Measuring and Adapting MQTT in Cellular Networks for Collaborative Smart Farming

Jan Bauer and Nils Aschenbruck
University of Osnabrück, Institute of Computer Science
Wachsbleiche 27, 49090 Osnabrück, Germany
Email: {bauer, aschenbruck}@uos.de

***Abstract*—In distributed Smart Farming applications, a reliable communication is often crucial, in particular during collaborating operations. In rural areas, the network connectivity via Public Land Mobile Networks is, however, not always sufficient. Network disruptions may occur hindering reliable transmissions. For the delivery of periodic sensor data streams of agricultural machines, several modern communication frameworks adopt the Message Queue Telemetry Transport (MQTT) protocol. Although MQTT is built on top of TCP, a reliable delivery, even of important data, cannot generally be guaranteed in rural environments. Due to handovers and spatial dead zones, this is in particular the case when clients are mobile.**

This paper presents a case-study-based performance evaluation of MQTT regarding a reliable data transmission in real-world scenarios with mobile clients. Therefore, we investigate both links, publisher to broker and broker to subscriber, separately. By doing so, we show that both links significantly benefit from a suitable parameterization of MQTT. Moreover, for many data streams in the considered scenario, the default FIFO queuing strategy of MQTT is not always the most suitable approach to cope with network disruptions. Hence, we implemented LIFO queuing as well as a novel hybrid approach in MQTT as a proof-of-concept and evaluated its impact.

I. Introduction

The progressive digitalization has changed modern agriculture in the last decades and an infrastructure of software systems has grown historically. In order to optimize processes in the value chain, these systems interconnect a plurality of various devices ranging from constrained sensors deployed in agricultural fields, stables, or silos to smart farm equipment, agricultural machinery, and services. However, existing and proprietary systems only cover limited sections of the value chain. Their heterogeneity inherently hinders this optimization as well as the development of new Smart Farming applications and services and a seamless integration of farm management information systems. For collaborative applications and optimizations, there is a demand for a decentralized (inter-)network for agriculture consisting of (centralized) self-sufficient networks of actors such as farmers or contractors. While the global communication of the decentralized network is based on a broadband Internet connection, predominantly Public Land Mobile Network (PLMN) communication is used within individual self-sufficient networks by devices located in the field. Eventually, a robust interconnection of these devices has become crucial for a coordinated management of agricultural processes. Thus, in this paper, we focus on a performance evaluation of transmission reliability and timeliness in suchlike networks. In the context of Smart Faming and this interconnection of various clients, the adaption of the popular Internet of Things (IoT) protocol, Message Queue Telemetry Transport (MQTT) [3], is very promising. Recently, it is also applied in other domains, particularly in the area of Smart Faming. However, if communication clients are mobile, e.g., machinery or livestock in fields, PLMN communication becomes inherently unreliable in rural areas [11] and impaired by transient network disruptions. Besides reliability aspects, in this case, different queuing strategies for data buffers are of special interest depending on the specific demands of applications.

The core contributions of this paper are (1) an empirical real-world evaluation of MQTT communication with mobile clients via PLMNs, (2) the introduction of scenario-specific challenges, (3) practical solutions for a robust communication, and (4) an advanced application-specific queuing scheme.

The remainder of this paper is organized as follows. Section II provides background about MQTT, the agricultural scenario considered in this work, and highlights specific challenges that arise. Section III then describes our evaluation architecture which was used to conduct real-world experiments. The presentation of the obtained results is given in two separate sections. First, evaluation results regarding the reliability of MQTT's message delivery are shown and discussed in Section IV. Afterwards, Section V introduces results about the timeliness of messages that are buffered due to network disruptions (either by MQTT publishers or brokers). Moreover, we introduce a novel queuing strategy tailored to specific demands of certain applications. Finally, we discuss our experiences and contributions of this case study in Section VI and conclude the paper in Section VII with a short outlook on our next steps.

II. Background

A. Message Queue Telemetry Transport

MQTT is a messaging protocol on the application layer based on top of TCP. It is standardized by OASIS [3] and is widely used in industry for Machine-to-Machine (M2M) and IoT communication. Its publish-subscribe pattern is a loose-coupled communication approach that uses hierarchical structured topics for individual message streams. Clients can act as a publisher or a subscriber and exchange messages according to specific topics. For that purpose, a publisher sends messages on a certain topic to a central broker that forwards these

 DOI: 10.1109/LCN.2017.81

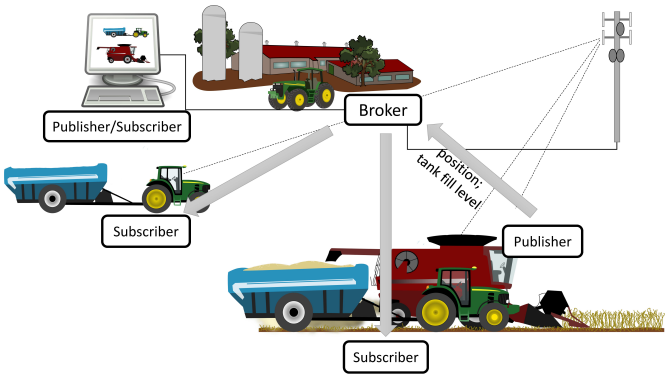


Fig. 1. Motivational scenario: collaborating machines during a harvesting process, interconnected by MQTT via PLMN. Reliable and timely communication contributes to the challenging infield-planning of unloading operations.

messages to each client that has subscribed this topic. Hence, it is suitable to exchange periodically gathered sensor data from agricultural machinery or agricultural Wireless Sensor Networks [4], [5], [14].

In MQTT, prior to every messaging, each client needs to initiate a connection establishment to the broker. This is achieved by a specific *CONNECT-CONNACK* handshake between both entities. In the *CONNECT* message, all relevant client information is present, e.g., the client ID, user credentials, and an *Keep Alive (KA)* parameter. This *KA* parameter is specified by the client and represents the longest possible time interval that both entities can endure without communication. It allows to determine if the communication partner is still reachable. As long as messages are exchanged frequently enough, no extra messages are required. Otherwise, special *PING* messages are used. Relevant for this work are particularly two features of MQTT. First, it supports three Quality of Service (QoS) types for different reliability levels and, second, there is a special *persistence mode* that maintains session state information for client reconnects. Amongst others, there are additional basic security features such as SSL/TLS or access control lists to manage the access to certain topics.

B. Mobile Scenario

In our study, we exemplarily consider the following simplified reference scenario that demands a reliable and timely MQTT communication: During the grain harvest, infield logistics of crop transportations are very challenging. Combine harvesters store crop in their internal tanks. If such a tank reaches its maximum fill level, the crop has to be unloaded to street transporters on site. To save expensive process time, the unloading has to be in time and takes place while driving, cf. [11], [12]. For the infield-planning that optimizes the orchestration of suchlike unloading processes, situational awareness is beneficial and can be realized by timely (MQTT) communication. Hence, the combined harvester has to publish its position and fill level periodically. These publications are then transmitted to a central broker, consumed by operators, and forwarded to transport vehicles as subscribers, as sketched in Figure 1.

In preliminary investigations, we observed a challenging connectivity of agricultural machines in rural areas. This challenge is also addressed by other research in literature and faced by complementary approaches, e.g., a delay-tolerant combination of wireless ad-hoc networks and PLMNs [11]. Hence, our main assumption in this scenario is that communication entities (i.e. MQTT publishers and subscribers) are mobile and that significant variations in PLMN connectivity with temporary disruptions occur. On the other hand, this work is not limited to this Smart Farming use-case but also relevant for similar scenarios that demand situational awareness.

C. Challenges

As MQTT has been designed for industrial environments, PLMN communication and mobility were originally not considered. Moreover, it uses TCP that provides a stream-based, reliable, and ordered data delivery as long as connections could be maintained. Nevertheless, MQTT offers its own reliability by adding an additional retransmission scheme and, thus, additional overhead. For that purpose, different QoS types are available. While messages are transmitted in a fire-and-forget fashion (i.e. best efforts of the underlying TCP/IP network) in QoS type 0, QoS 1 and 2 "guarantee" a successful delivery. In case of QoS 1, a message is delivered at least once and QoS 2 additionally guarantee a duplicate-free transmission. However, the characteristics of PLMNs in rural environments are very challenging in general. Furthermore, TCP that is known to result in a severe degradation of end-to-end performance in wireless networks [2]. The reason is TCP's lack of the ability to distinguish the packet losses caused by network congestions from wireless transmission losses. Although many mechanisms for improving TCP's performance in wireless and wired-wireless hybrid networks have been proposed in the last decades, existing TCP variants are still far from optimal for modern high-speed PLMNs [10]. Thus, questions concerning the choice of MQTT based on TCP and the performance of this combination in practice arose: Can MQTT cope with mobility, fluctuating network properties (GSM, UMTS, LTE), and transient disruptions? Does it provide sufficient robustness? Although MQTT is able to buffer messages in QoS ≥ 1 mode, it only provides First In – First Out (FIFO) queuing and also lacks in prioritization. Does FIFO buffering affect the timeliness in our scenario where latest information holds higher relevance? Is it easily possible to change this strategy?

III. Evaluation Architecture

A. Basic Setup

In order to answer the aforementioned questions, we began with basic field experiments and successively extended these performance evaluations. In all setups, we use Eclipse Mosquitto v3.1[1], an open-source MQTT broker that was installed on a stationary server in the Internet. As mobile clients, we use two devices: One device as *mobile publisher*

[1] https://mosquitto.org

TABLE I
OVERVIEW OF PLMN INFRASTRUCTURE-BASED FIELD EVALUATIONS OF MQTT WITH EVOLVING CONFIGURATIONS.

Nº	Distance (km)	Duration (min)	Mobility	#Messages	Keep Alive (s)	Pub Client	QoS	Queuing	Load (msg/s)	Sub Client	QoS	Persistence	PER* (%)	#CON/#RST	PER (%)	DPR (%)	
Evaluation					**Configuration**								**Results**				
					MQTT	**Publisher**				**Subscriber**			**TCP**		**MQTT**		**Fig.**
1(a)	150	123	car	14678	60	pub	**1**	FIFO	2	host	0	–	1.532	2 / 1	**0.129**	5.968	2(a)
1(b)	150	136	car	16205	60	pub	1	FIFO	2	**sub**	0	–	n/a		**6.819**	4.887	2(b)
2(a)	30	47	car	5693	**3600**	pub	1	FIFO	2	host	0	–	1.316	**1 / 0**	**0**	**34.709**	
2(b)	30	47	car	5693	3600	pub	1	FIFO	2	sub	0	–	1.629	1 / 0	6.974	23.274	3
3(a)	65	74	**train**	4449	3600	pub	1	FIFO	1	sub	0	–	5.293	2 / 0	**65.638**	0.180	
3(b)	65	74	train	4449	3600	pub	1	FIFO	1	sub	1	**FIFO**	5.293	2 / 0	**0**	2.180	4(a)
3(c)	65	135	train	8044	3600	pub	1	FIFO	1	sub	0	–	5.688	1 / 0	39.667	0.025	
3(d)	65	135	train	8044	3600	pub	1	FIFO	1	sub	1	FIFO	5.688	1 / 0	0	1.156	
4(a)	400	190	train	11411	3600	host	1	FIFO	1	sub	0	–	8.254	5 / 4	62.475	0	
4(b)	400	190	train	11411	3600	host	1	FIFO	1	sub	1	**LIFO**	8.254	5 / 4	0	7.020	
4(c)	400	179	train	10734	3600	host	1	FIFO	1	sub	0	–	4.667	4 / 3	69.955	0	
4(d)	400	179	train	10734	3600	host	1	FIFO	1	sub	1	LIFO	4.667	5 / 4	0	1.732	4(b)
5(a)	25	21	car	1251	3600	pub	1	FIFO	1	host	0	–	6.730	4 / 1	0	619.50	5(a)
5(b)	25	21	car	1251	3600	pub	1	**LIFO**	1	host	0	–	6.730	4 / 1	0	616.31	5(b)
5(c)	25	21	car	1251	3600	pub	1	**AFLO**	1	host	0	–	6.730	4 / 1	0	71.94	5(c)

client and the other one as *mobile subscriber*. For both devices, we use a Raspberry Pi 3, a small and low-priced single-board computer, and equipped each device with an LTE modem (Huawei E3372) and a GPS receiver. On both devices, either MQTT publishers or subscribers are realized by Paho-MQTT v1.2.2[2] as Python clients that operate on Raspian Jessie. This Raspberry Pi Linux distribution (Linux Kernel 4.4) includes the most widely deployed TCP version CUBIC [7] that provides an enhanced congestion control algorithm which is particularly scalable in networks with a large bandwidth-delay product. In order to evaluate different configurations of Paho clients while being able to directly compare these configurations, we occasionally run multiple instances of publishers or subscribers on the same physical device. Furthermore, there is an additional Python Paho-MQTT publisher and accordingly a subscriber. Both are executed on the server that hosts the broker. Thus, these stationary clients are hereinafter referred to as *host publisher* and *host subscriber*. Since they share the same server with the broker, they are neither affected by packet loss or transmission delays and, therefore, helpful for a separated evaluation of both "mobile" links (publisher to broker and broker to subscriber).

Each publisher periodically generates and publishes messages in a fixed interval. These messages are efficiently serialized with Google protocol buffers (protobuf)[3] and contain a time stamp, a sequence number, and longitude/latitude values resulting in 68 byte messages on application layer. MQTT does not forward any client identity or another kind of originator address to subscribers. Instead, it is implicitly assumed that this information is included in the predefined topic or maybe explicitly integrated within the payload. For that reason, we decided to use the topic-based differentiation. Hence, we use different topics for each publisher, i.e. `<sensors/mobile/n>` and `<sensors/host/n>` for the n-th mobile and stationary publisher, respectively. Each subscriber then simply subscribes to all relevant topics by using the MQTT multi level wildcard, i.e. `<sensors/#>`.

Despite of its additional overhead, we decided to use QoS 1 (unless otherwise stated) since we observed some packet loss with QoS 0 during phases in long dead zones. Moreover, on the mobile clients, but also on the stationary server, we log every MQTT event as well as every in- and out-going message on application layer and use tcpdump [4] to capture pcap-files for a deeper analysis. Besides, all PLMN status information which is available by the API of the LTE modem of mobile clients is periodically logged every second to provide insights into network connectivity in our experiments. The mobility of clients is "emulated" by intentional car and passenger train trips in rural areas in Lower Saxony and North Rhine-Westphalia, in the north-west of Germany. In fact, both mobile clients (publisher and subscriber) were carried along in the same vehicle during these trips. Although separated LTE modems were used, this leads to correlated PLMN effects that can impair the significance of the evaluation. To cope with this spatio-temporal correlation, both links, mobile publisher to broker and broker to mobile subscriber, can be analyzed separately by leveraging the host MQTT client entities.

B. Evaluation Overview

Prior to a detailed introduction of our evaluation results in the following two sections, Table I provides an overview

[2]https://pypi.python.org/pypi/paho-mqtt/1.2.2

[3]https://developers.google.com/protocol-buffers/

[4]http://www.tcpdump.org

of a representative selection of the large number of field experiments in advance. In this table, we summarize basic information of these experiments such as the distance and duration of a route, the means of transportation, and the accumulated number of all messages in the corresponding experiment. For later referencing, the experiments are numbered. Moreover, the configurations of individual experiments regarding publishers and subscribers are listed, representing the successive evolution of our setups. The evolving progress of variations in these configurations are emphasized in bold type and will explicitly be described in the following sections. Although not explicitly listed in Table I, the *KA* parameter of the subscriber are always identical to the corresponding parameters of publisher. Finally, evaluation results of TCP and MQTT are given concerning the following metrics, introduced in the next subsection.

C. Evaluation Metrics

We use the packet error ratio (PER) as performance metric that represents the reliability of data transmission. On the transport layer, in the case of TCP, this ratio does not indicate the real packet loss since transmission errors of underlying layers are usually compensated by TCP's retransmissions. The default number of retransmission retries is 15 and, since a random exponential backoff algorithm is implemented, this leads to a retransmission phase of approximately between 13 to 30 minutes [9].

During that phase, we, thus, leverage the number retransmissions for TCP's PER. That means, it is measured at the sender and actually represents the retransmission ratio. Therefore, it is abbreviated with PER*. However, if the network disruption exceeds the retransmission phase, packet loss is not considered by PER*. Hence, PER* represents a lower bound for the real loss ratio only. On application layer, in the case of MQTT, the PER indicates the actual packet loss, more precisely: MQTT message loss. Moreover, the duplicate packet ratio (DPR) describes the ratio of duplicated packets to the overall number of packets on application layer. Here, duplicated packets are MQTT messages which have been received repeatedly by a subscriber. As an additional indicator of network quality and disruptions, we listed TCP session information for each experiment represented by the number of TCP connections (# CONs) and session resets (# RSTs).

In order to evaluate the timeliness of the MQTT delivery in our scenario, we adopt a so-called *utility metric* from the context of situational awareness [6]. This metric particularly allows to evaluate the benefit of different queuing strategies from an application perspective. For every message msg in a set of messages M, it defines the value $um(msg)$ based on its delivery delay $d(msg)$ and two predefined thresholds δ_{min} and δ_{max}. If the delay exceeds the lower threshold but is less than the upper one, the value decays exponentially with the delay of the message (depending on the exponential decay constant λ, here: $\lambda = 6$). That can be formalized as:

$\forall\ msg_i \in M$:

$$um(msg_i) = \begin{cases} 100\,\%, & \text{if } d(msg_i) \leq \delta_{min} \\ 0\,\%, & \text{if } d(msg_i) \geq \delta_{max} \\ 100 \cdot e^{(-\lambda \frac{d(msg_i)-\delta_{min}}{\delta_{max}-\delta_{min}})}\,\%, & \text{otherwise.} \end{cases}$$

The utility metric UM is then determined by the average of all values of a given $M \neq \varnothing$ as follows

$$UM = \frac{1}{|M|} \cdot \sum_{i=1}^{|M|} um(msg_i). \tag{1}$$

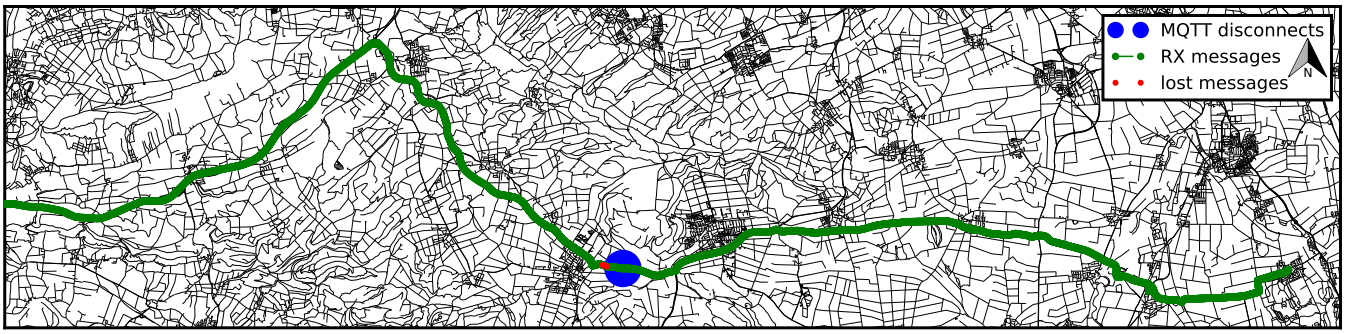
IV. TCP and MQTT in the Wild

In the first experiments, we investigate the interaction of TCP and MQTT. Using the aforementioned mobile publisher that transmits its GPS coordinates (2 msg/s) to a stationary subscriber, we primarily analyze the reliability of the MQTT messaging in the wild by measuring the PER.

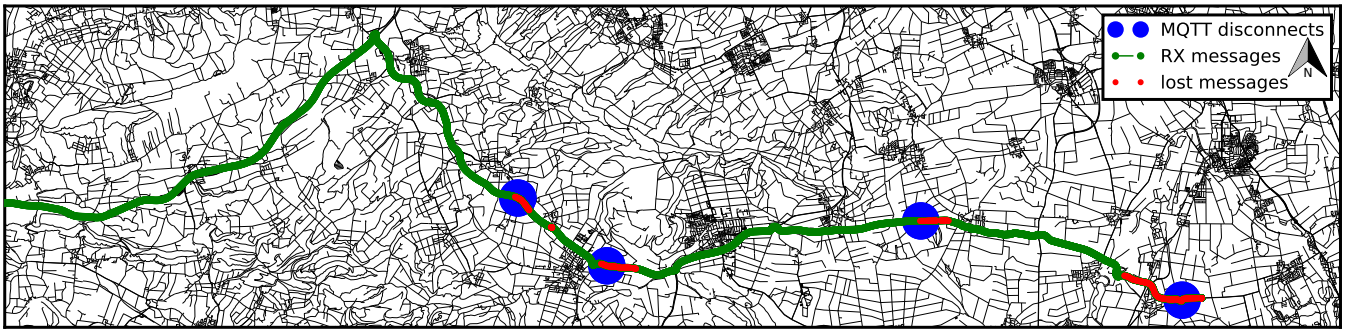
A. Mobile Publishing

Due to the underlying reliability mechanism of TCP, packet loss is very unlikely. Since PLMN capacity is far from being exceeded in our scenario and reliability is utmost importance for our application, we, nevertheless, set MQTT's QoS level to QoS 1. However, losses occasionally still occur, as demonstrated by the first evaluation in Figure 2(a). As shown in Table I (Eval. 1(a)), the PER observed is marginal (0.129 %) but existing. It turns out, that the reason for this existence is Paho's default *KA* interval (60 s) of MQTT. While TCP is still keeping the session alive, this interval is responsible for a premature disconnection (indicated by a TCP RST) triggered by MQTT, followed by discarding messages in spite of QoS 1. Thus, we observed two TCP and MQTT connection establishments in this experiment (cf. Table I, Eval. 1(a)).

Once, the *KA* parameter is large enough, e.g. 3,600 s, or even exceeds the corresponding TCP timer (default: 7,200 s [9]), the MQTT connection of the mobile publisher is stable and the QoS 1 delivery to the broker could be classified as reliable. We therefore use a 3,600 s *KA* value for mobile publishers from now on, as can be seen in Table I. The table confirms that whenever this is the case and messages are forwarded to a host subscriber (Eval. 2(a) and 5), MQTT packet loss can totally be prevented. The downside of the *KA* increase is that it is likely that the DPR increases significantly as well. The reason is that with an increased *KA* value, the MQTT connection bridges long disruption phases as intended, but still keeps resending (duplicated) messages for messages that cannot be confirmed by acknowledgments yet. On the other hand, the automatic repeat request (ARQ) mechanism of TCP simultaneously attempts to retransmit the original message and also each related duplicate. Hence, an increased number of duplicates occurs in particular in case of frequent short disruptions (i.e. in the lower minute range).



(a) Mobile publisher to broker (host subscriber) in Evaluation 1(a) (direction of travel: East).



(b) Mobile publisher to mobile subscriber in Evaluation 1(b) (direction of travel: West).

Fig. 2. Mapped representation of a 80 km section of the first evaluation.

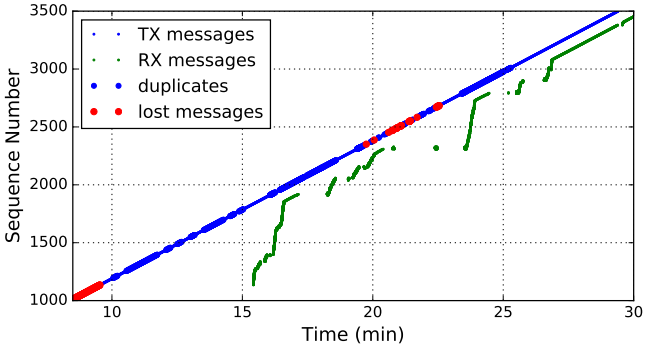


Fig. 3. Message delivery between mobile clients. Despite of an increased *KA* value, the delivery to a *mobile subscriber* still shows a serious loss ratio. In this visualized section of Eval. 2(b), the PER = 7.20 % and DPR = 50.10 %. For presentation purpose, a 1 min bias is used at the subscriber.

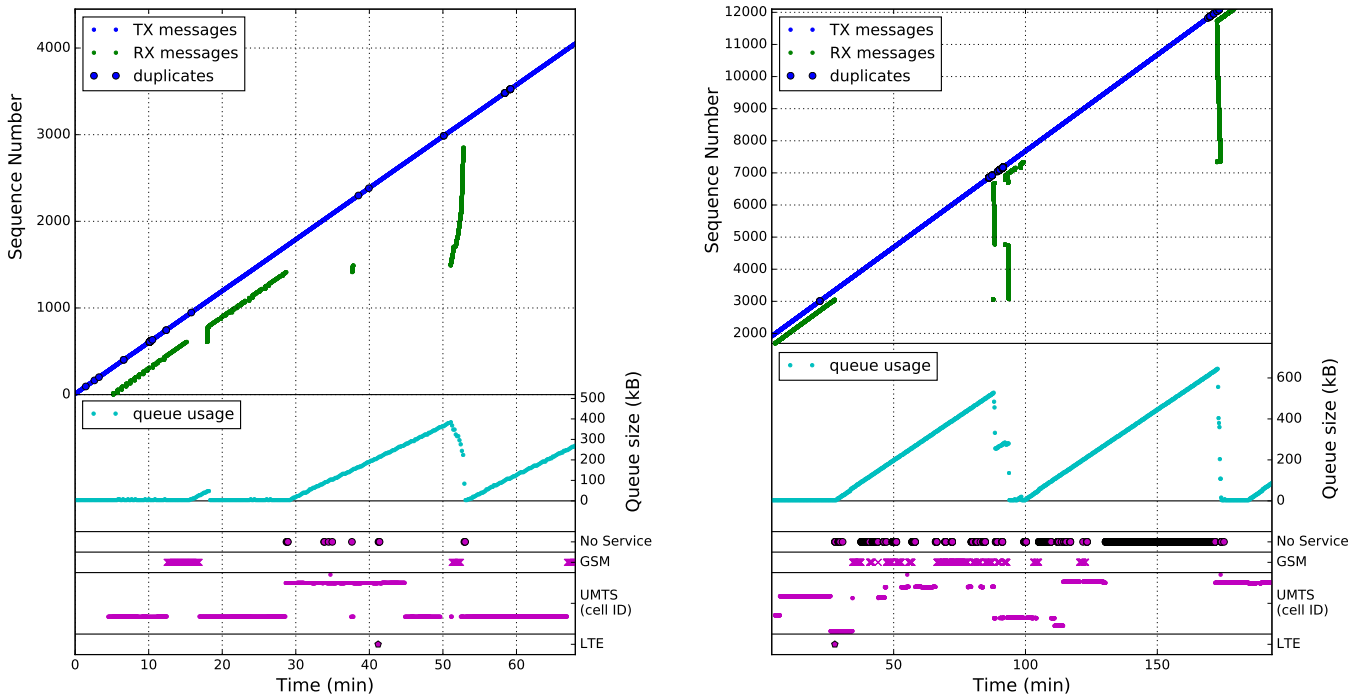
B. Mobile Subscribers and MQTT Persistence

A mobile subscriber is exposed to similar challenges as a mobile publisher. However, as a receiver, suchlike subscribers, cannot actively trigger retransmissions. Generally, they are not even aware of transmission attempts. Thus, a higher PER is expected at the link between the broker and a mobile subscriber in our scenario.

Figure 2 confirms this expectation for the first Evaluation. The increased PER experienced at the mobile subscriber (6.819 % in total, cf. Tab. I) is visible in Figure 2(b), indicated by red sections of the GPS trace. Even though in Evaluation 1, a *KA* value of 60 s is used, the map presentation also reveals that not the whole packet loss occurs in spatial proximity of MQTT disconnects (visualized by blue circles) as there is at least one exception in Figure 2(b). This observation already suggests that solely an increase of the *KA* value does not provide a sufficient solution towards a reliable messaging. Indeed, if the corresponding value is also set to 3.600 s, as done in Evaluations ≥ 2, occurrence of packet loss is not necessarily disappearing (cf. e.g., Eval. 2(b) in Tab. I). This is also shown by the time sequence diagram in Figure 3 which visualizes the delivery from the mobile publisher to the mobile subscriber in the second evaluation on a temporal scale. Besides packet losses, messaging delay is indirectly illustrated by the horizontal gap between transmission and receive events. This delay is noticeable by the application and, here, primarily caused by the publisher due to the spatio-temporal correlation of publisher and subscriber in our evaluation setup.

To cope with transient connection disruptions and, thus, with the existing packet loss on (mobile) subscriber side, Mosquitto provides a *persistence mode* that disables so-called MQTT *clean sessions*. This mode is designed to allow for automatic reconnects for particularly constrained clients with limited resources. If this mode is activated, the broker supervises the transmission to each QoS ≥ 1 subscriber. If necessary, i.e. if messages could not be acknowledged by the subscriber, these messages are then buffered in an internal database until they can be successfully redelivered.

In our third evaluation, we use the following configuration parameters for the persistence mode in the Mosquitto broker. Taking potentially long disruptions into account, we use a large



(a) Mobile publisher to a *persistent* mobile subscriber in FIFO mode. In this visualized section of Eval. 3(b), DPR is 1.16 % (3 min bias for subscriber and queue size).

(b) Broker (host publisher) to a *persistent* mobile subscriber in LIFO mode. In this visualized section of Eval. 4(d), DPR = 1.73 % and 79.06,% of all messages were delivered out-of-order as intended by the inverted strategy (5 min bias for subscriber and queue size).

Fig. 4. Reliable messaging using a persistence-capable broker (upper part). Additionally, the utilization of the broker's queue (central part) and fluctuations concerning the PLMN connection (lower part) are shown.

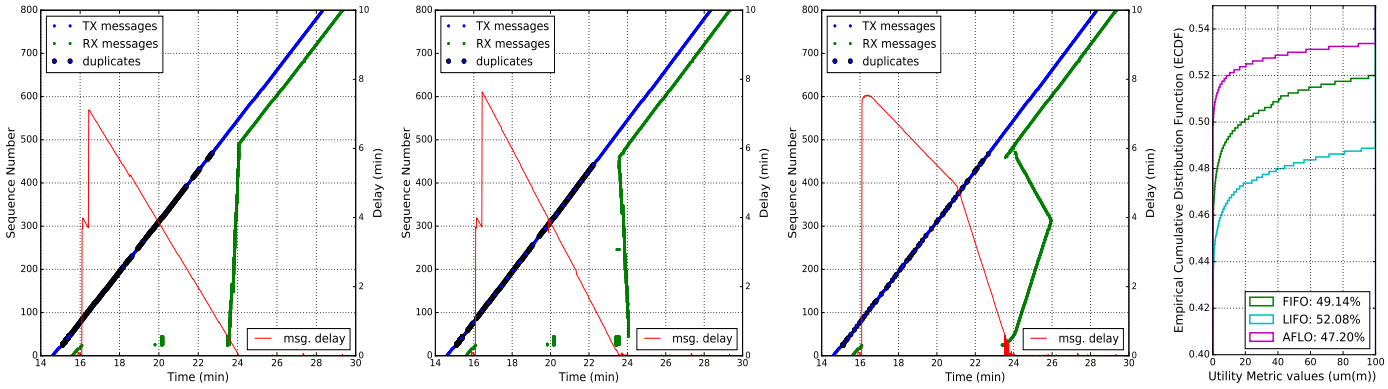
expiration period for buffered messages (1 day) and increase the queue size from 100 to 10,000 messages. Mosquitto's retransmission interval (default: 20 s) as well as the maximum number of inflight messages (default: 20) are unchanged. In Table I, the activation of the persistence mode in the broker and simultaneously in the (persistent) mobile subscriber is indicated by entries in the *Persistence* column. As highlighted, persistence is firstly used in Evaluation 3. At the same time, this evaluation is the first one conducted by train mobility which generally results in poorer PLMN connectivities in our experiments and, thus, in a significantly increased PER for non-persistent subscribers (cf. Eval. 3(a)/(c) and 4(a)/(b)). Nonetheless, MQTT's persistence feature successfully enables a reliable messaging for mobile subscribers over suchlike lossy and disruptive connections. This reliability and its impact on delivery delays are exemplarily illustrated by Figure 4(a). Compared to the previous scenario (Fig. 3), a larger delay (up to 25 min) due to longer network disruptions is observed. However, all messages are queued and retransmitted by the broker as soon as possible. Hence, large retransmission bursts occur, as can be seen in the figure. Moreover, the queue utilization is visualized as well and is obviously increasing during disruption phases. This increase is linear to the number of messages in the buffer, but not necessarily linear to the number of subscribers since messages are not redundantly stored in the Mosquitto broker. Additionally, results of Evaluations 3, 4, and 6 in Table I confirm that reliable messaging is feasible.

V. Application-specific queuing

Apparently, temporary PLMN disruptions caused by mobility in rural areas are frequent and can be quite long (cf., Fig. 4(a)). Hence, a large number of messages is required to be buffered in both, publishers' queues and brokers' databases, even with low data rates as in our experiments. At the same time, these messages need to be retransmitted as soon as possible, resulting in considerable retransmission burst phases that induce additional delay to messages at the tail of the queue. Thus, the queuing strategy used in both entities has significant impact. If the retransmission phase is, moreover, interrupted by further disruptions, the choice of an appropriate strategy gains in even more importance. In our scenario, timeliness of delivery is essential. We assume that the latest information (e.g., position of a vehicle) holds higher relevance for decision making processes. For that reason, we modified MQTT's FIFO queuing implementations.

A. Broker queuing

The Mosquitto broker stores messages in a linked list. For each *persistent* subscriber, an additional list of missed message IDs is maintained. In order to realize another queuing strategy, either the en- or dequeuing process needs to be adapted. As a proof-of-concept, we implement a LIFO queuing by simply inverting the enqueuing accordingly, provide a flag to toggle between it and the original FIFO mode, and evaluated the impact, as exemplarily visualized in Figure 4(b). One can see



(a) FIFO publisher (Eval. 5a). (b) LIFO publisher (Eval. 5b). (c) AFLO publisher (Eval. 5c). (d) Utility metric.

Fig. 5. Queuing strategies impact on the delivery delay. Different strategies are prototypically implemented in the publisher client that periodically transmits messages to a stationary subscriber in Eval. 5. The message delay is integrated in the diagrams (1 min subscriber and delay bias). Furthermore, (d) compares the ECDFs of the utility metric achieved by each strategy.

that, compared to the retransmission burst in Figure 4(a), the orientation of these bursts is inverted by the LIFO strategy and the number of out-of-order retransmissions grows accordingly. That means, in such a burst, firstly latest (i.e. "last-in") messages are retransmitted. This can be very beneficial for certain applications. In this evaluation, a retransmission burst (in the center of Fig. 4(b)) is interrupted, hence, as mentioned above, further increasing the delivery delay of corresponding messages and thereby the potential of LIFO queuing. Note that the first messages in burst phases that still arrive in the original order represent pending inflight messages (Mosquitto's default: 20 messages) which are handled by the retransmission mechanism of TCP.

B. Publisher queuing

Similar to the broker, QoS ≥ 1 publishers monitor the states of their publications and buffer all messages until a corresponding acknowledgment is received from the broker. From application perspective, it is also worth to consider different queuing strategies for this buffer. In the Paho-MQTT client that we use in our architecture, the FIFO queue can be easily adapted by inverting the dequeuing process.

For a quantitative comparison of the impact of FIFO and LIFO on the timeliness, i.e. the message delivery delay, two publisher clients were executed on the same (mobile) device and share the same LTE modem: one client in FIFO mode and another in LIFO mode. Again, these clients periodically generate (GPS) sensor data during a 25 km route and publish these data to the broker which is then forwarded to a host subscriber. For the evaluation, we apply the *utility metric* introduced in Section III-C with the *situational awareness* parameterization from [6], i.e. $\delta_{min} = 50\,\mathrm{s}$ and $\delta_{max} = 100\,\mathrm{s}$. Thus, a maximum delay of 50 s is considered as negligible, messages with delays lower then 100 s remain relevant, while message with higher delays are useless for the application.

The resulting utility metric UM shows that situational awareness can profit from LIFO queuing. As for the UM determination (Eq. 1 in Sec. III-C), the metric values of all messages are accumulated, we chose a empirical CDF plot to visualize the UM results of both queuing strategies. For the time period considered here, the UM results are shown in Figure 5(d). The regular FIFO-based retransmission results in a UM of 49.14 %, whereas LIFO achieves a significant increase (52.08 %) due to its early retransmission of later messages. Thus, in Figure 5(d), the UM graph of LIFO clearly outperforms the corresponding FIFO graph. However, the extent of the difference depends on the chosen UM parameterization. Eventually, this is strongly application-specific, hence, wether using FIFO or LIFO is a decision that has to be made by the end-user.

C. Hybrid queuing

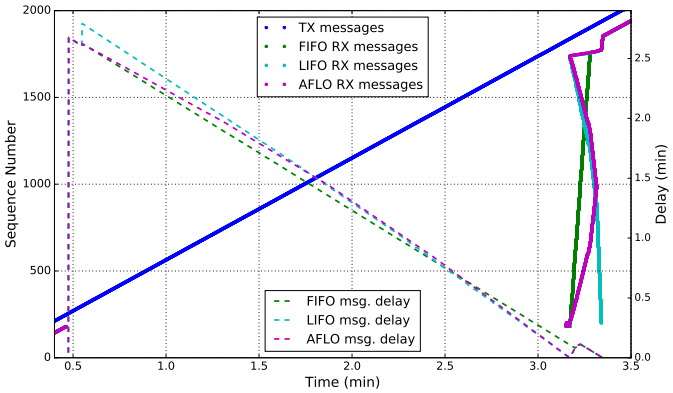
Both complementary strategies, FIFO and LIFO, have their reason for existence and, depending on the demands of the applications, one or another can be more suitable. This obvious consideration suggests our novel and hybrid strategy that simply combines FIFO and LIFO and, thus, their advantages. This is realized by a continuous alternation between both strategies. For that purpose, we maintain a FIFO buffer, i.e. messages are enqueued in FIFO order, and modify the dequeuing routine in that way that alternately the first and the last element of the buffer is taken for a retransmission attempt. As a result, alternately the oldest and the latest message is delivered during the retransmission burst. Hence, the hybrid approach, that is denoted as Alternating First-Last Order (AFLO) queuing scheme hereinafter, satisfies the demands of both types of applications. However, as a consequence of the combination, the temporal resolution of the delivery is, so to say, halved, since the oldest or the latest information is received with one half of the frequency as in the original FIFO or LIFO approach, respectively.

The impact of our novel queuing scheme depends on two factors: (1) it is simply determined by the amount of data that is queued and the duration of the dequeuing process, i.e. the load, the duration of the disruption, and the available data rate, and (2) the specific parameterization of the utility

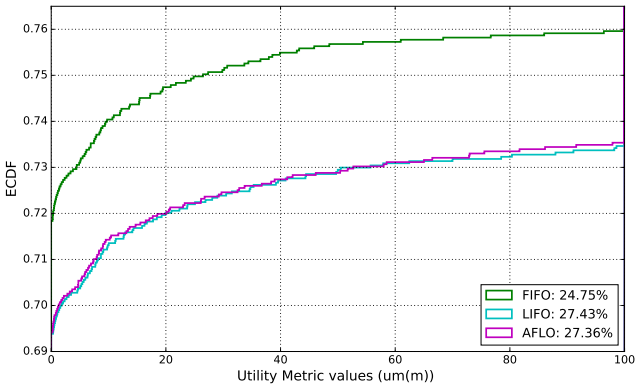
metric UM (cf. Eq. 1). Thus, it is difficult to generalize a quantitative impact. Therefore, we decided to exemplarily demonstrate this impact by an experiment. We therefore implemented the AFLO scheme as an optional scheme for both, publisher clients and brokers, and evaluated the impact of this scheme in Evaluation 5 (cf. Tab. I). The resulting delay leads to a characteristic shape of the AFLO retransmission burst in time sequence diagrams that is representatively shown by Figure 5(c).

Figure 5 exemplarily shows the impact regarding all three strategies. We have restricted the AFLO publisher to a stop-and-wait mode by changing its maximum number of in-flight messages to a single message in order to emphasize its characteristic dequeuing effect in this real-world experiments. At the same time, the default maximum inflight parameter (20 messages) was used for the two other strategies (cf. Fig. 5(a) and 5(b)). Thus, here, the duration of the AFLO retransmission burst is significantly longer and not directly comparable. As a result, its metric values are generally decreased (cf. Fig. 5(d)).

For a sound comparative evaluation, a controlled emulation environment is eventually used. Therefore, both client devices, publisher and subscriber, were connected to the broker via LAN. Three publishers were executed simultaneously on the publisher client with Paho-MQTT's default configuration, except for the queuing scheme. Again, we use one client as FIFO, another as LIFO, and the third as AFLO publisher. We speeded up our setup by increasing the sending frequency to 10 messages/s and adapting the metric parameters accordingly to $\delta_{min} = 5\,\mathrm{s}$ and $\delta_{max} = 10\,\mathrm{s}$. After a view seconds of regular operation, we then physically cut the connection between publisher and broker for roughly 3 min, forcing the publishers to buffer their messages. Once the connection is reestablished, each publisher starts its retransmission phase, as visualized in Figure 6(a), until the regular operation is reached again. Quantitative results of all three schemes are provided by the utility metric and given in Figure 6(b). As observed in the real-world experiment, the ECDF plot shows that LIFO outperforms FIFO regarding this metric. Moreover, the emulative plot reveals that our novel AFLO scheme achieves similar results with LIFO. However, in contrast to LIFO, AFLO not only serves applications that are primarily interested in the latest information, but likewise applications with opposite interests that would prefer the FIFO strategy.



(a) Messaging and its delay from a publisher to a persistent subscriber in the emulation (10 s subscriber and delay bias).



(b) ECDFs of the utility metric achieved by each strategy.

Fig. 6. Impact of different queuing strategies on the delivery delay.

VI. Discussion & Related Work

In our real-world experiments, mobility is emulated by car or train trips. Routes are well chosen to be representative for our motivation and we believe that PLMN conditions are comparable to a certain extend and our main assumption of mobile entities suffering from limited connectivity (cf. Sec. II-B) is satisfied. A comparatively lower speed of agricultural machinery will basically increase the duration of transient disruptions and, thus, increase the impact of our work.

The communication performance in PLMNs has also been empirically studied in [8] and [15] that also come to the conclusion that mobility has a significant impact. In particular in trains, the perceived network quality can be very limited. However, both studies focus on bandwidth predictability and rather consider bandwidth required for human demands than for M2M communication via publish-subscribe protocols.

Another issue which is worth to discuss is the redundant reliability of MQTT. On the one hand, MQTT relies on TCP, and on the other hand, it similarly features its own session and retransmission mechanisms. That has the advantage of allowing for queuing scheme customization on application layer, i.e. in the user space, as done in this paper. The OASIS standard explicitly states that unacknowledged packets are *only* re-sent in the persistence mode and *only* when a client with QoS ≥ 1 reconnects [3, Sec. 4.4]. While the standard consequently does not specify any retransmission timer, such a timer appears in common MQTT implementations. As a result, these implementations provide a complete ARQ mechanism. From that perspective, the underlying TCP can somehow be considered as obsolete. Thus, UDP will probably be a more effective choice, as also recognized by MQTT-SN [13]. However, this redundancy surely does not harm the reliability as long as there is network capacity available, but it implies overhead and, thus, wastes exactly this precious capacity. Indeed, this conflict is intensified by our recommendation to increase MQTT's KA value. As mitigation, i.e. to avoid

unnecessary MQTT duplicates, the corresponding retry interval (Paho's default: 5 s; Mosquitto's default: 20 s) should be considerably increased since TCP's retransmission mechanism operates at least roughly 13 min (cf. Sec. III-C). Since the standard does not specify such a retry interval, as mentioned above, it also suggests a KA interval of "typically a few minutes" [3, Sec. 3.1.2.10].

Application-specific queuing is not new and has been considered on transport layer in various middlewares in many scenarios. For instance, a protocol for sensor data transmission in the context of situational awareness is proposed in [1] and evaluated in disaster area maneuvers. This protocol also implements LIFO queuing and highlights its necessity. However, to the best of our knowledge, there exists no related work that considers application-specific queues in the context of MQTT. Thus, currently, it is not intended in the MQTT specification which consequently specifies that whenever a publisher re-sends any message, it must re-send them in the order of the original messages and, moreover, that a broker must treat each topic as an ordered topic [3, Sec 4.6]. Thus, a modification of the regular FIFO queuing scheme violates this specification. However, our case study empirically shows that such a modification would be beneficial for certain applications in lossy and disruptive networks. As a consequence, the application that uses our customized MQTT must be capable to handle disordered messages and to take advantage of it.

In static scenarios or in the case that there are no variations and transient disruptions in the PLMN connectivity, it is likely that no messages will be en- or dequeued at the sender's buffer. Hence, different strategies would not have any impact. However, even in the absence of connectivity variations, queuing potentially occurs due to network congestion and, thus, gains relevance. Because of the moderate data rates in our motivational scenario, this case is not addressed here.

A kind of application-specific queuing is also enabled by related IoT protocols, e.g. AMQP[5] or by DDS[6]. Both approaches are much more complex and imply certain communication overhead. Hence, for our scenario, we decided to rely on the "light-weight" MQTT.

VII. Conclusion & Future Work

In this paper, we presented a performance evaluation of MQTT used for sensor data exchange in mobile agricultural processes via PLMNs with regard to its reliability and timeliness of message delivery in disruptive environments. This evaluation is based on a comprehensive real-world case study in which mobility is emulated by purposive trips in rural areas. Our results showed that (1) an adequate parameterization of MQTT increases its robustness in rural environments and (2) depending on the application, a customized queuing is beneficial. Therefore, we introduced a novel alternating queuing scheme, evaluated its impact, and show that this strategy is simple, yet effective. Nevertheless, more sophisticated and also topic-based strategies will be regarded as our next step along with the interaction of publisher and broker queuing. In the future, we will accompany and interconnect agricultural machines during various collaborative processes for further case studies.

[5]http://www.amqp.org

[6]http://portals.omg.org/dds

Acknowledgments

This work was partially funded by the German Federal Ministry of Education and Research (BMBF) within the Program "Innovations for Tomorrow's Production, Services, and Work" (02K14A19K) and managed by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the contents of this publication.

References

[1] N. Aschenbruck and C. Fuchs, "STMP - Sensor data Transmission and Management Protocol," in *Proc. of the 36th IEEE Conference on Local Computer Networks (LCN)*, Bonn, Germany, 2011, pp. 475–483.

[2] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, Dec 1997.

[3] A. Banks and R. Gupta (Eds.), "MQTT Version 3.1.1 Plus Errata 01," OASIS Standard, December 2015. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/errata01/os/mqtt-v3.1.1-errata01-os-complete.html

[4] J. Bauer, B. Siegmann, T. Jarmer, and N. Aschenbruck, "On the Potential of Wireless Sensor Networks for the In-Situ Assessment of Crop Leaf Area Index," *Computers and Electronics in Agriculture*, vol. 128, pp. 149–159, 2016.

[5] R. Beckwith, D. Teibel, and P. Bowen, "Report from the Field: Results from an Agricultural Wireless Sensor Network," in *Proc. of the 29th IEEE Conference on Local Computer Networks (LCN)*, Tampa, FL, USA, 2004, pp. 471–478.

[6] V. Firoiu, G. Lauer, B. DeCleene, and S. Nanda, "Experiences with Network Coding within MANET Field Experiments," in *Proc. of the Military Commmunication Conference (MILCOM)*, San Jose, CA, USA, 2010, pp. 1363–1368.

[7] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-friendly High-speed TCP Variant," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[8] F. Kaup, F. Fischer, and D. Hausheer, "Measuring and Predicting Cellular Network Quality on Trains," in *Proc. of the Int. Conference on Networked Systems (NetSys)*, Göttingen, Germany, 2017, pp. 1–8.

[9] Linux User's Manual, *TCP(7). tcp – TCP protocol*, April 2002.

[10] K. Liu and J. Y. B. Lee, "On Improving TCP Performance over Mobile Data Networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2522–2536, Oct 2016.

[11] F. Nordemann, "A communication-optimizing middleware for efficient wireless communication in rural environments," in *Proc. of the 9th Middleware Doctoral Symposium of the 13th ACM/IFIP/USENIX Int. Middleware Conference (MIDDLEWARE)*, Montreal, Quebec, Canada, 2012, pp. 3:1–3:6.

[12] S. Scheuren, S. Stiene, R. Hartanto, J. Hertzberg, and M. Reinecke, "Spatio-Temporally Constrained Planning for Cooperative Vehicles in a Harvesting Scenario," *KI - Künstliche Intelligenz, German Journal on Artificial Intelligence*, vol. 27, no. 4, 2013.

[13] A. Stanford-Clark and H. L. Truong, "MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2," IBM, 2013. [Online]. Available: http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf

[14] T. Wark, P. Corke, P. Sikka, L. Klingbeil, Y. Guo, C. Crossman, P. Valencia, D. Swain, and G. Bishop-Hurley, "Transforming Agriculture through Pervasive Wireless Sensor Networks," *IEEE Pervasive Computing*, vol. 6, no. 2, pp. 50–57, 2007.

[15] J. Yao, S. S. Kanhere, and M. Hassan, "An Empirical Study of Bandwidth Predictability in Mobile Computing," in *Proc. of the ACM Workshop on Wireless Network Testbeds, Experimental Evaluation & CHaracterization (WiNTECH)*, San Francisco, CA, USA, 2008, pp. 11–18.