Demo: The Trade-offs of Connected VS Broadcast BLE Mesh Networking

Brecht Reynders, Yuri Murillo, Alessandro Chiumento and Sofie Pollin
KU Leuven, Department of Electrical Engineering (ESAT), Kasteelpark Arenberg 10, 3001 Heverlee, Belgium.
brecht.reynders, yuri.murillo, alessandro.chiumento, sofie.pollin @esat.kuleuven.be

Abstract

There is a lot of interest within the IoT community regarding the use of BLE mesh networks for reliable, energy efficient communication between resource constrained nodes. Even though the vast majority of industrial and consumer implementations tend to make use of broadcast BLE mesh, there is little to no analysis of the trade-offs between connected and broadcast mesh. In this demo we will show what are the strong and weak points of either of these two BLE meshing techniques and provide visual cues of the differences.

1 Introduction and Motivation

Bluetooth Low Energy (BLE) represents the low-power, low-cost extension of the Bluetooth communication technology envisioned for the Internet of Things (IoT). It has drawn strong interest from the industrial community for its good performance with respect to 802.15.4 [7] and its already well known indoor propagation characteristics [6]. The main issues with BLE nodes is that, because of their low power, they suffer of limited range, in order to overcome this hurdle, meshing protocols have been suggested. A communication protocol for multi-hop BLE networks is not yet available in the standard [3]. As of today, the vast majority of BLE mesh implementations employ broadcast messages to transfer the payload and do not allow the nodes to connect with one another. Although this method is generally considered to allow message spread with limited latency, there is no information on its reliability and energy consumption [4]. The packets are then then propagated over the network with a flooding mechanism, ensuring quick and widespread delivery but at great overhead cost [8].

In this demo we show that the choice to move towards broadcast mesh might have been an impulsive one and that connected mesh possesses properties quite complementary to its broadcast counterpart. In this work a neighbour-only routing scheme called FruityMesh [1] has been used to develop the connected mesh and the well known Trickle flooding scheme [5] has been implemented for broadcast meshing.

International Conference on Embedded Wireless Systems and Networks (EWSN) 2017
20–22 February, Uppsala, Sweden

ISBN: 978-0-9949886-1-4

2 The Algorithms

The two algorithms chosen to build the connected and broadcast mesh are described in this section. Their main characteristics and main differences are discussed.

2.1 Trickle Flooding

Trickle works by broadcasting a message to any node able to receive it. An internal suppression mechanism makes sure that the message is not re-transmitted if already overheard a certain number of times or the time to live of the message has passed [5]. Trickle's behaviour is given by some fixed parameters such as the limits of the Trickle Interval I_{MIN} and I_{MAX} and the redundancy constant k, which represents the threshold for which a node decides whether to re-broadcast a message or not. During the Trickle interval I, the node has a listen-only period from $[0, I/2)$ and a transmission period from $[I/2, I]$ in this interval, the transmission time t is chosen randomly by $t \in U(I/2, I)$. At any point in time, the redundancy counter c is incremented by 1 every time the latest message is overheard. If $c > k$ at time t then the transmission is suppressed. Finally, if the Trickle interval expires without receiving any message, it is doubled until I_{MAX} and c is reset to 0. If a new message different from the previous one is detected, the Trickle Interval is reset to I_{MIN} and the whole procedure is repeated.

2.2 FruityMesh

FruityMesh is a neighbour-only open-source routing scheme. No routing tables are stored but a connection is established between two neighbours and kept open until the packets reach their final destination (through multiple hops). The Algorithm works by having the node alternate between different states, in the *Initialisation* phase, a node is activated and generates some local information on its ID and cluster size. In the *Discovery* phase the node broadcasts its ID and cluster information. Also it scans on all three broadcast channels for similar messages. During the *Route Set-up* phase, the nodes decide to which neighbours to connect based on the their cluster size (the bigger the cluster the higher the connection priority, this way all the nodes in a network will automatically converge to the same cluster).

Information on the new cluster sizes is then propagated over the network via update messages over the connected mesh.

As the network reaches stability and all the nodes are part of the same cluster, the time between advertisements is decreased. If a connection is lost then the connection partner of a failed node switches to higher discovery mode. As a self-healing mechanism, alternative routes are generated every time a node disconnects. In this context, regular exchange of advertising packets constitute the key element for ensuring link maintenance and quick reaction to link changes.

Routing is carried out in a a neighbour-only fashion. No routing tables are used and each node is only aware of the nodes around it. When a sink is present in the network, this information is broadcasted over the FruityMesh network. Each node then, when it has data to send, selects the neighbour with the smallest hop-count to the sink and sends that device all the packets. In case a connection between nodes is lost, the nodes go back into discovery mode and a new connection is established.

2.3 Hardware and Implementation

For this demo the nRF52 development boards manufactured by Nordic Semiconductor have been used [2].

Figure 1: nRF52 Development Kit Board [2]

The boards, shown in Figure 1 are programmed with a pre-compiled and pre-linked binary file that contains the whole BLE protocol stack, known as the Softdevice. The two mesh protocols then run as applications on top of such stack. This approach decreases complexity in the development and implementation of the mesh protocols, but gives reduced freedom when trying to set and read BLE specific parameters given the closed nature of the Softdevice. This is not an issue for this work, as the main focus is on characterization of the mesh technologies that run on top of BLE.

3 Demonstration

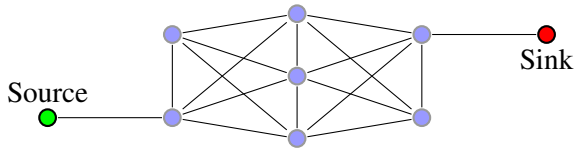

Figure 2: Multi-hop set-up: 2 devices will communicate with each other in a multi-hop fashion using the nodes in the middle.

In this demo, 9 nRF52 Bluetooth development boards are positioned as shown in Figure 2. The objective of this demonstration is to show the main differences between the two meshing schemes mentioned above. To demonstrate this, all boards will run the implemented software as described above, and hence, they will be able to run one of both schemes on demand. One device, the red dot in Fig. 2, will be configured as the sink. A second device, the green dot in Figure 2, will be configured as a sensor. This sensor will provide data at regular time intervals. This data will be send over the network using the devices in the middle to reach the sink in multiple hops.

To show the difference in performance between FruityMesh and Trickle, two different screens are presented to the user. The first screen allows the user to select the protocol to test and the arrival rate of the packets. On the second screen relevant network parameters such as power consumption, latency, packet error rate and throughput will be visualized. These information will be polled from each of the devices over USB. Also, LED's are attached to every device to provide a visual aid of the state in which the nodes are running.

The presented graphs show that when the network is ready, the performance in terms of packet error rate is similar. However, the graphs show as well that the latency in Trickle is significantly lower then in FruityMesh. This comes at a cost as the power consumption in Trickle will also be higher as the nodes will always work in either the receiving or the transmitting state.

4 Conclusions

In this demo we show that there are clear and substantial differences between connected and broadcasts BLE mesh. The strengths and weaknesses of each have been shown; latency, robustness and energy efficiency of the network are all heavily impacted by the meshing algorithm choice and a clear winner in all three is not identifiable.

5 References

[1] FruityMesh. `https://github.com/mwaylabs/fruitymesh/wiki`. Accessed: 2016-11-30.

[2] Nordic Semiconductors nrf52 development board. `https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-Preview-DK`. Accessed: 2016-10-28.

[3] Bluetooth Special Interest Group. *Core Specification Version 4.2*, 12 2014.

[4] M. E. M. Campista, P. M. Esposito, I. M. Moraes, L. H. M. k. Costa, O. C. M. b. Duarte, D. G. Passos, C. V. N. D. Albuquerque, D. C. M. Saade, and M. G. Rubinstein. Routing metrics and protocols for wireless mesh networks. *IEEE Network*, 22(1):6–12, Jan 2008.

[5] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, pages 2–2, Berkeley, CA, USA, 2004. USENIX Association.

[6] P. D. Marco, R. Chirikov, P. Amin, and F. Militano. Coverage analysis of bluetooth low energy and ieee 802.11ah for office scenario. In *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on*, pages 2283–2287, Aug 2015.

[7] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen. How low energy is bluetooth low energy? comparative measurements with zigbee/802.15.4. In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 232–237, April 2012.

[8] A. A. R. Thomas Zahn, Greg O'Shea. An empirical study of flooding in mesh networks. Technical report, April 2009.