AMS-IX) offers customers a layer 2 platform to exchange IP traffic. Internet service providers (ISPs), content providers and other Internet related organisations from all over the world are connected to AMS-IX. They use the AMS-IX to communicate with each other directly using a shared infrastructure, rather than expensive dedicated or transit connections.

Currently, AMS-IX has 317 members using a total of 580 ports generating a peak throughput of 675 gigabits per second.[2] The Border Gateway Protocol, version 4 [Rekhter et al., 2006] (BGP) is used to exchange routing information between members.

Continuity is very important for AMS-IX. An outage of only a few seconds can cause connection issues all over the world. To prevent that from happening, AMS-IX wants its platform to be as transparant and failsafe as possible. Therefore, only the use of certain protocols is allowed on the main ISP peering LAN.[3]

1.2 AMS-IX topology overview

The currently used topology is called AMS-IXv3. The next version, AMS-IXv4, is planned to replace AMS-IXv3 in late 2009.

1.2.1 AMS-IXv3

Both logically and physically, AMS-IXv3 implements a star topology. It centers around a Foundry Networks NetIron MLX-32 Ethernet switch. The MLX-32 is connected to smaller edge switches, which are located at different places in

[1] http://www.ams-ix.net/
[2] http://www.ams-ix.net/connected/
[3] http://www.ams-ix.net/technical/allowed.html

Amsterdam. Customers connect to the edge switches in various ways, depending on their port speed. A diagram is shown in figure 1.1.

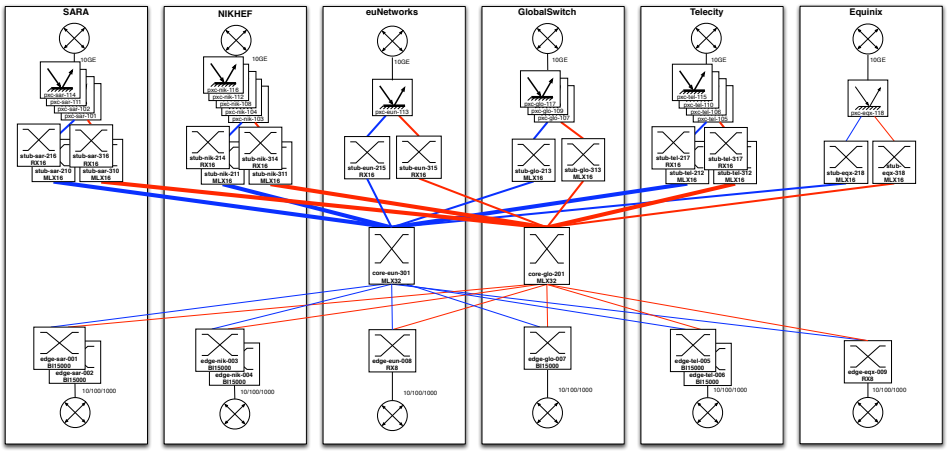

Figure 1.1: AMS-IXv3. Diagram supplied by AMS-IX.

AMS-IXv3 handles redundancy by having a failover network ready to take over. The core switches and 10 Gigabit Ethernet (10GbE) edge switches are all installed as pairs. Failover is done using the Virtual Switch Redundancy Protocol (VSRP), a Foundry-proprietary protocol. 10GbE customers are connected to an optical cross-connect which, in case of a fail-over situation, switches over customers to their redundant switch. The two networks are visualised in the diagram as red and blue.

The most important issue with AMS-IXv3 is scalability. There are currently no Ethernet switches available with enough 10GbE ports to accomodate the needs of AMS-IX in the near future. Switches with 100GbE ports would solve this problem, but won't be available until the 100GbE standard is defined by IEEE. Because of 100GbE being unavailable for some time to come, AMS-IX has decided to implement a more scalable topology, called AMS-IXv4.

1.2.2 AMS-IXv4

To be scalable, AMS-IXv4 has four core switches active at the same time. The plaform is designed to scale up to more core switches if necessary. To construct an Ethernet broadcast domain, Virtual Private Lan Service (VPLS) is used to connect all nodes on layer 2. VPLS will be supported by Multiprotocol Label Switching (MPLS), allowing AMS-IX to do load-balancing and other forms of traffic engineering. The edge switches will decide which core switch to use. The core switches forward traffic based on MPLS labels instead of Ethernet source addresses. The

core switches are not aware of VPLS and only switch based on MPLS labels.

This set-up is visualised in figure 1.2. Some parts are still based on link-failover, these parts are blue and red. Other parts in black are always active. All edge switches connect to all core switches. This allows for optimum load balancing of traffic.

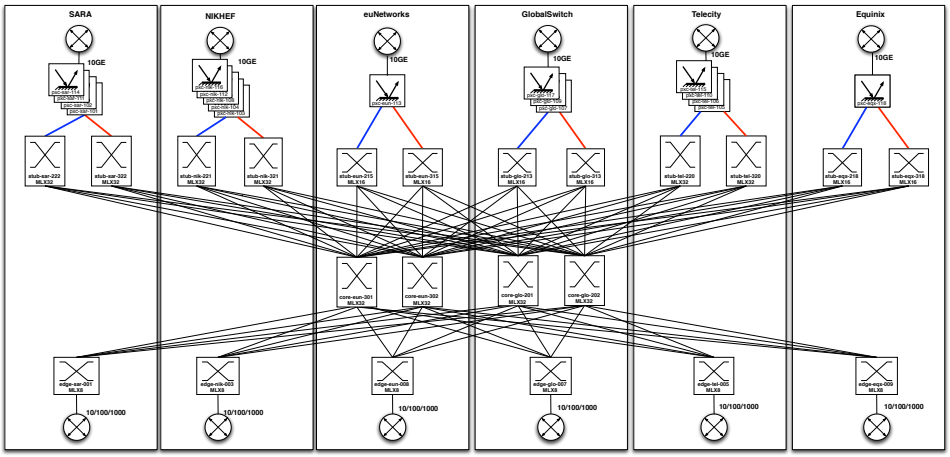

Figure 1.2: AMS-IXv4. Diagram supplied by AMS-IX.

With AMS-IXv4, failover is replaced by redundancy. All edge switches are connected to both core switches. In a normal situation, traffic is load-balanced over all four core switches. If one fails, the another one can take over the workload.

1.3 Problems caused by too much ARP traffic

On Ethernet networks, the Address Resolution Protocol [Plummer, 1982] (ARP) is used to find the MAC-address for a given IPv4 address.

ARP uses Ethertype `0x0806` together with Ethernet broadcasting. A node will broadcast an ARP Request packet to ask for the MAC address of an unknown IPv4 address. The node using the requested IP address replies (using regular unicast) with an ARP Reply packet, which includes its MAC address.

In order to work, it is important that all nodes using IPv4 listen for ARP packets and reply to them if necessary. The nodes therefore need to process all Ethernet broadcast messages with Ethertype `0x0806`. For each ARP packet, they must decide whether or not to reply.

Processing ARP packets can take a lot of processing power. Because all ARP pack-

ets need to be examined in order for ARP to work, processing ARP packets may take precedence over other activities, depending on the Operating System. As such, when there is a lot of ARP traffic, routers may be unable to do other processing tasks like maintaining BGP sessions.

This problem was noticed on AMS-IX when the ISP peering LAN was renumbered to new IPv4 addresses. Members in the new IPv4 range were trying to reach members in the old IPv4 range and vice versa. Larger amounts of ARP packets than usual crossed the network, consuming all available processing power on some customer routers, not leaving enough to process BGP in a timely manner, resulting in lost BGP sessions. Other routers started sending ARP packets to reestablish these BGP sessions, resulting in an ARP storm that brought even more routers down.

1.4 ARP Sponge – the AMS-IX solution

To help routers survive heavy ARP traffic, AMS-IX decided to try keeping the amount of ARP traffic down. For this purpose, AMS-IX developed a daemon, written in Perl, called *ARP Sponge*.

The ARP Sponge listens on the ISP peering LAN for ARP traffic. When the number of ARP Requests for a certain IP address exceeds a threshold, the ARP Sponge sends out an ARP Reply for that IP address using its own MAC address. From that moment, the IP address is *sponged*: all traffic to that node is sent to the ARP Sponge. This prevents ARP storms because it keeps the amount of ARP traffic down.

When the interface of a sponged IP address comes up again, it generally sends out a gratuitous ARP request packet [Perkins, 1996]. This is an ARP packet with both source and destination IP address set to the IP address of the node sending the packet. It is used mostly in case the MAC-address changed, so that other nodes can update their ARP caches.

When the ARP Sponge receives any traffic from a sponged IP address (including but not limited to gratuitous ARP requests, ARP requests for other nodes, BGP peering initiations, etc.), it ceases sponging the IP address, thus no longer sending out ARP replies for that IP address.

1.4.1 Effects and behaviour

From the start, the ARP Sponge has been sponging IP addresses continuously. There are two reasons for this. First, with over 300 members it is likely that, at any given moment, one or more nodes are down. Second, sometimes ARP Request packets are sent for IP addresses that are no longer in use or haven't been used at all. Two typical cases are outdated BGP configurations and network probes.

The ARP sponge results in a significant decrease in ARP requests on the network. For instance, on May 12th, 2008 the ARP sponge was down for more than one hour due to a request from a customer. The average number of ARP packets per minute in that month was about 1,450. During the outage, a peak of 13,902 ARP packets per minute was measured, almost ten times as much as normal. Fifteen minutes after the ARP Sponge was restarted, the amount ARP traffic was reduced to its normal rate of about 1,450 packets per minute.

One of the interesting side-effects of the ARP Sponge is that traffic destined to nodes that don't exist or don't exist anymore is diverted to the sponge machine. This allows AMS-IX some insight into customers who have not cleaned up their routers' BGP configurations when peers have disappeared. It is likely that this problem will be more prevalent with IPv6 as IPv4 addresses are re-used while IPv6 addresses are not necessarily; this is up to the customers.

1.5 Research question

What differences are there between IPv4 and IPv6 as relating to the ARP Sponge and infrastructure, and is an IPv6 implementation of the ARP Sponge necessary?

Chapter 2

Extending the ARP Sponge with IPv6 support

2.1 Currently

The current version op the ARP Sponge only deals with address resolution requests for IPv4. For IPv6 address resolution, a different protocol, Neighbor Discovery [Narten et al., 2007] (ND), is used. ND is part of IPv6, making use of ICMPv6 packets. This is in contrast to ARP, which is separately defined from IPv4 and has a different ethertype. Parts of the ND protocol need to be implemented in order for the ARP Sponge to deal with IPv6 address sponging.

2.2 IPv4 ARP versus IPv6 Neighbour Discovery

ND offers more than just address resolution in IPv6. The replacements for ARP Requests and ARP Replies are called Neighbour Solicitations and Neighbour Advertisements respectively. The neighbour discovery in IPv6 is also capable of discovering routers on the local subnet, but this is out of scope for this document.

One of the most important differences between ARP and ND is the way messages are distributed. ARP uses Ethernet broadcast messages while ND uses multicast messages. This difference seems subtle as multicasting on ethernet is mostly treated as broadcasting, but has some interesting effects.

Neighbour Advertisements, when sent unsolicited to mirror the function of gratuitous ARPs, are relevant to all nodes and are therefore sent to the *all-nodes* multicast address, `ff02::1`. This allows all nodes in the local broadcast domain to receive them. In effect, this is technically a multicast message but as it is received by all IPv6-capable nodes it can be considered a broadcast. Unsolicited advertisements are relatively rare, occuring mostly after a node has started operations on the network or changed its address.

In contrast, Neighbour Solicitations are only relevant to the node using the IPv6 address of which the MAC address is requested. Neighbour Solicitations are very similar to ARP requests with the very important difference that they are sent to the

solicited-node multicast address, `ff02::1:ffXX:XXXX`[1]. This is an IPv6 multicast address, created from the last three octets of the node's IPv6 unicast address which take the place of the X-characters. IPv6 multicast addresses map to an Ethernet multicast address, making them usable for address resolution. As with the all-nodes address, while the solicited-node address is technically a multicast address, Ethernet currently treats this as a broadcast address. All nodes therefore receive these notifications at their interfaces so they may process them if they are relevant (i.e., in the right multicast group) and reply if needed.

As a result of mapping the unicast address to a multicast group address, nodes are only in the same multicast group if the last three octets of their IPv6 unicast address match. In case of autoconfiguration, this typically isn't the case because those last three octets are normally also the last three octets of the node's MAC-address.

The benefit of using the solicited-node address is that nodes only receive Neighbor Solicitations that are likely to be relevant to them. This is by design and eliminates the need to process the content of *all* Neighbor Solicitations. Irrelevant Neighbour Solicitation packets can be quickly discarded because they were not sent to the right multicast group. This requires much less processing power as it can be done quickly and efficiently in hardware. All nodes still receive the message at their NICs, but these can ignore those that are not addressed to a subscribed multicast group.

2.2.1 Practically on AMS-IX

In the specific case of AMS-IX, unicast addresses are not generated by using MAC-addresses but by using the customer's Autonomous System number followed by a number for the specific router. An AMS-IX peering LAN IPv6 address uses the prefix `2001:07f8:0001:0000:0000:a5`. Appended is the customer's Autonomous System number and a router sequence number. The router sequence number is to be chosen by the customer, but generally starts at 1 and is increased by 1 for each subsequent router.

For example, for AS1200's first router, the unicast address on the AMS-IX peering lan would be `2001:07f8:0001:0000:0000:a500:1200:0001`. The last three octets of this address are `00:00:01`. This node would thus join the solicited-node multicast address `ff02::1:ff00:0001`. This multicast address then maps in a similar way, using the last 4 octets of the IPv6 multicast group address and prepending `33:33`, to an ethernet address, in this case `33:33:FF:00:00:01`.

[1] http://www.iana.org/assignments/ipv6-multicast-addresses/

A customer using a different AS-number, for instance AS1103, uses a slightly different unicast address. The first IPv6 router for this AS would be addressed `2001:7f8:1::a500:1103:1` (abbreviated), the last three octets being `03:00:01`. The corresponding multicast group therefore is `ff02::1:ff03:0001` with the associated ethernet address `33:33:FF:03:00:01`. This is different from the address used above, so it can be easily detected and ignored.

There exists one possible pitfall: because of how the AS-number is 'encoded' into the IPv6 unicast address, multiple customers could still share a single multicast group. Customers of which the last two digits of their respective AS-numbers match will have routers in the same multicast group using this addressing scheme. For instance, customers with AS-numbers 1100, 1200, and even 12300 will use the same multicast group.

AMS-IX has published a list of all IPv4 and IPv6 addresses is use on the ISP peering LAN.[2]. With that list, we have calculated the maximum and average number of hosts in an Ethernet multicast group. We found that at most 6 nodes are in the same Ethernet multicast group in the current situation. The average number of nodes in an Ethernet multicast group is 2.21. This was done by counting how many known nodes shared the last two digits of the AS number.

The number of nodes sharing an Ethernet multicast group can be alleviated somewhat by moving the AS-number in the IPv6 unicast address backwards slightly, occupying more of the last three octets of which the second one will in most cases be `00`. However ultimately this scheme will always overlap somewhat. Another option is to vary the last two octets such that a more unique number is used instead of simply a sequential number. As this number has no meaning to AMS-IX, it can be anything meaningful to the customer or even randomly chosen.

2.3 Reasons for adding IPv6 ND support

The main purpose of the ARP Sponge is to limit ARP traffic to save processing power on customers' routers. With IPv6, address resolution needs much less processing power because of the use of multicast for Neighbour Solicitations. Therefore, sponging IPv6 addresses may not save as much processing power on customer routers as sponging IPv4 addresses.

To test this, we built a test setup in the AMS-IX laboratory to measure the difference in processor load between ARP and ND. In chapter 3.3, the test setups are explained. In chapter 3.4, the results of the tests are explained.

[2] http://www.ams-ix.net/connected/table/

Besides limiting address resolution traffic, the ARP Sponge maintains a list of sponged IP addresses. AMS-IX considers this information because it can show the existence of network problems in an early stage. Sponged addresses can be nodes that are down, but also nodes that don't exist anymore, for example nodes that are mentioned in legacy BGP configurations.

Another use for a list of sponge IP addresses is limiting the impact of ping sweeps. Ping sweeps can result in an ARP or ND storms and are therefore potentially harmful.

Ping sweeps may seem rare but occur on AMS-IX on a regular basis. Some customers use tools that use ping sweeps to search for new nodes in the network. AMS-IX uses arpwatch[3] to detect new nodes in the network and ping sweeps to test customer router reachability. Ping sweeps from outside the AMS-IX network may be started by hackers or botnets and can be used for host discovery as well as Denial of Service attacks.

2.4 Pitfalls with IPv6 support

2.4.1 List of sponged addresses

The ISP peering LAN uses a /23 prefix for its IPv4 subnet. The maximum number of addresses on a /23 subnet is 512, so the list of sponged IPv4 addresses will never grow beyond 512 entries. Even when the sponge is used on a /16 network, the list of potentially sponged addresses is of a manageable length, 65536 addresses. For its IPv6 subnet, AMS-IX uses a /64 prefix, as is standard practice in IPv6 subnetting. This makes the possible number of IPv6 addresses on the subnet 2^{64}, or 1,844,674,4073,709,551,616. Maintaining a list of all sponged IPv6 addresses will be impossible over time, especially when a ping sweep is used to iterate (parts of) the address space.

A solution to this problem is to list only the most relevant sponged IPv6 addresses. There are several way to determine the relevance of a sponged address. One way is to look at the time elapsed since the address became sponged. This can be implemented as a timeout mechanism, just like ARP caches have. With this solution, the size of the list is variable.

One way to limit the size of the list is to use a ring buffer. A ring buffer always accepts new entries. When the buffer is full, the oldest entry is replaced. This prevents the list from getting too long; it has a fixed maximum size.

[3] http://www-nrg.ee.lbl.gov/

Currently, the ARP Sponge uses one list for sponged IP addresses. Another approach is to use a combination of a ring buffer and a white list. The white list contains all IP addresses that are known to exist because they have been seen transmitting traffic on the network. This list can be generated from received Neighbour Solicitations as well as onsolicited Neighbour Advertisements.

Addresses that appear on the whitelist are considered to exist and require sponging if necessary. This allows the sponge to differenciate between addresses that exist and always have them in the list of potentially sponged addresses, while addresses that are not known to exist can be kept in an expiring list such as a ring buffer. It also allows to set different timeouts and/or thresholds.

2.4.2 Neighbour caches

Another pitfall exists. All nodes keep caches of neighbours, aptly named the neighbour cache. This is a list of IPv6 addresses and the associated MAC addresses, parallel to the ARP cache used in IPv4. Once a Neighbour Solicitation is sent out for an address, it is placed in the cache as 'incomplete'. If the solicitation is answered, the entry is marked 'reachable'. If not, the entry is removed from the cache after a timeout. While an entry is in the cache, the node actively keeps track of the reachability of the neighbour. If it receives acknowledgement packets from the neighbour (signifying the ability of the neighbour to receive and process packets), the cache entry is again marked as 'reachable'. If nothing happens for a determined time, the entry is marked 'stale'. Depending on the circumstances, this means either the entry will time out and be removed after a longer time or a unicast neighbour solicitation is sent to the node, starting the process from the beginning. The latter is done only if the node is actively sending packets to the neighbour, such as when it is trying to set up a BGP session.

It is possible to fill a routing node's neighbour cache with entries if an IPv6 sponge were to exist and answer Neighbour Solicitations. This can be done simply by sending enough pings to each address to trigger the sponge. As with the sponge's lists as discussed above, this list could quickly grow to very large sizes, potentially causing a Denial of Service if the list is consistently filled to its maximum size. All that is needed is a host behind the router (or multiple for greater effect), able to send a number of IPv6 packets addressed to random addresses inside the IPv6 peering LAN subnet.

AMS-IX currently recommends customers to set their routers' ARP and Neighbour cache timeouts to at least two hours, or preferably even four. This practice, while useful in keeping ARP requests and Neighbour Solicitations to a minimum during normal operation, makes it even easier to fill up neighbour caches by extending greatly the time given to do so.

Chapter 3

Effects of ARP and ND on AMS-IX

To determine the necessity of an IPv6 ND Sponge, it is necessary to know what effects on switch and customer equipment performance large amounts of ND traffic has, and how this relates to ARP traffic. For this we set up a number of experiments.

3.1 Introduction to AMS-IXv4

The main difference between AMS-IXv3 and AMS-IXv4 is the number of active core switches. AMS-IXv3 has one active core switch and one hot-standby switch for failover, while AMS-IXv4 has multiple active core switches at the same time. This replaces failover with redundancy and makes traffic engineering (e.g. load balancing) possible.

AMS-IX offers a layer 2 platform for exchanging IP traffic. A pure layer 2 platform can not contain multiple possible routes between two points, as this would cause packets (in particular broadcast packets) to loop and multiply every time they were processed by a switch. AMS-IXv4 will use Multiprotocol Label Switching (MPLS) and Virtual Private LAN Service (VPLS) to create an Ethernet broadcast domain on top of an MPLS-based network. Because there is a clear distinction between inter-switch links (of which there is a full-mesh) and connections to customer equipment, the looping problem can be avoided by never flooding traffic to other switches if it was not received from an end-node.

3.1.1 MPLS for load-balancing

The MPLS protocol [Rosen et al., 2001] was originally designed to speed up routing in a network. The first MPLS router a packet crosses, called Label Edge Router, decides which predefined Label Switched Path (LSP) a packet will take through the MPLS network. It attaches the corresponding label to the packet and sends it to the next MPLS router.

All subsequent MPLS routers will read the label and forward the packet based

on that label. This differs from IP routing, where the destination IP address is matched against a list of prefixes. MPLS does label swapping rather than routing: in MPLS, routing decisions are made on Label Edge Routers only.

These days, routers do IP routing in hardware. This takes away the performance advantage of MPLS because IP routing is nearly as fast as packet switching. Therefore, the main reason for MPLS in AMS-IXv4 is traffic engineering.

The edge switches act as Label Edge Routers, adding labels to all packets before forwarding. Four LSPs are defined from each edge switch to another edge switch, one over each core. The edge switches do load-balancing over all LSPs. The core switches are not VPLS-aware, they simply switch MPLS packets.

3.1.2 VPLS to create a broadcast domain

MPLS creates a network on top of Ethernet, between Ethernet and IP. As such, it is often called layer 2.5. AMS-IXv4 uses VPLS [Lasserre and Kompella, 2007] to create an Ethernet broadcast domain on top of an MPLS network using RSVP-TE [Awduche et al., 2001] for label distribution. All edge nodes have two LSPs between eachother, for redundancy. VPLS combines this into a usable ethernet broadcast-domain without loops.

In VPLS, a Provider Edge (PE) thus emulates an Ethernet switch on top of a layer 2.5 network. VPLS routers in a network establish a full mesh. This involves two steps: discovery and signaling. Discovery is about locating other VPLS routers. Signaling is the process of establishing virtual Ethernet connections, which are called pseudo-wires.

To provide any-to-any connectivity, VPLS routers forward Ethernet traffic to each other, similar to generic Ethernet switches. VPLS uses split-horizon to prevent loops in the network.

3.2 Theoretical effects of AMS-IXv4 switch on hardware

Customer routers are unaware of MPLS and VPLS. All they see is an Ethernet broadcast domain. To customer routers, the switch to AMS-IXv4 is invisible. The change therefore will not affect customer equipment at all with regard to ARP or ND.

The role of the edge switches will change. Besides Ethernet switching, they have to act as Label Edge Routers and VPLS PEs. We expect this to mean that the

amount of processing per packet will increase slightly, due to the added functions of deciding what labels to attach, what LSPs to use, etc.

3.3 Test setups to verify theoretical effects

3.3.1 Lab setup AMS-IXv3

Our AMS-IXv3 lab setup consists of two routers, a Juniper M5 and a Cisco 7200, connected in a layer 2 VLAN to a traffic generator and a Linux machine. While using only a single switch, this setup emulates the current AMS-IX network setup, using 10GbE, 1GbE and Fast Ethernet ports.

Schematically the network looks as seen in figure 3.1.

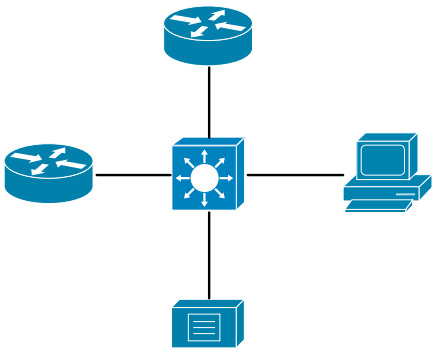

Figure 3.1: Lab setup AMS-IXv3

3.3.2 Lab setup AMS-IXv4

The AMS-IXv4 lab setup, like the above, consists of the Juniper M5, Cisco 7200 and Linux machine, but this time the nodes are connected to a switch with the relevant ports set to be in a VPLS instance.

Schematically this looks like figure 3.2. The switch is now represented by a VPLS cloud.

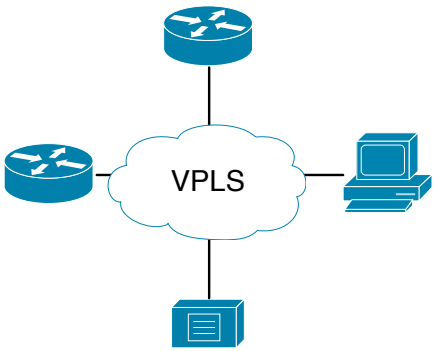

Figure 3.2: Lab setup AMS-IXv4

3.4 Tests

We tested five different kinds of ARP and ND-traffic for the routers and two for the switches. For the routers, these kinds were:

- ARP requests relevant to the node (i.e., requests for its IP)
- ARP requests not relevant to the node (i.e., requests for another IP)
- Neighbour Solicitations relevant to the node
- Neighbour Solicitations not relevant to the node
- Neighbour Solicitations not relevant to the node, but in one of its multicast groups

For the switches we tested only irrelevant ARP requests and Neighbour Solicitations, as the layer 3 addressing is not relevant to Ethernet in this case[1]. Using irrelevant requests (for a non-existent IP address) allows us to see the load caused only by the broadcast/multicast requests without that of the unicast answers.

To avoid measuring noise, we used elevated amounts of traffic, sending in all cases 10,000 frames per second. This is decidedly higher than AMS-IX is likely to see, but allows us to see more accurately the amount of processing needed for our traffic rather than the processing needed to process other packets, such as those generated by our collecting of the values.

[1] However, it would be if such features are ARP Protection were enabled.

Measurements from the Juniper and Cisco routers were collected using SNMP, measurements from the Linux machine using `vmstat` locally. Measurements were averaged over one minute.

In a normal situation on AMS-IX, the switches limit the amount of broadcast and multicast frames that are processed and/or forwarded to 1000 and 10,000 respectively. To achieve more accurate results, we disabled these limits for testing.

3.4.1 Results

The results of our tests are displayed in the following tables. All values are in percents of CPU utilisation.

Nodes

	ARP host	ARP other	ND host	ND other	ND group
Juniper M5	5	4	100	0	69
Cisco 7200	91	55	90	55	55
Linux	2	1	17	0	8

The Juniper M5 has no trouble handling 10k ARP requests in either case. Relevant requests cause a very slightly higher amount of activity, which is to be expected as it requires generating and sending an answer as well as processing the request. It appears to be much less optimised for handling Neighbour Discovery, as sending the exact same amount of Neighbour Solicitations per second caused the CPUs involved to be resource-starved. It handles irrelevant neighbour solicitations very well – this showed no activity at all. The CPUs involved are those of the Forwarding Engine Board (FEB) and the Flexible PIC Concentrator (FPC). These are tasked with packet processing. The main CPU, which runs the OS, had no measurable involvement in the process.

The Cisco requires most of its (main) CPU to handle the ARP packets, and the same goes for IPv6. The results for relevant ARP and ND are within expectations, but the non-relevant ND causes the CPU to be loaded as much as non-relevant ARPs and Non-relevant group-addressed ND, suggesting this platform does not program its MACs to filter out relevant ethernet addresses.

The Linux machine, being a full-fledged computer, allows slightly more insight into what is happening. Sending it 10kpps of ARP traffic shows almost no CPU load; less than two percent in both cases. This can be explained by the network interface card offloading this task from the main CPU. This is confirmed by reading through the source code for the driver. On this platform we could monitor

the amount of interrupts occurring during our tests. During relevant ARP-testing, interrupt-levels rose to approximately 2300 per second, from 300 while idle. Irrelevant ARPs caused half that amount. During relevant ND testing, however, between 18000 and 19000 interrupts were generated. During irrelevant group-adressed testing, roughly 10300 interrupts were generated per second. Subtracting the 300 interrupts per second when idle, this amounts to one interrupt per packet. It is obvious from the irrelevant ND test that the MACs on the ethernet card in this host are programmed with ethernet addresses to filter on: Neighbour Solicitations sent to another multicast group generated no increase in interrupt levels at all[2].

Note, though that this is not specific to Linux nor will every instance of Linux show these results. Whether or not these results can be duplicated is dependent on the NIC used and if its driver supports offloading features; these results apply to any generic PC with such features enabled. In our tests, we used an Intel 82544EI 1000BASE-SX Ethernet card. This card supports offloading ARP and various IPv4-features such as checksums. It does not support anything with regard to IPv6[3].

Switches

We tested using the switches AMS-IX currently uses in AMS-IXv3 and plans to use in AMS-IXv4: Foundry Networks (now Brocade) MLX-series switches. Our tests relate only to the Provider Edge switches, so we tested using the MLX-8 type switch as this type would be used for this purpose.

The test results for the switches are displayed in the table below. Although we tested this on two different switches to verify the results, they were identical and are thus only displayed once. These figures represent the usage of the CPU of the line card on which the frames were received. We set up our tests so multiple line cards would be included to test if any other CPU was involved. This was not the case.

	ARP L2	ARP VPLS	ND L2	ND VPLS
Foundry	42	63	40	62

As can be seen, 10k frames per second of Layer 2 broadcast/multicast traffic cause the line card CPU utilisation to reach 41 to 43 percent depending on the protocol. In the case of sending the the traffic over a VPLS instance, this rises to 63 or 64%. Sending ND packets caused slightly less load than ARP packets.

[2] Except, of course, while the NIC was in promiscuous mode.

[3] http://download.intel.com/design/network/manuals/8254x_GBe_SDM.pdf

The slight difference between ARP and ND in both cases we cannot explain directly because of lack of insight into the inner workings of the switch. We suspected the slightly different frame size to be of influence, but testing showed that utilisation did not differ between sending fabricated packets (non-ARP and non-ND) of the the two lengths (72 bytes for ARP and 98 bytes for ND) to broadcast and multicast addresses. On the other hand, sending packets of any length with the ethertype for ARP set but no other data inserted causes the utilisation to rise to 98% when the switch is set to an L2-configuration. The same frame with the ethertype set to that to anything else shows the same as earlier tests. This suggests the switch is doing some form of processing on ARP frames but does not do so for IPv6 ND frames. Unfortunately no more information on this could be found.

These results also tell us that the switch to MPLS/VPLS will cause slightly more resource utilisation on the switch for processing ARP and ND frames. The increase is similar for both protocols, however. It need therefore not be separately considered as being more or less problematic.

Chapter 4

Conclusion

The problem of ARP causing large amounts of interrupts in connected hardware are long known. It was thought of when IPv6 Neighbour Discovery was invented. As such, in IPv6 the problem appears to have been solved, at least well enough that on 'regular' networks the problem should not occur at all anymore. We define normal in this case as using stateless autoconfiguration and being of relatively limited size. With stateless autoconfiguration, nodes use their MAC-address to generate their unicast address. The chances of using the same multicast group in such cases are one in 2^{24}.

With the enlarged subnet size comes the problem of keeping track of all these addresses. This problem occurs both in the ARP sponge, which must keep track of addresses so it can decide when to start sponging, and in attached nodes as they must keep track of addresses to be able to communicate. Using an ND sponge opens up a possibility for launching a Denial of Service attach against the attached router equipment.

On the other hand the sponge provides a useful service: it allows AMS-IX network engineers to see traffic that would normally silently disappear. In cases where routers are configured to communicate with other routers so as to exchange routing information with them (peering) and one of these routers is decomissioned, AMS-IX engineers can detect which routers are still attempting to communicate with the router that has disappeared.

Our recommendation is currently to not implement an IPv6-capable version of the ARP-sponge. However, if the above service is important and the sponge *is* implemented, care must be taken to avoid the pitfalls described in this document.

4.1 Future Work

On some switching platforms it is possible to create layer 2 Access Control Lists. These lists allow a switch to decide to forward a frame (or not) depending on its source and/or destination MAC address. As layer 3 unicast IPv6 addresses map to a layer 2 ethernet multicast Ethernet address, it might be possible to limit the effects of an ARP sponge filling neighbour caches using this feature if the IPv6 addresses are structured similarly to how they are on AMS-IX. The effects of such

an approach on a switching platform and its effectiveness could be studied.

Chapter 5

Acknowledgements

This work could not have been done without the help of more than a few people. From AMS-IX, we want to thank Steven Bakker, Arien Vijn, Martin Pels and Niels Bakker (in no particular order) for information and ideas. Our fellow students Attilla and Yuri have also provided valuable input and a few laughs at that – Attilla in particular. Of course AMS-IX as an organisation should be thanked as well for providing us with a place to work at, people around for bouncing ideas off, a lab full of cool toys to play with and of course lunch every day.

Bibliography

[Awduche et al., 2001] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and Swallow, G. (2001). RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209 (Proposed Standard). Updated by RFCs 3936, 4420, 4874.

[Kompella and Rekhter, 2007] Kompella, K. and Rekhter, Y. (2007). Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling. RFC 4761 (Proposed Standard).

[Lasserre and Kompella, 2007] Lasserre, M. and Kompella, V. (2007). Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling. RFC 4762 (Proposed Standard).

[Narten et al., 2007] Narten, T., Nordmark, E., Simpson, W., and Soliman, H. (2007). Neighbor Discovery for IP version 6 (IPv6). RFC 4861 (Draft Standard).

[Perkins, 1996] Perkins, C. (1996). IP Mobility Support. RFC 2002 (Proposed Standard). Obsoleted by RFC 3220, updated by RFC 2290.

[Plummer, 1982] Plummer, D. (1982). Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard).

[Rekhter et al., 2006] Rekhter, Y., Li, T., and Hares, S. (2006). A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard).

[Rosen et al., 2001] Rosen, E., Viswanathan, A., and Callon, R. (2001). Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard).