Challenges when Realizing a Fully Distributed Internet-of-Things – How we Created the SensibleThings Platform

Stefan Forsström, Victor Kardeby, Patrik Österberg, and Ulf Jennehag
Department of Information and Communication Systems
Mid Sweden University
Sundsvall, Sweden
Email: {stefan.forsstrom, victor.kardeby, patrik.osterberg, ulf.jennehag}@miun.se

***Abstract*—The SensibleThings platform is an open source architecture for enabling Internet-of-Things based applications. During its development, multiple problems have been faced and solved, for example issues related to networking, information dissemination, sensors, and application access. This paper describes these problems and the technical solutions that are implemented in the platform. We also present the current progress and a series of demonstrator applications, which show the wide range of possibilities enabled by the platform. Finally, we present future work and how it will be used in future research endeavors and commercial interests.**

***Keywords*-overlay;sensors;actuators;internet-of-things.**

I. Introduction

Applications that utilize information from sensors attached to different things in order to provide more personalized, automatized, or even intelligent behavior are commonly referred to as Internet-of-Things (IoT) applications [1] or Machine-to-Machine (M2M) applications [2]. The prediction is that these kinds of applications will be able to interact with an IoT, a worldwide network of interconnected everyday objects, and thereby be able to display context-aware behavior [3]. These applications may address a variety of areas, such as environmental monitoring (pollution, earth quake, flooding, forest fire), energy conservation (optimization), security (traffic, fire, surveillance), safety (health care, elderly care), and enhancement of social experiences. IoT applications will probably have a big impact on how we interact with people, things, and the entire world in the future.

There is also an interesting relationship between the IoT and big data [4], since all of the connected things will produce and consume large amounts of data. Current estimations are in the order of 50 billion connected devices year 2020 [5]. In order to enable a widespread proliferation of IoT services there must be a common platform for dissemination of sensor and actuator information on a global scale. This is however a very difficult goal to achieve, because there is a large number of practical difficulties that must be solved. We state that applications on the IoT require the following from an underlying platform.

1) The platform must be able to quickly disseminate information to end points. The communication should be done with low overhead and there should be no unnecessary proxying of data. This due to that there exists many scenarios with real-time constraints for IoT applications.
2) The platform must be stable and handle devices joining and leaving the system with high churn rates. There should be no central points of failure and it should be possible for the system to heal itself, even when a large number of devices leave at the same time. For example in IoT scenarios with high mobility.
3) The platform must be lightweight enough to run on devices with limited hardware resources, such as mobile devices, small computers, and sensor motes. Hence, computational heavy algorithms and large amounts of data storage is not possible in such end devices. It is reasonable to expect that the IoT will include a wide range of different devices with varying computational and storage resources.
4) The platform must be extensive and adaptive to conform with a wide range of possible applications and devices. Applications should be able to create new functionality on top of the platform, suiting their own needs. This due to that IoT applications spreads across a wide range of scenarios and there might be unknown possible future scenarios.
5) The platform must be easy to adopt and free to use in commercial products. Since the goal is global proliferation, there should be no restrictions in terms of software licenses and fees related to the code for those companies or enterprises wishing to utilize the platform. Also, IoT applications must be viable for pure commercial interests, not just as research implementations.

This paper addresses the practical difficulties of facing the above mentioned requirements. We first present a short survey of related work in terms of the different IoT platforms currently available, including their relation to the requirements. We then present our solution called the SensibleThings platform, describing how it is designed and implemented to meet all of the requirements in a satisfactory manner.

The paper is outlined as follow. Section II presents the survey over currently available platforms. Section III describes the SensibleThings platform, whereas Section IV focuses on the problems faced during the processes. Section V discusses current results and possible applications. Finally, Section VI presents the conclusion and future research.

II. Related Work

There currently exists a vast amount of platforms which claims to enable an IoT, far more than can be listed in this paper. However, they can generally be categorized into three categories, centralized (or cloud distributed), semi-distributed, and fully distributed systems.

A. Centralized Systems

Most of the systems being released today seem to focus on distributing the data on some form of cloud-based IoT architecture. The cloud is a concept that rise in popularity, but it is in many cases simply a new word for traditional web services. Very few of the cloud based systems explain how the distribution and synchronization is actually done inside their architecture, how many servers they use, etc. Either way, cloud-based systems can be considered as centralized systems since they always relay the sensor and actuator information through a centralized point, in this case a cloud (be it one or many connected servers). The main problems with cloud oriented solutions are that they have difficulties achieving requirement 1 on direct communication between end devices, requirement 2 on no central points of failure, and requirement 5 on an open and free to use system, because they are based on large scale servers. Typical examples of these cloud based architectures include: SicsthSense [6], ThingSpeak [7], Sen.Se [8], Nimbits [9], ThingSquare [10], EVRYTHNG [11], Paraimpu [12], Xively [13], XOBXOB [14], Thingworx [15], One Platform [16], Carriots [17], and many more.

B. Semi-Distributed Systems

The semi-distributed systems are often based on session initiation protocols, whereas they afterward use direct communication between the connected devices. Because of this, they usually contain a centralized point for coordinating the communication. Thus, semi-distributed systems are faster and to some extent easier to scale than centralized solutions, but they still have difficulties coping with requirement 2 and 5. Typical examples of these semi-distributed architectures include: ETSI M2M [18], SENSEI [19], ADAMANTIUM [20], and other platforms based on 3GPP IMS [21].

C. Fully Distributed Systems

Fully distributed systems operate in a peer-to-peer manner, where they both store and administer the information locally on each entity. To achieve this, they often utilize hash tables to enable logarithmic scaling when the number of entities increases in magnitude. These systems do not contain any single point of failure and are thus more resilient, though the distribution itself often requires additional overhead in order to maintain an overlay. The main problem associated with fully distributed systems is however that they place a larger responsibility on the end devices, and thus have difficult to achieve requirement 3. Examples of such systems are SOFIA [22], COSMOS [23], and MediaSense [24].

III. The SensibleThings Platform

In order to solve the problem and address the stated requirements we have created the SensibleThings platform, which is a realization and implementation of the MediaSense architecture explained in [24]. The SensibleThings platform can be seen in figure 1, which presents the different layers and components of the platform. These include an interface layer, an add-in layer, a dissemination layer, a networking layer, and a sensor/actuator layer. The layers are explained in detail in the original article, but they will be summarized here as well. The actual SensibleThings code is based on a fork of the MediaSense platform, but has been significantly improved since then. The focus has been on the open source aspect and maintaining the commercialization possibilities of applications that are utilizing the platform. Other differences include the actual implementation of the lookup architecture, communication protocol, and code interfaces.

A. Interface Layer

The interface layer is the public interface through which applications interact with the SensibleThings platform. The interface layer includes a single component, the SensibleThings application interface, which is a generic Application Programming Interface (API) for developers to build their own applications on top of.

B. Add-in Layer

The add-in layer enables developers to add optional functionality and optimization algorithms to the platform. Add-ins can for example help the platform meet specific application requirements, such as specifically demanded features or handle the available capacity in regards to computational power and bandwidth. The add-in layer manages different extensible and pluggable add-ins, which can be loaded and unloaded in run-time when needed. These add-ins are divided into optimization and extension components, but the platform can include any number of them at the same time.

C. Dissemination Layer

The dissemination layer enables dissemination of information between all entities that participate in the system and are connected to the platform. A variant of the Distributed Context eXchange Protocol (DCXP) is used, which offers communication among entities that have joined a peer-to-peer network, enabling exchange of context or sensor information in real-time. The operation of the DCXP includes resolving of so called Universal Context Identifiers (UCI) and subsequently transferring context information directly. Therefore, the dissemination layer includes three components, a dissemination core, a lookup service, and a communication system. The dissemination core exposes the primitive functions provided by DCXP, the lookup service stores and resolves UCIs within the system, and the communication component abstracts transport layer communication. In short, the dissemination layer enables registration of sensors in the platform, resolving the location of a sensor in order to find it, and the communication to retrieve the actual sensor values.

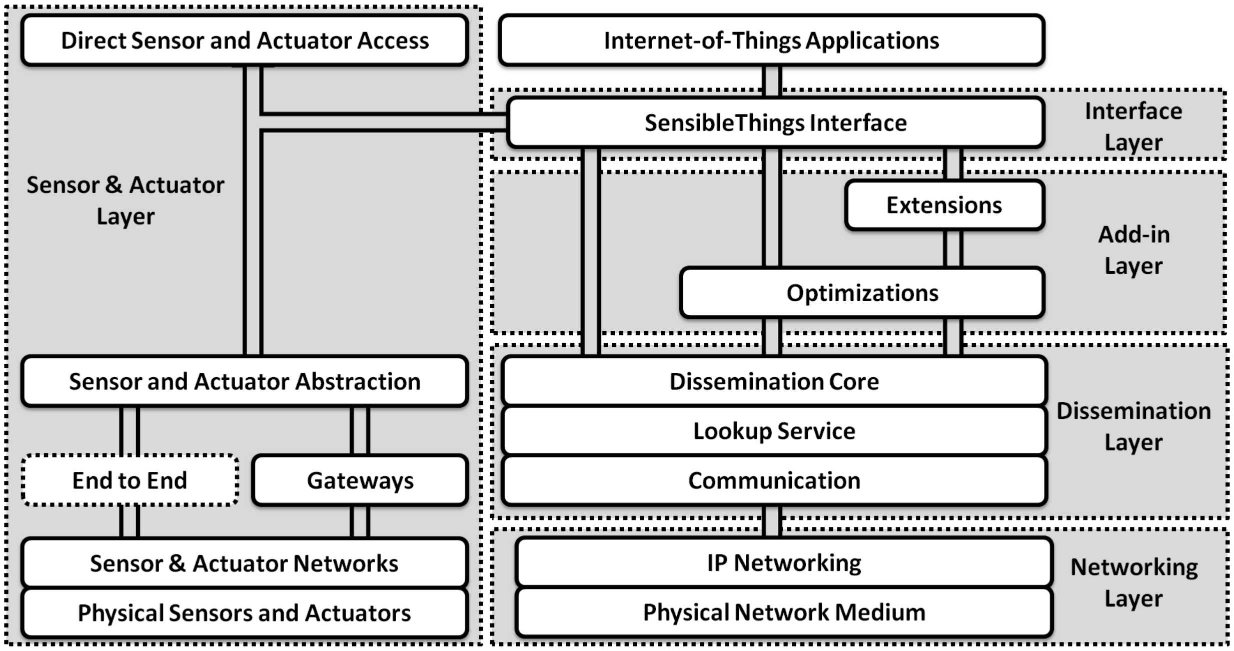


Figure 1. Overview of the SensibleThings platform's architecture

D. Networking Layer

The networking layer enables connection of different entities over current Internet Protocol (IP) based infrastructure, such as fiber optic networks or wireless and mobile networks. Hence, the networking layer is separated into two inner components, an IP network and the physical network medium. In short, the networking layer thus abstracts any underlying IP-based network architecture.

E. Sensor and Actuator Layer

The sensor and actuator layer enables different sensors and actuators to connect into the platform. The sensors and actuators can vary greatly and the platform therefore offers two options to connect them. Firstly, they can be connected directly if they are accessible from the application code, such as in the case of smartphone sensors. Secondly, the sensors and actuators can connect through the sensor and actuator abstraction. The abstraction enables connectivity either directly to wireless sensor networks or via more powerful gateways. Hence, the sensor and actuator layer is separated into five components: the directly accessible sensors and actuators, an abstraction component, different sensor and actuator networks, sensor and actuator gateways, and the physical sensors and actuators.

IV. Encountered Problems

This section outlines the different problems encountered during the development of the SensibleThings platform. The problems are divided according to what layer they belong to and explained in the following subsections. The last subsection describes issues related to the source code licensing.

A. Interface Layer Problems

The main problems of the interface layer were related to requirement 5 on easy usage, how to make the platform easy to understand and easy to implement. Different approaches were explored, but since almost all communication on the platform is done asynchronously, the listener java pattern is typically used in the application interface. Hence, almost all interface access with the platform is done through normal function calls, whereas the values are returned in event listeners.

B. Add-in Layer Problems

The add-in layer have also posed some specific problems, most prominently how the add-ins should be managed, loaded, and the API's chain of command. This relates to requirement 4 on extensibility and in requirement 5 on easy usage. There are also decisions to be made on what parts of the platform's API that should be considered primitive actions or be provided as add-in features. In the current platform, there are still some limitations as these issues have not been prioritized. For example, the add-ins have no chain of command and will therefore hijack functionality of the platform when enabled. Thus, some add-ins become mutually exclusive and will not function properly together.

C. Dissemination Layer Problems

The main problem faced when developing the dissemination layer was regarding the choice of lookup service that supports requirement 1, 2, and 3, namely quick dissemination, good scalability, and lightweight operation. There exists a number of Distributed Hash Tables (DHT) that the platform could use, where the most prominent are Chord [25], Kelips [26], and P-Grid [27]. All three choices have their separate advantages and disadvantages. Chord uses a ring structure which is difficult to maintain and has a logarithmic lookup time $O(log(N))$. Kelips uses affinity groups with a much simpler synchronization scheme and has fixed lookup time of $O(1)$, but it does not scale as well and has a larger overhead. P-Grid has a trie based structure with a logarithmic lookup time of $O(0.5\ log(N))$, but is more complex and difficult to maintain.

In the current platform, both Chord and Kelips are completely reimplemented to operate within the platform and with the same license as the rest of the code. We have also experimented with the currently available P-Grid code, but since that is using multiple source code licenses (including propagating open source licenses), it cannot be a part of the SensibleThings code at this stage. Currently the platform defaults to the Kelips DHT implementation, simply because that code is more stable than the Chord implementation when nodes join and leave rapidly.

The second problem faced in the dissemination layer was the choice of communication protocol. Requirement 1 states that the communication should be fast with low overhead. Therefore, the aim was to have the useful payload data already in the first packet. Because of this, a variant of a Reliable User Datagram Protocol (RUDP) is utilized as the default protocol. The problem with RUDP is however that the packets are sent in clear text, but to support industry applications the platform must provide the possibility of encryption. There exists several approaches for enabling this, such as different key exchange schemes with varying degrees of security and overhead. In the end, the decision was to support standard Secure Sockets Layer (SSL) encryption to at least make it possible to encrypt the data if needed. The encryption is however only useful to prevent eavesdropping, not man in the middle attacks, because all certificates will be self signed by the end devices. There is also a significant overhead related to SSL, and since there is an initial handshake the data will not come in the first packets.

There have also occurred different problems in relation to the serialization of messages, how the messages are coded when sent over the Internet (before any encryption). A binary serialization format is most suitable from a performance perspective, but a text based format is most usable from a human-readable perspective and a code-specific format is most easy to program. In the end, the choice was to support all different serialization formats but the default is set to Java's object serialization, to make it easier to develop new extensions. Likely, the Java serializer will be replaced in the future, in order to make transitions to other platforms and programming languages feasible.

D. Networking Layer Problems

The major problem faced in the networking layer was related to requirement 2 on stability and seamless communication. The first versions of the platform did not take Network Address Translation (NAT) and firewalls into consideration, it only worked if all devices was on the public Internet. However, today almost all consumer devices are connected to the Internet through either NAT or some type of firewall, either in their home or at their work, but also on the mobile phone networks. The NAT and firewall problem is only a question of configuration, given that the user is allowed to enable features such as port forwarding on the NAT routers, but that this rarely the case. For example, most normal users simply want things to work out of the box, most companies do not allow their employees to enable such features on their company network, and many Internet service providers are now enabling carrier grade NAT [28].

Multiple approaches were considered, ranging from IP version 6 (IPv6) solutions, to Universal Plug and Play (UPNP), different hole punching techniques, and finally simple proxy solutions. The chosen solution first tries the normal approaches, such as direct connections and UPNP. If this fails it instead utilizes distributed proxy nodes in the system. As the proxy solution stands in direct contradiction to requirement 1 on real-time communication without unnecessary relaying of information and requirement 2 on no central points of failure, it is only used as a last resort when there are no other solutions available.

There also exists problems related to the capacity of the Internet connections and their network delay, but since the SensibleThings platform is built on top of the existing Internet architecture, it cannot affect these parameters. Therefore, as long as useful payload data is sent in the first packet, the assumption is that it is being transmitted as fast as possible by the underlying network infrastructure.

E. Sensor and Actuator Layer Problems

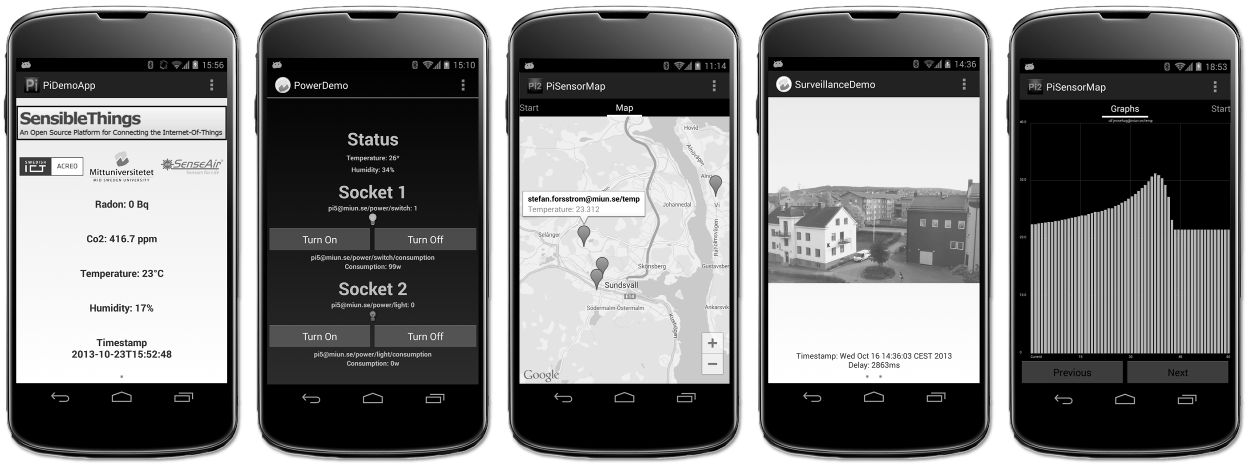
In the sensor and actuator layer, there were problems with the actual sensor hardware platforms that is available today, especially in regards to requirement 3 on being lightweight. Different vendors of sensors have different platforms that the sensors run on, especially when it comes to connecting large Wireless Sensor Networks (WSN). Typically, cheap analog sensors can be connected directly to a more powerful device, such as a smartphone or a Raspberry Pi [29]. But to connect traditional WSN architectures such as TinyOS [30] or Contiki [31], the platform must communicate via COAP [32] or other lightweight protocols that they can handle. Therefore, in most of the examples we have utilized either smartphones with sensors already built in, or Raspberry Pi devices with attached sensors. However, any device that can run the Java code for the platform can be a part of the system, and any low end device that can communicate via COAP can easily be connected via a more capable device.

F. Source Code License Problems

One purpose of the platform is to make it available for industry partners to develop their own applications and then commercialize the products, see requirement 5. This requirement made it impossible to use a strict and propagating open source license such as GNU General Public License (GPL). The amount of external code should also be kept to a minimum, in order to maintain the control over all the licenses in use. In the end, the decision was to use the GNU Lesser General Public License (LGPL) that allows companies to make commercial products on top of the platform, without forcing their products to be open source as well.

V. Results and Applications

The current results include launching our new development website for the SensibleThings platform [33]. This website will act as a portal for all developers who want to utilize the platform in their applications. The SensibleThings platform is provided free and under an LGPL version 3 open source license. Initial testing, demonstration, and evaluation of the platform has been conducted using a testbed with fixed and



(a) Sensor reading (b) Intelligent home (c) Object tracking (d) Surveillance (e) Historical values

Figure 2. Examples of applications using the SensibleThings platform.

mobile access to the Internet. In terms of performance we have measured the platform to be on par with UDP traffic. Table I shows the response times measured for resolving, and retrieving a specific sensor value in the platform. The table shows both the arithmetic mean μ and the standard deviation σ. The measurements were conducted with 103 devices connected to the platform. One workstation, one Raspberry Pi, and 100 emulated devices connected via 1 Gbit fiber optic based Internet connection. Lastly, one mobile device connected via 3G mobile Internet connection. The measurements were conducted on the workstation, the Raspberry Pi, and the mobile device, to show the difference between them and the effect of the proxy solution for NAT problems associated with the mobile device. The other 100 devices were put into the system to create background traffic, in order to emulate a real-world scenario with many connected devices.

Proof-of-concept demonstrator applications have been built using many different devices, sensors, and actuators, in order to show the versatility of the SensibleThings platform. The applications presented in figure 2 show some of these demonstrators. From left to right they are: (a) sensor value readings (radon, carbon dioxide, temperature, and humidity), (b) home automation with energy consumption measuring (for interacting with an intelligent home), (c) object tracking (for tracking different objects with attached sensors), (d) surveillance (for detecting trespassers), and (e) a set of historical measurements (for statistical usage). But the platform itself is versatile enough to be applied to even more areas. We foresee possible applications ranging from intelligent home, healthcare, logistics, emergency response, tourism, and smart-grids, to more social-oriented applications such as crowd sourcing, dating services, and intelligent collaborative reasoning.

VI. Conclusion and Future Work

In this paper we presented the challenges we have encountered when researching and developing the SensibleThings platform. The first contribution of this paper is the identification of the requirements for a functional and fully distributed IoT platform in Section I. The second contribution is a short survey of existing IoT platforms, presented in Section II.

TABLE I
Measured response times for platform.

		Workstation	Raspberry Pi	Mobile
Resolve	μ	4.8 ms	31 ms	230 ms
	σ	2.2 ms	14ms	68 ms
Retrieve	μ	4.8 ms	31 ms	280 ms
	σ	2.0 ms	12 ms	56 ms

The main contribution is however the explanation of the problems faced when building the SensibleThings platform, and the solutions that solve these problems. The proposed SensibleThings platform is shown to fulfill all the requirements stated in Section I. The platform can disseminate information to end devices quickly with low overhead (requirement 1). It is stable and operates without any central points of failure (requirement 2). The platform is lightweight and can run on mobile devices where the only central point is the bootstrap device (requirement 3), but any node can act as a bootstrap if this original node goes offline. It is extensible (requirement 4) as shown with the demonstrator applications. Finally, the platform is licensed under a well known and widely accepted open source license making it free to use and at the same time encouraging development of commercial a products (requirement 5). In comparison to related work, the SensibleThings platform can be classified as a fully distributed solution, where the overhead and communication is kept as lightweight as possible. We predict that this is the only type of solution that will scale for billions of connected devices and still be able to disseminate sensor information with real-time demands.

Our current efforts are directed toward improving and optimizing the existing code, as well as investigating other possible choices of DHT and communication protocol. We will also develop more extensions to satisfy specific applications demands, such as seamless integration with other IoT platforms and cloud infrastructures.

Acknowledgment

This research has been supported by grant 2010-00681 of VINNOVA the Swedish Governmental Agency for Innovation

Systems, and by grant 00163383 of the EU European Regional Development Fund, Mellersta Norrland, Sweden. The authors also want to thank our partners from industry and academia, in particular Acreo AB and Stockholm University.

References

[1] L. Atzori, A. Iera, and G. Morabito, “The internet of things: A survey,” Computer Networks, vol. 54, no. 15, 2010, pp. 2787–2805.

[2] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. Johnson, “M2M: From mobile to embedded Internet,” Communications Magazine, IEEE, vol. 49, no. 4, 2011, pp. 36–43.

[3] J. Hong, E. Suh, and S. Kim, “Context-aware systems: A literature review and classification,” Expert Systems with Applications, vol. 36, no. 4, 2009, pp. 8509–8522.

[4] D. E. O’Leary, “Big data, the internet of things and the internet of signs,” Intelligent Systems in Accounting, Finance and Management, vol. 20, no. 1, 2013, pp. 53–65.

[5] Ericsson. More than 50 billion connected devices. White Paper. [Online]. Available: http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf [retrieved: December, 2013]

[6] SicsthSense. [Online]. Available: http://sense.sics.se [retrieved: December, 2013]

[7] ThingSpeak. [Online]. Available: https://www.thingspeak.com [retrieved: December, 2013]

[8] Sen.se. [Online]. Available: http://open.sen.se [retrieved: December, 2013]

[9] Nimbits. [Online]. Available: http://www.nimbits.com [retrieved: December, 2013]

[10] Thingsquare. [Online]. Available: http://thingsquare.com [retrieved: December, 2013]

[11] EVRYTHNG. [Online]. Available: http://www.evrythng.com [retrieved: December, 2013]

[12] Paraimpu. [Online]. Available: http://paraimpu.crs4.it [retrieved: December, 2013]

[13] Xively. [Online]. Available: https://xively.com [retrieved: December, 2013]

[14] XOBXOB. [Online]. Available: http://www.xobxob.com [retrieved: December, 2013]

[15] ThingWorx. [Online]. Available: http://www.thingworx.com [retrieved: December, 2013]

[16] One platform. [Online]. Available: http://exosite.com/products/onep [retrieved: December, 2013]

[17] Carriots. [Online]. Available: https://www.carriots.com [retrieved: December, 2013]

[18] ETSI, “Machine-to-machine communications (m2m); functional architecture,” (TS 102 690 V1.1.1), Tech. Rep., 2011.

[19] M. Presser, P. Barnaghi, M. Eurich, and C. Villalonga, “The SENSEI project: integrating the physical world with the digital world of the network of the future,” Communications Magazine, IEEE, vol. 47, no. 4, 2009, pp. 1–4.

[20] H. Koumaras, D. Negrou, F. Liberal, J. Arauz, and A. Kourtis, “ADAMANTIUM project: Enhancing IMS with a PQoS-aware multimedia content management system,” International Conference on Automation, Quality and Testing, Robotics, vol. 1, 2008, pp. 358–363.

[21] G. Camarillo and M.-A. Garcia-Martin, The 3G IP multimedia subsystem (IMS): merging the Internet and the cellular worlds. Wiley, 2007.

[22] A. Toninelli, S. Pantsar-Syväniemi, P. Bellavista, and E. Ovaska, “Supporting context awareness in smart environments: a scalable approach to information interoperability,” in Proceedings of the International Workshop on Middleware for Pervasive Mobile and Embedded Computing. ACM, 2009, pp. 1–4.

[23] P. Bellavista, R. Montanari, and D. Tibaldi, “Cosmos: A Context-Centric Access Control Middleware for Mobile Environments,” in Mobile Agents for Telecommunication Applications, 2003, pp. 77–88.

[24] T. Kanter, S. Forsström, V. Kardeby, J. Walters, U. Jennehag, and P. Österberg, “Mediasense–an internet of things platform for scalable and decentralized context sharing and control,” in ICDT 2012, The Seventh International Conference on Digital Telecommunications, 2012, pp. 27–32.

[25] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, vol. 31. New Yourk, NY, USA: ACM Press, 2001, pp. 149–160.

[26] I. Gupta, K. Birman, P. Linga, A. Demers, and R. Van Renesse, “Kelips: Building an efficient and stable p2p dht through increased memory and background overhead,” in Peer-to-Peer Systems II. Springer, 2003, pp. 160–169.

[27] K. Aberer, “P-grid: A self-organizing access structure for p2p information systems,” in Cooperative Information Systems. Springer, 2001, pp. 179–194.

[28] S. Jiang, D. Guo, and B. Carpenter, “An Incremental Carrier-Grade NAT (CGN) for IPv6 Transition,” RFC 6264, Tech. Rep., 2011.

[29] Raspberry pi. [Online]. Available: http://www.raspberrypi.org [retrieved: December, 2013]

[30] P. Levis et al., “TinyOS: An operating system for sensor networks,” in Ambient intelligence. Springer, 2005, pp. 115–148.

[31] A. Dunkels, B. Gronvall, and T. Voigt, “Contiki-a lightweight and flexible operating system for tiny networked sensors,” in Local Computer Networks, 2004. 29th Annual IEEE International Conference on. IEEE, 2004, pp. 455–462.

[32] Z. Shelby, K. Hartke, and C. Bormann. Constrained application protocol (COAP). [Online]. Available: http://tools.ietf.org/html/ietf-core-coap-14.txt (2013)

[33] SensibleThings. [Online]. Available: http://www.sensiblethings.se [retrieved: December, 2013]