Analysis of the Message Queueing Telemetry Transport Protocol for Data Labelling: An Orthopedic Manufacturing Process Case Study

Mangolika Bhattacharya, Reenu Mohandas, Mihai Penica, Mark Southern, Karl Vancamp and Martin J. Hayes
University of Limerick, Limerick, Ireland

Keywords: Data Cleaning, Digital Manufacturing, Industry 4.0., Internet of Things (IoT), Message Queuing Telemetry Transport (MQTT) Protocol, Message-Oriented Middleware (MOM).

Abstract: The recent paradigm shift in the industrial production systems, known as Industry 4.0, changes the work culture in terms of human machine interaction. Human labours are assisted by smart devices and machines as in human-machine cooperation and human-machine collaboration. For enhancing this process, data processing and analyses are needed. Therefore, data collection has become one of the most essential functions of large organizations. In this work, a data engineering experiment for a grinding process within a commercial orthotics manufacturing company is presented. The data collection and labelling is assessed for time stamp latency using the Message Queuing Telemetry Transport (MQTT) protocol. This step is necessary to determine if alarm prediction or 'front running' is feasible. The paper analyses the procured dataset and discusses its merits as an alarm predictor, using sparsity indicators and concludes that a new investment in sensor infrastructure is necessary. This work highlights some of the limits of performance that exist for the use of MQTT with existing sensor infrastructure when retrofitting machine learning based alarm prediction in an industrial use case setting. A road-map for potential solution to this problem is provided which needs to be assessed by the company management before further progress can be made.

1 INTRODUCTION

Data cleaning and labelling has now become a major limiting factor in the application of digital technology within the manufacturing industries (Yin and Kaynak, 2015). By integrating smart devices, applying self-learning solutions and self-directional capabilities, development costs are reduced while flexibility, speed and quality of production is increased. With the arrival of 5G, the Industrial Internet of Things (IIoT) and intelligent *sensor-cloud* powered systems are evolving swiftly. A sensor-cloud system is a blend of wireless sensor networks with an integrated cloud computing capability that is combined with universal physical sensing ability, high-speed computation and huge storage (Wang et al., 2017). IoT gateways are used to communicate with sensors that are *agnostic* with respect to physical layer connectivity. Subsequent pre-processing and filtering are applied to the data being generated by the suite of sensors/devices to optimise transmission load on the network. The processed data is transmitted to the cloud servers with the use of standard protocols, via. the ***Message Queueing Telemetry Transport (MQTT)*** protocol. MQTT has become popular in Industry 4.0 applications due to its lightweight instruction set and low-power consumption properties. This makes MQTT ideal for battery powered sensors connected by wireless networks in a possibly *ad hoc* fashion. In this paper, data is collected using a dedicated Programmable Logic Controller (PLC) for a particular (orthotic grinding) commercial orthopedics manufacturing process. Although the MQTT protocol is well-suited for agile access to process variables, we found that a number of non-linear time varying artifacts are exhibited by the collected data. The dataset comprises a set of variables, associated values and cloud based timestamps that are validated in an irregular fashion by process operators. Moreover, the dataset consists of both input parameters to the machine and the alarms that have been generated in response to performance abnormalities that run outside established confidence limits. The objective of the work is to create an intelligent system that can use machine learning to capture and ultimately predict alerts so that corrective action can be taken before occurrence of catastrophic events

Bhattacharya, M., Mohandas, R., Penica, M., Southern, M., Vancamp, K. and Hayes, M.
Analysis of the Message Queueing Telemetry Transport Protocol for Data Labelling: An Orthopedic Manufacturing Process Case Study.
DOI: 10.5220/0010450402150222
In *Proceedings of the 6th International Conference on Internet of Things, Big Data and Security (IoTBDS 2021)*, pages 215-222
ISBN: 978-989-758-504-3

thereby improving product quality and minimising downtime. This paper focuses on the problems that are caused by the latency issues in large databases due to the application of MQTT and suggests potential solutions. This paper is organized as follows. Section 2 discusses related work and concepts in this area of research. Section 3 presents a case study on storage of data in the industry and the problems associated with it. Section 4 analyzes the MQTT experimental observations. Section 5 proposes a new system architecture for real-time data analysis that works well in simulation. Finally, Section 6 concludes the paper and suggests a roadmap for installation of the architecture on the actual commercial process.

2 PRELIMINARIES AND RELATED WORK

The work in this paper considers data engineering challenges for Cyber Physical System (CPS) applications in Industry 4.0. Such applications require optimized manufacturing processes for smart allocation of intelligent manufacturing subsystems for the purpose of production. The objective is to support customer optimized individual (so called 'Batch size 0') product manufacturing. This intelligent control of subsystem utilization enables cycle time, reliability, security, logistics and circular economy/sustainability considerations to be actively managed. Therefore, the first objective of the the work is to select an information transfer protocol that directly Industry 4.0 aims of *resource efficient* production through the use of state-of-the-art CPS infrastructure (Möller, 2016).

2.1 Industry 4.0

Industry 4.0 denotes the arrival of Internet of Things(IoT), smart devices, sensor networks, and the entire cyber-physical systems (CPS) which power the cloud based manufacturing systems (Vaidya et al., 2018). First industrial revolution was powered by the invention of water and steam powered engines and machines. Second industrial revolution or which can be termed as "Industry 2.0" was made possible due to the arrival of the mass production of goods using electrical engines and machines. Third industrial revolution or "Industry 3.0" marks the beginning of automation using PLCs and IT infrastructure. Industry 4.0 is an era of digital manufacturing. Digital manufacturing is a process of smart manufacturing, which fuses the virtual world with the real world through the cyber-physical infrastructure (Möller, 2016). This is aimed to improve the quality and quantity of goods being manufactured. Industry 4.0 dwells on the principles of *interoperability*, *virtualization*, *decentralization*, *real-time capability*, *service orientation* and *modularity* (Khan et al., 2017).

2.2 MOM Protocols for Industry 4.0

Message-oriented middleware (MOM) is software and/or hardware infrastructure that dynamically assists the transmission and receipt of messages in an IoT network. MQTT is a particular instance of the MOM model based protocol (Banks and Gupta, 2014). MQTT was developed by Andy Stanford-Clark of IBM in 1999 (Banks and Gupta, 2014) (Luzuriaga et al., 2015). Other commercial examples of the MOM protocol include the Advanced Message Queuing Protocol (AMQP), and the Data Distribution Service (DDS) protocols. AMQP, created by John O'Hara at JPMorgan Chase in London in 2003 (Kramer, 2009) (Luzuriaga et al., 2015), provides encryption for the purpose of security(Luzuriaga et al., 2015) (Novelli et al., 2018). This makes AMQP more secure but at the same time more resource heavy and not optimized in terms of power consumption. For this reason AMQP is not seen as being appropriate for industry 4.0 applications. DDS is an MOM based protocol that does not employ a brokerage system for event driven data flow. (Pardo-Castellote, 2003) (Yang et al., 2012). DDS systems generally consist of a "databus" which connects the publisher to the relevant subscriber. A vector of parameters are used to control the flow of information from the publisher to the subscriber thereby making it extremely suitable for secure real-time access. It is necessarily resource-heavy in its implementation and not suitable at present for deterministic low-power consuming systems. DDS was initially developed for critical military applications. DDS does not include a broker but a set of parameters which control the quality-of-service (QoS) through management of the information flow from the publishers to the subscribers. This fast and direct control of flow of information in DDS protocol makes it an excellent choice for real-time data analysis if it can be deployed in energy limited low power devices and sensor networks. The use of DDS is an on going research theme outside the scope of this paper and will be considered in future work.

2.3 MQTT Fundamentals

MQTT has 3 types of QoS (Silva et al., 2018). QoS 0 denotes that the messages are delivered at most once, and either the publisher or the subscriber stores the message. QoS 1 means the publisher stores the mes-

sage, sends it at least once and keeps sending the message until a confirmation is received from the subscriber. QoS 2 denotes that both the publisher *and* the subscriber store the message until both the parties receive confirmation that the message has been received by the subscriber at least once. MQTT does not have out-of-order delivery of messages, which creates latency issues that need to be mitigated for real time control applications. Separate factors such as network congestion and packet-losses need to be managed dynamically to minimise latency. This work considers a light-weight commercial implementation of MQTT that makes it suitable for low-power applications. To summarize, it can be said that AMQP is resource-heavy, power consuming but a secure implementation. DDS is suitable for real-time applications but not yet suitable for low-power and light-weight applications. MQTT is, at present, the only protocol that is capable of being deployed in an Industry 4.0 suitable for light-weight and low-power setting *only when* the latency issues are small enough to allow a floor level on information throughput. This paper considers how this floor level on performance can be dynamically assessed in a real time setting.

In this orthotics use case, the process relates to grinding of particular knee orthotics, to provide knee replacement solutions, with a comprehensive suite of orthopaedic knee implant products and instruments. It is an ultimate objective that last minute setpoint data can be transmitted to the process so that personalised medical applications can be supported.

MQTT is used to send data collected by a number of PLC supported sensors to the cloud (Silva et al., 2018). MQTT utilizes a publish-subscribe standard for the transmission of data. In (Silva et al., 2018) the authors dynamically investigated the latency in transmission of data using the MQTT protocol and reported significant real time control issues.

In order to address the latency issue that has been identified as the key limiting factor for performance, the authors of (Peralta et al., 2017) presented a 'fog' based computing scheme that also analysed the overall network energy consumption for a cluster of IIoT nodes and performed simulations on real data-sets using machine learning algorithms. They then compared the results with traditional MQTT schemes.

The authors of (Khan et al., 2017) considered the data engineering challenges presented by the use of MQTT in real-time digital manufacturing processes. This work actively considered the problem of data transformation wherein raw heterogeneous data is made suitable for interoperability with a number of different machines acting within a particular process. This approach enabled the prediction of machine failure that is considered here. Other problems that were considered include data integration and modelling, real-time access to the data, privacy and security, and data presentation. This work assesses the impact of latency issues on prediction. Other DDS type protocols have not been implemented as yet but will be considered in future work.

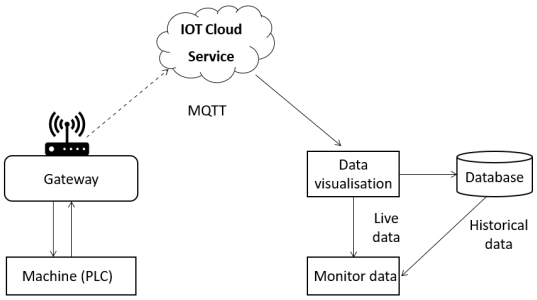

Figure 1: IoT architecture.

3 DATA STORAGE AND LABELLING CHALLENGES

A generalised diagram of the cloud architecture considered in this work is given in Fig. 1. The PLC is *any* industrial computer which is adapted for the direct control of a component of the manufacturing process and provides process fault diagnosis where appropriate for real time feedback (Bolton, 2015). The Open Platform Communication Unified Architecture (OPC UA) server is the software platform considered here for access to all PLC data and and is the MQTT broker in this use case. The data is transferred from the OPC UA client, and from there to the cloud. The Gateway in this setting is an industrial grade computer, containing an application which connects to the OPC UA server section 3.1. The critical data is filtered out at this level and sent to the cloud using a standardised MQTT protocol. The data visualisation is written in java using a spring boot framework in conjunction with NoSQL database and MongoDB.

3.1 OPC UA

The OPC UA standard, here controls all the communication that takes place on devices within machines, between machines, and from machines to systems in a convergence of Information Technology (IT) and Operational Technology (OT) for the grinding process within a commercial orthotics setting. The OPC UA is used with any software platform, *i.e.* it is scalable from minimum embedded controllers to massive

cloud infrastructures, but the Java platform is used here for reasons of convenience in relation to interfacing in real time with the NoSQL database.

3.2 Application of MQTT Protocol: The Methodology

As discussed in the previous section, this MQTT instance runs with a relatively small data packet size in order to keep the consumption of the power supply low (Atmoko et al., 2017). The physical layer protocol is 'data-agnostic', *i.e.* data can be transmitted in a variety of heterogeneous formats such as binary, text, XML, or JSON that is specified by the particular sensor that is employed on the machine. The PLC is responsible for individual sensor handshaking protocols.

This instance of MQTT requires two main software components:

- MQTT Client is the web platform developed to obtain the data.
- MQTT Broker is required to handle publish and subscribe data. A Linux platform handles the free available brokers (mosquitto and HiveMQ in this use case).

The data obtained from the process resulted in dataframes of two types of data, namely telemetry and alarm variables. More details on the dataset is discussed in sections 4.1 and 4.2.

MQTT protocol follows First-In-First-Out (FIFO) method. When an event occurs in the machine which raises an alarm, the MQTT protocol does not allow the event to be recorded in the server instantly but awaits it in a queue. Since the PLC contains over 500 telemetry variables, the queue at a certain time can be filled with a lengthy stream of data. This may cause significant delay in the recording of the alarm into the server. This time lag can vary depending on many factors such as the PLC being offline, number of awaiting notifications etc. Another probable reason for this random offset is the inbuilt clock in PLC which gets out-of-sync with the internet time.

4 MQTT DATASET OBSERVATIONS

This section deals with problems associated with the analysis of the MQTT data. Most data errors are detected during the processing of the downloaded dataset from the cloud. We present the data cleaning in a step-by-step process, involving cycles of screening, diagnosing, and editing of suspected data abnormalities.

Initially the data is downloaded from MongoDB in CSV files and imported into a pandas dataframe. There are over 500 variables in the PLC. The size of our database is over 30 GB, collected for a period of 3 months. Fig. 2 illustrates the first five rows of the data received from the database after initial cleaning.

4.1 Telemetry Variables

The telemetry variables refer to the process variables which generate values during the operation of the PLC, such as the speed of a spindle. Telemetry variables have:

- **id:** an identifier,
- **variable:** a variable name,
- **dateTime:** time at which the variable is uploaded to the cloud,
- **dataStatus:** True/False *i.e.* live/cached data,
- **alias:** description of the variable.

Once the telemetry variables are downloaded from the database as given in fig. 2, they are restructured with variable IDs in the columns and cloud timestamps in the rows. The data that is received from the database, has True/False tags. This is because, IoT cloud service broadcasts data at a predefined interval, which in this use case is 1 second. However, due to the issues defined in section 3.2, MQTT protocol broadcast cached data which are tagged as false. As given in fig. 3, the false data is filtered out for analysis purposes, which makes the resultant variable matrix quite sparse. The condition number, κ, of the variable matrix (A), is calculated as

$$\kappa = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} \approx 10^{10} \tag{1}$$

where σ_{max} and σ_{min} are the maximum and minimum values obtained after singular value decomposition of A. The high value of κ makes the post processing of the variable matrix quite difficult.

4.2 Alarm Variables

The alarm variables refer to the occurrence of different alarms during the operation of the PLC. The alarm data is similar to the telemetry variables except for the presence of one additional parameter *i.e.* the alarm timestamp. Alarm variables have:

- **id:** ordinal number of an alarm,
- **variable:** acknowledgement criterion for an alarm,

```
df_Var.head()
```

	_id	variable	value	dateTime	dataStatus	_class	alias	null
0	5e6121f93f96094544df5737	VarID_3109	I0	2020-03-05T15:59:53.380Z	False	com.example.demo.models.MqttInfo	NaN	1583423993380
1	5e6121f93f96094544df5736	VarID_3119	I-1	2020-03-05T15:59:53.376Z	False	com.example.demo.models.MqttInfo	NaN	1583423993376
2	5e6121f93f96094544df5735	VarID_3129	0	2020-03-05T15:59:53.373Z	False	com.example.demo.models.MqttInfo	NaN	1583423993373
3	5e6121f93f96094544df5734	VarID_100119	NCU	2020-03-05T15:59:53.369Z	False	com.example.demo.models.MqttInfo	SourceName	1583423993369
4	5e6121f93f96094544df5733	VarID_100129	Active	2020-03-05T15:59:53.366Z	False	com.example.demo.models.MqttInfo	ActiveState	1583423993366

Figure 2: Data downloaded from database after initial cleaning.

VarID_2529	VarID_2109	VarID_2179	VarID_2528	VarID_1218	VarID_2178	VarID_2158	VarID_1198	VarID_1217	VarID_1227	...	dateTime
9.326904	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.583416e+12
NaN	-0.157922	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.583416e+12
NaN	NaN	5.295	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.583416e+12
NaN	NaN	NaN	0.002686	NaN	NaN	NaN	NaN	NaN	NaN	...	1.583416e+12
NaN	NaN	NaN	NaN	84.943359	NaN	NaN	NaN	NaN	NaN	...	1.583416e+12
...	...	...	...	...	...	...	...	...	...	...	1.583416e+12
4.235107	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.583416e+12
NaN	NaN	6.071	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.583416e+12
NaN	NaN	NaN	-0.338379	NaN	NaN	NaN	NaN	NaN	NaN	...	1.583416e+12
NaN	NaN	NaN	NaN	NaN	-0.461	NaN	NaN	NaN	NaN	...	1.583416e+12
NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.036621	NaN	NaN	...	1.583416e+12

Figure 3: Variable matrix (A) with True values.

- **dateTime:** time at which the variable is uploaded to the cloud,
- **dataStatus:** True/False *i.e.* live/cached data,
- **Time stamp:** the time of the alarm occurring in the PLC, as per the PLC clock.
- **alias:** description of the alarm.

Fig. 4, denotes a snippet of the alarm variables and fig. 5 denotes the extracted alarm timestamps. As given in fig. 5, the alarm timestamp contains the time of the alarm according to the clock inside PLC, denoted by the column **value**. However, the actual/internet time of uploading of the alarm to the cloud by MQTT, is given by the column **timestamp1**, *i.e.* the unix time value of column **timestamp2**.

Fig. 6 denotes our alarm matrix created to front-run the alarms with the variables (fig. 3). However, due to the sparsity in the variable matrix and the validity of the alarm timestamps as discussed next, 'front-running', i.e., prediction of alarms is simply not possible. The alarms have a cloud timestamp and a machine timestamp, while the telemetry variables only have a cloud timestamp. For the alarm variables the time window between the machine time tags versus the cloud time tags is non linear and time varying. There exists a mean and standard deviation of 35 minutes each on this window. This is a major limiting factor on the utility of this sparse dataset for alarm prediction. Fig. 7, displays the gaussian distribution $g(x)$ of the delay in between the machine and cloud timestamps. Here,

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) \quad (2)$$

where x denotes the distribution, μ and σ denote the mean and standard deviation of the distribution respectively.

Any attempt to clean this data amounts to the creation of a *synthetic dataset* from raw data. Synthetic tag timestamping injects error into the creation of the synthetic data that renders it useless for alarm front running. This is due to the inability to produce a correlation function between the telemetry variables and alarms with any certainty due to the sparsity in the alarm and telemetry data. The outcome of this study is therefore that the alarms cannot be predicted using existing PLC based interfaces and that new sensor hardware needs to be deployed in order to predict alarms. This is useful intelligence in its own right for the company involved in this work. The initial design brief was to determine the limit of performance of the existing setup.

	_id	variable	value	dateTime	dataStatus	_class	alias
440023	5e6101c93f96094544d8a060	VarID_3019	0	2020-03-05T13:42:33.510Z	True	com.example.demo.models.MqttInfo	Acknowledgement criterion for an alarm
440027	5e6101c93f96094544d8a05c	VarID_3059	0	2020-03-05T13:42:33.495Z	True	com.example.demo.models.MqttInfo	Acknowledgement criterion for an alarm
440051	5e6101c93f96094544d8a044	VarID_3018	0	2020-03-05T13:42:33.412Z	True	com.example.demo.models.MqttInfo	Ordinal number of an alarm
440054	5e6101c93f96094544d8a041	VarID_3048	0	2020-03-05T13:42:33.402Z	True	com.example.demo.models.MqttInfo	Alarm Code 06
440055	5e6101c93f96094544d8a040	VarID_3058	0	2020-03-05T13:42:33.399Z	True	com.example.demo.models.MqttInfo	Ordinal number of an alarm

Figure 4: Alarm Values.

	_id	variable	value	alias	timestamp1	timestamp2	difference	tag
430	5df8b4a2f708502de88923f6	VarID_3017	1576578045661	Time Stamp	1576580258609	Tue, 17 Dec 2019 10:57:38 GMT	2212948	False
470	5df8b4a3f708502de889241e	VarID_3009	1576580701043	Time Stamp	1576580259024	Tue, 17 Dec 2019 10:57:39 GMT	442019	False
911	5df8b4acf708502de8892644	VarID_3017	1576578045661	Time Stamp	1576580268312	Tue, 17 Dec 2019 10:57:48 GMT	2222651	False
951	5df8b4acf708502de889266c	VarID_3009	1576580701043	Time Stamp	1576580268749	Tue, 17 Dec 2019 10:57:48 GMT	432294	False
1392	5df8b4b6f708502de8892892	VarID_3017	1576578045661	Time Stamp	1576580278002	Tue, 17 Dec 2019 10:57:58 GMT	2232341	False

Figure 5: Alarm Timestamps.

	VarID_3017	VarID_3009	VarID_3041	VarID_3033	VarID_3025	VarID_3049	VarID_3057	dateTime
0	1576578045661	NaN	NaN	NaN	NaN	NaN	NaN	1.576580e+12
1	NaN	1576580701043	NaN	NaN	NaN	NaN	NaN	1.576580e+12
2	1576578045661	NaN	NaN	NaN	NaN	NaN	NaN	1.576580e+12
3	NaN	1576580701043	NaN	NaN	NaN	NaN	NaN	1.576580e+12
4	1576578045661	NaN	NaN	NaN	NaN	NaN	NaN	1.576580e+12

Figure 6: Alarm matrix with True values.

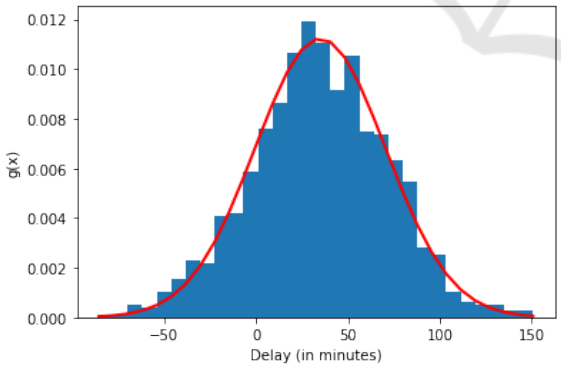

Figure 7: Probability density function of the delay.

5 PROPOSED SYSTEM FOR REAL-TIME DATA ANALYTICS

In order to advise the company on the next steps that are necessary to make the dataset suitable for any alarm throwing analysis/post processing, an architecture has been proposed for the upgrade of the existing data collection system. This availability of the data is often, in the authors experience, not known a priori by the industrial client and a project such as this is often very useful in terms of educating a company as to the limit of performance for a particular machine learning experiment. It is often the case that, just because a large quantity of data is being collected, there is an expectation that these vast quantities of data will per force lead to actionable manufacturing intelligence. However, studies such as these can often need to be carried out in order to once again confirm the law of *garbage in* leading to *garbage out*. For many client companies their current system architecture needs to be health checked in terms of the time syncing of timestamps before new investments will be made in sensor infrastructure. This is a necessary stage before proper post processing and then real-time data analysis can be performed.

For this client the following architecture has been proposed to enable predictive modelling. Fig. 8 illustrates the proposed system architecture in form of a flow chart. The steps of the flow chart are discussed as follow:

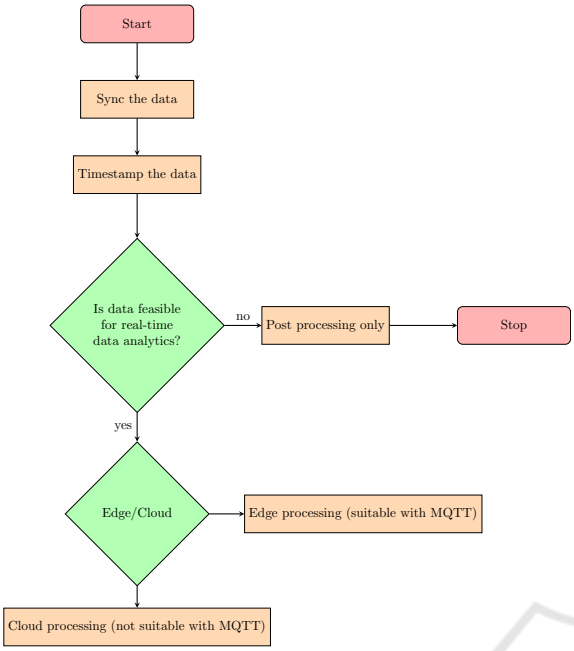

Figure 8: Proposed system.

- Maintenance of PLC clocks in the factory floor so that they are synced with internet time.
- For all variables, telemetry and alarm, the data needs to be time tagged by the machine when it is captured. Currently, the telemetry variables are not time tagged by the machine.
- Once the above steps are completed, it is checked if the data is feasible for real-time analytics or if the data is suitable only for post-processing. MQTT batches up the data before sending it to the cloud. This prevents real-time data analysis, as it requires a constant stream of data. However, when the data is batched up before sending, such as in the case study presented in this paper, post processing with a feasibility study can be done as long as it has associated time tags denoting when the measurements are made.
- For real-time data analysis batching has to be removed between the act of capturing the data and the act of processing because processing is done in the cloud or at the edge.

  Real-time data analysis can be done in two ways:

  – **Edge Computing:** For processing at the edge, the data and time tags are captured, processed, and then batched prior to being sent to the cloud. However, edge processing can only be performed in form of a closed loop correction. This means that to perform real-time data analysis, the processing of the data takes place at the edge and the result is notified to the operator to make changes in the PLC, before the data is uploaded in the cloud. Edge processing is suitable with MQTT as the corrective action takes place before the data is batched up and sent to the cloud.

  – **Cloud Computing:** For processing in the cloud, the data and time tags can be captured, sent to the cloud, and processed in real-time, then saved in the cloud, with batching removed. Cloud processing requires a constant stream of data and therefore MQTT may not be used for real-time data analysis via cloud. Other IoT protocols, such as DDS, can be useful to facilitate cloud computing.

The initial deployment cost of option 1. above is more significant than its cloud based alternative. There is a significant opex cost associated with the data transfer to the cloud. It is up to the business concerned to make a decision regarding which option best suits its business model. Once this decision is made we will revisit this analysis in order to determine the alarm tagging that is necessary for real-time process control.

6 CONCLUSION

Predictive modelling and real-time data analysis use cases are gaining popularity as the cost of data collection and transmission from sensors continue to drop. This paper has considered the application of MQTT in one such data streaming application for a grinding process within a commercial orthotics manufacturing company. Significant problems have been highlighted in the application of the MQTT protocol to this process as poor time tagging was exhibited within the dataset. The current setting of time tagging leads to the creation of sparse matrices for alarm prediction. Therefore, reliable front running of the grinding process is not possible with the present sensor infrastructure. Two design solutions have been presented to the company to mitigate this fundamental problem. We are presently awaiting a decision to be made that is dependent on the current post COVID 19 climate that is still taking shape. In future work, an Edge based MQTT protocol will be considered and its performance will be benchmarked against the corresponding DDS protocol to determine which approach is the most useful for alarm prediction in this commercial process.

REFERENCES

Atmoko, R., Riantini, R., and Hasin, M. (2017). Iot real time data acquisition using mqtt protocol. *Journal of Physics Conference Series*, 853(1).

Banks, A. and Gupta, R. (2014). Mqtt version 3.1. 1. *OASIS standard*, 29:89.

Bolton, W. (2015). *Programmable logic controllers*. Newnes.

Khan, M., Wu, X., Xu, X., and Dou, W. (2017). Big data challenges and opportunities in the hype of industry 4.0. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6.

Kramer, J. (2009). Advanced message queuing protocol (amqp). *Linux Journal*, 2009(187):3.

Luzuriaga, J. E., Perez, M., Boronat, P., Cano, J. C., Calafate, C., and Manzoni, P. (2015). A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 931–936.

Möller, D. P. (2016). Digital manufacturing/industry 4.0. In *Guide to Computing Fundamentals in Cyber-Physical Systems*, pages 307–375. Springer.

Novelli, L., Jorge, L., Melo, P., and Koscianski, A. (2018). Application protocols and wireless communication for iot: A simulation case study proposal. In *2018 11th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP)*, pages 1–6.

Pardo-Castellote, G. (2003). Omg data-distribution service: architectural overview. In *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings.*, pages 200–206.

Peralta, G., Iglesias-Urkia, M., Barcelo, M., Gomez, R., Moran, A., and Bilbao, J. (2017). Fog computing based efficient iot scheme for the industry 4.0. In *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pages 1–6.

Silva, D. R. C., Oliveira, G. M. B., Silva, I., Ferrari, P., and Sisinni, E. (2018). Latency evaluation for mqtt and websocket protocols: an industry 4.0 perspective. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 01233–01238.

Vaidya, S., Ambad, P., and Bhosle, S. (2018). Industry 4.0–a glimpse. *Procedia Manufacturing*, 20:233–238.

Wang, T., Li, Y., Chen, Y., Tian, H., Cai, Y., Jia, W., and Wang, B. (2017). Fog-based evaluation approach for trustworthy communication in sensor-cloud system. *IEEE Communications Letters*, 21(11):2532–2535.

Yang, J., Sandström, K., Nolte, T., and Behnam, M. (2012). Data distribution service for industrial automation. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pages 1–8.

Yin, S. and Kaynak, O. (2015). Big data for modern industry: challenges and trends [point of view]. *Proceedings of the IEEE*, 103(2):143–146.