A Blockchain-based Brokerage Platform for Fog Computing Resource Federation

Marco Savi[1], Daniele Santoro[1], Katarzyna Di Meo[1], Daniele Pizzolli[1], Miguel Pincheira[1], Raffaele Giaffreda[1], Silvio Cretti[1], Seung-woo Kum[2], Domenico Siracusa[1]

[1]Fondazione Bruno Kessler, Trento, Italy, [2]Korea Electronics Technology Institute, Seoul, Korea

***Abstract*—This demonstration aims at showcasing an initial version of the DECENTER Brokerage Platform, which leverages an Ethereum blockchain to enable resource federation among different Fog Computing infrastructures. We consider a scenario where an Italian Infrastructure Provider wants to seamlessly extend its pool of resources to get access to an IP camera located in Korea, so that it can deploy an application to locally perform text recognition from a live video stream.**

I. Introduction

Fog Computing [1] has been proposed as an effective means to bridge the gap between Cloud and IoT, making computing resources more easily available closer to where data is produced. While commercial deployments of Fog Computing Platforms already exist (e.g. [2]), such solutions are bound to work within a single administrative domain as they presume direct ownership of the distributed resources. As a result, complex interactions are required to operate federation of resources across administrative domains when any of the federated providers need resources at the edge in a location where they do not have an infrastructure deployed [3].

The DECENTER project [4] aims at addressing this shortcoming by creating a federated Fog Computing ecosystem where computing and IoT resources (processing, memory, storage, connectivity, sensing, actuating, etc.) can be harmoniously orchestrated in dynamically-created federated environments. DECENTER proposes a blockchain-based *Brokerage Platform* incentivising resource sharing across administrative domains through the use of Smart Contracts (SCs).

This demonstration showcases a first version of the DECENTER Brokerage Platform: to show the potential of our solution, we extended the functionalities of FogAtlas [5], a Fog Computing Platform based on Kubernetes (K8s) and designed for single-domain resource orchestration and application deployment. The extension integrates the Brokerage Platform through which spare resources can be advertised, selected and eventually paid for (after use) using blockchain cryptocurrencies. Providers that select some of the resources on offer will "self-provision" them over their own infrastructure that is managed by a local instance of FogAtlas. Specifically, in the demonstration we consider two providers, a seller (advertising resources located in Korea) and a buyer (located in Europe but willing to rent advertised resources) and we show, by means of a simple Graphical User Interface (GUI), how the buyer can easily extend its pool of resources. The specific example shows the deployment of a simple Optical Character Recognition (OCR) application where the buyer requires access to resources (an IP camera and processing power) from outside its administrative domain.

The remainder of this paper is organized as follows. Section II reports the architectural view of the proposed solution, Section III recalls the demonstration setup and workflow, while Section IV concludes the paper.

II. Architecture and Behavior

Figure 1 shows the high-level architecture of the software framework used in this demo. Two platforms are envisioned:

- **Fog Computing Platform:** Manages a Fog Computing infrastructure and handles requests for the deployment of microservice-oriented applications in a single administrative domain. We adopt FogAtlas [5], which leverages and extends open-source technologies (OpenStack, K8s, Docker, Ansible, ZeroTier, etc.) to seamlessly work from Cloud to things. Figure 1 shows two Fog Computing Platforms, managing infrastructures of two distinct administrative domains, namely Infrastructure Provider A (pink) and Infrastructure Provider B (green)[1].
- **Brokerage Platform:** Handles applications' deployment requests when different administrative domains are involved, i.e., when an application requires the usage of fog resources belonging to another Provider. A blockchain-based toolset (i.e., Ethereum [6]) is adopted for resource advertisement, negotiation and federation. In Fig. 1 the Brokerage Platform modules are highlighted in yellow.

A. Modules and interfaces

The main components of the Fog Computing Platform are:

- **App Composer GUI:** Used to model microservice-oriented applications and specify computational and networking requirements for microservices and related data flows.
- **Orchestrator:** Finds the best placement for applications' microservices, while meeting specified requirements. It is an extension of the K8s Scheduler.
- **IaaS Manager:** Offers a generic interface to the underlying infrastructure that can be customized.
- **Resource Manager:** Discovers, reserves and frees resources on the underlying infrastructure.
- **Infrastructure Provisioning:** Sub-component of the Resource Manager that enables the provisioning of new resources on the underlying infrastructure.

[1]Only two Infrastructure Providers (from now on simply "Providers") are considered in this demonstration, but more could be involved in a real scenario.

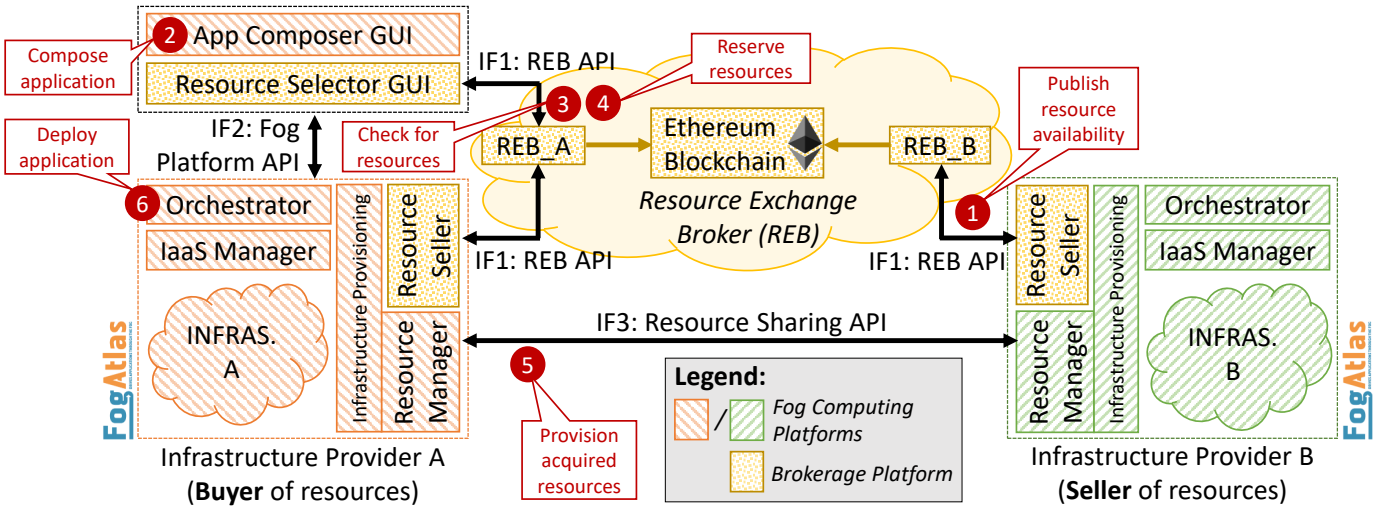

Fig. 1. Architectural view and behavior specification

The main components of the Brokerage Platform are:

- **Resource Exchange Broker (REB):** blockchain-based "conveyor belt", where different Providers can commit some resources for sharing with others. Sharing is regulated through SCs recorded in an Ethereum blockchain, which are submitted through a set of Application Programming Interfaces (APIs) distributed above each Fog Computing Platform (i.e., *REB_A* and *REB_B*) and used to enable communication with the blockchain.
- **Resource Selector GUI:** Click-based interface able to show resources advertised on the REB and used to select the most appropriate ones to meet application requirements.
- **Resource Seller:** Advertises available resources on the REB and implements the negotiation process to acknowledge a reservation request made by the Resource Selector GUI.

Three main interfaces have been designed and implemented:

- **REB API (IF1):** It allows (i) the Resource Seller to advertise resource availability on the REB and (ii) the Resource Selector to retrieve information about advertised resources and make reservations on them.
- **Fog Platform API (IF2):** Used (i) for applications' deployment and (ii) to request provisioning of third-parties rented resources on the underlying infrastructure.
- **Resource Sharing API (IF3):** Used by a Provider for automated resource provisioning of resources belonging to another Provider.

B. Behavior specification

Figure 1 reports the high-level behavior specification of this demonstration, where Provider A acts as *buyer* and Provider B as *seller* of resources. Six steps are executed:

1) *Publish resource availability:* Resource Seller of Provider B advertises available resources on the REB.
2) *Compose application and send it for deployment:* Through the App Composer GUI, a user (e.g. a service provider) models her application and sends its descriptor for deployment to Provider A through Fog Platform API.
3) *Check for available resources:* Advertised resources on the REB are retrieved through REB API and displayed by the Resource Selector GUI of Provider A. The most appropriate to meet application's requirements (i.e., resources advertised by Provider B) are selected.
4) *Reserve resources:* Resource Selector GUI requests to Provider B the reservation of chosen resources through REB API and the creation of a SC. Provider B acknowledges it via REB by amending the created SC.
5) *Provision acquired resources:* Infrastructure Provisioning module of Provider A starts the automated provisioning process through Resource Sharing API to include the rented resources of Provider B in its infrastructure.
6) *Deploy the application:* Orchestrator of Provider A deploys the application on the extended infrastructure.

III. Demonstration Setup and Workflow

This section describes how the demonstration environment has been set up and how its execution is carried out.

A. Environment and setup

The demonstration environment includes two Fog Computing infrastructures, one managed by Provider A and the other by Provider B. Both are composed of two *regions*: a Cloud region and an Edge region. The former infrastructure, called Infrastructure A for the sake of brevity, is located in Italy, while the latter (Infrastructure B) is located in Korea. The characteristics of the infrastructures are shown in Table I.

We use a simple microservice-oriented application to showcase the functionalities of the Brokerage Platform. It collects a video stream from an IP camera, extracts the text embedded in the images thanks to an OCR open-source library [7] and stores the text into a repository for future consultation through a web interface. The application, called *simple_ocr*, is composed of three microservices that are chained in sequence:

- **Processor:** Thanks to an OCR library, it extracts the text from the streamed images coming from the IP camera.
- **Repository:** Key-value store where extracted text is stored.

TABLE I
CHARACTERISTICS OF INVOLVED INFRASTRUCTURES

	Infrastructure A		Infrastructure B	
Feature	**Cloud reg.**	**Edge reg.**	**Cloud reg.**	**Edge reg.**
Name	CLOUD	POVO	SEOUL	SEOUL-KETI
CPU	6 vCPU	1 CPU	4 vCPU	2 CPU
RAM	6 GB	1 GB	4 GB	2 GB
Disk	500 GB	10 GB	500 GB	10 GB
Devices	None	IP camera (*cam1*)	None	IP camera (*camkor1*)
SW	OS, K8s	Ubuntu, K8s	OS, K8s	Ubuntu, K8s

- **Webserver:** Web application that allows to access text stored in the repository.

This application can benefit from having the Processor microservice deployed close to the data source (i.e., the IP camera) in an Edge region, while keeping CPU/RAM-hungry microservices (i.e., Repository, Webserver) in the Cloud region. In fact, (i) such a deployment reduces the requested bandwidth between Edge and Cloud and (ii) the video stream is processed locally (relevant for privacy-sensitive data).

B. Storyboard and workflow

The initial status of Infrastructures A and B is shown in Fig. 2(a) and 2(b) respectively. On Infrastructure A, *simple_ocr* has been already deployed: its microservices are identified by purple bullets. Note that the Processor microservice has been deployed on the POVO edge region. Infrastructure B has instead no applications deployed and is totally disconnected to Infrastructure A. Green and blue bullets in the CLOUD and SEOUL regions (i.e., Cloud regions) represent all deployed microservices of Brokerage and Fog Computing Platforms.

In the current situation, *simple_ocr* can gather and process the video stream from the camera located in the POVO region (*cam1*), while it clearly cannot access the video stream produced by the camera in the SEOUL-KETI edge region (*camkor1*). Granting access to the Korean IP camera, following the typical modes of operations, would involve human interaction, would need networking and infrastructural knowledge and is time consuming. In fact, the owner of the application should autonomously negotiate with Provider B the renting of needed resources, struggle for setting up a secure connection between the remote region and Infrastructure A and finally agree with Provider B on the deployment of part of her application (i.e. the Processor microservice) close to *camkor1*.

Thanks to the Brokerage Platform, Provider A can instead embrace the "one-stop-shop" business, offering the possibility to deploy automatically (part of) the application on resources belonging to Infrastructure B. The needed resources are reserved and rented through the Resource Selector GUI, which gets available resources through the REB (by reading advertised resources on the Ethereum blockchain), stipulates a SC with Provider B and provisions reserved resources through its Infrastructure Provisioning component. The acquired pool of resources is also automatically interconnected with Infrastructure A through a Virtual Private Network (VPN). All the needed software (e.g. K8s) is installed and finally the pool of resources joins Infrastructure A cluster in the form of a new K8s worker node. All this complex process is triggered by simply clicking on a button on the Resource Selector GUI.

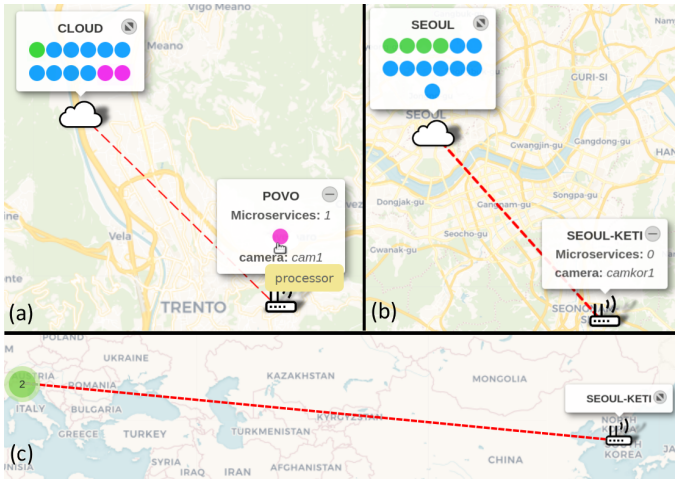

Fig. 2. Geographical representation of Infrastructure A (a), Infrastructure B (b) and Infrastructure A extended with resources from Infrastructure B (c)

The final result of the process is shown in Figure 2(c). On such an extended infrastructure, spanning from Italy to Korea, it is now possible to deploy a new instance of the OCR application to process the video streamed by *camkor1*, by placing a new instance of the Processor microservice in the SEOUL-KETI region. Once this microservice has been deployed, the Repository located in Italy can store the extrapolated text as sent from the Processor located in Korea.

IV. CONCLUSION

We designed an architecture where multiple Fog Computing Platforms, each managing a distributed infrastructure, can seamlessly federate their resources by means of a blockchain-based Brokerage Platform any time access to resources from a different administrative domain is needed. A proof-of-concept has been set up where an Infrastructure Provider, located in Italy, wants to extend its infrastructure by acquiring computational resources and gaining access to an IP camera located in Korea, with the aim of recognizing text from the camera-produced video stream while keeping the video processing operations local (e.g. for privacy reasons). In the future, we will extend our work to include negotiated Service Level Agreements (SLAs) among providers in the stipulated SCs and to allow an automated verification of those SLAs before payments for the usage of rented resources occur.

ACKNOWLEDGMENT

This work has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 815141 (DECENTER project).

REFERENCES

[1] A. V. Dastjerdi and R. Buyya, "Fog Computing: Helping the Internet of Things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.
[2] "Amazon AWS IoT Greengrass," https://aws.amazon.com/greengrass/.
[3] A. Boubendir, F. Guillemin, C. Le Toquin *et al.*, "Federation of Cross-Domain Edge Resources: A Brokering Architecture for Network Slicing," in *IEEE NetSoft*, 2018.
[4] "DECENTER project," https://www.decenter-project.eu/.
[5] "FogAtlas," https://fogatlas.fbk.eu/.
[6] G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, pp. 1–32, 2014.
[7] "Gosseract OCR," https://github.com/otiai10/gosseract.