# Harnessing XMPP for Machine-to-Machine Communications & Pervasive Applications

Antti Iivari, Teemu Väisänen, Mahdi Ben Alaya, Tero Riipinen, and Thierry Monteil

*Abstract* — **An ever increasing number of interconnected embedded devices, or Machine-to-Machine (M2M) systems, are changing the way we live, work and play. M2M systems as a whole are typically characterized by the diversity in both the type of device and type of network access technology employed, and such systems are often still today task-specific and built for just one specific application. Smart lighting, remote monitoring and control of all kinds of consumer devices and industrial equipment, safety and security monitoring devices and smart health and fitness products, exemplify this revolution of intercommunicating machines. However, the differences in communication technologies and data formats among such devices and systems are leading to a huge complexity explosion problem and a strongly fragmented market, with no true interoperability. Due to these problems, the full potential of M2M technology has yet to be fulfilled. In this paper, we examine the suitability of the Extensible Messaging and Presence Protocol (XMPP) and experiment with its potential to rise to the challenge of machine-to-machine communications and meet the needs of modern pervasive applications. Experimental implementations and some proof-of-concept solutions are also presented.**

*Index terms* — **Machine-to-Machine, XMPP, Internet-of-things, Interoperability, Applications**

## I. INTRODUCTION

Machine-to-Machine (M2M) refers to the automated exchange of information between servers, sensors, actuators, and various end devices such as mobile phones, vending machines, vehicles, and personal computers. All kinds of information exchange between various kinds of communicating machines, with limited or no human intervention, can be referred to as M2M communication. This gives rise to many obvious advantages such as real time data exchange between devices in the field, remote monitoring and operation according to real needs, immediate reactivity to business information, automatic processing of consumption statistics and operational data are made readily available. Different application domains are brought together by M2M technology. Data communication takes place via established ICT (Information and Communication Technology) infrastructure and communication media is based on both fixed and wireless (mobile) technology. Indeed, such devices are longer restricted to communicating only with other machines in the same local system, or even in the same application domain.

In essence, M2M (Machine-to-machine) [1] is always a mixture of various kinds of electronic devices, communication technologies and software implementations. M2M can be defined simply as machines intercommunicating without, or with limited, human intervention.

M2M constitute systems that enable the wireless and wired devices, both big and small, to exchange information with other devices and actors included in the M2M ecosystem. Typically, devices such as sensors and meters are used to capture raw data which is then transferred through a network to the M2M applications. These pieces of data are then transformed into meaningful pieces of information by the background system. Any interoperable M2M system will obviously need to support a mixture of legacy and modern technologies and protocols, which is already a tough challenge in itself. Furthermore, to have these machines really intercommunicate in a meaningful manner, it is not enough to simply enforce compatible communication protocols, as the transmitted information itself must also be processed and stored in a format that is accessible to both man and machine. Technologies used commonly in M2M include naming and identification of M2M entities, service discovery (SD), security including, e.g., authentication and encryption techniques, service platforms, wireless radios, data formats and communication protocols. Many different technologies and standards exists, all attempting to provide a solution for one or more of these issues given rise by M2M and pervasive applications, but it quickly becomes evident that the technology to unleash the full potential of M2M is still missing [1],[2].

In the second chapter of this paper, we will begin with discussion on machine-to-machine communications and related technologies in general. Then we will continue with more in-depth content on the main topic of this work, the XMPP protocol and a study on its applicability for M2M purposes in the third chapter. After this, carrying on to the fourth chapter, the experiments and proof-of-concept implementations based on the XMPP-technology will be presented. Finally, in the conclusions, we will summarize the work, discuss the results and consider some of the items that are still left for future works.

## II. MACHINE-TO-MACHINE COMMUNICATIONS

Machine-to-machine systems blur the line between the physical- and virtual worlds, making use of various communication technologies to enable remote monitoring, control, updating and interaction with all sorts of communication enabled machines and asset devices in any application domain. Typically, these interactions occur without or with limited human interaction. In this section certain characteristics, requirements, technologies and standards relating to M2M are discussed.

Machine-to-machine systems, from a communication engineering point of view, are an especially challenging endeavour. This is due to the fact that these systems typically consist of myriads of inter-communicating devices with extremely heterogeneous characteristics, requirements and capabilities operating in various application domains and environments [3]. From the tiniest power-constrained

Manuscript received May 9, 2014; revised September 5, 2014.

A. Iivari, T. Väisänen and T. Riipinen are with the VTT Technical Research Centre of Finland, Kaitoväylä 1, Oulu FI-90571 Oulu, Finland (e-mail: { antti.iivari, teemu.vaisanen, tero.riipinen}@vtt.fi).

M. B. Alaya and T. Monteil are with the LAAS-CNRS,7 avenue du Colonel Roche, 31077 Toulouse, France (e-mail: {ben.alaya, monteil}@laas.fr).

sensors and actuators to computationally powerful smart devices and servers, these complex distributed systems must be designed from the ground up with considerations such as; scalability, security, interoperability, flexibility, extensibility and robustness always at the forefront. In addition, with M2M systems vastly different network access methods and wired or wireless communication technologies are employed as required by the application domain at hand [1]. A M2M system always includes software services in addition to the actual machines and hardware devices, in one form or another. Indeed, one of the key challenges in designing interoperable M2M systems is the software that enables horizontal interoperability between various services and software agents. Existing M2M solutions are typically dedicated to a single specific application and serve only a limited set of usage scenarios or a single business need. In these cases, the software and hardware, sometimes even the communication protocols, are tailor-made for a certain proprietary products. This can be seen as "vertical M2M", where the developed solutions are customized and fitted specifically for a single solution, with no concern for interoperability or the bigger picture. Also, the multitude of technologies and competing standards, none of which are truly interoperable or able to understand each other, are causing major issues for engineers working within the M2M field. Consequently, there is a dire need for interoperable and future-proof M2M technologies that ensure horizontal interoperability between M2M actors and applications while still providing all the necessary functionalities to cover a multitude of application domains [3],[4].

End-to-end M2M communication can be established with one or more protocol conversion gateways, but the trend is to build systems where real end-to-end communication is possible, for example using IPv6 as advocated by Internet Protocol for Smart Objects (IPSO) Alliance [5]. In the literature, technologies enabling interoperable M2M communication are called M2M or IoT [6] middleware, other such as smart environment middleware [7] or just middleware [8], M2M overlays [9], service infrastructures for IoT [10] or for M2M, private M2M service space [11], M2M service networks or just M2M communication [12]. In recent years, there have been also research efforts to enable capabilities related to autonomic computation, opportunistic communication and self-configuration in the context of M2M and IoT. [13], [14].

### A. *Characteristics and requirements for M2M Communication*

The full scope of the M2M communication challenge is probably best elucidated by a high-level system diagram. An abstract overview diagram illustrating the various aspects and main functionalities of a complete M2M communication system is shown in Fig. 1. On the left side of the image, various existing systems are portrayed within their own small areas. These systems are connected then connected to the rest of the system via M2M gateways. The overlay communication between the different gateways, M2M devices (those that are not behind the gateway) and servers takes place over existing communication infrastructure. As already discussed earlier, the M2M system must be able to operate efficiently over existing ICT infrastructure utilizing internet protocols (IP) wherever possible and function correctly regardless of the existing technologies or protocols underneath. There are several implications from this, and we will discuss them in this section.

Secure, efficient and adaptive M2M message exchange over existing ICT-infrastructure in an overlay-type [15] of manner will become a critical part of the interoperable machine-to-machine solution. On top of the IP stack we will need to have some kind of an open and extensible (e.g. XML-based) messaging solution, that will effectively create an overlay or ,"a logical network on top of a network", if you will. Machine-to-Machine clients, asset devices, back-ends and

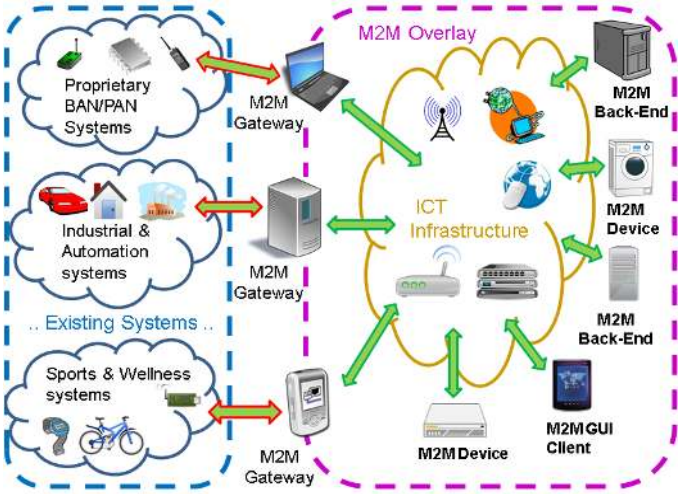


Fig. 1. A high-level overview of a M2M system.

gateways will exchange M2M messages with one another without the underlying IP world necessarily understanding anything about them. The regular IP-nodes are just relaying and routing them forward according to the legacy rules like any other packet over the internet. As not every modem, router or device along the way will directly be part, or even aware, of the M2M ecosystem, an overlay network will inevitably be formed [15], [16].

One of the most basic requirements for the system is the ability for simple, effective and secure client-server communication. This means, essentially, any communication from M2M clients or gateways towards the M2M server(s). The server side will effectively provide querying devices information on any other available devices, services and resources connected to the M2M ecosystem as required by pervasive applications and ubiquitous environments [17]. The servers can be seen to act as central points or hubs that collect and store data on nearby entities, their status, availability, services, resources, etc. The server will also be critical in the process for authentication and establishing trust between M2M devices. Also, communication between servers, or server federation, must be supported. Relying on devices and services to function properly using only pure peer-to-peer and ad-hoc mechanisms, such as flooding routing/discovery queries of various kinds, in a large-scale M2M ecosystem is not realistic especially when the scalability requirements are a part of the equation. Therefore, we assume that certain server side functionalities are required and form another key component of the interoperable M2M communication system. Furthermore, the interoperable M2M system must support multiple M2M servers working together and being part of the same overall system through secure server-to-server communication. The servers might even be operated by different service providers all sharing the goal of enabling truly interoperable M2M capabilities for their products and services for added value.

The key functionalities for M2M systems can also be looked at in the form of a simplified layered structure, as demonstrated in Fig. 2, where a subset of standards and technologies relevant for M2M are arranged into a stack-like layer diagram for illustration purposes. The layers in the stack correspond to the various level of abstract functionality within an M2M ecosystem to further illustrate where these technologies can be mapped to for a M2M system design. Thus, the diagram consists of three technology layers; one for devices, another for messaging and the third one for M2M information and services. M2M security, as a cross-layer issue, is also evident in the left hand side of the diagram to emphasize that fact that to bring forth a robust and trustworthy design for a system of intercommunicating machines from various application domains, concerns relating to

security and privacy must be seriously taken into account throughout all aspects of the system design. At this point, it must be emphasized that this division of M2M functionalities into three layers herein is purely an abstract notion to structure and classify conceptually the abundance of information and as such, it should not be compared to e.g. the Open Systems Interconnection model (OSI).

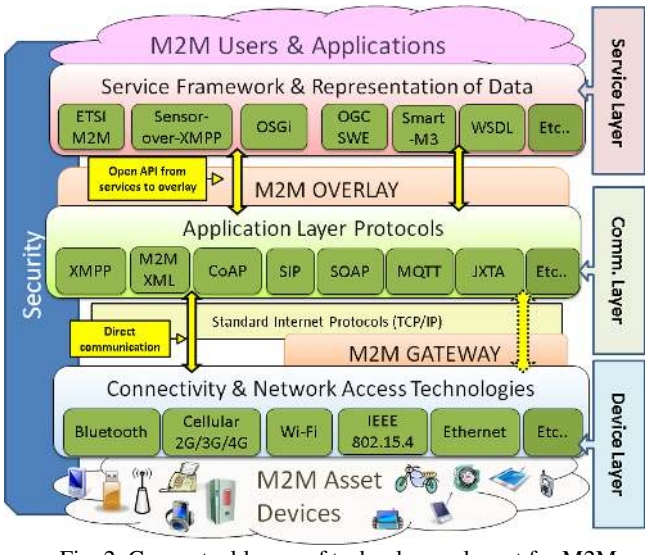


Fig. 2. Conceptual layers of technology relevant for M2M.

To gain a deeper understanding of the M2M problem, we shall discuss the layer diagram a bit further. At the lowest level there are the various physical end-devices themselves of the M2M-system constituting what is herein referred to as the "M2M Device layer", and these devices – ranging from miniscule sensors to large industrial machinery – operate with vastly different technologies, operational characteristics and application environments. These are commonly also referred to as M2M asset devices. In most cases on the field today, the machines are then connected via some network access technology - wired or wireless - to either a larger backbone network, some kind of central hub or only form a limited private area network. Typically, radio technologies such as Bluetooth, ZigBee or Wi-Fi are utilized by the myriads of such devices in addition to cellular communication, different wired interfaces and proprietary technologies of varying kind.

In the middle of the layer diagram, we have the concept of "M2M Communication Layer" which basically consists of an application layer communication protocol with an API for a communication platform enabling any and all messaging operations as required by the users, applications and services (both man and machine) of the M2M Ecosystem. This is, yet again, related to the idea of overlay communication, or logical networks on top of networks. The overlay communication functionality operates over existing ICT infrastructure and standard internet protocols (IP), hiding the complexities and problems associated with the different communication technologies underneath it. Now, between the device layer and the communication layer for M2M, a gateway may or may not be needed, as will be discussed in this paper. This is depending on whether the asset device is capable of handling the communication protocols, such as the chosen application layer protocol and IP, and other operations as employed by the rest of the machine-to-machine overlay ecosystem by itself. There are several application layer protocols that are suitable and have the potential to be employed for secure IP-based M2M message exchange, some of which are currently being studied and experimented with, such as XMPP. In our layer diagram, communication between the M2M device – and communication layers is shown to happen either directly, or through the M2M Gateway, as discussed previously. In short, The M2M overlay offers a set of communication capabilities to the service layer via an open API (application programming interface) upon which the services call upon when certain communication operations and functionalities are required by the applications or services.

Finally, at the very top of our layered abstraction in Fig. 2, we have the "M2M Service Layer", which can be seen as a selection of software platforms, service frameworks, data formats, information management solutions and other relevant tools and technologies. The service layer consists of everything needed for the users and applications to start utilizing the M2M ecosystem for the needs of the application domain at hand. The service layer calls upon and utilizes the communication mechanisms provided by the M2M communication layer below via an open API, as shown in the layer diagram. To summarize, the service layer is essentially the foundation upon which the applications and services for the end-user (whether human or machine) are built. The service layer utilizes the communication capabilities provided by the communication layer, which in turn hides the challenges at the lower levels (device heterogeneity, different communication protocols, radio technologies, etc.) by acting as a platform for M2M Overlay networking. In certain cases, the gateway is utilized to facilitate communication from the various asset devices to the rest of the overlay, and vice versa. Also, some of the technologies seen in the layer-diagram are discussed further within this document. While the service layer components on the diagram each belong to their own technological domain, interoperability between applications built based on varying technologies can be achieved by means of interoperability gateways or interworking proxies, which will be discussed further within this paper.

### *B. M2M Gateways for Interoperability*

It is clear that steps must be taken to ensure interoperability with existing commercial and resource constrained systems and devices to the extent reasonably possible. Another critical issue is the efficient utilization of the entire existing communication infrastructure, especially standard internet technologies, for secure and interoperable M2M message exchange. There are way too many competing protocols, tools, technologies and standards already in the field and, therefore, efforts should be made on finding viable approaches by combining and enhancing selected existing widely accepted technologies and harnessing them to serve M2M in new and interesting ways. The ultimate goal for M2M is to provide a functional backbone consisting of certain key universal building blocks in order to enable various mechanisms for applications, services and users to interact, operate, exchange messages, utilize each other's capabilities, share resources and discover each other in ways that are fitting and are useful in any application domain where M2M technology is applied. Furthermore, important security policies, such as authentication and channel encryption, must be built into the system and enforced. Indeed, for true M2M interoperability, it is simply not enough to design new communication protocols or service platforms and expect it to solve all the problems and everyone to suddenly start deploying it. There are already various existing systems and technologies on the field in operation which need to be taken into account, without altering them or affecting their current modes of operation. Issues related to interoperability with existing systems and extremely resource constrained devices are typically approached by building and designing M2M gateways to facilitate the communication to and from an auxiliary system or device [18]. This way, the gateway will handle the issues related to communicating with either a system based on an incompatible communication protocol, or resource constrained devices which are unable to communicate with the rest of the system directly due to limited

resources or capabilities. While some might consider the effort of design and deployment of an additional gateway component to be superfluous or suboptimal, the fact remains that having two systems with different protocols and data formats intercommunicate is simply not possible without a translating element of some sort. Other advantages of the gateway approach include congestion control for core networks (prevent various signalling messages from the devices from flooding the backbone network), convenient management of the end-devices in groups, easy deployment of automatic maintenance and configuration operations and the potential to leverage unlicensed frequency bands for the devices operating "behind" the gateway. Typically, gateways such as these are then connected to a server or a background system, as in most cases also a back-end side of some sort is also an important part of the overall architecture handling operations related to data storage, management, centralized control and enforcement of important security policies [18], [19]. From a communications point of view, the use of servers and background systems can also be seen to have a crucial role in combatting the various scalability and reliability issues found in pure peer-to-peer and ad-hoc -type systems.

### C. *Current standardization activities for M2M*

As the growth of the M2M communications industry has led to a clear need for standards and interoperability for M2M technologies, most of the major ICT standardization organizations have formed Machine-to-Machine related working groups. Some of these standardization activities are briefly discussed in this section.

The European Telecommunications standardization Institute (ETSI) [20], 3rd Generation Partnership Project (3GPP) [21], the Telecommunications Industry Association (TIA) of the USA [22], and Institute of Electrical and Electronics Engineers (IEEE) [23] are some of the main standardization bodies that are putting significant effort in investigating and identifying M2M challenges, issues and architectures. Perhaps the most encompassing mission is the one undertaken by oneM2M [24], in which the members are working towards the goal of developing technical specifications and reports to ensure M2M devices can successfully share information on a global scale through a common horizontal service platform.

### D. *The ETSI M2M Architecture*

Due to the fact that M2M has been declared as one of the main strategic topics of ETSI's standardization work, the ETSI has formed a M2M technical committee to facilitate the standardization of a M2M application layer. This committee in general has a clear focus on the service and application middleware rather than the actual M2M network and communication techniques. It has been tasked with the goal to develop an end-to-end overall high level M2M architecture. The committee includes experts and researchers from Europe, America and Asia representing research centres, telecommunication operators, device vendors, etc. All automated exchange of information between machines (even virtual ones) with limited human end-user influence is considered machine-to-machine communication. The focus is to develop a horizontal Service Capability Layer (SCL) that can be used by multiple M2M vertical applications. The group aims to provide an end-to-end view of Machine to Machine standardization that remains agnostic to the telecommunication technologies used at the lower layers.

The ETSI M2M high level architecture contains a Device and Gateway domain and a Network domain as described in Fig 3. An M2M Device runs M2M Applications using the SCL. It can connect directly to the Network Domain via the Access network and may provide service to other devices. It can also connect to the Network Domain via a Gateway through a Local Area Network. A Gateway also runs M2M Applications using the SCL, and acts as a proxy between local devices and the network domain. The access network allows M2M devices and gateways to communicate with the Core Network. The SCL provides functions that can be shared by different Applications. Network Management Functions enables to manage the Access and Core Networks, such as Provisioning, Supervision, and Fault Management. M2M Management Functions consist of all the functions required to manage the SCL in the Network Domain.

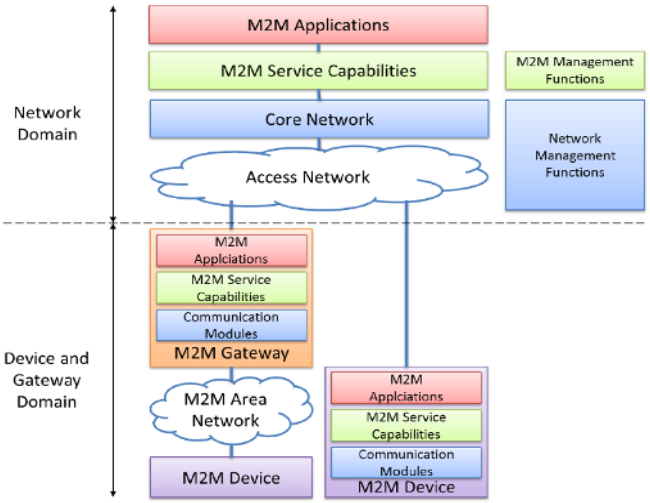


Fig. 3.ETSI M2M high level architecture

Fig. 4 depicts the ETSI M2M functional architecture. An SCL includes mandatory service capabilities such as application enablement, generic communication, reachability, addressing and repository, remote entity management, security. It may include optional service capabilities such as history and data retention, transaction management, telco operator exposure, and interworking proxy. An SCL can be deployed on an M2M network (NSCL), a gateway (GSCL), or a device (DSCL). Several primitive procedures are defined to enable machine registration, synchronous and asynchronous communication, resource discovery, access rights management, group broadcast, etc. Three reference points based on open APIs are specified: mIa, dIa, and mId. The mIa reference point allows a Network Application (NA) to access the NSCL. The dIa allows a Device or Gateway Application (D/GA) to access the D/GSCL. The mId reference point allows a D/GSCL to access the NSCL. These interfaces are defined in a generic way to support a wide range of network technologies and protocols to enhance interoperability.

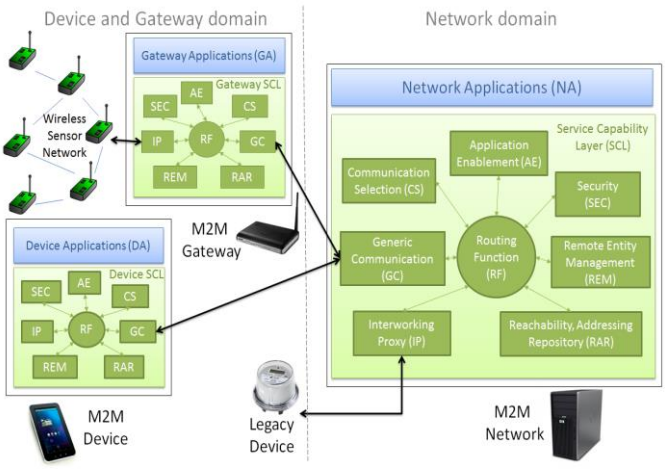


Fig.4. ETSI M2M functional architecture.

ETSI M2M adopts a RESTful architecture style. Each SCL contains a standardized resource tree where the information is stored. A resource is uniquely addressable via a Universal Resource Identifier (URI), and has a representation that can be transferred and manipulated with verbs (e.g. retrieve, update, delete, and execute). Each SCL is defined using a resource tree including different kind of resources. The "sclBase" resource describes the hosting SCL, and is the root for all other resources within the hosting SCL. The "scl" resource stores information related to SCLs residing on other M2M machines after successful mutual authentication. The "application" resource stores information about the application after a successful registration on the hosting SCL. The "container" resource acts as a mediator for data buffering to enable data exchange between applications and SCLs. The "contentInstance" resource represents a data instance in the container. The "accessRight" resource manages permissions and permissions holders to limit and protect the access to the resource tree structure. The "group" resource enhances resources tree operations by adding the grouping feature. The "subscription" resource allows subscribers to receive asynchronous notification when an event happens such as the reception of new sensor event or the creation, update, or delete of a resource. The "announced" resource contains a partial representation of a resource in a remote SCL to simplify discovery request on distributed SCLs. The "discovery" resource acts as a search engine for resources. The collection resource groups common resources together.

### *E. The OM2M project*

The OM2M project [25] provides an open source implementation of the horizontal service platform for M2M interoperability based on the ETSI-M2M standard. OM2M follows a RESTful architecture style with open interfaces to enable developing services and applications independently of the underlying network. It provides a simple API based on simple primitive procedures to enable machines authentication, resources discovery, applications registration, containers management, synchronous and asynchronous communications, access rights authorization, groups organisation, re-targeting, etc. OM2M proposes a modular architecture running on top of an OSGi layer, making it highly extensible via plugins. It supports multiple protocol bindings such as HTTP and CoAP. The OMA-DM protocol is used as a remote entity management service to perform efficient software updates and life-cycle managements on constraint devices. The TLS-PSK protocol is considered to enable secure communication based on pre-shared keys, and various interworking proxies are provided to enable seamless communication with legacy devices such as Zigbee and Phidgets technologies. OM2M has also been accepted as an open source project by the Eclipse Foundation in the IoT working group [26].

Furthermore, The XMPP-based experimental Interworking Proxy implementation discussed later in this paper is built to work together with the OM2M implementation of the ETSI M2M reference platform.

### *F. oneM2M*

ETSI is one of the main contributors to oneM2M, which contains members from already mentioned affiliations ETSI and TIA, but also industrial and research organizations from Open Mobile Alliance (OMA), Association of Radio Industries and Businesses (ARIB), Alliance for Telecommunications Industry Solutions (ATIS), China Communications Standards Association (CCSA), and Telecommunications Technology Association (TTA) of Korea, and Telecommunication Technology Committee (TTC) of Japan. oneM2M shall prepare, approve and maintain the necessary set of specifications and reports, e.g., for use cases and requirements, service architecture, open interfaces and protocols, interoperability, identification and naming of devices and applications and management aspects. Current specifications of oneM2M can be found from the oneM2M website [24].

### *G. IEEE's M2M standardization*

It could be said that IEEE has done M2M related standardization at lower level than ETSI and other members of oneM2M. IEEE already provides several wired and wireless communication standards for the purpose of enabling communication between M2M devices, but is still developing new ones and enhancing the existing ones to support M2M and IoT communications even further. For example, IEEE 802.11ah standardization task group is developing Wireless LAN standard suitable for M2M and IoT communication by creating large group of stations that cooperate to share air medium while minimizing energy consumption. Another IEEE's M2M related standardization activity is IEEE 1451 standard family. IEEE 1451 describes communication for the smallest devices for low communication range.

IEEE 1451.x standards specify a physical interface for data interconnection (PHY), which might use mixed mode analog and digital [27], digital point-to-point transducer to microprocessor [28], distributed multidrop bus [29], RFID [30], or Wireless [31] communication. Network Capable Application Processor (NCAP) module includes Common Functionality and Transducer Electronic Data Sheets (TEDS) defined in [32], and Smart Transducer Object Model [33]. TEDS is a set of electronic data in a standardized format specifying type, identification information, interfaces, technical information such as sensitivity, calibration, correction data, bridge type, manufacturer-related information, excitation, etc. of the transducer. TEDS is a memory device attached to the transducer (sensor or actuator). There are TEDS for each 1451.x communication mechanisms.

Purpose of IEEE project P21451-1-4 - Standards for a Smart Transducer Interface for Sensors, Actuators, and Devices - eXtensible Messaging and Presence Protocol (XMPP) for Networked Device Communication is to define a method for transporting IEEE 1451 messages over a network using XMPP to establish session initiation, secure communication, and characteristic identification between networked client and server devices using device Meta identification information based on the IEEE 1451 Transducer Electronic Data Sheets (TEDs). [34]. Furthermore, the ISO/IEC/IEEE P21451-1-4 will use a JID (EUI-64), an Universal Unique IDentifier (UUID), which is defined in ISO 29161 Automatic Identification for the Internet of Things. ISO 29161 is developed by ISO JTC1 SG31 WG2 Automatic Identification & Data Capture and TC122 Internet of Things (IoT).

Third IEEE's M2M related standardization activity is done in wireless broadband IEEE 802.16 family (commercialized under the name WiMAX). IEEE 802.16's M2M Task Group [35] has defined 802.16p [36] that describe how to enhance WirelessMAN-OFDMA Air Interface for broadband wireless access systems to enable support for modern M2M applications.

## III. THE EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL (XMPP)

XMPP is a set of open XML technologies for presence and real-time communication originally created for near-real-time messaging, presence, and request-response services [37], [38]. XMPP is described in [39],[40],[41], and updated by several Internet Drafts published by Internet Engineering Task Force (IETF) and by XMPP extension protocols (XEPs) published by the XMPP Standards

Foundation (XSF). In this chapter we will present the main features and architecture of XMPP technology. Also, as its name suggests, one of the key features of XMPP stems from its extensibility and we will therefore also discuss some of the most relevant XMPP extensions herein.

XMPP has gained quite a momentum recently. Although, it has been awhile since XMPP technology (or as it was called back then, Jabber) was first invented and released in 1998-1999, today the technology itself is alive and doing better than ever. After publishing the core XMPP specification in 2004 as RFCs, XMPP technology got adopted widely by bigger and smaller companies. For example, in August 2005 Google Talk IM and Voice over Internet Protocol (VoIP) services were released both based on XMPP technology. Significant companies using XMPP in their products include Apple, Cisco, Google, IBM, Nokia and Sun, to name a few. Today's trendy instant messaging service WhatsApp, which is also based on XMPP technology, although customized to fit their needs, now has over 350 million monthly users. XMPP is also utilized as a basis for many other commercial products and solutions, and is used, e.g., in gaming, geolocation and cloud computing [37], but also in sensing of campus areas [42], smart cities [43] and home environments [44],[12],[45], public transport [46], disaster management [47],[48], to retrieve Internet Abuse handling related information, in smart grids [49] and cyber security exercises, and in medical instruments and systems to monitor patients [6],[50],[51]. The potential of the XMPP technology is huge and, as we can see, some of the biggest companies are utilizing that potential on their top end products and services. Although, the examples given above are mostly related to the interaction between humans, XMPP's potential also for the M2M communication should not be ignored. Same kind of operations are needed for both human-to-human and machine-to-machine communications but the M2M systems need to be more automated, robust and smart. Regardless, the XMPP technology offers a great platform to implement M2M systems over it. There are already many communities, researchers or research groups who are studying the XMPP's potential on applications such as M2M or IoT fields [52], [10], [42], [46], [6], [53], [54], [7], [55], [56], [9], [57], [58], [59], [60], [61], [62].

One of these efforts is published by R. Klauck and M. Kirsche on their article "Chatty things - Making the Internet of Things readily usable for the masses with XMPP" [53]. Many other parties have been discovering the untapped potential of XMPP for the M2M world, strengthening the decision to employ and investigate the potential of XMPP within the context of M2M.

### A. *XMPP Features and the distributed client-server architecture*

The Extensible Messaging and Presence Protocol (XMPP) is a communication protocol based on the Extensible Markup Language (XML) providing exchange of structured yet extensible data between network entities in near-real-time. XMPP has similarities to other application-layer protocols like SMTP. In these architectures, a client with a unique name communicates with another client with a unique name through an associated server. Each client implements the client form of the protocol, where the server provides routing capability. Servers can also communicate for purposes of routing between domains. Farther, gateways can exist for purposes of translation between foreign messaging domains and protocols. As an extensible protocol, XMPP is an ideal backbone protocol to provide universal connectivity among different endpoint protocols. The XMPP gateway permits the termination of a given client-to-server (C2S) session and the initiation of a new session to the target endpoint protocol. Gateways are most often used in this context to translate between instant messaging (IM) protocols (for example, XMPP to Internet Relay Chat) or translate XMPP and other protocols, such as Bluetooth to control Lego NXT and Wii Remote [10].

The Extensible Messaging and Presence Protocol has been developed to enable message oriented communication services applicable in the internet context. XMPP can be seen as a communication overlay protocol as it operates on top of TCP/IP. The XMPP communication architecture is based on decentralized client-server model with also server-to-server (S2S) and serverless communication capabilities which can optionally be used. Each XMPP client has an account hosted by a XMPP server, and the client can be addressed by unique Jabber ID (JID). A client connects to the server to send and receive messages, to enable client-to-client (C2C) messaging over multiple domains the servers are responsible of routing messages to each other and all the way to the intended receiving client. Originally XMPP was developed for human to human, instant messaging (IM), communications; however, it can also be quite easily applied for M2M communications and many other tasks. One of the core features of the XMPP is the capability to handle presence information of the networked entities. Presence messages are used to detect the availability of each client. The information about client's presence is shared only with XMPP users that are in the roster/address book of sending client.

XMPP is essentially based on distributed client-server architecture, using at most three hops on the communication path from one client to another client. Architecture is presented in Fig. 5, where line number 1 presents C2S communication, line 2 presents S2S communication and line 3 presents direct C2C communication, for example a VoIP call that is negotiated using XMPP (that has been configured through servers in domains X and Z, but does not use XMPP for transmitting the actual VoIP data).

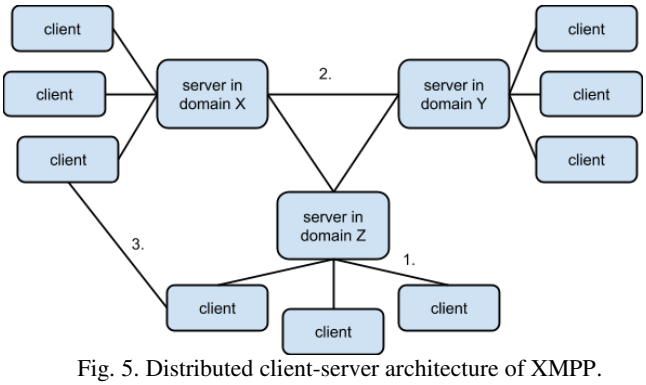


Fig. 5. Distributed client-server architecture of XMPP.

Of course, other types of XMPP communication also exist, some of which do not require servers at all, such as the one defined in XEP-0174: Serverless Messaging [63].

### B. *XMPP's publish-subscribe architecture*

XMPP offers publish-subscribe ("pubsub") functionality [64], visualized in Fig. 6, so polling is not always necessarily required between clients and services to disseminate information. Continuous polling can be seen as a wasteful operation in the case of small embedded M2M devices for example. The idea of publish-subscribe is simple, but the specification itself is larger:

- An entity publishes information to a node at a publish-subscribe service.
- The pubsub service pushes a notification to all entities that are authorized to learn about the published information.

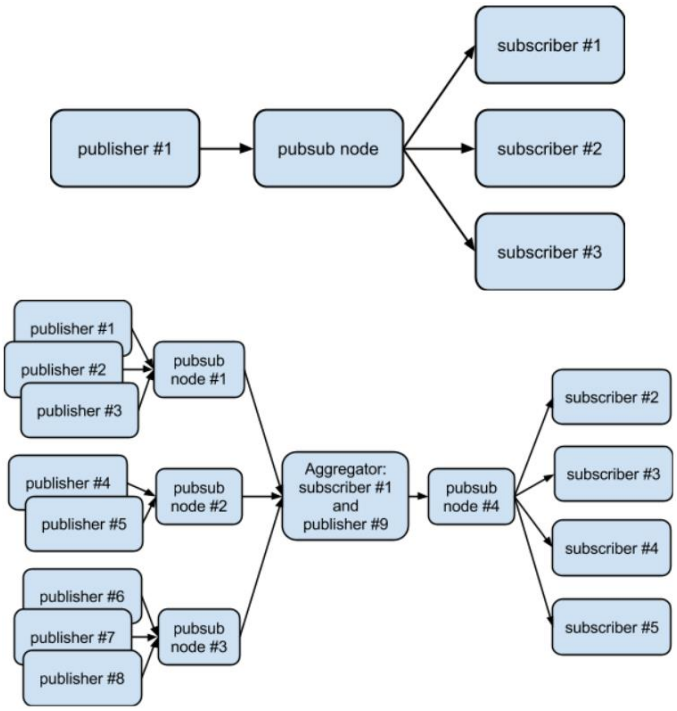


Fig. 6. Publish-subscribe functionality in its basic form and the principle of using pubsub for aggregators.

The specification for pubsub [64] defines several messages, such as request, success, and error messages for subscribing, unsubscribing and publishing, as also for configuring subscription options and retrieving items from node. It also defines messages for example for creating, configuring, deleting and managing nodes. In XMPP pubsub offers possibilities to create versatile configurations. An XMPP account / client can act as both publisher and subscriber. Pubsub nodes may have none, one or several publishers and subscribers. Some of the subscribers may act as aggregators, as seen also in Fig. 6, that collect information from several pubsub nodes, analyse, combine and sophisticate the information and publish it to other pubsub nodes.

Because the number of sensors in M2M and IoT systems can become immense, aggregators are needed to make the information more understandable.

In M2M and IoT systems, pubsub can be used for several basic operations. One of the most common one is to publish sensor information to a pubsub node, and provide it from there to all subscribers. Sensors and actuators can work as both publishers and subscribers, and it is possible to read, e.g., configuration files from pubsub nodes to sensor devices. Using pubsub and adding none contacts for devices, the amount of the traffic going between the sensor device and its server can be decreased with the approach presented in Fig. 7.

1. User starts the device.
2. Device logins, e.g., by using the manufacturer's preconfigured XMPP credentials.
3. Device subscribes to a pubsub node preconfigured by the manufacturer.
4. Owner or user of the device logins to the manufacturer's webpage and gives wanted credentials of the device.
5. The manufacturer publishes these to the preconfigured pubsub node.
6. An authorized device gets new credentials.

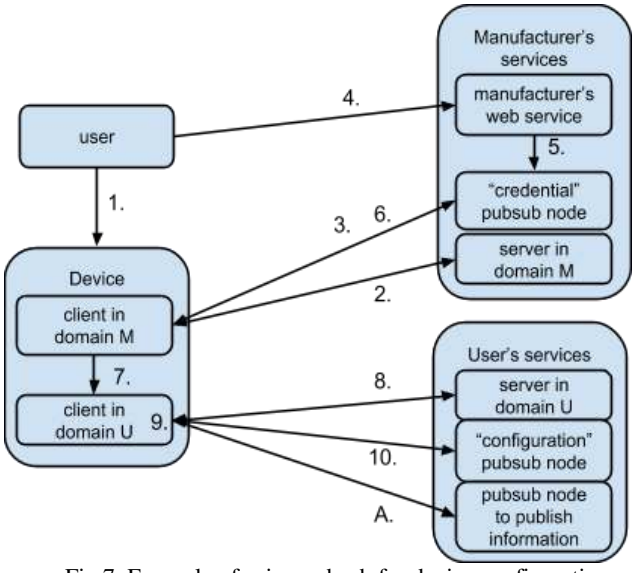


Fig 7. Example of using pubsub for device configuration.

7. The credentials in the device are changed. The XMPP client using the manufacturer's preconfigured account logs out. New credentials are used. If there is space left in the memory, it is possible to store the manufacturer's preconfigured credentials to be used later as backup mechanism for logins.
8. The device logins using the new credentials and discovers certain pubsub node used to store configuration information, and subscribes to it. Configuration information might include a new password, describe task to be executed, or provide names of pubsub nodes to be created and to publish information into. This is presented as arrow A. in Fig. 7. Notice that these pubsub nodes are not necessarily in domain U.
9. The device starts running based on the received configuration.
10. If the item in configuration pubsub node is updated, the device gets new configurations. The device may get this newest item also every login time.

Problem in such an approach is that the user must rely on the manufacturer and that the manufacturer does not use provided credentials maliciously before they have been changed, that is done after step 8. Notice that in the example only core XMPP and pubsub [60] are used to transmit all the information between XMPP clients and server. It would be possible, to use mechanisms described in [60] to configure credentials for device in step 2. It would be possible to publish configuration commands (such as ad-hoc [65], Jabber-RPC [66], <set> command of XEP-0325 [59]) to a pubsub node. This replaces an approach of embedding commands to <message> or <iq> stanzas sent directly to the device. When the device (sensor or actuator) subscribes to this pubsub node, it receives the command and performs or starts performing the wanted action or actions. This is especially useful if the full JID of the device is not known, and provides transmitting configurations simultaneously to thousands of devices using only one pubsub node.

Details of security of the pubsub architecture are further described in the section "XMPP security" of this paper.

### C. *Applicability of XMPP for M2M purposes*

XMPP core technology already offers a variety of key functionalities which are used for enabling reliable near real-time and secure communication between connected clients and servers. Some of these features and characteristics of XMPP are directly extremely useful for M2M and pervasive applications, such as:

- presence information is utilized for providing information about connecting and disconnecting entities along with information about changes in the services provided by the entities
- secure messaging (TLS): TLS is used to encrypt message channels from M2M clients to back-end servers.
- overlay communication over IP: The Internet Protocol (IP) is widely used protocol, not only in Internet, but also in closed networks. There are implementations that support both IPv4 and IPv6 protocols.
- near real-time: XMPP is based on XML streams that are direct connections between clients and servers. This allows messages to pass the system in minimum latency.
- authentication is essential to make the M2M overlay secure and available to only trusted entities
- contact list of XMPP account is stored in server so clients need minimal amount of persistent memory.
- service discovery is critical in order to connect entities providing services to entities requiring services. Service discovery [67] defines the basic service discovery mechanism of XMPP. It defines three kinds of information which need to be discovered about an entity:
    - its basic identity (type and/or category)
    - the features it offers and protocols it supports
    - any additional items associated with the entity, whether or not they are addressable as JIDs

The basic service discovery mechanism has also been extended with functionalities (XEP-0128) that enable extensions such as description for multi-user chat rooms, room subjects, number of occupants in the room and the JID of the room, using Data Forms (XEP-0004) and Field Standardization for Data Forms (XEP-0068) specs as building blocks. As explained in the [68], an XMPP client or other entity may need to discover services external to the XMPP network in order to complete certain XMPP-related use cases. An XMPP entity can discover external services in several ways, including:

1. The service is specified in the application's default settings.
2. The service is manually added into the application's configuration by a human user.
3. The service is discovered via non-XMPP service discovery protocols, such as:
    - DNS SRV records, RFC 2782
    - Service Location Protocol (SLP), RFC 2608
    - The Dynamic Delegation Discovery System (DDDS), RFC 3401
    - The NAPTR profile of DDDS, RFC 3403
    - The S-NAPTR profile of DDDS, RFC 3958
    - The U-NAPTR profile of DDDS, RFC 4848
4. Using protocol defined in [68] as a fall-back when the relevant service discovery technologies are not available to the XMPP entities involved.

As described in section “IEEE’s M2M Standardization”, IEEE has defined how to transmit IEEE 1451 sensor and actuator information with XMPP.

Another example of XMPP’s potential for M2M applications is Blueforce’s M2M Cloud Service [69], which takes input from multiple, and disparate, unattended ground sensors (UGS) and normalizes the output into a standards-based information flow that can be viewed inside of Blueforce Tactical, Blueforce Command Center Common Operational Picture (COP), or any XMPP standards-based client. UGS data may be overlayed alongside of other human-based assets as well as shared in real-time on Geospatial Information Systems using our dynamic KML output. This cloud agent has been built from the ground up and can support thousands of subscribed endpoints as well as hundreds of inbound sources and can be hosted on own infrastructure or different cloud service providers. Blueforce mentions “Inter-Agency Sensor Sharing”, “Technical Collection”, “Surveillance and Reconnaissance”, “Critical Infrastructure Protection”, “Force Protection”, and “WMD Monitoring” as use cases.

XMPP provides a general framework for messaging across a network. This has a multitude of applications beyond traditional IM and the distribution of data. A close application to IM is multi-party messaging or the development of multi-user chat rooms [70]. With group communication, features similar to micro-blogging as provided by Twitter can be implemented. But text is not only data that can be transmitted through XMPP. Other forms of communication could include audio, image, and video data. XMPP provides a solid base for both discovery of services on a network and advertisement of services and capabilities. XMPP offers a crucial set of features for online games, including authentication, presence information, chat and extensible near-real-time communication of game state information. XMPP has been used in research laboratories but also in commercial products for M2M communication and remote management of devices in different domains such as smart grids or infrastructure monitoring. Also, XMPP is a protocol of new area of cloud computing. Cloud computing and storage rely on various levels and forms of communication, including not only messaging between systems to relay state but also migration of larger objects, such as storage or virtual machines. XMPP can be applied at a variety of levels and is ideal as a middleware protocol.

One of the main reasons for choosing XMPP as a basis for our M2M-related experiments was that it already offered us directly most the basic functionality that were needed and that it was designed from the ground up to be easily customizable and extensible. The extensibility granted us the opportunity to implement needed modifications to the system and for our own data formats to transfer appropriate data between entities without “re-inventing the wheel”. There exist only two types of basic connections in XMPP network: client to server and server to server connections, although serverless messaging can also be accomplished by using appropriate XMPP extension [63]. Each connection is a XML stream through a TCP socket as specified in RFC 6120 [39]. Client to server connections are persistent but server to server connections can be established as need basis.

### *D. XMPP Security*

#### D.1 Encryption

XMPP has supported per-hop channel encryption using Transport Layer Security (TLS) through a STARTTLS upgrade mechanism. Per-hop encryption using TLS can be used to protect only messages transmitted between the client and the server, or messages transmitted between servers, but not inside servers that do the routing. TLS has been used mainly in C2S communication, and usually S2S communications are not encrypted. Mechanisms such as OpenPGP, Off-the-Record (OTR), TLS, S/MIME, SIGMA, XML encryption, and CMS with JOSE, to enable end-to-end encryption between two clients exist, but none of them has been deployed widely [71].

#### D.2 Pubsub security

XMPP pubsub and access models are described in XEP-0060 [64] and here in Table 1.

TABLE I.
NODE ACCESS MODELS OF XMPP PUBSUB

| Access Model | Description |
|---|---|
| Open | Any entity may subscribe to the node (i.e., without |

| | |
|---|---|
| | the necessity for subscription approval) and any entity may retrieve items from the node (i.e., without being subscribed). Pubsub XEP [XEP-0060] describes that this should be the default access model for generic pubsub services. It should be noted that such generic pubsub services must not publish any confidential data if the service itself is public to the Internet. Confidential data may only be published if the pubsub server is trusted and isolated. |
| Presence | Any entity with a subscription of type "from" or "both" may subscribe to the node and retrieve items from the node; this access model applies mainly to instant messaging systems. |
| Roster | Any entity in the specified roster group(s) may subscribe to the node and retrieve items from the node; this access model applies mainly to instant messaging systems. |
| Authorize | The node owner must approve all subscription requests, and only subscribers may retrieve items from the node. |
| Whitelist | An entity may subscribe or retrieve items only if on a whitelist managed by the node owner. The node owner MUST automatically be on the whitelist. In order to add entities to the whitelist, the node owner SHOULD use the protocol specified in the Manage Affiliated Entities section of this document, specifically by setting the affiliation to "member". |

In XMPP pubsub [64], a publisher can select who (XMPP entities) can subscribe to the published information. For example, the publisher can give access only to entities in its roster, only to certain entities in a whitelist, to all, or require an authorization from certain entities. When the information is private, it should be exercised care in approving subscription requests. The server must allow only authorized entities to complete “owner use case” actions such as create, configure, and delete node, request default node configuration options, purge all node items, manage subscription requests, process pending subscription requests, and manage subscriptions and affiliations. Manual authorization is done also when XMPP entities want to subscribe to presence information.

Access control of pubsub can be done remotely and/or externally, e.g., one possibility is to define a constrained device to create only private pubsub nodes to pubsub service. The private access control rule (access model) would be changed to a wanted one afterwards from unconstrained devices (e.g., using laptops). Changing of the access control rules would have to be done either using the same JID as the publisher, or using an administrator account. In XMPP, it is possible to login using the same XMPP account from different locations, only the resource part of the JID changes. XMPP’s pubsub enables implementation of more complex publishing and subscription modes. For example, it enables subscribing to nodes temporarily, i.e., only for as long as the subscriber is online in its current presence session. In addition, it is possible to deliver event notifications only when the subscriber is online. XMPP’s pubsub offers also time-based subscriptions aka leases that enable expiring old or stale subscriptions. It also offers error message to tell the subscriber, e.g., that the subscriber is not in the customer database or the customer’s account is not paid up. This could be used in commercial deployments [64].

### D.3 Sensitivity level

A security label aka a confidentiality label, is a structured representation of the sensitivity of a piece of information and it is used in conjunction with a clearance, a structured representation of what information sensitivities a person (or other entity) is authorized to access, and a security policy to control access to each piece of information. If such information would be transmitted using XMPP’s pubsub, a mechanism presented in XEP-0314 could be used, where XEP-0314 defines an extension that allows for the use of security labels in XMPP’s pubsub.

### D.4 Presence leak

There is a risk that presence information is leaked when publishing items, but this risk can be mitigated, e.g., by removing publisher related information from the pubsub nodes. XMPP might leak presence information in other use cases. In addition to access control (access model) mechanism of pubsub, XMPP offers mechanisms to handle persistent storage of published information that could be used in M2M communication, e.g., to store information that is private to M2M overlay entities and/or their owners. This is described in XEP-0223: Persistent Storage of Private Data via PubSub.

## *E. Relevant informational, final and draft standards XMPP extensions for M2M*

This section lists relevant XEP standards that are informational or have final or draft status. Designers and engineers planning to apply XMPP in M2M and IoT applications would do well to be aware of the XEPs listed the following:

- XEP-0009: Jabber-RPC [66] defines how to transport XML-RPC encoded requests and responses between two XMPP entities.
- XEP-0030: Service Discovery [67] defines how to discover information about other XMPP entities. Two kinds of information can be discovered, the identity and capabilities of an entity, including the protocols and features it supports, and the items associated with an entity. Service discovery has been used in most of the current IoT and M2M systems which are based on XMPP.
- XEP-0045: Multi-User Chat [70] defines an XMPP protocol extension for multi-user text chat, whereby multiple XMPP users can exchange messages in the context of a room or channel. It has been used for IoT and M2M systems, as described in [10], [48],[47]
- XEP-0050: Ad-Hoc Commands [65] defines how to advertise and execute application-specific commands. Ad-hoc commands have been used, e.g., in VIRTUS M2M middleware to provide workflow capabilities for any structured interaction between two XMPP entities [6].
- XEP-0060: Publish-Subscribe [64] provides generic publish-subscribe functionalities such as creating and managing nodes and publishing information at nodes, and subscribing to nodes. Pubsub is used in several XMPP based IoT and M2M systems, such as in [52], [10], [42] and [12] for example.
- XEP-0080: User Location [72] defines how to communicate information about the current geographical or physical local of an entity.
- XEP-0100: Gateway Interaction [73] defines best practises for interactions between Jabber clients and proxy gateways to legacy IM services. Even if the service behind the proxy gateway would not be an IM service, some described practises, such as ones in the Security Considerations -section apply to M2M systems that are using gateways to connect to several protocols or networks.
- XEP-0115: Entity Capabilities [74] provides a mechanism for caching, and hence eliding, the disco#info requests needed to

negotiate optional features.

- XEP-0127: Common Alerting Protocol (CAP) Over XMPP [75] defines a method for sending CAP data over XMP. It is of course suitable only to M2M systems that are using CAP data for alerts and notifications.
- XEP-0138: Stream Compression [76] provides application stream level compression, useful if the device TLS stack does not support TLS-based compression.
- XEP-0156: Discovering Alternative XMPP Connection Methods [77] defines how to discover alternative methods of connecting to an XMPP server. It could be suitable, e.g., in sensor networks which do not have or cannot use TCP connectivity.
- XEP-0174: Serverless Messaging [63] defines how to communicate over local or wide-area networks using the principles of zero-configuration networking. This XEP could be useful especially in local area sensor networks, which have no Internet connection always available. It has been used in IoT systems to provide possibility for sensors to communicate in disaster situations, as described in [47].
- XEP-0198: Stanza Acknowledgements [78] provides session resumption over TCP, enabling a client to handle the case where the coverage is patchy. The <r/> and <a/> elements also provide a keep alive facility in a small number of octets. Currently the name of this XEP is Stream Management.
- XEP-0222: Persistent Storage of Public Data via PubSub [79] defines practices for using the XMPP's pubsub to persistently store semi-public data objects such as public keys and personal profiles. In M2M systems this public information may be, e.g, public key of a sensor or an actuator. Also, Persistent Storage of Private Data via PubSub in XEP-0223 defines best practices for using the XMPP's pubsub to persistently store private information. In M2M systems this private information could be for example client or sensor device configuration options.
- XEP-0229: Stream Compression with LZW [80] defines how to use the LZW algorithm in XML stream compression. Other proposals for XML compression exist.

### *F. Relevant currently experimental XMPP extensions and not accepted XEP (and older Jabber Extension Protocol) proposals for M2M*

This section lists relevant XEP and older Jabber Extension Protocol (JEP) standards that have experimental status or have not yet been accepted. Notice that listed XEPs may have major changes before they are final.

- Smart Presence Distribution [81] compares the distribution strategies to cut down on server-to-server traffic. It is old, from 2005, and haven't got acceptance. We see that this would be especially useful if sensor networks in M2M systems would contain XMPP servers.
- Transmitting authentication factor information using Ad-Hoc Commands [82] defines how, e.g., the user may authenticate itself and gain access to a device.
- Sensor-Over-XMPP [61] proposal which defines a payload format for exchanging sensor and actuator data. XEP proposal mentions use cases such as power distribution metering, home automation, monitoring and control of heating and cooling systems and infrastructure monitoring. Sensor-Over-XMPP has been used, e.g., in large-scale campus-wide sensing and actuation [42].
- Stanza Repeaters [83] enables optimization of interdomain traffic that is intended for delivery to multiple recipients, using the concept of a stanza repeater. This could be useful, e.g., when configuring several M2M entities. On the other hand we see that similar functionalities could be implemented with pubsub [XEP-0060].
- XEP-0215: External Service Discovery [68] defines how to discover services external to the XMPP network.
- XEP-0322: Efficient XML Interchange (EXI) Format [84] describes how EXI compression can be used in XMPP networks.
- XEP-0323: Internet of Things - Sensor Data [57] provides a common framework for sensor data interchange over XMPP networks.
- XEP-0324: Internet of Things - Provisioning [58] describes an architecture for provisioning of services, access rights and user privileges in for the XMPP based IoT.
- XEP-0325: Internet of Things - Control [59] describes how to control devices or actuators in an XMPP-based sensor network.
- XEP-0326: Internet of Things - Concentrators [85] describes how to manage and get information from concentrators of devices over XMPP network.
- XEP-0337: Event Logging over XMPP [86] provides common framework for sending event logs over XMPP network.
- XEP-0347: Internet of Things - Discovery [60] describes mechanisms for installing and discovering Things in IoT.

### *G. Relevant currently deferred and obsoleted XMPP extensions for M2M*

Implementing any deferred and obsoleted XEPs is not recommended by XSF as such, but the ones listed below include valuable and interesting information from the point of view of M2M systems. For example, XEP-0286 [87] gives guidelines to optimize usage of power and bandwidth in XMPP and also lists few other interesting and related XEPs.

- XEP-0237: Roster Versioning [88] provided an extension for reducing the roster fetch bandwidth, in most cases reducing it to a simple affirmation that the client has the current roster. Today the XEP-0237 is incorporated into core XMPP in RFC 6121 [40]. Approach saves not only bandwidth, but also reduces local storage writes. If there are only few contacts in M2M devices' rosters, bandwidth usage caused by roster updates will not usually be a problem except in extremely resource constrained environments.
- XEP-0250: C2C Authentication Using TLS [89] defines how to negotiate TLS extensions when using TLS for end-to-end XML streams between two clients. It covers X.509 certificates with and without CA, the use of OpenPGP, Shared Remote Passwords (SRP) and how to use one extension to bootstrap a trust relationship. XEP-0250 has been used, e.g. in [47].
- XEP-0255: Location Query [90] defines how to query a compliant server or service for information about the geographical or physical location of an entity.
- XEP-0273: Stanza Interception and Filtering Technology [91] provides a mechanism which, amongst other things, would allow a presence "hush", buffering presence during certain states.
- XEP-0286: XMPP on Mobile Devices [87] provides information for engineers using XMPP concerned with mobile devices operating in a cellular network. Similar principles can be used in other wireless networks to reduce resource consumption.

## IV. THE PROOF-OF-CONCEPT IMPLEMENTATIONS

### *A. XMPP as a communication overlay for M2M*

The applicability of XMPP technologies for building M2M systems has been evaluated within this paper by implementing a proof-of-concept system around an electric bicycle and its accessory sensors. The M2M communication overlay of this experimental

system has been implemented using XMPP. In, Fig. 8, an illustration on the overall experimental system composition can be seen.

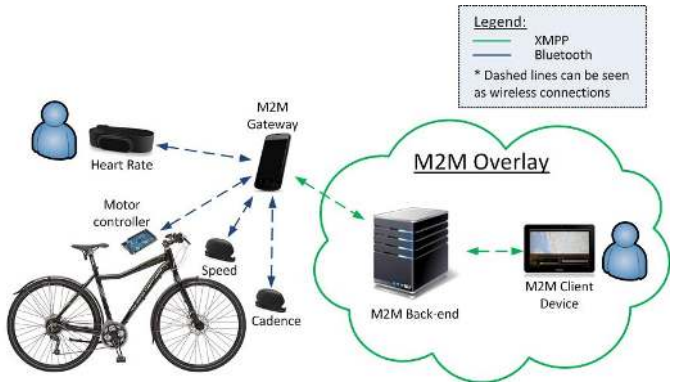


Fig. 8. Experimental M2M communication overlay based on XMPP.

The experimental environment consists mainly of two parts: The electric bicycle and the M2M overlay. These parts are connected together through the M2M gateway. Additionally, proprietary systems could also be connected to the M2M overlay with appropriate M2M gateways to further expand the overall M2M environment.

The two main parts of the environment can be broken down into smaller components. The components are as follows:

- Electric bicycle
- Bluetooth interface for bicycle electronics (motor controller)
- Bluetooth Smart (BLE) sensors (heart rate, cycling speed and cycling cadence)
- M2M Gateway software on smart phone
- Electric bicycle user interface on smartphone
- M2M back-end
- Database on back-end
- M2M Client devices with or without user interface

In this experimental system the electric bicycle can be seen as an example of a mobile M2M device. It contains an electric motor, several accessory sensors and a data connection to the back-end server. This setup allows experiments with the sensor data usage and the management of electrical devices over the M2M overlay network. Both, static and mobile clients communicate through the back-end server. This creates an uniform network over different wireless and wired networks in various different address spaces. This uniform network is called the M2M overlay. In the experimental environment the M2M overlay is simplified as a messaging platform, using XMPP as the basis, that provide an uniform address space, messaging and presence information for the M2M clients. At this point the environment consists only one back-end server, hence, leaving inter-domain communication out of the scope of this work.

The sensors on the bicycle are connected to the smartphone via Bluetooth Smart low energy connection for delivering the sensor data. There is also a classic Bluetooth connection between the smartphone and the motor controlling electronics to enable the controlling of the electric motor and reading its values using the smart phone. At the same time the smart phone acts as a gateway device towards the M2M overlay network, making the gathered data from the electric bicycle available to other M2M entities on the M2M overlay. Dispatching of this data is done according to the XMPP's publish-subscribe extension [64]. The XMPP-based gateway solution has been described more thoroughly in the next chapter.

### *B. XMPP-based M2M gateway for the Android platform*

M2M gateways are devices or software that enables connecting proprietary systems to the M2M overlay network. These systems can consist of large number of devices or for example just one tiny sensor. The main point here is that the gateways allow these separate systems to become part of the M2M network. As the experimental M2M environment described herein, the M2M gateway is used to connect multiple Bluetooth Smart and Bluetooth classic sensors to the M2M overlay network. The data that the sensors provide are published to the M2M overlay network via the smartphone acting as the M2M gateway. The gateway is capable of two-way communication which enables sending of commands to the actuators from the M2M overlay network, in other words, remote controlling.

The M2M gateway device in the experimental environment is an Android-based smartphone. Android was the preferred choice as the operating system on the smartphone, based on its large market share and Android's openness towards application development. Using of Bluetooth Smart sensors on a bicycle also set some criteria for choosing the applicable smartphone. The official support for Bluetooth 4.0 was published on Android 4.3 so it was clear that the phone needed to run at least that version of the Android operating system. The first Android-based smartphone to receive update to the Android 4.3 was the Google Nexus 4. Naturally, it is also the smartphone used in the experimental system, as it was the only available Android smartphone officially supporting Bluetooth 4.0 at the time of building the system.

The experimental setup has three Bluetooth Smart sensors connected to the M2M gateway, as seen in Fig. 8. The cycling speed and cadence sensors are mounted on the bicycle and the cyclist has the heart rate belt around his/her chest. All of these sensors use the Bluetooth Smart low energy communication to deliver the sensor information to the gateway device. Communicating with the cycling speed and cadence sensors is done by the Bluetooth 4.0 specification and its cycling speed and cadence profile specification [92]. Same goes with the heart rate sensor using the heart rate profile [93]. The Bluetooth API of Android system is utilized to implement the actual connection and communication between the sensors and the M2M gateway. The minimum API level for using the Android's Bluetooth Low Energy API is the API level 18, corresponding to Android version 4.3.

The control electronics of the electric motor uses the Bluetooth classic to connect to the M2M gateway. The battery voltage level, motor's assistance level and motor status, on or off, are gathered from the control electronics and then published to the M2M network. Also the remote controlling of the motor is enabled to demonstrate the ability of two way communication. The electric motor is the actuator on this setup. Control commands to it can be sent from the M2M client entity over the M2M network, passing through the M2M gateway and to the control electronics which translates the commands to the actual analog voltage signals for the electric motor.

The most interesting part of the implementation is the gathering of the sensor data and translating it to the appropriate format to be published on the M2M network. As the M2M overlay network uses XMPP as its basis, the Sensor-Over-XMPP extension [61] is used to format the gathered data appropriately. Basically the process is as simple as connecting the sensors to the phone and obtaining the sensed data from the sensors using the Android Bluetooth API, after that the data gets wrapped up inside the payload format according to Sensor-Over-XMPP specification and then published to the M2M overlay network using publish-subscribe capabilities of XMPP and its extensions. Other M2M entities on the network can then access the published data by subscribing it. The publish-subscribe functionality

is described thoroughly on the official XMPP publish-subscribe extension [64].

The XMPP part of the M2M gateway has been implemented using aSmack version 0.8.5, XMPP client library for Android API level 18. The basic XMPP functions of the aSmack library are used and above them the custom Sensor-Over-XMPP and publish-subscribe features are implemented according to their extension specifications.

### C. *Advanced two-factor user authentication for M2M*

[82] describes an authentication mechanism which defines how to transmit authentication factor information with ad-hoc commands defined in [65]. The [82] gives examples to enable two-factor authentication by using mechanisms to check if used XMPP account exists and/or if user knows or have credentials of the account, if used XMPP client knows Verifier's full JID, and if user of the client, the client, or some other entity is able to access a shared secret stored inside certain device, a trusted platform module (TPM) or application and provide it to the Verifier through the Prover during a wanted time-window. The current version (0.0.4) of the XEP proposal [82] describes transmitting a random string, a Time-Based One-time Password (TOTP) or a One-time pad and using them as authentication factors.

One purpose of designing and implementing such a mechanism for authentication is to enable renting the electric bike and/or M2M Gateway without need to tell XMPP credentials used in the M2M Gateway to the customer. In the simplest use case, the Verifier client only checks that the Prover client is running in the same device, such as in a smartphone, as it is. After the two-factor authentication, it is possible to execute access control to check if the user is authorized, e.g., to access or use certain resource. In the use case, this resource is the XMPP client in M2M Gateway which, e.g., controls the motor and publishes events from sensors. The XEP proposal does not take a stance on what the resource is, so it could be, e.g., ability to create XMPP accounts using in-band registration defined in [94].

Prototype implementation was created with SleekXMPP [95], PYOTP - The Python One Time Password [96], and Python's hashlib libraries.

### D. *An interworking mechanism for ETSI/XMPP interoperability*

Working with XMPP alongside the ETSI SCL platform, both of which have been discussed earlier in this paper, gives an interesting and valuable opportunity to perform experiments with and study one of the most important aspect of modern M2M systems, cross-domain interoperability and interworking capabilities. After all, in M2M world different systems should be able to communicate and cooperate with each other in a meaningful manner.

In this work, a proof-of-concept interworking module has been developed with the goal of further enhancing the interoperability of modern M2M systems and the various technologies used in M2M by bringing together the XMPP domain with the ETSI SCL –based system. This is approached by connecting an experimental XMPP-based M2M domain to an up-and-running ETSI NSCL. In concrete terms, this is done by building an interworking client that is able to talk to both the XMPP –based M2M back-end server in the experimental M2M ecosystem and an ETSI NSCL server. The ETSI NSCL implementation employed in this work is a part of the OM2M -project, which has been presented in this paper in the previous chapter. The most important part of the whole Interworking Proxy module is the Custom Interworking Logic unit, which handles all the crucial translation work of the messages and message formats from ETSI network to XMPP network and vice versa. To summarize, the ETSI-XMPP Interworking Proxy packs up two different client units in it with the Custom Interworking Logic. Clients handle the communication towards the appropriate networks and the logic unit translates the messages so that the receiving end can understand it.

The experimental implementation of the Interworking Proxy for XMPP / ETSI NSCL presented in this paper consists of the following modules, as shown in Fig. 9..:

- Interworking Logic module
- XMPP-Communication module
- HTTP-Communication module
- Interface/EventListener module

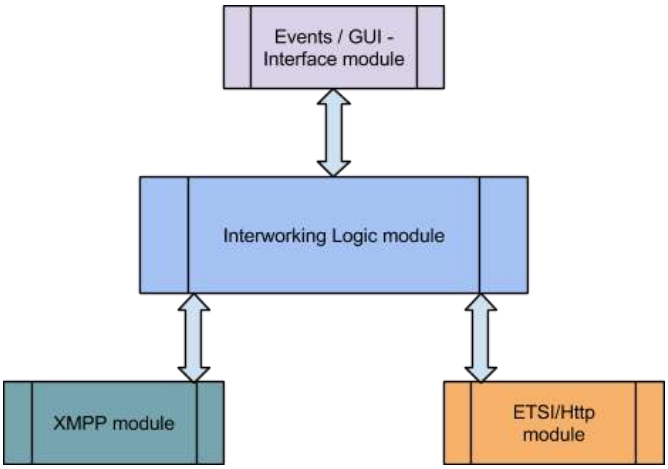


Fig. 9. The main functional modules and their relationships within the Interworking Proxy implementation.

The first prototype version of the IWP was designed and built to store information received from a XMPP domain via publish-subscribe mechanisms into the ETSI NSCL resource tree.

The XMPP-Communication module is essentially a set of XMPP-related functionalities built on top of an XMPP client library. The module contains all the necessary XMPP-related functionalities for actions such as opening a connection to a designated XMPP-server, providing authorization information (username, password), logging in to the server, subscribing to a designated pubsub node, listening for incoming messages, processing and storing incoming messages and sending XMPP-messages. It communicates with the XMPP Server in a standard client-server manner as specified by the XMPP standard. Additionally this client also has an interface towards the Custom Interworking Logic unit to handle the required translation of messages and resources.

For the Interworking Proxy implementation presented in this paper, the XMPP functionalities are built upon the Smack API 3.3.1 [97], which is a widely used open source XMPP client library, licensed under the Apache Software License. It is a library for the Java platform, and it can be embedded into any application to create anything from full blown XMPP clients to simple XMPP interactions, such as sending simple notifications. It is a convenient and functional library for enabling communications with XMPP servers to perform instant messaging and group chat, for example. The Smack library is currently licensed under the Open Source Apache License, which means that anyone can easily include the functionalities offered by it into any non-commercial or even commercial applications.

The HTTP-Communication module handles all the communication to the ETSI NSCL server via HTTP-protocol. Any interchange of

information with the ETSI NSCL is carried out through simple GET, POST, PUT and DELETE Http-calls, following the Representational state transfer (REST) model. The M2M service architecture proposed by ETSI has been discussed in more detail earlier in this paper. The ETSI module conforms to the ETSI specification and it communicates in an ETSI specified way towards the ETSI NSCL. This module also has the interfaces towards the Custom Interworking Logic unit to be able to exchange data with it.

The Interworking Logic module is, as the name suggests, where most of the logic and "intelligence" of the implementation resides. In technical terms, all the other modules are allocated and employed as member objects within the Interworking Logic module and its corresponding object. It gets input and event from the Interface/EventListener module and then carries out all the necessary processing and calls the necessary methods from the XMPP- or HTTP-Communication modules. The actual application "payloads" embedded into the ETSI specified XML-messages transmitted over HTTP/REST -calls, are encoded into the convenient Base64 –format [98]. Base64 is a binary-to-text encoding scheme with the goal of representing any binary data in an ASCII string format by means of translating the data into a radix-64 representation. Base64 encoding schemes are commonly applied when there is a need to encode various kinds of binary data that needs to be stored and transferred with methods that have been designed to work with textual/string -type data. The aim is to ensure that the data remains intact without modification during transport. Base64 formats are quite commonly used in a number of applications such as email and XML-based applications. In the experimental Interworking Proxy implementation, it was decided to employ the tried and true Base64 encoder from the Apache Commons Codec –library [99], licensed under the Apache License. The Apache Commons Codec software library provides a time tested and reliable implementations of common encoders and decoders such as Base64, Hex, Phonetic, etc. It consists of a set of utilities and a simple framework for encoding and decoding text and binary data. In general terms, the Commons is an Apache software project which concentrates on all sorts of reusable Java components.

The trials and experiments to employ the interworking implementation presented here in making resources and information from the XMPP-domain available in the ETSI NSCL have been successful. The IWP, in its current form, is indeed able to read, process and store data received via XMPP publish subscribe and package and make that information readily available using the resource tree in the NSCL, as specified by ETSI.

## V. CONCLUDING REMARKS

When considering the requirements for machine-to-machine communications and pervasive applications, interoperability with other systems will inevitably become one of the key issues. There are already dozens of existing solutions in the field already for M2M, but they are mostly designed to work only in a specific application environment with devices that conform to certain protocols and technologies. Scalability and end-to-end security can also be added among the list of key challenges. The various different application domains relevant to M2M communication must be taken into account by designing the main components and essential features to be general, widely applicable and flexible as much as possible. Most of the existing M2M systems today are tightly focused on narrow usage scenarios that are only relevant within certain highly specific application domains, though some efforts are being made towards global M2M standardization and cross-domain interoperability in the form of horizontal service platforms.

In this paper we have explored the applicability of the Extensible Messaging and Presence Protocol (XMPP) and related technologies for M2M systems and pervasive applications. Based on our study and the experimental implementations discussed in this paper, we conclude that many of the requirements and challenges (also discussed in this paper) in such systems can be met by harnessing XMPP-based technology. XMPP as a technology contains already in its core some of the functionalities that can be seen as the key enablers for M2M message exchange. Furthermore, due to its extensible nature, many extensions exists which provide mechanisms and functionalities required in interoperable M2M systems.

Some items that had to be left for future work include larger scale M2M experiments with XMPP, possibly including multiple federated servers, interconnected gateways, scalability tests and employing hardware-in-the-loop simulation techniques. Also, bringing in more extensive peer-to-peer functionalities, integration of more diverse sensor/embedded systems and experimenting with mechanisms to enable delay tolerant communication would be of considerable interest.

## VI. ACKNOWLEDGEMENTS

This work has been partially conducted within the European joint research project A2Nets under ITEA2 cluster project of EUREKA network, to which organisations the authors of this paper wish to express their gratitude.

## REFERENCES

[1] Wu, G.; Talwar, S.; Johnsson, K.; Himayat, N. & Johnson, N. D. (2011), 'M2M: From mobile to embedded internet.', *IEEE Communications Magazine* **49** (4) , 36-43 .

[2] D. Katusic, A. Marcev, R. Vulas, and G. Jezic, “Machine-to-machine: Emerging market and consequences on existing regulatory framework,” in Telecommunications (ConTEL), 2013 12th International Conference on , 2013, pp. 317–324.

[3] Lawton, G. "Machine-to-Machine Technology Gears Up for Growth," Computer, vol. 37, no. 9, pp. 12-15, September, 2004

[4] Chang, Kim, Anthony Soong, Mitch Tseng, and Zhixian Xiang. 2011. "Global Wireless Machine-to-Machine Standardization." *IEEE Internet Computing* 15, no. 2: 64-69. *Academic Search Research & Development*

[5] Internet Protocol for Smart Objects (IPSO) Alliance’s webpage, URL: http://www.ipso-alliance.org/

[6] Conzon, D.; Bolognesi, T.; Brizzi, P.; Lotito, A.; Tomasi, R.; Spirito, M.A., "The VIRTUS Middleware: An XMPP Based Architecture for Secure IoT Communications," Computer Communications and Networks (ICCCN), 2012 21st International Conference on , vol., no., pp.1,6, July 30 2012-Aug. 2 2012

[7] Kusznir, J.; Cook, D.J., "Designing Lightweight Software Architectures for Smart Environments," *Intelligent Environments (IE), 2010 Sixth International Conference on* , vol., no., pp.220,224, 19-21 July 2010

[8] Jouni Hiltunen, Mikko Ala-Louko, and Markus Taumberger, “Experimental Performance Evaluation of POBICOS Middleware for Wireless Sensor Networks,” ISRN Communications and Networking, vol. 2012, Article ID 180369, 10 pages, 2012. doi:10.5402/2012/180369

[9] Teemu Väisänen, "A Simple M2M Overlay Entity Discovery Protocol." ICCGI 2012, The Seventh International Multi-Conference on Computing in the Global Information Technology. 2012.

[10] Bendel, S.; Springer, T.; Schuster, D.; Schill, A.; Ackermann, R.; Ameling, M., "A service infrastructure for the Internet of Things based on XMPP," Pervasive Computing and Communications Workshops (PERCOM Workshops), 2013 IEEE International Conference on , vol., no., pp.385,388, 18-22 March 2013

[11] Latvakoski, J., Hautakoski, T., Väisänen, T., Toivonen, J., Lappalainen, A., & Aarnipuro, T. (2009, June). Secure M2M service space in residential home. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE* (p. 10). ACM.

[12] Walczak, D.; Wrzos, M.; Radziuk, A.; Lewandowski, B.; Mazurek, C., "Machine-to-Machine communication and data processing approach in Future Internet applications," *Communication Systems, Networks & Digital Signal Processing (CSNDSP), 2012 8th International Symposium on* , vol., no., pp.1,5, 18-20 July 2012

[13] Latvakoski, J., Hautakoski, T., & Iivari, A. (2010). Situated Service Oriented Messaging for Opportunistic Networks. In *Bioinspired Models of Network, Information, and Computing Systems* (pp. 50-64). Springer Berlin Heidelberg.

[14] Latvakoski, J.; Alaya, M.B.; Ganem, H.; Jubeh, B.; Iivari, A.; Leguay, J.; Bosch, J.M.; Granqvist, N. Towards Horizontal Architecture for Autonomic M2M Service Networks. Future Internet 2014, 6, 261-301.

[15] Galán-Jiménez, J., & Gazo-Cervero, A. (2011). Overview and challenges of overlay networks: A survey. *Int J Comput Sci Eng Surv (IJCSES)*, *2*, 19-37.

[16] Doval, D.; O'Mahony, D.; , "Overlay networks: A scalable alternative for P2P," Internet Computing, IEEE , vol.7, no.4, pp. 79- 82, July-Aug. 2003

[17] Niemelä, E., & Latvakoski, J. (2004, October). Survey of requirements and solutions for ubiquitous software. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia* (pp. 71-78). ACM.

[18] Starsinic, M. (2010, May). System architecture challenges in the home M2M network. In *Applications and Technology Conference (LISAT), 2010 Long Island Systems* (pp. 1-7). IEEE.

[19] Liu, Q., Leng, S., Mao, Y., & Zhang, Y. (2011, December). Optimal gateway placement in the smart grid Machine-to-Machine networks. In *GLOBECOM Workshops (GC Wkshps), 2011 IEEE* (pp. 1173-1177). IEEE.

[20] ETSI Machine to machine communications, URL: http://www.etsi.org/technologies-clusters/technologies/m2m

[21] About 3GPP, URL: http://www.3gpp.org/about-3gpp/about-3gpp

[22] Telecommunications Industry Association (TIA), Machine-to-machine (M2M), URL: http://www.tiaonline.org/m2m

[23] The Institute of Electrical and Electronics Engineers, URL: http://www.ieee.org

[24] oneM2M homepage, URL: http://www.onem2m.org

[25] (Accepted) Ben Alaya M, Banouar Y, Monteil T, Drira K. "OM2M: Extensible ETSI-Compliant M2M service platform with self-configuration capability". International Workshop on Recent Advances on Machine-to-Machine Communication (RAMCOM 2014). Hasselt, Belgium. June 2-5, 2014.

[26] Eclipse Projects, OM2M, URL: http://projects.eclipse.org/projects/technology.om2m

[27] 1451.4-2004 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Mixed-Mode Communication Protocols & TEDS Formats

[28] E1451.2-1997 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Transducer to Microprocessor Communication Protocols & TEDS Formats

[29] 1451.3-2003 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Digital Communication & TEDS Formats for Distributed Multidrop Systems

[30] 1451.7-2010 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats

[31] 1451.5-2007 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Wireless Communication Protocols & Transducer Electronic Data Sheet (TEDS) Formats

[32] 1451.0-2007 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats

[33] 1451.1-1999 IEEE Standard for a Smart Transducer Interface for Sensors and Actuators – Network Capable Application Processor Information Model

[34] Overview of P21451-1-4 - Standard for a Smart Transducer Interface for Sensors, Actuators, and Devices - eXtensible Messaging and Presence Protocol (XMPP) for Networked Device Communication, URL: http://standards.ieee.org/develop/project/21451-1-4.html

[35] IEEE 802.16 Machine-to-Machine (M2M) Task Group, URL: http://ieee802.org/16/m2m/index.html

[36] IEEE Standard for Air Interface for Broadband Wireless Access Systems--Amendment 1: Enhancements to Support Machine-to-Machine Applications," *IEEE Std 802.16p-2012 (Amendment to IEEE Std 802.16-2012)* , vol., no., pp.1,82, Oct. 8 2012

[37] P. Saint-Andre, K. Smith, and R. Tronçon, "XMPP: The Definitive Guide, Building Real-Time Applications with Jabber Technologies", O'Reilly, 2009

[38] P. Saint-Andre, "XMPP: lessons learned from ten years of XML messaging," Communications Magazine, IEEE, vol. 47, no. 4, pp. 92-96, April 2009, doi: 10.1109/MCOM.2009.4907413

[39] P. Saint-Andre, "RFC 6120: Extensible Messaging and Presence Protocol (XMPP): Core (2011)." URL: http://tools.ietf.org/html/rfc6120

[40] P. Saint-Andre, "RFC 6121: Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", URL: http://tools.ietf.org/html/rfc6121

[41] P. Saint-Andre, "RFC 6122: Extensible Messaging and Presence Protocol (XMPP): Address Format", URL: http://tools.ietf.org/html/rfc6122

[42] Anthony Rowe, Mario Berges, Gaurav Bhatia, Ethan Goldman, Ragunathan Rajkumar, and Lucio Soibelman. 2009. Demo abstract: The Sensor Andrew infrastructure for large-scale campus-wide sensing and actuation. In Proceedings of the 2009 International Conference on Information Processing in Sensor Networks (IPSN '09). IEEE Computer Society, Washington, DC, USA, 415-416.

[43] Szabo, R.; Farkas, K.; Ispany, M.; Benczur, A.A.; Batfai, N.; Jeszenszky, P.; Laki, S.; Vagner, A.; Kollar, L.; Sidlo, C.; Besenczi, R.; Smajda, M.; Kover, G.; Szincsak, T.; Kadek, T.; Kosa, M.; Adamko, A.; Lendak, I.; Wiandt, B.; Tomas, T.; Nagy, A.Z.; Feher, G., "Framework for smart city applications based on participatory sensing," *Cognitive Infocommunications (CogInfoCom), 2013 IEEE 4th International Conference on* , vol., no., pp.295,300, 2-5 Dec. 2013

[44] Hornsby, A.; Belimpasakis, P.; Defee, I., "XMPP-based wireless sensor network and its integration into the extended home environment," *Consumer Electronics, 2009. ISCE '09. IEEE 13th International Symposium on* , vol., no., pp.794,797, 25-28 May 2009

[45] Mong-Fong Horng; Mao-Hsiung Hung; Yi-Ting Chen; Jeng-Shyang Pan; Wen Huang, "A new approach based on XMPP and OSGi technology to home automation on Web," *Computer Information Systems and Industrial Management Applications (CISIM), 2010 International Conference on* , vol., no., pp.487,490, 8-10 Oct. 2010

[46] Szabo, R.; Farkas, K.; Wiandt, B., "Measurements of a real-time transit feed service architecture for mobile participatory sensing," *Wireless Days (WD), 2013 IFIP* , vol., no., pp.1,4, 13-15 Nov. 2013

[47] Klauck, R.; Gaebler, J.; Kirsche, M.; Schoepke, S., "Mobile XMPP and cloud service collaboration: An alliance for flexible disaster management," *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2011 7th International Conference on* , vol., no., pp.201,210, 15-18 Oct. 2011

[48] Klauck, R.; Kirsche, M., "XMPP to the rescue: Enhancing post disaster management and joint task force work," *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on* , vol., no., pp.752,757, 19-23 March 2012

[49] Khan, A.A.; Mouftah, H.T., "Secured web services for home automation in smart grid environment," *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on* , vol., no., pp.1,4, April 29 2012-May 2 2012, doi: 10.1109/CCECE.2012.6335018, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6335018&isnumber=6334811

[50] Pawara, S.R.; Hiray, S.R., "Instant Notification System in Heterogeneous Sensor Network with Deployment of XMPP Protocol," *Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference on* , vol., no., pp.87,92, 15-16 Nov. 2013

[51] Abousharkh, M.; Mouftah, H., "XMPP-enabled SOA-driven middleware for remote patient monitoring system," *Information Technology and e-Services (ICITeS), 2012 International Conference on* , vol., no., pp.1,5, 24-26 March 2012

[52] Hornsby, A.; Bail, E., "µXMPP: Lightweight implementation for low power operating system Contiki," *Ultra Modern Telecommunications & Workshops, 2009. ICUMT '09. International Conference on* , vol., no., pp.1,5, 12-14 Oct. 2009

[53] Klauck, R.; Kirsche, M., "Chatty things - Making the Internet of Things readily usable for the masses with XMPP," *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2012 8th International Conference on* , vol., no., pp.60,69, 14-17 Oct. 2012

[54] Kirsche, M.; Klauck, R., "Unify to bridge gaps: Bringing XMPP into the Internet of Things," *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on* , vol., no., pp.455,458, 19-23 March 2012

[55] Tolman, Anne, and Tommi Parkkila. "FM tools to ensure healthy performance based buildings." *Facilities* 27.11/12 (2009): 469-479.

[56] Kilpeläinen, Pekka, Rauno Heikkilä, and Tommi Parkkila. "Automation and wireless communication technologies in road rehabilitation." *Proceedings of the 24th International Symposium on Automation and Robotics in Construction (ISARC)*. 2007.

[57] Peter Waher, "XEP-0323: Internet of Things - Sensor Data", Experimental Standard, version 0.3, 04/2014

[58] Peter Waher, "XEP-0324: Internet of Things - Provisioning", Experimental Standard, version 0.3, 05/2014

[59] Peter Waher, "XEP-0325: Internet of Things - Control", Experimental Standard, version 0.3, 04/2014

[60] Peter Waher, Ronny Klauck, "XEP-0347: Internet of Things - Discovery", Experimental Standard, version 0.1, 04/2014

[61] Gaurav Bhatia, Anthony Rowe, Mario Berges, Charles Spirakis, "XEP-xxxx: Sensor-Over-XMPP", XEP proposal, version 0.0.20, 08/2012, URL: http://sensor.andrew.cmu.edu/xep/sox-xep.html

[62] Xiaoping Che; Maag, S., "A Passive Testing Approach for Protocols in Internet of Things," Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing , vol., no., pp.678,684, 20-23 Aug. 2013

[63] Peter Saint-Andre, "XEP-0174: Serverless Messaging", Final Standard, version 2.0, 11/ 2008

[64] Peter Millard, Peter Saint-Andre, Ralph Meijer, "XEP-0060: Publish-Subscribe", Draft Standard, version 1.13, 07/2010

[65] Matthew Miller, "XEP-0050: Ad-Hoc Commands", Draft Standard, version 1.2, 06/2005

[66] DJ Adams, "XEP-0009: Jabber-RPC", Final Standard, version 2.2, 11/2011

[67] Joe Hildebrand, Peter Millard, Ryan Eatmon, Peter Saint-Andre, "XEP-0030: Service Discovery", Final Standard, version 2.4, 06/2008

[68] Peter Saint-Andre, Sean Egan, Marcus Lundblad, "XEP-0215: External Service Discovery", Experimental Standard, version 0.6, 02/2014

[69] Blueforce M2M Cloud Service, URL: http://blueforcedev.com/products/blueforce-m2m-cloud-service/

[70] Peter Saint-Andre, "XEP-0045: Multi-User Chat", Draft Standard, version 1.25, 02/2012, URL: http://xmpp.org/extensions/xep-0045.html

[71] P. Saint-Andre, "STRINT Workshop Position Paper: Strengthening the Extensible Messaging and Presence Protocol (XMPP)", draft-saintandre-strint-workshop-xmpp-02, Internet-Draft, January 23, 2014, URL: http://tools.ietf.org/html/draft-saintandre-strint-workshop-xmpp-02

[72] Joe Hildebrand, Peter Saint-Andre, "XEP-0080: User Location", Draft Standard, version 1.7, 09/2009

[73] Peter Saint-Andre, Dave Smith, "XEP: 0100: Gateway Interaction", Informational, version 1.0, 10/2005

[74] Joe Hildebrand, Peter Saint-Andre, Remko Tronçon, Jacek Konieczny, "XEP-0115: Entity Capabilities", Draft Standard, version 1.5, 02/2008

[75] Peter Saint-Andre, Boyd Fletcher, "XEP-0127: Common Alerting Protocol (CAP) Over XMPP", Informational, version 1.0, 12/2004

[76] Joe Hildebrand, Peter Saint-Andre, "XEP-0138: Stream Compression", Final Standard, version 2.0, 05/2009

[77] Joe Hildebrand, Peter Saint-Andre, Lance Stout, "XEP-0156: Discovering Alternative XMPP Connection Methods", Draft Standard, version 1.1, 01/2014

[78] Justin Karneges, Peter Saint-Andre, Joe Hildebrand, Fabio Forno, Dave Cridland, Matthew Wild, "XEP-0198: Stream Management", Draft Standard, version 1.3, 06/2011

[79] Peter Saint-Andre, "XEP-0222: Persistent Storage of Public Data via PubSub", Informational, version 1.0, 09/2008

[80] Peter Saint-Andre, "XEP-0229: Stream Compression with LZW", Draft Standard, version 1.0, 09/2007

[81] Philipp Hancke, Carlo von Loesch, "JEP-xxxx: Smart Presence Distribution", JEP proposal, version 0.0.4, 05/2005, URL: http://www.xmpp.org/extensions/inbox/smartpresence.html

[82] Teemu Väisänen, "XEP-xxxx: Transmitting authentication factor information using Ad-Hoc Commands", XEP proposal, version 0.0.4, 03/2014

[83] Joe Hildebrand, Matt Yacobucci, Peter Saint-Andre, Craig Kaes, "XEP-xxxx: Stanza-Repeaters", XEP proposal, version 0.0.2, 03/2008, URL: http://xmpp.org/extensions/inbox/repeaters.html

[84] Peter Waher, Yusuke DOI, "XEP-0322: Efficient XML Interchange (EXI) Format", Experimental Standard, version 0.4, 03/2014

[85] Peter Waher, "XEP-0326: Internet of Things - Concentrators", Experimental Standard, version 0.2, 03/2014

[86] Peter Waher, "XEP-0337: Event Logging over XMPP", Experimental Standard, version 0.1, 01/2014

[87] Dave Cridland, "XEP-0286: XMPP on Mobile Devices", Deferred Standard, version 0.1, 09/2010

[88] Peter Saint-Andre, Dave Cridland, "XEP-0237: Roster Versioning", Obsoleted Standard, version 1.3, 02/2012

[89] Dirk Meyer, "XEP-0250: C2C Authentication Using TLS", Deferred Standard, version 0.2, 09/2008
[90] Helge Timenes, Simon Tennant, Ross Savage, "XEP-0255: Location Query", Deferred Standard, version 0.6, 04/2009
[91] Joe Hildebrand, Jack Moffitt, Peter Saint-Andre, "XEP-0273: Stanza Interception and Filtering Technology (SIFT)", Deferred Standard, version 0.4, 06/2011
[92] Bluetooth Special Interest Group, "Cycling Speed and Cadence Profile", version 1.0, URL: https://www.bluetooth.org/en-us/specification/adopted-specifications
[93] Bluetooth Special Interest Group, "Heart Rate Profile", version 1.0, URL: https://www.bluetooth.org/en-us/specification/adopted-specifications
[94] Peter Saint-Andre, "XEP-0077: In-Band Registration", Final Standard, version 2.4, 01/2012
[95] SleekXMPP: an MIT licensed XMPP library for Python, URL: http://sleekxmpp.com/
[96] PYOTP - The Python One Time Password Library, URL: https://github.com/nathforge/pyotp
[97] Ignite Realtime: Smack API, URL: http://www.igniterealtime.org/projects/smack/
[98] IETF: The Base16, Base32, and Base64 Data Encodings, URL: http://tools.ietf.org/html/rfc4648
[99] Apache Commons Codec, URL: http://commons.apache.org/proper/commons-codec/

**Antti Iivari**, a research scientist working for VTT since 2008, achieved his Master's degree in telecommunication engineering from University of Oulu - specializing in wireless communications, digital signal processing, engineering mathematics and embedded systems. He has worked in several international projects and has experience from a variety of topics including bio-inspired networking, software development, machine-to-machine communication and demanding network simulations. He has also worked as the leader of the M2M architecture work package in the ITEA2 A2Nets project.

**Teemu Väisänen** was born on 11th of December 1980 in Sonkajärvi, Finland. Väisänen joined VTT Technical Research Centre of Finland (VTT) in May 2005 as a Research Trainee, graduated with a Master of Science in Technology degree in Information Engineering, Embedded Systems, the University of Oulu in September 2006, and started his appointment as a Research Scientist at VTT, Oulu, Finland in October 2006. Topic of his thesis was "Security of a VoIP call in Hybrid Mobile Ad Hoc Network". His work as a research scientist consists different topics of information and cyber security. Väisänen worked as a voluntary Safety Expert of VTT, in Safer Internet Day (SID) project led by Finnish Communications Regulatory Authority 2006-2012 and since 2013 as one of VTT's voluntary Mediataitokummi in Media Literature School project led by the National Audiovisual Institute (of Finland). Currently Väisänen is studying for Ph.D. at the Department of Computer Science and Engineering at the University of Oulu and working for the Finnish Defence Forces as Researcher in the NATO Cooperative Cyber Defence Centre of Excellence (NATO CCD COE) in Tallinn, Estonia. His current research interests include cyber security, Internet of Things, privacy and anti-forensics.

**Mahdi Ben Alaya** is a Ph.D student at LAAS-CNRS laboratory in Toulouse, France. He has been awarded a master diploma in artificial intelligence and decision, and received an engineering diploma in computer science from the National School of Computer Science of Tunisia (NSCS). His research addresses Machine-to-Machine (M2M) interoperability, self-management of M2M based on Autonomic Computing (AC) paradigm, and routing optimization based on Information Centric Networking (ICN). He is an Expert of ETSI M2M and OneM2M standards for a Horizontal M2M architecture. He has been involved in the Europeans projects ITEA2-USENET and ITEA2-A2NETS. He is the co-founder of the open source Eclipse project OM2M (om2m.org) providing a full ETSI-Compliant M2M service platform.

**Tero Riipinen**, a former research scientist who worked for VTT during the period of 2009-2014, achieved his Master's degree in information processing science from University of Oulu while specializing in software engineering and digital media. During his time at VTT, he has worked in several large international research projects. Topics he is passionate about include software development, machine-to-machine communication, data visualization, user interfaces and game development.

**Thierry Monteil** is assistant professor in computer science since 1998 at INSA Toulouse and researcher at LAAS-CNRS. He received the Engineering degrees in Computer Science and applied mathematics from ENSEEIHT in 1992. He had a Doctorate in parallel computing in 1996 and a HDR degree in 2010. He works on parallel computing middleware (LANDA parallel environment), Grid resources management (AROMA project), computer and network modeling, load balancing with prediction models, autonomous policies to improve performance on distributed applications, parallelization of large electromagnetic simulation and autonomic middleware, and machine-to-machine architecture. He has managed a SUN microsystems center of excellence in the field of grid and cluster for network applications and a Cisco academy. Since 2011, he coordinates the industrial SOP project funded by ANR that creates hybrid cloud for personal service over ADSL network under energy and quality of service constraints. He is member of ETSI (European Telecommunication Standards Institute) and the contact for CNRS. He also represents CNRS in the eclipse foundation and co-leads the OM2M open source project. He is author of more than 50 regular and invited papers in conferences and journals.