BIOTIC
Binary Internet of Things Interoperable Communication
Executive Summary

Peter Waher

Waher Data AB, Kajutvägen 26, 13955 Värmdö, Sweden
peterwaher@hotmail.com

Abstract. This paper proposes a basis for a new protocol for the Internet of Things – the Binary Internet of Things Interoperable Communication protocol – that will combine the pros of available protocols for the Internet of Things while avoiding many of the cons. It explains the reasons for using it and the design principles behind it. It also compares it to existing protocols for the IoT.

Keywords: Internet of Things, Peer-to-peer networks, Protocols, XMPP, MQTT, HTTP, CoAP, BIOTIC.

1 Introduction

There are many different protocols and communication patterns currently used for the Internet of Things. Until now none has been able to provide a solution for all use cases. Some are preferred under certain conditions, others preferred for different scenarios, and still others promoted as part of a company policy.

This paper provides a short overview of the different technologies available for the Internet of Things, lists their corresponding pros and cons, and then proposes a novel protocol for the Internet of Things that combines the pros of all previous protocols while minimizing the cons. A more detailed and extensive discussion about the different protocols, patterns, as well as their pros and cons can be found in [1] and [2].

2 Comparison of existing technologies

Table 1 is a simple break-down of how certain features important to the Internet of Things are supported in four popular IoT protocols: HTTP, CoAP [3], MQTT [4] and XMPP [5]. Browser APIs and associated protocols are not discussed here since they are only applicable for web clients. Peer-to-peer solutions will be discussed later in the paper, and not included in the first comparison that follows.

In Table 1, if a feature is supported, it is marked by a green ✔. If a feature is only supported by that protocol, it is marked with a double green ✔✔ with a greyed background. If a feature is supported under certain conditions, it is marked by a green

checkmark within parenthesis (✔). If the feature is not supported by the protocol or associated extensions, it is marked by a red ✘. This does not mean it cannot be implemented using the corresponding protocol, it simply means that any such implementations must be done in a proprietary fashion on-top of the corresponding protocol, which would create solutions that are more difficult to make interoperable between manufacturers.

Table 1. Supported features for different protocols

Feature	HTTP	CoAP	MQTT	XMPP
Request/Response	✔	✔	✘	✔
Publish/Subscribe	✘	✘	✔	✔
Multicast	✘	✔	✘	✔
Events or Push	✔	✔	✔	✔
Bypasses firewall	✘	(✔)	✔	✔
Federation	✘	✘	✘	✔✔
Authentication	✔	✔	✔	✔
Network Identity	(✔)	(✔)	✘	✔
Authorization	✘	✘	✘	✔✔
Encryption	✔	✔	✔	✔
End-to-end encryption	✘	✘	✘	✔✔
Compression	✔	✘	✘	✔
Streaming	✔	✘	✔	✔
Reliable messaging	✘	✘	✔✔	✘
Message Queues	✘	✘	✘	✘

- Request/response is the capacity of a client to request something from a device and have a response returned (momentary or somewhat delayed).
- Publish/Subscribe allows publishers to publish items to a message broker, which in turn forwards the item to registered subscribers.
- Multicast is the ability of multiple parties to communicate together, i.e. messages are broadcast to all members of certain groups.
- Events or push notification is the capacity of a device to send data to an interested party without a previous request for each delivery.
- Bypass Firewall means that messages can be sent and received even if the sender and final receiver lie behind different firewalls.
- Federation means that different islands can be interconnected using federation of message brokers, and that messages can be forwarded between brokers enabling devices in one island to communicate with devices in the other.
- Authentication refers to user authentication, i.e. validation of user credentials in networks.
- Network Identity means if participants in the communication are aware of each-others authenticated Network Identity when they communicate.

- Authorization refers to a security mechanism where messages are only allowed to be sent between approved parties (i.e. friendships).
- Encryption means if encrypted communication is possible.
- End-to-end encryption refers to encrypting the communication in such a way that not even message brokers can eavesdrop on messages being transmitted.
- Compression is the capacity to compress transmitted data to reduce the number of bytes transmitted over the network.
- Streaming is the ability to send unending streams of data bytes over a channel.
- Reliable messaging means that the protocol has a mechanism to assure the delivery of messages and that the sender has the option to be informed of the successful receipt of the messages.
- Message Queues are popular in back-end protocols, but are generally not supported in the available Internet-of-Things protocols. It means the ability for multiple publishers to queue items in a queue. These items are then consumed on a first-in-first-out basis by multiple consumers (or workers). Items are sent using reliable messaging, to make sure each item is processed exactly once.

From the table above it is clear that among the mentioned protocols, the protocol that has the largest support for features important in Internet of Things applications is XMPP.

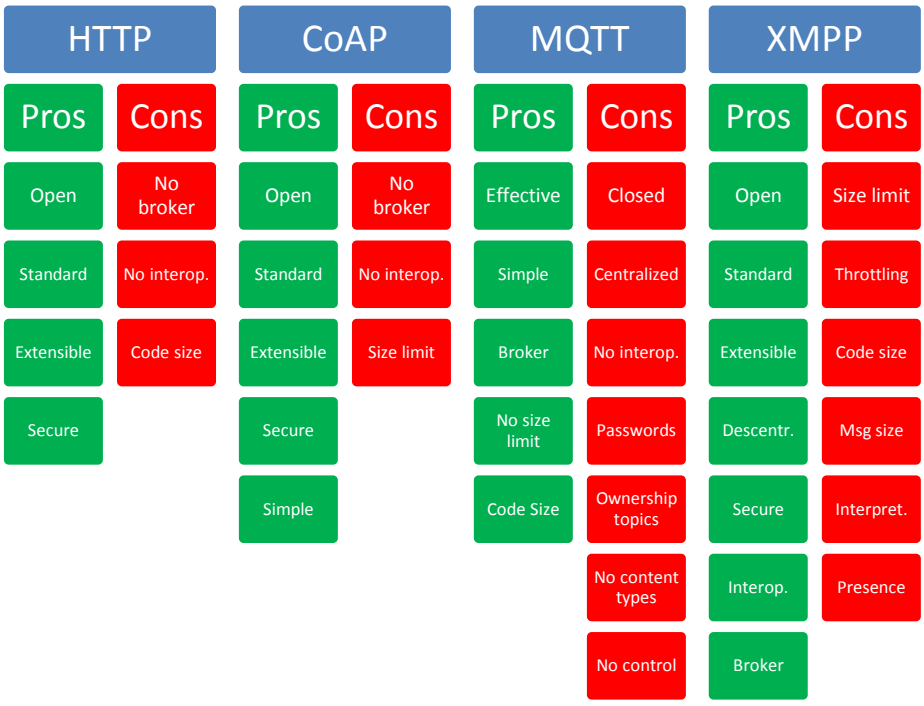

Fig. 1. Pros and cons for different protocols

But XMPP is not the obvious choice for many working with IoT. Why? Figure 1 provides a more subjective view of the existing protocols, comparing their perceived pros and cons. Even in this view, it is clear that XMPP has many more pros than the other solutions. The answer must, therefore, be found in the list of cons.

There's a size limit for sent messages on the XMPP network. This con is shared with the CoAP solution (UDP version). It limits the amount of data you can send inline and is server specific. This might affect Things that send multimedia (like cameras, microphones, etc.) but not sensors. It is therefore unlikely a reason to avoid XMPP for IoT in the general case.

Another con is the throttling of traffic, where the rate of transmission from a specific client is throttled as a means to balance the load on the broker between clients. This feature can be disabled and might, in some cases, limit the choice of public servers to use in some cases. Normally, this feature can be configured away. This con is also unlikely to be a reason for not selecting XMPP for use in IoT.

Another perceived con of XMPP is the code size of the implementation. It is felt to be too large for embedded devices. This is a con partly shared by HTTP. It is largely an effect of the many features supported by available clients, and the use of large XML DOM parser, perhaps suitable for larger applications. By reducing the clients to support only extensions that are required, and by using a more efficient parsing of the XML, the code size can be made small enough for embedded devices, as is shown in [6] and [7]. And in cases where SSL/TLS is only required to authenticate users and not protect the actual data, XMPP implementations can actually be made much smaller than corresponding MQTT implementations, since the SSL/TLS stack takes more memory than the underlying XMPP stack. Since secure user authentication in MQTT requires SSL/TLS at all times, while in XMPP it doesn't since XMPP uses SASL, small XMPP implementations not requiring encryption of data sent after user authentication can save a lot of memory by avoiding the use of SSL/TLS altogether, thus becoming smaller than their MQTT counterparts.

Another perceived problem is that XML messages become larger than necessary, consuming bandwidth unnecessarily. This problem is mitigated by using EXI compression creating very efficient binary messages, as defined in [8], but a generic EXI encoder/decoder still occupies memory. Very efficient EXI encoders and decoders can be created however, if used XML schemas are known at compile time, which is the case by embedded clients.

The problem of parsing incoming streams, or particularly knowing when one command ends and the next starts, is another perceived con resulting from the use of XML fragments in communication. There are no binary headers used in XMPP, which can make it trickier than necessary to implement an XMPP client library that correctly delimits messages. One obstacle for the use of XMPP in battery powered devices is the requirement for a live connection to the server is incoming messages and requests are to be allowed. This has often led device manufacturers to reverse the direction of communication when battery powered devices are used, such as the publish/subscribe pattern using MQTT or posting information regularly using HTTP or CoAP. Instead of reacting to incoming messages, they publish information regularly and sleep the rest of the time. The last con of normal XMPP for use in wireless appli-

cations is the use of presence messages. This creates a lot of messages – a problem in low bandwidth radio networks. XMPP servers disabling presence notification have been used to solve this problem [6] [7].

Faced with these XMPP cons, a device developer might be tempted to choose any of the other solutions, especially given that they are very simple to get started with, even though the feature set supported for that solution is not the same as for XMPP which makes it easier to get cornered in the future. If the protocol provides a solution for the most immediate need, it is seen as sufficient. The rest is perceived can be added later (albeit in a proprietary fashion that is difficult to make interoperable). In the same way, companies seldom consider interoperability with other companies (especially competitors). The result is that any technology that achieves the most immediate goal is often seen as the option of choice, even if it be proprietary and not interoperable.

3 Proposition – a new IoT protocol: BIOTIC

Rather than arguing for or against a specific technology, this paper makes the following proposition: It is possible to create a new protocol for the Internet of Things that provides the benefits of the binary protocols such as CoAP and MQTT when it comes to code size and simplicity of implementation and interpretation, but that retains the flexibility, extensibility, security, federated and decentralized architecture of the XMPP protocol, not to forget the extensive set of featured patterns listed above.

One of the perceived reasons for MQTT and CoAP to be simpler and easier to implement is its lack of XML parser and the packaging of each message in a **binary** package containing package size. The new BIOTIC protocol, will package each message in a binary package containing a header with package size, which makes it easier to receive the message, since you know beforehand how large it will be. The binary header would also include address and other relevant information relating to message service and reliability.

For security reasons the BIOTIC protocol will maintain the use of **message brokers** that authenticates users, authorizes who can talk to whom and forwards packets accordingly. As in the case of MQTT, there should be no practical limit to message sizes and no throttling of messages for it to be useful in IoT applications. Using a message broker in this way also effectively bypasses firewalls without creating security issues by punching holes in firewalls and only permitting authorized users to communicate with each other. Allowing federation of message brokers allows for secure decentralized solutions.

The extensibility of XMPP is provided by the use of namespaces and different elements in XML. In BIOTIC this is replaced by efficiently encoded unsigned integers representing **organization**, **command set**, **command** and **version**. Each organization needs to register and when registered, can produce any number of command sets, commands and versions. There would be a core BIOTIC organization containing core command sets for core features. The command set would represent an extension or feature, similar to the concept of an XMPP extension [9]. Each command in a com-

mand set would represent a specific command corresponding to that command set. The last number would be a version number, making it possible to extend commands in time in a controlled manner, and on a command level.

The **core organization** would provide command sets that provide the essential generic features expected of an Internet of Things protocol such as request/response, multicast including moderation, publish/subscribe including ownership and subscription control, push notification, message queues, encryption (including end-to-end encryption), compression, streaming, etc. It would also include IoT-specific features such as encoding of sensor data (analogous to [10]), provisioning (analogous to [11]), control (analogous to [12]) and concentrators and bridges to other protocols (analogous to [13]). Interoperability interfaces (analogous to [14]) will be an important aspect in creating a truly interoperable infrastructure where services and devices from different manufacturers can work together. Extensions of the Semantic Web onto BIOTIC networks would be done in a similar fashion as described in [1] with an extension similar to [15]. The core organization could also include command sets for related uses, such as chatting (instant messaging), social networking, etc. It is these extensions that make BIOTIC interoperable, by defining features that different manufacturers can use to communicate with each other, without restricting the manufacturers from creating their own extensions.

4 Novel features

4.1 Multi-layered federation

One novel feature of the BIOTIC protocol is a multi-layered federation system where BIOTIC servers (or message brokers) can act as clients in their own regard to other servers, but still permit federation across domain even through one server can only connect to another. This would also be visible in the BIOTIC address space: Devices connected to one server would be able to communicate with each other using addresses such as `device1` and `device2`. These would be synonymous with `device1@domain` and `device2@domain` if connected to a server on the same domain, but would require less space in transmitted packets. Devices connected to different servers could reach one another using addresses such as `device1@domain1` and `device2@domain2` as long as the domains `domain1` and `domain2` could be reachable from one another. Devices connected to private servers would still be able to communicate with each other by using public servers and multi-layered addresses such as `device1@privatedomain1@publicdomain1` and `device2@privatedomain2@publicdomain2`, as long as the servers `publicdomain1` and `publicdomain2` can reach each other. This can be repeated in any number of layers desired. Apart from having a very beneficent impact on architecture and privacy, it also has a very beneficent impact on scalability and manageability of devices and their connections as it can be distributed away from bottlenecks. The only thing needed on the public servers is an authorized connection between

`privatedomain1@publicdomain1` and `privatedomain2@publicdomain2`, and then the finer detail is left to the private domains.

Example: If each user (say a home) would have 50 devices, each controlled by a small gateway (a home gateway), and there would be 10,000 users (for instance apartments), instead of having 500,000 connections to a central broker (requiring perhaps a small data center), only 10,000 connections would be necessary. Now, if each building (containing on an average perhaps 25 apartments) controlled its own server or message broker, perhaps on its IP router, only 400 connections would be necessary for the central server, still allowing all 500,000 devices to talk to each other (if permitted). The address space in such a network would be addresses similar to `device@apartment@building@operator`. Of course, devices within the network of the same operator could be reached by using addresses like `device@apartment@building`, devices within the same building could be reached using addresses like `device@apartment`, and devices within the same apartment could be reached using addresses like `device`. Note that the 400 connections on the central server would only transport messages across buildings, which would probably count for only a small fraction of overall communication. In such a case, a small computer instead of a larger data center would probably suffice.

4.2 Battery powered devices

Since the BIOTIC protocol is aimed at the Internet of Things, support for battery powered devices is included. This means that all types of messages and requests support delayed delivery and intermittent connections at all levels of the multi-layered federated network. It will also report back to the original sender of a message when a message has been delivered (if requested) or if the message could not be delivered within the allotted time. Using this option, it will be possible to communicate with battery powered devices in the same way as normal devices, even though responses will be delayed until the battery powered device connects or a timeout occurs. This will enable battery powered devices to respond to requests when they can instead of constantly publishing information that might or might not be used (even if this latter option is still possible). The support for intermittent connections, even during operations that normally would require dedicated connections, both reduces the number of simultaneous connections necessary to support a network as well as the energy required for small devices to maintain live connections.

4.3 Peer-to-peer communication

Up until now we have not considered true peer-to-peer solutions. Even though XMPP and MQTT act like peer-to-peer solutions on the application layer, they still constitute hybrid approaches, since they use message brokers, even though XMPP can run in server-less mode, see [16]. The use of message brokers is a positive feature for IoT, since it removes much of the complexity relating to security including user authentication and authorization from the Things themselves, which otherwise often lack good interfaces or resources. However, there are many cases where a true peer-

to-peer connection may be preferred, especially if one can be negotiated securely, for instance when performing streaming or if the brokers themselves are intermittent.

BIOTIC takes advantage of the secure infrastructure provided by the multi-layered federated message broker structure to negotiate (in a secure manner) reusable peer-to-peer connections between parties based on Distributed Hash Tables and Kademlia [17], if they are desired and supported. Each party would use the security framework provided by the BIOTIC message brokers to enable trust between peer-to-peer connections, transforming the BIOTIC protocol into a true structured peer-to-peer network as well. This peer-to-peer communication would be visible in the BIOTIC address space as well: `device#hash` would be the address of a device reachable over the BIOTIC peer-to-peer layer and `hash` would be safely negotiated beforehand by the underlying BIOTIC protocol, or by the corresponding message broker (for instance a private broker in the home) and be known to correspond to a certain trusted party. Once known, a peer-to-peer address can be used synonymously in all BIOTIC commands sent to the corresponding device seamlessly. There would be no need to build such peer-to-peer functionality into the different command sets, since it would be provided by the underlying BIOTIC protocol itself. If using a private broker, the devices themselves do not need to support peer-to-peer functionality, since the broker can do that for them.

Peer-to-peer communication as provided by the BIOTIC protocol, used in all communication between devices, places less stress on the structural components provided by the message brokers, reducing the need for powerful datacenters even for very large networks.

One final note however: Peer-to-peer functionality is an optional extension and devices or the corresponding network architecture are not required to support it. If this is the case, the peer-to-peer address negotiation will fail. In this case, messages can still be sent through the message brokers.

5 Conclusion

There are many protocols in use today for Internet of Things applications. Each protocol solves a specific set of problems, but none of the existing protocols solves all use cases. This may be acceptable for certain manufacturers, since they select technology that solves only their particular set of use cases, extending their choice with proprietary solutions. But an Internet of Things perspective takes interoperability and Internet security into account, making it preferable to have one protocol that resolves all use cases currently identified as important. Such a protocol is definitely possible to create, as outlined in this paper. The new protocol, the basis which is described here, is called the BIOTIC protocol – the Binary Internet of Things Interoperable Communication protocol.

You can read more about this protocol at `biotic-community.com` [18]. A "Biotic Community" is otherwise defined as "all the interacting organisms living together in a specific habitat" [19]. For the Internet of Things, an organism is a Thing, or Things become organisms in the future Internet.

6 Acknowledgements

Financial support for this study was provided by KTC [20], Manodo [21] and Weevio [22]. The author wishes to thank Dr. Karin Forsell, Jeff Freund and Tina Beckman for their suggestions and comments on this document.

7 References

[1] P. Waher, "Extending the Semantic Web to Peer-to-Peer-Like Sensor Networks Based on XMPP".

[2] P. Waher, "Bridging MQTT & XMPP Internet of Things networks," 2013.

[3] IETF, "Constrained RESTful Environments (core)," [Online]. Available: https://ietf.org/wg/core/. [Accessed 16 12 2013].

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

[5] xmpp.org, "XMPP Technology Overview," [Online]. Available: http://xmpp.org/about-xmpp/technology-overview/. [Accessed 29 11 2013].

[6] 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/CollaborateCom_2012-Klauck_Kirsche.pdf.

[7] 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/PerCom_2012-WiP-Kirsche_Klauck.pdf.

[8] P. Waher and Y. DOI, "XEP-0322: Efficient XML Interchange (EXI) Format," [Online]. Available: http://xmpp.org/extensions/xep-0322.html. [Accessed 29 11 2013].

[9] xmpp.org, "XMPP Extensions," [Online]. Available: http://xmpp.org/extensions/. [Accessed 29 11 2013].

[10] P. Waher, "XEP-0323: Internet of Things - Sensor Data," 16 04 2013. [Online]. Available: http://xmpp.org/extensions/xep-0323.html.

[11] P. Waher, "XEP-0324: Internet of Things - Provisioning," 16 04 2013. [Online]. Available: http://xmpp.org/extensions/xep-0324.html.

[12] P. Waher, "XEP-0325: Internet of Things - Control," 06 05 2013. [Online]. Available: http://xmpp.org/extensions/xep-0325.html.

[13] P. Waher, "XEP-0326: Internet of Things - Concentrators," 06 05 2013. [Online]. Available: http://xmpp.org/extensions/xep-0326.html.

[14] P. Waher, "XEP-xxxx: Internet of Things - Interoperability," [Online]. Available: http://htmlpreview.github.io/?https://github.com/joachimlindborg/XMPP-IoT/blob/master/xep-0000-IoT-Interoperability.html. [Accessed 29 11

2013].

[15] P. Waher, "XEP-0332: HTTP over XMPP transport," 11 07 2013. [Online]. Available: http://xmpp.org/extensions/xep-0332.html.

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

[17] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-peer Information System Based on the XOR Metric," [Online]. Available: http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf. [Accessed 19 12 2013].

[18] "The Binary Internet of Things Interoperable Communication Protocol - Community Pages," [Online]. Available: http://biotic-community.com/.

[19] Wikipedia, "Biotic," [Online]. Available: http://en.wikipedia.org/wiki/Biotic. [Accessed 19 12 2013].

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

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

[22] "Weevio AMR," [Online]. Available: http://weevio.com/.