Bluetooth® Mesh
Johan Hedberg
What is Bluetooth Mesh?

- New standard which came out in 2017
- Many-to-many, multi-hop topology
- No new Bluetooth HW required
- Broadcast & relay in a flooding/ripple fashion
- Mainly for signaling – not large data transfer
- Message publication & subscription
- Multi-level security
- Greatly extended range
Mesh in terms of Bluetooth

**BR/EDR**
- for continuous connections
- pairing (1:1)
  - audio streaming
    - wireless speakers
    - wireless headsets
    - in-car infotainment
- 2016: 730M | 2020: 930M

**Low Energy (LE)**
- for short burst connections
  - pairing (1:1)
    - data transfer
      - sports & fitness devices
      - medical & healthcare devices
      - peripherals & accessories
- broadcasting (1:m)
  - localized info sharing
    - Pol information
    - item finding
    - way finding
- 2016: 573M | 2020: 975M
  - 2016: 12M | 2020: 380M
  - mesh networking (m:m)
    - large device networks
      - building automation
      - sensor networks
      - asset tracking
    - Launching mid-2017
Mesh in terms of LE roles

Central - Peripheral
- Connection-oriented, between two devices
- Sensor as peripheral, your phone or PC as the central

Observer - Broadcaster
- Observer scans for advertising packets
- Broadcaster sends advertising packets for everybody who is scanning
- The natural choice for Mesh
Node Types

GATT Proxy

GATT Client

Relay

Provisioner

Low-Power Node

Friend
Node Lifecycle

- **Node Reset**: Unprovisioned Device
  - **Provisioning**: ECDH, OOB, Network Key, Device Key, Address

- **Configured Mesh Node**
  - **Configuration**: Node Composition, Application Key(s), Group Subscription & Publication

- **Unconfigured Mesh Node**

- **Blacklisted Mesh Node**
  - **Blacklisting**: Key Refresh
Node Composition: Elements & Models

**Elements**
- Unique Network Address
- Implements one or more Models

**Models**
- OpCode addressing
- States & Messages
- Client & Server
Mesh Protocol Layers

- **Models**: states / messages / behavior
- **Access Layer**: opcodes, multiplexing models
- **Upper Transport Layer**: heartbeat/friendship, application encryption & authentication
- **Lower Transport Layer**: segmentation & reassembly
- **Network Layer**: message format, network encryption & authentication
- **Advertising Bearer**: message transport
- **GATT Bearer (Optional)**
# Anatomy of a Mesh Network PDU

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Bits</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVI</td>
<td>1</td>
<td>Least significant bit of IV Index</td>
</tr>
<tr>
<td>NID</td>
<td>7</td>
<td>Value derived from the NetKey used to identify the Encryption Key and Privacy Key used to secure this PDU</td>
</tr>
<tr>
<td>CTL</td>
<td>1</td>
<td>Network Control</td>
</tr>
<tr>
<td>TTL</td>
<td>7</td>
<td>Time To Live</td>
</tr>
<tr>
<td>SEQ</td>
<td>24</td>
<td>Sequence Number</td>
</tr>
<tr>
<td>SRC</td>
<td>16</td>
<td>Source Address</td>
</tr>
<tr>
<td>DST</td>
<td>16</td>
<td>Destination Address</td>
</tr>
<tr>
<td>TransportPDU</td>
<td>8 to 128</td>
<td>Transport Protocol Data Unit</td>
</tr>
<tr>
<td>NetMIC</td>
<td>32 or 64</td>
<td>Message Integrity Check for Network</td>
</tr>
</tbody>
</table>
Mesh Network Addresses

16-bit Network address with several categories/ranges

<table>
<thead>
<tr>
<th>Category</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unassigned</td>
<td>0000000000000000</td>
<td>No address assigned (typically used when not publishing or subscribing)</td>
</tr>
<tr>
<td>Unicast</td>
<td>0xxxxxxxxxxxxxxx</td>
<td>Every element has a unique unicast address</td>
</tr>
<tr>
<td>Virtual</td>
<td>10xxxxxxxxxxxxxxx</td>
<td>Special group addresses authenticated using a 128-bit virtual label UUID</td>
</tr>
<tr>
<td>Group</td>
<td>11xxxxxxxxxxxxxxx</td>
<td>Fixed (all nodes, all friends, etc) or dedicated (application specific)</td>
</tr>
</tbody>
</table>
Relaying

- Time-to-Live (TTL, 7-bit, i.e. max 127)
- Decrypt with Network Key
- Decrement TTL

If TTL > 0:
- Re-encrypt with Network Key
- Send out to Network
- Application layer payload remains encrypted & untouched
  - Relay Node may not even have the Application Key
Publish & Subscribe

Nodes
Switch 1
Switch 2
Switch 3
Switch 4
Switch 5
Switch 6
Switch 7

Groups
Hall
Den
Kitchen
Living
Deck

Nodes
Light 1
Light 2
Light 3
Light 4
Light 5
Light 6
Light 7
Security Features

• Authentication during provisioning
• Two level encryption
  – Network
  – Application
• Replay protection
  – IV Index (32-bits)
  – Sequence number (24 bits)
  – IV Index Update procedure
• Key Refresh
  – Node Blacklisting
A message can be either unsegmented or segmented

**Payload**
- Unsegmented: 15 bytes
- Segmented: 12 bytes per segment, max 32 segments = 384 bytes
- Contains 4 or 8 byte MIC at the end, reducing usable payload size

Unsegmented messages are inherently unreliable

Segments of a segmented message are acknowledged by the receiver
- One-segment “segmented” message can be used for reliable sending
Friendship

- 100% duty-cycle scanning needed for reliability, but consumes a lot of power
- Mix of battery & mains powered nodes
- Solution: pair up stable power supply nodes (Friends) with Low Power Nodes (LPNs)
- Friends queue up messages for the LPN
- LPN queries the Friend periodically if there are any messages for it
Implementation status & plans

Linux & Zephyr OS
Support in Zephyr* OS

- Available starting with Zephyr 1.9
- All mandatory features implemented
- Tested against multiple other implementations
- Ported to MyNewt
  - Multiple valuable fixes ported back to Zephyr
- Demos possible with many popular supported Zephyr boards
  - Come to the Zephyr booth to see it in action!
- Minimum RAM footprint (entire OS with Mesh) is ~12kB
  - Fits even the most constrained 16k boards, like BBC micro:bit
Support in Linux*

- meshctl tool released with BlueZ 5.47
  - GATT Client
  - PB-GATT Provisioner
- Ongoing work both in user space (BlueZ) and kernel
  - Kernel acts as a very thin layer, interface is almost the same as the vendor HCI extensions
  - Essentially everything else in a user space meshd
Future development

- Mesh Vendor HCI Extensions
  - Supported both by Linux & Zephyr
- More features
  - Provisioner role
- More standard models
- More demos with various boards
Questions?
Disclaimer

Intel technologies’ features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at www.intel.com.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

The Bluetooth® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by Intel Corporation is under license.

*Other names and brands may be claimed as the property of others.

© Intel Corporation