Collaborative Real-Time Business Communication Platform

Guadalupe Flores
Wellness Telecom Seville, Spain
Email: gflores@wtelecom.es

Ignacio Campos
Wellness Telecom Seville, Spain
Email: icampos@wtelecom.es

Jorge Perez
Wellness Telecom Seville, Spain
Email: jmperez@wtelecom.es

Spiros Koulouzis
University of Amsterdam Amsterdam, Neederland
Email: S.Koulouzis@uva.nl

***Abstract*—In this paper is presented a collaborative real-time business communication platform. This platform used the software workbench for Interactive, Time Critical and Highly self-adaptive cloud applications (SWITCH), improving development productivity, deployment efficiency and reducing operational costs. The pilot has been integrated with the SWITCH system, where the deployment and monitoring of the platform is shown.**

1. Introduction

The global spreading of Internet, along with the amount of devices internet connected, formed the need for IT mobility that drive to the emergence of Cloud [1]. In this context, the cloud infrastructures can provide virtualization, elasticity and high-quality services. Nevertheless, for real time critical applications the cloud environments can not support the early demand of time critical applications [2]. Within this frame, SWITCH project arises, a platform for time critical application in the cloud providing programmability and controllability of QoS and QoE for cloud environments [3]. In this sense, the requirements for time critical application are analyzed by real uses cases. In this paper we present a use case running on the cloud. The collaborative real-time business communication platform, also called Unified Communication as a Service (UCaaS) to provide cloud interoperability among different clouds.

The main novelty of this communication system is the adaptability of the service on the traffic demand while maintaining the quality of service QoS [4]. The SWITCH workbench assure this QoS by implementing efficiently customer requirements, deploying flexibly software and maintaining the run time system quality. In addition, the UC platform makes use of Docker containers [5], increasing the deployment time of the applications and opening the possibility of scaling on demand.

This paper is structured as follows, first the use case is described, with the different plausible scenarios. Secondly, the architecture of the system is shown, where the different hosts and dockers are explained. Thirdly, the integration with the SWITCH platform is illustrated, where the scalability on demand of the system is analyzed. Later on the the first results of the integration and management of the use case by SWITCH are pointed out and finally, the conclusion of the real-time business communication platform are summarized.

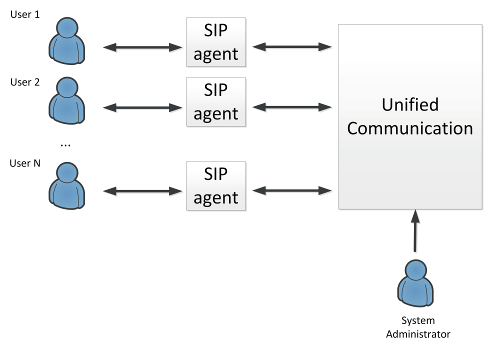


Figure 1. Use case Diagram.

2. Description

In this section will be described the pilot presented: A collaborative real-time business communication platform. The pilot can be defined as a platform to provide enterprise communications integrated in only one service. This service embraces the communication between two users or among a group of users. The services offers by the platform are presence detection, instant message service (chat), message delivery service (delayed message service), audio and video calls. The platform is thought to be used in a business close environment where the users are the employees of the company and the administrator of the system gives access to the service. The administrator has the control of the system. Once the users get access from the administrator, the users can access to the platform through a SIP client.

The different functionalities can be reached through a SIP agent installed in the a device. When a user wants to connect to the unified communication platform, it needs to initiate the process through the SIP agent. An schema of the platform is illustrated in Fig. 1.

Inside the functionalities, various scenarios can occur. The different scenarios are summarized as follows.

- **Peer to Peer audio call:** An users who wants to call other user of the system starts the call through the SIP agent. If the system recognize the receiver, a negotiation of message begins before the communication is established. Between those messages, the audio codec is set. In this way, the system does not have to process the audio.
- **Peer to peer video call:** The procedure is the same than the peer to peer audio call with the difference of having two flows: one for audio and other for video. As it happened in the previous scenario, the codec for audio and video are set during the negotiation protocol.
- **Multi-conference audio call:** The administrator needs to create a logical space where the user can hold an audio conference, this logical spaces are called rooms. When the users wants to access to the conference, the administrator provides the conference room and the conference is initiated through a SIP agent in the same way of peer to peer call.
- **Multi-conference video call:** In this case, the procedure is the same as in multi-conference audio call but with two flows one from audio and other for video.
- **Chat and message delivery service:** This functionality varies from the previous described. This lets users to share instant messages in real time (chat), as well as send files to other users if they are not connected at the moment. The SIP agent will be responsible for this detection. Moreover, the SIP agent will show the status of different contacts to any user.

3. Architecture

In this section is presented the architecture of the use case with the different hosts (VMs) and Docker containers. The platform comprises several services which should be configured to provide the complete set of functions. These services, needed for the use case, are running in different Docker containers, Fig. 2. The Dockers technology is an abstraction at the app layer that package code and dependencies together. Containers take up less space than VMs and are isolated from its surrounding, avoiding conflicts between different applications. Dockers offer the possibility of scaling the containers if more resources are needed, this property will be used by the use case and will commented later on the section 4. The architecture of the use case with the different Docker hosts and containers can be seen in Fig. 3. The use case is composed of different services which are listed below:

- **Kamailio:** This is a very flexible open source SIP server, which can be used to build complex real time communication platforms; even it can also support Presence and Instant Messaging functions. This service can also act as a load balancer of VoIP and video calls along with other servers. In the use case Asterisk and Doubango servers will be used as VoIP and video server respectively while Kamailio will be responsible for the complete signalization of the protocol.
- **MySQL:** An open-source management system for relational databases based on Structured Query Language. In this case, a MySQL database will be used to store registered users in the system, as well as calls registry and another dataset.
- **RTP Engine:** This service allows media data flows to circulate through itself supported by the RTP protocol. The data flows can be both audio and video flows.
- **Asterisk:** This service is an open-source initiative converted in a universal tool for building audio communication application acting as a communication server and handling data along different communications protocol. This platform is an engine to power IP PBX (Private Branch Exchange), IVR (Interactive Voice Response), ACD (Automatic Call Distributor), etc. Its specific function in the use case is to process VoIP calls according to signalization in the system.
- **Doubango:** This service is an open-source SIP TelePresence System. It is used as a video mixer used for shaping various streams at conferences
- **Web Portal:** This service will expose a website that

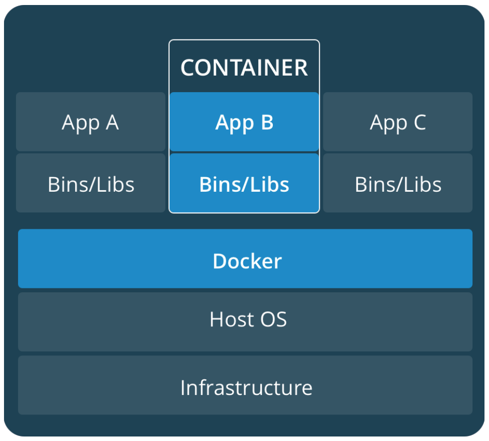


Figure 2. Docker container Diagram.

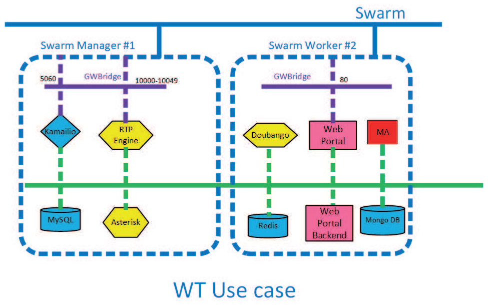


Figure 3. Architecture of the platform.

will offer access to the communications system without installing additional software in user devices. This portal will provide an agent (SIP agent) which will be offered to be installed in the browser and will let anybody registered in the system use the service without having any other program installed in his device.

- **Mongo DB:** It is a open-source cross-platform document-oriented database program. Mongo is the data base for the web portal.
- **Redis:** It is an open source (BSD licensed), in-memory data structure store, used as a database, cache and message broker. It is the data base for Doubango video mixer.
- **Monitoring Agent (AG):** It collects the information based on statsd. In all the containers there is a small agent that sends the metrics to the Monitoring agent. This monitor agent gathers all the information and sends those to the monitor server based on JCatascopia. These information is stored in Cassandra, a big data database.

4. Scalability: SWITCH integration

The Unified communication pilot is a real-time critical application which depends on the the load demands of the system. In order to meet with customers requirements and QoS, the system is design to cope with two types of scalability: vertical and horizontal. The vertical scalability is guarantee using Docker [6], it is a dynamic scalability, where each container take the resources needed from the host. However, this resource allocation is limited to the host resources and in some cases, a new application container is needed. In this sense, the UC pilot together with SWITCH subsystem have been designed to automatically perform vertical scaling and if needed, horizontal scaling.

SWITCH systems is designed to guarantee the traffic demand of the pilot while maintaining the proper operation of the system no matter the work load of the pilot. First, the SIDE subsystem allows developers to define the system, at container level with QoS requirements to describe the system. This user interface establishes a common ontology which can be used for different subsystems inside the service or even different services. Second, the DRIP subsystem checks the resources needed for the service before starting execution and deploy the pilot in the different VMs. Moreover, if application must be scaled up, DRIP will provision new resources in a suitable cloud to host new containers while maintaining QoS. Finally, ASAP is responsible to monitor metrics and resources remaining the proper operation of the system as well as QoS of the service by means of probes which will be deployed in the same host as containers.

In the Table 1 are summarized the relation between services and the type of scalability and the metrics affected.

TABLE 1. QoS IN UNIFIED COMMUNICATION PLATFORM

Resource	Type	Metric
Kamailio	Vertical	CPU metrics, Memory metrics
MySQL	Vertical	CPU metrics, Memory metrics
RTP Engine	Horizontal	Number of ports
Asterisk	Horizontal	Number of rooms and users
Doubango	Horizontal	Number of rooms and users
Mongo DB	Vertical	CPU metrics, Memory metrics

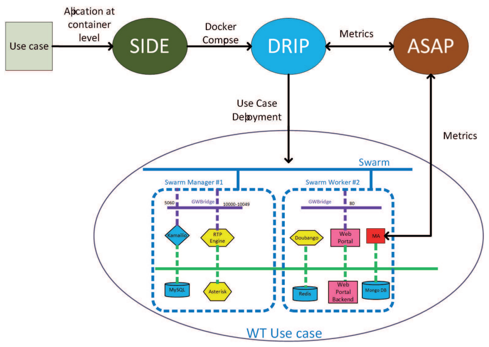


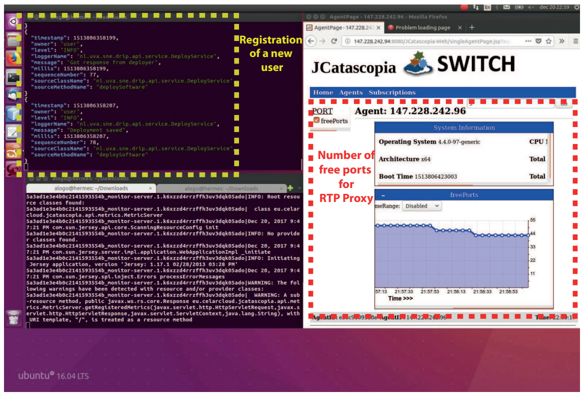
Figure 4. Integration of the platform with SWITCH.

5. Results: Use case deployment from DRIP

The objectives of the pilots related to the SWITCH environments are the proper operation of the system, assuring the scalability of the system. In the Fig. 4 are presented the integration of the use case with switch environment. In this section is tested the deployment of the pilot through the DRIP and the monitorization of the metrics for the scaling under demand.

In the SIDE, the different configuration for the pilot are defined, and the Tosca with the docker compose file is generated. This docker compose file is the one represented in fig Fig. 3. This Tosca file is acquire by the DRIP subsystem and in it contains the information needed in terms of VM and containers for the deployment of the system. With this information, DRIP through SWARM deploys the pilot with the different containers of the use case. Between this containers is located the monitoring agent (MA) which DRIP internally communicate with the Monitoring Server that has been already deployed ins ASAP. In this case, all the metrics mentioned in Table 1 are stored in the Cassandra database of monitoring server.

In this paper, the peer to peer video call with the monitoring of the ports is presented. DRIP is able to deploy the system with the docker compose file of the pilot. DRIP connect the pilot with the Monitor Server deployed in the ASAP. The monitor agent (MA) of the use case send the metrics for the RTP Proxy, the number of free port, to the monitor server deployed in the system. When a new user want to interact with the system, a new port is associated

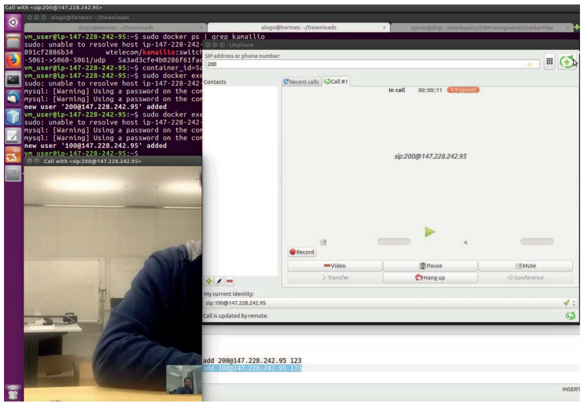


```
}
{
  "timestamp": 1513806358199,
  "owner": "user",
  "level": "INFO",
  "loggerName": "nl.uva.sne.drip.api.service.DeployService",
  "message": "Got response from deployer",
  "millis": 1513806358199,
  "sequenceNumber": 77,
  "sourceClassName": "nl.uva.sne.drip.api.service.DeployService",
  "sourceMethodName": "deploySoftware"
}
{
  "timestamp": 1513806358207,
  "owner": "user",
  "level": "INFO",
  "loggerName": "nl.uva.sne.drip.api.service.DeployService",
  "message": "Deployment saved",
  "millis": 1513806358207,
  "sequenceNumber": 78,
  "sourceClassName": "nl.uva.sne.drip.api.service.DeployService",
  "sourceMethodName": "deploySoftware"
}
```

```
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|INFO: Root resou
rce classes found:
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|  class eu.celar
cloud.jcatascopia.api.metrics.MetricServer
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|Dec 20, 2017 9:4
7:21 PM com.sun.jersey.api.core.ScanningResourceConfig init
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|INFO: No provide
r classes found.
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|Dec 20, 2017 9:4
7:21 PM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|INFO: Initiating
 Jersey application, version 'Jersey: 1.17.1 02/28/2013 03:28 PM'
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|Dec 20, 2017 9:4
7:21 PM com.sun.jersey.spi.inject.Errors processErrorMessages
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|WARNING: The fol
lowing warnings have been detected with resource and/or provider classes:
5a3ad1e3e4b0c2141593554b_monitor-server.1.k6xzzd4rrzffh3uv3dqk05ado|  WARNING: A sub
-resource method, public javax.ws.rs.core.Response eu.celarcloud.jcatascopia.api.met
rics.MetricServer.getRegisteredMetrics(javax.servlet.http.HttpServletRequest,javax.s
ervlet.http.HttpServletResponse,javax.servlet.ServletContext,java.lang.String), with
 URI template, "/", is treated as a resource method
```

Operating System 4.4.0-97-generic
Architecture x64
Boot Time 1513806423003

Figure 5. RTP Proxy ports monitoring



```
vm_user@ip-147-228-242-95:~$ sudo docker ps | grep kamailio
sudo: unable to resolve host ip-147-228-242-
891cf2886b34        wtelecom/kamailio:switch
-5061->5060-5061/udp   5a3ad3cfe4b0286f61fac
vm_user@ip-147-228-242-95:~$ container_id=5a
vm_user@ip-147-228-242-95:~$ sudo docker exe
sudo: unable to resolve host ip-147-228-242-
mysql: [Warning] Using a password on the com
mysql: [Warning] Using a password on the com
new user '200@147.228.242.95' added
vm_user@ip-147-228-242-95:~$ sudo docker exe
sudo: unable to resolve host ip-147-228-242-
mysql: [Warning] Using a password on the com
mysql: [Warning] Using a password on the com
new user '100@147.228.242.95' added
vm_user@ip-147-228-242-95:~$
```

SIP address or phone number: 200

In call 00:00:11

sip:200@147.228.242.95

My current identity: sip:100@147.228.242.95

Call is updated by remote.

```
add 200@147.228.242.95 123
```

Figure 6. Peer to peer video call.

to the user and the number of free ports register by the monitor server decrease. In Fig. 5 is shown the result from the process, in yellow dashed line are represented when the users are being registered into the system and in red dashed line, are represented the number of free ports used by the system. As it can be seen these are decreasing in time. In addition, the proper operation of the system through DRIP is illustrated in Fig. 6. The user through the SIP client, can initate a video conference, mantaining a QoS during the whole call.

The next step for the total monitoring will be to set a rule for the metrics and the scale of the containers. Nevertheless, these steps will be a simple task due to the more complex part, the deployment and integration of the use case by SWITCH has already been demonstrated.

6. Conclusion

In this paper is presented a collaborative real-time platform for audio, video and chat for business environment. These platform is designed to be used in the Cloud and thanks to the Docker technology used, the container provides standards and can be deployed in any Cloud technology. The main novelty of the approach is the scalability of the system on demand. The SWITCH workbench interacts with the platform assuring the QoS with a scalability of the necessary components. In the paper the work of integration, deployment and monitoring of the use case by SWITCH has been proved. This system has been tested with one of the scenarios, peer to peer video conference, showing the feasibility of the system, maintaining the QoS. This work leads the way to the final task of the SWITCH project, the total integration of the use case where the scalability of on demand is assured.

Acknowledgments

This work has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 643963 (SWITCH project).

References

[1] Jörg Domaschka, Frank Griesinger, Daniel Baur, and Alessandro Rossini. Beyond mere application structure thoughts on the future of cloud orchestration tools. *Procedia Computer Science*, 68:151–162, 2015.

[2] Keith Jeferry, George Kousiouris, Dimosthenis Kyriazis, Jörn Altmann, Augusto Ciuffoletti, Ilias Maglogiannis, Paolo Nesi, Bojan Suzic, and Zhiming Zhao. Challenges emerging from future cloud application scenarios. *Procedia Computer Science*, 68:227–237, 2015.

[3] Zhiming Zhao, Paul Martin, Junchao Wang, Ari Taal, Andrew Jones, Ian Taylor, Vlado Stankovski, Ignacio Garcia Vega, George Suciu, Alexandre Ulisses, et al. Developing and operating time critical applications in clouds: the state of the art and the switch approach. *Procedia Computer Science*, 68:17–28, 2015.

[4] Salman Taherizadeh, Ian Taylor, Andrew Jones, Zhiming Zhao, and Vlado Stankovski. A network edge monitoring approach for real-time data streaming applications. In *International Conference on the Economics of Grids, Clouds, Systems, and Services*, pages 293–303. Springer, 2016.

[5] Docker. `http://https://www.docker.com/what-container`.

[6] Ann Mary Joy. Performance comparison between linux containers and virtual machines. In *Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in*, pages 342–346. IEEE, 2015.