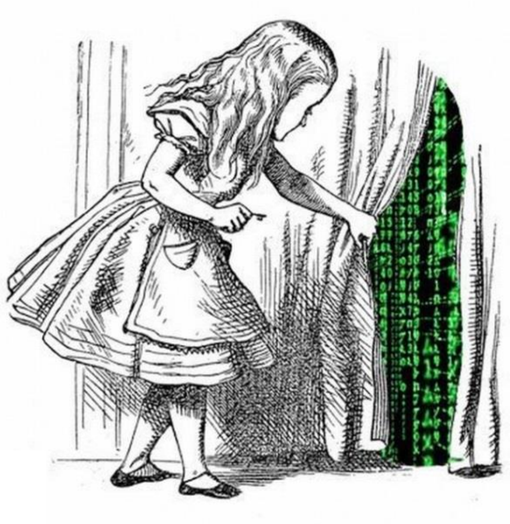
MLD Considered Harmful

Antonios Atlasis
aatlasis@secfu.net

Jayson Salazar
jsalazar@ernw.de

Rafael Schaefer
rschaefer@ernw.de

Road Map

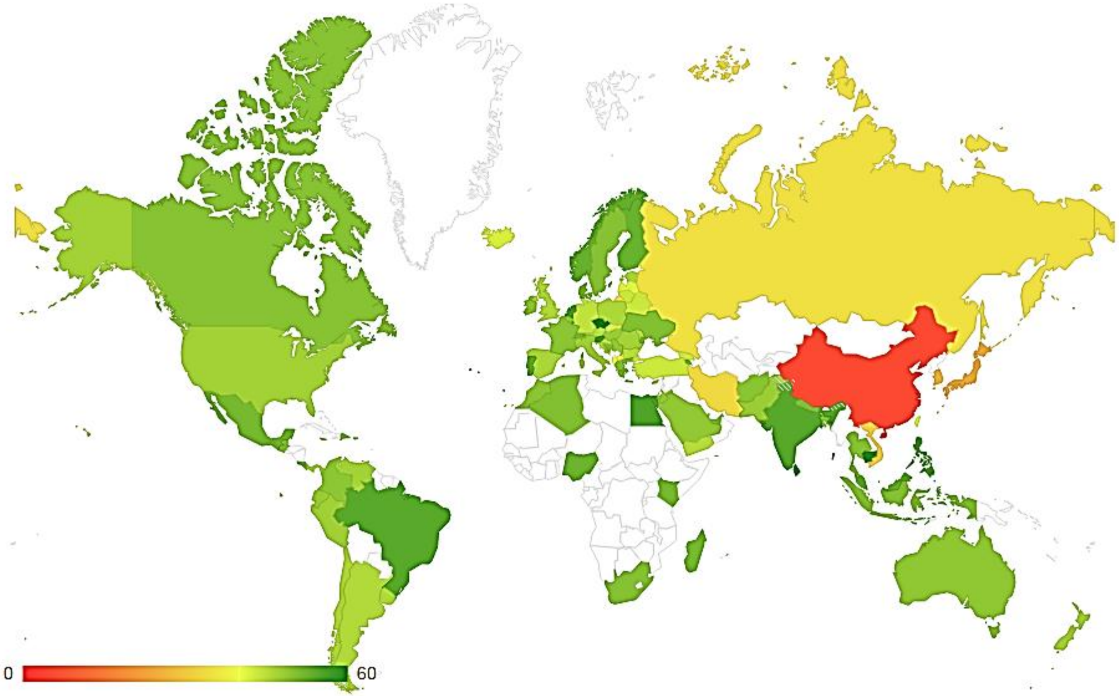


- Background Information
- MLD, Myths and Facts
- Profiting from MLD
- Mitigations
- Conclusions

Background Information

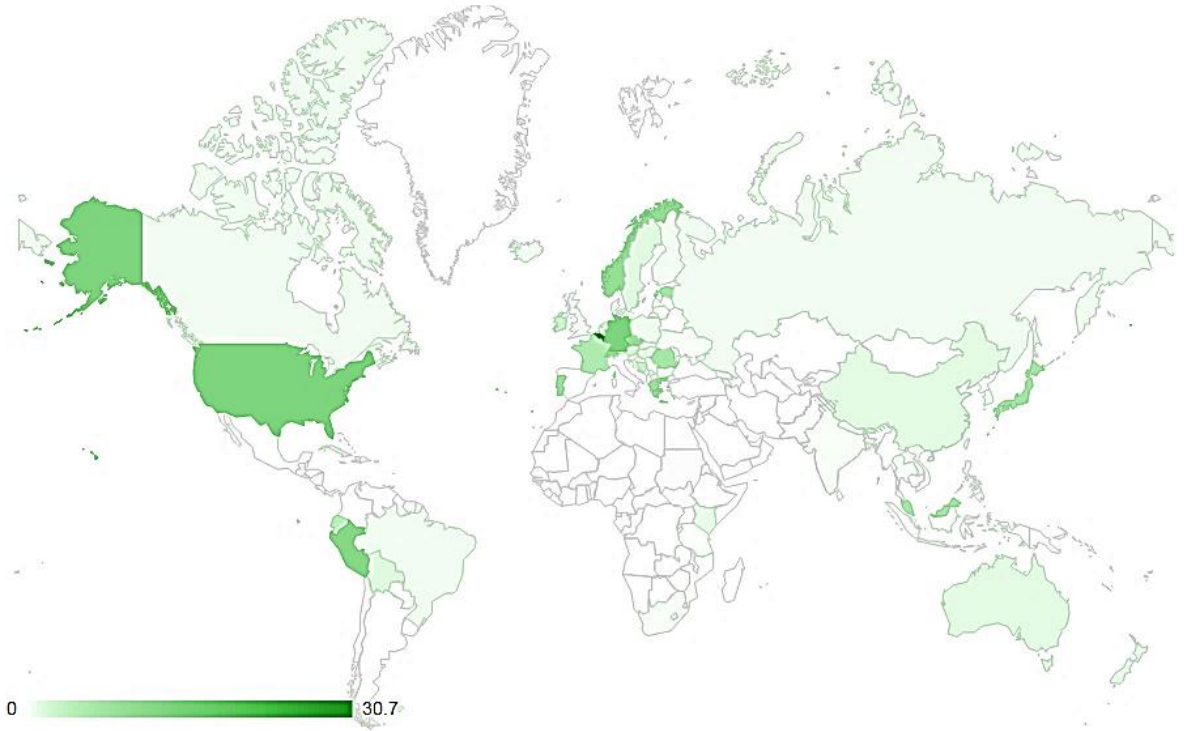
On IPv6, MLD and where the Internet is heading

Web Content Available over IPv6



From: http://6lab.cisco.com/stats/

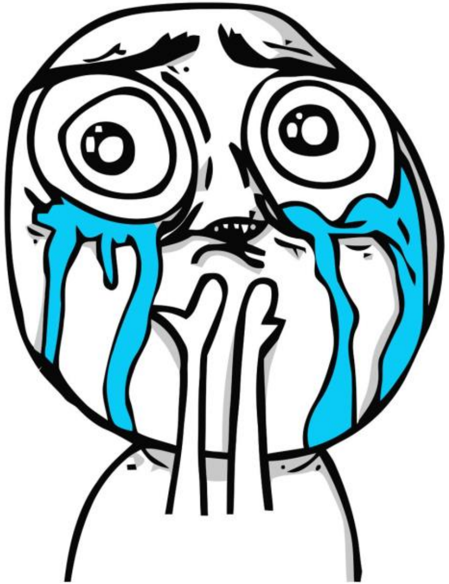
Users Accessing the Internet over IPv6



- Belgium: 37,28%
- Germany: 18,24%
- USA: 15,93%
- Japan: 10,83 %
- France: 5,46%

From: http://6lab.cisco.com/stats/

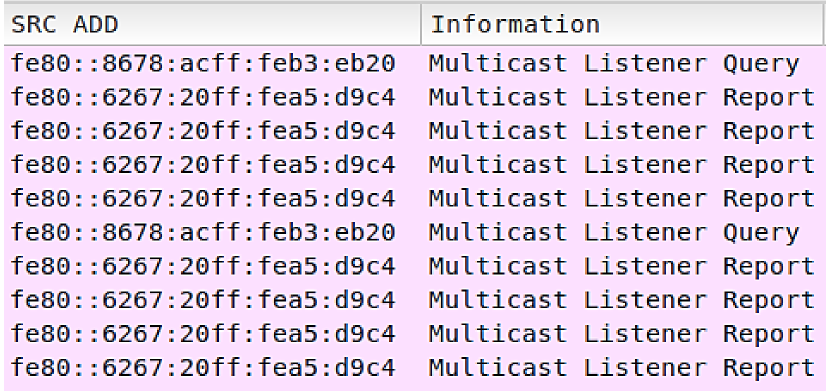
The IPv6 Vision



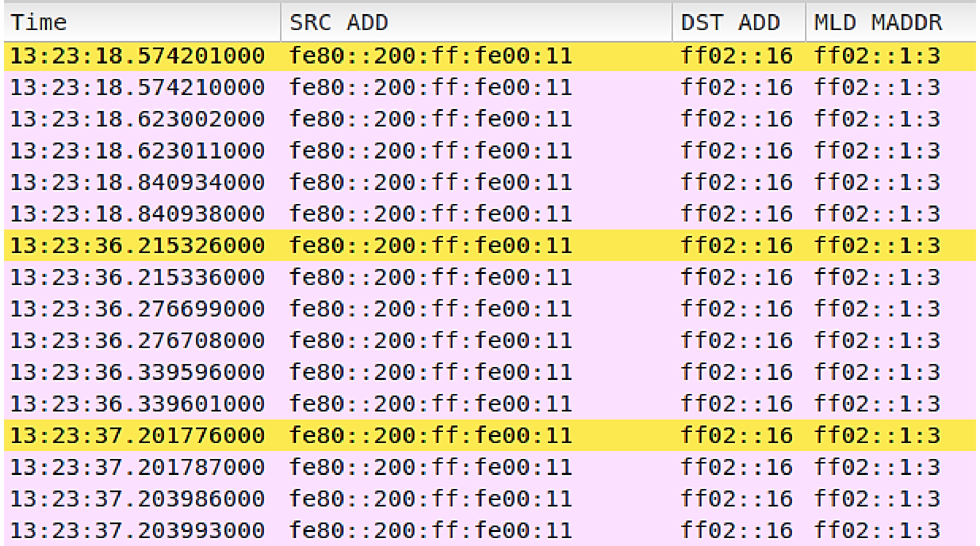
- Personal **appliances** are increasingly incorporating **networking capabilities**.
- Research and monitoring devices such as **sensor networks** are also looking towards IPv6 and multicasting.
- Concrete efforts are being directed towards materializing the "**Internet of Things**."

This All Sounds Great, but ...

- **Is** IPv6 **mature enough** for deployment and most important, **are we informed enough**?



SRC ADD	Information
fe80::8678:acff:feb3:eb20	Multicast Listener Query
fe80::6267:20ff:fea5:d9c4	Multicast Listener Report
fe80::6267:20ff:fea5:d9c4	Multicast Listener Report
fe80::6267:20ff:fea5:d9c4	Multicast Listener Report
fe80::6267:20ff:fea5:d9c4	Multicast Listener Report
fe80::8678:acff:feb3:eb20	Multicast Listener Query
fe80::6267:20ff:fea5:d9c4	Multicast Listener Report
fe80::6267:20ff:fea5:d9c4	Multicast Listener Report
fe80::6267:20ff:fea5:d9c4	Multicast Listener Report
fe80::6267:20ff:fea5:d9c4	Multicast Listener Report

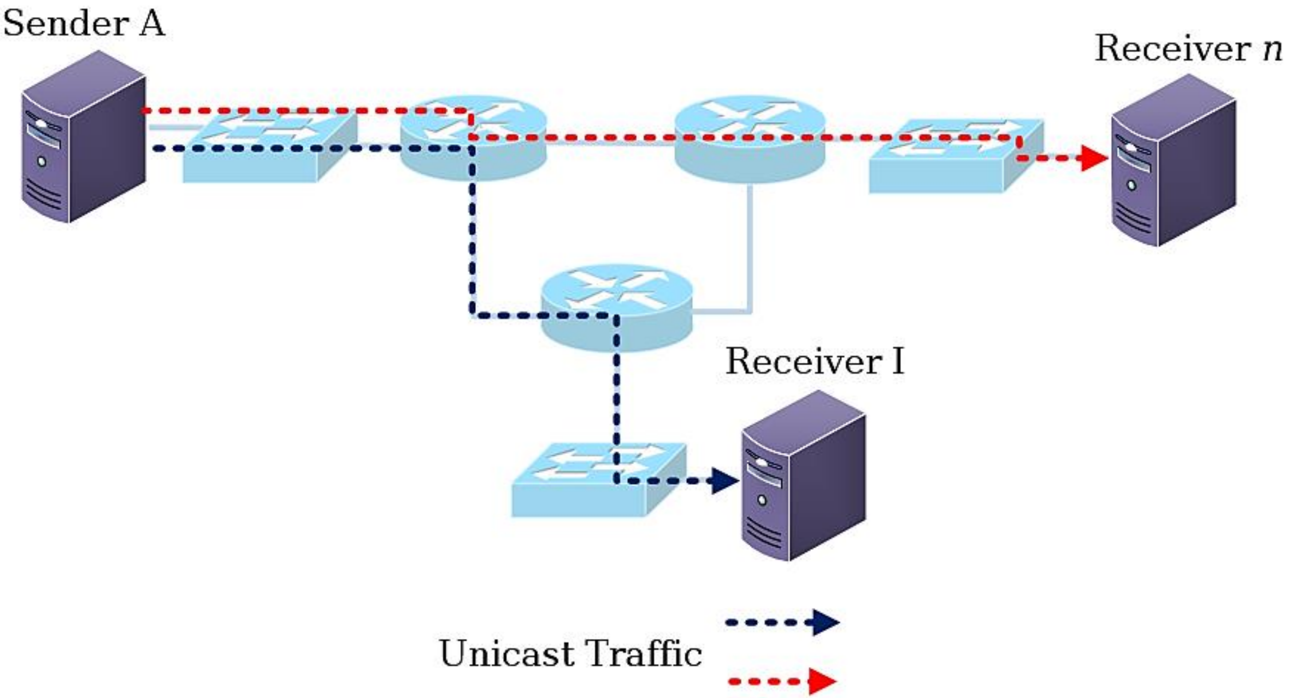


Time	SRC ADD	DST ADD	MLD MADDR
13:23:18.574201000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:18.574210000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:18.623002000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:18.623011000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:18.840934000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:18.840938000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:36.215326000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:36.215336000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:36.276699000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:36.276708000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:36.339596000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:36.339601000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:37.201776000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:37.201787000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:37.203986000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3
13:23:37.203993000	fe80::200:ff:fe00:11	ff02::16	ff02::1:3

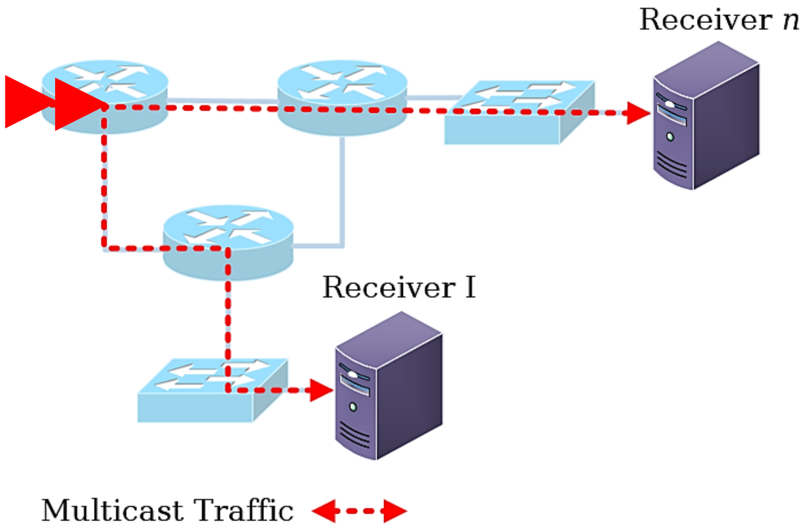
MLD, Every Protocol Has a Story

Hopefully, an entertaining one.

The Unicast Side of Things



Basic Concepts behind Multicasting



- The **sender** does **not require N data transmissions** to reach **N clients**.
- The **infrastructure** takes care of the **routing** and **replication**.
- The **sender sends** its data **once** and N clients receive it.
- **How** does the **infrastructure know** where the listeners are located?

Where is Multicast being Used? (I)

- The usual suspects:
  - Video-conferencing
  - IPTV
  - Sensor-networks
  - Monitoring and logging

Where is Multicast being Used? (II)

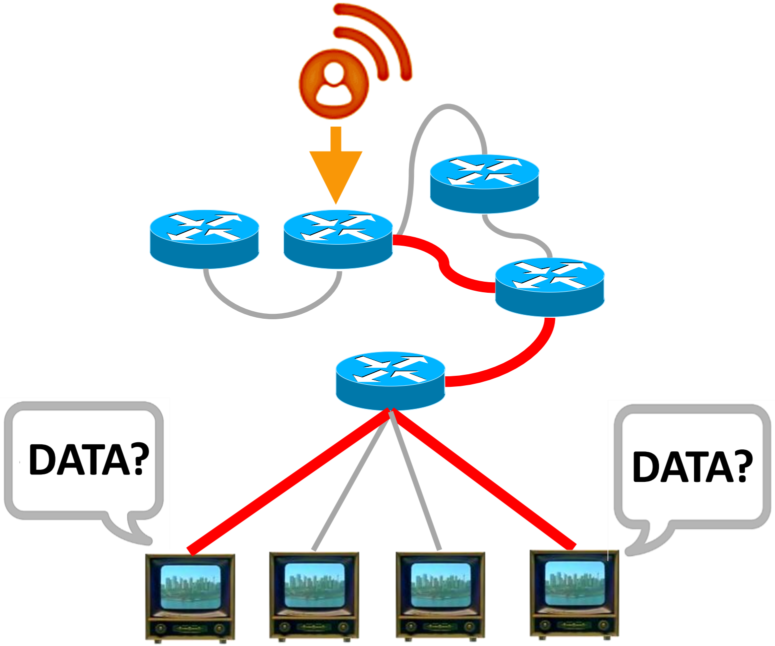
- **IPv6** has ‘**replaced**’ **broadcasting** with **multicasting** and multicast-related mechanisms
- **How**, you ask**?**

  By **mixing** the **Neighbor-Discovery** protocol, with **Solicited-Node** multicast **addresses** and **MLD**

MLD Will Make our Life much Easier

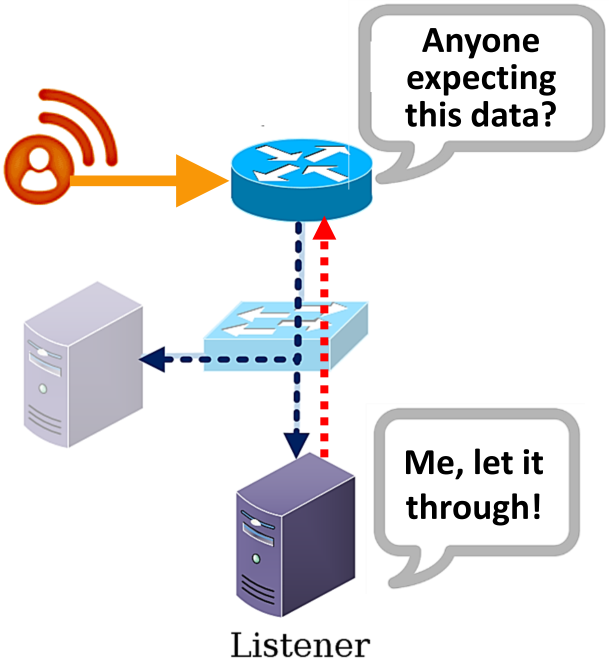
Well, at least it should …

The Initial Scenario



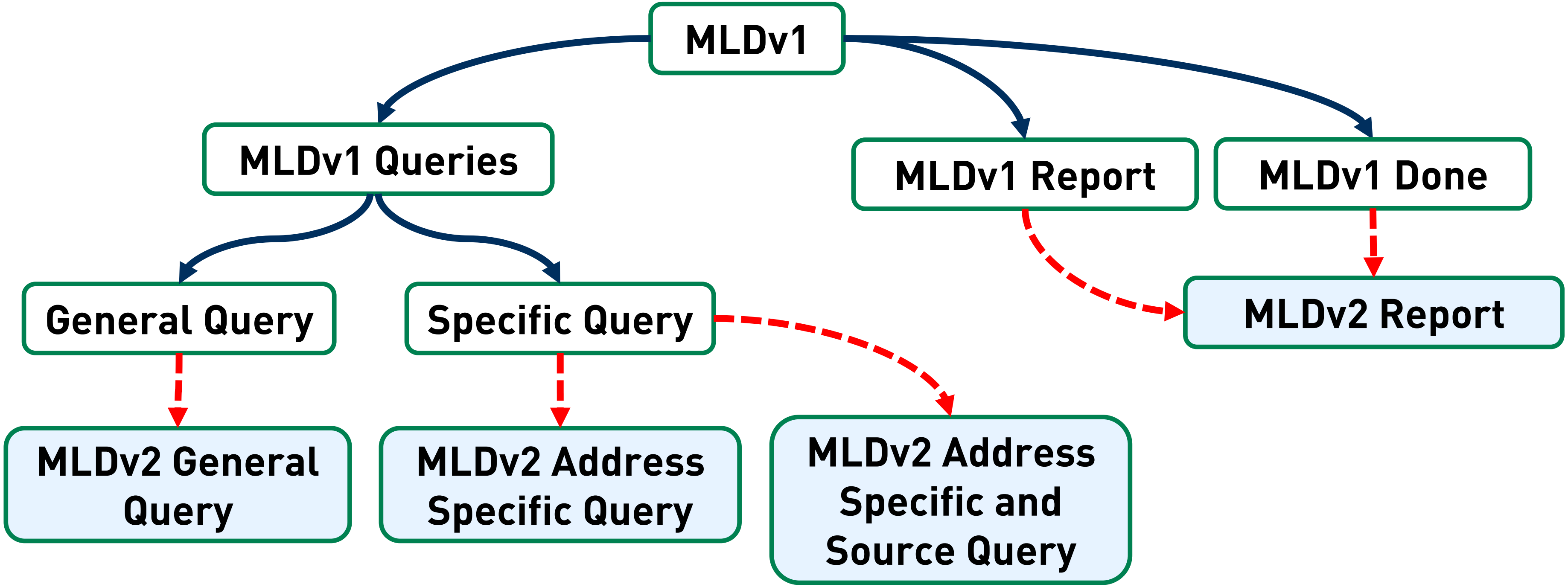
- IPv6 counterpart of IGMP
- MLD **enables** IPv6 routers to **discover** the presence of **multicast listeners** on its attached links
- Specifically, which **multicast addresses** are of **interest** to those neighboring nodes.
- **MLDv1** dates back to **1999** and was superseded by **MLDv2** in **2004**

Basic MLD Operation

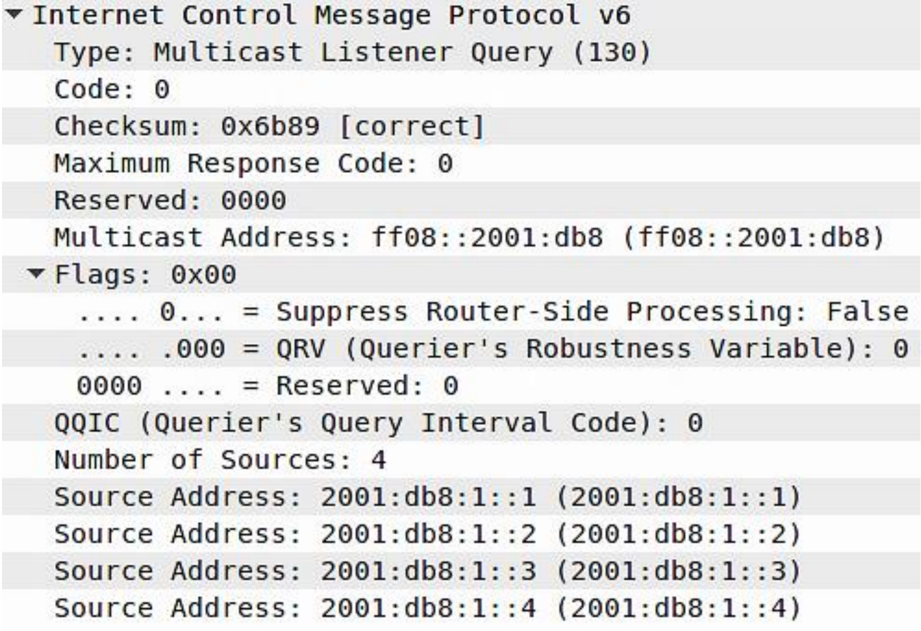


- The **Querier** sends **periodical Queries** to which Listeners with reportable addresses reply.
- The **Querier** does **not learn which** or **how many** clients are interested in which sources.
- The **Querier** uses reported information for deciding what **ingress data** to **forward**.

MLD Messages



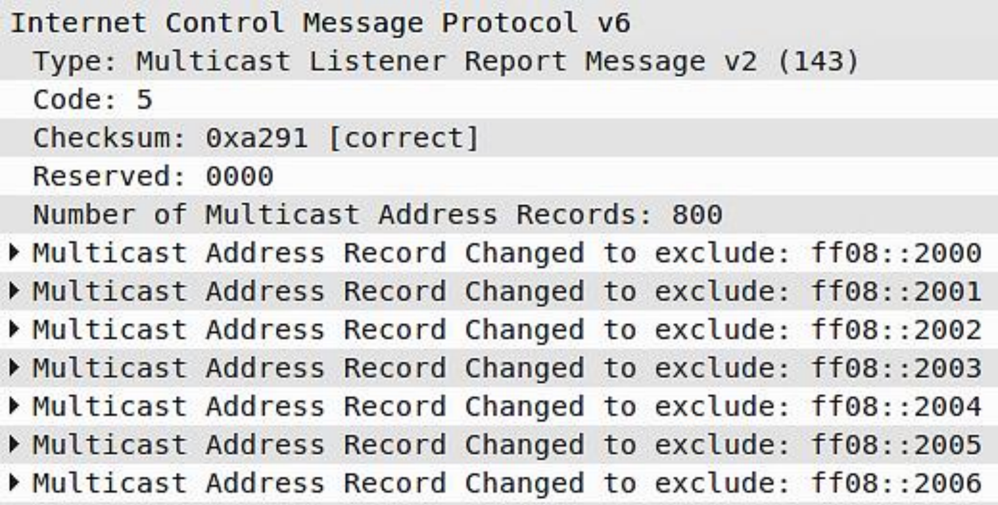
Querier-Sent Messages, Queries



```
▾ Internet Control Message Protocol v6
    Type: Multicast Listener Query (130)
    Code: 0
    Checksum: 0x6b89 [correct]
    Maximum Response Code: 0
    Reserved: 0000
    Multicast Address: ff08::2001:db8 (ff08::2001:db8)
  ▾ Flags: 0x00
      .... 0... = Suppress Router-Side Processing: False
      .... .000 = QRV (Querier's Robustness Variable): 0
      0000 .... = Reserved: 0
    QQIC (Querier's Query Interval Code): 0
    Number of Sources: 4
    Source Address: 2001:db8:1::1 (2001:db8:1::1)
    Source Address: 2001:db8:1::2 (2001:db8:1::2)
    Source Address: 2001:db8:1::3 (2001:db8:1::3)
    Source Address: 2001:db8:1::4 (2001:db8:1::4)
```

- Queries have ICMPv6 type 130
- General Queries are **sent to FF02::1**
- **Specific Queries** are sent **to** the multicast **address** being **queried**.

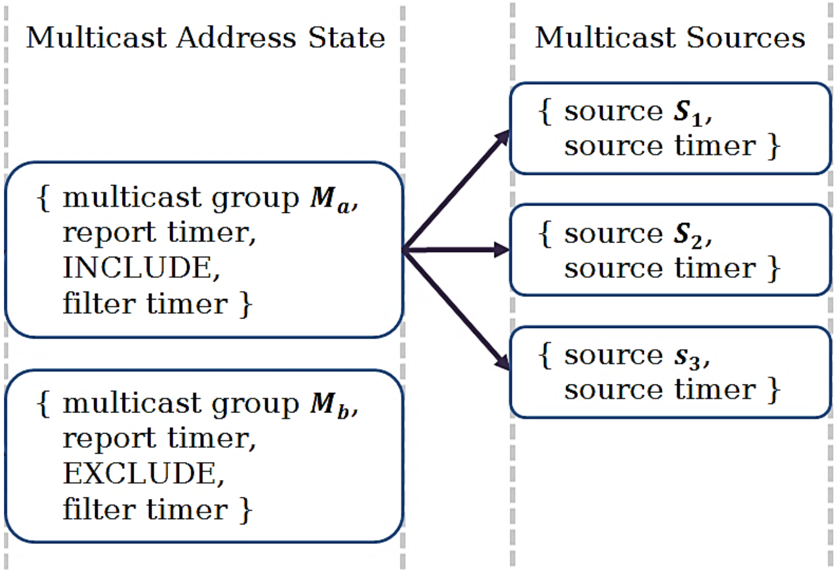
Listener-Sent Messages, Reports



```
Internet Control Message Protocol v6
  Type: Multicast Listener Report Message v2 (143)
  Code: 5
  Checksum: 0xa291 [correct]
  Reserved: 0000
  Number of Multicast Address Records: 800
▸ Multicast Address Record Changed to exclude: ff08::2000
▸ Multicast Address Record Changed to exclude: ff08::2001
▸ Multicast Address Record Changed to exclude: ff08::2002
▸ Multicast Address Record Changed to exclude: ff08::2003
▸ Multicast Address Record Changed to exclude: ff08::2004
▸ Multicast Address Record Changed to exclude: ff08::2005
▸ Multicast Address Record Changed to exclude: ff08::2006
```

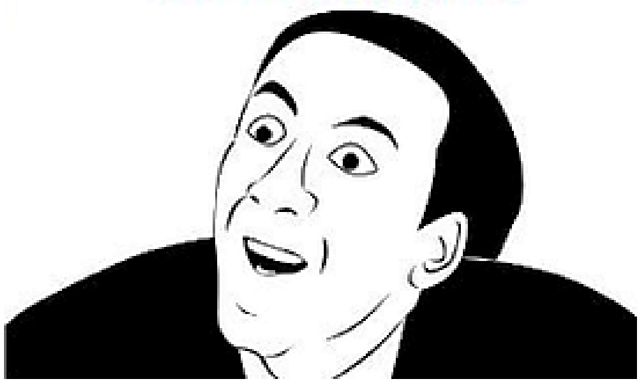
- MLDv2 Reports have ICMPv6 type 143
- Reports are **sent to FF02::16**
- Can report **several desired groups and sources simultaneously** in so-called MARs

Funky Note #1, State Keeping on Gateways


Multicast Address State
Multicast Sources
{ multicast group M_a, report timer, INCLUDE, filter timer }
{ multicast group M_b, report timer, EXCLUDE, filter timer }
{ source S_1, source timer }
{ source S_2, source timer }
{ source s_3, source timer }

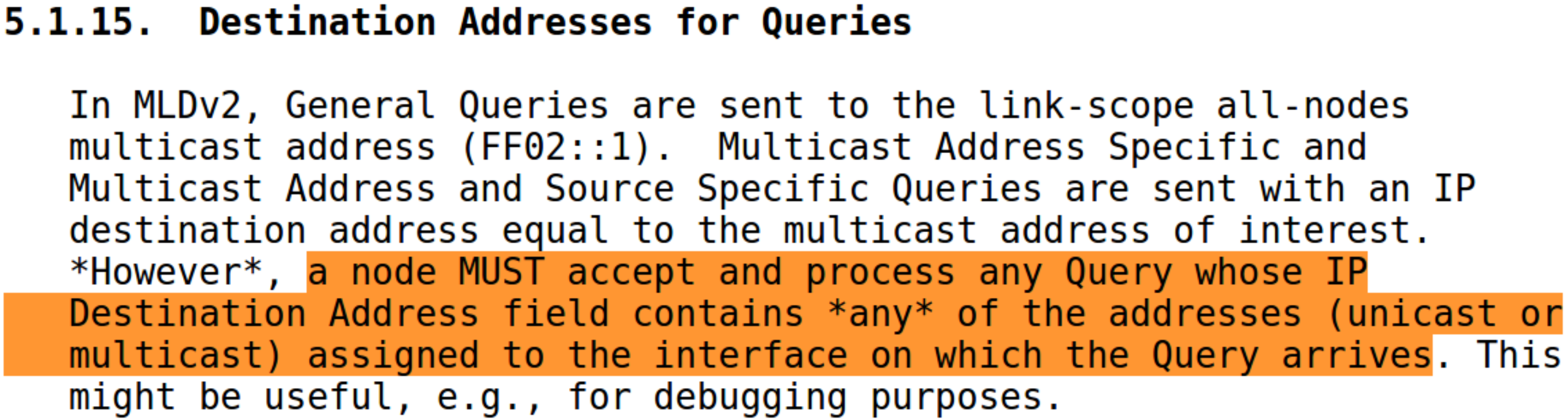
- A **gateway** must **keep** state regarding what "**kind**" of **content** must be **let through**
- **MLDv2 extended** state keeping mechanisms in order to also **keep track** of **accepted sources**
- **Timers** are **kept** per reported **group** and per accepted **source**

Funky Note #2, It Could've been Better



- MLD does **not learn** the **identity** or **number** of **Listeners** for a particular multicast group
- When there are multiple routers on the link the **Querier is elected** by **using** the **lowest IPv6 address** seen on a Query.
- In **MLDv1**, a client **may suppress** its **own report** when another node reports the same address.

Funky Note #3, One-to-one Communication

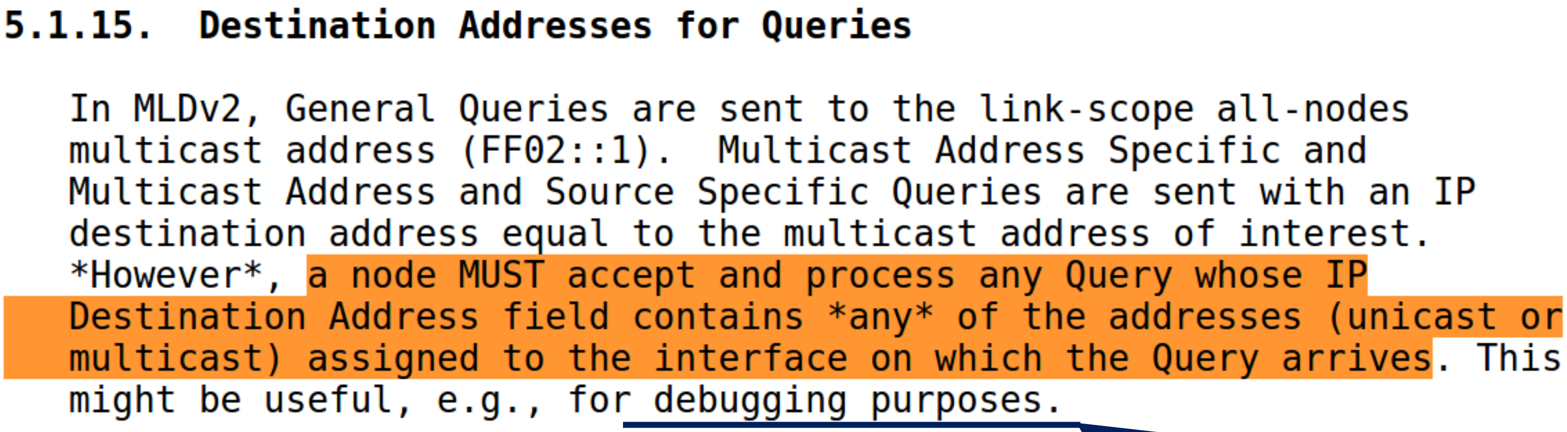


5.1.15. Destination Addresses for Queries

In MLDv2, General Queries are sent to the link-scope all-nodes multicast address (FF02::1). Multicast Address Specific and Multicast Address and Source Specific Queries are sent with an IP destination address equal to the multicast address of interest. *However*, a node MUST accept and process any Query whose IP Destination Address field contains *any* of the addresses (unicast or multicast) assigned to the interface on which the Query arrives. This might be useful, e.g., for debugging purposes.

RFC 3810

Funky Note #3, One-to-one Communication

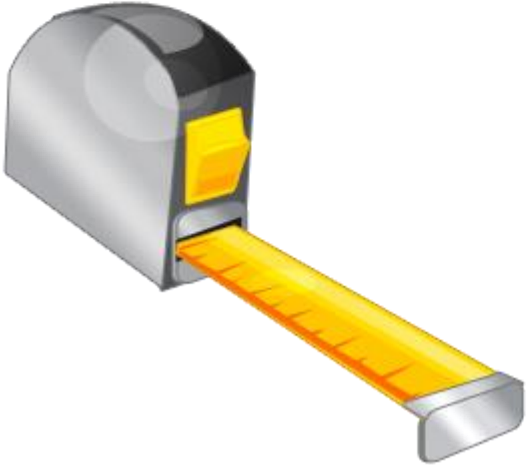


```
5.1.15.  Destination Addresses for Queries

   In MLDv2, General Queries are sent to the link-scope all-nodes
   multicast address (FF02::1).  Multicast Address Specific and
   Multicast Address and Source Specific Queries are sent with an IP
   destination address equal to the multicast address of interest.
   *However*, a node MUST accept and process any Query whose IP
   Destination Address field contains *any* of the addresses (unicast or
   multicast) assigned to the interface on which the Query arrives. This
   might be useful, e.g., for debugging purposes.
```

RFC 3810

MLDv2 Compared to MLDv1

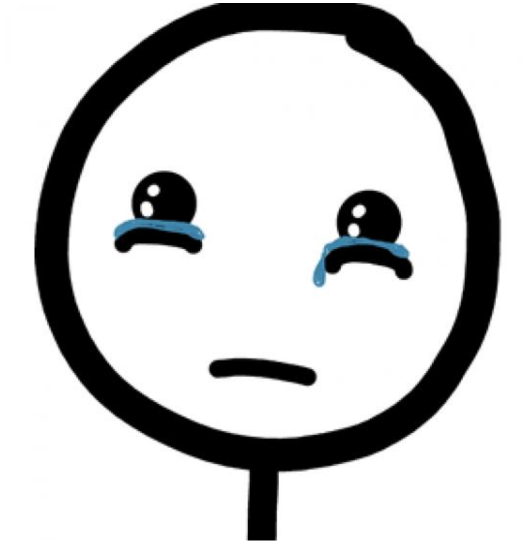


- MLDv2 **supports** for **source filtering**
- MLDv2 **Queries** and **Reports** can refer to **multiple sources**
- MLDv2 does **not** have a **suppression** mechanism nor **Done** messages
- **Groups** and **Sources** can be **included** or **excluded** and said **status** must be **tracked** by routers

There are Good News, Though

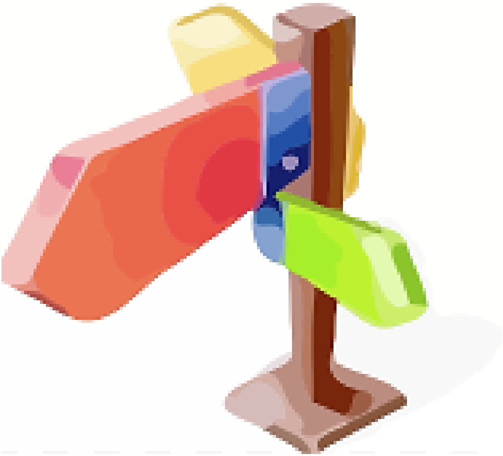
Well, it depends …

Up Until this Point, You don’t need MLD



- You only **need MLD** if you are operating **multicast applications**
- But, **needing** and **running isn’t** the **same**.
- **Except** for **OpenBSD** clients, **every** IPv6-capable **host** in your network **is running it**
- Great, **complexity for** the sake of **complexity**

So, Summarizing …

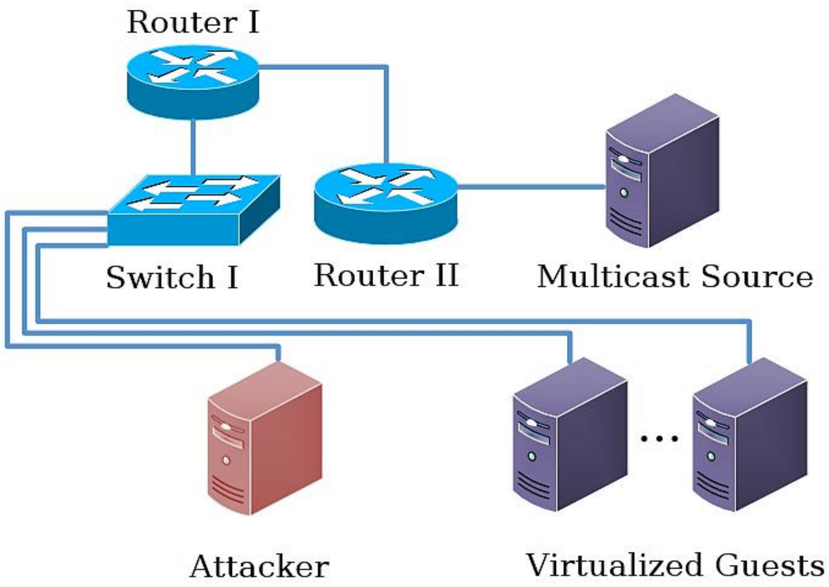


- You're running a **complex**, **resource-intensive protocol** although **you usually don't need it**
- It has some useful **"features"**
  - **Increases state-keeping** on the infrastructure side
  - One can **easily become** the **Querier**
  - One can **communicate** on a **one-to-one** basis
  - Some clients implement **Report suppression**
  - Forcing a **switch** to **MLDv1** is **trivial**
  - **Anything else?**

Playing with MLD

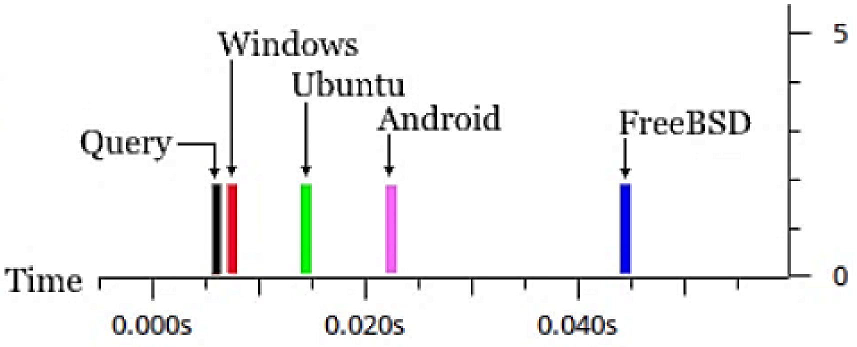
On how and what we tested

Test Environment



- **Cisco 1921** routers and **Cisco 2960s** switches
- Android, FreeBSD, Ubuntu and Windows virtualized guests
- **Tools**
  - Scapy
  - Chiron
  - Dizzy
  - THC IPv6 Toolkit
  - Wireshark

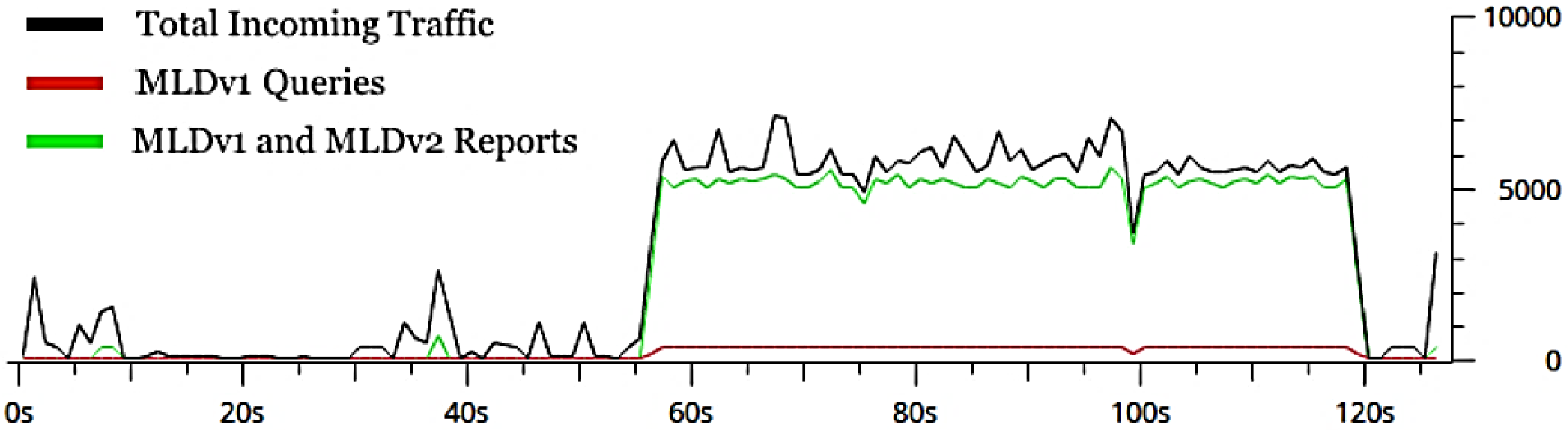
Clients' Response Time to MLD Queries



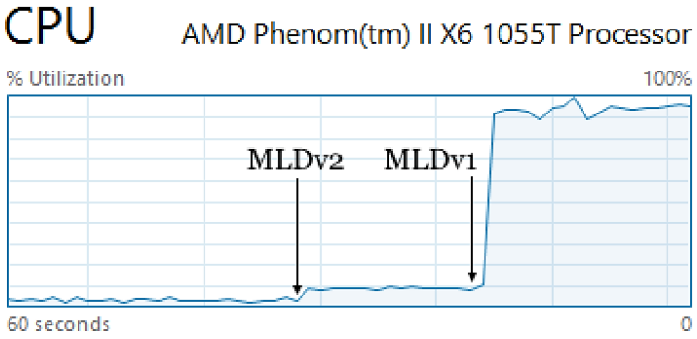
- **Most** clients **replied immediately** to Queries with Maximum Response Delay equal to zero
- **1,3kb/s** of MLDv1 Queries **become 49,8kb/s** on the Querier's side.
- Although the **RFC mentions** potential "ACK explosions" and **traffic amplification**, the clients just fire right away.

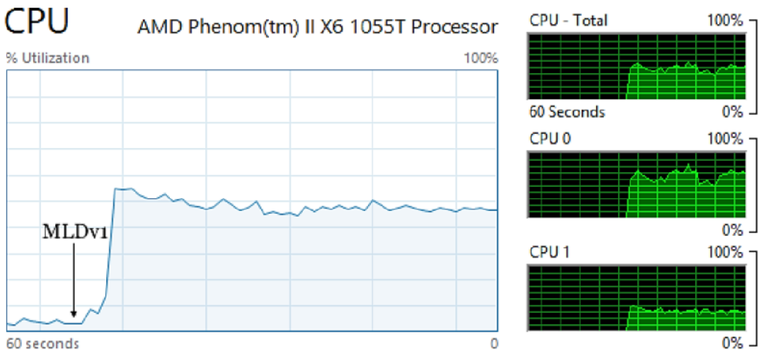
MLDv1 Traffic Amplification

- 1,3kb/s become 49,8kb/s on the router's side, **~3830%** the initial traffic



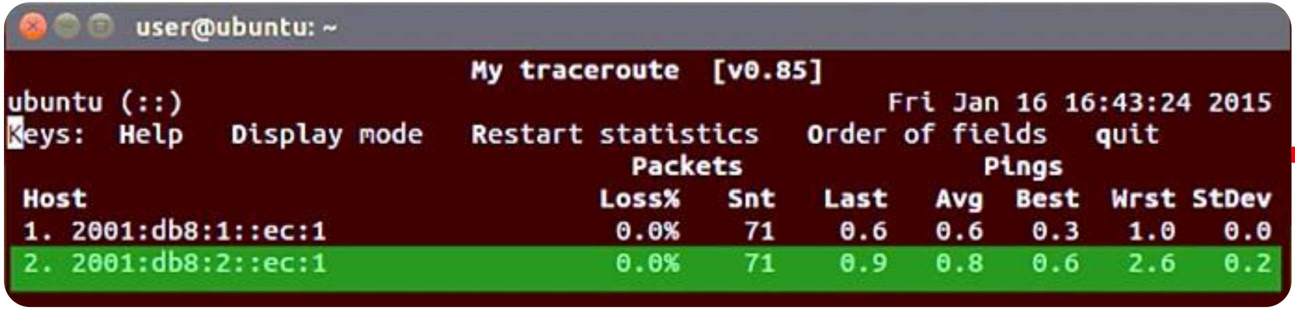
As Usual, Windows Must Behave Differently





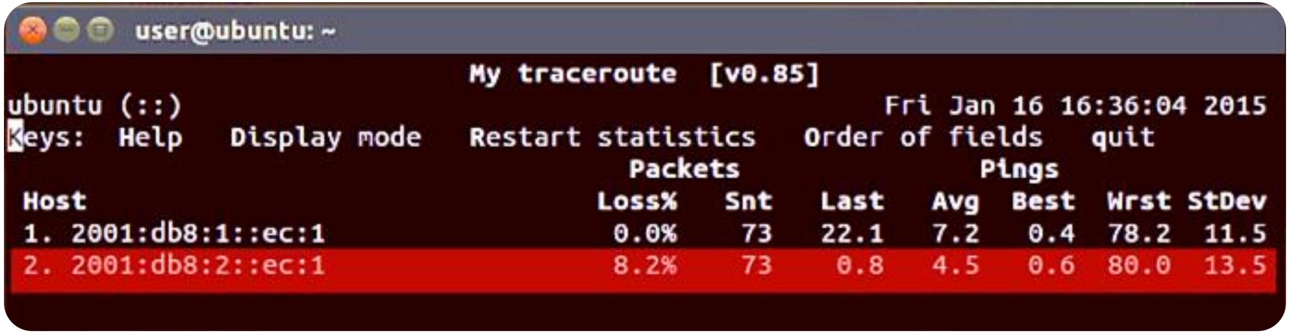
- In Windows 7 and 8.1 systems the process in charge of MLD + Interrupts processing can **consume up to one** processor **core**.

Big MLD Reports, Router Resource Depletion



user@ubuntu: ~

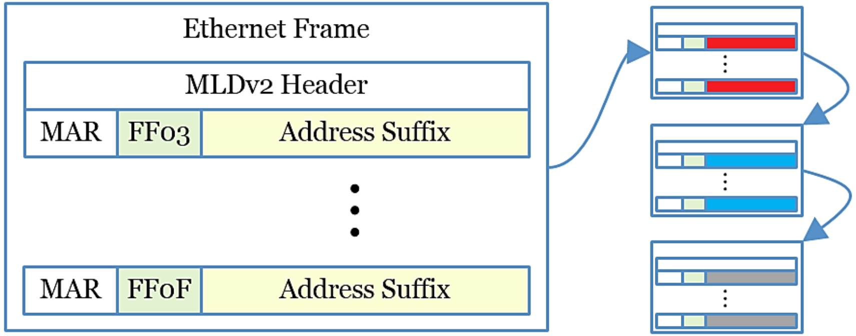
```
                              My traceroute  [v0.85]
ubuntu (::)                                             Fri Jan 16 16:43:24 2015
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                         Packets               Pings
 Host                                  Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 2001:db8:1::ec:1                    0.0%    71    0.6   0.6   0.3   1.0   0.0
 2. 2001:db8:2::ec:1                    0.0%    71    0.9   0.8   0.6   2.6   0.2
```



user@ubuntu: ~

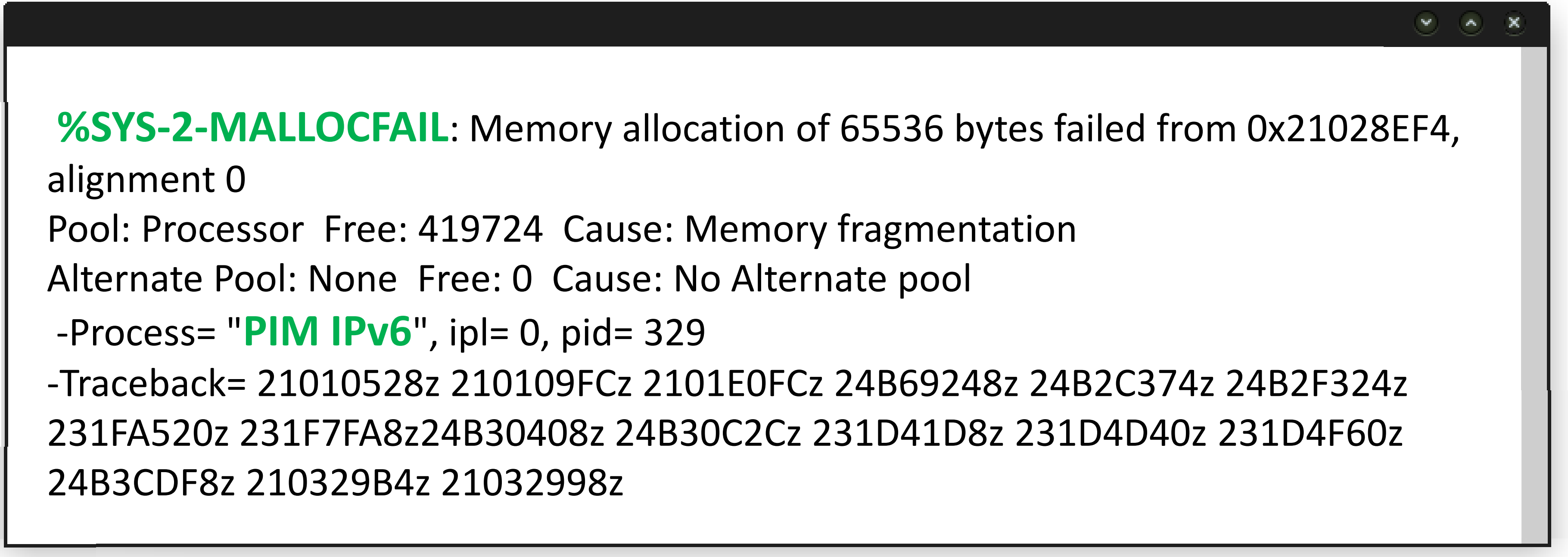
```
                              My traceroute  [v0.85]
ubuntu (::)                                             Fri Jan 16 16:36:04 2015
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                         Packets               Pings
 Host                                  Loss%   Snt   Last   Avg  Best  Wrst StDev
 1. 2001:db8:1::ec:1                    0.0%    73   22.1   7.2   0.4  78.2  11.5
 2. 2001:db8:2::ec:1                    8.2%    73    0.8   4.5   0.6  80.0  13.5
```

Big Reports Fill the Cache in about 30s



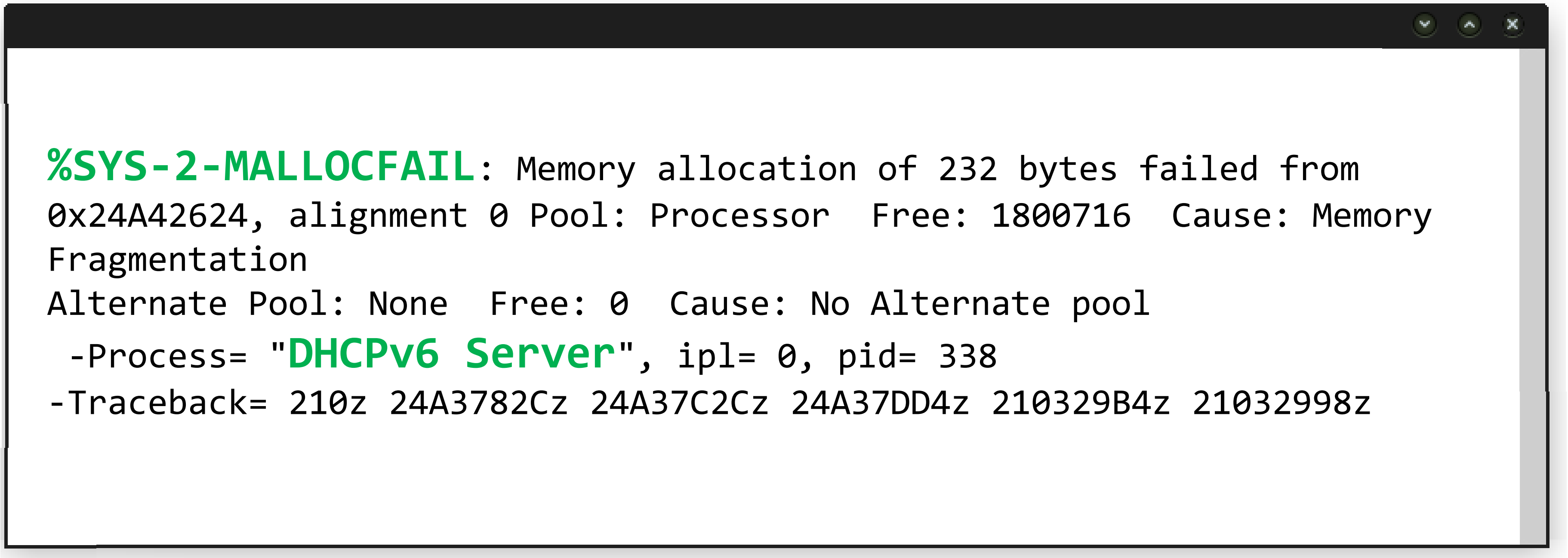
- Device **becomes unresponsive**, **packets** start being **dropped** and **latency** goes **up**
- Further **Listeners aren’t able** to **join** multicast groups since the table is effectively full
- Putting a **hard limit** on the number of entries **isn’t likely** to **help**

The PIM IPv6 Process Fails, Not that Bad



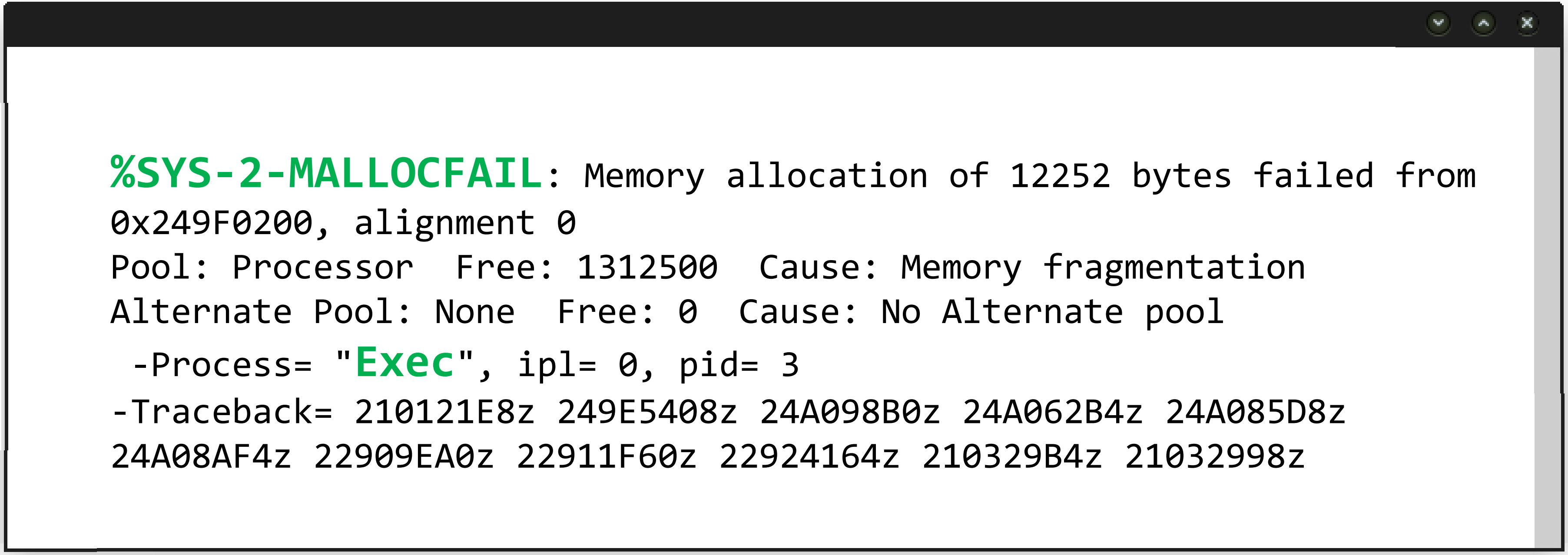
```
 %SYS-2-MALLOCFAIL: Memory allocation of 65536 bytes failed from 0x21028EF4,
alignment 0
Pool: Processor  Free: 419724  Cause: Memory fragmentation
Alternate Pool: None  Free: 0  Cause: No Alternate pool
 -Process= "PIM IPv6", ipl= 0, pid= 329
-Traceback= 21010528z 210109FCz 2101E0FCz 24B69248z 24B2C374z 24B2F324z
231FA520z 231F7FA8z24B30408z 24B30C2Cz 231D41D8z 231D4D40z 231D4F60z
24B3CDF8z 210329B4z 21032998z
```

IPv6 Addresses can’t be Leased, Hm



```
%SYS-2-MALLOCFAIL: Memory allocation of 232 bytes failed from
0x24A42624, alignment 0 Pool: Processor  Free: 1800716  Cause: Memory
Fragmentation
Alternate Pool: None  Free: 0  Cause: No Alternate pool
 -Process= "DHCPv6 Server", ipl= 0, pid= 338
-Traceback= 210z 24A3782Cz 24A37C2Cz 24A37DD4z 210329B4z 21032998z
```

Neither does SSH work, Oh Well ...

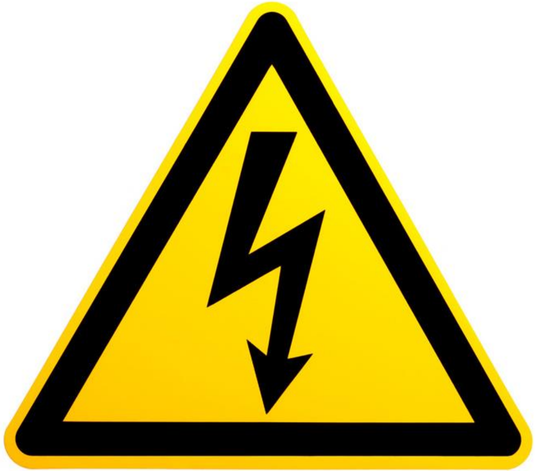


```
%SYS-2-MALLOCFAIL: Memory allocation of 12252 bytes failed from
0x249F0200, alignment 0
Pool: Processor  Free: 1312500  Cause: Memory fragmentation
Alternate Pool: None  Free: 0  Cause: No Alternate pool
 -Process= "Exec", ipl= 0, pid= 3
-Traceback= 210121E8z 249E5408z 24A098B0z 24A062B4z 24A085D8z
24A08AF4z 22909EA0z 22911F60z 22924164z 210329B4z 21032998z
```

Demo

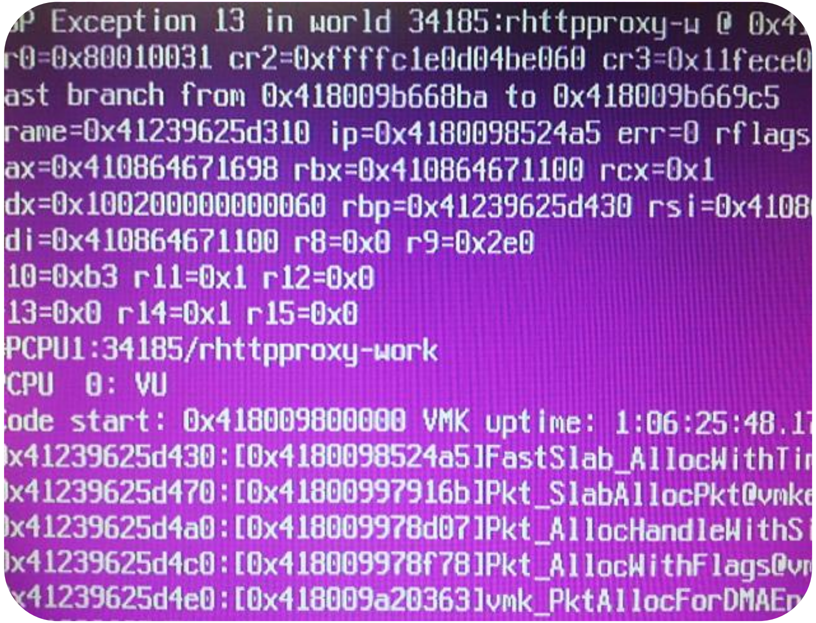
Overloading network infrastructure via MLD

Just Useless Defaults by Cisco



- **156.500** MLD entries cause the routers to malfunction.
- **Who** and what for **needs 150k** MLD **entries**?
- So much for useful defaults, **limit MLD state**!
- Not limited to the listed devices, **similar behavior** was **observed** with **ASR1000s**

Drivers, Always Drivers

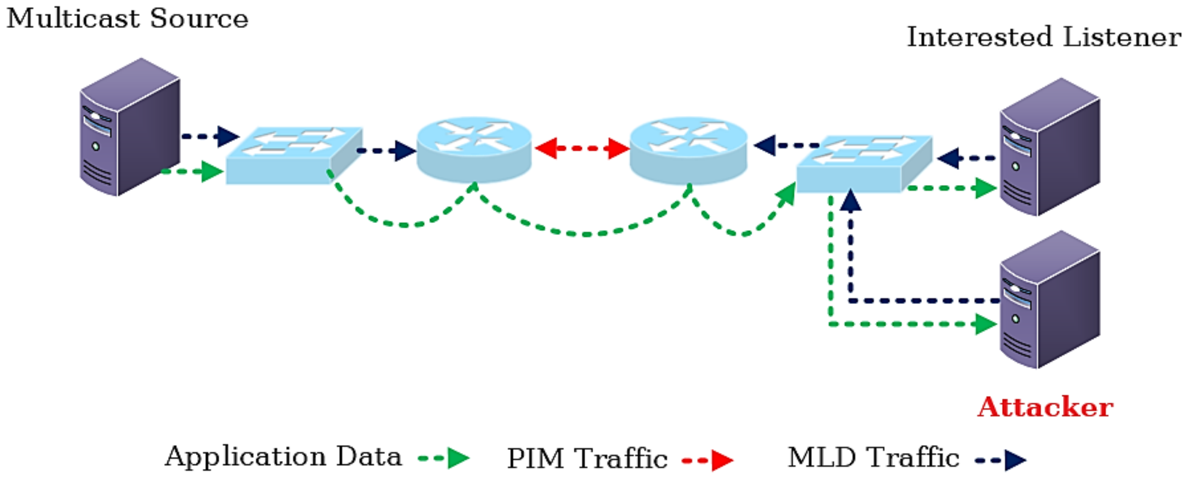


```
P Exception 13 in world 34185:rhttpproxy-w @ 0x4
r0=0x80010031 cr2=0xffffc1e0d04be060 cr3=0x11fece0
ast branch from 0x418009b668ba to 0x418009b669c5
rame=0x41239625d310 ip=0x4180098524a5 err=0 rflags
ax=0x410864671698 rbx=0x410864671100 rcx=0x1
dx=0x1002000000000060 rbp=0x41239625d430 rsi=0x4108
di=0x410864671100 r8=0x0 r9=0x2e0
10=0xb3 r11=0x1 r12=0x0
13=0x0 r14=0x1 r15=0x0
PCPU1:34185/rhttpproxy-work
CPU  0: VU
ode start: 0x418009800000 VMK uptime: 1:06:25:48.1
x41239625d430:[0x4180098524a5]FastSlab_AllocWithTi
x41239625d470:[0x41800997916b]Pkt_SlabAllocPkt@vmke
x41239625d4a0:[0x418009978d07]Pkt_AllocHandleWithS
x41239625d4c0:[0x418009978f78]Pkt_AllocWithFlags@vm
x41239625d4e0:[0x418009a20363]vmk_PktAllocForDMAEn
```

- **VMWare ESXi 5.5. crashes** when high rates of MLD traffic are received on an **Intel 82573L** network interface

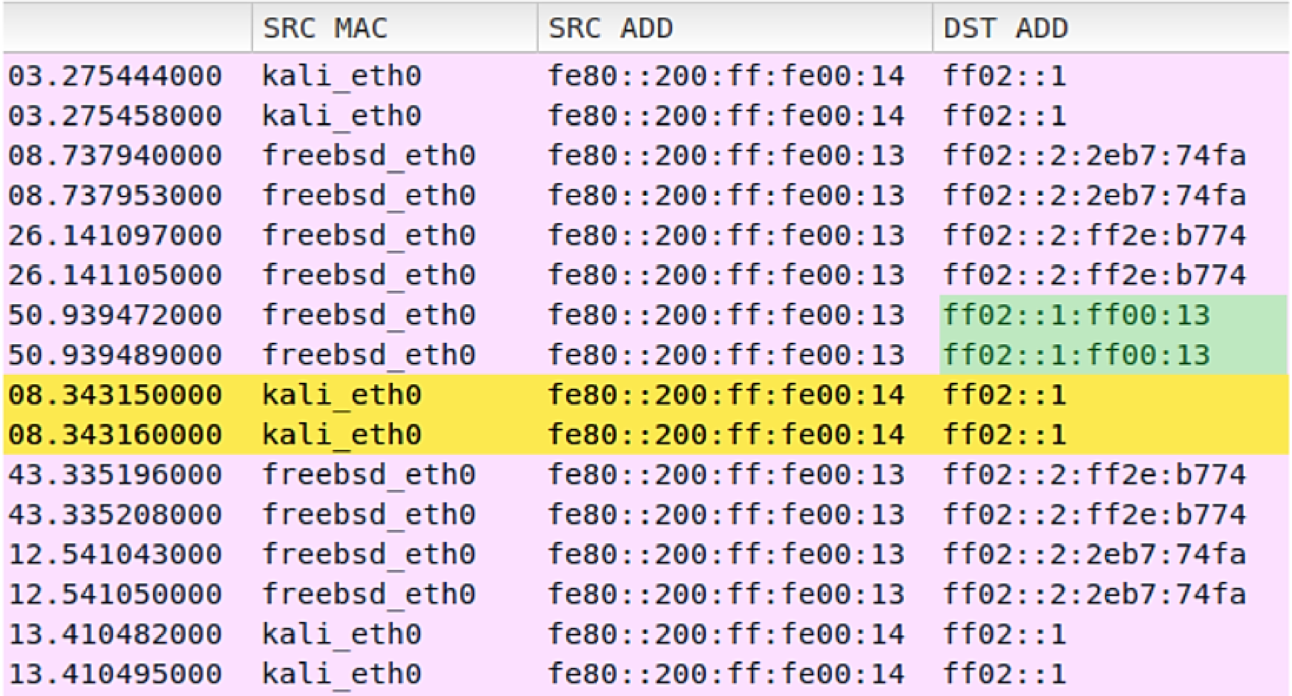
- **0-Day'ish**, relevant only as **DoS**, though.

Let's not Forget the Scenario



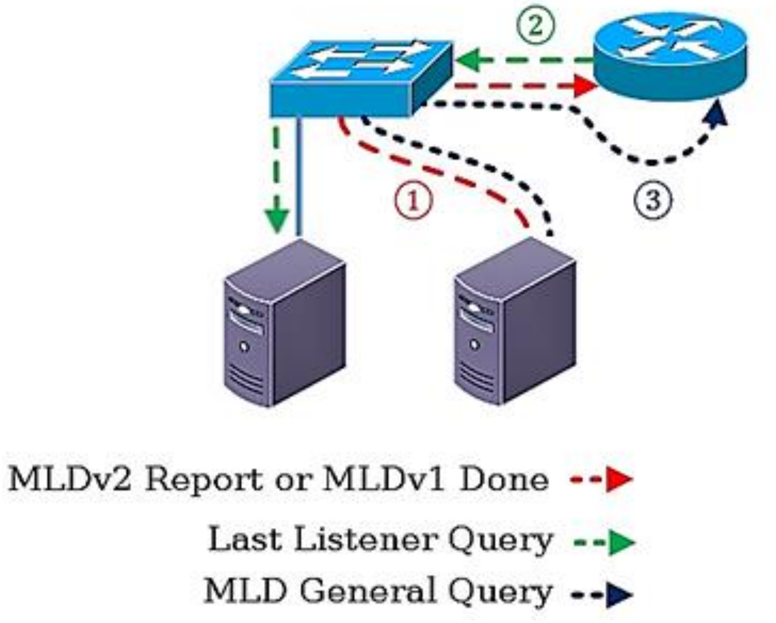
- MLD messages are **processed regardless** of **destination** address
- A malicious user can **trivially become** the **Querier** on the link

Force MLDv1 Usage and Reports Suppression



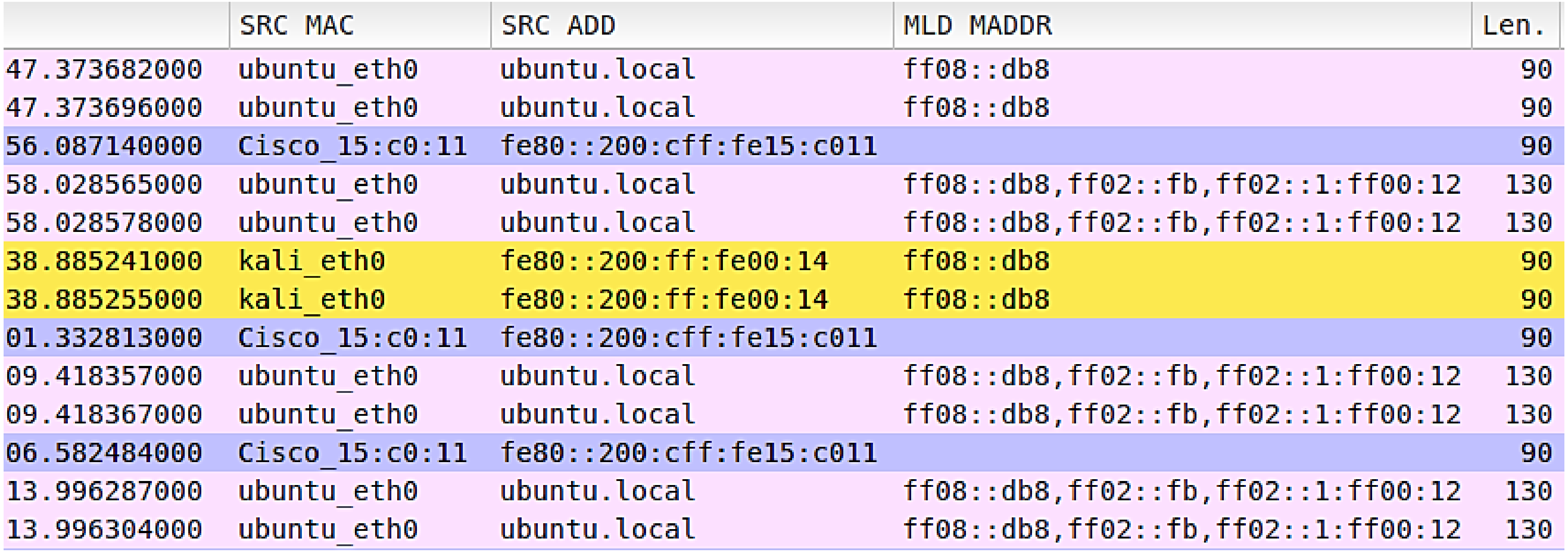
	SRC MAC	SRC ADD	DST ADD
03.275444000	kali_eth0	fe80::200:ff:fe00:14	ff02::1
03.275458000	kali_eth0	fe80::200:ff:fe00:14	ff02::1
08.737940000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::2:2eb7:74fa
08.737953000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::2:2eb7:74fa
26.141097000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::2:ff2e:b774
26.141105000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::2:ff2e:b774
50.939472000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::1:ff00:13
50.939489000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::1:ff00:13
08.343150000	kali_eth0	fe80::200:ff:fe00:14	ff02::1
08.343160000	kali_eth0	fe80::200:ff:fe00:14	ff02::1
43.335196000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::2:ff2e:b774
43.335208000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::2:ff2e:b774
12.541043000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::2:2eb7:74fa
12.541050000	freebsd_eth0	fe80::200:ff:fe00:13	ff02::2:2eb7:74fa
13.410482000	kali_eth0	fe80::200:ff:fe00:14	ff02::1
13.410495000	kali_eth0	fe80::200:ff:fe00:14	ff02::1

The Last Call for Drinks, Last-Listener-Queries



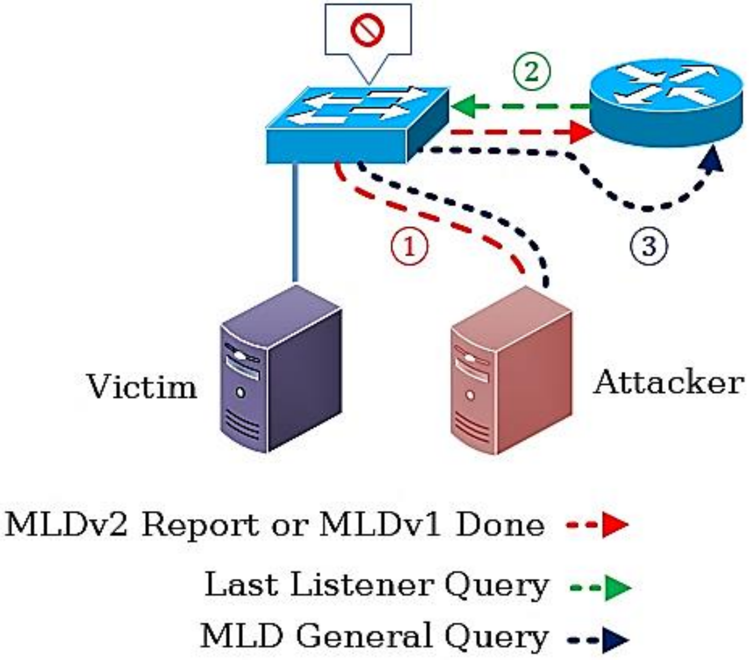
- **Last-Listener-Queries** are **sent** by the Querier **when** a Listener expresses its **lack of interest** in certain traffic
- Is **sent** as a **Specific-Query** to the multicast address which is being queried
- An **attacker** can **become** the **Querier**, **leave** a **group** on behalf of a client and **fake** a **Last-Listener-Query**

However, Something was Missing



	SRC MAC	SRC ADD	MLD MADDR	Len.
47.373682000	ubuntu_eth0	ubuntu.local	ff08::db8	90
47.373696000	ubuntu_eth0	ubuntu.local	ff08::db8	90
56.087140000	Cisco_15:c0:11	fe80::200:cff:fe15:c011		90
58.028565000	ubuntu_eth0	ubuntu.local	ff08::db8,ff02::fb,ff02::1:ff00:12	130
58.028578000	ubuntu_eth0	ubuntu.local	ff08::db8,ff02::fb,ff02::1:ff00:12	130
38.885241000	kali_eth0	fe80::200:ff:fe00:14	ff08::db8	90
38.885255000	kali_eth0	fe80::200:ff:fe00:14	ff08::db8	90
01.332813000	Cisco_15:c0:11	fe80::200:cff:fe15:c011		90
09.418357000	ubuntu_eth0	ubuntu.local	ff08::db8,ff02::fb,ff02::1:ff00:12	130
09.418367000	ubuntu_eth0	ubuntu.local	ff08::db8,ff02::fb,ff02::1:ff00:12	130
06.582484000	Cisco_15:c0:11	fe80::200:cff:fe15:c011		90
13.996287000	ubuntu_eth0	ubuntu.local	ff08::db8,ff02::fb,ff02::1:ff00:12	130
13.996304000	ubuntu_eth0	ubuntu.local	ff08::db8,ff02::fb,ff02::1:ff00:12	130

In Reality, It's Even Easier



- Cisco 1921 devices **do not forward Last-Listener-Queries**
- To prevent a client from receiving certain multicast data-flows one **simply** has to **spoof** an **MLD Report** or **Done** message
- The interested **Listener won't have** the **chance** to **reply** since, well, the switch doesn't forward the query

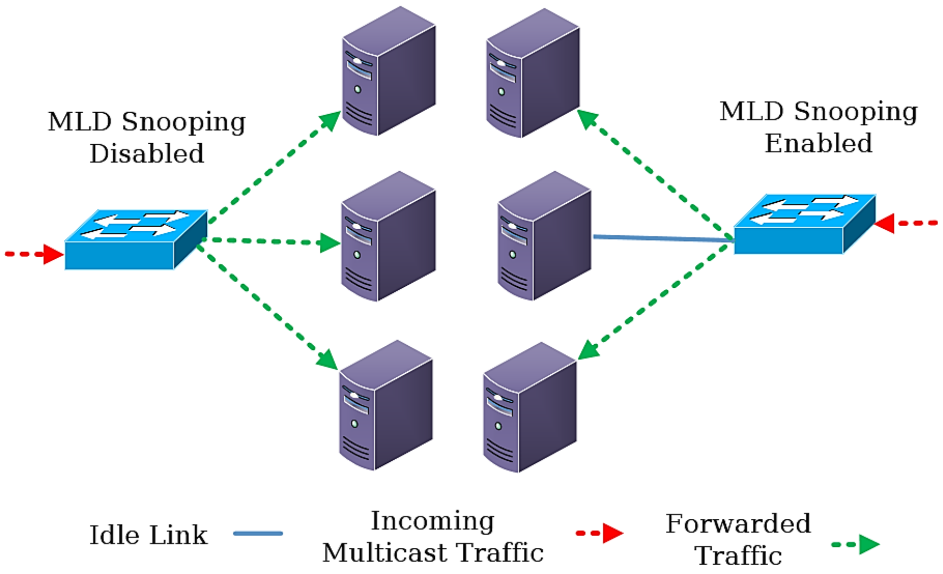
Demo

So, management wants video-conferencing?

But Someone had to Add Something else ...

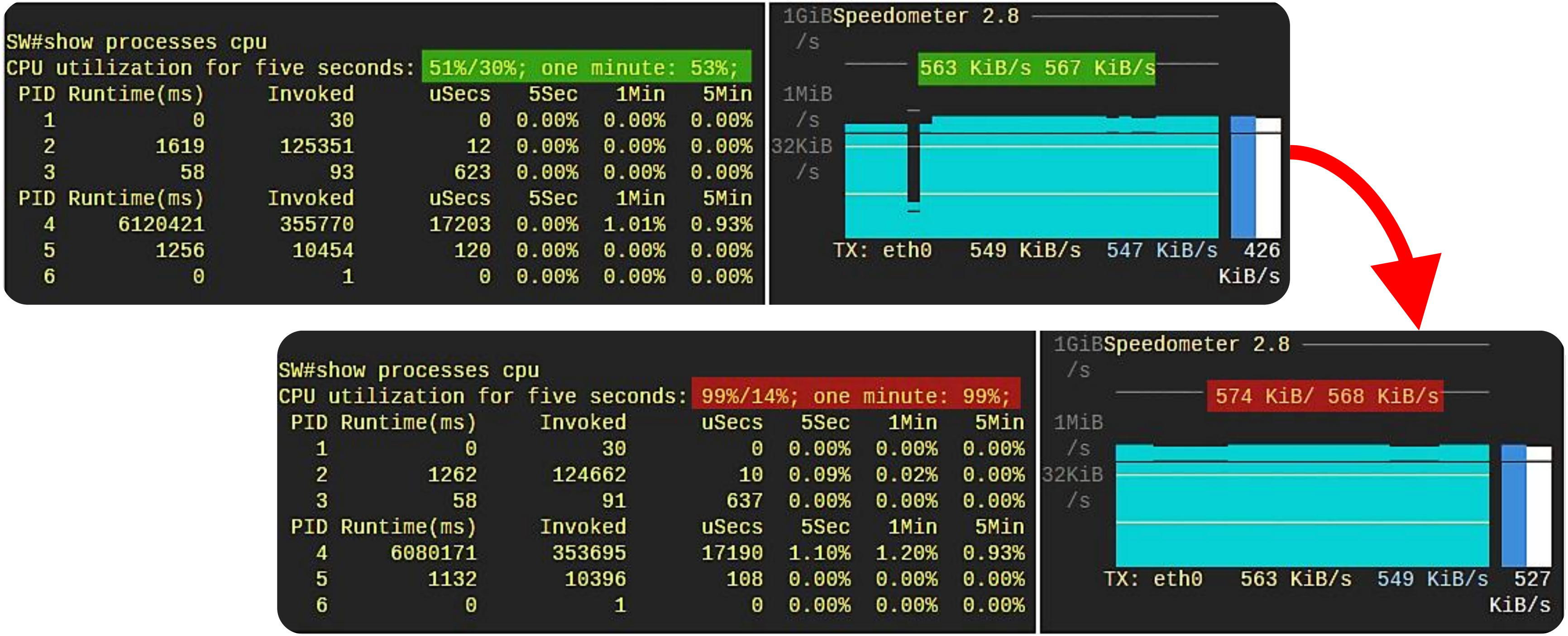
Because there is always room for more complexity

MLD-Snooping … Yes, More Complexity!



- Is **not standardized**
- There's an **informational RFC**
- Brings **state-keeping** behavior to the **switches**
- Considered by **RFC3810** and **others** where **ND** is specified.

Of Course, Nothing Could Go Wrong



```
SW#show processes cpu
CPU utilization for five seconds: 51%/30%; one minute: 53%;
 PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min
   1           0          30          0  0.00%  0.00%  0.00%
   2        1619      125351         12  0.00%  0.00%  0.00%
   3          58          93        623  0.00%  0.00%  0.00%
 PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min
   4     6120421      355770      17203  0.00%  1.01%  0.93%
   5        1256       10454        120  0.00%  0.00%  0.00%
   6           0           1          0  0.00%  0.00%  0.00%
```

```
1GiBSpeedometer 2.8
 /s
          563 KiB/s 567 KiB/s
1MiB
 /s
32KiB
 /s
     TX: eth0    549 KiB/s   547 KiB/s   426
                                         KiB/s
```

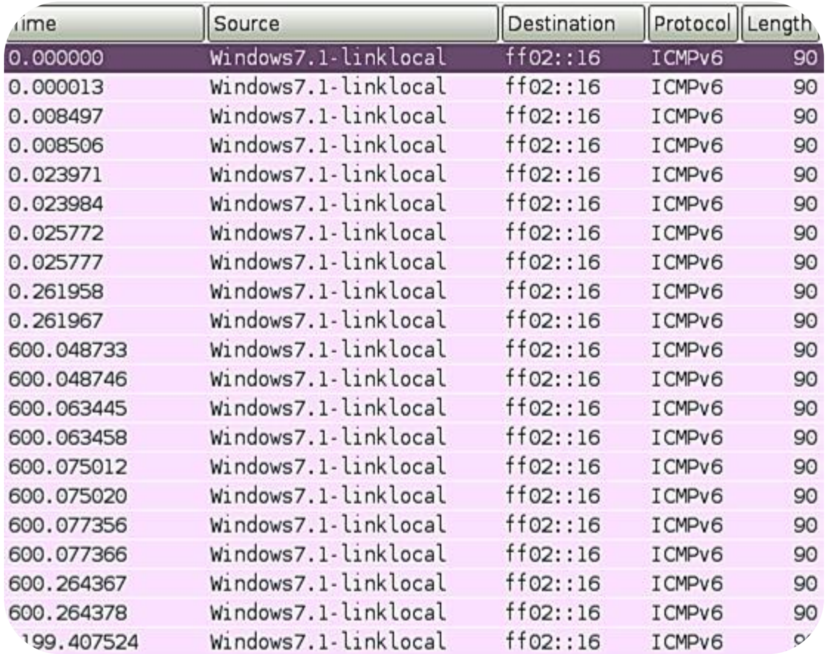
```
SW#show processes cpu
CPU utilization for five seconds: 99%/14%; one minute: 99%;
 PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min
   1           0          30          0  0.00%  0.00%  0.00%
   2        1262      124662         10  0.09%  0.02%  0.00%
   3          58          91        637  0.00%  0.00%  0.00%
 PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min
   4     6080171      353695      17190  1.10%  1.20%  0.93%
   5        1132       10396        108  0.00%  0.00%  0.00%
   6           0           1          0  0.00%  0.00%  0.00%
```

```
1GiBSpeedometer 2.8
 /s
          574 KiB/ 568 KiB/s
1MiB
 /s
32KiB
 /s
     TX: eth0    563 KiB/s   549 KiB/s   527
                                         KiB/s
```

Anything else?

One last minor detail

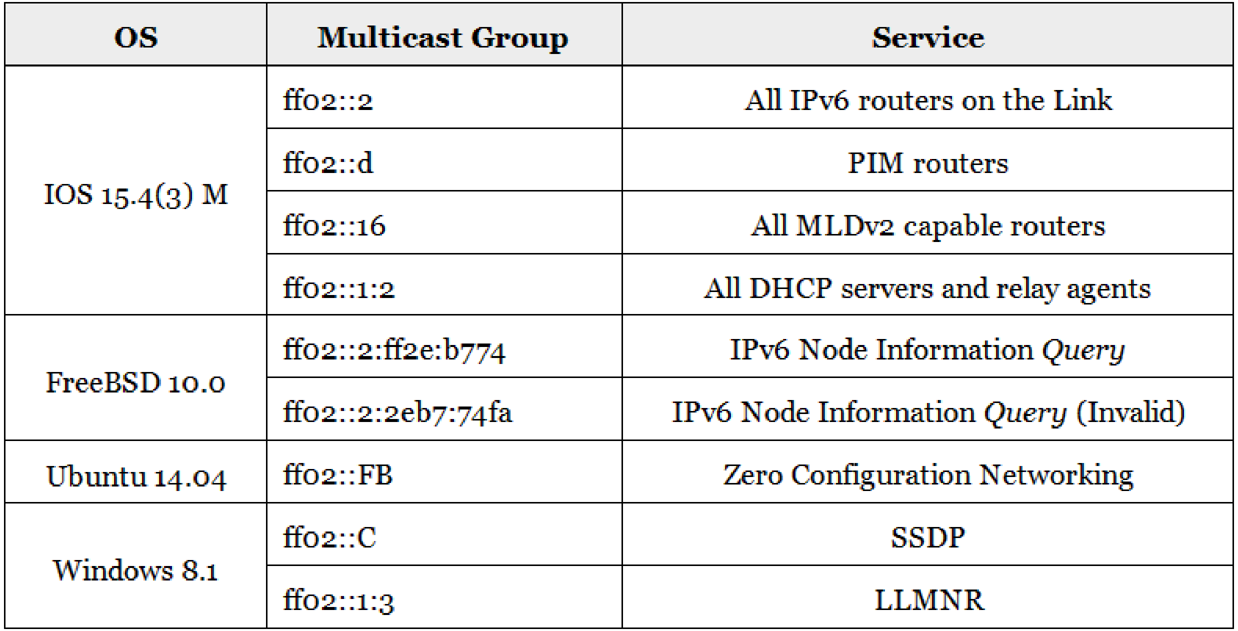
Trivial Host Discovery and Fingerprinting (I)



ime	Source	Destination	Protocol	Length
0.000000	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.000013	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.008497	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.008506	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.023971	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.023984	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.025772	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.025777	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.261958	Windows7.1-linklocal	ff02::16	ICMPv6	90
0.261967	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.048733	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.048746	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.063445	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.063458	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.075012	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.075020	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.077356	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.077366	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.264367	Windows7.1-linklocal	ff02::16	ICMPv6	90
600.264378	Windows7.1-linklocal	ff02::16	ICMPv6	90
99.407524	Windows7.1-linklocal	ff02::16	ICMPv6	9

- MLD is the perfect protocol for the job.
- Pre-enabled in Windows, Linux and FreeBSD
- Reports are sent even before the ND Process starts
- Hosts must respond to Queries
- Works even when responses to ICMPv6 are disabled

Trivial Host Discovery and Fingerprinting (II)



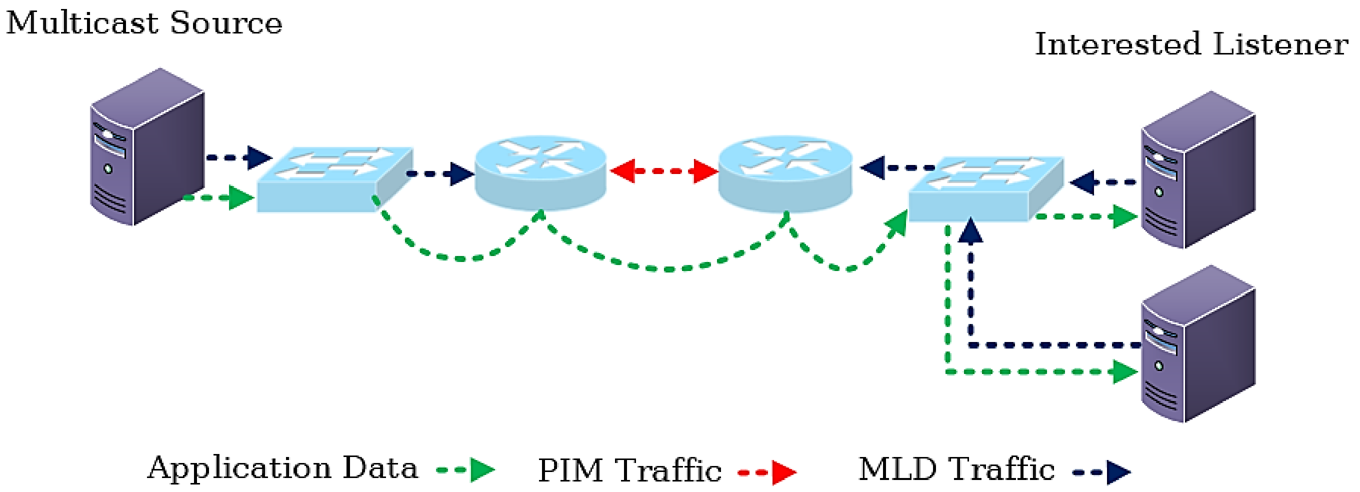
OS	Multicast Group	Service
IOS 15.4(3) M	ff02::2	All IPv6 routers on the Link
	ff02::d	PIM routers
	ff02::16	All MLDv2 capable routers
	ff02::1:2	All DHCP servers and relay agents
FreeBSD 10.0	ff02::2:ff2e:b774	IPv6 Node Information *Query*
	ff02::2:2eb7:74fa	IPv6 Node Information *Query* (Invalid)
Ubuntu 14.04	ff02::FB	Zero Configuration Networking
Windows 8.1	ff02::C	SSDP
	ff02::1:3	LLMNR

Is MLD really not used at all?

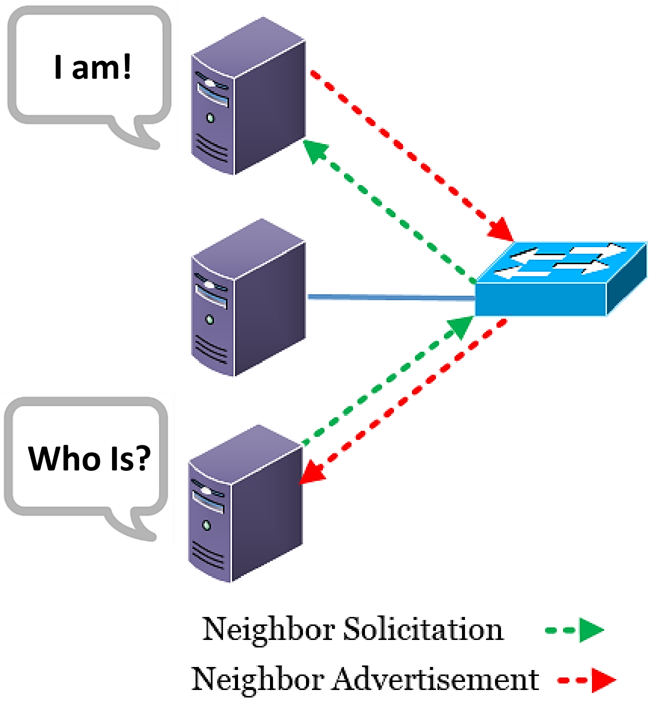
Well, it’s more complex than that …

Of Course, Multicast Applications

- **Whether intra or inter-domain**, you wouldn't want all those video streams to get broadcasted like crazy.

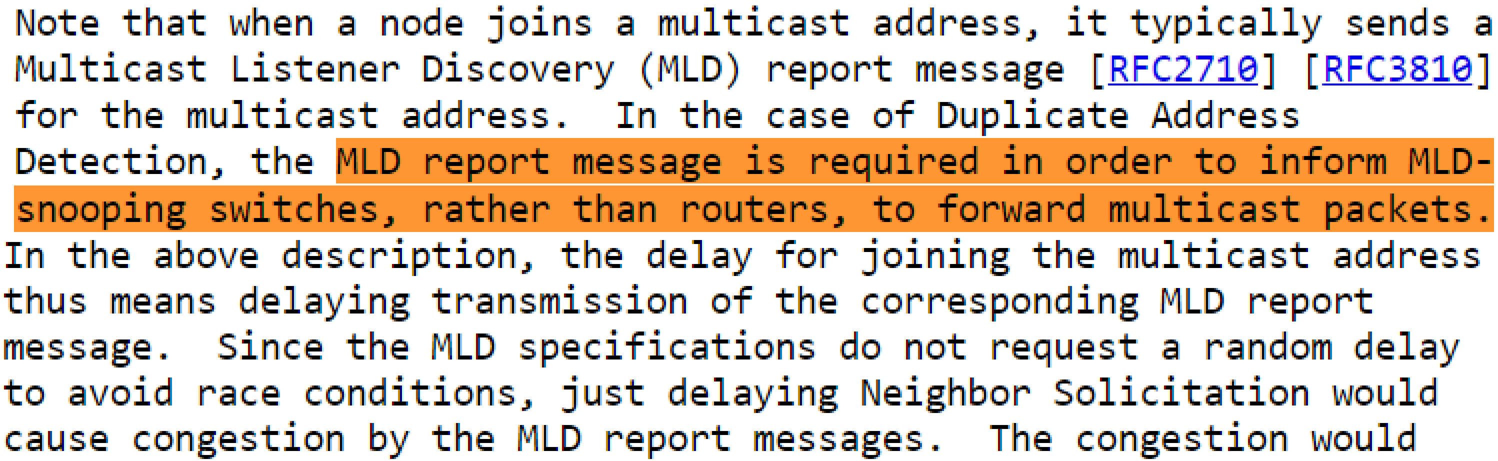


Funky Note #5, The Neighbor Discovery Protocol



- **No broadcast**, **all-nodes** multicast address **instead**.
- **Every IPv6** address has a **associated** derived **Solicited-Node** multicast **group**.
- All **relevant Solicited-Node** groups **must be joined** by a node during interface initialization.
- RFC 4861: "**joining** the **solicited-node** multicast address **is done using** a Multicast Listener Discovery protocol such as the [**MLD**] or [**MLDv2**] protocols."

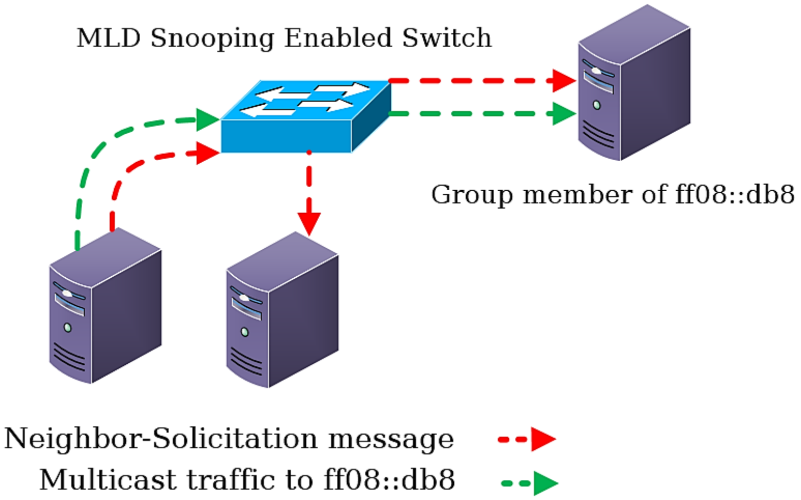
Funky Note #6, Duplicate Address Detection



Note that when a node joins a multicast address, it typically sends a Multicast Listener Discovery (MLD) report message [RFC2710] [RFC3810] for the multicast address. In the case of Duplicate Address Detection, the MLD report message is required in order to inform MLD-snooping switches, rather than routers, to forward multicast packets. In the above description, the delay for joining the multicast address thus means delaying transmission of the corresponding MLD report message. Since the MLD specifications do not request a random delay to avoid race conditions, just delaying Neighbor Solicitation would cause congestion by the MLD report messages. The congestion would

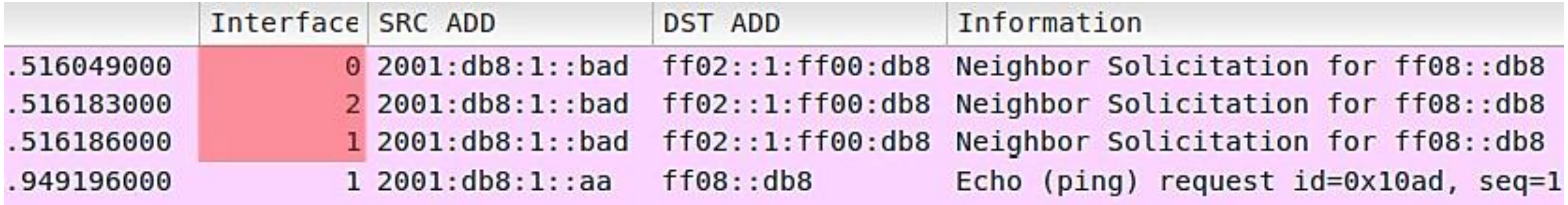
RFC 4862

All this for What? (I)


MLD Snooping Enabled Switch
Group member of ff08::db8
Neighbor-Solicitation message
Multicast traffic to ff08::db8

All this for What? (II)

- **Normal** multicast **traffic**, ICMPv6 in this case, is appropriately **forwarded**.
- **ND-related traffic** just gets **broadcasted**.
- Cisco seemingly followed the easy route here.
  See: http://tools.ietf.org/id/draft-pashby-magma-simplify-mld-snooping-01.txt



	Interface	SRC ADD	DST ADD	Information
.516049000	0	2001:db8:1::bad	ff02::1:ff00:db8	Neighbor Solicitation for ff08::db8
.516183000	2	2001:db8:1::bad	ff02::1:ff00:db8	Neighbor Solicitation for ff08::db8
.516186000	1	2001:db8:1::bad	ff02::1:ff00:db8	Neighbor Solicitation for ff08::db8
.949196000	1	2001:db8:1::aa	ff08::db8	Echo (ping) request id=0x10ad, seq=1

Wrap-Up

What have we learned?

Some Ideas for Admins

- **Limit** the **rate** at which your infrastructure components **process MLD messages**.
- If you're not running multicast applications, **stay away** from **MLD-Snooping**
- If pertinent, **consider filtering MLD** messages on your access and distribution layers; at least Queries.
- **Don't** enable full **multicast routing** or **MLD-Snooping** for **few services**. **Configure** multicast groups used for critical services **statically** (e.g. DHCPv6)

A Couple of Points for the IETF

- **MLDv2**: **Routers** must **not accept Queries** destined to FF02::2, FF02::16, or unicast addresses, link-local or global.
  - “For debugging purposes” **isn’t** a **valid** reason
- **MLDv1**: Nodes must **not accept** Reports to their unicast addresses.
- **Both**: **Querier election** by using the ‘lowest’ IPv6 address? Is such a trivial mechanism **really useful**?

Future Work

- **Telcos** are **deploying** IPv6 **multicasting** in their **IPTV** solutions
- **Surveillance using IP** cameras is widespread. As **IPv6 gains traction** IPv6 **multicast** is **likely** to also come into play
- **Video-conferencing** is now sought after by ‘the management’. Solutions also **rely** on **multicasting**
- **How** are **cheap appliances** and simple **networks** going to **deal** with what allegedly is **the** ‘**future**’ of the Internet?

Conclusions

- You have **MLD** traffic **in your IPv6 network**, yes you do!
- Theory says **MLD** is **required** for ND, practice shows it **isn't**
- **MLD** introduces **complexity** and a **immature** codebase
- **MLD** is crucial for IPv6 multicasting, but **not for** your **typical IPv6 network**.
- **If multicasting** is the **future**, more people **have** a **critical look** at the protocols that power it, among them MLD
- The IETF should **reconsider** the **role** and **design** of **MLD**

Thank You for Your Time!

Enjoy BlackHat Asia!

Antonios Atlasis
aatlasis@secfu.net

Jayson Salazar
jsalazar@ernw.de

Rafael Schaefer
rschaefer@ernw.de