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: 16.61%
- Japan: 10.96%
- Malaysia: 8.25%
- Singapore: 4.53%

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

<table>
<thead>
<tr>
<th>SRC_ADDR</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>fe80:8678:acff:feb3:eb20</td>
<td>Multicast Listener Query</td>
</tr>
<tr>
<td>fe80:6267:20ff:fea5:d9c4</td>
<td>Multicast Listener Report</td>
</tr>
<tr>
<td>fe80:6267:20ff:fea5:d9c4</td>
<td>Multicast Listener Report</td>
</tr>
<tr>
<td>fe80:6267:20ff:fea5:d9c4</td>
<td>Multicast Listener Report</td>
</tr>
<tr>
<td>fe80:6267:20ff:fea5:d9c4</td>
<td>Multicast Listener Report</td>
</tr>
<tr>
<td>fe80:6267:20ff:fea5:d9c4</td>
<td>Multicast Listener Report</td>
</tr>
<tr>
<td>fe80:6267:20ff:fea5:d9c4</td>
<td>Multicast Listener Report</td>
</tr>
<tr>
<td>fe80:6267:20ff:fea5:d9c4</td>
<td>Multicast Listener Report</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>SRC_ADD</th>
<th>DST_ADD</th>
<th>MLD_MADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:23:18.574201000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:18.574210000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:18.623020000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:18.623011000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:18.849934000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:18.849938000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:36.215326000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:36.215336000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:36.276699000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:36.276008000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:36.339596000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:36.339601000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:37.201776000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:37.201787000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:37.203986000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
<tr>
<td>13:23:37.203993000</td>
<td>fe80::200:ff:fe00:11</td>
<td>ff02::16</td>
<td>ff02::1:3</td>
</tr>
</tbody>
</table>
MLD, Every Protocol Has a Story

Hopefully, an entertaining one.
The Unicast Side of Things
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
  - Auto-Configuration and Monitoring
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

MLDv1
- MLDv1 Queries
  - General Query
  - Specific Query

MLDv2
- MLDv2 General Query
- MLDv2 Address Specific Query
- MLDv2 Address Specific and Source Query

MLDv1 Report

MLDv1 Done

MLDv2 Report
### Querier-Sent Messages, Queries

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

<table>
<thead>
<tr>
<th>Internet Control Message Protocol v6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type:</strong> Multicast Listener Query (130)</td>
</tr>
<tr>
<td><strong>Code:</strong> 0</td>
</tr>
<tr>
<td><strong>Checksum:</strong> 0x6b89 [correct]</td>
</tr>
<tr>
<td><strong>Maximum Response Code:</strong> 0</td>
</tr>
<tr>
<td><strong>Reserved:</strong> 0000</td>
</tr>
<tr>
<td><strong>Multicast Address:</strong> ff08::2001:db8 (ff08::2001:db8)</td>
</tr>
<tr>
<td><strong>Flags:</strong> 0x00</td>
</tr>
</tbody>
</table>

- .... 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) |
Listener-Sent Messages, Reports

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

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
Funky Note #1, State Keeping on Gateways

- 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.
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
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
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 and clients’ side
  - One can easily ‘impersonate’ the Querier
  - One can communicate on a one-to-one basis
  - Some clients implement Report suppression
  - Forcing a network to MLDv1 is trivial
  - Anything else?
Playing with MLD
On how and what we tested
Test Environment

- **Cisco 1921** routers and **Cisco 2960s** switches. **IOS 15.2** and **15.4** respectively.
- Android, FreeBSD, Ubuntu and Windows virtualized guests
- **Tools**
  - Scapy
  - Chiron
  - Dizzy
  - THC IPv6 Toolkit
  - Wireshark
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
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

- Certain enterprise grade virtualization solution, which shall sadly remain unnamed, crashes when high rates of MLD traffic are received on an Intel 82573L network interface.

- Relevant only as DoS.

- IPv6 ‘reliance’ on multicast should be expected to cause trouble with network cards.
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

<table>
<thead>
<tr>
<th>SRC MAC</th>
<th>SRC ADD</th>
<th>DST ADD</th>
</tr>
</thead>
<tbody>
<tr>
<td>03.275444000</td>
<td>kali_eth0</td>
<td>fe80::200:ff:fe00:14</td>
</tr>
<tr>
<td>03.275458000</td>
<td>kali_eth0</td>
<td>fe80::200:ff:fe00:14</td>
</tr>
<tr>
<td>08.737940000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>08.737953000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>26.141097000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>26.141105000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>50.939472000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>50.939489000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>08.343150000</td>
<td>kali_eth0</td>
<td>fe80::200:ff:fe00:14</td>
</tr>
<tr>
<td>08.343160000</td>
<td>kali_eth0</td>
<td>fe80::200:ff:fe00:14</td>
</tr>
<tr>
<td>43.335196000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>43.335208000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>12.541043000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>12.541050000</td>
<td>freebsd_eth0</td>
<td>fe80::200:ff:fe00:13</td>
</tr>
<tr>
<td>13.410482000</td>
<td>kali_eth0</td>
<td>fe80::200:ff:fe00:14</td>
</tr>
<tr>
<td>13.410495000</td>
<td>kali_eth0</td>
<td>fe80::200:ff:fe00:14</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>SRC</th>
<th>MAC</th>
<th>SRC ADD</th>
<th>MLD MADDR</th>
<th>Len.</th>
</tr>
</thead>
<tbody>
<tr>
<td>47.373682000</td>
<td>ubuntu_eth0</td>
<td>ubuntu.local</td>
<td>ff08::db8</td>
<td>90</td>
</tr>
<tr>
<td>47.373696000</td>
<td>ubuntu_eth0</td>
<td>ubuntu.local</td>
<td>ff08::db8</td>
<td>90</td>
</tr>
<tr>
<td>56.087140000</td>
<td>Cisco_15:c0:11</td>
<td>fe80::200:ff:fe15:c011</td>
<td>ff08::db8,ff02::fb,ff02::1:ff00:12</td>
<td>130</td>
</tr>
<tr>
<td>58.028565000</td>
<td>ubuntu_eth0</td>
<td>ubuntu.local</td>
<td>ff08::db8,ff02::fb,ff02::1:ff00:12</td>
<td>130</td>
</tr>
<tr>
<td>58.028578000</td>
<td>ubuntu_eth0</td>
<td>ubuntu.local</td>
<td>ff08::db8</td>
<td>90</td>
</tr>
<tr>
<td>38.885241000</td>
<td>kali_eth0</td>
<td>fe80::200:ff:fe00:14</td>
<td>ff08::db8</td>
<td>90</td>
</tr>
<tr>
<td>38.885255000</td>
<td>kali_eth0</td>
<td>fe80::200:ff:fe00:14</td>
<td>ff08::db8</td>
<td>90</td>
</tr>
<tr>
<td>01.332813000</td>
<td>Cisco_15:c0:11</td>
<td>fe80::200:ff:fe15:c011</td>
<td>ff08::db8</td>
<td>90</td>
</tr>
<tr>
<td>09.418357000</td>
<td>ubuntu_eth0</td>
<td>ubuntu.local</td>
<td>ff08::db8,ff02::fb,ff02::1:ff00:12</td>
<td>130</td>
</tr>
<tr>
<td>09.418367000</td>
<td>ubuntu_eth0</td>
<td>ubuntu.local</td>
<td>ff08::db8,ff02::fb,ff02::1:ff00:12</td>
<td>130</td>
</tr>
<tr>
<td>06.582484000</td>
<td>Cisco_15:c0:11</td>
<td>fe80::200:ff:fe15:c011</td>
<td>ff08::db8</td>
<td>90</td>
</tr>
<tr>
<td>13.996287000</td>
<td>ubuntu_eth0</td>
<td>ubuntu.local</td>
<td>ff08::db8,ff02::fb,ff02::1:ff00:12</td>
<td>130</td>
</tr>
<tr>
<td>13.996304000</td>
<td>ubuntu_eth0</td>
<td>ubuntu.local</td>
<td>ff08::db8,ff02::fb,ff02::1:ff00:12</td>
<td>130</td>
</tr>
</tbody>
</table>
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
Anything else?

One last minor detail
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
- Use **Chiron**, Scapy, THCv6 Toolkit or NMAP*

**Trivial Host Discovery and Fingerprinting (I)**

![Diagram showing IPv6 Node and Recon Node with MLDv2 Query and Report messages](image)
## Trivial Host Discovery and Fingerprinting (II)

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

Wrap-Up

What have we learned?
Some Ideas for Admins

- **Limit** the rate at which your infrastructure components **process MLD messages** (CoPP).
- 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**?
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?
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 Singapore!

Antonios Atlasis
aatlasis@secfu.net

Jayson Salazar
jsalazar@ernw.de

Rafael Schaefer
rschaefer@ernw.de