A Distributed Data Management Scheme for Industrial IoT Environments

Theofanis P. Raptis, Andrea Passarella
Institute of Informatics and Telematics
National Research Council
Pisa, Italy
Email: theofanis.raptis@iit.cnr.it, andrea.passarella@iit.cnr.it

***Abstract*—Industrial IoT networks are typically used for monitoring systems and supporting control loops, as well as for movement detection systems, process control and factory automation. To this end, data generated by monitoring IoT devices are collected, elaborated and sent to controllers and actuators. The routing of data from IoT sensors to actuators is an integral part of any large-scale industrial network for maintaining critical delay requirements. Centralised schemes are typically used, whereby data are transferred to a central network controller, from where they are accessed by any other node requiring them. This may result in significant overheads and suboptimal resource consumption. In this paper, we propose a distributed, cooperative *Data Management Layer (DML)*, whereby nodes cooperate to store data within the network. The DML is decoupled yet interacts with the underlying Network Plane. Specifically, given a set of data, the sets of nodes generating and requesting them, and a maximum access delay that requesting nodes can tolerate, the DML efficiently identifies a limited set of proxies in the network where data are stored. Given the mentioned constraints, we investigate the (computationally difficult) problem of finding which network nodes to select as proxies and we propose a simple method to address it. We demonstrate that the proposed method (i) guarantees that access delay stays below the given threshold, and (ii) significantly outperforms centralised and even distributed approaches, both in terms of access latency and in terms of maximum latency guarantees.**

***Index Terms*—Industry 4.0, Data Management, Internet of Things**

I. Introduction

Industrial IoT networks (Fig. 1) are typically used for monitoring systems and supporting control loops, as well as movement detection systems for use in process control (i.e., process manufacturing) and factory automation (i.e., discrete manufacturing) [1], [2]. To maintain the stability and control performance, industrial monitoring and control applications impose stringent end-to-end delay requirements on data communication between sensors and actuators [3]. Missing or delaying the process data may severely degrade the quality of control [4]. In recent years many standards have been issued by international bodies to support the development of industrial networks in different application domains, like IEEE 802.15.4e [5] and WirelessHART [6].

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

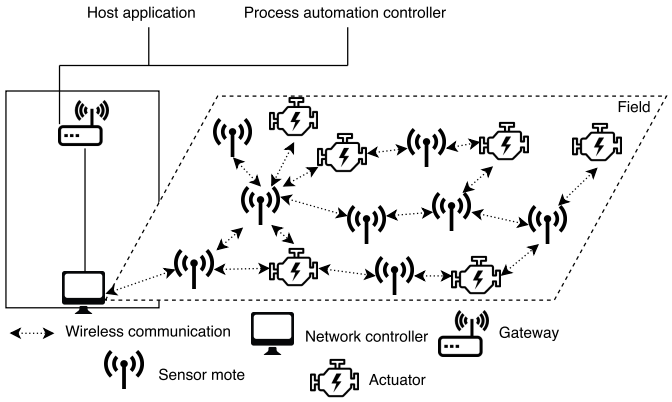


Fig. 1: A typical industrial IoT network setting.

The routing of data from IoT sensors to actuators and controllers is an integral part of any large-scale industrial network for maintaining the delay requirements [7]. On the other hand, in today's typical configurations, data management is quite primitive. Usually, the generated data are transferred to a central network controller using wireless links [8]. The controller analyses the received information and, if needed, changes the behaviour of the physical environment through actuator devices [9]. However, routing the data centrally, as well as imposing data transfers back and forth in the network may lead to severely sub-optimal paths [10], which in turn negatively affect the overall network latency. At the same time, those transfer patterns lead to poor network performance, as the devices often have to tolerate longer response times than necessary. This is becoming even more serious with decentralisation of control decision, by which not only a central controller, but also other nodes in the network may take decisions on how to configure IoT devices and actuators. Clearly, this calls for more distributed schemes for managing data. Therefore, cooperation becomes essential, as nodes in an industrial network will have to cooperate in storing and serving data to each other, in order to maximise the overall performance of the network.

Our contribution. In this paper, we introduce a *Data Management Layer (DML)*, which operates independently from and complements the routing process. Assuming that

978-1-5386-3839-2/17/$31.00 ©2017 IEEE

applications in industrial IoT networks require that there is (i) a set of *source* nodes producing data (e.g., IoT sensors), (ii) a set of *destination* nodes require those data to implement the application logic (e.g., IoT actuators), and that (iii) a maximum latency $L_{\max}$ that destinations can tolerate in receiving data after they have requested them, the DML offers an efficient method for regulating the data distribution methods among sources and destinations. In doing so, our DML realises an efficient cooperation scheme across all the nodes of the network: It selectively assigns a special role to some of the network nodes, that of the *proxy node*. Each node that can become a proxy potentially serves as an intermediary between sources and destinations, even though the node might be neither a source nor a destination. As shown in the paper, the cooperative scheme drastically outperforms both other simpler cooperative schemes, as well as one of the de-facto standards in industrial environments, i.e., a totally non-cooperative scheme where all data management functions are centralised in one unique controller.

We observe that not all data need to be transferred to the central network controller prior to delivery to the actuators (as traditional industrial routing approaches usually impose); in fact, data can be also stored managed locally at the proxies, exploiting, when needed, additional levels of information. The solution proposed in this paper takes the data distribution process one step further than the three obvious "naive" alternatives: (i) keep the data at the sources, which results in the longest access delays; (ii) keep the data at the network controller, which results in access delays possibly above the requirements; (iii) replicate the data on all destinations, which creates unnecessary traffic and energy dissipation in the network. We demonstrate that the employment of the DML is critical to guarantee a small access delay for the destination nodes (below $L_{\max}$) compared to other state of the art schemes and, at the same time, a small of proxies used.

We note that, proxy nodes are intermediate nodes acting as caching agents between source nodes and destination nodes. If properly selected, proxy nodes can reduce the access delay and alleviate the potential network congestion. However, when a node is selected as a proxy, it has to significantly increase its storing, computational and communicational activities, rendering the proxies as an important system resource. Simply put, selecting proxy nodes in high numbers and at the "wrong" places is not only costly, but also does little to improve the system performance[1].

Roadmap of the paper. In Section II, we provide a brief summary of concepts related to this work. Since (to the best of our knowledge) this is the first paper introducing a separation of the Data Management Plane from the Network Plane in industrial environments, we focus on relevant approaches in other networking domains. In Section III, we provide the model of the settings we consider, as well as the necessary notation. In Section IV, we introduce the DML, we provide some fundamental definitions and we present the problem that this paper addresses. Then, we break the problem in two modular subproblems: the data distribution and delivery subproblem and the selection of proxies subproblem. Finally, we evaluate the performance of the DML in Section V, also in comparison with two other methods used in industrial environments, and we conclude and provide some insights for future work in Section VI.

[1]Note that, the coherency of data that reside on proxy nodes can be achieved in a variety of ways [11], and is beyond the scope of this paper.

II. Related concepts

To the best of our knowledge, distributed data management approaches, like the ones proposed in this paper, have not been considered in industrial IoT environments in the past. However, similar concepts have been used in other networking fields, in order to satisfy different requirements with diverse technologies:

- *Data Management Layer:* There have been several architectures presented in different fields, which take advantage of a decoupling process of the data management plane from the routing plane. For example, in the field of *Content Distribution Networks*, authors of [12] have proposed SCAN. SCAN utilises an underlying distributed object routing and location system, which combines dynamic replica placement with a self-organising application level multicast tree to meet client QoS and server resource constraints.
- *Proxies:* Proxy placement strategies have been a usual research topic in the field of *Information Retrieval over the Internet*. For example, in [13], the authors presented one of the first ways to investigate placement policies in the web and optimise a given performance measure for the target web server subject to system resources and traffic pattern. Proxies also play an important role in the field of *Content Distribution Networks*. For example, the authors of [14] have presented a heuristic algorithm for proxy server placement in Content Distribution Networks, by taking into account the hierarchical Internet structure and the routing policy constraint resulting from it. Last but not least, proxy caching has been traditionally very important in the *Multimedia Streaming* field. In [15], the authors study three media segmentation approaches to proxy caching, and group media streams into various segments for cache management.
- *Data distribution:* Different ways of distributed data in diverse networking environments have been proposed over the years. The authors of [16] provide a useful survey of various diverse data distribution architectures in *Information-Centric Networking (ICN)* environments. In [17], they discuss context data distribution for *Mobile Ubiquitous Systems*.

Although we use similar concepts, the purpose of this paper is to highlight the necessity of adopting smart data distribution methods in industrial IoT environments, especially when it comes to critical end-to-end delays.

III. System modelling

The networks we consider consist of a set of $S = \{u_1, u_2, ..., u_n\}$ nodes and a central network controller $C = u_1$ which are deployed in an area of interest $\mathcal{A}$. The nodes abstract industrial IoT devices and can be either sensor motes or actuator and controller devices. The communication range r_u of node u varies according to requirements of the underlying routing protocol. Nodes u, v are able to communicate with each other iff $r_u, r_v \geq d(u, v)$, where $d(u, v)$ is the Euclidean distance between u and v.

A known *underlying routing protocol* is taking care of the propagation of generated data. For example, a representative routing protocol which can be used in the networking stack of industrial applications using constrained devices is the Routing Protocol for Low-Power and Lossy Networks (RPL, in storing or non-storing mode) [18]. We also assume the use of *low-power multi-hop MAC protocol*, capable of addressing the emerging needs of industrial IoT environments. For example, IEEE 802.15.4e or WirelessHART which include slotted access, shared and dedicated slots, multi-channel communication, and frequency hopping [19], [6].

In the traditional industrial IoT settings, sensor nodes perform monitoring tasks and in some cases their sensor data are needed either by other sensor nodes (which could need additional data to complement their own local measurements) or by actuator nodes (which use the sensor data so as to perform an actuation). When needed, a node u (destination) can ask for data of interest using the principles defined by the underlying routing protocol from a sensor node v (source). When v receives a data request, it propagates the requested data back, along the same routing path, starting at v and finishing at the destination node u. Note that the latency of this individual data delivery is twice the length of the path between u and v.

We assume that the controller C is able to maintain *centralised 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 C knows all the shortest paths in the network and comes with an $n \times n$ matrix $\mathbf{D}$, where $\mathbf{D}_{u,v}$ is the length of the shortest path between nodes u and v. This assumption is valid, since the offline shortest path computation between two nodes is a classic problem in graph theory and can be solved polynomially, using Dijkstra's algorithm [20]. Note that, as will be clear in the following, only the control of the data management plane is centralised, while the data plane itself is distributed and cooperative, as data are stored on multiple nodes, which cooperate to achieve the optimal performance of the network according to the objective function defined next.

It is customary in wireless networking applications to estimate that the energy required to transmit from u to v is proportional to $d(u, v)^\mu$, where μ is the path-loss coefficient. In perfect conditions $\mu = 2$, however in more realistic settings (in presence of obstructions or noisy environment) it can have a value between 2 and 4 [21]. In this paper we assume $\mu = 2$ for simplicity. However, it is possible to extend our results for other values of μ which are greater than 2.

In order to simplify the problem and the analysis, we assume that each $u \in S$ is able to request data from up to one source $v \in S$. In other words, we can define a vector $\mathbf{R}$, in which $\mathbf{R}_u = v$ iff u is requesting data from v, or $\mathbf{R}_u = 0$ otherwise. We define a set $S_r \subseteq S$, with $|S_r| = m$ (and obviously $m < n$), where $u \in S_r \iff \mathbf{R}_u \neq \mathbf{0}$.

We measure time in time units. During a time unit, every node can transmit a message to another node.

In this paper, we consider the following problem:

Problem 1. *Given a maximum data access latency threshold L_{max} imposed by the industrial operator, design a data distribution and delivery scheme which, using a minimal number of proxies, ensures that the average data access latency of the destination nodes in the network does not exceed L_{max}.*

IV. Data Management Layer

In order to manage the data distribution process and decrease the average latency in the network, we introduce the *Data Management Layer* (DML). The main idea behind the DML is decoupling the Network plane from the Data Management Plane. Fig. 2 depicts the DML high-level structure. The basic function of DML is the selection of some nodes which will act as *proxy* nodes and the definition of efficient techniques for data distribution and delivery. More specifically, the role of the DML is twofold:

1) *Provide an efficient data distribution and delivery method:* The DML imposes some rules on the circulation of data to the network, according to the available proxies, using the principles of the underlying routing mechanism in the Network Plane.
2) *Identify the network nodes that will operate as proxies:* The DML defines a set $P \subset S$, the elements of which are the selected proxy nodes. The number of the proxies can range from 1 to $n - 1$. The case of 1 proxy is equivalent to having only the controller C operating as a single point of data distribution. In this case, the data distribution is functioning as in traditional industrial IoT environments.

A. Internal and external latency

The introduction of proxies in the network, as well as the storage of data of interest at the proxy locations, is giving us the ability to spatially reconfigure the distribution of data in the network and break the data delivery latency into two individual sub-latencies; the *internal latency*, which concerns the access delay that destination nodes have until the get the data requested from the corresponding proxy, and the *external latency*, which concerns the delay of data distribution from the sources to the proxies:

Definition 1 (Average internal latency)**.** *We define as internal latency, L_u^{int}, the amount of time required for the data to reach*

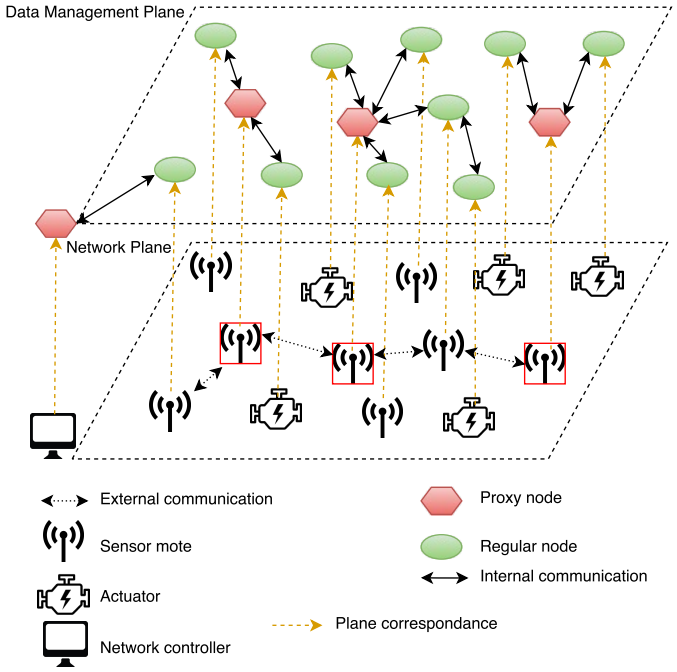


Fig. 2: Decoupling of the Data Management Plane and the Network Plane.

node u, after u's request. We define as average internal latency the quantity

$$\overline{L^{\text{int}}} = \frac{\sum_{\forall u \in S_r} L_u^{\text{int}}}{m}.$$

Definition 2 (Average external latency)**.** *We define as external latency, L_u^{ext}, the amount of time required for the data requested by $u \in S_r$ to reach the assigned proxy node $p \in P$, after p's request to source node $v \in S$. Note that when the only proxy node is the controller C (or equivalently $|P| = 1$), then there is no external latency in the network. We define as average external latency the quantity*

$$\overline{L^{\text{ext}}} = \frac{\sum_{\forall u \in S_r} L_u^{\text{ext}}}{m}.$$

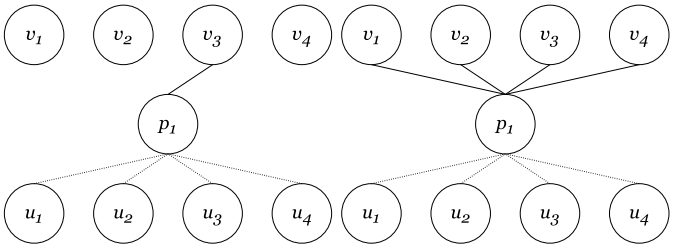
Note that, in this paper, we put focus on $\overline{L^{\text{int}}}$, and more specifically on providing an efficient solution in order to guarantee that $\overline{L^{\text{int}}} \leq L_{\max}$

B. Data distribution and delivery

The first subproblem, that the DML addresses, is the definition of a data distribution and delivery method to the destination nodes. We define the first subproblem as follows:

Subproblem 1.1 (Efficient Data Distribution and Delivery)**.** *Given a proxy selection, derive a method to distribute and deliver the data efficiently to the destination nodes.*

The demarcation of the total latency in internal latency and external latency allows us to break the data delivery process in two parts (each of which are using the standard routing mechanism provided by the Network Plane). At the first part,



(a) All destination nodes need data from a single source node. (b) All destination nodes need data from different source nodes.

Fig. 3: Toy example showing the two extreme cases of external communication, in a network consisting of 4 destination nodes u_1, u_2, u_3, u_4, 4 source nodes v_1, v_2, v_3, v_4 and a proxy p_1

there is the internal communication; a destination node u is able to request data of interest from the corresponding proxy p. Upon receiving the request from u, p is able to directly forward the data to u if this data is already available. This means that the data may have arrived at p from the corresponding source node v at another time in the past. At the second part, there is the external communication; a proxy p is able to receive and store data from a source node v.

This demarcated model of data exchanges can be formulated as a *publish/subscribe (pub/sub)* model [22]. In a pub/sub model, nodes interested in data subscribe to it, i.e., they denote their interest for it to the network, and nodes offering the data publish advertisements to the network. Inside the network, the proxies are responsible for matching subscriptions with publications i.e., they provide a rendezvous function, and for storing the available data according to the corresponding subscriptions.

The strength of the pub/sub communication model stems from the fact that publication and subscription operations are decoupled in time and space [23]. The communication between a publisher and a subscriber does not need to be time-synchronised, i.e., the publisher (source node) may publish data before any subscribers (destination nodes) have requested it and the subscribers may initiate data requests (at the proxies) after publication announcements. Publishers do not usually hold references to the subscribers, neither do they know how many subscribers are receiving a particular publication.

The main features that the pub/sub with proxies approach provides are (i) the avoidance of the data to flow multiple times from the source towards the destinations served by the same proxy; (ii) the proactive delivery of data generated by sources to the proxies, such that when a request for data comes from a destination node, it can be served in limited time by the proxy. A visual depiction of feature (i) can be found in Fig. 3, where we have a toy example with two extreme cases of external communication in a network of 9 nodes. In the first case, all destination nodes need data from a single source node and in the second case, all destination nodes need data from different source nodes. It is apparent that the amount of external communication can be significantly reduced in the first case, if an appropriate grouping method for the data

deliveries from v to p is employed[2].

The pub/sub process that we consider uses the underlying routing mechanism and starts with the source nodes advertising to the proxies the available data that they can generate. Typically, sensor data is characterised by its *content*, which is a tuple of its main characteristics, and can be carried along together with data, without imposing extra communication overhead. A very simple case of content is the pair (v, t), where v is the source node and t is the type of the sensor measurement (temperature, humidity, etc.). The proxies receive those advertisements and construct an announcement list for distribution to the destination nodes. After the distribution of the announcements, a destination node u is able to subscribe (by informing corresponding proxy p) to the data of their interest, coming from source v. At this point, p informs the source v that there is an interest for v to publish data to p. We assume that the publishing part, which corresponds to the external latency, is taking place periodically, following some fixed, predefined time interval[3]. Eventually, when the publishing process starts, p is holding the most up-to-date data coming from v. The destination node u is able to send a data request to p and then receive the data of interest, whenever access to data is needed.

C. Selection of the proxies

The second subproblem, that the DML addresses, is the selection of the exact proxies needed for the data distribution and delivery process, so that average internal latency remains below $L_{\max}$. As we mentioned before, the nodes in the network are considered as a very important resource, and thus, every selection of a node as a proxy (and its inherent overuse of storage, computational and communicational capabilities), is considered as an additional resource consumption in order to guarantee that $\overline{L^{\text{int}}} < L_{\max}$.

In order to make the effect of the number of selected proxies more clear, we provide a visual example. Fig. 4 displays the average internal latency $\overline{L^{\text{int}}}$ for different number of proxies in the network when we set $m = 0.4 \cdot |S|$ and $L_{\max} = 8$. The plot at the rightmost point lies even below the minimum average latency ($\overline{L^{\text{int}}} = 2$, one hop for the request and one hop for the delivery), which is represented by the green line, because some of the selected proxies might also request data and consequently, $L_p^{\text{int}} = 0$ for every p that is requesting data. By sequentially decreasing the number of proxies, $\overline{L^{\text{int}}}$ starts taking values between the minimum average internal latency and the maximum latency threshold $L_{\max}$, which is represented by the red line. As we will see in the performance evaluation section, the DML ensures that $\overline{L^{\text{int}}}$ will not surpass $L_{\max}$. If we further decrease the number of proxies, then we have that $\overline{L^{\text{int}}} > L_{\max}$, and the latency constraint is not met. At the leftmost point of the plot, we can see the latency achieved when using only one proxy (the controller C, or in other words, when the DML functionalities are absent), which is much higher than when employing additional proxies.

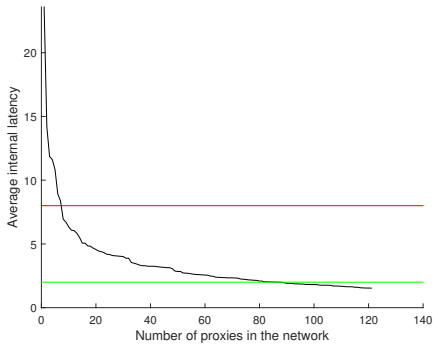


Fig. 4: $\overline{L^{\text{int}}}$ for different $|P|$, $n = 500$, $m = 0.4 \cdot |S|$.

It is clear that the second subproblem introduces two different issues. On the one hand, there is the constraint imposed by the industrial operator which necessitates that $\overline{L^{\text{int}}} < L_{\max}$ and on the other hand there is the need for a low resource consumption in terms of proxies used. For this reason we define the second subproblem as follows:

Subproblem 1.2 (Minimum Proxy Number (MPN))**.** *Given a set S of nodes, a set $S_r \subset S$ of destination nodes and a maximum latency threshold L_{max}, minimise the number of proxies needed in the network so as to guarantee $\overline{L^{\text{int}}} \leq L_{max}$.*

We now present an integer program formulation for MPN. We define two sets of decision variables, (a) $x_p = 1$, if $v \in S$ is selected as proxy and 0, otherwise, (b) $y_{u,p} = 1$, if node $u \in S$ is assigned to proxy $p \in S$ and 0, otherwise. Then, the integer program formulation is the following:

$$\text{Min.:} \quad \sum_{p \in S} x_p \tag{1}$$

$$\text{S. t.:} \quad \sum_{u \in S_r} \sum_{p \in S} \frac{\mathbf{D}_{u,p} \cdot y_{u,p}}{m} \leq L_{\max} \tag{2}$$

$$\sum_{u \in S_r} y_{u,p} = 1 \qquad \forall p \in S. \tag{3}$$

$$y_{u,p} \leq x_u \qquad \forall u \in S_r, \forall p \in S. \tag{4}$$

$$x_p, y_{u,p} \in \{0, 1\} \qquad \forall u \in S_r, \forall p \in S. \tag{5}$$

The objective function (1) minimises the number of proxies. Constraint (2) guarantees that $\overline{L^{\text{int}}} \leq L_{\max}$. Constraints (3) guarantee that each node has to be assigned to one and only one proxy. Constraints (4) guarantee that nodes can be assigned only to proxies. Constraints (5) guarantee that all nodes are considered for potentially being selected as proxies and that all nodes requesting data are assigned to a proxy.

[2]In this paper, we do not investigate methods for efficient external communication, which could potentially improve the DML performance. In fact, at the performance evaluation we consider instances like the one displayed in Fig. 3b, so as to demonstrate that the DML can efficiently perform the data management even in the worst case.

[3]Since the given latency constraint $L_{\max}$ concerns the access delay of the destination nodes to the data, we are specifically focusing on improving the internal latency achieved by the data distribution and delivery process (of course, we also measure the effect of our methods on the external latency). For this reason we are not putting more emphasis in the publishing part of the pub/sub model

Algorithm 1: `ProxySelection`

Input : $S, \mathbf{D}, S_r, L_{\max}$

$P = u_1$

counter $= 2$

while counter $< n$ **and** $\overline{L^{\text{int}}} > L_{max}$ **do**

  $P = u_1$

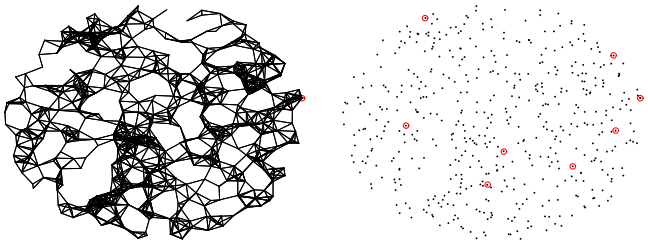
  for $i = 1 : 1 :$ counter **do**

    $$p = \arg\min_{u \in S} \sum_{v \in S_r} \min_{k \in P \cup \{u\}} \frac{\mathbf{D}_{k,v}}{m}$$

    $P = P \cup \{p\}$

  counter $++$

Output: P



(a) Network deployment. (b) P.

Fig. 5: `ProxySelection` Algorithm output for a network with $n = 500$.

MPN is computationally intractable, since it can be formulated as an integer program. This means that we are not able to optimally calculate the exact proxies needed so as to minimise their number while staying below $L_{\max}$. In order to address this problem, we designed Algorithm 1 (`ProxySelection`), in order to come up with a selection of proxies adequate for an for ensuring minimal internal latency. `ProxySelection` is a greedy algorithm which does not give the optimal solution. The reason why we chose this proxy selection method is to demonstrate that the introduction of the DML is able to outperform the traditional centralised methods, even with when adopting simple solutions.

Algorithm 1 initially sets as a first proxy the network controller C. Then, it gradually increases the number of proxies until it reaches a number with which the average internal latency $\overline{L^{\text{int}}}$ does not violate the maximum latency threshold $L_{\max}$. In every iteration, the exact selection of the of the next proxy in the network is performed using a very simple myopic greedy addition, similar to the k-Median greedy method presented in p. 26 of [24]. Each candidate node is examined and the one whose addition to the current solution reduces the average internal latency the most is added to the incumbent solution. Note that since we are interested in $\overline{L^{\text{int}}}$, `ProxySelection` does not take into account $\overline{L^{\text{ext}}}$.

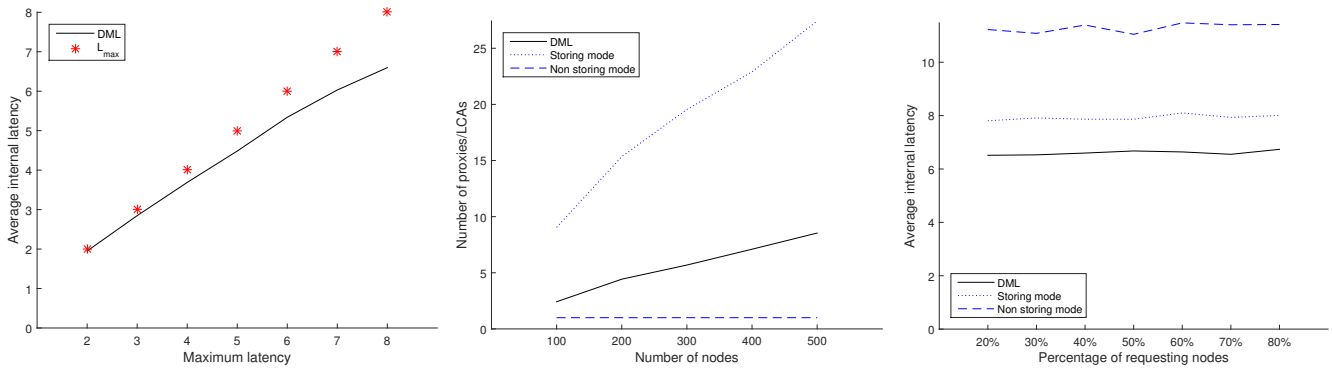
A visual example where the outcomes of Algorithm 1 can be observed is provided in Fig. 5. Fig. 5a displays a typical network deployment of 500 nodes, with the corresponding wireless links and with the controller C lying on the far right edge of the network, depicted as a red circle. Fig. 5b displays the locations of the final set P of proxies depicted as red circles after running Algorithm 1. Observing the spatial display of Fig. 5b, it is easy to understand that the final selection results in a balanced proxy selection, ensuring that even isolated nodes, which are located near sparse areas of the network, also have close access to a proxy.

V. Performance evaluation

For the evaluation of the performance of the DML, we simulate a typical industrial IoT setting. The deployment area $\mathcal{A}$ is set to be circular with radius $R_{\mathcal{A}} = 1$ and we deploy n nodes uniformly at random in $\mathcal{A}$. We construct networks of different number of nodes, inserting the additional nodes in the same network area and at the same time decreasing the communication range r_u appropriately, so as to maintain a single strongly connected component at all times. More specifically, we apply a well known connectivity threshold presented in [25], in order to maximise the probability that the produced random instances are connected. Since $\mathcal{A} \subset \mathbb{R}^2$, an instance of the *random geometric graphs model* $\mathcal{G}(\mathcal{X}_n; r)$ is constructed as follows: We select n points $\mathcal{X}_n$ uniformly at random in $\mathcal{A}$. The set $\mathcal{X}_n$ is the set of vertices of the graph and we connect two vertices if their euclidean distance is at most r. In [25] it is shown that the connectivity threshold for $\mathcal{G}(\mathcal{X}_n; r)$ is $r_c = \sqrt{\frac{\ln n}{\pi n}}$. In this paper we consider random instances of $\mathcal{G}(\mathcal{X}_n; r_u)$ of varying density, by selecting $r_u = \sqrt{\frac{c \ln n}{\pi n}}$, for different values of $c > 1$, which guarantees that the produced random instance is connected with high probability. An example of a generated network of 500 nodes is depicted in Fig. 5a.

We assume an ideal, infinite bandwidth, so as to get conclusions about the data distribution process independently of the bandwidth restrictions and the details of the physical and MAC layers. The simulation environment we used is is Matlab. For statistical smoothness, we apply several times the deployment of nodes in the network and repeat each experiment 100 times. The statistical analysis of the findings (the median, lower and upper quartiles, outliers of the samples) demonstrate very high concentration around the mean, so in the following figures we only depict average values.

In order to measure the performance of the DML w.r.t. traditional industrial IoT alternatives, we implement two additional data delivery strategies, which do not differentiate the Data Management Plane from the Network Plane, and can be viewed as traditional routing mechanisms. The first strategy is the most traditional data delivery strategy in current industrial IoT environments and imposes that all data requests and data deliveries are being routed through the controller C. More specifically, the request is routed from u to C and then from C to v. At the next step, the data is routed from v again to

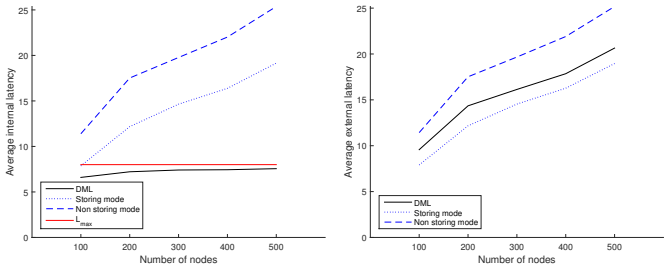


(a) Average internal latency for different values of $L_{\max}$.

(b) Number of proxies/LCAs for different number of nodes.

(c) Average internal latency for different percentages of requesting nodes.

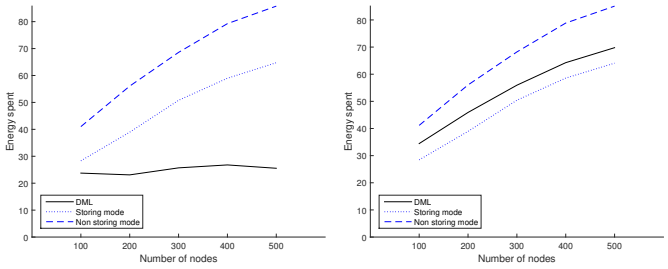
Fig. 6: Various metrics.



(a) Internal.

(b) External.

Fig. 7: Average latency for different numbers of nodes.



(a) Internal.

(b) External.

Fig. 8: Energy consumption for different numbers of nodes.

C and then from C to u. We call this mode of operation *non storing mode* and it is obvious that it is completely centralised and not cooperative. Note that this would be the simplest data management approach that can be implemented in routing mechanisms like RPL, where intermediate nodes are not allowed to cache data (thus, the RPL terminology non-storing mode). The second strategy is another, less commonly used in industrial IoT settings, but nevertheless useful alternative. It imposes that all data requests and data deliveries are being routed through the lowest common ancestor (LCA) of the routing tree instead of the controller C. The LCA two nodes u and v in the routing tree is the lowest (i.e., deepest) node that has both u and v as descendants. We call this mode of operation *storing mode*, because the LCAs should store additional information about their descendants, and it is obvious that it is a distributed alternative. Again, this is the simplest solution that one would implement with routing mechanisms like RPL in storing mode, i.e., when intermediate nodes between communication endpoints are allowed to cache content.

Different values of $L_{\max}$. We tested the performance of the DML for different values of $L_{\max}$, in networks of 100 nodes, with $m = 0.4 \cdot |S|$. The results are shown in Fig. 6a. The red points represent the values for the maximum latency threshold $L_{\max}$ provided by the industrial operator. The average internal latency achieved by the DML is always below the threshold, due to the provisioning of Algorithm 1. In fact, we can see that the more the value of $L_{\max}$ is increased, the larger the difference between $\overline{L^{\text{int}}}$ and $L_{\max}$ becomes. This happens because for higher $L_{\max}$ values, the latency constraint is more relaxed, and lower $\overline{L^{\text{int}}}$ can be achieved more easily.

Different number of proxies. We compare the three solutions w.r.t. the number of special nodes that they use. The DML is using proxies, the non storing mode is using the controller C and the storing mode is using LCAs. As we mentioned earlier, the use of special nodes is wasteful on resources. For example the proxies store the data requested and the correspondence of sources and destinations, and the LCAs hold routing information about their descendants. In fact, specifically in the case of industrial IoT applications, the devices which are used are very constrained in all aspects of operation (computation, communication, storage), and this issue has a larger impact. In Fig. 6b, we can see that the DML is performing really well compared to the storing mode and uses much less special nodes. Of course, the non storing mode is using just one special node for any network size, but this has a severe impact on the latency achieved.

Different percentages of requesting nodes. Another possible factor that could affect the average internal latency $\overline{L^{\text{int}}}$ achieved, is the percentage of destination nodes. In Fig. 6c, we can see that $\overline{L^{\text{int}}}$ remains constant for any percentage of requesting nodes, in every of the three alternatives. This is natural, considering that we do not take into account the effect of network congestion.

Average latency achieved. Fig. 7 displays the results on

the average latency for the three alternatives, for different numbers of nodes in the network. We provide details in terms of both average internal latency $\overline{L^{\text{int}}}$ (Fig. 7a) and average external latency $\overline{L^{\text{ext}}}$ (Fig. 7b). Regarding the internal latency, we can see that the efficient management of proxies provided by the DML results in a better performance compared to the other two alternatives. An interesting fact is that the $\overline{L^{\text{int}}}$ achieved by the DML respects the constraint provided by the industrial operator and always remains lower than $L_{\max}$ (red line). Regarding the external latency, we can see that, although the DML outperforms the non storing mode, its performance is somewhat poorer that the performance of the non storing mode. This is a result of the design of the `ProxySelection`, which is not taking into account any variables on the external latency. Consequently, the proxies of the DML are selected close to the destination nodes, but farther from the source nodes than in the case of the LCA placement of the storing mode.

Energy consumption. Fig. 8 displays the energy consumption in the network for the three alternatives, for different numbers of nodes. The energy consumption for internal communication is lower in the case of the DML. This is natural, since low latency comes with less transmissions in the network, which results in fewer energy demands. Although the energy consumption external communication is higher for the DML than for the storing mode, if a balanced period of publishing is maintained, then the overall energy consumption in the network is lower in the case of DML.

VI. Conclusions and future work

In this paper, we introduce a distributed, cooperative *Data Management Layer (DML)* for industrial IoT environments. The DML is decoupled yet interacts with the underlying Network Plane. Given constraints on the data access latency, we investigate the problem of finding which network nodes to select as proxies and we propose a simple algorithm to address it. We demonstrate that the proposed algorithm guarantees that access delay stays below the given threshold, and significantly outperforms centralised and distributed approaches, in terms of access latency and in terms of maximum latency guarantees. The purpose of this paper is to validate the introduction of the Data Management Layer in industrial IoT environments. For this reason, some simplifying assumptions were made, regarding the application of the proposed concepts. We verified that the DML is indeed enhancing the network performance and the next step for future work is to take into account more realistic assumptions and investigate their impact of the data distribution process. A first next step is to consider a limited bandwidth which can potentially lead to congestive collapse, when incoming traffic exceeds outgoing bandwidth.

References

[1] E. Molina, O. Lazaro, M. Sepulcre, J. Gozalvez, A. Passarella, T. P. Raptis, A. Ude, B. Nemec, M. Rooker, F. Kirstein, and E. Mooij, "The AUTOWARE framework and requirements for the cognitive digital automation," in *18th IFIP WG 5.5 Working Conference on Virtual Enterprises (PRO-VE)*, 2017.

[2] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-end communication delay analysis in industrial wireless networks," *IEEE Transactions on Computers*, vol. 64, pp. 1361–1374, May 2015.

[3] P. Gaj, J. Jasperneite, and M. Felser, "Computer communication within industrial distributed environment - survey," *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 182–189, Feb 2013.

[4] S. Han, X. Zhu, A. K. Mok, D. Chen, and M. Nixon, "Reliable and real-time communication in industrial wireless mesh networks," in *2011 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 3–12, April 2011.

[5] "IEEE standard for local and metropolitan area networks–part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) amendment 1: MAC sublayer," *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–225, April 2012.

[6] D. Chen, M. Nixon, and A. Mok, *WirelessHART^{TM}*. Springer, 2010.

[7] K. Pister, P. Thubert, S. Dwars, and T. Phinney, "Industrial Routing Requirements in Low-Power and Lossy Networks," RFC 5673, Network Working Group, October 2009.

[8] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 2233–2243, Nov 2014.

[9] M. Nobre, I. Silva, and L. Guedes, "Routing and scheduling algorithms for WirelessHART networks: A survey," *Sensors*, vol. 15, p. 97039740, Apr 2015.

[10] E. Baccelli, M. Philipp, and M. Goyal, "The P2P-RPL routing protocol for IPv6 sensor networks: Testbed experiments," in *SoftCOM 2011, 19th International Conference on Software, Telecommunications and Computer Networks*, pp. 1–6, Sept 2011.

[11] P. Rodriguez and S. Sibal, "SPREAD: Scalable platform for reliable and efficient automated distribution," *Computer Networks*, vol. 33, no. 16, pp. 33 – 49, 2000.

[12] Y. Chen, R. H. Katz, and J. Kubiatowicz, "SCAN: A dynamic, scalable, and efficient content distribution network," in *Proceedings of the First International Conference on Pervasive Computing*, Pervasive '02, (London, UK, UK), pp. 282–296, Springer-Verlag, 2002.

[13] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the internet," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1282–1290 vol.3, Mar 1999.

[14] H. S. Bassali, K. M. Kamath, R. B. Hosamani, and L. Gao, "Hierarchy-aware algorithms for CDN proxy placement in the internet," *Computer Communications*, vol. 26, no. 3, pp. 251 – 263, 2003. Scalability and Traffic Control in IP Networks.

[15] K.-L. Wu, P. S. Yu, and J. L. Wolf, "Segmentation of multimedia streams for proxy caching," *IEEE Transactions on Multimedia*, vol. 6, pp. 770–780, Oct 2004.

[16] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys Tutorials*, vol. 16, pp. 1024–1049, Second 2014.

[17] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A survey of context data distribution for mobile ubiquitous systems," *ACM Comput. Surv.*, vol. 44, pp. 24:1–24:45, Sept. 2012.

[18] 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," RFC 6550, RFC Editor, March 2012.

[19] D. D. Guglielmo, S. Brienza, and G. Anastasi, "IEEE 802.15.4e: A survey," *Computer Communications*, vol. 88, pp. 1 – 24, 2016.

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

[21] K. Pahlavan and A. H. Levesque, *Wireless Information Networks*. Wiley, 1995.

[22] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, pp. 114–131, June 2003.

[23] Y. Huang and H. Garcia-Molina, "Publish/subscribe in a mobile environment," *Wirel. Netw.*, vol. 10, pp. 643–652, Nov. 2004.

[24] M. S. Daskin and K. L. Maass, "The p-median problem," in *Location Science*, pp. 21–45, Springer International Publishing, 2015.

[25] P. Gupta and P. R. Kumar, "Critical power for asymptotic connectivity in wireless networks," in *Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming*, pp. 547–566, Birkhäuser Boston, 1999.