**GenSec Industries: Research document**

  
  
  
  
  
  
  
  
  
  
  
Johnson Domacasse 4471709  
Nedyalko Tenev 4808231  
Tuan Kiet Ho 4832649  
Farros Ramzy 3767353

[1. Introduction: 2](#_Toc76110575)

[2. Protocol Research Collection: 3](#_Toc1773962450)

[2.1 MQTT Protocol 3](#_Toc1921810815)

[2.1.1 MQTT Introduction 3](#_Toc1353895858)

[2.1.2 Is MQTT the ideal solution for IoT projects. 3](#_Toc36768917)

[2.1.3 MQTT Conclusion. 4](#_Toc2103503647)

[2.2 Gossip Protocol 5](#_Toc77600260)

[2.2.1 Gossip Introduction 5](#_Toc981894534)

[2.2.2 Gossip Performance 5](#_Toc2143134112)

[2.2.3 Gossip properties 6](#_Toc1451771567)

[2.2.4 Gossip Conclusions 6](#_Toc363721346)

[2.3 Protocol Conclusion: 6](#_Toc1361625985)

[3. References: 7](#_Toc404759484)

# Introduction:

*The focus of this document will be to hold all the information that was gathered when doing research regarding different parts of the occupancy management system. Different research topics and methods with relevancy to the occupancy management system can be found below.*

# Protocol Research Collection:

(**In this section we will talk about which topics were researched and why they are relevant to the system itself)**

## 2.1 MQTT Protocol

### 2.1.1 MQTT Introduction

This section discusses the importance of choosing an efficient method of communication when implementing an IoT solution and focuses on the research question of using MQTT as a communication protocol for multiple devices connected over a network. This research section aims to evaluate whether MQTT is compatible with the team's requirements, and sub-questions are provided, such as understanding MQTT, its applications and benefits, best practices for implementing it, and how to enable secure communication and data exchange among multiple devices over the network. The document also discusses the research strategy employed, which includes methods such as literature study, available product analysis, multicriteria decision making, and hardware validation.

When implementing a IoT (Internet of Things) solution, it is very important to choose an efficient method for devices to communicate to each other. The research question "**How can MQTT be used to let multiple devices communicate over network connections?**" focuses on the use of MQTT (Message Queuing Telemetry Transport) as a communication protocol to facilitate data exchange among multiple devices connected over a network. Along with these, additional sub-questions were defined to support the main research questions:

1. What is MQTT, and how does it work? What hardware does it require?
2. What are the applications and benefits of MQTT in facilitating communication among multiple devices?
3. What are the best practices and considerations for implementing MQTT in a networked environment?
4. How can MQTT be used to enable secure communication and data exchange among multiple devices over the network?

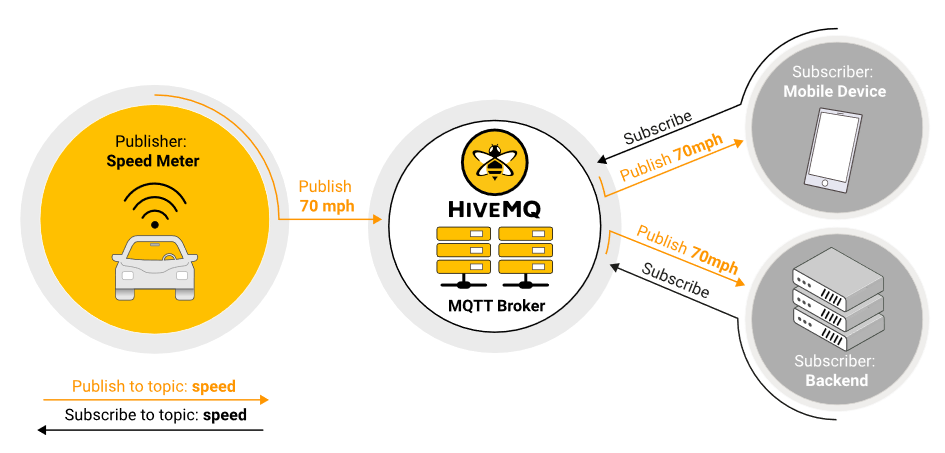
Answering these questions will help sufficiently in choosing the correct solution for the project. The outcome of the research should also provide the team with enough information to estimate the required effort for implementation.

### 2.1.2 Is MQTT the ideal solution for IoT projects.

This research involves the evaluation of similar protocols for Internet of Things systems. There are a lot of alternatives to the MQTT technology, like using a cloud-based solution or a server command system. However, to make sure that there is not a better choice for communication over a network, going over these options will show us the advantages and disadvantages of these technologies. See section **2.2 Gossip protocol** for more information.

Firstly, let us start with the specification of MQTT. It is a lightweight binary protocol which is good in transferring data over the wire in comparison to other protocols, like HTTP. Also, it is quite easy to implement on the client side, which helps with constrained devices with limited resources.

It works in a pattern of Publish / Subscribe. This means that the connection between the publisher and the client is decoupled. As they never contact each other directly, (in fact, they are not even aware that the other exists), the connection between them is handled by a third component (the broker). It filters all messages and distributes them to subscriber.



*Figure 1: MQTT example.*

There are a couple of aspects which we need to remember when working with a Publish / Subscribe system. Firstly, there are some benefits. The publisher and subscriber do not need to know each other, as they do not exchange any information about themselves like IP address or port. Also, they do not need to be run at the same time. This time decoupling allows the system to take disconnections without problem, because if the broker and the publisher are stable, subscribers can receive data at any time. However, there is also a negative – operations on both the publisher and the subscriber should not be interrupted during publishing or receiving.

To determine whether a subscribing client gets the message or not, MQTT uses subject-based filtering. This means that it differs a lot from the “message queue” systems, where a message is only consumed by one client and is only stored until it is consumed.

The MQTT protocol is based on TCP/IP, which means that both the client and the server have a TCP/IP stack. In most cases, they are located behind a router that uses NAT.

To make this work for an Arduino Uno, a library called ArduinoMqttClient is used. On the publisher device, a network name & password are set, together with a broker server. A message & a topic is being sent to the broker every time a change is found in the temperature sensor of the device. Another subscriber device is set up in a similar condition, but a poll function is added, so it can keep up with the latest information from the server.

### 2.1.3 MQTT Conclusion.

As a lightweight and efficient protocol, MQTT is widely used in the IoT and M2M communication fields, providing real-time data transfer and easy integration with various devices and systems.

## 2.2 Gossip Protocol

### 2.2.1 Gossip Introduction

Another message broadcasting technique in a distributed system is the Gossip protocol. The gossip protocol, also known as the epidemic protocol, is a decentralized peer-to-peer communication technique to transmit messages in an enormous distributed system.**[6][11]** The key concept of gossip protocol is that every node periodically sends out a message to a subset of other random nodes.**[11][7 ]** The entire system will receive this message eventually with a high probability.**[12][8]** Simply put, the gossip protocol is a technique for nodes to build a global map through limited local interactions. **[6]**

The gossip protocol is built on a robust, scalable, and eventually consistent algorithm. The gossip protocol is typically used to maintain the node membership list, achieve consensus, and fault detection in a distributed system.**[7]**

### 2.2.2 Gossip Performance

The number of nodes that will receive the message from a particular node is known as the fanout. The count of gossip rounds required to spread a message across the entire cluster is known as the cycle.**[11][10]**

For instance, it takes approximately 15 gossip rounds to propagate a message across 25,000 nodes. The gossip interval can be set to a value as low as 10 Ms to propagate a message across a big data centre in roughly 3 seconds. The propagation of a message in the gossip protocol should automatically age out to reduce the unnecessary load **[9]**. The performance of a gossip protocol implementation can be measured with the following metrics. **[11].**

* residue: number of remaining nodes that haven’t received the messages should be minimum
* traffic: average number of messages sent between nodes should be minimum
* convergence: every node should receive the message as quickly as possible
* time average: average time taken to send the message to every node should be low
* time last: the time taken for the last node to receive the message should be low

A case study showed that a system with 128 nodes consumed less than 2 percent of CPU and less than 60 KBps of bandwidth to run gossip protocol.**[12]**

2.2.3 Gossip properties  
There is no formal way to define gossip protocol. In general, the gossip protocol is expected to satisfy the following properties [**11]**:

* node selection must be random to perform a fanout
* only local information is available to every node and the nodes are oblivious to the state of the cluster
* communication between nodes involves periodic, pairwise, inter-process interactions
* bounded size transmission capacity per gossip round
* every node deploys the same gossip protocol
* unreliable network paths between nodes are assumed
* node interaction frequency is low
* node interactions result in a state exchange

### 2.2.4 Gossip Conclusions

The overall conclusion regarding the gossip protocol is that would be efficient for our system due to its easy failure detection. It was not the choice over MQTT due to its increased latency. Its way of distributing information and its difficulty to debug or test.   
 The usage of the gossip protocol results in increased latency because the node must wait for the next gossip cycle (interval) to transmit the message. The message does not trigger the gossip exchange, but the gossip protocol interval timer does.**[10]**  
 The protocol does not ensure that messages are received at a fixed rate and a specific location. The messages are being “thrown” around as much as possible so that it can be read by all the nodes. **[5]**  
 Because of its unpredictable behaviour, this protocol can be quite difficult to unit test or debug later during implementation.**[7]**  
 These three topics are the reason we will not be using the Gossip protocol within our system.

## 2.3 Protocol Conclusion:

Based on the information gathered on both protocols, we have concluded that we will be using the MQTT protocol for the following reasons:

1. **Easy to use**: The MQTT protocol is easy to use. Next to this, it is a protocol that we have some experience implementing.
2. **Latency**: Compared to the gossip protocol, the latency is significantly less. This is because the node must wait for the next gossip cycle (interval) to transmit the message. Although this would be good for a heartbeat implementation, it would not be good for situations to handle fast check-ins for example.
3. **Unpredictable behaviour:** On top of this we do not want the messages to be “thrown” around like they do in the gossip protocol. We need the messages sent over to the other modules in a robust matter.
4. **Difficult to test:** Due to its unpredictable behaviour, testing the program later would prove difficult.

The fact that MQTT is also not the best protocol to be used for this project is also aware to us. For the sake of the skeleton, we will be using MQTT. If applicable, we have an idea of what we would like to do regarding the protocol for multiple divider components.

# 3. References:

[1] - HBOi. (2021, July). *The DOT Framework - ICT research methods*.  
 <https://ictresearchmethods.nl/The_DOT_Framework>

[2] - *Getting to know MQTT*. IBM developer. (n.d.). <https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/>

[3] - Team, H. (n.d.). *MQTT client, MQTT broker, and MQTT Server Connection Establishment explained – mqtt essentials: Part 3*. <https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment/>

[4] - Team, T. A. (n.d.). *Sending data over MQTT*. Arduino Documentation. <https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-device>

[5] - *Gossip protocol explained*. High Scalability. (2023, July 16). <http://highscalability.com/blog/2023/7/16/gossip-protocol-explained.html>

[6] - Gupta, P. (2022, April 27). *Gossip protocol in distributed systems*. Medium. <https://medium.com/nerd-for-tech/gossip-protocol-in-distributed-systems-e2b0665c7135>

[7] - Systems, D. (2023, March 13). *How do you integrate a gossip system with other distributed components and services?*. Gossip System Integration: A Guide for Distributed Systems. <https://www.linkedin.com/advice/0/how-do-you-integrate-gossip-system-other-distributed>

[8] - Klepmann, M. (2020, October 28). *Distributed Systems 4.3: Broadcast algorithms*. YouTube. <https://www.youtube.com/watch?v=77qpCahU3fo>

[9] - Dutta, B. (2022). *A gentle introduction to gossip protocol*. Analytics Steps. <https://www.analyticssteps.com/blogs/gentle-introduction-gossip-protocol>

[10] - Acuna, G. (2020, June 1). *Parallel & Distributed Computing - Gossip protocol*. YouTube. <https://www.youtube.com/watch?v=qJpPjzg44R8>

[11] - López, F. (2016, April 17). *Introduction to gossip*. Just my thoughts. <https://managementfromscratch.wordpress.com/2016/04/01/introduction-to-gossip/>

[12] - Hoff, T. (2011). *Using gossip protocols for failure detection, monitoring, messaging and other good things*. High Scalability. <http://highscalability.com/blog/2011/11/14/using-gossip-protocols-for-failure-detection-monitoring-mess.html>