Efficient Device-to-Device Service Invocation Using Arrowhead Orchestration

Cristina Paniagua, Jens Eliasson, and Jerker Delsing, Member, IEEE

Abstract—The Internet of Things (IoT) enables interaction from real-world physical objects using sensors to the virtual world of computers and the Internet. The use of service-oriented architecture (SOA) is one step in the creation of basic and complex interactions between several sensors and actuators. However, the use of SOA-enabled technologies alone does not meet all requirements of how sensor and actuator systems could be integrated to create distributed monitoring and control applications. The centralized, traditional method of communication in wireless sensor networks via a gateway presents drawbacks that have to be addressed; device-to-cloud communication adds higher latency and higher power consumption and is less robust than the device-to-device communication approach. Moreover, all these characteristics reduce the scalability of the network, thus limiting the use of IoT in the industry.

In this paper, the proposed method utilizes the Arrowhead framework orchestration system to generate service composition within a (wireless) network formed by IoT devices. The aim is to achieve efficient device-to-device service invocation to reduce the drawbacks of today’s widely used device-to-cloud approach. The method in this paper performs efficient service composition for industrial IoT, including mapping SOA service composition in very small resource-constrained devices using the Arrowhead orchestration. The results presented in this paper at the service level can increase performance and robustness in fog computing on resource-constrained devices.


I. INTRODUCTION

The concept of ubiquitous computing, where computational power, such as smart sensors, is embedded in the environment, was first introduced by Mark Weiser at Xerox PARC in 1993 [1]. Ubiquitous computing is currently usually referred to as the Internet of Things (IoT). Today, we are starting to see the realization of Weisers ideas, for example, in-home automation, Internet-connected district heating measurement stations and smart appliances. Sensor nodes, i.e., small embedded systems equipped with a combination of sensors and/or actuators, are the core building blocks of an Internet of Things network. These nodes, combined with gateways, middleware, and cloud computing, perform distributed data gathering and analysis. Individual nodes gather data, perform local processing, and transmit their results to other nodes in the network using IPv6 on top of wireless technologies such as Bluetooth, 6LoWPAN or Wi-Fi or wired solutions such as Ethernet and power-line communication (PLC). All nodes thus collaborate in a distributed fashion to address the tasks assigned to the network. Since a network may consist of a large number of nodes, individual nodes may break down without causing total network failure. Gateways are used to enable sensor nodes, which typically use low-power radios, to communicate with external networks, such as cellular networks and the Internet.

Organizations such as IEEE, IETF, IPSO, and ZigBee Alliance are currently developing standards for communication in the field of the Internet of Things. Many protocols, for example, 6LoWPAN over IEEE 802.15.4, Bluetooth Low Energy, PLC, and RPL are starting to become de facto standards for low-power and low-cost devices. The use of SOA-enabled protocols such as CoAP enables machine-to-machine communication and interaction between end-users and sensors and actuator platforms. Although CoAP addresses issues such as low-power access to resource-constrained devices and powerful scripting frameworks for service composition targeted CoAP (see, for example, [2]) have been proposed, there is still little work in frameworks designed for allowing users other than programmers and engineers deep access to sensor data. The Arrowhead framework addresses this by providing automatic device, system and service coverage, runtime service composition and security.

The typical structure in a wireless sensor network, based on device-to-cloud communication, presents several drawbacks, such as high latency, high power consumption, and low robustness, that limits their use in many automation applications. The proliferation of mobile devices with high computational power and the limitation of the constrained devices lead to a lack of research in the area of constrained networks. However, the use of device-to-device communication in conjunction with the Arrowhead orchestration open new possibilities for resource-constrained networks. The benefits of small device use in industry and smart environments include measurement and collection of more data with lower cost than larger devices and traditional wired networks, which enables more possibilities in the monitoring of processes or predictive management.

This paper compares traditional wireless sensor network (WSN) mesh structure communication with new service-oriented structures based on wireless device-to-device IP communication between nodes. Therefore, this paper proposes a method for efficient device-to-device service invocation using the Arrowhead framework and its orchestration capabilities. The focus of this paper is on the application/service level but does not investigate issues in the lower layers in the

C.Paniagua, J.Eliasson and J.Delsing were with the Department Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå, Sweden SE-97187 e-mail: cristina.paniagua@ltu.se.

Copyright (c) 2019 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to pubs-permissions@ieee.org.
communication stack, such as MAC, routing etc.

The proposal includes the benefits of using Arrowhead orchestration, such as easy coordination of services, dynamic discovery, late binding, reusability, and scalability, in conjunction with the device-to-device communication characteristics that promise to improve efficiency, latency, and scalability. This helps in the development and use of constrained devices despite their limitations in memory, computational power, energy, and bandwidth.

The method addresses issues related to latency in the orchestration of services. The proposed method changes the point of view from the more centralized topology where the information flow node to node to the gateway and back again through the nodes in the wireless mesh network to a more individual communication path directly between devices, thus significantly reducing the latency and network traffic load.

This paper is structured as follows: Section II presents related work and Section III presents the background. Section IV presents the proposed solution in detail. In Section V the characteristics of the communication are defined. Section VI describes the evaluation of the proposed method, and Section VII presents the results and conclusions. Finally, Section VIII provides suggestions for future work.

II. RELATED WORK

This section presents the related work that can be found in the current literature in relation to the IoT wireless sensor networks and device-to-device communication.

A. Internet of Things

The concept of the Internet of Things evolved from WSN technology. WSNs can be seen as the starting point for R&D of small, wireless sensor devices connected to each other and to other networks. One major drawback of the WSN approach is the typical use of proprietary and highly customized protocols and solutions to achieve optimal performance in terms of power consumption and low cost, which leads to poor interoperability since there is a multitude of different MAC protocols and radio standards, thus forcing developers to duplicate development efforts for each new application. However, when 6LoWPAN [3] was accepted as a standard for IPv6 over low-power mesh networks, software development started to rely on existing mechanisms from the TCP/IP protocol suite for addressing allocation, naming, and discovery. The use of IoT technologies within industrial applications is also starting to be accepted; however, the industry requires much longer system lifetimes compared to consumer products [4].

The overhead of using the IP protocol on resource-constrained devices was shown to be very low in many applications, as reported by Hui and Culler [5]. One of the foremost drivers of IP-based, Internet-connected embedded devices has been the IPSO Alliance, helping to standardize protocols for the Internet of Things.

Different research groups, such as the Thing-to-Thing Research Group (T2TRG) and the Internet Group on the Web of Things (WoT), have also been researching communication by resource-constrained devices with the global Internet. Datta et al. presented a method for device and metadata discovery plus a framework for device management [6]. In [7], Caminha et al. described a framework for discovery mechanisms for the Internet of Things.

B. Wireless Sensor Networks

WSN play an important role in the Internet of Things, where sensor and actuator nodes interact with the environment, collecting raw data or events [8]. The advances in microelectromechanical systems (MEMS), wireless communication and significant progress in hardware platforms, their operating systems and applications have enabled the development of low-cost, low-power and multifunctional WSNs. Traditionally, the idea of a sensor network was based on the collaborative effort of a large number of nodes. The nodes are usually dispersed in the sensor field, and each node has the capability to collect and route data back to the sink and end-users by multihop (mesh) communication.

WSN design is influenced by many factors, including fault tolerance, scalability, production cost, operating environment, topology, hardware constraints and power consumption. In recent decades, R&D has focused on improving all of these aspects by developing applications, designs and communication architectures [9], including specialized application domains such as healthcare in Alemdar et al. [10] or habitat monitoring in Mainwaring et al. [11].

Multihop (mesh) wireless sensor networks are commonly used in the frame of IoT. In a mesh network, a packet may have to traverse multiple consecutive wireless links to reach its destination [12]. In recent years, companies and researchers have been developing different architectures to address issues such as the quality of service or the latency in this type of network. One example is presented in [13].

C. Device-to-device communication

Device-to-device (D2D) communications have become an important and current topic considered in a broad range of applications [14]. Especially in reference to IoT devices and mobile networks [15]. The D2D communications standardization process started in 2015 by the 3GPP consortium and is still in progress.

D2D communication appears to enable key technology in new scenarios, such as mobile 5G networks [16][17], edge computing [18] and IoT [19]. Some of the paradigms where D2D communication is growing are fog computing [20] and multiaccess edge computing (MEC) [20].

Some examples of this are presented in the literature. Orsino et al. [21] claimed that D2D communications offer decisive benefits for future mobile 5G scenarios and proposed a forwarding scheme that shows that D2D communication guarantees a significant reduction in delay and traffic load across the network. In [22], Santos et al. presented a fog computing framework for the management and orchestration of smart city applications in 5G wireless networks.

In [23], the use of simultaneous wireless information and power transfer (SWIPT) was used to mitigate interferences between D2D and cellular communications.
[24], present an edge computing system for orchestrating end devices to execute services, maintain user privacy and keep data closer to its source.

As shown in several contributions, the wireless system requirements are presented in terms of high data rate, reliability and low latency. However, most of the current efforts are focused on mobile devices and cellular networks. Fewer efforts are concerned with small resource-constrained devices.

The main difficulties that face the resource-constrained IoT edge devices are its limitation in power, memory, processing resources, and bandwidth. The RFC 7228 [25], which defines the terminology for constrained-node networks, determines a constrained node as a node where some of the characteristics that are otherwise pretty much taken for granted for Internet nodes at the time of writing are not attainable. The current problems of the constrained devices in the Internet of Things are widely explained in [26].

In [27], Samie et al. noted the limitation of resources in low-power devices and proposed a technique for managing computation offloading in a local IoT network. In [28], an LWM2M implementation was presented in constrained IoT devices, addressing the end-node side instead of the server application point of view.

In [29], Duquennoy et al. demonstrated a novel approach for enabling QoS (Quality of Service) even on low-power and low-bandwidth wireless technologies such as IEEE 802.15.4. The proposed orchestra software-enabled scheduling of the duty cycle of wireless devices to guarantee certain performance characteristics, such as packets per second or maximum latency, which eliminates many issues with beacon-based mesh networks and allows distributed applications to execute with controlled communication properties. Other examples of time-slotted channel hopping (TSCH) are presented in [30] and [31].

III. BACKGROUND

A contextual overview of the Arrowhead framework in the field of IoT, process automation and SOA for embedded systems is introduced in this section.

A. The Arrowhead Framework

The Arrowhead framework is an SOA-based framework for supporting the creation of scalable cloud-based automation systems [32]. Its main concept is to achieve the interoperability between IoT devices and systems at the service level, thereby providing an exchange of information irrespective of underlying protocols and semantic solutions.

The Arrowhead framework makes use of the following entities [33]:

- Service. Used to interchange, consume or provide, information between systems.
- System. Piece of software that provides or consumes a service.
- Device. Piece of hardware, equipment or machine with computational and communication capabilities that hosts one or more systems.
- Local cloud. Self-contained network.

The extended definition of a local cloud is a self-contained network with the required functionalities to support IoT automation applications through the utilization of the three mandatory core systems and some optional support systems [34]. The mandatory core systems are the service registry, orchestration, and authorization. On top of the core system services, users can develop application-specific services that address some specific tasks, implementing the application-specific parts and reusing most of the other functionalities in the local cloud.

The functionality of the mandatory core systems provides the minimum advisable services to set a local automation cloud. The characteristics of each system are presented as follows [35]:

- Service registry system. The service registry system is in charge of registering and keeping track of the active services within the local cloud to enable their discovery.
- Orchestration system. The orchestration system is the central component of the framework and is responsible for providing service consumption patterns and controlling the interconnection of the systems.
- Authorization system. The authorization system is in charge of providing credentials to the systems to control and restrict access to the services. It is responsible for authentication and authorization.

The Arrowhead Framework differentiates from other industrial SOA-based frameworks in its wide device support from the small constraints resource devices to high and complex systems, providing required automation capabilities such as real-time control, security, and scalability.

B. Industrial IoT

The term Internet of Things (IoT) alludes to the interrelated computing devices, machines, and objects that have the ability to transfer data over a network without requiring human interaction. The application of the IoT concept to the industry was the beginning of the industrial Internet of Things (IIoT), and the development and integration of IIoT presented numerous challenges to the industry that must be addressed. Devices, sensors and smart machines add value to the traditional view of the industry and open a new paradigm in which operational efficiency, data monitoring, and collaborative automation are key concepts. Moreover, the industrial control systems that take part in the future industrial Internet of Things have stringent latency and reliability requirements [36], which represent significant challenges to existing wireless communication technologies.

C. CoAP

CoAP [37] is a protocol that stands for constrained application protocol, and it is specifically designed for resource-constrained embedded systems while still being compatible with such web technologies as URLs. CoAP requests can easily be translated into HTTP, which makes it suitable for obtaining sensor data from resource-constrained devices and forwarding the information using a CoAP-HTTP proxy. CoAP is quickly
becoming the de facto protocol for IoT systems since it was designed to be highly efficient, runs on top of UDP, is event-based and works exceptionally well with 6LoWPAN. CoAP has been implemented for both TinyOS and Contiki, and open source implementations are available for C and Java. There even exists a JavaScript implementation for Firefox. CoAP is specifically designed for low-power devices with low-bandwidth communication capabilities. Therefore, it is a good alternative for use as a communication protocol for the proposed framework. In [38], Eliasson et al. presented a data compression model and methods for highly distributed application on top of mesh networks, such as 6LoWPAN.

IV. EFFICIENT ARROWHEAD ORCHESTRATION FOR RESOURCE-CONSTRAINED DEVICES

Within the framework of industrial wireless networks, there are particular features, constraints, and requirements for resource-constrained devices. The quest for low latency, flexibility, mobility or reliable and efficient communications is being added to the traditional requirements as scalability, robustness or power efficiency [39].

A wireless sensor network is formed by several nodes grouped in a sensor field, and the data are forwarded via multiple hops relaying to a local sink or a gateway to communicate with other networks. In the [40], a number of examples with this common feature can be observed.

Fig. 1. Wireless sensor network mesh topology.

In the local cloud field, communication between nodes is usually achieved via the gateway. A sensor sends data through multiple hops to the gateway, and the gateway communicates the information to the local cloud. The processed data are sent back to the network through multiple hops until the packet is received by the destination node. The communication path depends on the topology and distribution of the network. In Fig. 1, the multihop device-to-cloud approach (blue), previously explained, is shown in contraposition to the device-to-device approach (orange).

Nevertheless, this traditional communication approach has some drawbacks that must be addressed. Device-to-cloud communication adds higher latency and power consumption and is less robust than device-to-device communication. These characteristics reduce the scalability of the network and limit the use of industrial IoT devices in the industry.

The solution presented in this paper draws on device-to-device communication to reduce the drawbacks of the device-to-cloud approach. In the Arrowhead framework, the orchestration system is utilized to dynamically allow the reuse of existing services and systems to create new services and functionalities [32]. The proposed method uses the Arrowhead orchestration system to generate service composition within a wireless network formed by IoT devices to achieve efficient device-to-device service invocation.

The experiment presented in this paper considers the latency as an indicator of the feasibility of the method. In industry, applications require low latency when working with important parameters, such as delivering control instructions, real-time information or monitoring machine status [41].

According to the time of response, we can distinguish four different requirements:

- Instantaneous response. Systems require an immediate response; latency requirements are below 1 ms. Wired closed control loops are normally part of this category.
- Fast response. System response below 100 ms. Safety mechanisms where a sensor triggers an almost instant response are an example of it.
- Medium response. Control processes require fulfilling the time requirements to avoid issues in the control and monitoring of the process. Latency requirements in these cases are typically less than 1000 ms.
- Slow response. Some physical magnitudes have a slow evolution, for example, temperature. In these cases, the time of response is not a critical requirement, which allows more flexibility in the control of these variables. A slow response is considered above seconds to even minutes.

The experiment is focused on fast response systems, considering that when stricter requirements have been achieved, the rest of the cases can be achieved, except for the instantaneous response systems whose latency is difficult to achieve in wireless networks and, consequently, wired configurations are the most widely used approach.

This method, in the frame of the industry, increases the process value through the addition of new sensors and monitoring processes. The proposed approach, compared with wired traditional systems, reduces the cost of adding new sensors and providing new useful information. The cost of wired devices increases the cost of the installations, which limits the amount used.

A. Communication methods

The two communication methods that are applied in the presented solution are device-to-device communication and device-to-cloud communication. Both methods are broadly used in numerous applications. However, even though both methods can be used as communication devices within a network, their characteristics and performance are different.

1) Device-to-cloud: Device-to-cloud communication is performed via an intermediate device in charge of routing the data from the nodes or devices in the network to the cloud for processing. The intermediate device that connects the net with the cloud depends on the type of network, commonly devices such as gateways and border routers. This communication
method is called cloud computing, in which the computing takes place in the World Wide Web, reducing maintenance and managing resource problems [42]. Regarding its use in WSN, communication between the nodes is commonly routed via the gateway to the cloud. Data are sent in multiple hops through the nodes. This methodology increases the latency time and reduces scalability. However, it does not require computations at the nodes because the data are processed in the cloud.

2) Device-to-device: Device-to-device communication is based on communication between peers. The main difference between device-to-cloud communication lies in the use of one-hop networks compared with the multihop networks used in the D2C. Consequently, the utilization of resources and the efficiency of the network increase. Current investigations, see Section II, are focused on mobile networks and 5G technology. Gandotra et al. [43] presented the advantages of device-to-device cellular networks, which include one-hop communication, spectrum reusability, optimization of power levels and improved coverage area. In the framework of the WSN, the communication directly between nodes, in conjunction with efficient service orchestration, improves the performance of the net, including a reduction in latency, an increase in scalability and an optimization of resources. However, utilizing this type of communication is limited by the computational power and memory of the nodes.

B. Arrowhead orchestration

The services orchestration in the Arrowhead framework is mainly managed by the orchestration core system. The orchestration system stores orchestration rules and resulting orchestration patterns [44].

The orchestration process starts when the consumer sends a service request in the form of a JSON object with the information about the requested service. The orchestrator uses that information to look up the service registry of a provider hosting that specific service. After that, it compares the matches with the authorization system to ensure the security of the network and finally sends the response to the consumer with the information and endpoint of the provider hosting the required service.

The use of the Arrowhead orchestration system carries multiple benefits, such as easy coordination, control, and deployment of the services, and the support of dynamic discovery and late binding [35]. These characteristics enable the use of long-term life cycle components that can be reused in different applications and future configurations. Moreover, the security aspects are included in the orchestration process, making the use of the network more robust.

The orchestration process to setup IoT service data exchanges is an initiation process. This initial process enables IoT service exchanges to be executed directly between a service provider and a service consumer with no involvement of the orchestration system, thus providing an autonomous and distributed D2D exchange of service between a producer and a consumer. The direct communication between both systems orchestrated by the Arrowhead Framework facilitates the scalability and reduces the latency in a wide spectrum of devices, including resource-constrained devices.

C. Configuration

To test the applicability and performance of the method, two different configurations of the WSN are considered. Pathway A represents the traditional WSN scenario, and pathway B represents the new IoT scenario.

- Pathway A, blue arrows in Fig. 2. The sensor node communicates with the computer sending a request via the border router. In the computer, the CoAP server responds to the sensor request by sending a request to the actuator. Pathway A represents a centralized model where the information has to travel through the nets to the gateway, and once the information is processed, back through nodes to the actuator closing the control loop. This model is often used in the configuration of WSN (see, for example, [45]).

- Pathway B, orange arrows in Fig. 2. The sensor node knows the endpoint of the actuator, and the communication is directly between the two nodes. Pathway B represents an IoT service-oriented model where the control loop is limited to the nodes implicated in the service. In the case of not previously knowing the actuator endpoint, the orchestration system provides the information needed to achieve the communication. This peer-to-peer interchange is used in modern service-oriented nets.

D. Experiment

The implementation of both pathways has been developed to test the applicability and performance of our solution with a simple experiment. The scenario includes services to switch two LEDs on and off when a push button is activated.

The devices used in the experiment are Mulle platforms. The Mulle platform is a miniature wireless embedded internet system (EIS) suitable for wireless sensors connected to the Internet of Things developed by Eistec AB. A sensor node is equipped with an ARM Cortex-M4 microcontroller with a 2 MB flash memory and an IEEE 802.15.4 transceiver (868 MHz radio frequency).
The components that formed the experiment, shown in Fig. 2, are:

- M1. Mulle device integrated with a push button. It portrays a sensor node in the test, generating the interrupt that starts the communication.
- M2. Mulle device integrated with two LEDs in two different pins. M2 provides a service to toggle the state of the LEDs. If the LED is switched on, the service changes the state to off, and vice versa. The LEDs are the actuators in the loop, and when the LED changes its state, the test is completed.
- M3. The Mulle device is used to measure the latency, and the device is connected with wires to M1 and M2. Device M3 measures the time from when the button is pressed in M1 to when the LED is toggled in M2. The connections between M3 and M1-M2 are auxiliary and only implemented for measurement purposes (continue line in Fig. 2).
- Border router. The Mulle device runs as a generic border router application from RIOT
- Computer with the orchestration system.

The implementation presented in this paper is currently supported by IPv6 over 6LoWPAN, and CoAP is used as a communication protocol. The operating system used on the nodes is RIOT, and they are programmed in C. The orchestration application was developed in Java using the CoAP libraries.

The experiment has two scenarios, one that corresponds to pathway B and another more simple scenario that corresponds to pathway A. In case A, the test starts when the button is activated. When the push button is pressed in M1, the interruption generates the request (PUT) to the computer. The system in the computer sends a request to M2 (PUT) that activates the LED.

In case B, the test starts when the button is activated. When the push button is pressed, the interruption generates a request (GET) to the orchestration system on the computer. The orchestration system stores the endpoints for each resource; depending on one parameter introduced in the command line, it is possible to choose between the white LED or the red LED. The system sends (POST) the corresponding endpoint. M1 stores the endpoint and uses it to communicate using the service that toggles the LED. The M2 answers with the response status code allowing the status to be checked.

Node M1 stores M2s endpoint to use in the following occasions. The next time that the button is pressed, communication is direct with M2, switching the LED, Fig. 4. Consequently, the communication to the orchestration system that supplies the endpoint is executed only the first time. For the test performed in the experiment, time is considered part of the communication transition and is not contemplated for the latency measures. Both paths are compared in Fig. 5, showing the differences between the two approaches.

To automate the test, to take enough samples to realize statistical calculation, node M3 generates a digital signal emulating the push of the button and registers the output where the LED is placed.

V. COMMUNICATION CHARACTERISTICS

The experiment is performed considering an ideal bidirectional point to point topology. The type of D2D communication utilizes the standard IEEE 802.15.4, and the discovery of the devices is controlled via the Arrowhead orchestrator. The issues concerning the signaling of other devices and other wireless communication issues are not part of this paper and are considered part of future experiments. The underlying communication is considered a medium to realize the experiments and is not the objective. A change in the radio frequency and technology use would affect transmission times.

A. Communication stack

The proposed method is primarily designed to be used with resource-constrained (wireless) sensor and actuator platforms. Therefore, the currently supported communication protocol for embedded devices is IPv6 over 6LoWPAN. CoAP is used as an efficient binary protocol with interesting properties for microcontrollers with a few tens of kB of RAM. Fig. 3 shows the current protocol stack compared with a traditional stack for web communication. The use of CoAP enables event-based communication through the observe feature, with built-in security by DTLS. Good interoperability with HTTP enables data to be sent from a sensor using CoAP to end-users using HTTP in a standard web browser.

B. Communication Setup

The hardware is set up with Mulle sensor nodes running the RIOT [46] operating system for IoT. The border router is set up using a single serial interface, ethos (Ethernet over serial) to multiplex serial data and UHCP (microhost configuration protocol) to configure the wireless interface prefix and routes. The border router is programmed utilizing the gnrc_border_router that RIOT OS provides.

The Mulle sensor nodes are equipped with an Atmel AT86RF212B Low Power 700/800/900 MHz transceiver, which works in 868 MHz of radio frequency. Using the only channel available in Europe, with a raw data rate of 20 kbps. In addition, Mulles uses the RIOT standard duty-cycled MAC layer.

Fig. 3. Communication stack.
Due to the slow speed of the radio communication, the transmission time is the dominant time in the experiments. The time required for decoding a frame is approximately 6 ms for the proposed experiments, compared with the 16 ms used for sending. These circumstances change when other radio communication technologies are used; however, the efficiency in radio communication is not an objective of the proposed experiments.

VI. EVALUATION

The experiment described in the previous section demonstrates that despite the foreseeable increase in the latency in communication, replacing cables with wireless communication is possible if it is orchestrated by service composition. Latency is below the time requirements explained in Section IV since latency is reduced when the communication is directly between nodes.

A. Measuring method

The test was performed under the same network conditions for both pathways, avoiding differences in the measurements due to the border router performance. Due to the different time bases of each device, the measures were taken by another external device to obtain the measures at the same time base. The samples were taken automatically by another Mulle node. The test began when the Mulle generated the signal to trigger the communication in M1 (sensor), which was considered the start timestamp. The stop timestamp was registered when the response was received, and the corresponding port in M2 (actuator) was set. Both timestamps were logged in a text file, and the difference of times was considered the latency in the corresponding path. In the case that a packet was lost and the timeout was exceeded, the loss was registered, and the test was restarted.

B. Results

1) Latency: The tests were carried out such that there were 250 iterations per path type. The results were averaged (Table I). Pathway B (device-to-device communication) performed better compared to pathway A (communication via the gateway). The reduction in the latency was, in these experiments, approximately 86%.

The ICMPv6 Echo between the two nodes was, as a result, 75 ms on average. Therefore, when the CoAP transmission was used, the latency time was close to the time response in the communication. This result shows that the overhead introduced by CoAP is very small.

The average latency for pathway A was 287 ms. This time corresponds to the communication delay between the node and the border router, the communication with the Java client application, processing of the data, and finally, the communication back downstream from the second node.

<table>
<thead>
<tr>
<th>Pathway</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>287</td>
<td>39</td>
</tr>
<tr>
<td>Dispersion</td>
<td>(280-302)</td>
<td>(36-45)</td>
</tr>
<tr>
<td>Std. Deviation</td>
<td>3.3</td>
<td>1.5</td>
</tr>
</tbody>
</table>
In addition, both approaches had a setup time to which all nodes were set. This time corresponded to the administration time, and it is different in each configuration. In the case of pathway B, this setup was completed dynamically in runtime. The nodes requested orchestration to the system and received the endpoint and details that were needed to stabilize the direct communication with other devices. In the experiment, the average latency, including the setup time, was 1,006 ms in pathway B. This result implies an administrative cost that has to be considered. The time was the same as the latency times in pathway A, with the difference being that approach B only required this time once in the configuration, not in each communication between nodes. The WSNs had setup times corresponding to the configuration of the network, and most of the cases were hardcoded. For this reason, we do not provide any measure of this time.

2) Losses: As explained in the previous section, a loss is considered when the actuator does not answer in a determined amount of time after starting the communication. If a package has been lost at some point in the communication, the counter of losses is increased, and the test starts again. To test the number of communication failures, more samples were taken to obtain an accurate result; the statistical sample was formed by 2,500 iterations of the test.

The results of the experiment show that in less than 0.5% of the cases, the communication between the sensor and the actuator using the proposed solution was lost. This result may be affected by the size of the network or an increment in the number of cases, and consequently, an increase in the density of packets in the network can be translated into an increase in the loss number. The reduced number of nodes in the example is insufficient to study this characteristic of the network. For this reason, packet losses are not part of the deeper analysis in this paper and will be considered in future work.

3) Analysis: As shown in Fig. 8, there are substantial gains that can be obtained by using a device-to-device approach instead of a device-to-cloud-to-device approach. The number of required transmitted packets can be greatly reduced, especially in dense and deep mesh networks. By the Arrowhead approach of deploying services even on resource-constrained devices, service compositions can be performed at runtime.

The implemented network is the smallest network that can be used for the realization of the test. Considering the different WSN topologies ((a) star, (b) tree and (c) mesh, Fig. 9), the minimal number of hops to complete the communication between two nodes are two: From one node to the gateway

Fig. 5. Sequence diagram of A and B path comparison.

Fig. 6. Histogram of the latency in pathway A (device-to-cloud).
and from the gateway to the end-node. This fact implies that in larger networks, the latency in the pathway will increase due to the number of nodes and consequently, the number of necessary hops. However, the latency in pathway B remains similar; only the administration time (setup) increased. Hence, the scalability is improved using device-to-device communication (case B).

**VII. Conclusion**

This paper presents a theoretical method for enabling distributed applications to run on both service-oriented networks and traditional wireless sensor networks. The proposed device-to-device communication reduces the latency compared to the centralized methods of wireless sensor network implementations, in which the communication between sensors and actuators is performed via multihops to the gateway. Moreover, a mapping of service composition in device-to-device invocation using the Arrowhead orchestration is also introduced.

With the use of the Arrowhead framework, the orchestration of a wide spectrum of devices, including small resource-constrained devices, is possible. The service composition of the services in conjunction with the device-to-device communication facilitates the scalability of the sensor and actuator networks, reducing the latency and the number of needed packets.

To test the concept, a set of experiments was presented, in which three Mulle nodes, using CoAP on top of 6LoWPAN and RIOT as the operating system, created a sensor network in two different scenarios. The first scenario was the most common centralized WSN topology, and the second was the proposed device-to-device service composition orchestration. In both scenarios, the latency and packet loss was measured, thus demonstrating that the reduction in latency was substantial. In the performed experiments, the gain of latency reduction was approximately 86%. More work is needed to address large mesh networks. However, preliminary theories suggest that the scalability in large networks will increase with the use of this new approach.

The aim of this paper is to show that the IoT service-oriented device-to-device approach in conjunction with the Arrowhead orchestration system can be used in wireless sensor and actuator networks, even on resource-constrained devices. The proposed approach addresses the latency requirements of systems with fast-medium response times.

This paper presented results that show that service composition, which is normally used on powerful computers and servers, can be used on wireless resource-constrained IoT devices. Furthermore, by avoiding device-to-cloud communication and instead using device-to-device communication, latency is reduced. The achieved latency depends on a number of factors [47], such as MAC layer performance, network size and topology, external electromagnetic noise. However, by using robust radio technologies and time-synchronous duty-cycling, it is possible to deploy distributed applications with soft real-time performance.

**VIII. Future Work**

The work presented in this paper provides a foundation for SOA-based distributed functionality by allowing loosely coupled services on resource-constrained devices to be interconnected at runtime. To further improve the possibilities for
interoperability and low-latency service compositions, it would be beneficial to investigate how service interface descriptions could be machine readable to allow seamless device-to-device selection. Furthermore, it would be interesting to investigate how Arrowhead-compliant systems can be exploited in large-scale fog computing applications. Other parameters, such as the energy cost, must be analyzed and compared in the different approaches and included in future experiments. Another important aspect would be to integrate information about the service compositions with a duty-cycle-based MAC protocol such as 6TiSCH. By integrating the proposed approach in this paper with the Orchestra solution presented in [29] or [30], it is possible to develop distributed applications with controlled latency using TSCH and Arrowhead even on IEEE 802.15.4. The integration of both technologies will be considered in future implementations.

Finally, security mechanisms are needed to allow the formation of distributed service compositions. The Arrowhead framework AAA solutions must therefore, be fully integrated to allow authentication, access control, and accounting. In addition, related service interface issues, not only service orchestration, must be investigated in future developments.

ACKNOWLEDGMENT

The authors would like to thank the European Commission, Artemis and the Arrowhead and Productive4.0, ECSEL JU project no. 737459, projects for funding.

REFERENCES


Cristina Paniagua Cristina Paniagua is a Ph. D. student in Industrial Electronics at Luleå University of Technology in Sweden and received her M.Sc. in Industrial Engineering 2017 and her B.Sc.in Electronic and Automation Engineering 2015 at Zaragoza University in Spain. Her current research interests include the Industrial Internet of Things focus on resource-constrained devices and the collaborative performance in heterogeneous Service-oriented Architecture (SOA).

Jens Eliasson Dr. Jens Eliasson is an Associate Professor in Industrial electronics, and researcher in the field of Industrial Internet of Things and received his M.Sc. in Computer Engineering 2003, and Ph.D. in Industrial Electronics 2008 at Luleå University of Technology in Sweden. He has since then been working with sensor networks and Internet of Things for industrial applications at the Embedded Internet Systems Laboratory (EISLAB). He is currently working with Service Oriented Architecture (SOA) for monitoring and control on deeply resource-constrained devices. In 2015, Dr. Eliasson and his team won the IPSO Challenge competition in Silicon Valley, USA.

Jesper Delsing Prof. Jerker Delsing received the M.Sc. in Engineering Physics at Lund Institute of Technology, Sweden 1982. In 1988 he received the PhD. degree in Electrical Measurement at the Lund University. In 1994 he was promoted to associate professor in Heat and Power Engineering at Lund University. Early 1995 he was appointed full professor in Industrial Electronics at Luleå University of Technology where he currently is the scientific head of EISLAB, http://www.ltu.se/eislab. His present research profile can be entitled IoT adn SoS Automation, with applications to automation in large and complex industry and society systems. Prof. Delsing and the EISLAB group have been a partner and coordinator of several large to very large EU projects in the field, e.g. Socrates, IMC-AESOP, Arrowhead, FAR-EDGE, Productive4.0 and Arrowhead Tools.