SFDDM: A Secure Distributed Database Management in Combined Fog-to-Cloud Systems

Souvik Sengupta, Sarang Kahvazadeh, Xavi Masip-Bruin, Jordi Garcia
CRAAX Lab, Dept. of Computer Architecture (DAC), UPC BarcelonaTech, Vilanova i la Geltrú, Spain
Email: (souvik, skahvaza, xmasip, jordig) @ac.upc.edu

***Abstract*—Technological revolutions have greatly increased the use of IoT devices for our daily life. Driving the fact that everything surrounding us is getting connected what turns into an unstoppable increase in the amount of data produced. This data represents the state of diverse environmental events and helps to control a large set of distinct activities. So, accurate and secure management of this data is essential for any computing platform. Moreover, in order to provide real-time services in a distributed system (i.e., smart city), the data should be properly and securely managed. It is well known that shifting these tasks to the edge (i.e., near to the end users), highly facilitates these two objectives. The recently proposed Fog-to-Cloud (F2C) model is intended to enable data processing near to the edge, which helps to get better latency-sensitive services. However, some challenges remain to accurately and securely manage this data over the system, mainly due to the distributed F2C nature. Thus, considering these facts and challenges, in this paper we propose an architectural solution aimed at building a secure distributed database for F2C systems. Then, considering a real-case scenario, we perform some tests to measure the performance of our proposing schema. Finally, by comparing the performance between traditional cloud, fog/edge based execution model and our proposing SFDDM, we validate the effectiveness of our proposing schema.**

***Index Terms*—Fog-to-Cloud (F2C), Distributed Database, Internet-of-Thing (IoT), Security and Privacy**

I. Introduction

The connection of the large set of diverse devices and objects deployed at the edge to the network, undoubtedly eases the wide adoption of the set of envisioned Internet-of-Things (IoT) benefits, mainly rooted on the deployment of innovative services with a large impact on the society, out of which large scale situation-awareness applications may be strongly highlighted (e.g., large-scale air quality monitoring, large-scale fire detection monitoring, city's traffic monitoring or large area video surveillance). Interestingly, these applications rely on a large amount of data produced by a non-negligible amount of IoT devices. Moreover, for large scale situation-awareness applications, it is essential for all IoT devices to work in a sense-process-actuate control loop at a machine-perception speed, in order to guarantee a secure and much better performance. However, assuming all data processing is handled at remote clouds some relevant issues come up. Indeed, the network delay and connection unreliability between the remote cloud and the IoT devices deployed at the edge makes the cloud-based model not to be the proper one for latency-sensitive situation-awareness applications. Aimed at fixing these latency issues, fog computing [1] recently popped up, pushing for moving cloud to the edge, thus setting a user/IoT devices proximate infrastructure, able to manage some processing capacities and consequently shrinking the need for moving data processing to the cloud, tuning into a notable latency reduction. Even more, assuming cloud is here to stay, large benefits may be obtained by the smart coordination of fog and cloud infrastructures, what has been recently coined as Fog-to-Cloud (F2C) computing [2]. In fact, by enabling both a better resources utilization mechanism, an optimal data processing over the whole set of available resources and by improving the service execution at the edge devices, F2C ensures a much better service performance, also for latency-sensitive applications.

Interestingly, in order to make the most of the capacities brought in by the edge resources, it is essential to keep the data near to the edge. But this is not enough. It should be also noticed that the data required by several applications and particularly by situation-awareness applications (in general, any data coming from monitoring infrastructures, resources, people, etc), is usually quite sensitive, hence unauthorized access, process and tampering data might cause severe degradation of the overall system performance. Consequently, secure data distribution and processing are basic needs for these applications [3]. However, when dealing with edge devices, guaranteeing security and privacy for edge devices is a tough task [4], mainly motivated by the low capacities normally inherent to edge devices. In this complex and challenging context, this paper proposes an architectural solution for a secure distributed database management in scenarios combining fog and cloud resources, hereby referred to as Secure Fog-to-Cloud Distributed Database Management (SFDDM).

The rest of the paper is organized as follows. In Section II, we discuss the related work. After that, in Section III, we briefly describe the framework of the proposed SFDDM architectural solution. Then, in Section IV, we evaluate the performance of our SFDDM proposal. Finally, we conclude our paper presenting some future direction and final remarks.

II. Related Work

There are some contributions dealing with solutions for data management in cloud and fog without considering the coordinated and hierarchical F2C nature. In [5], [6], authors focus on an efficient data management in a fog computing ecosystem. More concretely, in [5] authors show their concerns to find out the best possible way to appropriately replicate the data over the distributed database system without issuing

inconsistency issues. On the other hand, authors in [6] propose to design a suitable data-model aimed at ensuring efficient data distribution to the overall distributed computing framework. However, none of these two contributions considered security functionalities and provisioning over the data management. Another contribution [7], focuses on building an appropriate framework for managing the huge amount of data in the federated fog/edge and cloud platform, although, the work is not considering any security implementation over the data management in the federated fog/cloud. In [8] authors design a framework to bring the full cloud facilities near to the edge minimizing the data-query response time, which helps to optimize the performance of latency-sensitive situation-awareness services. However, the proposed architecture relies on computational resources from volunteering contributors what can cause high-security risks. In [9], a holistic vision-centric approach, proposing an architectural vision for managing the data is introduced, including novel hypothetical leveraged data-centric model approaches for managing the system data over the distributed network in a trustworthy fashion. However, authors only propose an architecture with no details about security deployment and trust establishment between layers such as IoT, fog and cloud. Other papers, put special attention to highlight potential concerns to design access control policies [3] aimed at ensuring the security and privacy needs [10] for specific distributed database frameworks. However, in these proposals, access control and security provisioning are handled by a centralized component, becoming a single point of failure that can compromise the whole system. Therefore, after revisiting current works, we may conclude that an architectural solution for a secure and distributed management and access of the data in combined fog/cloud systems is a must and which is yet demanding specific efforts from the scientific community.

III. Architectural Description

The key benefits of the coordinated management of fog and cloud resources have been already demonstrated [2], in terms of both a much better service execution and optimal resources utilization. These benefits may be highly emphasized whether the collected data to be processed are kept close to the edge of the network or not. However, as already stated before, dealing with data, especially when playing with data sensitive applications (an almost mandatory requirement when running applications and services related to users or private systems, e.g., situation-awareness applications), demands strong guarantees on privacy and security. By aiming to sort out these challenges and making security a key functionality in combined fog and cloud scenarios, in this section, we present a novel architectural solution for data sensitive applications for smart computing scenarios. To that end, the envisioned scenario is presented, illustrating the proposed security services which are offered by the proposed architectural solution.

A. Application context: use-case scenario for SFDDM

There are many scenarios where the coordinated management of fog and cloud resources brings in substantial benefits, such as e-health, industry 4.0, smart transportation or smart cities, just to name a few. Particularly, as introduced in the first section, latency-sensitive situation-awareness applications put together a large set of highly demanding services with a large impact on smart cities scenarios. In fact, through a vast deployment of smart systems and devices, cities are evolving towards an improved and sustainable living facility offering a much better quality of life by reducing environmental pollution and also solving many other issues [11]. Several efforts are already done aimed at building up smart solutions for improving life quality in modern cities. For example, authors in [12] proposed a methodology for using WSN technology for reducing air pollution in modern cities. In the proposed scenario, many resource-constrained devices at the edge collect data to measure air quality. These data, in a smart city context, particularly when dealing with latency-sensitive situation-awareness applications, must be processed near to the edge.

As described earlier, the coordinated management of cloud and fog resources may help to achieve that set of requirements, taking real-time decisions about the set of resources to use according to the real needs, be it at the cloud or at fog. Hence, when putting together the needs required by the envisioned highly demanding scenario and the features offered by the F2C model, we indeed notice that F2C may be a nice management paradigm on which, a solution may rely on. For the sake of illustration, the proposed architectural solution is applied for air quality and traffic monitoring service in smart cities. Fig. 1 presents a topological approach of the tentative infrastructure required to deploy the air and traffic service on a city matching the F2C model. As shown in the picture, there are many quality measurement sensors and surveillance cameras geographically distributed over the city, aimed at measuring the air quality level and monitoring the traffic in real-time. Assisted by the surveillance cameras, information about the total number of vehicles moving in the city for a given time may be also collected. In practice, considering the volume of sensors and the volume of vehicles, a large volume of data is expected. Moreover, for a much better service experience in a smart city domain, it is essentially required to share the captured data with existing local government organization (e.g., meteorological dept., traffic controlling dept., etc.) and also among the citizens. Unfortunately, processing and securely sharing this high volume of data in real-time is pretty difficult when considering that edge devices are resource-constrained systems with poor storage and processing capacities, mainly due to two key factors. First, it is evident that cloud may be a nice candidate to handle these needs, but bandwidth limitations and large distances may make the cloud, as a standalone solution, not to be the ultimate one. Second, unreliable communication between edge devices and cloud brings additional complexity in the system while also increasing the chances for data losses.

B. Architectural schema & components: of SFDDM

In order to alleviate all the upper-mentioned issues, we develop a modified F2C-based execution model, supported

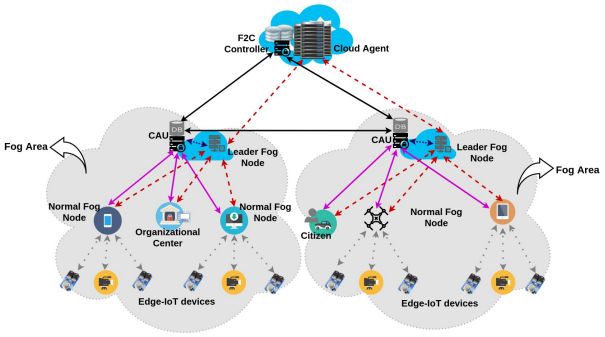

Fig. 1. F2C-based execution model for monitoring traffic and air quality in a Smart City scenario.

by an architectural schema named - Secure F2C Distributed Database Management system (SFDDM). The SFDDM mainly consists of three (3) layers and six (6) types of architectural elements, as discussed below.

1) Cloud Layer: This layer is considered as the most resource-enriched layer. It is the uppermost layer of our proposing architectural schema as shown in Fig. 3. According to our schema, two-types of architectural elements are working in this layer, the F2C Controller and the Cloud Agent (ClA). The F2C Controller is the master node acting as a certificate authority providing authentication for distributed control-area-units (CAUs) located at the edge of the network. F2C Controller gives authorization for security provisioning at the edge of the network to the distributed CAUs. Also, it acts as the data center node for our proposing distributed database oriented architectural schema. On the other hand, ClA is the cloud node, which is basically a composite form of cloud resources. It is responsible for providing the various kind of cloud facilities, including the storage facilities for historical data. The F2C Controller has secure intercommunication with the ClA. Key details and functionalities of the F2C Controller and CAUs are further extended in subsection III-C.

2) Fog Layer: In the proposed architecture, this layer is mainly responsible for bringing the cloud facilities near to the edge of the network. According to the architecture schema shown in Fig. 2, this layer is a composite layer of two different sub-layers. The upper sub-layer is called as the Fog Manager Layer (FML). Another sub-layer resides below of the FML, referred to as the Fog Employee Layer (FEL). The FML sub-layer consists of two different architectural elements - the Control Area Unit (CAU) and the Leader Fog Node (LFn). The CAUs are responsible for securely managing and storing the captured data, and whereas the LFns are responsible for bringing other cloud facilities near to the end-users. The FEL sub-layer elements are comparatively more resource constrained than the FML sub-layer elements, known as Normal Fog Node (NFn). The NFns act as the end-point or gateway for the Edge-IoT devices (e.g., sensor nodes).

From a topological perspective, we may consider that in a modern F2C-enabled smart city scenario, several small areas may emerge to properly provide F2C services near to the edge of the network. These small areas are individually known as fog areas. In the proposed architectural schema, we consider that each fog area has one CAU and one LFn, respectively responsible for providing the security functionalities, storing the data near to the edge and managing the other devices (i.e., NFn) within their scope.

3) Edge-IoT Layer: This layer is the bottom layer of the proposed architectural schema (Fig. 2), and consists of various kind of small IoT devices (e.g., sensor nodes, surveillance camera). These devices are responsible for both the continuous capturing of various environmental events (such as temperature sensing) and the execution of some actions (such as automatic fire extinguisher), according to some instructions.

Moreover, by recognizing the need for a strong data security management at edge devices, particularly in data-sensitive applications, the proposed architecture designs a distributed database cluster deployed on top of all CAUs and the F2C Controller. Next subsection briefly describes the main security features of the proposed architectural schema.

C. Security services

The proposed data management security schema leverages the decoupled distributed security framework introduced in [13], briefly described in terms of its main components next:

1) F2C Controller: The F2C Controller resides at cloud and acts as the master node for the distributed control-area-units (CAUs) deployed at the Fog Manager Layer *FML*. The F2C Controller provides authentication, authorization, access control and secure channels for distributed CAUs. Once authorized, CAUs are responsible for providing security in their corresponding fog areas.

2) Control-Area-Unit (CAU): The CAUs (deployed at the fog layer), once authenticated and authorized by the F2C Controller, are responsible for providing distributed authentication and access control, as well as for enabling the secure channels for their corresponding fog areas. Then, CAUs become trustable to act as distributed security controllers at other fog nodes (i.e., LFn and NFn) to provide the security requirements within their scope. CAUs provide security services for LFn, NFn and even other edge and IoT devices. All the CAUs are communicating with each other through a secure intercommunication channel. The CAUs, after being authenticated and authorized from the F2C Controller can provide security at the edge without relying on the F2C Controller at the cloud.

In the proposed architectural schema, the communication between all CAUs, CAUs and F2C Controller, LFn and CAU, as well as NFn and CAU are implemented through transport layer security (TLS), hence all communications are handled through secure channels. CAUs are in charge of providing certificate authorities for their corresponding fog areas. Indeed, by using X.509 public key certificates with the RSA cryptography algorithm, CAUs provide distributed authentication within their scope. Moreover, CAUs provide access control (Role-based) to the distributed database to prevent any unauthorized access. Finally, before storing the data, the CAUs encrypt the data by implementing the advanced encryption standard

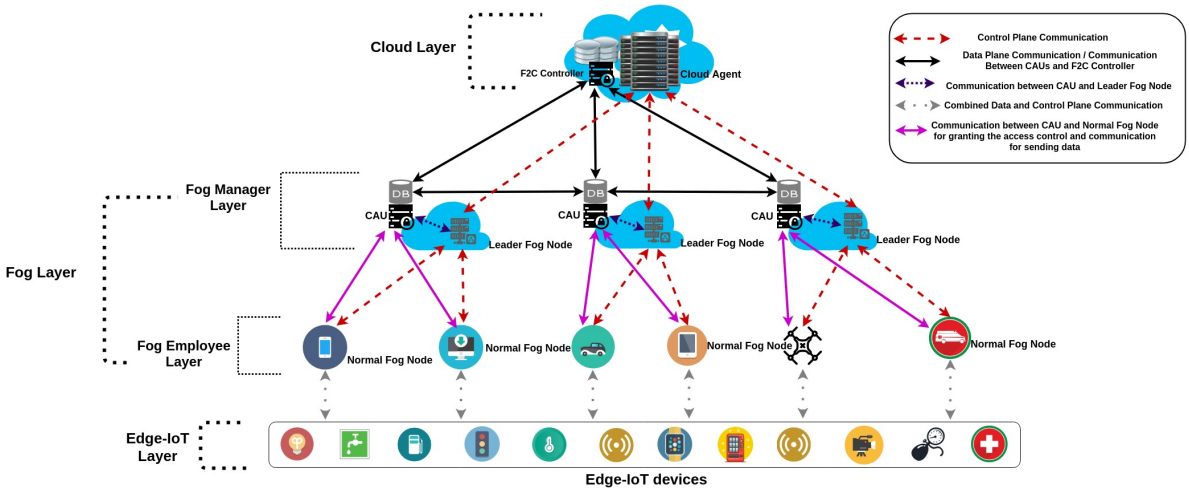

Fig. 2. CAU-based Distributed Database in the F2C scenario

(AES), thus preventing any data leakage, eavesdropping, and any type of passive and active attack over the database.

D. Functionalities of the SFDDM schema

The set of expected services provided by the proposing SFDDM architectural schema relies on two algorithms (Algorithm 1 & 2). Algorithm 1, describes the data aggregation procedure, while Algorithm 2 describes the secure data storing and retrieving procedure in the proposing SFDDM schema.

Algorithm 1 For preparing the aggregated data in a NFn

Initial Consideration: $MQ_s = [\,], Q_s = [\,],\ C_l,\ C_t$

procedure AGGREGATION_OF_DATA
  Check the data sources for context
  if contexts are same **then**
    check the values of C_l *and* C_t *for every data sources*
    if *All the* C_l *and* C_t *are same* **then**
      Add all the data into Q_s
      Perform data aggregation on Q_s
      Estimate Mean, Median, and Mode of Q_s
      Calculate the difference between Mode value and each data of Q_s
      if $difference > 1$, *for any data in* Q_s **then**
        Send alert sms to LFn and Add Estimated Mode value to the MQ_s
      else *Add the Estimated Mode value to the* MQ_s
    else *Add all the data in* MQ_s
  else *Add all the data in* MQ_s

In our envisioned scenario, considering latency-sensitive situation-awareness services in a smart city as an illustrative example, all Edge-IoT devices are responsible for both continuous capturing the different environmental events (i.e., air pollution measurement, passing vehicles counting) and generating (i.e., collecting) the data. Afterwards, once the data is generated, the Edge-IoT devices immediately send this data to their connected NFn. Once the NFn receives this data, it starts aggregating this data. To that end, the SFDDM solution adopts the statistical average method. Thus, on the basis of the data context and with the help of data capturing time (C_t) and capturing location (C_l), the aggregation is being done. Most importantly, by doing so, it is possible to detect faulty data. Then the clean data packet is built for sending to the upper layer in the SFDDM architectural schema. At this step the Algorithm 1 stops working.

Algorithm 2 For Storing/Retrieving data

Initial Consideration: $MQ_s = [\,],\ MQ_r = [\,], InV = [\,]$

procedure DATA_STORING/RETRIEVING
  Check the authentication of NFn and register them, if not registered
  ***FIRST PHASE*: For Storing Data**
  Aggregate the data in NFn and insert it in MQ_s, against the InV
  Send the MQ_s, and InV to CAU
  Confirm the accessing permission of NFn for storing data in distributed DB
  Encrypt the MQ_s and store it into the distributed DB (in a table) considering the InV value
  ***SECOND PHASE*: For Retrieving data**
  Confirm the accessing permission of NFn for retrieving data from distributed DB
  Make the query from other NFn, considering the desired InV value and table name to the CAU
  Then, CAU find the encrypted data from it's database, and then decrypt the data and also put it in the MQ_r
  CAU send the MQ_r to the NFn

In order to store and retrieve the data, the Algorithm 2

is deployed. According to this algorithm, before sending or retrieving the data, all NFns must be authenticated, before sending the data packet to the CAU of their corresponding fog area. Once the authorized NFn gets the accessing permission from the corresponding CAU, then it can start sending the data packet (MQ_s). In order to ensure privacy and data integrity, we encrypt the (MQ_s), before storing it against the index value (InV = [C_l, C_t]).

To that end, for retrieving the data, also all query makers (e.g., organizational centre, citizen) need to register themselves and claim the access permission from the CAU of their corresponding fog area. Once they get access permission, they can start querying for the desired data. When a CAU gets the query from the query makers, then the CAU starts looking for the data in the distributed database with the help of the mentioned InV. After finding the data from the distributed database cluster, the CAU decrypts the data and prepares the response packet (MQ_r) for sending to the query makers. Following this way, our schema securely delivers data to the query makers.

IV. Performance Evaluation

To evaluate the performance of our proposed architectural schema, we have configured our project test-bed [14] and emulated our schema. In addition, we have also used the Python-based simulation tool (YAFS [15]) to generate the traditional Cloud-based and Edge/Fog-based execution models, in order to compare our performance with the traditional execution models. We have set up a prototype of the F2C-enabled smart city scenario in our research lab, where the Cloud Layer elements (i.e., F2C Controller and Cloud Agent) are hosted on a server with Intel Xeon family E5-2620 V4 series (clock speed @3GHz), 96GB RAM, 1TB Hard Drive, and running on Ubuntu 16.04LTS Linux. In our experimental testbed, all the Fog Layer elements (i.e., CAUs, LFn and NFn) are relatively small computing devices: Raspberry Pi3 B+ models, running on Ubuntu 16.04LTS Linux and with Cortex A53 @ 1.4GHz processor, 1GB SD-RAM, and 64GB micro-SD card. Similarly, we are hosting the LFn and NFn on some Raspberry Pi Zero models, working with 1GHz single-core CPU, 512MB RAM and, for storage purposes, we are using 8GB microSD for each of the LFn and NFn. For storing the captured data and securely distributing it over the network, we are creating a distributed database cluster over all the CAUs and F2C Controllers. For that purpose, we are using the containerized Apache Cassandra *(Dockerized-Cassandra)*. Tests have been performed implementing the multi-datacenter, multi-node based Cassandra cluster over the considered distributed framework. Security in our proposing schema has been ensured by implementing the aforementioned security functionalities. Similarly, we have customized the YAFS simulator to create the traditional cloud-based and fog-based smart city scenario, considering the same resources specification (i.e., RAM size, CPU specs., bandwidth), in order to compare with our proposed schema. For, cloud-based and traditional edge/fog based execution model, we are permanently storing the data in the cloud. Whereas, in the traditional edge/fog based model, before permanent storing of the data, the fog nodes are caching and processing this data for a limited amount of time. The feasibility of our schema has been validated performing the test over several amounts of distinct data packets. Individually, the average size of each of the packet is 30kB. By imitating the smart city scenario, we understand that at the edge of the network the communication bandwidth is reduced. In our proposed schema, we consider the maximum connection speed between Edge-IoT devices and the NFn is 2 MBps. Similarly, the maximum connection speed between NFn and LFn/CAUs is 5-7 MBps. And the maximum connection speed between different FML's architectural elements (e.g. LFn and CAUs) and also with the cloud layer's element is 11 MBps. A thorough observation at any smart city scenario, easily shows that bandwidth utilization is one of the most critical issues to manage. Considering this requirement, we perform our first evaluation test to show the performance of our proposed schema.

A. Bandwidth utilization: Cloud vs Fog vs SFDDM

SFDDM aggregates the raw captured data in the NFn, before being sent to the distributed database. After that, the aggregated data is sent to be stored in the distributed database, located at the FML. Eventually, this reduces the overall bandwidth utilization for our proposing schema. In addition, as the distributed database cluster is residing near to the edge, it also reduces the data transmission time for storing the data into the database. In Fig. 3, we present the performance evaluation of our model. Here, the deep magenta coloured line represents the performance of the SFDDM schema, and the green coloured line and the red coloured line represent the performance of fog-based and cloud-based execution model, respectively. We observe that our model reduces the data packet transmission time by approximately 33% compared to the cloud-based execution model and approximately 44% compared to the traditional fog/edge-based execution model.

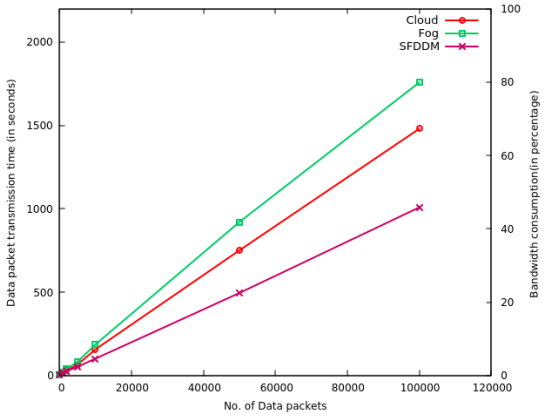

Fig. 3. Test result1: Data transmission vs Bandwidth usage

B. Query response time: Cloud vs Fog vs SFDDM

We consider two different cases to measure the query response time. In the first case (Fig. 4), we consider that the data source and the query makers (i.e., organizational centre or citizen) are within the same scope (i.e., same fog area). In the second case (Fig. 5), we consider the query for searching the data is made either from a remote area or from a different scope (i.e., different fog areas).

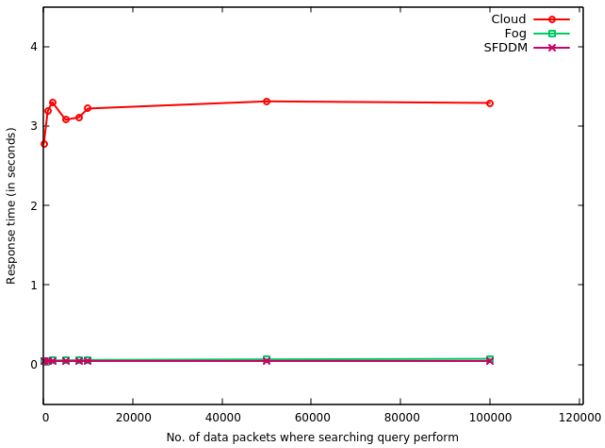

Fig. 4. Test result2: Query response time (data within local scope).

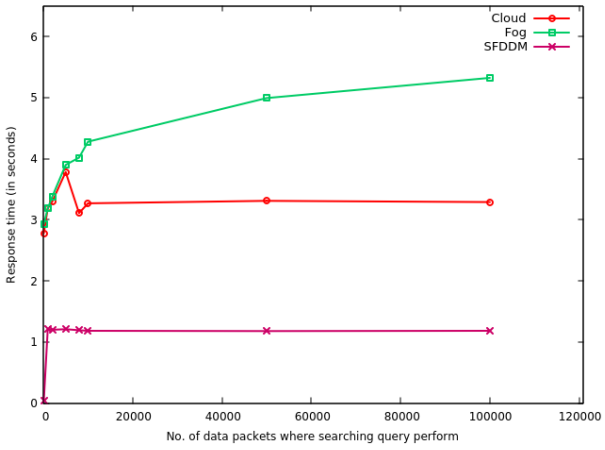

Fig. 5. Test result2: Query response time (data in remote area).

In the first case (Fig. 4), we have found that, since the data is residing within the same scope, the query response time for the normal fog-based execution model and our SFDDM schema is quite similar; however, the query response time at cloud is extremely high. Interestingly, in the second case (Fig. 5), we have found that our schema performs much better than regular fog-based execution model. As the SFDDM has the distributed database near to the edge, the data is dispersed and therefore closer to the edge of the network. But, in case of the normal fog-based execution model, the query has travelled through the upper layer (e.g., cloud layer), so this usually takes a long time and degrades the performance for normal fog-based execution model. The traditional cloud-based execution model again has higher response time, because the data is residing far from the query makers.

C. Data loss: Cloud vs Fog vs SFDDM

Finally, we measure the data loss. To that end, we consider the same two cases for some video streaming applications in a smart city scenario. In the first experiment (Fig. 6), the data (image) source node (which hosts the surveillance camera) and the monitoring node (i.e., organizational centre) are residing within the same scope (i.e., in the same fog area), and in the second case (Fig. 7), both nodes are residing in different scopes (i.e., in different fog areas). For both cases, we found that our SFDDM schema provides better performance than either traditional cloud-based or fog-based execution models. Indeed, implementing the distributed security functionalities and database near to the edge is guaranteeing a better performance for our SFDDM schema.

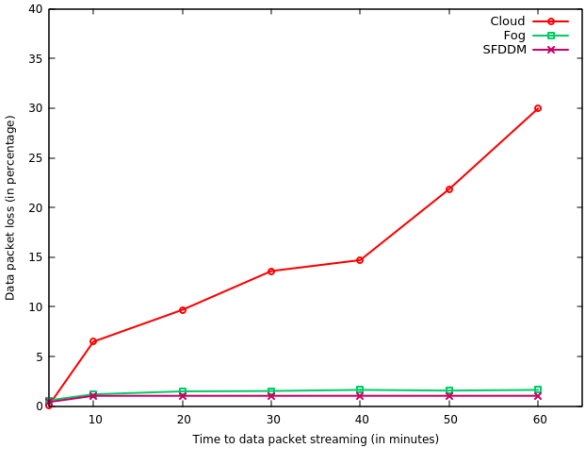

Fig. 6. Test result3: Data loss (Data within the local scope)

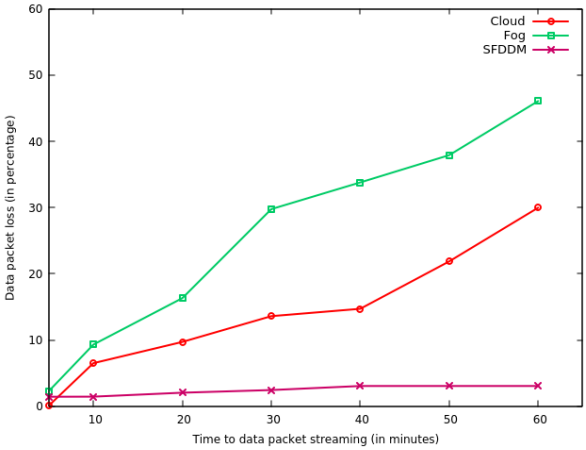

Fig. 7. Test result3: Data loss (Data is in the remote area)

V. Conclusion

In this paper, we have proposed an architectural schema for the secure distributed database management system in the F2C-based execution model. It provides enhanced features and performance for any latency-sensitive situation-awareness applications scenario. By performing some tests, we have measured the performance of our SFDDM architectural schema, and compared with those in a cloud-based and fog-based models. We have shown the improved performance of our model compared with any of the traditional models.

This research work is presented as the primary steps for making an efficient and secure F2C-based system, so many challenges still remain to be addressed. For that reason, additional work is essential. For example, in our framework, we did not consider the security functionalities for the edge IoTs (e.g., sensor nodes, surveillance camera). We know that, as these edge IoTs are mostly resource-constrained, so ensuring security and privacy functionalities for them, is one of the most challenging tasks. For this reason, these challenges and many other open issues will constitute the core of our future work.

Acknowledgment

This work has been supported by the Spanish Ministry of Science, Innovation and Universities and the European Regional Development Fund (FEDER) under contract RTI2018-094532-B-I00, and by the H2020 European Union mF2C project with reference 730929.

References

[1] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*. Springer, 2014, pp. 169–186.

[2] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G.-J. Ren, "Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 120–128, 2016.

[3] M. Gupta, F. Patwa, and R. Sandhu, "An attribute-based access control model for secure big data processing in hadoop ecosystem," in *Proceedings of the Third ACM Workshop on Attribute-Based Access Control*. ACM, 2018, pp. 13–24.

[4] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018.

[5] R. Mayer, H. Gupta, E. Saurez, and U. Ramachandran, "Fogstore: Toward a distributed data store for fog computing," in *2017 IEEE Fog World Congress (FWC)*. IEEE, 2017, pp. 1–6.

[6] T. Kudo, "Fog computing with distributed database," in *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2018, pp. 623–630.

[7] M. Malensek, S. L. Pallickara, and S. Pallickara, "Hermes: Federating fog and cloud domains to support query evaluations in continuous sensing environments," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 54–62, 2017.

[8] M. Ryden, K. Oh, A. Chandra, and J. Weissman, "Nebula: Distributed edge cloud for data intensive computing," in *2014 IEEE International Conference on Cloud Engineering*. IEEE, 2014, pp. 57–66.

[9] E. M. Schooler, D. Zage, J. Sedayao, H. Moustafa, A. Brown, and M. Ambrosin, "An architectural vision for a data-centric iot: Rethinking things, trust and clouds," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1717–1728.

[10] L. Lyu, K. Nandakumar, B. Rubinstein, J. Jin, J. Bedo, and M. Palaniswami, "Ppfa: Privacy preserving fog-enabled aggregation in smart grid," *IEEE Transactions on Industrial Informatics*, 2018.

[11] M. Thuzar, "Urbanization in southeast asia: developing smart cities for the future?" *Regional Outlook*, p. 96, 2011.

[12] M. S. Jamil, M. A. Jamil, A. Mazhar, A. Ikram, A. Ahmed, and U. Munawar, "Smart environment monitoring system by employing wireless sensor networks on vehicles for pollution free smart cities," *Procedia Engineering*, vol. 107, pp. 480–484, 2015.

[13] S. Kahvazadeh, V. B. Souza, X. Masip-Bruin, E. Marin-Tordera, J. Garcia, and R. Diaz, "Securing combined fog-to-cloud system through sdn approach," in *Proceedings of the 4th Workshop on CrossCloud Infrastructures & Platforms*. ACM, 2017, p. 2.

[14] R. Sosa, R. Sucasas, A. S. ENG, A. Leckey, C. Cordeiro, J. Jensen, C. Ketley, S. Crompton, and F. Carpio, "D5. 1 mf2c reference architecture (integration it-1)."

[15] I. Lera and C. Guerrero, "Simulation tool: Yet another fog simulator (yafs)," 2017. [Online]. Available: https://yafs.readthedocs.io/en/latest/index.html