Experimental comparison of Industrial Internet of Things protocol stacks in Time Slotted Channel Hopping scenarios

Pietro Boccadoro[(1,2)], Giuseppe Piro[(1,2)], Domenico Striccoli[(1)] and Luigi Alfredo Grieco[(1,2)]
[(1)]Dep. of Electrical and Information Engineering (DEI), Politecnico di Bari, Bari (Italy)
[(2)]Consorzio Nazionale Interuniversitario per le Telecomunicazioni, CNIT
e-mail: {pietro.boccadoro, giuseppe.piro, domenico.striccoli, alfredo.grieco}@poliba.it

***Abstract*—Time Slotted Channel Hopping (TSCH) is a recent enhancement of the IEEE 802.15.4 standard aimed at the industrial sector. To complement its unique features, the IPv6 over the TSCH mode of IEEE 802.15.4e (6tisch) Working Group released a set of specifications that define the way Time Slotted Channel Hopping could be integrated in embedded IPv6 networks based on the IPv6 over Low Power WPAN. TheI 6tisch standard is now available under several hardware/software platforms so that a cross-comparison of the different implementations deserves particular attention in order to properly identify performance bottlenecks and scavenge pros and cons of the 6tisch technology. The present contribution provides an experimental cross-comparison of protocol stack implementations running on top of different devices. In particular, experimental tests were conducted to characterize the behavior of OpenWSN and Contiki in Time Slotted Channel Hopping-based scenarios made up by TelosB, Zolertia Z1, and OpenMote motes. By considering star and chain topologies and different traffic load conditions, many performance indices were measured, including memory footprint, energy consumption, synchronization time, duty-cycle, end-to-end packet delays, and Packet Loss Ratio. Obtained results highlight strengths and weaknesses of each hardware/software platform, while providing trade-off considerations about flexibility, accuracy, energy-efficiency, and Quality of Service.**

***Index Terms*—Industrial Internet of Things; TSCH; 802.15.4; Contiki; OpenWSN.**

I. Introduction

Industrial Internet of Things (IIoT) is now emerging as a new communication paradigm for a class of low-power wireless networks used in critical applications, such as industrial process monitoring and automation [1]. Time Slotted Channel Hopping (TSCH) is widely considered a cornerstone technology for the IIoT [2]. On top of TSCH, Internet Engineering Task Force (IETF) defined a lightweight protocol stack for constrained IIoT devices, that grants communication requirements (i.e., reliability, security, latency, bandwidth) of typical industrial applications [3], as experimented in [4]. At the time of this writing, two main Open Source projects are emerging in both academia and industry, i.e., OpenWSN [5] and Contiki [6]. Both of them are already supported by many hardware platforms widely diffused in the market and extensively employed in research activities (like TelosB, Zolertia Z1, and OpenMote [1][7]).

From the hardware perspective, it is acceptable to assume that different platforms may experience variable performance levels, due to their peculiar processing and storage capabilities. Furthermore, despite the specifications imposed by reference standards, different software solutions may behave differently under test. Therefore, an experimental comparison becomes of great importance. Examples of studies in this direction can be found in [1], [8] and [9]. The first highlights the set of features and the list of hardware platforms supported by available implementations of the IETF protocol stack. It also presents preliminary performance indices captured by isolated experimental tests. The second investigates network performance metrics in terms of routing. The last one evaluates energy consumption of TelosB platforms operating in a TSCH-enabled scenario when running OpenWSN and Contiki operating systems.

Unfortunately, the literature does not offer, yet, a complete cross-comparison of the aforementioned IIoT technologies. This work jointly characterizes operating systems and hardware platforms in realistic operative conditions. To this end, different TSCH-enabled scenarios, with two network topologies and different traffic loads, were investigated. The experimental campaign was conducted to measure memory footprint, energy consumption, synchronization time, duty-cycle, End-to-End packet delays, and Packet Loss Ratio (PLR). Obtained results illustrate the impact of hardware on network performance. OpenMote offers the best memory and computational capabilities, and registers lower values for almost all the aforementioned parameters. Counterwise, TelosB and Zolertia Z1 are less energy demanding. The comparison between operating systems highlights that OpenWSN provides a better usage of hardware components, thus providing promising values for almost all of the chosen Quality of Service (QoS) indices.

The rest of the work is organized as what follows: Section II presents the technological background on IIoT, including IETF protocol stack, operating systems, and hardware platforms. Section III describes the experimental campaign and discusses obtained results. Section IV concludes the work and proposes future works.

II. Technological background

A. The protocol stack

The IIoT protocol suite is built on top of MAC and PHY layers defined by the IEEE 802.15.4 standard [3] and integrates solutions standardized by different IETF Working Group (WG)

1) Application layer: The Constrained RESTful Environments (CORE) WG developed Constrained Application Protocol (CoAP) [10], a lightweight implementation of the well-known HTTP protocol, suitable for constrained devices. It allows mapping of CoAP messages to HTTP messages, thus easing the integration of IIoT services in the web. Unlike HTTP, CoAP is based on request/response communication scheme and adopts User Datagram Protocol (UDP) at the transport layer. At the same time, it supports multicast and introduces low overhead.

2) Network layer: The IETF protocol stack uses Internet Protocol version 6 (IPv6). IIoT systems generally appear as multi-hop Low-power Lossy Networks (LLNs). To facilitate topology formation and management, the Routing Over Low power and Lossy networks (ROLL) WG designed a gradient based routing protocol, namely Routing Protocol for Low-power and Lossy networks (RPL) [11] that leverages multiple roots and adapts topology through parametric optimization functions.

3) Adaptation layers: With IPv6, the default minimum Maximum Transmission Unit (MTU) size is set to 1280 bytes. IEEE 802.15.4 technology allows a maximum frame length of only 127 bytes. Therefore IPv6 packet could be too large to fit in an IEEE802.15.4 frame. To solve this problem, the IPv6 over Low power WPAN (6lowpan) WG concluded the standardization of the 6LoWPAN protocol, that fits IPv6 packets in small payload size (up to 127 bytes) through header compression techniques.

4) IEEE 802.15.4 MAC: Time Slotted Channel Hopping represents a cornerstone technology for the IIoT; it works at the Medium Access Control (MAC) layer and offers a good reliability against interference and multi-path fading. Transmit, listen or sleep tasks are scheduled in time and frequency slots, which can be assigned to single network node or shared among many devices.

5) IEEE 802.15.4 - PHY: IEEE 802.15.4 leverages a low-power physical layer based on the Direct Sequence Spread Spectrum modulation scheme and operates at 2.4 - 2.485 GHz ISM frequency band with Offset-Quadrature Phase-Shift Keying 2 Mbps physical data rate modulation scheme. The resulting physical data rate is equal to 250 kbps. It is configured to reach a trade-off between energy-efficiency, communication range, and data rate.

B. Implementations

At the time of this writing, there exist two widely accepted implementations for the IIoT protocol stack discussed [7], OpenWSN [5] and Contiki [6].

1) OpenWSN: It is an Open Source and freeware project [5] implementing 802.15.4 specifications. It runs on top of two different kernel, i.e., openos and FreeRTOS. Among them, openos is widely used because of the lower hardware requirements[1]. OpenWSN embraces firmware and software parts. The former implements instructions and procedures of the protocol stack and the code to manage drivers on devices. The latter, namely OpenVisualizer, provides gateway functionalities to connect the IIoT network to Internet. It also allows network monitoring and debugging for motes physically connected to the host controller. OpenWSN relies on a standard-compliant Finite-State Machine (FSM) that orchestrates all the operations performed by means of a set of timer. The FSM directly controls radio transceiver activities and radio duty-cycling mechanism[2].

2) Contiki: It is a highly portable operating system [6], with a kernel based on event-driven and multi-threaded programming principle (namely Protothreads). Protothreads [12] provide an alternative way to implement a list of operations (namely *flow of control*) with the goal of reducing memory usage, which is useful in constrained systems. Bringing to a limited memory usage, this approach is useful in constrained systems. In Contiki, TSCH is implemented through a set of processes. Each process may be composed by one or more protothreads, that control the list of operations to execute during transmission and reception activities. Contiki's TSCH implementation is still undergoing [13].

C. Hardware platforms

The work presented herein considers three different platforms that are widely diffused in the market and extensively employed in research activities. They are TelosB[3] [2][4], Zolertia Z1[4][14][15] and OpenMote[5][16][17]. The main characteristics for each of them are reported in Table I.

Board	MCU	Memory (RAM/Flash)	Radio Chip
TelosB	MSP430	10 kB / 48 kB	CC2420
Zolertia Z1	MSP430 (2nd gen.)	8 kB / 92 kB	CC2420
OpenMote	ARM Cortex M3	32 kB / 512 kB	CC2538

TABLE I
HARDWARE SPECIFICATIONS

III. Experimental comparison

The behavior of OpenWSN and Contiki implementations, running on different hardware platforms and in different TSCH-enabled scenarios, were deeply characterized by means of experimental tests. The IPv6 over the TSCH mode of IEEE

[1]Note that openos is the operating system taken into account in this work.

[2]The passing of time is handled as a sequence of ticks, that are unitary time intervals measured by the on-board oscillator (working @32.768 kHz). Typically, ticks manage interrupts. Then, interrupts control Micro-Controller Unit (MCU) tasks and the switching among different states of the FSM.

[3]http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.eps

[4]http://zolertia.sourceforge.net/wiki/index.php/Main_Page

[5]http://www.ti.com/product/cc2538

802.15.4e (6tisch) WG proposes a baseline configuration for the TSCH schedule, namely 6tisch-minimal [18], that assumes 11 timeslots (10 ms each) per slotframe, 1 timeslot shared among all the devices, as well as 3 possible retransmissions of unacknowledged packets. To provide a deep insight in a realistic scenario, the experimental campaign considers a network composed by 8 nodes (i.e., 1 network coordinator and 7 end nodes). Nodes were arranged according to both star and chain topologies. As for TSCH configuration, a slotframe made up by 101 timeslots (15 ms each) has been implemented. The first timeslot is used for transmitting beacon messages. Then, 7 consecutive timeslots are allocated to each node pair. During the remaining timeslots, instead, all nodes are forced to stay in sleep mode. In each test, all the nodes (excepting for the network coordinator) have been configured in order to send 800 packets with a payload of 40 bytes each to the network coordinator. Furthermore, to evaluate the impact of the network load, two values of the inter-arrival packet time, ΔT, have been considered: 1 s and 10 s.

The following key performance indices have then been evaluated: memory footprint, energy consumption, synchronization time, duty-cycle, end-to-end packet delays, and PLR. Average values and the related 95% confidence intervals have been calculated and reported in the figures below.

A. Memory footprint

The firmware for each board is compiled using the appropriate toolchain (i.e., MSP-gcc and ARM-gcc). The resulting memory footprint strongly depends on these premises and it is of great importance to discuss the effective compatibility between protocol stack implementations and constrained platforms. Given the limited storage capabilities of IIoT devices, firmware must be optimized to limit the consumptions of both ROM and RAM memories. Measured ROM and RAM footprints are summarized in Table II. Both the absolute value and the percentage of memory occupied by the operating system over the total are reported. Obtained results show that a given protocol stack implementation produces a memory occupancy that depends on the hardware platform. On TelosB, OpenWSN and Contiki register similar ROM occupancy values, taking almost all the available space (83%). On Zolertia Z1, the ROM footprint gets closed to the half of available memory (specifically, OpenWSN takes 41% and Contiki takes 54%). OpenMote experiments the lowest ROM footprint, that is always less than 15%. As a result, OpenMote emerges as the most promising hardware platform, allowing future implementation of additional and/or updated algorithms and protocols. This possibility is almost prevented on the TelosB mote. Similar considerations can be done as for RAM footprint. Freeing RAM occupancy expresses the possibility to load additional variables against those used by baseline implementations. In this context, experiments demonstrated that Zolertia Z1 offers very limited capability to load additional content. Similar considerations can be done for the TelosB platform, when running the Contiki operating systems. With OpenWSN, instead, about 33% of RAM can be used for loading new variables and instructions. Also in this case, the OpenMote platform emerges as the most versatile platform because it always guarantees the lowest percentage of the RAM footprint (equal to 53% and 29% respectively in Contiki and OpenWSN). Lastly, RAM and ROM occupancies are not significantly affected by the network topology and the role of the device.

		OpenWSN		**Contiki**	
		RAM (B)	**ROM (B)**	**RAM (B)**	**ROM (B)**
TelosB	*Chain Coordinator*	6662 (66.7%)	39882 (83%)	8102 (81%)	39883 (83%)
	Chain End Node			7996 (79.9%)	
	Star Coordinator			8102 (81%)	
	Star End Node			7996 (79.9%)	
Zolertia Z1	*Chain Coordinator*	6560 (82%)	37676 (41%)	7523 (94%)	49954 (54%)
	Chain End Node			7890 (98%)	49628 (54%)
	Star Coordinator		37530 (41%)	7523 (94%)	49954 (54%)
	Star End Node			7890 (98%)	49628 (54%)
OpenMote	*Chain Coordinator*	9284 (29%)	71092 (14%)	17112 (53%)	52278 (10%)
	Chain End Node			17239 (53%)	52242 (10%)
	Star Coordinator			17112 (53%)	52278 (10%)
	Star End Node			17239 (53%)	52242 (10%)

TABLE II
MEMORY FOOTPRINTS

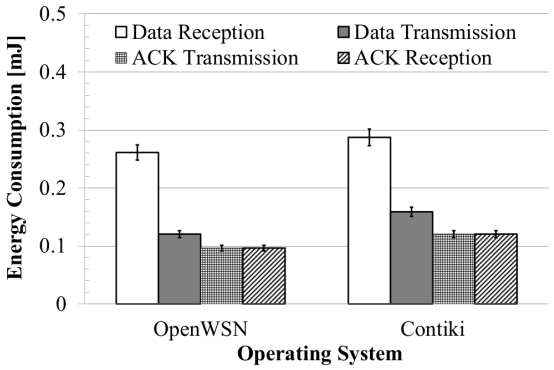
B. Energy consumption

During communication processes, both radio interfaces and MCUs drain current. Frequent transmissions and receptions activities can discharge batteries in a matter of days. While a datasheet can be used for preliminary evaluation on nominal values, conducted tests provide real energy consumptions. This enables realistic measurements of the amount of energy consumed during each operation within each time slot (i.e., data transmission, data reception, ACK transmission, and ACK reception). Overall values are detailed in Table III In OpenWSN, data transmission and reception processes take 2.5 ms and 5 ms, respectively. The transmission and the reception of ACKs take 2 ms. In Contiki, the same processes take: 3.3 ms for data transmission, 5.5 ms for data reception, and 2.5 ms for transmitting and receiving the ACK. In Figure 1 energy consumption values are reported[6]. The most noticeable results is that measured values are not in line with reference datasheets[7]. On TelosB and OpenMote platforms, higher discrepancies are measured with Contiki. At the same time, however, the measured energy consumptions of OpenMote are

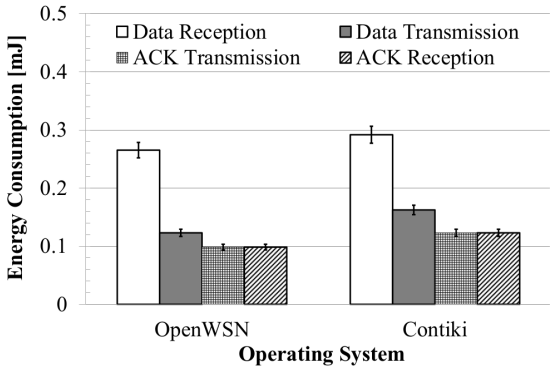
[6]The real energy consumption (expressed in Joule) are calculated as the product of the batteries voltage values, the current measured and the specific reference time interval.

[7]Zolertia Z1's nominal MCU current consumption value for active mode is $<$10 mA, so estimations are not as precise as for other platform solutions.

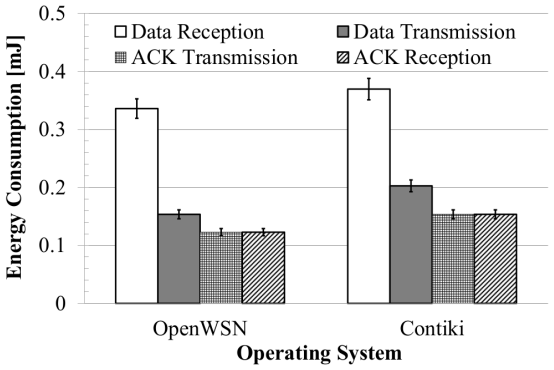
far from those reported in reference datasheets (sometimes up to 50%).



(a)



(b)



(c)

Fig. 1. Energy Consumption in (a) TelosB, (b) Zolertia Z1 and (c) OpenMote

C. Synchronization time and duty-cycle

Devices synchronization is mandatory to maintain connection between neighbors in a TSCH-based network. The entire network is configured by means of advertising and joining mechanisms. At the beginning, the network coordinator sends (periodically and in broadcast) the advertisement command frame. A new node willing to join the network, processes that command and sends in unicast the join request command frame. The coordinator confirms the success of the joining procedure by transmitting the activate command

		Data Tx	Data Rx	ACK Tx	ACK Rx
OpenWSN	**TelosB**	4.5	4.6	4.4	4.4
	Zolertia Z1	0	0	0	0
	OpenMote	6.7	40	6.7	6.7
Contiki	**TelosB**	38	16	30.6	30.6
	Zolertia Z1	0	0	0	0
	OpenMote	6.7	50	6.7	6.7

TABLE III
ENERGY CONSUMPTION OVERHEAD WITH REFERENCE TO NOMINAL VALUES, REPORTED IN %.

frame. Then, this procedure is replicated hop-by-hop until the farthest child node is reached. For each pair of nodes, IEEE 802.15.4 adopts an *ACK-based synchronization approach*. To this end, the receiver calculates the difference between the expected and effective time of arrival of the frame Then, it provides that information to the sender node via an ACK, and the sender synchronizes with receiver's clock. Once the network is established at the MAC layer, the RPL protocol configures the topology at the network layer as well. In this work, synchronization time is defined as the time taken by all the devices in the network to successfully complete the MAC joining procedure and the RPL configuration of both downward and upward communication paths. Results reported in Table IV remark that synchronization times are up to 71 s in the worst case. Nevertheless, OpenMote achieves the best performance by registering the lowest synchronization time, in all the considered conditions. Once the network is configured, nodes start exchanging data. According to the TSCH schedule, the communication occurs in specific time slots.

It is of relevance to verify effective duty-cycle values (e.g., the time percentage of the radio transceiver activity over the total slotframe definition) Results are shown in Figure 2. Values show that OpenWSN always provides the lowest duty-cycle: differently from Contiki, a reduction up to 64% in the star topology and up to 69.15% in the chain topology has been observed. This behavior can be justified by considering the lower amount of time (see Section III-B) taken for performing Tx and Rx operations. At the same time, the higher the interarrival packet time, the lower the duty-cycle. As expected, when the interarrival packet time increases, the quota of packets to be disseminated within the network decreases. Thus, the radio activity (and so the duty-cycle) decreases as well. With respect to the network, chain topology registers higher duty-cycles. Unlike the simple star topology, in a chain the intermediate nodes are in charge of disseminating application data coming also from other nodes. Therefore, their radio activity (and so the duty-cycle) experiences an increment. It is important to highlight that the TSCH schedule does not represent the only factor that influences the radio activity. Indeed, the traffic load (and therefore the presence of data to transmit) can introduce a significant overhead.

D. End-to-end packet delays and PLR

End-to-end packet delays are depicted in Figure 3. As expected, network latencies increase together with the traffic

		Star	Chain
OpenWSN	**TelosB**	47	56
	Zolertia Z1	59	52
	OpenMote	46	46
Contiki	**TelosB**	66	71
	Zolertia Z1	58	61
	OpenMote	42	49

TABLE IV
SYNCHRONIZATION TIME (EXPRESSED IN S).

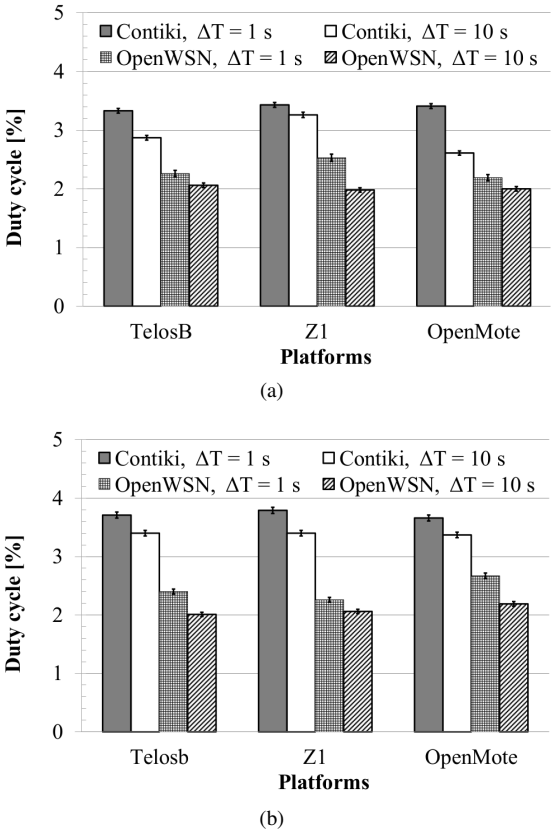


Fig. 2. Duty Cycle in (a) star topology and (b) chain topology

load. Surely, the lower the interarrival packet time, the higher the amount of packets to be disseminated. Given the fixed (and limited) number of time slots dedicated to link-level transmissions (that is equal to 1 time slot for per slot frame for each node pair), higher packet delays are measured when $\Delta T = 1s$. Furthermore, with chain topology the additional latency introduced by the multi-hop communication paths still create effects. In the star topology, packet delays are in the range between 0.8 s (TelosB with Contiki) and 2.9 s (Z1 with OpenWSN). In the chain topology, instead, they fall in the range between 4 s (OpenMote with Contiki) and 8.7 s (TelosB with Contiki).

By comparing different hardware platforms, it clearly emerges that the higher the profile of the device, the better the performances. In line with the previous comments, PLR is higher in scenarios with the highest traffic load and in chain network topology. Similar considerations can be done here analyzing performances reached by the hardware platforms.

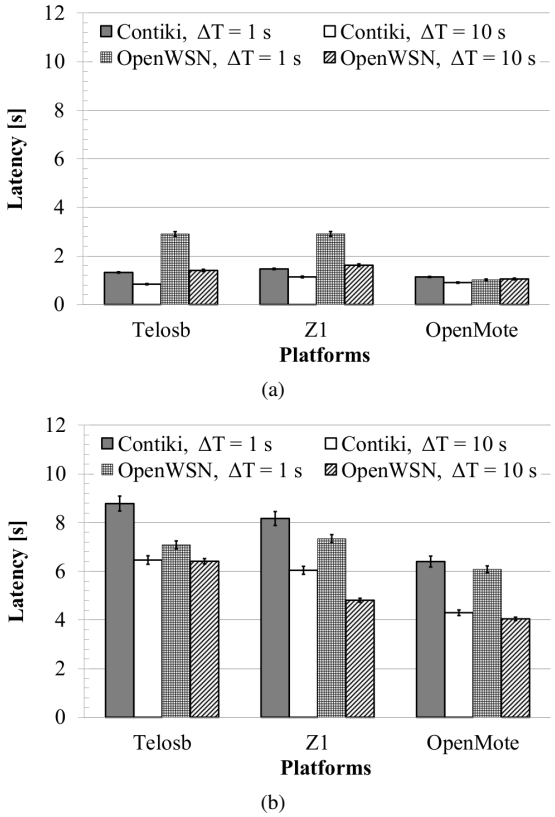


Fig. 3. Latencies in (a) star topology and (b) chain topology

It is also noticeable that OpenWSN registers, in this case, the worst behavior when compared with Contiki. Typically, IIoT networks are also classified as LLNs. Therefore, IIoT networks may experience important PLR values [4]. This may depend on the number of devices, their location and the overall QoS. Moreover, packets might be lost due to different phenomena, i.e., queue filling, channel interferences, multi-path fading, etc. In such conditions, lower interarrival packet time may lead to queue overflow. The probability of queue overflow is higher if the mote is more hops away. In general, increasing the interarrival packet time can help reaching longer distances. Counterwise, the increase in interarrival packet time can significantly lower the throughput.

To provide a further insight, both PLR and latencies for lower-end devices could be explained with portability issues related to the TSCH code. For example, drift correction may be the cause of unreliable frame identification thus leading to an increase in latencies, when packets are received, or packet losses, in the worst cases. The latter implies the consequent retransmission requests. The outcome is clearer when observing OpenMote behavior: values are considerably better and networks look more efficient with respect to TelosB and Zolertia Z1-based deployments.

IV. CONCLUSIONS AND FUTURE WORKS

The present work proposed an experimental comparison of IIoT protocol stacks in TSCH scenarios. In particular, two

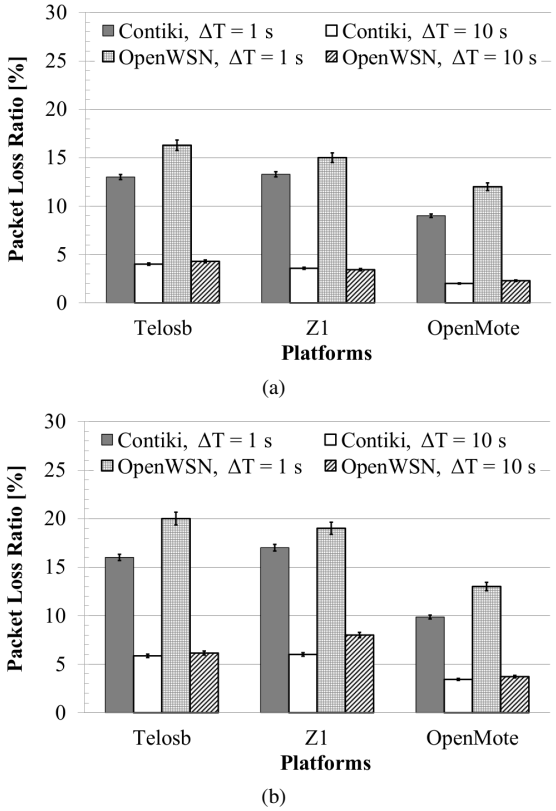


Fig. 4. PLR in (a) star topology and (b) chain topology

main implementations were evaluated in different network deployments made up by heterogeneous hardware platforms. The study highlighted pros and cons of available software and hardware solutions in terms of: memory footprint, energy consumption, synchronization time and duty-cycle, end-to-end packet delays, and Packet Loss Ratio. Obtained results demonstrated that: (i) adopted hardware and software solutions impact on memory footprints, (ii) computational capabilities impact on network performance; (iii) the OpenMote platforms guarantees lower memory footprint, lower synchronization time and duty-cycles, as well as lower end-to-end packet delays and PLR, at the expense of a higher energy consumptions (with respect to both TelosB and Zolertia Z1 motes); (iv) OpenWSN provides a better usage of hardware components, thus resulting in lower values in terms of memory footprint, energy consumption, synchronization time, duty-cycle, end-to-end packet delays; and (v) OpenWSN registers higher Packet Loss Ratio values than Contiki. Future work will improve the experimental analysis for large scale IIoT deployments by: considering new communication protocols, verifying Orchestra's efficiency, including RiOT OS, and investigating specific optimization in network synchronization mechanism.

V. Acknowledgment

This work was partially supported by the BONVOYAGE project, which received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 63586. It has also been partially funded by the research project E-SHELF (by the Apulia Region - Italy, code: OSW3NO1) and the FFABR fund (by the Italian MIUR).

References

[1] T. Watteyne, V. Handziski, X. Vilajosana, S. Duquennoy, O. Hahm, E. Baccelli, and A. Wolisz, "Industrial Wireless IP-Based Cyber-Physical Systems," *Proc. of the IEEE*, vol. 104, no. 5, pp. 1025–1038, May 2016.

[2] B. D. Darshini, A. Paventhan, H. Krishna, and N. Pahuja, "Enabling real time requirements in industrial IoT through IETF 6TiSCH," in *Proc. of International Conference on Internet of Things and Applications (IOTA)*, Jan. 2016, pp. 121–124.

[3] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel, "On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks," *IEEE Sensors Journal*, vol. 16, no. 2, pp. 550–560, Jan. 2016.

[4] A. Paventhan, D. D. B, H. Krishna, N. Pahuja, M. F. Khan, and A. Jain, "Experimental evaluation of IETF 6TiSCH in the context of Smart Grid," in *Proc. of IEEE World Forum on Internet of Things (WF-IoT)*, Dec. 2015, pp. 530–535.

[5] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. Glaser, K. Pister, "Openwsn: A standards-based low-power wireless development environment," *Wiley Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

[6] Adam Dunkels and Bjorn and Gronvall and Thiemo Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Proc. of IEEE International Conference on Local Computer Networks*, Nov. 2004.

[7] T. B. Chandra, P. Verma, and A. Dwivedi, "Operating systems for internet of things: A comparative study," 2016. [Online]. Available: https://arxiv.org/pdf/1504.02517

[8] T. Farnham, "Proactive wireless sensor network for industrial IoT," in *Proc. of IEEE International Conference on Communications (ICC)*, May. 2017, pp. 1–6.

[9] P. Boccadoro, M. Barile, G. Piro, and L. A. Grieco, "Energy consumption analysis of TSCH-enabled platforms for the Industrial-IoT," in *Proc. of IEEE International Forum on Research and Technologies for Society and Industry (RTSI)*, Sep. 2016.

[10] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, *Constrained Application Protocol (CoAP)*, IETF CoRE Working Group, Feb. 2011.

[11] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," *Internet Engineering Task Force (IETF) - Request for Comments: 6550*, 2012.

[12] A. Dunkels, O. Schmidt, and T. Voigt, "Using protothreads for sensor node programming," in *Proc. of the REALWSN*, vol. 5, 2005.

[13] S. Duquennoy, A. Elsts, B. Al Nahas, and G. Oikonomou, "TSCH and 6TiSCH for Contiki: Challenges, Design and Evaluation," in *Proc. of International Conference on Distributed Computing in Sensor Systems (DCOSS)*, Jun. 2017.

[14] S. Kharche and S. Pawar, "Node level energy consumption analysis in 6LoWPAN network using real and emulated Zolertia Z1 motes," in *Proc. fo IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Nov. 2016, pp. 1–5.

[15] A. N. Mian, S. A. Alvi, R. Khan, M. Zulqarnain, and W. Iqbal, "Experimental study of link quality in IEEE 802.15.4 using Z1 Motes," in *Proc. of International Wireless Communications and Mobile Computing Conference (IWCMC)*, Sep. 2016, pp. 830–835.

[16] T. Chang, P. Tuset-Peiro, X. Vilajosana, and T. Watteyne, "OpenWSN&OpenMote: Demo'ing a Complete Ecosystem for the Industrial Internet of Things," in *Proc. of IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Jun. 2016, pp. 1–3.

[17] D. Vasiljevi and G. Gardaevi, "Performance evaluation of OpenWSN operating system on open mote platform for industrial IoT applications," in *Proc. of International Symposium on Industrial Electronics (INDEL)*, Nov. 2016, pp. 1–6.

[18] X. Vilajosana and K. Pister, "Minimal 6tisch configuration draft-ietf-6tisch-minimal-16," 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6tisch-minimal-16