Context Interoperability for IoT through an Edge-centric Context Sharing Architecture

Everton de Matos, Ramão Tiago Tiburski, Leonardo Albernaz Amaral, Fabiano Hessel
Pontifical Catholic University of Rio Grande do Sul (PUCRS) - Porto Alegre - RS - Brazil
(everton.matos.001, ramao.tiburski, leonardo.amaral)@acad.pucrs.br, fabiano.hessel@pucrs.br

***Abstract*—The adoption of the Internet of Things (IoT) demands advances to cope with the large heterogeneity of IoT entities (i.e., systems, applications, and devices). Context information is an essential characteristic of these entities, which can store relevant details about their environments and related events. However, with the integration of different IoT vertical domains, providing isolated context is no longer enough. Sharing the context information is mandatory to have interoperability. Edge computing emerges as a promising approach to help in filling the context sharing gap by minimizing information overhead and reducing network latency. In this sense, this paper defines an Edge-centric Context Sharing Architecture able to make context sharing based on edge-to-fog approach. We also discuss the requirements for context sharing and the related work in the area to make clear the novelty of the architecture.**

I. Introduction

The Internet of Things (IoT) has been adding new dimensions to the world of information and communication technology [1]. As miniaturization still continues and computing capacity still increases, edge sensors (IoT devices) become more powerful. Edge Computing enables moving the IoT computation from the high-powered central Cloud to the edge of the network. The benefits of edge computing result from its proximity to data sources and end users [2].

There is a common sense that IoT devices generate a lot of data. The context-aware computing helps in interpret and understand these data in a proper way, producing context information to characterizes the entities (e.g., location, status, updates). However, most context management platforms may have heterogeneous characteristics and are designed to facilitate applications in separate factions. In this sense, sharing context information between different kinds of systems has become a mandatory requirement in the IoT ecosystem [1].

To the best of our knowledge, there are no efforts around the definition of an architecture for IoT environments that mitigates the context sharing requirements. In this sense, this article defines an Edge-centric Context Sharing Architecture able to manage and process the context information of different entities to have a pool of abstractions and provide the context in an interoperable way.

The remainder of the paper is structured as follows: Section II presents context sharing concepts and it's requirements. Section III presents the related work. Section IV presents the proposed context sharing architecture. Section V presents the evaluation tests. Finally, Section VI concludes the paper.

II. Context Sharing Requirements

Context, also called context information, is considered any high-level information, sometimes semantic, that can be used to characterize the situation of an entity (e.g., person, place, or computing device). In most cases, the context information is stored individually by the system entities with no access for system-to-system or entities outside the restricted domain. *Context sharing* is a feature that allows entities to "understand" different context information into the same smart space domain or across different domains. Besides, context sharing is considered a gap in the context-awareness area [1].

A context sharing platform should encompass many functions to deal with complex application scenarios such as those found in IoT. Next items present the context sharing requirements for a platform, based on [1] and [3].

Modeling: An efficient model for handling, sharing, and storing context data is essential for a working context-aware system. The context information must follow a model to ease interpretation.

Reasoning: It can be defined as a method of deducing new knowledge based on the available context. In the context sharing domain, the reasoning function is related to the process of deducing to whom the context information must be sent.

Heterogeneity: Components of shared systems might be highly heterogeneous along several dimensions and applications, such as: how the context is produced, consumed, and understood. Full heterogeneity systems try to care about the interoperability in many verticals, such as network, data format, and connectivity. Partial heterogeneity systems usually care about local interoperability, with similar systems.

Systems Management: The context sharing platforms must be open to new devices connecting with them, thus enabling the interoperability between similar and different systems.

Scalability: Massive quantities of all sorts of context information will need to be processed at high speed and in an efficient way.

Security and Privacy: It is indispensable for the management of context information, once it often includes private information such as location and preferences.

Architectural Model: Context sharing platforms may vary regarding the architectural perspective. The processing can be all done in the cloud, or on the edge of the network [4]. Some edge systems may have a centralized point-of-control. On the other hand, some systems can adapt itself depending on the environment.

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

TABLE I
EVALUATION OF SURVEYED RESEARCH FRAMEWORKS, SYSTEMS, APPROACHES, AND ARCHITECTURES.

Sharing platforms	Ref.	Modeling	Reasoning	Heterogeneity	Systems management	Scalability	Security	Architectural model
Bluewave	[5]	Text-based	Rule-based	Partial	Yes	Yes	Yes	Centralized-edge
Chitchat	[6]	Key-value, Text-based	Probabilistic	Full	Yes	Yes	No	Adapt
CS-Sharing	[7]	Key-value	Rule-based	Partial	Yes	Yes	No	Decentralized-edge
HEAL	[8]	Key-value	Probabilistic	Partial	Yes	No	No	Cloud-based
RCOS	[9]	Ontology-based	Ontology-based	Full	Yes	Yes	No	Centralized-edge
Our Work		**Ontology-based**	**Ontology/Rule-based**	**Full**	**Yes**	**Yes**	**Yes**	**Adapt**

III. RELATED WORK

Table I presents the comparison of analyzed works based on the context sharing requirements (see Section II). Our work is presented in details at Section IV.

Bluewave [5] is a Bluetooth-based technique that allows mobile devices to share context when they are nearby. Chitchat [6] is a suite of context representation that combines the communication and context sensing capabilities of the devices to support creating and sharing views of local context.

CS-Sharing [7] enables context sharing in vehicular networks to monitor the road conditions. HEAL (Healthcare Event Aggregation Lab) [8] acts as a bridge between different platforms of the healthcare domain. RCOS (Real Time Context Sharing) [9] main objective is to enable a semantic matching between entities.

As shown in Table I, the most popular technique for *modeling* context information is key-value pairs. It is a lightweight and straightforward technique since every context information has a unique key. The usage of ontologies is a trend for *reasoning* about context information [1]. RCOS uses ontologies to define a context model and also to reason on it.

Most platforms do not address the *heterogeneity* requirement that is desired for IoT environments. RCOS stands out by using different set of ontologies to reach full heterogeneity. *Systems management* feature is addressed by all of the analyzed platforms. Chitchat fulfills the *scalability* feature by creating a small context representation for lightweight sharing. Both Bluewave and CS-Sharing are based on short-range communication, helping in the real-time issue.

Security and privacy is the less addressed feature of the analyzed platforms. However, Bluewave provides security at communication protocol level. There is not a standard for the *architectural model*. Bluewave, can process some information at the edge of the network and only use the cloud as storage. CS-Sharing works with vehicle networks, in which the context sharing occurs in a decentralized/distributed way. Chitchat can adapt its architecture depending on the application.

None of the analyzed systems deals with all the context sharing requirements. Moreover, the high IoT heterogeneity is not addressed by the systems at all. Although the context sharing concept is used in the pervasive computing area by some systems, the sharing mostly occurs locally with a small group of similar entities. A context sharing architecture that works with IoT environments was not deployed yet.

IV. EDGE-CENTRIC CONTEXT SHARING ARCHITECTURE

The proposed Edge-centric Context Sharing Architecture takes benefit of fog and edge computing approaches to minimize network communications, thus reducing failure points and improving scalability. The definition of the architecture is illustrated in Fig. 1-A. It is composed of two main systems: Context Sharing and Context Provider.

Context Sharing Systems are placed at the fog level and are responsible for sharing the context information with other entities or Context Sharing Systems instances, including different fogs. Context Provider Systems are placed at the edge layer, embedded or connected directly to IoT devices, and are responsible for providing context information based on IoT devices data. The context information must be modeled following a pre-defined pattern. An example of a context information modeled in JSON can be seen in Fig. 1-B. It contains all the information that will be shared and can also be presented in XML format.

Besides the presented Fog and Edge layers, it also has a Cloud layer for store all the context information. The Context Sharing Systems has information only about the deployed scope. However, the Cloud has information about all the instances. Next, we detail the Context Sharing and Provider System modules.

A. Context Sharing System

This software system is placed at the Fog layer. It has two main functions: (i) to coordinate the Context Providers Systems of its domain, and (ii) to receive context information and share it with who may concern.

The *Security Manager* module ensures protection against attacks on data and communication channels. *Context Selection* takes place when an entity is interested in getting context information from the Fog. It works alongside with the *Repository Manager*, that has a Context database with updated context information regarding the Context Providers of its domain. The database is replicated in a Cloud, and only the newest information stays at the Fog level.

Communication Services module has all the communication interfaces to guarantee the data exchange in an interoperable way. It is also used to communicate with the Cloud when needed. *Context Providers Manager* ensures that the Context Providers (edge) be able to register to a Context Sharing System (fog). *Context Providers' Reasoner* is responsible for determining who are interested in the shared context.

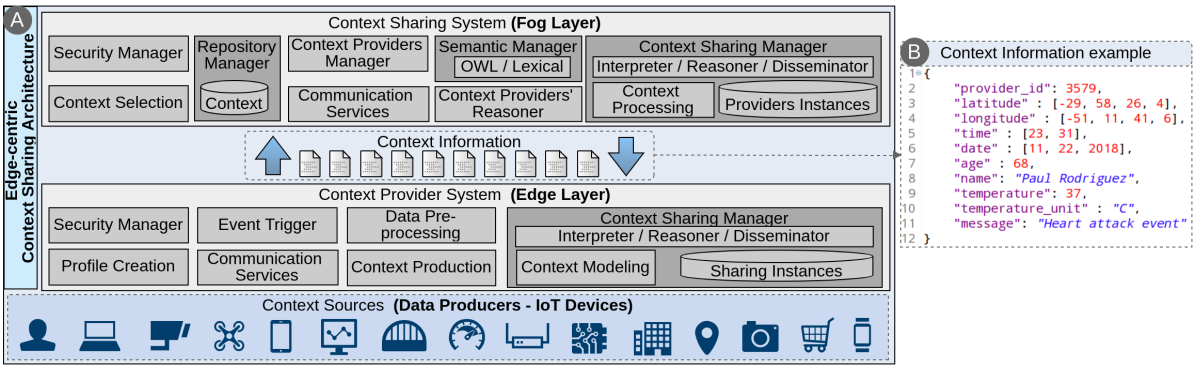

Fig. 1. A layered Edge-Centric Context Sharing Architecture.

Semantic Manager module works regarding heterogeneity trough OWL (Web Ontology Language). We defined an ontology for context information classification composed of three main subclasses: (i) *Context_Domain* subclass defines which domain the context came from (e.g., health, urban traffic); (ii) *Context_Format* to represents the context format (e.g., semantic, numeric); and (iii) *Context_Source* to denotes the context information source (e.g., user, network). The WordNet tool is also very useful for providing heterogeneity [10]. WordNet is a large lexical database of English. WordNet makes possible to understand that different words, such as *heat* and *warmth*, have the same meaning.

The *Context Sharing Manager* module interprets the context information, reason over it to produce new context information, and disseminate it to who may be interested. The reasoning process happens alongside with the Semantic Manager module ontologies. The reasoning output is the domain of the entities that may be interested in such context. The Providers Instances database has the address of the entities of each domain that will be interested in the context information to be shared. *Context Processing* provides the operations of aggregation and filtering in the context information.

B. Context Provider System

This software system is placed at the Edge layer and can be deployed alongside or within an IoT device. It has two main functions: (i) to model context information, and (ii) to receive shared context information for decision making.

Security Manager module offers protection to data and communication channels. *Profile Creation* is responsible for creating the Context Provider profile to be registered in the Context Sharing System. The profile follows standards of widespread sources, such as FIWARE, and oneM2M.

Event Trigger guarantees context information to be shared with the Context Sharing System automatically. *Communication Services* has the communication interfaces to guarantee the data exchange in an interoperable way. *Context Production:* It is a third-party system that produces context information. *Data Pre-processing* works for scalability. It avoids data stream from the Context Providers. Moreover, common data accessed by multiple applications is reused.

Context Sharing Manager module is responsible for the context sharing process at the edge. It has two main functions: (i) to model the context information to share it with the Context Sharing System infrastructure at the Fog level, and (ii) to interpret the received shared context information and reason over it to produce a new context. It also has the address of the Context Sharing System instance that the context information must be sent. A Context Provider can register to one or more Sharing Systems.

V. Validation

A. Internet of Things Application Scenario

A smart city is a complex IoT environment to encompass many different application verticals (e.g., healthcare, home-care, urban traffic, Emergency Medical Services - EMS) interacting with each other [11]. Let's consider a scenario in which the focus is on sharing the context of a home-care patient when some important events related to the health condition occurs (see Fig. 2-A). In this scenario, there are four deployed instances of the Context Sharing System: (i) home-care and home-automation, (ii) EMS, (iii) hospital infrastructure, and (iv) urban traffic infrastructure. Every instance is aware of the environment and knows with whom they must share context information. For every instance of the Context Sharing System, it will be various instances of the Context Provider System. For example, in the EMS Context Sharing System, every ambulance is a Context Provider System.

B. Environment Setup

Our main goal is to demonstrate the Edge-centric Context Sharing Architecture suitability in different networks, which is very common in IoT environments. For the tests, we consider three subsets of the previous IoT scenario as can be seen in Fig. 2-B. Each subset has its network peculiarities.

The Context Sharing Systems were hosted by Dell All-in-one computers. Both were configured with Ubuntu 16.04 LTS (64-bit), Quad-Core 2.8 GHz and 8GB of RAM. When using LTE, the Context Providers Systems were hosted by a cell phone configured with Android 7.0, Octa-Core 2.1 GHz and 3GB of RAM. When using ADSL, the Context Providers Systems were hosted by a Raspberry Pi 3 Model B, Quad Core

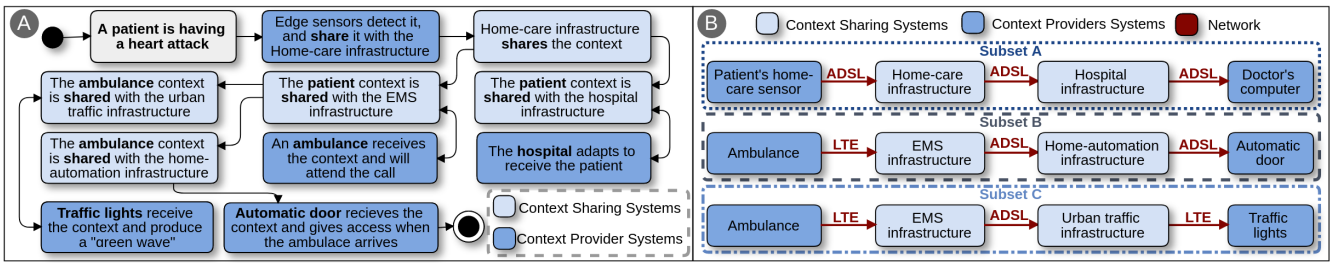

Fig. 2. (A) Flowchart for an Internet of Things application scenario. (B) Three different subsets of the application scenario.

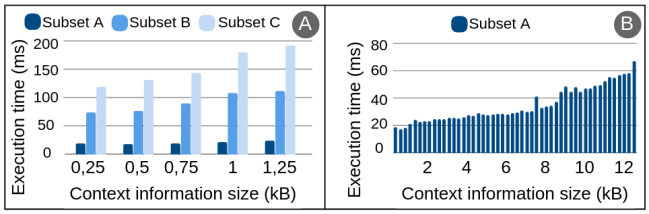

Fig. 3. Execution time in different networks (ms).

1.2 GHz and 1GB of RAM. The Context Sharing System was implemented using Java language, and the Context Provider System using C language.

C. Experiment Results

We measure the time taken for the context information be delivered from one Context Provider to another in a different domain. An example of a context information of approximately 250 bytes can be seen in Fig. 1-B. We performed two tests. In the first one, we compare the execution time of *Subset A*, *Subset B*, and *Subset C* for different context information sizes. We started from 250 bytes to 1250 bytes (see Fig. 3-A). The second one is related only to the *Subset A*, in which the size of context information has varied from 250 bytes to 12500 bytes (see Fig. 3-B). Fig. 3 shows the results (average of 30 executions for each context information size) for the simulations.

It was observed that the time taken by the architecture for share context information grows exponentially when LTE network is used. This is expected since mobile networks, as LTE, usually has bigger latency and packet loss. We consider the execution time results acceptable, mainly because the time taken for sharing context information between different domains was 192 ms (*Subset C*) for the worst case, and the best results are smaller than 20 ms (*Subset A*).

The edge-centric approach helps in the positive result, once the context information is produced in the edge of the network, avoiding the transportation of a large amount of data. Even in the worst case, the communication time for the proposed architecture to share context information over the Internet in less than 0,5 s for each of the 30 times execution, which is suitable for IoT environments. For the future, we plan to perform tests of the architecture in real-world scenarios with many vertical systems and thousands of sensors.

VI. Conclusions

Context sharing applied to IoT environments has become mandatory. Although the development of such approach is noteworthy, it is crucial to be careful with the way in which it is applied. There are two critical challenges to overcome that are missing in the existing systems. The first challenge is to deal with the large heterogeneity of IoT environments. The use and optimization of ontologies and web services can be a first step towards the mitigation of this issue. The second challenge is related to scalability and real-time sharing. There is a need to optimize mechanisms in order to minimize the data/context traffic between entities. The use of Edge Computing concept may be a way of reducing the extra information exchanged.

References

[1] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 414–454, 2014.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[3] K. Nihei, "Context sharing platform," *NEC journal of advanced technology*, vol. 1, no. 3, pp. 200–204, 2004.

[4] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of iot and cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 964 – 975, 2018.

[5] A. A. de Freitas, M. Nebeling, A. S. K. K. Ranithangam, J. Yang, and A. K. Dey, "Bluewave: Enabling opportunistic context sharing via bluetooth device names," in *Symposium on Engineering Interactive Computing Systems*, 2016, pp. 38–49.

[6] S. Cho and C. Julien, "Chitchat: Navigating tradeoffs in device-to-device context sharing," in *International Conference on Pervasive Computing and Communications*, March 2016, pp. 1–10.

[7] K. Xie, W. Luo, X. Wang, D. Xie, J. Cao, J. Wen, and G. Xie, "Decentralized context sharing in vehicular delay tolerant networks with compressive sensing," in *International Conference on Distributed Computing Systems*, June 2016, pp. 169–178.

[8] A. Manashty, J. Light, and U. Yadav, "Healthcare event aggregation lab (heal), a knowledge sharing platform for anomaly detection and prediction," in *2015 17th International Conference on E-health Networking, Application Services (HealthCom)*, Oct 2015, pp. 648–652.

[9] J. Dhallenne, P. P. Jayaraman, and A. Zaslavsky, "Rcos: Real time context sharing across a fleet of smart mobile devices," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 16th International Conference, NEW2AN 2016, and 9th Conference, ruSMART 2016, St. Petersburg, Russia, September 26-28, 2016, Proceedings*. Cham: Springer International Publishing, 2016, pp. 87–100.

[10] Princeton University, ""About WordNet." WordNet. Princeton University." 2010. [Online]. Available: http://wordnet.princeton.edu

[11] V. A. Memos, K. E. Psannis, Y. Ishibashi, B.-G. Kim, and B. Gupta, "An efficient algorithm for media-based surveillance system (eamsus) in iot smart city framework," *Future Generation Computer Systems*, vol. 83, pp. 619 – 628, 2018.