IoT Updates with IPv6 Multicast

Brett Sheffield, Librecast Project

@brett_sheffield

#FOSDEM2020
Before we begin...
"IP Multicast will play a prominent role on the Internet in the coming years. It is a requirement, not an option, if the Internet is going to scale. Multicast allows application developers to add more functionality without significantly impacting the network."

– RFC 3170, Sep 2001
Multicast

Efficient
Multicast

Scalable
Real-World
Multicast

Decentralisation
What is Multicast?
Definition

- Unicast
- Broadcast
- Multicast
Definition

Unicast

Broadcast

Multicast

Unicast
Definition

Unicast

Broadcast
Definition

Unicast

Broadcast

Multicast
Definition

Unicast

Broadcast

Multicast
What is Multicast?

Unicast, Broadcast

Multicast
What is Multicast?

Unicast, Broadcast PUSH

Multicast
What is Multicast?

Unicast, Broadcast  PUSH

Multicast  PULL
Multicast Misconceptions
Multicast Misconceptions

- only for streaming
Multicast Misconceptions

- only for streaming
- no use for video on demand
Multicast Misconceptions

- only for streaming
- no use for video on demand
- unreliable
Multicast Misconceptions

- only for streaming
- no use for video on demand
- unreliable
- insecure
Multicast Misconceptions

- only for streaming
- no use for video on demand
- unreliable
- insecure
- can’t work on Internet
Multicast Misconceptions

- only for streaming
- no use for video on demand
- unreliable
- insecure
- can’t work on Internet
Multicast is ...
Multicast is ... Group Communication
All Communication is Group Communication
IoT Updates
https://github.com/librestack/iotupd
Datagram:
Datagram:

- checksum
Datagram:

- checksum
- size of file
Datagram:

- checksum
- size of file
- size of chunk
Datagram:

- checksum
- size of file
- size of chunk
- offset
Datagram:

- checksum
- size of file
- size of chunk
- offset
- data
Receiving a File
Receiving a File
Receiving a File

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File
Receiving a File

Data received: 100%
Receiving a File

Data received: 100%
Checksum match
What Just Happened?
Are there other ways we can achieve TCP/IP-like reliability?
PGM
(RFC 3208 - Experimental)
NACKs, Replay
Loop and Repeat
Flow Control
Flow Control

t \downarrow

\mathcal{g}_1
Flow Control

\[
g_1 \quad 1
\]
Flow Control

g_1

1

2

t
Flow Control
Flow Control
Flow Control

\[ \begin{array}{c|c|}
\text{g}_1 & \text{g}_2 \\
\hline
1 & 2 \\
3 & 4 \\
\end{array} \]
Flow Control

g₁  g₂  g₃  g₄

1  2  3  4
Reliability

<table>
<thead>
<tr>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>$g_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Reliability

<table>
<thead>
<tr>
<th></th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$g_3$</th>
<th>$g_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

$t$
Anatomy of an IPv6 Multicast Address
Anatomy of an IPv6 Multicast Address

ff
Anatomy of an IPv6 Multicast Address

ff1
Anatomy of an IPv6 Multicast Address

ff1e
Anatomy of an IPv6 Multicast Address

ff1e: + group address (112 bits)
Multicast "DNS"

ff1e: + HASH("example.com")
Multicast "DNS"

example.com  ⟷  ff1e:873e:378f:f6a5:a1f6:fa49:95f1:0faf
Librecast
Librecast

- Developers Developers Developers

- Messaging Library

- Transitional Technology

- Improved Routing Protocol

- Build multicast-enabled applications

- Work with FOSS projects to enable multicast everywhere

- Ensure new standards (e.g., WebRTC, QUIC) support multicast
Librecast

- Developers
- Developers
- Developers
- Developers
- Messaging Library
Librecast

- Developers
- Messaging Library
- Transitional Technology
Librecast

- Developers Developers Developers
- Messaging Library
- Transitional Technology
- Improved Routing Protocol
Librecast

- Developers
- Messaging Library
- Transitional Technology
- Improved Routing Protocol
- Build multicast-enabled applications
Librecast

- Developers
- Messaging Library
- Transitional Technology
- Improved Routing Protocol
- Build multicast-enabled applications
- Work with FOSS projects to enable multicast everywhere
Librecast

- Developers
- Messaging Library
- Transitional Technology
- Improved Routing Protocol
- Build multicast-enabled applications
- Work with FOSS projects to enable multicast everywhere
- Ensure new standards (eg. WebRTC, QUIC) support multicast
lc_ctx_t *ctx;
lc_socket_t *sock;
lc_channel_t *chan;
lc_message_t msg;

ctx = lc_ctx_new();
sock = lc_socket_new(ctx);
chan = lc_channel_new(ctx, channelName);
lc_channel_bind(sock, chan);

lc_msg_init_size(&msg, strlen(msgtext) - 1);
lc_msg_send(chan, &msg);

/* clean up */
lc_socket_close(sock);
lc_channel_free(chan);
lc_ctx_free(ctx);
Brett Sheffield — Librecast Project

http://brettsheffield.com — Email: brett@librecast.net
Freenode: bacs — Twitter: @brett_sheffield
github.com/brettsheffield — github.com/librestack