How to configure STSW-BNRG-Mesh SDK options related to neighbor info

Introduction

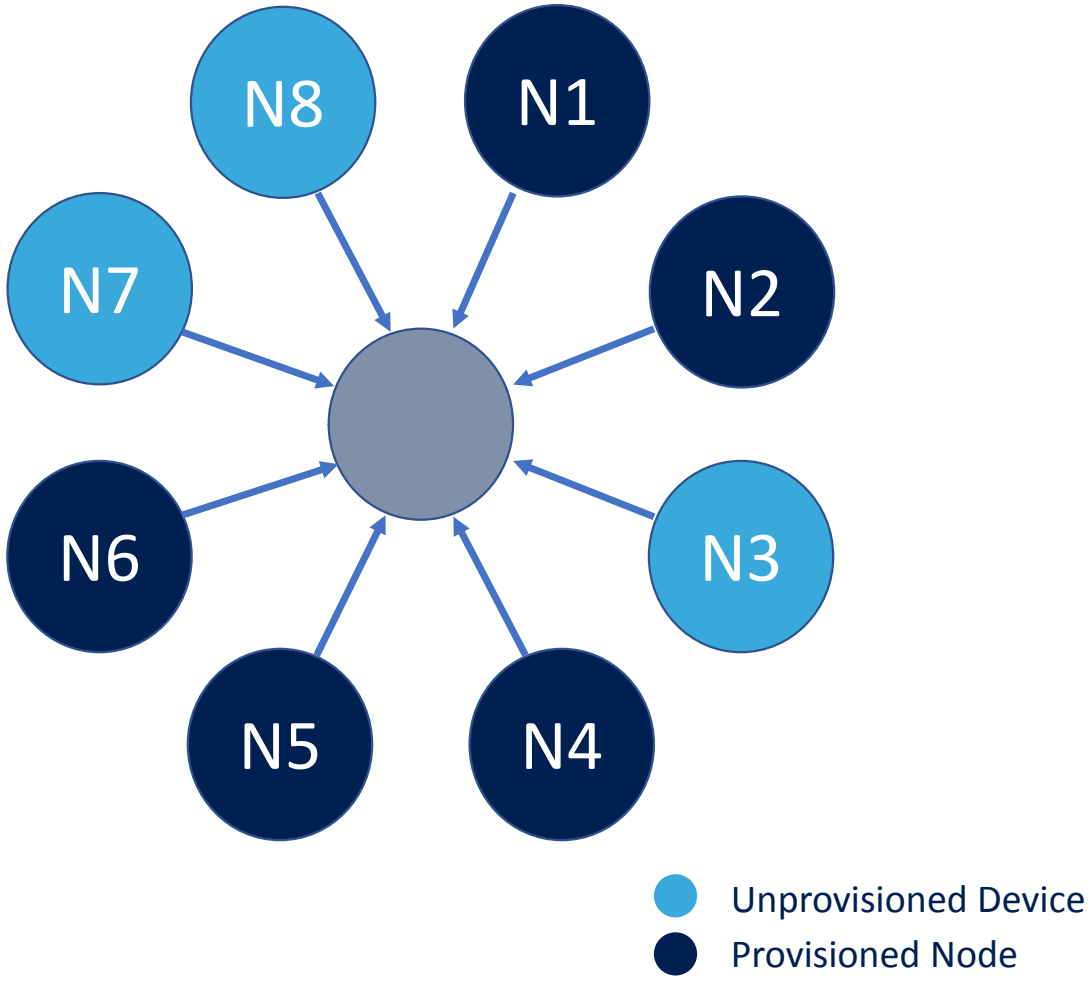
A Bluetooth Mesh-enabled device receives various types of advertisement packets from similar nearby devices.

A neighbor is a valid neighbor if it is an Unprovisioned Mesh device or a Mesh node provisioned in the same network.

Advertisements are only received from immediate neighbors. Based on its own state (provisioned or unprovisioned) and neighbor's state (provisioned or unprovisioned), a device can extract useful (or relevant) neighbor information from the corresponding advertisements.

STSW-BNRG-Mesh supports a neighbor information database. Various configurable options are available in the SDK related to neighbor info.

Figure 1. STSW-BNRG-Mesh example of Bluetooth Mesh-enabled device surrounded by 8 neighbors



1 Bluetooth Mesh packet types

Bluetooth Mesh defines different types of packets. Among them, all the non-connectable packet types are processed to update neighbor info:

- Unprovisioned Device beacon: sent by unprovisioned devices
- Secure Network beacon: periodically sent by provisioned nodes
- Mesh network message: handling of messages is categorized as
  - Mesh network message with TTL equal to 0
  - Mesh network message with TTL not equal to 0

The following parameters are defined for neighbors:

- bdAddr: Bluetooth Low Energy MAC address (valid if not NULL)
- provisioned: if neighbor is provisioned or not (always valid)
- uuid: UUID of neighbor (valid if not NULL)
- networkAddress: neighbor Mesh network address (valid if not 0x0000)
- rssi: last updated rssi value (always valid)

For each different type of Mesh packet, different sets of parameters are valid.

Table 1. Parameter validity and corresponding packet types

Packet type	Parameter validity (Y = valid, N = not valid)				
	bdAddr	Provisioned	uuid	networkAddress	rssi
Unprovisioned Device beacon	Y	Y	Y	N	Y
Secure Network beacon	Y	Y	N	N	Y
Message with TTL = 0	Y	Y	N	Y	Y
Message with TTL != 0	Y	Y	N	N	Y

An unprovisioned device can only process Unprovisioned Device beacons and is unable to understand secure communication among provisioned nodes. Instead, a provisioned node can process all kinds of packets (i.e., Unprovisioned Device beacon, Secure Network beacon and Mesh messages).

2 Neighbor info parameter configuration

STSW-BNRG-Mesh allows flexible configuration of neighbor implementation.

The neighbor info processing is enabled by defining `ENABLE_NEIGHBOR_TABLE` in the *mesh_cfg_usr.h* file.

Implementation is modular in terms of Flash and RAM use. If the neighbor info processing is disabled, the corresponding memory is free to be used for other application purposes.

Table 2. Neighbor info processing configurable parameters

Parameter	Comments
`NEIGHBOR_COUNT` Value: 1 - 25	Maximum number of neighbors that can exist in the neighbor info database. By increasing `NEIGHBOR_COUNT` the RAM usage increases
`NEIGHBOR_ALIVE_TIME` Value: 1 s – 65535 s	Time during which the neighbor exists in the neighbor info database. `NEIGHBOR_ALIVE_TIME` is calculated with reference to the appearance of a new neighbor or the last refresh of an existing neighbor
`NEIGHBOR_UNPRVND_DEV_BEACON_NTU` 0: Disable 1: Enable	To enable or disable neighbor info update with Unprovisioned Device beacon. If `NEIGHBOR_UNPRVND_DEV_BEACON_NTU` is disabled, Unprovisioned Device beacons are not used to update the neighbor info database
`NEIGHBOR_SECURE_NET_BEACON_NTU` 0: Disable 1: Enable	To enable or disable neighbor info update with Secure Network beacon. If `NEIGHBOR_SECURE_NET_BEACON_NTU` is disabled, Secure Network beacons are not used to update neighbor info database
`NEIGHBOR_MSG_TTLX_NTU` 0: Disable 1: Enable for messages with TTL = 0 2: Enable for all messages	To enable or disable neighbor table update with a Mesh network message. If `NEIGHBOR_MSG_TTLX_NTU` is disabled, Mesh messages are not used to update the neighbor info database. If it is enabled for messages with TTL = 0, messages with TTL > 0 are ignored. Enabling `NEIGHBOR_MSG_TTLX_NTU` for all messages might result in a high frequency processing of the neighbor info database which leads to high latency and slow response

3 Setup and API implementation

3.1 Setup

Step 1. Enable the neighbor table implementation by defining `ENABLE_NEIGHBOR_TABLE` in the *mesh_cfg_usr.h* file.

Step 2. Set the required neighbor table parameters.

Table 3. Neighbor table example parameters

Parameter	Value
NEIGHBOR_COUNT	5
NEIGHBOR_ALIVE_TIME	20 s
NEIGHBOR_UNPRVND_DEV_BEACON_NTU	0
NEIGHBOR_SECURE_NET_BEACON_NTU	1
NEIGHBOR_MSG_TTLX_NTU	1

3.2 BluenrgMesh_NeighborAppearedCallback

`BluenrgMesh_NeighborAppearedCallback` signals the presence of a new neighbor in the neighbor info database, placed according to the parameters given in Section 3.1 Setup. This callback passes the related information to the application.

The parameters available for this callback are bdAddr, provisioned state, uuid, networkAddress and rssi. Depending on the type of packet used for processing neighbor info, parameters can be valid or invalid.

3.3 BluenrgMesh_NeighborRefreshedCallback

`BluenrgMesh_NeighborRefreshedCallback` occurs when refreshing an existing neighbor. If the last neighbor appearance time or neighbor refresh time is less than `NEIGHBOR_ALIVE_TIME`, information is passed to application via this callback.

The parameters available for this callback are bdAddr, provisioned state, uuid, networkAddress and rssi. Depending on the type of packet used for processing neighbor info, parameters can be valid or invalid.

3.4 BluenrgMesh_GetNeighborState

`BluenrgMesh_GetNeighborState` can be used to retrieve neighbor info from an existing database.

Input parameters to be provided are `pNeighborTable` (pointer to the buffer which is updated with neighbor information) and `pNoOfNeighborPresent` (pointer to the variable which is updated with the number of valid neighbors).

It is important to note that bdAddr is used to process the neighbor info database. If neighbor's bdAddr changes, it appears as a new neighbor.

3.5 Memory footprint

Defining `ENABLE_NEIGHBOR_TABLE` occupies 484 bytes of Flash memory while RAM consumption is `32* NEIGHBOR_COUNT`.

If the application enables neighbor info processing with `NEIGHBOR_COUNT` = 10, the total Flash usage would increase by 484 bytes and RAM usage would increase by 320 bytes.

Revision history

Table 4. **Document revision history**

Date	Version	Changes
19-Nov-2019	1	Initial release.

Contents

List of tables