Journal Pre-proof

Evaluation of IoT stream processing at edge computing layer for semantic data enrichment

Fatos Xhafa, Burak Kilic, Paul Krause

PII: S0167-739X(19)32129-6
DOI: https://doi.org/10.1016/j.future.2019.12.031
Reference: FUTURE 5346

To appear in: *Future Generation Computer Systems*

Received date : 9 August 2019
Revised date : 24 October 2019
Accepted date : 25 December 2019

Please cite this article as: F. Xhafa, B. Kilic and P. Krause, Evaluation of IoT stream processing at edge computing layer for semantic data enrichment, *Future Generation Computer Systems* (2019), doi: https://doi.org/10.1016/j.future.2019.12.031.

Evaluation of IoT Stream Processing at Edge Computing Layer for Semantic Data Enrichment

Fatos Xhafa[†*], Burak Kilic[†], Paul Krause[‡]

[†]*Department of Computer Science, Technical University of Catalonia, Spain*
[On leave, University of Surrey, UK]*
[‡]*Department of Computer Science, University of Surrey, UK*

Abstract

The fast development of Internet of Things (IoT) computing and technologies has prompted a decentralization of Cloud-based systems. Indeed, sending all the information from IoT devices directly to the Cloud is not a feasible option for many applications with demanding requirements on real-time response, low latency, energy-aware processing and security. Such decentralization has led in a few years to the proliferation of new computing layers between Cloud and IoT, known as Edge computing layer, which comprises of small computing devices (e.g. Raspberry Pi) to larger computing nodes such as Gateways, Road Side Units, Mini Clouds, MEC Servers, Fog nodes, etc. In this paper, we study the challenges of processing an IoT data stream in an Edge computing layer. By using a real life data stream set arising from a car data stream as well as a real infrastructure using Raspberry Pi and Node-Red server, we highlight the complexities of achieving real time requirements of applications based on IoT stream processing.

Keywords: IoT Computing, Edge Computing, Data Stream Processing, Anomaly Detection, Data Stream Rate.

1. Introduction

Although the IoT as a concept, model and technology has been around for twenty years in the arena of new technologies, fully fledged IoT systems have become a reality at large scale only during the recent years. This paradigm shift in IoT computing is due to many factors such as the continuous improvements in networking, processing and data storage capabilities

of Cloud computing, which is the real factor for giving IoT such a prevalent role. Indeed, IoT systems have become an indispensable part of modern applications, from sensing to control and intelligent systems. Following the Cloud paradigm of "*everything-as-a-service*", Cloud computing has achieved the view of mobile devices, sensors, actuators and other kinds of devices "*as-a-service*" [9]. On the other hand, together with the *unlimited* capacity in processing power and data storage of Cloud data centers, it is possible not only to extract useful information from IoT data but most importantly to build knowledge, intelligence and decision making systems. Big companies have already such IoT systems in place. For instance, in IBM IoT Cloud, data collected from Arduino-Raspberry Pis are transmitted to IBM Cloud Node-Red servers. Then, the IBM Watson Analytics service processes the IoT data to generate knowledge and intelligence, which can finally be integrated into business workflows of enterprises. It remains a challenging issue for SMEs to achieve the benefits of IoT systems for their business processes.

Here, we start by discussing briefly on the challenges of the Cloud-to-thing continuum, which comprises Edge computing and Fog computing layers and how these new computing layers are propelling a whole family of offloading algorithms for IoT stream processing. Our goal is to study the challenges of processing an IoT data stream in an Edge computing layer by using real life data stream sets and a real infrastructure using Raspberry Pi and a Node-Red server. Through the study we highlight some of the challenges faced during the real life evaluation such as for anomaly detection in real time, accommodating various data rates for IoT data stream processing, etc.

The rest of the paper is organized as follows. In Section 2 we briefly overview the Edge computing as part of Cloud ecosystem. The IoT stream processing and semantic data enrichment are discussed in Section 3. In Section 4 we present the architecture of stream processing application and in Section 5 its experimental evaluation. We conclude the paper in Section 6.

2. Edge Computing

The new structure of Cloud-to-thing continuum poses new research challenges, among which we could distinguish three main ones [6, 16, 13, 11]: **(a)** processing and aggregating large data volumes generated at IoT layer before sending them to Cloud or to a *backend* system. Most of IoT devices do not have computing and storage capacities for this task, which becomes even a more complex challenge due to the heterogeneous nature of IoT devices.

(b) achieving low or very low latency, by avoiding IoT data transmission to Cloud, in order to support applications having real time requirements, requirements on user's mobility, critical applications that need to make decisions and control in short intervals of time (patients, production processes, etc.). **(c)** enhancing security, privacy and trust levels in IoT devices beyond the security provided by IoT and Cloud services providers.

The Edge computing layer can significantly contribute to the solutions of above mentioned challenges. To that end, new computing layers are developed in between of IoT and Cloud. While there is no any standard or formal definition of Edge computing (there is of course a common understanding of it as explained above), our model of Edge computing is based on that of [4]. Other definitions capture characteristics of Edge computing such as various computing granularities, notably, Fog computing, Cloudlets computing, Mobile Edge computing/Multi-access Edge computing (MEC) [2, 17].

The objective of Edge computing is to alleviate the computational burden of IoT caused to Cloud, by offloading data processing at the Edge devices, close to where the data is generated, instead of directly sending it to Cloud. The computing devices at this layer have larger computing and data storage capacities, when compared to IoT sensing devices. For instance, we could find in this layer devices like Raspberry Pi (either alone or grouped into virtual Edge nodes), embedded devices into vehicles, gateways, mini-clouds, cloudlets, MEC servers, micro-data centers, etc. Edge computing is currently propelled by 5G technologies [3, 7, 18], which improve substantially Quality of Service (QoS), reduce significantly latency (when compared to Cloud latency), provide support to mobility, scalability, real-time applications, etc.

3. IoT Stream Processing and Semantic Data Enrichment

One of the most prominent scenarios of offloading algorithms, from Cloud to Edge, is that of IoT data stream processing. Indeed, IoT is the real contributor to the exponential growth in data. In many applications based on IoT stream processing the aim is to just detect an "*event of interest*" (a pattern, an anomaly, a fault, a security breach, etc.) in real time. In such cases, data is processed but not stored (this kind of stream is also referred to as *trash data stream*). In some more advanced applications, especially those based on multi streams, there could be needed a complex event processing to correlate events from different streams and reasoning over complex events.

3.1. Semantic Data Enrichment

Semantic data enrichment is one of the plausible tasks to be offloaded to Edge computing. Indeed, IoT data comes in raw formats and usually contains redundant data, missing data elements, data errors, etc., due to possible faulty device or network behaviour. Data can therefore be pre-processed, cleaned and structured close to the data source where data is generated. Here, semantic data enrichment, that is, the process of adding new contextual information to original data using semantic format, is of special interest.

Semantic data enrichment can bring various advantages and benefits to IoT data processing, among which we could distinguish: (a) adding value to data by giving meaning to it; (b) enabling more efficient reasoning and mining; (c) linking contents from different IoT devices; and, (d) enabling complex event processing. In the first place, semantic data enrichment adds valuable information to the data such as contextual information of the form *where-when-who-how* on top of the original information of the event such as geo-location information, timestamp, actor (device) that originated the event and other potential related information. Secondly, semantically enriched data enables reasoning and mining more efficiently. Indeed, by using Resource Description Framework (RDF) the enriched data can be machine readable and interpretable enabling thus reasoning and rule-based engines to automatically run on data. Likewise, the use of meta-data facilitates mining of semantically enriched data. In the third place, through semantic data enrichment, it is possible to link contents coming from different data streams similarly as in the case of linked data and the semantic Web. Finally, semantic data enrichment makes possible data correlation and therefore complex event processing. Certainly, it is very difficult, if not impossible, to always correlate atomic events in an IoT data stream as events can happen independently of each other and at different times. It becomes therefore necessary to enrich data of atomic events so that by correlating them, more complex events can be defined and processed. Semantic IoT is currently an active research field inspired by semantic web technologies. However, unlike web resources description which largely considered data in a static context, it is very challenging to find a model for description and relationship of massive data sets (*aka* Big Data or Big Data stream). Nevertheless, semantic technologies have recently started to find their way into the IoT domain and addressing the inherent problems of interoperability and interpretation, time dimension, identification of data elements in a stream, etc. [1, 12].

3.2. Semantic data representation

Uniform Semantic Web data representations, such as Resource Description Framework (RDF), which can be unambiguously interpreted on the Internet, are appealing candidates for data transfer formats in IoT. Nevertheless, resource-constraints and latency requirements introduce challenges for implementing these technologies. RDF data can be expressed in various data formats for publishing and exchanging semantic data. RDF/XML, Turtle, and N-Triples are alternative W3C standard representations for RDF. Notation3 (N3) is another expressive format of RDF, which can also express rules with N3 Logic and RDF properties. All of these are based on the triple structure but differ in their expressive power. These RDF syntaxes are designed for Web applications (prior to IoT). JSON for Linked Data (JSON-LD), Entity Notation (EN) and Header-Dictionary-Triples (HDT) are more compact, lightweight representations for RDF. HDT is designed for compressed RDF data storage rather than a lightweight data exchange for IoT. Sensor Markup Language (SML) is a data format for representing sensor measurements and device parameters but is not based on RDF, although it has the necessary capabilities to annotate the type of data.

4. System Architecture

Our system architecture is divided into four layers: (1) IoT Data Sensing Layer; (2) Edge Processing Layer; (3) Data Analysis and Reasoning Layer and (4) User Application Layer (see Fig. 1).

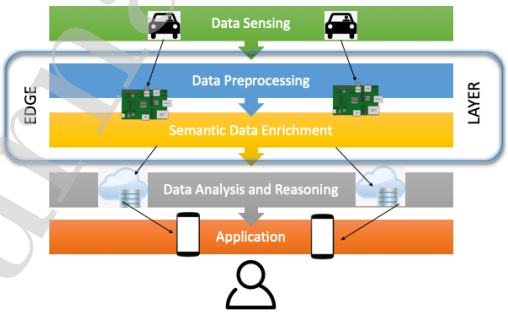

Figure 1: Architecture Overview

4.1. Data Sensing Layer

This is the lowest layer of the system architecture where the data is generated. To particularize the study with a real life application, we consider car sensory data, namely, the Controller Area Network (CAN bus) data that is coming from the car sensors in a road [8]. The car sensory data stream fits into characteristics of IoT data stream described earlier. This data stream could be useful for many purposes, in particular, it is useful for pothole detection in roads. Indeed, as part of CAN bus data, there is included also the speed of wheels. An anomaly in a wheel's speed can be observed when a car goes through a pothole, which causes an abrupt change in wheel's speed.

The generation layer is vital in determining the system's throughput characteristics, given that there is a data generation rate parameter at this layer. Different data rates have to be considered in experimental studies and real-life scenarios to ensure that the events are properly processed and not lost. In order to have the maximum benefit of the Edge computing characteristics and its processing capabilities, it is intended to keep the data sensing layer's capabilities at the lowest level. It means that only basic operations are applied in this layer corresponding to definition of which data will be sent, within which time frames, or by single or multiple measurements.

4.2. Edge Processing Layer

The Edge layer is the main distributed architecture component of the system. The data sensing layer is passing the data, which is coming from the car's sensors to this layer for pre-processing and semantic enrichment. The idea is to take advantage of the computing capacity of Edge devices to reduce the upper layer's processing amount, stress and time such as Clouds or Clusters. The Edge layer has two processing sub-layers, which have a hierarchy between them. The pre-processing always happens before the data enrichment, as it makes sense to enrich only over the qualified or appropriately selected data. Next, we examine the capabilities of each layer in more depth to further analyze the streaming processes.

Data Pre-processing

The pre-processing identifies blocks of events as units of processing (based on an atomic event definition or on a window processing). These fundamental units depend on the complexity and type of the collected data. For instance, redundant data elimination can be done before the data passes to the upper layers. If there is enough computing capacity in the existing infrastructure,

part of the pre-process can be handled in Edge devices, before sending them to any network. If network bandwidth is not a concern, or there is not enough computational power in the existing infrastructure, moving pre-processing to the Edge platform could be a good option in order to not increase the stress over the local devices. In this study, the pre-processing is handled inside the Edge device (RPi). Some data pre-processing options are considered in this layer, such as:

Event Detection: Any event stream processing assumes that some functions will be used to find the boundaries of atomic events in the stream. It should be noted that it is not always possible to precisely and efficiently compute such event boundaries. Window-based techniques are more general techniques for events identification otherwise.

Filtering: The classical filtering of the data can be applied based on one or more criteria. The context of the data can be selected manually, automatically or some of them can be discarded. The other filtering options are error detection and filtering out unclassified data for specific environments.

Data Reduction: This procedure aims to achieve a reduced representation of the data. In some cases, the amount of data can be huge (unfeasible to handle), therefore it is necessary to be able to reduce the utilized data without losing any of its accuracies or usefulness. It is also possible to reduce the intensity or data stream rate for the measurements. This might sound like reducing the data stream rate would automatically drop the data quality. However, in some cases, it might be useful to reduce the unnecessary amount of data if the high data stream rate is not the principal objective.

Anomaly Detection: Computation of anomaly scores is usually based on a statistical approach. Furthermore, this approach can be extended with the advances of machine learning algorithms that can state a prediction score to detect anomalies such as Hierarchical Temporal Memory (HTM) Algorithm to train the dataset [14, 5].

4.3. Data Analysis and Reasoning Layer

In recent years, a notable number of technologies that facilitate real-time and linked stream processing have also been released. Linked Stream Data is an annex of the SPARQL query language. It has a semantic engine to deal with stream sensor data and enrich them with the Linked Open Data cloud. C-SPARQL was an initial proposal of the streaming SPARQL system. A reasoning engine (a reasoner) is a software tool for reasoning with rules. Current reasoners can handle a broad set of RDF and Web Ontology Language

(OWL) vocabularies and most RDF data formats. A reasoner infers knowledge from semantic data and ontology-based on predefined rules. Common reasoning and inference engines such as the Jena Inference subsystem, Pellet, RacerPro, HermiT, RIF4J, and Fact++ support different rule languages, ontologies and OWL. IoT introduces additional challenges for reasoning; for example, reasoning can occur at any stage of the data delivery process, from the sensor node to back-end knowledge repositories. Distributing reasoning can therefore improve reasoning latency with large data sets.

5. Experimental Study and Evaluation

5.1. Deployment in Real Infrastructure

The real infrastructure for the deployment of the system is divided into three categories: Raspberry Pi integration, Ubuntu Machine in RDlab UPC to store and execute the server side scripts and IBM Node-Red server to visualize the incoming results on both desktop and mobile view.

```
@prefix ns1: <http://schema.org/> .
@prefix ns2: <http://www.w3.org/ns/sosa/> .
@prefix ns3: <http://qudt.org/schema/qudt#> .
@prefix ns4: <http://www.w3.org/ns/ssn/> .
@prefix ns5: <http://www.w3.org/ns/prov#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<edgeprocessor/27657/data/0> a rdf:Bag ;
    rdf:li <car/24898/measurement/0/value/0>,
        <car/24898/measurement/0/value/1>,
        <car/24898/measurement/0/value/2>,
        <car/24898/measurement/0/value/3> .

<car/24898/measurement/0/value/0> a ns1:Observation ;
    rdf:Property [ a ns2:PropertyValue ;
            ns2:name "AnomalyValue" ;
            ns2:value 7.3e-01 ] ;
    ns5:generatedAtTime "2.09" ;
    ns2:hasFeatureOfInterest <car/24898/wheel/0> ;
    ns2:hasResult [ ns3:Unit <http://qudt.org/vocab/unit#RevolutionPerMinute> ;
            ns3:numericValue 0e+00 ] ;
    ns2:madeBySensor <car/24898/wheel/0/rpmsensor> ;
    ns2:observedProperty <car/24898/wheel/0/speed> .

<car/24898/measurement/0/value/1> a ns1:Observation ;
    rdf:Property [ a ns2:PropertyValue ;
            ns2:name "AnomalyValue" ;
            ns2:value 1.6e-01 ] ;
    ns5:generatedAtTime "2.09" ;
    ns2:hasFeatureOfInterest <car/24898/wheel/1> ;
    ns2:hasResult [ ns3:Unit <http://qudt.org/vocab/unit#RevolutionPerMinute> ;
            ns3:numericValue 0e+00 ] ;
    ns2:madeBySensor <car/24898/wheel/1/rpmsensor> ;
    ns2:observedProperty <car/24898/wheel/1/speed> .

<car/24898/measurement/0/value/2> a ns1:Observation ;
```

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:ns1="http://schema.org/"
  xmlns:ns2="http://www.w3.org/ns/sosa/"
  xmlns:ns3="http://www.w3.org/ns/ssn/"
  xmlns:ns4="http://www.w3.org/ns/prov#"
  xmlns:ns5="http://qudt.org/schema/qudt#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
>
  <rdf:Description rdf:nodeID="N98d88ed698f845d4af6607e69add8948">
    <ns1:value rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.2</ns1:value>
    <rdf:type rdf:resource="http://schema.org/PropertyValue"/>
    <ns1:name>AnomalyValue</ns1:name>
  </rdf:Description>
  <rdf:Description rdf:about="car/12144/measurement/0/value/1">
    <ns2:hasResult rdf:nodeID="result1"/>
    <ns2:madeBySensor rdf:resource="car/12144/wheel/1/rpmsensor"/>
    <rdf:Property rdf:nodeID="N98d88ed698f845d4af6607e69add8948"/>
    <ns2:observedProperty rdf:resource="car/12144/wheel/1/speed"/>
    <rdf:type rdf:resource="http://www.w3.org/ns/sosa/Observation"/>
    <ns2:hasFeatureOfInterest rdf:resource="car/12144/wheel/1"/>
    <ns4:generatedAtTime rdf:datatype="http://www.w3.org/2001/XMLSchema#datetime">
      2019-06-08T18:38:08.957512</ns4:generatedAtTime>
  </rdf:Description>
  <rdf:Description rdf:about="car/12144/wheel/0/speed">
    <rdfs:label>the rotation speed of wheel 0 of car 12144</rdfs:label>
    <rdf:type rdf:resource="http://www.w3.org/ns/sosa/ObservableProperty"/>
  </rdf:Description>
```

Figure 2: Snippet of a Sample of Semantically Enriched Data in RPi –Serialized in Turtle Format (left) and XML Format (right)

Edge Platform Integration in Raspberry Pi. The integration of the Edge platform is handled by Raspberry Pi 3 model B+ as it is the current stable final release of the hardware. The new model has some advantages over the previous models in terms of quantities of the pre-installed software, 5GHz connectivity, Virtual Network Computing (VNC) to be able to access Raspberry Pi remotely with the Graphical User Interface (GUI). The Application Programming Interface (API) Server is a handler of the Server-Edge communications. The Message Queuing Telemetry Transport (MQTT) protocol has been integrated into Rasberry Pi to execute the Edge processing and sending data to the server via MQTT Broker.

As mentioned in the previous sections, the pre-processing procedure is responsible for retrieving and parsing the incoming CAN bus data files. In this case, instead of assuming the data is already at the Edge computing platform for processing, we implemented a simple File Transfer Protocol server to send the *CarData.csv* file to the Raspberry Pi. Fig. 2 shows snipets of a semantic annotation achieved by RPi serialized in two different formats (Turtle and XML, resp.).

Server Integration in RDLab at UPC. We used the RDLab infrastructure at the Technical University of Catalonia, which has over 160 physical servers, 1000 CPU cores, 3TB RAM and 10 Gbit high-speed network. The aim of using RDLab was to test the system behaviour within a larger distributed infrastructure. It means that each component of the system can be analyzed and monitored individually in terms of performance and reliability.

Application Integration in Node-Red. The final integration step was the implementation a Graphical User Interface (GUI) application for the software to enable the accessibility for the final users. While there are various options, in order to be more compatible with the tasks and requirements, Node-Red is chosen for the deployment of the application (see Fig. 3 (left)). Moreover, the Node-Red is supporting both desktop and mobile platforms.

Finally, the flows of the architecture that is shown above, can be visualized via built-in dashboard elements (see Fig. 3 (right)). There are many nodes in the Node-Red environment for creating dashboard elements, networking components, RPi and social media. There are plenty of functions to parse, modify, and interact with other nodes as well. The system is built entirely in NodeJS, hence, diversity can be increased by simply installing new additional nodes via the node package manager tool.

Figure 3: Node-Red Architecture (left); Node-Red User Interface (right)

5.2. Memory Usage Test

In this testing, each function related to the pre-processing and semantic enrichment has been analyzed (see Fig. 4) in the Edge Raspberry Pi, starting from connection establishment, processing the information, and the final delivery of the results. In the case of the semantic enrichment process, in order to get precise measurement results and comparisons, each type of syntax regarding RDF goes through the same process under the same conditions in the Edge unit by specifying the file format.

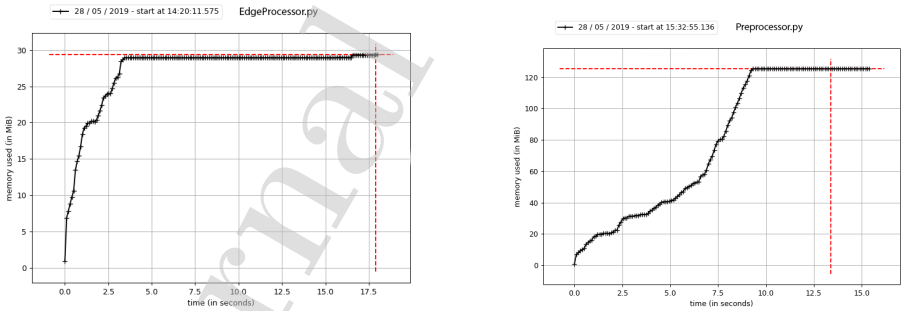

Figure 4: Memory Usage of Semantic Enrichment Unit in Turtle Format (left) and in Pre-processing Unit (right)

As we can be seen from Fig. 4, the memory usage of semantic enrichment is 30 MiB (1 MiB = 1048576 Mb) that is equal to approximately 3.3% of the total memory amount. The pre-processing usage is equal to 12% of the total amount. However, the memory usage depends on the volume of incoming

data. It should be noted that in case the memory cannot provide enough processing power for all the incoming data, more than one Raspberry Pi with parallel programming techniques should be considered.

5.3. Network Performance Test

The testing of network performance is divided into two cases. We intend to measure the latency and RTT–Round Trip Time of the data streaming process with different network configurations. Due to a lack of sim-card injection ability devices, the network performance is simulated by 2.4GHz and 5GHz Wi-Fi speed networks. Before testing, every possible resource demanding application such as online data streaming and downloads are eliminated in network connection to obtain more accurate results. The tests are obtained by processing 2000 CAN bus data entries and are sent over the network to the RDLab server. In Fig. 5 (left) the comparison between 5GHz and 2.4GHz speed performance can be seen, while in Fig. 5 (right) is shown how the latency between the RDlab server, Raspberry Pi, a local computer behaves. The rationale of doing this test is to have an idea of the Raspberry Pi's built-in 5GHz Wi-Fi hardware and software performance, compared to more advanced configurations.

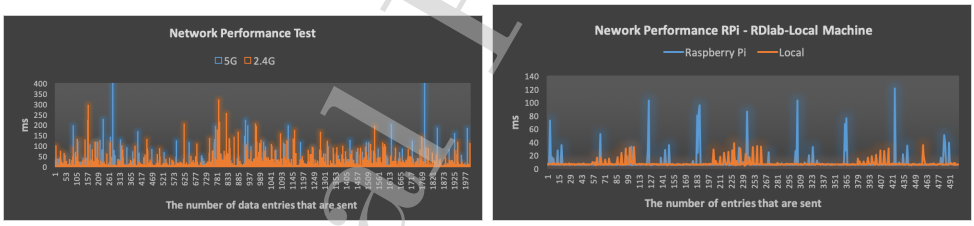

Figure 5: Latency Test with 5GHz vs 2.4GHz (left); Latency Test - 5GHz (right)

The final networking performance test refers to the Round Trip Time (RTT) between the Raspberry Pi and the RDlab Server with different Wi-Fi speeds. In Fig. 6 are shown the results of RTT measurements.

Despite their reputation as high-speed and accurate, 5GHz networks came out with unstable results as we can see in figures above. This may vary due to the established quality of 5GHz, the quality of the antennas and the distance and the infrastructure quality. During the tests for the 5GHz measurements, the average latency was observed between 6-18ms with an average of 16ms and only 300 data entries have been sent with less than 6ms.

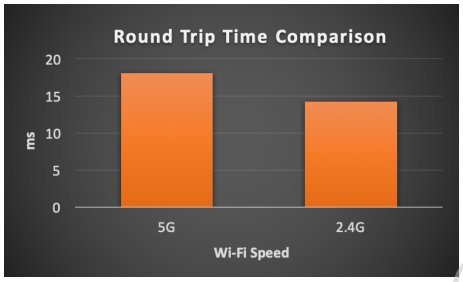

Figure 6: Round Trip Time Measurement - Raspberry Pi and RDLab

On the contrary, 2.4GHz provided quite acceptable results with this dataset. Most of the packets were sent between 5-14ms with an average of 12ms, and over 1000 data entries were sent with less than 6ms. However, it should be noted that for higher data streaming rates and resource demanding tasks, the bandwidth of the 2.4GHz may not be enough.

5.4. Data Stream Rate Test

Data streams might have dramatic spikes in data volume, such as high event rates during critical states in the areas where the traffic load is relatively higher. Usually, it is impractical to provide resources to handle the spike load adequately [10]. As accurate data stream processing is most crucial in such circumstances of high data load, our tests over the data stream rate aim to achieve high quality measurements when we increase the data streaming rate. The tests are covering the data streaming process through a CSV file by reading it with multiple lines per second, instead of just one as it is on a regular basis. The MQTT Broker was a connection provider to transfer all the information to the server-side. Hence, we would be able to see for how long the MQTT Broker and the Raspberry Pi hardware resources can hold its state as stable and working. In order to perform these tests, we have created multiple test scenarios with different streaming rates. The total amount of data entries were up to 2000. Each entry is representing a single **msg.payload** that is a size of 4000 bytes. The test case is created by reading 15, 20, 25, 30, 50, 100 entries per second that have been processed, and continuously sent from Raspberry Pi to the server.

According to our test results (see Fig. 7 (left)), the MQTT Broker cannot handle multiple entries simultaneously. The streaming up to 15 lines per second, was the most stable result without losing connection or data, despite

the execution time increase. After this threshold, the MQTT started to struggle to keep its state and failed at the end. However, with the appropriate data stream rate, there wasn't any data loss during the tests since MQTT operates on top of the TCP/IP protocol, and it guarantees the transmission of the packets. In Fig. 7, the performance of each streaming rate and the total amount of processed lines that are achieved by the MQTT Broker before the failure is shown. In Fig. 7 (right), the execution performances of each streaming rate are analyzed. The numbers are selected according to the lowest number of entries that are processed by the 100 lines/sec data rate streaming. As we can see from the graph, while the data streaming rate is increasing, Raspberry Pi is getting unstable. Also, it has crashed several times after 4-5 execution cycles due to workload and thermal problems that forced us to restart and cool down the machine.

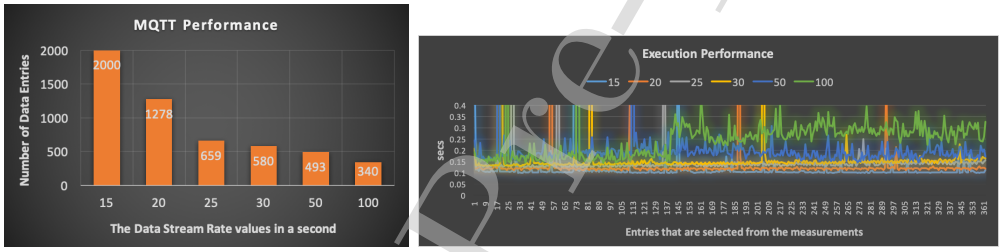

Figure 7: MQTT Performance Comparisons (left); Data Streaming Rate Execution Time Comparisons (right)

6. Conclusions and Future Work

In this paper we have discussed and analyzed the opportunities and challenges that bring the Edge computing in the Cloud-to-thing continuum to alleviate the computational burden of current Cloud computing systems, better support to efficient and scalable data processing, support to end user applications with low latency, mobility requirements, etc. We particularly emphasize the task of semantic data enrichment, which is a key enabler to further analysis and reasoning over IoT data streams. We have deployed a layered IoT-Edge-Cloud system in a real infrastructure by using car sensory data at IoT layer, RPi at Edge layer and Node-Red server at Cloud level. An extensive experimental study was then conducted to highlight the performance in terms of efficiency, memory usage, Round Trip Time with different

network configurations and different data stream rates. The results show the computational bounds while using a single RPi for processing data stream and prompting for a real compute fabric at Edge layer by using a number of RPi or other Edge devices.

In our future work we plan to extend the deployment of the system by adding a rule-based reasoning layer to detect complex events from semantically enriched data and support an Event-Condition-Action (ECA) system.

Acknowledgements

Fatos Xhafa's work is supported by Spanish Ministry of Science, Innovation and Universities, Programme "*Estancias de profesores e investigadores sénior en centros extranjeros, incluido el Programa Salvador de Madariaga 2019*", PRX19/00155.

References

[1] Anicic, D. et al. (2016) RDF stream processing: requirements and design principles. *RDF Stream Processing Community Group.*

[2] Beck, M. T., Werner, M., Feld, S., and Schimper, S. (2016). Mobile edge computing: A taxonomy. In *Proc. of the Sixth International Conference on Advances in Future Internet*, pp. 48–55.

[3] Farris, I., Orsino, A., Militano, L., Iera, A., and Araniti, G. (2018). Federated iot services leveraging 5g technologies at the edge. *Ad Hoc Networks*, 68:58–69.

[4] Hassan, N., Gillani, S., Ahmed, E., Yaqoob, I., Imran, M.(2018). The role of edge computing in internet of things. *IEEE Comm.*, 56(11):110–115.

[5] Hawkins, J., Lewis, M., Klukas, M., Purdy, S., and Ahmad, S. A framework for intelligence and cortical function based on grid cells in the neocortex. *Frontiers in Neural Circuits*, 12, 01 2019.

[6] Howard, P. N. (2015). *Pax Technica How the Internet of Things May Set Us Free or Lock Us Up.* Yale University Press.

[7] Hsieh, H., Chen, J., and Benslimane, A. (2018). 5g virtualized multi-access edge computing platform for iot applications. *J. Network and Computer Applications*, 115:94–102.

[8] T. Huybrechts, Y. Vanommeslaeghe, D. Blontrock, G. Van Barel, and P. Hellinckx, Automatic reverse engineering of can bus data using machine learning techniques, in *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, Springer, 2018, pp. 751–761.

[9] Ibbotson, J., Gibson, C., Wright, J. J., Waggett, P., Zerfos, P., Szymanski, B. K., and Thornley, D. J. (2010). Sensors as a service oriented architecture: Middleware for sensor networks. In *Sixth International Conference on Intelligent Environments, IE2010*, pp 209–214.

[10] Klein, A., Hackenbroich, G., and Lehner, W. How to screen a data stream - quality-driven load shedding in sensor data streams. *ICIQ2009*, pp. 76–90, 2009.

[11] Liu, D., Yan, Z., Ding, W., and Atiquzzaman, M. (2019). A survey on secure data analytics in edge computing. *IEEE Internet of Things*.

[12] Mauri, A., Calbimonte, J.-P., Dell'Aglio, D., Balduini, M., Brambilla, M., Della Valle, E. and Aberer, K. Triplewave: Spreading RDF streams on the web, in *The Semantic Web*, Springer, 2016, pp. 140–149.

[13] Noor, M. and Hassan, H. (2019). Current research on internet of things (iot) security: A survey. *Computer Networks*, 148:283–294.

[14] Numenta Inc 2011. Hierarchical Temporal Memory Including HTM Cortical Learning Algorithms. *White Paper*, page 4, 2011.

[15] Ritrovato, P., Xhafa, F., and Giordano, A. (2018). Edge and cluster computing as enabling infrastructure for internet of medical things. In *32nd IEEE AINA2018*, pp. 717–723.

[16] Roman, R., López, J., and Mambo, M. (2018). Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges. *Future Generation Comp. Syst.*, 78:680–698.

[17] Sánchez-Iborra, R., Sanchez-Gómez, J., and Skarmeta-Gómez, A. F. (2018). Evolving iot networks by the confluence of MEC and LP-WAN paradigms. *Future Generation Comp. Syst.*, 88:199–208.

[18] Shah, S. A. A., Ahmed, E., Imran, M., and Zeadally, S. (2018). 5G for vehicular communications. *IEEE Comm.*, 56(1):111–117.

Highlights

- Investigate challenges of processing an IoT data stream in a real edge computing layer.
- Analyze semantic data enrichment techniques
- Use real life data stream arising from a car data stream for semantic data enrichment and anomaly detection
- Use a real infrastructure using Raspberry Pi and Node-Red server for deploying the system
- Highlight the complexities of achieving real time requirements of applications based on IoT stream processing.

Fatos Xhafa received his PhD in Computer Science in 1998 from the Department of Computer Science of the Technical University of Catalonia (UPC), Barcelona, Spain. Currently, he holds a permanent position of *Professor Titular* at UPC, BarcelonaTech. He was a Visiting Professor at Birkbeck College, University of London (UK) during academic year 2009-2010 and Research Associate at Drexel University, Philadelphia (USA) during academic term 2004/2005. Prof. Xhafa has widely published in peer reviewed international journals, conferences/workshops, book chapters and edited books and proceedings in the field (http://dblp.uni-trier.de/pers/hd/x/Xhafa:Fatos). He is awarded teaching and research merits by Spanish Ministry of Science and Education, by IEEE conference and best paper awards. Prof. Xhafa has an extensive editorial and reviewing service. He is editor in Chief of *International Journal of Grid and Utility Computing* and *International Journal of Space-based and Situated Computing* from Inderscience and member of EB of several International Journals and Guest Editors of Special Issues. He is Editor in Chief of the Elsevier Book Series "*Intelligent Data-Centric Systems*" is actively participating in the organization of several international conferences and workshops. He is a member of IEEE Communications Society, IEEE Systems, Man & Cybernetics Society and Emerging Technical Subcomm. of Internet of Things.

His research interests include parallel and distributed algorithms, massive data processing and collective intelligence, optimization, networking, P2P and Cloud computing, security and trustworthy computing, machine learning and data mining, among others. He can be reached at fatos@cs.upc.edu and more information can be found at http://www.cs.upc.edu/~fatos/

Future Generation Computer Systems

Submission Cover Letter

Manuscript title: Evaluation of IoT Stream Processing at Edge Computing Layer for Semantic Data Enrichment

Authors: Fatos Xhafa, Burak Kilic, Paul Krause

Submission to Special Issue:

Dear Guest Editors of the Special Issue,

We would like to submit the manuscript "*Evaluation of IoT Stream Processing at Edge Computing Layer for Semantic Data Enrichment*" authored by Fatos Xhafa, Burak Kilic, Paul Krause to your special issue.

We affirm that the submission represents original work that has not been published previously and is not currently being considered by another journal.

We declare that there are no conflicts of interest.

Corresponding author

Name: Fatos Xhafa

Affiliation: Technical University of Catalonia, Barcelona, Spain

E-mail address: fatos@cs.upc.edu

Submission date: 9th August 2019