ARP Sponge

Niels Sijm
Marco Wessel

AMS-IX?

- One of the largest IXP in the world by members, ports and traffic
- 317 Members, 580 ports, 675Gb/sec peak
- All in one L2 subnet.

AMS-IX Set-up

- AMS-IXv3:
  - Big L2 subnet
  - Hub/spoke with backup network
  - VSRP for failover
  - No longer scalable.

AMS-IX Set-up

- AMS-IXv4
  - MPLS/VPLS
  - One network, redundancy replaces failover
  - Still one big L2 subnet for customers

ARP Sponge

- ARP Sponge exists to decrease amount of ARP traffic on AMS-IX
- Spoofs ARP replies when necessary

Research Question

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

ARP Problems

- ARP, needed for IPv4 over Ethernet
- Resolves IP addresses into MAC addresses
- Broadcast: 'who is at this IP?'
- Must be processed by everyone who receives it
- Too much ARP may cause CPU overload situations.

ARP Sponge

- Too much ARP happens when nodes are unavailable (down, nonexistent)
  - ARP requests are repeated (in case they were lost), often by multiple requestors
- ARP Sponge exists to notice this and reply in downed node’s stead.
  - Nodes are ‘happy’, so far as their ARP caches go

ARP Sponge

- Start ‘sponging’ when too many requests are received in small amount of time
- Stop ‘sponging’ when traffic is received from the real host
  - Gratuitous ARP, ARP request for other node, anything.

ARP Sponge Benefits

- Nearly ten-fold reduction of ARP traffic seen on an average day:
    - 1450 ARPs/min with
    - 13902 ARPs/min without
- Additionally, allows AMS-IX to see traffic for nonexistent nodes
    - Notably, BGP sessions with routers that no longer exist

IPv6

- Current Sponge only deals with IPv4
- What about IPv6?
  - IPv6 replaces ARP with ‘Neighbour Discovery’
  - Part of ICMPv6
  - Multicast instead of Broadcast
  - Also allows router discovery

Issues for IPv6 Sponge

- IPv6 subnet is 64 bits large
- 18446744073709551616 (2^{64}) potential addresses
- Sponge must keep state for IP addresses to determine when to sponge
- ‘limited’ memory capacity not enough

Issues for IPv6 Sponge

- How to solve?
  - Use two lists:
    - White list of hosts known to exist (limited amount), filled by watching for traffic, can be seeded
    - Ring-buffer or timed-expiry for other addresses so old addresses expire automatically

IPv6 ND

- ND consists primarily of:
    - Neighbour Solicitations and Advertisements
        - Functionally equivalent to ARP
        - multicast on Ethernet, using *solicited-node* address
    - Router Solicitations and Advertisements.

IPv6 ND

- Solicited-node address: ff02::1:FFXX:XXXX
- XX:XXXX replaced with last three octets of unicast address
- IPv6 Multicast address maps to ethernet multicast address: 33:33:XX:XX:XX:XX
- XX'es replaced with last 32 bits of multicast address

IPv6 ND

- Example: 2001:7b8:200:2202:216:cbff:fe90:fe41
- Solicited-node address: ff02::1:ff90:fe41
- Multicast Ethernet address: 33:33:ff:90:fe:41

IPv6 ND

- This allows ‘selection at the gate’, or: don’t process irrelevant solicitations
- MAC chips can be programmed for this
- Keeps CPU utilization down in comparison to ARP

Group overlap

- Multicast group addressing scheme on AMS-IX:
  - addresses are structured as 2001:7f8:1::a5xx:xxxx:yyyy
  - AS-numbers that end in the same two digits ‘overlap’:

    2001:7f8:1::a500:1200:0001 and
    2001:7f8:1::a512:3400:0001 result in
    33:33:ff:00:00:01

  - Average of 2.21 nodes per group, maximum 6

Comparisons

- Router CPU utilization ARP/ND, 10kpps

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

- Notes:
  - Juniper: FEB/FPC CPU; Cisco: main CPU
  - Cisco very busy handling packets in general, but nothing *extra* for irrelevant ND
  - Linux: used e1000 ethernet adapter which has ARP-offloading

Switch comparisons

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

- Tested 10kpps ARP/ND in L2 environment vs. VPLS
- Small difference between ND/ARP: processing in switch
- VPLS increases line-card processing load evenly between ARP/ND

IPv6 Sponge Issue

- 64-bit subnet means potentially *very* large neighbour cache for routers
    - Attacker behind router starts ping sweep of peering subnet
    - Router starts soliciting for neighbours (that don’t exist)
    - ARP Sponge answers
    - Neighbour cache fills up

Recommendation

- Given:
  - Multicasting of Neighbour Solicitations with ‘selection at the gate’
  - Potential to fill up neighbour caches
- We recommend not implementing IPv6 Sponge daemon (yet)
  - If implementing for other reasons: use small lists to prevent cache problem

Thank you.

Questions?