A NOVEL COMMUNICATION PROTOCOL DESIGN FOR INTERNET OF FLYING THINGS

UÇAN NESNELERİN İNTERNETİ İÇİN YENİ BİR İLETİŞİM PROTOKOLÜ TASARIMI

BURAK HAN ÇORAK

PROF. DR. SUAT ÖZDEMİR
Supervisor

Submitted to
Graduate School of Science and Engineering of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Engineering

January 2023

ABSTRACT

A NOVEL COMMUNICATION PROTOCOL DESIGN FOR INTERNET OF FLYING THINGS

Burak Han ÇORAK
Master of Science, Computer Engineering
Supervisor: Prof. Dr. Suat ÖZDEMİR
January 2023, 103 pages

In this thesis a new communication protocol, Message Queuing Telemetry Transport Extended (MQTT-XT) is proposed for the Internet of Flying Things (IoFT) network. The main objective of the proposed protocol is to enable UAVs to be considered as part of IoT nodes. The MQTT-XT protocol bridges the gap between UAVs and IoT devices by converting MQTT and MAVLINK message types to make them compatible with each other. Three different client types have been proposed to support long-distance and distributed communication with UAVs and IoT devices. A testbed was set up in simulation to evaluate the performance of MQTT-XT and MAVLINK protocols in terms of UAV management messages and 4 different IoFT scenarios. The experimental results show that MQTT-XT consumes 5 times less message size to transmit command messages and 20 times less for status messages. Additionally, the MQTT-XT protocol-based network completes missions 30% faster, and the packet loss ratio is reduced from 20% to 2%. In conclusion, MQTT-XT outperforms MAVLINK in IoFT autonomous applications.

Keywords: MQTT-XT, Internet of Things, Internet of Flying Things, Unmanned Aerial Vehicle, IoT Communication Protocols, MQTT, MAVLINK

ÖZET

UÇAN NESNELERİN İNTERNETİ İÇİN YENİ BİR İLETİŞİM PROTOKOLÜ TASARIMI

Burak Han ÇORAK
Yüksek Lisans, Bilgisayar Mühendisliği
Danışman: Prof. Dr. Suat ÖZDEMİR
Ocak 2023, 103 sayfa

Bu tezde, Uçan Nesnelerin İnterneti (IoFT) ağı için yeni bir iletişim protokolü olan Message Queuing Telemetry Transport Extended (MQTT-XT) önerilmiştir. Önerilen protokolün temel amacı, İHA'ların IoT düğümlerinin bir parçası olarak kabul edilmesini sağlamaktır. MQTT-XT protokolü, MQTT ve MAVLINK mesajlarını dönüştürerek ve birbirleriyle uyumlandırarak İHA'lar ve IoT cihazları arasındaki boşluğu doldurur. İHA'lar ve IoT cihazları ile uzun mesafeli ve dağıtık iletişimi desteklemek için üç farklı istemci tipi önerilmiştir. MQTT-XT ve MAVLINK protokollerinin performansının İHA yönetim mesajları ve 4 farklı IoFT senaryosu açısından değerlendirilmesi amacıyla simülasyonda test ortamı hazırlanmıştır. Deneysel sonuçlar, MQTT-XT'nin komut mesajlarını iletmek için 5 kat ve durum mesajları için 20 kat daha az mesaj boyutu tükettiğini göstermektedir. Ek olarak, MQTT-XT protokol tabanlı ağ, verilen görevleri %30 daha hızlı tamamlar ve paket kayıp oranı %20'den %2'ye düşürür. Sonuç olarak MQTT-XT, IoFT otonom uygulamalarında MAVLINK'ten daha iyi performans gösterir.

Anahtar Kelimeler: MQTT-XT, Nesnelerin İnterneti, Uçan Nesnelerin İnterneti, İnsansız Hava Aracı, IoT İletişim Protokolü, MQTT, MAVLINK

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my thesis advisor Prof. Dr. Suat ÖZDEMİR for his invaluable guidance, encouragement, and support throughout the entire research process. I am also grateful to my mentor Dear Ahmet Burak NASUHBEYOĞLU, for his generosity in sharing his wealth of knowledge and experience with me in my thesis journey. Besides, I would like to thank my thesis committee members, Assoc. Prof. Adnan ÖZSOY, Assoc. Prof. Ali Seydi KEÇELİ, Assoc. Prof. Mehmet DEMİRCİ and Asst. Prof. Yılmaz AR for reviewing this thesis and giving insightful comments. I am deeply grateful to my study participant Muhammed Emir Çakıcı, for his time and willingness to participate in this research.

My parents Naile and Yıldırım Beyazit ÇORAK have been a constant source of support and encouragement throughout my academic journey, and I am also eternally grateful to my little sweet sister Ayşegül ÇORAK, my elder sister Elif Nur GEZER and her husband Ph.D. Volkan GEZER for their love and support. I would like to extend my special thanks to Dr. Metehan AKTEPE, for his lifetime friendship, motivation, inspiration, and psychological support.

Finally, words cannot be sufficient to begin expressing myself but I would like to thank my beloved wife Merve ÇORAK, for her love, encouragement, and understanding during the time I spent working on this thesis. Without her love and support, this journey would not have been possible. Lastly, I want to thank my daughter Irmak Ada ÇORAK, for bringing joy, and happiness to my life on this journey. You have made me stronger, better, and more fulfilled than I could have ever imagined.

I would like to extend my most sincere appreciation to all of my friends who have provided me with unwavering support throughout the research process. Unfortunately, the constraints of this page do not permit me to mention each and every one of you by name, but please know that your contributions have been invaluable to my success. I am forever grateful for your support and be part of my life.

CONTENTS

Page

TABLES

FIGURES

Page

ABBREVIATIONS

6LoWPAN	:	IPv**6** over **Low**-Power Wireless **P**ersonal **A**rea **N**etworks
AMQP	:	**A**dvanced **M**essage **Q**ueuing **P**rotocol
API	:	**A**pplication **P**rogramming **I**nterface
CoAP	:	**Co**nstrained **A**pplication **P**rotocol
DDS	:	**D**ata **D**istribution **S**ervice
FANET	:	**F**lying **A**d Hoc **Net**work
GS	:	**G**round **S**tation
HITL	:	**H**ardware **in** **t**he **L**oop
HTTP	:	**H**yper **T**ext **T**ransfer **P**rotocol
IP	:	**I**nternet **P**rotocol
IoFT	:	**I**nternet **o**f **F**lying **T**hings
IoT	:	**I**nternet **o**f **T**hings
MANET	:	**M**obile **A**d Hoc **Net**work
MAVLINK	:	**M**icro **A**ir **V**ehicle **Link**
MQTT	:	**M**essage **Q**ueuing **T**elemetry **T**ransport
MQTT-XT	:	**M**essage **Q**ueuing **T**elemetry **T**ransport E**xt**ended
PSMI	:	**P**rioritization on **S**elect-able **M**ission **I**tem
QoDS	:	**Q**uality **o**f **D**rones **S**ervices
QoS	:	**Q**uality **o**f **S**ervices
RC	:	**R**emote **C**ontroller
RF	:	**R**adio **F**requency
ROS	:	**R**obot **O**perating **S**ystem
SDK	:	**S**oftware **D**evelopment **K**it
SITL	:	**S**oftware **in** **t**he **L**oop
SN	:	**S**ensor **N**etwork
TCP	:	**T**ransmission **C**ontrol **P**rotocol

UAV	:	**U**nmanned **A**erial **V**ehicle
UDP	:	**U**ser **D**atagram **P**rotocol
VANET	:	**V**ehicular **A**d Hoc **Net**work
WANET	:	**W**ireless **A**d Hoc **Net**work
WiFi	:	**Wi**reless **Fi**delity
XMPP	:	E**x**tensible **M**essaging and **P**resence **P**rotocol

1. INTRODUCTION

Communication has been one of the most important milestones of humanity in every aspect throughout history. The revolution of technologies is getting more advanced by interchanging information between many scientists. For example, the computer is one of the greatest inventions in technology history. It helps to solve many complex problems that require many calculations. However, advancement in this area has been taking a big step since the World-Wide-Web (WWW) was invented. Thanks to this invention, people could start sharing their knowledge, experience, and many things in a few minutes instead of weeks.

The same principle spreads to machines. The basic definition of the Internet of Things (IoT) is the things that connect to the internet. In other words, the devices that connect and communicate with each other. These tiny, low-resource, low-price, low-power consumer devices are integrated into almost every place in our daily lives. They are also integrated into many health, heavy industry, farming, etc., applications [1]. The application area of the IoT industry has expanded more widely in the last decades with many advancements in the communication area. Popular companies in tech area like Intel, Microsoft, Amazon, Google, and Cisco start different projects to achieve next-generation solutions in the IoT communication industry [2], as these devices are generally produced with low resource-consuming and provide cost-effective solutions.

Accordingly, many new-brand protocols are introduced such as Constrained Application Protocol (CoAP), Message Queuing Telemetry Transport (MQTT) etc., since the IoT devices run on low-bandwidth, low-resources, and extremely high latency networks in many cases [3]. These lightweight protocols aim to send messages under difficult circumstances by providing a reliable connection. Also, protocols are developed to cope with the high density of message traffic since the IoT environment consists of different devices. Furthermore, the scalability factor of IoT devices is higher when compared to the traditional networks, as we know in the WWW. The main purpose of the protocols is to organize the network by considering more complicated low-connection networks where the number of nodes can reach thousands while ensuring a low rate of packet loss.

The scalability factors of IoT devices were limited to existing terrestrial networks [4]. However, applications area of the IoT can be varied such as agriculture, disaster monitoring, etc., where terrains are more rough and terrestrial networks are not available. By considering different application domains, the scalability factor becomes a more challenging problem. Thus, many suggestions were provided by the researchers to overcome these communication problems. One of these suggestions includes a wireless ad hoc network (WANET). WANETs are specified for their domain and they are decentralized network types.

The Internet of Flying Things (IoFT) has emerged concept among all ad hoc networks [5]. The future of IoT has become another level with Unmanned Aerial Vehicle (UAV) technology, since these low-cost flight-based devices provide more flexible and mobile communication when compared to other types of ad hoc networks. Moreover, the management of the IoFT is more complex. The ad hoc networks provide communication over their routers or access points. Nevertheless they are not focused on communicating or managing vehicles. Despite that, UAV systems are required to be directed and managed.

At this point, a communication protocol showed up as one of the biggest challenges. Because on the one hand, IoT devices use their protocols while on the other hand, UAVs use their protocols. This problem causes the design of two different communication infrastructures which can be more complex as the number of nodes increases which is completely opposed to the idea of the IoT concept. For this reason, UAVs should be considered as a part of the IoT network. One of the solutions to achieve this is to propose a single protocol that provides communication for IoFT components.

1.1. Scope Of The Thesis

The main focus of the thesis, to solve the communication problem that creates a distinction between drone systems which require ground stations to receive commands, and IoT devices where a large scale of the network is needed but insufficient to distribute due to a lack of process capabilities. The IoT protocols are well-designed for low bandwidth, low resources, and packet loss. However, the current UAV' protocols are more talkative and need more resources than IoT devices from the perspective of the network load. Furthermore, UAVs need centralized management to take commands and report their status to ground stations.

In this thesis, a new MAVLINK enabled MQTT Extended (MQTT-XT) protocol design is proposed to consider UAVs as a part of IoFT nodes. For this reason, a single protocol is suggested which provides communication between IoT and UAVs by interpreting MAVLink and MQTT messages to each other in the UAV platform. Furthermore, IoT network devices work in resource-constrained environments. MQTT-XT makes it possible for UAVs to work in this type of network with its lightweight structure. Furthermore, MQTT-XT has specialized functions which employ multiple UAVs for communication. The proposed novel protocol is built on top of MQTT by extending and stretching the MQTT structure. The MQTT-XT protocol makes it more cost-effective for communication among IoFT while it responds to the requirements of both UAV and IoT nodes.

1.2. Contributions

In this thesis, the deficiency of the communication protocol of the IoFT concept is handled. The IoT environment includes heterogeneous devices, while often they work on constrained resources. Similarly, IoFT has not only many different devices but it also has UAVs. However, there is no proposed protocol to manage UAVs over existing IoT protocols. These constraints push researchers to create more unique architectures when they work on different scenarios.

The main motivation of the thesis is to make the IoFT concept more flexible, reliable, low-cost, and efficient with the proposed protocol. The approach proposes a single protocol to overcome communication problems for UAVs in IoFT networks by considering UAVs as IoT nodes. This novel design MQTT-XT protocol offers a simple and efficient approach while the UAVs' nodes are still able to communicate with other IoT devices. The key difference between the MQTT-XT and the existing proposed models is a single protocol that aims to communicate UAVs directly with over MQTT topics by interpreting MQTT and MAVLINK messages to each other in the UAV platform.

The main contributions of this thesis are summarized as follows:

- A new MQTT-XT (MQTT Extended) protocol for IoFT which provides to control UAV over MQTT topics without changing the structure of MQTT is proposed
- Unlike most of the previous work, a single protocol is built by considering the requirements of UAVs while extending MQTT for IoFT.
- For the first time, an IoT communication protocol that accepts UAVs as part of an IoT network is proposed
- Customized functions like QDoS and PSMI for managing UAVs is introduced. Also, specified functions such as broadcast messages that provide to manage drones with a single command message are presented which have not existed in previous studies.
- The experimental results show that the MAVLINK protocol is not suitable for the IoFT since it has a heavy message traffic.

1.3. Organization

The organization of the thesis is as follows:

- **Chapter 1** presents the motivation, contributions, and the scope of the thesis.
- **Chapter 2** provides basic knowledge of IoT and UAV protocol. Also, the existing problems for IoFT are given in this chapter.
- **Chapter 3** gives related works which aim to establish communication between IoT devices and drones. The related works will be examined under two different topics: Multi-Protocol Based Architectures and Single-Protocol Based Architectures
- **Chapter 4** introduces the novel protocol MQTT Extended (MQTT-XT) with its 3 different types of clients by inspecting the general architecture of the protocol.
- **Chapter 5** demonstrates the experimental results for our thesis. These results are obtained by comparing MAVLINK and MQTT-XT protocols with different communication metrics on different message sets and various scenarios in the simulation environment.
- **Chapter 6** states the summary of the thesis and possible future directions.

2. BACKGROUND OVERVIEW

The IoT has rapidly gained a reputation since the term includes all devices which can connect to the Internet. The IoT devices can be heavy computing machines, smartphones, micro-chip-based machines with basic functions, sensing the environment with a dozen sets of sensors, or daily devices that people used for their daily routines [6]. This wide and various definition consists of many different devices. For this reason, the term started to evolve more and more heterogeneous structures day by day. Besides, the complexity level is increasing and the management of networks that have very different smart devices is even harder than before.

The other feature that gives a reputation to IoT is the ability to work in low-resources and constrained environments. However, the trade-off is the reliability of the network. The development of the existing traditional application layer protocols and the new IoT protocols are introduced to overcome reliability problems. The IoT protocols are aimed to organize the network by considering the heterogeneous structure of devices which is presented in Figure 2.1. Also, they provide reliability for the constrained resource environments.

After these huge steps, the research domain of IoT research is spread among very wide areas like smart cities, smart farming, health-care monitoring, natural disasters, etc. However, the key problem for some of the research areas is many IoT devices that lack Internet networks. Since there is no Internet connection among these devices, they are not able to transmit data to the cloud. Even the local networks are provided to obtain certain data and transmit them when a network is available, the real-time dilemma exists for this type of connection [7]. Also, maintenance operations are very difficult for these types of networks. Besides, smart farming or natural disaster management application areas are more likely implemented on rough terrains which causes a couple of different problems for researchers.

One of the biggest challenges of long-distance communication in such solutions is the distribution in very wide areas as a nature of the IoT devices structure [8]. Furthermore, the terrestrial network cannot be accessible or many devices are faced with very small bandwidth

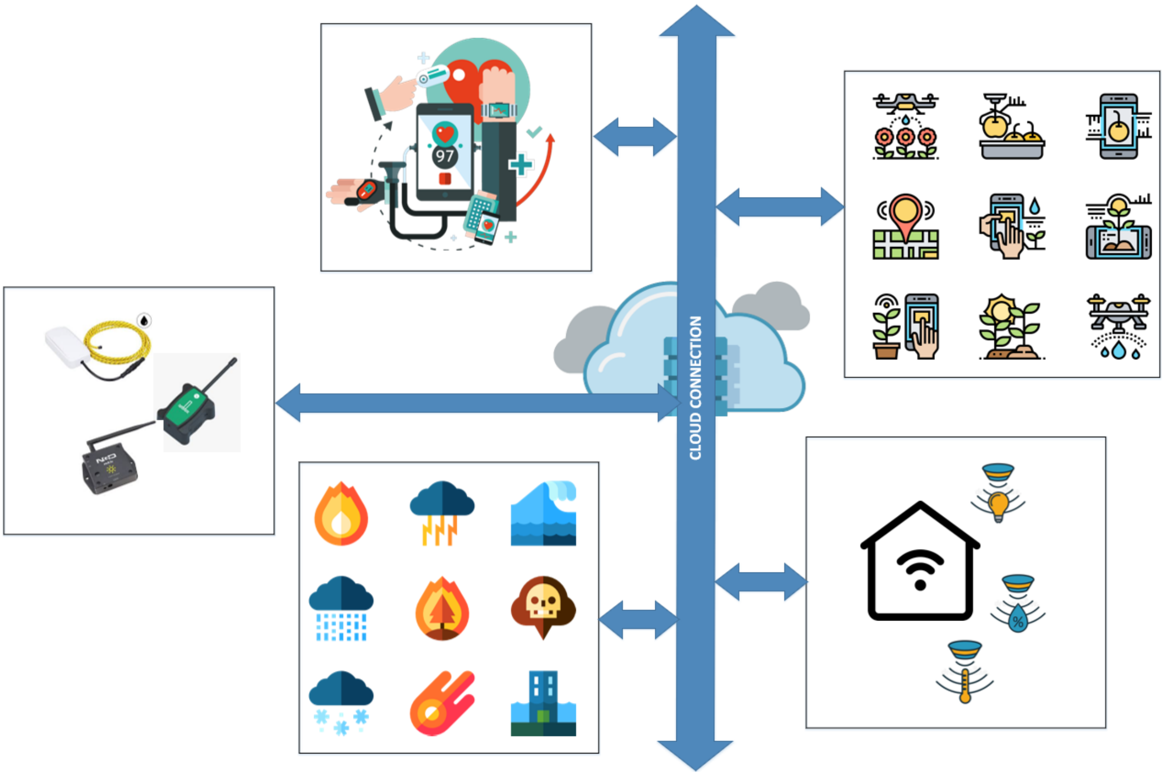


Figure 2.1 Heterogeneous Devices of the IoT Applications

even if it is accessed. In such resource-constrained environments, different IoT protocols such as MQTT and CoAP have been designed to make sure that the sensor network can be easily maintained, enabling devices to operate with limited resources. The importance of such protocols is vital to provide manageability of these devices which are scattered and spread over a wide environment. Additionally, due to the complete lack of access to the terrestrial network, at some points, different solution methods have been developed, such as satellite Internet, 4G, or 5G [9].

Besides these technologies, ad hoc networks such as Vehicular ad hoc networks (VANET), Mobile ad hoc networks (MANET), and Flying ad hoc networks (FANET) are provided to overcome accessibility problems. The main purpose of ad hoc networks is to solve accessibility problems with the different types of wireless connections and routing operations. One of the other ad hoc networks, which is named IoFT, emerged more than other types. The term includes many integrated UAVs to collect data from sensors. Figure 2.2 presents the aforementioned ad hoc network types.

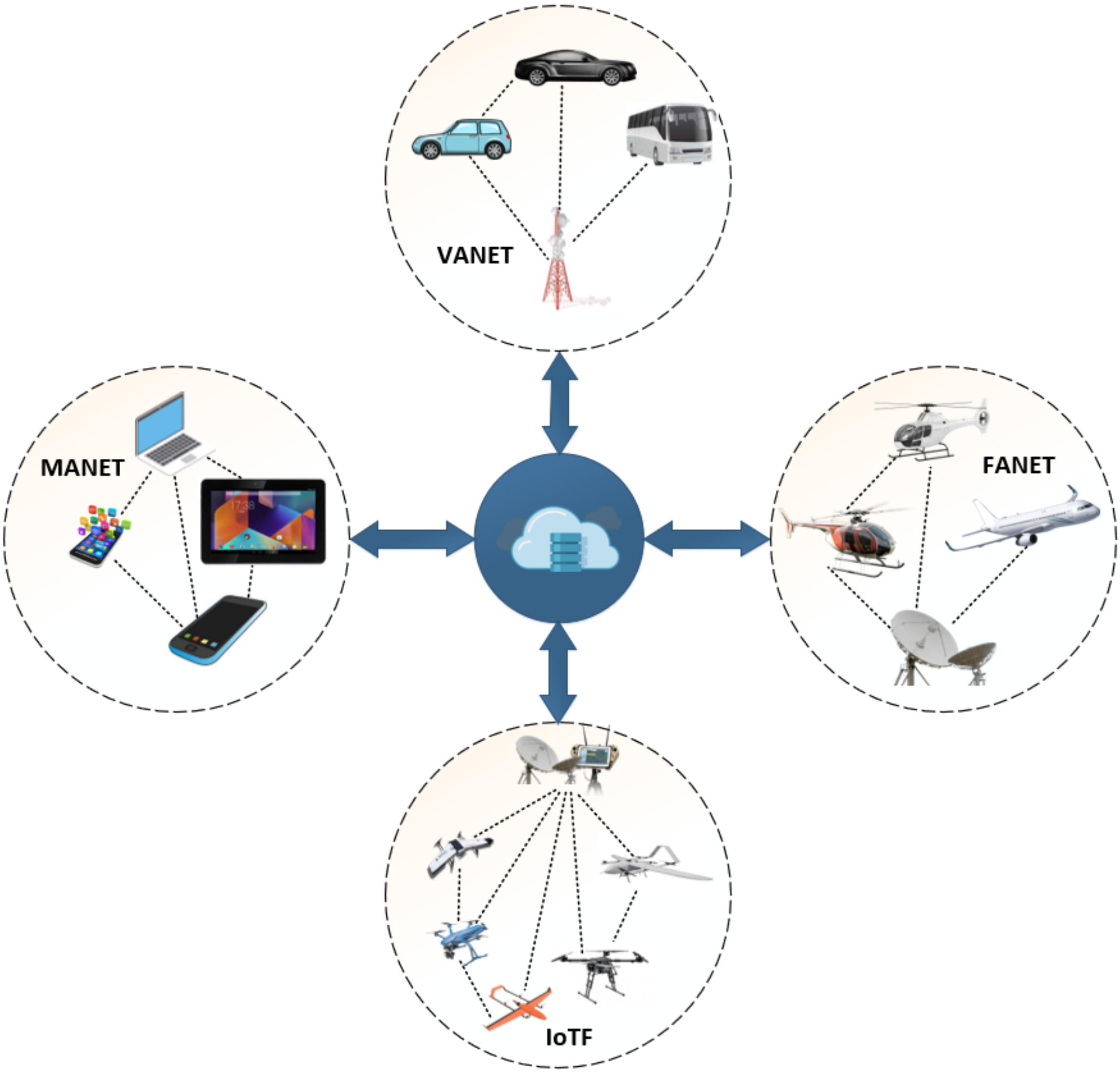


Figure 2.2 MANET, VANET, FANET, and IoFT

Even if the IoFT is inspected under the ad hoc network type, the term can be expanded to the application layer domain. Because the main purpose of the IoFT is to collaborate with drones to gather different data types. The typical problem for the IoFT type of network is the communication protocols since many UAVs are used in architecture. The management of drones stands out as another challenge. For example, MAVLINK, the most common of the existing drone protocols, is designed for RF communication with line-of-sight operations. Even if it has the capability to communicate with UAVs with Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), this capability is only limited to UAVs and not IoT devices. So, implementing the IoFT is not well described or handled regarding application layer protocols.

The IoFT is suggested by researchers to fill the gaps in IoT communication. Nevertheless, the details are not well defined and are still unclear in many aspects. The first problem is the management of UAVs in IoFT networks. UAVs are still considered to be like other types of devices. The IoFT employs UAVs to provide communication between IoT networks but does not offer to manage drones over existing network architectures. The second problem is that as the number of UAVs increases to tens or hundreds, the management of existing UAVs becomes harder and harder.

For example, given in Figure 2.3 and Figure 2.4, the drones are used to gather data from different IoT nodes and transmit end-user through the cloud. The multiple UAVs-based architectures offer mobility. However, the management of the multiple UAVs-based architectures are harder than the IoT network. The researchers should employ at least two different protocols to organize the architecture. One of them is needed to receive and transmit data from IoT nodes and one of them is needed to manage and control UAVs. This is the common dilemma of IoFT. Because, on the one hand, the IoFT provides high mobility networks by establishing communication over IoT nodes; on the other hand it has complexity since it has different types of devices.

To overcome this issue, the UAV should be considered as part of the IoT network and the management of UAV should be provided IoT protocols [10]. Thus, the UAV communication protocol becomes suitable for IoT environments and it can directly communicate with other IoT nodes. Moreover, existing UAV protocols such as MAVLINK use heavy payloads to communicate with ground stations. Fundamentally, IoT networks are generally built on constrained resources and the UAV protocols are not suitable in the long term for IoFT environments since they use heavy payloads. So, the IoFT needs to be supported with IoT communication protocols to provide efficient communication between UAVs and IoT nodes.

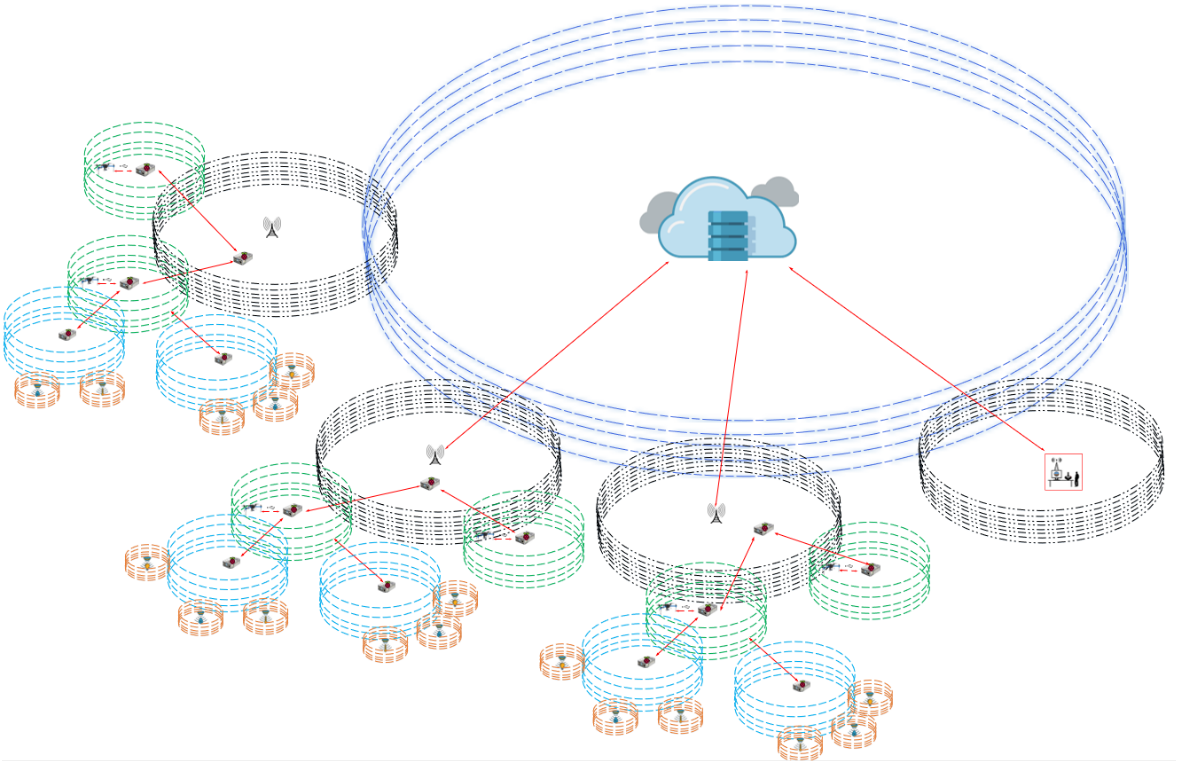


Figure 2.3 IoT nodes and UAV connections - 1

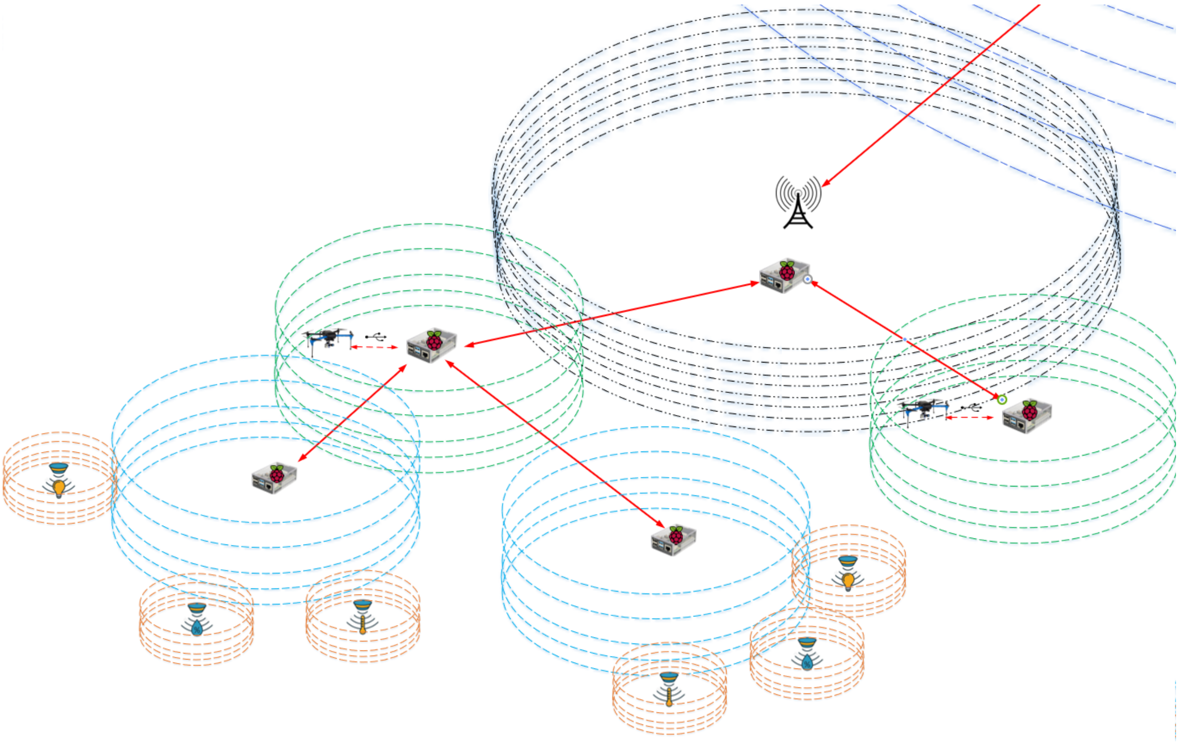


Figure 2.4 IoT nodes and UAV connections - 2

2.1. IoT Communication Protocols

One of the most popular IoT application layer communication protocols such as CoAP, MQTT, Advanced Message Queuing Protocol(AMQP), Extensible Messaging and Presence Protocol(XMPP), and ZeroMQ are varied and built on UDP and TCP network layer protocols. The main purpose of the protocols is to transmit messages in low-resource applications. Different research is presented to compare these similar protocols. Paridhika et. al. [11] compared the performance of CoAP, MQTT, and XMPP in the smart parking application domain. By measuring their response time, the researchers obtained the result that the CoAP performs better than MQTT and XMPP at lower usage of the server. In the contrary, if multi-threading is supported, XMPP is better than others.

The other research is presented by Naik [12] to compare IoT communication protocols. The MQTT, CoAP, AMQP, and HTTP are compared and CoAP performs better in many network metrics. However, MQTT is still the most preferred technology among the other protocols for IoT applications. Tightiz et. al. [13] analyzed the protocols from the perspective of a smart grid and the Data Distribution Service(DDS) and ZeroMQ emerged among all other protocols due to security and reliability characteristics.

Anusha et. al. [14] reviewed the AMQP, CoAP, XMPP, MQTT, MQTT-SN (MQTT-Sensor Network), and DDS with under packet loss, message size, bandwidth consumption, and latency metrics. CoAP and XMPP have the highest packet loss rates than other protocols when the actual telemetry loss rate ratio is increased. However, CoAP has better results on the latency of packets since it is built on UDP.

The MQTT is the most common preferred protocol of any others in IoT application domains [15, 16]. The main reason for more implementation of MQTT is that it has a more flexible structure than other protocols and is more eligible for IoT environments. Also, since it was developed earlier than others, the optimization and standardization parameters are better when it is compared with other IoT protocols. A brief comparison summary of IoT protocols is presented in Table 2.1.

Protocol	Transport Layer	Messaging Type	Architecture	Security
CoAP	UDP	Request/ Response	Client/ Server	DTLS, IPSec
MQTT	TCP	Publish/ Subscribe	Client/ Broker	TLS/SSL
XMPP	TCP	Publish/ Subscribe	Client/ Server	TLS/SASL
AMQP	TCP	Publish/ Subscribe	Client/ Broker	TLS/SSL IPSec SASL
ZeroMQ	UDP	Publish/ Subscribe	Client/ Broker(Opt.)	TLS/SSL

Table 2.1 Summary of the IoT protocols

2.2. MQTT (Message Queuing Telemetry Transport)

The MQTT protocol, released by IBM in 1999 and standardized by OASIS in 2013, is the most popular IoT protocol which has novel publish and subscribe architecture [17]. The main focus of the protocol was built as an extremely lightweight. The efficient bandwidth usage of the protocol provides to connect devices where the resources are constrained. The use-case of MQTT has spread very wide variety in many application and research areas.

The MQTT runs over TCP/IP and messaging relies on the topic structure. It allows multiple bidirectional connections for IoT devices. It has two different client types which are named publisher and subscriber. Also, it has a server known as the broker for MQTT which is used to exchange messages between clients through topics. The publisher clients transmit messages to the broker with publish mechanism into the specific topics which are defined by the user. The subscriber clients receive the message by subscribing to the specified topics. The general publish/subscribe architecture is given in Figure 2.5.

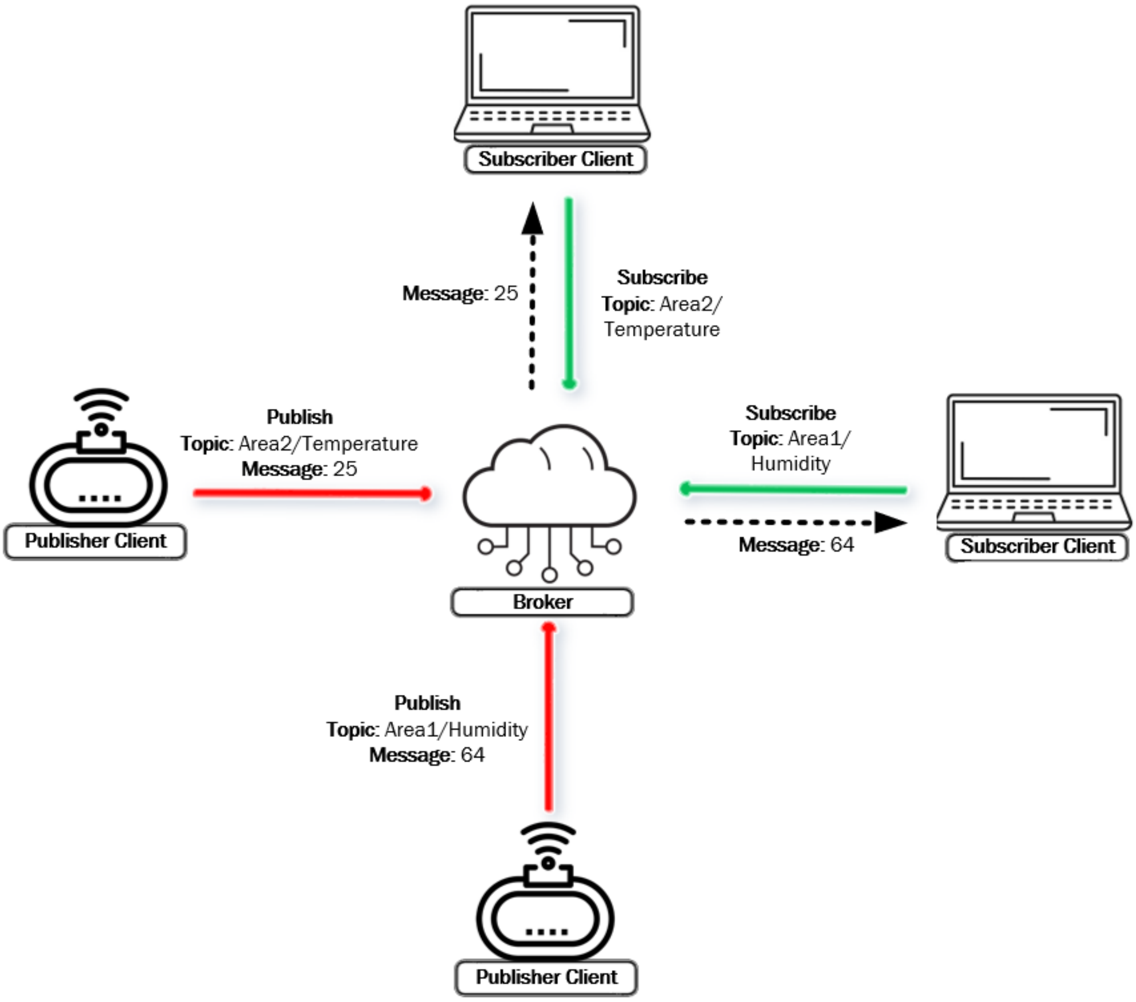
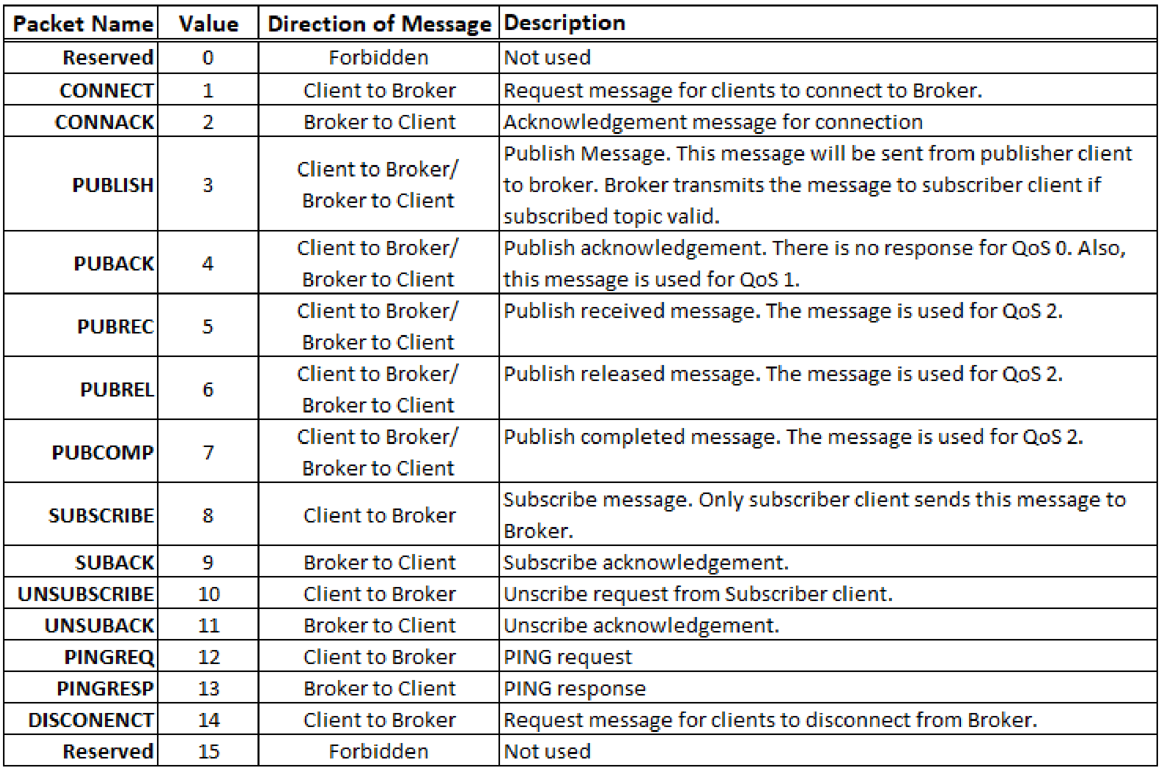


Figure 2.5 General MQTT Architecture

MQTT present 3 different Quality of Services (QoS) levels. The QoS 0 is the fire and forget level. The message delivery is not guaranteed since the message is sent once. The QoS is generally a good option where the connection is reliable or the packet loss is acceptable. The QoS 1 level is guaranteed packet delivery but it is not guaranteed duplication of data. The last QoS level 2 is certainly transmitting messages only once. Besides, QoS 2 is guaranteed the delivery of messages to the broker. But, the trade-off for QoS 2 is that the time consumption and workload are higher since it tries to send a message and waits for the acknowledgment message from the broker. However, it offers a more reliable connection where packet loss is not tolerated.



Packet Name	Value	Direction of Message	Description
Reserved	0	Forbidden	Not used
CONNECT	1	Client to Broker	Request message for clients to connect to Broker.
CONNACK	2	Broker to Client	Acknowledgement message for connection
PUBLISH	3	Client to Broker/ Broker to Client	Publish Message. This message will be sent from publisher client to broker. Broker transmits the message to subscriber client if subscribed topic valid.
PUBACK	4	Client to Broker/ Broker to Client	Publish acknowledgement. There is no response for QoS 0. Also, this message is used for QoS 1.
PUBREC	5	Client to Broker/ Broker to Client	Publish received message. The message is used for QoS 2.
PUBREL	6	Client to Broker/ Broker to Client	Publish released message. The message is used for QoS 2.
PUBCOMP	7	Client to Broker/ Broker to Client	Publish completed message. The message is used for QoS 2.
SUBSCRIBE	8	Client to Broker	Subscribe message. Only subscriber client sends this message to Broker.
SUBACK	9	Broker to Client	Subscribe acknowledgement.
UNSUBSCRIBE	10	Client to Broker	Unscribe request from Subscriber client.
UNSUBACK	11	Broker to Client	Unscribe acknowledgement.
PINGREQ	12	Client to Broker	PING request
PINGRESP	13	Broker to Client	PING response
DISCONENCT	14	Client to Broker	Request message for clients to disconnect from Broker.
Reserved	15	Forbidden	Not used

Table 2.2 List of the MQTT messages

The packet format for MQTT consists of 3 main sectors: Fixed Header, Variable Header, and Payload. The Variable Header and Fixed Header sectors are flexible and not always presented. But the Header size is fixed and always presented. Control Header and Packet length are part of the Fixed Header. The Control Header includes the message type which is given in Table 2.2 and the control flag to introduce a message for the client or broker. The control flag is used to specify the QoS level for publish message and it is not used for other message types. The packet length is used to indicate the payload length sum of the Variable Header and Payload. The Variable Header is not always used in MQTT packets. It provides to specify packets and include protocol names. The payload carries the data to be sent between the client and broker. The general structure of the MQTT packet is given in Figure 2.6

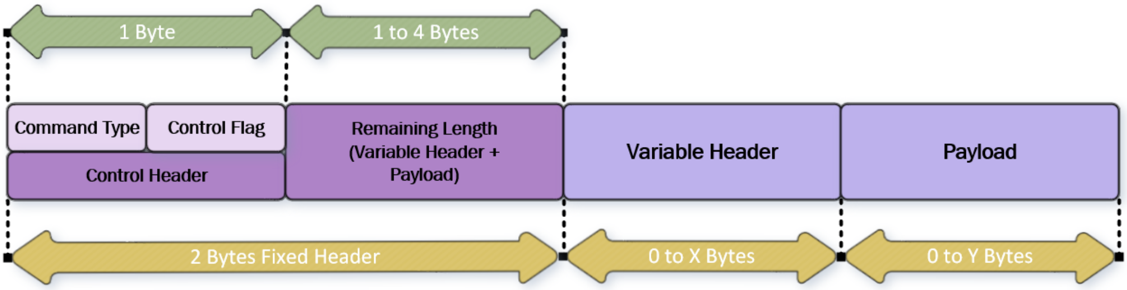


Figure 2.6 MQTT General Packet Structure

2.3. UAV Protocols

The protocols for the low-cost UAVs are mostly developed as on open-source projects. Many of them aim to establish communication between UAVs and ground stations. In preliminary, the management of the UAV is provided by the remote controller with the RF signals from Remote Controller by giving manual inputs to adjust rotor speed levels. In this situation, the ground station is only used to monitor drone status, and protocols are often dedicated to receiving status. With the development of the autopilot stage, the UAV has become a more robotic device and predefined activities such as takeoff, arm, land, etc. could be provided by the ground station. Thus, the protocols are also evolved to manage and control UAVs.

Many companies design their own protocol architecture since the applications of UAVs are more used for military or commercial purposes. In this type of application, communication infrastructure is called a Data Link which can be different variations. The most common types are presented in Figure 2.7. These types of systems benefit Radio Frequency (RF) signals to reach long-distance communication for sight-line operations. However, the RF is only able to transmit signals between the ground station and UAV in the visible range. Thus, satellite communication is the other option. The UAVs transmit the data to the satellite and then the ground station receives signals from the satellite.

Besides, open-source protocols have been developed to support research on UAVs. Also, these open-source protocols are not limited to specific architectures. But, the open-source application layer UAV protocols are very limited. Khan et. al. [18] presented three different protocols with their pros and cons. Furthermore, the first one is UranusLink which is designed for RF signals. The second one is the UAVCan which is designed for basic usage and is not useful for mission-critical systems. The most negative side of UranusLink and UAVCan protocols is that they have been developed for small overhead. The last one is the MAVLINK which is more widely known and has a more mature packet structure. MAVLINK is more stable than UranusLink and UAVCan. Also, MAVLINK supports many application layer protocols and has a multi-programming language.

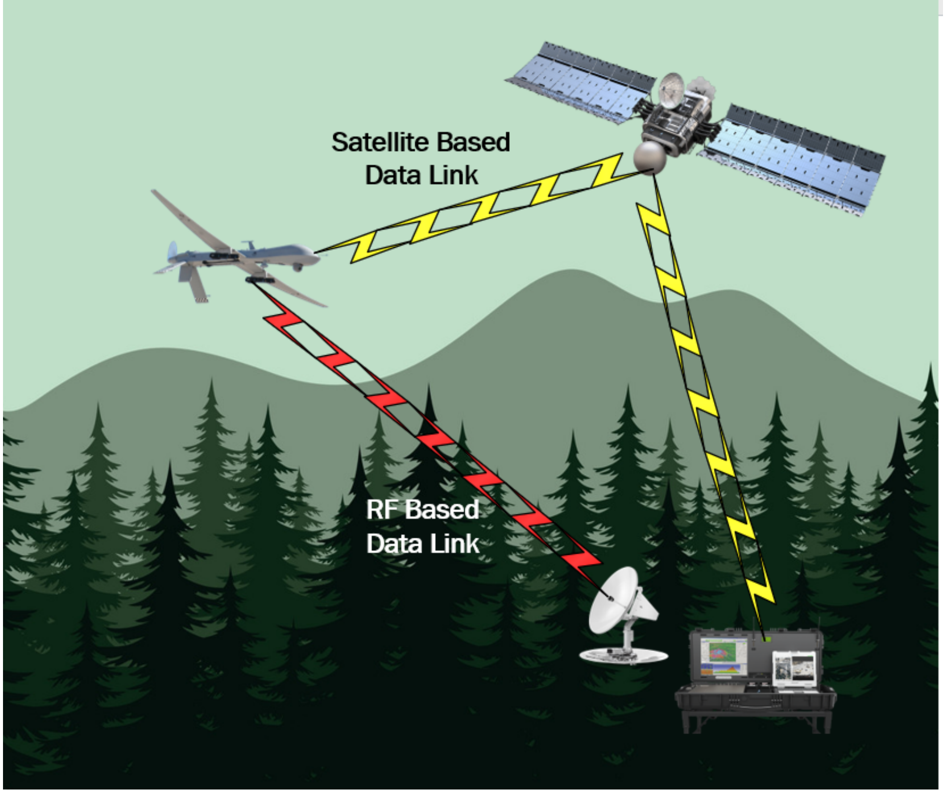


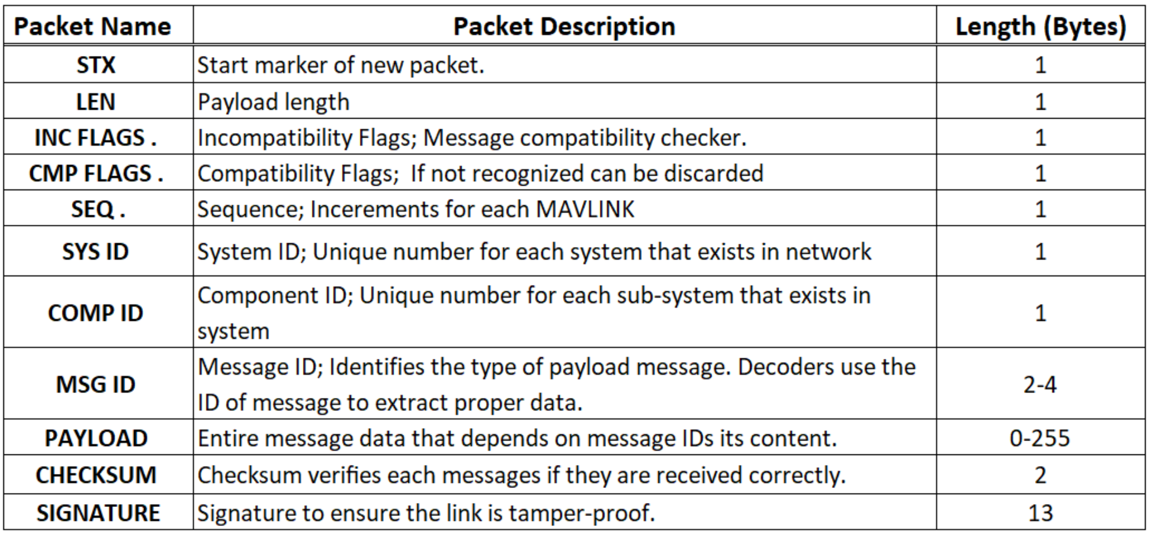
Figure 2.7 UAV Communication - Data Link

2.4. MAVLink

MAVLink which stands for micro air vehicle link is a protocol that provides communication between unmanned systems and ground stations. MAVLINK uses minimal overhead to provide lightweight communication. The approach of the protocol benefits serialization and supports different layers of message transportation [19]. The typical network interfaces like

UDP and TCP are also usable for MAVLINK to transmit messages. UDP provides more fast messaging while TCP is more reliable, but the management of connections becomes more difficult.

The typical message structure of MAVLINK depends on binary serialization. The serialization converts the messages into binary series which is given in Table 2.3 and transmits over different layers. The packet frame of MAVLINK includes the message types to indicate which messages are delivered. Also, the message types are divided into command and status messages. The status messages provide information about UAVs. The command messages are used to manage or initiate predefined actions like take-off, arm, disarm, etc. The frame of the MAVLINK is given in Figure 2.8.



Packet Name	Packet Description	Length (Bytes)
STX	Start marker of new packet.	1
LEN	Payload length	1
INC FLAGS .	Incompatibility Flags; Message compatibility checker.	1
CMP FLAGS .	Compatibility Flags; If not recognized can be discarded	1
SEQ .	Sequence; Incerements for each MAVLINK	1
SYS ID	System ID; Unique number for each system that exists in network	1
COMP ID	Component ID; Unique number for each sub-system that exists in system	1
MSG ID	Message ID; Identifies the type of payload message. Decoders use the ID of message to extract proper data.	2-4
PAYLOAD	Entire message data that depends on message IDs its content.	0-255
CHECKSUM	Checksum verifies each messages if they are received correctly.	2
SIGNATURE	Signature to ensure the link is tamper-proof.	13

Table 2.3 MAVLINK Packet Format

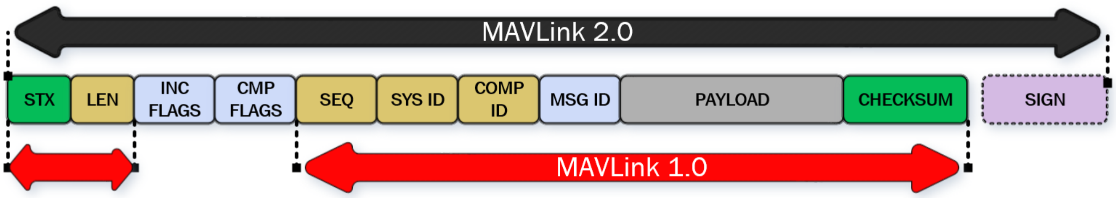


Figure 2.8 MAVLINK 1.0 and 2.0 Packet Frame

3. RELATED WORK

In this chapter, different studies are presented which are eligible to solve the IoFT communication problem and manage to drone over IoT protocols.

In the few last years, the research area for the domain of IoT and UAVs has increased and expanded significantly as non-terrestrial networks are much more popular nowadays. For the UAV-enabled IoT network, the communication between IoT devices and unmanned systems includes many different scenarios. However, most of the time, in related studies architecture of their systems and information about the management of the highly heterogeneous network are not given in detail which causes an insufficient review of the network requirements and scalability factor from the perspective of communication protocols. For example, Boursianis et. al. [20] reviewed IoT and UAV-assisted applications and prepared a survey on different aspects. The researchers described some different metrics, and technology and included different metrics in IoT and UAV-assisted agriculture application domain. Moreover, the researchers cited that they analyzed a total of 65 different papers. However, the researchers even mentioned to IoT protocols the participation of the UAV is not well covered. Also, the total Scopus database presents 865 total papers from 2004 to the research published day while the contributors do not go into detail about the UAVs management in IoT networks with the communication protocols.

A similar issue can be observed in different application domains such as smart cities [21, 22] disaster management[23, 24], virtually augmented reality [25, 26], edge computing [27, 28] or data transferring/gathering in different formats[29–33] and much more different Machine to Machine (M2M). The summaries of the research show us that the communication protocol is not well-defined to specify the UAV management in IoT networks. According to [34], the drone term should be kept as ”things” in given domain sets. The researchers are demonstrating the whole process where both terms are included. The key factor of the paper is that the introduced framework supports IoT-UAV-based applications. However, the given aerial networking is not yet defined as the whole process of the UAV-supported network to manage and orchestrate the very-wide spread network without the detailed aerial protocol. Only configurations are presented like MANETs, VANETs, and FANETS grouped under three main types: Proactive, Reactive, and Geographic.

Year	Title	Evaluation Metrics	Employed Protocol(s)	Key features perspective of IoFT network
2022	Development of an Efficiency Platform Based on MQTT for UAV Controlling and DoS Attack Detection	• Latency • Network • Memory Usage	MQTT	• Implementation of authentication and SSL/TLS certificates • ML model to detect DoS attacks on the UAV network
2017	Remote Wireless Sensor Network Range Extension using UAVs with Thread Protocol	• Latency • Power consumption	Thread	• Lower power consumption when compared to the current UAV protocols • Remote Control from a distance over UDP • Extending its range using radio telemetry
2017	Industrial Internet of things at work: a case study analysis in robotic-aided environmental monitoring	• Network joining time • Memory • Power consumption • Packet Loss	CoAP	• Revealing that the IoT devices on the UAV platform are more efficient way for managing them • Proves that the IoT protocols play essential roles in gathering data with the UAV
2019	Cellular UAV-to-X Communications: Design and Optimization for Multi-UAV Networks	• Uplink sum-rate	Sense-and-send protocol	• Proposed new algorithm that optimizes Multi-UAV networks • New communication technique is proposed to provide communication between UAVs and
2015	Integrating UAVs into the Cloud Using the Concept of the Web of Things	• Response time	HTTP	• Considering UAVs s a part of the Web of Things • Creating web services' resources for UAVs • Developing a server and clients to integrate UAVs into the cloud
2018	Ultra-Reliable IoT Communications with UAVs: A Swarm Use Case	• Reliability • Polling Delay	CoAP MQTT MAVLINK	• Proves that IoT technologies are eligible to manage drone swarm • States that IoT protocols are more reliable and provide efficiency in terms of latency
2020	EdgeDrone: QoS aware MQTT middleware for mobile edge computing in opportunistic Internet of Drone Things	• Average packet delivery • Latency • Memory • Bandwidth utilization • Power consumption	MAVLINK MQTT MQTT-SN	• MQTT and MQTT-SN are used to transfer messages to the UAV in mobile edge • The energy consumption of IoT protocols is much better • The performance of the MQTT and MQTT-SN has been analyzed in terms of UAV
2017	A Distributed Architecture for Unmanned Aerial Systemsbased on Publish/Subscribe Messaging and Simultaneous Localisation and Mapping (SLAM) Testbed	• Latency • Bandwidth utilization	MAVLINK MQTT DDS	• Shows that DDS is also suitable for UAVs • The MQTT and DDS are developed to communicate with distributed UAVs • IoT protocols are strong candidates for a large number of UAVs
2022	Microservices for autonomous UAV inspection with UAV simulation as a service	• Response Time • Load test	MAVLINK Fast DDS	• Building network architecture for UAVs with cost-effective, energy-efficient, and robustness • Developed cloud-based microservices application for UAV network
2022	DTUAV: a novel cloud–based digital twin system for unmanned aerial vehicles	• Transmission time • Packet size	MQTT MAVLINK	• The IoTF is transferred to virtuality to create a digital twin • Proposed method is tested under different network infrastructures such as 4G, 5G, WiFi, Wired networks, Cloud

Table 3.1 Summary table of related works about IoFT protocols

As a result of this, the complicated structures cannot be overviewed with the combined communication protocols which suit IoT devices and UAVs. The different studies are proposed to provide solutions to a given problem set by using UAV and IoT-based architecture. However, the main focus of these studies is solving the problems not comparing protocols or handling communication architecture. Under these considerations, in this thesis, the related work reviews are separated into 2 sections which are Multiple Protocol Based Architectures and Single Protocol Based Architectures. The purpose of dividing the 2 major sections is to show protocol differences in their own domains and IoFT domain. Also, the summary of related works is given in Table 3.1

3.1. Multiple Protocol Based Architectures

The multiple protocol based studies for UAV-IoT communication on at least two different protocols are presented in this section. Many studies benefit from different protocols for their application domains. Also, the majority of studies focus more on applications and scenarios without giving any details of network architecture [35–37]. The swarm technologies are much similar to the IoT network architecture and studies published about managing UAV swarms with IoT principle [38, 39]. Both terms need to deal with communication problems about a large number of devices.

Regarding the IoFT structures, many architectures employ different protocols since the UAV and IoT devices do not communicate with each other. For example, Yuan et. al. [40] designed architecture to provide UAV swarm communication in the 5G domain. The testbed consists of 30 different devices to reflect real-life conditions. The architecture consists of software and hardware designs. Also, in software design, the CoAP and MQTT protocols are suggested for the embedded operating system. But, the MAVLINK is used to manage or control UAVs for data encapsulation in software design. The main key feature of the design is that it provides more reliable and efficient connections in cellular networks.

Mukherjee et. al. [41] reviewed the Internet of Drones Thing (IoDT) concept and they developed message transfer with enhanced MQTT and MQTT-SN to provide communication in edge computing. Also, they state that topology estimation is needed since the mobility of drones is higher than standard IoT architecture. For this reason, they prepared an estimation algorithm to provide reliability. By using an enhanced message transfer strategy, they aim to decrease forwarding latency between routers and provide efficient bandwidth utilization. They review the performance of the proposed approach in various metrics such as average packet delivery, latency, memory, bandwidth utilization, and power consumption. They achieve 30% better performance with enhanced MQTT-SN and 17% with enhanced MQTT for average packet delivery. Furthermore, the MQTT-SN uses minimum energy consumption when compared to other strategies. Experimental result shows that MQTT-SN with the proposed algorithm has better performance for memory usage and bandwidth utilization.

Saib's book [42] reviewed publish/subscribe mechanism on distributed UAVs architecture. The message strategy of MQTT and DDS provides for managing UAVs in vision-based Simultaneous Localisation and Mapping (SLAM) system. Since the complexity and heterogeneous structure of these type of network is enormous level, the researcher states that MQTT and DDS publish/subscribe mechanism serve better to overcome communication problems. Also, the suitability and scalability factors can be enhanced by using IoT protocols in a distributed environment. To build infrastructure between IoT devices and UAVs, MAVLINK is employed. The latency and bandwidth utilization are measured for the evaluation criteria. But, the observation is obtained internally by measuring given metric sets.

Airborne measurements, inspections, or surveillance missions with UAVs are very common. Matlekovic et al [43] designed a cloud system that provides to inspect infrastructures with planning paths and assigning the task to UAVs. Microservices are defined to manage UAVs and requested tasks. The complexity of the design tried to be under control by defining different functional requirements and non-functional requirements. For providing communication between all nodes and transferring obtained data to the web interface, Fast DDS is operated. On the UAV side, the MAVLINK message protocol is supporting the architecture. The evaluation criteria from the perspective of IoFT are provided by measuring a response time and load test.

The digital twin concept is another important topic in the last few years. Meng et. al. [44] create a digital twin of the UAV-based IoT environment. The test scenarios include different network types such as WiFi, 4G, and 5G. Both real and virtual frameworks consist of MAVLINK and MQTT to establish a connection between IoT nodes and UAVs. The frameworks are evaluated with the transmission time and packet sizes. The researchers emphasize that creating a cloud-based digital twin of UAV systems makes them more intelligent.

3.2. Single Protocol Based Architectures

In this section, the single protocol based architectures are presented. Vangimalla et. al. [45] proposed a system architecture that aims to solve interoperability problems that exist in constrained resource areas by using THREAD protocol for UAVs. Thread protocol is useful regarding cost-effectiveness, low-power consumption, and reliability. It has no application layer which encourages the researchers to build their own protocol. The network layer of Thread protocol consists of UDP, IP Routing, and Low-Power Wireless Personal Area Networks (6LoWPAN). The implementation of the paper shows that the proposed method uses a Thread protocol between UAVs and IoT devices. The management of UAVs is still provided with MAVLINK.

Silva et. al. [46] introduced MQTT-based UAV control architecture. The researchers deployed MQTT. MQTT was used to receive commands from the controller and to transmit the status of the drone to the controller. The evaluation criteria of the architecture have three metrics: Platform Latency, Network, and Memory Tests. The measurement consists of Ethernet and WiFi connections. The researchers suggested a new ML model that prevents DDOS attacks for this type of architecture. The other contribution of the paper is data balancing and Bayesian optimization. However, the main problem of the research is the lack of comparison of their architecture with similar protocols and is more likely to test MQTT performance in UAV communication. Also, the key point of the research is the deployment of MQTT for the management of UAVs instead of solving communication problems in IoFT.

Another research which is presented by Scilimati et. al. [47] proves that the IoT protocols are much more effective where UAVs and IoT devices are incorporated. The researchers built a test environment in Robotic Environment System (ROS) and the testbed consists of IoT-aided UAV systems. To provide IoT communication and gather IoT sensor data with UAV, CoAP is employed with a bunch of message sets of PUT and GET. According to the network joining time, memory usage, and packet loss ratio, the researchers state that the performance of CoAP is sufficient regarding data gathering and message delivery. However, the study does not include any comparison of other protocols or proof of sufficient criteria. Furthermore, the architecture of CoAP is built only with PUT and GET messages which are useful to gather data, but it does not consider receiving the status message of the UAV. The scenarios are created to send only commands to UAVs and prepare mission items for them.

Integration of UAVs into the cellular networks is studied by various researchers [48–50]. Generally, UAVs were handled point of router protocols or case studies view in these studies. Zhang et. al. [51] proposed a sense-and-send protocol by comparing networks between UAV-to-UAV, UAV-to-infrastructure, and UAV-to-anywhere regarding a sum of up-link rates. Also, they proposed an iterative sub-channel allocation and speed optimization algorithm (ISASOA) that organizes multi-UAV-based networks which run 10% better than a greedy algorithm. However, this study has only been limited to gathering data through cellular networks in terms of the IoFT.

Web services incorporating UAVs are also another popular topic. This topic is researched in different studies [52, 53]. Mahmoud et. al. [54] used HTTP protocol to integrate UAVs as a part of the cloud. By specifying HTTP's RESTful Application Programming Interface (API) and Request/Respond mechanism, the researchers aimed to adapt UAVs to the Web of Things. The prepared architecture to communicate UAVs over web services consists of the command and status messages of UAVs. The researchers present response time results for each HTTP message. Even though the main purpose of the paper is that integrate UAVs into the web domain, the paper has not sufficient evaluation criteria to prove managing drones over web services is efficient or reliable.

4. PROPOSED METHOD

The base problem of the protocol is the assumption that UAVs should be considered like IoT nodes. Basically, the UAVs should have two vital functions: Receive Commands and Transmit Status. These two basic principles are managed with the ground station. Thanks to improvements in autopilot technology, many UAV behavior such as takeoff, land, cruise, etc. can be managed without manual input [55]. In other words, the UAV rotors are controlled through Remote Controller (RC) such as a joystick with a different signal. However, the direct inputs to provide managing on the drone are not needed anymore for predefined actions. The flight modes such as off-board are providing different predefined actions for specific tasks.

Accordingly, the ground stations which are more focused the managing drones in visual ways and interactions with MAVLINK protocol are not effective when it comes to scenarios involving hundreds of UAVs and different locations. As mentioned in the background section, the cloud is necessary to manage multi UAVs based networks such as swarm. Although, the MQTT and other preferable IoT protocols are employed to deal with multi-vehicle networking problems. Nevertheless none of them is not directly aiming to consider UAVs as IoT nodes. The main concern is managing the swarm, rather than including each UAV as part of the chain.

In the light of this information, the protocol should have a design that provides communication with the UAVs while it can be useful for the IoT network [56]. The main focus of the proposed method is re-configuring the MQTT with a MAVLINK interpreter to deal with the long-distance UAV communication within the IoT environment to provide communication among UAVs as the IoT node. The main principle of this perspective is preserving the UAV as an IoT node and closing the gap in the protocol differences when the UAVs are needed to manage from the end-user in the IoT network. Because the IoFT term includes both systems of the UAVs and IoT sensor networks to achieve better distances. However, the problem of the researchers is dealing to design 2 different network architectures to overcome the problem. One of them is needed to communicate with the IoT nodes while the other one is needed to manage or communicate with UAVs.

The MAVLINK is the most common protocol among UAV management in various tasks. It is specialized to communicate with ground-station and UAVs with RF signals. As the requirement of the communication is changed at some point, the TCP/UDP packets also support the MAVLINK frame. However, because of the nature of MAVLINK, it is really talkative protocol than the IoT protocols since the main aim of the protocol is directly controlling the drone, and the design approach of the protocol is only managing the UAVs and receiving the status of UAV just in time. Even though the protocol provides communication with multiple types of UAVs and it assigns unique identities for each of them, the cost of managing all of them is a really challenging situation as it has many packet frames in a couple of seconds. The router or different types of low-cost methods can be used to achieve long-distance communication. But as mentioned before the fact is that each hop of the network will demand more resources for each one of them. Also, MAVLINK is not tolerating constrained resources.

For this issue, we provide the MAVLINK enabled MQTT (MQTT-XT) to provide more flexibility for both systems where the MAVLINK is not needed to be considered to manage drones anymore in the IoT network by embedding it in the UAV platform as MAVLINK interpreter. The architecture of the MQTT allows one to specify the topics and separates them for specific tasks. The publish and subscribe structure of the MQTT can be altered to combine the MAVLINK frame which is preferred to manage the UAV system. These specific topics are configured to manage, command, receive status, and other different functionalities which are previously provided by MAVLINK. The MQTT allows to customize and organize topics. The MQTT is designed on top of the TCP layer where it has its own abilities for the publish-subscribe mechanism. In this thesis, we built our protocol with auto-generated topics which aim to manage drones with specific topics. This layer is illustrated in Figure 4.1

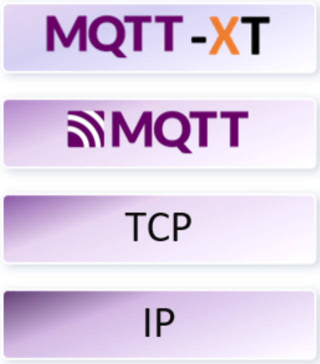


Figure 4.1 MQTT-XT Layers

Our next-generation structure of communication model design offers to decrease the workload of the network by moving the MAVLINK packet structure to the closest point to the UAV. The communication between the client-server is re-arranged to transmit UAV status to the end-user and to allow command from the end-user. The general structure of our communication protocol is given in Figure 4.2.

The architecture of the proposed method stands on three different client types and one cloud server. The protocol waits for the connection signal from the end user. Each client component of the protocol design has its own responsibilities. Each type of the MQTT-XT component and their summarized functionalities are given below:

- **Ground-station Client (End User):** Typical ground-station that provides to connect, send commands, and receive the status of UAVs which are connected to the cloud.
- **Cloud (Server):** The cloud server that provides to establish the connection between Activator client and End-User with its specific topics
- **Activator Client (FoG Area):** This client provides to activate communication that is not reachable by default for the end-user. It has its own broker that provides a connection point for its area. Also, this client has some novel functions such as QDoS and PSMI.

- **UAV Client (UAV Platform):** UAV client is deployed on the UAV Platform and its main aim is that interpret the MQTT message to UAV and UAV Messages to MQTT topics.

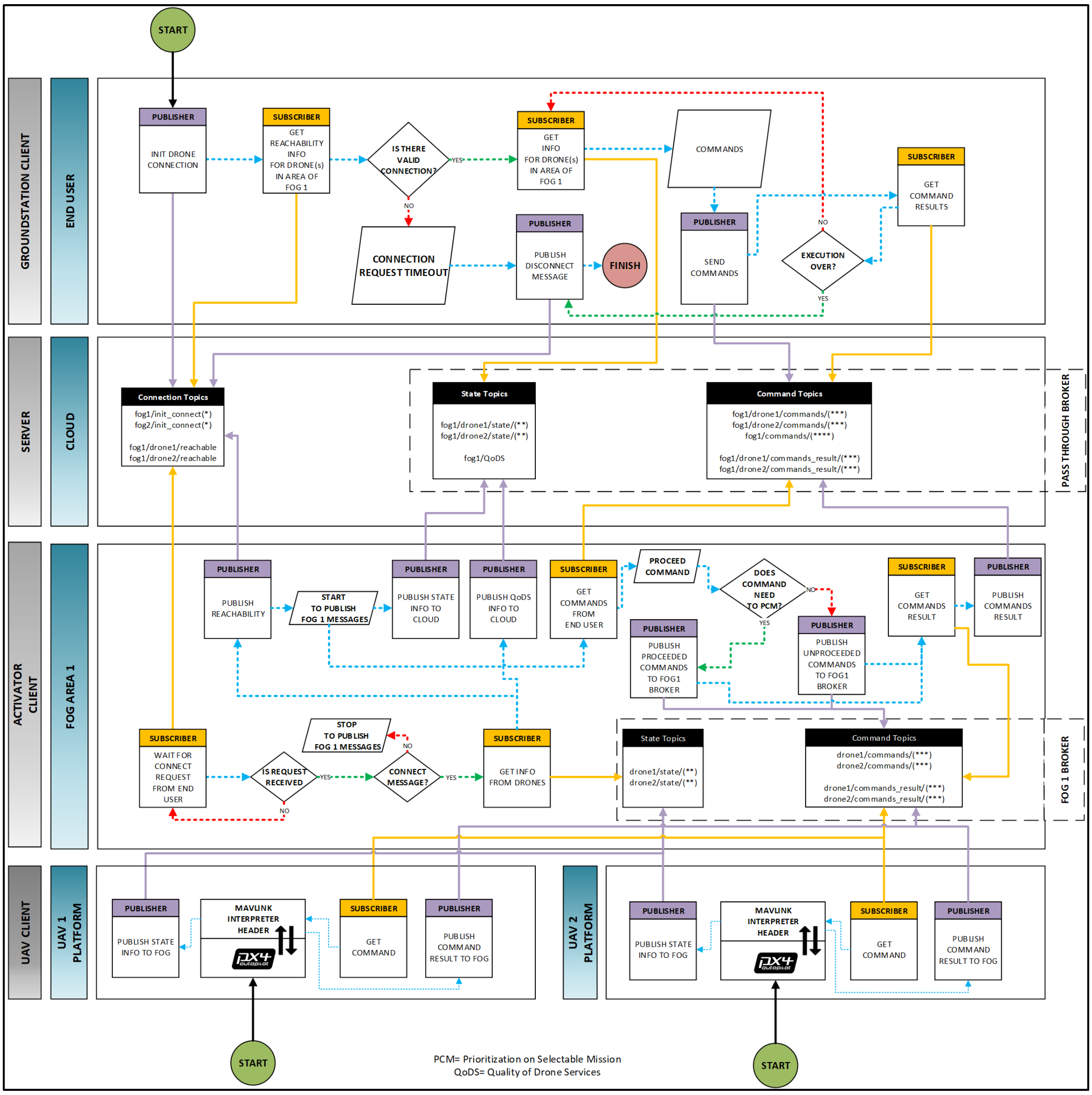


Figure 4.2 MAVLINK enabled MQTT (MQTT-XT) Top-Down level architecture flowchart

Since the motivation of the thesis is establishing communication of UAVs on the IoT network, we communication approach is designed to be more flexible that fit the requirement of the different areas. The key point of the architecture is the MAVLINK interpreter and auto-generated topics by clients. The subscribe and publish mechanism of the proposed protocol allows controlling UAVs more easily by generating custom topics over MQTT. In Figure 4.2, the beginning of the process is presented. The subscribe events are colored orange and publish events are colored purple. The sequential process of the protocol can be easily tracked with blue arrows. However, each type of component has a unique task to keep communication alive. Furthermore, even the backbone of the MQTT is preserved, as the long-distance communication is aimed the structure is varied than MQTT. To demonstrate each component functionalities' detailed processes are presented following subsections.

4.1. Ground-station Client (End User)

The typical strategy to manage UAVs stands on establishing communication between UAV and Ground-station [57]. As far as the communication or broadly known as data link is built, the user can interact with UAV. From the perspective of the UAV flight modes, some of the tasks require manual input from RC while some autonomous flight modes only require a command to perform predefined actions. Furthermore, these autonomous flight modes do not require active and robust communication as long as the commands are received once and it knows what to do.

For this reason, MQTT-XT protocol is designed for UAVs by considering the network connection is unstable like in many IoT environments. However, it is also useful where manual inputs are not required. The ground-station client is acting like the typical controlling center which is traditionally used for the command and control center illustrated in Figure 4.3

Unlike the traditional way, the end-user is connecting directly to the activator client in different areas instead of UAVs. Actually, the activator clients are a more vital role than the end-user in the perspective of the data link since the command and control are more vulnerable on cloud messaging. The reason for this approach, providing more efficient interaction with UAVs by providing a bridge on the edge of the FoG. A detailed explanation of the approach for reason is given under Activator client.

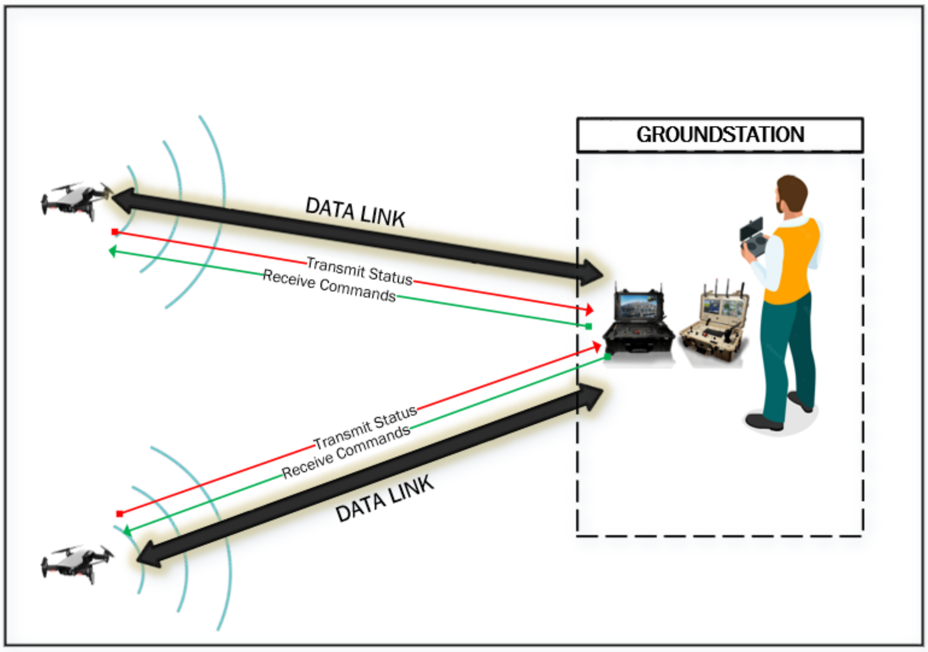
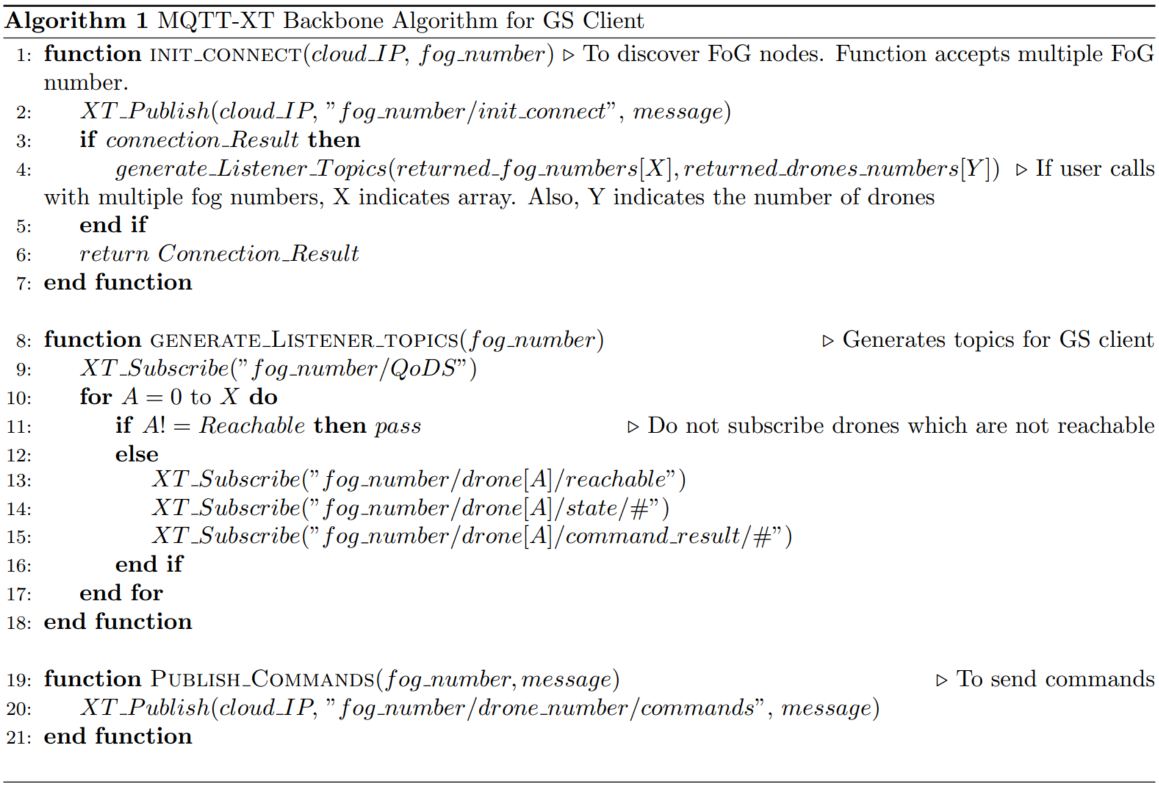


Figure 4.3 Typical communication between Ground station and UAVs

The ground-station (GS) client stands for 3 different publisher that allows sending message and 3 different subscriber that listens to topics on the cloud for each FoG area connection. The topics can be grouped under Connections, States, and Commands. The Connections Topic class provides a connection between GS client and activator client. Also, reachability information per drones that exist in the related connected area can be found under this group. For example, if the user wants to connect FoG-1 and FoG-2 areas, the protocol sends the connection message under "`fog[X]/init_connect`" and cloud responses this connect message under "`fog[X]/drone[Y]/reachable`" where the X indicates FoG ID and Y indicates the drone ID.

The topics are auto-generated on the cloud when the protocol is deployed. The user can easily connect the FoG with only FoG IP. The IP table is also available via the activator client. However, in order to ease case scenarios, the user interacts with the activator client instead of each drone. The backbone of the MQTT-XT algorithm for the GS client is given in Figure 4.4.

The State Topic includes the status of drones that are generated from autopilot and reported



```
Algorithm 1 MQTT-XT Backbone Algorithm for GS Client
function INIT_CONNECT($cloud\_IP$, $fog\_number$)  ▷ To discover FoG nodes. Function accepts multiple FoG number.
    $XT\_Publish(cloud\_IP,$ ”$fog\_number/init\_connect$”, $message)$
    if $connection\_Result$ then
        $generate\_Listener\_Topics(returned\_fog\_numbers[X], returned\_drones\_numbers[Y])$  ▷ If user calls with multiple fog numbers, X indicates array. Also, Y indicates the number of drones
    end if
    $return\ Connection\_Result$
end function

function GENERATE_LISTENER_TOPICS($fog\_number$)  ▷ Generates topics for GS client
    $XT\_Subscribe($”$fog\_number/QoDS$”$)$
    for $A = 0$ to $X$ do
        if $A! = Reachable$ then $pass$  ▷ Do not subscribe drones which are not reachable
        else
            $XT\_Subscribe($”$fog\_number/drone[A]/reachable$”$)$
            $XT\_Subscribe($”$fog\_number/drone[A]/state/\#$”$)$
            $XT\_Subscribe($”$fog\_number/drone[A]/command\_result/\#$”$)$
        end if
    end for
end function

function PUBLISH_COMMANDS($fog\_number, message$)  ▷ To send commands
    $XT\_Publish(cloud\_IP,$ ”$fog\_number/drone\_number/commands$”, $message)$
end function
```

Figure 4.4 MQTT-XT Backbone Algorithm for Ground station Client

by the activator client. The activator client automatically starts to publish the status of the drone when the connect message is received. The functions are defined to receive each status data which are received from all drones. When the user pulls the status data, cloud responses messages under "fog[X]/drone[Y]/state/[Status]" where the Status is a topic that indicates the different status of the drone (e.g. Location, Battery Percentage, Flight Mode, etc.).

The last topic which is named Commands Topics is used to provide giving the task to drones. Each command has its result in the same topic group. When sending a command, the end-user has two options. The first option is that give specific drones that exist in FoG. For this command set, the user sends a message under "fog[X]/drone[Y]/commands" topic and cloud responses this command under "fog[X]/drone[Y]/commands_result" topic. The second option is related to the FoG area. The protocol provides to receive one command that affects all nodes under FoG. To achieve this, the user can use a command to directly to the activator client. The command message will be broadcast for all nodes under "fog[X]/commands" topic and cloud responses this message for each drone again under "fog[X]/drone[Y]/commands_result" topic. These typical message requests and responses are given in Figure 4.5. The figure presents the typical connection, getting status, and giving the command.

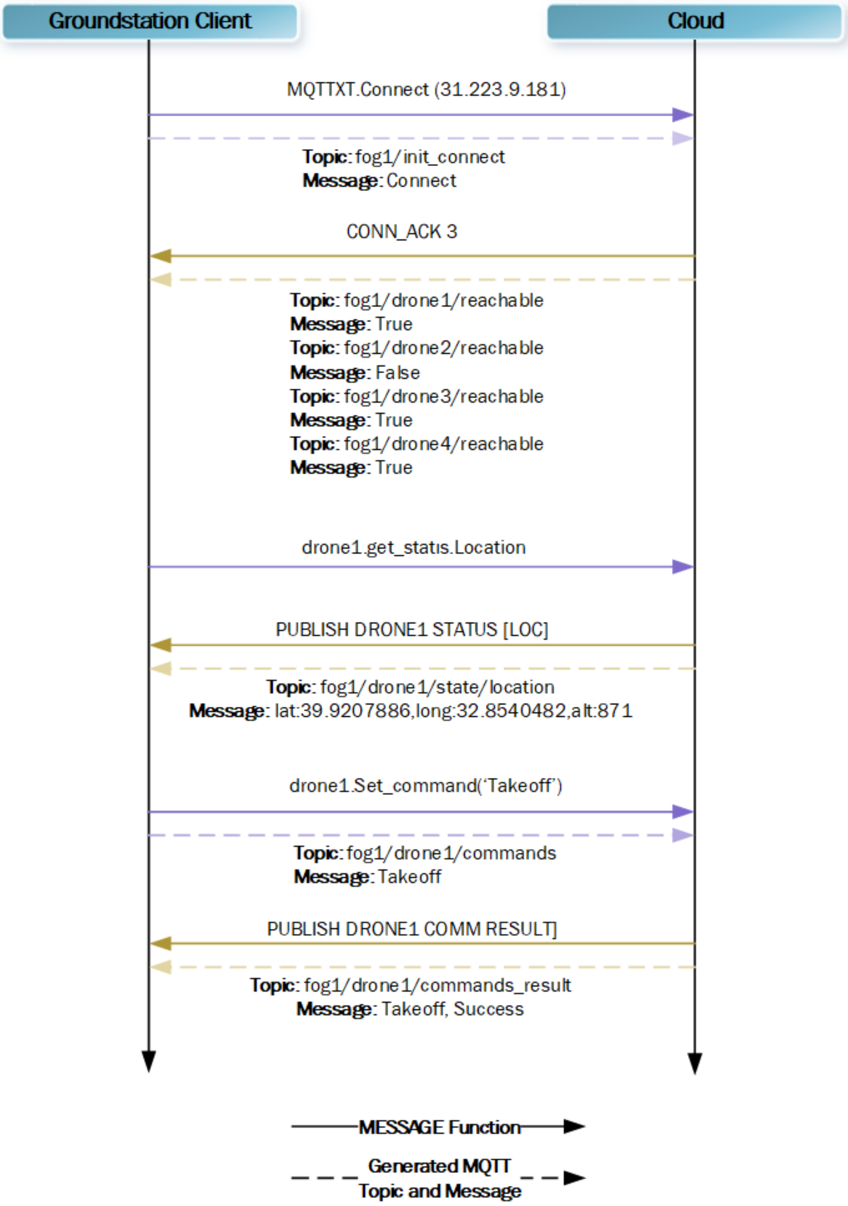

Groundstation Client
Cloud
MQTTXT.Connect (31.223.9.181)
Topic: fog1/init_connect
Message: Connect
CONN_ACK 3
Topic: fog1/drone1/reachable
Message: True
Topic: fog1/drone2/reachable
Message: False
Topic: fog1/drone3/reachable
Message: True
Topic: fog1/drone4/reachable
Message: True
drone1.get_status.Location
PUBLISH DRONE1 STATUS [LOC]
Topic: fog1/drone1/state/location
Message: lat:39.9207886,long:32.8540482,alt:871
drone1.Set_command('Takeoff')
Topic: fog1/drone1/commands
Message: Takeoff
PUBLISH DRONE1 COMM RESULT]
Topic: fog1/drone1/commands_result
Message: Takeoff, Success
MESSAGE Function
Generated MQTT Topic and Message

Figure 4.5 MQTT-XT Messaging between GS Client and Cloud

4.2. Server (Cloud)

The cloud component is providing a bridge between the end-user and activator clients. The main focus of the cloud side is preserving topics that are generated from end-user and activator clients. The topics are playing a vital role as the concept includes many different connections among the network. The FoG number which is connected can be various numbers as the main purpose of the protocol is to provide messaging between the activator client and end-user. In Figure 4.6, the different types of connection can be noticed.

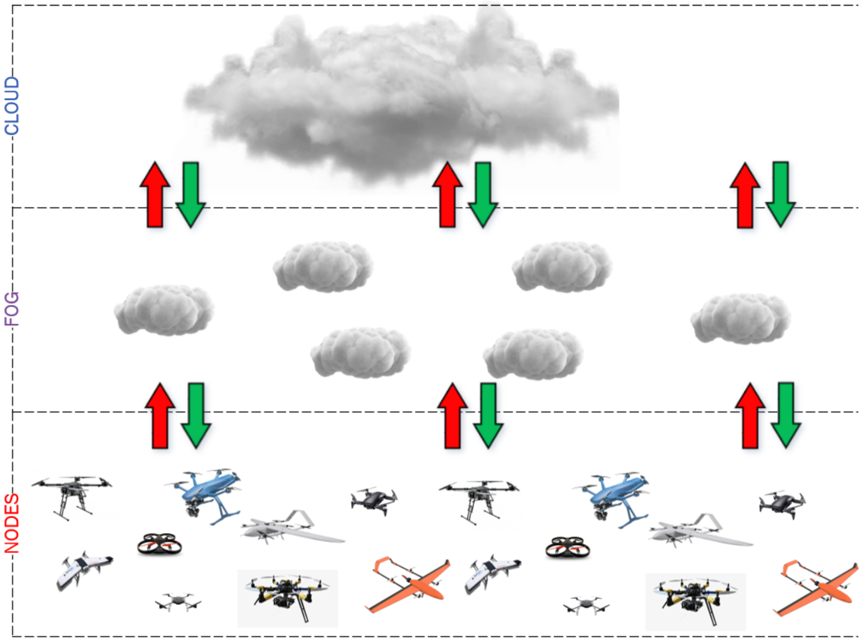


Figure 4.6 Cloud - Type of Connections

The topics are automatically generated when the connection is established. Thus, the cloud server settings are not needed as the activator client has its own server to provide services for the edge side. The cloud should be deployed where clients can communicate without any restrictions. Also, the messaging traffic will exponentially increase when the FoG node is added. The limitation of the network on the cloud is limitless when it is compared to the activator client. The activator client should have a more efficient load balance related to the connected drone number. The total network load for the activator client is calculated using the equation (1).

$$L(x) = \sum_{n=1}^{k} ByteStream_n \tag{1}$$

Where the k indicates the number of the drone, x indicates the FoG number, and L courier is used for the Total Load for FoG. Similarly, the total network load of the cloud can be represented with the equation (2).

$$TotalLoad = \sum_{x=1}^{t} L(x) \tag{2}$$

Where the t represents the number of FoG that is connected to the cloud. As can be seen in the equations, the load of the cloud is highly increasingly related to the FoG number. Furthermore, the resources of the cloud can be assumed as infinitely. Contrarily, the activator client has a limited process and connection capacity due to the decreased cost of the edge node.

4.3. Activator Client (Fog Area)

Activator client has the most vital role in protocol architecture design. The activator client has its own broker to communicate different UAVs. In other words, the broker is embedded for each FoG node. It receives the drone status and transmits them to the cloud. It also receives commands from the end-user client and publishes them to the related UAV clients.

The client has a communication infrastructure with two different servers. The cloud side has two subscribers and four publishers while the fog side has two subscribers and one publisher. The topic groups for the cloud are the same as the ones mentioned in the end-user client: Connections, States, and Commands. Except for the connection topic group, the states and commands groups are existing in the FoG broker.

For the connection topics, the activator client listens to the initialization from the end user. The name of the client comes from that function. The message publish mechanism for the cloud is not activated until the message received from "`fog[X]/init_connect`" topic. The approach for the communication between the cloud and activator client is reactive. The user needs to send connect message to activate communication. Also, the disconnect message means deactivating message streaming. As the connect message is received, the activator client publishes each drone reachability under "`fog[X]/drone[Y]/reachable`" topic.

The messaging between UAV client and the activator client is proactive. As the systems are alive, each UAV generates the status messages and publishes them under "`drone[Y]/state/[Status]`" topic. The activator client adds a tag with its ID and publishes the same messages under "`fog[X]/drone[Y]/state/[Status]`" topic as the connection message is received. Figure 4.7 presents the activation and deactivation of message traffic.

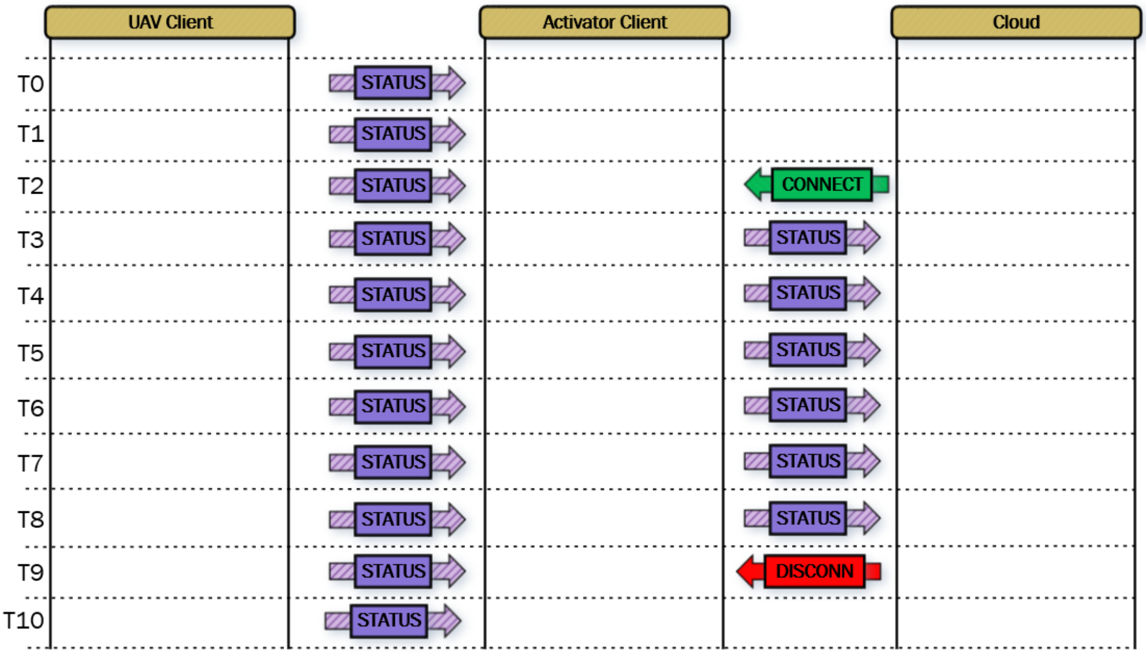


Figure 4.7 Activator Client Activation and Deactivation Message Traffic

For the command messages, the activator client listens to two different topics to process data. The first one is the drone-specified command messages under "fog[X]/drone[Y]/commands" topic and forwards the command by publishing under "drone[Y]/commands". The second one is received through "fog[X]/drone[Y]/commands" topic. When the command is not specified for the drone, the user can use "fog[X]/commands" topic. The activator client processes the command whether it is broadcast or Prioritization on Selectable Mission Item (PSMI). The command result message will be forwarded under "fog[X]/drone[Y]/commands_result" topic. The backbone of the MQTT-XT algorithm for the activator client is given in Figure 4.8.

For the common command message, related to its function, the activator client broadcasts or prioritizes the command. This function provides a more effective way to communicate with multiple nodes. For the broadcast message, the activator client publishes a message for each drone. However, the PSMI includes the calculation to assign a task for the specific drone.

4.3.1. Prioritization on Selectable Mission Item (PSMI)

The PSMI is a useful feature to manage and organize comprehensive tasks. The main aim is to create a decision support system for some commands. For example, if the user wants to send a drone to X location, for this request, the command message is received from "fog1/commands/goto" topic. If drone A location is assumed as Y and drone B as Z, for the geological distance calculation, the Haversine Formula (3) can be used [58].

$$\begin{gathered} a = sin^2\frac{\Delta\phi}{2} + cos\phi_1 \times cos\phi_2 \times sin^2\frac{\Delta\lambda}{2} \\ c = 2 \times atan2(\sqrt{a}, \sqrt{1-a}) \\ d = R \times c \end{gathered} \tag{3}$$

where the latitude is ϕ, longitude is λ and R represents earth's radius. The value for Y can be calculated as $d_{A\rightarrow X}$ and the value for Z can be calculated as $d_{B\rightarrow X}$. If the Z value is less than Y, the activator client will forward go to command for drone B. In this case, the PSMI value for ”go to” mission will be [2,1].

Algorithm 2 MQTT-XT Backbone Algorithm for Activator Client

```
function CLOUD_BROKER_LISTENER(cloud_IP)    ▷ Listens cloud broker to send commands
    if init_connection then
        cloud_connection = True    ▷ This variable provides activation to send status
        Publish_drone_reachable_states(cloud_IP)
        Publish_drone_status(cloud_IP)
        Publish_qods(cloud_IP)
    else
        cloud_connection = False    ▷ This variable provides deactivation to send status
    end if
    if command_message then
        if broadcast_message then
            if PSMI_message then
                XT_Publish(own_broker_IP, ”drone[A]/commands”, message)    ▷ The drone is selected with Haversine formula then Wait Command Result
                XT_Publish(cloud_IP, ”drone[A]/commands_result/[command_type]”, message)
            else
                for A = 0 to X do
                    XT_Publish(own_broker_IP, ”drone[A]/commands”, message)    ▷ Wait Command Result
                    XT_Publish(cloud_IP, ”drone[A]/commands_result/[command_type]”, message)
                end for
            end if
        else
            XT_Publish(cloud_IP, ”drone[A]/commands”, message)    ▷ Wait Command Result
            XT_Publish(cloud_IP, ”drone[A]/commands_result/[command_type]”, message)
        end if
    end if
end function

function PUBLISH_DRONE_REACHABLE_STATE(cloud_IP)    ▷ Send drone reachable states
    for A = 0 to X do
        XT_Publish(cloud_IP, ”fog_number/drone[A]/reachable”, message)
    end for
end function

function PUBLISH_DRONE_STATUS(cloud_IP)    ▷ Send drone states
    for A = 0 to X do
        XT_Publish(cloud_IP, ”fog_number/drone[A]/state/[data]”, message)    ▷ Data variable can be various. The topics are auto generated from received messages”
    end for
end function

function PUBLISH_QODS(cloud_IP)    ▷ Send QoDS values
    Analyze_Drone_Status()
    XT_Publish(cloud_IP, ”fog_number/qods”, message)
end function

function GENERATE_LISTENER_TOPICS()    ▷ Generates topics for Activator client
    for A = 0 to X do
        XT_Subscribe(”drone[A]/state/#”)
        XT_Subscribe(”drone[A]/command_result/#”)
    end for
end function
```

Figure 4.8 MQTT-XT Backbone Algorithm for Activator Client

However, the PSMI value is not only calculated based on the distance factor, the status of the drone is also included to calculate PSMI. The battery drainage to arrive at that location is the mostly used factor. For this reason, the estimated remaining distance for a drone is another filter. If the drone is not eligible to arrive at that location, the PSMI will be not listed for the end-user. Also, the distance between the commanded location and the drone location is not the only factor to estimate for PSMI. The protocol promises to be suitable for any kind of drone. So, the maximum groundspeed of the drone should be added PSMI calculation for the commanded location after filtering eligible ones. The PSMI values will be ordered with their time to arrive related to (4) formula.

$$TimetoArrive = \frac{d}{groundSpeed} \tag{4}$$

4.3.2. Quality of Drone Services

The Quality of Drone Services (QoDS) is the function that presents the best available UAVs which is ready for mission based on their status of them. The list is prepared by using the battery percentage and maximum ground-speed attribute. The QoDS is published under `"fog[X]/QoDS"` as the connection message is received. The user can call this list to see the best option for the task. The function provides decision support for a user similar to PSMI. Unlike the PSMI, the calculation is not related to any input from the user.

For the first step, the filter is defined to make a group of UAVs based on their battery percentage. The filter works by dividing tenths of a hundred for battery percentage. A total of ten groups are sorted for the first step. The higher battery percentage means to a higher QoDS order. The second filter is used to separate and sort the UAVs that are sorted into the same group. For this reason, the ground speed is used to sort. The higher ground speed takes first place in the list. The example filter and sort operations are given in Figure 4.9.

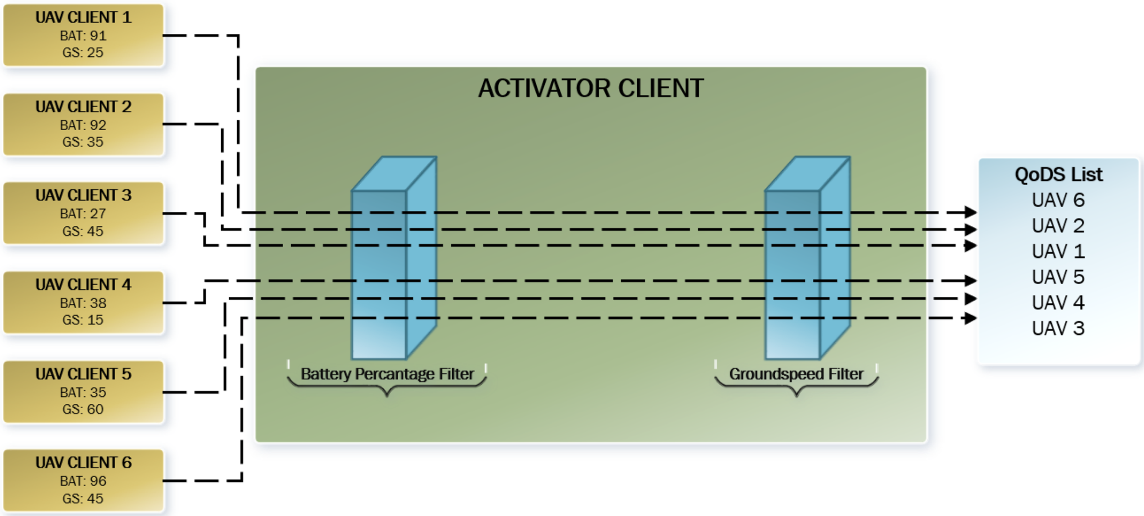


Figure 4.9 QoDS List Example

4.4. UAV Client (UAV Platform)

The last component is the UAV client which is placed on the UAV platform to communicate with Autopilot over the serial bus. The main attribute of the UAV client is the interpreting function. This function converts the MAVLINK message to the MQTT message and transmits the MAVLINK message through MQTT topics. The protocol automatically does the interpreting phase.

The MAVLINK message frame has a minimum packet length is 8 bytes while the maximum packet length can be up to 263 bytes. The MQTT-XT is receiving this various byte array from autopilot with serial interface and replaces it with predefined topics. Topics are generated over the activator client with a UAV number tag. Thus, UAV client creates access points for the end-user over the activator client. The activator client transmits its own status to the activator client broker when the system is activated. The topics are designed according to the MAVLINK packet structure and they are generic for all UAVs.

The user will be able to receive the status of desired UAV system with the `"fog[X]/drone[Y]/state/[Z]/"` topic where the X indicates the FoG number, Y indicates the UAV number, and Z is related to the directly MAVLINK topic like location, battery status, heading, etc. UAV client listens to commands from the activator client under "drone[Y]/commands" and responses under "drone[Y]/commands_result/[A]" where the A has received a command message. The command result includes the success or failure message to inform the end user. The MAVLINK packets' interpreting operation is designed by imitating the MAVLINK messages. The serialization structure of MAVLINK supports different communication methods for UAV communication. The UAV client of MQTT-XT is built on the serial interface of the autopilot. The MAVLINK messaging is active between autopilot and UAV clients. However, the user no anymore to struggles with MAVLINK packets since the communication is provided by MQTT topics. The UAV client interpreter is given in Figure 4.10.

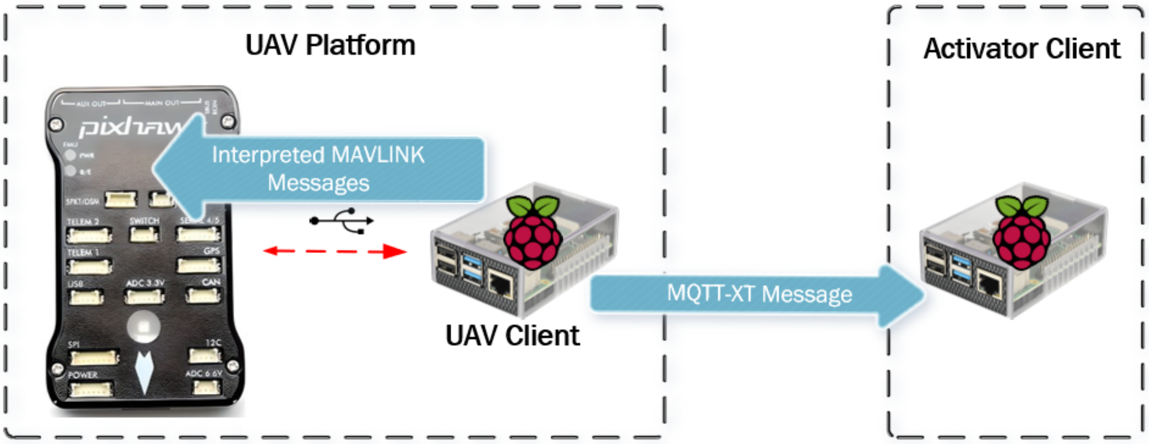


Figure 4.10 UAV Client MAVLINK Interpreter

The main purpose of UAV client is to make MAVLINK protocols accessible via MQTT topics for the any type of IoT network. This client structure can be used in one drone and can be used for dozens of different types of platforms. Deploying it on the drone is enough to start the communication. The backbone of the MQTT-XT algorithm for the UAV client is given in Figure 4.11.

Algorithm 3 MQTT-XT Backbone Algorithm for UAV Client

```
function MAVLINK_INTERPRETER_TO_MQTT()    ▷ Listens MAVLINK on serial interface, convert messages to publish on broker
    $status = Convert\_Serial\_Data()$
    $XT\_Publish(fog[X]\_IP, ”drone[Y]/state/[data]”, status)$
end function

function MQTT_INTERPRETER_TO_MAVLINK()    ▷ Listens MQTT topics’ messages, convert MQTT messages to MAVLINK to send through serial interface
    $command\_result = Convert\_Serial\_Data()$
    $XT\_Publish(fog[X]\_IP, ”drone[Y]/command\_result/[data]”, command\_result)$
end function

function GENERATE_LISTENER_TOPICS()    ▷ Generates topics for UAV client
    $XT\_Subscribe(”drone[A]/commands/\#”)$
end function
```

Figure 4.11 MQTT-XT Backbone Algorithm for UAV Client

5. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents the experimental results by comparing the payload, time delay, mission time, and packet loss between MQTT-XT and MAVLINK protocols. The protocols are deployed within the different levels of the network to measure for given metric sets. Furthermore, the experimental results are divided into two major sections: Message set comparison and Scenarios.

The message set section includes the 10 types of messages that are provided to control UAVs and to receive their status of them. The 5 vital status messages are selected for experimental results: Battery, Location, Groundspeed, Flight mode, and Heading. The other message types are commands to manage UAV movements with pre-defined functionalities: Arm, Takeoff, Land, Offboard, and Go-to.

The scenarios are created with the same conditions and environments as the message set, but they cover the complete task to manage drones. The last scenario is including the IoT packet transfer with three different sensor types while still including the management of UAVs. On the other side, the first three scenarios are only related to managing UAVs.

These two major experiments are tested in the same testbed; however, the number of UAVs is 6 in the scenarios while this number is limited to one in message sets. Also, the number of IoT nodes is different from each other since the scenarios are more focused on the performances of protocols while the message set tests aim to measure individual messages and have only 1 node per network level.

The key difference between the protocols is the network layers. The MQTT-XT benefits from the TCP protocol while the MAVLINK is built over UDP. The MAVLINK has been developed to provide RF communication to achieve long distances when the UAV is in line of sight. However, as the demands of the user are changed, it has implementations for different application layers. In this thesis, we prefer to use the UDP layer since it is more mature and widely used.

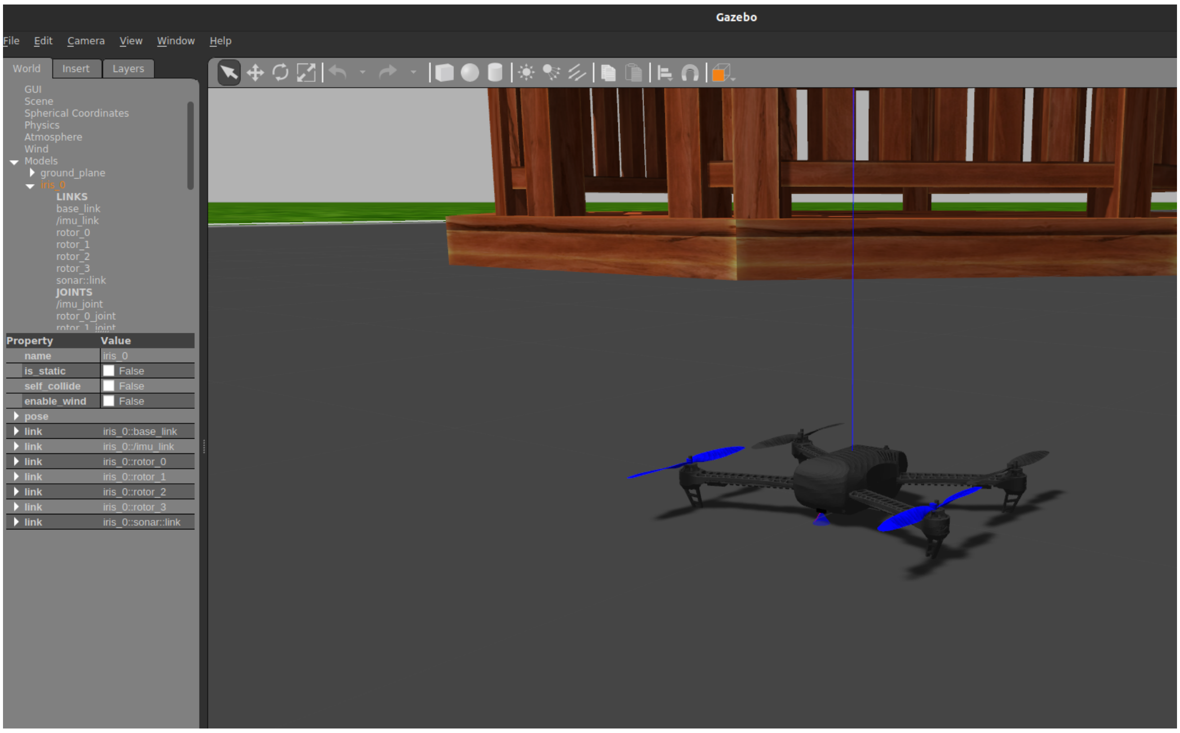
5.1. Testbed and Simulation Environment

The simulation environment is built using the Gazebo, which is an open-source project that provides the deployment of different types of UAVs and PX4 autopilot. The simulator is generally preferred to simulate robotic applications with its 3D physics engines. It generates the sensor data for UAVs and it has the ability to interact with drones with the defined topics.

The PX4 flight controller is used to simulate the UAV platform autopilot. PX4 autopilot has a strong open-source community. This flight control project is started at ETH Zurich and it is low-cost and has a plug-and-play architecture to use in different research projects and environments. It allows to use of different modes which are named Software in the Loop (SITL) and Hardware in the Loop (HITL). The SITL provides to run the autopilot software in the simulator without using any physical device while HITL needs a real device which is connecting the simulator with the serial bus. The HITL is useful for testing flight controllers in a simulator before deploying them on a UAV platform. Our experimental testbed is built on the SITL since the controller does not need to be tested. The view of the gazebo is given in Figure 5.1

The testbed is varied for the two major experimental results. For the first experimental results, the architecture was designed to receive and transmit messages for both protocols. The testbed includes one end-user computer, one cloud, one node, and one UAV node. For the MQTT-XT, the cloud is used to deploy MQTT. The FoG node has the activator client and the end-user has the ground station client to provide interaction with the nodes. Lastly, the UAV platform has the UAV client and it simulates the UAV platform by using the UAV Gazebo simulator.

For the MAVLINK, the cloud, FoG, and UAV nodes have the MAVLINK router. The user interacts drone with the MAVSDK library. The MAVSDK is developed to be used in different applications with software-based implementation. The MAVLINK router has the ability to forward the MAVLINK packets to both sides like a bridge. To decide on the MAVLINK router, two different candidate applications put themselves forward: mavproxy and mavp2p.



Property	Value
name	iris_0
is_static	□ False
self_collide	□ False
enable_wind	□ False
pose	
link	iris_0::base_link
link	iris_0::/imu_link
link	iris_0::rotor_0
link	iris_0::rotor_1
link	iris_0::rotor_2
link	iris_0::rotor_3
link	iris_0::sonar::link

Figure 5.1 Gazebo Simulator

Both of them serve the purposes of the experimental results. However, from the perspective of the simulation environment, the mavp2p promises to serve under constrained resource conditions. For this reason, mavp2p is used as a MAVLINK router for all test scenarios. The UAV platform has the same structure as the MQTT-XT side except for the UAV client.

The UAV client uses the MAVLINK router like other nodes. The general structure of the testbed for MQTT-XT and MAVLINK protocols is given respectively under Figure 5.2 and Figure 5.3.

The second major experimental result aims to test protocols under specific tasks to reveal the performances of the protocols under heavy load. For this reason, four different scenarios are defined. All scenarios have six UAV Clients with the two FoG nodes. The first 3 UAVs are connected to the FoG 1, while the last 3 of them are connected to the FoG 1. The number of cloud and end-user clients is 1 node for each of them. However, scenario 4 has an extra 6 IoT sensor nodes which produce temperature, humidity, and light data to transmit end-user. Thus, the main aim of Scenario 4 is to present the performance capabilities under

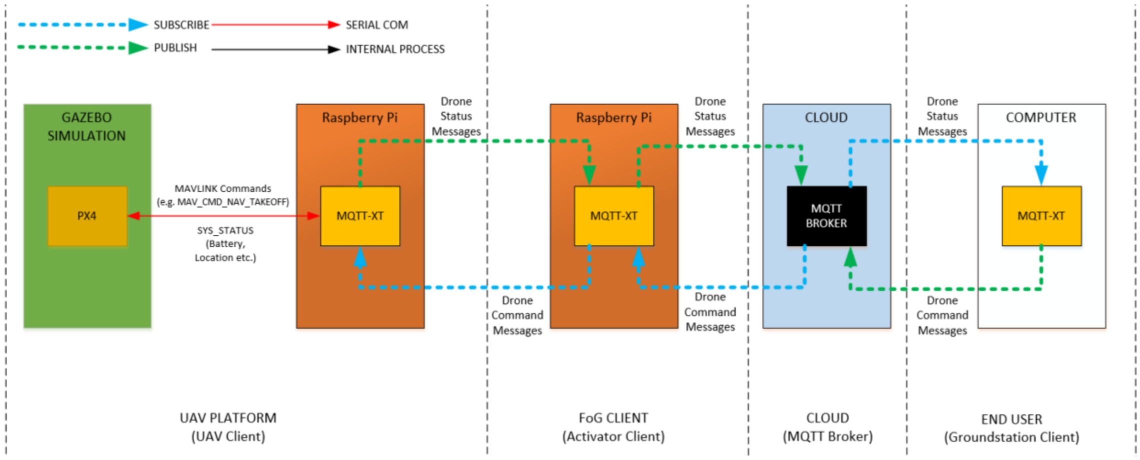


Figure 5.2 Testbed Structure for the MQTT-XT Protocol on Message Sets

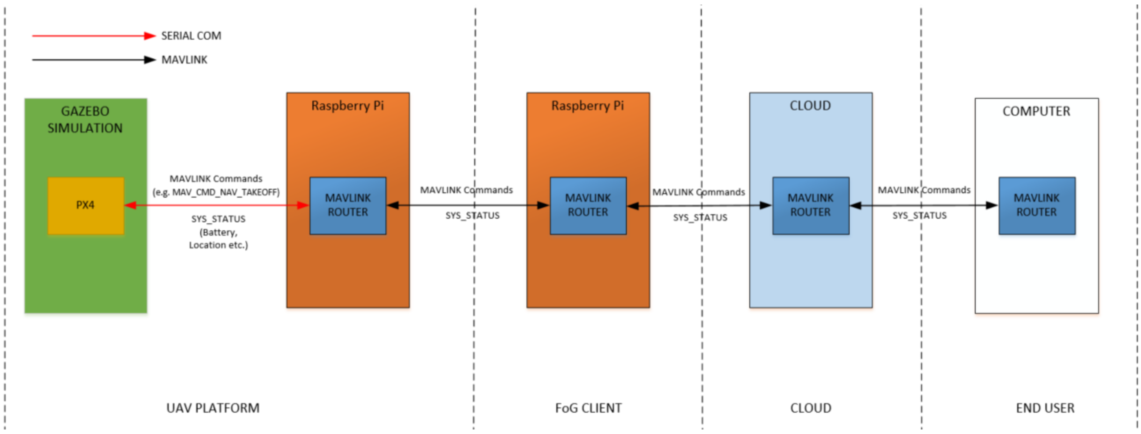


Figure 5.3 Testbed Structure for the MAVLINK Protocol on Message Sets

an IoT environment while the first three scenarios aim to manage UAVs. For the first three scenarios, the general structure of the testbed for MQTT-XT and MAVLINK protocols is given respectively, under Figure 5.4 and Figure 5.5.

Scenario 4 has a unique architecture while the UAV management is the same as the 3 scenarios. The key difference is related to IoT sensor networks. The scenario includes directing UAVs to specific tasks where the UAVs receive three different sensor types and transmit these data over their FoG nodes. A total of 18 sensor data is transmitted to the end-user in Scenario 4. The Scenario 4 problem definition helps to understand the differences between MQTT-XT and MAVLINK protocols from the point of IoT network view. UAVs are in service as the IoT node in MQTT-XT, while the MAVLINK lacks the transmission or interact with other IoT nodes. For this reason, the basic MQTT structure is built for MAVLINK to transmit sensor data over existing networks. The simulation view can be seen in Figure 5.6.

Furthermore, the detail of scenario expectation is given in Figure 5.7. Each UAV has a task to do such as go to given coordinate to obtain data. The range of the sensor is not able to transmit the sensor data to FoG. Thus, UAVs create a bridge by obtaining data from sensors and transmitting them directly to FoG. The MQTT-XT has the flexibility to create topics that serve the communication with other IoT nodes. Thus, UAVs can be considered a part of the IoT node. The UAVs are connected only to their FoG and this is pre-defined.

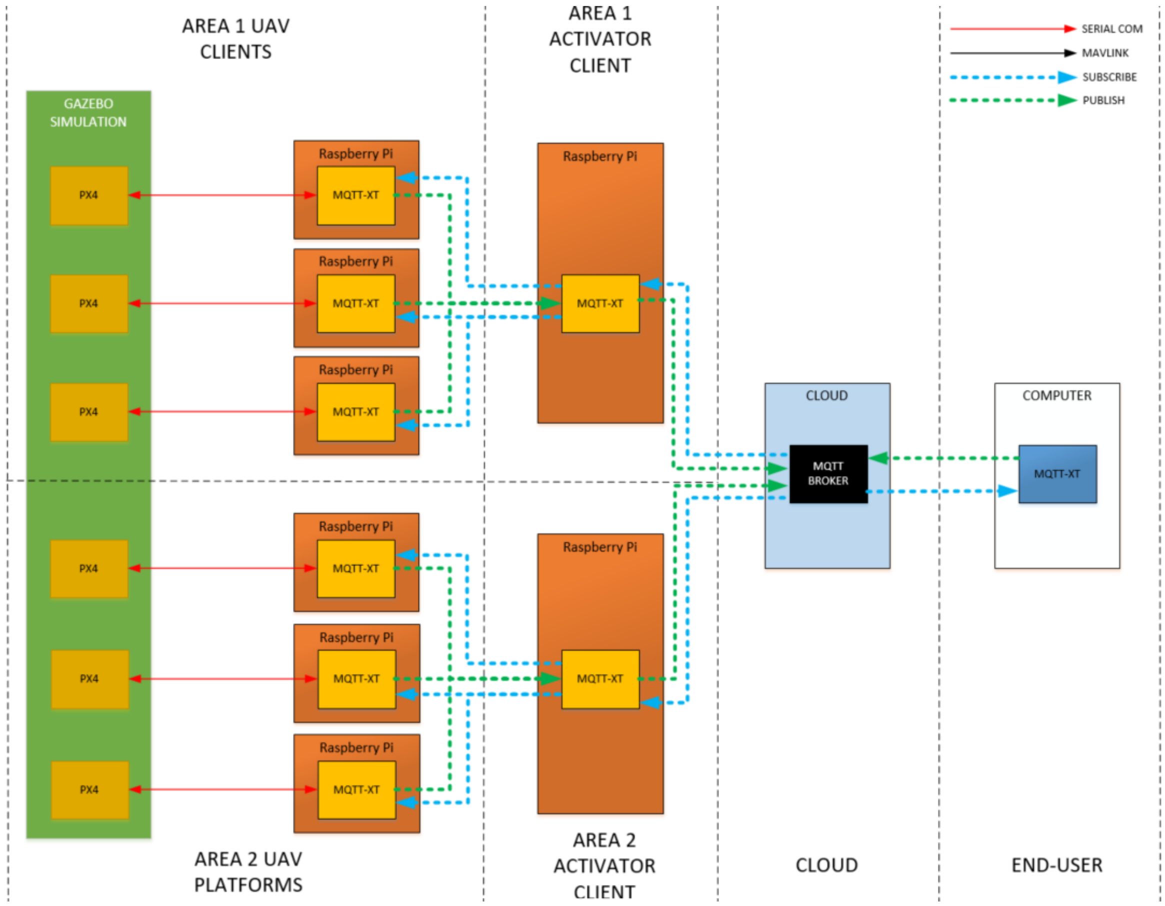


Figure 5.4 Testbed Structure for the MQTT-XT Protocol on Scenarios

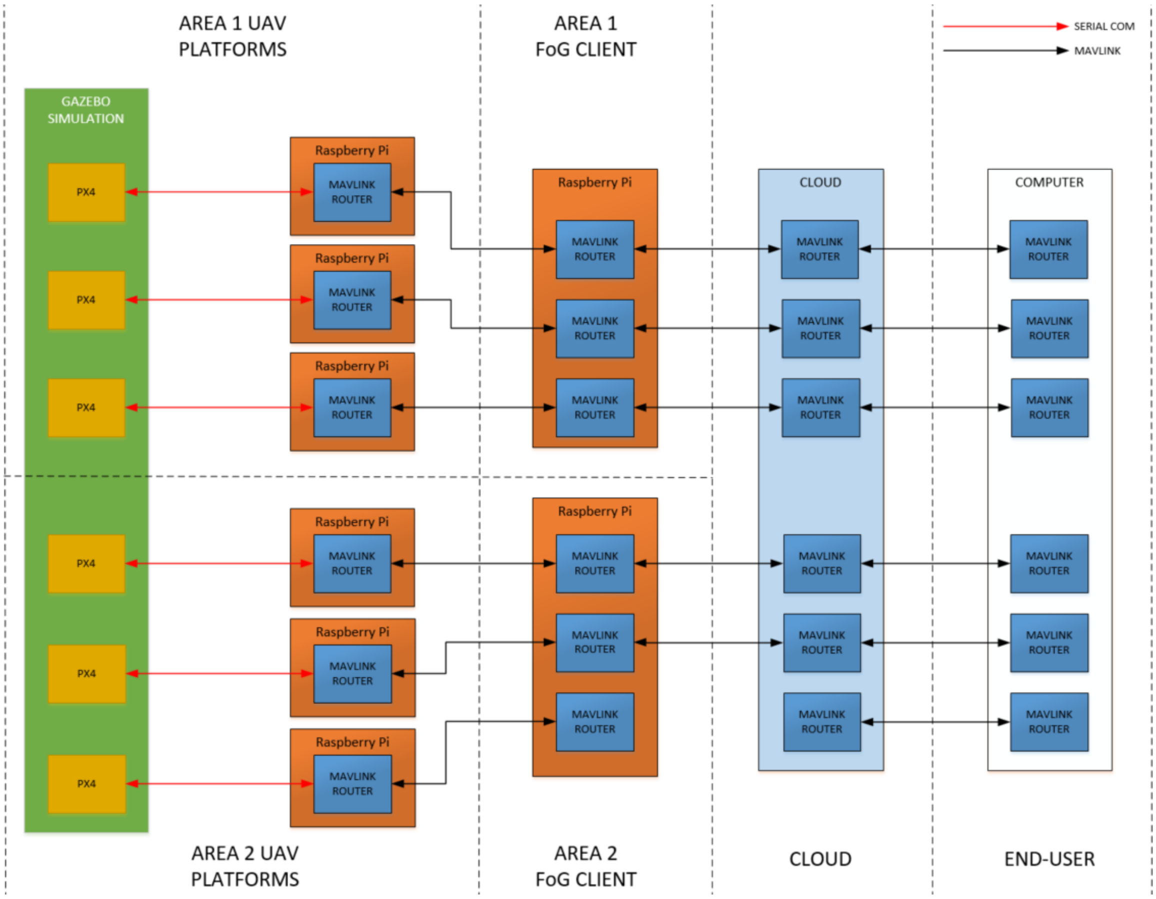


Figure 5.5 Testbed Structure for the MAVLINK Protocol on Scenarios

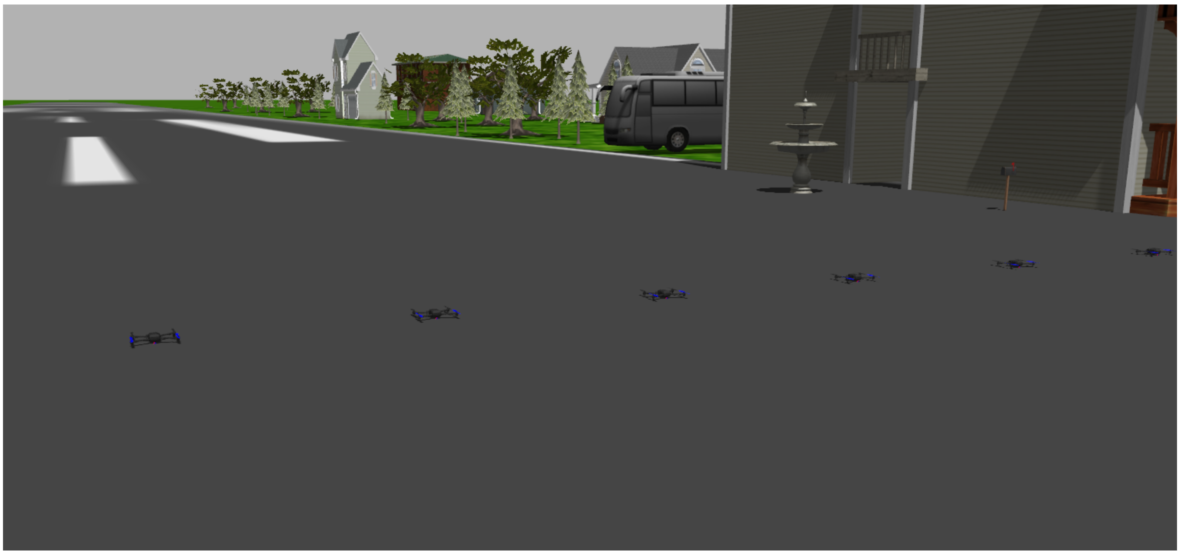


Figure 5.6 Gazebo Simulator for Scenario 4

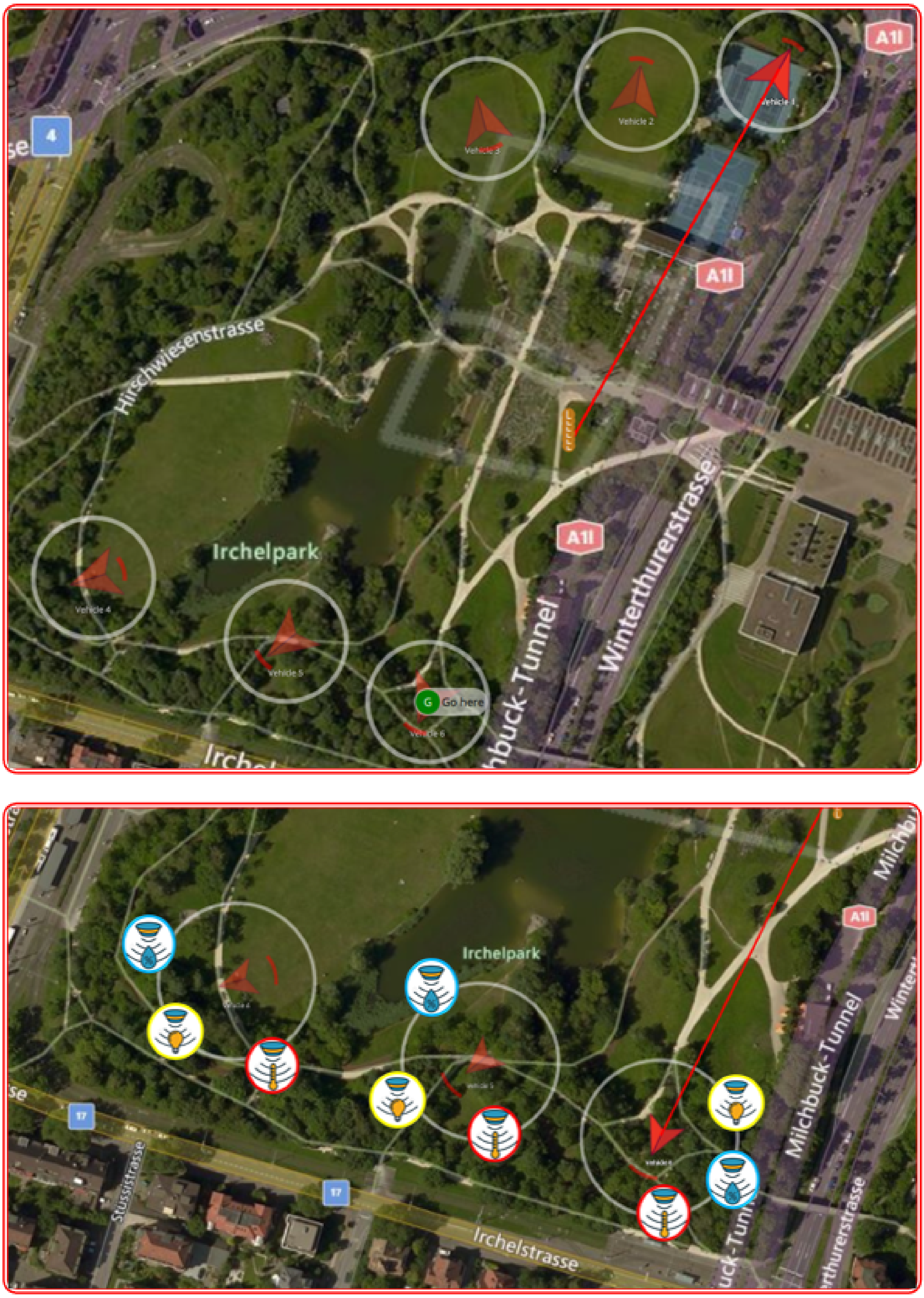


Figure 5.7 Scenario 4 design on the Groundstation Map View

5.2. Performance Metrics

The evaluation criteria of performance metrics for the network-based methods depend mostly on the time delay, packet size, and packet loss. Because, these three types of metrics are affected by the other types of metric data such as throughput, bandwidth, jitter, etc. For this reason, 3 different types of metrics are selected for evaluating MAVLINK and MQTT-XT protocols when comparing the message set: Packet size, Time delay, and Packet Loss. Furthermore, the other important metrics are the message traffic size after establishing a connection between nodes and the number of messages exchanged between all nodes. Consequently, it is aimed to compare the protocols over 5 different metrics.

The packet size is measured independently for the transmitted command message and received status message. The time delay metric for the message set is calculated end-to-end. In other words, required time for messages to reach the end-user node from the UAV node or vice versa. Packet loss is one of the most important metrics since UAVs are more susceptible than IoT devices regarding packet loss if they are not received a command message, and the given task is not able to execute.

The packet count and message traffic size metrics present the message traffic that occurs when the user needs to send a command or receive a status. The base architecture is not changed for MAVLINK or MQTT-XT. Since both protocols have their architecture to provide communication among nodes, the protocols can send or receive different messages other than the selected ones. The main purpose of the metrics is to present that the packet size which is consumed by the protocols to provide communication does not only consist of the messages sent or received but also the different types of messages that should be taken into account since the MAVLINK streams many different packets, even if the user needs the specific ones. Also, MQTT-XT has internal processes between nodes e.g., SYNC, CONNECT, ACK. In this way, traffic density can be observed among all nodes.

The packet loss is the other important criterion since the message types consist of required information for both UAV and end-user clients. For this reason, the QoS level of the

MQTT-XT is used as 0 to provide a more fair comparison. Because the QoS 1 and 2 levels guarantee the packet delivery. Also, the QoS level is not defined as 1 or 2 since the density of the message is high. Furthermore, QoS 2 is the slowest alternative among quality of service levels. The packet loss is calculated for only transmitted or received messages. The other message types are not taken into account since the main purpose is evaluating the compared messages and both protocols have different types of messages.

Lastly, the above 3 of 5 metrics are different for prepared scenarios. Because the point of the scenarios is to measure the robustness and efficiency of the protocols in the different conditions and tasks. The time delay metric is changed as the mission complete time to measure the response of the protocols under heavy-load conditions and pressure of the high density of traffic. The packet loss is obtained for the whole traffic which is produced by the protocols. The packet size is not used for the scenarios. However, message traffic size and packet count metrics have not changed. Also, unlike the message set tests, packet loss is calculated for all messages that are taken as part of scenarios. In other words, the whole packets which are transmitted or received by nodes are measured to obtain packet loss.

5.3. Command and Status Messages Evaluation

The various number message set is defined under MAVLINK to manage UAVs. These different message sets have their responsibilities. MQTT-XT imitates each MAVLINK message under MQTT topics to transmit one node to another. To provide a comparison between MAVLINK and MQTT-XT, the 5 different status messages and 5 different command message is used. Especially, the command message benefits the autopilot structure to direct UAVs and the command set does not require manual input continuously. These 10 different message types and their description is given in Table 5.1 and these types are evaluated with explanation in following subsection.

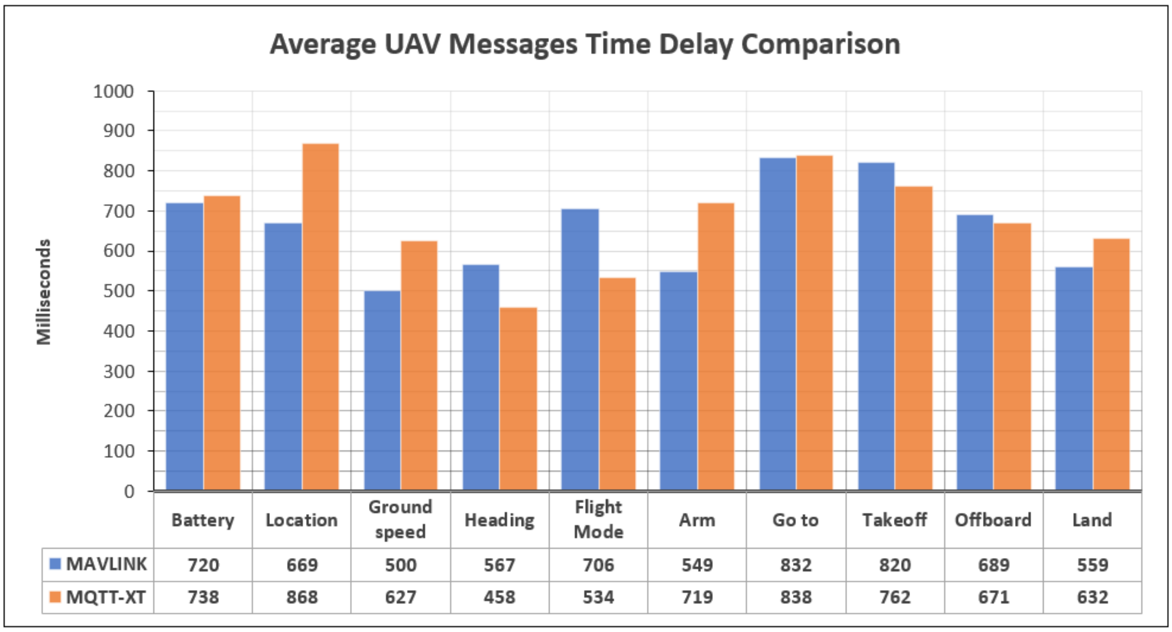
Message Name	**Type**	**Description**
Arm	Command	Provides to start (arm) the motors of the UAVs.
Takeoff	Command	Provides to take-off the UAVs vertically in current position.
Land	Command	Provides to land the UAVs vertically in current position.
Go to	Command	Provides to direct the UAVs given location and altitude.
Offboard	Command	Provides to direct the UAVs with given attitude.
Battery	Status	Provides remaining battery information of UAVs.
Location	Status	Provides current location and altitude of UAVs.
Heading	Status	Provides current heading of UAVs.
Groundspeed	Status	Provides current speed of UAVs relative to ground.
Flight Mode	Status	Provides current flight mode information of UAVs.

Table 5.1 Evaluated message types and their description

5.3.1. Time Delay

The time delay of each message is tested individually. This metric is obtained by measuring message transmission time for an end-to-end node. For each message type, 100 messages are used to eliminate intruders. The summary results of the time delay for MQTT-XT and MAVLINK are presented under Figure 5.8.

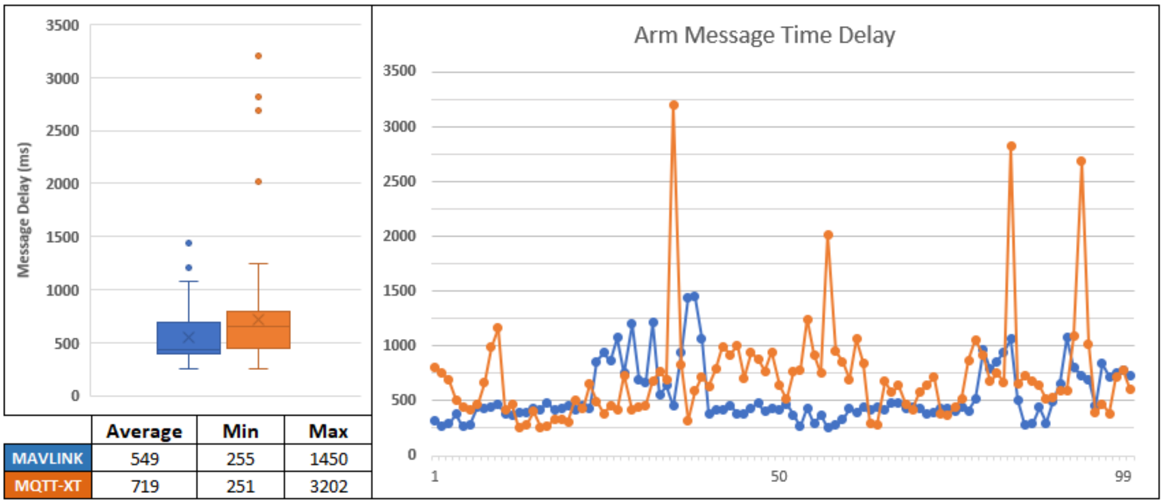
The results show that both protocols produce close values to each other. However, MAVLINK performs better since it is based on the UDP stream. Also, the MQTT has only a 6 ms difference from MAVLINK when the Go-To message type is analyzed. The highest difference between them is obtained for the location data. The time difference is 200 ms for this message.



	Battery	Location	Ground speed	Heading	Flight Mode	Arm	Go to	Takeoff	Offboard	Land
🟦 MAVLINK	720	669	500	567	706	549	832	820	689	559
🟧 MQTT-XT	738	868	627	458	534	719	838	762	671	632

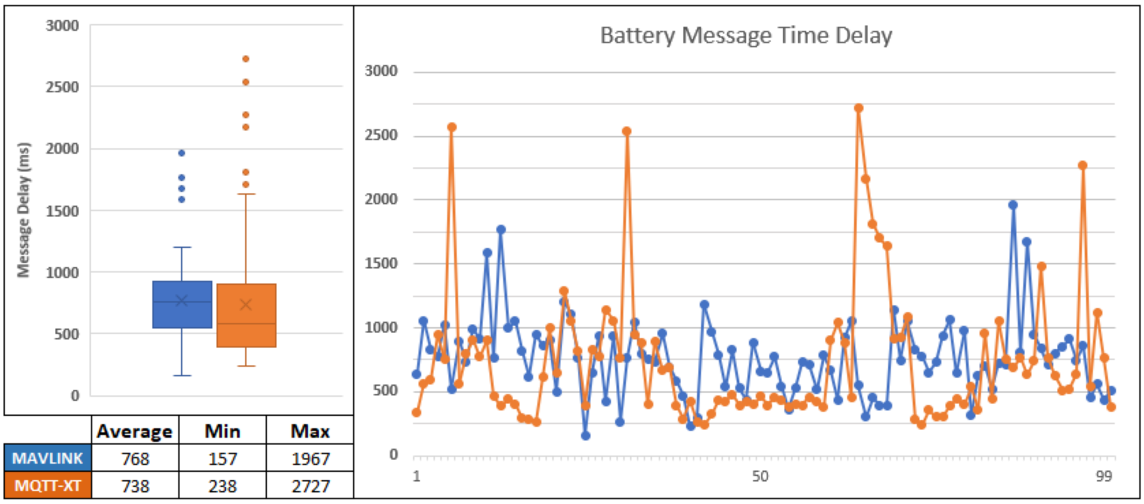
Figure 5.8 Average Time Delay for 100 Messages

The maximum and minimum points of time delay data have very different results when the message types are inspected individually. For example, in Figure 5.9, the maximum time delay is observed as 3202 ms for MQTT-XT where the value is 1450 ms for MAVLINK. A similar result can be seen in Figure 5.10. The maximum value is 2727 ms for MQTT-XT where the value is 1967 ms. The key difference between MQTT-XT and MAVLINK is related to their architecture. The principle of the MAVLINK stands on streaming that aims to fire and forget. This situation differs for MQTT-XT. Because MQTT-XT tries to transmit a message at least once since the QoS level is set as 1.



	Average	Min	Max
MAVLINK	549	255	1450
MQTT-XT	719	251	3202

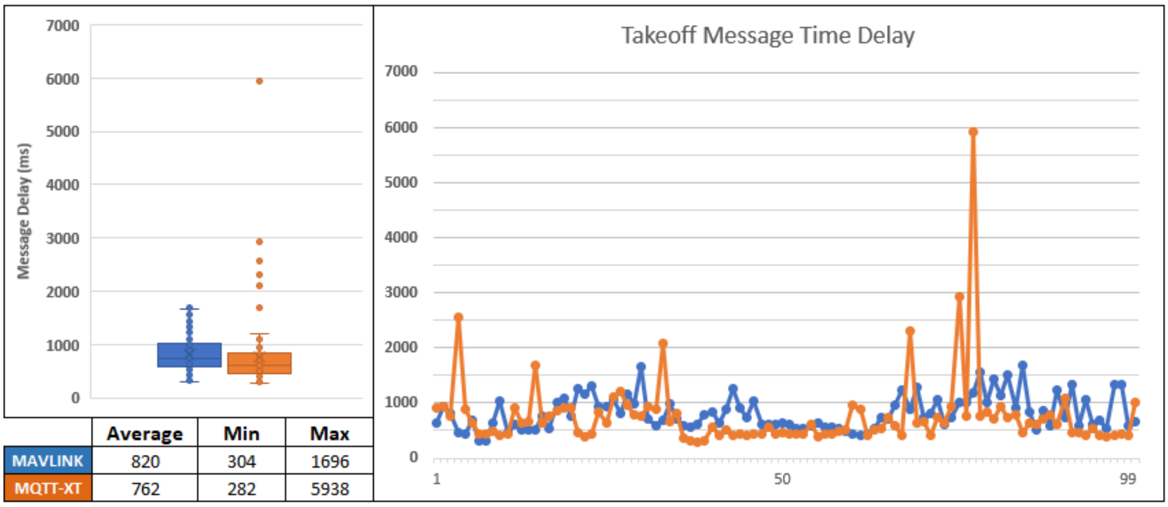
Figure 5.9 Time Delays for 100 Arm command message



	Average	Min	Max
MAVLINK	768	157	1967
MQTT-XT	738	238	2727

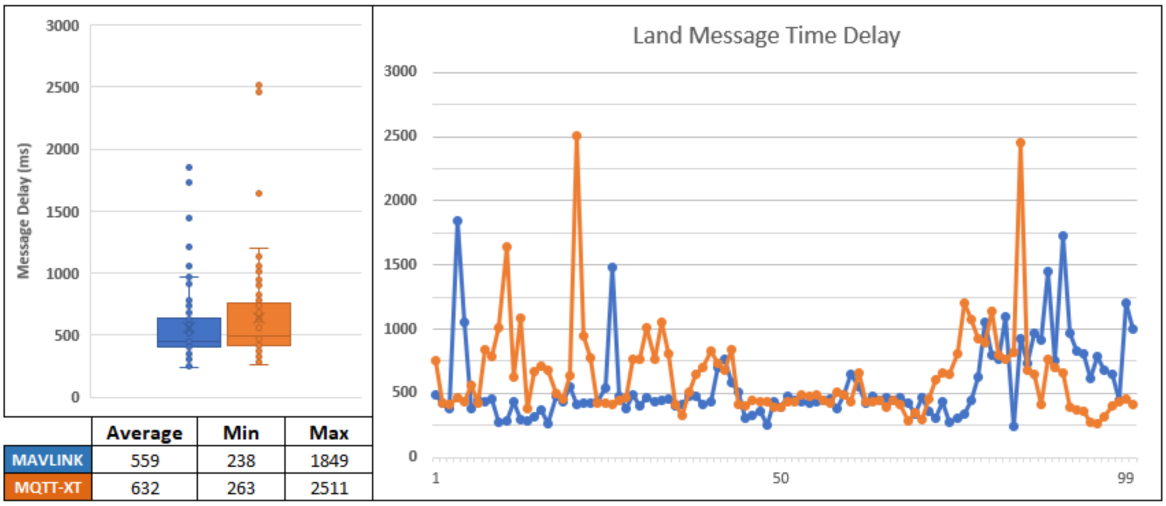
Figure 5.10 Time Delays for 100 Battery status message

The time delays for the command message set are given in Figure 5.11, Figure 5.12, Figure 5.13, and Figure 5.14. The result of the command message set shows similar trends to each other. When the time delay is inspected for command messages, the maximum value to transmit a message is offboard command with 2112 ms among the command messages. For the MQTT-XT side, this maximum value is received for the Takeoff command with 5938. However, the time delay is the exception and the average time delay of the commands is too far between MQTT-XT and MAVLINK.



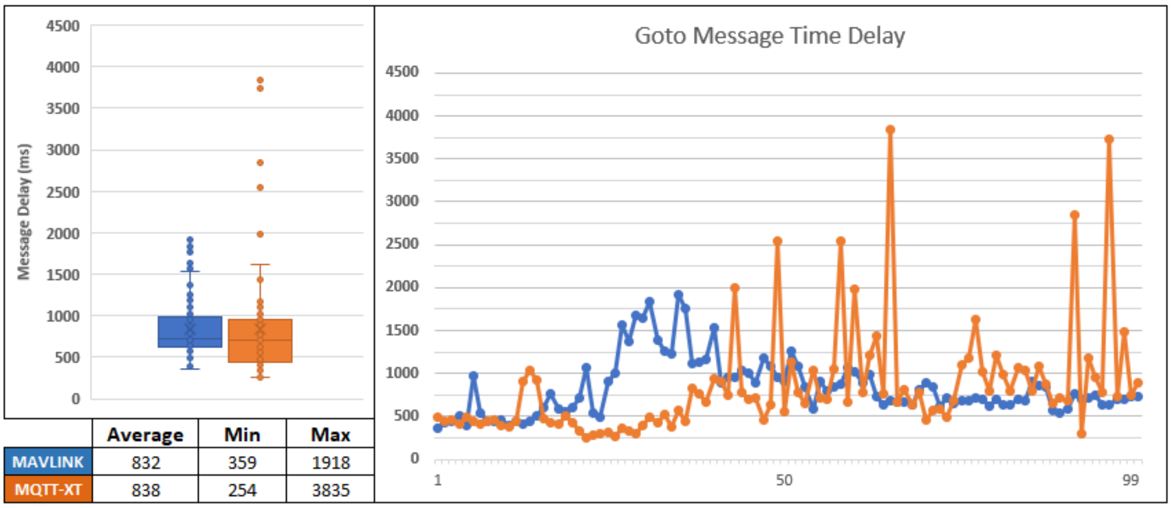
	Average	Min	Max
MAVLINK	820	304	1696
MQTT-XT	762	282	5938

Figure 5.11 Time Delays for 100 Takeoff command message



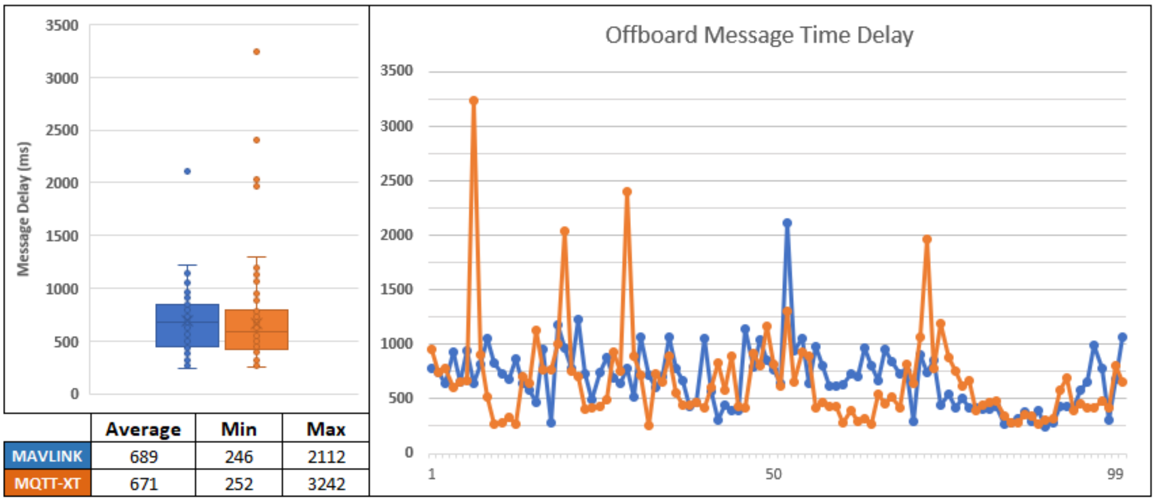
	Average	Min	Max
MAVLINK	559	238	1849
MQTT-XT	632	263	2511

Figure 5.12 Time Delays for 100 Land command message



	Average	Min	Max
MAVLINK	832	359	1918
MQTT-XT	838	254	3835

Figure 5.13 Time Delays for 100 Go to command message

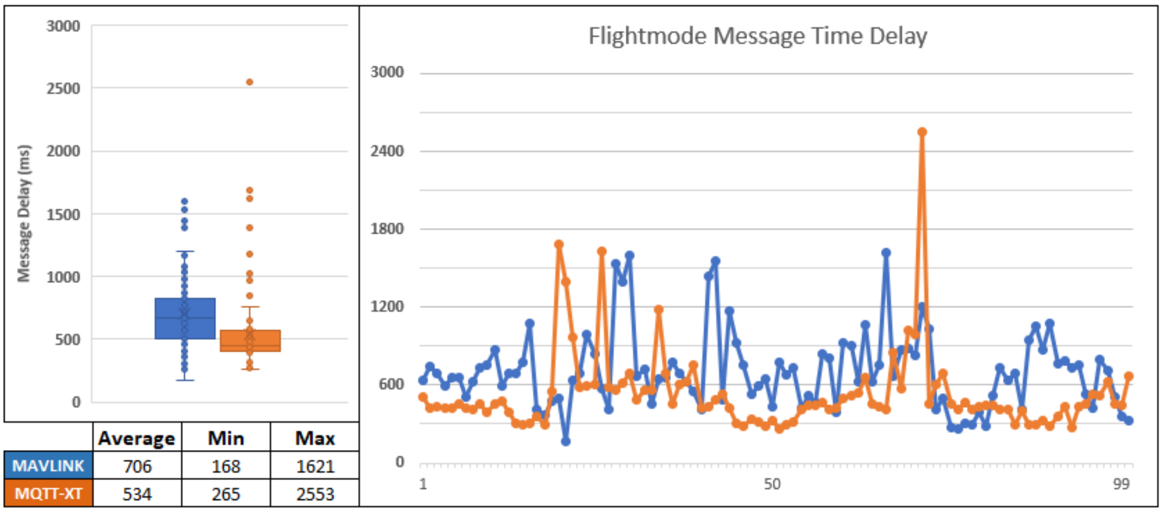


	Average	Min	Max
MAVLINK	689	246	2112
MQTT-XT	671	252	3242

Figure 5.14 Time Delays for 100 Offboard command message

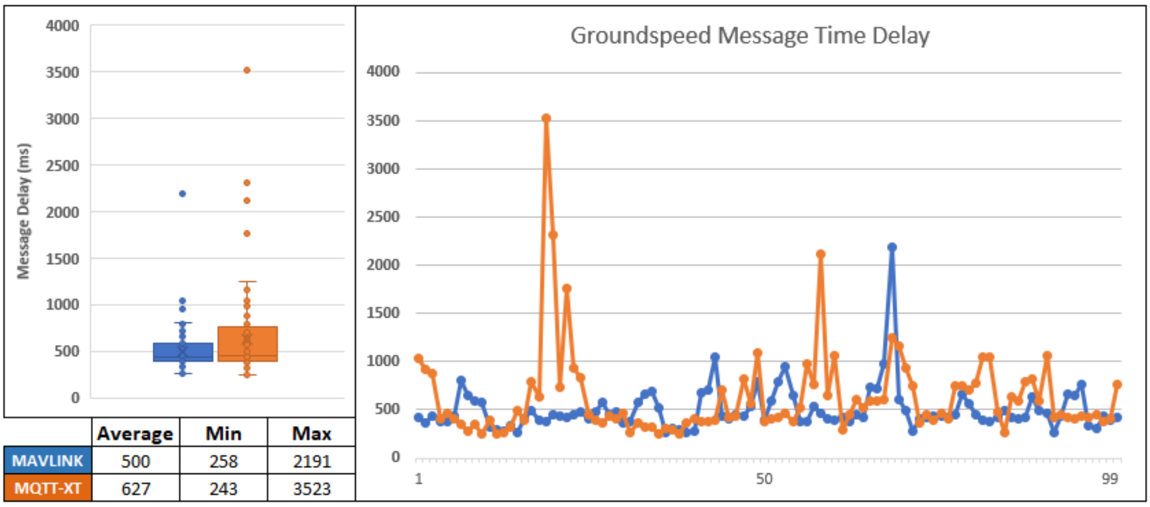
The time delays for the status message set are given in Figure 5.15, Figure 5.16, Figure 5.17, and Figure 5.18. The sudden rise in time delay is observed for the location status message and the linear decrease that follows it. The main root cause is that location data is the heaviest payload data. So, the time delay trend on the location can be considered as a result of this reason. The status message has a similar result to the command message set. Comparing both protocols, the MAVLINK has better time delay performance. The maximum value for transmitting a message is obtained in location status with 2678 ms among all status messages.

Also, the maximum value between status messages is received in the same Location status message for MQTT-XT status with 5523 ms. However, the time delay is the exception and the average time delay of the commands is too far between MQTT-XT and MAVLINK.



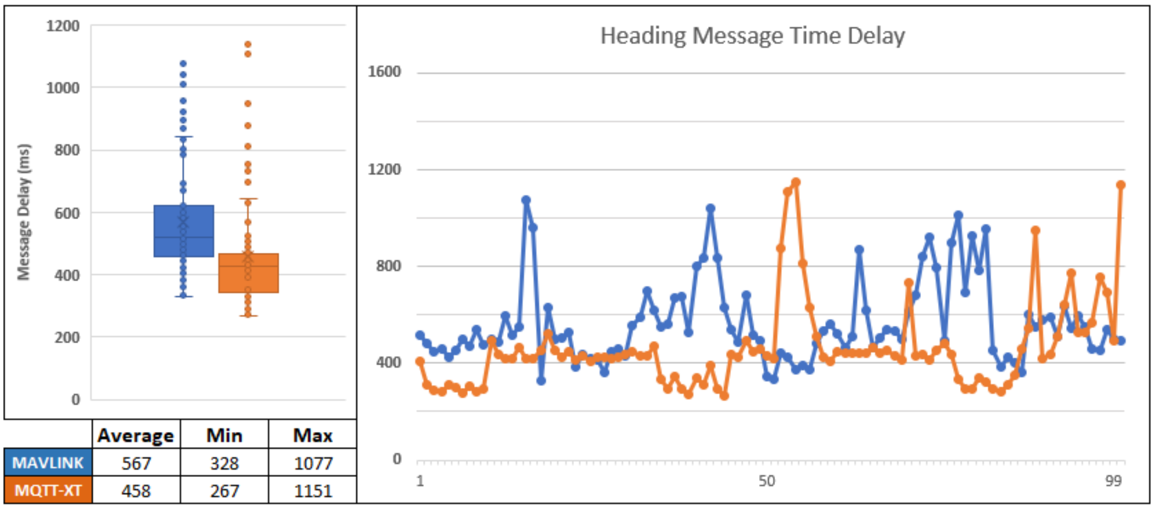
	Average	Min	Max
MAVLINK	706	168	1621
MQTT-XT	534	265	2553

Figure 5.15 Time Delays for 100 Flightmode status message



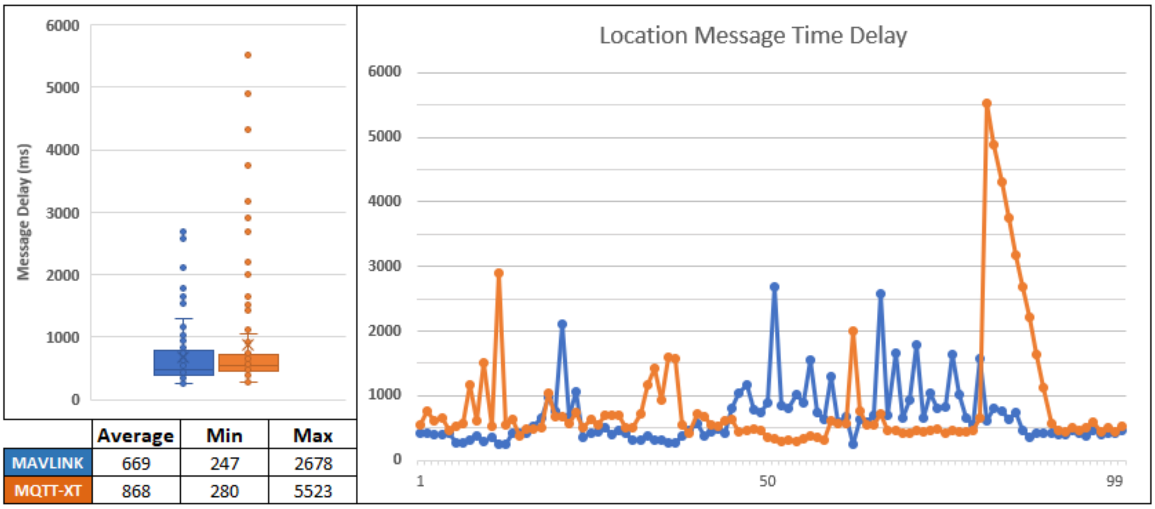
	Average	Min	Max
MAVLINK	500	258	2191
MQTT-XT	627	243	3523

Figure 5.16 Time Delays for 100 Groundspeed status message



	Average	Min	Max
MAVLINK	567	328	1077
MQTT-XT	458	267	1151

Figure 5.17 Time Delays for 100 Heading status message



	Average	Min	Max
MAVLINK	669	247	2678
MQTT-XT	868	280	5523

Figure 5.18 Time Delays for 100 Location status message

5.3.2. Packet Loss

The packet loss of the message set for MQTT-XT is a very low percentage when compared to MAVLINK. Thanks to TCP and QoS mechanisms, the packets are transmitting more reliably for MQTT-XT. On the other side, the MAVLINK is only streaming the packets and there is no control mechanism for re-transmission. However, this situation causes more time delay for MQTT-XT, but the trade-off gap is not so close between MQTT-XT and MAVLINK as in the time delay metric. The packet loss is obtained only for the exact number of transmitted and received packets. The other packet types are ignored since the main purpose of the metric is presenting a demanded message set. The command message has a lower packet loss when it is compared to the status message for both protocols. Among the message sets, the location and the heading have the most packet loss percentage than any other type. A total of 15 packets out of 100 are lost in MAVLINK while the packet loss rate is 5 for location and 1 for heading in MQTT-XT. For the different loss rate between Location and Heading message types in MQTT-XT is related to the MQTT-XT. MQTT-XT separates the heading data from other data and not transmits other data, unlike MAVLINK. The payload size can be up to 255 bytes. So, some of the packets have many data instead of the desired ones. The packet loss for all message sets are given in Figure 5.19.

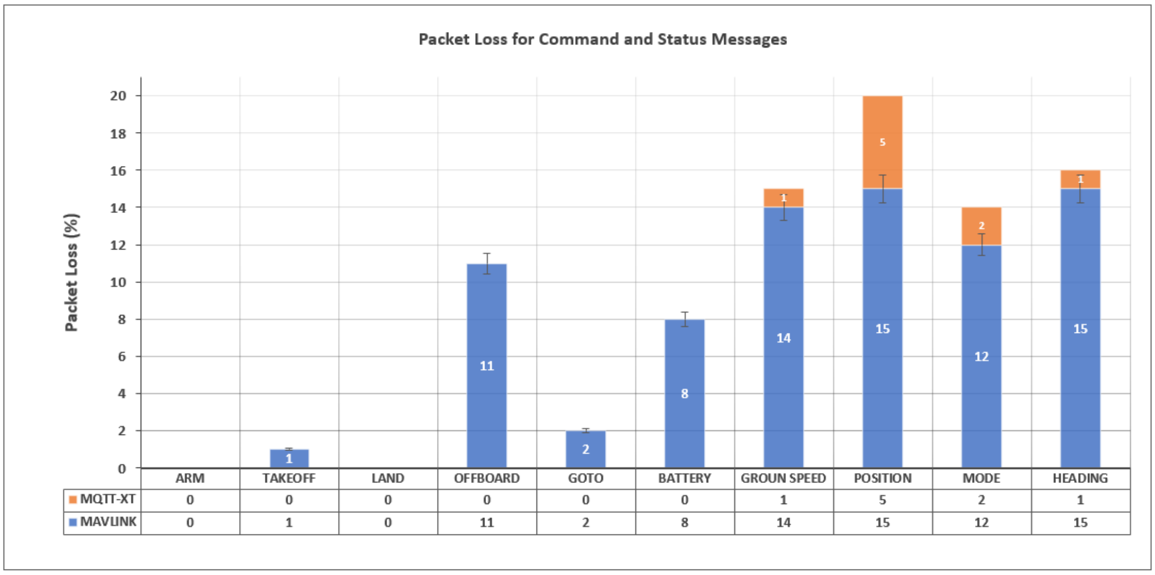


Figure 5.19 Packet Loss for Command and Status Messages

5.3.3. Packet Size

The packet size metric directly aims to present the size of the packet for both protocols. For this reason, all status and command messages are obtained to give perspective for performance evaluation considering only packet types. Also, the packet sizes are obtained between only 2 nodes. If the architecture has 3 different hops to be able to transmit a message, packet sizes will be increased relative to node number. The packet sizes for 10 different messages are given in Table 5.2.

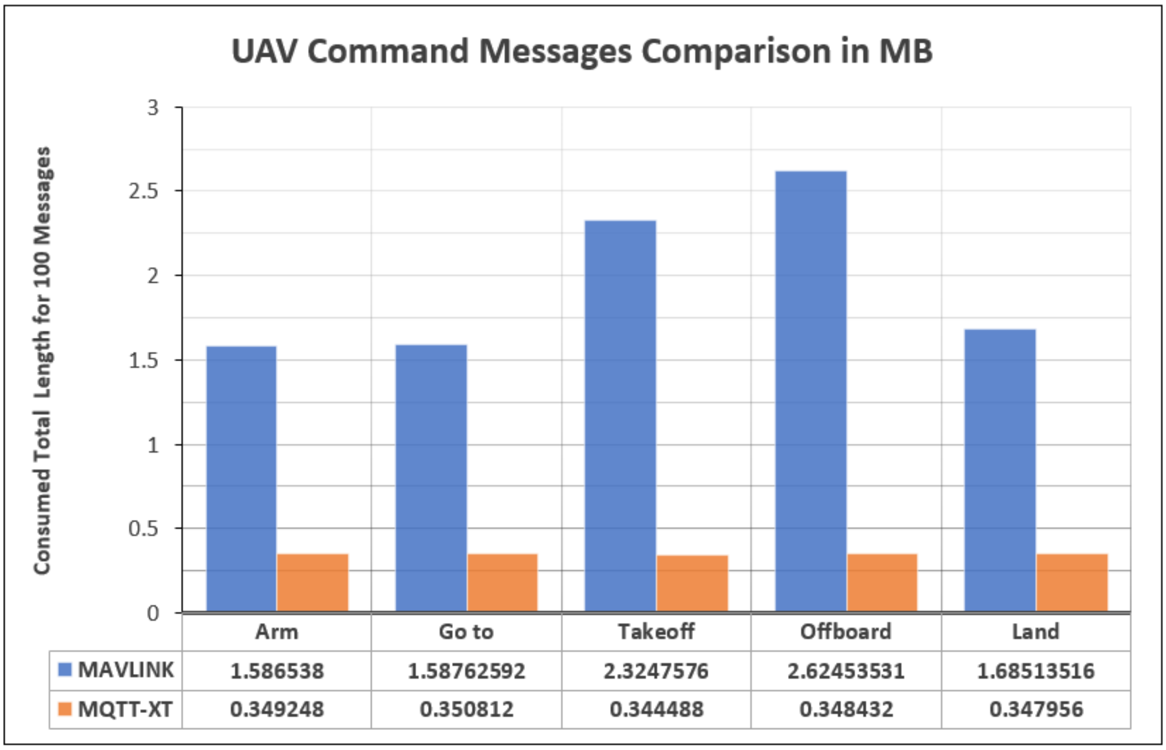
Message Name	MAVLINK (bytes)	MQTT-XT (bytes)
Arm	86	95
Takeoff	86	99
Land	86	96
Go to	86	96
Offboard	87	103
Battery	86	97
Location	124	242
Heading	124	112
Groundspeed	71	118
Flight Mode	70	95

Table 5.2 Packet size comparison for 10 message types between MAVLINK and MQTT-XT

The evaluation shows that the MQTT-XT has more payloads than the MAVLINK. The main reason for the packet size difference is related to MQTT-XT topics. MQTT-XT specifies the topic to indicate the address of nodes. Also, it carries the data with the same payload. However, MAVLINK fundamentally aims to stream data. The MQTT-XT reduces the MAVLINK heavy payload by interpreting messages for specific nodes and publishing them under specific topics. Also, MQTT-XT extracts the required data and transmits it when the end-user client sends the connection message.

5.3.4. Packet Count and Message Traffic Size

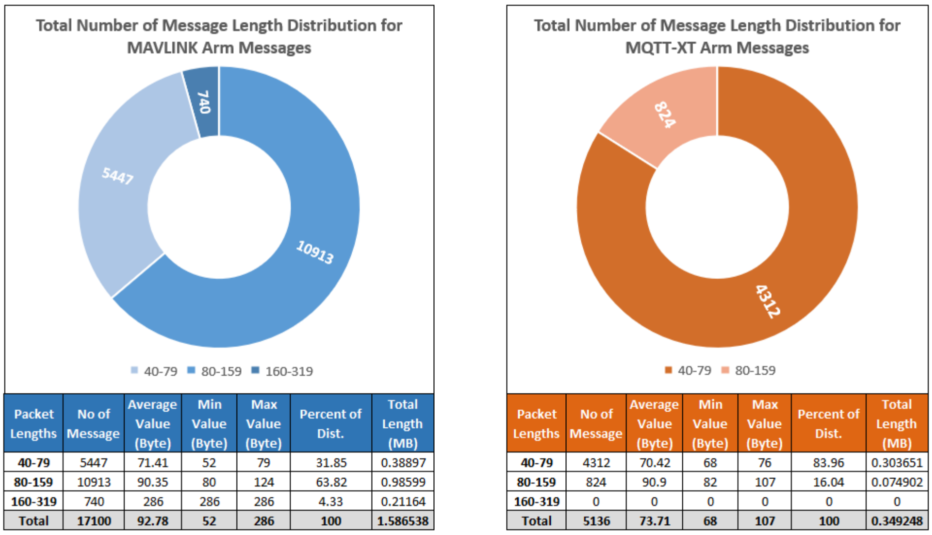
Both protocols have other message types other than the selected message set. When the communication is established protocols produce the traffic rather than only transmitting a specific message. For example, MQTT-XT has different processes between broker and clients like SYNC, ACK, PUBACK, etc. when it deals to transmit one single message. MAVLINK also has a message stream between nodes to provide communication continuously and this function is the fundamental part of it. Therefore, the packet count and the total message traffic size for protocols are measured to reveal the traffic density even if for one particular message type. However, the 100 command messages are sent continuously for both protocols and the connection is not broken until the last message is received. Also, MQTT-XT does not include status messages since it can prefer listening message topics or not. Also, the total sizes of ongoing messages are obtained between the first and last commands. The consumed total size to transmit 5 different 100 command messages is given in Figure 5.20.



	Arm	Go to	Takeoff	Offboard	Land
🟦 MAVLINK	1.586538	1.58762592	2.3247576	2.62453531	1.68513516
🟧 MQTT-XT	0.349248	0.350812	0.344488	0.348432	0.347956

Figure 5.20 UAV Command Messages Comparison in MB

The result shows that the packet size of MQTT-XT is at least 5 times smaller than MAVLINK. Also, the message sizes are changed between commands while the MQTT-XT remains almost the same size for each command message. For the Arm command, the amount of packets and their distribution for the MAVLINK is given in Figure 5.21



Total Number of Message Length Distribution for MAVLINK Arm Messages

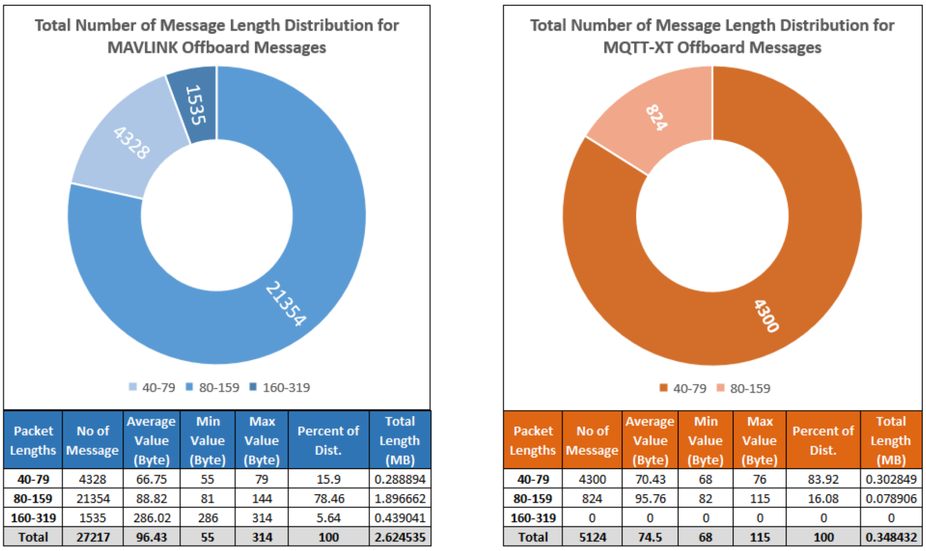
Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	5447	71.41	52	79	31.85	0.38897
80-159	10913	90.35	80	124	63.82	0.98599
160-319	740	286	286	286	4.33	0.21164
Total	**17100**	**92.78**	**52**	**286**	**100**	**1.586538**

Total Number of Message Length Distribution for MQTT-XT Arm Messages

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	4312	70.42	68	76	83.96	0.303651
80-159	824	90.9	82	107	16.04	0.074902
160-319	0	0	0	0	0	0
Total	**5136**	**73.71**	**68**	**107**	**100**	**0.349248**

Figure 5.21 Traffic Message analysis for the Arm command message

The total size of the command is 5 times greater if MQTT-XT and MAVLINK are compared. However, the packet count does not fit the same trend line. The MAVLINK is 3 times greater for the number of packets than the MQTT-XT which is caused by the packet density. The most packet density for the MAVLINK is between 80-159 bytes while it is between 40-79 for the MQTT-XT. The MQTT-XT can transmit the same command messages with smaller packets. Furthermore, the payload can be up to 160-319 for MAVLINK. As a result, traffic will exponentially increase for MAVLINK when the other nodes are added to the same network. Another example can be observed for offboard command in Figure 5.22.

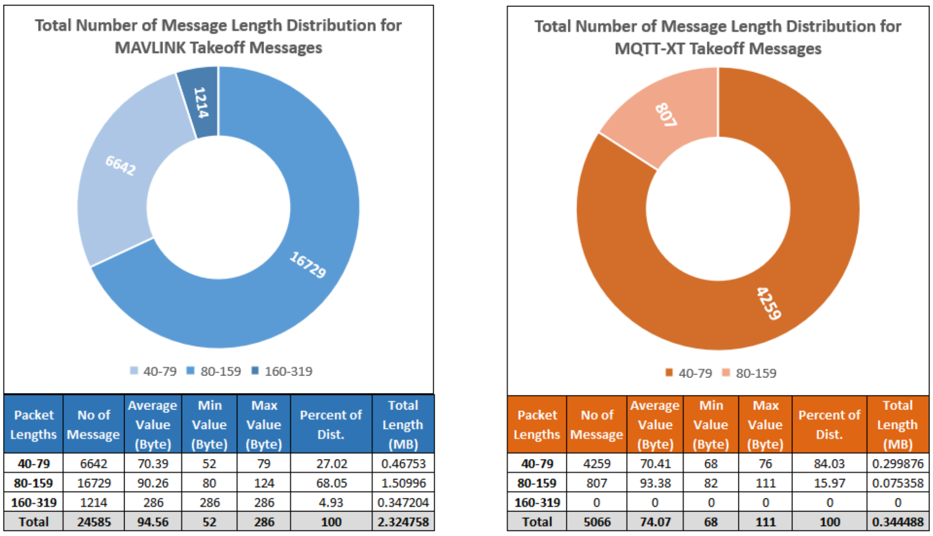


Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	4328	66.75	55	79	15.9	0.288894
80-159	21354	88.82	81	144	78.46	1.896662
160-319	1535	286.02	286	314	5.64	0.439041
Total	**27217**	**96.43**	**55**	**314**	**100**	**2.624535**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	4300	70.43	68	76	83.92	0.302849
80-159	824	95.76	82	115	16.08	0.078906
160-319	0	0	0	0	0	0
Total	**5124**	**74.5**	**68**	**115**	**100**	**0.348432**

Figure 5.22 Traffic Message analysis for the Offboard command message

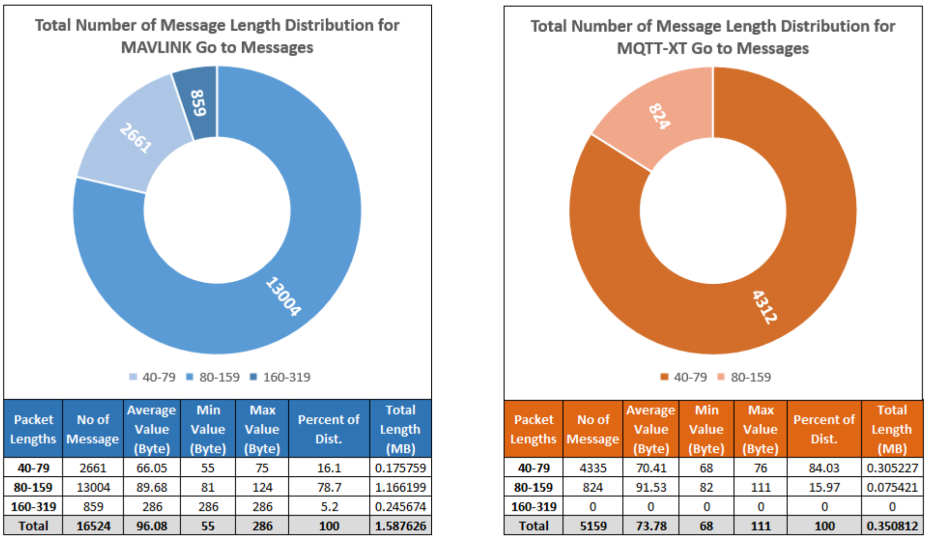
The packet size difference is almost 8 times while the packet count is only 6 times. The correlation between message traffic size and packet count is similar to each other. The other command messages Takeoff, Go to, and Land are respectively given in Figure 5.23, Figure 5.24, and Figure 5.25.



Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	6642	70.39	52	79	27.02	0.46753
80-159	16729	90.26	80	124	68.05	1.50996
160-319	1214	286	286	286	4.93	0.347204
Total	**24585**	**94.56**	**52**	**286**	**100**	**2.324758**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	4259	70.41	68	76	84.03	0.299876
80-159	807	93.38	82	111	15.97	0.075358
160-319	0	0	0	0	0	0
Total	**5066**	**74.07**	**68**	**111**	**100**	**0.344488**

Figure 5.23 Traffic Message analysis for the Takeoff command message



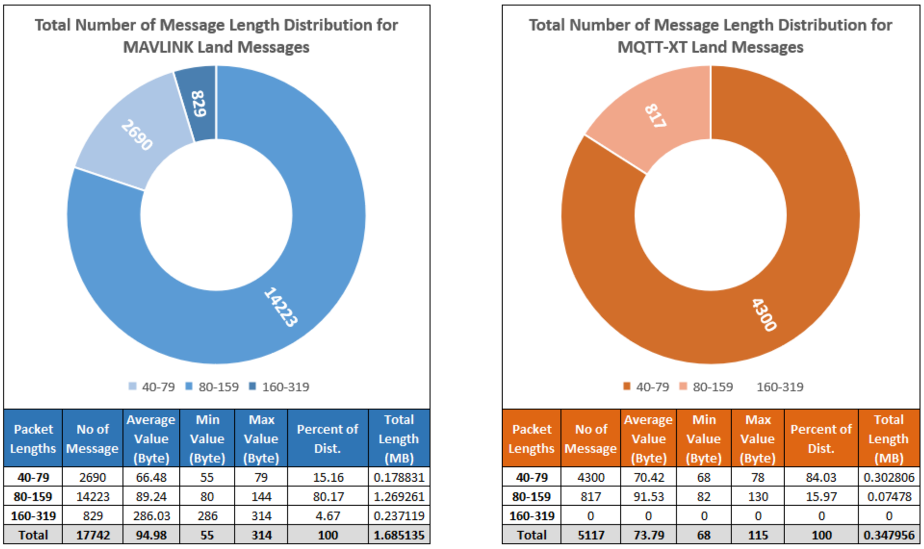
Total Number of Message Length Distribution for MAVLINK Go to Messages

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	2661	66.05	55	75	16.1	0.175759
80-159	13004	89.68	81	124	78.7	1.166199
160-319	859	286	286	286	5.2	0.245674
Total	**16524**	**96.08**	**55**	**286**	**100**	**1.587626**

Total Number of Message Length Distribution for MQTT-XT Go to Messages

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	4335	70.41	68	76	84.03	0.305227
80-159	824	91.53	82	111	15.97	0.075421
160-319	0	0	0	0	0	0
Total	**5159**	**73.78**	**68**	**111**	**100**	**0.350812**

Figure 5.24 Traffic Message analysis for the Go to command message



Total Number of Message Length Distribution for MAVLINK Land Messages

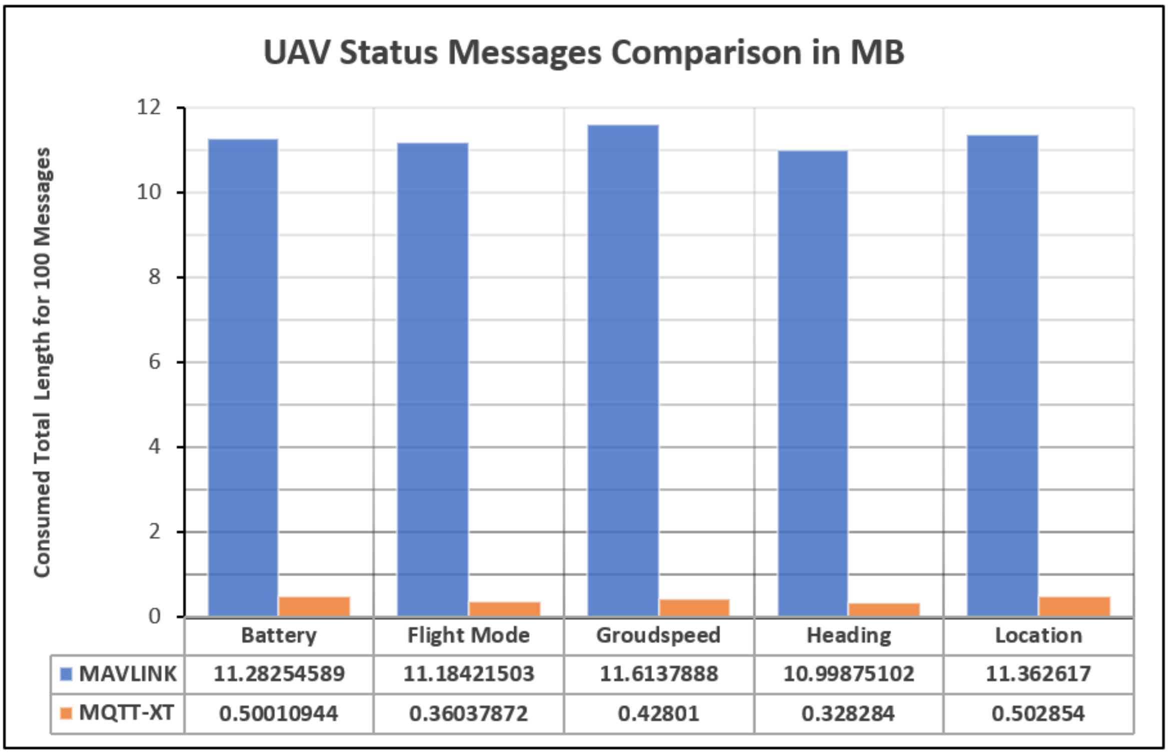
Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	2690	66.48	55	79	15.16	0.178831
80-159	14223	89.24	80	144	80.17	1.269261
160-319	829	286.03	286	314	4.67	0.237119
Total	**17742**	**94.98**	**55**	**314**	**100**	**1.685135**

Total Number of Message Length Distribution for MQTT-XT Land Messages

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	4300	70.42	68	78	84.03	0.302806
80-159	817	91.53	82	130	15.97	0.07478
160-319	0	0	0	0	0	0
Total	**5117**	**73.79**	**68**	**115**	**100**	**0.347956**

Figure 5.25 Traffic Message analysis for the Land command message

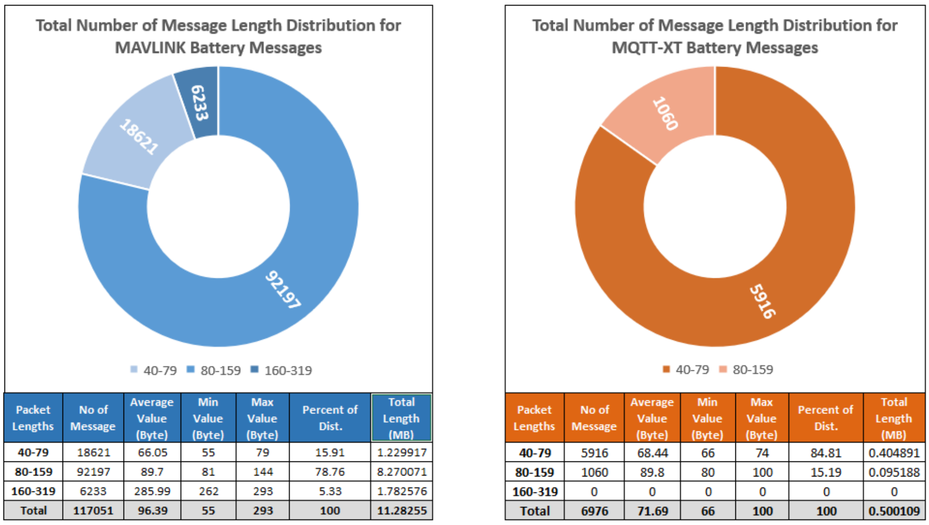
Status messages have different results than command messages for the MAVLINK. Status messages are transmitted continuously and automatically at specific rates by MAVLINK. However, obtaining exactly 100 selected status messages cause to increase in the size significantly on the status message set. The consumed total size to receive 5 different 100 status messages is given in Figure 5.20.



	Battery	Flight Mode	Groudspeed	Heading	Location
■ MAVLINK	11.28254589	11.18421503	11.6137888	10.99875102	11.362617
■ MQTT-XT	0.50010944	0.36037872	0.42801	0.328284	0.502854

Figure 5.26 UAV Status Messages Comparison in MB

The same payload distribution can also be seen for status messages. For example, the battery status is given in Figure 5.27. The packet count and sizes are still the same density level with the same byte lengths as obtained for command messages.

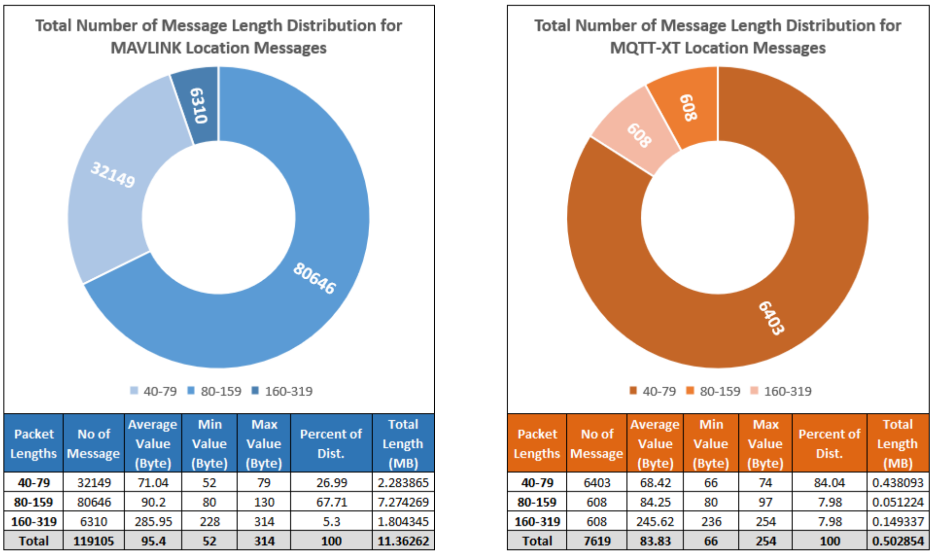


Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	18621	66.05	55	79	15.91	1.229917
80-159	92197	89.7	81	144	78.76	8.270071
160-319	6233	285.99	262	293	5.33	1.782576
Total	**117051**	**96.39**	**55**	**293**	**100**	**11.28255**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	5916	68.44	66	74	84.81	0.404891
80-159	1060	89.8	80	100	15.19	0.095188
160-319	0	0	0	0	0	0
Total	**6976**	**71.69**	**66**	**100**	**100**	**0.500109**

Figure 5.27 Traffic Message analysis for the Battery status message

The traffic message size result for the battery is almost 3 times greater than the command message set. However, the result is not changed dramatically for MQTT-XT since the stream function is not defined for MQTT-XT. But, as the packet lengths are increased, the traffic message size is also increased for MQTT-XT, which can be observed in Figure 5.28.

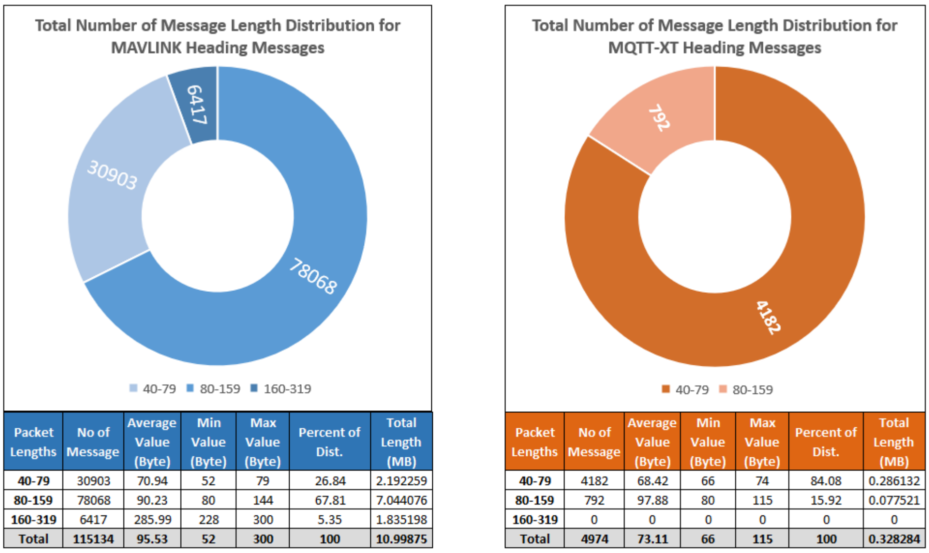


Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	32149	71.04	52	79	26.99	2.283865
80-159	80646	90.2	80	130	67.71	7.274269
160-319	6310	285.95	228	314	5.3	1.804345
Total	**119105**	**95.4**	**52**	**314**	**100**	**11.36262**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	6403	68.42	66	74	84.04	0.438093
80-159	608	84.25	80	97	7.98	0.051224
160-319	608	245.62	236	254	7.98	0.149337
Total	**7619**	**83.83**	**66**	**254**	**100**	**0.502854**

Figure 5.28 Traffic Message analysis for the Location status message

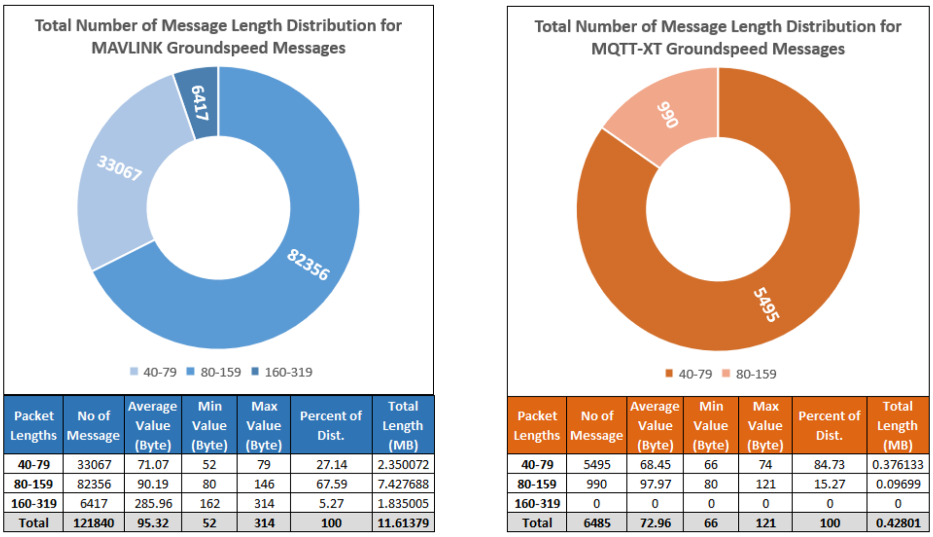
The location data is formed of longitude, latitude, and altitude data. Also, this data includes long numbers to transmit among the network. Therefore, unlike other MQTT-XT message sets, the location status message has a packet distribution between 160-319. But, it still outperforms the MAVLINK. The other status messages Heading, Groundspeed, and Flight Mode are respectively given in Figure 5.29, Figure 5.30, and Figure 5.31.



Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	30903	70.94	52	79	26.84	2.192259
80-159	78068	90.23	80	144	67.81	7.044076
160-319	6417	285.99	228	300	5.35	1.835198
Total	**115134**	**95.53**	**52**	**300**	**100**	**10.99875**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	4182	68.42	66	74	84.08	0.286132
80-159	792	97.88	80	115	15.92	0.077521
160-319	0	0	0	0	0	0
Total	**4974**	**73.11**	**66**	**115**	**100**	**0.328284**

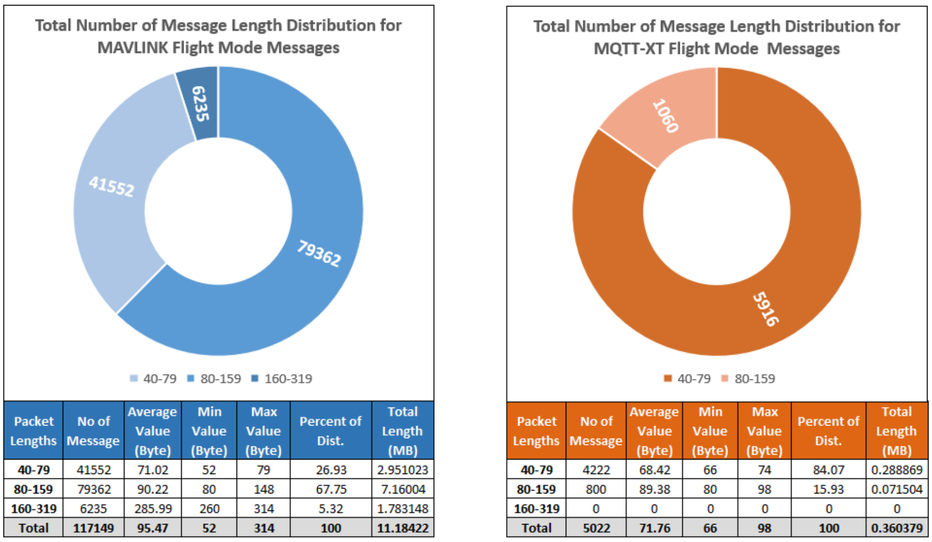
Figure 5.29 Traffic Message analysis for the Heading status message



Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	33067	71.07	52	79	27.14	2.350072
80-159	82356	90.19	80	146	67.59	7.427688
160-319	6417	285.96	162	314	5.27	1.835005
Total	**121840**	**95.32**	**52**	**314**	**100**	**11.61379**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	5495	68.45	66	74	84.73	0.376133
80-159	990	97.97	80	121	15.27	0.09699
160-319	0	0	0	0	0	0
Total	**6485**	**72.96**	**66**	**121**	**100**	**0.42801**

Figure 5.30 Traffic Message analysis for the Groundspeed status message



Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	41552	71.02	52	79	26.93	2.951023
80-159	79362	90.22	80	148	67.75	7.16004
160-319	6235	285.99	260	314	5.32	1.783148
Total	**117149**	**95.47**	**52**	**314**	**100**	**11.18422**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	4222	68.42	66	74	84.07	0.288869
80-159	800	89.38	80	98	15.93	0.071504
160-319	0	0	0	0	0	0
Total	**5022**	**71.76**	**66**	**98**	**100**	**0.360379**

Figure 5.31 Traffic Message analysis for the Flight Mode status message

5.4. Drone Management and IoT Sensor Data Gathering Scenarios Evaluation

The experiments of the scenarios are designed to reveal the performance of the protocols in different situations. The first 3 scenarios are related to managing a drone with a couple of messages. The task flow first scenario starts with the Arm command and continues with the takeoff, hold for 15 seconds and then land. For the second scenario, each drone has certain coordinates to arrive after an arm, takeoff sequence. After this step, they return to their takeoff positions and land. The third scenario includes only the arm, takeoff, and land command sequence.

The last scenario is similar to the second scenario except for the data transmission task. The task sequence starts with the arm, takeoff and continues with arriving given coordinates. After reaching a mission point, the transmission of IoT data starts and after finishing the transmission operation they return to the takeoff points and land exact position.

5.4.1. Mission Complete Time

The mission complete time is measured from the first arm command to disarming the last UAV. The mission complete time includes all processes of the mission. A total of 10 mission complete times are obtained per scenario to compare protocol behaviors. The average Mission complete time is presented in Figure 5.32

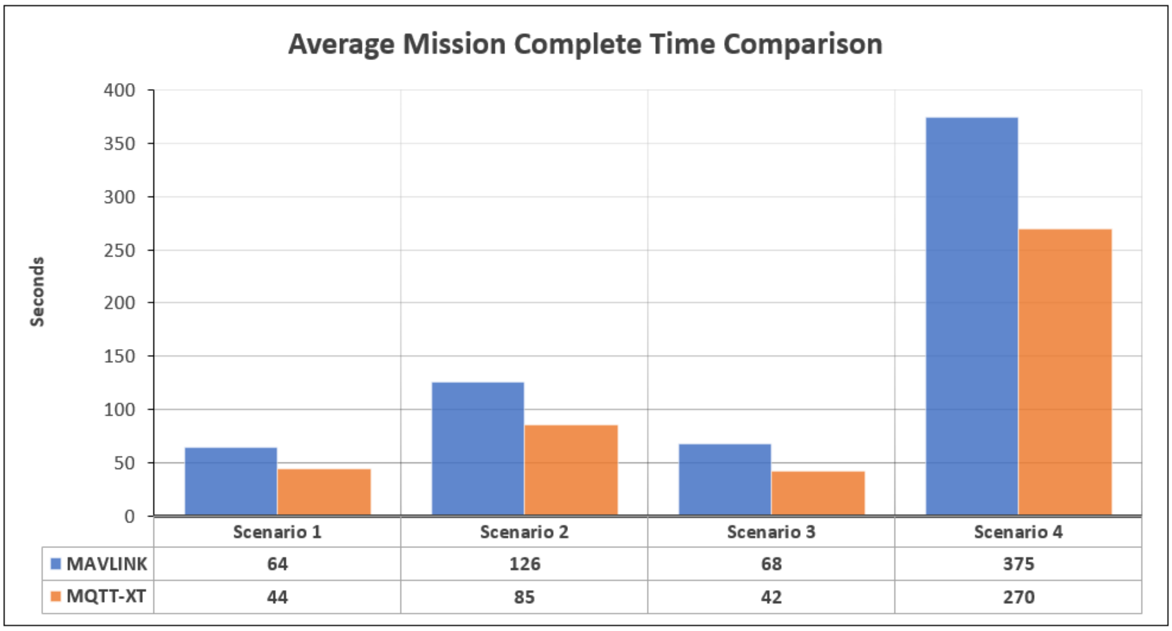
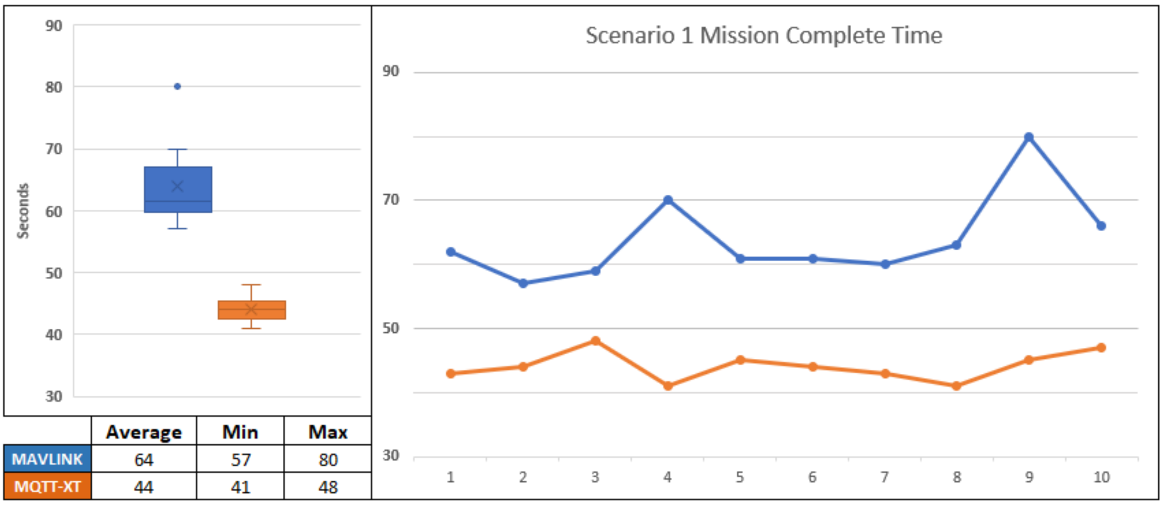


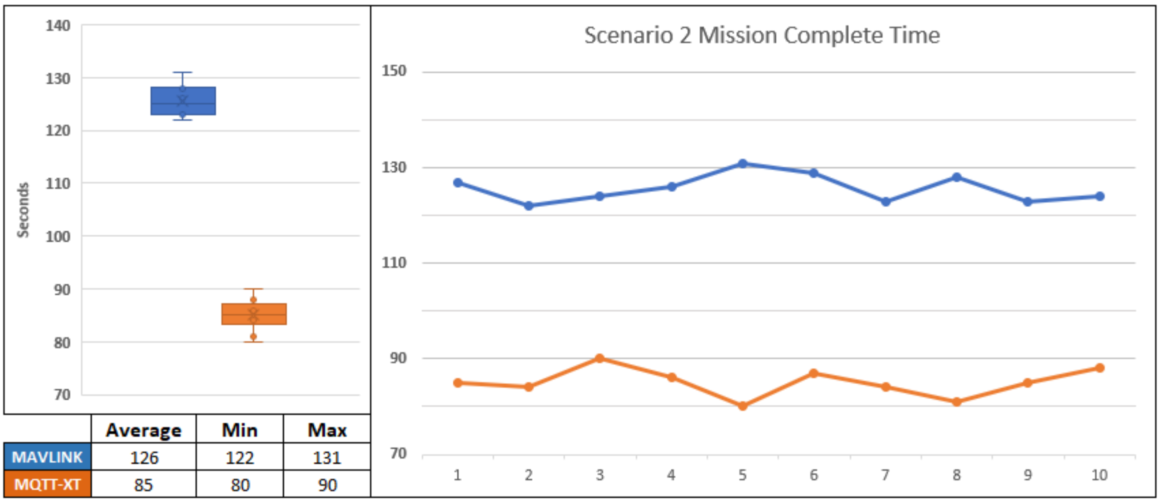
Figure 5.32 Average Mission Complete Time

The results show that the MQTT-XT completes missions approximately 30% faster than MAVLINK. The MQTT-XT utilizes communication for multiple vehicles with its broadcast function. Since MAVLINK aims to manage drones with peer-to-peer connections, some of the tasks are performed just in time. Also, for every 6 UAVs, the end-user and cloud need MAVLINK routers per connection and each fog needs 3 routers. The first 3 scenario mission complete time for 10 attempts and its distribution over time is given in Figure 5.33, Figure 5.34, and Figure 5.35.



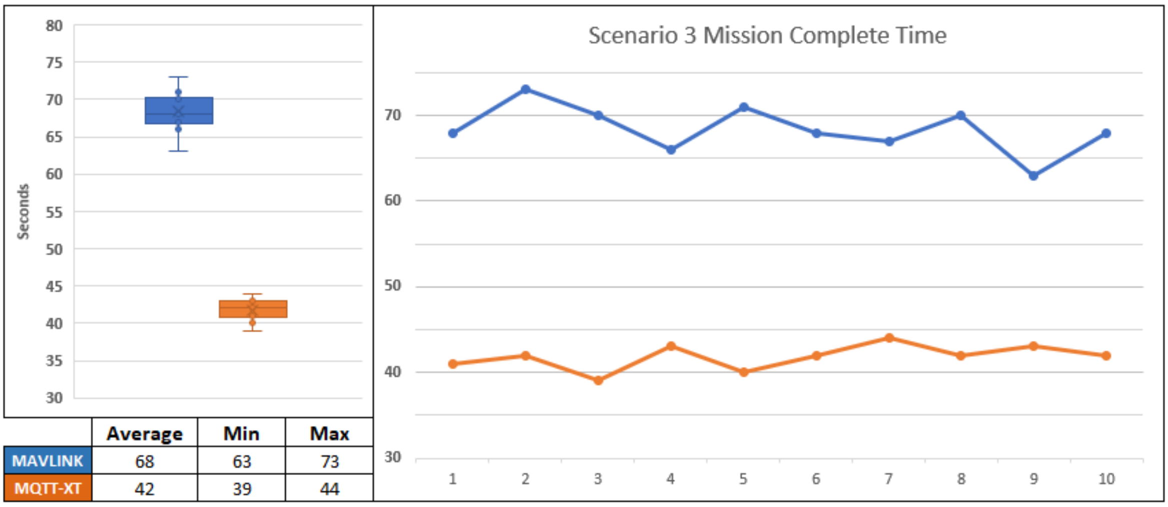
	Average	Min	Max
MAVLINK	64	57	80
MQTT-XT	44	41	48

Figure 5.33 Scenario 1 - Mission Complete Time



	Average	Min	Max
MAVLINK	126	122	131
MQTT-XT	85	80	90

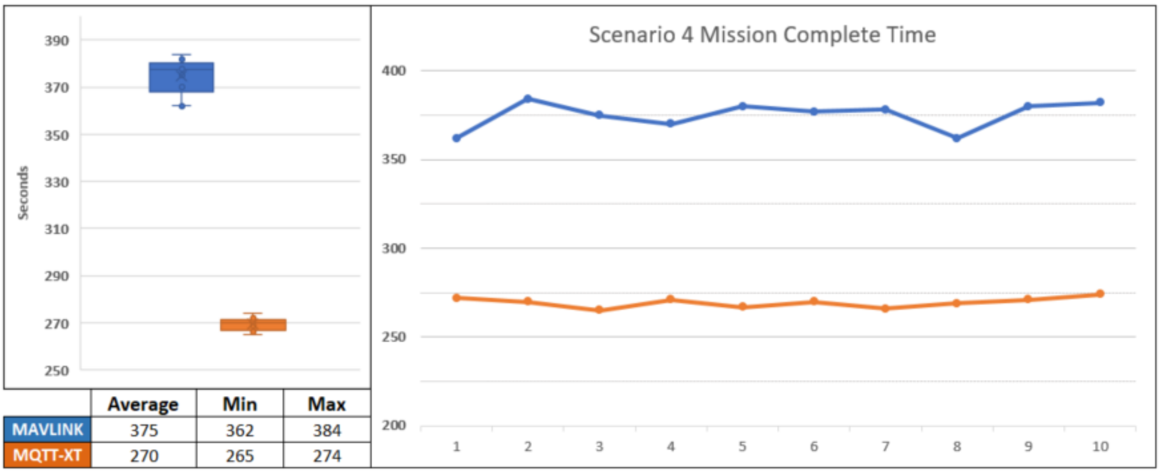
Figure 5.34 Scenario 2 - Mission Complete Time



	Average	Min	Max
MAVLINK	68	63	73
MQTT-XT	42	39	44

Figure 5.35 Scenario 3 - Mission Complete Time

The scenario 4 mission complete time also is originated from the extra connection for the IoT sensors' data gathering. The MQTT-XT uses its connection to transmit the data while the MAVLINK is not able to receive and transmit data to FoG. For this reason, alongside the MAVLINK connection, MQTT is established to provide sensor gathering operation. The mission complete time result of Scenario 4 is presented in Figure 5.36.

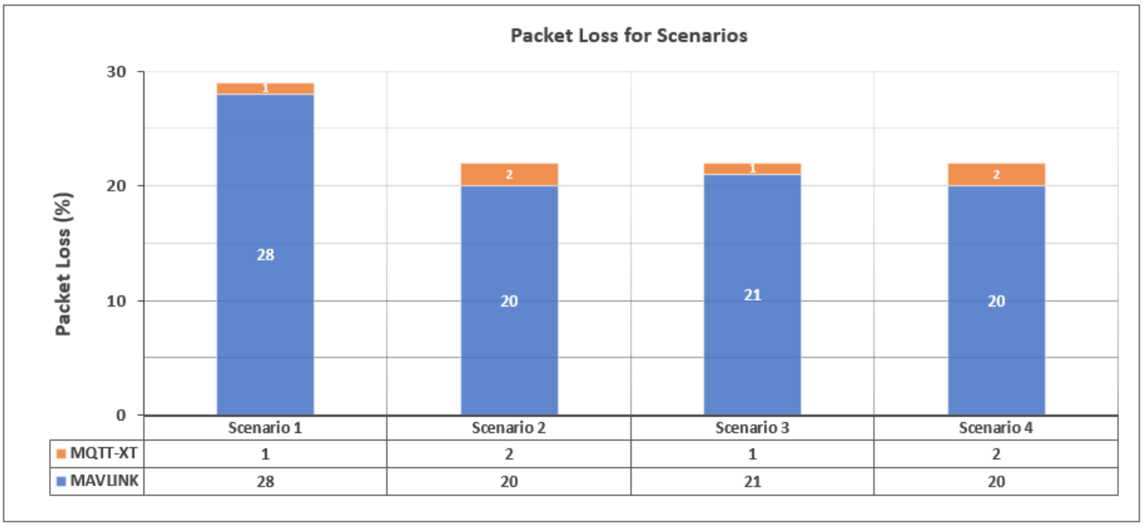


	Average	Min	Max
MAVLINK	375	362	384
MQTT-XT	270	265	274

Figure 5.36 Scenario 4 - Mission Complete Time

5.4.2. Packet Loss

Unlike the result of the message set, the packet loss is increased for the MAVLINK. As the MAVLINK streams different variables to the end user, most of them do not even arrive at the end user. Also, as the number of nodes is increased from 1 to 6, the MAVLINK has remarkable performance issues in terms of packet loss. However, the main aim of the MAVLINK is not to guarantee packet delivery. But, the MQTT-XT performs extremely better since the infrastructure of the MQTT-XT stands on the TCP. Thus, the packet loss ratio is approximately between 1-2% for MQTT-XT while this ratio goes up to 28%.



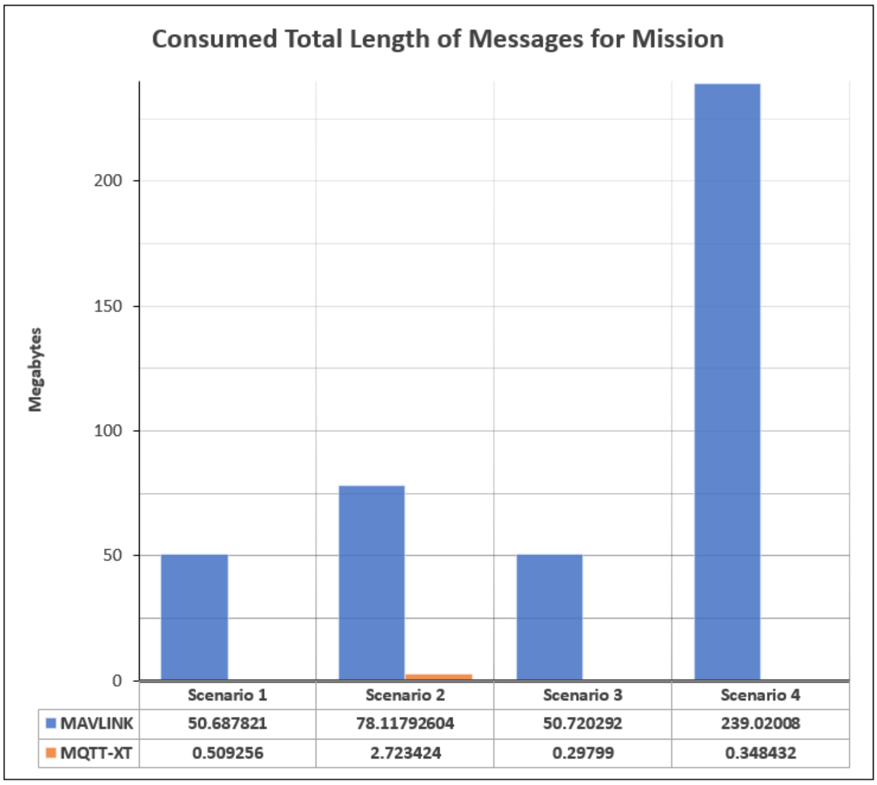
	Scenario 1	Scenario 2	Scenario 3	Scenario 4
🟧 MQTT-XT	1	2	1	2
🟦 MAVLINK	28	20	21	20

Figure 5.37 Packet Loss for Scenarios

5.4.3. Packet Count and Message Traffic Size

This evaluation criteria has the most remarkable results. The packet count and message traffic size rise drastically for MAVLINK, while these values are very significantly low for MQTT-XT. Since the MAVLINK is more talkative, the packet counts are extremely increased for it. In figure 5.38, the total consumed message sizes are given.

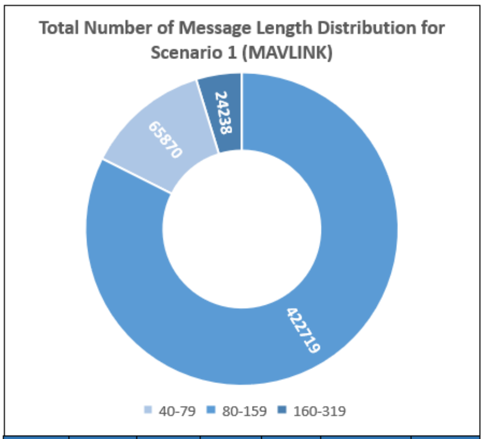
There is a directly proportional relationship between the elapsed time and the number of packets counted. On the other hand, the high number of UAVs greatly affects traffic density.

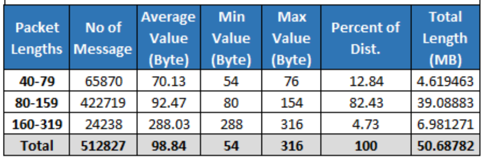


	Scenario 1	Scenario 2	Scenario 3	Scenario 4
■ MAVLINK	50.687821	78.11792604	50.720292	239.02008
■ MQTT-XT	0.509256	2.723424	0.29799	0.348432

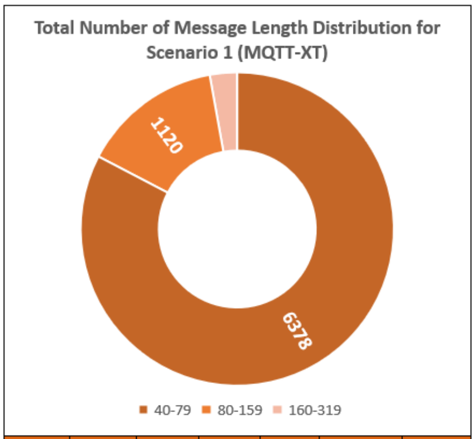
Figure 5.38 Total Consumed Message Size for the Scenarios

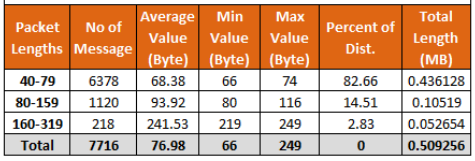
Because MAVLINK supports only peer connections for the UAVs and routers are only responsible for forwarding packets to end-users. The results of Scenario 1-3 are presented in Figure 5.39, Figure 5.40, and Figure 5.41.





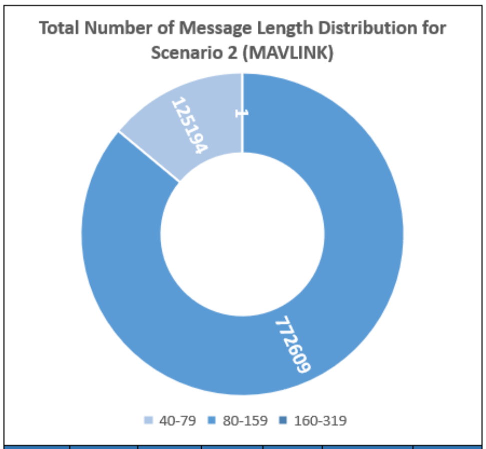
Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	65870	70.13	54	76	12.84	4.619463
80-159	422719	92.47	80	154	82.43	39.08883
160-319	24238	288.03	288	316	4.73	6.981271
Total	**512827**	**98.84**	**54**	**316**	**100**	**50.68782**

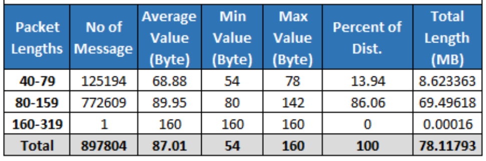




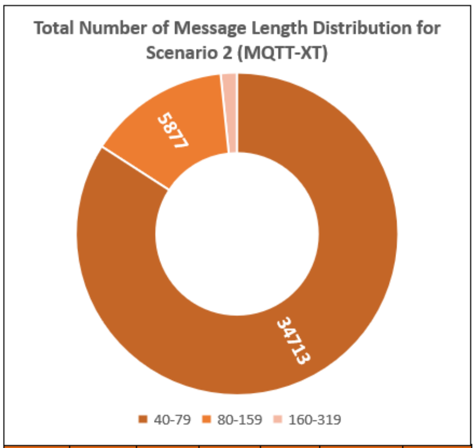
Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	6378	68.38	66	74	82.66	0.436128
80-159	1120	93.92	80	116	14.51	0.10519
160-319	218	241.53	219	249	2.83	0.052654
Total	**7716**	**76.98**	**66**	**249**	**0**	**0.509256**

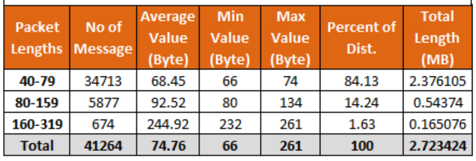
Figure 5.39 Traffic Message analysis for the Scenario 1





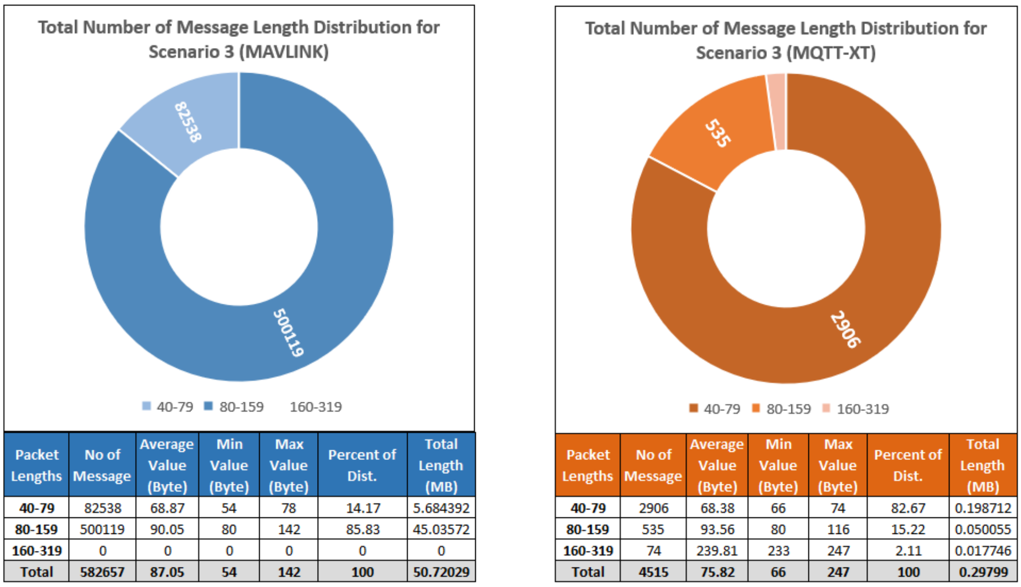
Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	125194	68.88	54	78	13.94	8.623363
80-159	772609	89.95	80	142	86.06	69.49618
160-319	1	160	160	160	0	0.00016
Total	**897804**	**87.01**	**54**	**160**	**100**	**78.11793**





Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	34713	68.45	66	74	84.13	2.376105
80-159	5877	92.52	80	134	14.24	0.54374
160-319	674	244.92	232	261	1.63	0.165076
Total	**41264**	**74.76**	**66**	**261**	**100**	**2.723424**

Figure 5.40 Traffic Message analysis for the Scenario 2

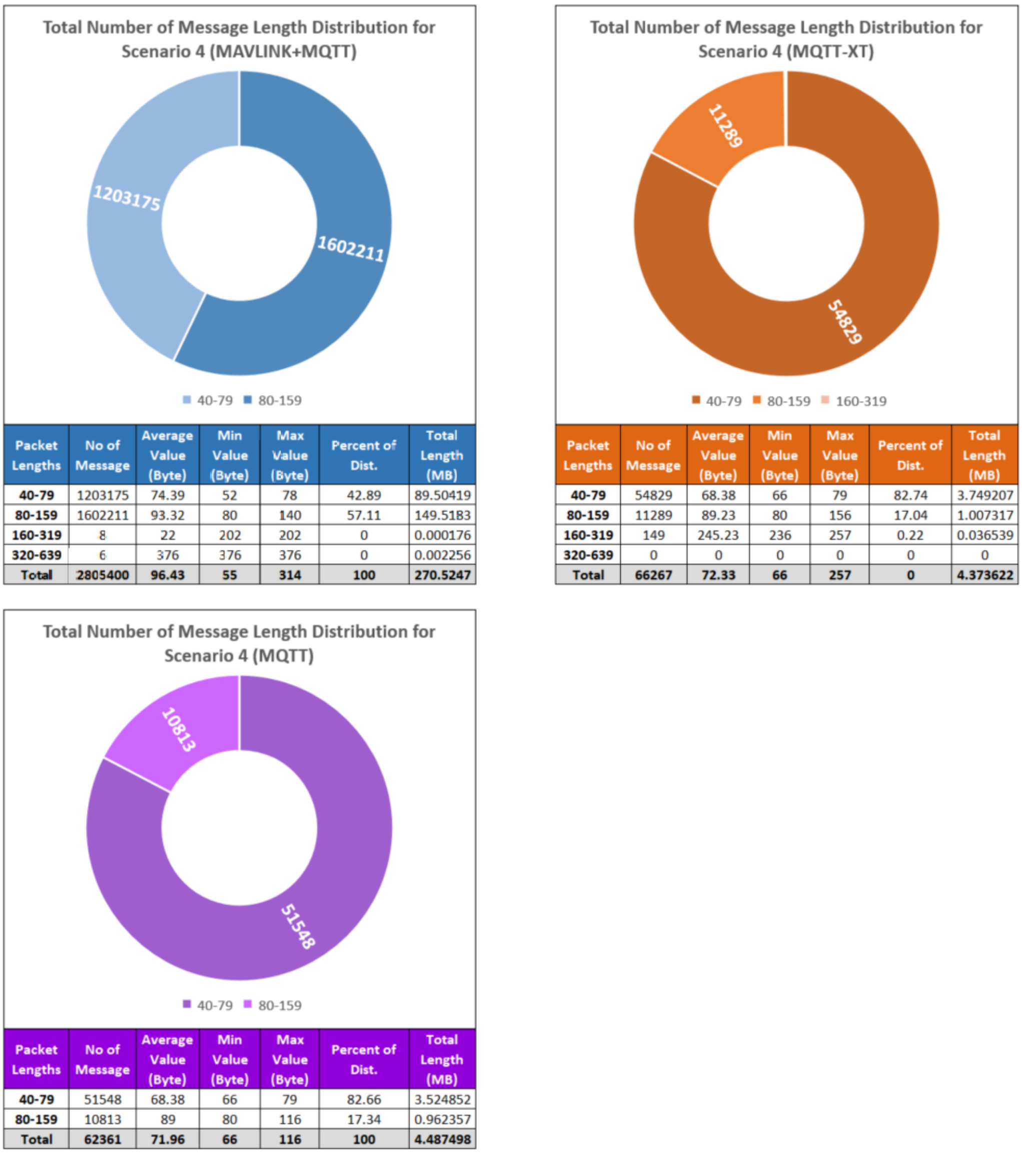


Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	82538	68.87	54	78	14.17	5.684392
80-159	500119	90.05	80	142	85.83	45.03572
160-319	0	0	0	0	0	0
Total	**582657**	**87.05**	**54**	**142**	**100**	**50.72029**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	2906	68.38	66	74	82.67	0.198712
80-159	535	93.56	80	116	15.22	0.050055
160-319	74	239.81	233	247	2.11	0.017746
Total	**4515**	**75.82**	**66**	**247**	**100**	**0.29799**

Figure 5.41 Traffic Message analysis for the Scenario 3

Scenario 4 has the most traffic messages among scenarios. It is almost 4 times greater than Scenario 1 and Scenario 3. However, the elapsed time is also greater by almost 5 times than others. However, the MQTT-XT does not spend much effort to complete a mission. Moreover, the great ratio of the message size comes from the sensor data. The MQTT-XT still has the same result as Scenario 2. The result of Scenario 4 is given in Figure 5.42.



Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	1203175	74.39	52	78	42.89	89.50419
80-159	1602211	93.32	80	140	57.11	149.5183
160-319	8	22	202	202	0	0.000176
320-639	6	376	376	376	0	0.002256
Total	**2805400**	**96.43**	**55**	**314**	**100**	**270.5247**

Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	54829	68.38	66	79	82.74	3.749207
80-159	11289	89.23	80	156	17.04	1.007317
160-319	149	245.23	236	257	0.22	0.036539
320-639	0	0	0	0	0	0
Total	**66267**	**72.33**	**66**	**257**	**0**	**4.373622**

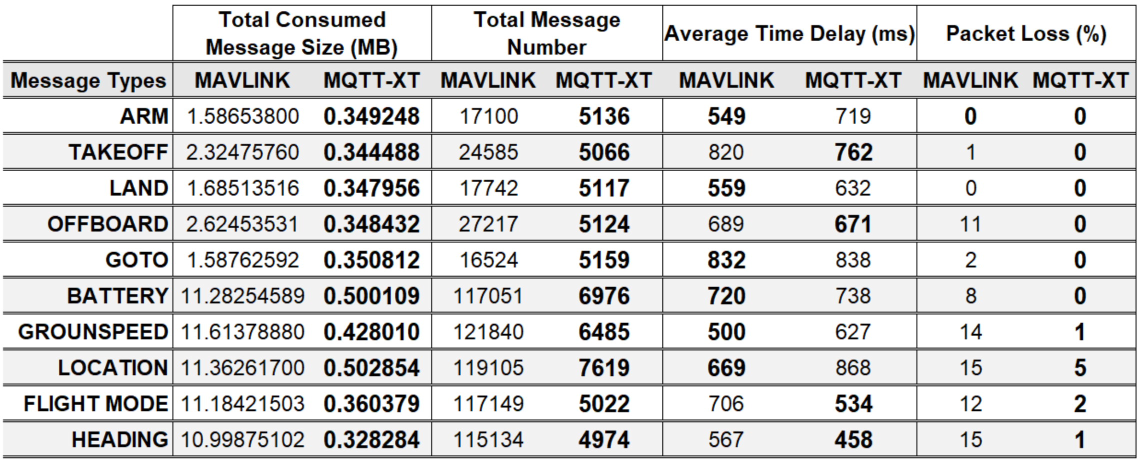
Packet Lengths	No of Message	Average Value (Byte)	Min Value (Byte)	Max Value (Byte)	Percent of Dist.	Total Length (MB)
40-79	51548	68.38	66	79	82.66	3.524852
80-159	10813	89	80	116	17.34	0.962357
Total	**62361**	**71.96**	**66**	**116**	**100**	**4.487498**

Figure 5.42 Traffic Message analysis for the Scenario 4

5.5. Discussion

For the evaluation, UDP-based MAVLINK and TCP-based MQTT-XT are compared in the Gazebo simulation environment. The evaluation is based on 2 different experimental results. The first experimental summary results, which are given in Table 5.3 present the 10 different command and status message types that provide to manage UAVs and to receive their status of them. For the first experimental results, 1 UAV platform, one Raspberry Pi to simulate

FoG, one cloud, and one End-user client were used. Each message type is sent or received 100 times for both protocols to obtain results.



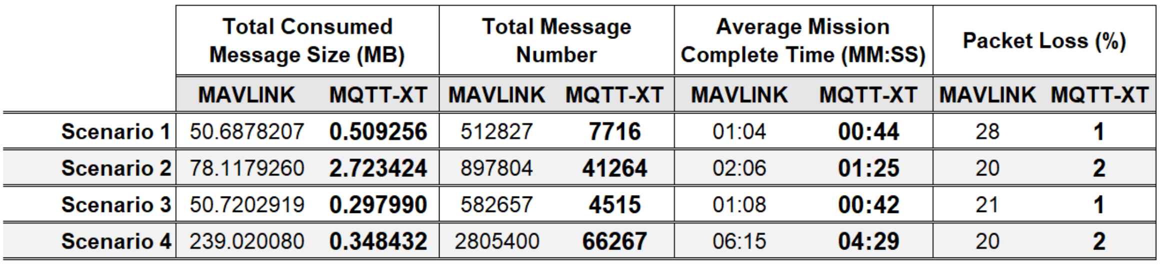
	Total Consumed Message Size (MB)		Total Message Number		Average Time Delay (ms)		Packet Loss (%)	
Message Types	MAVLINK	MQTT-XT	MAVLINK	MQTT-XT	MAVLINK	MQTT-XT	MAVLINK	MQTT-XT
ARM	1.58653800	**0.349248**	17100	**5136**	**549**	719	**0**	**0**
TAKEOFF	2.32475760	**0.344488**	24585	**5066**	820	**762**	1	**0**
LAND	1.68513516	**0.347956**	17742	**5117**	**559**	632	0	**0**
OFFBOARD	2.62453531	**0.348432**	27217	**5124**	689	**671**	11	**0**
GOTO	1.58762592	**0.350812**	16524	**5159**	**832**	838	2	**0**
BATTERY	11.28254589	**0.500109**	117051	**6976**	**720**	738	8	**0**
GROUNSPEED	11.61378880	**0.428010**	121840	**6485**	**500**	627	14	**1**
LOCATION	11.36261700	**0.502854**	119105	**7619**	**669**	868	15	**5**
FLIGHT MODE	11.18421503	**0.360379**	117149	**5022**	706	**534**	12	**2**
HEADING	10.99875102	**0.328284**	115134	**4974**	567	**458**	15	**1**

Table 5.3 Summary table of the experimental results for 100 command and status messages

The result shows that the MQTT-XT utilizes packet size at least 5 times better than MAVLINK for the command message set. In the status message set, MQTT-XT outperforms by shrinking at least 20 times. The main reason for the enormous difference between protocols is that MAVLINK streams many different messages, while MQTT-XT is more likely to focus on UAVs as IoT nodes and is designed to minimize packet sizes. Also, the number of packets to transmit or receive required messages is costly for MAVLINK. Even to receive or transmit 100 messages among nodes, it spends packets of at least more than 17000 for command types; 110000 for status types.

For the average time delay, the MAVLINK performs slightly better for 6 different message types since the principle of MAVLINK is fire and forget while MQTT-XT offers a more reliable connection for constrained environments. However, this difference causes packet loss for MAVLINK. The packet loss ratio is higher on MAVLINK. Especially, in the status message set, the packet loss ratio ups to 15 percent where it is 5 percent in MQTT-XT.

The second experimental result aims to reveal protocols' performances under more complicated network architectures with designed 4 different scenarios. For the first three scenarios are designed to control and manage six UAVs on two Raspberry Pi which simulate different located FoGs, one cloud, and one end-user is used. Scenario 4 includes the first 3 scenarios' purpose, but it is also designed to obtain three different sensors for each UAV and transmit data to the end user. The summary of the results for four different scenarios is given in Table 5.4.



	Total Consumed Message Size (MB)		Total Message Number		Average Mission Complete Time (MM:SS)		Packet Loss (%)	
	MAVLINK	MQTT-XT	MAVLINK	MQTT-XT	MAVLINK	MQTT-XT	MAVLINK	MQTT-XT
Scenario 1	50.6878207	**0.509256**	512827	**7716**	01:04	**00:44**	28	**1**
Scenario 2	78.1179260	**2.723424**	897804	**41264**	02:06	**01:25**	20	**2**
Scenario 3	50.7202919	**0.297990**	582657	**4515**	01:08	**00:42**	21	**1**
Scenario 4	239.020080	**0.348432**	2805400	**66267**	06:15	**04:29**	20	**2**

Table 5.4 Summary table of the experimental results for 4 different scenarios

The density of message traffic for MAVLINK is high. Also, as the MAVLINK has a heavy payload, the message sizes are extremely costly. The MQTT-XT uses packet sizes 100 times smaller than MAVLINK. Furthermore, in Scenario 4, the cost ups to 239 MB while it is only 0.34 MB for the MQTT-XT. The MQTT-XT also utilizes the network to serve better performance for multiple UAVs and mission complete time is 30% better than MAVLINK-based architecture.

The other main evaluation criterion is the packet loss ratio. The situation also differs from the message set test. Under heavy workload, the MAVLINK packet loss is between 20% and 28%. The MQTT-XT protocol provides a more reliable connection with its architecture and the packet loss ratio is between 1% and 2%. The main problem for the MAVLINK is that it aims to send packets as soon as possible and is not optimized for packet loss. The MQTT-XT is designed to work on constrained resources and it is more robust to communicate with enormous numbers of connected nodes.

6. CONCLUSION

In this thesis, a novel MQTT-XT protocol, also known as MAVLINK Enabled MQTT, is proposed to provide communication with UAVs over IoT networks while considering these devices as a part of the IoT. The main motivation is to close the gap between IoT devices and UAVs. The main problem is that the UAVs protocol differs from IoT protocols and the management of UAVs is not possible with the current IoT protocols. Since UAVs have their dialect protocol like MAVLINK, IoFT researchers are struggling to implement different protocols. Furthermore, the protocols are generally built on RF signals that are useful for line-of-sight operations but not for cloud operations.

With the advancement in technology, the requirements of the UAVs are changed and the MAVLINK provides communication over network layers to spread more different network layers. Thus, the researchers have the capability to build the MAVLINK with the network layer protocols like UDP and TCP. However, there are still problems with the perspective of IoFT. As well known, the IoT term includes many different devices which cause heterogeneous structures. When the number of UAVs is increasing, controlling and managing the network is becoming more complex.

For this reason, MQTT-XT plays a vital role in solving existing problems in IoFT, while on the other hand, it serves as an IoT protocol. The MQTT-XT has the capability of communicating with UAVs and IoT devices over MQTT. To provide this, MQTT-XT introduces 3 different client types, unlike the traditional MQTT approach client-server logic. The first client named UAV client has the responsibility to interpret MAVLINK and MQTT messages to each other. The main purpose of the UAV client is to be a bridge between IoFT devices by removing the limitation of MAVLINK. The second one is the activator client. It is designed for the FoG area and can be used on the cloud if the UAVs have direct communication with the cloud. The activator client provides communication between UAVs and users over FoG. The UAVs are directly connected to the activator client. The activator client has 2 different useful functions which are named QDoS and PSMI. Also, the user can use broadcast messages to communicate with all drones that are connected wtih activator client. Lastly, the end-user client is designed like the ground station, which aims to manage and control UAVs.

The future direction of this thesis is that implement all MAVLINK messages and prepare the MQTT-XT dialect with the MQTT topic structure. The activator client has a heavy workload when compared to other client types. The different artificial intelligence approaches can be applied to improve efficiency, reliability, and robustness. Also, the proposed protocol is designed with primitive functions and it is also not optimized. The main purpose and contribution of the protocol are to close the gap between IoT nodes and UAVs. Since the IoFT term is gaining more research areas, the network becomes a more complex structure. The MQTT-XT is designed to decrease the implementation process time and complexity of the IoFT area by offering a single protocol. Lastly, MQTT-XT is built on MQTT topics and security approaches are not considered in this thesis. The vulnerability of the MQTT-XT protocol can be reviewed in other studies.

REFERENCES

[1] Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu, and Hucheng Wang. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things journal*, 1(4):349–359, **2014**.

[2] Kashif Naveed, Chihiro Watanabe, and Pekka Neittaanmäki. The transformative direction of innovation toward an iot-based society-increasing dependency on uncaptured gdp in global ict firms. *Technology in Society*, 53:23–46, **2018**.

[3] Shadi Al-Sarawi, Mohammed Anbar, Kamal Alieyan, and Mahmood Alzubaidi. Internet of things (iot) communication protocols. In *2017 8th International conference on information technology (ICIT)*, pages 685–690. IEEE, **2017**.

[4] Mario Marchese, Aya Moheddine, and Fabio Patrone. Iot and uav integration in 5g hybrid terrestrial-satellite networks. *Sensors*, 19(17):3704, **2019**.

[5] Raed Kontar, Naichen Shi, Xubo Yue, Seokhyun Chung, Eunshin Byon, Mosharaf Chowdhury, Jionghua Jin, Wissam Kontar, Neda Masoud, Maher Nouiehed, et al. The internet of federated things (ioft). *IEEE Access*, 9:156071–156113, **2021**.

[6] CR Srinivasan, B Rajesh, P Saikalyan, K Premsagar, and Eadala Sarath Yadav. A review on the different types of internet of things (iot). *Journal of Advanced Research in Dynamical and Control Systems*, 11(1):154–158, **2019**.

[7] Bigi Varghese Philip, Tansu Alpcan, Jiong Jin, and Marimuthu Palaniswami. Distributed real-time iot for autonomous vehicles. *IEEE Transactions on Industrial Informatics*, 15(2):1131–1140, **2018**.

[8] Abdur Rahim Biswas and Raffaele Giaffreda. Iot and cloud convergence: Opportunities and challenges. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 375–376. IEEE, **2014**.

[9] Kazem Sohraby, Daniel Minoli, Benedict Occhiogrosso, and Wei Wang. A review of wireless and satellite-based m2m/iot services in support of smart grids. *Mobile Networks and Applications*, 23(4):881–895, **2018**.

[10] Qixun Zhang, Menglei Jiang, Zhiyong Feng, Wei Li, Wei Zhang, and Miao Pan. Iot enabled uav: Network architecture and routing algorithm. *IEEE Internet of Things Journal*, 6(2):3727–3742, **2019**.

[11] Paridhika Kayal and Harry Perros. A comparison of iot application layer protocols through a smart parking implementation. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, pages 331–336. **2017**. doi:10.1109/ICIN.2017.7899436.

[12] Nitin Naik. Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http. In *2017 IEEE International Systems Engineering Symposium (ISSE)*, pages 1–7. **2017**. doi:10.1109/SysEng.2017.8088251.

[13] Lilia Tightiz and Hyosik Yang. A comprehensive review on iot protocols' features in smart grid communication. *Energies*, 13(11):2762, **2020**.

[14] M Anusha, E Suresh Babu, L Sai Mahesh Reddy, A Vamsi Krishna, and B Bhagyasree. Performance analysis of data protocols of internet of things: a qualitative review. *International Journal of Pure and Applied Mathematics*, 115(6):37–47, **2017**.

[15] Biswajeeban Mishra and Attila Kertesz. The use of mqtt in m2m and iot systems: A survey. *IEEE Access*, 8:201071–201086, **2020**. doi:10.1109/ACCESS.2020.3035849.

[16] Tanin Sultana and Khan A Wahid. Choice of application layer protocols for next generation video surveillance using internet of video things. *IEEE Access*, 7:41607–41624, **2019**.

[17] Bryan Boyd, Joel Gauci, Michael P Robertson, Nguyen Van Duy, Rahul Gupta, Vasfi Gucer, Vladimir Kislicins, et al. *Building Real-time Mobile Solutions with MQTT and IBM MessageSight*. IBM Redbooks, **2014**.

[18] Navid Ali Khan, Noor Zaman Jhanjhi, Sarfraz Nawaz Brohi, and Anand Nayyar. Emerging use of uav's: secure communication protocol issues and challenges. In *Drones in smart-cities*, pages 37–55. Elsevier, **2020**.

[19] Anis Koubâa, Azza Allouch, Maram Alajlan, Yasir Javed, Abdelfettah Belghith, and Mohamed Khalgui. Micro air vehicle link (mavlink) in a nutshell: A survey. *IEEE Access*, 7:87658–87680, **2019**.

[20] Achilles D. Boursianis, Maria S. Papadopoulou, Panagiotis Diamantoulakis, Aglaia Liopa-Tsakalidi, Pantelis Barouchas, George Salahas, George Karagiannidis, Shaohua Wan, and Sotirios K. Goudos. Internet of things (iot) and agricultural unmanned aerial vehicles (uavs) in smart farming: A comprehensive review. *Internet of Things*, 18:100187, **2022**. ISSN 2542-6605. doi:https://doi.org/10.1016/j.iot.2020.100187.

[21] Fei Qi, Xuetian Zhu, Ge Mang, Michel Kadoch, and Wei Li. Uav network and iot in the sky for future smart cities. *IEEE Network*, 33(2):96–101, **2019**. doi:10.1109/MNET.2019.1800250.

[22] Andrey Giyenko and Young Im Cho. Intelligent uav in smart cities using iot. In *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, pages 207–210. **2016**. doi:10.1109/ICCAS.2016.7832322.

[23] N Nikhil, S M Shreyas, G Vyshnavi, and Sudha Yadav. Unmanned aerial vehicles (uav) in disaster management applications. In *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 140–148. **2020**. doi:10.1109/ICSSIT48917.2020.9214241.

[24] Waleed Ejaz, Muhammad Awais Azam, Salman Saadat, Farkhund Iqbal, and Abdul Hanan. Unmanned aerial vehicles enabled iot platform for disaster

management. *Energies*, 12(14), **2019**. ISSN 1996-1073. doi:10.3390/en12142706.

[25] Si Jung Kim, Yunhwan Jeong, Sujin Park, Kihyun Ryu, and Gyuhwan Oh. *A Survey of Drone use for Entertainment and AVR (Augmented and Virtual Reality)*, pages 339–352. Springer International Publishing, Cham, **2018**. doi:10.1007/978-3-319-64027-3_23.

[26] Metehan Unal, Erkan Bostanci, Evren Sertalp, Mehmet Serdar Guzel, and Nadia Kanwal. Geo-location based augmented reality application for cultural heritage using drones. In *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, pages 1–4. **2018**. doi:10.1109/ISMSIT.2018.8567073.

[27] Yassine Yazid, Imad Ez-Zazi, Antonio Guerrero-González, Ahmed El Oualkadi, and Mounir Arioua. Uav-enabled mobile edge-computing for iot based on ai: A comprehensive review. *Drones*, 5(4), **2021**. ISSN 2504-446X. doi:10.3390/drones5040148.

[28] Tiankui Zhang, Yu Xu, Jonathan Loo, Dingcheng Yang, and Lin Xiao. Joint computation and communication design for uav-assisted mobile edge computing in iot. *IEEE Transactions on Industrial Informatics*, 16(8):5505–5516, **2020**. doi:10.1109/TII.2019.2948406.

[29] Fadi Al-Turjman, Mohammad Abujubbeh, Arman Malekloo, and Leonardo Mostarda. Uavs assessment in software-defined iot networks: An overview. *Computer Communications*, 150:519–536, **2020**. ISSN 0140-3664. doi:https://doi.org/10.1016/j.comcom.2019.12.004.

[30] Hassan Daryanavard and Abbas Harifi. Uav path planning for data gathering of iot nodes: Ant colony or simulated annealing optimization. In *2019 3rd International Conference on Internet of Things and Applications (IoT)*, pages 1–4. **2019**. doi:10.1109/IICITA.2019.8808834.

[31] Sara Arabi, Essaid Sabir, Halima Elbiaze, and Mohamed Sadik. Data gathering and energy transfer dilemma in uav-assisted flying access network for iot. *Sensors*, 18(5), **2018**. ISSN 1424-8220. doi:10.3390/s18051519.

[32] Zehui Xiong, Yang Zhang, Wei Yang Bryan Lim, Jiawen Kang, Dusit Niyato, Cyril Leung, and Chunyan Miao. Uav-assisted wireless energy and data transfer with deep reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):85–99, **2021**. doi:10.1109/TCCN.2020.3027696.

[33] Omar Bouhamed, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud. A uav-assisted data collection for wireless sensor networks: Autonomous navigation and scheduling. *IEEE Access*, 8:110446–110460, **2020**. doi:10.1109/ACCESS.2020.3002538.

[34] Thomas Lagkas, Vasileios Argyriou, Stamatia Bibi, and Panagiotis Sarigiannidis. Uav iot framework views and challenges: Towards protecting drones as “things”. *Sensors*, 18(11), **2018**. ISSN 1424-8220. doi:10.3390/s18114015.

[35] Patrick McEnroe, Shen Wang, and Madhusanka Liyanage. A survey on the convergence of edge computing and ai for uavs: Opportunities and challenges. *IEEE Internet of Things Journal*, **2022**.

[36] Xiucheng Wang, Lianhao Fu, Nan Cheng, Ruijin Sun, Tom Luan, Wei Quan, and Khalid Aldubaikhy. Joint flying relay location and routing optimization for 6g uav–iot networks: A graph neural network-based approach. *Remote Sensing*, 14(17):4377, **2022**.

[37] Naser Hossein Motlagh, Miloud Bagaa, and Tarik Taleb. Uav-based iot platform: A crowd surveillance use case. *IEEE Communications Magazine*, 55(2):128–134, **2017**.

[38] Wen-Tsai Sung and Yen-Chun Chiang. Improved particle swarm optimization algorithm for android medical care iot using modified parameters. *Journal of medical systems*, 36(6):3755–3763, **2012**.

[39] Syed Agha Hassnain Mohsan, Muhammad Asghar Khan, Fazal Noor, Insaf Ullah, and Mohammed H Alsharif. Towards the unmanned aerial vehicles (uavs): A comprehensive review. *Drones*, 6(6):147, **2022**.

[40] Zhenhui Yuan, Jie Jin, Lingling Sun, Kwan-Wu Chin, and Gabriel-Miro Muntean. Ultra-reliable iot communications with uavs: A swarm use case. *IEEE Communications Magazine*, 56(12):90–96, **2018**.

[41] Amartya Mukherjee, Nilanjan Dey, and Debashis De. Edgedrone: Qos aware mqtt middleware for mobile edge computing in opportunistic internet of drone things. *Computer Communications*, 152:93–108, **2020**.

[42] K. Nuther-Saib. *A Distributed Architecture for Unmanned Aerial Systems Based on Publish/subscribe Messaging and Simultaneous Localisation and Mapping (SLAM) Testbed*. University of the Witwatersrand, Faculty of Science, School of Computer Science and Applied Mathematics, **2018**.

[43] Lea Matlekovic, Filip Juric, and Peter Schneider-Kamp. Microservices for autonomous uav inspection with uav simulation as a service. *Simulation Modelling Practice and Theory*, 119:102548, **2022**.

[44] Wei Meng, Yuanlin Yang, Jiayao Zang, Hongyi Li, and Renquan Lu. Dtuav: a novel cloud–based digital twin system for unmanned aerial vehicles. *Simulation*, 99(1):69–87, **2023**.

[45] Sivateja Reddy Vangimalla and Mohamed El-Sharkawy. Remote wireless sensor network range extension using uavs with thread protocol. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 902–906. **2018**. doi:10.1109/CSCI46756.2018.00178.

[46] Leandro Marcos da Silva, Henrique Bonini de Britto Menezes, Matheus dos Santos Luccas, Christian Mailer, Alex Sandro Roschildt Pinto, Adão Boava, Mariana Rodrigues, Isadora Garcia Ferrão, Júlio Cézar Estrella, and Kalinka Regina Lucas Jaquie Castelo Branco. Development of an efficiency platform based on mqtt for uav controlling and dos attack detection. *Sensors*, 22(17), **2022**. ISSN 1424-8220.

[47] Vito Scilimati, Antonio Petitti, Pietro Boccadoro, Roberto Colella, Donato Di Paola, Annalisa Milella, and Luigi Alfredo Grieco. Industrial internet of things at work: a case study analysis in robotic-aided environmental monitoring. *IET wireless sensor systems*, 7(5):155–162, **2017**.

[48] Yongs Zeng, Qingqing Wu, and Rui Zhang. Accessing from the sky: A tutorial on uav communications for 5g and beyond. *Proceedings of the IEEE*, 107(12):2327–2375, **2019**.

[49] Bin Li, Zesong Fei, Yan Zhang, and Mohsen Guizani. Secure uav communication networks over 5g. *IEEE Wireless Communications*, 26(5):114–120, **2019**.

[50] Bin Li, Zesong Fei, and Yan Zhang. Uav communications for 5g and beyond: Recent advances and future trends. *IEEE Internet of Things Journal*, 6(2):2241–2263, **2018**.

[51] Shuhang Zhang, Hongliang Zhang, Boya Di, and Lingyang Song. Cellular uav-to-x communications: Design and optimization for multi-uav networks. *IEEE Transactions on Wireless Communications*, 18(2):1346–1359, **2019**. doi:10.1109/TWC.2019.2892131.

[52] Soumya Kanti Datta, Jean-Luc Dugelay, and Christian Bonnet. Iot based uav platform for emergency services. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 144–147. **2018**. doi:10.1109/ICTC.2018.8539671.

[53] Han Hu, Cheng Zhan, Jianping An, and Yonggang Wen. Optimization for http adaptive video streaming in uav-enabled relaying system. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, **2019**.

[54] Sara Mahmoud, Nader Mohamed, and Jameela Al-Jaroodi. Integrating uavs into the cloud using the concept of the web of things. *Journal of Robotics*, 2015, **2015**.

[55] HaiYang Chao, YongCan Cao, and YangQuan Chen. Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, 8(1):36–44, **2010**.

[56] Robert L Lidowski, Barry E Mullins, and Rusty O Baldwin. A novel communications protocol using geographic routing for swarming uavs performing a search mission. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–7. IEEE, **2009**.

[57] Mahdi Asadpour, Domenico Giustiniano, Karin Anna Hummel, and Simon Egli. Uav networks in rescue missions. In *Proceedings of the 8th ACM international workshop on Wireless network testbeds, experimental evaluation & characterization*, pages 91–92. **2013**.

[58] C Carl Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, **1957**.