# Different Communication Protocols :

Smetankin, Daniel

## 1. Intro

In this report I will be looking into different communication protocols. And compare them at the end to then conclude what the best communication protocol for our project is.

## 2.TCP/IP

TCP/IP, or the Transmission Control Protocol/Internet Protocol. is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP can also be used as a communications protocol in a private network (an intranet or an extranet). ***(3)***

With TCP, a connection is established which verifies an entity is listening and expected to receive and send data, and this connection maintains data packet ordering and delivery so an application just needs to establish the connection and then data is guaranteed to get to the destination in the order it was sent or the socket API will return an error if it cannot. This connection is extra overhead as it takes a request, an ack to the request, and finally an ack to the ack to establish a connection.***(3)***

TCP/IP is suitable for purposes where error checking and correction are the number 1 priority. And the speed of witch the packets arrive are not as important.

## 3. UDP

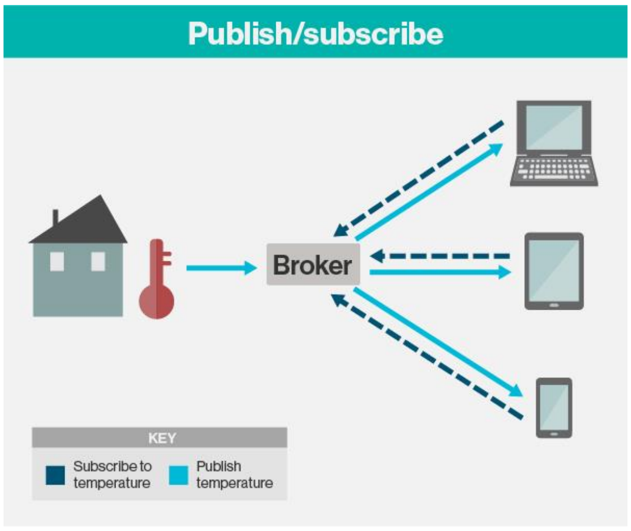
UDP uses a simple connectionless communication model with a minimum of protocol mechanism. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; There is no guarantee of delivery, ordering, or duplicate protection. If error-correction facilities are needed at the network interface level, an application may use the Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP) which are designed for this purpose.***(4)***

UDP is suitable for purposes where error checking and correction are either not necessary or are performed in the application; UDP avoids the overhead of such processing in the protocol stack. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for packets delayed due to retransmission, which may not be an option in a real-time system. ***(4)***

# 3. MQTT

MQTT (MQ Telemetry Transport) is a lightweight messaging protocol.that provides resource-constrained network clients with a simple way to distribute telemetry information. The protocol, which uses a publish/subscribe communication pattern, is used for machine-to-machine (M2M) communication and plays an important role in the internet of things.***(2)***

The MQTT protocol is a good choice for wireless networks that experience varying levels of latency due to occasional bandwidth constraints or unreliable connections. Should the connection from a subscribing client to a broker get broken, the broker will buffer messages and push them out to the subscriber when it is back online. Should the connection from the publishing client to the broker be disconnected without notice, the broker can close the connection and send subscribers a cached message with instructions from the publisher.***(2)***



**Client**
When talking about a client it almost always means an MQTT client. This includes publisher or subscribers, both of them label an MQTT client that is only doing publishing or subscribing. (In general a MQTT client can be both a publisher & subscriber at the same time). ***A MQTT client is any device from a micro controller up to a full fledged server, that has a MQTT library running and is connecting to an MQTT broker over any kind of network***. This could be a really small and resource constrained device, that is connected over a wireless network and has a library strapped to the minimum or a typical computer running a graphical MQTT client for testing purposes, basically any device that has a TCP/IP stack and speaks MQTT over it. The client implementation of the MQTT protocol is very straight-forward and really reduced to the essence. That's one aspect, why MQTT is ideally suitable for small devices. MQTT client libraries are available for a huge variety of programming languages, for example Android, Arduino, C, C++, C#, Go, iOS, Java, JavaScript.***(1)***

**Broker**
The counterpart to a MQTT client is the MQTT broker, which is the heart of any publish/subscribe protocol. Depending on the concrete implementation, a broker can handle up to thousands of concurrently connected MQTT clients. The broker is primarily responsible for receiving all messages, filtering them, decide who is interested in it and then sending the message to all subscribed clients. It also holds the session of all persisted clients including subscriptions and missed messages (More details). Another responsibility of the broker is the authentication and authorization of clients. And at most of the times a broker is also extensible, which allows to easily integrate custom authentication, authorization and integration into backend systems. Especially the integration is an important aspect, because often the broker is the component, which is directly exposed on the internet and handles a lot of clients and then passes messages along to downstream analyzing and processing systems. As we described in one of our early blog post subscribing to all message is not really an option. All in all the broker is the central hub, which every message needs to pass. Therefore it is important, that it is highly scalable, integratable into backend systems, easy to monitor and of course failure-resistant. For example HiveMQ solves this challenges by using state-of-the-art event driven network processing, an open plugin system and standard providers for monitoring.***(1)***

## 4. Conclussion.

TCP/IP would be an option that would work but certainly not the best option. It focusses on error checking and correction of packages and less on the speed of the packages. In our project reaction speed will be crucial to avoid collision between cars. Therefore UDP would be a great option as if focuses mainly on the speed of witch the packages are transmitted. MQTT works on top of the TCP/IP protocol so it won´t deliver the packages at the speed UDP can. But this disadvantage is compensated by the ***broker*** witch gathers information’s from different nodes , we are using 3 cars who all are connected to the server(computer) so the MQTT protocol is also a good option we could consider. So there is no “best” protocol there are in fact 2 good one´s UDP and MQTT , the choice between these 2 is defined by the programming environment we will chose and the MCU we will choose.

## Reference list

1. https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker-connection-establishment
2. http://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport
3. https://www.miser-tech.com/2015/05/esp8266-step-2-udp-vs-tcp-review/
4. https://www.diffen.com/difference/TCP_vs_UDP
5. https://en.wikipedia.org/wiki/MQTT