Data Centres Optimization for Energy-Efficient and EnvironmentalLy Friendly INternet

Funding scheme: Specific Targeted Research Projects – STREP

Co-funded by the European Commission within the Seventh Framework Programme

Project no. 609140

Strategic objective: FP7-SMARTCITIES-2013 (ICT-2013.6.2)

Start date of project: October 1st, 2013 (36 months duration)

Deliverable D4.3

SLA Renegotiation Controller component (Implementation)

Due date: 31/12/2015

Submission date: 28/01/2015

Deliverable leader: SIEMENS

Author list: Alina Nistor (SIEMENS), Septimiu Nechifor (SIEMENS), Stuart Clayman (UCL)

Dissemination Level

☒	PU:	Public
☐	PP:	Restricted to other programme participants (including the Commission Services)
☐	RE:	Restricted to a group specified by the consortium (including the Commission Services)
☐	CO:	Confidential, only for members of the consortium (including the Commission Services)

List of Contributors

Participant	Contributor
SIEMENS	Alina Nistor, Septimiu Nechifor, Constantin Aldea
UCL	Stuart Clayman
IRT	Domenico Gallico, Matteo Biancani
PSNC	Michal Balcerkiewicz

Amendment History

Version	Date	Partners	Description/Comments
0.1	04/12/2015	SIEMENS	Initial draft
0.2	11/01/2016	SIEMENS	Small changes and contributions
0.3	12/01/2016	IRT, PSNC,	Pre-final review and interactions section input
0.4	15/01/2016	SIEMENS	Addressed the IRT comments, ready for final review
0.5	18/01/2016	IRT	Final review
1.0	28/01/2016	IRT	Final version submitted to EC

Table of Contents

Figures Summary

Tables Summary

Abbreviations

AMQP	Advanced Message Queuing Protocol
CEP	Complex Event Processing
CPU	Central processing unit
CRUD	Create, Read, Update and Delete
DB	Database
DC	Data Centre
DCF	Discount Cash Flow
DCO	Data Centre Optimization
DNS	Domain Name Server
DVFS	Dynamic Voltage and Frequency Scaling
eCOP	Energy Consumption Optimization Platform
EP	Energy Provider
EPL	Event Processing Language
FTP	File Transfer Protocol
HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
ICT	Information and Communication Technology
HVAC	Heating, Ventilation and Air Conditioning
IOPS	Input/Output Operations per Second
IMAP	Internet Message Access Protocol,
IPR	Intellectual Properties Rights
KPI	Key Performance Indicator

MTTR	Mean Time to Repair
PM	Person Months
POP	Post Office Protocol
QoS	Quality of Service
RAM	Random Access Memory
RTT	Round Trip Time
SLA	Service Level Agreement
SLARC	Service Level Agreement Renegotiation Controller
SLO	Service Level Objective
SMTP	Simple Mail Transfer Protocol
ToC	Table of Content
VM	Virtual Machine
WP	Work package
XML	Extensible Markup Language

Executive Summary

This document represents the deliverable D4.3 “SLA Renegotiation Controller” of Work Package 4 “Synergetic Data Centers for energy efficiency” within the FP7-SMARTCITIES-2013 DOLFIN project.

This deliverable describes the SLA Renegotiation Controller detailed functional design, the implementation and the external interfaces of this module. The SLA Renegotiation Controller interacts with the Monitoring and Energy Efficiency Policy Maker and Actuators using a message exchange protocol (AMQP). Based on agreed messaging patterns, this component reads/receives a number of key monitored measurements parameters, computes and validates the state of energy impact (formally defined in SLA policies) and of energy efficiency conditions, and finally, generates the instructions to be sent to the Policy Maker & Actuators via the same messaging facility.

Another objective of this document is to describe the SLA renegotiation design, which refers to the process of automatically negotiating and agreeing with DC customers in balancing their SLA performance parameters, in exchange of decreased service’s prices. This performance degradation can also be exploited by the DOLFIN ecosystem toward energy efficient DC operation and seamless load relocation across synergetic DCs. Thus, the SLA renegotiation process is of paramount importance for the DOLFIN ecosystem, facilitating the DOLFIN operation at a synergetic DC level.

1. Introduction

The main objective of the SLA Renegotiation Controller is to use the technical specifications and guarantees in the SLA concept to steer the DC operations towards environmentally compatible behavior.

As a starting point for the detailed analysis we take into account that a Service Level Agreement (SLA) between DC and End User already exists. This agreement will offer an extended scope of eco-optimization to the service provider by relaxing traditional performance parameters, introducing novel energy performance parameters as classifying elements, and the parameters that determine the level of collaboration and offering incentives to customers in the form of dynamic pricing schemes. The pricing is tied to the level of collaboration, to the achieved system-wide environmental benefits, and to the achieved reduction of energy consumption in the DC.

This definition has a set of implications for the design analysis in this context, so we need to:

- Define the nature of the trade-off between performance degradation and energy savings as a basis for the compensation.
- Define energy performance parameters that are relevant for the EP (Energy Provider) motivated version of new green criteria to evaluate the quality/level of service.
- Suggest a dynamic pricing scheme as an incentive for the DC customers.

For which concerns the cross-DC optimization, the contracts signed between DCs are similar to the traditional SLA. The QoS specified for workload outsourcing depends on:

- the categories and range of services provided,
- the energy sources available at each DC's disposal,
- the QoS agreed upon with the counterparts,
- the amount of services provided for by contract.

Even if SLA requirements are similar, big customers are assigned with a higher priority, thus all the workload related to these customer will be treated accordingly. This prioritization mechanism introduces additional constraints in the jobs scheduler that needs to be taken into account by the DCF (Discount Cash Flow) when computing all the possible alternative solutions.

The SLA Renegotiation Controller will also comply with constraints derived from the **Green criteria.** The Green criteria represent an agreement between DCs and customers regarding the general rules

for the relocation of services in case of specific events (e.g. service crashes, hardware malfunctioning etc.). The criteria for the relocation are therefore an explicit convention and not an action dictated by the analytic result obtained through the processing of the Data Center services logs.

The specific events in DCs that can require relocation of services are defined on technical view as SLA statements. An example of a technical SLA statement is:

"Data needs to be constantly aligned between the Server Farms of the infrastructure provided by DC to the customer."

As this contractual requirement has to be translated at the architectural level, the statement can become:

"Data replication needs to be activated, with an active/passive configuration, without new servers or other components. The monitoring solution is configured to automate data movement failover and possibly send an alarm message to the central monitoring systems."

This gets further translated into a set of capabilities for the DC environment taking into account that the offered SLAs depends on the realized architecture:

- DC supports storage replication,
- DC storage configuration is active-passive,
- Data movement is automated,
- In case of failure of data movement, an alarm is sent to the central monitoring system.

Examples of constraints derived from Green SLA with end users include typically QoS constraints, such as:

- ***Computation location constraints:*** "*VM @VMx should run in DC @DCy located in region Y*".
- ***Computation amounts:*** "*The maximum number of instances of VMs from customer Y that are allowed to run on DC Z is X*".
- ***Time constraints:*** computation is restricted in terms of availability, performance or service management:
  - *Availability:* "*VM should be available within working days in region Y: 08:00 – 18:00 (of time zone UTC+1)*"
  - *Performance constraints:*
    - "*VM should be available (started) within xx minutes from any request to use it*"
    - "*VM machine should not be too loaded, i.e. its minimum Virtual CPU speed needs to be above the @threshold value.*"
  - *Service Management constraints (QoS):* "*Incidents reported about VM should be solved within X minutes from submitting the request to solve the incident*"
- ***Greenness constraints:*** "Green tasks" are tasks (with lower priority) to be run outside peak load periods.

The following sections of the document present the interactions, the design and implementation details of the SLA Renegotiation Controller.

2. SLA Renegotiation Controller

2.1. Module detail description

The SLA Renegotiation refers to the process of automatically negotiating and agreeing with DC customers in balancing their SLA performance parameters, in exchange of decreased service prices. This performance degradation can also be exploited by the DOLFIN ecosystem toward energy efficient DC operation and seamless load relocation across synergetic DCs.

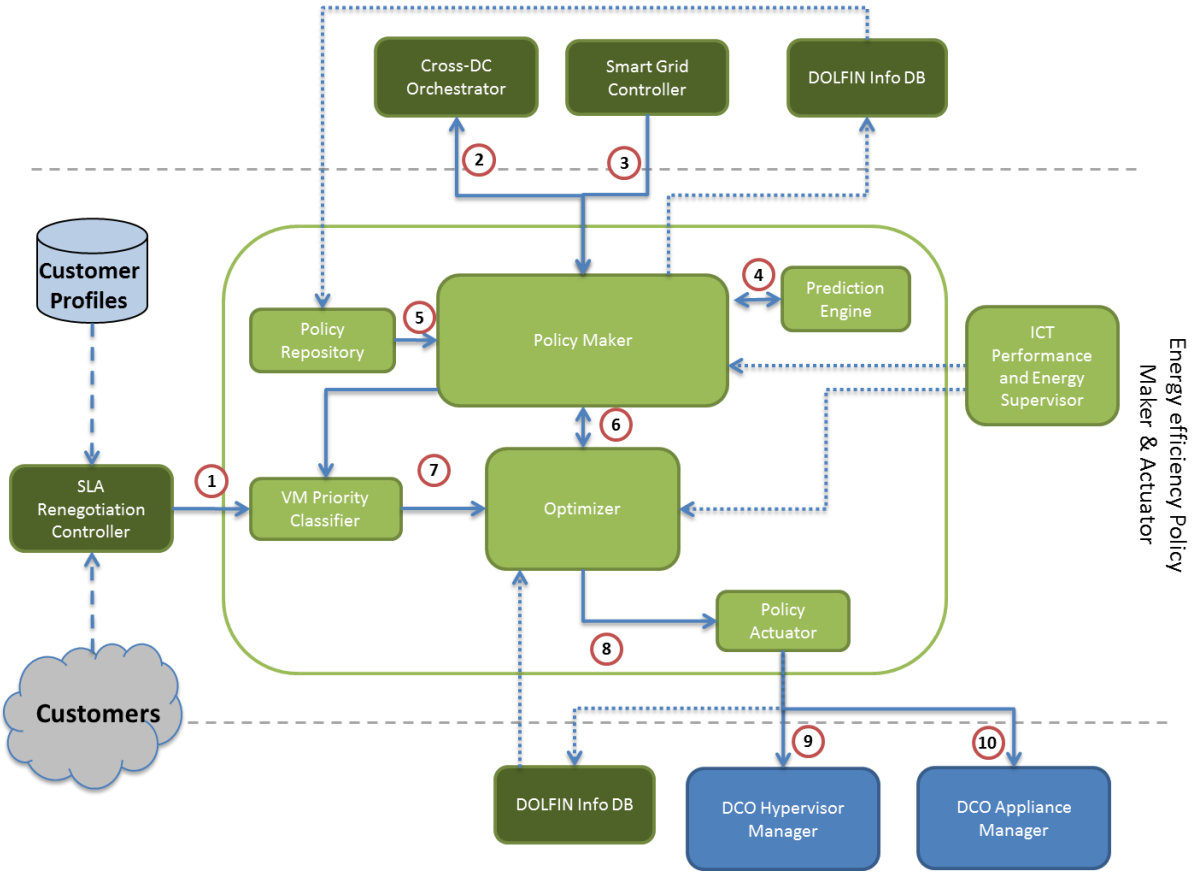

Figure 2-1: DOLFIN platform architecture

In order to capitalize on the merits offered by the SLA Renegotiation into the DOLFIN architecture the SLA Renegotiation Controller was designed as a component dedicated to:

- monitoring of the SLA parameters,
- enforcement of the SLAs,
- automatically renegotiate the SLAs, based on the predefined customer profiles.

The SLA Renegotiation Controller is responsible for supervising all stages of a SLA lifecycle, as these emerge in the context of a cloud market and described in numerous research works in the field, e.g. [1]. In particular, the involved stages are:

- the SLA Template Definition
- the SLA Instantiation and Management and
- the SLA Enforcement.

For proper monitoring and enforcing the SLA terms, the expected service guarantees are defined in the form of a SLA template. The defined SLA template links different levels of SLA guarantees (i.e. different combinations of SLA parameters) to different combinations of virtual resources [1]. Thus, prior to the instantiation of each VM by a DC customer, the SLA of the customer is instantiated, a format of virtual resources is reserved, while consenting to a certain level of service.

2.1.1. Customer profile

DC and cloud infrastructure users exhibit a wide variety of profiles and requirements as they originate from different sectors such as industry, academia, education, research and are widely geographically and culturally distributed. In the application logic adopted by the DOLFIN project, the customer profiles are an important constraint in the actuation of energy policies. One of the main objectives is that the actions taken by DOLFIN should aim to maximize efficiency while still observing the SLA negotiated with the customers. Thus before analysing the SLA issues, we will shortly analyse the issue of User Profiles, as they are closely interrelated. The following service characteristics and their specific prioritization for each customer constitute a solid base to build customer profiles relevant to DOLFIN innovation.

- **Cost:** A typical requirement for service users is to maintain their operation cost as low as possible. In several cases, when the rest of the characteristics are less important, cost can be a dominant factor in a customer's profile.
- **Performance:** In performance critical applications like super-computing or low-latency networking applications, performance in terms of throughput (operations/sec, requests served/sec) or latency (time to respond) can be very critical so that users request the highest possible sustained performance.
- **QoS:** Quality of service is of ultimate importance in several business scenarios where the provided service needs to be as stable as possible, minimizing threshold violations regardless external traffic or load.
- **Security:** Security can be of utmost importance for classes of users and applications that need to have strict guarantees for business critical or personal information.

- **Environmental friendliness:** In some cases, users tend to be quite sensitive about environmental matters, especially when they are aware of the energy and CO_2 footprint of modern DCs.

These service characteristics directly reflect the agreement reported in the SLA and define a set of measures which identify (first of all) the stability of the offered services. The SLA definition should focus on some major characteristics:

- What is the offer handed out by DC?
- How the DC will manage services?
- What is the mapping between SLA and measures?
- What happens if there is a violation on the SLA?

The first two points are directly linked to the technical assets of the DC, since they affect the service modelling (e.g. the provision of a VM) and how these services will be deployed in the infrastructure (e.g. the initial snapshot of a VM allocation). In case of changes in the DC infrastructure or in the VMs allocation, the SLAs must be taken into account. The third and fourth points establish which steps are needed for testing, whether or not the SLAs are compliant with what customer agreement requires and the measurements applied when an SLA violation occurs. The SLA is generally expressed in "textual form", for example:

"The SERVICE will be available 95.0% for 24h of operation 7 days a week. Any individual outage in excess of 15 min or sum of outages exceeding 1h per month will constitute a violation"

This textual form is mapped to a model for measuring the system capacity and identify when a violation occurs. The above example might be expressed as:

a) ***Service Availability as a Percentage of week hours:*** The measurement must be performed on the overall service availability. The maximum threshold is the maximum outage per incident and/or total sum of outage per month. Some typical KPIs to measure the SLA are:
  - ***Maximum capacity Outage***; defines the periods during which there is an unscheduled disruption to the Capacity, in term also of performance information.
  - **Mean Time To Repair (MTTR).** The MTTRs defines the time that provider needs to restore the system after a fault. The value may depend on the fault severity level (i.e. Critical, High, Medium, Low).

b) **Performance.** The performance parameters are (1) dependent from the service and (2) strictly correlated to the SLA and to the service cost. The customer profile should report a structured list of performance parameters which will be linked to the appropriate service. Some typical performance KPIs include: a) VM hosting: Average CPU utilization (%), Average memory utilization (%), Requests served (count/sec), Operation (count/sec), b) Storage: Average iops (count/sec), Average latency (ms),c) Networking: Path availability (%), Packet loss percentage (%), Average incoming throughput (bit/sec), Average outgoing throughput (bit/sec), Access average Round Trip Time (RTT, ms)

c) **Environmental friendliness:** This parameter quantifies the "sensitivity" of the user to the DC awareness on energy consumption and CO_2 footprint and reuse of green energy sources. The parameter generally aims to a global reduction of the performance, and can be expressed as "percentage of reduction in the Performance KPIs". Some (not necessary all) of

the performance KPIs parameters might be described in the form of "minimum granted value" conditioned to a maximum time duration and repetitiveness, for example: Minimum average CPU utilization (%), Time duration (hour), Rate (count/month).

The diagram in Figure 2-2 provides a schematic relational database model for the Customer Profiles DB.

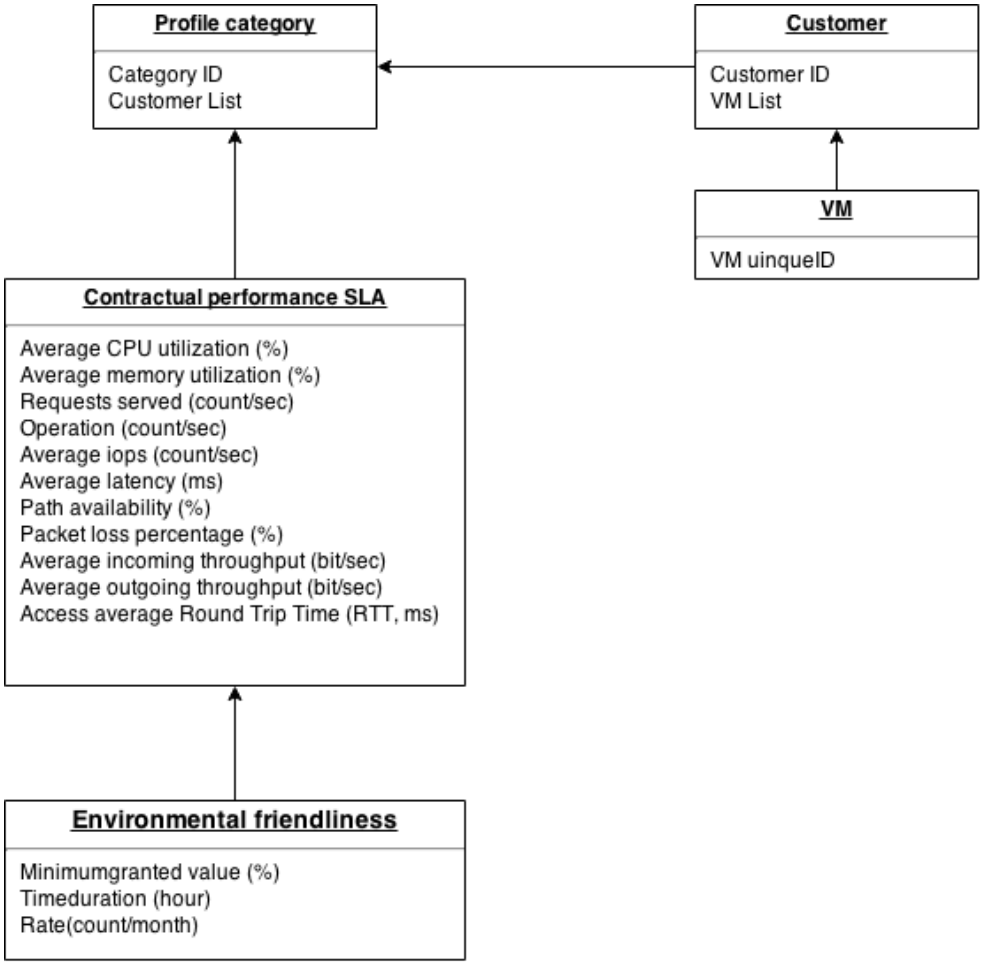

Figure 2-2: Schematic model for the Customer Profiles DB

2.1.2. SLA Templates

During the SLA instantiation process, the customer signs one of the contracts, predefined in the SLA Template Definition stage, detailing the charges for the accepted level of service and the combination of reserved resources. The charges for each level of service depend on one hand on the virtual resources reserved a priori in a stringent way and on the other hand on monolithic SLA parameters which are amenable to renegotiation. Since the parameters pertaining to reservation of virtual resources are not amenable to renegotiation they do not need to be monitored/enforced by the SLA Renegotiation Controller. However, as major price driving factors, the virtual resources reserved by new VMs are explicitly described during the SLA Template Definition. Thus, explicit

discrete values must be assigned in advance to computational resources such as the number of reserved CPU cores, the type of reserved CPU cores, the amount of RAM, the HDD and the HDD throughput.

Moreover, Green VM features which are not amenable to renegotiation but are the major price driving factors, are also defined in the SLA templates, along with the virtual resources reserved by new VMs. In particular, examples of constraints derived from Green VM features that are explicitly stated in the SLA definition, include computation location constraints (e.g. VM should run in Green Room), computation amount constraints (e.g. maximum number of VM instances per customer, per DC), etc.

Once the virtual resources and Green VM features as part of the SLA templates are defined, the actual SLA parameters referring to different levels of SLA guarantees need to be explicitly defined in the context of the SLA templates. In general, SLAs found in literature refer to generic SLAs corresponding to different levels of service (e.g. "Gold Level", "Standard Level", "Cost-optimized Level", etc.), with the SLA parameters being part of the run-time related non-functional properties of a service and presenting the main Quality of Service objectives [2].

The SLA parameters often encountered in practice, defining the different levels of service are the following:

1. VM down time
2. VM CPU speed (Subjected to DVFS)
3. VM RAM utilization
4. VM Network Throughput
5. Cumulative Outgoing Traffic per month (quota based)

To elaborate on the above, *the VM down time* refers to the percentage of time a live VM can be paused, providing an additional degree of freedom toward pausing a running VM, taking a snapshot of it and relocating it to a synergetic DC. The *VM CPU speed* is the threshold value below which the VM can be migrated by the DC operator towards servers with reduced CPU speed and therefore with reduced energy consumption (through DVFS). Thus, a renegotiation of the contractual VM CPU speed would allow the VM migration to a DC of obsolete or lower level resources and provide additional system flexibility. The *VM RAM utilization* and the *VM network throughput* offer additional degrees of freedom, similar to the flexibility provided by the VM CPU speed parameter. In summary, *the cumulative outgoing traffic per month* indicates the maximum acceptable traffic of the VM per month, acting as a safeguard toward system stability, allowing the DC operator to terminate VMs stressing the system, in case these VMs violate the contractual defined SLA parameters.

For each of those parameters a formal SLA policy can be agreed upfront with the customer and formalized in the contract. Usually, for the sake of clarity or legal constraints, the contracts define levels of severity for the cases when a breach is encountered in a single or in a combination of policies monitored, and also possible incentives towards the customers when an optimal behavior as the one detailed in Green SLA is observed. For current component, the effective contractual negotiation and signing aspects are not relevant as further implementations, but only the capture of formalized mechanisms for SLA policies which are accepted as object of automatic renegotiation or re-negotiation validation under DC operator action.

The following functionalities describe the design in more details:

- mapping of SLA parameters in available monitored and published topics (respecting the frequency of updates, e.g.: “VM CPU Speed is monitored every 10s, so a policy defined to asses and react every 2s will use the most recent available measured timestamp in a less than 10s observation window”)
- mapping of SLA parameters based on the quality of available data (having as an option the ability to calculate an inferred value from historic values, when measurements are missing)
- provide a formal language for SLA policy formulation, such as the ones used in rule-based systems, meaning it is possible to validate the generation and deployment of consistent sets of rules
- manage the lifecycle of SLA policies (deploy, activate, de-activate both automated and operator driven)
- manage the consistency of deployed policies (automated and operator driven)
- monitor the status of system (health check) based on agreed observation pattern for an SLA policy: event based, periodically, on demand from components such as Policy Maker & Actuators
- generate, push and log (for successful/un-successful push) new detailed states towards Actuators, eCOP data repository
- perform automated management actions on deployed SLA’s in the limit of agreed types of actions from the associated contract

Due to the fact that – as previously mentioned – there is no standard to detail what SLA parameters are to be considered and which is the process to follow regarding SLA renegotiation, the SLA Renegotiation Controller will be able to provide a set of tools that enables DC operators to define both technical objectives and business objectives listed above as an SLA parameter.

A predefined combination of the above listed parameters, determined during the SLA template definition stage, outlines the different levels of service. It is obvious that the above SLA parameters are dynamic and hence can be renegotiated during the SLA lifecycle.

2.1.3. SLA Instantiation

The definition of the contractual SLA templates defined by the above parameters will be followed by the SLA instantiation and the signing of the contract by each DC customer prior to the instantiation of each VM. Thus the DC customer consents to a certain level of service, explicitly stating the level of service he wants to stand in and, most importantly, defines the percentage of time that he is willing to be downgraded to a lower SLA level. However, if a customer is downgraded to a lower SLA level for a period of time longer than the period agreed, a penalty charge must be paid by the DC operator in terms of a price discount.

The determination of the time percentage that the DC customer is willing to be downgraded to a lower SLA level is of paramount importance for the SLA Renegotiation process, providing a significant leeway to the SLA Renegotiation Controller. Thus, the SLA Renegotiation Controller

continuously monitors the SLAs to enforce them and also exploit the flexibility provided by the acceptable SLA downgrade toward energy efficient DC operation at a synergetic DC level.

2.1.4. SLA Monitor and Enforcement

To enforce the SLAs and exploit the provided flexibility, the SLA Renegotiation Controller is responsible for the SLA monitoring. In this course, the SLA Renegotiation Controller continuously monitors the VM SLAs and updates its current status each time an event is logged to the ICT Performance and Energy Supervisor. Thus, the SLA Renegotiation Controller interacts with the ICT Performance and Energy Supervisor in order to obtain information on the current status of the VM SLAs. More details regarding the interaction of the SLA Renegotiation Controller and the ICT Performance and Energy Supervisor are exposed in Chapter 3.1

2.2. Module decomposition

In order to support all stages of the SLA lifecycle described above, the SLA Renegotiation Controller comprises a number of subcomponents each one dedicated to a different SLA lifecycle stage. In the following paragraphs a description of the constituent elements of the SLA Renegotiation Controller is provided. The SLA Renegotiation Controller subcomponents are:

- SLA Template Repository
- SLA Monitor
- SLA Renegotiation Core

Whenever a DC customers request the deployment of a new VM, the SLA Renegotiation Core will be accessed and will interact with the SLA Template Repository to provide information on the available SLA levels, on the available virtual resources and on the implicit cost of the selected service. The service charges depends on the leeway provided by the customer toward SLA renegotiation, expressed as the percentage of time he is willing to be downgraded to a lower SLA level.

Concluding the SLA instantiation process, the customer selects one of the provided SLA levels for a certain time period and for a combination of reserved resources at a given price charge. At the same time the discount provided by the operator as penalty in case of a SLA violation is explicitly defined as well. All of these contractual aspects of any newly admitted VM are then reported and maintained by the Customer Profiles DB throughout the deployment of the VM.

After the VM request has been accepted, the SLA Monitor subcomponent monitors the SLA parameters of the VM at run time. The SLA Monitor interacts with the ICT Performance and Energy Supervisor Core and the eCOP DB each time an event is logged to the ICT Performance and Energy Supervisor, in order to obtain all necessary historic and real-time data. After an event is logged, the SLA parameters are updated. Thus, the SLA monitoring is not performed at predefined time intervals, but is performed each time an event is logged to the DOLFIN public subscriber.

Subsequently, each time the SLA Monitor subcomponent updates its SLA status, this status is forwarded to the SLA Renegotiation Core, which is the key component of the SLA Renegotiation Controller. Whenever activated by the SLA Monitor, the SLA Renegotiation Core receives both the information on the current SLA status from the SLA Monitor and on the contractual constraints of

each SLA from the Customer Profiles DB and outputs the current leeway provided by the SLA of each VM, i.e. currently acceptable down time, currently acceptable CPU speed, etc. to the VM Priority Classifier of the Energy Efficiency Policy Maker and Actuator. Moreover, the SLA Renegotiation Core is the interface of the component with the outside world, since all new VMs requests arrive at the SLA Renegotiation Core, it subsequently interacts with the SLA Template Repository as described above. The internal module architecture is presented in Figure 2-3.

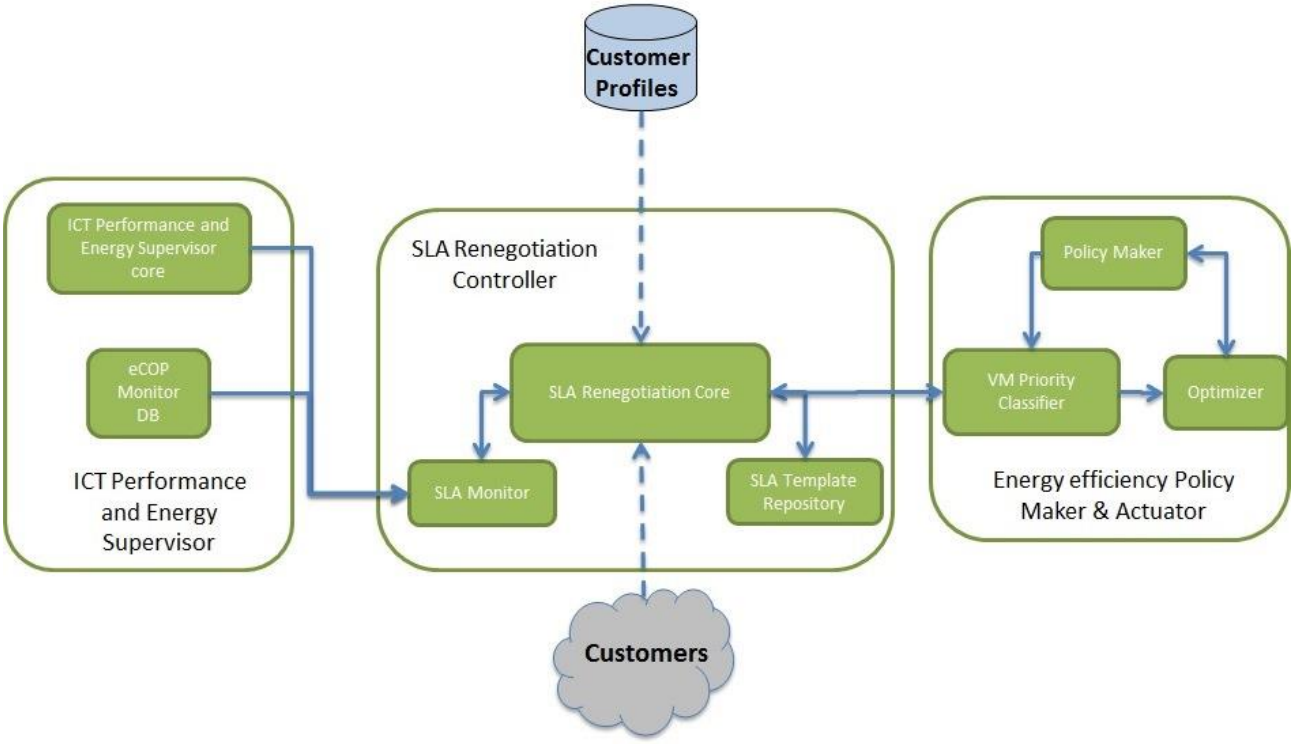

Figure 2-3 SLA Renegotiation Controller Internal Architecture

The SLA renegotiation process for the WIND Company is presented in Annex 1 as an indicative example, demonstrating the convergence with the DOLFIN renegotiation process described above.

2.2.1.1. *References to requirements*

This sub-section is presenting the list of requirements satisfied by this module. The complete list of identified requirements is described in the deliverable D2.2 [3].

Table 2-1: References to requirements (SLA Renegotiation Controller)

Requirement ID & Name	Requirement description
Q6 MgmntSLARenegotiation	The management subsystem shall be able to (automatically) request the SLA renegotiation to better adapt the SLA KPI to current optimal assets.
Q66 MonitorSLA Requests	The SLA Renegotiation Controller should keep track of the SLA Renegotiation requests for historical reasons.
Q8 MonitorTranslate	The monitoring subsystem shall always translate parameters to metrics (KPI).

Q50 Ask for PostponeExecutionPolicy	The Energy Efficiency Policy Maker and Actuator should be able to issue a request of the Workload and VM Manager to shift computing load (postpone it) when such a need arises.
Q51 Ask for RenegotiationPolicy	The Energy Efficiency Policy Maker and Actuator should be able to issue a request of the Renegotiation Policy e.g. renegotiation of Workload and VM Manager to shift computing load (postpone it) when such a need arises, when this load shifting could result in substantial cost reduction due to electricity price or it is considered to be profitable in the case of an end-user SLA renegotiation process, or such a request arrives from a DCO / Smart City operator.
Q57 HandleExceptions	When a customer denies a SLA renegotiation proposal or a synergetic DC denies cooperation, the DCO should be able to handle the rejection and consider alternative self-optimization actions
Q58: LogRenegotiationSteps	The SLA Renegotiation Controller should be able to persist the SLA renegotiation steps upon happening.
Q67 NegotiateSLAUpdate	When required, the SLA Renegotiation Controller should be able to actually perform the renegotiation (e.g. by the MgmntSLARenegotiation requirement)

3. Interactions

3.1. Detailed description

In order to fulfill its role regarding energy profile optimization of single and cross-DC landscapes, the SLA renegotiation controller interacts with Monitoring (both intra-DC and cross-DC) and Energy Efficiency Policy Maker and Actuators. Using the Rabbit MQ middleware this component reads/receive a number of key monitored measurements parameters, compute and validate the state of energy impact (formally defined in SLA policies) and on energy efficiency conditions, and generate decisions towards the Policy Maker & Actuators via the same messaging facility.

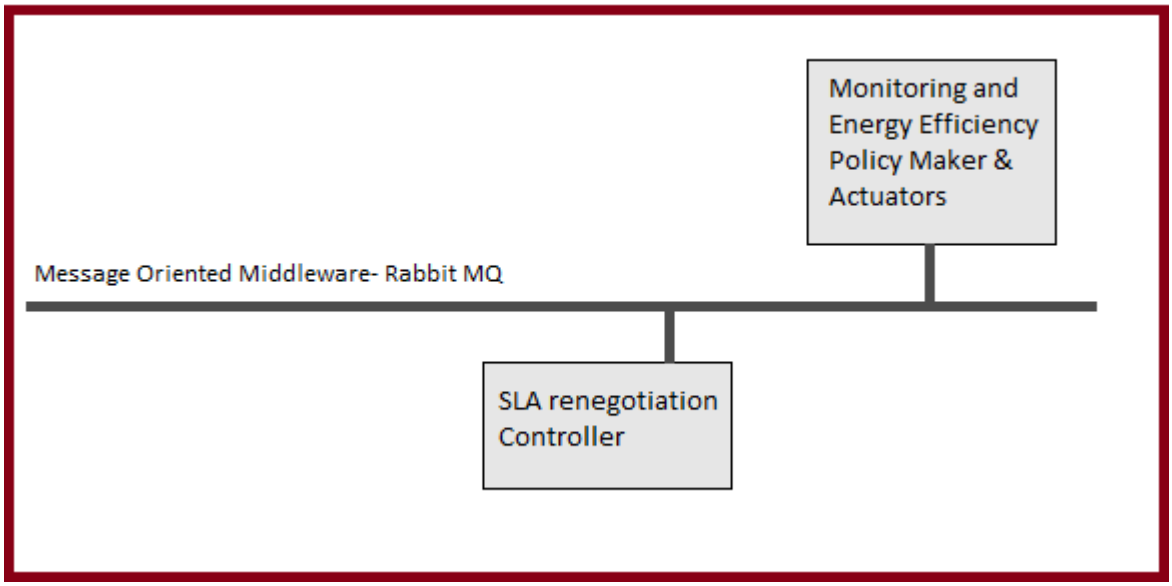

Figure 3-1: SLA Renegotiation Controller communication bus

Each time an event is logged to the ICT Performance and Energy Supervisor, the SLA Renegotiation Controller updates the status of the VM SLAs, collecting information related to the SLA parameters, i.e. the aggregate down time of each VM, the current VM CPU speed, etc. In order to obtain real time data streams such as the current VM CPU speed, the SLA Renegotiation Controller subscribes to the ICT Performance and Energy Supervisor Core, whereas in order to obtain historic data such as

the aggregate down time of each VM the eCOP DB is queried. Moreover, information related to SLA parameters of VMs migrating from federated DCs are encapsulated in the migrated VMs.

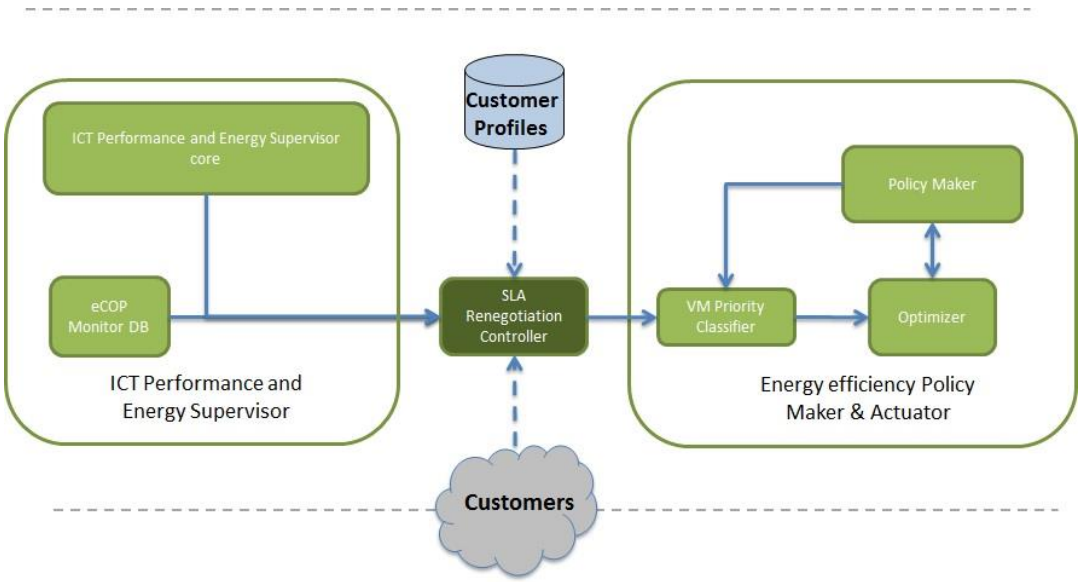

Figure 3-2: SLA Renegotiation Controller Interactions

The interaction between the above mentioned components enables the SLA Renegotiation Controller to continuously monitor the SLAs, whereas it also interacts with the Customer Profiles DB, which maintains information from the SLA instantiation stage on the DC customer contracts. After collecting all the needed information from the ICT Performance and Energy Supervisor, the eCOP DB, the Customer Profiles DB and the cross-DC VMs themselves, the SLA Renegotiation Controller interacts with the VM priority Classifier of the Energy efficient Policy Maker and Actuator to persist useful information on the SLA status. This information will be exploited later on by the Optimizer during the implementation of the revised optimization plan. The above interactions, during the SLA monitoring, the SLA instantiation and the SLA renegotiation process are detailed in Figure 3-2. This activity will be implemented by employing a Complex Event Processing (CEP) engine and connecting it to the publish-subscribe engine exposed by eCOP (ICT Performance and Energy Supervisor Core).

Once all monitoring information on the current SLA status have been collected, the SLA Renegotiation Controller provides information on the VMs that are amenable to certain policies (i.e. cross DC migration, load shifting, etc.) to the VM Priority Classifier. Moreover, the SLA Renegotiation Controller pinpoints the VMs which, for instance, have not reached their maximum down time or can still be downgraded to a lower SLA level and marks them as amenable to certain policies (i.e. VM cross-DC migration, etc.) providing these information to the VM Priority Classifier. Subsequently, this information is requested by the Policy Maker subcomponents of the Policy Maker and Actuator to be exploited by the Optimizer during the devise of a new optimization plan.

3.2. Interactions with the eCOP DB

In this section are described the used database entities for modelling of the SLA Renegotiation Controller. The eCOP database entities contains information regarding (1) the mapping between the users and their VMs, (2) the servers power source (Green energy or conventional energy sources), (3) the VM actions logs and (4) the contractual SLAs defined for each VM.

3.2.1. USER-VM Mapping (eCOP DB)

In the eCOP DB for every VM launched an entry is generated in the User–VM mapping table as described in Table 3-1.

Table 3-1: User VM mapping (eCOP DB)

VM_ID	User_ID
VM1	User1
VM2	User1
VM3	User1
VM4	User1

3.2.2. Green server mapping (eCOP DB)

When a physical server (host) is switched on, a log event is generated in the Green server mapping table. The power sources can be (1) conventional sources or (2) green power sources (e.g. photovoltaic, wind farms, etc.).

Table 3-2: Green server mapping (eCOP DB)

Server_ID	Is_green
Server1	TRUE
Server2	FALSE

3.2.3. VM action info (eCOP DB)

Each time a VM changes its state, a log event is generated and saved in the VM action information table in eCOP DB as described in Table 3-3. The possible statuses for a VM are: *Started, Stopped, Paused.*

Table 3-3: VM action information (eCOP DB)

VM_ID	Server_ID	Event	Timestamp
VM1	Server1	Started	11/6/2015 08:30:01
VM2	Server1	Started	11/6/2015 08:30:23
VM3	Server2	Started	11/6/2015 08:30:50
VM4	Server2	Started	11/6/2015 08:32:54

3.2.4. SLA contract info (DOLFIN DB)

The SLA contract info table represents a list of the contractual SLAs for each. Table 3-4 contains also data regarding the penalties applied in case of violating the SLA. The penalty parameter represents the incentives percentage refund to the customer in case of SLA violation. This parameter is a float value between 0 and 1, where penalty=1 means 100% refund.

Table 3-4: SLA contractual information (eCOP DB)

User_ID	VM_ID	Metric	Condition	Value	TimePeriod	Penalty
User1	VM1	ShutdownSecs	LESS_THAN	15	MONTH	A
User1	VM2	ShutdownSecs	LESS_THAN	20	MONTH	C
User1	VM3	ShutdownSecs	MORE_THAN	20	MONTH	D
User1	VM4	NotOnGSSecs	MORE_THAN	26	MONTH	E

4. Implementation and Integration description

4.1. Message Oriented Middleware - RabbitMQ

RabbitMQ is a message oriented middleware which is a message broker system and allows a process, *the publisher*, to write messages to an *exchange*. These messages are consumed by another process, called the *consumer* as depicted in Figure 4-1. The figure depicts the two main internal components of the message broker: (I) the *exchange*, which represents the entry point; and the (II) *queue* component, which acts like a container in which relevant messages are stored until their consumption.

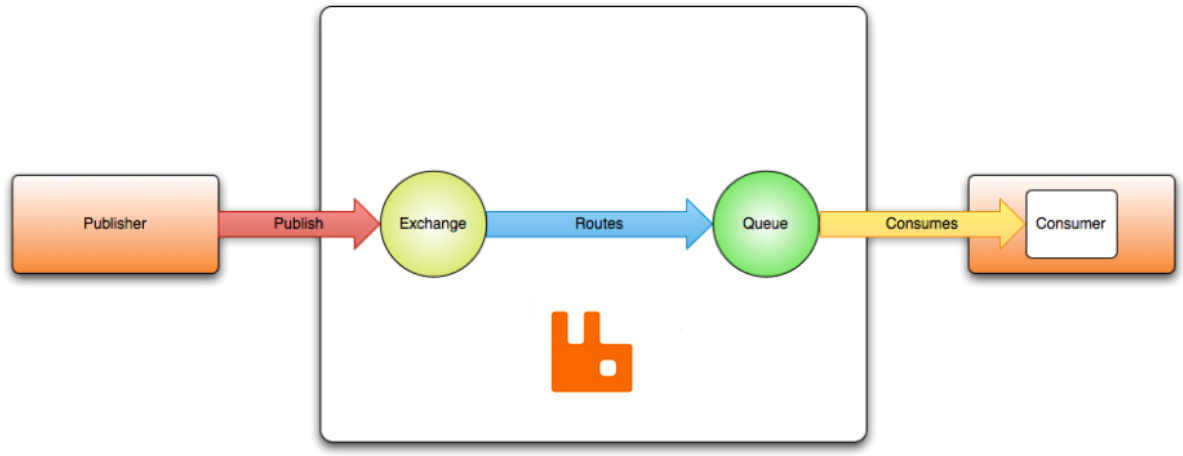

Figure 4-1: Direct message exchange in RabbitMQ [4]

The main advantages of using RabbitMQ, as listed in [4], are:

- **Reliability** - RabbitMQ offers a variety of features to trade off performance with reliability, including persistence, delivery acknowledgements, publisher confirms, and high availability.
- **Flexible Routing** - messages are routed through exchanges before arriving at queues. RabbitMQ features several built-in exchange types for typical routing logic. For more complex routing we can bind exchanges together or even write our own exchange type as a plug-in.

- **Highly Available Queues** - queues can be mirrored across several machines in a cluster, ensuring that even in the event of hardware failure your messages are safe.
- **Multi-protocol** - RabbitMQ supports messaging over a variety of messaging protocols.
- **Support for programming languages** - the clients for RabbitMQ broker can be implemented using several programming language, including JAVA, Ruby, Python, .NET, C/C++

4.2. Complex Event Processing – Esper

Esper is an open source event series analysis and Event Correlation Engine (Complex Event Processing - CEP) [5] that enables the detection of rich situations in event series (historical or currently-arriving) and triggers custom actions when event conditions occur among event streams as depicted in Figure 4-2.

Figure 4-2: Complex event processing workflow

Once event queries and pattern statements are registered in the Esper core container, event data are analyzed and can trigger arbitrary logic bound to the engine in the form of Plain Old Java Objects.

Esper and Event Processing Language (EPL) provide a highly scalable, memory-efficient, in-memory computing, SQL-standard, minimal latency and real-time streaming-capable Big Data processing engine for any-velocity online and real-time arriving data and high-variety data, as well as for historical event analysis.

Esper can be easily embedded in an existing Java application or middleware to add event-driven capabilities to existing platforms without paying high serialization cost or network latency for every message received and action triggered.

For the SLA Renegotiation Controller, Esper represents a great CEP engine providing fast and real-time log events processing and the possibility to generate warning events when a SLA violation has occurred or is about to occur.

4.3. Message structure

The SLA Renegotiation Controller communicates with the Monitoring (both DC and cross DC) and Energy Efficiency Policy Maker & Actuators via the Rabbit MQ middleware as described in 3.1, using

the AMQP protocol on port number 5762. The messages exchanged between the SLA Renegotiation Controller and the related components are (1) VM manager events, (2) VM events, (3) Host events and (4) Green host events.

These events are exchanged on a single topic (Exchange) using 4 different routing keys (Routes), as described in the RabbitMQ communication model in [4]. All messages are exchanged on a predefined topic, named *"vim"* and on 4 routing keys which are defined for each message type:

- VM_MANAGER_RK = "key.manager" ;
- VM_RK = "key.vm";
- HOST_RK = "key.host";
- GREEN_RK = "key.host.green";

In the next sections the message structure for each message type exchanged on AMQP protocol is described.

4.3.1. VM Manager Event

The VM manager event contains log information regarding the VM, the host (physical server) and the manager actions (e.g. *START_VM*, *STOP_VM*, *STOP_VM_CRASH (*generated when a VM crashes), *START_HOST*, *STOP_HOST*, *START_VM_MANAGER* and *STOP_VM_MANAGER*). This event is generated each time a VM, server or the manager changes its state. In Snippet 1 is presented an example of event generated when a VM is started.

```
{
  "message":"event_notification",
  "timestamp":1438343686617
  "payload":{
     "eventType":"START_VM",
     "id":29,
     "name":"Router-29"
     },
 }
```

Snippet 1: VM Manager event message example

4.3.2. VM Event

The VM events provide information regarding the VM status and the VM performance parameters. These events are generated at a predefined time period established by the DC. An example of such event is presented in Snippet 2.

To elaborate on the VM event parameters, the *cpuSys* refers to the CPU usage duration for system applications expressed in milliseconds. The *cpuUser* represents the time period used by the CPU for user's applications and the *cpuTotal* is the CPU usage duration for both system apps and user's apps in ms. The *energy parameter* represents the power consumed by the VM at the present moment (when the event is generated) and the *energyTotal* refers to the total energy consumed by the VM from the start.

```
{
  "message":"event_notification",
  "timestamp":1438343967009,
  "payload":{
    "vmId":10,
    "hostId":"localhost:10002",
    "vmName":"Router-10",
    "cpuSys":3.6989998817443848,       // CPU usage duration for system apps [ms]
    "cpuTotal":19.145000457763672,
    "cpuUser":15.446000099182129,    //CPU usage duration for user's apps [ms]
    "elapsedTime":475504,            // functioning duration [ms]
    "energy":2.1361015319823764,     // instantaneous power consumption [kW]
    "energyTotal":283.8441986083984, // energy used by the VM from start [kWh]
    "memory":87544,
    "starttime":1438343488841
   },
}
```

Snippet 2: VM event message example

4.3.3. Host Event

The host event provides information about the physical server performance parameters. These parameters include information regarding (1) the server CPU usage percentage, (2) the network traffic values (in/out Bytes and in/out Packets) and (3) server used/free memory as presented in Snippet 3.

```
{
  "message":"event_notification",
  "timestamp":1438343961986,
  "payload":{
    "hostId":"localhost:10001",
    "cpuIdle":70.14,                 // CPU used in idle mode (%)
    "cpuSys":15.67,                           // CPU used by OS (%)
    "cpuUser":14.17,                 // CPU used by user's apps (%)
    "freeMemory":2175,
    "inBytes":3857,
    "inPackets":35,
    "outBytes":15739,
    "outPackets":47,
    "usedMemory":14203
  },
}
```

Snippet 3: Host event message example

4.3.4. Green Host Event

The Green host event contains information with respect to the server energy source and is generated when a server is launched. The possible statuses for this event are GREEN when the server is powered using renewable or green energy sources, or NORMAL when conventional energy sources are used. Snippet 4 exemplifies the launching of a green server.

```
{
  "message":"event_notification",
  "timestamp":1438343961456,
  "payload":{
    "hostId":"localhost:10001",
    "status":"GREEN"
  },
}
```

Snippet 4: Green host event message example

This event is of paramount importance for enforcing the Green SLAs described in next section as a CEP rule.

4.4. CEP processing rules

The SLA Renegotiation Controller core is represented by the Esper CEP engine which receives and monitors all the event streams presented above, aggregate them and generates a warning event as soon as a predefined SLA rule is violated, (this process is presented in 4.2) via the RabbitMQ message middleware.

The Snippet 5 describes a critical event sent to the Energy efficient Policy Maker & Actuator when a violation occurs or a threshold has been reached. To elaborate on the event parameters, *type* represents the name of the SLA rule, the *level* parameter refers to the event criticality so the critical events will have the highest priority. The *timeRepresentation* is a description of the *time* parameter, so it could be 1) the time until an event or violation will occur, 2) the duration since an event has occurred (e.g. VM is running on a non-green server).

```
public class CriticalEvent{
        private String vmId;
        private String hostId;
        private String type;         // Green violation, Unexpected shutdown
        private String level;        // WARNING, CRITICAL
        private String timeRepresentation; // AFTER, UNTIL DURATION :
        description of what time variable means
        private long time;  // could be the time to an event or duration
        private float penalty;      // from 0 to 1
```

Snippet 5: CEP rule for VM unexpected shutdown

The CEP rules are defined in accordance with de DC Service Level Agreement (e.g. Annex 1 – WIND SLA Renegotiation) and as proof of concept, the next two subsections will present examples of SLA rules (Availability and Green SLAs).

4.4.1. SLA Rule for Unexpected Shutdown

As a proof of concept a CEP rule has been defined for Availability Service Level Objective (SLO) as described in Snippet 7. For this rule the CEP engine monitors the VM Manager events (see 4.3.1) and in particular the unexpected shutdown VMs (e.g. *eventType*= STOP_VM_CRUSH).

When using the *insert* keyword in CEP engine, a table in-memory is created containing the information requested after the *select* keyword. Each time an event is fulfilling the condition after *where* a new entry is created. The first two expressions are creating two tables: (1) launched VMs and (2) crushed VMs containing information regarding the *timestamp*, the *VM id* and the *eventType* (Snippet 6).

```
private String vmStartAction
private String vmCrashAction = "STOP_VM_CRASH";

expression = "insert into startVMs select timestamp, payload.eventType, payload.id
             from VmManagerMessage where payload.eventType = \""
             + vmStartAction + "\"";

expression = "insert into crashedVms select timestamp , payload.eventType,
             payload.id as id from VmManagerMessage where payload.eventType = \""
             + vmCrashAction + "\"";
```

Snippet 6: CEP expressions to create StartVMs and CrushedVMs tables

When a VM crush and remains in the same state for more than a predefined time period (e.g. 60 seconds) the CEP engine is adding a new entry in the *criticalAvailabilitySituation* table, after executing the expression presented in Snippet 7 and a critical event is generated.

```
expression = "insert into criticalAvailabilitySituation
              select cas.timestamp as timestamp, cas.id as id
              from pattern
              [every cas = unexpectedShutdownVms ->
              (timer:interval(60 + sec) and not stm=startVMs
              (cas.id=stm.payload.id))]"
```

Snippet 7: CEP rule for VM unexpected shutdown

4.4.2. SLA Rule for Non-Green Server

The SLA rule for non-green server refers to the time period in which a VM is running on a server powered with conventional energy sources (NORMAL). In order to create this rule the *Green host* events are monitored along with the VM manager events.

The first two expressions will create two tables 1) a non-green servers list and 2) a list with the stopped VMs as described in Snippet 8. In addition we will need the *startVMs* list presented in the previous subsection and a *nonGreenVMsStarted* table which will contain all the VMs started on a non-green host.

```
private String nonGreenStatus = "NORMAL";

expression = "insert into nonGreenHosts select payload.status, payload.hostId from
             GreenHostMessage where payload.status = \""
             + nonGreenStatus + "\"";

expression = "insert into stopVMs select timestamp, payload.id from
             VmManagerMessage where payload.eventType = \""
```

```
            + vmStopAction + "\"";
```

Snippet 8: CEP expressions to create nonGreenHosts and StopVMs tables

Each time a server is launched, a Green host event (see 4.3.4) is generated and, if the energy source is conventional, the server is saved in the non-Green hosts table. When a VM is started on a non-Green server and it is not stopped or moved on a green server for an established time period (e.g. 30 minutes) then an entry in the *criticalSituation* table is created and a warning event is generated.

```
expression =”insert into criticalSituation
            select cs.timestamp as timestamp, cs.vmId as vmId,
            cs.hostId as hostId, cs.status as status    from pattern
            [every cs = nonGreenVmsStarted ->
          (timer:interval(30 + min) and not svm=stopVMs (cs.vmId=svm.payload.id))]”
```

Snippet 9: CEP rule for running on a non-Green server

To exemplify the CEP rules defined above, a proof of concept has been developed to monitor the duration a VM runs on a non-Green host. The Snippet 10 presents a warning event generated after a non-Green server rule violation. The penalty for running 30 minutes on a non-green server is 50% from the monthly payment in this example, but as specified in the previous chapters, each DC is defining the SLAs, the thresholds and the incentives with each customer.

```
{
  "message":"Event warning",
  "timestamp":1438343752474,
  "payload":{
    "vmId":"36",
    "hostId":"localhost:10000",
    "type":"GREEN VIOLATION",
    "level":"WARNING",                          // can be CRITICAL, WARNING
    "timeRepresentation":"duration",            // description of what time variable means
    "time":30,                                  // can be time to an event or duration
    "penalty":0.5                               // from 0 to 1
  },
}
```

Snippet 10: Warning event message example

4.4.3. Integration between components

The integration of SLA Renegotiation Controller needs to be performed at two levels:

- Internal (Work Package 4 components only)
- Cross-package (selected Work Package 3 components)

For internal integration the following WP4 components must be present (depicted in Figure 2-1):

- Customers Profiles – provides persistence and management layer for the customer's profiles.
- Cross-DC Workload Orchestrator – is in charge of managing the full lifecycle of the virtual routers in the network and the allocation of the applications running on the virtual nodes.

Cross-package components required for integration with SLA Renegotiation Controller are shown in Figure 3-2:

- ICT Performance and Energy Supervisor core – performs analysis of performance data monitoring and energy data.
- eCOP Monitor DB – stores all real-time and historical energy related data collected from DCs. This information is used to predict VM loads. Provides API for CRUD (Create, Read, Update and Delete) operations for VMs, Servers, Network elements, Metrics, Measurements, Lighting and HVAC (Heating, Ventilation and Air Conditioning).

ICT Performance, Energy Supervisor core and eCOP Monitor DB make up the ICT Performance and Energy Supervisor module.

At both levels of integration, components talk to each other using a real-time and non-blocking communication channel Rabbit Message Queue. This make it possible for the SLA Renegotiation Controller to fetch messages pushed by ICT Performance and Energy Supervisor module, process them and post updates for VM Priority Classifier (part of Energy Efficiency Policy Maker and Actuator module).

5. Conclusions

This document has presented the SLA Renegotiation Controller detailed functional design, the implementation and the external interfaces. The document has described also the SLA Renegotiation process which refers to automatically negotiating and agreeing with DC customers in balancing their SLA performance parameters in exchange of decreased service prices.

In order to capitalize on the merits offered by the SLA Renegotiation into the DOLFIN architecture, the SLA Renegotiation Controller was designed as a component dedicated to:

- monitoring of the SLA parameters,
- enforcement of the SLAs,
- automatically renegotiate the SLAs, based on the predefined customer profiles.

The SLA Renegotiation Controller interacts with the Customer Profiles DB, which maintains information from the DC customer contracts and interacts also with the VM priority Classifier of the Energy efficient Policy Maker and Actuator to persist useful information on the SLA status.

As described in the previous chapters, the SLA Renegotiation Controller continuously monitors a number of events streams presented in 4.3 using the Esper CEP engine, aggregates them and generates warning events to be sent to the Policy Maker and Actuators when a predefined SLA rule is violated, via the RabbitMQ message middleware.

Besides the major importance of warning events generated by the SLA Renegotiation Controller used for SLA enforcement, this component provides also information on the VMs that are amenable to certain policies (i.e. cross DC migration, load shifting, etc.) to the VM Priority Classifier.

6. References

[1] EUROPEAN COMMISSION DIRECTORATE GENERAL COMMUNICATIONS NETWORKS, CONTENT AND TECHNOLOGY UNIT E2 - SOFTWARE AND SERVICES , CLOUD , "Cloud Computing Service Level Agreements," [Online]. Available: http://ec.europa.eu/information_society/newsroom/cf/dae/document.cfm?doc_id=249.

[2] C. S. Vladimir Stantchev, "Negotiating and Enforcing QoS and SLAs in Grid and Cloud Computing," *Advances in Grid and Pervasive Computing,* vol. 5529, pp. 25-35, 2009.

[3] D2.2 - DOLFIN requirements and system architecture, [Online]. Available: http://www.dolfin-fp7.eu/wp-content/uploads/2014/10/DOLFIN_D2.2_UCL_30-09-2014_FF.pdf.

[4] "RabbitMQ," [Online]. Available: https://www.rabbitmq.com.

[5] "Esper: Event Processing for Java," [Online]. Available: http://www.espertech.com/products/esper.php.

7. Annex 1 – WIND SLA Renegotiation

A service level agreement (SLA) is part of a service agreement between a service provider and a service customer – originally created in the 1980's by telecommunication companies in order to determine a reliable level of quality of service (QoS) by defining measurable parameters. Nowadays they are being employed nearly everywhere in the ICT sector. Executable SLAs are mostly denoted in the XML format.

Unfortunately there are no real standards for SLAs, however there is some typical content which are shown in the following picture:

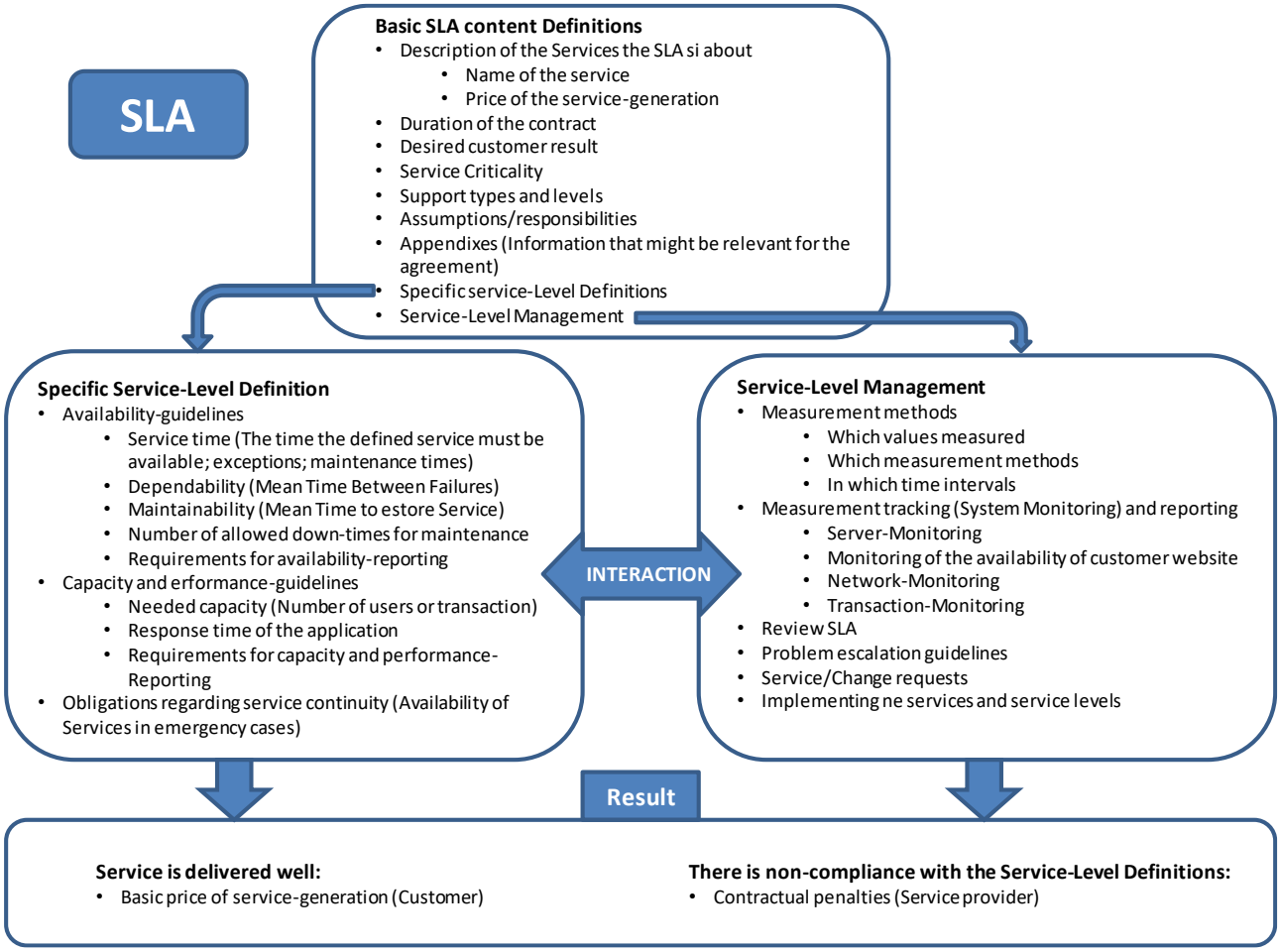

Figure 7-1: SLA Components

To elaborate more on Figure 7-1 the basic SLA content definitions are presented below:

1. *Description of the offered services:* First the service provider has to give a detailed description of the service he wants to offer. For example the name of the service must be defined. The pricing could also be part of an SLA.

2. *Duration of the contract*: The duration of the SLA has to be regulated. The beginning and ending date and other arrangements in relation to the termination of the acknowledgement have to be defined.

3. *Desired customer result:* What result the customer would like to reach? Which business-processes and activities of the customer are supported by the service? What's the desired result in relation to utility and warranty?

4. *Service criticality*: Estimated damage for the customer which results from the loss of the service.

5. *Support types and levels:* This part of the SLA describes the support types and support levels the customer needs. Does the customer need on-site support or Remote support? What is the response time or the time for solving a problem?

6. *Assumptions/responsibilities:* Which obligations does the service provider have? Which obligations does the customer have (for example in relation to it-security)?

7. *Appendices:* Appendices are used to include additional information that might be relevant for the agreement.

8. *Specific Service-Level Definitions:* Here technical guidelines of the service execution are defined: parameters are specified and metrics to monitor compliance are determined. In order to fulfil the contract, the service-provider has to demonstrate compliance with the guidelines. Among them, for instance:

9. *Service Level Monitoring and Management:* The agreed SLO (service level objectives) need to be monitored at runtime in order to determine if they are breached or not. In case of breaching the SLAs, countermeasures need to be taken.

10. *Penalties:* In the case of SLA breach the contract also defines the kind of consequences of such a breach:
    - A financial compensation like pricing reduction
    - Other kinds of compensation like the change of a customer group or other special treatment

7.1. Green Service Level Agreements (GreenSLA)

A GreenSLA is a type of SLA between a service provider and its customers that offers an extended scope of eco-optimization to the service provider:

- by relaxing traditional performance parameters,
- by introducing novel energy performance parameters as classifying elements,

- by offering incentives to the customers in exchange for a specified variation of traditional performance metrics.

A GreenSLA is always offered alongside a traditional SLA it compares to. Thus the GreenSLA modifies the attributes of the underlying service, not its main characteristics.

Usually SLAs are used as a legal document that provides the basic legal setting, technical objectives and penalties (legal and monitoring function); however, in SLA research it has also been used as an instrument that actively steers the DC operation (SLA-based scheduling). The target could be to enable DCs to comply with the trigger given by the energy provider and react to unpredictable events (sudden demand and supply spikes). As scheduling is mainly based on prediction and planning, the scheduling algorithm though, being linked to SLA restrictions, cannot be designed as a direct translation of an SLA into an algorithm but will be much more complex since it has to take many scheduling parameters into account.

7.2. Vision for a WIND Data Centre Federation

7.2.1. Introduction

This section describes the vision for the DC Federation. Firstly Green SLAs with its customers have to be introduced. Secondly, the DOLFIN collaboration tool within the DC federation system has to be deployed at the DC site. It is envisioned to apply Green KPIs for energy efficiency. Thirdly a broader product & services portfolio has to be introduced which aims at higher energy efficiency. Furthermore, a change of the cost and pricing model is applied across the DC Federation, to take into account the Green SLAs. From an IT Customer perspective, the existing products will be enhanced with a set of new products (e.g., “XL physical server with low energy consumption/emissions”) and services (e.g., “virtualized database server with energy-friendly data backup policy”, “energy-friendly cold standby and replication services”), offered in conjunction with the GreenSLAs. In addition to the existing standard SLAs, the DC infrastructure services provider (WIND) will add several green parameters such as allowed downtime, recovery time objective, maximum allowed emissions, maximum workload per server, maximum energy consumption per server. Additionally, it will provide incentives such as discounts, additional workload capacity, additional usage time and/or additional energy quota per server.

7.2.2. Measures for the DC Federation to better respond to Variations in Energy Delivery

WIND Data Centre nodes can evaluate and support the collaboration goals within the eco-system by applying some technical innovation measures (such as more energy efficient hardware, power and cooling systems) but also by applying better **workload planning** through the following measures:

- **DC Workload consolidation**: moving all related tasks on the machines which are most frequently and most heavily used. This might be done on demand, in which case it is the outcome of negotiations between DCs in the DC federation. It can also be performed in combination with **Server Consolidation** and with **Data Migration and/or Replication**, but these operations are done manually and are out of scope for the project.

- **DC Workload Control through re-scheduling** based on negotiation between DC and ITC agents:
    - **workload shifting through re-scheduling**: tasks are re-planned with minimal impact on green SLAs with ITCs
    - **workload migration**: workload is shifted between the individual DCs and/or redistributed across the servers of a DC federation, with the goal to minimize the overall energy consumption at server site level.
- **DC Workload Optimization Measures**: when a DC server is not servicing requests on its virtual machine instances, it is either turned off or hibernated, within the limits allowed by the Green SLAs applicable for the DC site.
- **DC measures to delay energy consumption in time**: the DC is able to delay some tasks (such as, backups, periodical security checks) for short periods of time, in order to give priority to requests from EP to temporarily reduce energy consumption. This is done through prioritization of tasks, understanding the impact of different user and DC-specific tasks (e.g. maintenance, authentication, security scans, etc) to the overall energy consumption of the DC.

DC agents try to minimize their Energy consumption, by:

- **Providing** "Special offers", i.e. **discount prices** or free workload time for End Users' agents that use Green SLAs;
- **Advertising available workload capacity** (time frames with available computing capacity) to other DC in the same DC federation, when their infrastructure is not used at the maximum capacity.

7.2.3. End User – Data Centre Relationship

Infrastructure

Service Level Reporting infrastructure will be directly connected to the monitoring infrastructure, and will provide the End User with real time insights into their energy consumption of their workload.

Workload Migration to another Data Centre and/or dynamic re-scheduling of tasks (delaying them to a time slot after the energy demand peak has passed) can be triggered by the DC. Any task rescheduling needs to occur within the limits allowed by the existing Green SLA ITC-DC.

Business Model

The WIND Federated DC is a provider-managed hosting service provider of a multi-tenant computing infrastructure. In this system, services are provided using menu-like fixed rate services, based on two types of tariffs: peak load versus normal load time slots. In addition, DC Federation can provide several energy-consumption optimization services towards its ITCs:

- Manage multi-node Federation workload agreements: Manage and Monitor other DC Federation Server statuses, which allows connecting, monitoring and optimizing existing Servers and Virtual Machines and the allowed workload transfer between the DC Federation nodes

- Provisioning and Optimization of energy-aware Data and Security services (e.g. application and data recovery, disaster management) which include explicit energy load and emissions impact, facilitating Service Level Agreement recommendations (Time slots and priorities) for tasks concerning eg: Data and Application Backup; Disaster Recovery; Virtualization services; Load balancing; Application-level server monitoring and performance tuning.

Changes and upgrades of SLAs must be possible in order to allow the End User to choose a "greener" service or otherwise to temporarily choose a better performing setting with some penalties for lower energy efficiency.

The frequency of changes needs to be established to match the change policy that results in the most energy efficient setting (allowing the interests of all parties in the ecosystem to reach equilibrium). The most likely period to apply changes will be the period used for billing (monthly basis) or the period used for service level reporting (which ranges from a few weeks to one year).

Cost and pricing model will change in the WIND Data Centre Federation due to the adoption of Green SLAs and workload control strategies.

In addition to the existing baseline pricing model, i.e., *price proportional with the use of the WIND infrastructure allocated* for each ITC, an additional *price discount structure* can be conceived, *proportional with the impacted Green SLA service levels* selected by ITC.

Another aspect of the changed pricing model could consists of introducing explicit penalties for a DC missing its Green SLA service level targets: for each hour of computing time delayed, i.e. DC missing its target service levels in the services provisioned to ITC, a penalty is specified for DC or, alternatively, additional computing time is made available for the ITC.

Another expected change of the pricing model should be differentiation of computing time cost between the high/peak workload hours and low workload hours, as these will involve higher workload demand and higher energy costs for the DC.

The costs of the traditional multi-tenant Federated Data Centre are based on server cabinet rates as the main cost contributor for server infrastructure rental, plus the estimated usage costs for each service in the service portfolio. In the new situation, the main cost contributors for the rented servers are: the total cost of computing, formed by the direct energy cost consumed by the rented IT equipment, plus the fraction of cooling and UPS cost proportional to the occupied space; the cost of network traffic, for transferring virtual machines data and applications through the communication lines; the cost of storage, determined by the cost of storage space and the number of storage connections; and the cost of outsourcing services to other nodes of the DC Federation.

7.3. Wind Data Centre Service Level Agreement

A typical Wind Data Center Service Level Agreement for a Web Site running on a VM is exemplified below.

1. Coverage and Definitions

1.1 This Web Site Availability Service Level Agreement ("**SLA**") applies to you ("**Customer**") if Customer has ordered Smart Machines, Smart Appliances, and/or Virtual Machines from WIND (the "**Services**") and Customer's account is current (*i.e.*, not past due) with WIND.

1.2 As used herein, the term "**Availability**" means the percentage of a particular month (based on 24-hour days for the number of days in the subject month) that the content of Customer's hosted system is available for access by third parties via HTTP and HTTPS, as measured by WIND.

2. Service Level

2.1 **Goal:** WIND's goal is to achieve 100% Availability for all customers.

2.2 **Remedy:** Subject to Sections 3 and 4 below, if the Availability of Customer's Services is less than 100%, WIND will credit the Customer 5%) of the monthly fee for each 30 minutes of downtime (up to 100% of Customer's monthly fee for the affected server).

3. Exceptions

3.1 Customer shall not receive any credits in connection with any failure or deficiency of Availability caused by or associated with:

(i) circumstances beyond WIND's reasonable control, including, without limitation, acts of any governmental body, war, insurrection, sabotage, armed conflict, embargo, fire, flood, strike or other labour disturbance, interruption of or delay in transportation, unavailability of or interruption or delay in telecommunications or third party services, virus attacks or hackers, failure of third party software (including, without limitation, e-commerce software, payment gateways, chat, statistics or free scripts) or inability to obtain raw materials, supplies, or power used in or equipment needed for provision of the SLA;

(ii) failure of access circuits to the WIND CLOUD Network, unless such failure is caused solely by WIND;

(iii) scheduled maintenance and emergency maintenance and upgrades;

(iv) DNS issues outside the direct control of WIND;

(v) issues with FTP, POP, IMAP, or SMTP Customer access;

(vi) false SLA breaches reported as a result of outages or errors of any WIND measurement system; Customer's acts or omissions (or acts or omissions of others engaged or authorized by Customer), including, without limitation, custom scripting or coding (*e.g.*, CGI, Perl, HTML, ASP, Ruby, PHP, Python etc), any negligence, wilful misconduct, or use of the Services in breach of WIND's Terms of Service and Acceptable Use Policy;

(vii) e-mail or webmail delivery and transmission;

(viii) DNS (Domain Name Server) Propagation;

(ix) outages elsewhere on the Internet that hinder access to Customer's account.

3.2 WIND is not responsible for browser or DNS caching that may make Customer's site appear inaccessible when others can still access it.

3.3 WIND will guarantee only those areas considered under the control of WIND: WIND CLOUD server links to the Internet, WIND CLOUD's routers, and WIND CLOUD's servers.

4. Credit Request and Payment Procedures

4.1 To receive a credit, the Customer must make a request by sending an e-mail. The e-mail message must include the domain name and the username of the Customer's account in the "Subject" line.

4.2 Each request in connection must include the Customer's account number (per WIND's invoice) and the dates and times of the unavailability of Customer's Web site and must be received by WIND within 10 (ten) business days after the Customer's Web Site was not available. If the unavailability is confirmed by WIND, credits will be applied within two billing cycles after WIND's receipt of the Customer's credit request. Credits are not refundable and can be used only towards future billing charges.

4.3 Notwithstanding anything to the contrary herein, the total amount credited to Customer in a particular month shall not exceed the total hosting fee paid by Customer for such month for the affected Services. Credits are exclusive of any applicable taxes charged to Customer or collected by WIND and are Customer's sole and exclusive remedy with respect to any failure or deficiency in the availability of the Services.

Note: Credits are not refundable and can be used only towards future billing charges.