QoS Monitoring in a Cloud Services Environment: The SRT-15 Approach

Giuseppe Cicotti, Luigi Coppolino, Rosario Cristaldi, Salvatore D'Antonio, and Luigi Romano

Epsilon srl, Naples, Italy

Abstract. The evolution of Cloud Computing environments has resulted in a new impulse to the service oriented computing, with hardware resources, whole applications and entire business processes provided as services in the so called "as a service" paradigm. In such a paradigm the resulting interactions should involve actors (users and providers of services) belonging to different entities and possibly to different companies, hence the success of such a new vision of the IT world is strictly tied to the possibility of guaranteed high quality levels in the provisioning of resources and services. In this paper we present QoSMONaaS (Quality of Service MONitoring as a Service), a QoS monitoring facility built on top of the SRT-15, a Cloud-oriented and CEP-based platform being developed in the context of the homonymous EU funded project. In particular we present the main components of QoSMONaaS and illustrate QoSMONaaS operation and internals with respect to a substantial case study of an Internet of Thing (IoT) application.

Keywords: Quality of Service, Cloud Computing, Complex Event Processing.

1 Introduction

Quality of Service (QoS) monitoring is key for a company's success, since assessing the actual quality of what service users are paying for has become a mission-critical business practice requirement, and it will be even more so in the future. The ever increasing complexity of individual components and interconnection among them has impaired our ability to measure the Key Performance Indicators (KPIs) of the service which is to be monitored. Emerging development paradigms, together with the amazing increase of the scale, have made this task even more challenging in upcoming Future Internet (FI) scenarios, since individual business processes - whose internals are completely unknown - are being integrated to quickly implement and cheaply deploy semantically richer business processes. Evidence is demonstrating that QoS monitoring is a very much needed facility in the current cloud computing scenario, and it will be even more so in the future. In particular, in [1] authors emphasize that, due to the dynamic nature of cloud computing environments, continuous monitoring of Quality of Service (QoS) attributes is needed to enforce Service Level Agreements (SLAs). In [2],

M. Alexander et al. (Eds.): Euro-Par 2011 Workshops, Part I, LNCS 7155, pp. 15–24, 2012.

authors state that "An independent tool for monitoring/validating performance of a heterogeneous set of applications" is one of the capabilities which are most needed in the cloud. The availability of a dependable (i.e. reliable and timely) QoS monitoring facility would make it possible for organizations to understand if any failure and/or performance issue which they experience is caused by their cloud service providers, network infrastructure, or the design of the application itself. This would play a key role in the real take up of cloud computing technology, since, by allowing them to receive the full value of cloud computing services, it would increase the level of trust they would place in the cloud paradigm.

From a technical perspective, cloud computing makes QoS monitoring extremely challenging, for a number of reasons, and in particular:

- Business applications currently run (mostly) on privately owned service infrastructures, whose technology is - to a large extent - known. In a cloud computing context, they run on infrastructures which are: (i) shared, and (ii) virtualized. This also applies to the QoS monitor itself.
- It has to cope with dynamic composition of services, meaning that it must be able to track the evolution of services and it must be able to provide continuous QoS monitoring, i.e., even while changes are taking place.

We claim that QoS monitoring should be made available to all cloud users in a seamless way. To this end, the "as a service" model stands out as the ideal solution. Thus, we decided to implement a QoS monitoring facility, called QoSMONaaS (Quality of Service MONitoring as a Service) which is provided according to this paradigm.

QoSMONaaS is realized as a pilot application into an SRT-15 project, a new cloud-based Platform as a Service (PaaS) on top of which it is possible to build/compose every software as a services (SaaS). The goal of the SRT-15 project (Subscription Racing Technology for 2015) [3] is to build a scalable platform for connecting FI business applications and services. The platform will enable the discovery and integration of dynamic enterprise services on the Internet. SRT-15 will allow for dependable and scalable cloud-based processing of data coming to and from a variety of heterogeneous enterprise services spread across multiple distributed locations. In order to be able to embrace the change in the enterprise information processing landscape, SRT-15 relies on technologies that support rapid change: cloud computing, content-based routing [4] and complex event processing [5]. Privacy issues, concerning to the anonymity of services (i.e without an exchanged event reveal the identity of sender) or content hiding, is also taken into account by SRT-15.

It is worth emphasizing that the approach we propose for a QoS monitoring is innovative with respect to several aspects which are briefly discussed in the following:

- most existing products only focus on resources monitoring (or on an homogeneous environment), while QoSMONaaS focuses on the performance delivered at the Business Process level into an heterogeneous environment such as Cloud Computing, which is what cloud users really care for.

- current solutions (including big player, e.g. IBM [6]) gather data at predefined interval time and analyze them in batch or streaming way merely reporting violation. On the contrary, QoSMONaaS efficiently can collect data in an asynchronous way processing it just when predetermined condition are met (e.g. when some KPI values go beyond their thresholds it forwards queries to get more information about the service) attempting to prevent violation condition via statistical and/or logical inference.
- the assumption of Trusted Third Party, usually accepted on QoS monitor, in our solution is released: the QoSMONaaS is a "peer among peers", which will be trusted by design thanks to the privacy functionality provided by SRT-15.
- by using Complex Event Processing (CEP) capabilities provided by the underlying SRT-15 platform, in place of getting single data, QoSMONaaS can extract sequences of events by means of queries built on purpose to discover particular patters so that to prevent and/or detect violation condition.

This work shows our novel approach to QoS monitoring, which has been conceived to perfectly suit a cloud computing context. The proposed methodology is based on a model description of the service to be monitored and a formal specification of the related SLAs. Based on this information QoSMONaaS is able to generate specific queries to be performed by a CEP in order to get events or sequences of events (patterns) allowing to recognize violations both ongoing or already happended.

The rest of the paper is organized as follows. Section 2 presents the novel approach that we propose for monitoring QoS in a cloud environment. The key idea is to provide QoS monitoring according to the "as a service" model. As such, the approach is implemented in a pilot application named QoSMONaaS, i.e. "QoS MONitoring as a Service". Section 3 illustrates how QoSMONaaS is implemented by using the SRT-15 platform key features, mainly Complex Event Processing (CEP). Section 4 presents how QoSMONaaS is being validated with respect to Smart Meters, a substantial case study of an Internet of Things (IoT) application developed by SAP for implementing remote monitoring of power consumption in a Smart Grid environment. Section 5 concludes the paper with lessons learned and directions of future research.

2 QoSMONaaS: Quality of Service MONitoring as a Service

QoSMONaaS implements a dependable QoS monitoring facility, which is made available to all applications running on top of the SRT-15 platform. A Conceptual view of the interaction model among the individual parties involved in the monitoring process is given in fig. 1.

SRT-15 Requirements: The underlying platform will have to guarantee:

- that internal SRT-15 information (log data), which are of interest for QoS monitoring, be made available to the QoSMONaaS application. This is made possible via the Extended Interface.
- that effective processing capabilities - i.e. relational algebra operators (or equivalent ones) - be made available, which allow QoSMONaaS to extract and process QoS-related data streams in real time. This is provided by the CEP facilities of the SRT-15 platform via the Basic Interface.
- that the real identity of parties being monitored is not revealed to the monitoring application. This is achieved by having SRT-15 facilities be mediated by the Anonymization System.
- that all facilities be reliable and timely.

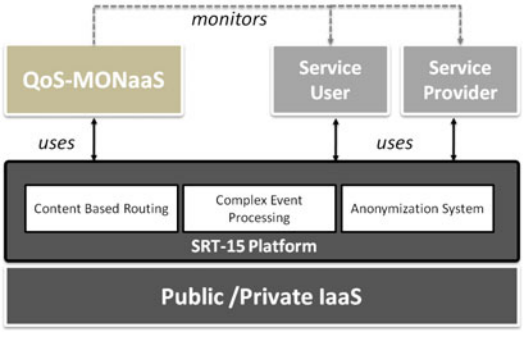

Fig. 1. SRT-15 and QoSMONaaS interaction overview

For more details about CEP facility of the SRT-15 platform, please refer to section 3, and to [3] and [7]. We explicitly emphasize that by introducing an Anonymization System (see fig. 1) in the loop, we avoid that the real identity of monitored parties be revealed to the monitoring application. This is essential for guaranteeing the aforementioned "trusted by design" and "peer among peers" characteristics of the QoSMONaaS application. The basic idea is that since the monitor ignores the real identities of parties, it cannot cheat. In the current implementation, we use a simple scheme, which is in all respects similar to those used in many social network applications [8] (basically, we rely on the simple technique of anonymizing graphs, by which we replace the identifying information of the parties with random ids). We are also exploring more sophisticated schemes, which will be the subject of a different paper.

3 QoSMONaaS Internals

QoSMONaaS main components and their operations are illustrated in fig. 2 and briefly described (additional details are provided in [9]).

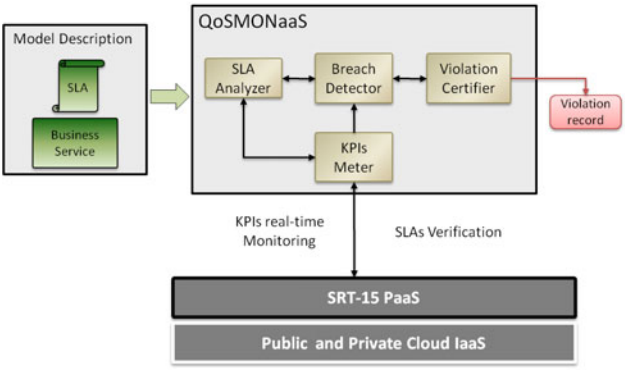

Fig. 2. QoSMONaaS Architecture

Model Description: In order for QoSMONaaS to be able to monitor the actual QoS delivered by the service provider, it is necessary to supply:

- a formal description of the specific business process (entities, relations and KPIs)
- a formal description of the SLA to be garanteed, that is a set of constraints that must be respected.

From a QoSMONaaS perspective, it is sufficient a service-centered model description that specify "what" rather than "how" the business process should do. To this scope, we are interested in a declarative specification of the main services involved in (used by) business process, relations among them, defined in term of exchanged events, and KPIs to keep under control.

The **SLA Analyzer** deals with two macro-functionalities: on the one hand read and processes (parsing) SLAs and model description provided as input, generating queries to pass through the KPI Meter. Furthermore, the SLA Analyzer collects all the information received by KPI Meter so as *analyze* them and *infer* new knowledge to be turned into fresh rules (CPE queries) to give again to the KPI Meter. Into QoSMONaaS application this component plays the role of decision maker allowing to have some advantages:

- adapting the monitoring (variable frequency of submitting queries) so as to minimize the downstream from SRT-15
- selecting just events and KPIs of interest and/or necessary to the analysis/monitoring process.
- efficiently submitting queries to CEP: just in case KPIs values raise a warning, the Engine triggers the Breach Detector sending it the SLA queries it must verify and the frequency of querying (e.g. check every 10 minutes).

The **KPI Meter** continuously monitors the actual value of the KPIs of interest by means of queries to submit to the CEP SRT-15's component. All the submitted queries can be distinguished on the basis of their result:

- Event Data - useful to take the current data value of interest
- Sequence of events - allows detection of a particular sequence of events (absent, if empty) in a data stream over a specified time window.

The **Breach Detector** combines the outputs of the KPI monitor and conditions from the SLA Analyzer to spot contract violations. In this case, report violation back to the SLA Analyzer and forward all related information to the Violation Certifier.

The **Violation Certifier** enriches the output of the Breach Detector with a timestamp and a digital signature, so to produce unforgeable evidence usable for forensic purposes (more details in [9]).

4 Case Study

The implementation of QoSMONaaS is being validated with respect to a substantial case study of an Internet of Things (IoT) application developed by SAP. The application, which is called Smart Meters, implements remote monitoring of power consumption in a Smart Grid environment.

Smart Meters use the SRT-15 platform for distribution and real-time processing of measurements related to energy consumption and production. This application is the ideal candidate to showcase the applicability of the proposed QoS monitoring approach to a realistic Future Internet (FI) scenario. The figure 3 shows an overview of the actors involved in this real Smart Meter application.

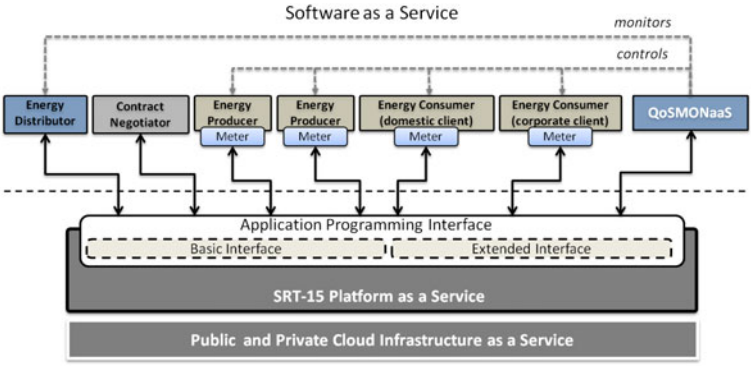

Fig. 3. Case Study: Smart Meter Application

The Energy Distributor (ED) sells energy power to Energy Consumers (ECs), based on a set of pre-defined SLAs. To do so, it buys energy from Energy Producers (EPs), also based on a set of others pre-defined SLAs. EPs and ECs are each one equipped with a Smart Meter which make measurements in (soft) real-time readily made available through the SRT-15 platform to the ED. In our case study EPs and ECs are directly attached to the Cloud through SRT-15. The ED

uses SRT-15 as well, gathering data on power consumption, and processes it in an attempt to predict the energy demand for the next time frame. The ED's objective is to efficiently manage the unstable production of renewable energy and balance supply and demand. Based on the result of the prediction process, the ED knows that it needs to either contact additional EPs and add them to its pool or it may spare some of it (in case of surplus production). Contracts between parties are handled by the Contract Negotiator. The above mentioned actors (namely: Energy Distributor, Energy Consumers, Energy Producer, and Contract Negotiator) altogether form the Smart Meter application.

In our case study we use labels `EP1`, `EP2`, `EC1`, `EC2` and `ED` for respectively referring to Energy Producers, Energy Consumers and Energy Distributor services all made available (published) on the SRT-15 platform. From the point of view of the `ED` service, that we need to monitor, it is linked to (uses) both Producer and Consumer services by means of events of measurement, called `SmartMeterMeasureEvent`, coming from their Smart Meter devices. However, there could be many other type of relations like, for example, `DeviceStateEvent` event, that conveys information details on consumption of each single device, appliance and/or complex system (e.g. machinery, networks, plant, etc.) installed at ECs so to remotely control any individual electrical device. An example of model description for an ED business process could be as follows:

```
1   Service Model:
2       complexType Services  = {Producers, Consumers};
3       complexType Producers = {EP1, EP2};
4       complexType Consumers = {EC1, EC2};
5       ED: Producers.SmartMeterMeasureEvent
6       ED: EC1.SmartMeterMeasureEvent
7       ED: EC2.*
8       ED: Services.SmartMeterInfoEvent
9       EC2: ED.RemoteControlEvent
10  KPI definition:
11      <define simple/complex KPI>
12  Constraints:
13      <conditions to be satisfied>
```

Rows 2 to 5 define the structure of the process, specifically they show the dependence of `ED` (row 5) from Producers, that is `EP1` and `EP2`; rows 6 to 9 define the interest of some entities (`ED` and `EC2`) in specific events. The '`*`' symbol in row 7 represent the interest of `ED` in all the events produced by `EC2`.

In the `Constraints` section we can define some kind of special properties that must be satisfied over a whole (possibly endless) runtime execution of the service[1], whereas within `KPI definition` section all the KPIs concerning the business service are defined. In the case study under consideration we have two

[1] Typical properties considered on dynamic systems are those of safety, liveness, fairness, etc.

composite KPIs, `Consumption` and `Production`, respectively indicating the energy consumed by all ECs and the energy produced by all EPs (i.e. they are aggregated data). For the sake of simplicity we consider the real energy power put into distribution network by ED equal to the energy bought from all EPs; thus we can constrain the model with a safety property that `Consumption < Production` is always true in every state of the service model. Finally, we can identify the KPIs of interest for the process as follows:

```
1   KPI definition:
2    compositeKPI Consumption = \
        forall C in Consumers \
        select sum(measure) from C.SmartMeterMeasureEvent;
3    compositeKPI Production  = \
        forall P in Producers \
        select sum(measure) from P.SmartMeterMeasureEvent;
4    simpleKPI : EC2_PCnet = \
        select state from EC2.DeviceStateEvent;
5   Constraints:
6     Consumption < Production
```

Row 2 provides the relationship between the derived KPI `Consumption` and the basic event generated by each consumer. At the same way, the row 3 specifies a similar relation between the KPI `Production` and basic event produced by every producer.

The service level to be monitored is specified by a Service Level Agreement (SLA) between service provider and service user. An SLA is defined in WS-Agreement, a formal xml-based definition language written with the goal to provide a great flexibility for future extensions and customizations. In particular, we can decide the appropriate language useful to specify the quality conditions to be verified.

The choice on such a language should consider its ability of capturing:

- the KPI value conveyed by a single event
- a specified pattern of events into an ordered sequence of events.

Our choice was to use an Interval Temporal Logic (ITL), a language that combines first order and temporal operators. For the sake of brevity we will not introduce ITL, refer to [10] for a detail description of its syntax and semantics.

For our SLA some simple clauses could be:

1. between `ED` and each `EP`: the energy supplied by an `EP`, we say it p, is never less than a predefined threshold, t, for no longer than an I interval time.
2. between `ED` and `EC1`: the "availability" of energy power must be of 95% along 24h (about 1 hour per day of power outage)
3. between `ED` and `EC2`:
   - SLO 1: the "availability" of energy power must be of 99% along 24h during weekday

- SLO 2: the "availability" of energy power must be of 95% along 24h during weekend and holydays

Once the SLA has been provided with an ITL-like expression, the SLA Analyzer is able to interpret and translate it into queries executable by a CEP. For instance, clause 1) can be formalized through the following ITL-like expressions:

```
EP1.SmartMeasureEvent.measure > t over [15 min]
```

where I is sets to 15 minutes.

The SLA Analyzer parses this expression and, then, extracts the internal KPI, `measure` in this case, generating the two following CEP queries:

```
- select measure from EP1.SmartMeterMeasureEvent
  where measure > t
- select sequence(measure) as Seq
  from EP1.SmartMeterMeasureEvent
  where time.interval(15min) and Seq.forall(measure > t)
```

On the one hand, the first query is used by QoSMONaaS for continuously monitoring a possible "symptom" of the SLA violation and is forwarded to the KPI Meter. On the other hand, based on anomalous values and statistics of history data, the SLA Analyzer can decide to submit the second query to the CEP for detecting an SLA violation (see fig. 4). In this way the SLA Analyzer optimizes the number of requests submitted to the CEP. Clause 2) represents a more

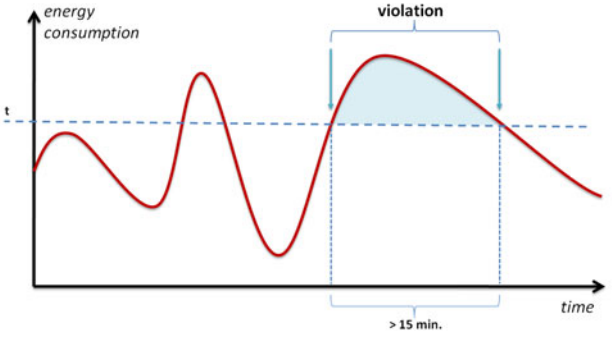

Fig. 4. QoSMONaaS Violation Detection

complex case as it requires the correlation of two events; in particular, we need to verify that the "absorbed power" field of each `SmartMeterMeasureEvent` has a value equal to the one of the "supplied power" in the `SmartMeterInfoEvent`. As for clause 3), breaches can be detected looking for periods of 1, 2 h out of 24 h without data events intended. Such an anomalous behavior can be the result of a power outage elapsed over the 5% of a day.

5 Conclusions and Directions of Future Work

QoS monitoring is an essential aspect to take under control SLAs contracted among business partners. Into a Cloud Computing context, which is based on the "as a service" paradigm, a continuous control of QoS attributes is a primary issue that the current state of the art products still have difficulties to deal with.

In this paper we have discussed the implementation of a new monitoring facility within the context of the SRT-15 project. The paper makes three important contributions. First, it proposes an innovative approach to QoS monitoring, which perfectly suits the cloud computing context. Second, it shows how such an approach can be efficiently implemented in a cloud computing platform which provides advanced features, specifically Complex Event Processing (CEP). Third, it demonstrates the advantages brought by the developed QoS monitoring facility to a realistic Future Internet (FI) application.

For the future we have been studying the possibility to combine the potential of statistical and logical reasoning to both estimate (in a probabilistic way) and predict violations. Furthermore, we will focus our attention on the performance of a "root cause analysis" in an SRT-15-based Cloud environment, for finding and linking together causes (anomalous states) and effects (violations).

References

1. Pankesh Patel, A.S., Ranabahu, A.: Service level agreement in cloud computing. In: UKPEW (2009)
2. Four keys for monitoring cloud services. White Paper from ManageEngine, `http://www.manageengine.com/products/applications_manager/four-keys-for-monitoring-cloud-services.pdf`
3. White Paper (2010), `http://www.srt-15.eu/`
4. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.-M.: The many faces of publish/subscribe. ACM Computing Surveys 35, 114–131 (2003)
5. `http://www.complexevents.com/event-processing/`
6. IBM, Service Level Agreement Monitoring with IBM cognos now (2010)
7. Martin, A., Knauth, T., de Brum, D.B., Weigert, S., Creutz, S., Brito, A., Fetzer, C.: Low-overhead fault tolerance for high-throughput data processing systems. In: ICDCS (2011)
8. Ying, X., Pan, K., Wu, X., Guo, L.: Comparisons of randomization and k-degree anonymization schemes for privacy preserving social network publishing. In: SNA-KDD (2009)
9. Romano, L., Mari, D.D., Jerzak, Z., Fetzer, C.: A novel approach to qos monitoring in the cloud. In: 1st International Conference on Data Compression, Communication and Processing. IEEE Computer Society Press (June 2011)
10. Cau, A., Moszkowski, B.: Interval temporal logic (March 2010)