Simulation Based Performance Evaluation of FIWARE IoT Platform for Smart Agriculture

Kari Kolehmainen[1], Marco Pirazzi[1], Juha-Pekka Soininen[1] and Juha Backman[2]
[1]*VTT, Technical Research Centre of Finland, Finland*
[2]*LUKE Natural Resources Institute of Finland, Hyvinkää, Finland*

Keywords: IoT, Robotics, Performance, Smart Agriculture.

Abstract: In the domain of smart agriculture there is a growing demand for the development and implementation of robotics and Internt of Things (IoT) solutions. Using robots and autonomous vehicles such as Unmanned Aerial Vehicles (UAVs) for increasingly complex tasks requires coordinating robotic operations taking into account other robots doing complementary tasks. Using IoT platforms for adding intelligence to cooperation and coordination is a lucrative possibility. Performance constraints limit the tasks in which co-operation can be used. Information latency is a key factor for moving autonomous robots in many cases. Using the FIWARE IoT platform for information integration offers the flexibility of combining cloud-based AI analysis with robot operations, however it comes with the cost of increased latency. The messaging frequency that is dependent on the number of parallel robots, as well as their configuration, affects the overall latency of the IoT system. We present the composition of latency in the FIWARE IoT system and its limit in a practical deployment scenario.

1 INTRODUCTION

The need to increase the yield of agriculture per land area as well as the cost of labor unit drive the deployment of robotics in agriculture. As a result, autonomous or semi-autonomous devices are increasingly being utilized in modern agriculture (Grieve et al., 2019) (Friha et al., 2021). This can encompass Unmanned Aerial Vehicles (UAVs), robot tractors, and other utility vehicles designed to reduce manual labor, increase efficiency, or provide detailed data on the fields and crops. Each robotic system still requires its operator and support systems and as such coordination becomes challenging. Coordinating multiple robot operations on the field simultaneously often lacks systematic approaches.

Internet of Robotic Things (IoRT) is a sub-theme of IoT focusing on inter-operating robots and devices. (Ray, 2016). IoRT started as a cloud robotics concept where part of the computation was executed in the cloud. Later it has included also approaches where robots are considered as things and robot virtual models have been created into IoT systems. There have been solutions where robots are modeled as virtual entities, i.e. digital representations of physical things, but also as digital twins for robots. Digital representation as IoT nodes gives possibilities to manage robots in the same way as other things and to use techniques developed in IoT solutions such as Artificial Intelligence (AI), data analytics, and data fusion solutions that are typical in IoT platforms. The Digital twin approach gives tools for example predictive maintenance and to a simulation of control options. In the case of heterogeneous robot fleets, the IoT platform can be used as a common framework for managing fleet operations. Latency is identified as one of the key factors affecting the use of IoRT (Vermesan et al., 2020). Forming a realistic perception of latency involved with robots sharing data affects the feasible use cases of collaborative robots.

In this paper, we analyze the performance constraints of using a modern IoT platforms such as FIWARE for monitoring and operating a fleet of multiple robots. Understanding bottlenecks on the latency of overall multi robot control system is important for assessing feasibility of the overall system. Robots encompass various types of autonomous devices operating in different roles in the same field. Collecting data from different sources enables safer operation and building a system that is capable of integrating data from many sources as needed. The complexity of managing the operations for a robot fleet can be reduced by integrating non-robot specific middleware. Using open interfaces for robot fleet management has provided encouraging results in Smart Manufacturing use cases. (Quadrini et al., 2020)

Kolehmainen, K., Pirazzi, M., Soininen, J. and Backman, J.
Simulation Based Performance Evaluation of FIWARE IoT Platform for Smart Agriculture.
DOI: 10.5220/0011918700003482
In *Proceedings of the 8th International Conference on Internet of Things, Big Data and Security (IoTBDS 2023)*, pages 73-81
ISBN: 978-989-758-643-9; ISSN: 2184-4976

2 MOTIVATION

The motivation for increasing the complexity of robotic operations by introducing collaboration of robots is to leverage and benefit from the different capabilities of robots. One use case for such collaboration is to improve situational awareness of robot tractors in smart agriculture. An example use case is silage harvesting where one or more tractors are mowing and raking the hay while other tractors are baling the hay. Tractors can benefit from Situational awareness provided by UAVs coupled with AI-based object recognition. The combination of these methods allow for collaboration and enables the farm to use resources more efficiently and safely.

Another example use case is seeding where multiple work phases can be potentially automated with an autonomous robot fleet. Leveraging cloud-based AI analytic help overcome some of the challenges related to work distribution between robots.

AI analytic services subscribe to data provided by UAV and analyse video stream provided by UAV to detect objects from the field. Hazard detection algorithms can then analyse the detected objects along with their heading vectors to anticipate possible collision events. Latency of the information is critical in this scenario in order to determine wide enough safety boundaries for the objects and robots.

The latency in this proposed multi-robot fleet monitoring system can be divided into three categories. First, is latency coming from analyzing the data provided by the robots. The second is latency caused by data transfer in wired and wireless networks. The third is the overhead latency caused by the IoT platform. This paper focuses on determining the IoT system latency and boundary conditions which affect that.

The reason for this focus is that it is the source of latency that can be affected by the deployment and configuration of the system. Network latency is the major source of latency but is not possible to affect it. Latency from data analysis is relatively easy to assess. Latency caused by the IoT system on the other hand is easy to overlook and include in the network proportion.

This system was deployed in the European Union (EU) research project FlexiGroBots where it demonstrates the use of a drone(s) with robot tractors to enhance overall situational awareness. This approach can be also used to identify needs for analytics and robot services, i.e. identifying weeds and pests from the monitoring data as input for weeding robots, etc.

3 PROPOSED SYSTEM

Devices needed for implementing the system are first of all robots and robot control systems that can communicate with the IoT system. The use of IoT systems in smart agriculture use cases is well-documented (Farooq et al., 2019) (García et al., 2020). FIWARE is used as a baseline IoT system. FIWARE is a context management system that implements **N**ext **G**eneration **S**ervice **I**nteface (NGSI) data model for internal data structure. The heart of the FIWARE system is a context broker (i.e. Orion) that maintains entities and their context information. Context broker implements a publish/subscribe interface to inform other components about entity changes.

MQTT is an extra layer of abstraction in the sense that it decouples the IoT platform from the robots. While MQTT allows flexibility for the system setup it induces extra latency (Pereira et al., 2018). Whether that is an acceptable trade-off depends on system requirements.

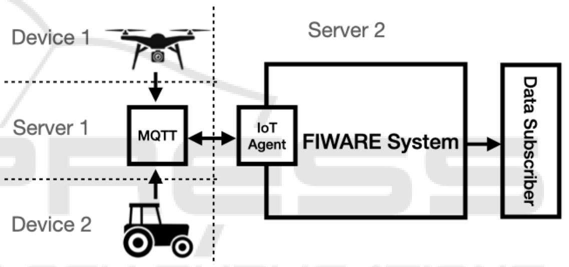

Figure 1: Deployment architecture of simulation and final systems.

The motivation for using the MQTT broker is to simplify the deployment of the system and to decouple devices from underlying IoT platform implementation. Robot and robot control systems publish data to the MQTT broker to which FIWARE IoT Agents subscribe. IoT system handles data using NGSI but none of the robot systems provide information in NGSI format directly. IoT Agent is the FIWARE component that is responsible for translating information for the NGSI data model.

Existing research on FIWARE performance characteristics has proven that the FIWARE platform deployed in commercial cloud computing platforms can scale up to a rate of 1000 requests per second before becoming unstable. After that latency also starts to increase significantly. Even then the platform imposes a heavy load on the underlying system. (Araujo et al., 2019)

4 MEASUREMENT SETUP

Performance measurements for this study were conducted in two phases. Performance of IoT platform is measured in isolated setup illustrated in figure 1 and described in detail in table 1. Performance characteristics are then validated with a simulator deployed in a way that corresponds to real deployment as far as possible.

The performance measurement system is composed of IoT Platform, MQTT Broker, and Load Generator each running on separate virtual servers as per in figure 2. IoT platform is a standard FIWARE platform with key components and Orion Context broker as a centerpiece of FIWARE. MQTT Broker is a separate publish/subscribe service used by devices for communication as per figure 1. A load generator is a component used for simulating traffic coming from robots. Load generator generates robot position messages to MQTT broker. A load generator is used for getting repeatable data for IoT platform stress testing.

Robot simulators (jMAVsim and ground robot simulator) replace load generators in performance validation tests. Simulators run on their separate systems as described in figure 1. FIWARE IoT platform and MQTT Broker are deployed in validation measurement the same way as with performance measurements.

jMAVsim is a Java-based drone simulator running on a Linux system. jMAVsim is a standard drone simulator used for simulating Dronecode based UAVs that use MAVlink communication protocol. A custom ground robot simulator is a robot simulator implemented in Python to simulate robot tractor movements on the field.

4.1 Environment Setup

MQTT broker and FIWARE components run on the same computer hardware but in separate virtual servers. FIWARE components are deployed as docker containers in a virtual server.

The environment is installed on a Dell R740 server running CentOS Linux. The virtualization environment is built on Linux Kernel-based Virtual Machine (KVM) and libvirt virtualization library. Different components are running on their dedicated virtual servers that are running a Docker environment. Docker is used for components as it is a common service run on many cloud services.

For Measurement accuracy reasons the simulators are run on the same physical hardware to eliminate the effects of clock synchronization issues. The logical structure of the setup is illustrated in the figure 2

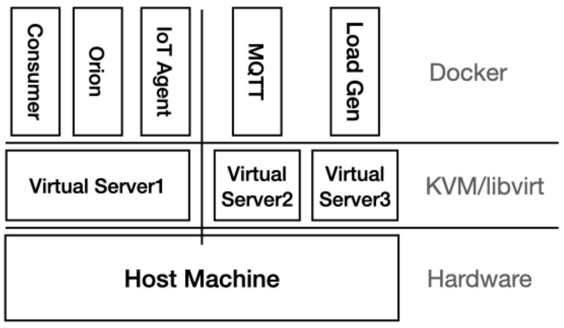

Figure 2: Measurement system setup.

Cloud infrastructure is running on a virtual cloud server that has four CPU cores (Intel Xeon Silver 4208), 16Gb RAM, and 120Gb hard drive space.

Timestamps are collected as EPOCH timestamps in microsecond resolution. Hooks for timestamp collection are added to the relevant components.

- MQTT Broker the hook is added to "handle__publish" -function. Topic, as well as a timestamp, are printed to match timestamp with appropriate robot message
- IoT Agent that is used is implemented with Node.js. The IoT agent has a handler for incoming device updates called "deviceUpdatingHandler". Node.js timestamp resolution is milliseconds
- Orion Context Broker is implemented with C++ and for NGSIv2. Orion prints log entries with millisecond precision on incoming updates.

For practical purposes, additional FIWARE Generic Enablers like time-series data collection facilities are left out of the measurements as they are not relevant for context-based operations.

4.2 Clock Synchronization

Clock synchronization is a major challenge in collecting performance metrics based on timestamps with distributed systems. The main protocols for synchronizing clocks of networked devices are NTP (Network Time Protocol) and PTP (Precision Time Protocol). Errors in NTP-based systems are in the 10-100ms range while sub-millisecond synchronization is achievable with software-only PTP implementation and sub-microsecond synchronization with hardware PTP implementation.(Neagoe et al., 2006)

Another complication in the measurement environment is that there are several layers of virtualization involved which potentially can affect timestamp collection. Cloud services are running on a virtual server on a Linux host machine and individual services running on a virtual server are running as

Table 1: System components.

Component	Level	Description
Host Machine	HW	CentOS 7.1 running on Dell PowerEdge 740 with 32 core Intel Xeon Silver 4208 CPU @ 2.10GHz
Server1	KVM	FIWARE Host virtual server with Ubuntu 20LTS Linux with 4 CPU cores, 16GB RAM, and 120GB disk space
Server2	KVM	MQTT virtual server Ubuntu 20 LTS server with 4 CPU cores, 16GB RAM and 120GB disk space
Server3	KVM	MQTT Load generator virtual server Ubuntu 20 LTS server with 4 CPU cores, 16GB RAM, and 120GB disk space
Orion	Docker	Orion context broker
IoT Agent	Docker	Component that implements data translation from robot to NGSI format
Consumer	Docker	Mock AI measurement component for data coming from FIWARE platform
MQTT Broker	Docker	Mosquitto MQTT Broker modified to log timestamps of incoming context updates
Load Gen	Docker	Python MQTT client generating MQTT messages for the broker that simulate Robot context updates

Docker containers. To ensure correct timestamp values, the virtual server is set to use the Constant Time Stamp Counter (TSC) of the host machine processor and PTP synchronization. Clock synchronization is relatively accurate (within 1ms skew) with the virtual server as long as the host system load is kept under 60%. (Ahuja et al., 2021)

Clock synchronization is implemented with Chrony daemon set to take reference clock from /dev/ptp0 that is mapped to the host system constant timestamp counter (TSC). Data is collected as recorded log files during simulation and then analyzed after the data collection phase. This enables components running on cloud service to have synchronized clocks but leaves open the synchronization of clocks of geographically distributed computers.

Network latency and clock skew are measured from geographically distributed computers separately. Depending on the computer as high as 200ms differences in clock synchronization were observed. Clock difference and latency are measured simple timestamp echo client/server script that implements a two-way method for clock synchronization (Levine, 2016). Script sends local EPOCH timestamp to other system and receives remote timestamp in reply. This allows us to calculate round trip delay and clock difference assuming symmetrical delays. Delay and clock difference is characterized by taking 100 samples and averaging the values.

In a simulation environment using stable wired networks, this method gives sufficiently accurate results. In a real-world implementation using mobile networks, delays can not be assumed to be symmetric, and more comprehensive methods should be used.

4.3 Measurement Points and Latency Metrics

Times Stamps (TS) are collected from several points for analysis of latency and throughput constraints. Time stamps are collected in micro (or milli) second resolution. The first timestamp (TS1) is from the moment when message is published to the MQTT broker. The second step (TS2) is when the MQTT broker receives the published message. The third step (TS3) is the moment when IoT Agent receives the message from MQTT Broker. The fourth step (TS4) is when Orion receives an NGSI format Entity update request from IoT Agent. The fifth and final timestamp (TS5) is the moment that the Orion Entity update message is received at subscribing Mock AI service.

The latency that we measure is derived from collected timestamps. Timestamps are taken at the beginning of each step. For instance, TS3 is the timestamp of when a message is received at the IoT Agent incoming port. Logging of the timestamp creates a small overhead but it is determined to be irrelevant.

Lat1 and Lat 4 of these latency measurements are mostly related to networking lantency as they are measuring latency in and out from the FIWARE platform. Lat2 and Lat3 are related to latency of the IoT platform that consist of MQTT Broker as well FIWARE Platform. Lat2 and Lat3 are the points where

Table 2: Latency measurements.

Measure	Calculation	Description
Lat1	TS2-TS1	Network latency from robot to MQTT Broker
Lat2	TS3-TS2	Latency from MQTT Broker and IoT Agent
Lat3	TS4-TS3	Data conversion latency in IoT Agent
Lat4	TS5-TS4	Orion publish latency
Lat5	TS5-TS1	Overall latency of the whole system

data conversions and storage takes place.

5 SIMULATION RESULTS

5.1 IoT Platform Performance Measurements

MQTT and FIWARE Latency were estimated first with a load generator to determine the sequential message processing capacity of the system. Measurements are completed with 1ms steps from 1ms to 10ms where the rate of change in latency is significant. Two measurements with longer 5ms intervals are taken to verify that there is no unexpected behavior with longer intervals. On each step, over 800 samples are generated to generate a statistically significant number of samples. Sample mean difference of Lat5 measurement settles to 0.1% difference in about 600 sample size so 800 samples gives some buffer to achieve reliable arithmetic mean for latency. As expected relative difference in latency measurement is highest on longer messaging interval cases.

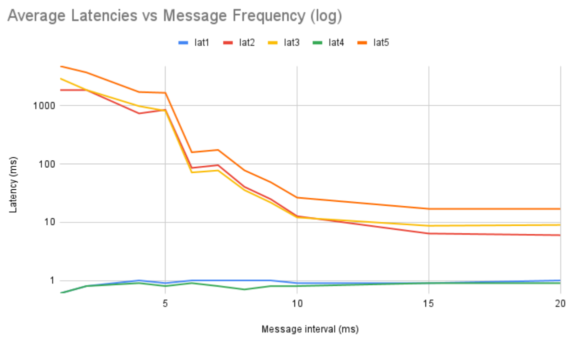

Figure 3: Latency measurement results.

The load was generated with different intervals starting from 1ms to 30ms. Data was published via MQTT in UltraLight format that can be interpreted by FIWARE IoT agent. IoT Agent then updated the Orion context broker entity with new values. Orion provided subscribed applications with new values via REST API. Program providing REST API for Orion updated resided in the same system as the FIWARE platform.

Figure 3 illustrates the behavior of the system. Latency drops dramatically with messaging intervals from 1ms to 10ms and stabilizes after a 15ms message interval.

Table 3: Measurement results (milliseconds).

Interval	lat1	lat2	lat3	lat4	lat5
1	0.6	1863.9	2953.7	0.6	4818.6
2	0.8	1870.7	1876	0.8	3747.8
4	1	740.7	988.5	0.9	1727.1
5	0.9	854.1	820.9	0.8	1674.1
6	1	85.9	71.7	0.9	159.3
7	1	95.6	77.6	0.8	174.9
8	1	40.6	35.7	0.7	78
9	1	24.9	21.8	0.8	48.7
10	0.9	12.7	12.1	0.8	26.5
15	0.9	6.4	8.7	0.9	17
20	1	6	9	0.9	17

From this data represented in figure 3 and table 3 it can be deduced that measurement latency is kept reasonable when the system is loaded with a maximum of 100 messages per second. Distribution of latency indicated that MQTT Broker as well as Orion event notifications cause minimal overhead. The majority of latency is spent with IoT Agent and Orion Entity update as shown in figure 4 even in a lightly loaded case with a 15ms message interval.

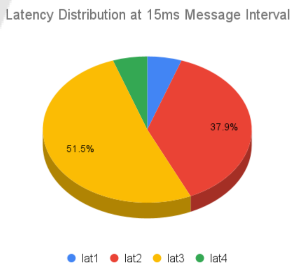

Figure 4: Distribution of latency.

Another relevant metric is the system tolerance to load spikes. From the generated load data we can observe how latency behaves at the beginning of the test at high message frequencies. The premise is that buffering of messages enables the system to process several samples at the beginning of the test at a higher rate than later in the test when the system is working at maximum capacity.

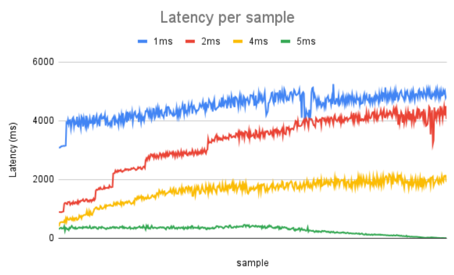

Figure 5: Sample Latency with different frequencies.

From figure 5 it can be observed that this phenomenon is happening to a degree even though the latency is high from the very first samples onward.

The filling of messaging buffers can be observed with incremental steps with 1ms and 2ms sample intervals. Latency increases in clear steps while with longer message intervals increase of the latency is more gradual. Interestingly latency decreases over the test time with a 5ms sample interval. Between 4ms and 5ms message intervals there seems to be a tipping point where the system is capable of processing messages and cope with load over time.

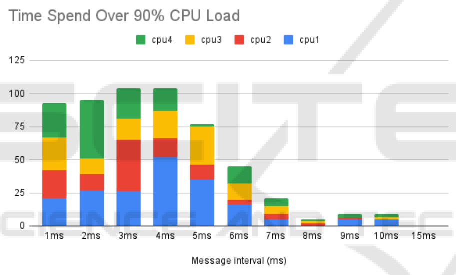

Figure 6: FIWARE CPU load per messaging interval.

Figure 6 shows how the FIWARE system CPU load behaves with different message intervals. This further highlights the tipping point at the 4ms message interval. Before the 4ms tipping point system load is less meaning that latency is dictated by network and Input/Output (I/O) throughput rather than CPU processing capacity.

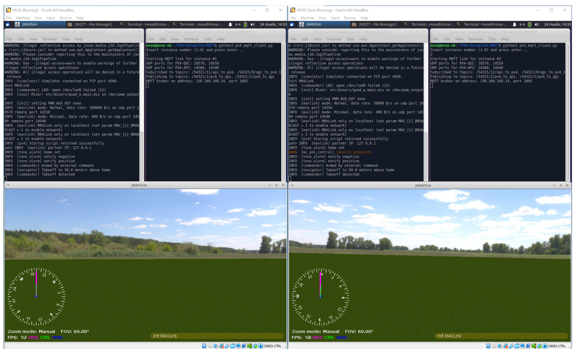

Figure 7: jMAVsim simulators running PX4 firmware.

5.2 Robot Simulator Performance Measurements

The IoT platform was tested with simulated drones and robots to validate platform performance as close to the real use case as possible. Two parallel PX4-based drones were deployed with simulated missions with the same virtual farm plot as four simulated robot tractors (figure 7).

jMAVsim simulators update their position fix roughly 40 times per second. In practice, frequency is slightly less than that around 30 to 35 updates per second. Simulated robot tractors update their positions once per second. In total maximum combined messaging frequency is slightly less than 80 messages per second as can be seen from the dotted line on figure 8

80 messages per second is an average message interval of 12.5ms which is well within the performance bracket of the FIWARE IoT platform running on the test environment. On simulator measurements test arrangement differ from platform performance performance measurements in a way that instead of load generator Drone simulators are used to generate realistic data. Two simulated drones are run on their own virtual machines that are hosted on a single HP Laptop. The laptop is connected to internet via WiFi network and with measured network latency of 17ms between the laptop and the MQTT broker. Four simulated robot tractors were running on a different computer and different network that had 30ms network latency to the MQTT broker. Simulation was run on single long execution of mission where drones and robot tractors execute collaborative tasks on smart agricultural scenario. Latency and message transmission statistics collected from the simulatation run are visualised in the figure 8. From data presented in figure 8 it can be seen that the latency of simulated tractors and drones remains relatively constant when the number of messages is kept within the performance bracket of the FIWARE system determined by prior performance analysis. This is emphasized by the intermediate pauses on the mission of Drone2 when number of messages per second is nearly halved but not affecting the latency of the other robots. This is consistent behaviour with the stress testing results.

6 DISCUSSION

Based on the measurements it can be argued that FIWARE is capable to cope with messaging as long as the message interval is kept longer than 5ms. Messaging frequency used in simulations proved the assessment from platform stress testing that at 12.5ms

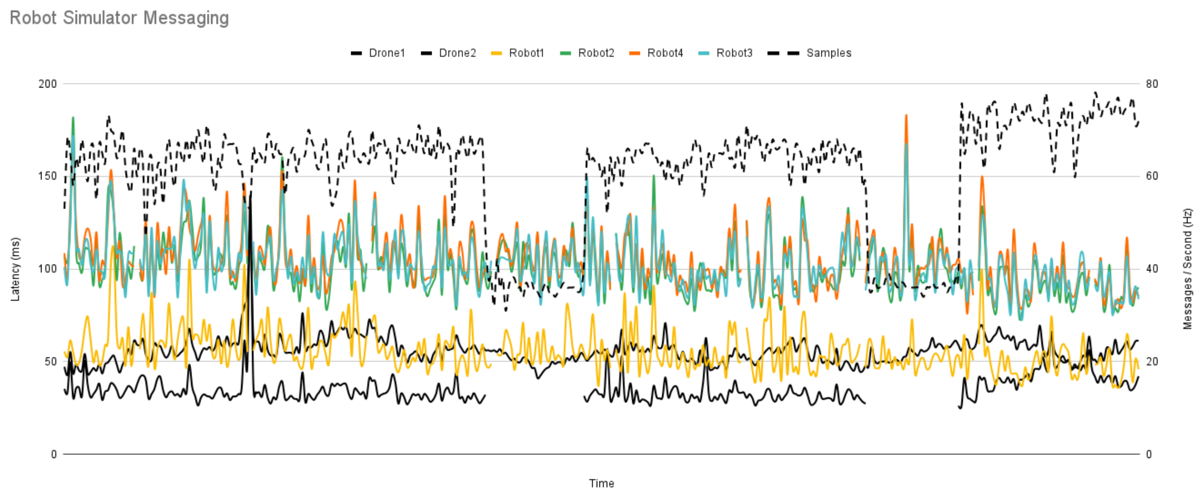

Figure 8: Simulator messages processed by IoT platform. Continuous lines present latency of robots and dotted line overall messaging frequency.

messaging interval platform latency is not significant. That can be seen from figure 8 where the gaps of one simulator going offline do not change the latency of other running simulators.

The effect of payload size with a combination of messaging frequency was not measured in this study. The payload of the Entity update on the MQTT message was in the ultralight format. For example, position update payload can be

$lat|65.0001|lon|25.2728|ele|100.00|h|270.1|v|10.15$

where lat, lon, ele, h, and v map to NGSI Entity attributes. This short message is interpreted by IoT Agent to match the Device specification which is in turn mapped to a specific NGSI Entity in the Orion context broker. NGSI is an ETSI standard for context information model. Translating message payload from ultralight format to more complex NGSI representation requires data processing which is computationally heavier than simply passing data between components. This mapping takes place in the IoT Agent that is represented by the Lat3 in our simulation measurements.

Our results of messaging latency of FIWARE differ from the study by Araujo et al. so that latency starts to increase in the messaging frequency of 200Hz whereas in the previous study the latency started to spike at 1000Hz. In that previous study focus was more on throughput which is less relevant in our case. The amount of the data is small in size and latency is the more important factor. The system was also set up in the Amazon Web Service (AWS) cloud which explains part of the difference. Using Linux sysbench tool CPU performance was given a score of 3950 while AWS cloud machines provide a score of around 7500. (Araujo et al., 2019) Another difference is that we are measuring latency until the update to the subscriber of the context information, and not just the database. This may have some effect on performance as well. This leads to the conclusion that the performance of the FIWARE IoT system can be scaled up if necessary by either improving the CPU performance of the host machine or by horizontally scaling the database as proposed by Araujo et al. (Araujo et al., 2019) (Zyrianoff et al., 2018)

6.1 Performance Requirements in Smart Agriculture Case

The motivation of the analysis is the piloting of multi-robot missions in a smart agriculture use case. Multi-robot mission deploys autonomous harvesting robot tractors on the field while drones provide situational awareness for the tractors using AI components from the IoT platform.

Having a realistic understanding of detection latency objects and robot locations is essential in determining sufficient safety buffers. Object detection itself generates some latency. Popular object detection frameworks such as You Only Look Once (YOLO) are capable of detecting objects in video streams that have a frame rate of 25fps in which case latency would be 40ms. (Lee and Hwang, 2021)

Our data was collected on a simulator environment running on wired broadband internet. The typical latency was 30ms. On mobile 4G network latency can be assumed to be around 100ms. In practice, the tractors operate on 3-4m/s speed and the total round trip latency in a realistic system would be about 250ms. This would lead to uncertainty of one meter of tractor location and safety margins should be set accordingly.

When message interval is kept within the capability bracket of the FIWARE platform it will cause only a marginal overhead. With 10ms messaging interval 26.5ms latency was measured. If loaded beyond the processing capacity, overhead very quickly increases to a significant amount. Scaling up cloud computing capability is necessary if the capacity of the FIWARE system becomes a bottleneck.

As network latency is dominating factor in normal use case some strategies to mitigate it is needed. Coping with network latency in this use case can be done in one of four ways:

- Increase safety zone to take into account the latency
- Decrease working speed of robots
- Move computation into Edge to mitigate network latency
- Use low latency networks such as 5G

In a practical use case scenario safety perimeter and working speed may be the easiest ways of coping with latency, but adding the edge capabilities or faster network would have less impact on actual work performance.

7 CONCLUSIONS

Understanding the distribution of latency in IoT data collection systems helps to pinpoint bottlenecks in the data collection system. Suitability to different use cases and their real-time requirements requires an understanding of system behavior.

The performance of the FIWARE system is dependent on underlying computing resources and the messaging load of the system. When going over the threshold, the latency increases from 17ms to over 4800ms. Determining the system load in terms of context updates per second is important to keep the latency within an acceptable limit. The underlying computing platform processing capability needs to be sized according.

Based on the findings here capability of the IoT platform can be determined with a test involving data subscriber and publisher. In the FIWARE system majority of the latency is coming from the IoT agent and Orion. Optimizing cloud service performance parameters for those two components provide the best results for overall performance gains. Additional latency is introduced with actual analysis as well as communicating messages back to the robot.

Based on the findings here it can be argued that it is feasible to use FIWARE in the simulated use cases where the number of simultaneously operating robots is limited. The latency caused by the IoT platform is reasonable. If the platform is serving several farms or fields simultaneously computing resources may need to be allocated consecutively to keep QoS acceptable.

ACKNOWLEDGMENT

This research was conducted by VTT and LUKE as part of the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017111.

REFERENCES

Ahuja, A., Jain, V., Saini, D., and Al-Turjman, F. (2021). Measuring clock reliability in cloud virtual machines, real-time intelligence for heterogeneous networks: Applications, challenges, and scenarios in iot. In *Het-Nets, 2021, 87-98.*

Araujo, V., Mitra, K., Saguna, S., and Åhlund, C. (2019). Performance evaluation of fiware: A cloud-based iot platform for smart cities. In *Journal of Parallel and Distributed Computing 132 (2019) 250–261.*

Farooq, M. S., Riaz, S., Abid, A., Abid, K., and Naeem, M. A. (2019). A survey on the role of iot in agriculture for the implementation of smart farming. In *IEEE Access, vol. 7, pp. 156237-156271, 2019, doi: 10.1109/ACCESS.2019.2949703.*

Friha, O., Ferrag, M. A., Shu, L., Maglaras, L., and Wang, X. (2021). Internet of things for the future of smart agriculture: A comprehensive survey of emerging technologies. ieee/caa journal of automatica sinica, 8(4), 718–752. https://doi.org/10.1109/jas.2021.1003925.

García, L., Parra, L., Jimenez, J., Lloret, J., and Lorenz, P. (2020). Iot-based smart irrigation systems: An overview on the recent trends on sensors and iot systems for irrigation in precision agriculture. In *Sensors 2020, 20, 1042. https://doi.org/10.3390/s20041042.*

Grieve, B. D., Duckett, T., Collison, M., Boyd, L., West, J., Yin, H., Arvin, F., and Pearson, S. (2019). The challenges posed by global broadacre crops in delivering smart agri-robotic solutions: A fundamental rethink is required, global food security. In *Volume 23, Pages 116-124, ISSN 2211-9124.*

Lee, J. and Hwang, K. (2021). Yolo with adaptive frame control for real-time object detection applications. In *Multimed Tools Appl (2021). https://doi.org/10.1007/s11042-021-11480-0.*

Levine, J. (2016). An algorithm for synchronizing a clock when the data are received over a network with an unstable delay. In *IEEE Trans Ultrason Ferroelectr Freq Control. 63(4):561-570. doi:10.1109/TUFFC.2015.2495014.*

Neagoe, T., Cristea, V., and L., B. (2006). Ntp versus ptp in com puter networks clock synchronization. In *IEEE*

International Symposium on Industrial Electronics, 2006, pp. 317-362.

Pereira, C., Cardoso, J., Aguiar, A., and R., M. (2018). Benchmarking pub/sub iot middleware platforms for smart services. In *Journal of Reliable Intelligent Environments (2018) 4:25–37*.

Quadrini, W., Negri, E., and Fumagalli, L. (2020). Open interfaces for connecting automated guided vehicles to a fleet management system. In *Procedia Manufacturing, Volume 42, 2020, Pages 406-413, ISSN 2351-9789, https://doi.org/10.1016/j.promfg.2020.02.055*.

Ray, P. (2016). Internet of robotic things: Concept, technologies, and challenges. In *IEEE Access, 4, p. 9489-9500, 2016, doi:10.1109/ACCESS.2017.2647747*.

Vermesan, O., Bahr, R., Ottella, M., Serrano, M., Karlsen, T., Wahlstrøm, T., Sand, H. E., Ashwathnarayan, M., and Gamba, M. T. (2020). Internet of robotic things intelligent connectivity and platforms. In *Frontiers in Robotics and AI, volume 7, 2020, https://www.frontiersin.org/article/10.3389/frobt.2020.00104*.

Zyrianoff, I., Heideker, A., Silva, D., and Kamienski, C. (2018). Scalability of an internet of things platform for smart water management for agriculture. In *Proceedings of the 23rd Conference of Open Innovations Association FRUCT (FRUCT'23). FRUCT Oy, Helsinki, Uusimaa, FIN, Article 58, 432–439*.