Evaluating Publish/Subscribe Protocols for use in Constrained Networks

Emil Paulin Andersen and Frank T. Johnsen

Norwegian Defence Research Establishment (FFI)
Instituttveien 20, Kjeller, Norway

Abstract. Considering the case of disaster relief and search and rescue operations, we can anticipate personnel operating with a total or partial lack of any pre-existing infrastructure. This means that ad hoc communication solutions must be established. Coupled with the abundant sensing capabilities provided through the innovation behind the Internet of Things (IoT), we need to identify suitable publish/subscribe protocols that can convey such data when facing various constrained networking conditions.
In this paper, we evaluate MQTT, MQTT for Sensor Networks (MQTT-SN), and ZeroMQ in several different relevant networking conditions. Based on the analysis of each protocol, we have concluded that they are suitable for use in less constrained networks but struggle with efficient communication in the more challenging ones. We found that MQTT shows good performance in the less limiting networks we have tested, only surpassed by MQTT-SN in some cases. However, due to tooling maturity and unsurpassed community support, we consider MQTT the "overall winner".

Keywords: Publish/Subscribe · Edge Computing · IoT

1 Introduction

Gartner defines the Internet of Things (IoT) as "a network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment" [1]. IoT has become an essential aspect of daily life, helping us automatically manage simple and complex tasks without much hassle. The quantity of data produced by these devices is increasing, and it is not always efficient nor possible to process this data on the system locally. Therefore, there has become a need to move the data responsibility to other systems with more computing power and better data processing. The data should be sent from one system to another in an efficient and fault-tolerant manner. This message exchange must happen across a network, which might not always be reliable. Operating in a controlled environment, for example a factory, communication resources are predictable. Conversely, out in an operation, performing search and rescue in distant areas, or providing disaster relief to an area affected by a natural disaster, communication is a different matter. In such operations, it is often a need for collaboration between military and

civilian entities, and communication equipment is often provided in an ad hoc manner. There is here a requirement for edge computing, where, "Edge computing is part of a distributed computing topology where information processing is located close to the edge, where things and people produce or consume that information.", according to Gartner [2]. In this paper, we have looked at several protocols within the publish/subscribe pattern [3] and evaluated how they performed in constrained networks. The protocols include MQTT [4], MQTT for Sensor Networks (MQTT-SN) [5] and ZeroMQ [6].

This paper is a summary of work that started in the autumn of 2021 with a preparatory project [7], and that concluded with a master's thesis delivered the summer of 2022 [8].

The paper is organized as follows: Section 2 presents related work. Section 3 presents the experiment design. Results are presented, analyzed and summarized in Section 4. Section 5 concludes the paper.

2 Related Work

In NATO, several Research Task Groups (RTGs) have investigated different technological aspects of search and rescue operations. An RTG titled "Military Applications of IoT" investigated incorporating IoT data into military systems, leveraging the MQTT protocol. The group culminated with a technical demonstration of the feasibility of such an approach [9].

MQTT has been identified as a promising publish/subscribe protocol for use in disaster relief, among other things in a prototype tested by the German Red Cross [10]. Further, various publish/subscribe protocols, including MQTT, have been tested in constrained military networks for command and control systems by a RTG [11]. This research shows that MQTT and MQTT-SN are fairly lightweight, and preferable to other industry standards like Advanced Message Queuing Protocol (AMQP) for use in constrained networks.

ZeroMQ also finds itself to be one of the more promising protocols for use in constrained networks [12]. It has also been evaluated by an RTG on group communication in military networks, where it achieved good results [13]. Both these studies show ZeroMQ outperforming industry standards like Data Distribution Service (DDS) and AMQP.

For this reason, we further investigate the MQTT, MQTT-SN and ZeroMQ protocols to determine their usability for use in constrained networks. In this paper, our findings support the findings of these studies, from which the overall conclusion is that UDP gives lower latency in general (almost the same as MQTT with Quality of Service (QoS) level 0).

3 Experiment Design

As part of the experiment, we needed to establish and emulate relevant networking technologies. We also needed a testbed to provide an environment for controlled, repeatable experiments and analysis.

3.1 Networks

In order to emulate networks with varying degrees of constraints, we have designed several network models that emulate realistic networks that could be used in search and rescue or disaster relief operations.

The network models can be seen in table 1. We have a model to emulate low-band 5G networks, which we expect to play a prominent role in future operations, with the use of mobile base stations and edge computing.

Another class of networks, more constrained than 5G, but very relevant to establishing ad hoc communication capacity is military networks. Tactical Broadband radios are designed to be used in challenging outdoor environments, i.e., the tactical battlefield [14].

Satellite communication (SATCOM) may provide super high frequency (SHF) and ultra-high frequency (UHF) communications access. SHF is used for static and deployed stations on the ground, while UHF is used for tactical communications [15]. We have focused on testing with a SATCOM model similar to what is provided by UHF, due to the nature of operations we consider.

NATO Narrowband Waveform is a standard developed by NATO to provide a single-channel ad hoc network that will serve voice and data traffic [16].

Combat Network Radio (CNR) operates in a network that provides both a half-duplex circuit and either a single radio frequency or a set number of radio frequencies [17].

Table 1. Network models

Network	**Data rate**	**Latency**	**Loss percentage**
Low-band 5G	100 mbit/s	20 ms	0%
Tactical Broadband	2 mbit/s	100 ms	1%
SATCOM	250 kbit/s	550 ms	0%
NATO Narrowband Waveform	16 kbit/s	500 ms	0%
CNR	9.6 kbit/s	100 ms	1% / 10%

3.2 Testbed

In order to test the performance of the protocols in constrained networks, we have developed an analysis tool running in Python (version 3.8.10). The tool is built using the packet capture library tcpdump [18]. Tcpdump saves packet information in a PCAP file [19], which can later be analyzed in the network analysis tool, Wireshark [20]. We implemented support for the networks presented in the previous section, namely 5G, Tactical Broadband, SATCOM, NATO Narrowband, and CNR. This was done using the netem tool [21] as part of our framework. Netem provides functionality that can emulate network properties like bandwidth, packet loss, delay, duplication and re-ordering. The tool is enabled in the

Linux kernel and is controlled by a command line tool. In our experimentation we used it to simulate challenging network conditions by limiting bandwidth, increasing delay and increasing the percentage chance of packet loss occurring. We opted for netem because it has been shown to be empirically and statistically correct [22]. Further we implemented support for testing MQTT, MQTT-SN and ZeroMQ across these networks.

We have used the Pandas library in Python (version 1.4.0) [23] to store logs and create statistics during each test run.

We did consider building on existing testbeds, and investigated several options. These testbeds, and the rationale behind why we opted to develop our own testbed is further described in the thesis [8].

3.3 Execution

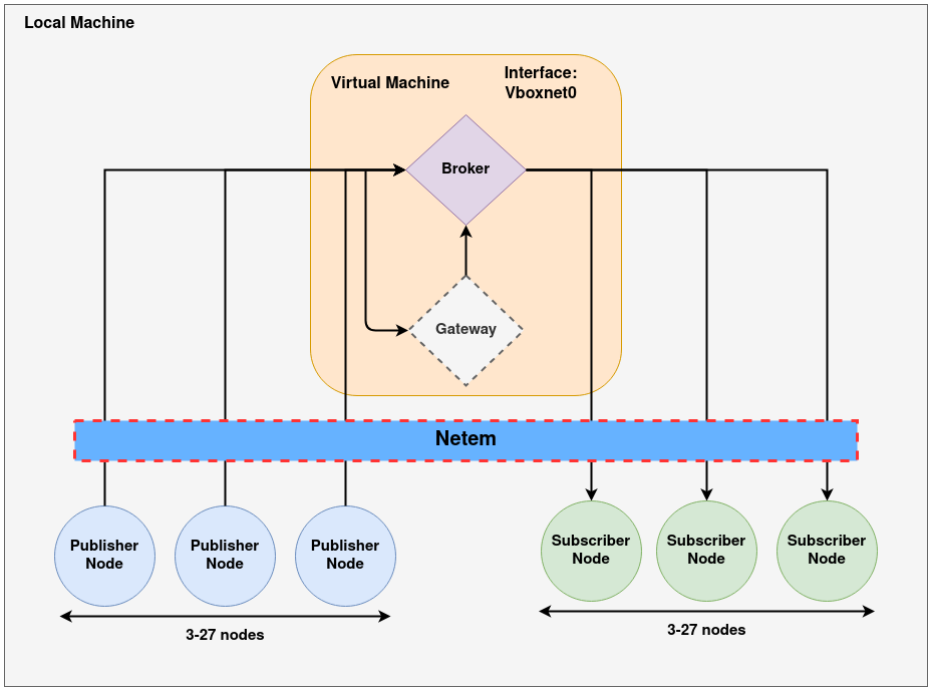

Fig. 1. The test setup used by all protocols

The tests that we ran during our experimentation were conducted with a similar base setup as can be seen in figure 1. To run the setup, we used a Linux machine running Ubuntu (version 20.04.3 LTS), where we configured and ran the tests using our analysis tool. On the machine, we set up a virtual machine (also running on Ubuntu), which worked as a server that the clients connected to

when starting a test. The server has a broker, which is needed in the case of MQTT, and a gateway used by MQTT-SN. The broker filters and forwards messages from the publisher to the subscriber, while a gateway converts MQTT-SN messages from the publishers to MQTT messages that can be accepted by the broker. While testing with MQTT, we used 27 publisher clients and 27 subscriber clients. MQTT has a QoS setting, which enables a range of guarantees for message delivery. This is on top of the guarantees provided by TCP which MQTT uses, and ranges from 0 (fire and forget), 1 (at least once), and 2 (only once).

The data we have chosen to use for these tests, include GPS and image data. These data types are highly relevant to the use cases we have considered, namely disaster relief and search and rescue operations. GPS provides vital information about not only the position of search crews and personnel, but possibly also the victims in need of assistance. Images may provide additional information vital to an operation by providing an overview of an area, or a visual representation of an asset.

Some limitations were put on our testing of the MQTT-SN protocol since the implementation we had to use for the clients was not optimal. The limitations include issues sending data with larger payloads and using QoS 2. Therefore, we only ran tests with MQTT-SN using QoS 0 and 1, only with smaller data sizes and we only used 18 clients of each type. We decided to test the protocol using two different test variations. The standard test variation is MQTT-SN to MQTT-SN, meaning that the publishers produce MQTT-SN messages, and the subscribers are configured to receive MQTT-SN messages. The other variation is a hybrid one, more commonly seen in other implementations. The hybrid variation has publishers sending MQTT-SN messages to the gateway, which then forwards them to the broker as MQTT messages. The subscribing clients are standard MQTT clients that receive MQTT messages form the broker. Unlike MQTT, MQTT-SN uses UDP for message transport.

In our experimentation with ZeroMQ, we opted to test it with two different transport protocols, namely TCP and Pragmatic General Multicast (PGM). PGM is a protocol for reliable multicast transport of data over IP networks. ZeroMQ has a socket API [24], where the sockets used are dependent on the implementation. The setup used for ZeroMQ is similar to MQTT with 27 clients of each type, which exchange messages over a broker located on the virtual machine. Clients in ZeroMQ need to be implemented using sockets from the socket API. We have implemented clients for both TCP and PGM using sockets from the publish/subscribe pattern and using an architecture similar to what is provided as an example by the ZeroMQ developers. Since ZeroMQ does not have a predefined broker component, we implemented our own with a filtering mechanism comparable to what was used by MQTT. ZeroMQ does not have the QoS setting found in MQTT and MQTT-SN and only relies on the delivery semantics used by the sockets.

Of the protocols we have tested in this study, MQTT is the most mature technology, with well-supported tools, good documentation, and a large community.

ZeroMQ has good tools and is quite mature, but the community is somewhat small compared to that of MQTT. Implementing certain solutions with ZeroMQ can provide a challenge due to lacking documentation and support. MQTT-SN is the least mature of the protocols, with poor tooling and a very small community as of today.

4 Results

Using the analysis tool we developed, we tested the MQTT, MQTT-SN, and ZeroMQ protocols using different network models. We aimed to provide recommendations on which protocol(s) to use for the different networks we tested. To compare the protocols, we the evaluate average transmit delay and packet loss. We have tested with GPS data, which is useful for providing information about the location of people or vehicles. We have also tested with image data, which provides additional situational information. The results presented here are from tests with the 5G and SATCOM network models, commonly used in the civilian sector, and CNR, which is on the lower limit of what is used in military operations. The complete results can be found in the thesis [8].

4.1 5G

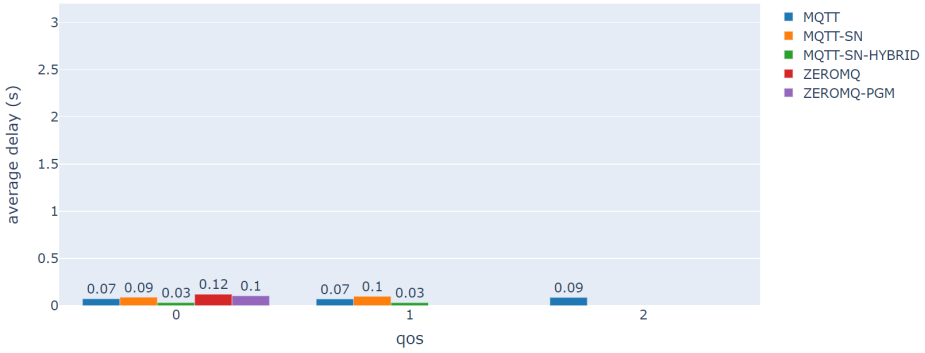

Fig. 2. Average transmit delay times using the 5G network model with GPS data

Figure 2 shows how the protocols performed regarding average transmit delay during the tests with the 5G network model using GPS data. Note that both tests with ZeroMQ are positioned in the QoS 0 category. This is because ZeroMQ does not have any inherent QoS settings and only relies on the delivery semantics provided by the socket type used. This will be the same for all the coming figures. Table 2 shows the packet loss for each protocol. Most of the protocols had low

delays, at around 100 milliseconds per message. The protocol that stands out most is MQTT-SN with the hybrid variation, which is significantly quicker than the others. This was likely due to UDP, which has much less overhead than TCP. The standard variation is in line with the other protocol, so there might be some delay introduced by the clients used since this variation uses both MQTT-SN clients for publishing and subscribing. Looking at the QoS levels for MQTT and MQTT-SN, we see that the higher QoS levels had no significant impact on the average delay of the protocols, which is to be expected in this type of network.

Table 2. Packet loss from tests with the 5G network model using GPS data

Protocol	QoS	Packet Loss
MQTT	**0**	0%
	1	0%
	2	0%
MQTT-SN Standard	**0**	0%
	1	0%
MQTT-SN Hybrid	**0**	0%
	1	0%
ZeroMQ TCP	**None**	0%
ZeroMQ PGM	**None**	0%

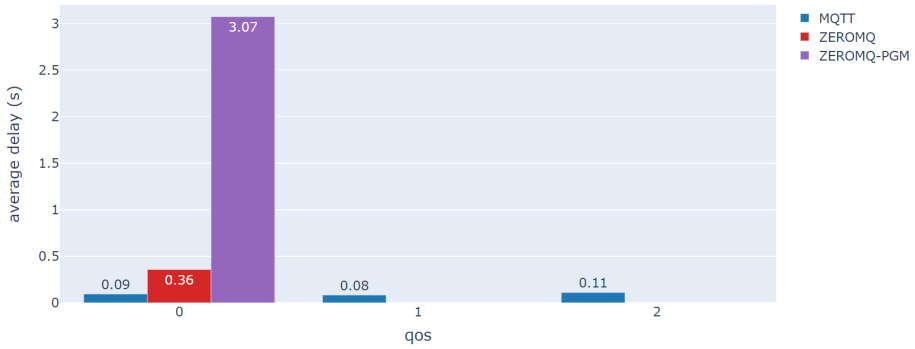

Fig. 3. Average transmit delay times using the 5G network model with image data

Figure 3 shows how the protocols performed regarding average transmit delay during the tests with the 5G network model using image data. Note that since we did not perform tests with MQTT-SN using image data, the figure shows no results for this protocol. Table 3 shows the packet loss for each protocol. Here we see that the MQTT protocol had marginally higher delays than the tests with GPS data, which is expected due to the increase in message size. The ZeroMQ

protocol showed a large increase in delay times for the TCP and PGM tests. The increased message size seems to have a greater effect while sending with these protocols than with MQTT, especially for PGM. The network statistics show that the number of packets created by multicasting is high, which may have affected performance somewhat.

Table 3. Packet loss from tests with the 5G network model using image data

Protocol	QoS	Packet Loss
MQTT	**0**	0%
	1	0%
	2	0%
MQTT-SN Standard	**0**	NA
	1	NA
MQTT-SN Hybrid	**0**	NA
	1	NA
ZeroMQ TCP	**None**	0%
ZeroMQ PGM	**None**	0%

4.2 SATCOM

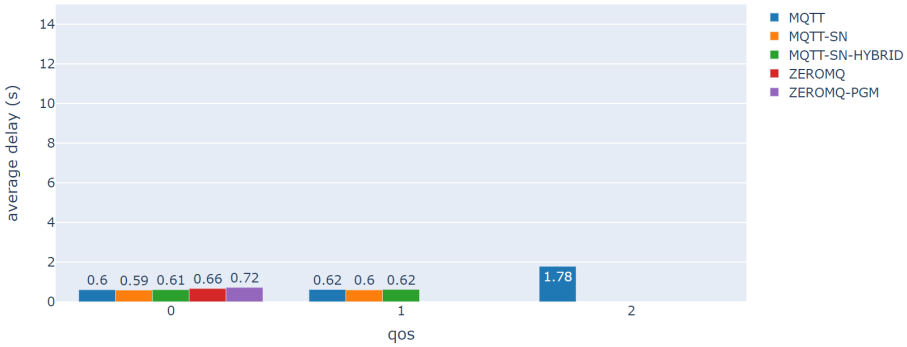

Fig. 4. Average transmit delay times using the SATCOM network model with GPS data

Figure 4 shows how the protocols performed in regards to average transmit delay during the tests with the SATCOM network model using GPS data. Table 4 shows the packet loss for each protocol. As with the previous network models, we can see that the average delay is similar across the protocols. The only exception is MQTT with QoS 2, which has a delay of almost two seconds. This is again because QoS 2 introduces quite a bit of overhead in the form of handshakes

and acknowledgments compared to QoS 0 and 1. The average delay is somewhat higher than the previous network models, but this was expected as the SATCOM network model introduces a delay of 550 milliseconds.

Table 4. Packet loss from tests with the SATCOM network model using GPS data

Protocol	QoS	Packet Loss
MQTT	**0**	0%
	1	0%
	2	0%
MQTT-SN Standard	**0**	0%
	1	0%
MQTT-SN Hybrid	**0**	0%
	1	0%
ZeroMQ TCP	**None**	0%
ZeroMQ PGM	**None**	0%

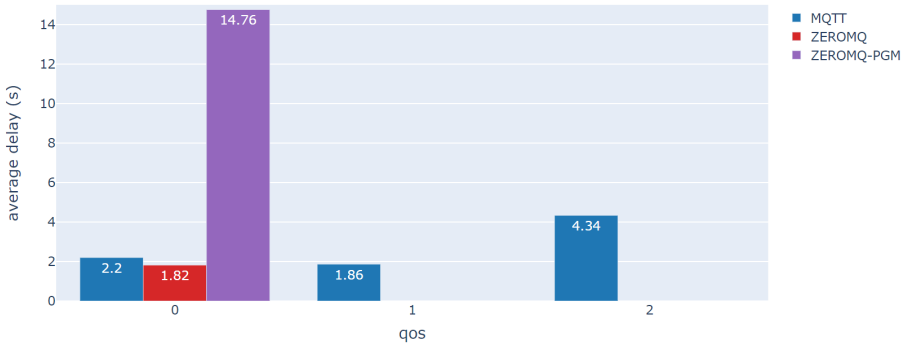

Fig. 5. Average transmit delay times using the SATCOM network model with image data

Figure 5 shows how the protocols performed in regards to average transmit delay during the tests with the SATCOM network model using image data. Table 5 shows the packet loss for each protocol. The results follow a similar pattern to the other tests with image data in that the ZeroMQ using the PGM protocol is slower than MQTT. The main difference we see here is that ZeroMQ with TCP has a lower transmit delay than MQTT, even with QoS 0. In the previous comparison with Tactical Broadband, we saw that ZeroMQ had a much higher delay than MQTT. PGM also had a significant packet loss, which we have not experienced while testing any other protocol using SATCOM. There was also some packet loss from the Tactical Broadband test, but this was very little and

may be due to connection issues at startup. The packet loss for SATCOM is much higher at over 12%, which cannot be attributed to startup issues. The results also show that the delay has increased more for MQTT than for the tests using GPS data. This was especially the case with QoS 2. We know from the results on the network layer that the data rate decreased significantly in the test using this network model compared to the other models.

Table 5. Packet loss from tests with the SATCOM network model using image data

Protocol	QoS	Packet Loss
MQTT	**0**	0%
	1	0%
	2	0%
MQTT-SN Standard	**0**	0%
	1	0%
MQTT-SN Hybrid	**0**	0%
	1	0%
ZeroMQ TCP	**None**	0%
ZeroMQ PGM	**None**	12.77%

4.3 CNR with 10% Loss

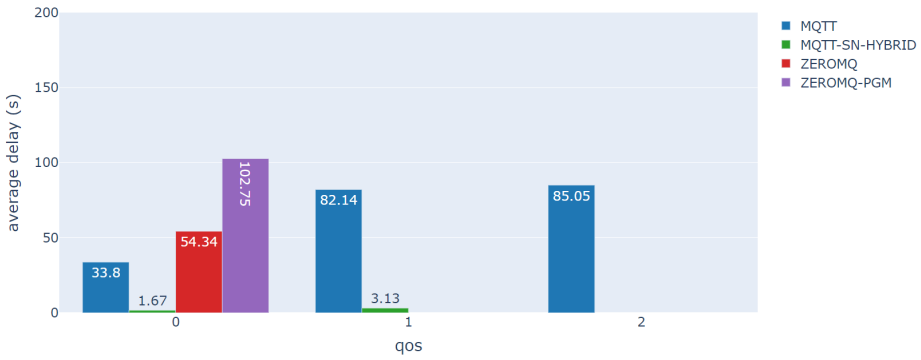

Fig. 6. Average transmit delay times using the CNR network model with 10% loss with GPS data

Figure 6 shows how the protocols performed in regards to average transmit delay during the tests with the CNR network model with 10% loss using GPS data. Table 6 shows the packet loss for each protocol. This is the most constrained network that we have tested, and again we see that most of the protocols have

high delays and cannot deliver messages efficiently regarding speed. The exception is MQTT-SN using the hybrid test variation, which shows low delays. The standard variation of the MQTT-SN test did not manage to receive any messages and therefore performed worse than the hybrid variation. This is the only test showing a significant decrease in data rate while using MQTT-SN, which shows that this protocol struggles somewhat in transmitting messages.

Table 6. Packet loss from tests with the CNR network model with 10% loss using GPS data

Protocol	QoS	Packet Loss
MQTT	**0**	8.55%
	1	19.37%
	2	61.78%
MQTT-SN Standard	**0**	100%
	1	100%
MQTT-SN Hybrid	**0**	20.21%
	1	25%
ZeroMQ TCP	**None**	8.35%
ZeroMQ PGM	**None**	60.96%

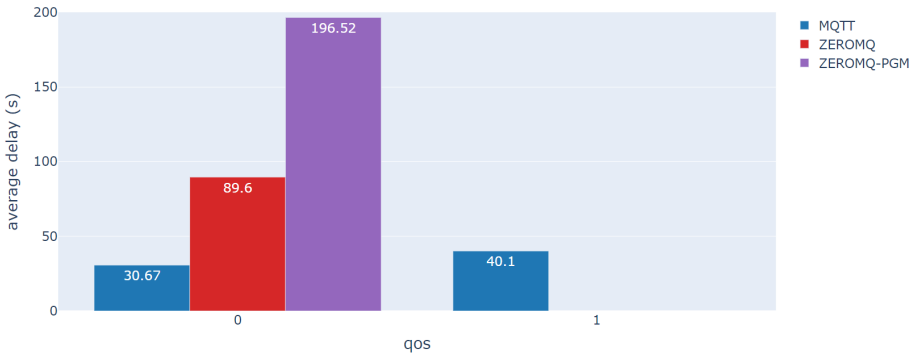

Fig. 7. Average transmit delay times using the CNR network model with 10% loss with image data

Figure 7 shows how the protocols performed in regards to average transmit delay during the tests with the CNR network model with 1% loss using image data. Table 7 shows the packet loss for each protocol. We see similar results here where no protocols could transmit messages efficiently. This shows that these networks are challenging, even if the protocols are developed for use in limited networks with poor bandwidth.

Table 7. Packet loss from tests with the CNR network model with 10% loss using image data

Protocol	QoS	Packet Loss
MQTT	**0**	94.29%
	1	91.51%
	2	100%
MQTT-SN Standard	**0**	NA
	1	NA
MQTT-SN Hybrid	**0**	NA
	1	NA
ZeroMQ TCP	**None**	97.30%
ZeroMQ PGM	**None**	93.28%

4.4 Discussion

MQTT shows good performance in the less limiting networks we have tested. The protocol shows no issues in transmitting messages, both in regard to delay and packet loss. Using QoS 2 showed that the amount of overhead caused issues for the protocol, primarily due to the extra steps needed before a message transmission was completed. Due to this, we only recommend using QoS 0 and 1, as the overhead is mostly reduced with these settings. The tests with more limiting networks show that the protocol struggled and is therefore not found suitable for use. The protocol was the easiest to set up and use compared to the other protocols, and this is to be expected, as MQTT is an industry-standard protocol in IoT.

Our results show that MQTT-SN had good results for almost all tests. The delay was low with both test variations across all network models, and the protocol outperformed the other ones in most tests. The packet loss was also comparatively low. Both QoS 0 and 1 showed good results, but QoS 0 was most times faster than QoS 1. The only issues were found during the test using the CNR network model with a 10% loss. Here, only the hybrid test variation could send and receive messages. Even though the network is quite limited, it showed good results, both in terms of delay and packet loss. Still, the packet loss was too high for us to be able to recommend its usage in such a network. While overall results are good, the implementation limitations have restricted us from adequately being able to compare it to the other protocols. Setting up the protocol was not an easy task and was significantly more difficult than the other two protocols we have investigated, except for ZeroMQ with PGM.

ZeroMQ showed the overall poorest results from the protocols we have tested. TCP had some promise with smaller message sizes, where the results were comparable to that of MQTT and MQTT-SN. However, the test with larger message sizes showed that the protocol had issues, both in terms of delay and packet loss. PGM provided the overall worst results, especially with larger message sizes. The poor results might be due to the test implementation, which is difficult to say without more testing. The results may have been better without a broker, but

the downside is the increased complexity of setting up and connecting the clients. Setting up the protocol with clients was relatively simple, but there were certain networking issues with getting PGM to work, as it is not well supported in ZeroMQ.

5 Conclusion

In this paper, we have evaluated three publish/subscribe protocols for use in constrained networks. The networks encompass both civilian and military communication solutions that are relevant to search and rescue operations. Using network emulation, we have evaluated the protocols when conveying position information (representative of tracking the units involved in an operation) as well as images (in support of a shared awareness between units).

We found that MQTT shows good performance in the less limiting networks we have tested. We recommend using MQTT with QoS levels 0 (fire and forget messages) and 1 (at least once semantics). Due to good tooling support, MQTT is easy to use.

Using a higher QoS level would usually be preferable in operations where the requirement for guaranteed delivery is higher than the need for speedy delivery, such as search and rescue operations. Our results show that in most networks QoS 2 provides neither. In less constrained networks, there is no packet loss and thus no need for anything higher than QoS 1, while in more constrained networks, setting the QoS higher only induces additional network congestion.

MQTT-SN, while providing the overall best performance, we found that the tooling support was lacking, and the protocol is not straightforward to use in this respect. So, while performance is good, implementation maturity is poor.

ZeroMQ exhibited the overall poorest performance for the networks we tested.

So, from a pure performance perspective we can recommend using MQTT-SN. Conversely, for the best community and tooling support, MQTT is the overall winner.

References

1. Gartner, "Internet of things (iot)," https://www.gartner.com/en/information-technology/glossary/internet-of-things, accessed on 2022.06.08.
2. ——, "IT Glossary — Edge Computing," https://www.gartner.com/en/information-technology/glossary/edge-computing, accessed on 2022.09.05.
3. Matthew O'Riordan, "Everything You Need To Know About Publish/Subscribe," https://ably.com/topic/pub-sub, accessed on 2022.09.05.
4. OASIS, "MQTT Version 3.1.1," http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html#_Toc398718018, accessed on 2022.09.05.
5. ——, "MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2," https://www.oasis-open.org/committees/download.php/66091/MQTT-SN_spec_v1.2.pdf, accessed on 2022.09.05.
6. The ZeroMQ authors, "ZeroMQ An open-source universal messaging library," https://zeromq.org/, accessed on 2022.09.05.

7. Emil P. Andersen, "Creating an analysis tool for testing and evaluating the mqtt protocol," https://github.com/EPA1/Preparatory-Project/, accessed on 2022.10.28.
8. ——, "Evaluating publish/subscribe protocols for use in constrained networks," https://hdl.handle.net/11250/3006200, accessed on 2022.09.01.
9. F. T. Johnsen, Z. Zieliński, K. Wrona, N. Suri, C. Fuchs, M. Pradhan, J. Furtak, B. Vasilache, V. Pellegrini, M. Dyk *et al.*, "Application of iot in military operations in a smart city," in *2018 International Conference on Military Communications and Information Systems (ICMCIS)*. IEEE, 2018, pp. 1–8.
10. M. Pradhan, "Interoperability for disaster relief operations in smart city environments," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, 2019, pp. 711–714.
11. M. Manso, N. Jansen, K. Chan, A. Toth, T. H. Bloebaum, and F. T. Johnsen, "Mobile Tactical Force Situational Awareness: Evaluation of Message Broker Middleware for Information Exchange," ICCRTS 2018.
12. Z. Kang and A. Dubey, "Evaluating DDS, MQTT, and ZeroMQ Under Different IoT Traffic Conditions," 2020. [Online]. Available: http://www.dre.vanderbilt.edu/~gokhale/WWW/papers/M4IoT2020
13. N. Suri, M. Breedy, R. Fronteddu, A. Morelli, E. Cramer, J. Nilsson, A. Hansson, K. Marcus, and A. Martens, "Evaluating the scalability of group communication protocols over synchronized cooperative broadcast," in *2021 International Conference on Military Communication and Information Systems (ICMCIS)*, 2021, pp. 1–9.
14. ASELSAN, "GRC-5220 Tactical Broadband ETHERNET Radio," https://www.aselsan.com.tr/en/capabilities/military-communication-systems/military-broadband-multimode-radiolinks/grc5220-tactical-broadband-ethernet-radio, accessed on 2022.09.05.
15. NATO, "SATCOM Post-2000 (Archived)," https://www.nato.int/cps/en/natohq/topics_50092.htm, accessed on 2022.09.05.
16. T. J. Berg, "NATO Narrowband Waveform (NBWF) - multilevel-level precedence and preemption for IP traffic," https://ffi-publikasjoner.archive.knowledgearc.net/handle/20.500.12242/953, accessed on 2022.09.05.
17. military-history.fandom, "Combat-net radio," https://military-history.fandom.com/wiki/Combat-net_radio, accessed on 2022.09.05.
18. The Tcpdump Group, "tcpdump and libcap," https://tcpdump.org/, accessed on 2022.09.05.
19. G. Harris and Ed. and M. Richardson Sandelman, "Pcap capture file format," https://tools.ietf.org/id/draft-gharris-opsawg-pcap-00.html, accessed on 2022.06.08.
20. Wireshark Foundation, "About Wireshark," https://www.wireshark.org/, accessed on 2022.09.05.
21. Linux Foundation, "netem," https://wiki.linuxfoundation.org/networking/netem, accessed on 2022.06.08.
22. A. Jurgelionis, J.-P. Laulajainen, M. Hirvonen, and A. I. Wang, "An empirical study of netem network emulation functionalities," in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, 2011, pp. 1–6.
23. NumFOCUS, "pandas," https://pandas.pydata.org/, accessed on 2022.09.05.
24. The ZeroMQ authors, "Socket api," https://zeromq.org/socket-api/, accessed on 2022.09.12.