A Preliminary Fuzzy Markup Language based Approach for the Queue Buffer Size Optimization in Fog Nodes for Stream Processing

Gregorio Corpas-Prieto
Dpt. Electr. & Computer Engineering
University of Córdoba, Spain
email: gcorpas@uco.es

Fernando Leon-Garcia
Dpt. Electr. & Computer Engineering
University of Córdoba, Spain
email: fernando.leon@uco.es

Juan Carlos Gámez-Granados
Dpt. Electr. & Computer Engineering
University of Córdoba, Spain
email: jcgamez@uco.es

José Manuel Palomares
Dpt. Electr. & Computer Engineering
University of Córdoba, Spain
email: jmpalomares@uco.es

Joaquín Olivares
Dpt. Electr. & Computer Engineering
University of Córdoba, Spain
email: olivares@uco.es

Jose Manuel Soto-Hidalgo
Dpt. Comp. Architecture & Technology
University of Granada, Spain
email: jmsoto@ugr.es

2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) | 978-1-6654-6710-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/FUZZ-IEEE55066.2022.9882741

***Abstract*—The Internet of Things (IoT) is usually divided in three layers: *Edge*, *Fog*, and *Cloud* layers. The whole IoT infrastructure deals with large amount of data between layers. Focusing on the Fog layer, the sending/receiving data and further cascade processing of those data in the Fog layer enable the *Stream Processing* paradigm. Thus, aspects such as the number of connections, delays, buffer size, memory usage, among others, have to be considered to optimize the network traffic. Moreover, these characteristics are affected by uncertainty and imprecision since, for example, the number of connections or the buffer size may be considered low in some cases and high in others. Fuzzy Rule-Based Systems (FRBS) are suitable for addressing complex data and managing their imprecision. The objective of this paper is to propose an approach that optimizes network traffic with the main goal of dynamically and automatically adjusting the queue buffer size in a node to avoid network collapse. The IEEE std 1855-2016 for Fuzzy Markup Language and the open source library JFML are used for their flexibility and interoperability offered by these technologies. The proposal has been simulated in three basic different scenarios involving several network traffic states in a fog infrastructure.**

I. Introduction

Internet of Things (IoT) has enabled different devices to share heterogeneous data between them without human actions [1]. The architecture of IoT is usually considered to be 3 layer approach: edge, fog and cloud. The edge layer represents the things where heterogeneous and massively distributed devices are present. The fog layer involves deploying computing services at the edge of the network in order to improve the user experience and service resilience in the event of failure [2], and finally, the cloud layer encompasses the powerful Internet servers.

Thanks to this architecture, large amounts of data are sensed anywhere, anyplace, and anytime by means of IoT over the different involved layers (edge, fog and cloud). In consequence, the volume and type of data as network traffic is increased. Focusing on data sending/reception between nodes in the edge/fog layer, optimising the use of the network resources has been a subject of scientific interest [3]–[7], for which several aspects have to be considered due to the heterogeneity [8].

In this context, the stream processing concept plays an important role due to it takes action on a series of data at the time the data is created. Stream processing often entails multiple tasks on the incoming series of data (the "data stream"), which can be performed serially, in parallel, or both. This workflow is referred to as a stream processing pipeline, which includes the generation of the stream data, the processing of the data, and the delivery of the data to a final location [9]. When resources at the Edge are used, data is channelled from the sources to the Cloud through chained processing stages. These stages are distributed among the available nodes, emerging networked pipeline structures [10]. The linear logic of the pipelines, in which each node is a consumer of its predecessor and a producer of its successor, gives rise to congestion dynamics that inherit this linearity, in terms of bottlenecks and chain delays [11]. As a consequence, in this process, several characteristics directly related to the network nodes have to be taken into account for the stream processing due to when a node receives more data than it can handle, congestion occurs. For example, the number of connections, delays, buffer size, memory usage, etc. are aspects that are not the same in each node and each time. Due to the linear nature of the pipeline structure, where network traffic is cascaded, the size of the receiving buffers of the nodes is a particularly relevant parameter in flow control and decoupling dynamics. Moreover, these characteristics are affected by uncertainty and imprecision since, for example, the number of connections or the buffer size may be considered small in some cases and big in others.

On the other hand, Fuzzy Logic Systems (FLS), which are rule-based expert systems based on fuzzy set theory to

978-1-6654-6710-0/22/$31.00 ©2022 IEEE

represent the semantics of rules and to process inference when input data are provided [12], [13], have been successfully used in a wide range of real-world problems. Concretely, some traffic network approaches based on Fuzzy Systems were found in previous literature [14]–[16]. In addition, FLS have also been considered in stream processing architectures to define expert systems to take data streams in input and which outputs decisions on the fly [17]. FLS can include a priori expert knowledge and represent systems for which it is not possible to obtain a mathematical model. In this context, the IEEE Computational Intelligence Society (IEEE-CIS) has sponsored the publication of the new standard for FLS (IEEE Std 1855-2016). This standard was established with the main objective of providing the fuzzy community with a unique and well-defined tool allowing a system to be designed completely independently from the specific hardware/software. The new standard defines a new W3C eXtensible Markup Language (XML) based language, named Fuzzy Markup Language (FML) [18] aimed at providing a unified and well-defined representation of interoperable and interpretable FLS. Additionally, in order to make the IEEE standard operative and useful for the fuzzy community, the library JFML [19] has been developed to offer a complete implementation of the new IEEE standard as well as a module to integrate IoT solutions into JFML [20]. Some developments based on JFML have also been successfully done in different areas such as controller for embedded systems [21], fall from height [22] or risk of falling object accidents in the construction area [23], among others.

Therefore, the integration of FLS and FML technologies in the management of the traffic network in stream processing poses an interesting challenge that can provide significant benefits for the network traffic in a Fog infrastructure. These aspects have motivated the present work by the idea of defining a preliminary approach which is able to manage the network traffic with the main goal of dynamically and automatically adjusting the buffer size on a node to avoid network collapses. We propose to integrate support expert's decision making in the development of a smart node to manage stream data. To do this, firstly, a panel of experts defines a set of rules for the management of the queue buffer size, secondly, we represent the FLS according to the IEEE std 1855-2016 where input variables will be associated with the previously mentioned variables (number of connections, time of delay, buffer size, etc) and a set of fuzzy rules will be considered to model five different buffer sizes: very small, small, medium, big and very big. This solution will be implemented in nodes that will integrate JFML and the defined FLS according to FML to automatically adjust the node buffer size.

This paper is organized as follows. Section II provides some preliminary concepts related to the Internet of Things, stream processing and the JFML library. Section III describes the approach for the optimization of the network traffic in stream processing based on JFML. Section IV presents a study case to show the applicability of the proposal in a simulation for different scenarios. Finally, the main conclusions are highlighted in Section V.

II. Preliminaries

In this section, firstly a brief introduction to the Internet of Things and Stream Processing is presented. Secondly, some concepts on network congestion in chained stages are also shown. Finally, an overview of the main concepts related to the IEEE std 1855-2016 and JFML are exposed.

A. Internet of Things and Stream Processing

Internet of Things (IoT) is a disruptive technology characterized by the interconnection on a global scale of a dynamic infrastructure of intelligent and self-configuring nodes (things). IoT enables ubiquitous and pervasive computing scenarios across heterogeneous devices with limited processing and storage capacity [24]. The architecture of IoT is usually considered to be 3-layer approach, called Perception, Network, and Application layers. Each one is characterized by a framework of supporting technologies and devices. Perception layer is also known as the sensor layer, is implemented as the bottom layer in IoT architecture. The perception layer interacts with physical devices and components through smart devices, and it involves technologies such as RFID and (Wireless) sensor networks (WSN). The network layer interconnects network elements and provides Internet access. Finally, the application layer guarantees confidentiality, integrity, and authenticity to the data [25]. However, a different architecture emerges from the challenge of integrating Cloud into the IoT universe. In this architecture, the global infrastructure is divided into three layers called Edge, Fog, and Cloud. The edge layer represents the things: heterogeneous and massively distributed devices. At the other extreme is Cloud, which encompasses the powerful Internet servers. At the center of this scale is Fog, a concept that aims to bring processing as close as possible to IoT data generators and actuators [26]. The development of the fog layer involves deploying computing services at the edge of the network in order to improve the user experience and service resilience in the event of failure. With the advantage of distributed architecture and close to end-users, Fog Computing can provide faster response and higher quality of service for IoT applications [27].

On the other hand, Big Data uses billions of data acquired usually from IoT to obtain useful knowledge to provide enhanced decisions. Big Data is mainly carried out using Cloud Computing technologies to centralize a wide variety of services on powerful Internet servers. In this sense, IoT and Cloud, opposite in terms of computing capacity, heterogeneity, and distribution; complement each other in the vision of the technological paradigm that is being developed. In this context, large data flows must be processed constantly to obtain results with very low latency and high performance per data. To achieve this, computational processes have been subdivided into consecutive stages in order to achieve segmented processing, similar to that of pipelines. This methodology, called Data Stream Processing, started in large Cloud processing centers as a computational process [28]. However, due to the expansion of the Internet of Things (IoT), the processing has been approaching the Edge level, the level of data sources

or generators, deriving in what is known as Near Edge Computing. In this case, the Stream Processing methodology has also moved towards the Edge level, at the so-called Fog level. This approach allows lower latencies, being closer to the data sources, affecting the performance of the network due to a large amount of data to be transmitted [29].

B. Network congestion dynamics in chained stages

In networks, when a node or link receives more data than it can handle, congestion occurs. It can be defined as a circumstantial degradation of quality of service (QoS) (in one of its many aspects) and can present different levels of severity. Both the shared medium and the node processors are limited resources, so congestion is not something that can be avoided when service demand exceeds certain limits. However, the dynamics associated with congestion can be analysed to act accordingly, for example: to provide extra resources to avoid congestion; to manage a controlled degradation of service; or to redistribute network workload [30].

Stream processing is a distributed computing paradigm that addresses the challenge of processing data in the form of endless streams. At the edge, data is channelled from the sources to the Cloud through chained processing stages which are distributed among the available nodes, emerging networked pipeline structures [10]. In this scenario, the buffers associated with the incoming traffic in each socket of each node of the pipeline play a dual role: in addition to their function in the communication protocols, they are decoupling elements between stages, providing a temporary storage with certain plasticity to mitigate the problems associated with desynchronization and processing queues.

In relation to communication, when this buffer is too large, the flow control mechanism is avoided, in the same way that an oversized fuse would avoid electrical safety in a device. This action is positive only if the network is so congested that flow control is counterproductive. However, if this action is not justified, an oversized buffer can lead to *bufferbloat* effects [31], [32]. On the other hand, as a decoupling element, the buffer size should be chosen according to the CPU and storage load, as a temporary processing queue memory.

As it can be appreciated, the problem of appropriate buffer size assignment involves a trade-off between several factors, suggesting the potential advantage of using expert systems that can effectively manage the imprecision of the magnitudes involved in this logic with some knowledge based on experience.

C. IEEE std 1855-2016 and JFML

The IEEE std 1855-2016 [33] proposed the syntax of a new language called Fuzzy Markup Language (FML). FML was created to represent Fuzzy Logic Systems (FLS), and specifically, Fuzzy Rule Based Systems (FRBS) in a human-understandable language. This is possible because FML shares the syntax with Extensible Markup Language (XML), another famous human-readable language. Furthermore, FML provides components to use various inference methods such as AnYa [34], or well-known methods such as Tsukamoto and Takagi-Sugeno-Kang (TSK) [35] and Mamdani [36]. In addition to these features, FML improves the IEC 61131-7 FCL [37] solving the limitation of proprietary formats and making FLS or FRBS usable in other systems, such as embedded systems.

However, the IEEE std 1855-2016 only introduced the FML syntax and its elements, but there was no available a library to design the FLS. For this reason, in 2018 an open-source library was published to design FLS according to IEEE std 1855-2016 under the name of JFML [19].

JFML is a library implemented in Java, although it is also available in Python 3.x through Py4JFML [38]. JFML provides not only an implementation of IEEE std 1855-2016, but also modules to import and export the generated FLS to other widely used formats such as Predictive Model Markup Language (PMML) or the format used by Matlab Fuzzy Logic Toolbox among others. As this library is based on the FML language, the five main components that define an FLS are also present: fuzzy knowledge base; fuzzy rule base; inference technique; fuzzification subsystem; defuzzification subsystem. Also, due to the library's package scheme, it is possible to update the library's functionality with the introduction of new modules.

III. Node Design

In general, when a message arrives at a node, the thread associated with the network card buffer wakes up and automatically queues that message in the decoupling buffer in the nodes. Starting from this idea, a general design scheme for the nodes to solve the problem of queue buffer size in a Fog network is presented. The main elements for each node included in this design are sensors/actuators, JFML instance, and FML files representing expert knowledge for the optimization in stream processing according to the IEEE std 1855-2016. A graphical representation of the node design and fog architecture is shown in Fig. 1.

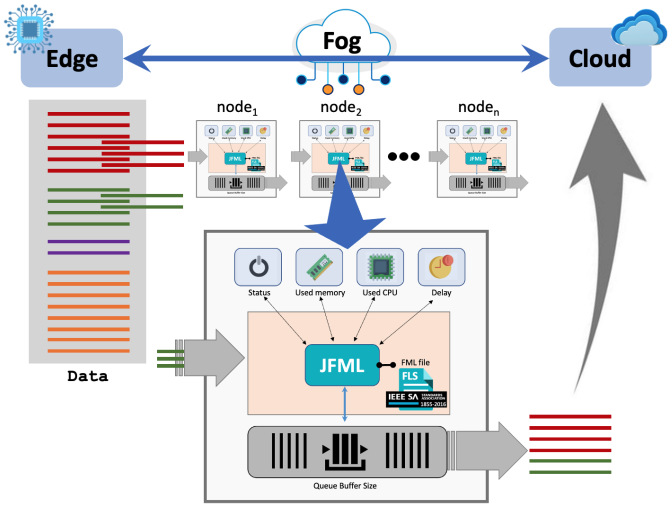

Fig. 1. General design for the nodes in a fog architecture

The main elements for each node are described in the following:

- $Status$: This element represents a sensor for reading the status of the node: active or inactive.
- $Used\ Memory$: This element represents the used memory percentage for the node. For example, 10%, 25%, etc.
- $Used\ CPU$: This element is in charge of managing the node CPU usage. For example, 70%, 90%, etc.
- $Delay$: This element represents the delay in the queue of the node. For example, $10ms$, $60ms$, $1s$, etc.
- $Queue\ Buffer\ Size$: This element represents the queue buffer size for the node. For example, $50KB$, $20MB$, etc.
- $JFML$: this element represents devices capable of using the JFML library. For example, embedded systems, a computer, or a server in the cloud where JFML can be run.
- FML: this element contains a repository of FML files where the knowledge base, the rule base and the fuzzy inference model according to the IEEE std 1855-2016 are coded.

A. Integration with JFML

In general, input/output elements or sensors/actuators provide data which are used for the JFML to make the inference according to the expert knowledge represented in FML files. This architecture allows communication between all the ubiquitous elements thanks to the JFML capabilities. The behavior of this communication procedure and the integration with JFML is summarized below:

1) Input elements provide data. In consequence, they must be associated to input variables. For example, the element for reading the $Status$ provides 0 for inactive and 1 for active status or the $Used\ Memory$ provides values in the interval $[0, 100]$.
2) JFML receives input data from reading elements and it assigns them to the input variables which are defined in the FLS (represented in the FML files according to the IEEE std 1855-2016). For example, JFML receives data from the elements $Status$, $Delay$, etc. and they are associated with the corresponding input variables.
3) When all reading elements have been provided their data and JFML has assigned these values to the input variables, the inference is carried out. Rules are fired according to the input values and the rule base defined in the FML files.
4) Once the inference process is finished, the output variables obtain values from the corresponding defuzzification method. Then, JFML writes these values to the corresponding output elements. For example, the value of the output variable *Buffer Size* is considered to adjust the buffer size of the node.

IV. Study Case

In order to illustrate the suitability of this proposal in the optimization of the queue buffer size in a node, a basic case study is developed. Specifically, a case focused on several network traffic situations where both the expert knowledge in network traffic monitoring and the fog node design are considered. A Raspberry PI Zero 2W, with 512MB RAM LPDDR2 and a queue buffer with 64MB are used. Notice that the SO requires about 200MB, resulting in 312 MB of available RAM for user processes. The proposal does not require high computational complexity, allowing it to run on this device without impacting the overall performance of the device. In this case, a queue buffer of 64MB would represent just over 20% of the available RAM. In this sense, the use of so much RAM for the buffer would have a very high impact on the management of the node itself, limiting the possibilities of effective use of it, and at the same time, showing an interesting example to illustrate the applicability of this proposal.

In subsection IV-A, the FLS taking into account expert knowledge is defined while in subsection IV-D some results are also shown.

A. Defining the Fuzzy Logic System

Several network traffic situations related to fog traffic can be considered by using the proposed node design. These situations are involved to gradual concepts. For example, if the memory usage is full and the CPU is very busy the buffer size should be big to avoid delays. If the delay is low or the CPU usage is idle, the buffer size could be little, while if the delay is high the buffer size should increase. Hence, to represent this expert knowledge the Fuzzy Logic System (FLS) is defined. We propose a methodology for defining this FLS. Fig. 2 illustrates the flowchart of the proposed methodology.

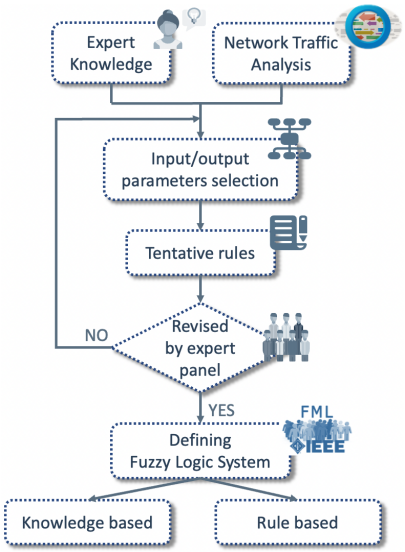

Fig. 2. Workflow for defining a FLS

Firstly, expert knowledge on network management in fog infrastructures is considered in order to select variables to support the problem at hand. Based on the information gathered, tentative rules are proposed and reviewed by a panel of experts in this field of knowledge (5 persons with high knowledge

on network monitoring and fog node design). If the experts do not validate the rules, the process is initiated in order to propose new variables and rules. However, if the proposal is validated, a fuzzy logic system is defined. For this purpose, firstly, the knowledge base is defined with several fuzzy variables and their fuzzy terms (subsection IV-A1)). Secondly, the rule base is also defined considering the expert knowledge and the relationships between the fuzzy variables (subsection IV-B). Finally, a representation of the FLS according to the IEEE 1855-2016 standard for the Fuzzy Markup Language is detailed (subsection IV-C).

1) Defining the Knowledge Base: Several fuzzy variables which are related with some elements in the node are defined. Concretely, four input and one output variables are considered:

- *Status* represents the status of the node as active or inactive. It is an input variable defined by the singletons: *"On"*, *"Off"*.
- *Memory* represents the memory usage of the node. It is an input variable composed by the fuzzy terms *"Empty"*, *"Normal"* and *"Full"* in the domain $[0, 100]$ represented in percentage for the total memory capacity.
- *CPU* indicates the CPU usage of the node. It is an input variable defined by the fuzzy terms *"Idle"*, *"Normal"*, *"Busy"* or *"Very busy"* in the domain $[0, 100]$ represented in percentage for the total CPU power.
- *Delay* represents the delay for processing the batch in the node. It is an input variable defined by the fuzzy terms *"Low"*, *"Medium"* or *"High"* in the domain $[0, 10]$ represented in ms.
- *Buffer size* represents the queue buffer size of the node. It is an output variable defined by the fuzzy terms *"Very small"*, *"Small"*, *"Medium"*, *"Big"* and *"Very Big"* in the domain $[0, 64]$ represented in MB.

B. Defining the Rule Base

Expert knowledge from the traffic network in a fog scenario in the form of rules is considered in order to determine the buffer size of the nodes. Specifically, 9 fuzzy rules have been defined:

1) *IF Status IS Off THEN Buffer IS Very small*
2) *IF Status IS On AND Memory IS Empty AND CPU IS Idle AND Delay IS Low THEN Buffer IS Very small*
3) *IF Status IS On AND Memory IS Empty AND CPU IS Normal AND Delay IS Low THEN Buffer IS Small*
4) *IF Status IS On AND Memory IS Empty AND Delay IS Medium THEN Buffer IS Medium*
5) *IF Status IS On AND Memory IS Normal AND CPU IS Normal THEN Buffer IS Medium*
6) *IF Status IS On AND Delay IS Medium AND CPU IS Busy THEN Buffer IS Big*
7) *IF Status IS On AND Delay IS High THEN Buffer IS Big*
8) *IF Status IS On AND Memory IS Full AND CPU IS Very busy AND Delay IS Medium THEN Buffer IS Big*
9) *IF Status IS On AND Memory IS Full AND CPU IS Very busy AND Delay IS High THEN Buffer IS Very big*

C. Fuzzy Logic System according to the IEEE 1855-2016

All the previous rules and fuzzy variables have been represented in a FML file according to the IEEE std 1855-2016 specifications. As illustrative example, some parts of the file are shown in the code 1 although the complete FML file can be downloaded from the JFML official website.

```
<fuzzySystem xmlns="http://www.ieee1855.org" name="
    Buffer Size optimizacion">
  <knowledgeBase>
    <fuzzyVariable name="Status" domainleft="-1.0
        " domainright="10.0" type="input">
      <fuzzyTerm name="On">
        <singletonShape param1="1.0"/>
      </fuzzyTerm>
      <fuzzyTerm name="Off">
        <singletonShape param1="0.0"/>
      </fuzzyTerm>
    </fuzzyVariable>
    ....
    <fuzzyVariable name="Buffer" domainleft="0.0"
        domainright="64.0" type="output"
       accumulation="MAX" defuzzifier="COG"
       defaultValue="0.0">
      <fuzzyTerm name="Very_small">
        <triangularShape param1="0.0" param2="
            0.0" param3="16.0"/>
      </fuzzyTerm>
      <fuzzyTerm name="Small">
        <triangularShape param1="0.0" param2="
            16.0" param3="32.0"/>
      </fuzzyTerm>
      <fuzzyTerm name="Medium">
        <triangularShape param1="16.0" param2="
            32.0" param3="48.0"/>
      </fuzzyTerm>
      <fuzzyTerm name="Big">
        <triangularShape param1="32.0" param2="
            48.0" param3="64.0"/>
      </fuzzyTerm>
      <fuzzyTerm name="Very_big">
        <rightLinearShape param1="48.0" param2=
            "64.0"/>
      </fuzzyTerm>
    </fuzzyVariable>
  </knowledgeBase>
  <mamdaniRuleBase name="rulebase1"
      activationMethod="MIN" andMethod="MIN"
      orMethod="MAX">
    <rule name="rule1" andMethod="MIN" connector=
        "and" weight="1.0">
      <antecedent>
        <clause>
          <variable>Status</variable>
          <term>Off</term>
        </clause>
      </antecedent>
      <consequent>
        <then>
          <clause>
            <variable>Buffer</variable>
            <term>Very_small</term>
          </clause>
        </then>
      </consequent>
    </rule>
    ...
  </mamdaniRuleBase>
</fuzzySystem>
```

Code 1. Some part of the FML file designed for the queue buffer size

D. Some illustrative results

In this subsection, some examples are detailed as a simulation of real situations in an edge-to-cloud data stream processing scenario involving the designed fog nodes by using our proposal. In these examples, node receives data from stream processing and depending on the node status (CPU, memory, delay, etc.), the queue buffer size is adjusted according to the output values provided by both the JFML and the FLS defined in the previous section. Each result is validated by the expert panel that defined the FLS. This validation was carried out with experts, because for this example it was not appropriate to include quantitative evaluation metrics, as the results coincided with the opinion of the expert panel.

1) Case 1: Low network traffic - idle nodes : In this example, the memory and the CPU usage are low and the delay is medium. On this situation, the queue buffer size should be small.

```
RESULTS
(INPUT): Status=1.0, Memory=10.0, CPU=30.0, Delay
    =11.0
(OUTPUT): Buffer=19.860588
(ACTIVATED RULES):
  RULE 3: rule3 - (0.8) IF Status IS On AND Memory
       IS Empty AND CPU IS Normal AND Delay IS Low
       THEN Buffer IS Small [weight=1.0]
 RULE 4: rule4 - (0.2) IF Status IS On AND Memory
     IS Empty AND Delay IS Medium THEN Buffer IS
     Medium [weight=1.0]
```

Case 1. Node is On, Memory usage is 10%, CPU usage is 30% and Delay is 11 ms

In this case, two rules have been fired. Rule 3 and Rule 4 with 0.8 and 0.2 respectively. The buffer size can be adjusted to 19.860588. This would result in an increase of the memory usage of just 16.4%. This buffer size is able to include the required amount of data to maintain the process throughput without increasing the delay of associated to the internal processing. This buffer size is able to keep the delay below 11 ms with no increase in the CPU usage and very limited impact on the memory occupancy.

2) Case 2: Moderate network traffic - moderately idle nodes: In this example, the memory and the CPU usage are low and the delay is medium. On this situation, the queue buffer size should be medium.

```
RESULTS
(INPUT): Status=1.0, Memory=50.0, CPU=50.0, Delay
    =10.0
(OUTPUT): Buffer=31.999846
(ACTIVATED RULES):
  RULE 5: rule5 - (1.0) IF Status IS On AND Memory
       IS Normal AND CPU IS Normal THEN Buffer IS
      Medium [weight=1.0]
```

Case 2. Node is On, Memory and CPU usage are 50% and Delay is 10 ms

In this case, the rule 5 has been fired corresponding to a medium situation. Then amount of buffer space can be fixed to 32 MB. This scales up the memory usage to 60.3% which is quite notable. The CPU usage will be increased to handle such amount of memory, however, the amount of free memory is still large enough not to constraint the computing process and therefore, the delay is kept below 10 ms.

3) Case 3: High network traffic - overloaded nodes : In this example, the memory and the CPU usage are high but the delay is medium. On this situation, the queue buffer size should be big to avoid collapse.

```
RESULTS
(INPUT): Status=1.0, Memory=80.0, CPU=95.0, Delay
    =45.0
(OUTPUT): Buffer=48.591198
(ACTIVATED RULES):
  RULE 7: rule7 - (0.25) IF Status IS On AND Delay
       IS High THEN Buffer IS Big [weight=1.0]
 RULE 8: rule8 - (0.5) IF Status IS On AND Memory
     IS Full AND CPU IS Very_busy AND Delay IS
     Medium THEN Buffer IS Big [weight=1.0]
 RULE 9: rule9 - (0.25) IF Status IS On AND Memory
      IS Full AND CPU IS Very_busy AND Delay IS
     High THEN Buffer IS Very_big [weight=1.0]
```

Case 3. Node is On, Memory usage is 10%, CPU usage is 95% and Delay is 45 ms

In this case, three rules have been fired. Rule 7 and Rule 9 with 0.25 and Rule 8 with 0.5. The buffer size can be adjusted to 48.591198 MB. Such buffer size consumes up to 95.5% of the memory. The CPU increases the usage to 100% to be able to work with that limited amount of free memory for data processing. However, the increase in delay due to the rise in the processing time is hidden by the use of such a large buffer size.

V. Conclusions

In this paper, a preliminary approach to dynamically and automatically adjust the queue buffer size in a node for stream processing in fog computing is proposed. This approach presents a node design that integrates a JFML instance and a FML file representing expert knowledge as a Fuzzy Logic System (FLS) according to the IEEE std 1855-2016 for the queue buffer-size problem in network traffic. This integration provides an intelligent environment configured by "smart nodes" in the fog layer responding in real-time to adjust the queue buffer size of nodes for stream processing without human intervention. In this way, aspects such as the number of connections, delays, memory and CPU usage, etc. are taken into account so that the node behaves as a human would. In order to showcase the utility of the proposal, a case study has been carried out. Concretely, a case focused on several network traffic situations where expert knowledge in the network traffic monitoring and fog node design are considered. A methodology for defining the FLS is also proposed. Finally, 3 different scenarios are simulated where edge-to-cloud data stream processing are considered. One scenario considers low network traffic with idle nodes. Another scenario contemplates moderate network traffic and the last one high network traffic with overloaded nodes. The results are in accordance with the expert panel assessment. Moreover, the simulated system shows a clear improvement in the overall performance of Stream Processing by keeping all data packets undropped with the maximum possible throughput.

It is worth mentioning that the system is a non–linear complex problem, in which in some cases, if one of the output

variables is maximized, another one could become worse. For instance, if the network is highly saturated, the *buffer size* may grow larger, in order not to drop any packet from the network. However, that would decrease the amount of memory in the nodes, and that would make each node to increase the CPU usage to deal with that low–memory environment. Therefore, it is not a trivial situation, that could be managed with other simpler mechanisms.

ACKNOWLEDGEMENTS

This research has been partially supported by the Spanish Ministry of Science, Innovation and Universities and the European Regional Development Fund - ERDF under the projects RTI2018-098371-B-I00 *Intelligent distributed processing architectures in Fog level for the IoT paradigm (Smart-Fog)* and PGC2018-096156-B-I00 *Recuperación y Descripción de Imágenes mediante Lenguaje Natural usando Técnicas de Aprendizaje Profundo y Computación Flexible*, and by the ERDF/Regional Government of Andalusia (1380974-F) *Fog Computing para mejorar la navegación de robots asistenciales (FOCRA)*.

REFERENCES

[1] E. Siow, T. Tiropanis, and W. Hall, "Analytics for the internet of things: A survey," *ACM computing surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[2] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, 2018.

[3] G. Campobello, A. Segreto, and S. Serrano, "Data Gathering Techniques for Wireless Sensor Networks: A Comparison," *International Journal of Distributed Sensor Networks*, vol. 2016, 2016.

[4] T. Wimalajeewa and P. K. Varshney, "Application of Compressive Sensing Techniques in Distributed Sensor Networks: A Survey," pp. 1–32, 2017. [Online]. Available: http://arxiv.org/abs/1709.10401

[5] F. Leon-Garcia, J. Palomares, and J. Olivares, "D2R-TED: Data—Domain Reduction Model for Threshold-Based Event Detection in Sensor Networks," *Sensors*, vol. 18, no. 11, p. 3806, nov 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/11/3806

[6] Z. Gao, L. Dai, S. Han, I. Chih-Lin, Z. Wang, and L. Hanzo, "Compressive Sensing Techniques for Next-Generation Wireless Communications," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 144–153, 2018.

[7] F. Leon-Garcia, F. J. Rodriguez-Lozano, J. Olivares, and J. M. Palomares, "Data Communication Optimization for the Evaluation of Multivariate Conditions in Distributed Scenarios," *IEEE Access*, vol. 7, pp. 123 473–123 489, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8809670/

[8] B. Varghese, N. Wang, D. S. Nikolopoulos, and R. Buyya, "Feasibility of fog computing," in *Handbook of Integration of Cloud Computing, Cyber Physical Systems and Internet of Things*. Springer, 2020, pp. 127–146.

[9] B. Gedik, S. Schneider, M. Hirzel, and K.-L. Wu, "Elastic scaling for data stream processing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1447–1463, 2013.

[10] M. D. de Assuncao, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *Journal of Network and Computer Applications*, vol. 103, pp. 1–17, 2018.

[11] M. E. Oliver, C. F. Mason, and D. Finnoff, "Pipeline congestion and basis differentials," *Journal of Regulatory Economics*, vol. 46, no. 3, pp. 261–291, 2014.

[12] S. Guillaume, "Designing fuzzy inference systems from data: An interpretability-oriented review," *IEEE Transactions on Fuzzy Systems*, vol. 9, no. 3, pp. 426–443, 6 2001.

[13] J. M. Soto-Hidalgo, P. M. Martinez-Jimenez, J. Chamorro-Martinez, and D. Sánchez, "Jfcs: A color modeling java software based on fuzzy color spaces," *IEEE Computational Intelligence Magazine*, vol. 11, no. 2, pp. 16–28, 2016.

[14] R. Logambigai, S. Ganapathy, and A. Kannan, "Energy–efficient grid–based routing algorithm using intelligent fuzzy rules for wireless sensor networks," *Computers & Electrical Engineering*, vol. 68, pp. 62–75, 2018.

[15] Y. Li, Z. Sun, L. Han, and N. Mei, "Fuzzy comprehensive evaluation method for energy management systems based on an internet of things," *IEEE Access*, vol. 5, pp. 21 312–21 322, 2017.

[16] S. K. Basha and T. Shankar, "Fuzzy logic based forwarder selection for efficient data dissemination in vanets," *Wireless Networks*, vol. 27, no. 3, pp. 2193–2216, 2021.

[17] J.-P. Poli and L. Boudet, "A fuzzy expert system architecture for data and event stream processing," *Fuzzy Sets and systems*, vol. 343, pp. 20–34, 2018.

[18] G. Acampora and V. Loia, "Fuzzy control interoperability and scalability for adaptive domotic framework," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 2, pp. 97–111, 2005.

[19] J. M. Soto-Hidalgo, J. M. Alonso, G. Acampora, and J. Alcala-Fdez, "JFML: A java library to design fuzzy logic systems according to the IEEE Std 1855-2016," *IEEE Access*, vol. 6, no. 1, pp. 54 952–54 964, 2018. [Online]. Available: http://www.uco.es/JFML/

[20] F. J. Rodríguez-Lozano, J. C. Gámez-Granados, O. Banos, J. Alcalá-Fdez, and J. M. Soto-Hidalgo, "An approach to bridge the gap between ubiquitous embedded devices and JFML: A new module for internet of things," in *IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2021*. IEEE, 2021, pp. 1–6. [Online]. Available: https://doi.org/10.1109/FUZZ45933.2021.9494483

[21] J. M. Soto-Hidalgo, A. Vitiello, J. M. Alonso, G. Acampora, and J. Alcalá-Fdez, "Design of fuzzy controllers for embedded systems with JFML," *International Journal of Computational Intelligence Systems*, vol. 12, no. 1, pp. 204–214, 2019.

[22] M. del Carmen Rey-Merchán, A. L. Arquillos, and J. M. Soto-Hidalgo, "A fall from height prevention proposal for construction sites based on fuzzy markup language, jfml and iot solutions," in *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2021, pp. 1–6.

[23] M. Martínez-Rojas, M. J. Gacto, A. Vitiello, G. Acampora, and J. M. Soto-Hidalgo, "An internet of things and fuzzy markup language based approach to prevent the risk of falling object accidents in the execution phase of construction projects," *Sensors*, vol. 21, no. 19, p. 6461, 2021. [Online]. Available: https://doi.org/10.3390/s21196461

[24] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: a survey," *Future generation computer systems*, vol. 56, pp. 684–700, 2016.

[25] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (iot) security: Current status, challenges and prospective measures," in *2015 10th international conference for internet technology and secured transactions (ICITST)*. IEEE, 2015, pp. 336–341.

[26] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the internet of things: A review," *big data and cognitive computing*, vol. 2, no. 2, p. 10, 2018.

[27] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE internet of things journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[28] D. Namiot, "On big data stream processing," *International Journal of Open Information Technologies*, vol. 3, no. 8, pp. 48–51, 2015.

[29] V. Cardellini, G. Mencagli, D. Talia, and M. Torquati, "New landscapes of the data stream processing in the era of fog computing," *Future Generation Computer Systems*, vol. 99, pp. 646–650, 2019.

[30] M. Welzl, *Network congestion control: managing internet traffic*. John Wiley & Sons, 2005.

[31] C. Kulatunga, N. Kuhn, G. Fairhurst, and D. Ros, "Tackling bufferbloat in capacity-limited networks," in *2015 European Conference on Networks and Communications (EuCNC)*. IEEE, 2015, pp. 381–385.

[32] O. Hohlfeld, E. Pujol, F. Ciucu, A. Feldmann, and P. Barford, "Bufferbloat: how relevant? a qoe perspective on buffer sizing," *Technische Universität Berlin, Tech. Rep*, 2012.

[33] G. Acampora, B. di Stefano, and A. Vitiello, "IEEE 1855^{TM}: The first IEEE standard sponsored by IEEE Computational Intelligence Society," *IEEE Computational Intelligence Magazine*, vol. 11, no. 4,

pp. 4–7, 2016. [Online]. Available: https://standards.ieee.org/findstds/standard/1855-2016.html
[34] P. Angelov and R. Yager, "Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density," in *IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*, 4 2011, pp. 62–69.
[35] T. J. Ross, *Fuzzy logic with engineering applications*, 4th ed. John Wiley & Sons, 2016.
[36] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Human-Computer Studies*, vol. 51, no. 2, pp. 135–147, aug 1999.
[37] International Electrotechnical Commission Technical Committee Industrial Process Measurement and Control, document IEC 61131,Programmable Controllers, 2000.
[38] J. Alcalá-Fdez, J. M. Alonso, C. Castiello, C. Mencar, and J. M. Soto-Hidalgo, "Py4jfml: A python wrapper for using the ieee std 1855-2016 through jfml," in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–6.