Optimization of avatar's life cycle in a cloud infrastructure

LIRIS - Université Claude Bernard - Lyon 1

09 september 2016

Contents

List of Figures

List of Tables

1 Introduction

Nowadays the web is the major medium of communication between humans and applications. Moreover, web services have been proven to be indispensable for interoperable applications. Existing web technologies and protocols are used to unify the logical and the physical world. WoT (Web of Things) is a non traditional view of the IoT (Internet of Things). IoT is a network of different physical objects that enables these objects to collect and exchange data. WoT enables a communication between connected devices using a same language, in order to communicate and interoperate freely on the Web. In addition, WoT extends the IoT by considering that each physical object can be accessed and controlled using Web-based languages and protocols.

An avatar is an important element of the WoT and is built on a software platform that addresses most of the challenges imposed by the WoT. It is a software artifact dedicated to a particular physical object that represents the virtual part of its corresponding cyber-physical object. It has several characteristics, possesses a complex component-based architecture, allowing to add intelligence to objects behaviors by taking into account additional information available from other environments . An avatar is hosted in the cloud and can be migrated between cloud infrastructures to ensure a better service for users. Through this work we present a method for avatar migration between cloud infrastructures. To decide if migration is more beneficial, a model of criteria has been proposed to calculate the cost of the migration.

This report consists of several parts and is divided into several sections as follows. The first section presents the work environment and concepts while describing the avatar, its architecture and features. It explains also the functioning of the avatar. Then the second section focuses on the research problem, the motivation scenario and the state of the art of migration techniques. The third section describes the model of criteria and how it was implemented in our solution to evaluate the migration. In addition, it describes finely the migration technique developed during the internship. Finally, the last section presents the results obtained and the interpretation of these results.

2 Environment and Concepts

Through this section, we present a brief description of the environment and the platform used during our study.We start with a definition of the Web of Things, on which avatars are instantiated. Then, we present the avatar, its features, architecture and life-cycle. We finish with the types of migration, the research problem of the internship and the motivation scenarios.

2.1 Web of Things

Nowadays, the web is the major medium of communication between humans and applications. Moreover, web services have been proven to be indispensable in order to create interoperable applications. Existing web technologies and protocols will be used to unify the www world and the physical world. WoT (Web of Things) is a non-traditional view of the IoT (Internet of Things) that enables a communication between connected devices using a same language, this way interconnected objects can communicate and interoperate freely on the Web [1]. The WoT extends the IoT by considering that each physical object can be accessed and controlled using Web-based languages and protocols [2]. WoT runtime environment imposes some requirements. In what follows some of the requirements are mentioned [2] [3].

- Live reactivity allows the platform to adapt its behavior and structure to its environment at runtime.
- Resource management estimates the global cost of physical actions in terms of different metrics (device usage, computation and networking).
- Interoperability to ensure that WoT platform is able to automatically discover and interact with heterogeneous physical objects.
- Disconnection tolerance allows the platform to support connectivity disruptions occurring between mobile objects.
- Safety ensures that the platform is reliable and secure. This way objects and applications are harmless.
- Delegation allows identifying the most suitable location to execute code module and deploying these modules on the object processing unit or on a cloud infrastructure.
- Collaboration allows a set of objects to exhibit a collective behavior, in order to accelerate the processes.

2.2 Avatars

Avatars, important elements of the WoT, are a central piece in the ASAWoO project (https://liris.cnrs.fr/asawoo/). The avatar is built on a software platform that addresses most of the challenges imposed by the WoT. An avatar is a software artifact dedicated to a particular physical object and represents the virtual part of its corresponding cyber-physical object [2]. This component has several characteristics, it possesses a complex component-based architecture, allowing to add intelligence to objects behaviors by taking into account additional information available from other Web sources [3]. In addition avatars can be defined as systems that allow supporting a fast, scalable, reliable, and energy efficient distributed computing over mobile devices by leveraging cloud resources [4]. Finally, it provides hardware vendors, software developers and end-users with a comprehensible abstraction that makes physical objects accessible on the Web. All the characteristics mentioned are supported through different features. In this section we will define avatar's features, then avatar's architecture and finally avatar's life-cycle.

2.2.1 Avatar architecture

The avatar runtime designed on a felix Apache server as an OSGi service-oriented architecture is decoupled from its logical architecture. The fact of having a decoupled logical architecture offer the possibility to adapt the avatar on different types of objects dynamically and to distribute avatar's services to different locations. [3] The avatar architecture is formed by different modules with a specific role for each one. Figure 1 shows the architecture of the avatar, where services are grouped into functional module.

The first module is the Core module. This module includes central components that are reused in different steps of the lifecycle of the avatar. The reasoner is used by the local functionality manager from the collaboration module. It reasons about knowledge representations. Local cache speeds up data exchange between services. Component Deployment Manager decides where and when to deploy other components in the architecture [3]. Web service module is formed by two component. The WoT Application Server that exposes functionalities available as applications. The HTTP Client that allows an interaction between the avatar and an external Web service available on the Web or applications provided by other avatars. In addition this module through its component is in charge of implementing the inter-avatar negotiation processes. [2][3] WoT Application module, through this mod-

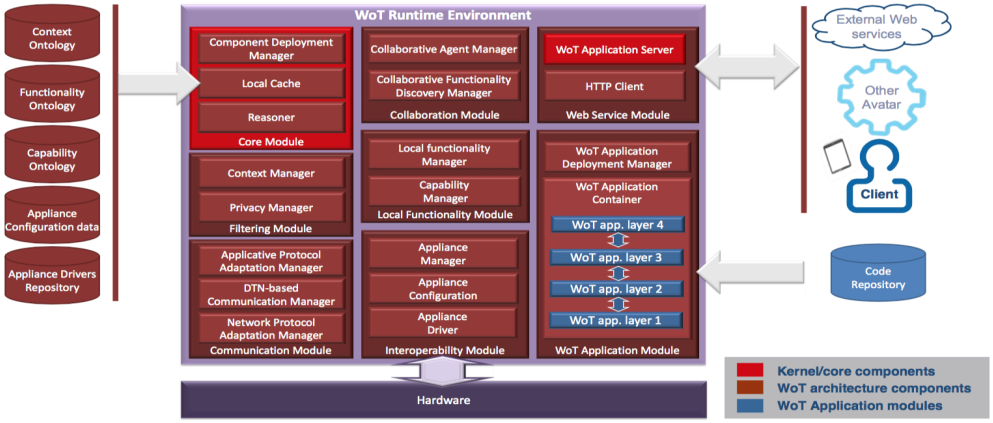

Figure 1: Avatar architecture

ule a WoT application is executed inside a WoT Application Container. The container can be physically distributed over different objects or locations (device, gateway, cloud) by the WoT Application Deployment Manager. The different parts of the application are implemented and compiled to be either executed on the object or on the gateway/cloud [2][3]. Local functionality module, include the Capabilities Manager that discover the capabilities of an object while communicating with the object and by using introspection techniques detailed previously. The Local Functionality Manager deducts functionalities while combining capabilities discovered by the Capabilities Manager. Local Functionality Manager needs the help of the reasoner to reason about capabilities [2][3]. Collaboration module formed by the Collaborative Functionality Discovery Manager and Collaborative Agent Manager.

Avatars respect the WoT requirements and specifically the collaboration between devices and components. To do so, an avatar should identify functionalities of other avatars in order to enable a collective behavior. Collaborative Functionality Discovery Manager looks for external functionalities in the avatar community. Collaborative Agent Manager observes the activity of other avatars in its immediate environment to identify if its goals are compatible with the goals of other avatars or if there is a conflict. Communication module that has the responsibility to select the right network interface, the right network (Wi- Fi, Bluetooth, Zigbee, etc.) and application protocols (CoAP, HTTP) according to availability and needs (Throughput and needs). DTN Communication Manager configures and initializes the communication protocol based on the "store, carry and forward principle". DTN Communi-

cation Manager allows the avatar to support connectivity disruptions [2][3]. Filtering module includes the Context Manager that aggregates data from different sources into contextual situations, [9] in order to identify on which avatar to expose a collaborative function; to decide which functionalities to expose; to choose where to deploy each architecture component and application code module and to determine the most appropriate protocol stack for the current communication scheme and contextual conditions. The privacy manager will reason about privacy constraints and protect data [10][3]. The Interoperability module formed by the Appliance communication manager that communicates with other components in the architecture. As well it works with the appliance configuration manager and the appliance driver to communicate with the object. The Appliance configuration manager relies on an object configuration tools database to associate communication methods to objects. The Appliance driver loads and uses the drivers to send and receive messages to and from the object [2] [3].

2.2.2 Avatar features

An avatar presents different features used to present services for users. This part presents some important features of an avatar. Semantic description of capabilities and functionalities is the first feature. The description of capabilities and functionalities in ASAWoO project is done using OWL (Web Ontology Language) semantic Web language [2]. OWL web semantic language is used when there is a need to process the content of information instead of just presenting information [5]. The Avatar discovers the capabilities of an object using introspection techniques by using SAJE (System-Aware-Java Environment). SAJE give the possibility to offer information about the capabilities of physical objects, and to control some components of these objects such as the communication interfaces. [3] Then the avatar infers its functionalities using a common ontology that keeps trace of the different combinations of capabilities required to achieve each functionality. Semantic descriptions of capabilities and functionalities also provide intelligent resource management facilities [2]. Another feature is Context-Aware Adaptation. The avatar has a multi-level, semantic context model giving the possibility to be deployed on different types of objects from different manufacturers. The model can be processed at different abstraction levels and must respond to requests regarding different adaptation goals [19]. These goals are chosen to first deploy the different application modules. Then to choose the most appropriate communication protocols and schemes for a given task. Also to decide whether an object has enough resources to perform a task or not and if the task is safe.

Finally, to take part in negotiations to achieve collaborative functionalities. The context engine created will be both compatible with the high-level features defined in the project and the contextual and QoS data provided by the objects or other resources [2]. Disruption tolerant service provisioning is the third one. Today's network can suffer from short communication range of wireless interfaces, interferences and limited power resource of devices . This can cause disruptions in the communication links. To cope with these issues, routing protocols used with avatars implement the "store, carry and forward" principle. If two nodes encounter a difficulty to communicate with each other, because they are not in the transmission range, "store, carry and forward" principle offers the possibility to exploit other mobile nodes as intermediate relays. These intermediary relays can carry a copy of a message when they move and forward it afterwards to other nodes so that it eventually reaches its destination [6]. In addition, other functionalities are supported by the avatar.

2.2.3 Avatar life-cycle

First, the avatar is instantiated, the avatar builder should be located on the local network gateway to detect the arrival of an object in the network. After creation, the avatar deploy the main components for the good functioning, connects to and exchanges messages with the object it extends to discover its actuators and sensors to build a list of capabilities. Based on this list the avatar decides with the help of a reasoning engine which capabilities to expose as functionalities. When the lifecycle comes to an end after object disconnection, the avatar notifies its community that its services are not available and terminates all the processes [3].

2.2.4 Avatar deployment on cloud infrastructures

A cloud infrastructure is an ideal location for avatars, since they can rely on unlimited resources provided by the interconnected cloud infrastructures. In order to deploy the avatar on a cloud, an instance (Virtual machine) will be created , then java will be installed to obtain a JVM (Java Virtual Machine) on which we will upload the avatar. The avatar is deployed as a java project on the JVM as it is developed as an OSGi project formed by different bundles and components.

Our contribution uses two different cloud infrastructure interconnected and available on the same network. This way, the interconnection between the

two infrastructures will be easier. We used the OpenStack environment (http://docs.openstack.org) of Université Lyon 1, formed by different compute nodes as the first infrastructure. For the other cloud infrastructure, DevStack (http://docs.openstack.org/developer/), the testing environment of OpenStack, was installed on a local machine.

2.3 ASAWoO project

The ASAWoO project (https://liris.cnrs.fr/asawoo/) is a 4 year research project funded by the French National Agency for Research (ANR), and started on January 2014, in the context of the INFRA ANR program. IMAGINOVE competitiveness cluster (http://www.imaginove.fr), a witness for its interesting impact on the society and industrial domains had certified the project. The ASAWoO project has for objective to enhance appliance integration into the Web. Finally, ASAWoO project is designed on an OSGi (Open Service Gateway initiative) platform, a module system and service platform for the Java programming language and has become a standard component for Java platforms. ASAWoO project includes different avatars instantiated for different physical objects and each one is formed by different OSGi components or bundles. An ideal location permitting to the avatar to rely on unlimited resources is a cloud infrastructure. That's why we use two cloud infrastructures for the migration of the avatar. In what follows, the four essential bundles of the ASAWoO project will be presented.

The Core that represents the main part of a java program. It is the core of the ASAWoO middleware project. Different sub-bundles are defined: annotation, capability, config, context, deployment, functionality, net, proxy, util and webapp. In addition it consists of three global classes Activator, ASAWoOServiceTracker and runtime. By using this bundle the system will define variables as functionalities, capabilities, methods and parameters, etc. Services that represents the internal components of the avatar. One of the components is Capability that activates a bundle through the activator class. In addition Capability can add or remove listeners through the LocalCapabilityManager class. Bundles that includes the different implementations for objects and applications. Through this bundle all functions and services needed by applications and objects are instantiated. It should include all functions that may be used in different scenarios. And finally, Domains that represents the functions available on interfaces like motion detector and data collector. This bundle represents the interface used by the user to control the avatar and to specify its needs.

An avatar is a software artifact corresponding to a physical object, it possesses different functionalities available on the WoT. It is an important element to study and explore. The research problem and the motivation scenarios are developed in the next chapter.

3 Motivation scenarios and related work

Different scenarios motivate us to study the migration life-cycle of avatars. In fact avatars migrate from one infrastructure to another when it is crucial for the good functioning of the services and it is more beneficial than keeping it on the source infrastructure.

3.1 Motivation scenarios

In what follows, we discuss different examples that explain the need of migration of avatars in order to clarify the idea and the objectives of the project. An avatar as mentioned before is dedicated to a specific physical object, the avatar and its physical object can be on different infrastructures or on the same one. When the physical object moves from an infrastructure to another, the fact of migrating the avatar with its physical object is important to consider. As shown in figure 2, an avatar dedicated to the mobile device of a researcher (Alex) is instantiated on its home cloud infrastructure (Cloud 1), Alex leaves home for work, then logically his mobile device will be connected to a new infrastructure, the infrastructure of the laboratory (Cloud 2). Two choices are available; leave the avatar on Cloud 1 -if it is more beneficial, it has a good internet connexion and has reasonable capacities for calculations -while the mobile device is on Cloud 2 . The other choice is to migrate the avatar to Cloud 2, then Alex's device and its avatar are on the same infrastructure. The migration can be done periodically in this case, a trigger will be created for this process. For example the trigger migrates the avatar from the Cloud 1 to Cloud 2 at 8 am when Alex heads to work and from Cloud 2 to cloud 1 at 6 pm when he returns home from Monday to Friday. Another example for a migration scenario is when the cloud infrastructure on which the avatar is instantiated/running had become overloaded with calculations. In this case the services presented by the avatar can be interrupted and the time of calculations will increase, the higher is this time the lower is the performance, so migration will probably resolve the problem. In addition, in some cases, the infrastructure hosting the avatar should

be shut off for maintenance or for technical problem, etc. Then the avatar should be migrated to another infrastructure to ensure a continuity of service.

An additional scenario is when different avatars need to be migrated at the same time as the different physical object migrate together. For example a family goes out, all the mobile devices will be disconnected together from the gateway. Then the avatars instantiated for the physical devices need to migrate together. To ensure that the model of criteria is respected, and there is no ambiguity, the migration should be done sequentially. For this purpose, a single token will be in used. Then we ensure that migrations are done one at a time.

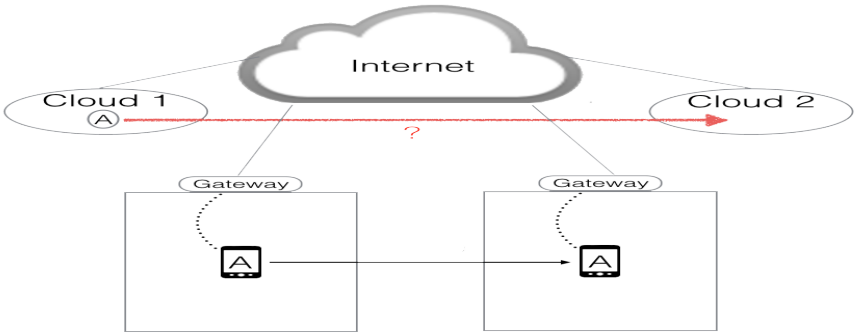

Figure 2: Scenario of migration

When the migration is one of the possible choices, mandatory questions should be answered : Should we migrate or not to the new infrastructure? What is the best solution for migration? Do we use Live migration, Freeze and Resume migration or a compromise between the two? How to migrate with a minimal cost? etc. Those decisions should be based on a model of criteria that includes several metrics on which our study will be based and it will be developed later. The cost of the migration will be calculated through the measurements of the different metrics of the model by using a point system and thresholds. Additionally the migration in our case is accomplished in a WoT environment. Consequently it should respect the requirements for this environment [3] [4]: Live reactivity, Resource management, interoperability, disconnection tolerance, safety, delegation and collaboration.

Then the problems to study for the migration scenario are listed below.

- Necessity: To study the need for migration and the advantages offered by this process. How the migration will be more beneficial for the user

and for a better service.

- Cost: To understand the effective cost of the migration. The price can be paid for the migration or for the communication between infrastructures by using the public network; Internet.
- Type: To learn about the available types of migration, their feasibility and advantages. The choie of the migration type is mandatory to benefit from the migration.
- Frequency: To study how to minimize the cost needed in case of multiple migrations per day. In addition, to indicate if there is a need to create a trigger for the migration.
- Duration: To understand the effect of integration time and how it will affect the global cost of migration. In fact, during the integration time, services on the source and on the destination are used simultaneously.
- Downtime: To deduce how the service continuity will be affected. During this period service will be not available neither on the source nor on the destination and the user will not have access to the service.

3.2 Migration

Migration is the process of moving an application, virtual machine, project, etc. from one environment to another while considering at the destination the last state of the migrated object on the source. It brings the benefit of being able to move workload from an infrastructure to another. In general, migration facilitates fault management, load balancing, and low-level system maintenance. In consequence it improves the capacities and the features of Cloud environments and implies more flexible resource management. It presents different advantages for the continuity of services, services' cost, systems' performance, etc. Although this is a complex and a difficult technique to use over MAN/WAN. IP addressing is one of the biggest issue as the system changes the address of the migrated virtual machine which may not remain in the same network domain. Moreover, it impacts the performance of VMs by adding a non negligible overhead [9]. Additionally, migration can be complicated because differences between the original and destination environments exist always.

Live migration and "Freeze and Resume" migration are two types of migration that should be finely defined. Live migration refers to the process of moving a running object between different physical infrastructures or machines

without disconnecting the client or application; with a minimal downtime. Practically, downtime exist in all live migration techniques and results from the fact of copying memory content. Different studies are trying to reduce downtime by using different approaches such as pre-paging and the probability density function of memory. Live migration ensure that all resource state consisting of CPU, memory and storage are created on the destination infrastructure. Then the subject of migration is suspended on the source infrastructure, copied and initiated on the destination infrastructure. At the end memory, storage, and network connectivity of the object are transferred from the source to the destination [10]. Non-live migration or Freeze and resume migration, as we decided to call it, is the slowest type of migration and performs a network copy of the virtual machine or of the application after freezing it on the source. It is the process of copying the object on the destination without any consideration for the downtime and the continuity of service during the migration. After migration, the application or virtual machine should be resumed on destination. The downtime is proportional to the size of the data transfer [11].

3.3 Related work

Different examples are done for the migration of OSGi components and OSGi projects. The existing works present different solutions for component migration as OSGi-PC that presents migration through internal Java classes and Kalimucho that offers the possibility to apply dynamic reconfigurations. These two works ensure a component migration; or the avatar is formed by multiple components. Another work present a solution for service migration by using mobile agents and Agent Transfer Protocol. Other works present a list of requirements that should be respected during the migration in OSGi platforms. In what follows, a brief description of different works on the migration in OSGi platforms.

3.3.1 Component migration - OSGi-PC

OSGi-PC, OSGi-based pervasive cloud infrastructure is developed in paper [12]. It uses the cloud computing capabilities and the component flexibilities from OSGi. The migration through OSGi-PC need to determine the components subject of migration and to provide these components to specific remote framework. A remote deployer service at the destination needs to be registered to allow the migration and to manage the migrated component after migration. OSGi-PC is implemented on Apache Felix OSGi framework as it

follows the OSGi standards. In addition this implementation reuse code from DOSGi (Distributed OSGi) [13] which specify that standard meta-data information of components and services is used to disclose service information. The component migration in OSGi-PC uses different classes to determine the component that should be migrated, to shut down those components and then to deploy them remotely. The authors defined three different types of component migration: component migration between frameworks on remote power nodes (where a new framework joining the OSGi-PC should register itself with necessary information), between frameworks on small devices (a simple process as the components are codes that are transferred, it is a process of receiving the components, installing and starting them) and between frameworks on remote power nodes and frameworks on small devices (small devices can not read Jar files, the RemoteDeployerImpl service convert components from Java code to the code used by small devices and vice versa).

3.3.2 Component migration - Kalimucho

Another work on migration in OSGi platforms, Kalimucho platform is described in paper [14]. Kalimucho is a supervision platform that can perform dynamic reconfiguration decisions to ensure service continuity and is distributed on all peripherals of the application. The work in [14] present a possible way for service reconfiguration during runtime and different mechanisms to ensure structural reconfigurations including functionalities as migration, replacement, etc. This platform gather all the information (status of the application, components, peripherals, connectors and the environment) in order to process it and perform reconfiguration actions. Authors in [14] identify three actions used to modify the structure of an application: service migration (Modifying the distribution of a service without modifying its components. Components are moved, with their statuses, from one peripheral to another without changes.), deployment or redeployment of a service (Offering a new assembly of components to perform a service or an equivalent service.) and modification of a service (Replacing one or more components with others.). To perform the actions listed before, the platform should monitor operation of components and data flows circulation by using containers. The architecture of Kalimucho is based essentially on a middleware and a core. The middleware handles the communications between components, it consists of connectors. It present the possibility to create, delete, connect and disconnect components while the application is running.

In addition, it involves different operations as creation of connectors and

migration of components. The core comprises services that can be installed, if needed, depending on the configurations. The services that can be installed are Services Registry, Application supervision services, Code loading services and Network communication services. Service Registry allows the platform and applications to access services offered by Kalimucho. Application supervision services execute creation/deletion/connection/disconnection commands coming from the platforms. Code loading service used to load the code for classes. Network communication services enable platforms to communicate with each other and are used as middleware medium. Migrating a component running on a host A to a host B is performed with Kalimucho as followed, the platform on A stops the component (like for a deletion), then sends it to the platform on B by serializing its properties. Finally its input and output connectors are redirected so that they now end up on B and not on A. In fact the communication between components is provided by connectors using a client/server model.

Based on the idea of these two works we had created a solution for migrating the platform when needed and migrating component if the platform exists on the destination host. The solution will be described later.

3.3.3 Service migration - ATP

Paper [20], discusses the migration in OSGi platforms while using mobile agents. A mobile agent is a composition of computer software and data which is able to migrate autonomously. The migration can reduce the network load, provide dynamic adaptation and a fault tolerant flexible network. The migration is realized through Agent Transfer Protocol (ATP). Every mobile agent framework is an ATP server. Dispatch and retract are two activities used for the agent's migration. Dispatch demands a destination agent framework to reconstruct an agent from the content of a request. Then the destination framework should start to execute the agent. In case of a successful request, the sender must terminate the agent and release any resources consumed by it on the source. Retract used when the destination agent framework should send the specified agent back to the source.

3.3.4 Migration requirements

Author in [21] indicates that some requirements and non-functional concerns need to be fulfilled to enable service migration in OSGi platforms. In what

follows, the requirements are listed. Knowledge about the presence, type and context of the devices are prerequisite to guarantee a minimum usability and QoS. Before relocating a service to another host, a service discovery protocol is used to verify if the service is not already available. Stateful services require a state transfer after redeployment in order to continue their operations on the destination host. Furthermore, the service must be able to resume from its last state. The service platform must provide functions to manage the life cycle of services. If a service cannot be run perfectly due to a lack of resources on the platform, the service needs to be migrated to another powerful device or infrastructure to ensure a better service.

Author in [22] described four requirements four framework migration: Transparency, portability, minimum overhead during normal execution and strong migrations properties. He indicates that the transparency requirement is needed because the OSGi framework should appear as a local framework to the outside world. For portability, he declares that all the platforms participating in the virtual framework should be portable. In addition, the techniques adopted to capture the last state should not cause a worse service performance while remaining at the same location; that's why minimum overhead is required. Finally, strong migrations properties are a necessity to automate all the tasks to perform a migration with no interference of programmers or users.

3.4 Summary of the research problem

All the examples listed before present the need for migration; but migration should be done if it is more beneficial, while minimizing the cost and while conserving the last state of the avatar on the source. Our study offers a model of criteria that can decide which case (migration or non migration) is more beneficial based on several criteria implemented in a model through a shell code. Additionally this study presents a way to migrate the avatar from an infrastructure to another while considering the last state of the avatar on the source. This section present the migration that will be discussed meticulously afterwards.

4 Contributions

After defining the problem and the motivation scenarios for a migration it is time to describe our migration method. In addition, the model of criteria on

which the decision of migration is taken will be explained.

4.1 Migration Methods

As mentioned before, our migration technique is based on the several works presented in a previous section. Our contribution is presented as a shell code executed on the source infrastructure. The migration of avatars between cloud infrastructures can be classified into three different scenarios or cases. In what follows, each scenario is explained meticulously.

The first one is to migrate the avatar from an instance on a source cloud to a different plain instance on the destination infrastructure with no installations of prerequisites or environments. For this purpose, the destination environment should be prepared, by installing Java JDK8 to obtain a JVM (Java virtual Machine). In addition all prerequisite softwares and tools as curl, expect, etc. should be installed. After preparing the environment, migration can take place. To do so all the project, as the avatar is an OSGi project, should be migrated from the source to the destination including the felix runtime environment.

The second scenario is to migrate the avatar from an instance on a source cloud infrastructure to a JVM instance on a different infrastructure. In this case, only prerequisite softwares and tools should be installed if needed. Then the avatar with felix runtime environment will be migrated from the source to the destination.

The last scenario is to migrate the avatar from an instance on a source cloud infrastructure to a prepared instance including the felix runtime environment on the destination infrastructure. In this scenario, all prerequisite and the runtime environment are available on the destination. For this, only the bundles used by the avatar subject of migration will be migrated to the destination.

Our contribution will migrate the avatar to the instance on the destination cloud infrastructure while respecting the corresponding scenario of migration. To do so, the destination instance on the receiving infrastructure will be tested to check first if the JVM is installed, then if the prerequisites are available. Finally, the presence of felix runtime environment will be tested.

The migration of the avatar will be ensured through different phases. If different avatars need to be migrated at the same time, sequential migration

will be used. In fact, avatar's migration should be sequentialized to respect the model of criteria. This idea is developed and explained in the next subsection. First, when the avatar possesses the right for migration, the JDK on the destination will be tested by checking its version. If there is a need, the JDK will be updated or installed. Second, expect and curl, two tools on linux used by the avatar, are checked on the destination to ensure their presence. To install or update tools and environments on the destination the installation command will be sent using a ssh tunnel from the source to the destination. Third, the felix runtime environment will be checked on the destination. In case felix is available, a synchronisation between the environment on the source and on the destination is executed through the remote synchronisation command (rsync). In fact, this phase will ensure that all the components needed for the good functioning of the felix runtime environment are available and with the correct version. If this runtime environment is not available, it will be copied on the destination using the Secure Copy Protocol command (scp) that will add security to the copying process by using also a ssh tunnel. Finally, the bundles used by the avatar need to be available on the destination. For this purpose, a remote synchronisation for the bundles between the source and the destination is executed. The correct version of the bundles and the needed bundles will be available on the destination through this final phase.The scp and rsync commands used in the migration code use different arguments; the source file, destination file and the security key. In what follow we can visualize two examples of the usage of these command.

```
rsync -e "ssh -i /home/ubuntu/key4.pem" -r /home/ubuntu/middleware/
ubuntu@1 : /home/ubuntu/middleware/
```

```
scp -i /home/ubuntu/key4.pem -r /home/ubuntu/middleware ubuntu@1 :
/home/ubuntu
```

After preparing the destination instance on the destination infrastructure for hosting the avatar, the services should be stopped on the source. To do so, each bundle used should be stopped correctly to conserve its last state. Using the remote shell bundle, we will access the gogo shell, that monitors the different bundles running on the source felix server, using a telnet connection. After accessing the gogo shell, stop commands will be executed for each bundle. The state of the bundles will be saved in the configuration file automatically after the stop commands. Telnet connection is created on the localhost, using the 6666 port, so no security flaw.

To migrate the avatar with its last changes and states, a configuration file storing these values will be copied to the destination using also the "scp" command. This configuration file will be sent before starting the services on the destination. When the configuration file is received, the avatar can be started on the destination by sending the corresponding commands using a ssh tunnel for security reasons.

During the migration, the different metrics used by the model of criteria are measured using specific tools installed on the source. The model of criteria and the tools needed are developed later in this section. Some tools need also to be present on the destination; the installation commands will be executed also through a ssh tunnel to remove security breaches. The values of measurements will be returned after the migration and will be saved in a specific file for later use.

Our contribution will allow a fast and easy migration of avatars and OSGi projects between cloud infrastructures. In case, the migration will be periodic between two infrastructures as for Alex's example described in previous section; the migration will be optimized by minimizing its cost and offering a better service. This migration technique ensure a convenient migration also in other scenarios like for infrastructure maintenance, etc. Figure 3 explain our algorithm of migration through a model.

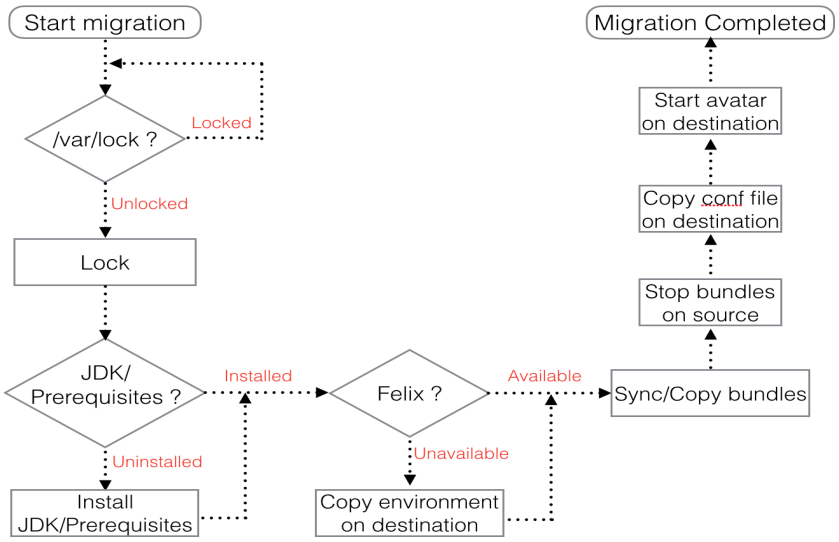

Figure 3: Schema of the migration

4.2 Mutex for chained migration

A mutex is a programming concept that is frequently used to solve multi-threading problems. It is a mutually exclusive flag used to synchronize multiple threads. In some cases, when different attempts to access a single resource is present, only the first attempt should gain access and the others should wait until the resource is free again. In fact, if different avatars should be migrated and asked for migration at the same time; the migration processes should be sequentialized. This way, persistent values for the different metrics of the model of criteria are obtained. For example, if the destination infrastructure has no running avatar yet and the source infrastructure deploy them all. The migration will be privileged as the source is overloaded and the destination is free. Without a mutex, all the avatars will be migrated to the destination, in case they ask for it at the same time. The mutex will ensure the reliability of the criteria's model by sequentializing the migrations, and repeating the calculations of the different metric in the model of criteria after each migration.

To implement the mutex a file (/var/lock/mylock) is created to ensure that there is a lock while one avatar is migrating. The file should be accessed only by one avatar, thanks to the function flock() available on Linux that locks the file until the migration of the avatar is over. The code allowing to implement the mutex for our migration method is represented below.

```
for line in (cat/var/lock/mylock)
do
while[″line" = "locked" ];
do
echo waiting for token
sleep 15;
done
if [ "line″ =″ nolock″];
then
echolocked > /var/lock/mylock
flock − x.007/var/lock/mylock
.....
```

4.3 Model of criteria

The migration of the avatar is not always more beneficial than keeping it on the source. To find which case is more beneficial , the cost of the mi-

gration should be estimated. Based on this cost all the questions related to the execution of the migration will be answered. In order to accomplish the cost calculation, a model of criteria should be proposed and should use existing techniques and technologies and adapt them to the different scenarios of usage. For each criteria of our model, measurements in different scenarios are considered and results will be compared to take decisions. The implementation of our model is based on a weighted average, for each criteria a weight is affected. After comparing results, the weight will be added to the migration or non migration variable depending on the results. Finally, these new values will be compared in order to take a decision based on the higher value. In what follows, for each criteria, the techniques of measurement and theirs adaptation to our scenario will be considered.

4.3.1 Capacity of the source infrastructure

The capacity of the source infrastructure, in term of cpu and memory is a mandatory criteria. This criteria indicates if the source infrastructure can reply to the avatar's needs. Before passing to other criteria, the capacity of the source infrastructure will be studied. If there is not enough capacity, migration will be executed and other criteria will not be taken into consideration. To measure this capacity, the nova client on the source infrastructure will be requested to return the number of available CPUs and available memory space. Based on returned values, the decision will be taken to continue to measure other criteria or to migrate immediately.

This first criteria in our model of evaluation, ensure that the avatar has its needs to offer the best services for the user. In case there is a difficulty to obtain its needs, the avatar should be migrated to offer the best service from the most suitable infrastructure.

4.3.2 Power consumption

Power consumption while migration is a non negligible criteria, as it affects the real cost of the migration and indicates if the migration is eco-friendly. The avatar is implemented on a cloud infrastructure, so power management techniques and models for grids and data centers should be used to estimate and minimize this consumption.

The power consumption of any system consists of a fixed part and a variable part. The fixed part depends on the system size and components type. The

variable part results from computing, network and storage ressources usage [15]. To save energy at data centers, studies in [16] indicates to minimize the transmission losses by locating the center close to where the electricity is generated. In addition, free cooling techniques as outside air or sea water should be used. Another way to decrease energy consumption is to decrease heat production by creating algorithms that deal with energy and thermal problems.The best way to minimize energy is to place the workload on an infrastructure where the outside temperature is low and so the system will use less energy for cooling the infrastructure [17]. The fact of minimizing the energy consumed on cooling systems, power consumption of other components can be studied precisely. For example, if the infrastructure is used only for avatar instantiation, the migration will be prioritized during time where the cooling energy is minimal to minimize energy consumption. Our infrastructures are not designed only for avatars instantiation so we can not use this technique in our model.

To monitor power consumption, sensors as power meters should be implemented on the infrastructure to return live values during migration. In our contribution, a wattmeter available in LIP Lab had been used to return power consumption values. The wattmeter used is LMG 450 model of ZES ZIMMER, a Multi-channel precision Power Meter. It is a high precision wattmeter that returns several measurement in a second. This frequency of measurements is more than enough to monitor the power consumption. The values returned by the power meter will be stored in a file for future use. Additionally, a mean value will be calculated and compared to a threshold defined by the user. The comparison will help for taking a decision of migration. If the mean value is higher than the threshold, the weight of this criteria will be added to the non migration variable. If not, the weight will be added to the migration variable. Idle consumption is subtracted from the values returned by the power meter to obtain the effective power consumption of the migration. The power consumption of the infrastructure will be measured to calculate the idle consumption. Last, the mean value of the power consumption of the migration will be multiplied by the time of migration (time of integration of the avatar on the destination) to obtain the energy consumed in Joules. For this purpose, the formula E (J) = P (W) * T (s) is used.

This criteria, ensure that the migration of the avatar will take place if it is more beneficial from energy consumption point of view. This way, the user can monitor the power consumption while ensuring a migration when needed. Figure 4 and 5 below show the idle power consumption with no

migration activity and the power consumption during the migration.

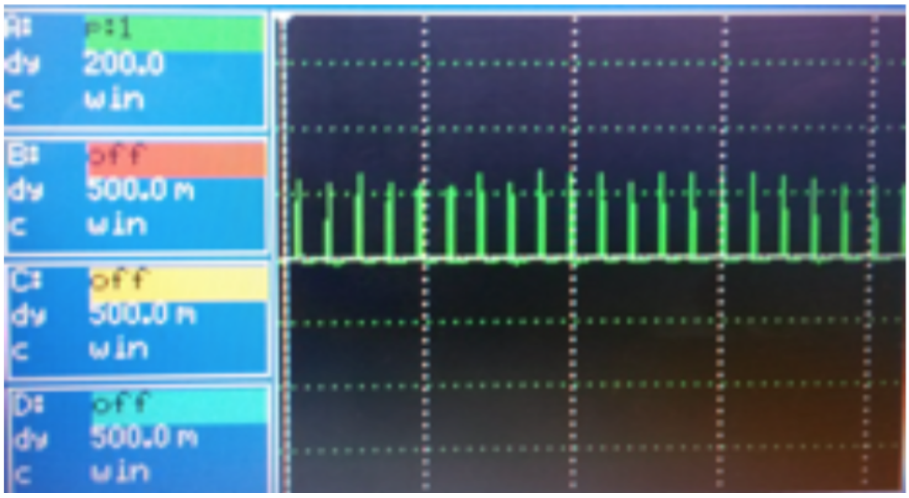

Figure 4: Idle power Consumption of the infrastructure

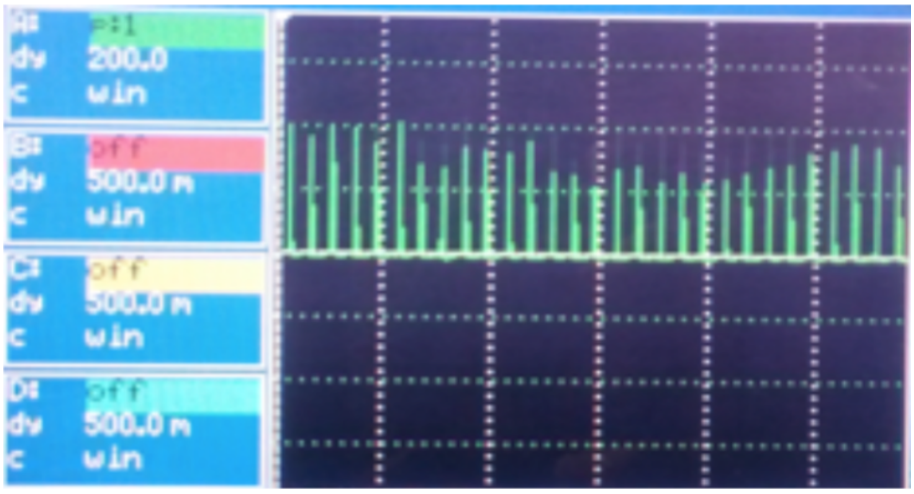

Figure 5: Power consumption on the infrastructure during the migration

4.3.3 Used Data

The bandwidth usage while migration is an important criteria to study, as it influences the bandwidth available on the network between the two infrastructure. On the other side the bandwidth used to communicate with the

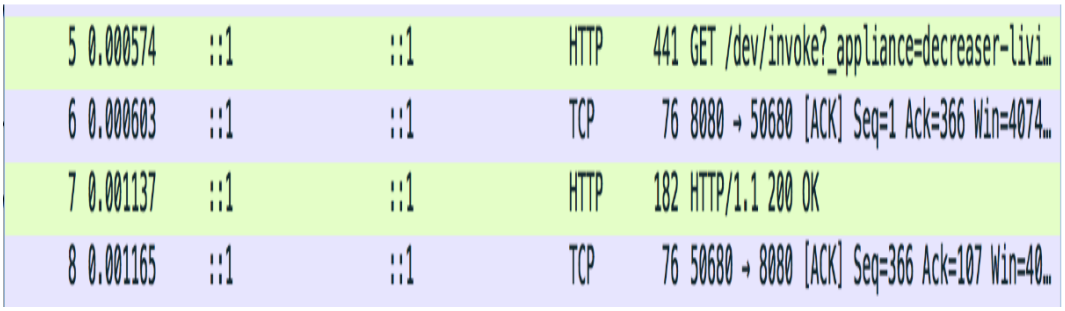

Figure 6: Inter-avatar communication packets

avatar residing on the source cloud infrastructure should be measured too. Those two values should be compared to decide which case is more beneficial.

To measure used bandwidth, Spruce and Pathload represents the two techniques on which all the measurement tools are based. Spruce sends packet pairs spaced back to back according to the capacity of the tight link, and the amount of packet dispersion at receiver determines the Bandwidth. Pathload creates short-lived congestion in the tight link then find trends in one way probe packet delays [18]. Studies in [18] showed that Pathload is the most accurate tools for big infrastructures with high speed. iPerf, a network testing tool that measures the network throughput between end hosts, uses the same technique as Pathload and gives accurate results. Although this tool eats a lot of bandwidth for measurement. iPerf used in study [18] can return values in a manageable time, about 6 seconds but it returns the bandwidth available on the link between the two infrastructures.In our study we will use the ifstat tool available on linux that return used data amount. It returns the used data amount during every second. These values will be summed to obtain the total usage for data during migration (Total-Data).

Figure 6 and 7 show respectively the packets of a single inter avatar communication and the packets of TCP connection establishment.
If the avatar is not migrated, a communication between the physical object on the first infrastructure and the avatar on the second will be needed. These communications will also consume bandwidth. A single communication for the avatar is formed by four packets; two HTTP packets (GET, HTTP/OK) and two acknowledgment packets (TCP ack) after building a TCP connection. These packets were sniffed using Wireshark. The mean value for the packets size is calculated (Mean-Value-Packets). In addition the data used for TCP connection establishment is calculated (TCP- Con-Data).

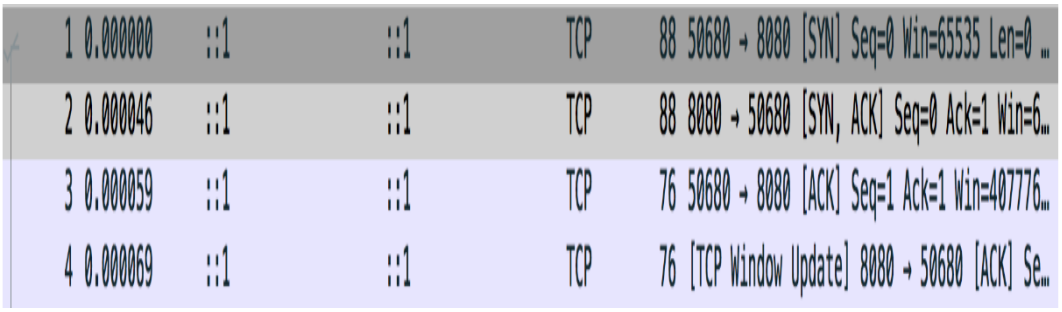

Figure 7: TCP connection establishment packets

The communication between avatar and physical object can be based on different basic models : a frequent communication model, three communications every minute (90 communications per hour) or a non frequent communication, one communication every five minutes (30 communications per hour).The communication model is chosen based on the type of the avatar. In our implementation the user will choose between these two models. We should compare the used data for migration and the used data for communication. To do so, we will calculate the duration of communication, between avatar and physical object on different infrastructures, in which used data will be the same as for migration. If this time is less than a threshold defined by the user, migration variable will be increased, if not the non migration variable will be increased by the weight of this criteria. First the TCP-Con-Data is subtracted from Total-Data then the result will be divided on Mean-Value-Packets. The number of communication needed for having the same data usage as for migration is obtained. This number will be divided then by the number of requests per hour based on the model of communication (30 communications / hour or 90 communications / hour) to obtain the final result and to compare it to the threshold value.

Data usage during migration can influence the performance of the network interconnecting the two infrastructures or the network of the same infrastructure. For this reason, this criteria was essential to be used and added to our model of criteria.

4.3.4 Time of residence

Mobile devices can be moved a lot during a short time period. If the device will be on the new infrastructure for a short period, it will be more beneficial

to leave the avatar on the source infrastructure. To take decisions about this criteria, a database created for each device should be created if this criteria will be implemented in the model. The decision will be taken based on the date (weekdays or weekends), time (Morning, noon, afternoon), place (a lot visited place, new place), etc.

Using this criteria, non necessary migration will not be executed. Then the non necessary cost of those migrations will be avoided. This way, our model presents an optimized evaluation for migration decisions.

4.3.5 Delay

Duration needed for an information to travel across the network from one endpoint to another is an important criteria where the application needs live reactivity. To calculate the delay, RTT (Round Trip Time) will be measured and divided by two to obtain an approximation of delay. In fact an approximate value is needed; a simple ping can return the mean value for delay. We will use 10 ping requests to obtain the result. Two cases are used for the delay comparison. Delay used for the communication between two infrastructures and the delay needed to communicate in the same infrastructure, these two values are compared to favorite migration or non migration.

In fact, some avatars needs live reactivity with a minimal delay to ensure a fast reactivity and avoid problems. This way the best delay will be selected to present the best and adequate service.

4.3.6 Time of integration and downtime

While migrating project and applications between infrastructures, some requirements should be installed. In our case ASAWoO project should be deployed and there is a possibility that the java environment should be updated or installed. All those requirements demand some time. To calculate this time, a simple subtraction between the start time of migration and the time the project is running perfectly will be executed. To do so, the date command will be used. In addition, the downtime is a crucial criteria as it represents the time of unavailability of the service neither on the source nor on the destination. This time is calculated by subtracting the time where the service is stopped on the source and the service is running perfectly on the destination using the same date command on Linux.

This criteria, is the most important criteria of our model. As it affects

directly the continuity of service. Some avatars needs a minimal downtime, in this case downtime should be finely monitored and chosen.

4.3.7 Capacity of destination infrastructure

The capacity of the destination infrastructure is another criteria to be considered. If one of the infrastructures can't take more calculations, it will be better to place the avatar on the non overloaded infrastructure. This criteria will be judged by consulting the number of available VCPU, free memory, RAM and storage on the destination cloud infrastructure. On OpenStack those values are represented on horizon (graphical interface of OpenStack) or can be accessed through requests on nova client.

This last criteria, ensure that the avatar has its needs to offer the best services. In case there is a difficulty to obtain the needs on the destination, the avatar should not be migrated to offer the best service from the most suitable infrastructure.

4.3.8 Available functionalities on the destination

The functionalities offered by other avatar can induce in a better service of the whole environment. Some avatars can rely on the functionalities offered by other avatars to ensure a better service and so on a better client satisfaction.

Other avatars capacities can be an important criteria to study. The functionalities available on the destination by other avatars are important to consider in our study for migration decisions. If the destination can present important functionalities to the avatar, it will be more beneficial to execute the migration. Semantic description of capabilities and functionalities is done using OWL (Web Ontology Language) semantic Web language used when there is a need to process the content of information. The Avatar discovers the capabilities of an object using introspection techniques by using SAJE that gives the possibility to offer information about the capabilities of physical objects [8] [9]. Then the avatar infers its functionalities using a common ontology that keeps trace of the different combinations of capabilities required to achieve each functionality. Semantic descriptions of capabilities and functionalities also provide intelligent resource management facilities. This way the available functionalities presented by different avatars on the infrastructure can be processed and used to take decision of migration[8] [9].

To implement those criteria in our model of criteria, we should try to find a way to check the existence of some algorithms and some tasks on the destination infrastructure.

4.4 Additional Criteria

Through this section we will focus firstly on the availability of QoS mechanisms in the destination cloud computing environments. Those mechanisms can ensure a prioritized processing for priority communication packets. In addition, the security and the privacy of the data on the cloud infrastructure are essential to study. In fact, some data used by the avatars needs to be secured as it consists in personal data. Additionally, the available functionalities on the destination can induce in a better offer of the services, so these functionalities should be represented correctly.

4.4.1 Quality of Service (QoS)

QoS is a new trend in the numeric world. Using the QoS mechanisms can ensure a better client satisfaction and a better service. In order to respect the client requirements some priority tasks should be executed before others. The QoS mechanisms in cloud infrastructures will be presented in what follows.

QoS mechanism in destination infrastructures is important to consider. To apply QoS mechanisms in any environment, two functionalities are necessary. The first one is setting tasks (or packets) into one or different queues. The other is scheduling between the queues. In fact, some communication packets between avatars should be prioritized from other packets. In addition, the infrastructure should offer the required capabilities for proper service functioning. It is more beneficial to apply QoS mechanisms in the cloud infrastructure that hosts the avatar. This way, the infrastructure will ensure a better service to the user by prioritizing some packets that need to be delivered faster than others and will offer the required needs.

When we are migrating to a destination cloud infrastructure, and after building the needed environment, the cloud will deliver cloud computing services as a SaaS (Software as Service). Consequently, to offer QoS mechanisms, specific SLAs (Service Level Agreements) that respect the requirements of SaaS should be established.

Data on clouds are hosted on servers. To satisfy the user on the performance

and the availability level, the service provider must have enough ressources in order to be in line with the avatar needs. Considering the wide variety of services in SaaS and specifically avatar's services, it is difficult to provide a comprehensive and representative list of SLO (Service Level Objective) for our case. So users should expect the general objectives as the monthly downtime, response time, the persistence of consumer information, and automatic scalability. In addition, data held on the cloud should be stored using standard formats to ensure data portability. For this, SaaS services must be self-elastic to assign the number of servers, databases and computing capacities required to respect the SLA [23][24]. To ensure that client specifications are respected, an algorithm, taking the objectives as entry variables, calculates and applies the configurations on the cloud.

Several works like in [25] are done till now to improve the utilization of servers allocated to the jobs and the resource utilization, to process the job having higher priority and finally to minimize the waiting time and the switching time. To do so, a scheduling technique is used in cloud computing systems. Other works like in [26], use a proposed Workload consolidation method supported by virtualization technologies for improving utilization of resources in data centers and clouds.

Scheduling mechanisms in cloud computing environments induce in managing the processes and in increasing the performance of the servers and the ressources. Different scheduling algorithms and techniques are available for the cloud. The goal of cloud task scheduling is to achieve high system throughput, to allocate various computing resources to applications and to prioritize some tasks on others. Different examples of scheduling algorithms can be discussed: FCFS (First come First serve), Round-Robin, Min-Min algorithm, Max-Min algorithm, etc. Other works had created there own protocols and techniques for scheduling in cloud computing environments to improve the performance on several levels. The fact of having a scheduling algorithms implemented on the infrastructure can ensure a certain level of QoS [25] [26]. We are not studying which algorithm is the most optimal one, we are testing if the infrastructure can ensure QoS mechanisms by supporting any scheduling algorithm.

Queuing is another important QoS mechanism, response time is defined as the time for a request to be serviced which corresponds to the waiting time in the queue and the service time. Consequently, queuing technique induce response time. The fact of choosing the best queuing technique is important. Different queuing techniques and queuing model theories are available [27].

Again, we are not studying which technique is the most optimal, the simple fact of using one of them will add QoS mechanisms on the cloud.

So as a conclusion to ensure that QoS mechanisms are available on the infrastructure; an algorithm checking if the service's needs, a scheduling algorithm and a queuing technique should be used on the infrastructure. To check, if service's needs are available on the infrastructure, the capacity of infrastructure criteria will be used. To check the availability of queuing and scheduling, techniques cloud providers or administrators should present informations about those two criteria.

4.4.2 Security

Security had become an essential criteria in any environment, data should be protected , communications should be encrypted to respect the privacy of the communications, etc. In a cloud infrastructure where the data is available on remote servers, the privacy of the users should be respected. That's why security mechanisms and functions should be implemented. The degree of security risk associated with cloud computing is directly related to the sensitivity of the data that will be stored, processed, or transmitted [28]. The cloud will deliver avatar's services as a SaaS. Security challenges in SaaS applications are the same for any web application technology. Traditional security solutions are not sufficient, so new approaches are necessary as the Open Web Application Security Project (OWASP) that identifies ten critical web applications security threats [29]. In what follows the essential security mechanisms needed are discussed and defined.

Security mechanisms on cloud infrastructures is another criteria to take into consideration in order to ensure the data privacy. Security of data and of communications is a main concern for the cloud environment.While migrating the avatar from an infrastructure to another, all the mechanisms are secured via ssh tunnels using public key systems. The security concerns consist on how to secure the communications in the same infrastructure and how to secure the data on the infrastructure for respecting the privacy of the users.

To ensure the security in an infrastructure, three essential security services should be available: Confidentiality, Authentication, Integrity. The confidentiality is the measures undertaken to prevent sensitive information from

reaching the wrong people, while making sure that the right people can in fact get it. Data encryption is a common method of ensuring confidentiality, public key encryption is the best way till now to ensure this service. Authentication is the process used by a system in order to verify the identity of a user trying to access it. Different techniques of authentication are available: public key authentication, login/password authentication, etc. Integrity ensure the consistency, accuracy, and trustworthiness of data [30]. It ensures also that data did not changed in transit. These measures include file permissions, user access controls, some means to detect any changes in data caused by non-human events. These means include checksums, even cryptographic checksums, for verification of integrity. Secure communication between the different part of the avatar residing on different places on an infrastructure is needed also to ensure the security. Secure communication is when two entities can communicate in a way not susceptible to interception. To ensure this, an encryption is needed, public key encryption will ensure that the communication is just readable by the the two communication entities [30] [31].

Consequently, data encryption, authentication methods and integrity check are essential for securing the infrastructure. The presence of such techniques indicates that security is taken into consideration on the infrastructure and for the communication between different part on the destination. We need to check the availability of security mechanisms on the destination to consider the target infrastructure as secured.

4.5 Deployment of the model of criteria

Our model based on several criteria is implemented through a shell code. Two variables are used (migration and non migration) to make at the end a decision to perform. After evaluating each component and based on the results obtained during the evaluation, the weight of the criteria will be added to the corresponding variable. At the end, these two values will be compared and the decision will be taken.

First, the capacity of the source infrastructure is tested: if it is not suitable for the avatar, the avatar will be migrated to the destination and there is no more measurements for other criteria. If not, the evaluation based on the other criteria will take place. Next, the bandwidth usage will be tested to find if the migration is more beneficial. Then, integration time and

Downtime are evaluated. Finally, energy consumption, and the capacity of destination infrastructure are evaluated. As mentioned before, the two variables, migration and non migration, will be compared and the best decision will be made.Figure 8 shows a model explaining the functioning of the model of criteria .

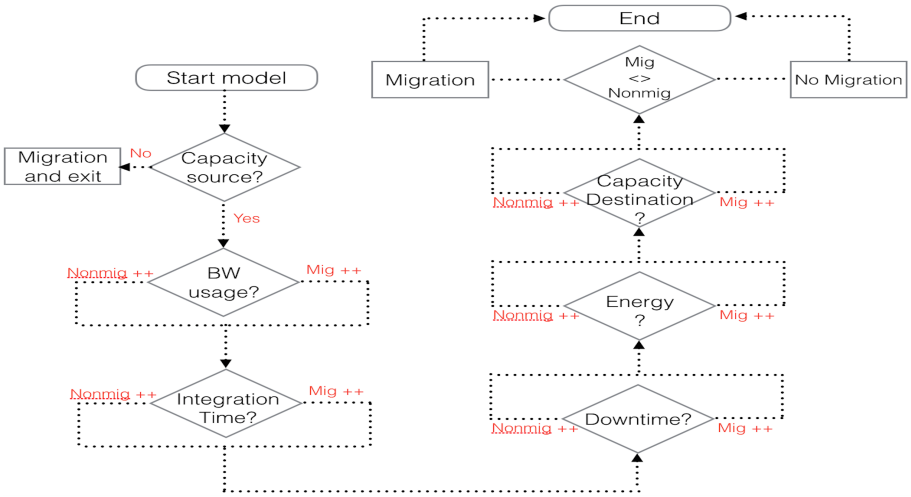

Figure 8: Schema model of criteria

This section presented the migration method used in our approach. In addition, the model of criteria that had been discussed is the base of our study. Using this model the results that will be explained next are measured and calculated.

5 Experiments and evaluation

This section presents the different measurements and values for each criteria in the model described in the previous section. In addition a mean value and a standard deviation for each criteria are presented also. Finally, the results are analyzed and explained.

5.1 Measurements

The values and measurements are represented using tables to simplify the representation. Table 1 shows different measurements for different metrics of the criteria's model for each scenario case. In addition, Table 2 shows the mean value and the standard deviation for each case. Case 1 corresponds to

the scenario where all dependencies are available on the destination without the felix runtime environment. The second case corresponds to the scenario where all dependencies are available on the destination with the felix runtime environment. Finally, case 3 corresponds for a migration to a plain destination infrastructure.

5.2 Interpretation

As we can see in the result before the time of integration is considerable. In fact the DevStack environment we are using as a second infrastructure is on a local machine. This local machine has limited capacities in term of power calculations (CPU) and in term of memory resources which create this considerable time. To be sure we had monitored the CPU usage and the memory usage during migration in our source infrastructure (DevStack). The CPU usage before migration is about 25,9 %, during the migration the usage of the CPU passes to 50 % , 98 % and then 100 %. In addition the memory access before migration was 38/116 MB and during migration, it passes to 92/116 MB. In addition, as we can see the percentage of downtime over the integration time is around 1,14 % for the first case, 2,67 % for the second case and 0,89 % for the third case. The continuity of service will be ensured and the service will be unavailable for around 10 seconds for the three cases. For the energy consumption, we obtained reasonable values. For the first case the energy consumption is approximately about 1,72 Wh (Watt hour), for the second it is about 0,69 Wh and for the third it is about 1,86 Wh. These values are acceptable as the time of integration is less than half an hour so for each migration, the energy consumption is minimal.

	Case 1			Case 2			Case 3		
	Experiment 1	Experiment 2	Experiment 3	Experiment 1	Experiment 2	Experiment 3	Experiment 1	Experiment 2	Experiment 3
Integration Time (s)	1015	996	1137	341	325	343	1194	1252	1209
Downtime (s)	11	11	14	5	10	11	11	10	14
Used Data (MB)	214,24	276,14	212,26	62,17	67,22	80,65	300,07	301,12	301,02
Delay (ms)	3,83	6,93	4,54	3,30	3,52	2,35	4,24	8,83	3,52
Energie Consumption (J)	6101,16	6215,04	6311,25	2315,39	2312,05	2802,31	6961,03	6873,4	6274,71

Table 1: Values of different metrics

The interpretation shows that our approach can be used on a real application and on powerful infrastructures. In fact our results can be improved by using more powerful infrastructures as the time of calculations will be reduced and the CPU percentage also.

	Case 1		Case 2		Case 3	
	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Integration Time (s)	1049,33	76,51	336,33	9,86	1218,33	30,11
Downtime (s)	11,85	1,73	8,66	3,21	11,66	2,08
Used Data (MB)	234,21	36,23	70,01	9,55	300,73	0,57
Delay (ms)	5,10	1,62	3,05	0,62	5,53	2,87
Energie Consumption (J)	6209,15	105,17	2476,58	282,09	6617,87	383,62

Table 2: Mean and standard deviation for different metrics

6 Conclusion

The existence of avatars as an extension to physical objects is currently an ongoing work in the ASAWoO project. From here it is important to study the different life-cycles of the avatar and specifically the migration. In this report, we had presented a simple method to migrate the avatar between cloud infrastructures. Our method is based on basic commands on Unix. In addition, this report presents a concrete model for migration evaluation. It is a model of criteria based on a multiple criteria to evaluate the relevance of migration and on a set of shell scripts to drive the migration process from cloud to cloud. The migration, one of the most important life cycle of avatars is studied through this internship. Specifically, the study focuses on how and when to ensure the migration. In addition, it focuses on the cost of the migration.

Through the model of criteria, different physical and software tools had been used to measure the different metrics to evaluate the project migration. Our study has shown that service continuity can be ensured through our approach with a minimal downtime. The energy consumption of our approach can be explained as we are migrating a big project, so we are using a lot of CPU and memory.

Future works consists in studying different phases of the life-cycle of the avatar. Replication is one of these life-cycle. An example, is the avatar's replication when there is a need. In fact, the replication of the avatar is considered when different physical object have the same capabilities. In this case, these objects need the same avatar, and replication can be useful. Additionally, the study can be based on how to create the replication on an environment that can support only one avatar. Furthermore, the model of criteria proposed depend on the type of the avatar. For this, a study can be

made to automatically adapt the importance of each criteria and give them the corresponding weight for weighted average calculations. To do so, metadata informations should be used. The studies on this project are important as a lot of work should be done to have a complete solution that adapt itself for the scenario.

References

[1] Deze Zeng, Song Guo, and Zixue Cheng. *The Web of Things: A Survey*, School of Computer Science and Engineering, The University of Aizu, Japan. 15 June 2011.

[2] Michael Mrissa, Lionel Mni, Jean-Paul Jamont, Nicolas Le Sommer and Jme Laplace. *An Avatar Architecture for the Web of Things*, LIRIS Laboratory, Universit Lyon, France. July 2014

[3] Michael Mrissa, Lionel Mni, Jean-Paul Jamont, Nicolas Le Sommer and Jme Laplace. *Towards An Avatar Architecture for the Web of Things*, LIRIS Laboratory, Universit Lyon, France. http://www.itu.int/osg/spu/publications/internetofthings/

[4] Cristian Borcea, Xiaoning Ding, Narain Gehani, Reza Curtmola, Mohammad A Khan, Hillol Debnath. *Avatar: Mobile Distributed Computing in the Cloud*, Department of Computer Science, New Jersey Institute of Technology University Heights, Newark, New Jersey, 07102, USA. 2015

[5] W3C Recommendation. "OWL Web Ontology Language Overview". 10 February 2004 https://www.w3.org/TR/owl-features/

[6] Vinicius F. S. Mota, Felipe D. Cunha, Daniel F. Macedo, Jos S. Nogueira, and Antonio A. F. Loureiro. *Protocols, Mobility Models and Tools in Opportunistic Networks: A Survey*, Computer Communi- cations, vol. 48, pp. 5-19, March 2014

[7] Gita Sukthankar, Robert P. Goldman, Christopher Geib, David V. Pynadath, and Hung Bui. *Plan, Activity, and Intent Recognition: Theory and Practice*, Morgan Kaufmann 2014

[8] Michael Mrissa, Mohamed Sellami, Pierre De Vettor, Djamal Benslimane, and Bruno Defude, *A decentralized mediation-as-a-service architecture for service composition* in WETICE, Sumitra Reddy and Mohamed Jmaiel, Eds. 2013, pp. 80-85, IEEE.

[9] ANNE-CECILE ORGERIE, MARCOS DIAS DE ASSUNCAO, LAURENT LEFEVRE *A Sur- vey on Techniques for Improving the Energy Eciency of Large Scale Distributed Systems* INRIA, LIP Laboratory, University of Lyon, France, CNRS, IRISA Laboratory, France, IBM Research, Brazil

[10] *Live Migration*, https://www.techopedia.com/definition/16813/live-migration

[11] *Migrating Virtual Machines and Storage Overview*, https://technet.microsoft.com/en-us/library/jj628158.aspx

[12] ZHANG WeiShan, CHEN LiCheng, LIU Xin, LU QingHua, ZHANG PeiYing, YANG Su. *An OSGi-based flexible and adaptive pervasive cloud infrastructure*, Department of Software Engineering, China University of Petroleum - College of Computer, Fudan University. 2013

[13] Zhang Y, Huang G, Liu X Z, et al. *Refactoring android java code for on-demand computation offloading. In: Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, New York: ACM, 2012. 233-248

[14] Keling Da, Marc Dalmau, Philippe Roose *Kalimucho: Middleware for mobile applications*, Universit Pau et des Pays de l'Adour - France. March 2014

[15] ANNE-CECILE ORGERIE, MARCOS DIAS DE ASSUNCAO, LAURENT LEFEVRE *A Sur- vey on Techniques for Improving the Energy Eciency of Large Scale Distributed Systems* INRIA, LIP Laboratory, University of Lyon, France, CNRS, IRISA Laboratory, France, IBM Research, Brazil

[16] Greenpeace 2011. *How dirty is your data?*, Greenpeace report.

[17] Sharma, R., Bash, C., Patel, C., Friedrich, R., and Chase, J. *Balance of Power:Dynamic Thermal Management for Internet Data Centers* IEEE Internet Computing 9, 1, 42 - 49. 2005

[18] Hitesh Khandelwal, Ramana Rao Kompella, Rama Ramasubramanian *Cloud Monitoring Framework* December 8, 2010

[19] Mehdi Terdjimia, Lionel Médinia, Michael Mrissaa *Towards a Metamodel for Context in the Web of Things* KSS Research Workshop February 25 -26, 2016

[20] Mikael Desertot, Si-Ho Do, Didier Donsez, Marc Bui *Mobile Agents Platforms over OSGi* 4th International Conference on Computer Sciences, Research Innovation and Vision for the Futur February , 12-16, 2006,

[21] Davy Preuveneers and Yolande Berbers *Context-driven migration and diffusion of pervasive services on the OSGi framework* Autonomous and Adaptive Communications Systems 2010

[22] Damianos Maragkos *Replication and Migration of OSGi Bundles in the Virtual OSGi framework* Swiss Federal Institute of Technology Zurich 2008

[23] Cloud Standards Customer Council *Practical Guide to Cloud Service Level Agreements Version 1.0* 2012

[24] Yousri Kouki, Damierrano, Thomas Ledoux, Pierre Sens, Sara Bouchenak *SLA et qualit service pour le Cloud Computing* 2013

[25] Lipsa Tripathy and Rasmi Ranjan Patra *SCHEDULING IN CLOUD COMPUTING* International Journal on Cloud Computing: Services and Architecture (IJCCSA) , October 2014

[26] Xiaocheng Liu, Albert Y. Zomaya, Fellow IEEE, Chen Wang, Bing Bing Zhou, Junliang Chen, Ting Yang, *Priority-Based Consolidation of Parallel Workloads in the Cloud.* IEEE Transactions On Parallel And Distributed Systems, Vol. 24, No. 9, September 2013

[27] Er. Shimmy and Mr. Jagandeep Sidhu *DIFFERENT SCHEDULING ALGORITHMS IN DIFFERENT CLOUD ENVIRONMENT* Chitkara University, India, September 2014

[28] Jordi Vilaplana, Francesc Solsona, Ivan Teixidrdi Mateo, Francesc Abella, Josep Rius *A queuing theory model for cloud computing* Universitat de Lleida, 09 April 2014

[29] GEMINI Security Solutions *A Brief Security Overview of Cloud Computing* http://geminisecurity.com/wp-content/uploads/tools/a-brief-security-overview-of-cloud-computing.pdf

[30] KeikoHashizume, DavidGRosado, EduardoFernez-Medina and EduardoBFernandez *An analysis of security issues for cloud computing* Journal of Internet Services and Applications, 2013, https://jisajournal.springeropen.com/articles/10.1186/1869-0238-4-5

[31] Matthew Haughn, Stan Gibilisco, *http://whatis.techtarget.com/definition/Confidentiality-integrity-and-availability-CIA*

[32] Cloud Standard Customer Council *Security for Cloud Computing Ten Steps to Ensure Success Version 2.0, March, 2015*