IoT Harmonization using XMPP

How XMPP can be used to interconnect isolated protocol islands

Peter Waher
Little Sister
Stockholm, Sweden
peter@littlesister.se

***Abstract*—The world of Internet of Things (IoT) is comprised of a multitude of incompatible islands, separated by different protocols and communication patterns. In this paper, we describe how the eXtensible Messaging and Presence Protocol (XMPP) can be used to bridge these seemingly incompatible islands in real-time, to harmonize the Internet of Things, regardless of underlying technology. XMPP is standardized by the Internet Engineering Task Force (IETF), and provides an open and free alternative to commercial or bespoke middleware platforms.**

***Index Terms*—IoT, Harmonization, Protocols, XMPP**

I. Introduction

The Internet of Things has been dominated by commercial and bespoke enterprises, creating an avalanche of new technologies, communication protocols and standardization efforts. Most of these efforts have also been dominated by large companies willing to invest in new technology. The backside of this *ad hoc* development, is that different choices result in incompatible devices. Furthermore, companies want to protect their interests and control the market, resulting in walled gardens, where efforts to promote their own technologies overshadow concerns for interoperability between existing technologies. Such concerns are mainly raised from the public investing in the technology. This approach further blocks the potential of *Smart Cities*, were the interoperability between services and things is a key enabler.

This paper presents an approach to bridge existing technologies, creating a global interoperable network of things, apt for *Smart Cities*. It does so without limiting the use of existing or future communication technologies, or requiring proprietary middleware. Instead of trying to devise a new standard for communication between things, a new standard for interoperability between a plurality of communication protocols is proposed. This standard can then be implemented in any middleware or gateway, to achieve the capability to interoperate across technology boundaries on the Internet. Furthermore, this standard must be based on open, free, scalable and proven technologies, suitable for the purpose, and without limiting the possibilities of the underlying technologies used.

II. Layers

Bridging different IoT solutions must be done on multiple levels. It is not sufficient to simply map semantics from one system to another, or from one protocol to another. Bridging must be done first on a transport level, then on a semantical level.

The transport level includes network topology concerns: Who can connect to whom? How are messages distributed? We will denote this *Communication Patterns*. Different protocols support different patterns for communication.

Semantics starts where communication patterns end. It relates to what operations can be performed, and how *interoperability* between entities in the network is achieved.

Overarching the Internet of Things, hangs the veritable Sword of Democles: *Security* and *Privacy*. Without taking these attributes into account, bridging is not complete.

III. Network Topology

Different protocols impose different restrictions on network topology. Clients connect to servers, require servers to be accessible from the clients. Servers are in the general case not able to connect to clients, for instance, due to firewall constraints. XMPP allows any client in the global federated network to reach any other client, regardless of firewalls. From a topology point of view, XMPP therefore allows bridging between entities in networks that require connectivity to the Internet, i.e. the Internet of Things, regardless of use of firewalls.

IV. Communication Patterns

To be able to bridge across the plurality of protocols that exist today, you need to build on a technology that support the varying communication patterns that are in use. There are mainly four patterns in common use:

a) *Asynchronous messages*, sent from one entity to another.
b) *Request/Response*, where one entity requests information from another, who responds once for each request.
c) *Publish/Subscribe*, where publishers publish information to a broker, which persists it, and later distributes it to a negotiated set of subscribers.
d) *Multicasting*, like *Publish/Subscribe*, but without data persistence.
e) *Queues*, to who Data providers push data when available, and where Consumers pull data, when they are ready.

XMPP has native support for (a)-(c), through the message, iq and presence *stanzas*, defined by the IETF [1]. The name for packets being transmitted in the XMPP network, is *stanza*. The presence stanza support persistence of the last content published. A second Publish/Subscribe method with more persistence options is available in XEP-0060 [2]. Multicasting is available through XEP-0045 [3]. Queues have no standardized extension yet, but is straight-forward to implement. For these reasons, XMPP allows simple bridging of information using any of the well-known and well-used communication patterns available. The step to make all patterns standardized is very small.

V. Interoperability

For the transfer of IoT-related content, interoperability interfaces exist, such as for sensor data [4], control operations [5] and concentrators, integration of subsystems and bridging between protocols [6]. These have been retracted from the XMPP Standards Foundation (XSF) by their author for the purpose of moving and managing them and their features within the IEEE IoT Harmonization working group [7]. Mapping of HTTP, including semantic web technologies, over XMPP, is also possible [8].

The interoperability interfaces defined for IoT in XMPP are all *loosely coupled*. They also contain sufficient meta-data to be able to encapsulate all information required, including localized information, by both devices and humans. As such, it is very easy to map existing data models in other protocols to the XMPP interfaces. It is easier to map messages in a *tightly coupled* architecture to messages in a *loosely coupled*, than vice versa.

VI. Security

XMPP has great support for security. Apart from performing *authentication* using SASL [9], it has a built-in *authorization* mechanism built into it, in the form of presence subscription negotiation [10], blocking of users [11] and spam reporting [12]. It furthermore has support for fine-grained *provisioning* within the realm of Internet of Things [13]. It also allows for management of *ownerships* and transfers of *ownerships*, to make sure things know who their owners are [14]. Apart from transport encryption being built into XMPP, XMPP also support interoperable interfaces for *end-to-end encryption* using OTR [15], OpenPGP [16] and OLM [17].

VII. Privacy

The interoperability interfaces do not require central storage of data related to things. Since such data might be related to physical individuals, it should be considered personal information, and must be treated as such. XMPP provides a means to interchange such data, without central storage, providing a means to protect the privacy of any data subjects concerned.

VIII. Scalability

XMPP is by its design *federated*. Brokers authenticate users on their domain, and then cooperate to interchange stanzas between domains. XMPP always forwards the identities of senders of stanzas, making authorization in distributed environments easy. Brokers validate the identities of each other to avoid the insertion of malicious brokers [20].

The federated nature of XMPP, and its extensive usage today within mobile, chat and social interaction with billions of users, makes it a good candidate as an IoT architecture and infrastructure.

IX. Openness

XMPP is standardized by the *Internet Engineering Task Force*, and XMPP extensions by the *XMPP Standards Foundation*, a non-profit organization with a free membership. The technology is free to use, and driven by a large community in individuals, where companies and large enterprises have limited input. This is an interesting alternative to the many commercial alternatives available.

X. Conclusion

It is possible to bridge, with relatively little effort, IoT-technologies based on a variety of technologies, such as Web of Things (HTTP, CoAP), LWM2M, OneM2M, UPnP, etc., and traditional or proprietary M2M technologies such as those based on MQTT, AMQP or a myriad of other protocols, using XMPP, without imposing limitations on the underlying technologies, and without requiring changes be made to the underlying devices. As long as mapping is possible, reverse bridging is also possible, so that devices talking one protocol can communicate with devices somewhere else using another protocol, bridged seamlessly by XMPP in between. Since XMPP IoT uses a loosely coupled architecture, it can describe more information than a tightly coupled architecture can.

References

[1] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Core”, RFC 6120, “Instant Messaging and Presence”, RFC 6121, “Address Format”, RFC 6122, Internet Engineering Task Force (IETF).

[2] P. Millard, P. Saint-Andre, R. Meijer, “XEP-0060: Publish-Subscribe”, XMPP Standards Foundation (XSF).

[3] P. Saint-Andre, “XEP-0045: Multi-User Chat”, XMPP Standards Foundation (XSF).

[4] P. Waher, “XEP-0323: Internet of Things – Sensor Data”, XMPP Standards Foundation (XSF).

[5] P. Waher, “XEP-0325: Internet of Things – Control”, XMPP Standards Foundation (XSF).

[6] P. Waher, “XEP-0326: Internet of Things – Concentrators”, XMPP Standards Foundation (XSF).

[7] IEEE Project “P1451-99 - Standard for Harmonization of Internet of Things (IoT) Devices and Systems”, hosted by the DASH - Devices and Systems Harmonization Working Group.

[8] P. Waher, “XEP-0332: HTTP over XMPP transport”, XMPP Standards Foundation (XSF).

[9] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Core”, RFC 6120, section 6, “SASL Negotiation”, p 77 ff.

[10] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence”, RFC 6121.

[11] P. Saint-Andre, “XEP-0191: Blocking Command”, XMPP Standards Foundation (XSF).

[12] S. Whited, “XEP-0377: Spam Reporting”, XMPP Standards Foundation (XSF).

[13] P. Waher, “XEP-0324: Internet of Things – Provisioning”, XMPP Standards Foundation (XSF).

[14] P. Waher, R. Klauck, “XEP-0347: Internet of Things – Discovery”, XMPP Standards Foundation (XSF).

[15] S. Whited, “XEP-0364: Current Off-the-Record Messaging Usage”, XMPP Standards Foundation (XSF).

[16] T. Muldowney, “XEP-0027: Current Jabber OpenPGP Usage”, XMPP Standards Foundation (XSF).

[17] A. Straub, “XEP-0384: OMEMO Encryption”, XMPP Standards Foundation (XSF).

[18] J. Miller, P. Saint-Andre, P. Hancke, “XEP-0220: Server Dialback”, XMPP Standards Foundation (XSF).