On-demand updates after a node failure in a wireless network

Davide Villa, Chih-Kuang Lin
United Technologies Research Center
Cork, Ireland
{villadav, linck}@utrc.utc.com

Adam Kuenzi, Michael Lang
Carrier Corporation
Salem, Oregon, USA
{adam.kuenzi, michael.lang}@carrier.com

***Abstract*—Wireless networks are ubiquitous in our modern world, and we rely more and more on their continuous and reliable operation for battery-powered devices. Networks that self-maintain and self-heal are inherently more reliable. We study efficient and effective network self-healing and update methods for routing recovery following routing failures in a wireless multi-hop network. Network update processes are important since they enable local nodes to maintain the latest and updated neighbor information for routing given the network changes caused by failures. Network update also introduces control signals overhead. In this paper, we investigate the trade-off between routing performance and overhead cost with different network update algorithms and we characterize the performance of the proposed algorithms using network simulations. We show that network updates have positive impacts on routing. In particular, the on-demand route update method provides better results among compared techniques. The improvement is varying depending on the network topology and failure condition scenario.**

***Index Terms*—Bluetooth Low Energy, self-healing, on-demand update, wireless network, multi-hop.**

I. Introduction

Wireless communication is prone to error and failure. When there are failures, a robust network will self-heal and mitigate the problem by a recovery technique. Recovery techniques including the failure recovery and network update have been extensively studied in the past. The packet failure recovery has been thoroughly investigated, and it could be recovered in active or reactive way. Next, the network update challenge is to enable the relevant nodes that share information with neighbors and maintain accurate routing data. However, there is still room for an efficient and effective method for wireless networks with battery-powered devices. The challenge is to maintain accurate routing tables and update them after a failure in the network. The updating of routing paths, creation of new paths, and automatic selection of alternative paths are important for continuous robust operation of the network.

The routing failure recovery has focused on the finding of alternative routes in proactive and reactive way [1] [2] [3] [4] [5] [6]. Afterwards, local network information is updated. In [7], the controller is notified about the failure node and the routes are updated in the server. In [8], after a failure, the nodes broadcast a control packet to the nodes with smaller hop-count. In [9], P2P network performs failure recovery updates based on informing k sequential nodes and improves the selection of nodes to send the notification to. As shown in the state of art above, some algorithms provide on-demand recursive failure notification, but there is no method that guarantees at the same time a configurable, smart and on-demand maintenance and update of the network in both node and route information.

This paper will focus on the network update following a failure identification and subsequent recovery [10]. The network update has two goals: one is to find the latest and accurate network information for routing operation, while the other is to minimize the overhead cost of control signal. In the paper, we first describe the proposed network update methods in Section II. Then, Section III covers the performance evaluations through network simulations and discusses its key findings. Finally, the conclusions are included in Section IV.

II. Network information update

In this section, two on-demand methodologies are proposed to automatically update the information of the nodes subsequent to a failure and a recovery [11]. The system model that we took into account for the proposed methods is a Bluetooth Low Energy (BLE) mesh network, but the same approaches can be extended to all of wireless multi-hop networks.

A. System model

The Bluetooth Low Energy technology exploits an advertisement/scan mechanism for the access control. It operates with 40 channels within the 2.4-2.4835 GHz ISM radio bands with 2 MHz channel spacing. The channels are divided between 3 advertisement channels, which provide the node discovery and where the handshake for the link establishment takes place, and 37 data channels, for the bidirectional data stream flow. A node periodically advertises his presence and availability by sending an advertisement in each advertisement channel. The 3 advertisements are sent with a period composed of a fixed time, called advertising interval (τ_{AI}), plus a variable random delay (δ) to make the sending of advertisements asynchronous and increasing the chance of reception. On the other side, when a node has a pending packet to send, it switches in scan mode where it periodically listens for advertisements in the 3 advertisement channels. The time spent in each channel before switching to the next one circularly is called scan interval (τ_{SI}) which consists of an active listening part, named scan window (τ_{SW}), and a low-power state mode.

978-1-7281-9418-9/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: Northeastern University. Downloaded on November 23,2020 at 21:29:21 UTC from IEEE Xplore. Restrictions apply.

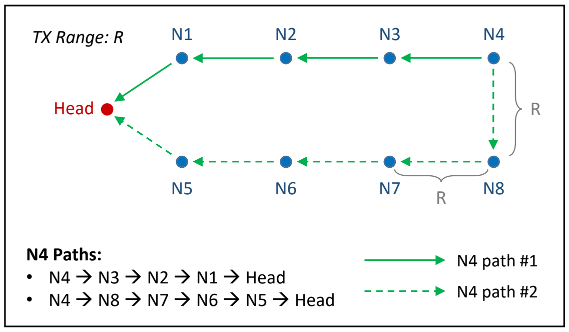

Fig. 1. Multi-path setup algorithm example.

When a proper advertisement is received, the node in scan mode initiates the link establishment by sending a connection request message which contains parameters used for the data delivery. After that, both nodes switch in the data channels where the data delivery is performed. The scan node (transmitter) starts sending data packets one by one and the advertiser node (receiver) is going to reply with an acknowledgment after each packet has been correctly received. When the scan node has completed to deliver all of the packets, both nodes switch to advertisement mode to continue their lifecycle and activities.

The network layer used in this study exploits a multi-path routing approach based on the minimum hop-count distance. The hop-count of a node is defined as the minimum distance in terms of number of hops between the node and a head node. The head node is a data sink in the BLE mesh network. This value is included inside the advertisement message. During the preliminary phase, a node retrieves its own hop-count by listening to the advertisements of its 1-hop neighbors [10]. In the paths creation operations, a node selects the next hop considering the hop-count of its neighbors and making the decision according to multi-path setup algorithm. This algorithm is designed to build a series of disjoint paths sequentially based on the minimum number of hop-count distances [12]. Figure 1 presents an example of the multi-path setup algorithm.

B. *On-demand table update*

After a multi-path routing is set up in a wireless network, failures may occur during a data delivery and trigger a failed path recovery. We next propose an on-demand table update method following the failure recovery. It aims to inform the 1-hop neighbors about a change that has occurred in the network in order to maintain the information of the neighbors up-to-date for future uses. It might be triggered if a first node discovers a failed node, or if a first node re-computes its hop-count and the new value has changed. These triggering conditions can be activated individually or together. When a first node triggers the on-demand table update, it generates an information packet (Table I). This message contains the hop distance of the first node and the node ID of the failed nodes discovered. Then, the first node sends the information packet to each of its 1-hop neighbors. Upon receipt of the information packet, a second node updates the hop distance of the first node and re-computes its own hop-count. If the hop distance has

TABLE I
INFORMATION PACKET OVERVIEW

Packet ID	...	Failed node ID	Sender's hop-count	...

changed, the on-demand table update is triggered recursively and a new information message is created and broadcasted. Additionally, the second node stores the IDs of any failed nodes and checks if any of the failed nodes are its 1-hop neighbor. If so, the second node invalidates the affected failed routes so that they are not selected in future packet deliveries.

The broadcast information message has a lower priority compared to the data packet that triggered the on-demand update, therefore it is served after the data message. This is because the system is configured to push the data packet, with higher priority, as soon as possible towards its final destination. Additionally, a node is able to include extra details in the same information packet if further failed nodes are discovered.
The number of 1-hop neighbors linked to a node mainly depends on the topology and the network density. In some situations, this number might be high and the time and energy required to perform the broadcast of the information packet may be too much energy consuming. For this reason, the sender can perform a ranking of its 1-hop neighbors before sending the message. The ranking can be based upon different parameters, e.g. minimum hop-count distance or higher received signal strength indicator (RSSI). The sender will select just a subset of nodes to send the message to starting from the highest ranked. The bigger the selected subset size is, the higher is the network update, but the higher is the resources consumed. The ranking aims to retrieve nodes that might have a higher probability to be interested in the current update. The on-demand table operations are summarized in Algorithm 1.

Algorithm 1 On-demand table update
```
1:  procedure NODE DETECTING A FAILURE
2:      discover a failed node;
3:      re-compute hop-count distance;
4:      serve the data packet;
5:      if failed node detected or hop distance changed then
6:          goto 16;
7:
8:  procedure NODE RECEIVING INFORMATION PACKET
9:      receive information packet;
10:     store failed node ID;
11:     invalidate paths with failed node;
12:     re-compute hop-count distance;
13:     if hop distance changed then
14:         goto 16;
15:
16: procedure INFORMATION PACKET OPERATIONS
17:     generate an information packet;
18:     perform a ranking of the 1-hop neighbors;
19:     select a subset of the ranked nodes;
20:     send the information packet to the selected nodes;
```

Authorized licensed use limited to: Northeastern University. Downloaded on November 23,2020 at 21:29:21 UTC from IEEE Xplore. Restrictions apply.

The update of the node structures performed by the information message has two main benefits: routing accuracy and energy saving. For instance, during future alternative route generations or failure recovery situations for data deliveries, the nodes are able to select the next hop quickly and correctly thanks to the up-to-date information stored. It can be inferred that the real benefits of the on-demand table update may not be always visible in a short period of time, but it might strongly help the network in the long term by saving energy and time. Figure 2 shows an example of the on-demand table update. At the beginning, N2 wants to send an uplink message to the head node via its path #1. Because of N1 failure, the packet is not delivered (2.A). N2 updates its own hop-count and discovers that it has changed from 2, passing via N1, to 3, passing via N6 (2.B). At this point, both the triggering conditions of the on-demand table update are satisfied since a failed node has been discovered and the hop distance has changed. Hence, N2 generates an information packet. After that, it pushes the data packet through its second path #2 via N6, since the data packet has higher priority. After this delivery is completed, the information message can be served. Since the network in this example is small, there is no need of the ranking operations, and the packet can be sent to all of the 1-hop neighbors, i.e. N3 and N6 (2.C). Upon the reception of the information message, the nodes update the hop-count of N2 and re-compute their own distances. The hop-count of N6 is not changed since it relies on N5. On the contrary, N3's distance has changed from 3 to 4 (2.D) since it was relying on N2. At this point, the condition of the on-demand table update is triggered again. N3 generates another information message and sends it to its 1-hop neighbors (2.E). The same process is going to happen with N4 (2.F). Additionally, if N3 has the information regarding all of the intermediate nodes of its paths, it might decide to invalidate its path #1 since it contains a failed spot. Same actions can be performed by the other reached nodes. In this example, the benefits of the update are immediately visible. Indeed, N3 is able to mark its path #1 failed and to select its #2 as active, saving energy and time. Additionally, all the nodes have the information regarding the hop-count distances of their neighbors correctly up-to-date.

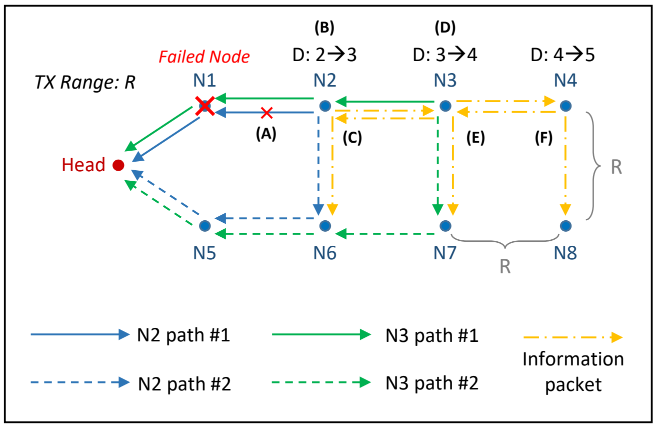

Fig. 2. On-demand table update example.

TABLE II
NOTIFICATION PACKET OVERVIEW

Packet ID	...	Failed node ID	Path ID failed	...

C. *On-demand route update*

The on-demand route update is another technique to deal with node failures in the network. While the on-demand table focuses on updating the hop-count and neighbor information, the on-demand route instead aims to provide an update of the routes containing a failed spot. The on-demand route update might be triggered after a failed node is discovered. When a sender discovers a failure, the link sender-failed node is recorded unavailable at the sending node. Then, the sender checks its routing tables to determine whether that link is used by other routes. If so, the on-demand route update is triggered. The sending node sends a unicast notification packet (Table II) to the original source of the path comprising the failed link if that link is part of an uplink route, or to the head node if that link is part of a downlink path. When the final node receives the notification message, it invalidates the notified path so that the path containing the failed link is not used in a future data delivery. Any intermediary nodes may also record any path containing the failed link as being invalid. As for the on-demand table, the notification packets of the on-demand route update have lower priority than the original data packet, meaning that they are sent after the current data packet. The on-demand route operations are summarized in Algorithm 2. The benefits of this technique are visible when a node skips the routes comprising a failed link thanks to the notification updates. This benefit situation may or may not happen in the future based on the failure locations and the topology structure. If a failed link is used by the current active path of a node, the benefit is immediately displayed. However, if the failed link is used by a non-active path, the on-demand route update will take more time and more failure nodes to show its benefits.

Algorithm 2 On-demand route update
```
1:  procedure NODE DETECTING A FAILURE
2:      discover a failed node;
3:      mark link with failed node unavailable;
4:      serve the data packet;
5:      check routing tables for failed link;
6:      for each upstream failed link found do
7:          send a notification packet downstream;
8:      for each downstream failed link found do
9:          send a notification packet upstream;
10:
11: procedure NODE RECEIVING NOTIFICATION PACKET
12:     receive notification packet;
13:     store failed node ID;
14:     if notification has reached final destination then
15:         invalidate paths with failed link;
16:     else
17:         forward notification;
```

Authorized licensed use limited to: Northeastern University. Downloaded on November 23,2020 at 21:29:21 UTC from IEEE Xplore. Restrictions apply.

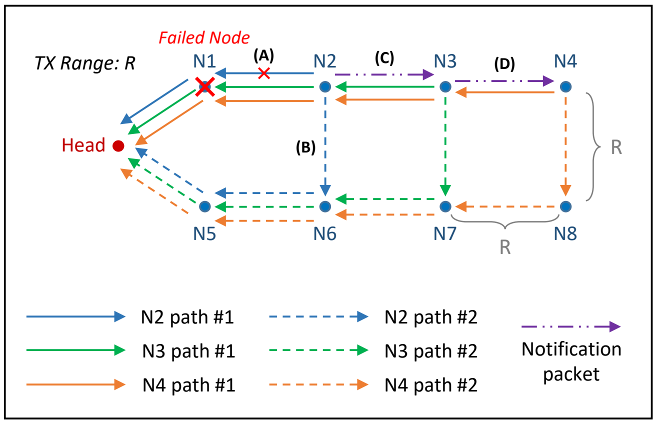

Fig. 3. On-demand route update example.

The on-demand updates can be used individually or combined together in order to update the network in a more comprehensive way in a failure event at the expense of time and resources. This allows the network to be more adaptable in response to failed node situations and able to self-heal to ensure effective and efficient data delivery through the network without needing for an external operator to continually maintain the system. On the other hand, if the failed nodes are quickly recovered by introducing a high maintenance cost thanks to a prompt technician intervention, no benefits might be provided to the system by the on-demand table and route update techniques.

Figure 3 shows an example of the on-demand route update methodology. Initially, N2 wants to send an uplink packet via its path #1. Upon attempting to do so, N2 discovers that node N1 has failed and the data packet cannot be properly received (3.A). At this point, N2 updates the information of node N1 marking the link N1-N2 as being unavailable. Because a failed node has been discovered, the condition for the on-demand route update might be triggered. First of all, N2 sends the current data packet through its path #2 via N6 (3.B). Then, N2 checks the routing tables to see if the link N1-N2 is used by other paths. It discovers that the link N1-N2 is used by N3 and N4 in their upstream routes. This is the proper condition that triggers the on-demand route update. N2 generates two notification messages, one with destination N3 and one with destination N4, and forwards these messages by following their correspondent downlink routes (3.C). After N3 has received the notifications, it invalidates its path #1 and forwards the second notification to N4 (3.D). Upon the notification reception, similarly N4 invalidates its path #1. In this example, the on-demand route update provides immediate benefits, since the routes involved in the notification were the current active paths #1 of the notified nodes. The nodes select as active their paths #2 and they will use directly those for the future data deliveries by saving time and resources.

III. Performance Evaluation

This section presents network simulation models and results performed using the Matlab [13] software environment to evaluate the proposed methods discussed in previous sections.

TABLE III
SIMULATION PARAMETERS.

Parameter	Value
τ_{AI} for head node	100 ms
τ_{AI} for BLE nodes	1 $second$
τ_{SW} equal to τ_{SI}	10 ms
Packet size	240 $bits$

A. Simulation setup

The network in this study is shown in Figure 4. It consists of a 3-floors topology which would simulate an indoor office or a student dormitory environment. Each floor is 4 $meters$ high and contains 10 nodes arranged in a 2x5 grid for a total of 30 nodes. Each node has a distance of 2 $meters$ from the other line, 12 $meters$ within a node in the same line, and 1.2 $meters$ from the ground. One head node is located in the central floor at a height of 3.9 $meters$, i.e. on the ceiling.

The channel modeling exploits the BLE physical signal with Gaussian Frequency Shift Keying (GFSK) modulation. The International Telecommunication Union (ITU) indoor propagation model [14] used takes into account the transmission frequency (f), the distance between transmitter and receiver (d) with a power loss coefficient (N) equal to 22, typical value for a commercial area in $2GHz$, and a floor penetration loss (P_f) of 6 dB per crossed floor (n) based on the nodes location. The resulting path loss (PL) equation is shown in (1).

$$PL_{d_0 \to d} = 20log(f) + Nlog(d) + P_f(n) - 28 \qquad (1)$$

A path loss threshold of 70 dB is used taking into consideration the BLE radio's receiver sensitivity and channel fading effect. This determines if two nodes are considered neighbors and are able to communicate directly and reliably. Additional parameters for the simulations are summarized in Table III. We make the assumptions that each node has 5 disjoint paths already stored in the routing tables towards the head node together with the hop-count values of its 1-hop neighbors [10] [12]. Each simulation cycle consists of the first phase with 1 second long where all the nodes wake up randomly and start the advertisement cycle. After that, the traffic is generated. Two main use cases are considered. The first tests the on-demand table update, while the second the on-demand route.

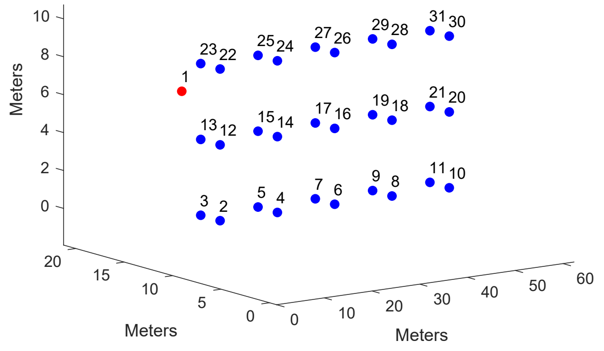

Fig. 4. Network topology used for the simulations in this study. In red the head node and in blue the other BLE nodes.

Authorized licensed use limited to: Northeastern University. Downloaded on November 23,2020 at 21:29:21 UTC from IEEE Xplore. Restrictions apply.

The background traffic for the first use case consists of one uplink packet generated by one node at random in the network with a distant of at least 2 hops towards the head. During each cycle, a node failure is injected along the path that the packet should follow to reach the head node in order to trigger the failure conditions to activate the on-demand table update process. The number of 1-hop neighbors to reach after the ranking process is fixed and varies between 1, 3, 5, 7 and 9. The system handles the delivery of a message to multiple 1-hop destination nodes in a sequentially unicast way creating the connections one at a time [15]. The sending node follows the following operations: connect to nearby MAC address; send an instance of the message; terminate the link; switch again in scan mode to establish another connection till all of the destination nodes have received the message correctly.

The second use case tests the on-demand route update. The background traffic consists of one packet uplink, while the failed node hop-count distance is fixed and varies between 1, 2, 3 and 4, the minimum and maximum values allowed in the network considered in this study. The source node, i.e. the node that generates the uplink message, is chosen between the nodes with a hop-count distance greater than the failed node in order to be able to inject the failure along the path that the packet should follow and thus trigger the on-demand route.

The main Key Performance Indicators (KPI) taken into considerations are latency and current consumption. The latency considers the average time needed by the system to complete all of the on-demand update activities from the time they are triggered. On the other hand, the current consumption takes into account the average instantaneous current needed by a node to perform the current operations. The energy model considers only the power consumption of the radio operations of transmitting/receiving, which are the most expensive in terms of energy in a wireless device. Their values are based on the Texas Instrument Bluetooth chipset *CC2642* [16] and are summarized in Table IV. In the first case only the current consumption of the node that discovers the failure is considered, since it is the only node more involved in the operations. On the other hand, in the second case the average of all of nodes current consumption is presented since more nodes in the network are involved in the on-demand route activities.

TABLE IV
ENERGY MODEL

State	**Time** $[\mu s]$	**Consumption** $[mA]$
Low-Power	/	0.0107
Preprocessing	1261	3.34
Postprocessing	685	2.45
Rx	184	6.47
Tx	150	7.47
Rx2Rx (change ch)	372	3.56
Rx2Tx (change ch)	370	3.43
Tx2Rx (change ch)	370	3.43
Tx2Tx (change ch)	372	3.56
Rx2Tx (no change ch)	150	5.49
Tx2Rx (no change ch)	112	4.66

B. *Results analysis*

The simulation results for the first case are shown in Figure 5. The latency increases linearly with the increasing of the number of 1-hop neighbors to reach. Its value starts with an initial offset since the node has to first send the data packet that has generated the failure discovery before serving the on-demand table information message. The 5.b shows the average current consumption during 1 $minute$ of simulation of the node that discovers the failure and sends the information packet. These values follow the same trend as the latency ones. However, these increments lead to an enhancement of the accuracy that ranges from 11% to 77%. The accuracy (5.c) is computed as the percentage of the average active 1-hop neighbors per node to whom the information message is sent to. It shows the fraction of 1-hop neighbors that have the information up-to-date at the end of the on-demand table. In this study the number of 1-hop neighbors per node range from 9 to 17 based on the location with an average of 12.6 neighbors. The accuracy is a good indicator of the network maintenance rate. The more resources are spent in the on-demand table process, the higher is the maintenance achieved.

Figure 6 presents the simulation results for the second use case. The increasing of the failed node's hop distance leads to a decreasing of all of the three KPIs taken into account: latency (6.a), current consumption (6.b) and total number of notification packets generated (6.c). As the failed node moves away from the head node, it is used by less nodes in their routes. This leads to a less number of on-demand

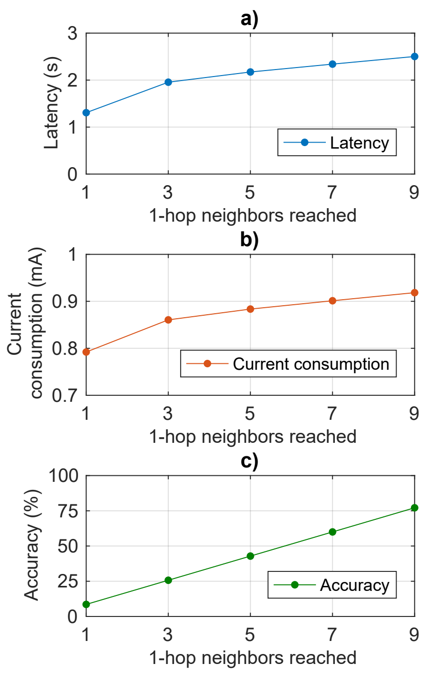

Fig. 5. First case with on-demand table update results: a) average latency for the completion of the on-demand table activities from the time they are triggered; b) average current consumption in 1 $minute$ of simulation for the node that has discovered the failure and has triggered the on-demand table update; c) accuracy of the on-demand table update given by the number of 1-hop neighbors reached over the average active 1-hop neighbors per node.

Authorized licensed use limited to: Northeastern University. Downloaded on November 23,2020 at 21:29:21 UTC from IEEE Xplore. Restrictions apply.

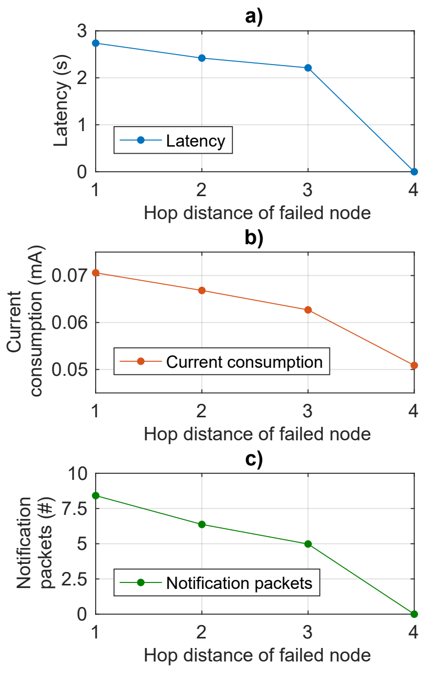

Fig. 6. Second case with on-demand route update results: a) average latency for the completion of the on-demand route activities from the time it is triggered; b) average node current consumption in 1 *minute* of simulation for all the nodes in the network; c) average number of notification packets generated by the network when the on-demand route update is triggered.

route notification packets generated since fewer nodes have to be informed. The overall latency and power consumption decrease accordingly. The current consumption in 6.b shows the average values for all the nodes in the network in 1 *minute* of simulation, in contrast with 5.b that presents only the current consumption of the node that triggered the on-demand update. It can be noticed that with a failed node's hop distance equal to 4 the number of notification packets is zero. This is because the node that discovers the failure has no paths stored in its routing tables that exploit the failed link, therefore no notification packets are required. This doesn't imply that the failed node is not used in other routes, but it only means that the specific link discovered is not used. The failed node could still be used by other surrounding nodes through other links. In this situation, the on-demand table update might be very useful. In fact, the information packet might be able to inform the neighbors about the node failure and they could invalidate any routes comprising the failed spot. Compared to the on-demand table, the on-demand route needs more information to generate its notifications, but, on the other hand, the notification packets have a more specific scope and destination nodes to reach.

The use of the on-demand updates means that with a smaller amount of resources spent in advance, the system is able to save energy and latency in future deliveries. Since the routing tables and information of the neighbors are already up-to-date, the senders are capable of selecting the next hop properly, without needing to spend time on discovering the failures. On the other hand, if the failed node is not used in future deliveries, or it is quickly repaired, there is no clear benefit in using the on-demand updates, since they are a mechanism of network self-maintenance. The real savings may also vary based on the network topology and density and how the system handles broadcast deliveries, failure discovery and recovery.

IV. Conclusion

In this paper we present and demonstrate an effective and efficient self-healing method for network routing. The method includes two on-demand update techniques. One method targets all of neighboring nodes with a ranking function while the other method provides a more focused update using early routing information. The methods are demonstrated to be efficient based upon KPIs for latency, current consumption and overall network self-maintenance percentage. In particular, when the updates reach nodes directly affected by the failure, as it has been shown in Section II, the methods provide strong and robust mechanisms of self-healing. On the other hand, the methods are weak when a technician intervention is actuated quickly or when the updates are not able to reach interested nodes. Their cost difference and effectiveness is varying depending on the selected system parameters, topology, failure and maintenance scenarios. The authors believe that the methods presented can help networks self-heal and robustly route around failed nodes with minimal effort and energy cost compared to previously published methods.

References

[1] H.-S. Kim, J. Lee, and J. Jang, "Blemesh: A wireless mesh network protocol for bluetooth low energy device," *International Conference on Future Internet of Things and Cloud*, Rome, 2015.
[2] A. Scaglione, M. Coates, M. Gastpar, J. Tsitsiklis, and M. Vetterli, "Introduction to the issue on gossiping algorithms design and applications," *IEEE J. Sel. Topics Signal Processing*, August 2011.
[3] B. K. Maharjan, U. Witkowski, and R. Zandian, "Tree network based on bluetooth 4.0 for wireless sensor network applications," *Embedded Design in Education and Research Conf. (EDERC)*, September 2014.
[4] K. Mikhaylov and J. Tervonen, "Multihop data transfer service for bluetooth low energy," *13rd ITS Telecomm.*, Finland, November 2013.
[5] Z. Guo, I. G. Harris, L. Tsaur, and X. Chen, "An on-demand scatternet formation and multi-hop routing protocol for ble-based wireless sensor networks," *IEEE Wireless Com. and Netw. Conf. (WCNC)*, March 2015.
[6] C.-K. Lin and O. Kure, "Energy-aware path selection in mobile wireless sensor networks: A dynamic bayesian game approach," *IEEE PIMRC Conference*, Tokyo, 2009.
[7] H. B. Nakil, P. R. Marques, H. Ajay, A. Ranjan, and A. Singla, "Rerouting network traffic after link failure," US8953441B2, 2013.
[8] Y. Nishimura, H. Sakauchi, and S. Hasegawa, "Self-healing network with distributed failure restoration capabilities," US5235599, 1990.
[9] G. Shi, J. Chen, and H. Gong, "Method, device and system for updating routes after node fails in p2p network," US8248919, 2009.
[10] C.-K. Lin, D. Villa, A. Kuenzi, M. Lang, and A. Tiwari, "Method and system for data transfer in a bluetooth low energy network," 107431EP01, 2019.
[11] D. Villa, C.-K. Lin, A. Kuenzi, and M. Lang, "On-demand table and route update after a node failure in a wireless network," 123926EP01, 2019.
[12] C.-K. Lin, D. Villa, A. Kuenzi, and M. Lang, "Adaptive multipath routing failure recovery in a wireless network," 124326EP01, 2019.
[13] MATLAB, *version 9.1.0.441655 (R2016b)*. Natick, Massachusetts: The MathWorks Inc., 2016.
[14] V. Sucasas, G. Mantas, and S. Althunibat, *Broadband Communications, Networks, and Systems*. Springer, 1st ed., January 2019.
[15] D. Villa, C.-K. Lin, A. Kuenzi, and M. Lang, "Broadcast delivery techniques in a wireless network," 123931EP01, 2019.
[16] T. Instruments, *CC2642R SimpleLink™ Bluetooth® 5 low energy Wireless MCU*, swrs 194 ed., January 2018.

Authorized licensed use limited to: Northeastern University. Downloaded on November 23,2020 at 21:29:21 UTC from IEEE Xplore. Restrictions apply.