Extending the Semantic Web to Peer-to-Peer-like Sensor Networks based on XMPP

Peter Waher

Clayster Laboratorios Chile S.A, Blanco 1623, of. 1402, Valparaíso, Chile
peter.waher@clayster.com

Abstract. This paper provides a novel approach in bridging the traditional semantic web based on the HTTP protocol and peer-to-peer-like sensor networks based on the XMPP protocol, thus extending the reach of semantic technologies to private spheres otherwise not accessible due to firewalls and other security issues, but still maintaining a high level of security and end-user data privacy and access control.

Keywords: Internet of Things, Peer-to-peer networks, Semantic web, data privacy, access control, plug computing, grid computing.

1 Introduction

The use of Semantic Web technologies [1] within the Internet of Things is a very promising area of research and development. It allows for the unification of a huge amount of proprietary APIs into one set of standardized APIs for accessing and linking data not only between Things in the network, but also between services and consumers, regardless of who has published the data, and where it is stored. It replaces the more difficult problem of implementing and supporting a huge array of proprietary communication APIs and protocols with the simpler problem of mapping or understanding the information provided by different Things all provided using standardized communication and data extraction techniques.

The traditional way of basing the Semantic Web on the HTTP protocol [2] however, has implied several limitations on the usefulness of this technique, especially when Things move closer to the private spheres of end-users, such as within people's homes, inside office buildings, etc. As long as all Things are publicly available on the Internet, HTTP works fine, but as soon as the content starts moving into areas where access is limited by firewalls, HTTP as a transport method starts failing, since connections most often can only be established from the inside out. If the SPARQL [3] endpoint resides outside of the firewall, it cannot reach Things residing inside the firewall using normal HTTP.

The same problem exists within the Internet of Things in general and not only to semantic web applications. All request/response-based communication protocols inherently have this problem. To solve this problem of communication between Things

behind firewalls within the Internet of Things community, various solutions have been proposed:

1. Publish/Subscribe architecture patterns
2. Cloud storage of data
3. Peer-to-Peer communication
4. Hybrid approaches

1.1 Publish/Subscribe architecture pattern

The publish/subscribe architecture pattern [4] basically consists of three types of actors: Publishers, message brokers and subscribers. Publishers generate content and publish it to a message broker. The message broker then immediately distributes the content, or information about the content, to subscribers having subscribed to the particular content.

Here, publishers and subscribers connect to the message broker and can therefore reside behind firewalls. The message broker however, needs to be reachable by all actors in the network.

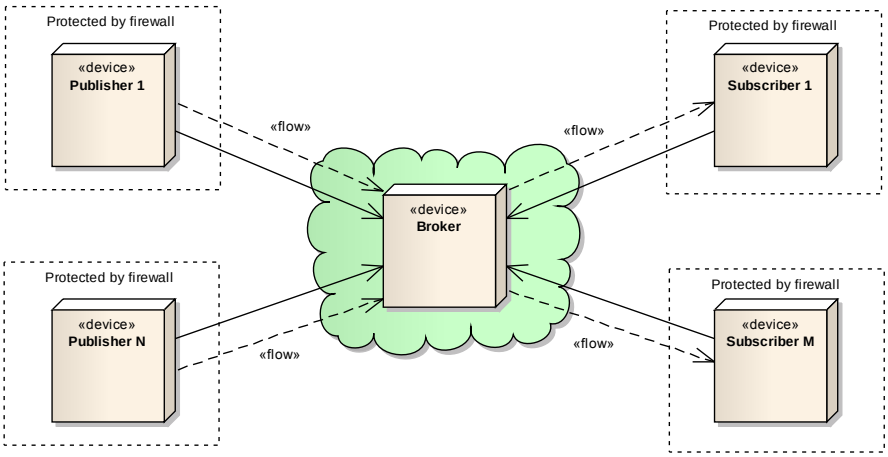


Fig. 1. Publish/Subscribe pattern

Here, all publishers and subscribers connect to the broker, bypassing any firewall restrictions on traffic in the opposite direction. Subscribers are also required to maintain the connection with the broker, to be able to receive the corresponding information. The flow of information however, is from left to right in the image. Publishers can of course be subscribers also.

The publish/subscribe architecture is very efficient in distributing messages in large networks, however, it only supports one type of communication pattern: Things publishing information, consumed by others. It is very difficult to create an environ-

ment permitting two-way communication and impossible for the publishers to control who gets access to what, which makes it impractical and useless in semantic web applications.

There are various protocols that readily provides publish/subscribe architecture support, like XMPP [5] with its publish-subscribe extension [6]. There are also protocols or platforms that are designed primarily with publish/subscribe in mind, like MQTT [7]. One could also see web platforms as Twitter [8] working according to the publish/subscribe pattern.

1.2 Cloud storage of data

Cloud storage of data uses a similar approach as publish/subscribe, except that there's no actual subscription of data involved providing data in real-time. Publishers publish data to a server, which in turn always stores it for future access. Consumers of the data are required to poll data from the server regularly or on demand. There exist however solutions implementing custom triggers that can notify clients of events. These can in turn be used to closely mimic the publish/subscribe pattern of immediate content delivery to the final recipients.

Data published on servers are typically available through some sort of API. Common APIs can be RESTful web services [9] returning XML, either proprietary formats or standardized formats such as RSS [10] or ATOM [11]. JSON is also popular since it allows for easy implementation in script languages.

Examples of platforms that use this technique for Internet of Things applications include Xively [12], Open.sen.se [13], SicsthSense [14] etc.

Even though this architectural pattern partly lends itself to incorporation of Internet of Things into the semantic web, it does so only with difficulty and great limitations imposed on it:

First, only historic data published on the server will be available on the semantic web. There can be no direct interaction with the Things that published the information without introducing significant latency. Secondly, the publishers have no control of who can see what data. Thirdly, the publishers lose control and ownership of the data, making changes or removal difficult. Fourthly, the risk of exposing private data to unknown corporate or government interests is great if the data is stored centrally, minimizing the desire of end-users to use the approach when it comes to private information. Companies that want to exploit the information and sell data mining and big data services of course see no problem with this approach.

1.3 Peer-to-peer communication

Peer-to-peer communication techniques originally developed for file sharing, instant messaging or gaming applications have become a promising field of research for Internet of Things also, in particular since these techniques provide a mechanism for devices to talk to each other, even though they reside behind different firewalls.

There are many different peer-to-peer protocols available that solve the problem on how to bypass firewall in different ways. Following is a non-exhaustive list of popular ways to achieve peer-to-peer functionality in networks:

Internet Group Management Protocol IGMP [15] is a way to send IP multicast messages. In networks where firewalls permit IGMP communication peers can subscribe to IP multicast addresses and routers will forward communication to all subscribed peers. Anyone partaking in a conversation can both send and receive information. The architecture is similar to that of the publish/subscribe pattern, with the exception that routing is done on a network layer in routers and not in the application layer of a message broker. Also, IGMP doesn't allow for a fine grained subscription model, where you can subscribe to specific topics. Anything sent on a multi-cast channel will be received by all subscribers of that channel, unless packets are lost. Secure packet delivery is not available. Another big disadvantage is that everybody subscribed to the multicast address will receive all messages. CPU load will increase as the number of senders grows, and it is difficult to send private messages.

IGMP is popular in streaming services, especially in IP-TV networks, as it allows for efficient distribution of information where loss of packets doesn't affect the quality of the service. In order to create point-to-point communication using IGMP it is required to add a secondary addressing mechanism to make sure the recipients know what packets are meant for them and what are meant for others. For this reason IGMP only serves to solve certain aspects of IoT-based communication, mostly concerned with discovery of devices. Examples that use this technique for device discovery include SSDP [16] on which UPnP [17] and DLNA [18] are based. Multicast DNS [19] is another discovery method using IGMP. Multicast DNS is used by Bonjour [20] and XMPP server-less messaging [21].

To circumvent the problems of multicasting, and create a Peer-to-peer protocol based on single-casting, i.e. point-to-point communication, other mechanisms have to be incorporated to bypass any firewalls. One such collection of methods goes by the collective name NAT transversal [22]. It includes a series of different techniques, none guaranteed to work in all settings, since NAT transversal is not standardized. Basically it includes methods where the firewalls are programmed to forward messages received on their public IP addresses, with given port numbers, to corresponding private IP addresses and corresponding port numbers behind the firewall. This sometimes goes by the name of punching holes in the firewall. It can also include the incorporation of publically available servers that route messages, just like message brokers do.

Unbeknown to many, NAT transversal may actually create a big security problem for its users, since it partly removes the original function of the firewall: Not let unauthorized or unauthenticated users access private resources on the network. Popular protocols such as UPnP allows devices to automatically punch holes in the firewall allowing external actors, friends but also foes, to access devices on the network, creating security holes that are easy to exploit [23].

Peer-to-peer networks are normally divided into two different types of networks: Unstructured and structured peer-to-peer networks. Unstructured networks have no explicit network topology, and peers connect to each other "randomly" or though

friendship requests. One of the problems such networks have is that finding useful resources is a difficult problem. Normally, peers can only ask known peers if they have access to a given resource. These peers can forward the question to their peers, and so on, until the resource if found. This normally works well for well-known resources. But for scarce resources, such questions impose a great load on the network.

To solve this problem, many solutions have been developed defining an explicit network topology including centralized resources to manage content, searching, access privileges, etc. These solutions range from content directories, to access privileges, friendships, scheduling, task lists, etc. Even though they require the use of centralized publically available servers they are considered peer-to-peer networks, albeit structured peer-to-peer networks, as the actual peer-to-peer communication is later done directly between peers.

1.4 Hybrid approaches

Had it not been for the security issues described in the previous section, peer-to-peer network architecture might have been a perfect candidate for the Internet of Things due to its flexibility when it comes to point-to-point communication between peers in the network. The publish/subscribe pattern described earlier does not have this vulnerability: As devices behind firewalls all connect to a message broker that redistributes messages to interested parties, and no holes are punched in the firewall, it's impossible for external parties to connect directly to the device.

This motivates the use of hybrid approaches, using federated message brokers, but having an architecture permitting point-to-point communication instead of one-to-many types of communication. As devices all connect to a message broker, external entities cannot connect to the devices, unless the message broker authenticates the device and authorizes its relationship with the original device. Even though this adds a component to the network, it is not much different from other publically available components available in structured peer-to-peer networks, as described earlier. It is even similar to the case where NAT-transversal requires the use of a public proxy server to forward messages between peers. For this reason, we will call this hybrid approach **peer-to-peer-like communication**. It works as a peer-to-peer protocol on the application layer, but not on the network layer.

Apart from fulfilling these requirements we also want the protocol to be open, standardized, efficient and easy to extend without the possibility of confusion. XMPP meets all these requirements [24]. XMPP was originally defined for use in instant messaging applications, which can be seen in the acronym "eXtensible Messaging and Presence Protocol". It is based on XML and use of namespaces makes XMPP extensible and easy to extend without creating conflicts. It is also standardized by IETF. Extensions to the protocol are published and maintained by the XMPP Standards Foundation [25]. There's a huge list of extensions available to XMPP [26], showing how the protocol has grown from its original domain to become a versatile protocol for the Internet in general and the Internet of Things in particular. As will be shown in this paper, XMPP also provides architectural support for a logical extension of the Semantic Web into the Internet of Things. A recent extension also provides a

mechanism of efficiently compressing XMPP messages, permitting the use of XMPP in wireless sensor networks with limited maximum package sizes [27].

XMPP also adds a security mechanism whereby clients are authenticated, and the broker also makes sure each client sending a message to another is authorized to do so. This adds a layer of added security to the network. A recent extension of the protocol permits even better control of who can talk to whom, and what they can talk about, what services are available to whom, permitting provisioning of devices and services in Internet of Things networks [28]. If end-to-end encryption is desired, to make sure the message broker cannot eavesdrop on the conversation, work is being done within IETF to solve this issue as well [29]. Extensions also exist for communication of sensor data [30], controlling devices [31] and bridging legacy or proprietary protocols and interfacing subsystems [32].

Furthermore, XMPP can be used in server-less mode using a small memory footprint, as is demonstrated by Ronny Klauck and Michael Kirsche in their work related to Chatty Things [33] [34]. Interesting Internet of Things-related projects and groups include a working group within IEEE/IEC/ISO [35], KTC [36] [37] and SUST [38] [39]. Work is also underway to define common interoperable interfaces for Internet of Things, based on available extensions [40]. A repository for XMPP-related research papers is also available at Mendeley [41]. The XMPP wiki also has a section dedicated to the Internet of Things [42].

2 Bridging the semantic web and XMPP networks

As was described in the previous section, the peer-to-peer-like network architecture provided by XMPP and available extensions permits us to create secure and interoperable networks for the Internet of Things, including an architecture for provisioning with fine-grained control of who can talk to whom, who has access to what information, who can control what and what services should be available to whom. This section will describe how this architecture can be used by semantic web applications as well.

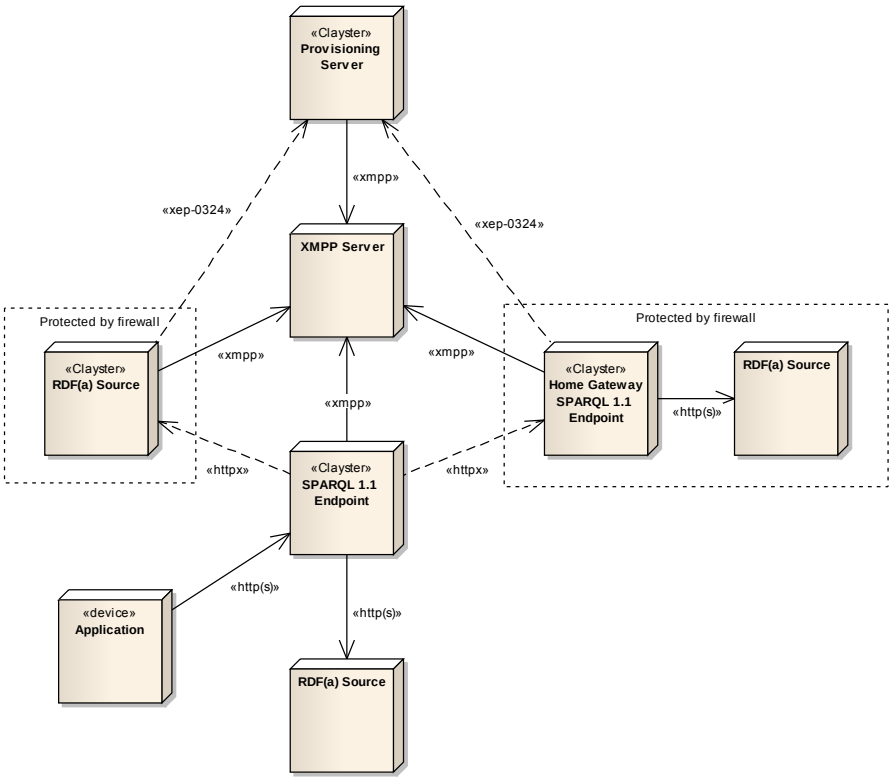
Traditional semantic web applications are forced to use normal HTTP over TCP or TLS connections. Security is limited to HTTP-based authentication which is very coarse and difficult to implement and manage on small devices. Leaving the implementation of web security to device manufacturers furthermore increases the risk of it being completely ignored. Reutilizing the existing security framework provided by XMPP networks, where such security features as user authentication and authorization is automatically provided by the message broker, therefore automatically provides the network with a better security model than what can be provided by normal HTTP over TCP or TLS.

The HTTP over XMPP transport extension [43] provides a mechanism to transport ordinary HTTP requests and responses over an XMPP network. Apart of supporting different HTTP versions, all HTTP methods and HTTP header semantics, it allows for various efficient encodings and transport schemes for efficient transfer of different types of data, including text, XML (allowing efficient compression if EXI is used),

binary base64-encoding, chunked encoding for dynamic content, file transfer [44] for transfer of existing files, In-band byte streams [45] and Jingle [46] for different types of streaming.

It also proposes a new URI scheme: **httpx** for easy integration into systems and browsers using URLs to identify resources. As the Semantic Web and Linked Data are based on IRIs, IRI being a generalization of URI, this permits the extension of the Semantic Web onto peer-to-peer-like XMPP networks seamlessly.

The Clayster platform [47] has been used to build various semantic web applications in different constellations. The Clayster platform can be hosted on both Windows servers in clustered environments, or on Linux platforms using Mono [48], and provides a web 3.0[1] runtime environment including a SPARQL 1.1 endpoint, web server, integrated XMPP support, pluggable Internet of Things architecture including multi-protocol support, pluggable object database and a powerful engine for 10-foot user interface [49] applications optimized for mobile phones, televisions and touch pads.



[1] Web use the term web 3.0 as a synonym of a distributed semantic web fused with Internet of Things. The fundament of web 3.0 [52] is Linked Data.

Fig. 2. Federated query accessing private information

Figure 2 shows a constellation where an application is asking a central web server running Clayster to create a report using a single federated SPARQL query. In the graph, solid lines represent actual socket connections, and the direction of the arrow shows the direction of the connection. Dashed lines represent peer-to-peer-like communication made over the underlying XMPP network and the direction of the arrow represents the direction of the request/response-based communication. The SPARQL engine joins data together from publically available RDF(a) data sources using normal HTTP or HTTPS, but also from privately hosted RDF data sources and from the results of federated queries to private SPARQL endpoints behind firewalls. In these cases, the httpx URI scheme [43] is used. Both the private RDF source and the private SPARQL Endpoint are hosted on separate Clayster platforms. The XMPP server acts as a guarantor that the main SPARQL endpoint can access the private data. Since XMPP is used, no holes are punched in the firewalls, and the private data remains private and only accessible from parties with access according to the provisioning server (also hosted on the Clayster platform). The provisioning server can also provide fine-grained control on what data from what devices the end user has the right to see.

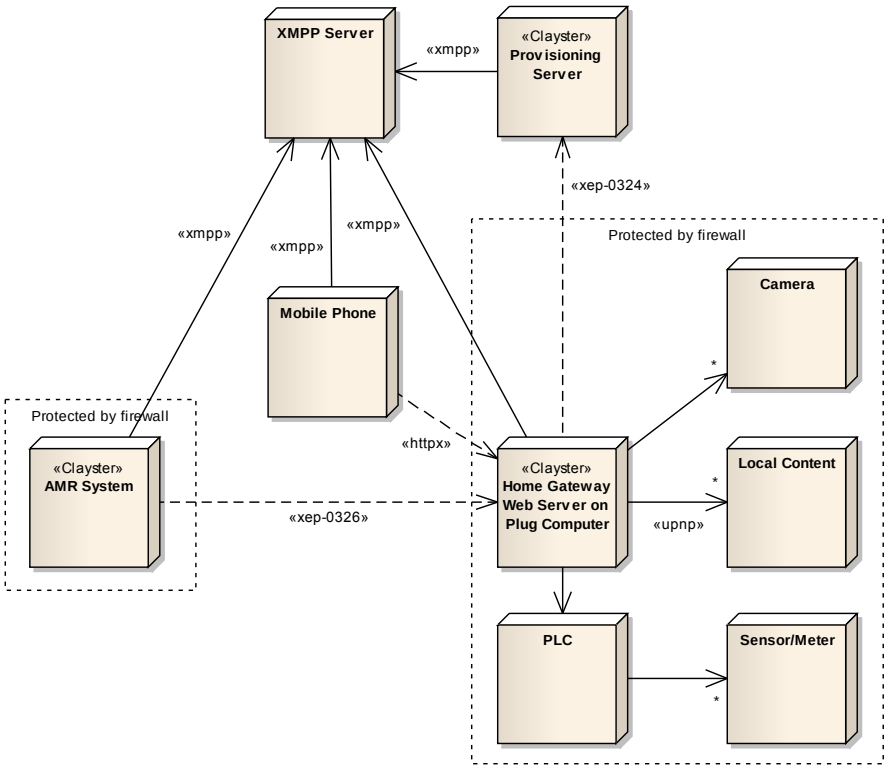


Fig. 3. Mobile phone accessing private web application

Figure 3 shows another constellation using the same protocols achieving a completely different application. This time, it is a web application hosted on a plug computer running the Clayster platform on Mono. Instead of hosting private information like private photos, videos, cameras at home, etc., on a web server in the cloud, the content is hosted privately on a cheap and power efficient plug computer. It is accessible as a normal web application within the boundaries of the local network protected by the firewall. But by using the httpx URI scheme the web application becomes available from everywhere where the federated XMPP server is reachable, but only to authenticated and authorized parties. Which ones is controlled by the XMPP Server and complemented by the provisioning server. The web application can also access private content in other domains, such as for example, the web camera in a common room or a neighbor that trusts you, by returning URL's to the content, using the httpx URI scheme itself. The provisioning server gives added control of what can be shared with what parties. In the example above, the mobile phone would have access to all local content, sensor values and cameras, while an Automated Reading System would only have access to the Accumulated Energy value of a given Electricity Meter, and no other sensor values.

3 Conclusion

Semantic web technologies provide for a very powerful set of tools to be used within the Internet of Things. Not only do semantic technologies provide a powerful abstraction of data, the technologies also resolve the problem of maintaining uncountable number of different proprietary APIs for communication with devices or systems from different manufacturers. Through the use of federation semantic web technologies also provide a standardized way to perform actions on a grid of devices as a whole.

The biggest challenge of semantic web technologies however, is how to solve access rights to private information, which is of paramount importance to the Internet of Things. This cannot be sufficiently solved by using the traditional HTTP model. HTTP authentication simply does not provide sufficient protection, granularity and manageability across large networks of devices with limited user interfaces. Furthermore, the use of previous existing architectural patterns adapts very poorly to the semantic web or implies huge restrictions on the Internet of Things as a concept.

The introduction of HTTP over XMPP changes this radically. It permits access to HTTP resources behind firewalls without the use of unsafe firewall hole punching techniques, or without publishing private and sensitive information in the cloud. It furthermore allows the end-user in a simple way to control who gets access to the material. Using provisioning servers based on published extensions of XMPP furthermore permits fine-grained control of what data can be accessed by whom and which services they are allowed to use.

4 Acknowledgements

Financial support for this study was provided by KTC [50] and Manodo [51]. The author wishes to thank Dr. Yusuke DOI and Dr. Karin Forsell for their suggestions and comments on this document.

5 References

[1] W3C, "Semantic Web," [Online]. Available: http://www.w3.org/standards/semanticweb/.

[2] R. Fielding, U. Irvine, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach and T. Berners-Lee, "RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1," June 1999. [Online]. Available: http://tools.ietf.org/html/rfc2616.

[3] W3C, "SPARQL Current Status," [Online]. Available: http://www.w3.org/standards/techs/sparql.

[4] Publish-subscribe pattern, "Wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern. [Accessed 22 July 2013].

[5] XSF, "XMPP Standards Foundation," [Online]. Available: http://xmpp.org/.

[6] P. Millard, P. Saint-Andre and R. Meijer, "XEP-0060: Publish-Subscribe," 2002-2010. [Online]. Available: http://xmpp.org/extensions/xep-0060.html.

[7] "MQ Telemetry Transport," [Online]. Available: http://mqtt.org/.

[8] Twitter, "Developers," [Online]. Available: https://dev.twitter.com/.

[9] Wikipedia, "Representational state transfer," [Online]. Available: http://en.wikipedia.org/wiki/Representational_state_transfer. [Accessed 22 July 2013].

[10] Harvard Law, "RSS 2.0 Specification," 2003. [Online]. Available: http://cyber.law.harvard.edu/rss/rss.html. [Accessed 22 July 2013].

[11] M. Nottingham and R. Sayre, "RFC 4287: The Atom Syndication Format," December 2005. [Online]. Available: http://tools.ietf.org/html/rfc4287.

[12] Xively, "Xively cloud platform," [Online]. Available: https://xively.com/.

[13] Open.Sen.Se, "Open.Sen.Se cloud platform," [Online]. Available: http://open.sen.se/.

[14] SICS, "SicsthSense cloud platform," [Online]. Available: http://sense.sics.se/API.

[15] Wikipedia, "Internet Group management Protocol," [Online]. Available: https://en.wikipedia.org/wiki/Internet_Group_Management_Protocol. [Accessed 22 July 2013].

[16] Wikipedia, "”Simple Service Discovery Protocol," [Online]. Available:

http://en.wikipedia.org/wiki/Simple_Service_Discovery_Protocol. [Accessed 22 July 2013].

[17] Universal Plug and Play Forum, "UPnP," [Online]. Available: http://www.upnp.org/.

[18] Digital Living Network Alliance, "DLNA," [Online]. Available: http://www.dlna.org/.

[19] Wikipedia, "Multicast DNS," [Online]. Available: http://en.wikipedia.org/wiki/Multicast_DNS. [Accessed 22 July 2013].

[20] Wikipedia, "Bonjour," [Online]. Available: http://en.wikipedia.org/wiki/Bonjour_(software). [Accessed 22 July 2013].

[21] P. Saint-Andre, "XEP-0174: Serverless Messaging," 2008. [Online]. Available: http://xmpp.org/extensions/xep-0174.html.

[22] Wikipedia, "NAT traversal," [Online]. Available: http://en.wikipedia.org/wiki/NAT_traversal. [Accessed 22 July 2013].

[23] H. Moore, "Security Flaws in Universal Plug and Play: Unplug, Don't Play.," January 2013. [Online]. Available: https://community.rapid7.com/docs/DOC-2150. [Accessed 22 July 2013].

[24] "XMPP Technologies Overview," [Online]. Available: http://xmpp.org/about-xmpp/technology-overview/.

[25] "About XMPP Standards Foundation," [Online]. Available: http://xmpp.org/about-xmpp/xsf/.

[26] "XMPP Extensions," [Online]. Available: http://xmpp.org/xmpp-protocols/xmpp-extensions/.

[27] P. Waher and Y. DOI, "XEP-0322: Efficient XML Interchange (EXI) Format," 2013. [Online]. Available: http://xmpp.org/extensions/xep-0322.html.

[28] P. Waher, "XEP-0324: Internet of Things - Provisioning," 2013. [Online]. Available: http://xmpp.org/extensions/xep-0324.html.

[29] M. Miller, "End-to-End Object Encryption and Signatures for the Extensible Messaging and Presence Protocol (XMPP)," 2013. [Online]. Available: http://tools.ietf.org/html/draft-miller-xmpp-e2e-06. [Accessed 22 July 2013].

[30] P. Waher, "XEP-0323: Internet of Things – Sensor Data," 2013. [Online]. Available: http://xmpp.org/extensions/xep-0323.html.

[31] P. Waher, "Internet of Things - Control," 2013. [Online]. Available: http://xmpp.org/extensions/xep-0325.html.

[32] P. Waher, "Internet of Things - Concentrators," 2013. [Online]. Available: http://xmpp.org/extensions/xep-0326.html.

[33] R. Klauck and M. Kirsche, "Chatty Things – Making the Internet of Things Readily Usable for the Masses with XMPP," 2012. [Online]. Available: https://www-rnks.informatik.tu-cottbus.de/content/unrestricted/staff/mk/Publications/.

[34] M. Krische and R. Klauck, "Unify to Bridge Gaps: Bringing XMPP into the Internet of Things," 2012. [Online]. Available: https://www-rnks.informatik.tu-cottbus.de/content/unrestricted/staff/mk/Publications/.

[35] "ISO/IEC/IEEE P21451-1-4 Standard for a Smart Transducer Interface for Sensors, Actuators, and Devices based on the eXtensible Messaging and Presence Protocol (XMPP) for Networked Device Communication," [Online]. Available: http://wiki.xmpp.org/web/Tech_pages/IoT_Sensei. [Accessed 22 July 2013].

[36] KTC, ”KTC tar klivet in i den digitala tidsåldern för fastighetsautomation,” 2013. [Online]. Available: http://www.ktc.se/2013/06/ktc-tar-klivet-in-i-den-digitala-tidsaldern-for-fastighetsautomation/. [Använd 22 July 2013].

[37] KTC, ”KTC förbinder sig till att skydda sina kunders och deras kunders data,” 2013. [Online]. Available: http://www.ktc.se/2013/07/ktc-forbinder-sig-till-att-skydda-sina-kunders-och-deras-kunders-data/. [Använd 22 July 2013].

[38] SUST, "Intelligent Energy Services," [Online]. Available: http://iea.sust.se/. [Accessed 22 July 2013].

[39] SUST, ”Kommunikationsstandarder för energieffektivisering i Almedalen,” 2013. [Online]. Available: http://www.ktc.se/2013/06/kommunikationsstandarder-for-energieffektivisering-i-almedalen/. [Använd 22 July 2013].

[40] P. Waher, "Proto-XEP: Internet of Things – Interoperability," 2013. [Online]. Available: http://htmlpreview.github.io/?https://github.com/joachimlindborg/XMPP-IoT/blob/master/xep-0000-IoT-Interoperability.html. [Accessed 22 July 2013].

[41] "XMPP research paper repository at Mendeley," [Online]. Available: http://www.mendeley.com/groups/3516891/xmpp/.

[42] wiki.xmpp.org, "Tech pages/IoT systems," [Online]. Available: http://wiki.xmpp.org/web/Tech_pages/IoT_systems. [Accessed 22 July 2013].

[43] P. Waher, "XEP-0332: HTTP over XMPP transport," 2013. [Online]. Available: http://xmpp.org/extensions/xep-0332.html.

[44] T. Muldowney, M. Miller, R. Eatmon and P. Saint-Andre, "XEP-0096: SI FileTransfer," 2004. [Online]. Available: http://xmpp.org/extensions/xep-0096.html.

[45] J. Karneges and P. Saint-Andre, "XEP-0047: In-Band Bytestreams," 2012. [Online]. Available: http://xmpp.org/extensions/xep-0047.html.

[46] S. Ludwig, J. Beda, P. Saint-Andre, R. McQueen, S. Egan and J. Hildebrand, "XEP-0166: Jingle," 2009. [Online]. Available: http://xmpp.org/extensions/xep-0166.html.

[47] "Clayster platform," [Online]. Available: http://clayster.com/.

[48] "The mono project," [Online]. Available: http://www.mono-project.com/.

[49] Wikipedia, "10-foot user interface," [Online]. Available:

http://en.wikipedia.org/wiki/10-foot_user_interface. [Accessed 22 July 2013].

[50] ”KTC,” [Online]. Available: http://www.ktc.se/.

[51] "Manodo," [Online]. Available: http://www.manodo.se/.

[52] Wikipedia, "Semantic Web & Web 3.0," [Online]. Available: http://en.wikipedia.org/wiki/Semantic_Web#Web_3.0. [Accessed 22 July 2013].