On the Performance of Data Distribution Methods for Wireless Industrial Networks

Theofanis P. Raptis*, Andrea Formica†, Elena Pagani†, Andrea Passarella*
*Institute of Informatics and Telematics, National Research Council, Via G. Moruzzi 1, 56124 Pisa, Italy
†Department of Computer Science, University of Milan, Via Celoria 18, 20135 Milano, Italy
{theofanis.raptis, andrea.passarella}@iit.cnr.it, {andrea.formica@studenti, pagani@di}.unimi.it

***Abstract*—The vast amounts of data generated in wireless industrial networked deployments introduce significant challenges on the data distribution process to consumer nodes within the timeframes imposed by the requirements of the Industry 4.0 paradigm. Using technological and methodological enablers, we can compose centralized or decentralized data distribution methods, which are able to help meeting the data requirements of the industrial applications. In this paper, using the technological enablers of WirelessHART, RPL and the methodological enabler of proxy selection as building blocks, we compose the protocol stacks of four different methods (both centralized and decentralized) for data distribution in wireless industrial networks over the IEEE 802.15.4 physical layer. Although there have been several comparisons of relevant methods in the recent literature, we identify that most of those comparisons are either theoretical, or based on abstract simulation tools, unable to uncover the specific, detailed impacts of the methods to the underlying networking infrastructure. We implement the presented methods in OMNeT++ and we evaluate their performance via a detailed simulation analysis. Interestingly enough, we demonstrate that the careful selection of a limited set of proxies for data caching in the network can lead to increased data delivery success rate and low data access latency.**

***Index Terms*—WirelessHART, RPL, Proxy Selection, Data Caching, OMNeT++, Industry 4.0**

I. Introduction

The proliferation of data generated from various networked industrial devices have led to data sizes which can range from terabyte to petabyte even for a single industrial deployment. In core industrial applications, the relevant devices generate about 1000 exabytes in total volume annually, and a great increase can be expected in the next ten years [1]. Those data amounts introduce great challenges not only to capture, manage, and store large data sets but also to efficiently deliver them to the data consumer nodes within the timeframes imposed by the requirements of the emerging paradigm of Industry 4.0.

Traditionally, data generated in wireless industrial networks are delivered to the data consumers via a central network controller. This approach, although robust, has limited flexibility, and makes the process of data distribution in the network more difficult and time consuming [2]. An alternative to this entirely centralized approach, is the use of intermediate, local cache nodes, which can provide distributed data replication, and lead to high flexibility, reconfigurability, and maintenance [3]. In order to achieve this decentralized mode of operation, the industrial network and the data management strategy must be built upon a flexible combination of the available communication and data management technologies, capable of meeting the data requirements of the industrial applications, with particular attention on time-critical automation.

Relevant technologies and methods have already been designed and implemented, and some of them are already used in networked industrial deployments. Some notable examples (able to operate on top of IEEE 802.15.4 physical layer) include WirelessHART [4], which provides backward compatibility with older industrial standards, RPL [5], which was designed with the objective to meet the requirements of industrial routing with low-power and lossy networks, as well as proxy utilization methods within the existing networking infrastructure [6], with the aim of meeting the data access latency constraints by locally caching industrial data. A key objective in recent literature is the performance comparison of methods employing such technological and methodological enablers, in order to fully understand the benefits and shortcomings of each.

Related works. In the recent literature, there have been several comparisons (both theoretical and simulative) among enablers that we consider. On the theoretical side, a comparison of WirelessHART and ZigBee is provided in [7], a comparison of WirelessHART and ISA100.11a, both from a technical and a systematical point of view is provided in [8], and a theoretical study on the relationship between WirelessHART and IEEE 802.15.4e, as well as on what impact the latest IEEE 802.15.4e has on WirelessHART and how industrial operators could benefit from both standards is provided in [9]. On the simulative side, RPL comparisons to other routing protocols like LRP and LOADng in different application areas are provided in [10], [11], and a latency aware proxy selection method comparison to RPL is provided in [12]. To the best of our knowledge, there is no direct performance comparison of methods incorporating all three aforementioned elements (WirelessHART, RPL, proxies), and in most cases, opposite to our approach, the comparisons are either theoretical ([7]–[9]), or they employ high-level simulation tools, thus not being able to uncover the underlying network specifics ([12] - Matlab).

Our Contribution. In this paper, contrary to the recent

This work has been funded by the European Commission through the FoF-RIA Project AUTOWARE: Wireless Autonomous, Reliable and Resilient Production Operation Architecture for Cognitive Manufacturing (No. 723909).

theoretical and abstract simulative comparisons, we provide the following contributions:

- We analyze the characteristics of four protocol stacks for data distribution in wireless industrial networks, built mainly from proposed standards such as IEEE 802.15.4, IETF RPL and IEC WirelessHART, and adopting either a centralized (data accessed only at the network controller) or distributed (data replicated in *proxies*) approach.
- We implement the methods in OMNeT++ Network Simulator, by extending and customizing the simulator according to the needs of our study.
- We evaluate the performance of the four methods via a detailed simulation analysis. The use of OMNeT++ reveals detailed performance insights about industrial data distribution. We show that a limited number of data *proxies* – carefully placed in order to limit delay in data retrieval – effectively helps in fulfilling latency constraints.

II. Methods for Industrial Data Distribution

In this section, we design four methods targeting data distribution for wireless industrial networks. We first present the network model we consider, then the technological and methodological building blocks that we use and finally the centralized and decentralized methods that we design. The layout of the methods is presented in Table I.

We consider networks of industrial devices which consist of sensor motes, actuators and controllers and are operating at the physical layer in the 2.4 GHz ISM band using IEEE 802.15.4 radios. In network configurations of such type, a central network controller maintains centralized network knowledge. This is usual in industrial applications, in which the locations of the nodes are known, traffic flows are deterministic and communication patterns are established a priori. We assume that the controller knows all the shortest paths in the network.

Sensor nodes perform monitoring tasks (producers), and in some cases, their sensor data are needed either by other sensor nodes, which could need additional data to complement their own local measurement, or by actuator nodes, which use the sensor data so as to perform an actuation (consumers). When needed, a consumer can ask for data of interest using the primitives defined by the underlying routing protocol from a sensor node. The control of the data distribution process is centralized; however the data distribution itself can be decentralized and cooperative, by caching the data in identified nodes in the network, the proxy nodes. Proxy nodes are able to store data that can be accessed in a timely manner from the consumer nodes of the network.

We assume that the data generation and access processes are not synchronized. Specifically, we assume that data consumers request data at an unspecified point in time after data have been generated by data producers. This demarcated model of data exchanges can be formulated as a publish-subscribe (pub/sub) model. In a pub/sub model, a consumer subscribes to data, i.e., denotes interest for it to the corresponding proxy, and the relevant producer publishes advertisements to this proxy. The pub/sub process is regulated at the central network controller, which maintains knowledge on the sets of producers, consumers and requests. Based on this, the network controller can find an appropriate set of proxies based on the algorithm we present next. Inside the network, the proxies are responsible for matching subscriptions with publications, i.e., they provide a rendezvous function for storing the available data according to the corresponding subscriptions. The producers do not hold references to the consumers, neither do they know how many consumers are receiving their generated data.

A. Building blocks

We now present the main technological and methodological enablers which we use as building blocks for our methods.

1) WirelessHART (IEC 62591): WirelessHART [4] is a wireless sensor networking technology based on the Highway Addressable Remote Transducer Protocol (HART). Developed as a multi-vendor, interoperable wireless standard, WirelessHART was defined for the requirements of industrial field device networks. The protocol utilizes a time synchronized, self-organizing, and self-healing mesh architecture. An important characteristic of WirelessHART MAC layer which differentiates it from the typical 802.15.4 CSMA/CA MAC layer is that it combines slow frequency-hopping and a TDMA scheme that utilizes a centralized a-priori slot allocation mechanism [13]. Two different mechanisms are provided for message routing: graph routing and source routing. In this work we use the source routing option, which uses ad-hoc created routes for the messages (instead of the pre-determined, redundant routes of graph routing) without providing any path diversity.

2) RPL (IETF RFC 6550): The RPL routing framework [5] is suitable for networks composed of many embedded devices with limited power, memory, and processing resources and is able to meet the requirements of a wide range of monitoring and industrial applications. It is widely considered as an important networking solution for various industrial sectors. There are two modes RPL can choose from; one is called non-storing mode, and the other is called storing mode [14]. In non-storing mode, the routing tree root (in our case the central network controller) collects and maintains topology information of the whole network. In storing mode, a node keeps a complete list of routing entries for nodes in its sub-routing trees. Consequently, there is a trade-off between storing and non-storing mode of operation in terms of computing and communication resources. For example, storing mode requires more memory in intermediate nodes while non-storing mode causes data packets to increase in size which uses more power and bandwidth.

3) Latency-aware proxy selection: In the context of wireless industrial networks, one could leverage the set of nodes present at the edge of the network to distribute functions that are currently being implemented by the central network controller. Many flavors of distributed data management exist in the networking literature, depending on which edge devices are used. In [12], the authors consider a distributed approach

TABLE I
LAYOUT OF THE COMPARED METHODS

	C1	D1	D2	C2
Data cache	Netw. Contr.	LCAs	Proxies	Netw. Contr.
APP	publish-subscribe			
NET	RPL non-st.	RPL st.	source routing	
MAC	CSMA/CA			WirelessHART
PHY	802.15.4 (2.4 GHz)			

and use the multitude of sensor nodes present in an industrial physical environment (e.g., a specific factory) to implement a decentralized data distribution, whereby sensor nodes cache data they produce and provide these data to each other upon request. In this case, the choice of the sensor nodes where data are cached must be done to guarantee a maximum delivery latency to nodes requesting those data. The objective function of the proxy selection minimizes the number of proxies. The main constraint is put on the guarantee that the value of the average end-to-end data access latency by the consumers in the network has to remain below a given time threshold imposed by the industrial operator. Each node has to be assigned to one and only one proxy, and all nodes are considered for potentially being selected as proxies.

B. Centralized Methods for Industrial Data Distribution

The composition of the two centralized methods is shown in Table I labeled as C1 and C2. The common aspect of the centralized methods is that they cache the generated data at the central network controller. Consequently, when a consumer node requires data access, the data request has to be sent to the network controller and the network controller has to make the data available to the consumer. The main differences of the two centralized methods are lying on the MAC and NET layers. Method C1 is using RPL in non-storing mode in order to route the relevant data over the typical CSMA/CA protocol of 802.15.4. That means that the slots in each frame are generally contention-based. On the contrary, method C2 is using source routing over the WirelessHART MAC protocol. In this case, the use of TDMA and pre-scheduled timeslots prevents message collisions, and frequency hopping and retransmissions limit the effects of temporal and frequency interference (a retransmission occurs on a different frequency).

C. Decentralized Methods for Industrial Data Distribution

The composition of the two decentralized methods is shown in Table I labeled as D1 and D2. The common aspect of the decentralized methods is that they cache the generated data not at the central network controller, but at selected nodes, distributed in the network. Method D1 caches the data at the locations of the lowest common ancestors (LCAs) for each producer-consumer pair, while method D2 caches the data at the locations of the proxies. The proxies are selected before the industrial data distribution process begins, in the following manner: the network controller is initially set as the first proxy of the network. Then, the number of proxies is gradually increasing, until it reaches a number such that the average access latency does not violate the maximum data access latency threshold. In every iteration, the exact selection of the next proxy in the network is performed using a myopic greedy addition. Each candidate node is examined, and the one whose addition to the current solution reduces the average access latency the most is added to the incumbent solution. To this end, the latency between a candidate proxy and a consumer is estimated as the length of the shortest path that is connecting them multiplied by the expected latency on each hop. The NET layers of the two methods are different. Method D1 is using RPL in storing mode in order to route the relevant data to the LCAs, while method D2 is using source routing to the proxies.

III. Simulation Model

We conducted a performance comparison among the methods described in section II through simulations. We used OMNET++ Network Simulator version 4.4.1.[1] As far as the modeling of the WirelessHART protocol stack is concerned, we started with NIST modules [15],[2] which we customized according to our needs. In this section, we describe the main peculiarities of the models we developed for the presented methods. As a preliminary consideration, we point out that all the simulations consider networks with a square grid topology (Fig. 1), where the network controller is always the node with coordinates $(0, 0)$ in the top left vertex. We did not reproduce node failures nor battery discharge; hence, all nodes correctly work for the whole simulation duration.

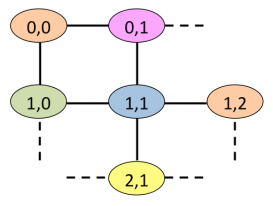

Fig. 1. Square grid topology scheme.

A. Route computation

Since nodes never crash in our simulations and the topology does not change, routes in the networks may be computed offline, and then uploaded in the models in order to speed up simulation execution. As the routing algorithm, we used Dijkstra's algorithm [16] to compute the shortest-path tree rooted in the network controller and connecting all nodes. The computed tree is used to obtain the needed routes in the presented methods.

We consider full-duplex links. For RPL in storing mode, the path between any two nodes s and d follows the branches of Dijkstra's tree from s up to the LCA of both s and d, and then downward along the appropriate branch toward d. Hence, in RPL storing, every node has in its routing table the routes to both its parent, and all its descendants. In RPL non-storing, the

[1]https://omnetpp.org/

[2]https://github.com/usnistgov/tesim_omnetpp

routing tables of all nodes but the controller contain just the address of the parent node, while the controller has routes to every other node. In WirelessHART, nodes know just the route to reach the controller, which by contrast knows the routes to every other node. Finally, in proxy source routing, every node knows just the route to its own proxy. In all cases, the routing tables are filled as a preliminary operation when the simulation starts.

B. *Channel scheduling for WirelessHART*

One main problem we encountered in modeling WirelessHART is that the channel scheduling mechanism has yet to be standardized, and just some requirements are indicated in the standard document [17]. Two problems have been dealt with: how to assign slots to pairs of nodes in order to avoid collisions, and how to form the superframe.

As far as the former problem is concerned, we implemented a graph coloring algorithm on the tree topology, with two constraints to be fulfilled. Considering the maximum number of neighbors owned by inner nodes, 5 colors are needed for the solution, and they must be assigned to nodes so that two neighbor nodes have different colors, so as to send their frames in different slots, and avoiding collisions between their transmissions. Moreover, the neighbors of a certain node n must have different colors, in order to avoid the hidden terminal problem. The example coloring in Fig. 1 satisfies both requirements; e.g., contemporary transmissions from nodes with coordinates $(0,0)$ and $(1,1)$ cannot occur, thus avoiding collisions in nodes with coordinates $(0,1)$ and $(1,0)$. Graph coloring too is computed offline, and assigned slots are configured in the nodes when the simulation starts.

As far as the latter problem is concerned, we opted for assigning slots in the superframe with the aim of maximizing the sleep time of nodes and minimizing latency. The trivial solution of building a superframe formed by N slots, with N the number of nodes in the network, is unsatisfactory. Indeed, it neglects the possibility of nodes that contemporarily use a slot of a given color without provoking collisions. This is for instance the case of nodes with coordinates $(0,0)$ and $(1,2)$ in Fig. 1, which are not neighbors nor they have any neighbor in common. Moreover, it forces a node with k neighbors to use k superframes to send one frame to each one of them. As an example, in a network with 180 nodes, and with slot duration of 10 ms., a node with 4 neighbors will spend $180 \times 4 = 720$ ms. to send a frame to each one of its neighbors. A node stays awake for up to 5 slots (1 for transmission and 4 for reception to/from neighbors) in each superframe. The alternative solution of having one slot per color in a superframe is disadvantageous under the point of view of energy consumption. Indeed, a node with four neighbors must stay awake in all 5 slots of the superframe to both send in its slot and receive from its neighbors.

In our solution, by contrast, a superframe is such that, for each node, there are as many slots as the number of its neighbors; e.g., node 3 in Fig. 1 is entitled to use 4 blue slots in each superframe. This way, in a certain superframe, a node n may send a frame to each one of its neighbor, and parallelism between distant nodes is allowed. With this policy, the upper bound on the superframe size is $\#colors \times node\_degree$, which in our networks amounts to $5 \times 4 = 20$ slots – corresponding to 200 ms. – thus being independent of the network size N. The minimum superframe length is achieved when the tree degenerates to a list: in this case 3 colors are enough, each node has two neighbors, and $3 \times 2 = 6$ slots per superframe are needed. Hence, latency is limited. Furthermore, this solution allows to save battery energy with respect to the choice of using one slot per color. In fact, a node stays awake for at most 8 slots per superframe (4 for transmission and 4 for reception to/from neighbors), if it owns four neighbors. This is seldom the case in our simulations: with 180 nodes we obtained a superframe size of 15 slots and the maximum node degree is 3. We modeled queues in the MAC layer as multiple FIFO queues, one for each destination, so that a node may send a frame to a neighbor as soon as a slot for that neighbor occurs, also if older frames to other neighbors exist (but in the respective, different, queue).

In the next section, the simulation conditions are detailed, including the number of consumers and proxies configured for each simulation.

IV. Performance Evaluation

We study the performance of the presented methods in two different scenarios, namely, either with variable number of nodes and fixed percentage of both producers and consumers, or with fixed number of nodes and variable number of consumers. In all simulations, the producers generate new data according to a uniform distribution probability with a period of 3s. Each consumer generates 400 requests; generation begins with a few seconds of delay from the simulation start in order to allow producers to produce the data. Each simulation reproduces 40 minutes of system operation. Every experiment is run 5 times, with different random seeds, and results are averaged; in the plots, confidence intervals are also shown. Some preliminary tuning was needed in order to optimize system behavior. We set the queue length at MAC layer to 500, in order to avoid message loss when the network is underloaded. As far as CSMA/CA is concerned, preliminary experiments allowed to reach 100% message delivery when up to 7 retransmissions can be performed in order to overcome collisions. As far as the proxy selection algorithm is concerned, we set the average data access latency threshold to 60 ms.

The main considered performance metrics are: *success rate*, defined as the percentage of requests from consumers that obtained a response, and *latency*, defined as the time elapsed between a request generation by a consumer, and the reception of the corresponding response.

In the first scenario, the number of nodes varies between 20 and 180. For all cases, both producers and consumers amount to 10% of the total number of nodes. In Fig.2(a), the success rate is shown. The use of proxies in method D2 allows to maintain a success rate very close to 100%, and

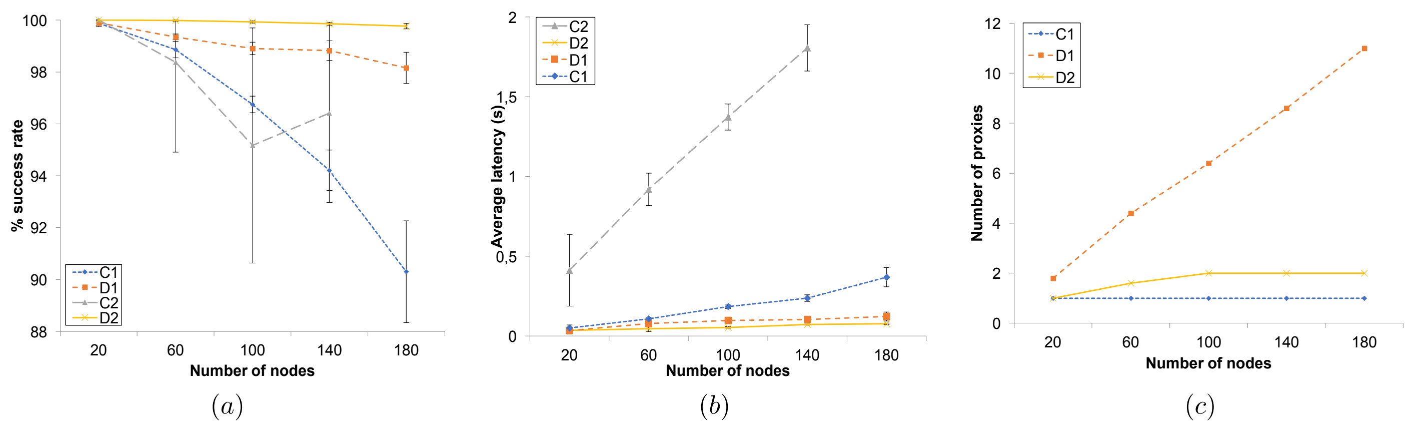

Fig. 2. Performance for variable number of nodes: (a) success rate; (b) average latency; (c) number of proxies.

D1 (RPL in storing mode) mimics quite well this behavior thanks to the proxies placed in the LCAs of the producer-consumer pairs. By contrast, centralized solutions are heavily penalized with the growing number of nodes, and therefore of both consumers and producers: this leads to a higher number of generated messages that converge toward the central network controller. In the case of C1 (RPL in non-storing mode), an increasing number of losses due to collisions is observed as a consequence. In the case of C2 (WirelessHART), which uses time slotting, collisions do not occur. Rather, messages accumulate in the queues and are lost due to buffer overflow, while nodes wait the appropriate slot for transmitting. This also explains the discontinuity for 100 nodes: with increasing number of nodes, the tree produced by the Dijkstra's algorithm changes, and buffer congestion distributes differently across nodes. Results for C2 with 180 nodes are not available as the model was computationally too heavy for the simulator, which ran out of memory.

The average latency, which is shown in Fig.2(b), confirms that the employment of proxies allows to reduce latency. The huge growth in C2 latency is due to messages waiting in the queues for longer times before being transmitted, which provokes higher delays than the retransmissions performed by CSMA/CA in the case of C1. We also measured the maximum latency observed by consumers: it shows the same trend as the average latency, with a greater separation amongst curves. Method D2 allows to fulfill the requirement about latency quite well: average latency is below 60ms for networks of up to 100 nodes. For increasing number of nodes, the constraint is slightly violated (average latency is 77ms with 180 nodes) and the algorithm tries to deal with this by increasing the number of proxies from 1 to 2, as shown in Fig.2(c) where the number of LCAs used as proxies by D1 is shown as well. The lowest latency achieved by D2 indicates that the proxies are placed in a more appropriate place – depending on the positions of both producers and consumers – than with D1 where the placement just depends on the tree topology. Both C1 and C2 have just one proxy that is the central network controller.

In order to study the traffic distribution, and also to estimate energy consumption by different nodes, we analyze heatmaps (Fig. 3) taken in a network of 100 nodes and based on the number of messages transmitted and received by nodes, which are the most energy-consuming operations. A centralized solution such as C1 (and similarly C2) concentrates traffic on the nodes of the tree nearer to the controller (Fig.3(a)). The number of managed messages, compared against the maximum queue length, allows to highlight the problem of buffer overflow mentioned above for C2: nodes nearer to the controller discard almost 10000 messages. By contrast, the heatmap for D1 (Fig.3(b)) evidences that the tree backbone – where LCAs lie – is formed by nodes on the left side of the network, which are also the most congested in a centralized solution (Fig.3(a)). The two proxies selected by D2 (Fig.3(c)) are the controller (0,0) and the node with coordinates (5,6).

In the second scenario, we consider a network of 100 nodes (hence, 10 producers) and vary the number of consumers such that ≥ 1 consumers correspond to each producer. The results confirm what observed before: while proxies allow to maintain a success rate (Fig.4(a)) very close to 100% (99.785% with 20 consumers), the other solutions – and in particular the centralized ones – suffer the higher traffic, which produces either collisions (in the case of CSMA/CA), or higher latencies and message drop (in the case of TDMA). The average latency is shown in Fig.4(b) and is similar to what obtained in the first scenario. This time, the network size is constant while consumers become more dense. This is why the values are almost constant, with a small decline for 15 consumers where possibly some consumer is nearer to the controller than with 10 consumers. The D2 method succeeds in fulfilling the latency threshold with an average latency of at most 66ms for the case of 20 consumers. As far as the maximum experimented latency is concerned, in C2 it is constant and equal to 2.8s, determined by the consumer furthest from the controller. By contrast, the decentralized solutions achieve maximum latency of 0.62s for D1 and 0.46s for D2 respectively, showing their adaptability to the positions of both producers and consumers. It is worth noting that D2, while able to guarantee an average latency comparable to the configured threshold, provides no guarantee for the performance observed by the worst consumer. The maximum latency of 0.46s is obtained with 20 consumers, while in the first scenario the maximum latency was 0.40s for a network of 180 nodes.

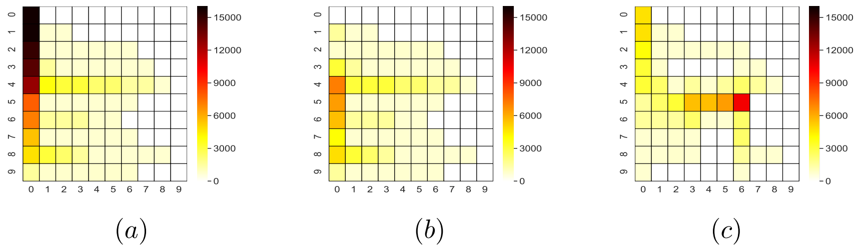
Fig. 3. Traffic heatmap for: (a) C1 (RPL in non-storing mode); (b) D1 (RPL in storing mode); (c) D2 (proxy selection algorithm).

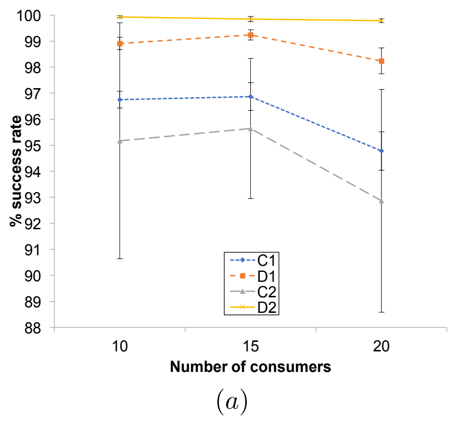
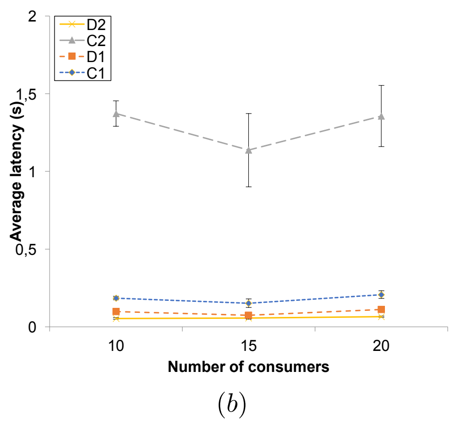
Fig. 4. Performance for variable number of consumers: (a) success rate; (b) average latency.

V. Conclusions

In this paper, we used WirelessHART, RPL and proxy selection as building blocks, in order to compose the protocol stacks of four different industrial data distribution methods (both centralized and decentralized) in wireless industrial networks. We implemented the methods in OMNeT++ and we evaluated their performance via a detailed simulation analysis. We demonstrated that the careful selection of a limited set of proxies for data caching in the network can lead to increased data delivery success rate and low data access latency.

References

[1] S. Yin, X. Li, H. Gao, and O. Kaynak, "Data-Based Techniques Focused on Modern Industry: An Overview," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 1, pp. 657–667, Jan 2015.

[2] D. V. Queiroz, M. S. Alencar, R. D. Gomes, I. E. Fonseca, and C. Benavente-Peces, "Survey and systematic mapping of industrial Wireless Sensor Networks," *Journal of Network and Computer Applications*, vol. 97, pp. 96 – 125, 2017.

[3] M. C. Lucas-Estan, M. Sepulcre, T. P. Raptis, A. Passarella, and M. Conti, "Emerging Trends in Hybrid Wireless Communication and Data Management for the Industry 4.0," *Electronics*, vol. 7, no. 12, 2018.

[4] J. Song, S. Han, A. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control," in *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2008, pp. 377–386.

[5] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," IETF RFC 6550, March 2012. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6550.txt

[6] T. P. Raptis, A. Passarella, and M. Conti, "Maximizing industrial IoT network lifetime under latency constraints through edge data distribution," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, May 2018, pp. 708–713.

[7] T. Lennvall, S. Svensson, and F. Hekland, "A comparison of WirelessHART and ZigBee for industrial applications," in *2008 IEEE International Workshop on Factory Communication Systems*, May 2008, pp. 85–88.

[8] S. Petersen and S. Carlsen, "WirelessHART Versus ISA100.11a: The Format War Hits the Factory Floor," *IEEE Industrial Electronics Magazine*, vol. 5, no. 4, pp. 23–34, Dec 2011.

[9] D. Chen, M. Nixon, S. Han, A. K. Mok, and X. Zhu, "WirelessHART and IEEE 802.15.4e," in *2014 IEEE International Conference on Industrial Technology (ICIT)*, Feb 2014, pp. 760–765.

[10] M. Vucinic, B. Tourancheau, and A. Duda, "Performance comparison of the RPL and LOADng routing protocols in a Home Automation scenario," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2013, pp. 1974–1979.

[11] H. Audeoud and M. Heusse, "Experimental Comparison of Routing Protocols for Wireless Sensors Networks: Routing Overhead and Asymmetric Links," in *2017 29th International Teletraffic Congress (ITC 29)*, vol. 1, Sep. 2017, pp. 55–62.

[12] T. P. Raptis, A. Passarella, and M. Conti, "Performance Analysis of Latency-Aware Data Management in Industrial IoT Networks," *Sensors*, vol. 18, no. 8, 2018.

[13] O. Khader, A. Willig, and A. Wolisz, "WirelessHART TDMA Protocol Performance Evaluation Using Response Surface Methodology," in *2011 International Conference on Broadband and Wireless Computing, Communication and Applications*, Oct 2011, pp. 197–206.

[14] W. Gan, Z. Shi, C. Zhang, L. Sun, and D. Ionescu, "MERPL: A more memory-efficient storing mode in RPL," in *2013 19th IEEE International Conference on Networks (ICON)*, Dec 2013, pp. 1–5.

[15] Y. Liu, R. Candell, K. Lee, and N. Moayeri, "A simulation framework for industrial wireless networks and process control systems," in *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, May 2016, pp. 1–11.

[16] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec 1959.

[17] M. Nobre, I. Silva, and L. A. Guedes, "Routing and Scheduling Algorithms for WirelessHART Networks: A Survey," *Sensors*, vol. 15, no. 5, pp. 9703–9740, 2015.