Future Directions for Providing Better IoT Infrastructure

Márcio Miguel Gomes
Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950
São Leopoldo, RS Brazil
marciobivs@gmail.com

Rodrigo da Rosa Righi
Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950
São Leopoldo, RS Brazil
rrrighi@unisinos.br

Cristiano André da Costa
Universidade do Vale do Rio dos Sinos
Av. Unisinos, 950
São Leopoldo, RS Brazil
cac@unisinos.br

UbiComp'14 Adjunct, September 13-17, 2014, Seattle, WA, USA
ACM 978-1-4503-3047-3/14/09.
http://dx.doi.org/10.1145/2638728.2638752

Abstract
Internet of Things (IoT) supports a connection between objects and humans, enabling the ubiquitous computing in our daily lives. Future research directions in IoT infrastructure should consider real-time communication and scalability to provide a better experience to the users. We justify this sentence by developing an IoT micro-benchmark, which was evaluated over a real IoT middleware. Considering the observed gaps, this article describes the ideas on redesigning the IoT infrastructure, not imposing any modifications in the users' source code. The modeling combines cloud virtualization and elasticity, service decomposition and multithreading programming. The scientific contribution of the article consists of both a novel IoT infrastructure and the algorithms to control the functioning and scalability of each component.

Author Keywords
IoT, EPCglobal, RFID, scalability, cloud computing

ACM Classification Keywords
D.4.7 [Organization and Design]: Distributed Systems.

Introduction
The idea of ubiquitous computing presupposes a strong integration among resources that are seamlessly integrated in the environment that surround us. To meet

this vision, many objects can be connected with the use of context-aware computing in an almost pervasive network. One way of enabling this idea consists of using the Internet of Things (IoT) concept, in which sensor network technologies, such as RFID, are used to exchange information without human intervention [1].

Aiming at dealing with the huge amount of data generated by IoT efficiently, GS1 proposed a standard named EPCglobal [4]. This standard defines how RFID data, $i.e$ Electronic Product Code (EPC), is collected, filtered, aggregated and then stored. To accomplish these objectives, EPCGlobal defines the ALE and EPCIS components to compose an IoT middleware. Several RFID readers can access this middleware in a real-time fashion to input data, while users query it to get up-to-date information about the objects.

The use of IoT is clearly growing in different areas, such as urban planning, environmental sensing, continuous care, intelligent shopping and home automation[2]. This direction of IoT adoption, with many readers and applications acting in parallel, impacts directly in the user experience on using this new vision of the Internet. This fact is yet more impaired when considering a centralized organization for the IoT middleware, because its lack of scalability. In this context, this article addresses a novel organization of the IoT infrastructure that is malleable in accordance with the readers and user applications demands. This infrastructure modeling represents our scientific contribution, combining concepts of multithreading, service decomposition and cloud computing elasticity and virtualization.

In proposing the infrastructure, we did not impose any modification in user's source code. Thus, representing an effortless way of obtaining performance and better experience in IoT scenarios. To emphasize the rationale of this work, the next section presents briefly some scalability experiments with the Fosstrak system[1], which emerges as one of the most used middlewares for IoT.

Analyzing Fosstrak IoT Middlewares

Fosstrak is an EPCglobal-compliant middleware in which its components commonly execute in an single machine. Considering its worldwide adoption in the IoT area, the justification of our work passes through a Fosstrak evaluation. At architectural viewpoint, the client module is in charge of launching the threads which send SOAP queries to server, synchronize them and take the elapsed time by computing $t2 - t1$ (see Figure 1). The server module runs Fosstrak in a single machine and also a monitoring system that periodically stores data related to CPU load and incoming and outcoming network traffic. The RFID system consists of Rifidi emulating a certain variety of readers and tags.

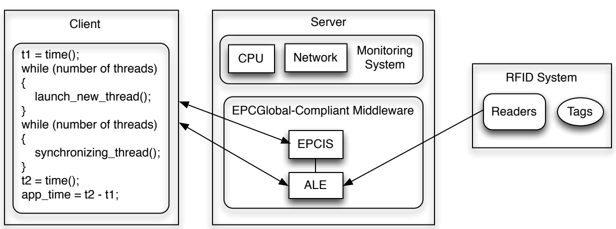

Figure 1: Fosstrak evaluation methodology

When analyzing the ALE component (for collecting and filtering data), we observe that the CPU usage and the network traffic increases in a linear rate as the number of data to be processed grows up as well, as illustrated in Figure 2 (a). ALE tends to present a constant response time with a short variation while the number of threads is

[1]http://fosstrak.org

increasing. This component has indicated a technical limitation that could process up to only 200 requests simultaneously. If a client runs 201 or more threads requesting SOAP queries, a timeout occurs and the ALE component crashes.

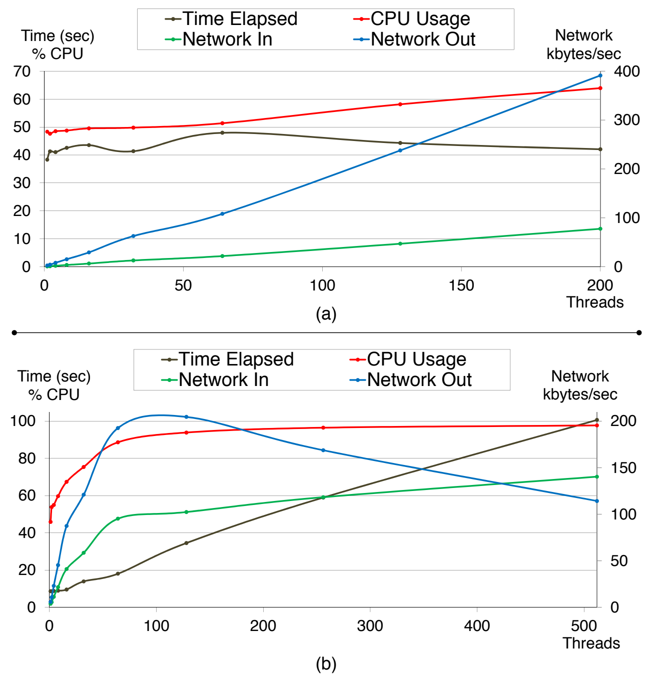

Figure 2: IoT middleware components: (a) ALE and; (b) EPCIS. We used 16 tags and requests from 16 threads

To the EPCIS component (for receiving and storing filtered data), CPU usage, network traffic and response time increase as the threads and requests are growing, as depicted in Figure 2 (b). Close to 90% of CPU load, we observe that the inclination of both the CPU and incoming network curves is not so aggressive.

Furthermore, the outbound network traffic starts to decrease suddenly when CPU usage crosses the value of 95%. Analyzing the stored data, we can see that the CPU exhaustion caused by the EPCIS affects the performance of ALE, resulting in a delayed data storage in the repository. This explains the decrease in the outbound network traffic after a high and persistent use of the CPU.

Proposal of a new IoT Infrastructure

Besides technical limitations, we observed that Fosstrak could not be the best option when executing both ALE and EPCIS together in a single machine. Thus, we are proposing a novel infrastructure (see Figure 3) with three main changes when compared to the usual deployment of an EPCGlobal middleware. First, aiming at providing better scalability, high availability and fault-tolerance, we are using a NoSQL P2P database as EPC data repository instead of MySQL. Second, we reorganize the ALE module to work with a multithreading library, so we can run faster when using a multicore server over a high parallel demand of RFID readers. Third, we redesign the EPCIS module to be deployed with virtual machines on the cloud.

At EPCIS level, firstly we model templates for EPCIS interfaces and repository. They are useful to create (or deallocate) virtual machines in accordance with elasticity algorithms. Furthermore, templates act in favor of ubiquitous computing, since provide system replication easier. As presented in Figure 3, user applications do not change and our final idea is to propose modifications at infrastructure level to offer better system usability for them. We split the "Capturing Interface" from the "Query Interface" composing different VM templates for them. This separation is necessary due to the "Capturing Interface" responsibility for storing RFID data coming from capturing applications (writing operation), while

"Query Interface" is intended to answer to users' queries (reading operations). Finally, both aforementioned subcomponents and the EPCIS repository present a moldable behavior taking profit from cloud elasticity.

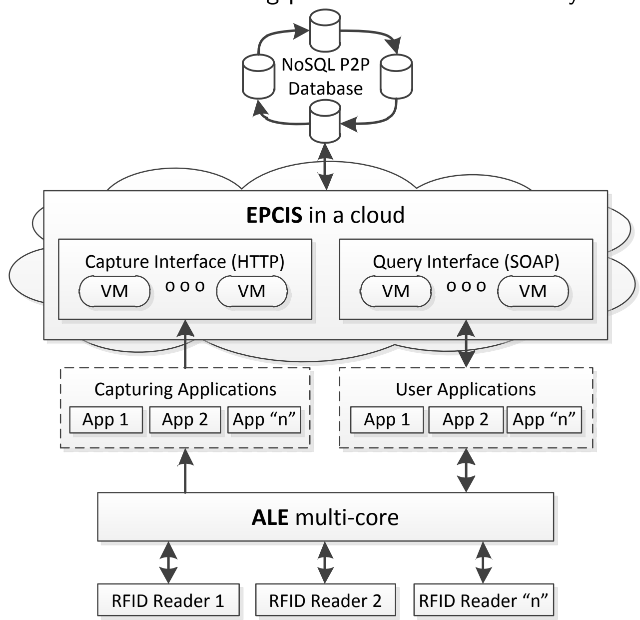

Figure 3: Proposed model for infrastructure scalability.

We plan to use reactive elasticity in the IoT scope. It normally uses rules-condition-action statements and predefined maximum and minimum thresholds for elasticity management. Periodically, the elasticity algorithm takes place and analyzes a load prediction *lp* for each evaluated metric (CPU, network, disk or memory). *lp* uses the concept of Aging, aiming to detect false-positives and false-negatives on allocation and deallocation actions. Basically, the Aging assigns a high height for the most recent observation, dividing this by 2 at each subsequent element in the time series. Our strategy can amortize the importance of peaks since an erroneous allocation will not really be needed if we analyze the historical data of the IoT system. After the testing of allocation, the same metrics are tested against the minimum threshold in order to deallocate VMs eventually. Lastly, the algorithm tries to migrate already allocated VMs between nodes (maintaining an established QoS) for consolidation purposes. Some recent cloud systems offer live-migration facility for enabling this task.

Conclusion

The concept goal of the Internet of Things is to enable things to be connected anytime, anyplace, with anything and anyone ideally using any path/network and any service. In this context, we are current developing a prototype of the proposed infrastructure using OpenNebula, pthreads and SOA.

References

[1] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems 29*, 7 (2013), 1645 – 1660.

[2] Guinard, D., Floerkemeier, C., and Sarma, S. Cloud computing, rest and mashups to simplify RFID application development and deployment. In *Proc. 2nd Int. Work. Web of Things*, WoT '11, ACM (2011), 9:1–9:6.

[3] Islam, S., Lee, K., Fekete, A., and Liu, A. How a consumer can measure elasticity for cloud platforms. In *Proc 3rd joint WOSP/SIPEW Int. Conf. Performance Eng.*, ICPE '12, ACM (2012), 85–96.

[4] Le, T. D., Kim, S. H., Nguyen, M. H., Kim, D., Shin, S. Y., Lee, K. E., and Righi, R. Epc information services with no-sql datastore for the internet of things. In *RFID, IEEE Int. Conf.* (2014), 47–54.