Avoiding Free Riders in the Cloud Federation Highways

Marcio Roberto Miranda Assis and Luiz Fernando Bittencourt
Institute of Computing, University of Campinas, Av. Albert Einstein 1251, Campinas, São Paulo, Brazil

Keywords: Cloud Computing, Inter-clouds, Cloud Federations, Free Riders, Multi-clouds Tournament.

Abstract: The maturity of the Cloud Computing paradigm has highlighted a set of obstacles which isolated cloud providers are not being able to handle. To overcome these obstacles, isolated providers can organize themselves in entities called Inter-Clouds, mainly to share resources. However, this kind of resource-sharing environment may face the emergence of free riders, which only consume resources without caring for the whole, in a modern version of the Tragedy of the Commons. This work characterizes the free riders and proposes an Inter-Cloud architecture to avoid them based on the main features of Cloud Federations. This Inter-Cloud architecture, called Multi-Clouds Tournament, organizes multiple cloud providers in a tournament-based fashion, allowing those with better *scores*, determined by a function of *offer* and *consumption*, to take advantage of the system. On the other hand, those with low expected returns to the system, free riders for example, face disadvantages or even are eliminated from the tournament. We show preliminary tests of a score function within the tournament, illustrating how the system promotes or eliminates participants according to their behavior.

1 INTRODUCTION

Cloud computing (Zhang et al., 2010; Mell and Grance, 2009) is a paradigm that has emerged as the answer to the search for solutions for delivering computational assets as utilities (Assis et al., 2016). A precursor of this paradigm is the Grid Computing (Berman et al., 2003; Foster et al., 2008), which has been inspired by the electric power grids delivery model to provide computing power to its stakeholders. As grid computing was focused on non-utility resource sharing, it has been applied for more successfully in academic collaborations. On the other hand, Cloud Computing has been focusing on more diverse types of customers, making it suitable to a wider range of areas and applications, including both enterprises and academia.

In Cloud Computing, the computational assets are made available to clients in the form of *services* - infrastructure (IaaS), platform (PaaS), and Software (SaaS). Customers in general acquire these services in a *self-service* way, most often via portals. Such services can be accessed *ubiquitously*, as long as there is a network communication between the client and the provider, and the quality of service requirements are defined in a service level agreement (SLA). Another feature of the paradigm is the *elasticity*, which allows customers to increase or decrease the amount of resources acquired according to their current needs in a *pay-as-you-go* basis.

As the Cloud Computing paradigm matured, a number of limitations began to emerge. Because the services offered are directly tied to the physical resources, isolated Cloud Service Providers (CSPs) are led to deny customer requests when their resources become exhausted, thereby limiting the elasticity. QoS is also compromised due to the impact of the possibility of providers having their physical facilities in regions temporarily subject to high data traffic for example. Moreover, certain customers, such as governments, may have restrictions that prevent them from storing data on providers physically located in other countries. In addition, the lack of standards and market competition lead providers to implement their own data schemas, libraries, and functionalities. Named *Lock-in* (Petri et al., 2014), this behavior imprisons the clients to their respective providers, making changes too costly, which can generate disinterest in this type of technology. To overcome these and other limitations (Toosi et al., 2014) cloud providers have begun to interconnect, forming associations called Inter-Clouds (Grozev and Buyya, 2012).

Inter-Clouds are pools of resources shared among providers. These pools can stimulate the emergence of free riders, which have a selfish profile aimed only at maximizing its own needs to the detriment of the

Assis, M. and Bittencourt, L.
Avoiding Free Riders in the Cloud Federation Highways.
DOI: 10.5220/0006304901970207
In *Proceedings of the 7th International Conference on Cloud Computing and Services Science (CLOSER 2017)*, pages 169-179
ISBN: 978-989-758-243-1

needs of other providers. Such behavior may lead to degradation of the association, either through resource exhaustion or disinterest in use. Some solutions to avoid free riders in resource sharing environments have been proposed in the literature recently, such as approaches involving game theory (Mashayekhy et al., 2015; Khethavath et al., 2013), definition of supply and demand policies (Toosi et al., 2011), and networks of favors (Falcão et al., 2015). However, they are focused on specific types of Inter-Clouds (centralized federations, decentralized federations, etc.), and do not focus at the same time on promoting the offer/consumption of resources and the elimination of selfish providers in the environment. In addition, they are implemented as a service over the Inter-Clouds, and not built on the architecture itself. This may lead to increased complexity and labouring maintenance (Joe-Wong et al., 2016).

Considering the context described above, this work formalizes one of the main economic problems related to environments where there is resource sharing: the presence of free riders. To deal with this problem, we describe an architecture called Multi-Clouds Tournament (MCT). Inspired by a soccer tournament, the MCT implements, at the architectural level, a mechanism to prevent free rider providers.

Section 2 shows the definition of free riders and the historical/economic context from which the term appeared. The set of Inter-Clouds composed of several solutions of multiple cloud organizations proposed to solve the limitations of isolated clouds providers is described in Section 3. The main solutions that multi-cloud organizations implement to prevent free riders are covered in Section 4. Section 5 describes the MCT, the solution used by the proposed Inter-Cloud architecture to avoid free riders. Section 6 discusses the simulation and preliminary results, and section 7 concludes the paper.

2 THE TRAGEDY OF THE COMMONS

In 1968 Garrett Hardin published *The Tragedy of the Commons* (Hardin, 1968). This article was inspired by a pamphlet written by William Forster Lloyd describing the overuse of a common area, called Commons[1] by several users. In the article published by Hardin, the consequences of the disordered consumption of a set of resources were described when they are made available to various stakeholders.

According to Hardin, there are several factors that influence the dynamics of using a set of shared resources where users are not owners but can consume the resources present there. This behavior was described in an example where a pasture area was offered, without any control mechanism, to the various stakeholders to raise cows. Since no regulations have been enforced, **each pastor can put the amount of animals that he/she finds convenient in the area**. Initially, this behavior generated benefits (products, revenues, etc.). However, with the multiplication of this behavior, the pasture consumption increased proportionally to the number of animals (Figure 1), which led to the exhaustion of the food and the decrease in the number of cows, causing injury to all herders.

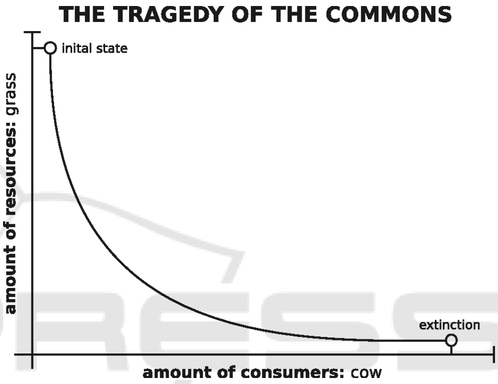

Figure 1: Without control mechanisms as the number of consumers increases the resources are finishing.

In resource-sharing environments, consumers can adopt different behaviors in relation to the consideration of the stakeholders and the consumption of existing resources, which in one case can degrade the entire environment (as in The Commons example). Thus, we consider three roles of consumers in those resource sharing environment: *altruistic, conscious*, and *selfish*.

2.1 Altruist

The altruist prioritizes the neighbor, not expecting any kind of reward for his actions. The altruist offers the maximum resources to the other stakeholders in the shared environment, while maintaining the consumption of resources to meet his/her needs at the minimum level.

Considering a behavior function $f : B \rightarrow B$ ($B = \mathit{Behavior}$), for the altruistic elements of an environment:

$$f(\mathit{altruist}) = (\text{offer } \uparrow, \text{consume } \downarrow)$$

It is possible to conclude that, because of disinterested and selfless concern for the well-being of oth-

---

[1] Private area offered to a group of people for exploration: grazing, gathering of firewood, etc.

ers, the altruists consume as little as possible and offer the most they can. Consequently, in case of resource shortage, altruists may face lack of resources due to no reserve. With this, it has a fragile life-cycle, being the first type of stakeholder to be harmed when the amount of resources available starts to decrease.

2.2 Conscious

The conscious are aware of resources scarcity and tries to keep the balance of the environment, thus consuming and providing resources at a proportional rate.

$$f(conscious) = (\text{offer } \simeq \text{consume })$$

The conscious **elements are more resistant to enviroment's resources shortage**. This resistance allows them to make decisions regarding maintenance of their own environment (periods of greater and lesser consumption) as well as abandonment of the environment when the resources are being extinguished.

2.3 Selfish

Also referred as free rider elements in the context of multiple-cloud associations, this is the opposite of the altruist. His/her behavior is oriented to maximize attention to particular needs, consuming disorderly the resources present in the environment and offering as little as possbile (resources) back. Selfish do not care about maintaining the environment and focus on maximum benefit in the shortest possible time.

$$f(selfish) = (\text{offer } \downarrow, \text{consume } \uparrow)$$

In resource shortage scenarios, this consumer is one of the last elements to disappear from the environment because they have a greater reserve of resources than altruists and conscious. They can also migrate to another resource pool and maintain the same behavior in order to keep resources avaliable.

2.4 Commons Maintenance

Elinor Ostrom et al. (Ostrom et al., 1999) complement Hardin's work by stating that to prevent the degradation of environments that have shared resources, there must be mechanisms of control of the environment. Among these mechanisms, two of them stand out: i) the election of a leader to regulate access to the pool, or ii) incentives and punishment toward the preservation of shared resources. In (i) an element (individual, groups of individuals, or an organization) is responsible for maintaining order of consumption within the environment. In this approach, this element can be established by consensus or by imposition, and both cases involve problems regarding the leading acceptance factors that can be difficult to solve in environments where there are elements with heterogeneous behavior (altruist, conscious, and selfish). The point (ii) involves the offer of incentives for good behavior and punishment if the behavior is not appropriate. The challenge in this approach is to characterize "good behavior" and find ways to incentivize it, as well as establish the set of punishments to be applied to offenders without disencouraging participation.

3 INTER-CLOUDS

With the maturation of the Cloud Computing paradigm, some providers have begun to associate themselves. The purpose of this association was increasing their respective revenues with the commercialization of idle resources and overcoming limitations of service provision. Additionally, expect to overcoming challenges (Buyya et al., 2010) they face when acting alone (limited elasticity, legal restriction, and affected QoS).

- *Limited elasticity*: inability that some CSPs have in providing scalability of resources to their customers. This is because scalability is related to the limited amount of physical assets present in CSPs, which mainly affects small- and medium-sized providers that have few resources and have difficulty acquiring more computational assets. The main consequences of this property is the containment of resources, and possible loss of QoS.
- *Legal restrictions*: providers have the ability to be physically located in a particular region and offer their services to their target audience, regardless of the geographical region they are in. This property affects those customers who by rule or force of laws (e.g. government) have restrictions on the location of their data. This can limit their interest in using Cloud Computing, harming CSPs.
- *Affected QoS*: the main communication network used by CSPs to provide services is the Internet. The Internet is composed of a set of networks with distinct characteristics that are exposed to the respective local traffic. For example, during the sale off day called *Black Friday* there is a significant increase in the access of **virtual stores in the USA**. This behavior increases the network traffic in that region and can affect the providers present there compromising the QoS of services provided.

Interconnected-Clouds (Grozev and Buyya, 2012; GICT, 2010), or simply Inter-Clouds, can be defined as a set of all associations of interconnected cloud providers. The subsets of Inter-Clouds most cited in the current literature are: *Multi-Clouds*, *Hybrid Clouds*, *Sky Computing*, and *Cloud Federations*. The Multi-Clouds (Kurze et al., 2011; Grozev and Buyya, 2012; Toosi et al., 2014) are associations of CSPs made from applications with the support of libraries – e.g. Libcloud (Foundation, 2015b) – or frameworks – Jcloud (Foundation, 2015a). In the Hybrid Clouds (Mell and Grance, 2009; Bittencourt and Madeira, 2011), CSPs intend to a specific group of customers to use resources from provider open to the general public to meet their needs when it is convenient (price, availability, resource expertise, etc.). Sky-Computing (Keahey et al., 2009) is a facilitator service that, through the execution requirements of an application, creates an association of several independent CSPs with sufficient resources that meet the demand of the application. Another significant subset is Cloud Federations which will be described below.

3.1 Cloud Federations

The cloud federations (Grozev and Buyya, 2012; Buyya et al., 2010; Celesti et al., 2010; Manno et al., 2012; Chaurasiya et al., 2012; Panarello et al., 2014), are Inter-Clouds organizations with a set of particular properties (Assis et al., 2016) that make them attractive to customers and CSPs. The main properties are:

- *Autonomous and voluntary providers*: CSPs can leave the federation at any time, when they find it convenient.
- *Geographic dispersion*: the diversity of physical location of the CSPs allows services to be migrated to another region in a timely manner.
- *Defined by contract*: all technical aspects of the federation and the behavior of the providers are described in a contract signed between the federation and the providers – the Federated Level Agreement (FLA) (Assis et al., 2016; Toosi et al., 2011).
- *Real elasticity*: providers can use federation to scale horizontally (amount of resources for a service) and/or vertically (classes of services).
- *SLA end-to-end*: SLAs between clients and providers are honored when services migrate to other providers within the federation.

From a customer point of view, federations are attractive because they enable clients to acquire services with resilience, guaranteed QoS, reliability that their data will be in the region of interest, and the freedom to migrate to another provider whenever convenient. Providers can maximize their revenue by selling or consuming resources and services to/from the federation, ensuring that the agreement among them and their customers are being respected if there is a need to use resources from other providers, and leave federation when convenient. In addition, the FLA guarantees each provider the knowledge of the possible actions of the other elements present in the environment, resulting in more predictability and stability to the Cloud Federation. However, they are still resource-sharing environments that are prone to the presence of free riders.

4 RELATED WORK

Some solutions have been proposed to address the presence of free riders in Cloud Federations. These proposals can be grouped into three classes that describe the approach used to solve the problem: *Policies*, *Game Theory*, and *Network of Favors*.

4.1 Policies

Toosi et. al (Toosi et al., 2011) propose a set of resource provisioning policies by providers to increase the profitability of the federation and thereby encourage sharing. However, this approach only covers cloud federations at the IaaS level. Another limitation is that defining a set of policies that meet the wishes of all providers can be an arduous task, considering the conflicting local policies that may exist among providers. In addition, this approach does not directly address free riders, but rather offers incentives to “good providers” (i.e., autruist and conscious).

4.2 Game Theory

In (Mashayekhy et al., 2015) game theory is used to create a model of uniform and proportional distribution of profitability obtained in the federation to providers. Distribution works as an incentive mechanism, while allowing the rational definition of the amount of resources offered to the organization by each service provider. This is another approach that deals with free riders, but in an indirect manner. In addition, it is focused only on centralized federations, which have a controlling element.

4.3 Network of Favors

Falcão et. al (Falcão et al., 2015), to avoid the presence of free riders, propose the incentive to share resources in decentralized federations (Peer-to-Peer) through the use of Network of Favors (Andrade et al., 2004) focused on justice. Cloud providers when offering resources to the federation receive credits that can be used later in the acquisition of resources. Consequently, the approach hopes to avoid the presence of free riders within the federation while maintaining the satisfaction of the participating providers. However, the solution only includes decentralized federations focused on IaaS.

5 MULTI-CLOUDS TOURNAMENT

MultiClouds Tournament (MCT) (Assis and Bittencourt, 2015) is an Inter-Clouds environment based on cloud federations and focused on the resources of the CSPs. Its main objective is to keep the resource-sharing environment free of free riders. To do this, MCT establishes (at an architectural level) the incentive to the provision of resources through the constitution of a tournament inspired by a divisions scheme and a ruleset of a soccer tournament. The MCT is defined as:

> *An Inter-Clouds organized in a tournament format, where volunteer CSPs, named players, are grouped into divisions ruled by a ruleset that defines the players' behavior within the environment. Depending on the progress (monitoring in regular periods of time) a player can be raised to higher divisions obtaining advantages (e.g. access to specialized resources, more resources) or demoted to a lower division, decreasing its access privileges.*

5.1 Architecture

Architecturally (Figure 2), MCT is composed of n divisions, the players (cloud service providers); the shared resources available in the environment; the statute with the set of tournament rules, and the referee that mediates the environment.

Formally, the proposal is described as a tuple $\mathcal{T} = (D, J, \Psi)$ consisting of three elements: a finite set of divisions D ($|D| \geq 2$), a finite set players J, and a bijective function $\Psi : J \rightarrow D$ responsible for mapping each player $j \in J$ to a division $d \in D$.

5.1.1 Divisions

The *divisions* D is a logical partition of the set of players. Each division has a specific set of rules described in the statute (Section 5.1.3). These rules define the behavior of the players within the environment. In MCT there may be n divisions, among which two are special and should be present in any implementation: the *access* and the *premier division*. The access division is where new players stay when they become part of the tournament. In this division are found many resources but there are no guarantees of specialization or QoS. In this division is expected to contain those players with multiple clients, lots of features and/or specialized resources because this profile contributes to increase the provider's score.

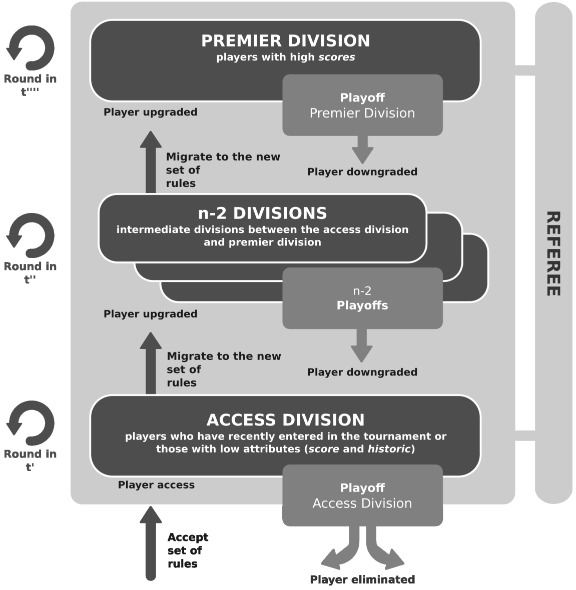

Figure 2: Elements whitin MCT architecture.

The divisions are hierarchical in the rules as well as in the access to resources. Regarding the rules, in higher divisions, more rules are inserted in the corresponding subset. With this procedure, it is expected that the excellence level from the players present in those divisions increases. Regarding to the resources, players at higher divisions will have more resources available: in MCT, players have access to the resources of their own division as well as the resources belong to all other lower divisions.

The players' life-cycle in the divisions is defined by a set of individual attributes. In MCT, two attributes are provided: *scores* and *history*. The first has an immediate character, and describes the state of the relation *offer* × *consumption* of a player at a certain moment. The score is calculated at the end of each time period called *round*. Also at the end of this time period it is verified if the player's score is within the criteria to remain in that division, enabling the referee to rise or to demote a player to other divisions. As the history reflects the behavior over time of each player, it can influence for example the amount of rounds of tolerance that a given player stays in the division if it does not have a high enough score at a certain point in time. The implementations of the score and history functions are defined in the implementation of the MCT, making it flexible to different niches and situations possible.

Within each division, there is an area called *play-off*, where players who failed to keep the established criteria can be sent if they have a history that subsidizes this decision. Once in that area, a player can be reinstated or recessed from the corresponding division. After a predetermined period of time, if the player can improve its own score it is reinstated to the division. Otherwise, it is demoted. This procedure aims to give a chance to players who perform well over time, but for exceptional reasons have failed to maintain the division permanence criteria.

Another peculiarity of the divisions is the presence of *dimensions*, which are structures that allow a given player to be classified in more than one division at the same time. This depends on the dimensions present in the tournament. For example, in a tournament where dimensions are service levels (IaaS, PaaS, and SaaS) a player can be in several divisions according to each level of service, or even give up exposing one of these three.

5.1.2 Players

A *player* is a CSP where there are resources that will be offered. These resources can be seen as the prizes of the tournament. Players may be heterogeneous in what regards to the type and amount of resources available in their own domains. Another feature inherent to the players is that they are volunteers to join the organization as well as free to leave the MCT at any time. Each player or team $j \in J$ is defined as a tuple $j = (A_j, \overrightarrow{L_j}, \overrightarrow{S_j}, \overrightarrow{H_j})$, where A_j stands for a set of technical properties that the player j has, $\overrightarrow{L_j}$ is a dimensional vector exposed by the player to allow the classification by divisions, and the vectors $\overrightarrow{S_j}$ and $\overrightarrow{H_j}$ respectively contain the calculated scores and the history of scores for each dimension in $\overrightarrow{L_j}$.

5.1.3 Statute

The *statute*, named Tournament Level Agreement (TLA), is a set E consisting of a subset R representing the FLA in the MCT context. Unlike the FLA at a global scope, each subset R the from TLA is applied to a specific division of MCT. Another feature is that a player may refuse a subruleset, so the player can remain in a division even having a score high enough to be promoted to a superior division .

Subruleset R should not be empty ($R \neq \emptyset$). Each subset must have at least four rules called Fundamental Rules (FRs): FR_1 – score calculation criteria for each player belonging to division; FR_2 – how the history of each player is defined in the respective division; FR_3 – upper and lower score limits to stay in the division; and FR_4 – round time.

5.1.4 Referee

The referee is the mediator of the architecture and acts as a Broker (Grozev and Buyya, 2012) centralizing all activities within the MCT. It makes the process of managing the players and the environment softer. Among the activities of the referee there are: *access management and identity*, *monitoring*, *calculation of player's attributes*, *mapping player*, *scheduler*, *catalog*, and *data repository*.

5.2 MCT Avoiding Free Riders

The MCT avoids the proliferation of free riders through the divisions, accessible by a score that each player has and that is determined by its behavior during their life cycle within the tournament. Two situations may occur in relation to a free rider within the MCT: i) the player is already a free rider when entering the tournament or ii) the player modifies its behavior during the evolution of the divisions. In the first case, as the player does not offer resources to the other players of the respective division its score is not increased. Thus, free rider does not reach the lower boundary of the access division at the same time that it does not present sufficient history to stay in the play-off area, being eliminated from the tournament after some rounds. In the second situation, a similar behavior occurs, however, depending on the division the player is, while it is conducted to the lower divisions, it is still able to consume resources until it becomes eliminated from the tournament.

5.3 Prototype

For the initial proposal validation, a prototype of the MCT was implemented in Python and submitted to 8 test cases. The prototype code and data resulting from the tests below are stored in a public repository[2].

Three divisions $\{d_1, d_2, d_3\}$ named *premier*, *second* and *access* have been defined. All divisions share the same value of round interval (t), but they aren't synchronous. The resources considered were: vCPU, memory and storage. They were grouped into three types of virtual machine flavors describes in Table

---

[2] https://github.com/mrmassis/mct.git

1. The dimension addressed was the Infrastructure as Service ($\forall d \in D, \overrightarrow{v} = [IaaS]$). Finally, the statute comprises three sets of rules R, $E = \{R_{div} | div = d_1, d_2, d_3\}$. One peculiarity is that the VMs used in the tests are simulated and do not perform actual workload, so they have a fixed duration time.

Table 1: Description of the types of VMs in the test cases.

VIRTUAL MACHINES				
VM's FLAVORS	vCPUS	MEMORY (MBytes)	DISC (GBytes)	RUNNING TIME (minutes)
Big	8	8192	80	10
Small	4	4096	40	20
Tiny	2	1024	20	30

The Figure 3 illustrates the prototyped MCT.

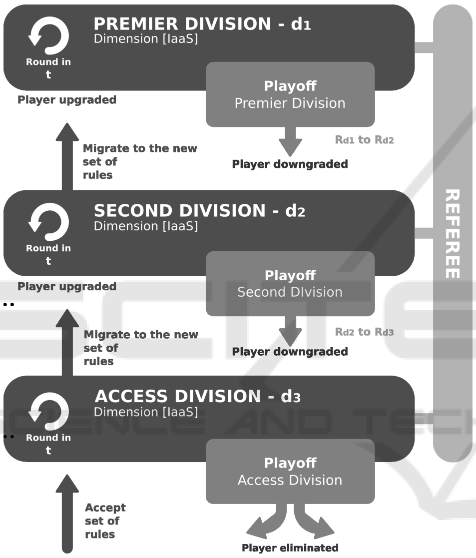

Figure 3: Layered diagram representing the implementation of the MCT prototype.

Equation 1 describes the function used to calculate the players' score (s_j). In it, the score is determined by two terms: the first one considers the sum of the execution times of the VM instances ($k = t, s, b$ – tiny, small, and big) from the requests the provider has accepted ($I_{VM_k}^{execution}$). The execution time of each type of virtual machine is multiplied by a weight (p_k) that is proportional to the size (in capacity - Table 1) of the VM type considered: the larger the executed virtual machine, the greater the weight applied ($Big > Small > Tiny$). The second term of the equation considers the requests not fulfilled by the player, i.e., the number of requests denied by the provider ($Qty_{request}^{reject}$). This amount is multiplied by a value ($C_{request}^{reject}$) that represents the cost that the player will be burdened for not fulfilling a request. The history of each player is represented by the number of times it had its score above the minimum limit of the division at the end of each round.

$$s_j = (\sum_{k=t,s,b} (p_k \times I_{VM_k}^{execution})) - (Qty_{request}^{reject} \times C_{request}^{reject}) \quad (1)$$

An initial score was also assigned to the new players, defined as the harmonic mean of the scores of all the players present in the access division. The definition of this value had as a premise to avoid that the player is disadvantaged in the environment or that stands out exaggeratedly in front of the others players as soon as it enters the tournament. If the disadvantage was not considered, a new player when entering the access division (depending on the dynamics of the division) could be eliminated in the next round even though it should remain in the tournament. On the other hand, if a new player is offered a lot of advantage, in the next round it could be raised to higher divisions without having collaborated to the other players in the tournament.

6 SIMULATION AND PRELIMINARY RESULTS

The purpose of the simulation is to verify, as a proof of concept, basic situations regarding the design and objectives of the MCT. This proof of concept helped validate and guide the research and development of MCT. The basic situations were: i) permanence of the players in the MCT; ii) flow of players through the divisions; and iii) elimination of free rider players from the tournament. To perform this verification, eight types of players identified as $A, B, \ldots, H$ have been implemented. All players were implemented in Python and based on the features of OpenStack (OpenStack Foudantion, 2015). Each type of player is distinguished from the others by the amount and flavors (Table 1) they accept. Another characteristic of the players is the possibility of choosing which division they wants to achieve. The types and characteristics of each player are shown in the Table 2.

The players were arranged in a set of eight test cases. In each test case there are five instances of certain types of players (Table 2) that send requests for VM_{big}, VM_{small}, and VM_{tiny} every 4 minutes (value obtained empirically for the initial tests). The execution time of each type of VM is described in Table 1 (value obtained empirically). Since there are divergences between request period and VM execution time, some players can refuse new VM instances request at certain occasions.

The performed tests were arranged as follows:

Table 2: Types of players. Type B, C, and F have typical free riders behaviors. The FINAL DIVISION field indicates which division a player expects to achieve.

PLAYER'S TYPE				
PLAYER' TYPE	QTDE OF VM_{Big}	QTY OF VM_{small}	QTY OF VM_{tiny}	FINAL DIVISION
A	10	0	0	First Division
B (free rider)	0	10	0	First Division
C (free rider)	0	0	10	First Division
D	10	10	0	First Division
E	10	0	10	First Division
F (free rider)	0	10	10	First Division
G	10	10	10	First Division
H	10	10	10	Variable

- Test 1 and 5:
  - Description: in test 1 there are five players in the tournament who only accept requests for virtual machines of type *Tiny*. In test 5, the five players accept requests for $VMs_{Tiny,Big}$.
  - Behavior: the chart of Figure 4 describes a cyclic variation of players across divisions. Each player after being promoted to intermediate division is put into a playoff state returning to the access division after the number rounds has been exhausted.
  - Discussion: cyclic behavior happens because when entering the second division, the players generate negative *scores* because they do not accept requests of virtual machines of *Small* type. This returns them to the previous division once the number of rounds of permanence in the playoff ends.

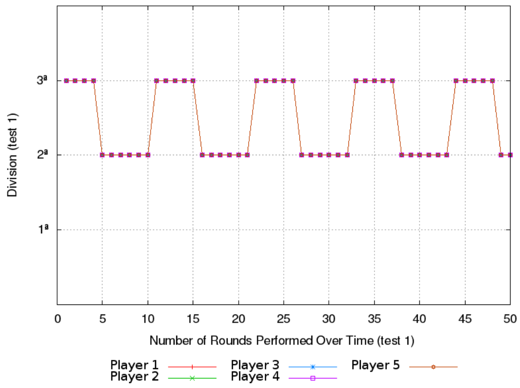

Figure 4: Behavior of players in test cases 1 and 5.

- Test 2, 3, and 6:
  - Description: there are five players in the tournament that do not accept requests for virtual machines of type *Tiny*;
  - Behavior: because players do not accept requests from instances of VMs_{Tiny}, they are quickly eliminated from the competition as they do not have enough *score* nor enough *history* to enter playoff status (Figure 5).
  - Discussion: the behavior of the five players in the three test cases is similar to that of free riders who only expect to consume resources. Thus, as expected, the MCT excludes this type of player as quickly as possible.

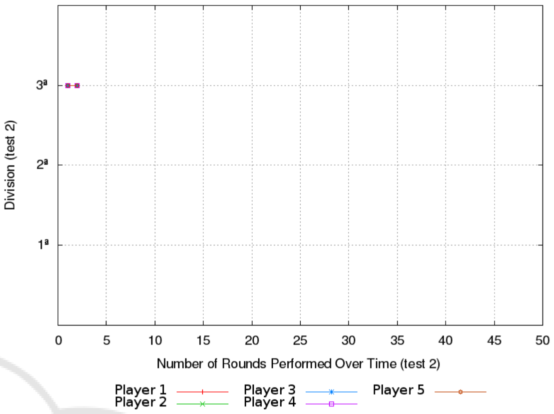

Figure 5: Evolution of the five players in the test cases 2, 3, and 6.

- Test 4:
  - Description: there are five players in the tournament that accept requests for virtual machines of types *Tiny* and *Small*. All players intend to reach the premier division.
  - Behavior: as the five players accept requests for virtual machines of types *Tiny* and *Small*, they can reach at most the second division. There is a recurring behavior of rise to the premier division and back to the second division.
  - Discussion: this behavior comes from the fact that no player is able to generate good scores when they are in premier division. Good scores are not generated because the five players do not accept requests for virtual machines of type *Big*. This behavior is described in Figure 6.
- Test 7:
  - Description: there are five players in the tournament. These players accept requests from all types of *VMs*. All players intend to reach the premier division.
  - Behavior: the chart (Figure 7) shows the evolution of the five players for the three divisions available in the MCT instance. As the five players accept all types of *VM* requests, and there is no restriction on permanence in divisions or variation of behavior, they reach the premier division and remain in it indefinitely.

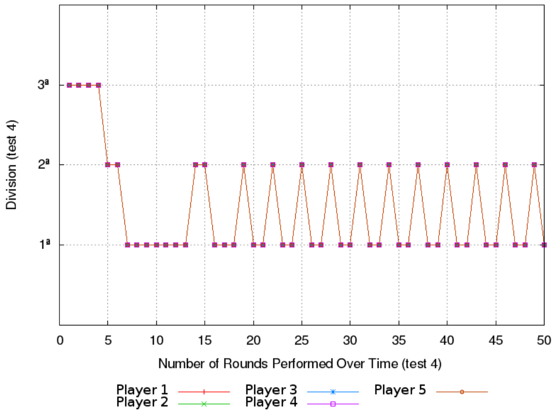

Figure 6: Cyclical behavior between divisions 2 and 1.

- Discussion: at first, the result of this test is presented as the ideal to MCT execution. However, this situation can lead to a reduction in the incentive to resource offers and specialization of the environment. If this behavior is observed and it is not welcome, it may be considered the creation of dynamic divisions that change in quantity and rules according to the overall behavior of the tournament.

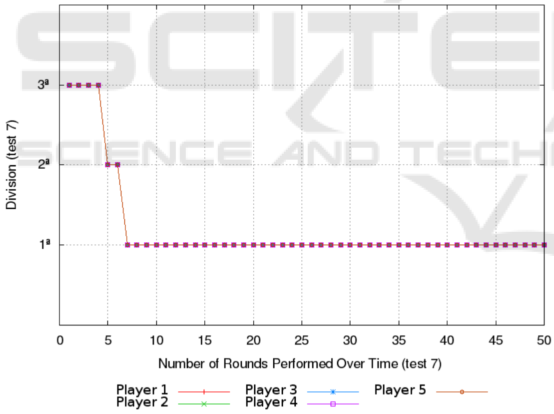

Figure 7: All players accept requests by $VMs_{Tiny,Small,Big}$.

- Test 8:
  - Description: test case similar to test 7, however these players differ by the division they intend to reach: player 1 - premier division; players 2 and 3 - intermediate division; and the other players - access division.
  - Behavior: as shown in the graph of Figure 8, groups of players develop distinct behaviors throughout rounds even though they have the ability to reach the premier division. This is because each player explicitly defines which division it wants to reach.
  - Discussion: the definition of division is a proactive property of each player and should be indicated if appropriate. In addition, each player may not accept a subset of rules from superior divisions and thus remain in the current division. With this property players can choose which division to stay.

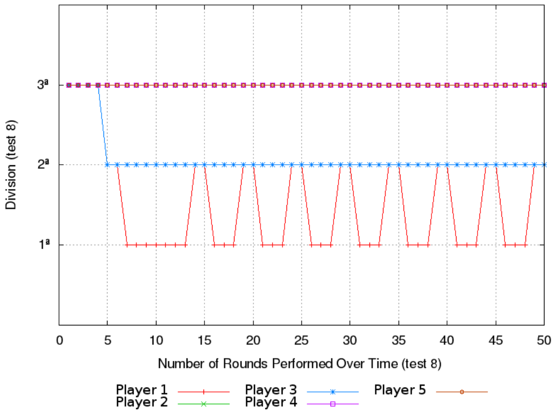

Figure 8: Distribution of players in the divisions.

7 CONCLUSION

The increasing use of Cloud Computing in different domains has been highlighting some paradigm limitations while leveraging new approaches to overcome these limitations. One such approach is the association of multiple providers called Inter-Clouds. However, these resource-sharing environments are suitable to the appearance of elements called free riders, which focus only on satisfying their own interests without regard to the other elements present in the environment. In this scenario, this paper proposed an Inter-Clouds architecture called MultiClouds Tournament that has as premisses the main characteristics of Cloud Federations. The MCT intends to encourage the consumption and supply of resources by providers through a tournament. In this way, it is expected to eliminate the free riders of this type of resource sharing environment. The literature shows that this is still an open challenge, and the preliminary results of this work point to an effective development of MCT.

7.1 Future Works

As future work, we will map the impacts of the actions that eventual free riders can perform inside the divisions in the period from the beginning of the behavior to the elimination of it from the tournament. This mapping will be analyzed and depending on the results will be considered the use of a mechanism to mitigate the action of these elements. It will also be evaluated the use of means that allow more specialized providers to be led to the superior division, avoiding to interpret them as free riders. The impact of the insertion of adaptive thresholds (lower and upper limits) in each division will be verified. These adaptive thresholds will be based on the players' $offering \times consumption$ relationship dynamics inside of each division (or entire tournament). In addition, it is expected that adaptative thresholds will help maximize this relation while avoiding players who offer the minimum resources to stay or evolve in the tournament. Also as future work, the implementation of a functional version of the MCT will be finalized and evaluated in a real environment.

ACKNOWLEDGMENTS

This research was partially funded by the European Commission H2020 programme under grant agreement no. 688941 (FUTEBOL), as well from the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through Brazilian National Research and Educational Network (RNP) and CTIC. The authors also would like to thank CNPq and CAPES for the financial support.

REFERENCES

Andrade, N., Brasileiro, F., Cirne, W., and Mowbray, M. (2004). Discouraging free riding in a peer-to-peer cpu-sharing grid. In *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*, pages 129–137.

Assis, M. R. M. and Bittencourt, L. F. (2015). Multiclouds tournament blueprint. In *IEEE/ACM 8th International Conference on Utility and Cloud Computing*, Limassol, Cyprus.

Assis, M. R. M., Bittencourt, L. F., Tolosana-Calasanz, R., and Lee, C. A. (2016). Cloud federations: Requirements, properties and archictectures. In Kecskemeti, G., Kertesz, A., and Nemeth, Z., editors, *Developing Interoperable and Federated Cloud Architecture*, chapter 1, pages 1–41. IGI Global, 701 East Chocolate Avenue, Hershey, PA 17033, USA.

Berman, F., Fox, G., and Hey, A. J. G. (2003). *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, Inc., New York, NY, USA.

Bittencourt, L. F. and Madeira, E. R. M. (2011). HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. *J. Internet Services and Applications*, 2(3):207–227.

Buyya, R., Ranjan, R., and Calheiros, R. N. (2010). Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Proceedings of the 10th International Conference on Algorithms and Architectures for Parallel Processing - Volume Part I*, ICA3PP'10, pages 13–31, Berlin, Heidelberg. Springer-Verlag.

Celesti, A., Tusa, F., Villari, M., and Puliafito, A. (2010). How to Enhance Cloud Architectures to Enable Cross-Federation. *In 2010 IEEE 3rd International Conference on Cloud Computing*.

Chaurasiya, V. K., Srinivasan, K., Thyagarajan, K., Govil, S. B., and Das, S. (2012). An approach to identify the optimal cloud in cloud federation. *Institute of Advanced Engineering and Science*, Vol 1, No 1 (2012).

Falcão, E. d. L., Brasileiro, F., Brito, A., and Vivas, J. L. (2015). Incentivising resource sharing in federated clouds. In Bessani, A. and Bouchenak, S., editors, *Distributed Applications and Interoperable Systems*, volume 9038 of *Lecture Notes in Computer Science*, pages 45–50. Springer International Publishing.

Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop*, pages 1–10. IEEE.

Foundation, T. A. S. (2015a). Apache jclouds – the java multi-cloud toolkit. https://jclouds.apache.org/. Accessed: 2015-28-02.

Foundation, T. A. S. (2015b). Apache libcloud – one interface to rule them all. https://libcloud.apache.org/. Accessed: 2015-28-02.

GICT (2010). Use cases and functional requirements for inter-cloud computing. Technical report, Global Inter-Cloud Technology Forum.

Grozev, N. and Buyya, R. (2012). Inter-Cloud Architectures and Application Brokering: Taxonomy and Survey. *Software: Practice and Experience*.

Hardin, G. (1968). The tragedy of the commons. *Science*, 162(3859):1243–1248.

Joe-Wong, C., Im, Y., Shin, K., and Ha, S. (2016). A performance analysis of incentive mechanisms for cooperative computing. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 108–117.

Keahey, K., Tsugawa, M., Matsunaga, A., and Fortes, J. (2009). Sky computing. *IEEE Internet Computing*, 13(5):43–51.

Khethavath, P., Thomas, J., Chan-Tin, E., and Liu, H. (2013). Introducing a distributed cloud architecture with efficient resource discovery and optimal resource allocation. In *2013 IEEE Ninth World Congress on Services*, pages 386–392.

Kurze, T., Klems, M., Bermbach, D., Lenk, A., Tai, S., and Kunze, M. (2011). Cloud federation. In *Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2011)*. IARIA.

Manno, G., Smari, W. W., and Spalazzi, L. (2012). Fcfa: A semantic-based federated cloud framework architecture. In *HPCS*, pages 42–52.

Mashayekhy, L., Nejad, M., and Grosu, D. (2015). Cloud federations in the sky: Formation game and mechanism. *Cloud Computing, IEEE Transactions on*, 3(1):14–27.

Mell, P. and Grance, T. (2009). The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology, Information Technology Laboratory.

OpenStack Foudantion (2015). OpenStack Framework. https://www.openstack.org/. Accessed: 2016-15-01.

Ostrom, E., Burger, J., Field, C. B., Norgaard, R. B., and Policansky, D. (1999). Revisiting the commons: local lessons, global challenges. *science*, 284(5412):278–282.

Panarello, A., Celesti, A., Fazio, M., Villari, M., and Puliafito, A. (2014). A requirements analysis for iaas cloud federation. In *Proceedings of the 4th International Conference on Cloud Computing and Services Science*, CLOSER 2014, pages 584–589, Portugal. SCITEPRESS - Science and Technology Publications, Lda.

Petri, I., Beach, T., Zou, M., Montes, J. D., Rana, O., and Parashar, M. (2014). Exploring models and mechanisms for exchanging resources in a federated cloud. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 215–224.

Toosi, A. N., Calheiros, R. N., and Buyya, R. (2014). Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Comput. Surv.*, 47(1):7:1–7:47.

Toosi, A. N., Calheiros, R. N., Thulasiram, R. K., and Buyya, R. (2011). Resource provisioning policies to increase iaas provider's profit in a federated cloud environment. In *Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications*, HPCC '11, pages 279–287, Washington, DC, USA. IEEE Computer Society.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.