OPEN UNIVERSITY OF CATALONIA

DOCTORAL THESIS REPORT

Integrating secure mobile P2P systems and Wireless Sensor Networks

Author:
Marc Domingo Prieto

Supervisor:
Dr. Joan Arnedo Moreno
Dr. Xavier Vilajosana Guillen

A thesis submitted in fulfillment of the requirements
for the degree of Doctoral Programme on Network and Information Technologies

in the

IT, Multimedia and Telecommunications Department

November 15, 2016

Declaration of Authorship

I, Marc Domingo Prieto, declare that this thesis titled, "Integrating secure mobile P2P systems and Wireless Sensor Networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

"*Great things in business are never done by one person.*
They're done by a team of people."

Steven Paul "Steve" Jobs (1955-2011)

OPEN UNIVERSITY OF CATALONIA

Abstract

IT, Multimedia and Telecommunications Department

Doctoral Programme on Network and Information Technologies

Integrating secure mobile P2P systems and Wireless Sensor Networks

by Marc Domingo Prieto

Wireless Sensor Networks (WSNs) have become very important in the current networked society, since they can acquire and send useful and valuable information at low installation and maintenance costs when compared with their wired counterparts. WSNs enable the digitalization of new physical things, which have not been tracked until now. The benefits of WSNs are not only economical but also functional, as they allow easy and fast deployment as well as node mobility and dynamic network topologies, which greatly impacts temporary installations. This technology additionally enables the retrieval of a large amount of useful data coming from the environment. WSNs have traditionally been related to industries such as military, oil and gas; however, as their popularity has increased, they have become ubiquitous in many other fields and are already part of our daily lives. For example, in modern cities the concept of smart city is being realized through the deployment of sensors throughout the urban environment in order to monitor noise, the atmospheric pollution, garbage levels, parking occupancy, traffic flows and structural health.

WSN technology is still in its initial stages, with many challenges that need to be overcome on various levels. On the one hand, sensors have limited resources for collecting and sending data. On the other hand, the cloud platform must be able to process, store and spread a lot of information to many consumers in a secure fashion. This thesis focuses on the limitations existing at different levels. At a lower level, we concentrate on energy consumption while, at a higher level, we focus on the dissemination and security of information.

WSN motes are usually battery-powered and expend their energy with periodic triggers that perform a reading of one of their integrated sensors, with the information then being processed and sent over the radio. Since batteries have limited power, the motes have a limited lifetime. Additionally, the consumption of a mote is normally related to external triggers that can be different between motes on the same WSN, thus the lifetime of each mote ends up being different. Moreover, since motes are generally installed in inaccessible places, there is a significant cost incurred for each intervention to replace a mote or change its battery. Therefore, reducing the operation costs caused by the limited and unpredictable lifetime of motes is one of the greatest challenges to making this technology viable in new environments. In this thesis, we focus on reducing the consumption of an individual mote and extending the consumption of networks as a whole.

To reduce the consumption of an individual mote, we focus on the radio because it is one of the main components responsible for expending energy. In general, energy is consumed when sending application messages from sensor motes to the gateway, but also when performing additional message exchanges required by the network protocol. Many mechanisms exist, but few of them take into consideration the dynamic traffic patterns that are typical of WSNs. Therefore, we have proposed a Scheduling Function (SF) to be applied over the 6top sub-layer. In order to reduce the consumption in networks with dynamic traffic patterns, the proposal is based on the well-known Proportional-Integral-Derivative (PID) controller.

The second proposal analyses how to more equally distribute the consumption of all the motes in a network when the number of messages sent by each mote is different. We do so by creating messages with variable packet size. In this way, the interventions and therefore the maintenance costs of WSNs are reduced.

At a higher level, we take into account that a great amount of information must be distributed and processed by constrained resources and that, even though such information can be easily gathered locally, global access is often necessary. Therefore, we consider it convenient to have some middleware that integrates access to and configuration of sensor motes, making it easier to access from a desktop and mobile devices. Peer-to-peer (P2P) networks are an efficient method for distributing information in a self-organized manner. In terms of the connectivity of their devices, these networks also provide incredible benefits such as scalability and reliability. For these reasons, we consider that P2P systems are a perfect match in this scenario. In this thesis, we have proposed jxSensor, which is an integration layer between WSNs and the well-known JXTA P2P system. WSN sensor motes are treated as normal peers inside a network, taking advantage of P2P capabilities. Additionally, this integration has been

created while taking into account sensor mote limitations, thus ensuring that the sensor mote constraints are not accentuated with this integration.

Finally, the employed security mechanisms cannot be neglected, because real data coming from the environment is combined with P2P and thus new threads appear. One example is that a real subject can be identified and tracked. For these reasons, an anonymity layer was proposed in the P2P framework, and a light authentication mechanism was added to its mobile version, thereby taking into account the system's limitations and minimizing the overhead produced. Authentication is one of the first security requirements, and it ensures that no one injects false data from a WSN into the P2P system.

To sum up, this thesis addresses different limitations found in WSNs in order to enable their deployment in new scenarios as well as to make it easier to disseminate the gathered information.

List of Research Contributions

The novelty and impact of this thesis is supported by several contributions in journals and international conferences. In this section, we present the five most relevant publications. This doctoral candidate was the main author in all of these contributions, and they were published within a 4-year period before the publication of this thesis. These publications, which are indexed by either JCR or SCOPUS, include:

- Two journal papers indexed in ISI-JCR (Q2 and Q4).
- Two conference papers published in IEEE Computer Society proceedings, all of them indexed in SCOPUS.
- One article in a peer-review journal, not indexed.

These publications, sorted by relevance, are:

- M. Domingo-Prieto, T. Chang, X. Vilajosana, T. Watteyne, (2016). "Distributed PID-based Scheduling for 6TiSCH Networks". in IEEE Communication Letters. vol. 10, no. 5, pp. 1006-1009. doi: 10.1109/LCOMM.2016.2546880. IF: 1.291, Q2: 34/82. Category: TELECOMMUNICATIONS.

  This paper proposes a distributed scheduling mechanism for WSNs in order to efficiently deal with bursty traffic. The proposal is integrated into 6TiSCH networks and is based on the well-known Proportional Integral Derivative controller (PID).

- M. Domingo, J. Arnedo, (2012). "JXTAnonym: An anonymity layer for JXTA services messaging". in IEICE Transactions on Information and Systems. vol. E95-D, no. 1, pp. 169-176. issn: 0916-8532. IF: 0.218, Q4: 129/132. Category: COMPUTER SCIENCE, INFORMATION SYSTEMS.

  This paper defines an anonymity layer for JXTA services messaging. It is integrated in a transparent way, so any peer can use anonymous services without having to perform extra operations.

- M. Domingo-Prieto, J. Arnedo-Moreno, and X. Vilajosana-Guillén, (2012). "jxSensor: a sensor network integration layer for JXTA". in 2012 15th International Conference on Network-Based Information Systems (NBiS-2012), pp. 435–440. doi: 10.1109/NBiS.2012.125. IEEE Press. Indexed in SCOPUS.

  This paper shows how WSNs can be integrated with P2P networks. Specifically, jxSensor is designed as an integration layer between WSNs and the well-known JXTA P2P middleware. In this integration, WSN motes are treated as normal

peers, so any other peer can find their resources without knowing that they are actually sensor motes. This whole process is performed in a transparent way for sensor motes.

- M. Domingo-Prieto, B. Martinez, M. Monton, I. Vilajosana-Guillen, X. Vilajosana-Guillen, J. Arnedo-Moreno, (2014). "Balancing Power Consumption in IoT Devices by Using Variable Packet Size". in Complex, Intelligent and Software Intensive Systems(CISIS14), pp. 170-176. doi: 10.1109/CISIS.2014.25. IEEE Press. Indexed in SCOPUS.

  This paper presents a mechanism for reducing the number of interventions required in a WSN due to the lifetime of its motes. The proposed method balances mote power consumption by using variable packet size. Motes which send more messages have shorter codification than those that send more often.

- M. Domingo, J. Arnedo, J. Herrera, and J. Prieto, (2012). "Towards secure mobile P2P applications using JXME". in Journal of Internet Services and Information Security. vol. 2, issue. 1/2, Pág. 1-21. issn: 2182-2069

  This paper proposes a light authentication method for JXME-Proxied, the mobile version of JXTA. This mechanism uses hash-chains as the identifier of a Proxied Peer. With this enhancement, a peer cannot be impersonated.

The following are other contributions that are related to the topic of this thesis. Although the doctoral candidate is the first author, they are too old to be incorporated into the first group; and they are:

- M. Domingo-Prieto and J. Arnedo-Moreno, (2011). "Lightweight security for JXME-Proxied relay authentication", in The 14-th International Conference on Network-Based Information Systems (NBiS-2011). pp 104-111. doi: 10.1109/NBiS.2011.25. IEEE Press.

- M. Domingo-Prieto, J. Arnedo-Moreno, and J. Herrera-Joancomartí, (2010). "JXTA security in mobile constrained devices", in 6th International Symposium on Frontiers of Information Systems and Network Applications. pp. 139–144. doi: 10.1109/WAINA.2010.178 IEEE Press.

- M. Domingo-Prieto, J. Arnedo-Moreno, and J. Herrera-Joancomartí, (2010). "Security analysis of JXME-Proxyless version", in Actas de la XI Reunión Española sobre Criptologia y Seguridad de la Información (XI – RECSI). pp. 301–306. isbn: 978-84-693-3304-4. Publicacions URV.

Finally, other contributions that are related to the subject of this thesis should be mentioned. The doctoral candidate is not the first author, but they helped him carry out this thesis, and they are the following: [1, 2, 3, 4, 5].

Acknowledgements

The realization of this thesis would not have been possible without the support of the following people, to whom I will always be grateful.

First of all, I would like to thank my directors, Dr. Joan Arnedo and Dr. Xavi Vilajosana. Without their incommensurable and continuous support, I would not have finished this journey. They carried me to the finish line when I was exhausted by this race. I know I have not been the ideal PhD student, but I will always be in their debt. Continuing my university career while starting a new adventure in a startup company was difficult to combine, and I appreciate their patience.

Secondly, to my extended family. They never stopped supporting me through all these years. A special thanks to Jose, who never let me give up, and who periodically reminded me of the importance of finishing this project before losing momentum.

Thirdly, I would like to thank Worldsensing for opening their doors to me and allowing me to pursue an industrial PhD in there facilities. While in their company, I have learned that it is generally complicated to align "typical university" research with industrial research. This is very unfortunate. Nevertheless, I would like to thank the "fantastic four" for letting me be part of their dream. From the first moment, I was hooked not only on the project, but also on their enthusiasm, dedication and talent. The great team that we have built at Worldsensing will be difficult to replace.

Fourthly, I cannot forget the institutions that have supported me economically while carrying out this project: AGAUR and the Generalitat de Catalunya. I truly believe that transitioning from University to Industrial grants is the first step in conducting more applied research.

My sincere thanks also goes to the universities where I carried out some research stays. To the KISON group in the UOC, where I worked as a research assistant for many years. Everything began there. To the DEBI Institute at Curtin University in Australia, where I began sharing research experiences with Jaipal Singh and Pedram Radman. To the Telecommunication Networks Group at Technische Universität in Berlin, where I had a profitable summer sharing experiences with Dr. Vlado Handziski, among others.

Last but not least, I would like to thank Sara for enduring me throughout the painful day to day frustration. I know that I was not often the easiest person to deal. Thank you.

Contents

List of Figures

List of Tables

Als meus pares

Chapter 1

Introduction

This chapter presents the motivations for carrying out this thesis as well as its relevance. In Section 1.1, the problem face by this thesis is contextualized and introduced. Section 1.2 explains similar research found in the literature review of the two main areas covering this thesis. The objectives of this thesis are presented in Section 1.3. In Section 1.4 the research methodology used to conduct this thesis is shown. Finally, the structure of the rest of the document is found in Section 1.5.

1.1 Statement of the problem

In the current networked society, Wireless Sensor Networks (WSNs) have become very important because they can acquire and send useful, valuable information at low installation and maintenance costs, particulary when compared with their wired counterparts. WSNs enable the digitalization of new physical things which have not been tracked until now. The benefits of WSNs are not only economical but also functional, as they allow easy and fast deployment as well as node mobility and dynamic network topologies, which greatly impacts temporary installations. Additionally, this technology enables the retrieval of a large amount of useful data coming from the environment. WSNs have been traditionally related to industries such as military, oil and gas, but their increasing popularity has led to them becoming ubiquitous in many other fields and they are now already part of our daily lives [6].

For example, in modern cities the concept of smart city is being realized through the deployment of sensors throughout the urban environment in order to monitor noise, atmospheric pollution, garbage levels, parking occupancy, traffic flows and structural health. WSNs are also coming into the homes and monitoring environmental conditions such as temperature, humidity and light in order to improve comfort while minimizing power consumption. Additionally, WSNs are starting to become commonly

used for keeping track of everyday machines such as cars in order to minimize maintenance and increase security by monitoring, for instance, tire pressure. Furthermore, WSNs are used in health to monitor chronically ill patients or the elderly.

WSN technology is still in its initial stages, with many challenges that need to be overcome on various levels. On the one hand, sensors have to collect and send data with its limited resources. On the other hand, the cloud platform must be able to process, store and spread a lot of information to many consumers in a secure fashion. This thesis focuses on the limitations existing at different levels. At a lower level, we concentrate on energy consumption while, at a higher level, we focus on the dissemination and security of information.

Beginning at the low-level viewpoint, we take into account an individual WSN sensor, whose main functions are:

- Sensing the environment
- Processing the captured data
- Sending some data to the gateway

WSNs motes are usually battery-powered and expend their energy with periodic triggers that perform a reading of one of their integrated sensors, with the information then being processed and sent over radio. Since batteries have limited power, the motes have a limited lifetime. Additionally, the consumption of a mote is normally related to external triggers that can be different between motes on the same WSN, thus the lifetime of each mote ends up being different. Moreover, since motes are generally installed in inaccessible places, there is a significant cost incurred for each intervention to replace a mote or change its battery. Therefore, reducing the operation costs caused by the limited and unpredictable lifetime of motes is one of the greatest challenges to making this technology viable in new environments. This challenge can be faced in different ways:

- Reducing the consumption of an individual mote.
- Extending the lifetime of the network as a whole.
- Increasing battery capacity or harvesting energy from the environment.

In this thesis we are going to focus on the first two approaches: reducing the energy consumption of an individual mote and extending the lifetime of the network as a whole. The third approach is more of a physics problem related to the composition of both the battery and the harvesting sensor; thus, it is beyond the scope of this work.

In order to address the first approach toward reducing the consumption of an individual mote, it is necessary to evaluate how each mote wastes power. As stated in [7], the main sources of WSN battery drain are:

- Gathering data
- Processing data
- Sending data

There is a broad set of different types of sensors for sensing the environment, such as inclinometers and magnetometers. Their energy consumption results from the physics behind of the sensor. In the processing stage, complex calculations are performed in the minimum amount of time and with the lowest power consumption. In the data-transmitting stage, the main challenge lies in defining a protocol that keeps the mote connected to the network so it can send and receive data while using the minimum amount of energy. This is additionally complex since typical WSN applications require bursty traffic, meaning that the mote does not need to send messages during long periods of time, but instead the network must have enough capacity to transmit a bunch of messages when a trigger is pulled.

Sensor, processor and radio manufacturers are improving both performance and consumption every year, but their efforts must also be accompanied by the development of new protocols that take advantage of the new features and make WSNs viable in new scenarios. Currently, one of the main components responsible for expending energy is the radio. In general, energy is consumed when sending application messages from sensor motes to the gateway, but also when performing additional message exchanges required by the network protocol. In WSNs there are mainly two types of networks: those that are long range and low bit rate, and those that are short range and high bitrate.

On the one hand, the first type of network is called Low-Power Wide-Area Network (LPWAN), and it uses direct communication between the mote and the gateway. The most well known technologies of this type are LoRaWAN [8] and Sigfox [9]. These types of networks have no coordination problem, since a mote needs to send a message to only a single entity. The main issue is how to reduce the information sent, since battery consumption per byte sent is very high.

On the other hand, the second type of network communication is known as multi-hop, meaning that the packet sent by a mote can go through different motes until it reaches the base station or gateway. In these networks, each mote has two functions: send messages to neighbors, and receive messages from neighbors. For this reason

these networks tend to be synchronized. A typical example of these networks is the IEEE802.15.4e [10]. In this kind of network, messages are sent frequently between motes to maintain synchronization, so the main issue is not to reduce the amount of data sent, but to reduce the signaling information exchanged between motes in order to keep them synchronized and adapt the communication channels to the transmission necessities, thus minimizing its consumption. This last part has been gaining the attention of many researchers, since new protocols are appearing every day and opening new research areas with a lot of potentially new improvements.

In the second approach to extending the lifetime of the network as a whole in order to reduce the number of interventions, one idea is balancing the consumption between all motes in the network. Although it might seem a bit strange, each mote in a WSN generally has different consumption, because consumption is related to the number of external triggers each mote has. Imagine a sensor mote that sends a message every time a car goes over it, or a sensor mote that sends an alarm when a rock falls. Of course there are exceptions in which all motes send messages at regular, prefixed intervals of time. The clearest exception is a water meter that reports consumption in a fixed time period, independently of external events. But even in this scenario an alarm system can be built that could make sensor motes consume differently. By defining a protocol that can balance consumption between motes independently of the application, the lifetime of the whole network is going to be maximized, therefore leading to reductions in the number of interventions and its cost.

Until now, we have seen WSNs from a lower level perspective, considering the constraints of the sensor motes themselves. But, if seen from a higher level, new concerns appear. WSNs are able to provide detailed information about the environment in which they are placed. However, a great amount of information must be distributed and processed by constrained resources, and the fact that even though such information can be easily gathered locally, global access is often necessary. Therefore, it is convenient to have some middleware that integrates sensor mote access and configuration, thus making it easier to access from desktop and mobile devices.

Some companies offer a centralized cloud service with a web interface for storing data coming from WSNs, such as senseiot [11] and ThingSpeak [12]. Their cloud services can be used to create a private database and store the data coming from the sensors. Additionally, they provide some visualization tools for analyzing the sensor mote data as well as a trigger mechanism for performing various actions when pulled. However, they present some limitations:

- These solutions are cloud-based, so you have to rely on the service of an external company. In order to use them, the WSN must have an internet connection and

you cannot create your own cloud.

- In general, the information is private for each user, so it is difficult to get data from other WSNs.
- There is no standardized way to treat the sensor data. So, even if you could get data from different WSNs, a complex process is needed to unify all data.
- Two-way communication with sensors is limited. Although some downlink messages can be sent, they use private protocols. This complicates the remote configuration of sensors.

Other strategies for unifying a common interface exist. From below, IP has been pushed to the sensor motes to standardize its interface with 6LoWPAN [13]. On top of that, the Constrained Application Protocol (CoAP) [14] is defined, which enables motes to talk through a simplified HTTP protocol. In this way, motes can be accessed in a standard way while two-way communication with a mote is standardized. An example of a cloud service that uses this technology is thethings.io [15]. Different implementations exist for running your own CoAP server, such as CoAPthon [16]. Although CoAP can be executed with low overhead in some networks, the overhead is too high when adding extra headers to the LPWAN with a low bit rate. Additionally, most of the development is taking place at universities, and the industry has not yet decided if this strategy fits their needs. When you are using industrial sensors you want to maximize battery life to the extreme, and the overhead is unaffordable when using these protocols that were originally designed for computers and adapted to sensor motes. Furthermore, the data coming from sensors is still not standardized. However, some recent initiatives have emerged for defining an open standard, such as OGC SensorThings API [17], which builds on Web protocols and the OGC [18] Sensor Web Enablement standards, while also applying an easy-to-use REST-like style. But they yet remain neither a public nor a de facto standard. For these reasons an alternative is needed.

Instead of focusing on cloud services, another interesting approach – and the one chosen in this thesis – is to integrate WSNs with a P2P middleware. Peer-to-peer (P2P) networks are an efficient method for distributing information in a self-organized manner. In terms of the connectivity of their devices, these networks also provide incredible benefits, such as scalability and fault-tolerance. For these reasons, we consider that P2P systems might be a very good fit in some environments.

There are many P2P middlewares, but most of them are designed for a specific application. The most common use of P2P technology is for file sharing applications, where

many protocols can be found, such as bitTorrent [19] or gnutella [20]. There are others that implement a distributed hash table, such as Chimera [21] or Freepastry [22]. P2PNS [23] provides a secure distributed name service for P2PSIP. A toolkit for reliable messaging is proposed by jgroups [24]. Freedomjs [25] provides a framework for building P2P web applications. As far as we know, in the few P2P middlewares that are generic enough to implement any P2P application, the most popular are sip2peer [26] and JXTA [27].

Sip2peer is an open-source SIP-based middleware for the implementation of any peer-to-peer application or overlay without constraints on peer nature (traditional PC or mobile nodes) and specific architecture.

JXTA is an open source P2P protocol specification that is defined as a set of XML messages, which allows any device connected to a network to exchange messages and collaborate independently of the underlying network topology. JXTA has a long history. It was initiated by Sun Microsystems in 2001 and over the years has been the system of choice for many P2P-based applications, such as a clipboard and a file sharing between different computers [28] as well as a distributed e-learning system [29]. These applications can take advantage of the integration with WSNs. Additionally, JXTA has a limited version called JXME, which allows less powerful devices such as mobile phones to access the JXTA network.

Once WSNs are deployed massively, a large part of the real environment where we live is going to be digitalized. This means that a lot of sensitive information will have to be managed by an upper platform that collects and process data from many WSNs. While P2P provides help in distributing the information gathered by WSNs, one thing that cannot be neglected is its security baseline, since some potentially critical environmental information is being exposed to the internet. For instance, the data on your home power consumption can be used to predict when will be at home, which means thieves may be able to use that information against you. Therefore, security has to be taken into account when integrating WSNs to P2P.

On the one hand, adding security to the sensor motes is a great challenge, since they have limited computational resources to perform the complex mathematical operations required by most security algorithms. Additionally, because sensor motes are battery-powered, every extra operation performed to satisfy the security requirements means shortening their lifetime. On the other hand, the security requirements in the cloud are not only common end to end encryption. WSNs produce sensitive data, such as people's habits or their health status. Therefore, extra security mechanisms like anonymity are deemed essential for making this technology viable in certain businesses.

In summary, the goal of this thesis is twofold: first, to propose some algorithms that overcome the main limitation of WSNs, i.e., its battery life; second, to propose a secure integration layer between P2P and WSNs that allows dissemination of the information gathered by the sensor motes as well as a distributed configuration of them by means of desktop computers and mobile devices.

1.2 State of the art

This section provides a brief overview of the literature on the two main topics of this thesis:

- Critical limitations of WSNs that focuses on an efficient schedule mechanism for TSCH networks with bursty traffic while also balancing the energy spent on mote communication in a network.
- How to integrate WSNs with P2P, with a focus on the integration layer as well as security.

1.2.1 Critical limitations in WSNs

In this thesis we focus on lowering and balancing the power consumption of motes when transmitting data. Specifically, we focus on an efficient schedule mechanism for TSCH networks with bursty traffic while also balancing the energy spent on mote communication in a network.

Schedule in TSCH networks

Time-Slotted Channel Hopping (TSCH) networks, such as 802.15.4e [10], use synchronization combined with time-slotted medium access to enable collision-free communication. This slotted structure is also multiplexed in frequency so that it can scale up communications and improve reliability [30]. The IETF 6TiSCH working group is currently standardizing IP convergence and control plane management for these networks [31].

However, according to RFC7554 [10], a missing component in the TSCH architecture is an entity in charge of scheduling the TSCH cells for the nodes in the network. This entity is referred to as “Logical Link Control” (LLC), and it manages and controls the network schedule.

The IEEE802.15.4e standard defines the mechanisms for a TSCH node to communicate; 6TiSCH defines the basic configuration [32]. But no standard defines the policy for computing the communication schedule, for matching that schedule to the multi-hop paths maintained by the routing protocol, or for adapting the link-layer resources allocated between neighbor nodes to the data traffic flows.

Several approaches to scheduling TSCH networks have been proposed, though few of them are implemented in real hardware/software ecosystems. Palattella *et. al.* developed TASA [33], a centralized scheduler that calculates optimal schedules at the cost of intensive signaling. Tinka *et. al.* [34] developed a decentralized algorithm for scheduling the network with little requirement for peer-to-peer signaling between nodes. Morell *et. al.* [35], developed a hybrid approach with the idea of label-switching in TSCH networks, and they further propose the use of end-to-end path reservation signals to transport bandwidth requirements to the nodes. Allocation is done between parents and children recursively along the path. A similar approach was taken later by Accettura *et. al.* [36]. Recently, Duquennoy *et. al.* presented Orchestra [37], a best-effort decentralized approach to scheduling the network. Orchestra uses randomization to allocate slots without requiring node communication. While the approach yields 99.999% packet delivery, it does not address bursty traffic.

Therefore, as far as we know, there exists no decentralized scheduler for 802.15.4e networks that deals with bursty traffic while adding little overhead.

Balancing energy spent on communications

Recent years have seen a growth in research activities that focus on increasing the lifetime of IoT devices. Optimizing the battery life of multi-hop networks has been a hot topic, and it has usually been addressed through data aggregation by intermediate hops [38] or by compressing the packets sent [39]. Data is compressed or aggregated in each intermediate hop in order to reduce the amount of data sent by radio, which is the main source of battery drain. Traditional compression algorithms cannot be used directly because of their complexity. Therefore, simplified algorithms are used.

Another common approach is to use fare routing protocols [40]. These routing protocols not only avoid congestion and maintain connectivity of the network but also take into account that motes have limited energy. For this reason, when the routing protocol calculates optimal routes in the network, it also takes into account the energy left in each mote.

Another proposal addressed this problem by using a more efficient MAC protocol [41]. In the proposed MAC protocol, a time-out scheme is used instead of a fixed duty cycle. All messages are grouped and sent in bursts of variable length to increase sleeping between bursts. It decreases the power consumption in an environment with varying message rates.

However, few of the presented approaches consider the application itself and how the monitoring service is provided. Most monitoring applications cover large areas where different activity zones are observed. Common energy optimization approaches use the same policies independently of the observed level of activity in a specific area. Even though, from an individual device standpoint, battery lifetime may be extended using the described approaches, some devices will still deplete their power quicker than others.

1.2.2 Securely integrating WSNs with P2P

This section focuses on how to integrate WSNs with P2P. It is divided into two parts, one centered on the integration layer itself and the other on its security.

Integration layer

The chosen middleware for this integration is JXTA. The reason for choosing JXTA is because, at the point of beginning this research, it was the most commonly used P2P middleware due to being mature and interesting for its great functionalities such as Peer Groups, which is a mechanism for segmenting the global network into smaller groups of peers.

The latest stable revision of JXTA, version 2.7, became available in May 2011 and incorporated several long-awaited functionalities, of which the most relevant are related to security and simplifying local deployment for the testing stage. JXTA's development was stopped for a while due to licensing restrictions. Sun Microsystems began developing it as an Open Source project for a brief time under a custom license. Then, when Sun Microsystems was bought by Oracle, the community started to worry about its future because, in November of 2010, Oracle officially announced its withdrawal from the JXTA project. For this reason, the JXTA community asked to change the license to the Apache License 2.0 so they could continue developing the project by themselves and continue using the JXTA name. But Oracle would not make a decision about what to do with this project. So the development of JXTA was halted after the 2.7 release. In August of 2013, Kees Pieters published a series of articles on the DZone community

website about JXSE (the java implementation of the JXTA protocols) and Equinox (the engine on which Eclipse runs), which revived enthusiasm in the JXTA community. In October, it was proposed to start developing a new 2.8 release of the JXSE implementation of JXTA protocols using the name Chaupal [42] under the Apache v2.0 license. Its last alpha version is from June of 2015 [43].

There are not many proposals on P2P-WSN integration, but several of them are based on JXTA. One of the most thorough works may be found in [44]. The authors of this paper propose a quite complex general purpose architecture (although VANETs are the main scenario they have in mind) for sensor mote data acquisition over a JXTA middleware overlay operating on 2.5G/3G mobile devices. This is achieved by heavily modifying the standard JXTA distribution, thus creating an alternate version. It must be mentioned that the paper places much more emphasis on the mobile device integration aspect of the system. Nevertheless, from the perspective of WSN integration, the most interesting contribution is the design of the XMLSens protocol, which allows sensor motes to announce their characteristics to the WSN gateway by using XML structures in a manner that is similar to JXTA's lower layer protocols.

In [45], JXTA is again the middleware of choice for distributing sensor mote data, in this case in a healthcare services scenario. Sensor mote data from a Body Area Network are aggregated using a PDA, which then relies on a JXTA Relay Peer that acts as a proxy for accessing the P2P network. However, not much emphasis is placed on the WSN part of the system. What makes this proposal especially relevant in the context of this literature review is the fact that it is the only approach in which JXTA Peer Groups are fully considered as a natural method for segmenting the P2P network, both in terms of efficiency and security.

Sharesense, a P2P environment based on JXTA for monitoring multiple WSNs is presented in [46]. Its core concept is to rely on integrating JXTA into jWebDust, an external Java environment. This had been proposed previously by the same authors, as it allows developing and managing WSN-based applications. Being Java-based, this environment is easy to integrate with the Java implementation of JXTA. Within the system's architecture, it seems to be the one doing the heavy work, as far as sensor mote data management and access is concerned. Thus, the system can only integrate sensor mote networks based on this particular environment. Additionally, a Google Earth-based interface is included in the demo application, allowing precise location of each sensor mote.

Nevertheless, apart from the specifics of each proposal, some common features are shared by all of them to some degree. Sensor motes are always considered as resources to be shared, and two of JXTA's core services are used mostly to access their data. On

the one hand, JXTA's Discovery Service is used to publish and locate available sensor motes and WSNs, doing so by means of advertisements in the form of XML metadata documents in order to describe the sensor mote's characteristics. In that regard, each proposal provides its own custom-made advertisement structure. On the other hand, the WSN gateway acts as a single peer in the P2P substrate, relying on JXTA's Pipe Service to receive messages from other peers. From a design point of view, this is a sound and straightforward way to integrate the JXTA middleware into a set of WSNs.

Finally, it is important to note that all but one of the proposals either completely forget or use in a very rudimentary way another of JXTA's main architectural features, which is the one that actually sets it apart from other P2P middlewares: Peer Groups. We deem taking JXTA Peer Groups into consideration important since they allow peers with similar capabilities to create a context for peer operation, thus segmenting the P2P network and facilitating advertisement publication and retrieval.

Secure integration

As stated before, the P2P middleware chosen in this thesis is JXTA. Therefore, all the analysis and related work are going to be centered on its desktop and mobile versions.

The full specification of JXTA's capabilities and architectural design can be found in [27]. A very thorough security survey already exists in [47], so it makes no sense to repeat it here. In brief, JXTA currently provides an acceptable level of security, although this security is provided by using a very specific group membership model: PSE. This has the drawback of using a centralized authorization server. Also, the survey points out that no mechanism exists for securing messages, especially one that provides some degree of privacy.

Not many proposals exist in the literature for JXME. First of all, it is necessary to point out that JXME is divided into two different versions, named Proxied and Proxyless. On the one hand, the JXME-Proxied version is a very simple implementation for limited devices, which delegates all the heavy work to an external super-peer, the Relay Peer. On the other hand, the JXME-Proxyless version is a more complex one, where mobile peers may directly interact with the JXTA network.

As far as the Proxied version is concerned, Kawulok et al. [48] show a framework which allows wireless and remote peers to participate in a JXTA network. Authors describe the most interesting implementation details of the framework as well as all changes made in the JXTA core and JXME packages. The proposed framework adds a new authentication scheme based on certificates and PKI [49]. This authentication is

provided by the Relay Peer, which uses an external Sign and LDAP Server, breaking completely the P2P model proposed by JXTA.

In regards to the Proxyless version, it is a direct though somewhat simplified port of JXTA that supports the same protocols and architecture, and therefore proposals that apply to JXTA also apply to this version.

To sum up, JXTA has a good baseline security level, but it has a security flaw if the application requires securing the privacy of its messaging mechanism. JXME-Proxyless version maintains the same structure as JXTA and therefore can inherit its improvements, such as new security schemes. In contrast, the JXME-Proxied version uses a simplified protocol and cannot directly inherit security improvements from JXTA. Furthermore, to the best of our knowledge, there is only one authentication proposal that tries to increase the JXME-Proxied security baseline, although it is provided in a centralized manner.

1.3 Objectives

The aim of this thesis is to identify the synergies between WSNs and P2P protocols in order to propose an integration layer between both technologies as well as to improve some individual weaknesses found in each of them. To this end, the objectives of this thesis are listed below.

- O1 To propose a solution to some of the limitations found in WSNs
  - O1.1 To study and assess the limitations of current WSNs in scheduling mechanisms for TSCH networks with bursty traffic
  - O1.2 To design and implement a scheduling system for TSCH networks with bursty traffic
  - O1.3 To analyze the limitations of current WSN lifetime due to unpredicted energy consumption
  - O1.4 To design and implement a mechanism to increase the lifetime of a whole WSN
- O2 To propose and implement an integration layer between WSNs and P2P
  - O2.1 To develop and test a middleware that allows integration between WSNs and P2P
  - O2.2 To integrate the system securely with mobile devices

- O2.3 To propose and implement solutions that guarantee a security baseline

1.4 Research methodology

All the work presented in this thesis has followed the scientific method [50]. A simplified version of the process defined by this method can be seen in Figure 1.1, and is composed of the following steps:

- A question has to be posed in the context of existing knowledge.
- A hypothesis has to be formulated as a tentative answer.
- Predictions have to be made as well as consequences deduced.
- The hypothesis has to be tested in the new field. If minor contradictions are found, the hypothesis has to be adjusted. In case of finding major discrepancies, the hypothesis must be redefined.
- Hypothesis is accepted as provisionally true if no contradictions are found in previous steps.

The specific research methodology used for each contribution follows. This includes the specific data collection methods employed as well as the tools used for the data analysis.

1.4.1 Schedule in TSCH networks

Regarding TSCH network schedules, both theoretical and practical research methods were used. The former was used to choose a well known mechanism to reduce consumption in networks with bursty traffic, while the latter was employed to test the performance of the implemented algorithm and adjust it.

The data collection methods and tools utilized are:

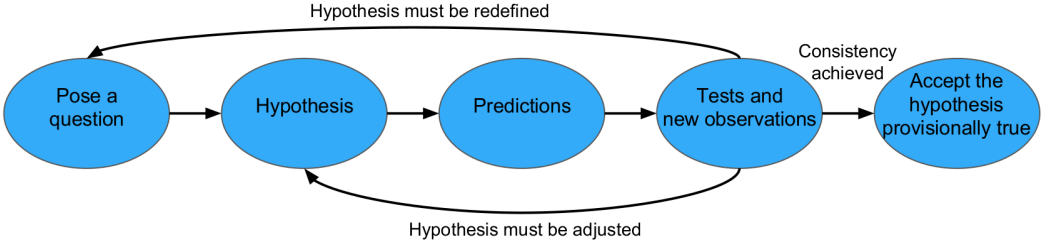


FIGURE 1.1: Diagram describing the scientific method

- A mathematical evaluation was carried out to determine whether the proposed solution would give the expected result.
- The OpenSim emulator of the OpenWSN project [51] was used to simulate a WSN and store logs with the behavior of the network.

Additionally, two tools were used for data analysis: pandas [52] and the jupyter notebook [53]. Pandas is a python data analysis library that provides a useful set of analysis tools that help evaluate the log files and extract results. The jupyter notebook is a tool that allows easy visualization of data, and it stores both the obtained results and the tests that were performed.

1.4.2 Balancing energy spent on communications

In terms of balancing energy spent on communications, theoretical research was first carried out to establish which mechanisms can be applied for distributing the consumption in a network. Second, practical research was used to test and tune the developed algorithms as well as to check their performance.

The data collection methods and tools used are:

- Information on parking events was generated by Fastprk sensors in a real Worldsensing installation, which was then extracted from their database by means of common tools provided by the database engine.
- A network simulator developed by Worldsensing was used to test the consumption of the sensor motes, depending on the algorithm used. Log files were obtained for the step by step consumption of each mote.
- A mathematical evaluation was carried out to determine whether the proposed solution would give the expected results.

To analyze the data, some tools provided by Worldsensing were used along with Matlab [54] scripts.

1.4.3 Integration layer

Concerning the integration layer, theoretical research was followed to determine how to perform the integration layer. The architectures of both JXTA and P2P were examined to determine the best integration. Additionally, other integration layers were

studied to learn from their designs. Therefore, this part involved most of the information being extracted by reviewing documentation and source code.

Finally, the practical methodology was also employed in order to build a proof of concept of this integration layer.

1.4.4 Secure integration

Regarding the secure integration, the first part consisted of theoretical research. This was used to find out the weaknesses of integrating JXTA with WSNs as well as which protocols could be used to mitigate these weaknesses.

The second part consisted of practical research. In this part, the algorithms were implemented and tested in real devices. A final step adjusted different parameters in each algorithm in order to obtain the best performance.

The data collection methods were mainly logs of the execution of the programs. Following that, spreadsheets were used to perform data mining.

1.5 Structure of the Thesis

The rest of this thesis is structured as follows. In Chapter 2, an overview of the main contributions is provided. Chapter 3 concludes this thesis by presenting the obtained achievements and describing lines that have been opened for future research. Finally, Appendixes A, B, C, D and E contain copies of the five main articles used in this contribution.

Chapter 2

Contributions of the Thesis

2.1 Distributed PID-based Scheduling for 6TiSCH Networks

M. Domingo-Prieto, T. Chang, X. Vilajosana, T. Watteyne, (2016). “Distributed PID-based Scheduling for 6TiSCH Networks”. in IEEE Communication Letters. vol. 10, no. 5, pp. 1006-1009. doi: 10.1109/LCOMM.2016.2546880. IF: 1.291, Q2: 34/82. Category: TELECOMMUNICATIONS.

The full text of this paper is available in Appendix A.

Abstract

Industrial low power networks are becoming the nexus of operational technologies and the Internet thanks to the standardization of networking layer interfaces. One of the main promoters of this shift is the IETF 6TiSCH WG, which addresses network management and IP integration of Time Synchronized Channel Hopping (TSCH) networks as those developed by the IEEE802.15.4 TG. The 6TiSCH WG is defining the operational interface and mechanism by which the network schedule can be distributed amongst the devices in the network. This operational sub-layer, called 6top, supports distributed scheduling and enables implementers to define the scheduling policy, only standardizing the distribution mechanism. This letter proposes a novel distributed scheduling policy based on the well-known industrial control paradigm referred as Proportional, Integral and Derivative (PID) control. The proposed technique is completely decentralized, enabling each node to determine the number of cells to schedule to one another, according to its traffic demand. The mechanism is reactive to sudden or bursty traffic patterns, while staying conservative in over-provisioning cells.

2.2 JXTAnonym: An anonymity layer for JXTA services messaging

M. Domingo, J. Arnedo, (2012). "JXTAnonym: An anonymity layer for JXTA services messaging". in IEICE Transactions on Information and Systems. vol. E95-D, no. 1, pp. 169-176. issn: 0916-8532. IF: 0.218, Q4: 129/132. Category: COMPUTER SCIENCE, INFORMATION SYSTEMS.

The full text of this paper is available in Appendix B.

Abstract

With the evolution of the P2P research field, new problems, such as those related with information security, have arisen. It is important to provide security mechanisms to P2P systems, since it has already become one of the key issues when evaluating them. However, even though many P2P systems have been adapted to provide a security baseline to their underlying applications, more advanced capabilities are becoming necessary. Specifically, privacy preservation and anonymity are deemed essential to make the information society sustainable. Unfortunately, sometimes, it may be difficult to attain anonymity unless it is included into the system's initial design. The JXTA open protocols specification is a good example of this kind of scenario. This work studies how to provide anonymity to JXTA's architecture in a feasible manner and proposes an extension which allows deployed services to process two-way messaging without disclosing the endpoints' identities to third parties.

2.3 jxSensor: a sensor network integration layer for JXTA

M. Domingo-Prieto, J. Arnedo-Moreno, and X. Vilajosana-Guillén, (2012). "jxSensor: a sensor network integration layer for JXTA". in 2012 15th International Conference on Network-Based Information Systems (NBiS-2012), pp. 435–440. doi: 10.1109/NBiS.2012.125. IEEE Press. Indexed in SCOPUS.

The full text of this paper is available in Appendix C.

Abstract

Nowadays, Wireless Sensor Networks (WSN) are already a very important data source to obtain data about the environment. Thus, they are key to the creation of Cyber-Physical Systems (CPS). Given the popularity of P2P middlewares as a means to efficiently process information and distribute services, being able to integrate them to WSN's is an interesting proposal. JXTA is a widely used P2P middleware that allows peers to easily exchange information, heavily relying on its main architectural highlight, the capability to organize peers with common interests into peer groups. However, right now, approaches to integrate WSN's to a JXTA network seldom take advantage of peer groups. For this reason, in this paper we present jxSensor, an integration layer for sensor motes which facilitates the deployment of CPS's under this architecture. This integration has been done taking into account JXTA's idiosyncrasies and proposing novel ideas, such as the Virtual Peer, a group of sensors that acts as a single entity within the peer group context

2.4 Balancing Power Consumption in IoT Devices by Using Variable Packet Size

M. Domingo-Prieto, B. Martinez, M. Monton, I. Vilajosana-Guillen, X. Vilajosana-Guillen, J. Arnedo-Moreno, (2014). "Balancing Power Consumption in IoT Devices by Using Variable Packet Size". in Complex, Intelligent and Software Intensive Systems(CISIS14), pp. 170-176. doi: 10.1109/CISIS.2014.25. IEEE Press. Indexed in SCOPUS.

The full text of this paper is available in Appendix D.

Abstract

Currently, IoT devices are becoming more and more popular, being deployed in different scenarios such as monitoring the power consumption of a house or the state of an outdoor parking spot. These networks tend to be densely populated by a huge amount of sensors that send really short messages. Given these characteristics, their main problem is the great variability and unpredictability of battery lifetime, when they cannot be plugged to a power outlet. In this paper, we analyze the mote behavior on a real IoT network and use the extracted data to propose a mechanism to distribute the power consumption more equally between all motes, regardless the number of messages each one sends. This new proposal decreases the numbers of interventions required to replace batteries, minimizing costs and increasing network lifetime.

2.5 Towards secure mobile P2P applications using JXME

M. Domingo, J. Arnedo, J. Herrera, and J. Prieto, (2012). “Towards secure mobile P2P applications using JXME“. in Journal of Internet Services and Information Security. vol. 2, issue. 1/2, Pág. 1-21. issn: 2182-2069

The full text of this paper is available in Appendix E.

Abstract

Mobile devices have become ubiquitous, allowing the integration of new information from a large range of devices. However, the development of new applications requires a powerful framework which simplifies their construction. JXME is the JXTA implementation for mobile devices using J2ME, its main value being its simplicity when creating peer-to-peer (P2P) applications on limited devices. On that regard, an issue that is becoming very important in the recent times is being able to provide a security baseline to such applications. This paper analyzes the current state of security in JXME and proposes a simple security mechanism in order to protect JXME applications against a broad range of vulnerabilities.

Chapter 3

Conclusions and future work

This chapter summarizes the achievements of this thesis by detailing how the originally proposed objectives have been accomplished. As a result of this research, various important contributions for improving WSNs have been proposed: at low levels with protocols that focus on energy consumption, and at high levels by designing an integration layer between WSNs and P2P to ease and secure information dissemination. Finally, the lines of research opened up by this thesis are outlined.

3.1 Thesis achievements

This section details how the results presented in the different contributions match the objectives of this thesis.

3.1.1 O1.1. To study and assess limitations of current WSNs in scheduling mechanisms for TSCH networks with bursty traffic

In [55], we explained one of the current limitations with WSNs, which is that most of them use internal proprietary protocols to handle the communication network, which means that there is no common means for taking the data gathered by the sensors and providing it to outside networks. In this direction, the IETF 6TiSCH Working Group is currently defining the architecture that enables the convergence of IPv6 and low-power industrial deterministic networks. At its core, there is a management layer called 6top [56], which is in charge of handling and managing the underlying MAC layer resources. 6top displays clear management interfaces and mechanisms so that distributed management entities can operate the network. Scheduling policies are left

open to implementers, enabling vendors to develop their optimal solutions while still being standards-compliant.

According to the 6TiSCH minimal configuration [32], any 6TiSCH node must provide one shared slot to bootstrap and advertise the network. Once a node has joined the network, it uses that slot to agree on a schedule with its neighbor. The 6top sublayer is in charge of handling slot reservation requests between neighbors, i.e., it installs the slots required by the schedule in its own node as well as its target neighbor. 6top contains an algorithm, called Scheduling Function (SF), which triggers when to add/delete one or more cells to a neighbor. The SF can pre-provision extra cells to cope with sudden bandwidth increases without loosing packets. Similarly, the SF can monitor the performance and utilization of each cell and possibly relocate the cell within the schedule in the case that it detects collisions happening in that cell. The SF uses local information to take decisions about the schedule. The problem is that the SF is a really critical part, because it affects the performance of the network as well as the consumption of its participants (i.e., the sensor motes), but there is no unique implementation that can fit all scenarios. Since this function is open to being customized for each application, there is a need to design an SF that fits the requirements of current applications.

3.1.2 O1.2. To design and implement a scheduling system for TSCH networks with bursty traffic

In order to overcome the limitations of WSN global connectivity and fill the gap that the IETF 6TiSCH WB has left open for development, we presented in [55] a distributed and efficient scheduling policy for networks with bursty traffic. This policy is based on the well-known industrial control paradigm referred to as Proportional, Integral and Derivative (PID) control, and it uses the 6top management layer being standardized at the IETF. We use this policy to drive the communication schedule in a TSCH network, doing so in a distributed manner. We demonstrate that with PID scheduling, a node in the network dynamically manages the TSCH schedule of its neighbors without requiring a central management entity or network-wide information. The mechanism is reactive to sudden or bursty traffic patterns while remaining conservative about over-provisioning cells.

A PID controller is a well-known control loop feedback mechanism used for stabilizing industrial control loops. Error minimization is performed by adjusting the process variables according to the current state and the desired end point. The further off-target the process variable is, the more aggressively the PID controller corrects it.

The objective of the mechanism is twofold: (1) keep a low (or zero) number of packets in the queue; (2) minimize the number of cells allocated in the node's schedule in order to reduce unnecessary energy consumption.

We implemented the PID scheduling controller in the OpenWSN project [51]. OpenWSN is an open-source implementation of a standards-based protocol stack, which includes IEEE802.15.4e TSCH, 6LoWPAN, RPL, CoAP and the latest IETF 6TiSCH standards.

In our experiments, we aimed to evaluate the performance of the PID scheduler under constant and bursty traffic and compared it to a well-known state of the art mechanism referred to as Orchestra [37].

In Fig. 3.1, the duty cycle of a network with a static schedule (Orchestra) is compared to the PID scheduler. In the experiment, we use bursts of 8 packets/slotframe during 3 slotframes every 50 slots for a three-node network. Three different cases are considered while following the Orchestra approach: Receiver oriented (Rx cells), Sender oriented (Tx cells) and Relay schedules (Both Tx and Rx cells). The static scheduler has been configured to deal with that burst by using a fixed capacity of 8 dedicated slots per slotframe according to the evaluated case (Tx, Rx or Relay). The figure presents the duty cycle per slotframe according to the node's role (Sender, Receiver or Relay). At the top, we see the overhead caused by the signaling of the PID, which causes a slightly higher duty cycle that tends to zero as the traffic demand decreases. In this case, the static schedule configuration slightly outperforms the PID scheduler on average, as transmission cells are not used if queues are empty; thus, the PID introduces an overhead to remove them from the schedule. However, as indicated in the middle and bottom figures, which show the receiver and relay oriented measurements, the PID removes Rx cells that are not used after a burst, thus saving the nodes to idle-listen in those slots. Averages are indicated by dotted lines for the three cases, showing that the PID approach outperforms static scheduling in terms of duty cycle when the network activity has a bursty nature.

We demonstrate that a PID schedule is able to cope with different types of traffic and react autonomously to sudden demand variations, doing so by targeting the stabilization and minimization of cells in the schedule and the queue. In most of the cases, this schedule outperforms the current state of the art mechanisms, which is going to increase the lifetime of motes in a WSN with bursty traffic.

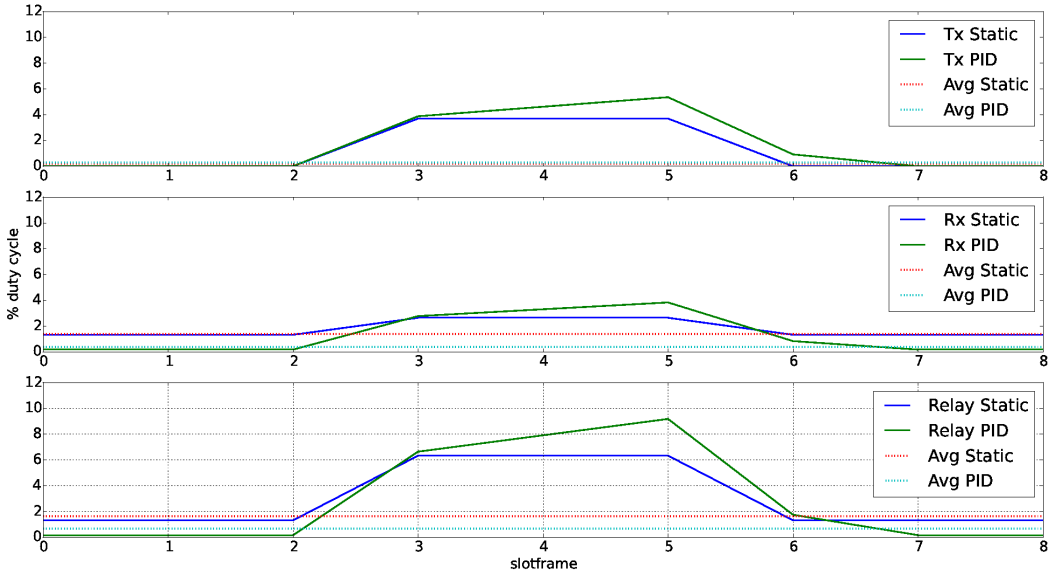


FIGURE 3.1: Duty cycle comparison between the transmission-oriented, sender-oriented and relay scenarios during a burst. Average traffic in dotted lines.

3.1.3 O1.3. To analyze current WSNs lifetime limitations resulting from unpredicted energy consumption

In [57], we analyzed the global battery consumption of a real IoT network deployment. The analysis revealed that not all motes behave in the same way inside a network. Some motes send messages more often than others, which results in having motes with different power consumption and battery lifetime.

Differences in battery usage across the same network become a major factor if we take into account that battery replacement interventions must be scheduled. In that regard, it must be taken into account that each intervention costs money and time. Furthermore, in some scenarios such as smart cities, it may actually become inconveniently bothersome to citizens. In scenarios with large differences in battery usage across devices, interventions become more complex because only some devices have their battery replaced and a log must be kept. The interventions must also be frequent, since the devices which were not replaced will now deplete their batteries sooner. Of course, it may be possible to just replace all batteries regardless of their power level at each intervention, but that would be wasteful from both a cost and environmental perspective. Therefore, in addition to increasing battery life, we consider that it is also very important to homogenize battery consumption across all devices in order to facilitate interventions and reduce maintenance costs.

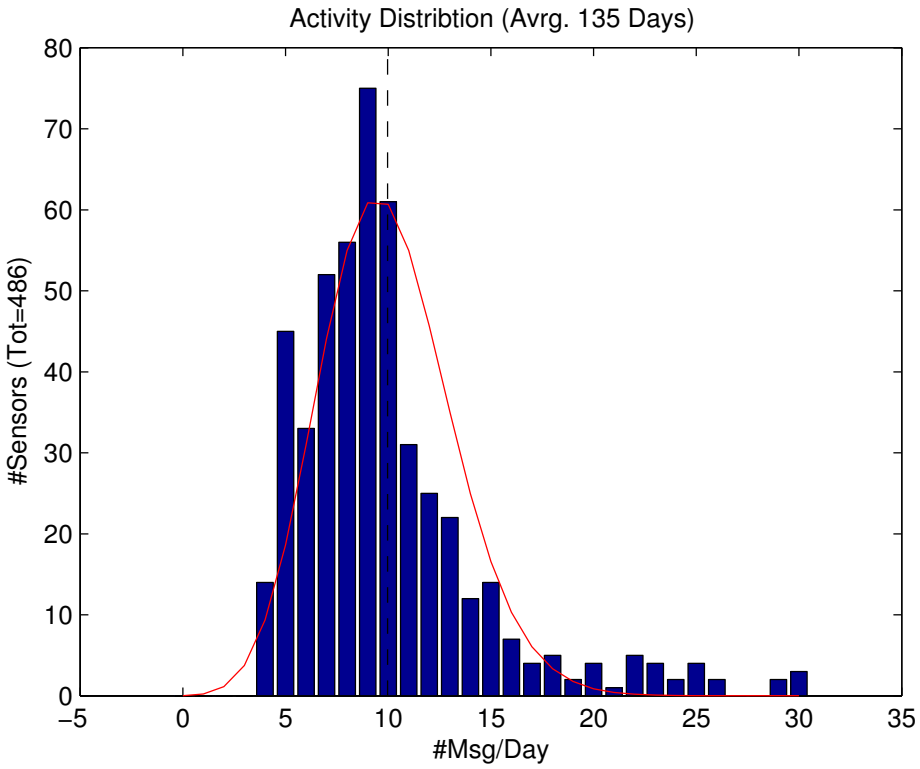


FIGURE 3.2: Number of messages, distribution per mote

IoT applications can be divided into two groups, depending on how messages are produced: one where motes report data periodically at fixed intervals; and another where motes report data asynchronously by reacting to certain triggers. In the latter group, the consumption of the motes (and, therefore, battery lifetime) depends on the number of state changes, which can be very different between motes installed at different spots.

In order to quantify this behavior in a real scenario, we collected four months' worth of real data from an installation of 500 outdoor parking sensor motes. The data was provided by one of Wordsensing's [58] company deployments in a real-life scenario that controlled the available parking spots on the streets. In Figure 3.2, which shows the average number of messages that motes send per day, it can be seen how most of the sensors send between 5 to 15 messages per day. However, some sensors send more packets, up to 30 packets per day.

From this data, it can be detected how different sensors in a real scenario send different numbers of messages, which will result in different battery lifetimes. In order to quantify how these different numbers of sent messages impact power consumption, a simulation was run using a power consumption model for WSN devices [59, 60]. This model was used to calculate Figure 3.3, which shows how battery level decreases as years pass, depending on the number of messages that a mote sends. The different

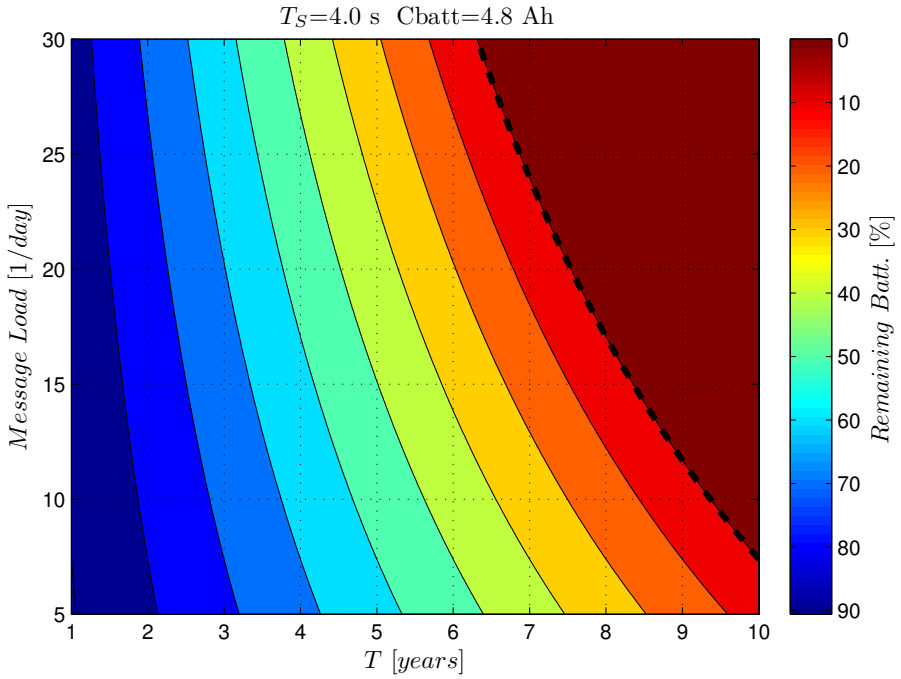


FIGURE 3.3: Mote battery duration, depending on the number of messages sent

ranges in this figure show the battery charge in 10% increments. More specifically, the left-most area represents remaining battery percentage that is between 100% and 90%, and the right-most area is the remaining battery percentage when the charge is between 10% and 0%, i.e., a flat battery.

Taking the maximum and minimum number of messages from Figure 3.2, it can be seen how the battery will last 10.5 years when sending 5 messages per day, while it will last only 6 years when sending 30 per day. This means that some interventions to replace batteries have to be scheduled 6 years after installation, and more interventions will be required over the next 4.5 years until the batteries of all motes are replaced.

Therefore, one of the main problems in WSNs is the great variability and unpredictability of battery lifetime.

3.1.4 O1.4. To design and implement a mechanisms for increasing the lifetime of the whole WSN

As stated before, one of the main challenges in WSNs is to increase the battery lifetime, for both the motes in particular and the whole network in general, in order to minimize the great variability and unpredictability of battery lifetime between motes. To overcome this last problem, we proposed evaluating two mechanisms in [57] in

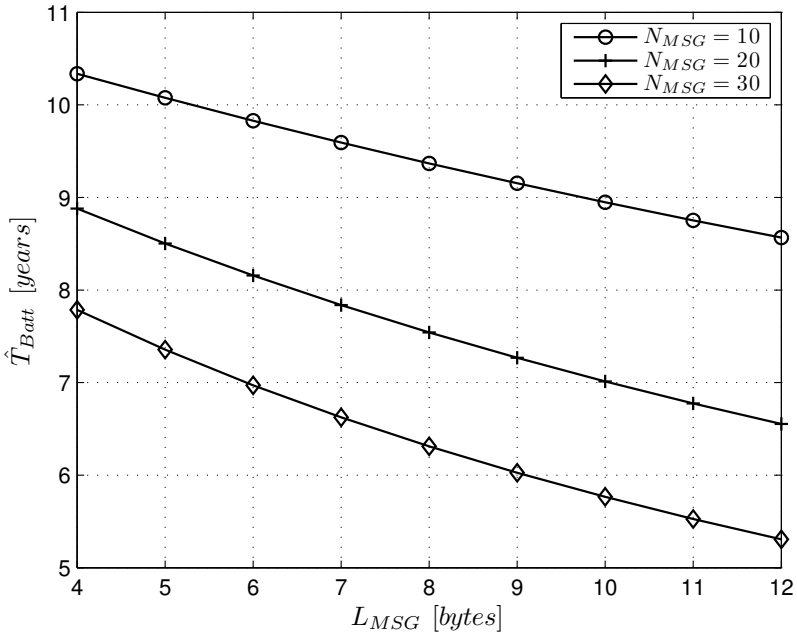


FIGURE 3.4: Mote battery duration, depending on the bytes sent

order to distribute the power consumption more equally between all motes, regardless of the number of messages each one sends. In that regard, we used datasets extracted from a real smart parking deployment in order to study two approaches: Huffman coding [61], which is very popular; and address clustering, which is used in other areas to improve the efficiency of systems.

Both proposals base their functionality on using variable packet size. Motes which are prone to send messages more frequently will use shorter packets in such a manner that the overall transmission time (and therefore battery depletion rate due to radio use) will be similar across the network. The potential impact of this idea is shown in Figure 3.4. Assuming N_{MSG} is 10, the mote will last more than 10 years when sending messages of 4 bytes, whereas it will last less than 9 if it sends messages of 12 bytes.

Huffman coding is an entropy encoding algorithm used for lossless data compression. It uses a variable-length code table for encoding source symbols. Huffman codes are established by storing the symbols of the alphabet in a binary tree according to their probability. As the tree is traversed from root to leaf, the code grows in length. Thus, symbols which occur frequently are stored near the root of the tree and have the shortest codes. While less frequently used symbols have longer codes.

We used a basic Huffman coder implementation to build a Huffman codification for our data set in order to demonstrate that a variable packet size is easily implementable. Figure 3.5 shows the lifetime of motes depending on the number of messages sent

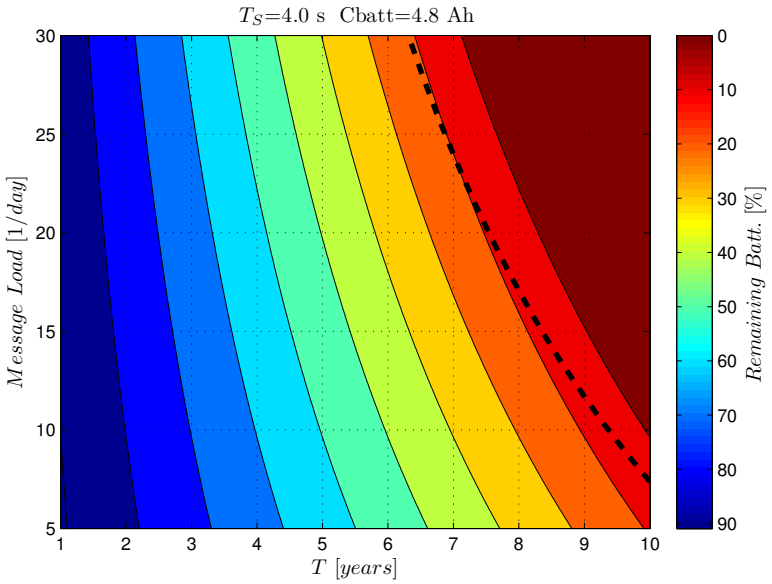


FIGURE 3.5: Mote battery duration for Huffman encoded messages, depending on the number of messages sent

when using Huffman encoded messages. The dotted black line is drawn for easy comparison with Figure 3.3, and it represents 0% of battery in that figure. The somewhat more vertical lines, which were theoretically expected, show that the power consumption of all the motes has been slightly balanced. Additionally, it can be seen how the 0% line has moved to the right, meaning that the power consumption of all motes has decreased. This effect is produced because Huffman coding not only performs variable encoding but also compression.

From this analysis, we can see that Huffman coding leads to increased lifetime of the network. However, from our observations, compression in this case rather than consumption balance seems to play a greater role in the improvements.

Another approach to variable packet size is address clustering. As can easily be seen in Figure 3.6, natural clustering is possible because nearby sensors have similar behavior. These clusters can be labeled as having “High”, “Std” or “Low” activity. In order to reduce the length of the packet, it is possible to perform variable-length source address encoding for the whole system. In this case, we can redefine the header in order to store the cluster number (using only 2 bits, i.e., up to 4 clusters), and we can make the header source address field variable, depending on the size of the cluster and the cluster to which each sensor belongs.

Figure 3.7 shows the lifetime of motes, depending on the number of messages sent when using clustering addresses. Again, the dotted black line is drawn for easy comparison with Figure 3.3, and it represents 0% of battery in that figure. This new figure

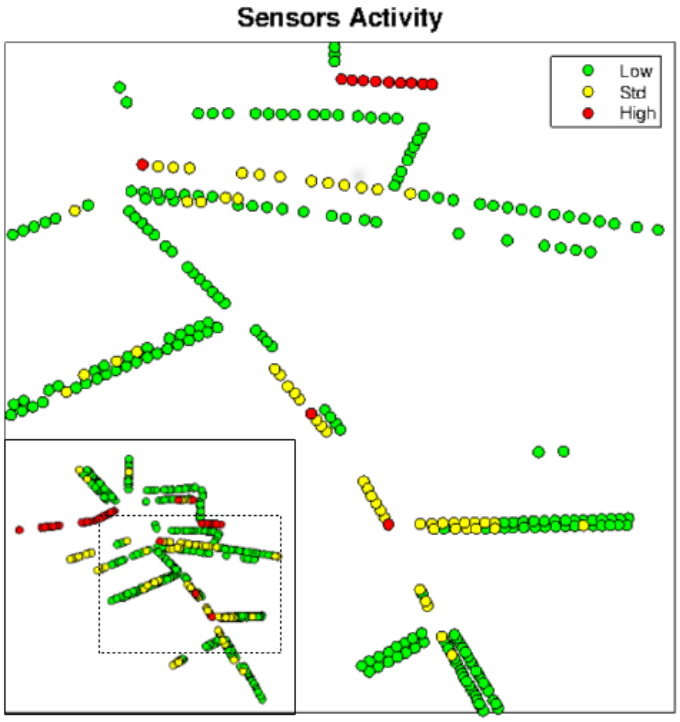


FIGURE 3.6: Heat map of messages sent

shows how there are three very differentiated sections, which match with the three kinds of clusters. The top group represents the “High” cluster, the middle represents the “Std” and the bottom the “Low”.

Additionally, the figure shows how all lines have a similar slope to the original but are shifted. The top cluster has moved to the right and the bottom cluster has moved to the left. After applying address clustering, motes that originally had a lifetime of between 6 to 7 years have moved to the 7- to 8-year range. In contrast, motes that were originally expected to last between 9 to 10 years now last between 8 to 9 years.

This assessment shows that the lifetime of most power hungry motes is similar in both cases. However, whereas address clustering shows better power balance, Huffman has better overall improvements. This is because, in addition to using variable size packets, Huffman coding also compresses data. However, the implementation of Huffman coding does not seem to be a trivial task in a real deployment, whereas address clustering is straightforward.

These results show that address clustering also improves the network lifetime by balancing the power consumption of motes. In comparison to the Huffman coding approach, address clustering is much easier to implement and provides similar benefits.

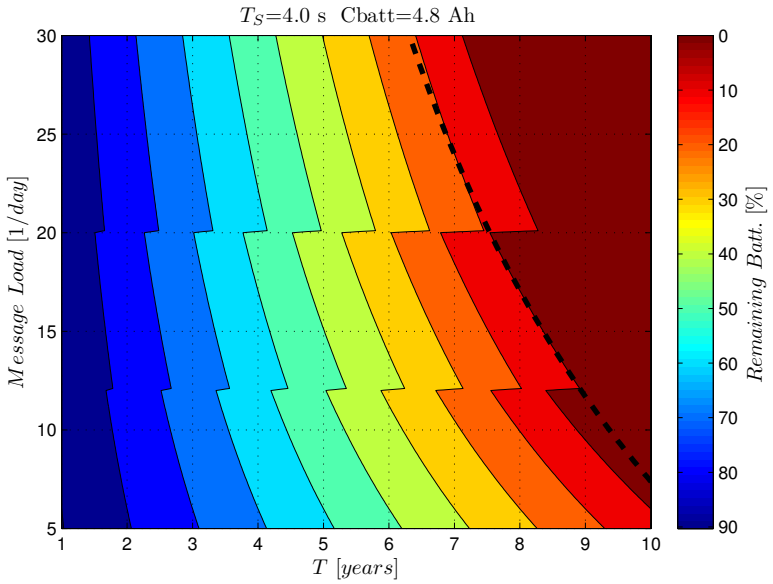


FIGURE 3.7: Mote battery duration for messages encoded using address clustering, depending on the number of messages sent

For these reasons, we conclude that using address clustering is the best choice for reducing the number of battery replacement interventions required, thus saving time and money.

3.1.5 O2.1. To develop and test a middleware that allows integration between WSNs and P2P

In [62], we presented jxSensor, an abstraction layer that allows JXTA peers to interact with WSNs. Through this integration, it is possible to provide a feasible solution to some of the WSN limitations in regards to data processing and dissemination. Our proposal uses novel approaches that set it apart from others, by taking into account JXTA's idiosyncrasies and proposing novel ideas such as *Virtual Peer*, a group of sensors that acts as a single entity within the peer group context.

First of all, each sensor mote or an aggregation of them can be treated as a single peer within the context of a peer group, allowing JXTA peers to communicate with them transparently while maintaining WSN gateway operation completely in the background. This allows sensor motes to use the versatility offered by JXTA Peer Groups to segment the network, such as enabling sensor motes to limit their presence to particular groups or allow different actions, depending on the groups. Most importantly, the membership of each sensor mote is not restricted by its geographical location, which

is the case in the only existing proposal where peer groups are even considered. Sensor motes connected to different gateways can share the same group and thus provide greater flexibility for JXTA network segmentation. In addition, our approach allows sensor motes to become members of several groups, not just a single one. Peer groups are also the main gateway for deploying security in JXTA.

Secondly, an abstraction layer is set in the gateway. This allows incorporating deployed sensor motes into the JXTA network without modifying its firmware.

Thirdly, two-way communication is permitted, allowing both the reading of data from the sensor motes and their configuration. Nevertheless, networks with one-way communication (sensor mote to gateway) are also supported, which is very useful in smart metering scenarios.

Fourth, the concept of Virtual Peer is created. This allows a group of sensor motes to be seen as a single peer in the JXTA network, which allows integrating a large quantity of sensors into the WSN without clogging up the JXTA network.

Finally, historic data is stored in the gateway in order to back up the information sent from the sensor motes and also to reduce readings by the sensor motes, thus optimizing resource utilization.

The general structure of the jxSensor network is shown at the logical and physical level in Figure 3.8. This figure serves as an overview of how jxSensor is integrated within the context of JXTA and WSN layers. The whole P2P-sensor integration layer is executed in a sensor gateway, which is a specialized hardware that acts as a bridge between the local WSN environment and external networks such as the internet. This aspect is imposed by the design of the WSNs. In that regard, on the one hand, applications which seek to obtain sensor data from the JXTA side of the architecture are executed by peers in the JXTA Applications layer. On the other hand, the sensor motes at the WSN side execute ordinary Sensor Apps that capture and send data while also receiving and executing actions. These actions include changing the configuration of the application or requesting data from a sensor. In both kinds of applications, their behavior is up to the developers and does not have to be specific to jxSensor.

The hardware gateway executes the main P2P-WSN integration module, **WSNGateway**, in the JXTA community services layer. As a result, it easily plugs into the standard distribution of JXTA 2.7 without the need to modify the source code, which would require the creation of a custom-made version of JXTA. Furthermore, it is not necessary to install a specific client component into the other peers before they can actually communicate with the sensor motes. Any JXTA Application may interact with the

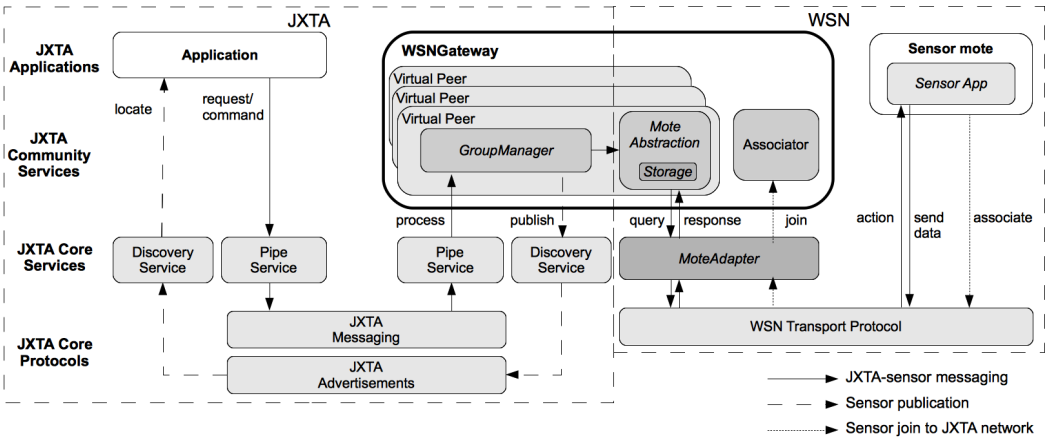


FIGURE 3.8: Overview of jxSensor's modular architecture

WSNGateway module by using the standard JXTA primitives for peer location and message exchange. The main component of this module is the Virtual Peer.

Virtual Peer acts as a peer entity within the JXTA network on behalf of one or a group of sensor motes, and it does so by storing and managing all the information an actual JXTA peer has (for example, a unique JXTA ID, advertisements, etc.). A single WSNGateway may support several Virtual Peers and, thus, several instances of this component may be executing concurrently at any given time, each one representing a separate sensor mote proxy. A set of motes may be managed by using a single Virtual Peer, which groups all of them together and may be interacted with a single entity. Thus, configuration commands are applied to all sensor motes at the same time, and the answer of a query can be resolved by any sensor mote in the group, which allows distribution of energy consumption.

At the lower level layer, the **MoteAdapter** service is a pluggable component which acts as the abstraction layer that translates queries to the actual protocols accepted by Sensor Apps deployed in the WSN. This is one of the most important modules, as it decouples the primitives processed by the MoteAbstraction component and the sensor motes. The implementation of the MoteAdapter service is specific for every WSN protocol and application used, making it highly flexible since all the heavy work falls on the gateway instead of each sensor mote.

To sum up, jxSensor was developed by providing an integration layer between WSNs and P2P. In this integration, sensor motes are treated as normal peers from the viewpoint of the other peers, but all their extra work is performed by the Relay Peer. This provides full capabilities to the integrated sensor motes without their having to spend extra energy.

3.1.6 O2.2. To integrate the system securely with mobile devices

JXTA has a mobile version called JXME. This version provides direct integration of mobile devices into the JXTA network. Therefore, in any system that uses JXTA, such as jxSensor [62], the integration layer we have developed between WSNs and JXTA can be used directly.

However, due to the restrictions of mobile devices, the JXME-Proxied version was developed using a special protocol between the mobile phone and its Relay Peer in the JXTA network. Therefore, this protocol has to be evaluated to ensure that it has a security baseline and there is no compromise of the information exchanged between mobile devices and the Relay Peer, as compromising this connection is going to compromise the whole system.

There were a few minor contributions trying to improve the security baseline of JXME-Proxied, but as far as we know there were not any that analyzed the current security provided. Therefore, in [63] we carried out a complete security analysis of JXME-Proxied. This assessment focuses on the communications between the Proxied and Relay Peer while also taking into account the way a Relay Peer stores and manages its subscribed Proxied Peers data. All communications between the Relay Peer and the rest of the JXTA network operate under the standard JXTA security model.

This analysis takes into account common attacks by categorizing them either as passive attacks, in which the attacker simply monitors peer activity and network traffic, or active attacks, in which the attacker purposely interferes with data or network activity.

Passive attacks which have been considered are Eavesdropping and Traffic analysis. The former consists of searching for sensitive information such as passwords in message exchanges, whereas the latter analyses traffic data looking for patterns and relevant peers.

Active attacks include Spoofing, Man-in-the-middle, Replay, Local data alteration and Software security flaws. Spoofing consists of impersonating another peer. Man-in-the-middle (MitM) intercepts the communications between two parties by transparently relaying forged messages to each one. Replay captures messages so that they can be reused at a later time to simulate a real message exchange initialization. Local data alteration modifies local data to corrupt system behavior. Finally, Software security flaws exploit vulnerabilities due to bugs in the source code by trying to obtain unexpected actions in the software.

Op./Threat	Evs	TAn	Spf	MitM	Rp	LDA	SFF
Startup	V(2)	N/A	V(4) P(TGMS)	V(2, 4) P(TGMS)	N/A	V(1)	P(OSS)
Join	V(2)	V(3)	V(4) P(TGMS)	V(2, 4) P(TGMS)	N/A	N/A	P(OSS)
Publish/ Discover	V(2)	V(3)	V(4)	V(2, 4)	V(4)	N/A	P(OSS)
Messaging	V(2)	V(3)	V(4)	V(2, 4)	V(4)	N/A	P(OSS)
Disconnect	V(2)	V(3)	V(4)	V(2, 4)	N/A	N/A	P(OSS)

N/A: Non-applicable.

TABLE 3.1: JXME-Proxied peer operation cycle security summary

The standard JXME-Proxied peer general operation cycle can be summarized in the following stages: Platform startup, Peer Group joining, Resource discovery and publication, Message exchange and Disconnection. The security analysis follows the actions performed by a Proxied Peer according to this lifecycle.

The analysis of possible attacks and the existing security mechanisms of JXME-Proxied, as classified by peer operations, provides a vulnerability map that is summarized in Table 3.1.

The four main vulnerabilities found are:

- **V(1)**: malicious executable code can easily be built and cannot be automatically discovered when installed
- **V(2)**: no encryption mechanism exists
- **V(3)**: no data flow masquerading mechanism exists
- **V(4)**: no real authentication is enforced

The available security mechanisms are:

- **P(OSS)**: Open Source Project
- **P(TGMS)**: Trusted Group Membership Service [64]

In analyzing the current JXME-Proxied security, it was found that the security baseline provided by JXME-Proxied was not enough to secure standard mobile applications. This is because the current version is vulnerable to a wide range of attacks. However, some attacks can be prevented by simple schemes that extend the basic protocols used in JXME-Proxied and thus add only a bounded complexity. In [63], we presented JXME-PLAuth (JXME-Proxied Light Authentication), a proposal to avoid Spoofing and Replay attacks. This proposal does not try to solve every single vulnerability which was identified, as this would require a much broader set of security mechanisms, but it

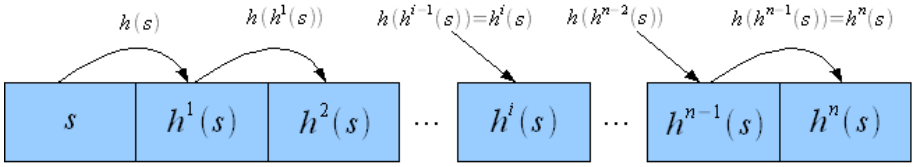


FIGURE 3.9: Hash-chain creation

provides initial protection to communication by guaranteeing lightweight authentication. This is the first step in providing a secure mobile framework for JXME-Proxied.

Protection against Spoofing and Replay attacks may be obtained by securely identifying the Proxied Peer using any well known authentication scheme. Our proposal relies on a lightweight scheme, since the constrained resources of devices where Proxied Peers are executed must be taken into account. For that reason, protection against Spoofing and Replay is obtained not by linking messages to a particular peer identity but by guaranteeing that, given a set of messages, all of them come from the same source peer, whichever that source might be. Thus, once an identifier is assigned, it can be guaranteed that no intruder is able to insert false messages by impersonating the source peer. To achieve such a goal, we propose using a hash-chain based scheme to create a set of linked values which will be used as local identifiers.

Therefore, we propose adding a security extension to the basic PeerId generation protocol at the platform startup stage in order to prevent attacks regarding authenticity at every stage in a peer's lifecycle.

Instead of obtaining a PeerId from the Relay Peer, Proxied Peers themselves generate a sufficiently long hash-chain (taking into account available resources), as can be seen in Figure 3.9. Starting with $h^n(s)$, each intermediate value of the hash-chain is used as its PeerId in each successive message exchange with its associated Relay Peer. In this way, the PeerId attached to a message changes for each successive message in such a manner that a potential hacker cannot predict it. However, as shown in Figure 3.10, the Relay Peer will be able to easily track identifier changes and recognize each message as originating from the same source. Internally, the Relay Peer stores a local translation table in order to translate local PeerIds to those of JXTA. Using a changing PeerId allows us to use exactly the same original protocol format, without any need for additional fields.

When a hash-chain is about to reach s or expire, a new one must be generated and its initial value refreshed at the Relay Peer in order for the new hash-chain values to be used. s will be used as the PUUID part of the PeerId in the refresh message. To allow this process, the set of operations that a Proxied Peer can perform by using HTTP-GET

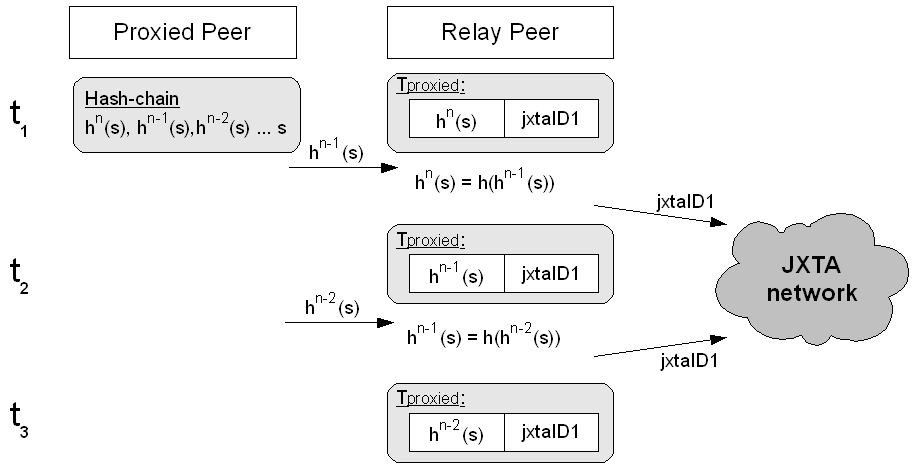


FIGURE 3.10: Hash-chain consumption and identifier translation

is extended with a renew command. This new parameter is used to announce a refresh in the hash-chain, which contains the new Id as its associated parameter.

As mentioned before, the proposed approach avoids Spoofing and Replay attacks. Furthermore, this proposal minimizes the modifications of the JXME-Proxied protocols, since no additional request type is defined. The renew command piggybacks on any other naturally occurring request, such as a listen request, in the same way as an additional parameter, and it will be processed along with the original request. Therefore, there is no need to send a single message for the sole purpose of transmitting hash-chain data, thus reducing overhead by taking advantage of existing transmissions. Additionally, as no refresh is needed, the secure scheme does not impact even the HTTP-GET protocol because the peer identifier field is used invisibly instead of with additional message element types.

Some real experiments were performed in order to test how the proposed security improvement to the JXME-Proxied framework impacts its overall behavior. The results showed that the overhead produced by utilizing security is generally low. Some overhead was expected to occur as a result of some security measures being used.

Therefore, a lightweight authentication scheme has been designed and implemented for JXME-Proxied, thus guaranteeing a security baseline for mobile applications that use JXTA. This ensures secure integration between mobile phones and jxSensor.

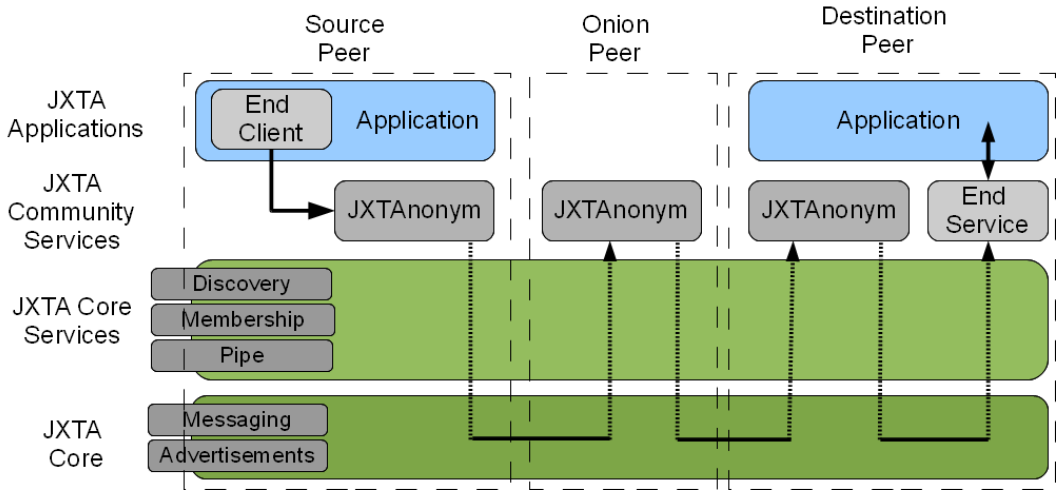


FIGURE 3.11: JXTAnonym service operation in the context of JXTA's architectural design.

3.1.7 O2.3. To propose and implement solutions that guarantee a security baseline

An extensive survey on the security of JXTA exists in [47]. One of its conclusions was that JXTA did not provide any mechanisms for anonymity. Privacy preservation and anonymity are deemed essential in maintaining a sustainable information society, but unfortunately it may sometimes be difficult to attain anonymity unless it is included in the system's initial design. When accessing real data from WSNs, anonymity is required in many use cases. For example, when providing the consumption of one's home to an external service, the consumer would not want to be identified because that information could be used to predict when they are not home.

It is for these reasons that in [65] we developed JXTAnonym, an anonymity layer for JXTA services messaging. Its general architecture is summarized in Figure 3.11.

The proposed layer is based on a popular approach within the context of P2P applications: onion routing [66]. However, it is specifically adapted to the idiosyncrasies of JXTA by taking advantage of service access mechanisms that are already provided by the platform rather than defining additional protocols. In addition, it minimizes the number of changes required on an existing system in order to integrate anonymous messaging. In that manner, peers which support anonymity may coexist with those who do not, without incompatibilities.

This new mechanism allows bidirectional message exchanges when accessing services using the JXTA protocols. JXTAnonym provides consumer and provider anonymity in any JXTA service access. The anonymity service has been implemented as a standard

JXTA service, and any JXTA Peer which belongs to a PSE Peer Group can use it. This restriction is necessary because a valid pair of public-private keys are required for implementing onion routing.

Apart from the fact that JXTA currently does not provide anonymous messaging by itself, the main contributions of the chosen approach are twofold. First, the tests performed on JXTAnonym demonstrate that anonymity is feasible in JXTA. Although the time spent on consuming a service increases when using JXTAnonym, this amount is not so high if we take into consideration the amount of connections which are performed and the fact that encryption is employed. Second, JXTAnonym is completely invisible to end services and clients in terms of processing the received data. Their internal operation stays the same whether anonymity is used or not.

These contributions are founded mainly on the fact that JXTAnonym is built over simple JXTA elements, such as Pipe Service, Discovery Service or advertisements. This allows using JXTAnonym without modifying the JXTA binary release, since no new JXTA protocols are defined. Moreover, Peer Group members can freely choose whether or not they want to belong to the anonymity set within the peer group. Obviously, the higher the number of peers who decide to use JXTAnonym, the higher the degree of anonymity.

To sum up, an anonymity mechanism has been defined for JXTA, thus allowing the sharing of sensitive data, such as that originating from WSNs, without the risk of being exposed.

3.2 Future work

This section presents the lines of research that remain open, specifically in regards to the different areas treated in this thesis.

In the area of extending WSN lifetime, our idea of variable packet size packets can be analyzed in other applications and tested in the field. Additionally, with the appearance of new LPWAN radios, new protocols and standards are also springing up. Our approach to improving the consumption of WSNs with radio burst patterns in 6TiSCH can be integrated into new protocols. Also, new protocols can be implemented to address other traffic patterns, which really depends on the application. Once the performance requirements are achieved, new challenges related to security will arise, since the data coming from sensors could be really sensitive. One issue that must arguably be analyzed is the question of ensuring anonymity. To achieve this end, an approach that is similar to our JXTAnonym proposal can be implemented in 6TiSCH networks.

In regard to the integration between a WSN and JXTA, the proposed solution called jxSensor can be implemented and tested in a real scenario. Thanks to the *MoteAdapter* service, already deployed installations can be easily integrated. This component can also be modified to use newer messaging protocols like MQTT or XMPP, which could be useful for sensors that implement them directly or to decouple the different pieces of jxSensor that run in the gateway. Other future plans are: a study on how to improve the association procedure that allows the usage of credentials and secure Peer Groups; the creation of an application to monitor sensor mote information from a mobile phone using JXME; and finding ways to replicate historic data between different gateways in order to improve its availability.

Some further research can be conducted on JXTA security. The anonymity layer proposed for JXTA can be extended to publishing and retrieving Advertisements, as well as to providing anonymous multicast communications. Also, it is worth studying how to apply mechanisms that thwart global attackers. In the JXME security proposal, additional mechanisms can be provided to secure against other attacks rather than Spoofing and Replay. However, the task is expected to be difficult, since the obtained solutions must be consistent with the constrained computational environment of a Proxied Peer device. Another possibility is to use the proposed security scheme in new protocols.

Finally, in the field of integration between P2P and WSNs, there is great potential in continuing to explore new ideas. Once WSNs are deployed in new scenarios, more different data will have to be handled efficiently by a distributed system.

Appendix A

Paper: Distributed PID-based Scheduling for 6TiSCH Networks

M. Domingo-Prieto, T. Chang, X. Vilajosana and T. Watteyne, "**Distributed PID-Based Scheduling for 6TiSCH Networks**," in IEEE Communications Letters, vol. 20, no. 5, pp. 1006-1009, May 2016. doi: 10.1109/LCOMM.2016.2546880. IF: 1.291, Q2: 34/82. Category: TELECOMMUNICATIONS.

Distributed PID-based Scheduling for 6TiSCH Networks

Marc Domingo-Prieto, Tengfei Chang, Xavier Vilajosana *Senior Member, IEEE*, Thomas Watteyne *Senior Member, IEEE*

***Abstract*—Industrial low power networks are becoming the nexus of operational technologies and the Internet thanks to the standardization of networking layer interfaces. One of the main promoters of this shift is the IETF 6TiSCH WG, which addresses network management and IP integration of Time Synchronized Channel Hopping (TSCH) networks as those developed by the IEEE802.15.4 TG. The 6TiSCH WG is defining the operational interface and mechanism by which the network schedule can be distributed amongst the devices in the network. This operational sub-layer, called 6top, supports distributed scheduling and enables implementers to define the scheduling policy, only standardizing the distribution mechanism. This letter proposes a novel distributed scheduling policy based on the well-known industrial control paradigm referred as Proportional, Integral and Derivative (PID) control. The proposed technique is completely decentralized, enabling each node to determine the number of cells to schedule to one another, according to its traffic demand. The mechanism is reactive to sudden or bursty traffic patterns, while staying conservative in over-provisioning cells.**

***Index Terms*—IEEE802.15.4e, 6TiSCH, Wireless Sensor Networks, Distributed Network Scheduling, Industrial IoT, TSCH.**

I. Introduction

The IETF 6TiSCH Working Group is defining the architecture that enables the convergence of IPv6 and low-power industrial deterministic networks. At its core is a management layer called 6top which is in charge of handling and managing the underlying MAC layer resources. Fig. 1 illustrates a Time Synchronized Channel Hopping (TSCH) network structure. 6top exposes clear management interfaces and mechanisms so distributed management entities can operate the network. Scheduling policies are left open to implementers, enabling vendors to develop their optimal solutions, while still being standards-compliant.

This letter presents a distributed and efficient scheduling policy. This policy is based on the well-known Proportional, Integral and Derivative (PID) control algorithm, which we use to drive the communication schedule in a TSCH network, in a distributed manner. We demonstrate that with PID scheduling, a node in the network dynamically manages its TSCH schedule to its neighbors, without requiring a central management entity or network-wide information.

This letter is organized as follow. Section II relates our work to the current state of the art, and introduces the scheduling operation in TSCH/6TiSCH networks. Section III introduces the concept of PID scheduling and its integration to OpenWSN. Section IV presents the implementation and evaluation details. Section V concludes this letter.

M. Domingo and X. Vilajosana are with Worldsensing and Universitat Oberta de Catalunya, Barcelona, Spain.

T. Watteyne is with Inria-Paris, EVA Team, France.

T. Chang is with both Inria-Paris, EVA Team, France and University of Science and Technology, Beijing, China.

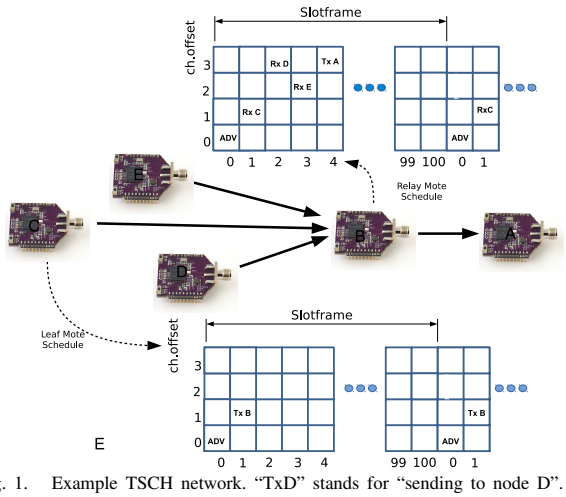


Fig. 1. Example TSCH network. "TxD" stands for "sending to node D". "RxD" stands for "receiving from node D". Cells marked "ADV" are used to advertise the presence of the network [1].

II. Scheduling in TSCH networks

A. Related Work

TSCH networks use synchronization combined with time slotted medium access to enable collision-free communication. This slotted structure is also multiplexed in frequency to be able to scale up communications and to improve reliability [2]. The IETF 6TiSCH working group is standardizing IP convergence and control plane management for these networks [3].

According to RFC7554 [1], a missing component in the TSCH architecture is an entity in charge of scheduling the TSCH cells for the nodes in the network. This entity is referred to as "Logical Link Control" (LLC), and manages and controls the schedule of the network.

The IEEE802.15.4e standard defines the mechanisms for a TSCH node to communicate; 6TiSCH defines the basic configuration [4]. No standard defines the policy to compute the communication schedule, match that schedule to the multi-hop paths maintained by the routing protocol or adapt the link-layer resources allocated between neighbor nodes to the data traffic flows.

Several approaches have been proposed to schedule TSCH networks, few of them being implemented in real hardware/software ecosystems. Palattella *et. al.* developed TASA [5], a centralized scheduler that calculates optimal schedules at the cost of intensive signaling. Tinka *et. al.* [6] developed a decentralized algorithm to schedule the network, while requiring little peer-to-peer signaling between nodes. Morell *et. al.* [7], developed an hybrid approach with the idea of label-switching in TSCH networks which proposes the use of end-to-end path reservation signals to transport

bandwidth requirements to the nodes. Allocation is done between parents and children recursively along the path. A similar approach was taken later by Accettura *et. al.* [8]. Recently, Duquennoy *et. al.* presented Orchestra [9], a best-effort decentralized approach to schedule the network. Orchestra uses randomization to allocate slots without requiring node communication. While the approach yields 99.999% packet delivery, it does not address bursty traffic as is shown later.

B. Network Scheduling in 6TiSCH Networks

According to the 6TiSCH minimal configuration [4], any 6TiSCH node must provide one shared slot to bootstrap and advertise the network. Once a node has joined the network, it uses that slot to agree on a schedule with its neighbor. The 6top sublayer is in charge of handling slot reservation requests between neighbors, i.e. install the required slots in the node and its target neighbor schedule. 6top contains an algorithm, called Scheduling Function (SF), which triggers when to add/delete one or more cells to a neighbor. The SF can pre-provision extra cells to cope with sudden bandwidth increases without loosing packets. Similarly, the SF can monitor the performance and utilization of each cell and possibly relocate the cell within the schedule in case it detects collisions are happening on that cell. The SF uses local information to take decisions about the schedule. The PID-based SF we propose can be plugged directly into the 6top sublayer.

III. PID SCHEDULING

A PID controller is a well-known control loop feedback mechanism used for stabilizing industrial control loops. The error minimization is done by adjusting the process variables according to the current state and the desired end point. The further off-target the process variable is, the more aggressively the PID controller corrects it.

A PID controller is generally defined as in (1), where e_t stands for the error between the current state and the desired objective. e_t is multiplied by a K_p constant which is used to fine-tune the process. The integral term of the PID, controlled by K_i, is used to take into account the error evolution along time (based on past values) and introduce certain inertia to the system. This prevents the control to be reacting proportionally to the immediate error and to maintain the correction trend according to the previous iterations. The derivative term, controlled by a K_d constant, provides a sense of the speed of the error variation (predicts future values of the error) and works in opposite direction to the integral term by reducing the inertia.

$$p_t = K_p * e_t + K_i \int_0^t e_t dt + K_d * \frac{d}{dt} e_t \tag{1}$$

This letter looks at TSCH scheduling as a closed loop control problem, with the objective to optimize the schedule of the TSCH network so the number of cells in it accommodates to the node's demand. The approach is fully decentralized: scheduling decisions are taken according to the current state of the node, without requiring global or partial network information.

The variables we optimize are the size of the queue and the number of scheduled slots that are unused in a slotframe. The PID controller computes the number of required cells in the schedule, according to the traffic demand on that particular node. At each slotframe, the PID controller determines the number of cells that need to be added or removed in a per-neighbor basis, according to the state of the queues and the previous slotframes cells usage. For example, if the queue fills up, the controller determines that more cells in the schedule are needed. Similarly, if the number of unused cells increases, the controller decides to remove some of them. Controlling the inertia of the PID is fundamental to avoid constants increments/decrements to the slot allocation because this introduces unnecessary energy consumption and packet overhead due to unnecessary allocations.

We propose a PID controller which calculates the number of cells to be scheduled at every start of slotframe, for each of the possible traffic destinations. The controller is also in charge of re-scheduling under-performing cells given a configurable threshold, e.g. if a link has an average Packet Delivery Ratio (PDR) under the threshold, the cell is relocated after N PID iterations, with N configurable. Once the controller indicates the number of cells to add/delete, the 6top protocol [10] carries out the negotiation and reservation with the neighbor. The proposed cells are selected randomly from those not being allocated in the requester's schedule. The objective of the mechanism is twofold: (1) keep a low (or zero) number of packets in the queue, (2) minimize the number of cells allocated in the node's schedule in order to reduce unnecessary energy consumption.

The number of cells to be scheduled per target node (C_{Sj}) is calculated by the PID controller using (2) as a discrete approximation of (1).

The proportional error of the PID to a particular neighbor (e_{tj}) can be seen in (3) and is calculated by the current number of packets in queue (P_{cj}) to that particular neighbor j, minus the current number of cells in the schedule to that particular neighbor (C_{cj}), and minus the target number of packet in queue (P_{tj}).

$P_{tj} = 0$ is the objective of our PID controller to maintain the queue empty. $\sum_0^{nj-1} e_{t_{nj}} * \delta_{SF}$ is the integral of the errors, that is the sum of the errors in the last N slotframes. δ_{SF} represents the slotframe duration. The last term is a derivative term that indicates how fast the error changes. In addition, note that the total number of cells to be scheduled in one iteration of the PID must not exceed the total schedule capacity as noted in (2).

$$C_{Sj} = K_p * e_{tj} + K_i \sum_0^{n-1} e_{t_{nj}} * \delta_{SF} + K_d * \frac{e_{tj} - e_{tj-1}}{\delta_{SF}},$$
$$j \in source\ nodes,$$
$$\sum_j C_{Sj} \leq Total\ slotframe\ capacity \tag{2}$$

$$e_{tj} = P_{cj} - C_{cj} - P_{tj} \tag{3}$$

To illustrate, let's imagine we use $K_p = 1$. When multiplied by e_{tj}, having the rest of K's set to zero, we would have a PID result proportional to the current error. That is, if the queue is 2 packets too large, the PID result C_S is that 2 slots need to be allocated to the schedule. By reducing the weight of the proportional error and adding weight to the integral term, we smoothen the reaction of the controller by restricting the impact of sudden queue variations as we introduce certain inertia to the system. Note that *Ks* are represented as float values [0..1] that depend on the scheduling objective that wants to be achieved. K_p weights the proportional term; it determines the amount of cells to schedule given a certain error. It is desirable to have one extra cell per extra packet in the queue; therefore K_p should be close to 1.0 but giving some space to both the integral term to avoid excess proportionality,

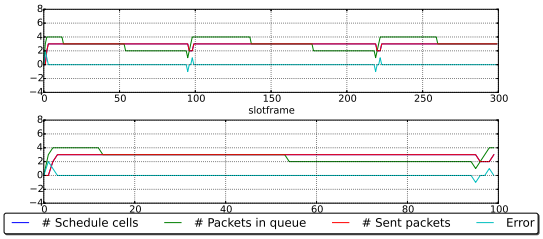


Fig. 2. Number of cells allocated for constant traffic. Top: results for 300 slotframes. Bottom: zoom on the first 100 slotframes.

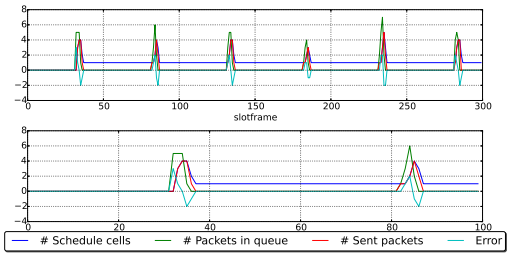


Fig. 3. Number of cells allocated for bursty traffic. Top: results for 300 slotframes. Bottom: zoom on the first 100 slotframes.

and therefore instability, and also to the derivative term. K_i is used to stabilize the controller, giving weight to the past actions. K_i depends on the window size and on the speed of the controller; its value should be approximately $\delta(SF)^n$ times smaller than K_p. Finally, K_d breaks the system inertia as it anticipates future values of the system and should have a value proportional to the speed of the system.

In the proposed approach, is important to recall that the 6top negotiation protocol does not ensure collision-free allocations, but it is the responsibility of the SF to address under-performing slots. A scheduling collision may cause a delay of several slots in the allocation procedure. These slots are dependent on the number of retries. The PID policy is simple in this case. The PID computes the required cells according to the current status. If an allocation failed in the previous attempt, the result of the PID calculation in the current iteration will be more aggressive to reach the objective as the system is drifting from it. If we are in the extreme case that no more cells in the schedule of the counterpart are available, the queue of the node grows and eventually starts dropping packets. This behavior is not a PID problem but a network planning problem as the required bandwidth is larger than the available bandwidth.

IV. Evaluation

We implemented the PID scheduling controller in the OpenWSN project [11]. OpenWSN is an open-source implementation of a standards-based protocol stack, including IEEE802.15.4e TSCH, 6LoWPAN, RPL, CoAP and the latest IETF 6TiSCH standards. OpenWSN implements the 6top sublayer [10]. The PID scheduling controller is a component that uses the 6top sublayer. Different SFs can be used in 6top to decide when to add/delete cells to each neighbor. Once the SF decides to add/delete cells, the 6top protocol locally negotiates with the neighbor. Negotiation occurs using

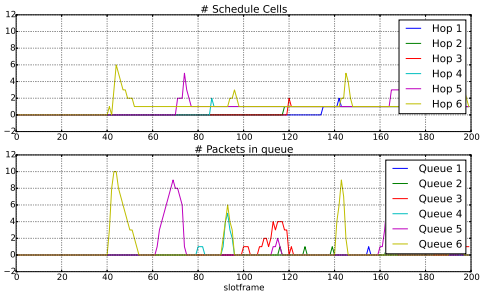


Fig. 4. Number of cells allocated for bursty traffic in a mulithop setting (6 hops). Top: For each of the hops, evolution of allocated cells in the schedule. Bottom: number of packets in the queue evolution at each hop.

pairwise communication between nodes exchanging allocation requests/responses. The PID controller is an SF which determines the number of cells that should be scheduled to a particular neighbor and invokes the corresponding 6top protocol function to request the cells. 6top is standardized in such a way that a SF can be plugged-in.

To test the performance of the PID under various types of traffic, we utilized `cstorm` application in OpenWSN. This application enables us to control the packet generation rate and therefore emulate different scenarios such as those with constant traffic and others with bursty traffic.

The evaluation aims to demonstrate that the mechanism is able to dynamically and in a distributed manner manage the schedule of the network without requiring centralized information.

A. Description of the test environment

OpenSim is the emulator of the OpenWSN project. It compiles the firmware running on real nodes as a Python extension module and enables to run them in a single computer emulating a real network. The simulated code is exactly the same as the one that runs on real devices.

The first network built with OpenSim contains two emulated nodes: node A configured as root node, node B configured as slave. The slot duration and length of the slotframe of this network are set to 15 ms and 101 slots long, respectively. Both nodes use 5 pre-configured shared cells for best effort and 6top reservation traffic. The cells reserved through 6top negotiation are dedicated cells between both nodes and hence without collision. The `cstorm` application is used to generate traffic from node B to A. The rate of the traffic is controlled by manipulating the packet generation interval. The smaller the interval, the higher the traffic. To accurately test the performance of the PID controller, `cstorm` packets are forced to be sent on reserved cells. All other packets (beacons, routing signaling traffic, etc.) can only be sent through shared cells. Only the packets in the queue generated by `cstorm` and the 6top layer are used for the PID calculation.

The experimental data is obtained at the last slot in the slotframe. The recorded information for each slotframe consists in a tuple with six elements: the number of scheduled cells, the number of packets in the queue, the number of sent packets, the traffic, the average duty cycle for the slotframe and the error (as e_t in (3)). The traffic represents the number of packets generated every slotframe.

A second setting is used to evaluate the performance of the PID scheduling in a multihop network. We use a 7 node linear

network, or 6 hops. Only the last node is generating `cstorm` traffic. In addition, we limit the number of shared slots to 2 between each pair of nodes, aiming to see more cells being reserved all along the 6 hops.

B. Experiments

In our experiments, we aimed to evaluate the performance of the PID scheduler under constant and bursty traffic and compared it to a well-known state of the art mechanism referred as Orchestra [9]. Two experiments have been conducted with a predetermined PID configuration with values ($K_p = 0.7$, $K_i = 0.075$, $K_d = 0$, $n = 4$). The PID configuration dependents on the application objective and part of the predeployment adjustment and tuning. We plot the variation of the number of scheduled cells in the schedule, packets in the queue, the sent packets and the error, as shown in Fig. 2 and Fig. 3. Fig. 2 presents the cases for constant traffic. For those, the `cstorm` packet generation rate is set to 3 packet every slotframe. Per Fig. 2, the scheduled cells are stable at 3 allocated cells. Despite the fact that packets in the queue eventually oscillate from 4 to 2, the scheduled cells stay at 3 and do not drop to 2 until the number of packets in the queue drop to 1; then, when the packets in the queue increase again the PID reacts in just 2 slotframes to reallocate the 3 slots.

To evaluate a bursty traffic scenario, we use a period of 50 slotframes to generate traffic. During the first 3 of the 50 slotframes, the packet generation rate is set randomly between 3 and 5 packets per slotframe. No packets are generated in the remaining 47 slotframes. As seen in Fig. 3, every time a traffic burst occurs, the PID controller allocates the determined number of cells through the 6top management interface, and recycle them after the end of the burst within 4 to 5 slotframes (due to the integral weight and the overhead of the negotiation protocol). The number of allocated cells is reduced to one after the end of the burst. At that point, the number of cells to be removed is less than one; thus, no cell can be removed, and the allocated cells are kept at one until next traffic burst.

A second experiment, depicted in Fig. 4 presents the evolution of cell allocations along a multihop path. Only the leaf node (hop 6) generates traffic using `cstorm` every 50 slotframes. We can see the filtering effect caused by progressive allocation of cells (smooth ramp due to inertia introduced by the PID integral term) which causes the next hop to require less "extra" slots to relay the traffic as their queue size grows more progressively.

Finally, in Fig. 5, the duty cycle of a network with a static schedule (Orchestra [9]) is compared to the PID scheduler. In the experiment, we use bursts of 8 packets/slotframe during 3 slotframes every 50 slots for a three-node network. Three different cases are considered following the Orchestra approach. Receiver oriented (Rx cells), Sender oriented (Tx cells) and Relay schedules (Both Tx and Rx cells). The static scheduler has been configured to deal with that burst using a fixed capacity of 8 dedicated slots per slotframe according to the evaluated case (Tx, Rx or Relay). The figure presents the duty cycle per slotframe according to the node's role (Sender, Receiver or Relay). In the top, we see the overhead caused by the signaling of the PID which causes a slightly higher duty cycle that tends to zero as the traffic demand decreases. In this case, the static schedule configuration outperforms in average slightly the PID scheduler as transmission cells are not used if queues are empty; the PID introduces an overhead to remove them from the schedule. However, the middle and bottom figures, showing the receiver and relay oriented measurements,

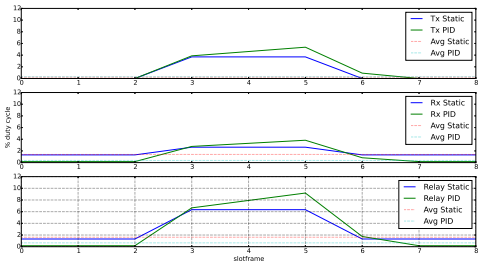


Fig. 5. Duty cycle comparison between the transmission oriented, sender oriented and relay scenarios during a burst. Average traffic in dotted lines.

indicate that the PID removes Rx cells not being used after a burst, saving the nodes to idle-listen in those slots. Average doted lines are provided for the three cases, showing that the PID approach outperforms static scheduling in terms of duty cycle when the network activity has a bursty nature.

V. Conclusions

This letter presents a distributed and efficient scheduling policy for 6TiSCH networks. This policy is based on the PID industrial control paradigm and uses the 6top management layer being standardized at the IETF. The mechanism targets schedule stabilization for coping with dynamic application demands in a distributed manner, and uses pairwise communication between neighbor nodes. We demonstrate that a PID schedule is able to cope with different types of traffic and react autonomously to sudden demand variations, targeting stabilization and minimization of cells in the schedule and the queue.

References

[1] T. Watteyne, M. R. Palattella, and L. A. Grieco, *Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement*, IETF Std. RFC7554, May 2015.

[2] T. Watteyne, A. Mehta, and K. S. Pister, "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense," in *Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*, Tenerife, Canary Islands, Spain, 29-30 October 2009.

[3] D. Dujovne, T. Watteyne, X. Vilajosana, and P. Thubert, "6TiSCH: Deterministic IP-enabled Industrial Internet (of Things)," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 36–41, 2014.

[4] X. Vilajosana and K. Pister, *Minimal 6TiSCH Configuration*, IETF Std. draft-ietf-6tisch-minimal-12 [work-in-progress], 21 September 2015.

[5] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia, "Traffic Aware Scheduling Algorithm for Reliable Low-power Multi-hop IEEE 802.15.4e Networks," in *International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. Sydney, Australia: IEEE, 9-12 September 2012, pp. 327–332.

[6] A. Tinka, T. Watteyne, K. S. Pister, and A. M. Bayen, "A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping," *Transactions on Mobile Communications and Applications*, vol. 11, no. 1, pp. 201–216, September 2011.

[7] A. Morell, X. Vilajosana, J. L. Vicario, and T. Watteyne, "Label Switching over IEEE802.15.4e Networks," *Wiley Transactions on Emerging Telecommunications Technologies*, vol. 24, no. 5, pp. 458–475, 3 May 2013.

[8] N. Accettura, M. R. Palattella, G. Boggia, L. A. Grieco, and M. Dohler, "Decentralized Traffic Aware Scheduling for multi-hop Low power Lossy Networks in the Internet of Things," in *International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 4-7 June 2013, pp. 1–6.

[9] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *International Conference on Embedded Networked Sensor Systems (SenSys)*. Seoul, South Korea: ACM, 1-4 November 2015.

[10] Q. Wang and X. Vilajosana, *6TiSCH Operation Sublayer (6top)*, IETF Std. draft-wang-6tisch-6top-sublayer-03 [work-in-progress], 6 July 2015.

[11] T. Watteyne, X. Vilajosana, B. Kerkez, F. Chraim, K. Weekly, Q. Wang, S. D. Glaser, and K. Pister, "OpenWSN: a Standards-based Low-power Wireless Development Environment," *Transactions on Emerging Telecommunications Technologies*, vol. 23, no. 5, pp. 480–493, 2012.

Appendix B

Paper: JXTAnonym: An anonymity layer for JXTA services messaging

M. Domingo, J. Arnedo, (2012). "**JXTAnonym: An anonymity layer for JXTA services messaging**". IEICE Transactions on Information and Systems. Núm. E95-D, Pág. 169-176. ISSN.0916-8532. IF: 0.218, Q4: 129/132. Category: COMPUTER SCIENCE, INFORMATION SYSTEMS.

PAPER Special Section on Trust, Security and Privacy in Computing and Communication Systems

JXTAnonym: An anonymity layer for JXTA services messaging*

Marc DOMINGO-PRIETO[†a)], *Nonmember and* **Joan ARNEDO-MORENO**[†b)], *Member*

SUMMARY

With the evolution of the P2P research field, new problems, such as those related with information security, have arisen. It is important to provide security mechanisms to P2P systems, since it has already become one of the key issues when evaluating them. However, even though many P2P systems have been adapted to provide a security baseline to their underlying applications, more advanced capabilities are becoming necessary. Specifically, privacy preservation and anonymity are deemed essential to make the information society sustainable. Unfortunately, sometimes, it may be difficult to attain anonymity unless it is included into the system's initial design. The JXTA open protocols specification is a good example of this kind of scenario. This work studies how to provide anonymity to JXTA's architecture in a feasible manner and proposes an extension which allows deployed services to process two-way messaging without disclosing the endpoints' identities to third parties.

key words: *peer-to-peer, security, anonymity, JXTA, Java, onion routing.*

1. Introduction

Just as the popularity of P2P applications has risen, so have concerns regarding their security. As P2P applications move from simple data sharing, for example BitTorrent [1], to a broader spectrum, such as e-learning environments [2], they become more and more sensitive to security threats. Therefore it becomes very important to design P2P frameworks which not only can be easily adapted to a broad set of application scopes, but also take into account an acceptable security baseline. Under today's standards, it is expected that it is possible to deploy some degree of privacy, ensuring that the contents of a message exchange are not revealed to an eavesdropper, and authentication, guaranteeing which is the identity of each endpoint during any message exchange.

Manuscript received January 1, 2010.
Manuscript revised January 1, 2010.

[†]The author is with the Estudis d'Informàtica, Multimèdia i Telecomunicació. Universitat Oberta de Catalunya. Rambla del Poblenou 156, 08018, Barcelona, Spain

*This work was partially supported by the Spanish MCYT and the FEDER funds under grant TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 "ARES", funded by the Spanish Ministry of Science and Education.

a) E-mail: mdomingopr@uoc.edu
b) E-mail: jarnedo@uoc.edu

DOI: 10.1587/trans.E0.??.1

As P2P systems evolve and become widespread in new scenarios, more advanced security capabilities should be considered. An example of them is message anonymity [3], allowing a peer to send requests to an arbitrary service in such a manner that nobody can determine the endpoints' identities. The need for this feature ranges from everyday situations such as a corporate suggestion box, a peer evaluation form or personal data sharing [4], to those related to much serious topics, such as freedom of speech or whistleblowing. Unfortunately, because of its architecture, P2P systems are specially weak to anonymity attacks unless this capability is included in the system's initial design. Even when providing the aforementioned basic security capabilities, it may be easy for other peers to acquire information by monitoring message flows or intercepting queries routed through them [5]. Consequently, message sources and final services are completely exposed to neighboring peers and super-peers.

JXTA (or "juxtapose") [6] is an example of a P2P system which already considers some basic security capabilities, but not anonymity. Consisting in a set of open protocols that enable the creation and deployment of P2P networks, it provides applications with the capability to easily discover and observe peers, exchange messages and publish resources. Even though in its successive revisions security mechanisms have slowly crept in, up to an acceptable degree [7], in its latest version (2.7RC1), available on January the 12th 2011, no previsions are made about being able to provide any degree of anonymity some day.

In this paper we extend our previous work in [8] and present JXTAnonym, an anonymity layer which allows peers to exchange messages with JXTA services without disclosing the identity of the participating parties to neighboring peers or super-peers. Thus, anonymous two-way messaging is made possible in JXTA. The proposed layer is based on a popular approach within the context of P2P applications, onion routing [9]. However, it is specifically adapted to the idiosyncrasies of JXTA, taking advantage of service access mechanisms already provided by the platform, instead of defining additional protocols. In addition, the amount of required changes on an existing system in order to integrate anonymous messaging is minimized. In that manner, peers which support anonymity may coexist with those who don't, without incompatibilities.

The paper is structured as follows. In Section 2 we provide a brief summary of the current mechanisms based on onion routing, focusing on those methods which provide bidirectional messaging capabilities in P2P networks. Section 3 describes how our base system has been extended to encompass bidirectional messaging, suiting to the idiosyncrasies of JXTA services. The outcome of our experimental results in order to evaluate its performance is provided in Section 4. Section 5 provides a security discussion of JXTAnonym. Finally, Section 6 concludes this paper and outlines further work.

2. Related work on onion routing

Anonymity in P2P networks can be achieved using three different approaches: *Unimessage-based*, *Split message-based* and *Replicated message-based*. Our work focuses on the first one, since it is the most popular and efficient, while maintaining a high degree of anonymity. Also it assumes that nodes are completely autonomous [10]. These characteristics mesh with the structure of JXTA and allows anonymous bidirectional communications.

A unimessage-based approach, also known as *Onion Routing* [9], provides anonymity by sending the message through a random sequence of proxies before it reaches the destination. Those proxies are labeled *OnionRouters*. Generally, the path of OnionRouters is pre-constructed by the sender before the message is sent. The message is repeatedly encrypted in a manner that, during transit towards the destination, a single encryption layer can be taken out, one at a time, at each OnionRouter. Each time a layer is peeled off, the identity of the next hop is obtained. Thus, at each hop, an OnionRouter does not know whether the message is being sent to another proxy or to the actual final destination. In the same manner, the destination peer cannot know whether the received message comes directly from the source or it has been relayed through a set of OnionRouters.

The onion routing approach is also able to provide anonymous two-way communications in two ways:

- Constructing both a query (source to destination) and response (backwards, destination to source) onion at the source peer and including it inside the onion message together with the data to be sent. Therefore the destination peer, after replying the query, can forward its reply data to the source using the response onion path.
- Including a session ID within the onion route, which is stored on transit, thus creating a kind of return path virtual circuit [11]. When the destination peer responds the query, the message is routed back following the established path, but in the opposite direction.

Onion Routing is used as the core of many anonymity protocols to achieve providing mutual anonymity, being *APFS* (Anonymous P2P File Sharing) [11] and *Tor* [12] the ones that share more features when constructing of the anonymous path.

As its name suggests, the APFS protocol is used for P2P file sharing, providing mutual anonymity of the initiator and the responder in a connection. In APFS, each peer has to choose a proxy peer and create a Onion Route to it. This proxy peer will be the entry point to the anonymous network for that peer. Additionally, exists a well known bootstrapping peer called coordinator which maintains a list of all the peers connected to the network and which are acting as a servers. However, instead of peers address, their proxy address are stored.

Tor is a circuit-based anonymous communication service. The sender's anonymity is maintained by constructing a circuit from the source to an exit node, who will communicate with the destination. The construction of this route is performed by incrementally extending the circuit, hop by hop, and negotiating a different symmetric key with each intermediate node. This circuit is periodically recalculated and can be used at its entire length, or shortened to modify the exit node. Answers to queries are transmitted through the same circuit.

Mutual anonymity can be achieved on Tor by using hidden services. The basic idea of hidden services is that the service provider is hidden behind some peers who act as his Introduction Points (IP) while the service consumer is behind a Rendezvous Point (RP). The communication between these points is done by following circuits. The steps required to use a hidden service are:

- The service consumer contacts with one of the IPs, announcing his RP.
- The IP forwards the message to the service provider, who will create a circuit to the RP
- At this point, mutual anonymity communication between both can be established through the RP.

3. A proposal for JXTA anonymous messaging

JXTA provides mechanisms to share services. Such services are commonly consumed by exchanging bidirectional messages. In order to provide anonymity in services consumption, is important to inspect the JXTA messaging architecture. From this review, it is possible to identify the elements that can be taken advantage of in order to define an anonymity layer for which is transparent and finely integrated to JXTA architecture, without the need of defining additional protocols.

3.1 JXTA Messaging architecture

In this subsection, we provide an overview of those of JXTA's main architectural elements related to our proposal. A detailed explanation can be found in [13].

The main idiosyncrasy in JXTA's design, which sets it apart from other P2P frameworks, is introducing the concept of *Peer Group*, a segmentation of the global JXTA network. All peers publish and consume services within the context of a group, interacting with each other by using some JXTA core services, the most important ones being the *Discovery Service* and the *Pipe Service*.

Every resource in a JXTA group is described by an *Advertisement*, a metadata document. A resource cannot be accessed unless its Advertisement is previously retrieved. Advertisements must be periodically published, since they expire and are flushed from the network after some time (by default, 2 hours). The Discovery Service's responsibility is managing advertisements, allowing peers to publish and find available resources. The most important types of Advertisements are:

- *Peer Advertisement*: Describes a peer and the resources and services it provides. Each peer is responsible for the publication of its own Peer Advertisement, and is considered online only while it continues to do so.
- *Pipe Advertisement*: Describes a *JXTA Pipe*, the main mechanism in JXTA to exchange data between two applications or services.

The Pipe Service is responsible for managing message exchanges using JXTA Pipes. The simplest pipe in JXTA, the *JxtaUnicast*, provides an asynchronous, unidirectional message transfer mechanism which can be easily established and managed. Nevertheless, there is a higher-level communication abstraction provided by the *JxtaBiDiPipe* which provides a bidirectional communication channel. The latter is usually preferred by services, since it allows the direct use of a straightforward query-response exchange. The description of JXTA's standard service model based on this procedure follows:

1. Each service provider starts a *JXTAServerPipe* using the Pipe Service, which exposes and listens to an *input pipe* in order to process communication requests. This input pipe is defined using a Pipe Advertisement.
2. This Pipe Advertisement is made public by the service provider using the Discovery Service.
3. The Advertisement is propagated within the group by *Rendezvous Peers*, special super-peers who efficiently distribute Advertisements.
4. To consume a service, a peer must retrieve the Pipe Advertisement (via the Discovery Service) and then a bidirectional connection must be established via the Pipe Service, creating a JxtaBiDiPipe.
5. Once a communication request is received at the server side, an independent JxtaBiDiPipe is created and bound with the request. A message exchanges may begin from now on.

Message exchanges can be secured in JXTA by using a group based on the *PSE (Personal Security Environment) Membership Service*. Under this kind of peer group, each peer is provided with a credential based on PKIX [14] certificates. This guarantees that each peer has initialized a valid pair of public-private keys and that the public key of each peer is automatically distributed inside its Peer Advertisement, in a special service parameter entry.

3.2 Anonymizing procedure

We propose an anonymity layer that causes the minimum interferences on the JXTA messaging architecture, according to the review done in Section 3.1. JXTAnonym, an anonymizing service, is deployed in those peer group members which want to anonymously exchange messages, creating an anonymous subnetwork within the context of a peer group. Group members may freely join and leave this network.

The proposed service is tailored to JXTA's core services features, such as invisible publication, discovery and access to services (via the Discovery Service), message management via the Pipe Service and usage of the PSE secure environment for cryptographic data generation and distribution. Therefore, the deployment procedure follows the same steps as for any other peer service, making use of JXTA's service model without the need of modifying JXTA's initial design.

The service's anonymity mechanism is based on an onion routing approach, examined in detail in Section 2, protecting the identity of end clients (consumers) and services (providers) from the rest of the group members. In addition, the end service is also unable to establish the end client's identity. Its general architecture is summarized in Figure 1.

The execution of JXTAnonym in any peer encompasses three different steps: *JXTAnonym Service Publication*, *Message Setup* and *Message Processing*.

3.2.1 JXTAnonym Service Publication

Just like any other JXTA service, an instance of the JXTAnonym service in a peer group member listens to incoming queries using an input pipe. This pipe is made available to other peer group members by periodically publishing its Pipe Advertisement, via the Discovery Service. However, we propose that Pipe Advertisements are not published as standalone documents.

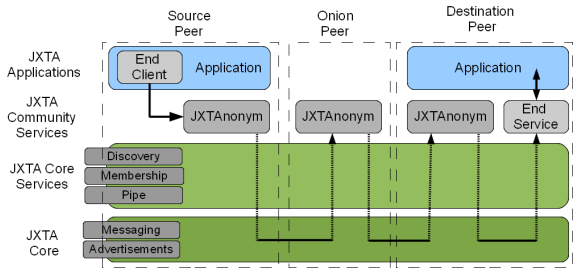


Fig. 1 JXTAnonym service operation in the context of JXTA's architectural design.

They are piggybacked inside the service parameters section of the Peer Advertisement, identified by a hard-coded well-known JXTAnonym service ID. The inclusion of service related-data in this section is also used by some other JXTA services, such as the PSE Membership Service. The main advantage of this approach is that it is guaranteed that the service's Pipe Advertisement is always being published while the peer is online, thus being always available and up to date. In addition, each peer's Peer Advertisement contains all the information required by JXTAnonym: the local service Pipe Advertisement and the peer's cryptographic data.

Once a peer decides to stop participating in the anonymous network, the service parameter entry is removed from the Peer Advertisement, continuing its publication as normal. Due to the JXTA's architecture, this new Advertisement will be propagated across the Peer Group, replacing the old one. If a peer becomes unreachable, its Peer Advertisement is not going to be updated any longer, being flushed after the expiration time is reached. Once this happens, the JXTAnonym Pipe Advertisement will stop being available too, guaranteeing that at any time only active instances of the service are published.

3.2.2 Message Setup

This step initiates the onion routing process and is performed by the peer who wants to anonymously access an end service deployed at a remote peer within the same group.

First of all, we will describe the onion layering procedure in $JXTAnonym$, since it is used at several steps in the message setup process. The layering procedure accepts two parameters: $OnionCore$, which is a JXTA Message structure which may contain any arbitrary data, and $PeerAdvList = Adv_1, \cdots, Adv_n$, an ordered sequence of Peer Advertisements.

For each Peer Advertisement Adv_i, from $n \ldots 1$, the following process is iteratively executed. $Onion_i$ is considered the result of each iteration, being $Onion_1$ the final result:

1. This iteration's input, *inputData*, is chosen.
   a. For the first iteration ($i = n$) the $OnionCore$ structure is considered *inputData*.
   b. For the rest of iterations ($i = n-1, \ldots, 1$), the result of previous iteration, $Onion_{i+1}$ is considered *inputData*.
2. The public key PK_i is retrieved from Adv_i's Membership Service definition entry. Under the context of a peer group which implements the PSE Membership Service, it is guaranteed that PK_i actually exists.
3. *inputData* is encrypted using PK_i under a wrapped key scheme [15], generating *EncInputData*.
4. Adv_i's PID field (the peer's unique identifier) is retrieved.
5. A new $OnionLayer$ structure is generated by creating a JXTA Message with the following fields:
   - `NextHop` $= PID$
   - `OnionRoute` $= EncInputData$
6. The $OnionLayer$ structure becomes this iteration's result ($Onion_i$).

Once the layering procedure has been established, the message setup process description follows:

1. A peer S decides to use a service ($EndService$) executing on peer D.
2. S's corresponding client ($EndClient$) creates a query message ($JXTAQueryMessage$), specifying which is the destination peer (peer D), and a callback structure ($PipeListener$). Such structure is used by JXTA applications to allow asynchronous processing of the incoming response, so they do not need to block until the reply is received.
3. $EndClient$ locates the $EndService$'s *JXTABidiPipe* Pipe Advertisement, from now on *EndServicePipe*, using the Discovery Service.
4. So far, steps 1-3 describe the required steps to access a generic JXTA service. At this point, $EndClient$ would open a connection using $EndServicePipe$ and use it to directly send $JXTAQueryMessage$. Instead, it decides to use the anonymity service, $JXTAnonymSvc$.
5. If an instance of $JXTAnonymSvc$ is being executed at S, an anonymous client ($JXTAnonymClient$) is also available. Such client is invoked, receiving $EndServicePipe$, D's identity, $JXTAQueryMessage$ and $PipeListener$ as parameters. From now on, $JXTAnonymClient$ will manage the rest of the message setup process. $EndClient$ considers message processing completed as far as it is concerned.
6. A random symmetric key ($FinalSymmetricKey$) is generated and used to encrypt both $JXTAQueryMessage$ and $EndServicePipe$, obtaining $QueryEncryptedMessage$.

7. A new record is created in a local table *PendingQueries*, containing $FinalSymmetricKey$ and $PipeListener$, the former being the primary key.
8. A set of random data ($RndData$) is generated. The length of this data should be between 2411 and 4614 bytes. The reasons will be explained in detail in Section 5.
9. S generates a *ResponseCore* structure. This structure is a JXTA message composed of the following name-value pairs:
   - `RandomData` = $RndData$
   - `SymmetricKey` = $FinalSymmetricKey$
10. At this point, a number of OnionRouter peers must be chosen in order to create a response path. It is considered that 3 hops is good enough [12]. This is done by using the Discovery Service to get a set of Peer Advertisements $RP = Adv_1, Adv_2, Adv_3, Adv_4$ where is it true that $\forall Adv_i, (i < 4), Adv_i \neq Adv_S \& Adv_i \neq Adv_D$ and $Adv_4 = Adv_S$, and in all of them the $JXTAnonymSvc$ service parameter field exists (all of them deploy *JXTAnonym*). It is worth remarking that, since S holds the end client waiting for the response, thats the reason why it must be the last peer in the response path.
11. Once the response path is established, an onion structure *ResponseOnion* is created using the layering procedure previously described. The input parameters are *ResponseCore* and RP.
12. From *ResponseOnion* the NextHop (*ResponseFirstHop*) and the OnionRoute (*ResponseRoute*) fields are extracted.
13. A *QueryCore* structure is created. This structure is a JXTA Message composed of the following name-value pairs:
   - `NextHop` = $ResponseFirstHop$
   - `OnionRoute` = $ResponseRoute$
   - `SymmetricKey` = $FinalSymmetricKey$
14. Now the query path, QP must be generated. This process is identical to step 10, but in this case, $Adv_4 = Adv_D$ instead of Adv_S. Ideally, $RP \neq QP$.
15. Once the query path is established, an onion structure *QueryOnion* is created using the layering procedure previously described. The input parameters are *QueryCore* and QP.
16. From *QueryOnion* the NextHop (*QueryFirstHop*) and the OnionRoute (*QueryRoute*) fields are extracted.
17. An $OnionMessage$ structure is generated. This structure is a JXTA Message composed of the following name-value pairs:
   - `OnionRoute` = $QueryRoute$
   - `EncryptedMessage` = $QueryEncryptedMessage$
18. The Peer Advertisement of $QueryFirstHop$ is obtained via the DiscoveryService, and its JXTAnonymSvc Pipe Advertisement extracted. A new connection to this pipe is created and *OnionMessage* sent through it.

3.2.3 Message Processing

This step is performed when an anonymous message is received at any peer that has deployed $JXTAnonymSvc$. The process starts when a running instance of $JXTAnonymSvc$ receives an incoming message ($OnionMessage$), which contains a $EncryptedMessage$ and *OnionRoute* fields. Then, the value in the *OnionRoute* field is extracted and decrypted using the peer's local private key, accessible using JXTA's PSE Membership service. At this point, three things may happen:

1. The extracted data becomes an *OnionLayer* (see Figure 2). Thus, the message must be routed to another peer.
   a. The values stored in the $NextHop$ and *OnionRoute* fields, respectively PID and *nextRoute*, are extracted.
   b. A new $OnionMessage$ structure is generated, where:
      - `OnionRoute` = *nextRoute*
      - `EncryptedMessage` = *EncryptedMessage*
   c. Using the Discovery Service, PID's Peer Advertisement is located. From this advertisement, the $JXTAnonymSvc$ service Pipe Advertisement is extracted.
   d. A new connection to the next hop is established using the previously recovered Pipe Advertisement. The newly generated $OnionMessage$ is sent through it.

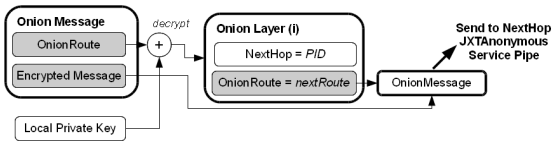


Fig. 2 Processing of OnionMessage at an Onion Peer

2. The extracted data becomes a *QueryCore* (see Figure 3). Thus, the current peer is the destination, D, and holds *EndService*. The query must be processed by this service. In this scenario, the $EncryptedMessage$ field holds the value $QueryEncryptedMessage$.
   a. The values stored in the $NextHop$, *OnionRoute* and $SymmetricKey$ fields, *ResponseFirstHop*, *ResponseRoute* and *FinalSymmetricKey* respectively, are extracted.

b. *QueryEncryptedMessage* is decrypted using *FinalSymmetricKey*, obtaining the original *JXTAQueryMessage* and *EndServicePipe*.
c. A bidirectional connection is established to the *EndService* using *EndServicePipe* and the client's query, *JXTAQueryMessage*, is sent through it. As far as the *EndService* is concerned, a normal connection has been established. It is oblivious to the anonymity layer.
d. The process waits until a response, *JXTAResponseMessage*, is received.
e. The response is encrypted using *FinalSymmetricKey*, generating *ResponseEncryptedMessage*.
f. A new *OnionMessage* structure is generated, where:
   - `OnionRoute` = *ResponseRoute*
   - `EncryptedMessage` =*ResponseEncryptedMessage*
g. *OnionMessage* is sent to the JXTAnonym's pipe of *ResponseFirstHop*.

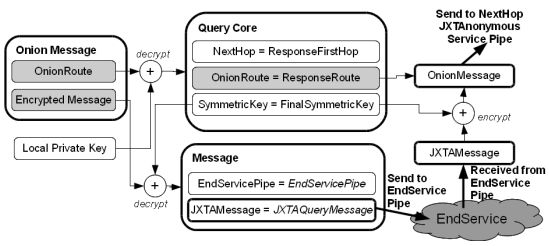


Fig. 3 Processing of OnionMessage at Peer *D*

3. The extracted data becomes in a *ResponseCore* (see Figure 4). Thus, the current peer is the source, *S*. The query-response has completed its round-trip. In this scenario, the *EncryptedMessage* field holds the value *ResponseEncryptedMessage*.
   a. The value stored in the *SymmetricKey* field is extracted (*FinalSymmetricKey*). The *RandomData* field is ignored.
   b. *ResponseEncryptedMessage* is decrypted using *FinalSymmetricKey*, obtaining the original *JXTAResponseMessage*.
   c. The *PendingQueries* local table is searched for the record which uses *FinalSymmetricKey* as its primary key.
   d. Using the *PipeListener* structure, the message is provided to the original *EndClient*. As far as the end client is concerned, JXTA has acted just like during any standard service access. The anonymity layer is invisible.

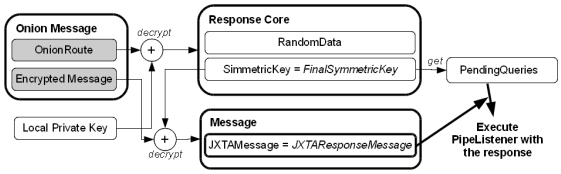


Fig. 4 Processing of OnionMessage at Peer *S*

e. The record is deleted from the *PendingQueries* table.

4. Experimental result evaluation

As a proof-of-concept, we have implemented our proposal and deployed it on a private PlanetLab-based network. We report the key features of our implementation and its performance in order to assess the feasibility of deploying anonymous messaging in a JXTA network.

The testbed deployed a JXTA peer group consisting in a single node operating as a Rendezvous Peer and 32 nodes as a Edge Peers. This is considered a typical group [16]. This peer group was based on the PSE Membership Service and all Edge Peers deployed the *JXTAnonym* service, as well as an additional test end service called *EchoService*, which was anonymized. This service basically replies with the same content as the query.

At this point, some tests were conducted in order to test the performance of using JXTA services with and without anonymity. The goal of these tests is not finding a performance improvement when using JXTAnonym, since applying security is expected to always result in a performance overhead. The goal is to determine if whether an onion routing approach is feasible in a JXTA network, a peer-to-peer framework developed without considering anonymity in its design and architecture. With feasible we refer that the produced overhead is acceptable compared with the benefits it provides.

Our set of tests measure the average time it takes to consume a service, from the moment the query is sent until the response is received, and the percentage of lost queries during transit. This test was executed in three different scenarios and with two different messages loads. The two different message loads consist in sending messages at maximum speed from just one peer or from all peers at once, evaluating the network behavior when it is very saturated. In both tests up to 250 messages are sent.

The first scenario consists in directly consuming the EchoService through a JXTABiDiPipe, as it is usually done in JXTA. This measure is the expected time during JXTA's normal operation. The second scenario sends the queries and the responses to the EchoServices through three intermediates nodes, as JX-

TAnonym would do, but without using any kind of security. This scenario is called 8 hops. Using this test, we can discern which part of the overhead produced by anonymity is due to multiple hops and which is produced by additional computing and transmission costs at each onion peer. The third one evaluates the cost of anonymizing EchoService using JXTAnonym, also using three intermediates nodes.

The average time it took to consume the EchoService in the previously mentioned scenarios is shown in Figure 5. It can be seen that when only one peer consumes the EchoService, the performance is better in all cases, as expected. Consuming the service directly through a BiDiPipe took 1.6 seconds, while consuming it after 8 hops took around 6 seconds. This is a good result, since the difference (4.4s) is less than 6.4 seconds, the better time one can expect considering using 4 extra BiDiPipes to perform 8 hops. This is because instead of using 4 BiDiPipes, 8 simple unidirectional pipes were used. Also, the usage of encryption and the higher message length in JXTAnonym compared with 8 hops produces a really low overhead, just 200ms.

However, when all peers consume the EchoService at the same time, the time required to use it increases significantly when extra hops are performed. This is because when the 32 nodes send messages at the same time, each connection to the EchoService produces 8 extra messages, which is 256 extra messages for each of the 250 messages sent. Consuming the service directly through a BiDiPipe, 8 hops and JXTAnonym took 1.8, 12 and 16.9 seconds respectively. Although there is an important overhead in this scenario when using extra hops, it has to be pointed out that this is the worst case, where each peer sends messages at maximum speed. A real scenario will be between this and the previous scenario. The difference between 8 hops and JXTAnonym (4.9s) is produced due to the use of encryption and higher message length in a saturated network.

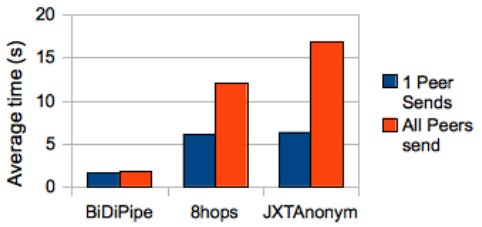


Fig. 5 Average time it takes to use EchoService

Figure 6 shows the amount of lost messages in previous scenarios. Only when BiDiPipe was used no packet loss was produced. This is because BiDiPipes are a higher level implementation of communication than unidirectional pipes, and perform extra communications to set up connections. BiDiPipe was set as no reliable but with a timeout of 1 minute. This decreases the lost packets but increases the communication time, as has been seen previously. When extra hops are performed using unidirectional pipes some packets are lost. In 8 hops scenario the percentage of lost messages ranges from 6 to 20 % while in JXTAnonym it ranges from 10 to 21 %, depending the congestion of the network. These results show that the increase of lost messages in JXTAnonym compared with 8 hops is very low. In conclusion, using unidirectional pipes produce better exchange time but increase the lost messages. Furthermore, modify the expiration time of Advertisements can change these results. A high expiration time (as default) increases the lost messages. In opposite, a lower expiration time minimizes the number of lost messages but increases the congestion of the network, since advertisements should be updated more often.

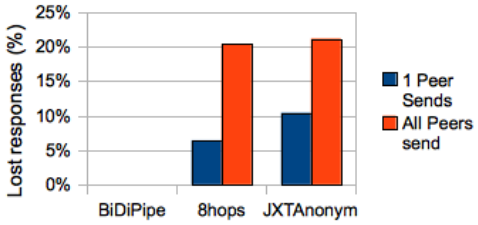


Fig. 6 Percentage of lost petitions when using the EchoService

5. Security discussion

JXTAnonym achieves anonymity by using Onion Routing, whose security has been analyzed in deep [17]. Its main vulnerable is due to traffic analysis attacks, such as packet size analysis. For this reason, as mentioned in Section 3.2.2, the ResponseCore structure contains some random data. This is used to protect the scheme against packet size analysis and make it difficult for an attacker to predict, just for its size, whether a message is the last hop, an intermediate hop, a query message or a response message. The size of each OnionCore is around 208 bytes whereas creating a new OnionLayer adds 399 bytes. From this data extra hops and extra paths are calculated, obtaining that the size of the random data has to be between 2411 and 4614 bytes.

6. Conclusions

In this paper, we have presented JXTAnonym, an anonymity layer that uses the onion routing approach to allow bidirectional message exchanges when accessing services using the JXTA protocols. JXTAnonym provides consumer and provider anonymity in any JXTA service access. The anonymity service has been implemented as a standard JXTA service, and any

JXTA Peer which belongs to a PSE Peer Group can use it. This restriction is necessary because a valid pair of public-private keys are necessary for implementing onion routing, and with this assumption is guaranteed.

Apart from the fact that JXTA currently does not provide anonymous messaging by itself, the main contributions of the chosen approach are twofold. First, the tests performed over JXTAnonym support that anonymity is feasible in JXTA. Although the time used to consume a service increases when using JXTAnonym, this amount is not so high if we take into consideration the amount of connections which are performed and the fact that encryption is employed. Second, JXTAnonym is completely invisible to end services and clients in terms of processing the received data. Their internal operation stays the same whether anonymity is used or not.

These contributions are mainly founded in the fact that JXTAnonym is built over simple JXTA elements, such as Pipe Service, Discovery Service or Advertisements. This allows using JXTAnonym without modifying the JXTA binary release, since no new JXTA protocols are defined. Moreover, Peer Group members can freely choose whether they want to belong or not to the anonymity set within the peer group. Obviously, the higher the number of peers who decide to use JXTAnonym, the higher the anonymity degree obtained.

Further research goes toward extending the anonymity layer to the publication and retrieval of Advertisements, as well as providing anonymous multicast communications. Finally, it is also worth studying how to apply mechanisms that thwart global attackers.

References

[1] B. Cohen, "Incentives build robustness in bittorrent," 1st Workshop on the Economics of Peer-2-Peer Systems, 2003.

[2] K. Matsuo, L. Barolli, F. Xhafa, A. Koyama, and A. Durresi, "Implementation of a JXTA-based P2P e-learning system and its performance evaluation," International Journal of Web Information Systems, vol.4, no.3, pp.352–371, 2008.

[3] A. Pfitzmann and M. Hansen, "Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management – a consolidated proposal for terminology." http://dud.inf.tu-dresden.de/Anon_Terminology.shtml, Feb. 2008. v0.31.

[4] M. Iguchi and S. Goto, "Anonymous P2P Web Browse History Sharing for Web Page Recommendation," IEICE TRANSACTIONS on Information and Systems, vol.E90-D, no.9, pp.1343–1353, 2007.

[5] J. Gu, J. Nah, H. Kwon, J. Jang, and S. Park, "Random Visitor: Defense against Identity Attacks in P2P Networks," IEICE TRANSACTIONS on Information and Systems, vol.E91-D, no.4, pp.1058–1073, 2008.

[6] L. Gong, "JXTA: A Network Programming Environment," Internet Computing, IEEE, vol.5, no.3, pp.88–95, 2008.

[7] J. Arnedo-Moreno and J. Herrera-Joancomart, "A survey on security in JXTA applications," Journal of Systems and Software, vol.82, no.9, pp.1513–1525, 2009.

[8] J. Arnedo-Moreno and M. Domingo-Prieto, "An anonymity layer for JXTA services," 2011 Workshops of International Conference on Advanced Information Networking and Applications, pp.102–107, IEEE Press, 2011.

[9] P. Syverson, D. Goldsclag, and M. Reed, "Anonymous connections and onion routing," Proceeding of the IEEE 18th Annual Symposium on Security and Privacy, pp.44–54, 1997.

[10] X. Ren-Yi, "Survey on anonymity in unstructured peer-to-peer systems," Journal of Computer Science and Technology, vol.23, no.4, pp.660–671, July 2008.

[11] V. Scarlata, B. Levine, and C. Shields, "Responder anonymity and anonymous peer-to-peer file sharing," Proceeding of the ACM CCS, pp.17–26, 2001.

[12] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second generation onion router," Proceeding of the 13th USENIX Security Symposium, pp.303–320, 1998.

[13] Sun Microsystems Inc., "JXTA v2.0 protocols specification," 2007. `http://java.net/projects/jxta-spec`.

[14] CCITT, "The directory authentication framework. recommendation," 1988.

[15] B. Kaliski and J. Staddon, "PKCS1: RSA Cryptography Specifications. Version 2.0," 1998. `http://www.ietf.org/rfc/rfc2437.txt`.

[16] E. Halepovic and R. Deters, "The JXTA performance model and evaluation," Future Generation Computer Systems, vol.21, no.3, pp.377–390, 2005.

[17] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an analysis of onion routing security," in Designing Privacy Enhancing Technologies, Lecture Notes in Computer Science, vol.2009, pp.96–114, 2001.

Marc Domingo-Prieto holds the degree of Computer Systems and the master in Computer Architecture, Networks and Systems from Universitat Politècnica de Catalunya (UPC). He is a research assistant in the K-ryptography and Information Security for Open Networks (KISON) research group in the Open University of Catalonia (UOC). His research interests include scalable distributed algorithms and applications, energy efficient and security in mobile and peer-to-peer applications.

Joan Arnedo-Moreno is a lecturer at Estudis d'Informàtica, Multimimèdia i Telecomuncicació in the Open University of Catalonia (UOC) and works as a part-time assistant at the Universitat Politècnica de Catalunya (UPC). From the latter, he earned his degree in Computer Science in 2002 and his PhD. degree in 2009. He has published several papers in international conferences and journals and has been invited as keynote speaker at several conferences. Both his teaching and research interests are related to the fields of networking and security, more specifically in peer-to-peer systems.

Appendix C

Paper: jxSensor: a sensor network integration layer for JXTA

Marc Domingo-Prieto, Joan Arnedo-Moreno, and Xavi Vilajosana-Guillén, "**jxSensor: a sensor network integration layer for JXTA**", in 2012 15th International Conference on Network-Based Information Systems (NBiS-2012). 2012, pp. 435–440, IEEE Press. Indexed in SCOPUS.

2012 15th International Conference on Network-Based Information Systems

jxSensor: a sensor network integration layer for JXTA

Marc Domingo-Prieto, Joan Arnedo-Moreno
Internet Interdisciplinary Institute (IN3)
Universitat Oberta de Catalunya
C\Roc Boronat, 117 , 7 flour 08018 Barcelona, Spain
mdomingopr, jarnedo@uoc.edu

Xavi Vilajosana-Guillén
Computer Science, Multimedia and Telecommunication
Department, Universitat Oberta de Catalunya
Rambla del Poblenou 156, 08018 Barcelona, Spain
xvilajosana@uoc.edu

***Abstract*—Nowadays, Wireless Sensor Networks (WSN) are already a very important data source to obtain data about the environment. Thus, they are key to the creation of Cyber-Physical Systems (CPS). Given the popularity of P2P middlewares as a means to efficiently process information and distribute services, being able to integrate them to WSN's is an interesting proposal. JXTA is a widely used P2P middleware that allows peers to easily exchange information, heavily relying on its main architectural highlight, the capability to organize peers with common interests into peer groups. However, right now, approaches to integrate WSNs to a JXTA network seldom take advantage of peer groups. For this reason, in this paper we present *jxSensor*, an integration layer for sensor motes which facilitates the deployment of CPS's under this architecture. This integration has been done taking into account JXTA's idiosyncrasies and proposing novel ideas, such as the *Virtual Peer*, a group of sensors that acts as a single entity within the peer group context.**

Keywords: peer-to-peer, WSN, JXTA, Java, sensor mote, Cyber-Physical System.

I. Introduction

In the current networked society, Wireless Sensor Networks (WSN's) have become very important, since they can acquire and send really useful and valuable information with a lesser cost of installation and maintenance, compared with their wired counterparts. The benefits of WSN's are not only in economical terms but also in functional ones, allowing an easy and fast deployment, allowing the mobility of nodes and dynamic network topologies, which has a big impact in temporal installations. WSN's have been traditionally related to industries such as military, oil and gas, but as their popularity has increased, they have become ubiquitous in many other fields and are already part of our daily life [1].

WSN's are able to provide detailed information about the environment, and therefore, are the key to the deployment of Cyber-Physical Systems (CPS), being their main means for data input. However, it is convenient to have some middleware that integrates access and configuration of sensor motes, overcoming the two common limitations in WSN's: that a great amount of information must be distributed and processed by constrained resources and the fact that, even though such information can be easily gathered locally, global access is often necessary.

Peer-to-peer (P2P) networks are an efficient method to distribute information in a self organized manner. These networks also provide incredible benefits in terms of the connectivity of its devices, such as scalability and reliability. For these reasons, we consider that the P2P systems might be a very good fit for developing CPS's in some environments. Many different P2P middlewares have become quite popular, helping the deployment of such networks, but between the most well-known ones with a long history, JXTA can be found [2]. JXTA is a set of open protocols specifications initiated by Sun Microsystems in 2001 that has been the system of choice for many P2P based applications, such as clipboard and file sharing between different computers [3] and a distributed e-learning system [4]. These applications can take advantage of the integration of WSN. Still under development, the latest revision at this date, version 2.7, became available in May 2011 and incorporates several long awaited functionalities, the most relevant improvements being in security and the simplification of local deployments for the testing stage.

This paper presents *jxSensor*, an abstraction layer that allows JXTA peers to interact with WSN's, facilitating the deployment of CPS's under this architecture. Even though extensive work already exists on the use of JXTA as a P2P substrate to distribute sensor information, our proposal uses a novel approach that sets it apart from others: instead of considering sensor motes as resources to be shared inside the JXTA network, they are considered full fledged peers. This small twist allows the main contributions of this paper: it takes into consideration actual two-way interaction, allowing active configuration of sensor motes, and the WSN gateways' operation remains completely in the background. From the other network participants' standpoint, sensor motes appear as a normal JXTA peer, allowing the use of some very useful JXTA capabilities which are not considered in current proposals, such as group membership across multiple gateways.

Following, we introduce the structure of this paper. First of all, Section II performs a literature review with all the relevant work on P2P-WSN integration which relies on JXTA as the underlying P2P middleware. The description of our proposed architecture and its underlying modules is included in Section

[1]This work was partly funded by the Spanish Government through projects TSI2007-65406-C03-03 "E-AEGIS", TIN2011-27076-C03-02 "CO-PRIVACY" and CONSOLIDER INGENIO 2010 CSD2007-0004 "ARES".

978-0-7695-4779-4/12 $26.00 © 2012 IEEE
DOI 10.1109/NBiS.2012.125

III. Section IV specifies the protocol for both the P2P and the WSN sensor side of the integration layer. Finally, Section V concludes this paper with a brief discussion about the paper's main contributions and future work.

II. Related Work

In this section, we provide a brief literature review on current proposals which rely on a P2P middleware to distribute WSN data. Even though proposals using different middleware exist [5], [6], we will only focus on those which specifically use JXTA, so it is possible to analyze how JXTA's particular capabilities are integrated into each approach. A full review of such capabilities and JXTA's architectural design can be found in [2].

One of the most thorough works may be found in [7]. In this paper, the authors propose a quite complex general purpose architecture, even though VANETs are the main scenario in mind, for sensor mote data acquisition over a JXTA middleware overlay operating on 2.5G/3G mobile devices. This is achieved by heavily modifying the standard JXTA distribution, thus creating an alternate version. It must be told that the paper gives much more emphasis to the mobile device integration aspect of the system. Nevertheless, from the WSN integration standpoint, the most interesting contribution is the design of the XMLSens protocol, which allows sensor motes to announce their characteristics to the WSN gateway using XML structures, in a manner similar to JXTA's lower layer protocols.

In [8], JXTA is again the middleware of choice to distribute sensor mote data, in this case in a healthcare services scenario. Sensor mote data from a Body Area Network are aggregated using a PDA, which then relies on a JXTA relay peer, acting as a proxy, to access the P2P network. However, not much emphasis is done on the WSN part of the system. What makes this proposal specially relevant in the context of a literature review is the fact that it is the only approach where JXTA peer groups are fully considered as a natural method to segment the P2P network, from both an efficiency and security standpoint.

Sharesense, a P2P environment based on JXTA for monitoring multiple WSN's is presented in [9]. At its core concept, its approach relies on integrating JXTA to jWebDust, an external Java environment, previously proposed by the same authors, that allows developing and managing WSN based applications . This environment, being Java based, is easy to integrate with the Java implementation of JXTA, and from the system's architecture, seems to be the one doing the heavyweight work as far as the sensor mote's data management and access is concerned. Thus, the system can only integrate sensor mote networks based on this particular environment. Additionally, a Google Earth-based interface is included in the demo application, allowing precise location of each sensor mote.

Nevertheless, apart from the specifics of each proposal, some common features are shared by all of them to some degree. Sensor motes are always considered as resources to be shared, and two of JXTA's core services are mostly used to access their data. On one hand, JXTA's Discovery Service is used to publish and locate available sensor motes or WSN's, relying on advertisements, XML metadata documents, to describe the sensor mote's characteristics. On that regard, each proposal provides its own custom-made advertisement structure. On the other hand, the WSN gateway acts as a single peer in the P2P substrate, relying on JXTA's Pipe Service to receive messages from other peers. This is a sound and straightforward way to integrate the JXTA middleware to a set of WSN's from a design standpoint.

However, it is important to note that, except in a single case, current proposals completely forget, or use in a very rudimentary way, another of JXTA's main architectural features, and the one which actually sets it apart from other P2P middlewares: peer groups. We deem taking JXTA peer groups into consideration important since they allow peers with similar capabilities to create a context for peer operation, segmenting the P2P network and facilitating advertisement publication and retrieval.

III. jxSensor architecture overview

The general structure of the jxSensor network at a logical and physical level is shown in Figure 1. This figure serves as an overview of how jxSensor is integrated within the context of the JXTA and WSN layers. The whole P2P-sensor integration layer is executed in a sensor gateway, a specialized hardware that acts as a bridge between the local WSN environment and external networks, such as the Internet. This aspect is imposed by the design of the WSN's. One example of such devices is the alix3d3 [10]. On that regard, on one hand, at the JXTA side of the architecture, applications which desire to obtain sensor data, usually the core behind a CPS, are executed in peers at the JXTA Applications layer. On the other hand, at the WSN side, the sensor motes execute ordinary Sensor Apps, capturing and sending data, and receiving and executing actions. These actions include changing the configuration of the application or requesting data from a sensor. In both kind of applications, their behavior is up to the developers, and does not have to be specific to jxSensor.

The hardware gateway executes the main P2P-WSN integration module, **WSNGateway**, at the JXTA's community services layer. As a result, it is easily pluggable to the standard distribution of JXTA 2.7 without the need to modify the source code, which needs the creation of a custom made JXTA version. Furthermore, it is not necessary to install a specific client component in other peers before they can actually communicate with the sensor motes. Any JXTA Application may interact with the WSNGateway module using the standard JXTA primitives for peer location and message exchange. This module is composed by the following components:

Virtual Peer: Acts as a peer entity within the JXTA network on behalf of one or a group of sensor motes, storing and managing all the information an actual JXTA peer has (for example, a unique JXTA ID, advertisements, etc.). A single WSNGateway may support several Virtual Peers, and thus,

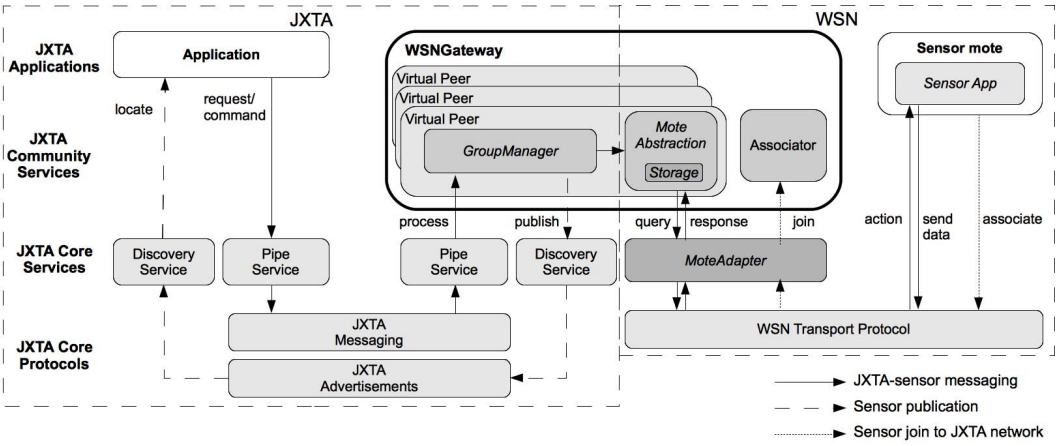


Fig. 1. Overview of jxSensor's modular architecture

several instances of this component may be executing concurrently at any given time, each one representing a separate sensor mote proxy. A set of motes may be managed using a single Virtual Peer, which groups all of them and may be interacted with a single entity. Thus, configuration commands are applied to all sensor motes at a time, and the answer of a query can be resolved by any sensor mote in the group, allowing for example to distribute the energy consumption.

GroupManager: Manages all data regarding a Virtual Peer's membership to a particular JXTA peer group. A Virtual Peer may belong to several JXTA groups, and therefore may deploy several GroupManagers.

Mote Abstraction: Processes queries received by a Virtual Peer from the JXTA network, requesting actions on sensor motes if necessary. If a query requests current data from a sensor mote, it is forwarded to the sensor mote, its response is cached to an internal storage and then sent to the requester. In the case of a query for historical data , the data is directly retrieved from the cache, without the need to query the actual sensor. Access control to the WSN, when required, is also enforced at this component.

Associator: Allows sensor motes to associate with the gateway, requesting the creation of a Virtual Peer that proxies them within the context of the JXTA network.

At a lower level layer, the **MoteAdapter** service is a pluggable component that acts as the abstraction layer that translates queries to the actual protocols accepted by Sensor Apps deployed at the WSN. This is one of the most important modules, decoupling the primitives processed at the MoteAbstraction component and the sensor motes. The implementation of the MoteAdapter service is specific for every WSN protocol and application used, providing high flexibility, since all the heavy work falls on the gateway, instead of each sensor mote.

A clearer idea of each component's functionality may be expressed with a description of a sensor mote's operation under the jxSensor architecture. Whenever a sensor mote (or a group of them) desires to be accessible from the JXTA network, first of all, its Sensor App must send its sensing capabilities and the JXTA groups it wants to be member of to the WSNGateway's Associator component. This data is transmitted using the WSN Transport Protocol and translated by the MoteAdapter service, which will forward the petition to the Associator component. As a result, a new Virtual Peer is created, which immediately joins the specified peer groups. A GroupManager component is deployed within the Virtual Peer for each joined group, which will manage JXTA data exchanges within the context of that group. The sensor mote is then considered associated to the WSNGateway, which will proxy all data exchanges with the JXTA network via the new Virtual Peer.

At this point, the sensor mote is available on the JXTA network and can be discovered and accessed using JXTA primitives as any other peer would be. Once discovered, any other peer can send requests for data retrieval or configuration commands using JXTA's standard messaging capabilities, which are managed by the WSNGateway and processed by the corresponding Virtual Peer. Whenever a request implies communications with a physical sensor mote device, it is sent to the mote by using the MoteAdapter service. Responses are then sent back to the JXTA network also via the corresponding Virtual Peer component.

Even though our proposal still follows some of the most basic JXTA integration architectural approaches used in the related literature, as described at the end of Section II, there are still some important differences. In our case, sensor motes become part of the network as peer themselves, instead of simply resources shared by the gateway. The main benefits of this decision are twofold.

First of all, if a sensor mote is considered a resource, in order to access its data the following steps must be followed:

1) Locate the sensor mote with the chosen sensing capabilities using the Discovery Service.
2) Locate the gateway, also using the Discovery Service.
3) Request the sensor mote's data using the Pipe Service.

However, under our scenario, step 2 may be completely omitted. Once the sensor mote of choice has been located, it is possible to directly send a request. Obviously, the gateway is still acting as a translator in the background, since sensor motes, due to their underlaying protocols and limited nature, are not actual JXTA peers. But it is not necessary to explicitly look it up, sending additional JXTA discovery requests.

Secondly, considering sensor motes as JXTA peers allows the full use of JXTA's Peer Group capabilities. Every sensor mote may become a member of any peer group, and most importantly, each sensor mote's membership is not restricted by its geographical location, which is the case in the only existing proposal where peer groups are even considered. Sensor motes connected to different gateways can share the same group, providing a larger flexibility in JXTA network segmentation. For example, seismographic sensor motes spread across disparate locations may be configured into a single peer group, so applications interested in only such data may locate them in a much more simpler and efficient manner. In addition, our approach allows sensor motes to become members of several groups, not just a single one.

IV. JXSENSOR PROTOCOL SPECIFICATION

This section presents the specification of the protocols used by jxSensor at the JXTA and WSN layers. On one hand, the former case details how individual sensor motes or groups of them can be located within the context of a JXTA peer group, as well as message syntax to send and retrieve information to/from them. On the other hand, the latter scenario details the message exchange between the Mote Abstraction component and the MoteAdapter service, not sensor motes. This is because, as mentioned before, the MoteApdapter service provides the abstraction layer which translates queries/responses to the format used by the non-standard Sensor Apps deployed at the WSN.

A. P2P layer protocols

jxSensor relies on three basic operations at the JXTA layer, all executed by JXTA peers: discovering existing sensor motes in a peer group, sending data requests to them and applying new configuration to sensor motes.

1) Sensor mote discovery: The JXTA presence mechanism within a peer group are Peer Advertisements, metadata XML documents which offer a basic description of a peer (JXTA ID, name and description), the means to reach them (routing information) and a list of service parameters, which usually include Pipe Advertisements, the necessary information to establish connections with each particular service being executed at that peer. In jxSensor, a Virtual Peer's GroupManager component publishes Peer Advertisements on behalf of the proxied sensor motes, thus being present within a peer group as any other normal peer from the other member's standpoint.

In order to ease the location of sensors with particular capabilities, the Virtual Peer's advertisement includes the following extra information:

- *Peer Type:* Specifies that this is actually a jxSensor Virtual Peer.
- *Capabilities:* A list with the available characteristics of the sensor mote.
  - Location: Position of the peer
  - Sensor list: The list of sensing capabilities that the mote controls, specifying each type and kind of operations that can be executed.
- Pipe: The Pipe Advertisement that will be used to contact this sensor mote, thus behaving just like any other JXTA service.

A sample Virtual Peer Advertisement is shown in Listing 1. The *Virtual Peer type* is included directly in the Advertisement's *Desc* field, using the "jxSensor" string. In this way, it is easy to search all the Virtual Peers inside a JXTA network, by looking up advertisements with this particular description. The *capabilities* and *pipe* information are included in the service parameter list, indexed using the *Svc* advertisement tag.

XML Listing 1 - Virtual Peer Advertisement

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxta:PA>
<jxta:PA xml:space="default" xmlns:jxta="http://jxta.org">
  <PID>urn:jxta:uuid-59616...7B03</PID>
  <GID>urn:jxta:uuid-425A5...5002</GID>
  <Name>sensorPeer_1</Name>
  <Desc>jxSensor</Desc>
  <Svc>
    <MCID>urn:jxta:uuid-DEADB...01105</MCID>
    <Parm>
      <jxta:PipeAdvertisement>
        <Id>urn:jxta:uuid-425A5...4704</Id>
        <Type>JxtaUnicast</Type>
        <Name>JXSENSOR-PIPE:sensorPeer_1</Name>
      </jxta:PipeAdvertisement>
    </Parm>
    <Parm>
      <jxSensor:Capabilities>
        <location>41.383333, 2.183333</location>
        <sensor>1,1,RWX</sensor>
        <sensor>2,1,RWX</sensor>
        <sensor>3,2,RWX</sensor>
      </jxSensor:Capabilities>
    </Parm>
  </Svc>
</jxta:PA>
```

The list of sensing capabilities a sensor mote controls is specified with an id, a descriptor code for its type and the permissions flags for the allowed operations. The id is used to identify a specific sensing capability in a sensor mote when a query is executed. We define the following code types: 1-Temperature, 2-Vibration, 3-Light, 4-Humidity, 5-Location, 6-Battery, and 7-Other.

The available permission flags are:

- R: Only historic data of this sensing capability, the one stored in the gateway, can be read.
- W: The configuration of the sensing capability can be modified.

- X: Data can be directly read from this sensing capability.

2) Sensor mote request data: The communication between a JXTA Peer and a Virtual Peer is done through a JXTA pipe, the one that can be found in the Virtual Peer Advertisement, as has been previously detailed in *sensor mote discovery* section. The Virtual Peer listens at its Pipe and receives messages. In general, the information received from a pipe is binary data, but in jxSensor we have decided to use JXTA Messages in order to improve compatibility and readability. JXTA Messages can be encoded as XML documents formed by name-value couples, under a namespace. Additionally to the information that can be added in a JXTA Message, it contains extra information that is internally needed by JXTA pipes. But for simplicity and readability, those information is not shown in examples messages bellow.

The *sensor mote request data* operation consists in querying data that has been captured from one sensing capability in a Virtual Peer. This data can be directly read from a sensor mote or obtained from the historical cache stored at the Virtual Peer.

In the first case, a query JXTA Message requesting a *read current data* action has to be sent to the Virtual Peer. An example of this message is shown in Listing 2. The *operation* tag specifies the type of operation which in this case is X. The *id* and *timeout* tags specify the sensor id and the maximum time the Virtual Peer is going to wait for the answer of the sensor mote before sending an error response. Furthermore, the *transaction id* has to be added in order to match the query with the response at a later time.

XML Listing 2 - read current data msg

```
<jxSensorMessageQuery>
  <operation> X </operation>
  <id> 3 </id>
  <timeout> 5 </timeout>
  <transactionId> 93872088 </transactionId>
</jxSensorMessageQuery>
```

In the second case, a query JXTA Message requesting a *read historic data* action has to be sent to the Virtual Peer. This message shares the same structure as the one before but with the difference that is a different operation, R. An optional tag *time* can be added specifying the time when the measure was taken. If not specified, the most recent stored data is sent.

Two possible kind of responses can be obtained from these queries: the requested data or an error. If the data could be accessed, it will be received in a message which contains the transaction id, the data from that sensor and the time when that data was measured. If an error occurs, a different message will be received with the transaction id, and the error. The possible errors are operation not allowed, timeout and no data. The first case happens when an operation is performed and we do not have the permissions to do so, whereas the second case when there is no data for that period of time, and the last case, when the response was not retrieved after the specified time.

3) Sensor mote apply configuration: The third operation consists in executing a command to change the configuration of a sensing capability from a sensor mote. In general, all sensing capabilities allow setting how often a measure is taken and sent to the gateway (in seconds), but other configurations can be set if implemented in the *MoteAdapter* service. The message sent is similar to the one of the *request data* operation but containing how often a data will be read from the sensor mote and sent to the WSNGateway. The answer is similar to the one of the previous operation, but only the time when the operation was done is returned. If an error is produced, the same error message is received.

B. Sensor mote layer protocols

The format of messages exchanged between the sensor motes and the gateway is defined by the Sensor App running on sensor motes, which can be different in each scenario. The MoteAdapter service is responsible of translating the message format to the operations defined by jxSensor, which are invoked on the MoteAbstraction component, decoupling the messages sent over the WSN and the ones of JXTA. This allows to use the most efficient implementation for each operation at any case. Thus, this subsection focuses in the message exchanges between the Mote Abstraction component and the MoteAdapter service. The operations that should be implemented at the MoteAdapter service are:

- **Associate** a sensor mote to a gateway, and join to some groups.
- **Request data** from a sensor mote.
- **Apply configuration** of a sensor mote.
- **Send periodic data** from a sensor mote.

1) Association: The first operation a sensor has to do to belong to the JXTA network is associating with a WSNGateway. In this operation the sensor should describe itself in order to allow the WSNGateway to spoof its identity. The information required is its location, capabilities and available sensors, and groups and permissions. This association can be done in two ways: initiated by the sensor mote, or manually set at the WSNGateway.

If the sensor mote sends an association message to the WSNGateway, the *MoteAdapter* service translates it and sends the message to the *Associator* component, which will perform the operation and answer with an ACK message. An example of an association message is shown in Listing 3. The first line is the id of the sensor mote. The second line is the location of that sensor mote. The next two lines are the groups the Virtual Peer will belong and the last three lines describe three sensing capabilities and their permission. The first element is the id of the sensing capability, the second is the type of sensing capability (see Section IV) and the rest are the permissions for each group. For instance, the first sensing capability is a temperature one, allowing all the operations for the group A but not allowing to change the configuration of the sensor from group B.

On the other hand, information can be directly set at the WSNGateway, without the need of a message exchange. This is achieved using a console terminal session command. This method allows two powerful features. The first one is that

a live WSN could be integrated to JXTA without modifying anything at the sensor motes. Only the gateway has to be modified with the JXTA software and jxSensor services and MoteAdapter service adapted for the internal WSN protocol. Then the information of the motes can be added manually to the WSNGateway by this association procedure, and the WSN is directly integrated into JXTA. The second interesting scenario is when you have a WSN network where there is just one way communication between sensor motes to the Gateway but not the opposite. It is possible to add sensor motes manually, only allowing "R" permission to its sensing capabilities. Then, the JXTA network can take advantage of the information of such sensor motes although there is not really a way to directly query them.

XML Listing 3 - Association missage

```
5438764863;
41.383333, 2.183333;
A,urn:jxta:uuid-425A5C703CD5454F9C03938A0D65BD5002;
B,urn:jxta:uuid-425A5C703CD5454F9A49857DBF78765F002;
1,1,RWX,RX;
2,1,RWX,RX;
3,2,RWX,RX;
```

2) Request data and apply configuration: The MoteAdapter service, whether it is permitted based on the sensing capabilities, can send two types of petitions to the sensor mote. A query for a sensing capability, or a command to modify the configuration. The messages sent in the JXTA level have been described in Sections IV-A2 and IV-A3. These messages are going to be exchanged between the *Mote Abstraction* component and the *MoteAdapter* service.

3) Send periodic information: Sensor Apps usually send periodic messages with data captured from the mote's capabilities. Under our architecture, this information is sent to the Mote Abstraction component. In this way, historic data can be queried from JXTA peers without sending additional messages to the motes, and therefore not increasing its energy consumption. The message format is similar as the one of association, the first line is the id of the sensor mote, the second line is the time when measures were taken and the last three lines contain the id of the sensing capability, and the captured data.

V. Conclusions

In this paper we have presented jxSensor, an abstraction layer that allows JXTA peers to interact with WSN's. Through this integration, it is possible to provide a feasible solution to some of the WSN limitations on regards to data processing and dissemination.

Our proposal uses novel approaches that set it apart from others. First of all, each sensor mote, or an aggregation of them, can be treated as a single peer within the context of a peer group, allowing JXTA peers to communicate to them transparently. This allows sensor motes to use the versatility offered by JXTA Peer Groups to segment the network, such as enabling sensor motes to limit their presence to particular groups or allow different actions depending on the groups. Peer groups are also the main gateway to deploying security in JXTA. Secondly, an abstraction layer is set in the gateway. This permits to incorporate deployed sensor motes into CPS's without modifying its software. Thirdly, two-way communication is permitted, allowing both reading data from the sensor motes and configuring them. Nevertheless, networks with one way communication (sensor mote to gateway) are also supported, which is very useful in smart metering scenarios. Fourth, the concept of Virtual Peer is created. This allows a group of sensor motes to be seen as a single peer in the JXTA network, which allows to integrate WSN with a great quantity of sensors without clogging up the JXTA network. Finally, historic data is stored in the gateway in order to backup the information sent from the sensor motes and also to reduce readings by the sensor motes, optimizing resource utilization.

Our future work includes finishing the jxSensor implementation and testing it in a real scenario. We have sensor motes installed in civil infrastructures, such as bridges, which monitor the healthiness of the infrastructure. Since the network is already deployed, our goal is to incorporate the sensor motes to a JXTA network without modifying them. The flexibility and abstraction defined in jxSensor, via the *MoteAdapter* service, allows this kind of adaptation. Therefore, just the gateway software has to be updated. Other future plans are the study on how to improve the association procedure to allow the usage of credentials and secure Peer Groups, the creation of an application to monitor sensor motes information from a mobile phone using JXME[11] and finding ways to replicate historic data between different gateways to improve its availability.

References

[1] M. Dohler, I. Vilajosana, X. Vilajosana, and J. Llosa, "Smart cities: An action plan", in *Smart City Expo & World Congress. Dec*, 2011.

[2] Sun Microsystems Inc., "JXTA v2.0 protocols specification", 2007, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[3] smokindoug, "Cut and paste with JXTA (clipboard manager)", 2009, http://kenai.com/projects/candpjini.

[4] K. Matsuo, L. Barolli, J. Arnedo-Moreno, F. Xhafa, A. Koyama, and A. Durresi, "Experimental results and evaluation of smartbox stimulation device in a P2P e-learning system", 2009, pp. 37–44.

[5] W. Song, S. Kim, S. Seok, and D. Cho, "A peer-to-peer environment for monitoring multiple wireless sensor networks", in *Proceedings of 18th Internatonal Conference on Computer Communications and Networks, 2009*, 2009, pp. 1–6.

[6] S. Seok, N. Kim, D. Choi, and W. Song, "An Implementation of P2P System for Sharing Sensory Information", in *Management Enabling the Future Internet for Changing Business and New Computing Services*, vol. 5787 of *Lecture Notes in Computer Science*, pp. 191–200. 2009.

[7] S. Krco, D. Cleary, and D. Parker, "Enabling ubiquitous sensor networking over mobile networks through peer-to-peer overlay networking", *Computer Communications*, vol. 28, no. 13, pp. 1586 – 1601, 2005.

[8] B. Lim, K. Choi, and D. Shin, "A JXTA-based Architecture for Efficient and Adaptive Healthcare Services", in *Information Networking. Convergence in Broadband and Mobile Networking*, vol. 3391 of *Lecture Notes in Computer Science*, pp. 776–785. 2005.

[9] A. Antoniou, I. Chatzigiannakis, A. Kinalis, G. Mylonas, S. Nikoletseas, and A. Papageorgiou, "A peer-to-peer environment for monitoring multiple wireless sensor networks", in *Proceedings of the 2nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. 2007, pp. 132–135, ACM.

[10] PC Engines, "Alix3d3", 2007, http://pcengines.ch/alix3d3.htm.

[11] Sun Microsystems, "JXME", 2003, https://jxta-jxme.dev.java.net.

Appendix D

Paper: Balancing Power Consumption in IoT Devices by Using Variable Packet Size

M. Domingo-Prieto, B. Martinez, M. Monton, I. Vilajosana-Guillen, X. Vilajosana-Guillen, J. Arnedo-Moreno, "**Balancing Power Consumption in IoT Devices by Using Variable Packet Size**" (CISIS14). 2014, pp. 170-176. IEEE Press. Indexed in SCOPUS.

_:

Balancing power consumption in IoT devices by using variable packet size

M. Domingo-Prieto*‡, B. Martinez‡, M. Montón‡, I. Vilajosana-Guillen‡, X. Vilajosana-Guillen†‡ and J. Arnedo-Moreno†
*Internet Interdisciplinary Institute (IN3), Universitat Oberta de Catalunya
C/ Roc Boronat, 117, 7 flour 08018 Barcelona, Spain
Email: mdomingopr@uoc.edu
†Computer Science, Multimedia and Telecomunication Department, Universitat Oberta de Catalunya
Rambla del Poblenou 156, 08018 Barcelona, Spain
Email: jarnedo, xvilajosana@uoc.edu
‡WorldSensing, C/ Aragó 383, 4th flour, 08013, Barcelona, Spain
Email: mdomingo, bmartinez, marius, ivilajosana, xvilajosana@worldsensing.com

***Abstract*—Currently, IoT devices are becoming more and more popular, being deployed in different scenarios such as monitoring the power consumption of a house or the state of an outdoor parking spot. These networks tend to be densely populated by a huge amount of sensors that send really short messages. Given these characteristics, their main problem is the great variability and unpredictability of battery lifetime, when they cannot be plugged to a power outlet. In this paper, we analyze the mote behaviour on a real IoT network and use the extracted data to propose a mechanism to distribute the power consumption more equally between all motes, regardless the number of messages each one sends. This new proposal decreases the numbers of interventions required to replace batteries, minimizing costs and increasing network lifetime.**

***Index Terms*—IoT; WSN; address clustering; Huffman coding; power consumption**

I. Introduction

The Internet of Things (IoT) is producing a profound transformation to global industry. Hardware miniaturization, communications standardization and analytics are allowing connecting more and more objects. The IoT devices, the growing analytics and big data capabilities have the potential for enhancing industry and services operations in multiple sectors. Undoubtedly the ubiquitous presence of IoT devices combined with their sensing and communicating capabilities is enabling the opportunity to lower maintenance and operational costs in an unprecedented scale.

Operational costs are especially relevant in critical infrastructures, where timely information can be used to optimize maintenance and minimize unplanned downtimes. The IoT is in pure expansion and that means connecting more and more objects to the Internet. However, there are still some limitations, mainly the power consumption of the wireless sensors, the ease of handling the communication interfaces and the global coverage.

In the recent years, research activities addressed to increase the lifetime of IoT devices have augmented. Optimizing the battery life of multi-hop networks has been a hot topic and it has usually been addressed through data aggregation by intermediate hops [1] or by using fare routing protocols [2]. Others addressed this problem by reducing the consumption of every mote by compressing the packets sent [3] or by using a more efficient MAC protocol [4].

However few of the presented approaches consider the application itself and how the monitoring service is provided. Most monitoring applications cover large areas where different activity zones are observed. Common energy optimization approaches use the same policies independently of the observed level of activity in a specific area. Even though, from an individual device standpoint, battery life may be extended using the described approaches, some devices will still deplete their power quicker than others.

Differences in battery usage across the same network becomes a big factor if we take into account that battery replacement interventions must be scheduled. On that regard, it must be taken into account that each intervention costs money and time, and in some scenarios, such smart cities, it may actually become a hassle to citizens. In a scenario with big differences in battery usage across devices, interventions become more complex, since only some devices have their battery replaced and a log must be kept. And they still become frequent, since the devices which were not replaced will deplete their battery sooner now. Of course, it may be possible to just replace all batteries regardless of their power level at each intervention, but that would be wasteful, from both a cost and environmental perspective. Therefore we consider that, in addition from increasing battery life, it is also very important to homogenize battery consumption across all devices, in order to facilitate interventions and reduce maintenance costs.

Given this scenario, in the present work we present a novel approach where we consider areas with different levels of activity. To this end, we suggest and analyse two communication strategies based on variable packet size mechanisms which will help to increase the global lifetime of the network. We back our proposal by studying the behavior of a real smart parking deployment.

The paper is structured as follows. First of all, in section

II, we analyse the power consumption of a real network. Section III evaluates the effect of variable size packets in power consumption. An assessment of two possible solutions based on variable packet size is detailed in section IV. Finally, section V concludes the paper and proposes further work.

II. Studying power consumption of IoT networks

IoT devices are small devices with limited processor, memory and battery. Their purpose is to capture data from the environment, wirelessly sending messages to a central sink, which is generally called gateway. On that regard, IoT applications can be divided into two groups, depending on how messages are produced: one where motes report data periodically, at fixes intervals, and another where motes report data asynchronously, reacting to certain triggers. House power consumption monitoring is an example of the former, since reports are usually generated at fixed intervals, whereas smart parking would be an example of the latter, where motes send messages only when the state of a parking spot changes.

In the first group, messages are sent in determined periods of time, for instance, one report message per hour. Therefore, the power consumption of all the motes in the network is quite similar, since all motes will send the same amount of messages, overall. As a result, it is easy to schedule battery replacement interventions, since the battery level of all motes will decrease similarly as time passes. However, in the second group, the consumption of the motes, and therefore, battery lifetime, depends on the number of state changes, which can be very different between motes installed at different spots.

A scenario where the lifetime of every mote is very different complicates the process of having all the devices up. This process is worsened when the complexity of interventions to replace batteries must be taken into account. Following with the smart parking example, every time a device must be accessed, it has to be planned in advance to warn about the intervention on the street, which costs money and has an impact in the citizens life. For these reasons, minimizing the difference between IoT devices lifetime and increasing the global network lifetime could be a very important achievement.

In order to quantify this behavior in a real scenario, we have collected real data from an installation of 500 outdoor parking sensor motes, for more than 4 months. The data has been provided by [5] from one of the company deployments in a real life scenario, used to control available parking spots on the streets. Figure 1 shows the average number of messages motes send per day. It can be seen how most of the sensors send between 5 to 15 messages per day. However, some sensors send more packets, up to 30 packets per day. This distribution matches to a Poisson distribution with $\lambda = 9.96$ that can be used for further simulation models.

From this data, it could be detected how, in a real scenario, different sensors send a different number of messages, which will result in different battery lifetimes. In order to quantify how these different number of messages sent impact on power consumption, a simulation was run using a power consumption model for WSN devices [6], [7].

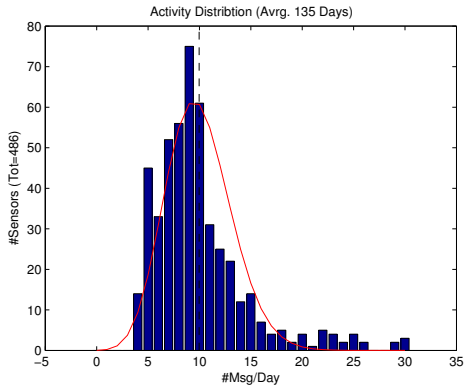


Fig. 1. Number of messages distribution per mote

A. Models of Power Consumption in WSNs

The power required to operate a device (P_{DEV}) can be broken down into three parts (or building blocks): energy required for data sensing (acquisition, P_{ACQ}), for application (data processing, P_{PRC}), and communication (networking, P_{NET}). This model is based on an atomic breakdown of each building block, where the instantaneous power consumption is integrated over the duration of the corresponding task, and averaged out its characteristic temporal scale (period of repetition). Yet, putting these systems together requires a clear understanding of their cyclic behavior.

$$P_{DEV} = P_{NET} + P_{ACQ} + P_{PRC} \tag{1}$$

The main contribution to the acquisition block is due to the sensor itself. Driven by the application demands, when a new sample is required the sensor is turned on and, after a fixed time, given by the sensor characteristics, the sample is recorded. The charge required by the sensor in the data acquisition stage is computed with the time the sensor must remain active and the current consumption of the sensor in the corresponding state, but this charge is constant for each record. Then, the acquisition current can be easily averaged for the sampling period T_{SMP} and, as in most applications the sampling process repeat cyclically, this average current in one period represents directly the global average consumption (Eq. 2).

$$\bar{I}_{ACQ} = \frac{Q_{SMP}}{\bar{T}_{SMP}} \tag{2}$$

Similarly, the average current in the communications block can be obtained by dividing the total charge required to send a radio message by the time between consecutive events (Eq. 3). The time between messages is fixed for periodic reporting applications but should be properly characterized

through averaging or other statistical methods in a general case.

$$\bar{I}_{NET} = \frac{Q_{MSG}}{\bar{T}_{MSG}} \tag{3}$$

Finally, the processor current load can be considered as constant, independently of the network and sampling key parameters. [1].

Combining the three contributions and assuming that all components operates at the same voltage level, the total average current can be modeled as shown in Eq. 4. It should be noted that constants A, B and C only depend on the particular choice of technology for the MCU, radio and sensor, respectively.

$$\bar{I}_{DEV} = A + \frac{B}{\bar{T}_{MSG}} + \frac{C}{\bar{T}_{SMP}} \tag{4}$$

B. Calculating Lifetime depending mote's activity

The power model described in the previous subsection has been used to identify how the different number of messages sent by motes exactly impacts on battery lifetime. The results can be seen in figure 2, which shows how battery level decreases as years pass, depending on the number of messages that a mote sends. The different ranges in this figure show the percentage of charge of the battery in 10% increments. Meaning that the left-most area represents that the remaining battery percentage is between 100% and 90% and the right-most one is the remaining battery percentage when the charge is between 10% and 0%, a flat battery.

This figure has been calculated assuming that the sensor mote is monitoring the environment every 4 seconds and battery capacity is 6Ah. Additionally, the size of the messages sent has been fixed to 8 bytes. In this figure, it can be seen that the battery of a sensor mote really depends on the number of messages they send. Taking the maximum and minimum number of messages from Figure 1, it can be seen how sending 5 messages per day the battery will last 10.5 years, while if it sends 30 it will last only 6 years. This means that after 6 years of the installation some interventions to replace batteries have to be scheduled, and these interventions will be required during 4.5 years, until battery of all motes are replaced.

An interesting result that can be extracted from the figure is that lines are not straight. This is because battery consumption is not only caused by message transmission. There are also other components, such as the one that monitors if a parking spot is free or occupied, that are periodic and don't vary with the number of messages sent.

Another interesting result that was obtained after analyzing the data is that sensors that are physically near have similar activity, which can be classified in geographical separated regions. This analysis is shown in Figure 3. We considered a classification in which *High* activity means more than 20 messages per day, *Standard* between 12 and 20, and finally

[1]This term should be handled with care in applications requiring high processing and filtering on each sample

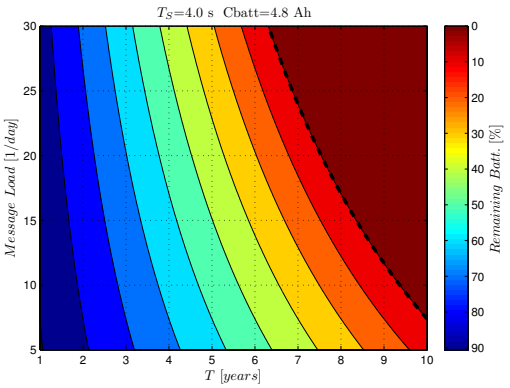


Fig. 2. Mote's battery duration depending the number of messages sent

Low represents less than 12 messages per day. In this figure is seen how motes with high activity are located in the top of the figure, while the ones with low activity are mainly located at the borders.

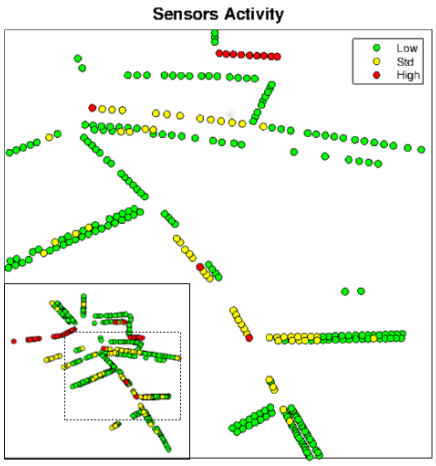


Fig. 3. Heat map of messages sent

C. Every bit matters

Figure 4 shows the lifetime for the motes sending max/mean/min messages depending the bytes sent. It clearly shows the importance of keeping packets with a small size in order to extend battery lifetime. Just saving a few bits can have a big impact.

These results encourage building an ad-hoc implementation of the messages, in order to minimize their size, instead of following standards. This is nothing new, since, in general, it is expected that use standards is going to add overhead.

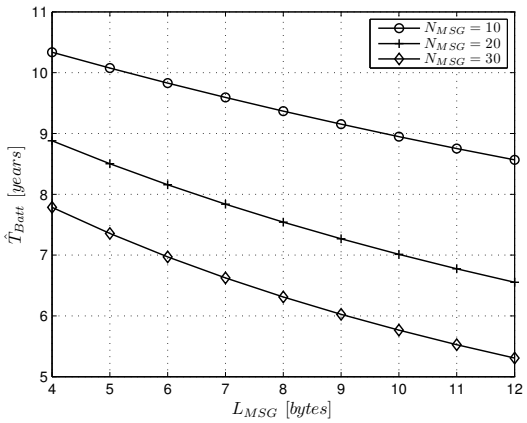


Fig. 4. Mote's battery duration depending the bytes sent

However, in cases where the lifetime is too crucial, it may be better to optimize as much as possible the internal protocol. Interoperability is sacrificed for a greater battery lifetime, and thus, decreased costs.

These optimizations can be in the form of avoiding Byte-padding small fields, like the *Packet identifier* field. This field, generally used to identify the packet type, is mapped to an 8 bit field, when generally, no more than 4 or 16 different packet types are in use. The same happens with the *Sequence number* field, which is incremented in every packet sent by the sensor in order to detect packet loss. Usually, this field is mapped to 8 bits, behaving as a 0 to 255 counter, but it is easy to see that only 4 bits (or even less) is enough to detect whether some packets are lost in the communication.

Additionally, most of the implementation of the IoT applications hide devices under a translation layer behind the gateway. Therefore, using handmade optimized internal protocols seems a good approach. Though this adds some complexity to the gateways, these devices are powerful enough to support these requirements.

III. Evaluating effect of variable size packets in power consumption

The results obtained in the previous section show that IoT networks based on events can have a very unbalanced power consumption distribution between motes. This makes it hard to predict when motes should be replaced, and what is worst, require many interventions in long periods of time, increasing maintenance costs.

In order to establish a better balance in battery usage between different motes, despite the different number of messages sent, we propose to implement variable packet size. Motes which are prone to send messages more frequently will use shorter packets in such a manner that, overall, the transmission time, and therefore battery depletion rate due to radio use, will be similar across the network. However, before even taking into consideration which approaches should be studied to decrease packet length, it is important to study the potential benefits of this strategy. To achieve this end, we will study the overall impact a given decrease in packet length has on the system.

Using the same model in Figure 2, but modifying the size of the messages depending the number of messages that are sent in a linear way, Figure 5 is generated. The size of messages in motes more prone to transmit is decreased up to 25%, while the size of messages for motes that send few packets is increased up to 25%. It has to be pointed out that the total amount of information sent over the network doesn't change, just the distribution between motes. In this new figure, it can be seen how lines are more vertical, showing that the power consumption of all motes is more similar. The black dotted line is drawn to ease this comparative, and represents 0% of battery on Figure 2. Additionally, the shortest lifetime of a mote in the network is increased, meaning that the whole network will last longer. Battery duration for the most active sensors has increased from 6 to more than 7 years. This would help battery replacement operations, doing this replacement once every 7 years for the whole installation, with no other interventions in between.

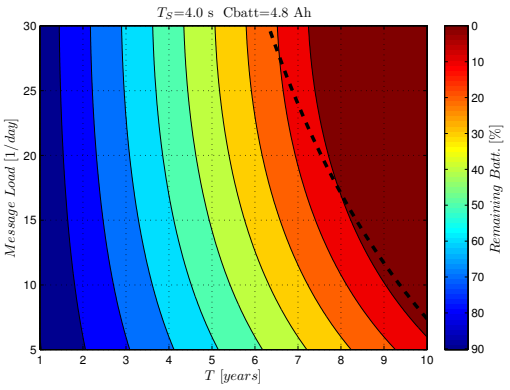


Fig. 5. Mote's battery duration for variable N bits depending the number of messages sent

These results show that the strategy of using variable packet size has the potential to achieve our objective of balancing the power consumption between motes.

IV. Evaluating the implementation of variable packet size

Once the potential benefits of a decreased packet length approach has been assessed, now we can analyze in much more detail which methods can be used to achieve such reduction. On that regard, we have studied two approaches using datasets extracted from the IoT device deployment. One one hand, Huffman coding [8], given its popularity. On the other hand, address clustering [9], since it has been used in other areas to improve the efficiency of systems. But, as far as we know, it

has not been applied before to provide variable packet size to balance consumption of IoT devices. Thus, this analysis is of some interest.

A. Huffman coding

Huffman coding is an entropy encoding algorithm used for lossless data compression. It uses variable-length code table for encoding source symbols. Huffman codes are established by storing the symbols of the alphabet in a binary tree according to their probability. As the tree is traversed from root to leaf, the code grows in length. Thus, symbols which occur frequently are stored near the root of the tree and have the shortest codes. While less frequent used symbols have longer codes.

Additionally to providing a variable-length codification Huffman coding also compresses data. How much data can be compressed is quantified by entropy, defined as a measure of the diversity or randomness of the data as:

$$H \equiv -\sum_{i=1}^{n} \frac{m_i}{m} \log \frac{m_i}{m}$$

Entropy gets a value of zero when all values in the data are the same and a maximum value of $\log m$ when all values are different. We normalize the entropy as $H/\log m$.

We calculated the entropy of all gathered packets during a period of time from all motes, as an indicator of the redundant information present in the transmissions. The resulting entropy, applying the previous formula to the data-set, gives a result of $H = 5.54$ bits per byte. This result can be seen as a medium-high degree of randomness in all packets. Therefore, it could be difficult for habitual compressing methods to achieve a good compression rate.

We used a basic Huffman coder implementation to build a Huffman codification for our data set in order to demonstrate that a variable packet size is implementable easily.

Figure 6 shows the lifetime of motes depending on the number of messages sent when using Huffman encoded messages. The dotted black line is drawn to ease a comparative with Figure 2, and represents 0% of battery on that figure. It is possible to see how lines are a bit more vertical, as theoretically expected, showing that the power consumption of all the motes has been balanced just a bit. Lines are not as vertical as in figure 5, where variable encoding was ideal. Additionally, it can be seen how the 0% line has moved to the right, meaning that the power consumption of all motes has decreased. This effect is produced because Huffman coding in addition to do variable encoding, does compression.

From this analysis, we can see that, using Huffman coding, the lifetime of the network could be improved. However, from our observations, compression seems to have a bigger role in the improvements, rather than consumption balance.

Nevertheless, this is just a first and fast approach, since the implementation of the Huffman algorithm in a real deployment is not trivial. The drawback of this approach is that Huffman coding requires analyzing the data prior to the compression,

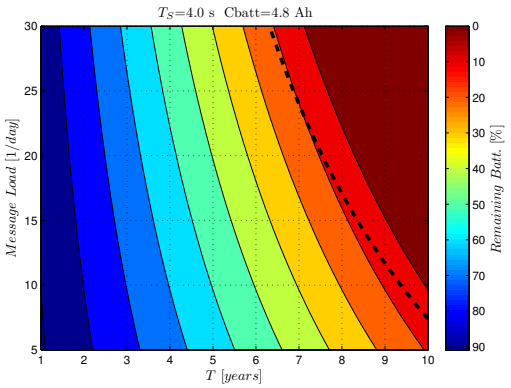


Fig. 6. Mote's battery duration for Huffman encoded messages depending the number of messages sent

TABLE I
CLUSTERED ADDRESSES

Cluster	motes in cluster	Number of bits	% occurrence
High	25	5	5.2
Std	82	7	16.9
Low	377	9	77.9

in order to build the symbol tree. This requirement would involve a non-optimized operation of the system (with no compression) for a certain amount of time (i.e. 3 months), collecting all data packets produced by the whole system. Then, it would be necessary to train the Huffman encoder with this collected data in a laboratory and disseminate the Huffman table to all motes in the deployment. This operation is totally unfeasible in industrial and large deployments, where access to all sensors can be very complicated or even impossible due to the environment or legal circumstances.

B. Address clustering proposal

Another approach to variable packet size is address clustering. As seen at the end of Section II, sensor activity can be grouped in different clusters. These clusters can be labeled as "High", "Std" and "Low" activity. In order to save length on the packet, it is possible to perform a variable-length source address encoding for the whole system. In this case, we can redefine the header in order to store the cluster number (using only 2 bits i.e. up to 4 clusters) and the header source address field to be variable depending on the size of the cluster and the cluster that each sensor belongs.

As can be seen in Figure 3, is easy to see that a natural clustering is possible because near sensors have similar behavior. In this work, this behavior is obtained studying the collected data, but it would be possible to analyze streets *before* whole deployment is done to know what clusters can be used.

In Table I we show the whole system clustered in 3 different address lengths, using less address bits for High activity sensors and more bits for the ones with Low activity.

Figure 7 shows the lifetime of motes, depending on the number of messages sent when using clustering addresses. Again, the dotted black line is drawn to ease the comparative with Figure 2, and represents the 0% of battery in that figure. In this new figure, it is shown how there are three very differentiated sections, which match with the three kinds of clusters. The top group represents the cluster "High", the middle represents the "Std" and the bottom the "Low".

Additionally, the figure shows how all lines have a similar slope to the original one, but shifted. The top cluster has been moved to the right and the bottom cluster has been moved to the left. This means that the power consumption of all the motes has been balanced. Motes that send more messages have improved their lifetime, while motes sending few messages have worsened it. Also, it is possible to see how the lifetime of the motes from the middle cluster does not change. This is an expected result, since the method balances consumption between motes, but does not improve the battery consumption of all individual motes. In this case, the lifetime of the whole network is not as high as it would be in the ideal case, but it is just a bit worse. The total gain is similar to applying Huffman coding.

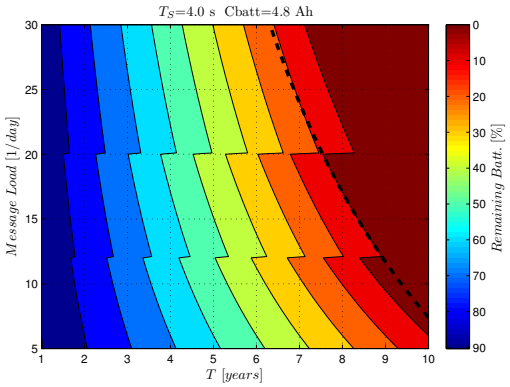


Fig. 7. Mote's battery duration for messages encoded using address clustering depending the number of messages sent

Figure 8 shows more graphically how the balance of the power consumption is achieved in all the motes in the network. After applying address clustering, motes that originally had a lifetime between 6 to 7 years have moved the 7 to 8 year range. And the opposite, motes that originally were expected to last between 9 to 10 years now last between 8 to 9 years.

These results show that using cluster addressing also improves the network lifetime by balancing mote's power consumption. In comparison to the Huffman coding approach, address clustering is much more easy to implement and provides similar benefits. Additionally, the clustering strategy used for addressing could be used to optimize other fields of the packet. Therefore, address clustering would be our choice when implementing a mechanism to balance power consumption between motes in the real world.

V. Conclusion

In this paper, the global battery consumption of a real deployment of an IoT network has been analyzed. From this analysis, it has been found that not all motes behave in the same way inside a network. Some motes send messages more often than others, which results in having motes with different power consumption and battery life. As a result, there is an unwanted side-effect as far as network maintenance is concerned: it decreases the time between required interventions to replace batteries and makes them more expensive and inconvenient, since interventions will be required in shorter periods of time. Additionally, we have found that motes physically near behave in a similar way, which allows a physical classification of motes depending on their activity.

From this initial study, we have evaluated decreased packet size as a viable solution to differences in power consumption across the network. On that regard, we assessed two approaches: Huffman codes and address clustering. This assessment shows that lifetime for the most power hungry motes is similar in both cases. However, whereas address clustering shows a better power balance, Huffman has better overall improvements. This is because Huffman coding, in addition to using variable size packets, also compresses data. However, the implementation of Huffman coding does not seem a trivial task in a real deployment, whereas the address clustering is straight forward.

For these reasons, we conclude that using address clustering is the best choice to improve the periodicity of battery replacement interventions, saving time and money.

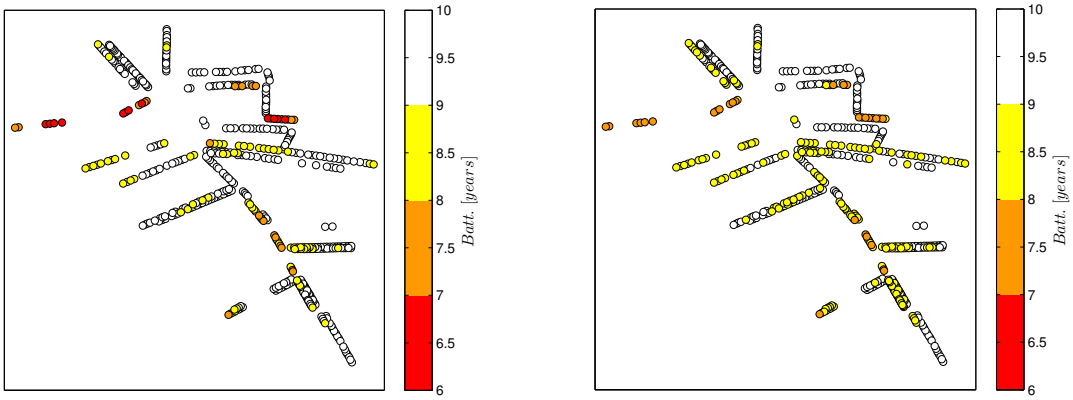
Once the viability and benefits of using address clustering have been evaluated, further research includes studying if it is possible to apply this approach when encryption is used. Also, a mechanism to send different types of packets depending the mote's remaining battery is going to be researched. Finally, this approach will be implemented and tested it in a real deployment.

Acknowledgments

This research is supported by the FP7-Calipso European project(FP7-ICT-2011.1.3, PN:288879) and the European Commission with contract INFSO-ICT-317826 (RELYonIT). Marc Domingo-Prieto is funded by the Generalitat de Catalunya under the TALENT Empresa grant. Xavier Vilajosana is funded by the Spanish Ministry of Education under a Fullbright-BE grant (INF-2010-0319). Borja Martinez is funded under grant Inncorpora-TU-1558 from the Ministry of Industry. Màrius Montón is funded under the Torres Quevedo Program PTQ-11-04864. Joan Arnedo-Moreno is partly funded by the Spanish Government through projects TIN2011-27076-C03-02 (CO-PRIVACY) and CONSOLIDER INGENIO 2010 CSD2007-0004 (ARES).

References

[1] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, vol. 2, 2005, pp. 8–13 Vol. 2.



(a) Original

(b) After applying address clustering

Fig. 8. Heat map of motes consumption

[2] C. Schurgers and M. Srivastava, "Energy efficient routing in wireless sensor networks," in *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*, vol. 1, 2001, pp. 357–361 vol.1.

[3] J. Kolo, L.-M. Ang, S. Shanmugam, D. Lim, and K. Seng, "A simple data compression algorithm for wireless sensor networks," in *Soft Computing Models in Industrial and Environmental Applications*, ser. Advances in Intelligent Systems and Computing, V. Snel, A. Abraham, and E. S. Corchado, Eds. Springer Berlin Heidelberg, 2013, vol. 188, pp. 327–336.

[4] T. van Dam and K. Langendoen, "An adaptive energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, ser. SenSys '03. New York, NY, USA: ACM, 2003, pp. 171–180. [Online]. Available: http://doi.acm.org/10.1145/958491.958512

[5] "Worldsensing," 2010, http://www.worldsensing.com.

[6] X. Vilajosana, Q. Wang, F. Chraim, T. Watteyne, T. Chang, and K. Pister, "A realistic energy consumption model for tsch networks," *IEEE Sensors*, 2013.

[7] D. Zordan, B. Martinez, I. Villajosana, and M. Rossi, "On the performance of lossy compression schemes for energy constrained sensor networking," *[early version: arXiv:1206.2129]. Accepted for Publication. ACM Transactions on Sensor Networks*, 2014.

[8] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[9] A. Macii, E. Macii, and M. Poncino, "Improving the efficiency of memory partitioning by address clustering." in *DATE*, 2003, pp. 10 018–10 023.

Appendix E

Paper: Towards secure mobile P2P applications using JXME

M. Domingo, J. Arnedo, J. Herrera, and J. Prieto, (2012). "**Towards secure mobile P2P applications using JXME**". Journal of Internet Services and Information Security. Núm. 1/2, Pág. 1-21. ISSN.2182-2069

Towards secure mobile P2P applications using JXME

Marc Domingo-Prieto
Internet Interdisciplinary Institute (IN3)
Universitat Oberta de Catalunya
Carrer Roc Boronat, 117
08018 Barcelona, Spain
mdomingopr@uoc.edu

Joan Arnedo-Moreno*
Internet Interdisciplinary Institute (IN3)
Universitat Oberta de Catalunya
Carrer Roc Boronat, 117
08018 Barcelona, Spain
jarnedo@uoc.edu

Jordi Herrera-Joancomartí
Escola Tècnica Superior d'Enginyeria
Universitat Autónoma de Barcelona
Campus de Bellaterra, Spain
jherrera@deic.uab.cat

Josep Prieto-Blázquez
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya
Rambla del Poblenou, 156
08018, Barcelona, Spain
jprieto@uoc.edu

Abstract

Mobile devices have become ubiquitous, allowing the integration of new information from a large range of devices. However, the development of new applications requires a powerful framework which simplifies their construction. JXME is the JXTA implementation for mobile devices using J2ME, its main value being its simplicity when creating peer-to-peer (P2P) applications on limited devices. On that regard, an issue that is becoming very important in the recent times is being able to provide a security baseline to such applications. This paper analyzes the current state of security in JXME and proposes a simple security mechanism in order to protect JXME applications against a broad range of vulnerabilities.

Keywords: peer-to-peer, security, JXME, JXTA, distributed systems, Java, J2ME

1 Introduction

Peer-to-peer (P2P) applications have become highly popular due to its great potential to scale and the lack of a central point of failure. Slowly, they have evolved from simple file-sharing environments, such as Gnutella [10], to more complex ones such as GIS (Geographic Information Systems) or e-learning [33, 17]. However, currently, Internet has become witness to the transition from a desktop-centric environment towards one based on the ubiquity of mobile devices [26]. Therefore, it was natural that the next step in the evolution of P2P applications would be following this trend [27], since mobile environments are based on node autonomy and decentralization, just like P2P.

There are different platforms that allow programmers to develop mobile P2P applications [15, 8], among which JXME [29] can be found, the mobile version of the well known JXTA platform [28]. The JXTA specification defines a set of generic protocols which allow peers to communicate and publish, find or consume remote resources, independently of the actual transport layer and the implementation language. Such protocols are generic enough so they are not bound to a narrow application scope, but adaptable to a large set of application types. Nevertheless, JXTA was designed with desktop devices in mind. Thus, JXME was developed in order to allow mobile devices to create standalone mobile JXTA networks or to participate in a JXTA network using a mobile device.

Journal of Internet Services and Information Security (JISIS), volume: 2, number: 1/2, pp. 1-21
*Corresponding author: Tel: +34934505342, Fax: +34934505201

The main characteristic of JXME is that it seriously takes into account the fact that a transition from a desktop environment to a mobile one requires facing challenges such as maintaining the trade off between scalability and efficiency, as well as the idiosyncrasies of mobile devices, such as power and storage limitations. However, the maturity of research in the field of P2P and mobile environments has pushed through new problems often neglected in a framework's design: those related with security. Even under the constraints of limited devices, a security baseline must be kept in any P2P system in order to protect it against common network vulnerabilities.

Our purpose in this paper is twofold. First of all, we examine the current state of JXME security, focusing in one of the two existing versions, JXME-Proxied. This study analyses basic peer operations by taking into account the whole peer life cycle, instead of in an isolated manner. From this study, it is possible to identify the available security mechanisms and how they operate, which may prove useful to application developers. Once the current vulnerabilities have been identified, we propose JXME-PLAuth, a simple protocol that bypass some of JXME's security shortcomings while still taking into account that it will be executed on limited devices.

The paper is organized as follows. Section 2, provides a brief overview of the JXME architecture. In Section 3, we present a security analysis of JXME from a typical peer operation cycle standpoint. Once current security has been assessed, Section 4 describes our proposal, JXME-PLAuth. The protocol security is evaluated in Section 5 and Section 6 presents the experimental results on regards to its performance on mobile devices. Finally, Section 7 summarizes the paper's main contributions and outlines further work.

2 JXME overview

As the response from the JXTA developer community to accelerate development in the wireless applications over mobile devices, Sun Microsystems presented a version of JXTA for Java 2 Micro Edition (J2ME), called JXME, providing JXTA capabilities to mobiles devices [4]. In fact, two distinct versions were developed, in order to accommodate to a broad set of scenarios. On one hand, the *JXME-Proxied* version, with very limied devices in mind, which delegates all heavyweight work to a JXTA super-peer. On the other hand, the *JXME-Proxyless* version is a straightforward port of JXTA, where peers may directly interact with the JXTA network. In this paper, we will mainly focus on the Proxied version, being the one which actually takes into account mobile device limitations on its protocols and diverging from the basic JXTA architecture, thus needing special consideration.

JXME is clearly based on JXTA, since they share the same basic specification. A detailed explanation of JXTA's generic protocols and services can be found in [30, 20] and in several papers in the literature, such as [4, 14, 24, 31]. In both JXTA and JXME, the basic organizational foundation is the *Peer Group*, a set of peers with common interests which agree on common services. Peer Groups are managed by the *Membership Service*, one of JXTA's core services, which manages the group members' identities within the group context. Identities are assigned by successfully completing an authentication process prior to actually joining the group. The Membership Service is defined as generic in the JXTA specification, leaving it up to developers to implement their own version, with the security level required by their applications.

Once a peer has joined a Peer Group, any resource may be shared with other group members by distributing its associated *Advertisement*, an XML metadata document describing the resource properties and how it may be accessed. A network resource cannot be accessed without previously recovering its associated Advertisement. Advertisements are located and distributed using the *Discovery Service*. Every time an Advertisement is retrieved by a peer, it is stored in a local cache and assigned an expiration date. On that date, the Advertisement will be automatically flushed. Once a resource has been successfully

located, messaging may begin using JXTA *pipes*, abstract endpoints which provide an asynchronous unidirectional communication channel.

Figure 1 shows the main components of the JXME architecture and how their devices are integrated into a JXTA network. Devices using the Proxyless version may directly communicate with JXTA peers, whereas those which use the Proxied version version are named *Proxied Peers* and, since they are assumed to have very limited resources, cannot directly communicate with other peers.

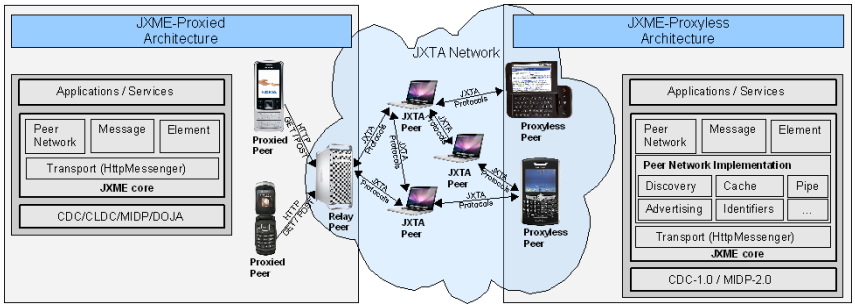


Figure 1: JXME-Proxied architecture

All communications from a Proxied Peer are actually destined to a super-peer which will overcome its limited capabilities, called *Relay Peer*, which implements the Relay and Proxy JXTA services. The Relay Peer assigns JXTA PeerID to the Proxied Peer before it may interact with the JXTA network. Furthermore, it translates or summarizes requests and responds to queries on its behalf.

The communication between the Proxied and Relay Peer is performed with a simplified protocol based on HTTP. By default, a single Relay Peer can support up to 150 Proxied Peers. The main responsibilities of the Relay Peer with regards to its Proxied Peers are to listen to and answer requests from them, translate messages received to XML and retransmit them to the JXTA network, store messages received from the JXTA network for Proxied Peers and summarize and translate XML messages from the JXTA network into a simple format which Proxied Peers are able to understand.

Due to its limitations and reliance on a Relay Peer, the kind of operations that a Proxied Peer can actually execute are limited to a very small set. First of all, Proxied Peers may **join** a group. Once the peer has successfully performed this operation, it may interact with other group members by **searching** or **creating** resources (such as Peer Groups or pipes), **listening** to a pipe to receive data, **sending** data to a specific pipe, **closing** a pipe and **polling** the Relay Peer for messages from the JXTA network that have the Proxied Peer as the final destination.

All message exchanges between a Proxied and Relay Peer share a special simple protocol encapsulated using HTTP-POST. To reduce the number of messages sent to the Relay Peer, they are stored in a queue at the Proxied Peer and each time a poll operation is performed, the first message in the queue is actually sent to the Relay Peer.

In the communication between a Proxied and Relay Peer each message starts with a special header where all namespaces are declared, and then is structured as a list of individual elements which contain the request type and its associated parameters. Every element is formatted as simple text, in contrast with standard JXTA, which uses XML, and always follows the same syntax. The element fields have the following order and functionality:

- Starting, a `jxel` string.
- The element's namespace identifier, chosen from the ones declared in the message header.

- An optional flag specifying additional element properties, such as content encoding.
- The element type, which is composed by two sub-fields: the type string length and the type itself.
- The element content, also composed by two sub-fields: the content string length and the type itself.

A sample message, is shown in Figure 2. All elements would actually be sent in a single line, but the header and each element have been put in different lines and the text has been formatted for the sake of readability. In this example, for instance, in the first element (second line), the namespace with id `2` is used, the element type is `request` (a string 7 characters long), and the content is `search` (6 characters long). This means that a `search` operation is being requested to the Relay Peer. The other elements are the parameters associated to such kind of request, and will vary depending on the request type.

Listing 1 - JXME-Proxied sample message

```
jxmg 0 01 05 proxy     08
jxel 2  0 07 request   0006 search
jxel 2  0 04 type      0005 GROUP
jxel 2  0 04 attr      0004 name
jxel 2  0 05 value     0006 myName
jxel 2  0 09 threshold 0001 2
jxel 2  0 09 requestid 0001 6
jxel 1  0 26 EndpointDestinationAddress xxxx <destination address>
jxel 1 0 21 EndpointSourceAddress       xxxx <source address>
```

Figure 2: JXME-Proxied sample message

From this overview of the JXME, focusing on the Proxied version, it is obvious that the need for Relay Peers is the main design divergence and limitation of this approach from original JXTA. The main consequences are twofold. First of all, if a set of Proxied Peers join the network using a single Relay Peer, then a central point of failure is created for all of them. However, and secondly, if a single Proxied Peer simultaneously connects via different Relay Peers in order to avoid the former pitfall, it will be assigned a different JXTA PeerId by each Relay Peer. Thus, the Proxied Peer, by simultaneously having different identities, will be considered as several different peers within the JXTA network.

2.1 Related research on JXME security

As it has been pointed out, security is a key feature in current P2P middlewares. Unfortunately, to our best knowledge, not many proposals exist in the literature for JXME.

Authors in [22] present an infrastructureless network composed of limited mobile devices called Mobile Ad hoc NETworks (MANETs). For them, JXME is a significant attempt at designing middleware for mobile devices.

On regards to the Proxyless version, since it is is a direct, though somewhat simplified, port of JXTA, supporting the same protocols and architecture, proposals that apply to JXTA also apply to this version, such as [3]. Nevertheless, an extensive survey on the state of the security in JXTA can be found here [2].

As far as the Proxied version is concerned, Kawulok et al. [14] show a framework which allows wireless and remote peers to participate in a JXTA network. Authors describe the most interesting implementation details of the framework as well as all changes made in the JXTA core and JXME packages. The proposed framework adds a new authentication scheme based on certificates and PKI [12]. This authentication is provided by the Relay Peer, which uses an external Sign and LDAP Server, breaking completely the P2P model proposed by JXTA.

To sum up, JXME-Proxyless version maintains the same structure as JXTA and therefore can inherit its improvements, such as new security schemes. Oppositely, JXME-Proxied version uses a simplified protocol and cannot inherit directly security improvements from JXTA. And, to our best knowledge, there is only one authentication proposal that tries to increase JXME-Proxied security baseline, but provides it in a centralized manner.

3 JXME-Proxied Security analysis

A global security analysis on standard JXTA already exists in [2]. Since the Proxyless version operates just like standard JXTA, most of its conclusions apply. However, it does not apply to the Proxied version because of its divergence from the base JXTA model, by relying some operations on a Relay Peer. Therefore, we will only focus on analyzing the security degree of the Proxied version. From this study, we can identify and assess existing vulnerabilities. Nevertheless, the analysis follows the same methodology proposed in the aforementioned study, where the typical peer life cycle is examined rather than isolated peer actions.

In order to perform a security assessment, it is useful to identify and categorize the most common types of attack in P2P networks. All attacks can be divided into two distinct groups, according to the degree of involvement of the attacker [6]: passive attacks, where the attacker just monitors peer activity and network traffic, and active attacks, where the attacker purposely interferes with data or network activity. Each group can be further classified according to the particular action performed by the attacker.

Passive attacks which have been considered are *Eavesdropping* and *Traffic analysis*. The former consists of searching in message exchanges for sensitive information, such as passwords, whereas the latter analyzes traffic data looking for patterns and relevant peers.

Active attacks include *Spoofing*, *Man-in-the-middle*, *Replay*, *Local data alteration* and *Software security flaws*. Spoofing consists in impersonating another peer. Man-in-the-middle (MitM) intercepts the communications between two parties transparently relaying forged messages to each one. Replay captures messages so they can be reused at a later time to simulate a real message exchange initialization. Local data alteration modifies local data to corrupt the system behavior. Finally, Software security flaws exploit vulnerabilities due to bugs in the source code, trying to obtain unexpected actions on the software.

This assessment will focus on the communications between the Proxied and Relay Peer, also taking into account the way a Relay Peer stores and manages its subscribed Proxied Peers' data. All communications between the Relay Peer and the rest of the JXTA network operate under the standard JXTA security model, so it is out of the scope of ths paper since it has been deeply analyzed in [2].

The standard JXME peer general operation cycle can be summarized in the following stages: Platform startup, Peer Group joining, Resource discovery and publication, Message exchange and Disconnection. The security analysis follows the actions performed by a Proxied Peer according to this lifecycle.

3.1 Platform startup

The first step during platform startup is loading the JXME libraries into the system. Unfortunately, no security model has been considered in order to distinguish correct JXME binary releases from another one with malicious code (*Local data alteration*).

Once the platform binaries have been loaded, but before a Proxied Peer may join the JXTA network, it must associate with any available Relay Peer. During this process, according to the JXME specification, the Relay Peer generates a new peer identifier, *PeerId*, and sends it back to the Proxied Peer. Such an identifier is only used within the context of message exchanges between Proxied and Relay Peers and

has no prevalence in the general JXTA network. At this point a Proxied Peer joins (next operation) to a default Peer Group, the NetPeerGroup, obtaining a JXTA PeerId.

The PeerId generation process is very important, since the Relay Peer is only able to identify Proxied Peers by their PeerId. However, we have found that, in the actual implementation, Proxied Peers can freely generate their own PeerId and connect to the Relay Peer, skipping most of this process. In fact, due to this occurrence, any peer may trivially impersonate others by self-assigning a PeerId already in use.

The PeerId generation process is an exception to the JXME-Proxied message format described in Section 2, since the request is sent using HTTP-GET. An example of an identifier request message is shown in Figure 3. It can be recognized as such since the PeerId is specified as `unknown-unknown` in the GET command.

Listing 2 - Proxied Peer PeerId request message

```
GET /unknown-unknown?0,-1,http://192.168.0.37:2481/
EndpointService:jxta-NetGroup/uuid-DEADBEEF...0F05/pid HTTP/1.1
Connection: close Content-Length: 0
User-Agent: UNTRUSTED/1.0
Host:172.16.0.37:2481
```

Figure 3: Proxied Peer PeerId request message

Sending the PeerId requires an initial communication protocol, which makes Proxied Peers very vulnerable at startup, since the identifier is transmitted in clear text over the network, allowing an attacker to easily learn it (*Eavesdropping*). Furthermore, since no authentication exists between a Proxied and Relay Peer, an attacker can also act as an invisible intermediate with the Relay Peer, redirecting HTTP messages (*Man-in-the-Middle*). At this stage, reusing intercepted data (*Replay*) makes no sense, since each peer starts the platform only once.

Even though JXME does not take into consideration a secure startup stage, an initial authentication protocol based on PKI is described in the Trusted Group proposal [14], out of the scope of the JXME's specification. The mechanism provides unidirectional authentication, only the Proxied Peer authenticates the Relay Peer, ensuring it is a legitimate one. However, this authentication is provided using a central Sign Server and LDAP database, which partially breaks the P2P model.

3.2 Peer Group joining

A Proxied Peer may join any Peer Group through the Relay Peer. As a requisite to proceed with this process, all group members must agree to use the same Membership Service implementation. This is achieved by sending a message with a `request` element which contains the `join` string. Group Membership parameters are included in the rest of the message elements.

The default Membership Service implementation in JXME-Proxied is the *None Membership Service*, which is used in groups without any kind of authentication, where any peer may claim any identity. It was designed for applications with no security requirements. As a result, all data exchanges based on this Membership Service are completely insecure.

Since the information in transmitted messages is sent in clear text, an attacker may discover the group any Proxied Peer is trying to join (*Eavesdropping*) and identify important peers by its traffic (*Traffic analysis*). Also, an attacker may easily impersonate any peer by claiming the other peer's identity within the Peer Group (*Spoofing*). However, during the join operation, reusing a directly captured message

(*Replay*) is pointless since a Proxied Peer can only join once to a Peer Group during a single session. Therefore, replay attacks are not a real concern.

In order to address some of these vulnerabilities, the proposal in [14] also extends its basic principle to provide an additional implementation of a Membership Service, external to the JXTA specification. An authentication mechanism is provided between the Proxied Peer and the Peer Group. However, in this case, it is based on mutual authentication (the Peer Group itself may also be authenticated). This authentication is based on certificates from both the Proxied Peer and the Peer Group and an external Sign and LDAP server.

3.3 Resource discovery and publication

Resource publication and discovery are also fully managed by the Relay Peer. The Proxied Peer just sends requests, using the HTTP-POST protocol, asking for Advertisements to be created or located.

Unfortunately, as a result of sharing the same simple protocol, an attacker can easily publish false resources (*Spoofing*) and modify/delete the Advertisements (*Man-in-the-middle*) of any other Proxied Peer, since no authentication is enforced. Any peer may create Advertisements on any other peer's behalf at leisure. Furthermore, an attacker can resend captured messages performing the original operation several times (*Replay*) in order to produce multiple resource discovery queries.

Finally, Advertisements are transmitted without encryption and can be easily intercepted (*Eavesdropping*) by an attacker, which can recognize important peers, those sharing many resources, by analyzing its traffic (*Traffic analysis*).

3.4 Message exchange

In JXTA, network messages are exchanged using pipes, briefly introduced in Section 2. Unfortunately, Proxied Peers are not able to use pipes between them and the Relay Peers. Since pipe usage is a complex mechanism which requires a non-negligible amount of system resources, it is the Relay Peer which, again, actually manages pipes, connecting to services in the JXTA network on behalf of the Proxied Peer. The communication between a Proxied and Relay Peer is performed using HTTP.

Pipe management requests are also based on a generic `request` element type. The element content dictates the actual operation: `create`, to create the pipe, `listen`, to receive messages, and `send`, to send messages. There is no specific element type to receive pipe messages. They are automatically transmitted from the Relay Peer each time any request is received from the Proxied Peer, along with the request reply.

As a result, different attacks can be performed: *Eavesdropping*, *Traffic analysis*, *Spoofing*, *Man-in-the-middle* and *Replay*. These attacks mainly allow an attacker to send/receive and sniff messages, as well as impersonate any peer. Moreover, an attacker can close legitimate peer pipes at will, abruptly ending message exchanges.

3.5 Disconnection

Before a peer may disconnect from the JXTA network, all pipes should be previously closed. However, the main limitation at this step is that no operation currently exists in JXME-Proxied for this purpose. It is also the Relay Peer that has to decide when to unsubscribe a peer from a Peer Group. Therefore the victim's PeerId may be easily spoofed, stealing his open pipes, opening new pipes, preventing pipe disconnection, and using all the groups previously joined.

Op./Threat	Evs	TAn	Spf	MitM	Rp	LDA	SSF
Startup	V(2)	N/A	V(4) P(TGMS)	V(2, 4) P(TGMS)	N/A	V(1)	P(OSS)
Join	V(2)	V(3)	V(4) P(TGMS)	V(2, 4) P(TGMS)	N/A	N/A	P(OSS)
Publish/ Discover	V(2)	V(3)	V(4)	V(2, 4)	V(4)	N/A	P(OSS)
Messaging	V(2)	V(3)	V(4)	V(2, 4)	V(4)	N/A	P(OSS)
Disconnect	V(2)	V(3)	V(4)	V(2, 4)	N/A	N/A	P(OSS)

N/A: Non-applicable.
V(type): Vulnerability exists.
P(mechanism): Security mechanism used

Table 1: JXME-Proxied peer operation cycle security summary

3.6 JXME-Proxied security evaluation summary

Since JXME-Proxied is Open Source Software (OSS), supported by a community of developers, and it is also very simple and small, it could be considered relatively safe from *Software security flaws*. Furthermore, since Proxied Peers do not store data locally, they are not vulnerable at execution time to *Local data alteration*.

The analysis of possible attacks and the existing security mechanisms of JXME-Proxied, classified by peer operations, provides a vulnerability map summarized in Table 1.

The four main vulnerabilities found are:

- **V(1)**: malicious executable code can easily be built and cannot be automatically discovered when installed
- **V(2)**: no encryption mechanism exists
- **V(3)**: no data flow masquerading mechanism exists
- **V(4)**: no actual authentication is enforced

The available security mechanisms are:

- **P(OSS)**: Open Source Software
- **P(TGMS)**: Trusted Group Membership Service [14]

4 A secure protocol extension for JXME-Proxied: JXME-PLAuth

The security analysis presented in Section 3 shows that the security mechanisms provided by JXME-Proxied are still not sufficient to secure standard mobile applications. This is because the current version is vulnerable to a wide range of attacks. However, some attacks can be prevented by simple schemes that extend the basic protocols used in JXME-Proxied, adding only a bounded complexity. In this paper, we present JXME-PLAuth (JXME-Proxied Light Authentication), a proposal to avoid *Spoofing* and *Replay* attacks. This proposal does not try to solve every single vulnerability which was identified, which would require a much broader set of security mechanisms, but provides a initial protection in the communication by guaranteeing lightweight authentication. This is the first step in providing a secure mobile framework for JXME.

The main vulnerabilities found in JXME-Proxied are produced by the insecure communication link between the Proxied Peer and its associated Relay Peer. For that reason, the proposed scheme tries to

secure the common message protocol over HTTP between the Proxied and Relay Peer. A secure extension for this communication protocol is built. Under the assumption that a Proxied Peer is a very limited device, the proposed protocol extension is based on lightweight cryptographic operations, mainly hash functions. This approach has been successfully used in other proposals for MANET based environments, such as [13].

Protection against Spoofing and Replay attacks may be obtained by securely identifying the Proxied Peer using any well known authentication scheme. Our proposal relies on a lightweight scheme, since the constrained resources of devices where Proxied Peers are executed must be taken into account. For that reason, the protection against Spoofing and Replay is obtained not by linking messages to a particular peer identity but by guaranteeing that, given a set of messages, all come from the same source peer, whichever that source might be. Thus, once an identifier is assigned, it can be guaranteed no intruder is able to insert false messages impersonating the source peer. To achieve such goal, we propose the use of a hash-chain [16] based scheme to create a set of linked values which will be used as local identifiers.

Therefore, we propose a security extension to the basic PeerId generation protocol at the platform startup stage in order to counter attacks regarding authenticity at every stage in a peer's lifecycle.

Instead of obtaining a PeerId from the Relay Peer, Proxied Peers generate themselves a sufficiently long hash-chain (taking into account available resources) and use each intermediate value as its PeerId in each successive message exchange with its associated Relay Peer. In this way, the PeerId attached in a message changes for each successive message in a manner that cannot be predicted by a possible attacker. However, the Relay Peer will be able to easily track identifier changes and recognize each message as originating from the same source. Using a changing PeerId allows us to use exactly the same original protocol format, without the need to add additional fields.

4.1 Protocol initialization

The proposed scheme needs an initialization process executed during the Proxied Peer's startup step and boot operation. In this process, the hash-chain is created, values are stored in the Proxied Peer's internal memory and the first PeerId is transmitted to the Relay Peer.

The detailed initialization process is next described:

(1) At the startup stage, the Proxied Peer chooses a random seed, s.

(2) The Proxied Peer generates a hash-chain $hc(s) = \{h^0(s), \cdots, h^n(s)\}$ by iteratively applying n times the hash function $h(\cdot)$ on s, so that $h^0(s) = s$ and $h^i(s) = h(h^{i-1}(s))$. All values in hc are stored in the peers' local memory. Figure 4 summarizes this process.

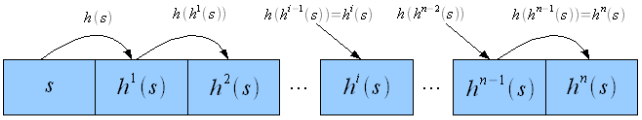


Figure 4: Hash-Chain creation

(3) The Proxied Peer's initial PeerId, $initId$, is created from $h^n(s)$, fulfilling the JXTA peer identifier specification. A PeerId may be created from any value in $h^n(s)$ according to the following steps (summarized in Figure 5):

- The PeerId starts with the string $uuid-$.
- The 16 most significant bytes, $\text{msb}_{16}(initId)$, are the *NetPeerGroup* identifier.

- From the 16^{th} to the 31^{st} byte the Peer UUID (from now on, summarized as as PUUID) is specified. The 16 least significant bytes of the hash value, $lsb_{16}(h^n(s))$, are assigned.
- The 32^{th} byte describes the ID type identifier. In this case, being a PeerId, the value 0x03 is assigned.

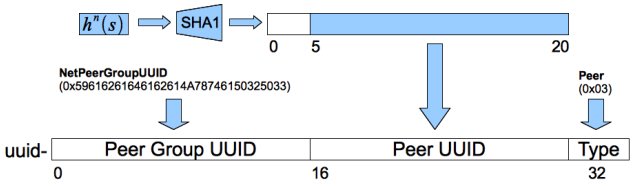


Figure 5: initId generation

(4) A PeerId request message is sent to the Relay Peer using its well-known address. However, instead containing an `unknown-unknown` string, as would be used in the original insecure protocol, *initId* is announced.

(5) When the request is received, the Relay Peer randomly generates a new JXTA PeerId, *jxtaID*1, as would be done in standard JXME. At this point, the Proxied Peer is considered associated to the Relay Peer.

(6) The Relay Peer keeps track of the identifiers for its possible different Proxied Peers in a local translation table $T_{proxied}$ that contains two fields: `lastId` and `globalId`. The former contains PeerId's and the latter JXTA PeerId's. The Relay Peer translates local PeerId's to JXTA ones (as explained in Section 2) when acting as some Proxied Peers behalf in the JXTA network. At this point a new entry is added to the table, `lastId` = *initId* and `globalId` = *jxtaID1*, where *initId* is considered the entry's key.

4.2 Protocol execution

After the initialization process has been performed, secure communication between the Proxied Peer and the Relay Peer can begin. Each message will use a new PeerId generated from the successive values extracted from the $hc(s)$. These values are retrieved in the descending order from their generation, starting from $h^{n-1}(s)$ and ending in s. That is, in the second message, the Proxied Peer will use the identifier:

$$newId = \text{"}uuid-\text{"}||NetPeerGroupUUID||\text{lsb}_{16}(h^{n-1}(s))||\text{"}03\text{"}$$

In general, the j-th message between the Proxied and the Relay Peers will contain the identifier:

$$newId = \text{"}uuid-\text{"}||NetPeerGroupUUID||\text{lsb}_{16}(h^{n-j+1}(s))||\text{"}03\text{"}$$

The identifier consumption from $hc(s)$ and the translation between the changing PeerId and the static JXTA PeerId is represented in Figure 6. To simplify this figure the identifier is represented as the PUUID part of the Proxied Peer identifier, which is the only one which varies at each message exchange.

The verification process is executed at the Relay Peer, validating that all successive messages come from the same Proxied Peer. This validation may be actually performed since, assuming that $lastPUUID$ is the PUUID section in the identifier from the last message sent from the Proxied Peer (stored in the

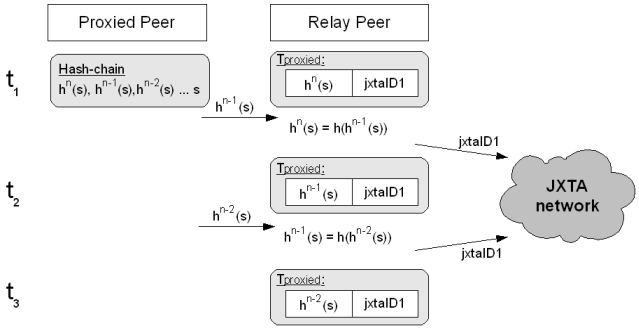


Figure 6: Hash-Chain consumption and identifier translation

`lastId` field of $T_{proxied}$), then the PUUID section from the current message's identifier, $currentPUUID$, must hold true that:

$$h(currentPUUID) = lastPUUID$$

The detailed verification process for the PUUID of the j-th message follows:

(1) The Relay Peer takes the message identifier $currentId$.

(2) The PUUID section of the identifier, $currentPUUID$, is extracted from $currentId$.

(3) If $currentPUUID$ matches with $T_{proxied}$'s `lastId` field, it means that someone is trying to perform a spoofing or replay attack. Therefore, this petition is obviated. Otherwise, the authentication protocol continues.

(4) The Relay Peer calculates $h(currentPUUID)$. From this value a PeerId is generated following the steps described in Figure 5.

(5) The result is looked up among the currently stored values in $T_{proxied}$'s `lastId` field.

(6) If a match exists, the message is not a result of spoofing or a replay attack, since no other peer would be able to predict the currentPUUID ($h^{n-j+1}(s)$) from lastPUUID ($h^{n-j+2}(s)$) and use it as a portion of the message identifier. Only the legitimate hash-chain generator is able to calculate it, from its hash-chain.

(7) The Relay Peer stores $currentId$ into $T_{proxied}$ replacing the old value matched in step (5). It becomes the entry's new key.

(8) If, as a result of the received request, the Relay Peer needs to send messages towards the JXTA network on behalf of the Proxied Peer, the value stored in the $globalId$ field is used.

4.3 Hash-chain refresh

When a hash-chain is about to reach s, a new one must be generated and its initial value refreshed at the Relay Peer so the new hash-chain values may be used. s will be used as the PUUID part of the PeerId in the refresh message. To allow this process, the set of operations that a Proxied Peer can perform using HTTP-GET is extended with a `renew` command. This new parameter is used to announce a refresh in

the hash-chain, containing as its associated parameter the new Id:

$$"uuid-"||netPeerGroupUUID||h^n(newSeed)||"03"$$

When the Relay Peer receives any message which contains a `renew` command, in step 6 of the verification process, its content value (*newId*) is the one stored instead of the message's *currentId*, initializing the new set of local identifiers.

As it has been mentioned before, the proposed approach avoids *Spoofing* and *Replay* attacks. Furthermore, this proposal minimizes the modifications of the JXME protocols, since no additional `request` type is defined. The `renew` command piggybacks inside any other naturally occurring request, such as a `listen` request, just like an additional parameter, and will be processed along the original request. Therefore, there is no need to send a single message with the sole purpose of transmitting hash-chain data, reducing the overhead by taking advantage of existing transmissions. Furthermore, while no refresh is needed, the secure scheme does not even impact on the HTTP-GET protocol, since the peer identifier field is invisibly used, instead of using additional message element types.

On regards to the computational cost of this proposal, it is worth mention that the only cryptographic operations used are hash value computations. Hash values are lightweight cryptographic operations which can be efficiently computed even in constrained devices.

5 JXME-PLAuth security evaluation

JXME-PLAuth adds simple mechanisms to provide protection against Spoofing and Reply attacks. Other attacks, such as Man-in-the-middle attacks, are not protected. This is because our proposal tries to be as simple as possible. Also, these attacks are not expected in some scenarios, such as when communications are direct.

The mechanism used to renew the hash-chain is authenticated using the same mechanism of hash-chain identifiers, therefore it is as secure as normal authentication using hash-chains.

Some messages can be lost, this can be solved in several ways [23, 7]. An easy integration of this mechanism in JXME-PLAuth can be done by adding a new parameter in exchanged messages containing the position of the peer identifier in its hash-chain. Also, the Relay Peer has to store and maintain updated this new information in its local translation table. Then, when the Relay Peer receives a message which contains a peer identifier that does not match in the local translation table, it can perform further hashes based on the identifier and the position in the hash-chain. If a matching is produced means that a previous message has been lost, but by this mechanism the local table is updated and further messages of this Proxied Peer can still be secure authenticated.

A security analysis similar as the one performed in Section 3 is conducted to our security proposal, JXME-PLAuth. This analysis follows the same methodology as before and just the security in the communications between the Proxied and the Relay Peer are analyzed.

5.1 Platform startup

In this stage the Proxied Peer becomes linked with its Relay Peer. Now it is safe of Spoofing attacks, since no other peer can authenticate itself impersonating another peer.

If one Peer tries to do so, it is going to receive an error. This is explained in Section 4.2. Peers can know current identifiers of other peers but cannot predict their next identifiers and therefore peers are secure authenticated.

There are two possible scenarios potentially prone to suffer security threads:

(1) Two Proxied Peers use the same identifier but in different Relay Peers.

(2) A Proxied Peer (P1) is authenticated in a Relay Peer (R1). Another Proxied Peer (P2) capture P1's identities and authenticate itself in R1 but always using the previous identifier of P1.

Both scenarios are extreme but possible cases. To understand how this protocol deal with these scenarios is important to have in mind Figure 6, where is shown that there are two identifiers: One internal identifier between the Proxied Peer and the Relay Peer, and one global identifier inside the JXTA Peer Group. This analysis, and in general all the paper, is only focused on the first identifier, the internal one.

In the first scenario both Proxied Peers will have the same internal identifier but in different Proxied Peers. And these peers will probably have different JXTA identifiers, but this is responsibility of the JXTA Membership Service, outside of the scope of this analysis. Therefore, secure authentication is provided.

In the second scenario both Proxied Peers (P1 and P2) will be authenticated at the same Relay Peer. At any moment, P1 will have its identity based on $h^i(s)$ while P2 in $h^{i+1}(s)$ (previous P1's identifier). Therefore, both internal identifiers are different. P2's identifier must always come, at minimum, one step before P1's identifier, otherwise P2 is going to receive an error when authenticating. Also, as previous scenario, the JXTA identifier of both peers should be different.

5.2 Peer Group joining

A Proxied Peer has to send a message to its Relay Peer in order to request to join a Peer Group. In this case, this operation is protected against Spoofing attacks since only the Proxied Peer can know the next identifier in the hash-chain.

5.3 Resource discovery and publication

This operation is also performed by a Proxied Peer by sending a message to its Relay Peer. In this case the Proxied Peer is also secured from spoofing attacks since non other peer different than it can send a message with its next identifier to its Relay Peer to find or create a resource in its behalf. Also, the message is protected against Replay attacks since when a message is resend with the same identifier, the Relay Peer will discard it.

5.4 Message exchange

Performing this operation produces a similar message as the previous step. In this case, Spoofing and Replay attacks are as well secured since no one knows Proxied Peer's next identifier and captured packages will be discarded by the Relay Peer if replayed.

5.5 Disconnection

In this operation, when a Proxied Peer decides to leave the network, it still does not have a specific operation to remove completely all its information stored in its Relay Peer. But using this authentication scheme prevents that other peers could perform Spoofing attacks, like closing its open pipes. Also, this mechanism guarantees that when a Proxied Peer disconnects from the network no one can steal its information contained in its Relay Peer. This is because the authentication scheme will remain working until the Relay Peer decides to remove Proxied Peer's data.

Op./Threat	Evs	TAn	Spf	MitM	Rp	LDA	SSF
Startup	V(2)	N/A	**P(JXME -PLAuth)** P(TGMS)	V(2) P(TGMS)	N/A	V(1)	P(OSS)
Join	V(2)	V(3)	**P(JXME -PLAuth)** P(TGMS)	V(2) P(TGMS)	N/A	N/A	P(OSS)
Publish/ Discover	V(2)	V(3)	**P(JXME -PLAuth)**	V(2)	**P(JXME -PLAuth)**	N/A	P(OSS)
Messaging	V(2)	V(3)	**P(JXME -PLAuth)**	V(2)	**P(JXME -PLAuth)**	N/A	P(OSS)
Disconnect	V(2)	V(3)	**P(JXME -PLAuth)**	V(2)	N/A	N/A	P(OSS)

N/A: Non-applicable.
V(type): Vulnerability exists.
P(mechanism): Security mechanism used

Table 2: JXME-Proxied using JXME-PLAuth extension peer operation cycle security summary

5.6 JXME-PLAuth security summary

The security analysis presented in this section verify that our security proposal JXME-PLAuth provides an enough strong authentication mechanism that prevents Proxied Peers from Spoofing and Replay attacks. From Table 1,

Table 2 has been generated in order to summarize this analysis. It shows that the vulnerability produced by the lack of an authentication scheme (V(4)) has been patched. Therefore, those attacks that were allowed only due to this vulnerability are avoided.

6 JXME-PLAuth experimental results

Some real experiments were done to test how the proposed security improvement to the JXME-Proxied framework impacts its overall behavior. The actual performance of our proposal implementation has been evaluated by assessing how the protocol extensions in this new security scheme would affect a Proxied Peer in terms of resource utilization. A mobile device acts as a Proxied Peer and a computer acts as a Relay Peer in these tests.

The mobile device needs network connectivity to exchange data with the Relay Peer and an open operating system which allows the execution of J2ME applications and obtaining information about the state of its resources, such as memory or battery usage. There are many operating systems for mobile devices, such as iOS [1], WebOS [21], Windows Phone 7 [19] or Android OS [11], but the one that fulfilled most our requirements was the Android OS. The main reason is the fact that this mobile operating system is based in a modified version of the linux kernel, which allows a high degree of customization and direct access to device resources. In addition, Android OS is becoming very popular and is being used in a large range of devices. Finally, although it does not allow to natively run J2ME applications, it natively supports Java. But, since native Android OS applications are being more popular than J2ME ones, it motivates us in spending some effort in customizing JXME-Proxied to be able to execute it in the large amount of devices that run Android OS.

The mobile phone chosen was the HTC Hero. This device was one of the most popular Android mobile phones at the moment of selecting a device for our tests but compared with new mobile phones its hardware is obsolete. Therefore, if JXME-PLAuth can be handled by this device, new devices will deal with it better. The HTC Hero has a processor Qualcomm MSM7200A 528 MHz, 512 MB of ROM,

288 MB of RAM, display of 3.2-inch TFT-LCD touch-sensitive with 320x480 HVGA resolution, Wi-Fi IEEE 802.11 b/g and rechargeable Lithium-ion battery with 3.7 V and 1350 mAh of capacity.

On regards to the computer, it has to be powerful enough to act as a Relay Peer and addressable, to receive communications from the Proxied Peer. Most of today's computers meet these requirements. The chosen computer has been a laptop computer, a MacBook Pro, which can be addressed through Wifi or Ethernet.

The communication channel used was Wi-Fi, since it is the one shared between both devices, and a wireless router was used in order to connect them.

The tests have to provide an idea of the overhead produced in a Proxied Peer implementing security. It is also assumed that a minimum overhead will be produced in the Relay Peer, but since it is expected that it has enough capabilities to deal with the extra work, these tests are not going to focus on it. The main test consisted in assess the behavior of a real JXME-Proxied application, a chat application, comparing a version which implements security with one which does not.

6.1 Test application

This test assessed the impact of the proposed security mechanism in a real application. The goal of this test is to detect the general influence of using security in the JXME messaging service.

The default JXME-Proxied distribution comes with a couple of sample applications to show the potential of this framework. One of these applications is a simple chat, which allows Proxied Peers to exchange messages between them.

This chat was chosen as the application for this test for many reasons:

- Message exchange is a typical operation in mobile devices. Until recently, this operation has been done using Short Message Service (SMS).
- With the extension of having access to the Internet network from mobile devices, some chat applications [32, 18] are getting really popular in this environment.
- This application is focused in communication rather than in computation, which allows an intensive test of JXME.
- The core structure of a chat application is simple.

However, since this chat is written in J2ME language and the Android OS cannot run it directly, some modifications have been performed. This is because while Android applications are written in Java, there is no Java Virtual Machine in the platform and Java bytecode cannot be executed. Java classes get recompiled into a Dalvik executable and run on a Dalvik virtual machine. Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. The modifications done in this chat application are basically the graphical user interface (GUI) and the J2ME framework specification used in the class responsible of the communication, the `HttpMessenger` Java class.

JXME-Proxied supports both CLDC and CDC J2ME frameworks specifications. The use of a different framework specification only affects in the HttpMessenger class, which is the one responsible for the communication. By default, JXME-Proxied uses CLDC, but Android does not support the classes used. Fortunately, all the classes used in CDC are supported by Android. Therefore, in our chat application, we have chosen the CDC version of the HttpMessenger class. Theoretically, all versions of this class should have worked properly, but the tests performed showed that the CDC version was not working correctly. However, after some small changes (always using the CLDC version as a model) the behavior of this class was fixed.

In this test the measured and compared resource was the battery consumption. Figure 7 shows some snapshots of the chat application running in Android.

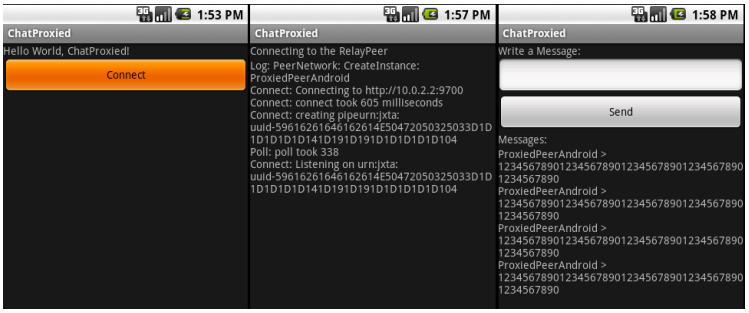


Figure 7: Chat Proxied GUI

6.1.1 Experimental results

This test measures the impact of JXME-PLAuth in a real application. Some considerations were taken:

- Screen light: Set to the minimum, since it is enough in most of the environments.
- Message exchange rate: A chat application is used in this test and we want to run it according to an average usage. This data is extracted from [5], where an analysis of the typical characteristics in instant messaging is conducted. The conclusion is that, typically, a message is sent each 10 seconds and the average size of each message is 32 chars.
- Hash function: SHA1 [9] has been used as the hash function, but any other hash function can be used.

The main test compares the application with and without security. In both cases the time interval between polls (operation performed by the Proxied Peer to send and receive messages to/from the Relay Peer) has to be defined. When security is activated, the size of the hash-chain also has to be specified.

Based on the poll time used in sample applications, and the time it takes to compute hash-chains of different sizes , the poll interval values that are considered significant are 1, 5 and 10 seconds (15 seconds are not considered since in this application the time between sent messages sent is 10 seconds). Different hash-chain sizes can be used for each test, but a logical restriction has been applied: The maximum time required to compute the hash-chain must be smaller to the time between polls. Table 3 shows the different possible combinations.

In Figure 8 the accumulated energy consumption of the chat application when using security and different time poll intervals and hash-chain sizes is presented (just trend lines are shown for the sake of

	HC size		
Poll interval (s)	50	250	500
1	✓	×	×
5	✓	✓	×
10	✓	✓	✓

Table 3: Combination of values of time between poll intervals and size of hash-chain used for the tests

readability). This shows that, in general, increasing poll time intervals and hash-chain sizes decreases the consumption of the battery. Having a constant hash-chain size and increasing the poll time interval always decrease the consumption. If the hash-chain size is 50 the percentage decrease of consumption for 1 and 5 seconds of poll interval is 10,77%, and for 5 and 10 seconds is 55,54%. Whereas when using a hash-chain of 250 elements the percentage decrease of consumption is 84,28% for 5 and 10 seconds. But having a constant poll time and increasing the hash-chain size not always decreases the battery consumption. If we use a poll interval of 5 second, the percentage decrease of consumption is 1,14% for hash-chains of 50 and 250 elements (this small difference between these two scenarios causes that in the Figure 8 both green lines are overlapped). And if 10 seconds is used, the percentage decrease of consumption is 17,15% for hash-chains of 50 and 250 elements, and -49,81% for hash-chains of 250 and 500 elements. The increase of consumption in this last case we think that is caused because the hash-chain is too big and does not fit into the L1 cache of the mobile phone (32 KB) and therefore more energy and time is required to calculate and consume the hash-chain. These values indicate that increasing the poll time intervals has higher impact in decreasing the consumption than increasing the hash-chain size.

It shows that renewing less and computing a longer hash-chain is more efficient than doing it more frequently.

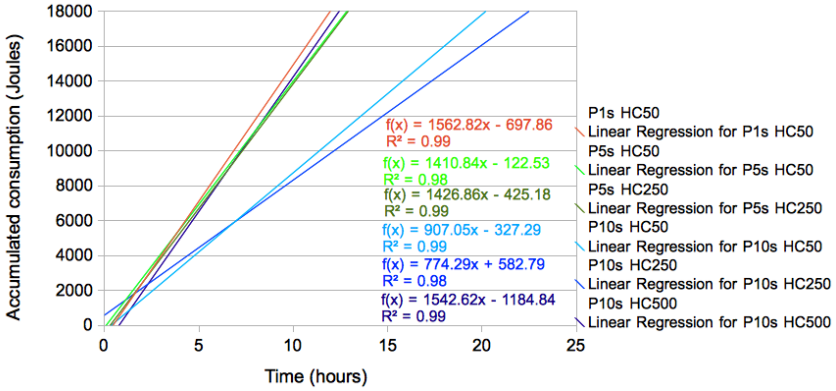


Figure 8: Accumulated consumption when running the chat application with security and different poll intervals and hash-chain sizes

To provide some perspective on energy consumption, the device's battery capacity is 1350mAh and its voltage is 3.7V. Just multiplying both values (converting the hours to seconds) we get that the battery has about 17982 Joules. When the device is functioning at maximum bright screen, the consumption is 22,5 J/min, which amounts to 6750 Joules/5hours, 13500 J/10h or 20250 J/15h. Therefore, even in the most intensive scenario (P1s HC50), the consumption is very similar to having the device idle with the maximum bright screen.

Figure 9 compares the accumulated battery consumption between using security or not in JXME-Proxied when running the chat application at different poll time intervals. When security is used, the hash-chain size which produces the better result in terms of battery usage is chosen for the different poll time intervals. This figure shows that in general the overhead produced by utilizing security is low. This overhead was expected because some security measures were used, but this impact also depends on the poll time. The percentage increase of consumption when using security and 1, 5 or 10 seconds as poll time interval is respectively 0,86, 48,77 and 2,3. And in the worst case, 5 seconds, the overhead produced can be reduced by playing with the size of the hash-chain.

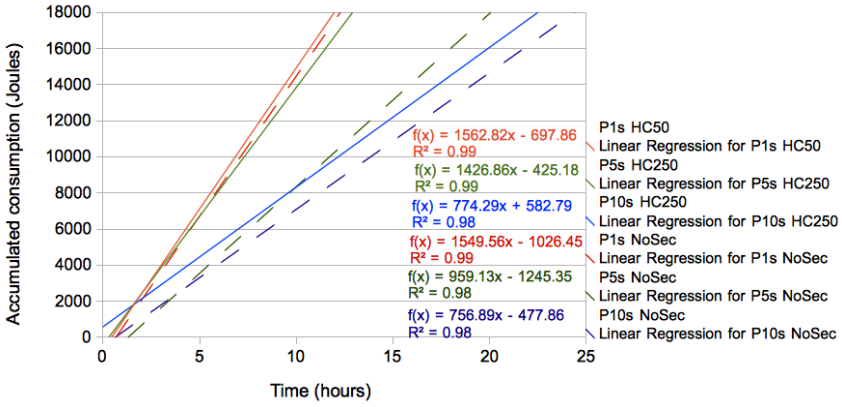


Figure 9: Accumulated consumption when running the chat application with security (continuous line) and without security (dashed line)

6.2 Evaluating the performance compared to HTTPS

All previous performance experiments have been done over our light authentication proposal, JXME-PLAuth. But is also important to compare the obtained results with the overhead produced with one of the typical authentication scheme, such is HTTPS [25]. Fist of all, It has to be pointed out that HTTPS additionally of guaranteeing authentication also provides encryption. But a trusted entity is required to manage certificates to guarantee this security level.

A experiment was conducted in order to compare the behavior of the system when using HTTPS instead of simple HTTP in the same scenario used in Section 6.1. The results showed that while the battery consumption of using HTTPS instead of HTTP is quite similar, the time required to perform HTTPS is enormous. Establishing connection and sending data in HTTP in average only took 0.5 seconds, while in HTTPS it took 26 seconds.

6.3 Tests conclusions

Previous tests analyze the impact caused by using security in mobile devices. These tests show that in general the overhead produced by JXME-PLAuth is low and acceptable. This overhead depends basically on the time between polls and the size of the hash-chain used. Increasing the time between polls always reduce the overhead. However, increasing the size of the hash-chain does not always reduce the overhead. For this reason, is important to chose a long hash-chain in order to reduce the times the hash-chain has to be renewed, but not too long that modify memory access patterns. Using higher authentication schemes, such as HTTPS, produces an important overhead, which justify the usage of simpler authentication schemes.

7 Conclusions and Future Work

JXME is the JXTA simplification that allows to run P2P applications using the JXTA milddleware when some peers are mobile devices with computational constraints. As we have analyzed in this paper, such simplification takes an extreme effect in the JXME-Proxied version, where the mobile device constraints imply that some of the tasks cannot be performed by the platform installed in the mobile peer and should

be delegated to a Relay Peer with more computational resources.

From the security point of view, we have performed an analysis that shows the relevance of the link between the Proxied Peer and its Relay Peer. Without properly securing such link, a wide range of attacks can be performed, from passive attacks, like Eavesdropping or Traffic analysis to more active ones such as Spoofing, Replay or Man-in-the middle.

The security analysis performed has allowed us to understand the nature of such vulnerabilities and then we have been able to provide an effective mechanism that allows to protect the communication between the Proxied Peer and its Relay Peer from Spoofing and Replay attacks. The obtained protection has been achieved using the concept of hash-chain in order to offer linkability between different communication sessions performed between the Proxied Peer and the Relay Peer. Using the onewayness of hash functions, linkability can be obtained at a low computational cost.

Such computational saving is translated in an affordable energy consumption, a really important feature when dealing with mobile devices. As it has been shown in the presented experimental results, in a standard mobile application like a messaging service running over an Android OS mobile device the inclusion of the proposed security mechanism does not impact significantly on the energy consumption of the device. Furthermore, the proposed solution does not imply to define new message protocols in JXME, since it can be implemented using standard JXME messages.

As we have discussed, other attacks rather than Spoofing and Replay can be performed on a JXME-Proxied implementation, and for that reason further research has to be performed in order to achieve the desired protection, a difficult task since the obtained solutions must be consistent with the constrained computational environment of a Proxied Peer device.

Acknowledgments

This work is partially supported by the Spanish Ministry of Science and Innovation and the FEDER funds under the grants TSI2007-65406-C03-03 E-AEGIS, CONSOLIDER CSD2007-00004 ARES and TIN2010-15764 N-KHRONOUS.

References

[1] Apple Inc. iOS. `http://www.apple.com/ios/`, 2008.

[2] J. Arnedo-Moreno and J. Herrera-Joancomartí. A survey on security in JXTA applications. *Journal of Systems and Software*, 82(9):1513–1525, 2009.

[3] J. Arnedo-Moreno and J. Herrera-Joancomartí. JXTA resource access control by means of advertisement encryption. *Future Generation Computer Systems*, 26(1):21–28, 2010.

[4] A. Arora, C. Haywood, and K. Pabla. JXTA for J2ME, extending the reach of wireless with JXTA technology. Technical report, SUN Microsystems, Inc, March 2002.

[5] D. Avrahami and S. E. Hudson. Communication characteristics of instant messaging: effects and predictions of interpersonal relationships. In *Proc. of the 20th Anniversary Conference on Computer Supported Cooperative Work (CSCW '06), Banff, Alberta, Canada*, pages 505–514. ACM Press, November 2006.

[6] D. Brookshier, D. Govoni, N. Krishnan, and J. Soto. *JXTA: Java P2P Programming - Chapter 8: JXTA and Security*. Sams, 2002.

[7] C. Cano, M. Guerrero, and B. Bellalta. Secure and efficient data collection in sensor networks. In *Proc. of 4th International Workshop of the EuroNGI/EuroFGI Network of Excellence, LNCS, Barcelona, Spain*, volume 5122, pages 37–48. Springer-Verlag, January 2008.

[8] B. Christensen. Experiences developing mobile P2P applications with lightpeers. In *Proc. of the 6th IEEE International Conference on Peer-to-Peer Computing (P2P'06), Cambridge, UK*, pages 229–230. IEEE, September 2006.

[9] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). IETF RFC 3174, September 2001. `http://www.ietf.org/rfc/rfc3174.txt`.

[10] J. Frankel and T. Pepper. Gnutella. `http://rfc-gnutella.sourceforge.net`, 2000.

[11] Google Inc. Project Android'. `http://code.google.com/intl/es/android`, 2007.

[12] R. Housley, W. Ford, W. Polk, and D. Solo. Internet x.509 public key infrastructure. IETF RFC 2459, January 1999. `http://www.ietf.org/rfc/rfc2459.txt`.

[13] H. Janzadeh, K. Fayazbakhsh, M. Dehghan, and M. Fallah. A secure credit-based cooperation stimulating mechanism for MANETs using hash chains. *Future Generation Computer Systems*, 25(8):21–28, 2009.

[14] L. Kawulok, K. Zielinski, and M. Jaeschke. Trusted group membership service for JXME (JXTA4J2ME). In *Proc. of IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'05), Montreal, Quebec, Canada*, volume 4, pages 116–121. IEEE, August 2005.

[15] G. Kortuem. Proem: a middleware platform for mobile peer-to-peer computing. *Mobile Computing and Communications Review - SIGMOBILE*, 6(4):62–64, 2002.

[16] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[17] K. Matsuo, L. Barolli, F. Xhafa, A. Koyama, and A. Durresi. Implementation of a JXTA-based P2P e-learning system and its performance evaluation. *International Journal of Web Information Systems*, 4(3):352–371, 2008.

[18] Meebo Inc. Meebo. `http://www.meebo.com/android/`, 2008.

[19] Microsoft. Windows Phone 7. `http://www.microsoft.com/windowsphone/`, 2010.

[20] Oracle. JXTA java standard edition v2.5: Programmers guide. `https://jxta-guide.dev.java.net/`, 2007.

[21] Palm Inc. WebOS. `http://developer.palm.com/`, 2009.

[22] G. Paroux, I. Demeure, and D. Baruch. A survey of middleware for mobile ad hoc networks. Technical Report Technical Report 2007/D004, Ecole Nationale Supérieure des Télécommunications, 2007.

[23] A. Perrig, R. Canetti, and I. T. J. Watson. Efficient authentication and signing of multicast streams over lossy channels. In *Proc. of the 2000 IEEE Symposium on Security and Privacy (SP'00), Berkeley, California, USA*, pages 56–73. IEEE, May 2000.

[24] T. Piedrahita and E. Montoya. Performance analysis of JXTA/JXME applications in hybrid fixed/mobile environments. *Revista Colombiana De Computación*, 7(1):5, 2006.

[25] E. Rescorla. HTTP Over TLS. IETF RFC 2818, May 2000. `http://tools.ietf.org/html/rfc2818`.

[26] L. Shou, X. Zhang, P. Wang, G. Chen, and J. Dong. Supporting multi-dimensional queries in mobile P2P network. *Information Sciences*, 181(13):2841–2857, 2011.

[27] Skype. Skype on your mobile. `http://www.skype.com/mobile`, 2004.

[28] Sun Microsystems. Project JXTA. `http://www.jxta.org`, 2001.

[29] Sun Microsystems. Project JXME. `https://jxta-jxme.dev.java.net`, 2003.

[30] Sun Microsystems. JXTA v2.0 protocols specification. `https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html`, 2007.

[31] T. Tahsin, L. Choudhury, and L. Rahman. Peer-to-Peer mobile applications using JXTA/JXME. In *Proc. of the 11th International Conference on Computer and Information Technology (ICCIT'08), Busan, Korea*, pages 702–707. IEEE, December 2008.

[32] WhatsApp Inc. WhatsApp. `http://www.whatsapp.com/`, 2009.

[33] J. Zhu, J. Gong, W. Liu, T. Song, and J. Zhang. A collaborative virtual geographic environment based on P2P and Grid technologies. *Information Sciences*, 177(21):4621–4633, 2007.

Marc Domingo-Prieto holds the degree of Computer Systems and the master in Computer Architecture, Networks and Systems from Universitat Politècnica de Catalunya (UPC). He is doing his phd at Worldsensing company about wireless sensor networks (WSN) communications security. Also, he is a research assistant in the Kryptography and Information Security for Open Networks (KISON) research group in the Open University of Catalonia (UOC). His research interests include scalable distributed algorithms and applications, energy efficient and security in WSN, mobile and peer-to-peer applications.

Joan Arnedo-Moreno is a lecturer at Estudis d'Informàtica, Multimimèdia i Telecomuncicació in the Open University of Catalonia (UOC) and works as a part-time assistant at the Universitat Politènica de Catalunya (UPC). From the latter, he earned his degree in Computer Science in 2002 and his Ph.D. degree in 2009. He has published several papers in international conferences and journals and has been invited as keynote speaker at several conferences. Both his teaching and research interests are related to the fields of networking and security, more specifically in peer-to-peer systems.

Jordi Herrera-Joancomartí is an associate professor at Department of Information and Communications Enginneering in the Universitat Autòma de Barcelona. He is graduated in Mathematics by Universitat Autòma de Barcelona in 1994 and he received his Ph.D. degree in 2000 from Universitat Polit?nica de Catalunya. His research interests include topics in the field of computer security and more precisely in privacy in social networks and security in distributed networks and RFID systems. He has published more than 80 papers in national and international conferences and journals and he has been main researcher in several research projects. He is now with the SeNDA research group.

Josep Prieto-Blázquez completed his Ph.D. degree in Computer Science in 2009 from the Universitat Oberta de Catalunya and received his Masters in Computer Science from the Universitat Politecnica de Catalunya in 1993. Since 1998 he has worked as a lecturer in the department of Computer Science, Multimedia and Telecommunication at the Universitat Oberta deCatalunya, where he has been a Director of the Computer Engineering (CE) programmesince 2001 and a Vice-Dean, in the same department, since 2009. His line of research centres on exploratory and application technology in the field of ICT. He has also participated in the wireless, free software and virtual learning environments projects and he is also a member of the Kriptography and Information Security for Open Networks (KISON) research group.

Bibliography

[1] J. Arnedo-Moreno, N. Pérez-Gilabert, and M. Domingo-Prieto. "Anonymously accessing JXTA community services through split message forwarding". In: *Mathematical and Computer Modelling* 58.5-6 (2013), pp. 1313–1327.

[2] J. Arnedo-Moreno, N. Perez-Gilabert, and M. Domingo-Prieto. "Experimental Evaluation of Anonymous Protocols under the JXTA Middlewar". In: *2012 15th International Conference on Network-Based Information Systems* (2012), pp. 52–57. DOI: http://doi.ieeecomputersociety.org/10.1109/NBiS.2012.28.

[3] À. García-Domínguez, M. Domingo-Prieto, and J. Arnedo-Moreno. "JXTA anonymity through a replicated message-based approach". In: *Actas de la XII Reunión Española sobre Criptologia y Seguridad de la Información (XI - RECSI)*. Publicacions URV., 2012, pp. 201–206. ISBN: 978-84-693-3304-4.

[4] J. Arnedo-Moreno and M. Domingo-Prieto. "An anonymity layer for JXTA services". In: *2011 Workshops of International Conference on Advanced Information Networking and Applications*. IEEE Press, 2011, pp. 102–107. ISBN: 978-0-7695-4338-3. DOI: 10.1109/WAINA.2011.9.

[5] P. Radmand et al. "ZigBee/ZigBee PRO Security Assessment Based on Compromised Cryptographic Keys". In: *1st International Workshop on Securing Information in Distributed Environments and Ubiquitous Systems*. IEEE Press, 2010, pp. 465–470. DOI: 10.1109/3PGCIC.2010.79.

[6] M. Dohler et al. "Smart Cities: An Action Plan". In: *Smart City Expo & World Congress. Dec*. 2011.

[7] B. Martinez et al. "The Power of Models: Modeling Power Consumption for IoT Devices". In: *IEEE Sensors Journal* 15.10 (2015), pp. 5777–5789. ISSN: 1530-437X. DOI: 10.1109/JSEN.2015.2445094.

[8] N. Sornin et al. "LoRaWAN Specifications". In: *LoRa™ Alliance* (2015).

[9] *Sigfox*. https://www.sigfox.com/. 2009.

[10] Thomas Watteyne, Maria Rita Palattella, and Luigi Alfredo Grieco. *Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement*. IETF, 2015.

[11] *senseiot*. http://www.sense-iot.com/. 2015.

[12] *ThingSpeak*. https://thingspeak.com/. 2010.

[13] *IPv6 over Low power WPAN (6LoWPAN)*. https://tools.ietf.org/wg/6lowpan/. 2007.

[14] *The Constrained Application Protocol (CoAP)*. https://tools.ietf.org/html/rfc7252. 2014.

[15] *thethings.io*. https://thethings.io/. 2015.

[16] G. Tanganelli, C. Vallati, and E. Mingozzi. "CoAPthon: Easy development of CoAP-based IoT applications with Python". In: *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*. 2015, pp. 63–68. DOI: 10.1109/WF-IoT.2015.7389028.

[17] *sensorthings*. https://github.com/opengeospatial/sensorthings. 2015.

[18] *Open Geospatial Consortium*. http://www.opengeospatial.org/standards. 1994.

[19] B. Cohen. *The BitTorrent protocol specification*. 2008.

[20] *Gnutella*. http://rfc-gnutella.sourceforge.net. 2000.

[21] Chimera. *Chimera*. http://current.cs.ucsb.edu/projects/chimera/index.html. 2006.

[22] FreePastry. *A substrate for peer-to-peer applications*. http://www.freepastry.org/. 2001.

[23] P2PNS. *P2PNS: A Secure Distributed Name Service for P2PSIP*. http://www.freepastry.org/. 2001.

[24] JGroups. *JGroups - A Toolkit for Reliable Messaging*. http://www.jgroups.org/index.html. 2002.

[25] freedomjs. *freedomjs*. http://www.freedomjs.org/. 2012.

[26] *Sip2Peer*. https://github.com/dsg-unipr/sip2peer. 2011.

[27] Sun Microsystems Inc. *JXTA v2.0 Protocols Specification*. https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html. 2007.

[28] smokindoug. *Cut and Paste with JXTA (Clipboard Manager)*. http://kenai.com/projects/candpjini. 2009.

[29] K. Matsuo et al. "Experimental Results and Evaluation of SmartBox Stimulation Device in a P2P E-learning System". In: 2009, pp. 37–44.

[30] Thomas Watteyne, Ankur Mehta, and Kris S.J. Pister. "Reliability Through Frequency Diversity: Why Channel Hopping Makes Sense". In: *Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN)*. Tenerife, Canary Islands, Spain, 2009.

[31] Diego Dujovne et al. "6TiSCH: Deterministic IP-enabled Industrial Internet (of Things)". In: *IEEE Communications Magazine* 52.12 (2014), pp. 36–41.

[32] Xavier Vilajosana and Kris Pister. *Minimal 6TiSCH Configuration*. IETF, 2015.

[33] Maria Rita Palattella et al. "Traffic Aware Scheduling Algorithm for Reliable Low-power Multi-hop IEEE 802.15.4e Networks". In: *International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*. IEEE. Sydney, Australia, 2012, pp. 327–332.

[34] Andrew Tinka et al. "A Decentralized Scheduling Algorithm for Time Synchronized Channel Hopping". In: *Transactions on Mobile Communications and Applications* 11.1 (2011), pp. 201–216.

[35] Antoni Morell et al. "Label Switching over IEEE802.15.4e Networks". In: *Wiley Transactions on Emerging Telecommunications Technologies* 24.5 (2013), pp. 458–475.

[36] Nicola Accettura et al. "Decentralized Traffic Aware Scheduling for multi-hop Low power Lossy Networks in the Internet of Things". In: *International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE. 2013, pp. 1–6.

[37] Simon Duquennoy et al. "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH". In: *International Conference on Embedded Networked Sensor Systems (SenSys)*. ACM. Seoul, South Korea, 2015.

[38] N. Kimura and S. Latifi. "A survey on data compression in wireless sensor networks". In: *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*. Vol. 2. 2005, 8–13 Vol. 2. DOI: 10.1109/ITCC.2005.43.

[39] JonathanGana Kolo et al. "A Simple Data Compression Algorithm for Wireless Sensor Networks". In: *Soft Computing Models in Industrial and Environmental Applications*. Ed. by Václav Snášel, Ajith Abraham, and Emilio S. Corchado. Vol. 188. Advances in Intelligent Systems and Computing. Springer Berlin Heidelberg, 2013, pp. 327–336. ISBN: 978-3-642-32921-0. DOI: 10.1007/978-3-642-32922-7_34.

[40] C. Schurgers and M.B. Srivastava. "Energy efficient routing in wireless sensor networks". In: *Military Communications Conference, 2001. MILCOM 2001. Communications for Network-Centric Operations: Creating the Information Force. IEEE*. Vol. 1. 2001, 357–361 vol.1. DOI: 10.1109/MILCOM.2001.985819.

[41] Tijs van Dam and Koen Langendoen. "An Adaptive Energy-efficient MAC Protocol for Wireless Sensor Networks". In: *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*. SenSys '03. Los Angeles, California, USA: ACM, 2003, pp. 171–180. ISBN: 1-58113-707-9. DOI: 10.1145/958491.958512. URL: http://doi.acm.org/10.1145/958491.958512.

[42] Chaupal. *The Chaupal Organisation*. https://chaupal.github.io/index.html. 2013.

[43] Chaupal. *Chaupal Releases*. http://chaupal.github.io/releases.html. 2015.

[44] S. Krco, D. Cleary, and D. Parker. "Enabling ubiquitous sensor networking over mobile networks through peer-to-peer overlay networking". In: *Computer Communications* 28.13 (2005), pp. 1586 –1601.

[45] B. Lim, K. Choi, and D. Shin. "A JXTA-based Architecture for Efficient and Adaptive Healthcare Services". In: *Information Networking. Convergence in Broadband and Mobile Networking*. Vol. 3391. Lecture Notes in Computer Science. 2005, pp. 776–785.

[46] A. Antoniou et al. "A peer-to-peer environment for monitoring multiple wireless sensor networks". In: *Proceedings of the 2nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*. ACM, 2007, pp. 132–135.

[47] J. Arnedo-Moreno and J. Herrera-Joancomartí. "A survey on security in JXTA applications". In: *Journal of Systems and Software* 82.9 (2009), pp. 1513–1525.

[48] L. Kawulok, K. Zielinski, and M. Jaeschke. "Trusted group membership service for JXME (JXTA4J2ME)". In: *Wireless And Mobile Computing, Networking And Communications, 2005.(WiMob'2005), IEEE International Conference on*. Vol. 4. IEEE. 2005, pp. 116–121.

[49] R. Housley et al. *Internet x.509 public key infrastructure*. http://www.ietf.org/rfc/rfc2459.txt. 1999.

[50] G. Dodig-Crnkovic. *Theory of Science*. 2003.

[51] Thomas Watteyne et al. "OpenWSN: a Standards-based Low-power Wireless Development Environment". In: *Transactions on Emerging Telecommunications Technologies* 23.5 (2012), pp. 480–493.

[52] W. Mckinney. *Python Data Analysis Library*. http://pandas.pydata.org/. 2008.

[53] F. Pérez. *The Jupyter Notebook*. https://ipython.org/notebook.html. 2014.

[54] MathWorks. *Matlab*. http://mathworks.com/products/matlab. 1984.

[55] M. Domingo-Prieto et al. "Distributed PID-Based Scheduling for 6TiSCH Networks". In: *IEEE Communications Letters* 20.5 (2016), pp. 1006–1009. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2016.2546880.

[56] Qin Wang and Xavier Vilajosana. *6TiSCH Operation Sublayer (6top)*. IETF, 2015.

[57] M. Domingo Prieto et al. "Balancing power consumption in IoT devices by using variable packet size". In: *Complex, Intelligent and Software Intensive Systems (CISIS), 2014 Eighth International Conference on*. IEEE. 2014, pp. 170–176. DOI: 10.1109/CISIS.2014.25.

[58] *Worldsensing*. http://www.worldsensing.com. 2010.

[59] Xavier Vilajosana et al. "A Realistic Energy Consumption Model for TSCH Networks". In: *IEEE Sensors* (2013).

[60] Davide Zordan et al. "On the Performance of Lossy Compression Schemes for Energy Constrained Sensor Networking". In: *[early version: arXiv:1206.2129]. Accepted for Publication. ACM Transactions on Sensor Networks* (2014).

[61] David A Huffman. "A method for the construction of minimum-redundancy codes". In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101.

[62] M. Domingo-Prieto, J. Arnedo-Moreno, and X. Vilajosana-Guillen. "jxSensor: A Sensor Network Integration Layer for JXTA". In: *2012 15th International Conference on Network-Based Information Systems* 0 (2012), pp. 435–440. DOI: http://doi.ieeecomputersociety.org/10.1109/NBiS.2012.125.

[63] M. Domingo Prieto et al. "Towards secure mobile P2P applications using JXME". In: *Journal of Internet Services and Information Security (JISIS)* 2.1/2 (2012), pp. 1–21.

[64] L. Kawulok, K. Zielinski, and M. Jaeschke. "Trusted group membership service for JXME (JXTA4J2ME)". In: *Wireless And Mobile Computing, Networking And Communications, (WiMob'2005), IEEE International Conference on*. Vol. 4. 2005, pp. 116–121.

[65] M. Domingo-Prieto and J. Arnedo-Moreno. "JXTAnonym: An anonymity layer for JXTA services messaging". In: *IEICE TRANSACTIONS on Information and Systems* 95.1 (2012), pp. 169–176.

[66] Goldsclag D. Syverson P. and Reed M. "Anonymous connections and onion routing". In: *Proceeding of the IEEE 18th Annual Symposium on Security and Privacy* (1997), pp. 44–54.