Resource Self-management under an SLA within a Cloud Networking Environment

Mohamad Hamze, Nader Mbarek, Olivier Togni

Le2i Laboratory UMR 6306 CNRS
University of Burgundy
Dijon, France
e-mail: {Mohamad.Hamze, Nader.Mbarek, Olivier.Togni}@u-bourgogne.fr

***Abstract*—Today, cloud networking is one of the recent research areas in the cloud computing research communities. The main drawback of cloud networking consists in the lack of Quality of Service (QoS) guarantee and management in conformance with a corresponding Service Level Agreement (SLA). In this paper, we propose a framework for self-establishing an end-to-end SLA between a Cloud Service User (CSU) and several Cloud Service Providers (CSPs) in a cloud networking environment (inter-cloud Broker and Federation architecture). Then, we propose the self-management of cloud resources under the established SLA using specific autonomic cloud managers. We simulate our proposed framework to provide videoconferencing and intensive computing applications with self-management and QoS guarantee. We observe that the Broker architecture is the most economical, while ensuring QoS requirements.**

Keywords–Cloud Networking; Autonomic Computing; Self-management; Service Level Agreement; Quality of Service.

I. INTRODUCTION

Cloud computing is a promising technology for the realization of large, scalable and on-demand computing infrastructures. Many enterprises are adopting this technology to achieve high performance and scalability for their applications while maintaining low cost. However, a CSU that can be an end-user, an organization, a Software as a Service (SaaS) provider, or a Platform as a Service (PaaS) provider, requires for its services an end-to-end QoS guarantee with a high level reliability and a continued availability. In addition, cloud computing success requires that CSUs and CSPs can be confident that established SLAs are supporting their respective business activities to their best extent. However, an SLA may be violated when using a single CSP model due to unpredictable workload, resource failure and security attack.

Thus, geographical distributed data centers offer better end-to-end performance between CSU and CSP, while improving reliability when failure occurs. However, the inter-cloud should be designed as a multi-vendor environment with the ability to migrate services from one provider to another and to locate the best resources not only in terms of computing capacity and storage, but also connectivity, bandwidth and delay. Thus, the networking aspect of cloud computing is a critical factor.

In this context, cloud networking is defined as the ability to connect the user to his cloud services and interconnect services within an inter-cloud. It is built upon two main concepts, the integration of the networking resources onto existing data centers and the deployment of distributed computing and storage resources in the network [1]. It is difficult to guarantee the QoS of data transfer in cloud networking [2]. Thus, a convenient solution is to use a Bandwidth on Demand (BoD) service provided by a Network as as Service (NaaS) CSP. In such an environment, the development of an autonomic cloud control is necessary to simplify the complexity, maximize efficiency and minimize user interactions.

In this paper, we propose a framework for self-establishing an end-to-end SLA and self-managing CSU resources in a cloud networking environment thanks to the specification of an autonomic cloud networking architecture and the Autonomic cloud Managers description with their interactions and lifecycle. Then, we enable videoconferencing and intensive computing applications to take full advantage of our framework.

The remainder of this paper is organized as follows: Section II highlights the most relevant research works and trends in this area. In Section III, we present an overview of our proposed cloud networking architecture. In Section IV, we describe our autonomic cloud networking framework including autonomic cloud managers, SLA self-establishment and self-management lifecycle with cost calculation. Section V presents videoconferencing and intensive computing usage cases and the framework evaluation. Lastly, Section VI concludes the paper and points out future work.

II. RELATED WORK

The research project Scalable & Adaptive Internet Solutions (SAIL) [3] describes a cloud networking architecture and focuses on security, but it does not consider the QoS and SLA. The research project Foundation of Self-governing ICT Infrastructures (FoSII) [4] is proposing solutions for autonomic management of SLAs in the cloud. In addition, the research project Contrail [5] aims to vertically integrate an open-source distributed operating system for autonomous resource management in Infrastructure as a Service (IaaS) environments and PaaS systems. We consider our work to be very much in alignment with the objectives of these projects. But, our research work is innovative by considering the resource self-management under a self-established SLA in a cloud networking environment, and by focusing on minimizing service cost with QoS guarantee for IaaS and NaaS services.

From standardization perspective, IEEE Cloud Computing formed the Inter-Cloud Working Group (ICWG). It announced the launch of two new standards development projects: P2301 [6], a guide for Cloud Portability and Interoperability Profiles (CPIP) and P2302 [7], a Standard for Intercloud Interoperability and Federation (SIIF). Open Grid Forum (OGF) is active in the definition of the Open Cloud Computing Interface (OCCI) [8] for the interoperability between clouds. Global Inter-Cloud

Technology Forum (GICTF) [9] studies the standard Inter-Cloud interfaces to improve the reliability of the Clouds, and presents SLA metrics for Inter-Cloud environments. IBM presented in 2011 CloudNaaS [10], a cloud networking platform for enterprise applications. In our research work, we propose to develop a cloud networking framework and we aim to enable communications not only between CSPs Data Centers (DC), but also between CSU, CSPs (DC) and CSPs offering BoD. For that purpose, we use Web Services standard technologies.

Finally, there are many related research works on QoS [11][12], but QoS mentioned in these works is for SaaS, PaaS or IaaS. Smit et al. [13] present a methodology for an implementation of a service-oriented application that provides relevant metadata information describing offered cloud services via a uniform RESTful web service. In addition, several research works present SLA for cloud computing only: the project Mycloud [14] proposes Cloud Service Level Agreement (CSLA) and Patel et al. [15] propose to use Web Service Level agreement (WSLA) [16] in a cloud computing context. However, our research work considers, in addition, NaaS services and an SLA within different QoS attributes for autonomic cloud networking environment.

III. Cloud Networking Architecture Overview

A. Architecture Description

We consider two kinds of architectures, one for inter-cloud Broker and the other for inter-cloud Federation. We assume an environment with multiple CSPs interconnected together. Within these architectures, we ensure consistency between the QoS requirements requested by the CSUs, and service levels proposed by CSPs to allow multiple CSPs working together to meet the CSU requirements.

In the proposed cloud networking Broker architecture (Figure 1), the Cloud Broker is emerged as an intermediate entity between a CSU and CSPs to help the establishing of a service level that meets the CSU requirements. In addition, we have proposed two kinds of CSPs. The first one is the CSP (BoD) providing BoD network service (e.g., network operator) and playing the role of a NaaS CSP. The second one is the CSP (DC) providing IaaS and NaaS services. The IaaS service concerns Virtual Machine (VM) and storage resources and the NaaS service concerns network DC resources. The CSP (DC) can offer resources from one or several data centers. Moreover, CSPs can offer different service levels for example (Platinum, Gold, Silver, and Bronze), each one with different QoS guarantee and cost.

However, in the proposed cloud networking federation architecture, the federation provides an alliance among several CSPs (DC/BoD) that join together to help the establishing of service level that meet the CSU requirements. In addition, CSPs provide IaaS and/or NaaS services with different service levels. Furthermore, we consider that the CSU is connected to a Lead Cloud Service Provider CSP_L.

Within the Broker architecture, we consider three types of SLA constructed using the XML language for interoperability and portability between different entities:

1) inter-cloud Service Level Agreement (iSLA): it is a contract between a CSU and a Cloud Broker. It guarantees QoS for NaaS (BoD and/or DC) and/or IaaS (VMs and/or storage) services. QoS parameters could be quantitative or qualitative. In addition, it contains cost when violations occur.

2) BoD inter-cloud Service Level Agreement (B_iSLA): it is a contract between a cloud Broker and a CSP (BoD) interconnecting CSU sites, CSPs (DC) or connecting CSU sites to CSP (DC). It guarantees QoS for NaaS (BoD) services.

3) Datacenter inter-cloud Service Level Agreement (D_iSLA): it is a contract between a cloud Broker and a CSP (DC) for NaaS and IaaS services.

Moreover, in the Federation architecture, we consider the same three types of SLA. However, the iSLA is a contract between CSU and CSP_L, the D_iSLA is a contract between CSP_L (DC) and CSPs (DC), and the B_iSLA is a contract between CSPs (BoD) or between CSPs (DC) and CSPs (BoD) that enable CSU sites to reach them. In addition, we have two scenarios, the CSP_L can meet the CSU requirements without other CSPs resources usage (scenario 1) or using other CSPs resources in the alliance (scenario 2).

B. Problem Statement and Proposed Algorithms

1) Problem Statement: Constraint Optimization Problem: in general, the CSU requests IaaS services with or without NaaS services to run specific applications with QoS guarantee for its different sites using Brokerage or Federation services. Our goal is to minimize the cost subject to QoS Constraints. We must ensure a feasible and optimal CSPs selection. A feasible selection means that aggregated QoS values from selected CSPs satisfy the global CSU QoS requirements. Then, we consider as an optimal selection the feasible one minimizing the overall cost value (we propose two algorithms: Algs. 1 and 2). However, if we have the same minimum cost for different offers, our goal becomes to maximize a utility function for these offers to select the optimal one using different normalized weights w_i assigned by the CSU for each i-th QoS parameter based on its importance and the type of application. For that purpose, we specify two algorithms (Algs. 1 and 2) [17] that ensure best CSPs (DC) selection offering IaaS resources with best path selection between CSU sites and selected CSPs (DC).

2) Optimization and Path Selection Algorithm: the Cloud Broker in Broker architecture or the CSP_L (scenario 1) with other CSPs (DC) (scenario 2) in Federation architecture select the best CSPs (BoD) in terms of minimal cost and NaaS QoS guarantee using a proposed optimization and path selection algorithm (Alg. 1), subject to NaaS QoS parameters constraints when the CSU requests IaaS with NaaS QoS guarantee, or subject to minimal cost in case of IaaS only. Indeed, Alg. 1 calculates and sorts the cost of different service levels combinations offered by CSPs (BoD) for each route between each site and a specific CSP (DC) while guaranteeing the CSU QoS requirements. Finally, Alg. 1 selects the route corresponding to the minimal cost value.

3) Optimization and Resource Selection Algorithm: when the CSU requests IaaS with/without NaaS services, it interacts with the Cloud Broker or the CSP_L specifying its requirements. Then, the Cloud Broker or the CSP_L selects best IaaS CSP resources in terms of minimal cost and IaaS QoS guarantee using a proposed optimization and resource selection algorithm (Alg. 2) subject to IaaS QoS parameters constraints. At first, using Alg. 1, the Cloud Broker in Broker architecture gets best routes between CSU sites and each CSP (DC), whereas in Federation architecture, only the CSP_L (scenario 1) or the latter with other CSPs (DC) (scenario 2) get best routes between CSU sites and themselves. Furthermore, the

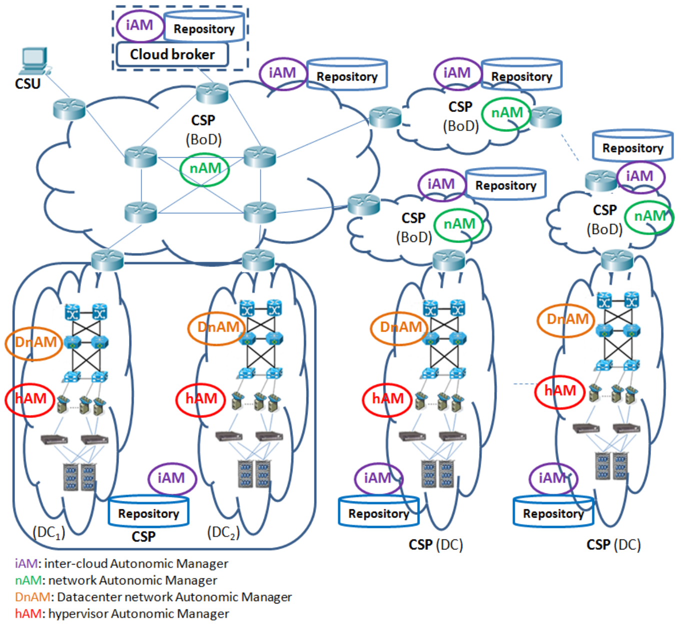
iAM
Repository
Cloud broker
CSU
CSP (BoD)
nAM
DnAM
hAM
(DC1)
CSP
(DC2)
CSP (DC)
iAM: inter-cloud Autonomic Manager
nAM: network Autonomic Manager
DnAM: Datacenter network Autonomic Manager
hAM: hypervisor Autonomic Manager

Figure 1. Autonomic Cloud Networking Architecture.

Cloud Broker or the CSP_L selects best resources that meet the CSU requirements while taking into account best routes (Alg. 1). Note that in the federation architecture, the CSP_L selects at first its available resources then it selects remaining resources in other CSPs (DC) if needed.

IV. Autonomic Cloud Networking Framework

Due to the on-demand self-service characteristic of cloud computing, it is required that a cloud infrastructure supports SLA self-establishment and resource self-management. In this section, we propose to self-manage resources and establish iSLA, B_iSLA and D_iSLA in our cloud networking architectures autonomously using Autonomic cloud Managers (AMs).

A. Autonomic Cloud Networking Architecture

In general, a domain refers to resource collections managed by a single entity, e.g., cloud Data Center (DC) or communication network. If we manage this domain in an autonomic manner, we call it Autonomic cloud Domain (AD). In this context, we present the proposed cloud networking architecture with several ADs (Cloud Broker or CSPs (DC/BoD)). Figure 1 represents this architecture including a cloud Broker entity for the Broker scenario and without it for the Federation scenario. Each AD is under the authority of an inter-cloud Autonomic Manager (iAM). iAM communicates with other iAMs to achieve an agreement on a service level. In addition, it controls one or more low level AMs to configure resources in conformance with the agreed service level. These AMs are playing different roles within our autonomic architecture:

1) network Autonomic Manager (nAM): it is responsible for creating and managing CSP (BoD) virtual networks and monitoring workload and performance in conformance with the agreed B_iSLA.

2) Datacenter network Autonomic Manager (DnAM): it is responsible for creating and managing CSU virtual networks within the cloud DC and monitoring workload and performance in conformance with the NaaS part of D_iSLA.

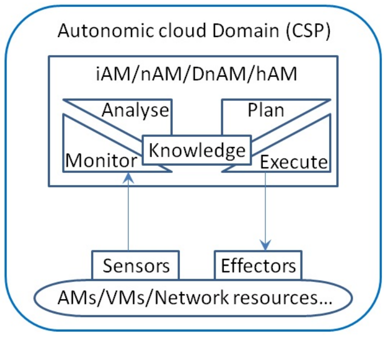

Figure 2. Autonomic cloud Manager Functional Details.

3) hypervisor Autonomic Manager (hAM): it is responsible for creating and managing VMs and storage capacities in conformance with the IaaS part of D_iSLA. Therefore, the CSP (DC) can consequently decide the allocation or deallocation of resources to maintain an acceptable performance.

We provide these autonomic cloud managers with the capability to achieve an agreement between ADs. This agreement covers QoS aspects for different cloud service models such as IaaS and NaaS with different service levels. In addition, the iAM entities use a repository to store resource management information and to facilitate their interactions with other AMs.

B. Autonomic cloud Manager Description

An Autonomic cloud Manager (Figure 2) has to know its environment and how to keep it in optimal conditions without the need of any external operation. Therefore, our proposed AM (iAM, nAM, DnAM, or hAM) can manage a single resource or set of resources (AMs, VMs, storage and network resources, etc.) thanks to sensors and effectors interfaces.

After integrating various policies in its knowledge base (thresholds, algorithms for best cloud resource selection, etc.), the AM begins with the monitoring phase (QoS parameter values, etc.) to ensure data collection, aggregation, filtering and reporting from managed resources thanks to sensor interfaces. Then, the collected data is passed to the analysis phase to correlate these data in accordance with the knowledge base policies (QoS parameter violation, failure, congestion, etc.). Then a request for a change could be sent to the planning phase to indicate actions needed to achieve specific objectives in accordance with specified policies (cloud resource allocation or releasing, etc.). Finally, these actions are sent to the execution phase that allows changes to be made in the managed cloud resource thanks to effectors interface (resource configuration, VM migration, etc.). In addition, the changes are checked to update the knowledge base by the monitoring phase. These phases constitute the closed control loop (MAPE-K) of cloud resource self-management implemented by our AMs.

C. Autonomic Cloud Managers Interactions

To provide our cloud networking architecture with autonomy while offering an end-to-end QoS guarantee, two kinds of interactions could take place between autonomic cloud managers (Figure 3 for Broker scenario and Figure 4 for Federation scenario). Thanks to the first one, an iAM initiates a peer to peer communication process with the corresponding

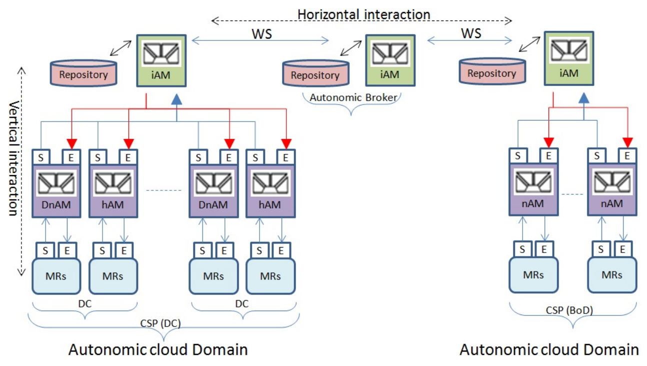

Figure 3. AM Interaction Framework for Broker scenario.

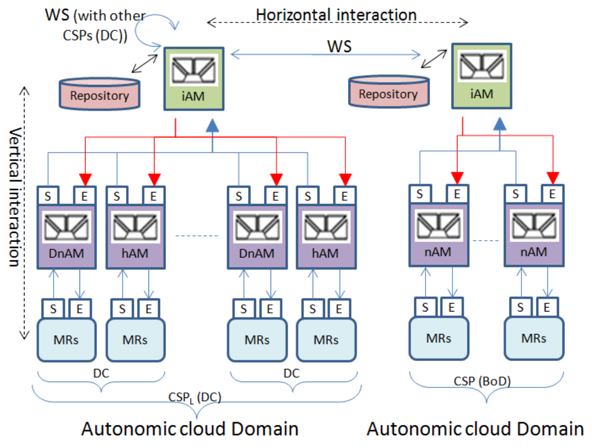

Figure 4. AM Interaction Framework for Federation scenario.

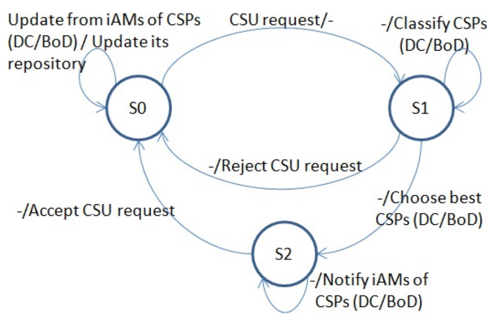
Update from iAMs of CSPs (DC/BoD) / Update its repository
CSU request/-
-/Classify CSPs (DC/BoD)
S0
S1
-/Reject CSU request
-/Accept CSU request
-/Choose best CSPs (DC/BoD)
S2
-/Notify iAMs of CSPs (DC/BoD)

Figure 5. FSM for Broker iAM Lifecycle.

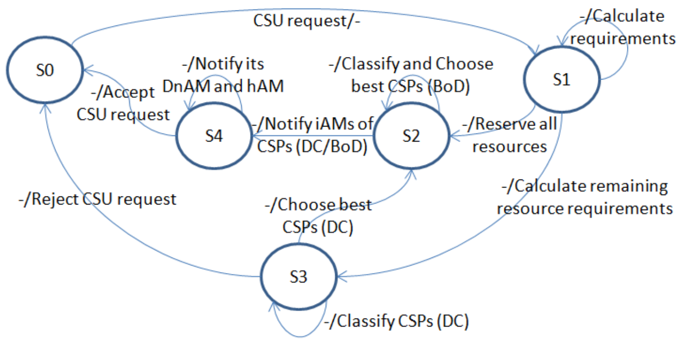

Figure 6. FSM for CSP_L iAM Lifecycle.

iAMs in a horizontal interaction using Web Services (WS) technologies to achieve an agreement on a service level. In addition, each iAM is responsible of the service level guarantee within the corresponding AD. This guarantee will be possible thanks to a second kind of interaction. Indeed, the iAM controls one or more low level Autonomic cloud Managers (nAM, DnAM or hAM) thanks to the manageability interfaces (effectors and sensors) using WS technologies to achieve this service level guarantee. Therefore, iAM provides these low level AMs with the corresponding service level (B_iSLA or D_iSLA) in a vertical interaction, so that they use a similar interaction to allocate, release, or modify the configuration of their Managed Resources (MRs: router, VMs, storage, etc.) according to the received service level.

D. SLA Self-establishment Lifecycle

The following Finite State Machines (FSMs) presents the lifecycle of the proposed AMs for SLA self-establishment:

1) Broker Architecture: The FSM concerning Broker iAM (Figure 5) includes three states. In the first state S0, the Broker iAM receives periodically available services with different service levels or any changes from CSPs (DC/BoD) iAMs in the alliance. Then, it updates its repository. After receiving CSU service requirements to construct an iSLA, the Broker iAM goes to the second state S1.

In state S1, the Broker iAM consults its repository and compares the CSU requirements with different services and their corresponding service levels offered by CSPs iAMs to select the appropriate CSPs that meet the CSU QoS requirements according to the proposed optimization and selection algorithms (see Sec. III-B). If the Broker iAM does not find any CSPs that meet the CSU requirement, it rejects the CSU request and goes to the initial state S0. Else, the Broker iAM chooses the suitable CSPs, and goes to state S2.

In state S2, the Broker iAM notifies each iAM of selected CSPs and establishes a D_iSLA and a B_iSLA with respectively each CSP (DC) iAM and CSP (BoD) iAM. Then, CSPs (DC) iAMs and CSPs (BoD) iAMs provide their AMs with the corresponding service level to allocate resources and deliver IaaS and/or NaaS services with QoS guarantee according to the received service level (B_iSLA or D_iSLA). Finally, the Broker iAM accepts the CSU request, establishes the iSLA, and goes to the initial state S0.

2) Federation Architecture: The FSM that we specify for the CSP_L iAM (Figure 6) includes five states. In state S0, when the CSP_L iAM receives a CSU request, state S1 is reached to calculate resource requirements. In state S1, if the CSP_L can meet all CSU requirements, the corresponding CSP_L iAM goes to state S2 to reserve resources. Else, it calculates remaining resource requirements and goes to state S3. In this state, it contacts other CSPs (DC) iAMs in order to meet remaining CSU resource requirements. On the other hand, each iAM of these CSPs (DC) selects best CSPs (BoD) according to the proposed optimization and path selection algorithm. Also, it describes IaaS and NaaS services for available resources with different service levels and sends all information to the CSP_L iAM. Then, if the CSP_L iAM does not find any CSPs that meet requirements, it rejects the CSU request and goes to the initial state S0. Else, it selects best

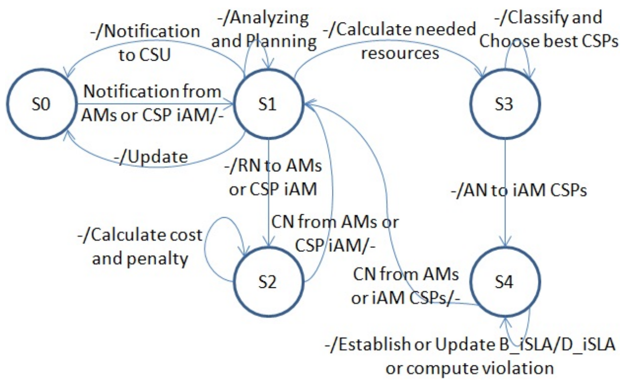
-/Notification to CSU
-/Analyzing and Planning
-/Calculate needed resources
-/Classify and Choose best CSPs
S0
Notification from AMs or CSP iAM/-
S1
S3
-/Update
-/RN to AMs or CSP iAM
-/AN to iAM CSPs
-/Calculate cost and penalty
CN from AMs or CSP iAM/-
S2
CN from AMs or iAM CSPs/-
S4
-/Establish or Update B_iSLA/D_iSLA or compute violation

Figure 7. FSM of Broker or CSP_L iAM Control Loop Lifecycle.

CSPs that meet CSU QoS requirements for IaaS with/without NaaS services according to the proposed optimization and resource selection algorithm and goes to state S2.

In state S2, the CSP_L iAM classifies and selects best CSPs (BoD) that enable CSU sites to reach it according to the proposed optimization and path selection algorithm. Then, the CSP_L iAM only (scenario 1) or the latter with iAMs of selected CSPs (DC) (scenario 2) notify iAMs of selected CSPs (BoD) in order to allocate BoD network resources and establish B_iSLA with them. In addition, each iAM of CSPs (BoD) provides its nAM with the corresponding service level to deliver NaaS services with QoS guarantee. Moreover, in scenario 2, the CSP_L iAM sends a request to establish a D_iSLA with iAMs of selected CSPs (DC). Then, each concerned CSPs (DC) iAMs provide their DnAM and hAM with the corresponding service level to deliver IaaS and/or NaaS services with QoS guarantee.

After this, the CSP_L iAM goes to state S4 while notifying its hAM and DnAM to reserve and configure VMs, storage and network resources in order to deliver IaaS and NaaS services with QoS guarantee. Finally, the CSP_L iAM accepts the CSU request, establishes the iSLA, and goes to initial state S0.

E. Self-management Lifecycle

After the SLA self-establishment, the objective is to guarantee cloud service performances by periodically monitoring the CSU agreed SLA at CSPs using AMs sensors and policies defined by each AD to avoid SLA violation. In addition, the strategy of detecting SLA violations is based on the use of predefined violation thresholds specified by each AD. On the one hand, a violation threshold is a value indicating the least acceptable performance level for an application in conformance with the agreed SLA and real-time monitoring information. Therefore, exceeding the violation threshold value for a particular QoS parameter indicates the occurrence of SLA violation and the system logs the necessary information for calculating the appropriate penalties. On the other hand, a risk threshold is a value indicating the risk of a violation for a performance level. Therefore, with this information the system can react quickly to avoid the violation threat and save the CSP from costly SLA penalties. Moreover, each AD specifies a threshold that indicates if the CSU have exceeded the maximal traffic throughput defined in the iSLA.

When an unexpected surge of access to cloud services occurs, there is a risk of degradation in QoS parameters at one or more CSPs (DC/BoD). Therefore, when a risk concerning a QoS parameter violation is detected at a CSP, available resources in this CSP or in other CSPs must be autonomously discovered and reserved with self-optimization and self-reconfiguration functions to solve this problem thanks to the control loop and AMs interactions using our proposed optimization and selection algorithms. Thus, the cloud self-management process is described thanks to an FSM including five states as presented in Figure 7.

1) State S0: in this state, the Broker iAM in Broker architecture is waiting to receive a notification from an iAM of a CSP to go to the second state S1. However, in Federation architecture, the CSP_L iAM is waiting to receive a notification from an iAM of a CSP or from AMs under its control (nAM, DnAM or hAM) to go to the second state S1. This notification collected by iAMs sensors can be a service termination, a risk for QoS parameter violation, an exceeding of throughput threshold, or an update at the AD of a CSP_L or a CSP.

2) State S1: in this state, the Broker or CSP_L iAM analyzes the notification and plans to react with the appropriate actions. Thus, if the notification is a service termination, the Broker or the CSP_L iAM sends a Release Notification (RN) to the CSP_L AMs or to a CSP iAM according to the established B_iSLA or D_iSLA to release resources and goes to state S2. However, if the notification is a risk of QoS parameter violation based on the QoS parameter threshold, the Broker or the CSP_L iAM can resolve the problem by allocating new resources free of charge to avoid violation. Therefore, it calculates needed resources and goes to state S3. Note that, we consider that a CSP cannot react only to a risk of QoS parameter violation to resolve the problem. Otherwise, it resolves this problem by allocating new resources in its AD in conformance with the established D_iSLA or B_iSLA. Then, it sends an update notification to the Broker or the CSP_L iAM that updates its repository and goes to state S0.

3) State S2: in this state, the Broker or the CSP_L iAM calculates the cost of terminated resources and penalties based on violations and the established B_iSLA or D_iSLA. Then, the CSP_L AMs or a CSP iAM release these resources using vertical AMs interaction and the corresponding CSP pays costs to the Cloud Broker or to the CSP_L and sends a Confirmation Notification (CN) to the Broker or the CSP_L iAM. Next, the Broker or the CSP_L iAM goes to the state S1.

4) State S3: in this state, the Broker or CSP_L iAM selects best CSPs based on the proposed optimization and selection algorithms in conformance with the established iSLA. Then, it sends an Allocation Notification (AN) to each iAM of selected CSPs to allocate resources and goes to the state S4.

5) State S4: in this state, the Broker or CSP_L iAM establishes or updates the B_iSLA or the D_iSLA. In addition, when the Broker or CSP_L iAM cannot avoid the violation, it considers this violation to calculate the penalty when the service is terminated. Moreover, the CSP iAM, CSP_L AMs or each iAM of selected CSPs send a CN to the Broker or CSP_L iAM that enabling their transition to state S1.

The Broker or CSP_L iAM in state S1 analyzes the new state of resources. If a problem is detected, the Broker or the CSP_L iAM tries to plan and resolve this problem. Else, it updates its repository (Knowledge base) with changes, notifies the CSU for resource allocation or releasing, and goes to the initial state S0. Note that, if some CSU requests exceed the corresponding throughput threshold, the CSP iAM can drop or delay these requests or allocate new paid resources to the CSU based on the established iSLA. Then, it sends an exceeding of throughput notification to the Broker or the CSP_L iAM that allocates new paid resources to the CSU if needed.

F. *Cost Calculation*

This section introduces a general methodology to calculate costs (1) similar to the way Amazon is charging its clients.

$$Cost_{total} = Cost_{VM} + Cost_{BW} \tag{1}$$

where, $Cost_{total}$ is the total cost of different CSU resource consumption, $Cost_{VM}$ is the total cost of VMs resources (2) and $Cost_{BW}$ is the total cost of bandwidth resources (3).

$$Cost_{VM} = \sum_{j}\sum_{i}(VMc_{ij} \times t_i) \tag{2}$$

$$Cost_{BW} = \sum_{k}(BWc_k \times BW_k) \tag{3}$$

where, VMc_{ij} (\$ per hour) is the cost unit of a selected VM type vt_i in a selected CSP_j (DC), and t_i is the number of vt_i consumption hours. BWc_k (\$ per GB) is the cost unit of the traffic traversing a selected CSP_k (DC and/or BoD) with a QoS level and BW_k (GB) is the CSU traffic traversing the selected CSP_k.

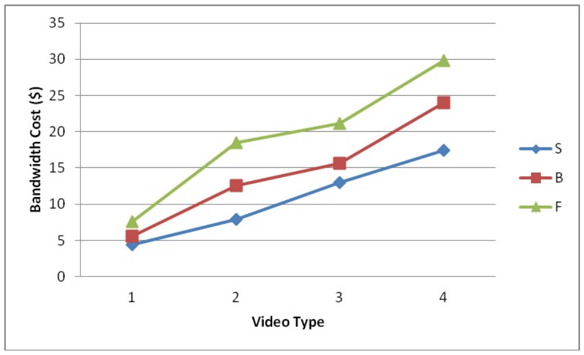

Figure 8. Global bandwidth cost comparison.

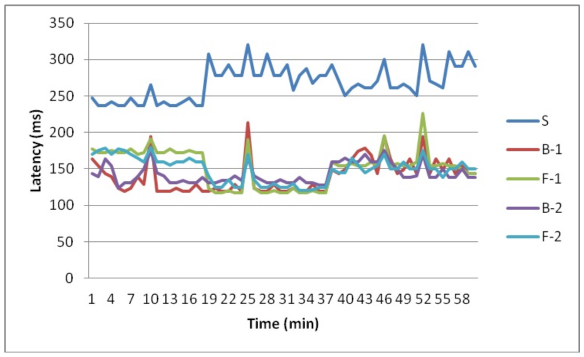

Figure 9. Global end-to-end latency comparison.

V. Usage Cases and Evaluation

To take full advantage of our proposed autonomic cloud networking framework, we present in this section two usage cases. The first one is for a large-scale cloud videoconferencing application which is one of the most demanding multimedia applications in terms of bandwidth, end-to-end delay and jitter QoS parameters. The second usage case is for intensive computing application which is based on requests for the execution of computationally intensive tasks. Therefore, we use our autonomic framework to self-establish SLAs and self-manage these applications while minimizing the total application cost without violating end-to-end QoS parameters.

We test corresponding usage cases as a proof of concept and evaluate performances by conducting a set of simulations using the CloudSim toolkit [18]. We extend CloudSim to support three new entities. The first one is a CSU entity and the second is a Cloud Broker entity, instead of DatacenterBroker entity. In addition, we propose a CSP (BoD) entity that interconnects the Broker, CSPs (DC) and CSUs using BRITE topology [18] for modeling link bandwidth and latencies. In addition, we use the Sensor class for monitoring resources and specifying thresholds and the SimEven class for specifying notifications. Therefore, the control loop algorithms at each entity constitute the iAM, nAM, DnAM and hAM.

The simulated model is composed of one Broker, four CSPs (BoD) and four CSPs (DC) containing each one 10 hosts. A host has quad-core processors (4×1,2GHz) and 16GB of RAM. All entities are initiated at the beginning of the simulation. For a Gold service level for example, a CSP (DC) can have the following characteristics in our simulation model: 750 MHz VM CPU capacity, 99.999% IaaS availability, NaaS QoS parameters: {Latency 7 ms, Jitter 1 ms, Packet Loss Ratio 2.5×10^{-3}, Bandwidth 10Mb/s, NaaS availability 99.999%}, Bandwidth cost 0.05 (\$ per GB) and CPU cost 0.3 (\$ per hour). In addition, a CSP (BoD) can have the following characteristics: NaaS QoS parameters: {Latency 12 ms, Jitter 2 ms, Packet Loss Ratio 2.5×10^{-3}, Bandwidth 10Mb/s, NaaS availability 99.999%} and Bandwidth cost 0.3 (\$ per GB). Other CSPs (DC/BoD) have different values for each service level (Platinum, Gold, Silver, Bronze).

In each usage case, we evaluate three simulation scenarios. The first one corresponds to a static selection of resources without SLA self-establishment and QoS guarantee. The second scenario is a broker and a federation based architecture with SLA self-establishment and QoS guarantee but without resource self-management, i.e., in case of violation the system will not react. The third scenario is the same as the second one but with resource self-management. In each scenario, we have multiple CSU sites connected to different CSPs (BoD) that use IaaS with NaaS services in broker and federation scenarios. In addition, the violation is simulated as a traffic sent by another party that affects QoS parameters of CSU resources.

A. *Usage Case 1: Cloud videoconferencing scenario*

We simulate 4 video types with one hour length, a size of 1, 2, 3 and 4 GB respectively and a bandwidth of 2.2Mb/s, 4.5Mb/s, 6.8Mb/s and 9.1Mb/s, respectively. The CSU specifies the latency less than 180 ms in Broker and Federation architectures. We calculate the global bandwidth cost of each video type while comparing the first and the third scenarios (Figure 8). In addition, we calculate the global network latency of the first video type, with a sampling interval period of one minute, to compare the three scenarios (Figure 9).

As shown in Figure 8, the bandwidth cost increase when the video bandwidth increases. Moreover, in a static selection (S),

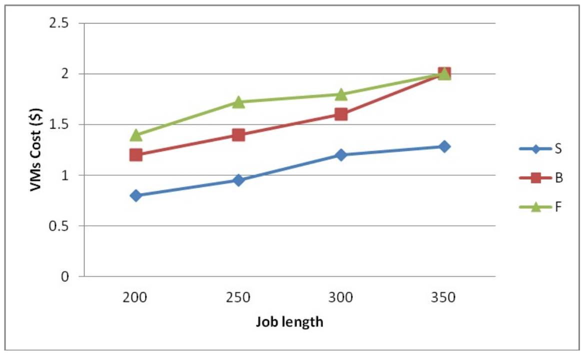

Figure 10. Global VMs cost comparison.

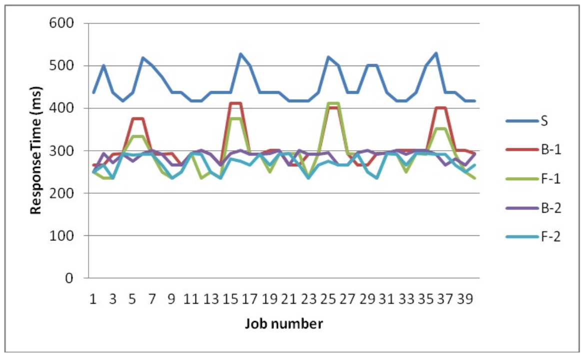

Figure 11. Global response time comparison.

the bandwidth cost is less than the Broker (B) and Federation (F) costs due to the NaaS QoS guarantee. Furthermore, the bandwidth cost in the Broker scenario is less than the Federation scenario due to the selection of resources in the CSP_L firstly. Therefore, CSP_L resources are not usually the best and so the Broker architecture is the most economical while ensuring QoS requirements. In addition, as shown in Figure 9, the results reveal a good streaming latency achieved by our Broker (B-1/2) and Federation (F-1/2) proposal, as compared to a static (S) selection. However, in case of violation, the Broker or the CSP_L (B/F-2) can react and avoid the violation as compared to results without self management (B/F-1).

B. Usage Case 2: Cloud Intensive Computing

We simulate different job lengths (200, 250, 300 and 350 instructions) that are executed by different VMs. For each job length, the CSU sends 10 jobs to four VMs. The CSU specifies a response time less than 300 ms. We calculate the global VMs cost of each job length during an hour while comparing the first and the third scenarios (Figure 10) and the global average response time to compare the three scenarios (Figure 11).

As shown in Figure 10, the VMs cost increase when the job length increases. Moreover, the VMs cost in static selection is less than the VMs cost in Broker and Federation scenarios due to the IaaS QoS guarantee. Furthermore, the VMs cost in the Broker scenario are less important than the Federation scenario according to the same reasons mentioned in usage case 1. Thus, the Broker architecture is the most economical. In addition, as shown in Figure 11, the response time is well controlled (B/F-1/2) as compared to a static selection (S) thanks to IaaS QoS guarantee. However, in case of violation, the Broker or the CSP_L (B/F-2) can react and avoid the violation as compared to results (ex. Job 5/16/37) without self management (B/F-1).

VI. Conclusion and Future Work

In this paper, we presented a framework for self-establishing an SLA and self-managing CSU resources within cloud networking environment. At first, we have proposed an autonomic cloud networking architecture for Broker and Federation scenarios. In addition, we have presented the description, interactions and the lifecycle of our autonomic cloud managers. Finally, we have evaluated our proposed framework for cloud videoconferencing and intensive computing applications and we have obtained good performance results. We observed that the Broker architecture is the most economical.

As a future work, we aim to define the penalty calculation in the SLA. In addition, security parameters will be included in our proposed iSLA to provide our autonomic cloud networking framework with self-protection and study their impact on QoS parameters.

References

[1] P. Murray, A. Sefidcon, R. Steinert, V. Fusenig, and J. Carapinha, "Cloud Networking: An Infrastructure Service Architecture for the Wide Area," in Proceedings of the 21st Annual Conference on Future Network & Mobile Summit, 2012, pp. 1–8.

[2] ITU-T, "Requirements and framework architecture of cloud infrastructure," in Focus Group on Cloud Computing, TR part 3, 2012, pp. 1–59.

[3] "SAIL Project," URL: http://www.sail-project.eu/ [retrieved: 01, 2015].

[4] "FoSII," URL: http://www.infosys.tuwien.ac.at/linksites/FOSII/index.html [retrieved: 01, 2015].

[5] "Contrail-project," URL: http://contrail-project.eu/ [retrieved: 01, 2015].

[6] "CPIP," URL: http://standards.ieee.org/develop/project/2301.html [retrieved: 01, 2015].

[7] "SIIF," URL: http://standards.ieee.org/develop/project/2302.html [retrieved: 01, 2015].

[8] "OCCI," URL: http://occi-wg.org/ [retrieved: 01, 2015].

[9] "GICTF," URL: http://www.gictf.jp/index_e.html [retrieved: 01, 2015].

[10] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: a cloud networking platform for enterprise applications," in Proceedings of the 2nd ACM Symposium on Cloud Computing, 2011, pp. 1–13.

[11] Z. Zibin, Z. Xinmiao, W. an Yilei, and M. Lyu, "QoS Ranking Prediction for Cloud Services," IEEE Transaction on Parallel and Distributed Systems, vol. 24, 2013, pp. 34–47.

[12] R. Karim, D. Chen, and A. Miri, "An End-to-End QoS Mapping Approach for Cloud Service Selection," in IEEE Ninth World Congress on Services, 2013, pp. 341–348.

[13] M. Smit, B. Pawluk, P. ad Simmons, and M. Litoiu, "A Web Service for Cloud Metadata," in IEEE Eighth World Congress on Services, 2012, pp. 361–368.

[14] Y. Kouki and T. Ledoux, "CSLA: a Language for improving Cloud SLA Management," in Proceedings of the International Conference on Cloud Computing and Services Science, 2012, pp. 586–591.

[15] P. Patel, A. Ranabahu, and A. Sheth, "Service Level Agreement in Cloud Computing," in Proceedings of the OOPSLA Cloud Computing workshop, 2009, pp. 1–10.

[16] H. Ludwig, A. Keller, A. Dan, R. King, and R. Franck, "Web service level agreement (WSLA) language specification," IBM Corporation, 2003, pp. 815–824, version 1.0.

[17] M. Hamze, N. Mbarek, and O. Togni, "Autonomic Brokerage Service for an End-to-End Cloud Networking Service Level Agreement," in IEEE 3rd Symposium on Network Cloud Computing and Applications, NCCA, 2014, pp. 54–61.

[18] R. Calheiros, R. Ranjan, A. Beloglazov, C. Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environment and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, 2011, pp. 23–50.