Reliable IoT Firmware Updates: A Large-scale Mesh Network Performance Investigation

Ioannis Mavromatis*, Aleksandar Stanoev*, Anthony J. Portelli*, Charles Lockie†,
Marius Ammann†, Yichao Jin*, and Mahesh Sooriyabandara*
* Bristol Research and Innovation Laboratory (BRIL), Toshiba Europe Ltd., Bristol, UK
† Department of Electrical and Electronic Engineering, University of Bristol, Bristol, UK
Emails: {Ioannis.Mavromatis, Aleksandar.Stanoev, Yichao.Jin}@toshiba-bril.com

Abstract—Internet of Things (IoT) networks require regular firmware updates to ensure enhanced security and stability. As we move towards methodologies of codifying security and policy decisions and exchanging them over IoT large-scale deployments (security-as-a-code), these demands should be considered a routine operation. However, rolling out firmware updates to large-scale networks presents a crucial challenge for constrained wireless environments with large numbers of IoT devices. This paper initially investigates how the current state-of-the-art protocols operate in such adverse conditions by measuring various Quality-of-Service (QoS) Key Performance Indicators (KPIs) of the shared wireless medium. We later discuss how Concurrent Transmissions (CT) can extend the scalability of IoT protocols and ensure reliable firmware roll-outs over large geographical areas. Measuring KPIs such as the mesh join time, the throughput, and the number of nodes forming a network, we provide great insight into how an IoT environment will behave under a large-scale firmware roll-out. Finally, we conducted our performance investigation over the UMBRELLA platform, a real-world IoT testbed deployed in Bristol, UK. This ensures our findings represent a realistic IoT scenario and meet the strict QoS requirements of today's IoT applications.

Index Terms—IoT; Bluetooth; Large-scale Testbed; Firmware Update; IEEE 802.15.4;

I. INTRODUCTION

Internet of Things (IoT) has become synonymous with everyday computing. It provides the ability for interconnected devices to exchange information over a wireless medium without human intervention [1]. IoT systems and platforms are already utilised in home environments for voice assistants and security systems to energy, heating, and lighting control [2]. Moving forward, IoT is expected to impact commercial applications in a number of areas, including Industry 4.0, connected vehicles and smart grids [3].

Operational stability and security are two critical factors of every IoT network and application. Both can be achieved with frequent firmware roll-outs [4]. However, autonomous firmware update without physical proximity is susceptible to adverse wireless medium conditions, can drastically increase the roll-out times, or lead to downtimes. Interconnected IoT devices make it possible for billions of nodes to exchange information and transform raw data into meaningful inferences. This was made possible due to the standardisation of IoT communication protocols. These constrained devices, having limited resources, can connect and form huge distributions of networks [5]. Such networks can ensure massive firmware roll-outs over large geographical areas. However, the current IoT protocols were initially designed to work within home environments [5] and do not scale well in city-wide deployments [6].

The contribution of this paper is two-fold. Initially, we investigate the requirements for such a firmware roll-out. Based on that, we explore the coexistence, cooperation, and interoperability of IoT communication protocols in a large-scale city-wide deployment. We intend on showing how different protocols perform within a real-world environment and their limitations as the number of devices increases. We also discuss how Concurrent Transmissions (CT) [7] can extend the scalability of IoT protocols. Well-established IEEE 802.15.4 protocols such as the Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA) [8] have been widely tested and implemented in the IoT world. For our investigation, we use CSMA-CA as a benchmark. As a second step, we conduct a large-scale performance investigation using a CT-based implementation [7]. Our analysis shows that CT can increase the current system performance in a real-world environment, especially when the number of interconnected devices increases.

Our performance investigation was based on Urban Multi Wireless Broadband and IoT Testing for Local Authority and Industrial Applications (UMBRELLA). UMBRELLA is currently deployed in Bristol, UK, and is a large-scale urban testbed. Owing to its size and versatility, it can be used as a playground for wireless experimentation and assessing Key Performance Indicators (KPIs) for large-scale urban IoT scenarios. Our investigation focuses on various KPIs like mesh creation times, reliability and data throughput, linking them with an example application of a large-scale firmware roll-out.

The rest of the paper is organised as follows. Sec. II describes the system architecture, our scenario, and the experimental setup. Sec. III presents the workflow and provides a better insight on the implementations and the large-scale evaluation. Our comparative results can be found in Sec. IV. Finally, the paper summarises our investigation in Sec. V.

II. SYSTEM DESCRIPTION

Our experimentation is based on a realistic IoT scenario. We consider a use-case where a firmware update is rolled out to a wide-reaching network of IoT devices. This can be the case when, for example, a security vulnerability is found, subsequently, a patch is introduced, and the new firmware is reflashed on all the wireless IoT interfaces. Within such a scenario, the downtime on the network should be minimised while ensuring the new firmware reliably reaches all the
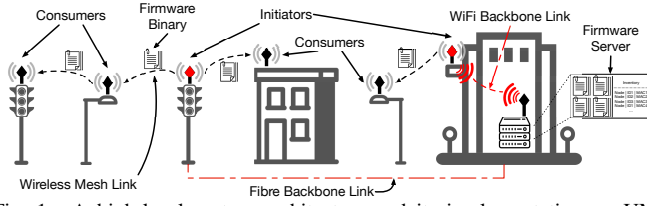
Fig. 1. A high-level system architecture and its implementation on UM-BRELLA. The nodes share the firmware binaries over the wireless mesh network. The initiators are reachable from the server via a backbone link.

destination nodes without significant delays. To emulate that, a $100\,\mathrm{kB}$ file is used to represent the firmware binary. This is roughly the size of an uncompressed binary file for the Nordic Semiconductor nRF52840 [9] System on Chip (SoC). Data compression could be used to reduce the size of the payload substantially. However, this would require the receiving nodes to perform the decompression. Such functionality is somewhat onerous when considering the limited computing capacity of IoT devices, particularly if they are battery-powered. Therefore, the uncompressed file size was chosen for our experimentation.

We assume that a firmware server stores the firmware images and manifests, and distributes them to the IoT devices (as in Fig. 1). The firmware binary is sent to all firmware consumers over a wireless mesh link. What is more, the server has direct access to the roll-out initiators (source nodes) and initiates the deployment. To do so, a list of all active consumers is always stored on the server-side and can be periodically updated and maintained. In the following sections, we describe the wireless protocol stacks used for our investigation.

A. IEEE 802.15.4, Bluetooth and the Different PHYs

Bluetooth Low Energy (BLE) and Low Rate Wireless Personal Area Networks (LR-WPANs) (IEEE 802.15.4) have emerged as wireless communications technologies of choice in many IoT applications. We utilise both during our investigation. The most recent version (BLE 5) supports four PHYs that largely differ in terms of data rate and robustness [10], i.e. $2\,\mathrm{Mbps}$, which doubles the nominal throughput of the original $1\,\mathrm{Mbps}$ PHY, and two coded PHYs with coding rates of $1/2$ and $1/8$ (i.e., the $500\,\mathrm{Kbps}$ and $125\,\mathrm{Kbps}$ respectively). The IEEE 802.15.4 supports a datarate of $250\,\mathrm{Kbps}$ in the $2.4\,\mathrm{GHz}$ frequency band and up to $40\,\mathrm{Kbps}$ at the sub-GHz one. For our experimentation, we will use all four PHYs from BLE and the $250\,\mathrm{Kbps}$ PHY for IEEE 802.15.4.

Both IEEE 802.15.4 and BLE operate in the global unlicensed ISM band of $2.4\,\mathrm{GHz}$. The same frequency band is also used by wireless technologies such as IEEE 802.11. Furthermore, IEEE 802.15.4g operates in the sub-GHz frequency band as well ($868\,\mathrm{MHz}$ to $868.6\,\mathrm{MHz}$ in Europe and $902\,\mathrm{MHz}$ to $928\,\mathrm{MHz}$ in North America). Our experimentation will focus on the $2.4\,\mathrm{GHz}$ frequency band. The above result in a cross-technology interference that affects the Quality-of-Service (QoS) of the network, particularly the reliability and latency that could lead to application failures. The appropriate physical layer should be considered when developing an IoT application to ensure the highest QoS. The impact is even more prominent in the case of CT-based communications [11].

B. Carrier Sense Multiple Access with Collision Avoidance

CSMA-CA is a random access protocol and works by sensing the channel before transmitting to avoid collisions. When the channel is occupied, it waits for a variable amount of time (back-off period) before checking again. Should a collision occur, the back-off period is increased to avoid the next packet also colliding [12]. The main advantage of CSMA-CA is its simplicity, as it does not require time synchronisation, scheduling between nodes, or the need to wait to join a network. However, as the number of nodes increases, it becomes saturated, and collisions become more likely. Finally, CSMA-CA operates on a predefined fixed frequency, thus is susceptible to environmental interference from nearby transmitters operating on the same or adjacent channels in the ISM band (such as WIFI).

For our experiments, we use CSMA-CA with Routing Protocol for Low-Power and Lossy Networks (RPL) [13]. RPL is an optimised routing protocol for wireless networks susceptible to packet loss. The routing tables are maintained as a Destination-Oriented Directed Acyclic Graph (DODAG). Each node is assigned a rank that increases as we move away from the root node. Two modes are supported, i.e. storing and non-storing modes. For the storing mode, routing tables are stored on each node, imposing a significant memory footprint in large networks and being hard to maintain consistency. For the non-storing mode, IPv6 source routing is employed. This means that routing tables are not stored in the nodes but are embedded in the source routing header. In larger networks with many hops, this can lead to increased header size.

While using the IEEE 802.15.4 channel, there is a limit to how much data can be sent. Researchers in [14], introducing various modifications in the Contiki stack, reliably achieving up to $45\,\mathrm{Kbps}$ with a CSMA-CA channel, using only ten devices. However, as described, increasing the load to $100\,\mathrm{Kbps}$ introduces large latencies and significant packet loss. For our implementation, a standard CSMA-CA stack is considered, thus the throughput is expected to be less than that.

C. Concurrent Transmissions and Synchronous Flooding

CT is the concept that nodes synchronously transmit in-contention with their neighbours. Synchronous Flooding (SF) builds on the idea of CT, and it can support one-to-all communication within a single flood, minimise the latency and enhance reliability. Its time-synchronised nature can help decouple network synchronisation from other network processes. An example of SF-based protocol is Atomic [7], which showed great results in terms of reduced latency, better reliability and energy efficiency on small testbeds. Finally, Atomic supports Multicast Protocol for Low-Power and Lossy Networks (MPL) and will be the SF protocol of use.

During an Atomic period Δ_{SF}, a packet is flooded across the whole network. Each Δ_{SF} is partitioned into slots. The maximum number of slots (MAX_SLOTS), and the maximum number of transmissions (MAX_TX), are configured at the start of each flooding. A node transmits a packet at the start of the Atomic period to all nearby listening devices. Then, all devices that successfully received the packet re-transmit it alongside the source node. This repeats until each node has
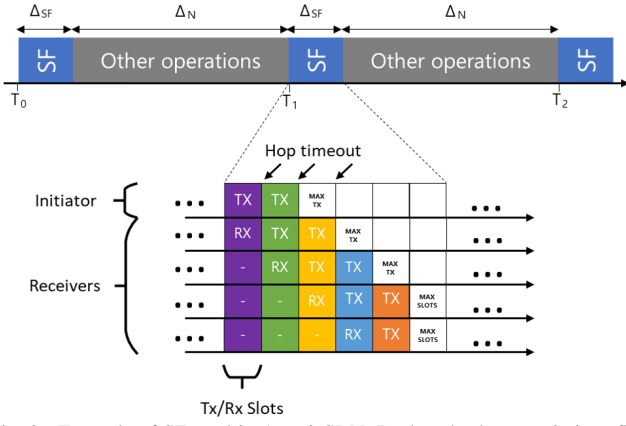
Fig. 2. Example of SF used in AtomicSDN. Back-to-back transmissions flood the network with minimal latency. The MAX_TX and MAX_SLOTS values are a trade-off between latency, and greater temporal/frequency diversity.
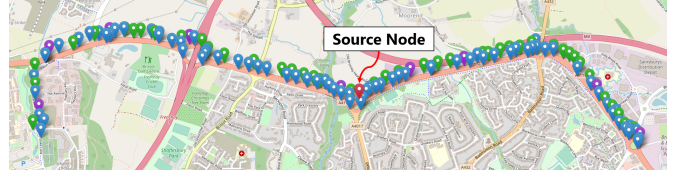

Fig. 3. The UMBRELLA network. All nodes are installed on public lampposts across a road of ~7.2km. The colors represent the nodes connectivity, i.e. green is fibre connected, blue is WiFi connected, and purple is fibre connected and can act as a LoRa gateway too. The red node is our experiment source node.
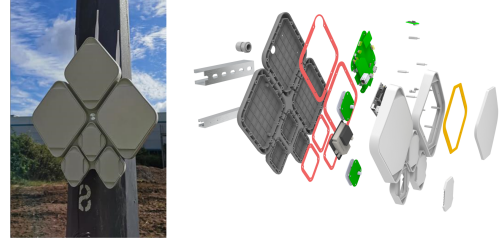

Fig. 4. Umbrella Node on a lamppost, with its exploded view.

resent the packet MAX_TX times. With each re-transmission, the data spread to exponentially more nodes. An example of the above can be seen in Fig. 2. Starting from the initiator, a node transmits a packet repeatedly until slot MAX_TX. Then, all others receive the packet and relay that during the next slot to all the forwarding nodes. More information about Atomic can be found in [7].

With such an approach, nodes receive the same symbol from multiple transmitters. This results in greater reliability for the whole network. Facilitating the propagation of control messages across the network in a flood and within dedicated control timeslots, Atomic allows an operation without the knowledge of the topology and benefits from the spatial and temporal diversity inherent within flooding protocols. Finally, Atomic uses channel hopping to increase performance, which avoids sources of interference on a fixed frequency and stops the Atomic flood from saturating one channel for too long.

D. UMBRELLA Testbed and Network Setup

As mentioned, UMBRELLA testbed [15] was used for our evaluation. UMBRELLA supports various IoT-related applications and use-cases. It allows the users to develop different applications and deploy them as containers on the provided edge nodes (UMBRELLA nodes). The applications supported range from air quality monitoring, street light maintenance, swarm robotics, private 5G for warehousing, logistics, and large-scale *over-the-air* wireless experimentation. The large-scale wireless testbed was the particular functionality that we utilised for our performance investigation.

UMBRELLA consists of ~200 nodes, installed on public lampposts and buildings, spread across South Gloucestershire region in the UK. The core UMBRELLA testbed is installed across a ~7.2km stretch of road (Fig. 3). Each node annotated in Fig. 3 is equipped with ten sensors (e.g., Bosch BME680, accelerometers, microphones, etc.) and seven network interfaces. The two interfaces used for experimentation are a Nordic Semiconductor nRF52840 [9] and a Texas Instruments CC1310 [16]. Our investigation was based on nRF52840. Two more interfaces (WiFi and fibre one) are used for backbone connectivity. Finally, all applications are supported by a unified

backend implementation. The developed platform provides the required messaging interfaces and protocols and some high-level APIs that an end-user can leverage to send requests, collect log files, or process the data and visualise them.

Between the nRF52840 interface and its dipole antenna exists a Skyworks RF Front-End Module [17], integrating a Low Noise Amplifier (LNA) and Power Amplifier (PA). This results in $22\,\mathrm{dB}$ of TX power gain, and increases RX sensitivity up to $6\,\mathrm{dB}$, approximately doubling the range of a typical IoT device [17]. Fig. 4 shows an Umbrella node attached to a lamppost. Each rhombus segment contains custom PCBs, all connected to a main processing unit (Raspberry Pi 3b+ Compute Module [18]). The UMBRELLA Raspberry Pi runs Raspbian GNU/Linux 10 (buster) and a custom kernel based on ver. 4.19.95-v7+. Finally, the three dipole antennas seen in Fig. 4 are for the $2.4\,\mathrm{GHz}$ nRF52840 (left), the sub-GHz CC1310 (middle), and the $2.4\,\mathrm{GHz}$ WiFi (right) interfaces.

III. LARGE SCALE EXPERIMENTS

As shown in Fig. 1, all firmware binaries are exchanged via the wireless mesh network (the nRF52840 interface in our case). Furthermore, all nodes are also accessible via a backbone link (WiFi or fibre), used for data collection, monitoring, and experiment execution purposes. The dedicated backbone link enhances the: 1) results collection without interference or disruption of on-going experiments, 2) available bandwidth to support large scale multi-user parallel experiments. At the beginning of each experiment, all nodes are assigned a unique ID and a role, i.e., source or receiver. The source node is the roll-out initiator (as described in Sec. II). Only a single node is assigned this role. The receivers act as the firmware consumers. With regards to Atomic, the source node also manages the SF periods and sets up the routing and timing for all nodes. The chosen source node (Fig. 3) is centrally located, giving us roughly an equal amount of nodes on each side of the network.

The $100\,\mathrm{kB}$ of data (emulated firmware) is discretised into packets, utilising the maximum payload size available without
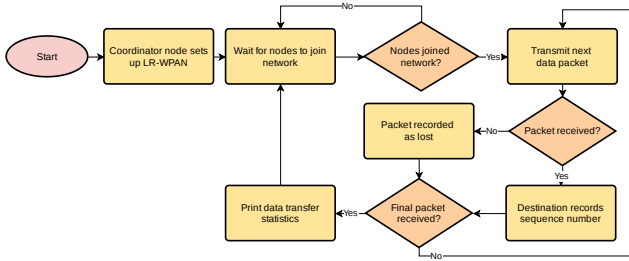
Fig. 5. Experiment flowchart. The source node initiates the network setup and is responsible for the transmission of the firmware later. When all packets have been sent, the statistics of each node are collected.

causing fragmentation (Tab. I). The remaining header bytes are used for control information, source and destination addresses and a checksum for the frame verification at the reception. Once the network connection is established, all packets are sent sequentially via the nRF52840 interface of the source node. Each packet contains a unique identifier, which is their sequence ID, so the firmware consumers are aware of which packets were received or lost. Fig. 5 shows the high-level structure of the procedure followed.

A. Transmitting and Receiving Sides

When using Atomic, we schedule a stream of packets for transmission at the end of each Atomic period Δ_{SF} (Fig 2). Our protocol stack is based on Contiki-NG [19]. Thus, we use various control processes and callbacks from the existing stack. A control process scheduled on a $1\,\mathrm{s}$ timer checks the readiness of the network and the completion of a transfer. Once the connection is established, the `send_next_packet` function from Contiki-NG is polled, and the "next packet" is added to the queue and sent once the current process finishes. On the other hand, when CSMA-CA is used, we schedule a `send_next_packet` process in regular intervals. After preliminary experimentation, we identified the timing required for our network to maximise the throughput without causing packets to be dropped due to timeouts.

On the receiving end, a node, upon joining the network, waits for a stream of packets to arrive. A stopwatch timer is triggered when the first packet is received, and the successful receptions are monitored. We maintain two lists monitoring their unique IDs, i.e. "packets received" and "packets lost". "Packets lost" are the ones that arrived out of order (e.g., their sequence ID preceding the last packet received), have a checksum fail, or are not received at all. To emulate a real-world-like scenario, simplicity, and fair comparison at a large-scale, retransmissions were set to 0. In a real-world application, this could be handled by a higher layer, e.g., passing the list of the lost packets to the transmitter and requesting the retransmission of the missing payloads. When the last packet is received, the node stops its local timer and prints its statistics.

B. Experimental Pipeline

For our evaluation, an automated deployment pipeline was implemented. To simplify the experiment execution, the same firmware code is uploaded on all nodes, containing both the roll-out initiator and firmware consumer codes. Using the Contiki-NG deployment service, the role of each node is decided

during run-time, i.e. hardcoding a single node (the same) as the Atomic SF initiator and the firmware roll-out source. The compiled binaries are uploaded on all nodes before flashing. Upon successful uploading, all nodes are flashed in parallel, and the experiment is initiated.

At the end of the experiment, all the results are recorded on the Raspberry Pi via the serial interface. They are later collected centrally on our UMBRELLA servers for post-processing. It was observed that the interaction with the serial interface causes a considerable delay for the SoC. Thus, it was decided to collect all the data at the end of each experiment. Furthermore, old-executed binaries can compromise the results when running sequential experiments (e.g., due to backbone network downtime nodes could be left earsplitting old data on the channel). Therefore, all nodes were flashed with a dummy binary at the end of each experiment to mitigate that. Finally, all the above processes and steps are automated with Ansible and a series of scripts to ensure smooth execution.

IV. Performance Investigation

For our performance investigation, we started with a small-scale experiment to identify the maximum achievable datarate. For that, we ran a small-factor experiment within a controlled lab environment using both Atomic and CSMA-CA and four nodes. Particularly for CSMA-CA, in order to find the optimal transmission time, we ran multiple experiments decreasing the inter-packet time until the receiver became unreachable. The last stable configuration was used as our time interval.

For our large-scale experiment, roughly ~150 nodes were used. These are the nodes seen in Fig. 3 and is a subset of the core UMBRELLA network. The rest of the nodes were excluded either due to being out-of-range or being installed inside buildings (thus not capable of forming a mesh with the outdoor nodes). All the nodes eastwards from the source are installed in a denser setup, with an average distance separation of ~$87\,\mathrm{m}$. The nodes westwards are separated by ~$94\,\mathrm{m}$ on average. Finally, channel no. 26 from the IEEE 802.15.4 frequency band was used for CSMA-CA, and both small- and large-scale experiments (carrier frequency of $2.480\,\mathrm{GHz}$). This channel was chosen due to its minimal observed interference after a channel sounding on the available spectrum.

A. Small-scale Comparison

Tab. I shows a summary of the results and the experimental configuration used. The time period (measured in ms) is the interarrival time between two packets and is the lowest value perceived with the network still reliably transferring data. These results indicate a maximum achievable data rate under ideal-like conditions. As seen, Atomic managed to outperform CSMA-CA in terms of the throughput perceived. However, as we will see in the next sections, more conservative configuration values should be used under a real-world scenario, and thus the achieved data rate decreases.

B. Atomic and CSMA-CA: a large-scale experiment

For the large-scale experiment, and using the indicative values from Tab. I, we adapted several parameters to fit the larger scale of the experiment. More specifically, the payload

TABLE I
SMALL SCALE THROUGHPUT EXPERIMENT RESULTS

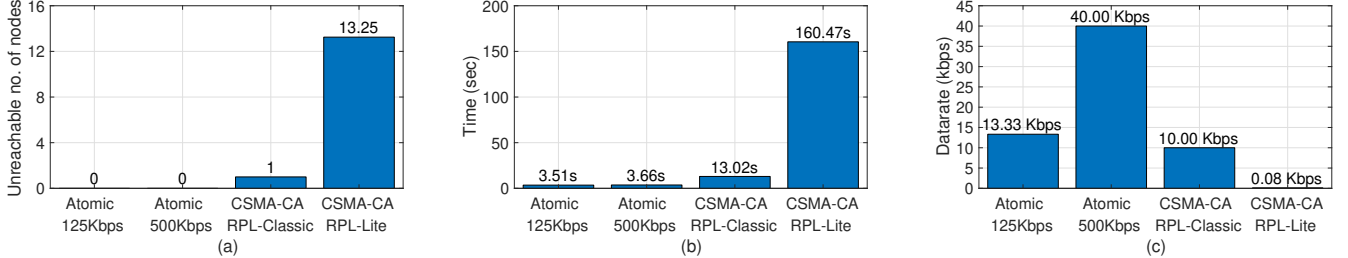| MAC	PHY	MAX_TX	MAX_SLOTS	Time Period	MTU	Payload Size	Throughput	Datarate (%)
Atomic	BLE 2 Mbps	3	7	16 ms	256 B	230 B	115.19 Kbps	5.76%
Atomic	BLE 1 Mbps	3	7	20 ms	256 B	230 B	92.17 Kbps	9.22%
Atomic	BLE 500 Kbps	3	7	29 ms	256 B	230 B	63.56 Kbps	12.71%
Atomic	BLE 125 Kbps	3	7	77 ms	256 B	230 B	23.94 Kbps	19.15%
Atomic	IEEE 802.15.4 250 Kbps	3	7	29 ms	127 B	121 B	33.40 Kbps	13.36%
CSMA	IEEE 802.15.4 250 Kbps	1	-	31 ms	127 B	86 B	21.98 Kbps	8.79%

Fig. 6. A comparison between Atomic and CSMA-CA. (a) shows the number of nodes non-reachable on average per experiment, (b) is the time required to form the mesh network, and (c) is the maximum achievable datarate.

size of the CSMA-CA was reduced to 50 Kbps. This is because the routing table included inside the MAC header was larger, thus, more space was required to accommodate that. Also, both RPL-Classic and RPL-Lite were used for CSMA-CA. A worst-case unicast CSMA-CA scenario was chosen to approximate the effects of a MPL approach, as in [20]. With what regards Atomic, the MAX_SLOTS was increased to 8, and the MAX_TX was increased to 12 to accommodate the increased number of nodes that take part in the experiment. These values were chosen after an exploratory investigation on the testbed. We chose the best performing pair for the given setup. Different time periods were tested, these being between 50 ms to 500 ms with an interval of 25 ms and between 500 ms to 800 ms with an interval of 100 ms, and two PHY were used, i.e., 125 Kbps and 500 Kbps. Finally, the denser side of the network was utilised for this experiment (all nodes eastwards from the source node).

We compare Atomic and CSMA-CA (Fig. 6) against the number of nodes able to form a mesh network, the time required to join, and the achievable datarate while sending 100 kB of data. The nodes reached and the mesh join time are averaged across all values, while the datarate is the maximum achieved. As shown, in terms of the number of unreachable nodes, Atomic outperforms CSMA-CA (Fig. 6-(a)), constantly forming a mesh network with all the available nodes. Particularly when using RPL-Lite, we observed that about 1/10th of the nodes was not part of the end network. When rolling out a firmware update, it is necessary to ensure that all the available nodes receive it. Atomic was proven a lot more reliable in such a scenario. Similar results can be seen in Fig. 6-(b) where Atomic requires less time to form the network, thus achieving less downtime and being less susceptible to network changes. This reflects on the increased perceived datarate (Fig. 6-(c)). As shown, Atomic, even with a lower PHY capacity (125 Kbps against 250 Kbps for CSMA-CA), still manages to achieve better performance. For the case of PHY 500 Kbps the results are even more significant, showing four times higher datarate than CSMA-

CA. These results show that the firmware binaries can reach the end destination faster, thus reducing the network's "downtime" and "maintenance" phases while making the roll-out less prone to errors or external factors.

C. Atomic: A more thorough experimentation

Above, we presented the dominance of Atomic compared to CSMA-CA. We now further investigate Atomic performance using the entire UMBRELLA network. A single experiment was executed throughout the entire network, but as we describe, we considered the performance perceived on the dense and sparse sides independently. As before, we used 125 Kbps and 500 Kbps PHYs. Due to the limited size of the paper, some of the results will only be described in text.

In Fig. 7 we compare Atomic using two different densities and PHYs and the same periods mentioned above. The MAX_SLOTS was set to 8, and the MAX_TX was set to 16. For 125 Kbps, the mesh join time was slightly reduced compared to the experiment in Sec. IV-B. The larger number of nodes around the source node helps with the mesh initialisation, insofar, sightly decreasing the join time. Considering the 500 Kbps PHY, as before, for the dense network side, the increased number of nodes around the source decreases the time required. However, when considering the sparser side, it is shown that the time is almost doubled. The increased distance separation and the higher PHY result in more hops required for each node to be reached, thus the difference in the results. The number of unreachable nodes is again zero for both PHYs, implying that all nodes were part of the mesh formation. The above findings give us a good indication of the ability of Atomic and SF to form networks under various scenarios and conditions.

We later investigate the reliability of Atomic when rolling out the firmware update on all the consumers. As seen in Fig. 8, lower periods can increase the maximum datarate, however significantly reduce the reliability of the protocol. This is because short Atomic periods reduce the number of nodes
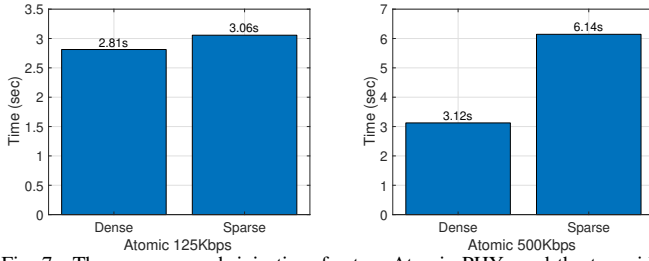
Fig. 7. The average mesh join time for two Atomic PHYs and the two sides of the UMBRELLA network.
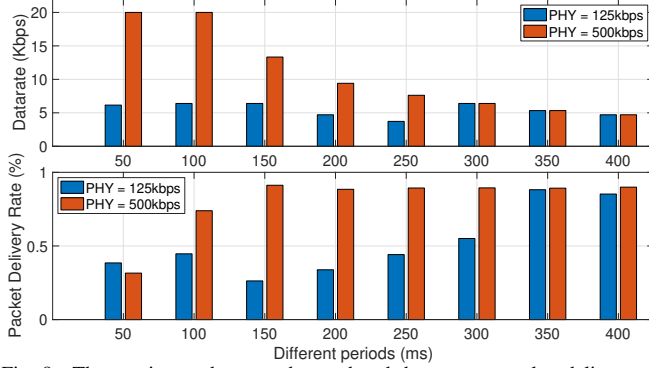

Fig. 8. The maximum datarate observed and the average packet delivery rate between all the UMBRELLA nodes.

participating in the Atomic SF operation. Thus, many nodes become unreachable until the next Atomic period. When the period is increased, the datarate is reduced. Allowing more time for Atomic to configure reduces the available channel capacity, increasing the time mandated for a firmware roll-out. These results show us the importance of choosing the appropriate configuration based on the available setup. By doing so, we can maximise the throughput without compromising the system stability. Focusing on the two sides of the network now (dense-sparse), it was observed that on average ~6 nodes on the sparse side, and ~2 on the dense one, were not able to finish the binary reception and dropped from the network during the roll-out. This was due to the operation of Atomic and the number of slots chosen. After further investigation, these nodes were identified to have very low RSSI, being at the edge of their neighbours' coverage area. Such nodes are susceptible to the way Atomic works, especially when the number of MAX_SLOTS increases, as they cannot reliably finalise the Atomic initialisation after every Atomic period, and thus are prune to misconfigurations. The performance in such networks could be improved further by combining SF with traditional routing like in [21].

V. Conclusions

This paper described a large-scale IoT wireless investigation using two protocols, i.e., Atomic and CSMA-CA. We adopted a typical firmware roll-out as our experimental use-case using different configurations. We used the publicly available UMBRELLA testbed for our large-scale investigation. Our results showed that Atomic and SF-like protocols outperform traditional protocols like CSMA-CA in such use-cases and larger networks. We also discovered how denser or sparser

setups affect SF and Atomic and the importance of proper configuration based on the given network setup. In the future, we plan to investigate the feasibility of other protocols and extend Atomic's capabilities to accommodate considerable distance variations between neighbouring nodes.

References

[1] D. Singh, G. Tripathi, and A. J. Jara, "A Survey of Internet-of-Things: Future Vision, Architecture, Challenges and Services," in *Proc. of IEEE WF-IoT 2014*, Mar. 2014, pp. 287–292.

[2] P. P. Gaikwad, J. P. Gabhane, and S. S. Golait, "A Survey based on Smart Homes System using Internet-of-Things," in *Proc. of Int. Conf. on ICCPEIC*, Apr. 2015, pp. 0330–0335.

[3] H. Arasteh, V. Hosseinnezhad, V. Loia *et al.*, "IoT-based Smart Cities: A survey," in *Proc. of IEEE EEEIC 2016*, Jun. 2016, pp. 1–6.

[4] J. L. Hernández-Ramos, G. Baldini, S. N. Matheu *et al.*, "Updating IoT Devices: Challenges and Potential Approaches," in *Proc. of GIoTS 2020*, 2020, pp. 1–5.

[5] J. C. Cano, V. Berrios, B. Garcia *et al.*, "Evolution of IoT: An Industry Perspective," *IEEE Internet of Things Magazine*, vol. 1, no. 2, pp. 12–17, May 2019.

[6] L. Oliveira, J. J. P. C. Rodrigues, S. A. Kozlov *et al.*, "MAC Layer Protocols for Internet of Things: A Survey," *Future Internet*, vol. 11, no. 1, Jan. 2019.

[7] M. Baddeley, U. Raza, M. Sooriyabandara *et al.*, "Atomic-SDN: Is Synchronous Flooding the Solution to Software-Defined Networking in IoT?" *IEEE Access*, vol. 7, May 2019.

[8] P. Kolodzy, "Communications Policy and Spectrum Management," in *Cognitive Radio Technology (Second Edition)*, B. A. Fette, Ed. Oxford: Academic Press, 2009, pp. 27–64.

[9] Nordic Semiconductors, "nRF52840 Product Specification, v1.1. [Online]," https://infocenter.nordicsemi.com/pdf/nRF52840PSv1.1.pdf, Accessed: 2021-12-15.

[10] M. Spörk, C. A. Boano, and K. Römer, "Performance and Trade-Offs of the New PHY Modes of BLE 5," in *Proc. of PERSIST-IoT 2019*. New York, NY, USA: Association for Computing Machinery, 2019, p. 7–12.

[11] M. Baddeley, C. A. Boano, A. Escobar-Molero *et al.*, "The Impact of the Physical Layer on the Performance of Concurrent Transmissions," in *Proc. of IEEE ICNP 2020*, Oct. 2020.

[12] A. Tanenbaum and D. Wetherall, *Computer Networks*, 5th ed. Pearson, 2013.

[13] O. Iova, P. Picco, T. Istomin *et al.*, "RPL: The Routing Standard for the Internet of Things... Or Is It?" *IEEE Communications Magazine*, vol. 54, no. 12, pp. 16–22, 2016.

[14] M. O. Farooq and T. Kunz, "Contiki-based IEEE 802.15.4 Node's Throughput and Wireless Channel Utilization Analysis," in *Proc. of IFIP Wireless Days 2012*, Nov. 2012, pp. 1–3.

[15] T. Farnham, S. Jones, A. Aijaz *et al.*, "UMBRELLA Collaborative Robotics Testbed and IoT Platform," in *Proc. of IEEE CCNC 2021*, Jan. 2021, pp. 1–7.

[16] Texas Instruments, "CC1310 SimpleLink Ultra-Low-Power Sub-1 GHz Wireless MCU datasheet (Rev. D) [Online]," https://www.ti.com/lit/gpn/cc1310, Accessed: 2021-12-15.

[17] Skyworks Inc., "RF Front-end Modules Boost Wireless Performance," Accessed: 2021-12-15. [Online]. Available: https://www.skyworksinc.com/-/media/SkyWorks/Documents/Articles/RF_Bluetooth_FEMs.pdf

[18] RaspberryPi, "Compute Module 3b+," https://www.raspberrypi.org/products/compute-module-3-plus/, 2021, Accessed: 2021-12-15.

[19] "Contiki-NG: The OS for Next Generation IoT Devices," https://github.com/contiki-ng/contiki-ng, Accessed: 2021-12-15.

[20] I. Ishaq, J. Hoebeke, I. Moerman *et al.*, "Experimental Evaluation of Unicast and Multicast CoAP Group Communication," *MDPI Sensors*, vol. 16, no. 7, Jul. 2016.

[21] M. Baddeley, A. Aijaz, U. Raza *et al.*, "6TiSCH++ with Bluetooth 5 and Concurrent Transmissions," in *Proc. of EWSN 2021*, Feb. 2021, p. 25–30.