Demo Abstract: Fog Computing for Improving User Application Interaction and Context Awareness

Enrique Saurez, Harshit Gupta
Umakishore Ramachandran
Georgia Institute of Technology
Atlanta, Georgia
harshitg,esaurez,rama@gatech.edu

Ruben Mayer
Institute for Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
ruben.mayer@ipvs.uni-stuttgart.de

ABSTRACT

New generations of cloud applications are increasingly complex and pose lower latency requirements. The latter is forcing the industry to reduce network latency by adding computation nodes near the edge of the network, also known as Fog Computing. To utilize the Fog nodes efficiently, the dynamic placement and migration of application components must be supported. To this end, a Fog-aware application programming and deployment framework, called Foglets, has been proposed. This demonstration shows how the Foglets framework can be easily used to deploy applications in the Fog Computing infrastructure: A video streaming application allows a moving user to continue watching a video on the closest available screen.

CCS CONCEPTS

•Computer systems organization →Distributed architectures;
Sensor networks;

KEYWORDS

Fog Computing, Situation Awareness, Deployment

ACM Reference format:

Enrique Saurez, Harshit Gupta, Umakishore Ramachandran, and Ruben Mayer. 2017. Demo Abstract: Fog Computing for Improving User Application Interaction and Context Awareness. In *Proceedings of The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA USA, April 2017 (IoT/II 2017)*, 2 pages.
DOI: <http://dx.doi.org/10.1145/3054977.3057297>

1 INTRODUCTION

With the proliferation of sensors in the surrounding world, e.g., in the Internet of Things, situational-aware applications gain increasing importance. Stream processing and Complex Event Processing systems are capable of processing the sensor streams and deducing higher-level situational knowledge from the basic events [3, 5]. In doing so, they span an operator graph (or data-flow graph) between the data sources and sinks, where each operator aggregates or correlates incoming data streams and provides (higher-level) outgoing

data streams to its successor operators. To avoid network congestion and yield low latency, the placement [2] and migration [6, 7] of operators has been addressed in research.

In the emerging paradigm of Fog Computing [1], also known as Edge Computing, computational capabilities are added toward the edge of the Internet, close to the data sources and sinks. This opens new opportunities for operator placement and migration. In particular, due to the heterogeneity of Fog resources in different layers of the network, the emerging Fog infrastructure can be regarded as a hierarchical multi-tier infrastructure, from nodes with low computational capabilities at the edge up to powerful nodes in the core in cloud data-centers. Traditional (flat, non-hierarchical) deployment and migration methods [2, 6, 7] cannot fully utilize the emerging multi-tier Fog architecture. Therefore, the Foglets system [8] has been proposed.

In this paper, we describe a demonstration of Foglets for real-time context-aware application adaptation. It shows the ability of Foglets to migrate application state in a context-aware manner (e.g. knowing the relative positions of users). Migration, or application adaptation in general, needs to be performed in real time because the contexts of users change rapidly. Using a centralized entity, such as the cloud, to host this control logic could cause irritating lags or be unstable for performing the actions required by the user, due to network latencies associated to sending the state.

2 APPROACH

Our demonstration shows how we benefit from utilizing a hierarchical Fog infrastructure with the Foglets system [8] in a stream processing application.

Foglets is a system management framework for the geo-distributed environment composed of Fog nodes and Cloud. Foglets handles the incremental deployment of applications, the migration of application components due to mobility of the sensors and Fog nodes, and the discovery of other Fog nodes. Additionally, Foglets provides an API for storing and fetching data generated by the application, self-triggered migration of the application, as well as communication primitives.

The programming model of Foglets requires that the developer implements a series of handlers using the API. The four main handlers that need to be implemented are:

onSendUp: This handler processes messages coming from the nodes below in the hierarchy; responses can be sent via the **sendDown** primitive.

onSendDown: This handler processes messages coming from the parent (the next higher node in the hierarchy); responses can be sent via the **sendUp** primitive.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IoT/II 2017, Pittsburgh, PA USA

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM.
978-1-4503-4966-6/17/04...\$15.00

DOI: <http://dx.doi.org/10.1145/3054977.3057297>

onMigrationStart: This handler is called before migrations on a child (nodes lower in the hierarchy) to make sure that all the associated computations are completed and saved using the storage API.

onMigrationEnd: This handler acknowledges to the new parent node that the application was migrated correctly and contains the associated state.

3 DEMONSTRATION

3.1 General Setup

The demonstration has participants walking around a space covered by multiple screens. A video of the user's choice is playing in the closest screen. By means of this demonstration, we showcase the ability of Foglets to migrate application instances across machines in real-time based on changes in the users' context.

The demonstration setup consists of multiple screens spread across space. A participant in the demonstration selects a Youtube video playlist of their choice from a list of options. This chosen playlist is paired with a Bluetooth Low Energy (BLE) beacon that is handed over to the participant. As the participant walks in front of the screens, their playlist follows them, playing the videos on the screen that is closest, which keeps changing based on the position of the participant with respect to the screens. In each screen change, the video starts playing from where it left off in the previous screen.

Physical Setup. The infrastructure is intended to represent a Fog Computing deployment, with computational nodes (laptops) close to the users. Each node is connected to a display (screen) to play videos assigned to it. The nodes are equipped with BLE 4.0 hardware for sensing the beacons. Each node has an instance of the slave worker running; one node is additionally hosting the master worker.

Software Setup. The application running the demonstration has been built using the MobileFog programming model [4] and executes on the Foglets framework [8]. It is composed of two levels of workers, deployed on the two-tier architecture (Fog and Cloud):

- (1) **Slave workers:** This worker performs the task of sensing the received Bluetooth beacons and extracting information about the people nearby and the strength of their signal. This information, together with the list of videos playing on this node, is periodically conveyed to the master worker using **sendUp** API call. The slave is also responsible for actuation, that is playing videos on the screen attached to it. The slave maintains state corresponding to each participant it is serving - which comprises the current progress of her video.
- (2) **Master worker:** This component hosts all the intelligence involved in the application. It receives perceived Bluetooth signal strengths of beacons from all slave worker nodes. Based on this information, it determines the node a particular participant is closest to and migrates her video to that node if needed.

3.2 Context-aware Migration of User State

The master worker is aware of the contexts of each participant, where context comprises her distance from each node in the system,

Figure 1: Schematic of application under demonstration

in terms of signal strength. With reference to Figure 1, when a user moves away from the node *P*, which is displaying her video, towards another node *Q*, the master recognizes this movement and initiates a migration (by calling the Foglets API) of the user's state from the slave worker on *P* to the slave worker on *Q*. Upon migration initiation, the handler **onMigrationStart** is called on node *P*, which sends the participant's state over to node *Q*. After receiving the state, the **onMigrationEnd** handler is called on node *Q* to notify the slave of the incoming state. The slave on node *Q* uses the received state to start playing the video from the point it stopped playing on node *P*. This workflow demonstrates the ability of Foglets to seamlessly move application state across nodes in a context-aware manner.

REFERENCES

- [1] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. 2014. Fog computing: A platform for internet of things and analytics. In *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 169–186.
- [2] Valeria Cardellini, Vincenzo Grassi, Francesco Lo Presti, and Matteo Nardelli. 2016. Optimal Operator Placement for Distributed Stream Processing Applications. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems (DEBS '16)*. ACM, New York, NY, USA, 69–80.
- [3] Gianpaolo Cugola and Alessandro Margara. 2012. Processing Flows of Information: From Data Stream to Complex Event Processing. *ACM Comput. Surv.* 44, 3, Article 15 (June 2012), 62 pages.
- [4] Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. 2013. Mobile Fog: A Programming Model for Large-scale Applications on the Internet of Things. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing (MCC '13)*. ACM, 15–20.
- [5] Ruben Mayer, Boris Koldehofe, and Kurt Rothermel. 2015. Predictable Low-Latency Event Detection With Parallel Complex Event Processing. *IEEE Internet of Things Journal* 2, 4 (Aug 2015), 274–286.
- [6] Beate Ottenwälder, Boris Koldehofe, Kurt Rothermel, and Umakishore Ramachandran. 2013. MigCEP: Operator Migration for Mobility Driven Distributed Complex Event Processing. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems (DEBS '13)*. ACM, New York, NY, USA, 183–194.
- [7] Beate Ottenwälder, Ruben Mayer, and Boris Koldehofe. 2014. Distributed Complex Event Processing for Mobile Large-scale Video Applications. In *Proceedings of the Posters & Demos Session (Middleware Posters and Demos '14)*. ACM, New York, NY, USA, 5–6.
- [8] Enrique Saurez, Kirak Hong, Dave Lillethun, Umakishore Ramachandran, and Beate Ottenwälder. 2016. Incremental Deployment and Migration of Geo-distributed Situation Awareness Applications in the Fog. In *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems (DEBS '16)*. ACM, New York, NY, USA, 258–269.