Proceedings of the Federated Conference on Computer Science and Information Systems pp. 645–649
DOI: 10.15439/2018F232
ISSN 2300-5963 ACSIS, Vol. 15

Using Publish/Subscribe for Short-lived IoT Data

Frank T. Johnsen
Norwegian Defence Research Establishment (FFI)
P.O. Box 25
2027 Kjeller, Norway

***Abstract*—Efficient distribution of IoT sensor data requires one-to-many communication, for which publish/subscribe is a better communication approach than request/response. In this paper, the goal is to identify the/those publish/subscribe protocol(s) that are best suited for IoT data. The premise is that data should be as fresh as possible. Hence, the metric is end-to-end delay and the recommended approach is the solution that yields the lowest delay under the test conditions. Raspberry Pi 3 was used as the testbed, since it is representative as an IoT platform. The protocols evaluated are: AMQP, MQTT, MQTT-SN, STOMP, WSN, and XMPP, as well as using a mediation service to translate between them.**

I. Introduction

The term Internet of Tings (IoT) can be traced back as early as 1999 when Kevin Ashton used it to describe a network that linked physical "stuff" to the Internet. Nevertheless, it would be a few years before "IoT" became an active research area and the buzzword it is today. There are many different interpretations of what IoT is, but the core idea stems from Ashton. A more recent and elaborate definition of IoT is as follows:

> The Internet of Things (IoT) describes the revolution already under way that is seeing a growing number of internet enabled devices that can network and communicate with each other and with other web-enabled gadgets. IoT refers to a state where Things (e.g. objects, environments, vehicles and clothing) will have more and more information associated with them and may have the ability to sense, communicate, network and produce new information, becoming an integral part of the Internet. A widespread Internet of Things has the potential to transform how we live in our cities, how we move, how we develop sustainably, how we age, and more. – From [1]

The reason why IoT has become commonplace over the last five years is that a number of factors that can be considered mandatory precursors to this phenomenon have come into place:

- Cheap sensors (easily accessible from eBay, deal extreme, etc).
- Cloud computing (serves as a backend for IoT systems and can handle big data)
- Powerful smartphones (often used as a consumer's control panel in the IoT context)

Given these precursors, there have been many business ideas in the healthcare sector, logistics and other areas that give rise to a number of applications that fuel the current IoT trend. IoT as a concept is definitely relevant in a defense context. An example of this is the pioneer work described in [2], which deals with the use of sensor networks and lightweight processing platforms that require low power. IoT includes several disciplines, as one needs networking, embedded hardware, software architectures, sensors, information management, data analysis and visualization to fully leverage the concept. A key component within IoT is the use of distributed online devices that communicate using Internet protocols. A "thing" in IoT may be any device that is able to communicate, gather data or offer some kind of control. With this wide interpretation of "things", IoT may include, but is not limited to: Vehicles, appliances, medical equipment, power grids, transport infrastructure and production equipment.

Military organizations can exploit IoT deployed in battlefields and operational theaters to improve situational awareness, mission performance and achieve information superiority [3]. Within NATO, the Research Task Group (RTG) IST-147 "Military Application of Internet of Things" is investigating how to best employ IoT in a coalition force, particularly in the context of augmenting situational awareness in military operations in smart cities [4].

Today there is a great focus on using Commercial off-the-shelf (COTS) products where possible because it is considered a cost-effective way of acquiring a capability. This idea is well rooted in NATO, and has been considered foundational for an effective Network Enabled Capability (NEC) as identified in the NATO NEC Feasibility Study [5]. In this study it was also pointed out that the principles of service orientation must be taken into account when building distributed systems. These observations can be continued within the IoT venture, as there will also be a need to build large, efficient and interoperable systems.

NATO has identified a set of *Core Services*, which provide common communication functionality that other services (e.g., C2 services) depend upon. An example of a Core Service is messaging, which includes both request/response and publish/subscribe services. In this paper, the focus is publish/subscribe services as applied to support IoT. Here, the focus is on short-lived IoT data, in other words data that comes fresh from a sensor and needs to be delivered as soon as possible. Publish/subscribe is considered the most efficient communication paradigm for this type of data, in contrast to long-lived data, where the new approach of Information-Centric Networking offers some desirable properties [6].

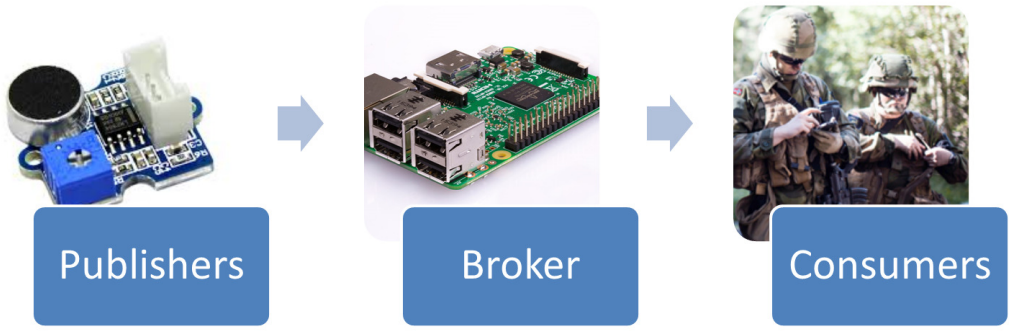

Fig. 1. Publish/subscribe information flow. It is assumed that subscriptions have been set up prior to this occurring.

II. Publish subscribe protocols

The publish/subscribe pattern implies that a consumer explicitly signals its interest in a given type of data by registering a subscription. The most common approach to signal such an interest is through a *topic*, i.e., a string that is used to identify the data a consumer is after. When new data is available on a certain topic, all consumers that have expressed interest in that topic receive it. This pattern is particularly well suited for IoT, as many sensors produce information more or less periodically, and thus, a push-pattern can reduce network activity considerably when compared to using a request/response or pull-pattern. A *broker* is used between the producer and consumer. Its tasks include subscription management and message dissemination according to topics, so that the producer only has to send new data to the broker, which then handles all further dissemination. For an illustration of the publish/subscribe pattern, see Fig. 1.

A number of publish/subscribe standards exist. This paper considers several of the most commonly available standards today. A short overview of the protocols follows.

A. Message Queue Telemetry Transport (MQTT)

MQTT is a popular publish/subscribe protocol for IoT, standardized as ISO/IEC PRF 20922 [7]. It provides publish/-subscribe messaging for resource-constrained devices: Low processing power, low memory, as well as network constraints. MQTT is designed to function well over unreliable networks by providing three levels of Quality of Service (QoS): Level 0, “at most once”-semantics – messages are delivered on a “best effort” basis. As MQTT is based on TCP, this is usually enough to ensure delivery. However, if the TCP connection is broken there will be no retransmission later on reconnection with this QoS level. So, though not likely, message loss can occur. Level 1 provides “at least once”-semantics, where messages are assured to arrive but duplicates can occur, hence systems must be able to handle duplicate packets. Level 2 gives “exactly once”-semantics, so messages are assured to arrive exactly once. This latter method requires an exchange of four packets, and decreases performance of the broker.

B. MQTT for Sensor Networks (MQTT-SN)

MQTT-SN is, in short, a version of MQTT that is optimized specifically for sensor networks [8]. The major difference is that it uses UDP as the underlying transport protocol rather than TCP.

C. Advanced Message Queuing Protocol (AMQP)

AMQP [9] is a binary wire protocol, which was designed as a reliable and interoperable open replacement for existing proprietary messaging middleware. As the name implies, it provides a wide range of features related to messaging, including reliable queuing, topic-based publish-and-subscribe messaging, flexible routing, transactions, and security. AMQP has been shown to be scalable and reliable, and is much used for civilian applications, notably for supporting financial transactions and also as a backbone in cloud computing clusters.

D. Web Services Notification (WSN)

NATO has chosen WSN [10] for publish/subscribe in its SOA baseline [11]. WSN is a part of the family of SOAP Web services standards. SOAP services promote interoperability, but being based on XML the cost is increased overhead compared to other protocols.

E. Simple/Streaming Text Oriented Messaging Protocol (STOMP)

STOMP [12] is text-based, making it somewhat similar to how HTTP operates. The main design principle was to create something simple to use and understand. However, the STOMP flavor of topics (called a *destination* in STOMP) is not mandated in the protocol specification, meaning that different brokers may support it differently. This, in turn, lowers interoperability across brokers, since publishers and consumers that function well with one broker implementation may not work with another. If you don’t encounter portability issues, then STOMP is simple, lightweight, and offers a wide range of language bindings.

Fig. 2. The Raspberry Pi 3B single board computer.

F. Extensible Messaging and Presence Protocol (XMPP)

Just like WSN, XMPP [13] is an XML-based protocol. Unlike WSN, XMPP does not use SOAP, so it does away with the extra abstraction layer (and thus extra overhead). XMPP is most known as a chat (instant messaging) and presence protocol, however it does offer additional features as well, like SIP-compatible multimedia signaling for voice, video, file transfer, and other applications as well as publish/subscribe functionality. XMPP aims to be the main competitor to MQTT for civilian IoT applications, and is, as such, interesting to compare with MQTT to see which protocol is "best".

III. Testbed setup

The testbed was put together of two Raspberry Pi 3B single board computers. The main motivation for using this as the testbed was that the Raspberry Pi 3B is a somewhat capable yet cheap computer that is representative for IoT development. The board is shown in Fig. 2. The technical specifications of the board are as follows [14]:

- Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board Computer running at 1.2GHz
- 1GB RAM
- BCM43143 WiFi
- Bluetooth Low Energy
- 40pin extended GPIO
- 4x USB 2 ports
- 4 pole Stereo output and Composite video port
- Full size HDMI
- CSI camera port for connecting the Raspberry Pi camera
- DSI display port for connecting the Raspberry Pi touch screen display
- Micro SD port for loading your operating system and storing data

One Raspberry Pi 3B functioned as the client: That is, it set up subscriptions to pre-determined topics up front, published messages to said topics, and ran the consumers that received the messages. This was done so that time measurements across publishers and consumers (via the broker) would be accurate, since timestamps would originate from one and the same node rather than several, where clock skew could become an issue. The second Raspberry Pi 3B offered the protocol brokers and the mediation service to translate between protocols. All publishers and consumers were implemented using Java, and the respective protocols' native Java libraries. The brokers and mediation service (let us call this component a *multi-protocol broker*) was also Java software. In fact, the multi-protocol broker was an extended version of the federation mechanism described here [18], enhanced for the purpose of this paper to support all the protocols discussed above. For networking, 100 Mbps Ethernet was used, and both Raspberry Pi 3B's were connected to a switch. This was done to ensure the best possible networking conditions during the tests, so that local disturbances and interference should not affect the results, which could have been an issue if using e.g., WiFi.

Tests were executed as follows:

1) The multi-protocol broker was started.
2) A subscription to a topic was set up for a particular protocol α.
3) A publisher was initiated to fire off a burst of 100 messages over protocol β.
4) Having received 100 messages, the consumer terminated its subscription to protocol α.
5) The duration of steps 3-4 above was measured.
6) Steps 1-5 above were repeated for all α, β of protocol permutations.

IV. Results

Tab I shows the results when transmitting 100 messages. This table shows when the publisher sends 100 consecutive messages via the multi-protocol broker. The consumer receives the messages, and terminates the subscription after message number 100. The time (in seconds) of this entire burst of messages was measured here.

We see that WSN is the overall loser when considering our protocol delay metric. Consistently WSN shows the highest delays here. We see that having WSN as either the publisher or subscriber results in a high delay, with an even higher delay exhibited (just over 20s – the highest in the test) when both publisher and subscriber used WSN. This can be attributed to the SOAP layer used in the protocol; having this extra abstraction layer on top of HTTP does have an impact performance wise.

Of the other protocols, the performance difference is not so large when considering publisher/subscriber pairs of the same protocol (no translation is involved – indicated in **bold** in the table). Here, we see that AMQP is the "worst" (just above 4.5s) and STOMP is the "best" (just above 2.5s). This is understandable when thinking about the fact that AMQP provides a reliable message queue. Messages are ensured delivery and acknowledged in the queue. STOMP does not perform this added value service. MQTT achieves slightly better results than MQTT-SN, which again is slightly more efficient than XMPP (just over 3s). At first glance this may seem strange, since MQTT-SN is based on UDP which inherently has lower overhead than TCP, which is the underlying transport in

TABLE I
PROTOCOL DELAY PUBLISHING AND RECEIVING 100 MESSAGES.

	AMQP Sub.	**MQTT Sub.**	**MQTT-SN Sub.**	**STOMP Sub.**	**WSN Sub.**	**XMPP Sub.**
AMQP Publisher	**4.577313**	5.054525	5.065977	3.529224	16.309429	4.443499
MQTT Publisher	3.022697	**2.655891**	2.303168	2.448346	14.992253	4.470155
MQTT-SN Publisher	3.438891	2.716955	**2.895711**	2.466580	14.462532	4.340074
STOMP Publisher	3.017157	3.215914	4.876303	**2.256608**	15.458013	4.347493
WSN Publisher	12.518343	11.745184	12.116381	11.534597	**20.769935**	13.644167
XMPP Publisher	7.165085	6.414229	6.492992	6.086224	16.023309	**3.001673**

MQTT. The reason why MQTT-SN has slightly higher delays here, is that it is actually implemented as a gateway that uses plain MQTT in the backend. Hence, MQTT-SN gets a slight performance impact going through this gateway which moves it from UDP to TCP and vice versa locally on the broker node (UDP is used between client and broker across the network).

The remaining rows in the table show the different protocols' delay where the impact of translating between them is also included. We see that XMPP and WSN here show the cost of translating to/from XML based protocols. WSN has the impact of using both XML and SOAP, whereas XMPP only uses XML. An interesting observation is the above mentioned performance of the XMPP publisher/subscriber pair, which shows that the XML-based XMPP does not perform too badly when no translation is involved.

V. RELATED WORK

The NATO IST-090 RTG has demonstrated the use of WSN at the tactical level. WSN has the benefit of being a NATO recommended standard for information exchange in a coalition environment. However, it is a resource heavy protocol and its application at the tactical level requires applying proprietary optimizations [15].

More recently, the IST-118 RTG conducted initial experiments comparing different publish/subscribe approaches on tactical broadband radios. Namely, WSN, MQTT and AMQP were investigated in a preliminary small-scale study [16]. Here, MQTT was found to be a very lightweight alternative to the other two protocols when applied in the tactical network. Currently, the IST-150 RTG is continuing work where IST-118 left off, and is considering MQTT specifically for use in soldier systems on the tactical level [17].

In [18], the authors provided a solution for federating between different publish/subscribe protocols, i.e., WSN, MQTT, and AMQP. The work in this paper uses an extended version of that open source implementation with support for additional protocols as the broker and mediation service (aka multi-protocol broker) in the testbed.

VI. CONCLUSION

If you need to use UDP from your sensor to the mediation service, then your choice is MQTT-SN, which is the only one based on UDP of the protocols tested.

If you need interoperability with NATO (i.e., you need WSN subscribers), then the most efficient protocols to use when going through the mediation service are MQTT-SN (UDP) and MQTT (TCP). You definitely don't want your sensors to deliver data using WSN directly, as that is the protocol with the overall highest delay.

If you are free to choose any protocol you want for the entire network, then using STOMP as both publisher and receiver was marginally quicker than using any of the other protocols. However, if you need advanced capabilities like multi broker meshing, then MQTT or AMQP can offer that, though they had slightly larger delays than STOMP. Though never best, XMPP shows overall favorable results given that it is based on XML. It goes to show that it is possible to implement a somewhat efficient XML based protocol, in comparison with WSN, which has very high delays. The reason for this is that WSN, being based on SOAP, adds an extra abstraction layer in the protocol that none of the other protocols have.

The overall recommendations are summarized in Tab. II.

TABLE II
RECOMMENDATIONS FOR PUBLISH/SUBSCRIBE COMBINATIONS.

GOAL	**Publisher**	**Subscriber**
Lowest overall delay	STOMP	STOMP
UDP necessary	MQTT-SN	MQTT-SN
NATO Interoperable	MQTT or MQTT-SN	WSN
Meshable brokers	MQTT or AMQP	MQTT or AMQP

The table gives an overview of which protocol combinations to use to achieve a specific goal. Note that where the lowest delay is not the primary goal, it is considered the secondary goal when giving the recommendations.

VII. FUTURE WORK

The testbed consisted of a publisher and subscriber running on one Raspberry Pi and the broker on another. Hence, it is only a small scale test of how the protocols perform. One of the challenges of IoT is the scale, so for future work it would be interesting to make similar tests of a larger scale setup.

Also, it would be beneficial to test the protocol performance over a typical IoT networking technology, such as LoRa. Other relevant networking options would be 4G and WiFi, to name a couple.

REFERENCES

[1] IoT Special Interest Group. Technology Strategy Board. 2013.

[2] Wind River Systems. The Internet Of Things For Defense. White Paper, 2015.
[3] Niranjan Suri et al. Analyzing the Applicability of Internet of Things to the Battlefield Environment. IEEE ICMCIS 2016, Brussels, Belgium, May 2016.
[4] Frank T. Johnsen, Zbigniew Zielinski, Konrad Wrona, Niranjan Suri, Christoph Fuchs, Manas Pradhan, Janusz Furtak, Bogdan Vasilache, Vincenzo Pellegrini, Michal Dyk, Michal Marks, and Mateusz Krzyszton. Application of IoT in Military Operations in a Smart City. IEEE ICMCIS 2018, Warsaw, Poland, 22nd – 23rd May 2018.
[5] P. Bartolomasi, T. Buckman, A. Campbell, J. Grainger, J. Mahaffey, R. Marchand, O. Kruidhof, C. Shawcross, and K. Veum. NATO network enabled capability feasibility study. Version 2.0, October 2005.
[6] A. Carzaniga, M. Papalini, and A. Wolf. Content-based Publish/Subscribe Networking and Information-centric Networking. Proceedings of the ACM SIGCOMM workshop on Information-centric networking, ACM, 2011.
[7] ISO/IEC 20922:2016. Information technology – Message Queuing Telemetry Transport (MQTT) v3.1.1. ISO/IEC JTC 1 Information technology. Publication date: June-2016. https://www.iso.org/standard/69466.html
[8] Andy Stanford-Clark and Hong Linh Truong. MQTT For Sensor Networks (MQTT-SN) Protocol Specification. Version 1.2. November 14, 2013. http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf
[9] RabbitMQ. AMQP 0.9.1 protocol specification. https://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf
[10] OASIS. WSN specifications. https://www.oasis-open.org/committees/wsn/
[11] Consultation, Command and Control Board (C3B). CORE ENTERPRISE SERVICES STANDARDS RECOMMENDATIONS: THE SOA BASELINE PROFILE VERSION 1.7. Enclosure 1 to AC/322-N(2011)0205, NATO Unclassified releasable to EAPC/PFP, 11 November 2011.
[12] STOMP Protocol Specification, Version 1.2 http://stomp.github.io/stomp-specification-1.2.html
[13] XMPP is the open standard for messaging and presence. https://xmpp.org/
[14] Farnell.com. Raspberry Pi 3 Model B. http://www.farnell.com/datasheets/2020826.pdf
[15] IST-090. SOA Challenges for Real-Time and Disadvantaged Grids, Final Report of IST-090. AC/323(IST-090)TP/520. NATO. Published April 2014
[16] IST-118. IST-118 SOA recommendations for Disadvantaged Grids: Tactical SOA Profile, Metrics and the Demonstrator Development Spiral. Paper presented at the SCI-254 Symposium on “Architecture Assessment for NEC”. 14-15 May, 2013 in ESTONIA.
[17] Marco Manso, Frank T. Johnsen, Ketil Lund, and Kevin Chan. Using MQTT to Support Mobile Tactical Force Situational Awareness. IEEE ICMCIS 2018, Warsaw, Poland, 22nd – 23rd May 2018.
[18] Eirik Bertelsen et al. Federated publish/subscribe services. 9th IFIP International Conference on New Technologies, Mobility & Security 26 to 28 February 2018. Paris, France.