Data Centers 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 1th, 2013 (36 months duration)**

Deliverable D4.1

Synergetic Data Centres for energy efficiency (Design)

Due date:	1/03/2015
Submission date:	30/04/2015
Deliverable leader:	UCL
Author list:	S. Clayman (UCL), A. Galis (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**
Interoute	M. Biancani, A. Kollaras
Siemens	S. Cosmin Nechifor, A. Nistor
WIND	A. Reale
UCL	S. Clayman, A. Galis, L. Mamatas
GRNET	E.Angelou, G.Goumas, J.Siakavaras, C. Georgiou, S. Aristeides, G. Spyroglou
Synelixis	A.Voulkidis, A. Aravanis, K. Pramataris, Ch, Tsiopoulou, Th. Zahariadis
I2CAT	I. B. Rodrigues, I. Canyameres, A. Legarrea

Table of Contents

Figures Summary

Tables Summary

Abbreviations

AP	Action Point
BMS	Building Management System
CEN	European Committee for Standardization
DB	Data Base
DC	Data Centre
DC	Data Centre
DCF	DC Federation
DCIM	Data Centre Infrastructure Management
DCO	Data Centre Optimization
DoW	Description of Work
EC	European Commission
EC	European Commission
EMS	Environmental Management System
EPs	Energy Providers
ESO	European Standards Organization (i.e. CEN, CENELEC and ETSI)
ETS	Emissions Trading System
ETSI	European Telecommunications Standards Institute
EU	European Union
FCoE	Fibre Channel over Ethernet
GUI	Graphical User Interface
ICT	Information and Communication Technology
IDB	Information Data Base
IPR	Intellectual Properties Rights
ISO	International Organization for Standardization
ITEU	IT Equipment Utilization

ITU-T	International Telecommunication Union - Telecommunication standardization sector
KPI	Environmental Key Performance Indicator
KPIEC	Energy Consumption
KPIREN	Renewable Energy Use
PM	Person Months
PMO	Project Management Office
QCT	Quality Check Team
SDC	Synergetic DCs
SG	Smart Grid
SGC	Smart Grid Controller
TMC	Technical Management Committee
ToC	Table of Content
UPS	Uninterruptible Power Supply
VM	Virtual Machines
WBD	Working Day Before Deadline
WP	Work package

Executive Summary

DOLFIN aims to significantly contribute towards improving the energy efficiency of Data Centres and the stabilization of Smart Grids, across networks of Data Centres and Smart Grids, following a holistic approach. The DOLFIN view of tomorrow's Data Centres (DCs) involves energy-aware, interconnected, interworking DCs that are able to efficiently operate in a federated environment, wherein all DC resources are optimized at multiple levels ranging from server-level to network-of-DCs level. In order to address the challenges that arise from such a perspective, DOLFIN comprises the local and federated energy efficiency optimization sets of components. The federated optimization components form the Synergetic DCs (SDC) part of the DOLFIN architecture. Although a first glance of this part has been already provided through the deliverable D2.2 [1], a detailed, clear view of the exact functionality and interworking characteristics of it were not provided in the former deliverable.

This deliverable is the detailed design specification of SDC part of the consolidated DOLFIN architecture as developed in the WP4. In addition to the presentation of the SDC components detailed specification, their interworkings interfaces and they relationship to the requirements a number of topics addressing both WP4 and WP3 developments have been tacked, including a consolidation of the DOLFIN Software Platform, a presentation of the basic control flows governing the interoperability of the eCOP (WP3) and SDC (WP4) subsystems, and a presentation of the energy models to be used in all DOLFIN optimization processes.

The document starts with a consolidation of the aforementioned common topics, in chapters 1 - 3. The detailed specification of the SDC components and their interworking characteristics are presented in chapter 4. The appendices contain detailed specifications of the interworking interfaces for eCOP and SDC components respectively. Specifically regarding chapter 5, three main sub-sections are given:

- ***SDC functional description*** – it provides a functional overview of the WP4 components that are enabling a distributed network of co-operating DCs.
- ***Modules interfaces*** – it provides has a high-level view of cross DC interactions, showing the distributed interaction of the Cross-DC. These elements of the Synergetic DCs create a distributed infrastructure that interacts with and is reliant on the elements of the eCOP system within each DC.
- ***Internal design for cross-DC and Smart Grid specific modules –*** it provides details in the Cross-DC Workload Orchestrator, the Cross-DC VM Manager, the Cross-DC Monitoring Data Collector, the DOLFIN Information Data Base, the Smart Grid Controller, the SLA Renegotiation Controller and Customer profile management.

Section 6 concludes the document, deriving hints for future development work. Section 7 presents the reference list. In addition to the complete specification of the various SDC components, this deliverable has 3 Annexes: Annex 1 presents a first set of APIs that will form the basis of the on-going and following SDC

components implementation processes, Annex 2 presents the API of the VLSI (the Very Lightweight Software-Defined Infrastructure), which we consider very important for the future Software Defined DCs, and Annex 3 presents WIND's vision and DCs SLA agreements.

1 DOLFIN architecture focusing on synergetic DCs

Large DCs are industrial scale operations using as much electricity as a small town and sometimes are a significant source of air pollution in the form of diesel exhaust. A DC is a logically centralized physical and virtual repository for computation, storage, management, services and the dissemination of data and information organized around a particular body of knowledge or pertaining to a particular business. The software elements of these DCs are realised and based on computing cloud principle and technologies. In general the hardware of the DCs includes redundant server and network node capabilities, as well as backup power supplies, redundant data communications connections, environmental controls (e.g., air conditioning, fire suppression) and various security devices.

DOLFIN's primary objective is to design, develop, and validate the DC platform capable of monitoring the energy usage of the DC and react accordingly for efficient energy management. The design of the DOLFIN DC framework would explicitly accommodate energy management and it is aimed at:

- Improving capital and operational efficiencies for DC operators through the use of a common organization, automation, and operations of all energy functions across the different domains
- Migrating from an ecosystem of separate energy management functions towards a coordinated arrangement of energy management functions as represented in Figure 1-1:

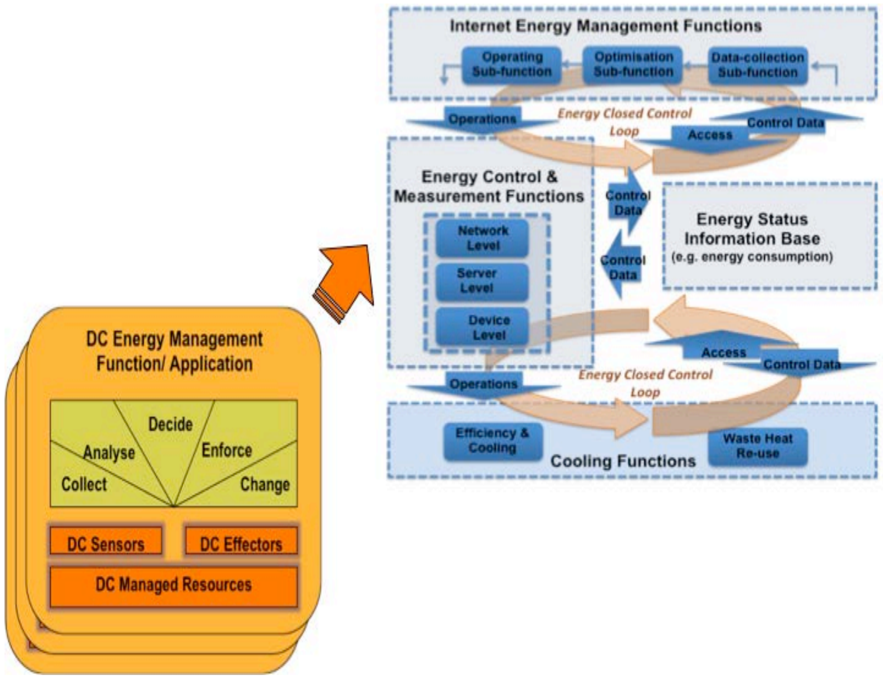

DC Energy Management Function/ Application
Decide
Analyse
Enforce
Collect
Change
DC Sensors
DC Effectors
DC Managed Resources
Internet Energy Management Functions
Operating Sub-function
Optimisation Sub-function
Data-collection Sub-function
Operations
Energy Closed Control Loop
Access
Control Data
Energy Control & Measurement Functions
Network Level
Server Level
Device Level
Control Data
Control Data
Energy Status Information Base (e.g. energy consumption)
Access
Control Data
Operations
Energy Closed Control Loop
Efficiency & Cooling
Cooling Functions
Waste Heat Re-use

Figure 1-1: Migration from separate energy control loops to a coordinated arrangement of multiple DC energy control loops

The next section §1.1 is a repetition of the DOLFIN deliverable D2.2: "DOLFIN requirements and system architecture". It is provided here in order to enable a smooth reading of the deliverable. Moreover, the deliverable is closely related to deliverable D3.1: "Data Centre energy consumption optimization platform (eCOP)". Cross reading may be required in some cases in order to better understand this deliverable.

1.1 Consolidated DOLFIN Software Platform

The DOLFIN system is an ecosystem of collaborative DCs. DOLFIN considers a number of DCs, each one having its own DC customers and links with the energy network, as shown in Figure 1-2.

From the energy network perspective, DOLFIN considers both traditional Energy Providers and Smart Grid Networks in a Smart City scenario. Moreover, DOLFIN considers that each DC may achieve further internal energy efficiency, by recycling and reusing the warm water that is used for cooling the ICT equipment for warming the DC offices (e.g. with an under floor warming system). We consider the DOLFIN System as a complete ecosystem that has a group of customers and an interaction with the energy network. These are not directly connected with a specific DC but use the DOLFIN system as a Virtual Data Centre. The main objective of the DOLFIN solution is to reduce the energy consumption of the elements within the DOLFIN ecosystem as a whole (especially the consumed brown energy) and stabilize the smart grid electricity network, wherever needed, without breaking the agreed SLAs with the end users (or renegotiating the SLAs based on signed contracts).

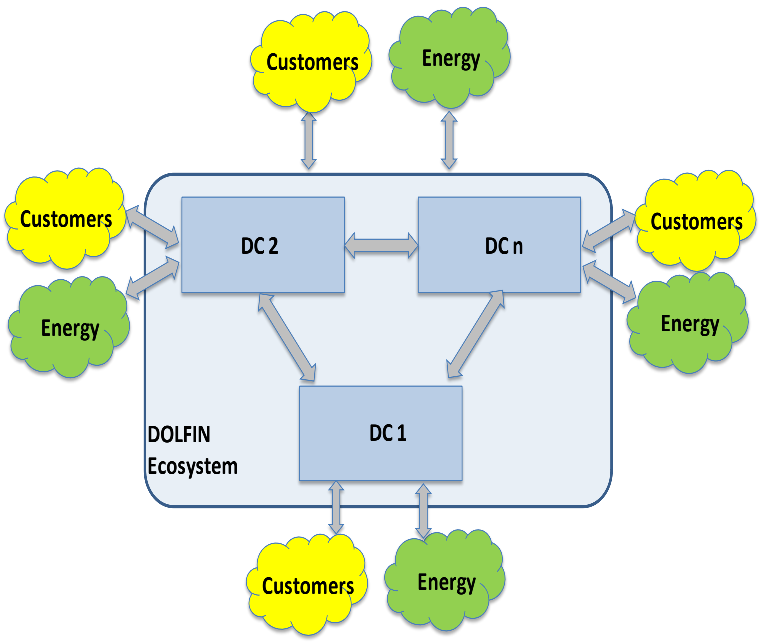


Figure 1-2: DOLFIN Interworking DCs – High Level Architecture

Current and future DCs are complex systems, comprised of diverse cloud management and autonomic functions, Virtual Machines (VM) hypervisors, legacy Energy Management and Uninterrupted Power Supply (UPS) systems, Building Management Systems (BMS) and HVAC (heating, ventilation, and air conditioning) systems. In most cases, DCs have legacy Data Centre Infrastructure Management (DCIM) systems which are quite sophisticated. Therefore, totally replacing this existing infrastructure would be unrealistic. Only new DOLFIN-aware DCs would significantly lower the deployment potential of DOLFIN.

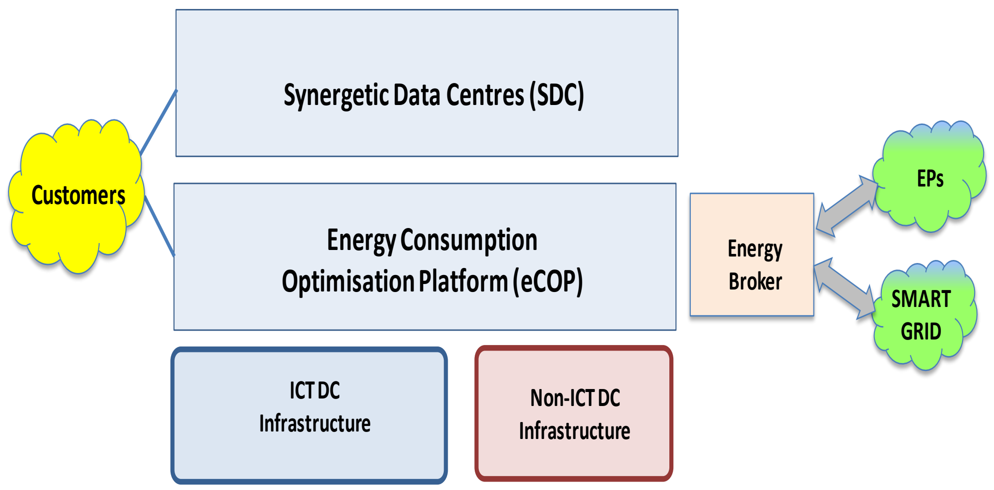

Synergetic Data Centres (SDC)
Customers
Energy Consumption Optimisation Platform (eCOP)
Energy Broker
EPs
SMART GRID
ICT DC Infrastructure
Non-ICT DC Infrastructure

Figure 1-3: DOLFIN Functional Architecture

The DOLFIN functional architecture of each individual DC is defined in terms of new DC functional blocks, revised DC functional blocks, and the interworking interfaces needed to realise different energy closed control loops and their interactions with the normal operation of the DC. The functional architecture is depicted in the Figure 1-3.

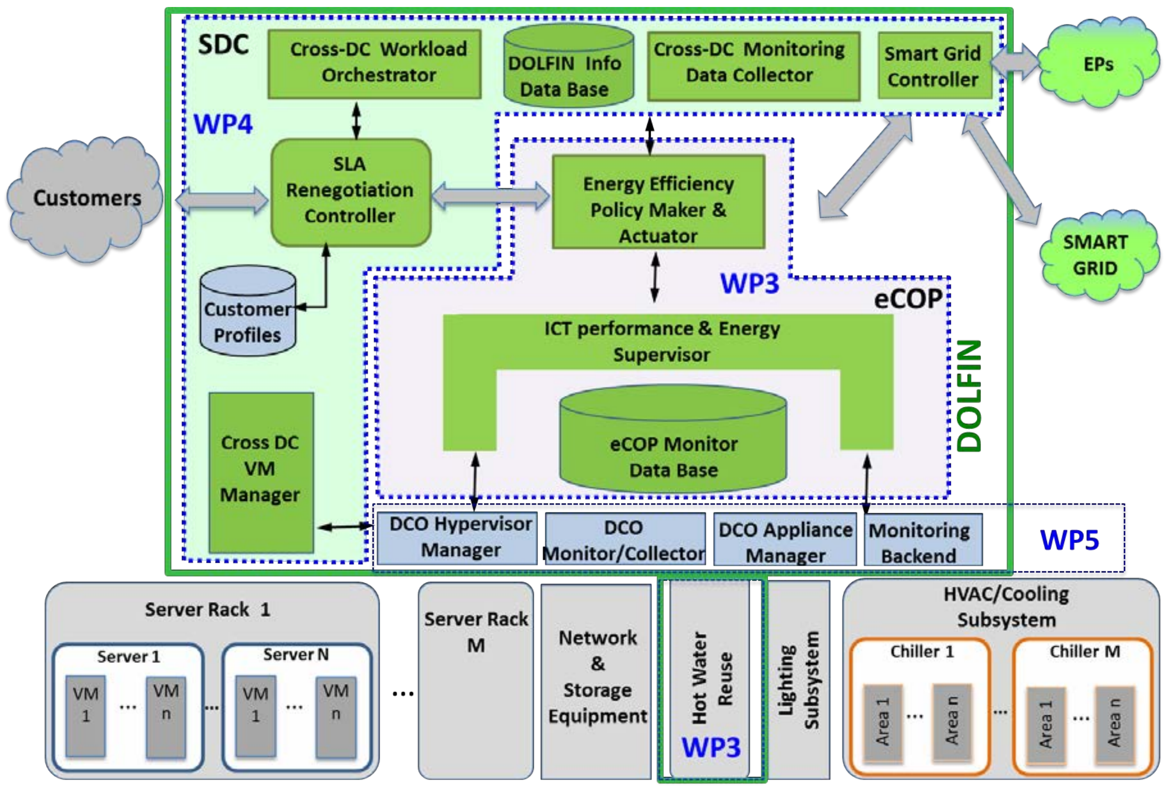


Figure 1-4: Overall DOLFIN architecture

The DOLFIN architecture is represented by a complex echo-system of functional modules that are structured (and grouped) in specialized sub-systems, which are composed by new functional blocks, especially defined in the context of the DOLFIN environment, and revised entities that introduces and adaptation to legacy DC functions needed to realise the energy closed control loop introduced by the DOLFIN energy efficiency approach. The two primary DOLFIN sub-systems are:

- The Energy Consumption Optimisation Platform (eCOP) (development objective for the WP3) represents the core platform in DOLFIN, where are performed the relevant actions to manage and optimize the energy consumption at Data Centre level, through the application of energy benchmarking, dynamic control, and adaptive optimisation of the DC infrastructure. Moreover eCOP includes a subsets of modules that allow continuous monitoring of DC resources (for both ICT and non-ICT entities) and provide specific functionalities for metrics calculation and data collection and storage.
- The energy-conscious Synergetic DCs (SDC) (detailed in WP4 context), which provides a dynamic, service-effective and energy-efficient allocation of demands, across a distributed network of co-operating DCs. In addition the SDC provides the control modules for the integration with the Smart Grid environment and that are responsible for the energy stabilisation through the interconnection with the smart grid network, providing responses on the changing demands for energy.

The overall description of DOLFIN architecture is depicted in Figure 1-4. The figure reports a schematic representation of components, highlights the main relations/interactions between subsystem and external entities (such as the Energy Brokers), the modules position within the DOLFIN architecture and finally how the DOLFIN are globally integrated in the normal DC environment.

A brief introduction of all involved modules is reported in DOLFIN deliverable D2.2. In this section we provide some additional information so that D4.1 may be read as a stand alone document. The DOLFIN architecture entities are listed in two groups that identify the elements specially developed in the context of DOLFIN and those that represent a typical asset in a standard DC environment, but request some adaptation to be used in DOLFIN.

1.1.1 New DC functionality offered by DOLFIN specific components

This section describes a list of components that have been introduced by DOLFIN in the DC environment and provide the following functionality:

- **eCOP Monitor Data Base:** the eCOP Monitor Data Base has the sole purpose of storing all real-time and historical energy related data collected from DCs. This information is used for energy efficiency decisions and VM load predictions, along with data from all the components within a DC architecture.
- **ICT Performance & Energy Supervisor:** focuses on the analysis of performance monitoring data and energy data. The objective of this component is to provide information on the actual performance of the applications (typically VMs) utilizing the resources of the DC devices and the energy consumed by the non-ICT components.
- **Energy Efficiency Policy Maker & Actuator:** this is the most intelligent part of the eCOP that makes the decisions realizing the requested policy of the DC. Following a spiral optimization approach in the context of DOLFIN: at Servers' rack level, at DC segment level, at DC level and finally at DOLFIN level, this component realizes the 3 first levels of the energy optimization, achieving optimization at DC level. Moreover, it aims at initiating a stream of control commands, which can be translated into actual actions by other DOLFIN subsystems within a DC.
- **Cross-DC Monitoring Data Collector:** collects knowledge not only from the Synergetic DC resources, but also from the network routers and in general from the network resources point of view.

- **Cross-DC Workload Orchestrator:** distributed software element that gets the SDC decisions and does most of the different types of resource optimisation including energy optimisation and management of their trade-offs in a cross DC optimization scenario.
- **Cross-DC VM Manager:** realises a DC Interconnect (DCI) interface and performs the actual migration of VMs cross DCs.
- **DOLFIN Information Data Base:** offers an abstracted and logically-centralised information manipulation (including information collection, aggregation/processing, storage / indexing and distribution) across all DOLFIN architectural components. It includes information on the local energy requirements e.g. Demand/Response requests from the local Smart Grid operator
- **SLA Renegotiation Controller:** will take into account the existing approaches in modelling SLA criteria (e.g. the overall cost of an offered service) and augment these approaches with the use of energy related criteria. This module will also handle the negotiation process between the DC and the end-users.

1.1.2 Revised DC functionality offered by components modified by DOLFIN

This section describes a list of components that already exists in a DC environment and they are revised in order to provide the following DOLFIN specific functionality:

- **DCO Hypervisor Manager:** is an adaptation layer that maps the high-level decisions taken by the DOLFIN system into low-level technology-dependent commands that control the DC ICT infrastructure. The DCO Hypervisor Manager analyses the new DCs ICT hardware including DCs configuration and status. It is responsible for managing new ICT hardware and VMs configuration. In addition, it is also responsible for the translation of new hardware configurations to dedicated commands to manage the specific virtualization hypervisor and the execution of commands to provide a new DCs hardware configuration.
- **DCO Appliance Manager:** the counterpart of the DCO Hypervisor Manager, on the non-ICT DC infrastructure. The DCO Appliance Manager adapts and guarantees the execution of strategies elaborated by DOLFIN expert system on the non-ICT infrastructure by translating high level commands into low-level technology-agnostic actions.
- **DCO Monitor/Collector:** DCO Monitor/Collector is to interface with both the ICT and the non-ICT DC infrastructure and collect all operational and energy related information to be stored in the eCOP Data Base.
- **Customer profile management:** provide the information for describing the requirements that need to be satisfied for service provision to the end customers. The Customer profile is strictly related to the service agreement contracted between customer and DC owner and directly affects the actions may be activate on the user resources.

2 DOLFIN Platform Control Flows

To envisage the actual control flow between the eCOP and SDC groups of components and elaborate on the occurring control handovers between the constituent subcomponents, the major possible control flows are presented in this section. In order to conceptualize the functionality and interworking of the two DOLFIN-enabling platforms and the way different events are handled, three different events need to be examined. These events are:

- a request for serving new workload from a set of domestic DC customers;
- a request for serving new workload from a cooperating, synergetic DC;
- a request from the Smart Grid or Prediction Engine[1], mandating the devise of a new DC state optimization plan.

In the following paragraphs, the response of the DOLFIN Platform to these events is presented. Note that this part makes explicit references to components belonging to the eCOP and are, thus, detailed in the deliverable D3.1.

2.1 New workload accommodation request from a set of domestic DC customers

In this case, the new workload request enters the DOLFIN plan by contacting the Policy Maker subcomponent of the Energy Efficiency Policy Makes & Actuator. Initially, the Policy Maker accepts or rejects the load accommodation after consulting the Policy Repository on the set of policies pertaining to the customers making the new workload request and after receiving input from the ICT Performance and Energy Supervisor on the current DC status.

Following the acceptance of the load accommodation (through the instantiation of the appropriate number of VMs), the control is handed over to the Optimizer, residing within the Energy Efficiency Policy Maker & Actuator, to optimize the VM allocation. At the end of the optimization process the control is handed back to the Policy Maker to confirm the applicability of the optimization plan and subsequently the control is handed over to the Policy Actuator. The Policy Actuator then begins issuing commands to the DCO managers, implementing the accepted optimization plan, e.g. instantiating the VMs in the appropriate physical servers.

[1] The Prediction Engine is a functional part of the Energy Efficiency Policy Maker and Actuator component of the eCOP platform. Its scope and role in the Dolfin architecture are detailed in the deliverable D3.1.

As opposed to the above process, the rejection of a workload accommodation request by the Policy Maker is followed by a request to the Cross-DC Workload Orchestrator to relocate the workload to a synergetic DC. As a result, the Cross-DC Workload Orchestrator receives the control and subsequently queries the Cross-DC Workload Orchestrators of the cooperating DOLFIN synergetic DCs in an attempt to find a DC able to host the workload (VMs) in hand. After the acceptance of the workload by a synergetic DC the control is passed to the Cross-DC VM Manager which interacts with the corresponding Cross-DC VM Manager to organize the transfer of the VMs. Last, the Cross-DC VM Manager issues commands to the DCO Hypervisor Manager to interact with the corresponding DCO Hypervisor Manager of the synergetic DC and complete the VM transfer. The above process is depicted in Figure 2-3.

In addition to the aforementioned cases, the possibility that the Policy Maker can neither accept nor reject an incoming VM request, in the borderline case, where the DC has almost reached its maximum capacity. In this case, the Policy Maker hands the control to the Optimizer, which devises an optimization plan of the current VM allocation and based on the outcome of the optimization process, the Optimizer reports to the Policy Maker, whether the workload accommodation request should be accepted or not. This process might introduce a non-ignorable delay to the workload acceptance process, since the optimization plan needs to be performed prior to the acceptance. Therefore, simultaneously to this process, a request is sent to the Cross-DC Workload Orchestrator to investigate the possibility of relocating the Workload to a synergetic DC. As a result a rejection of the workload by the Optimizer can be handled in a timely manner, through the already initiated VM migration.

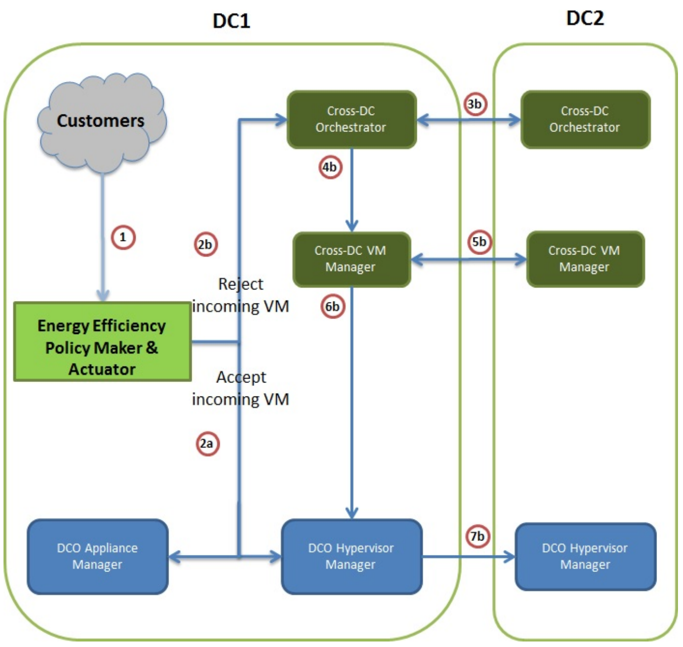


Figure 2-1: Control flow of the DOLFIN Platform response to an intra-DC workload accommodation request

2.2 New workload accommodation request from a synergetic DC

In this case, the new workload accommodation request arrives at the Cross-DC Workload Orchestrator. Subsequently, the request is forwarded to the local Policy Maker, which accepts or rejects the workload following the process detailed in the previous paragraph. If the VM creation is rejected, the Cross-DC Workload Orchestrator informs the respective DC that the VM request has been rejected. However, if the workload is accepted by the Policy Maker, the Cross-DC Workload Orchestrator is informed to accept the load from the synergetic DC and the two Cross-DC VM managers interact to organize the transfer of the workload (VMs). Subsequently, the respective DCO Hypervisor Managers interact to complete the VMs transfer. At the same time, the Policy Maker of the target DC passes the control to the respective Optimizer to optimize the allocation of the VMs, once the Cross-DC Workload Orchestrator is informed. At the end of the optimization process the control is handed back to the Policy Maker to confirm the applicability of the optimization plan and if confirmed the control is passed to the Actuator. The Actuator then begins issuing commands to the DCO Hypervisor Manager, in order to implement the derived optimization plan, once the accepted VMs are transferred from the DCO Hypervisor Manager of the workload source DC. As a result, the time consuming processes of optimization and Cross-DC VM migration run in parallel, allowing the timely service of the synergetic DC request. However, as already stated above in borderline cases of maximum DC utilization, the Policy Maker has to consult the Optimizer on the acceptance of a synergetic VM. This process does not necessarily introduce a delay, since other synergetic DCs can accept the request in a timely manner. The above process is depicted in Figure 2-4.

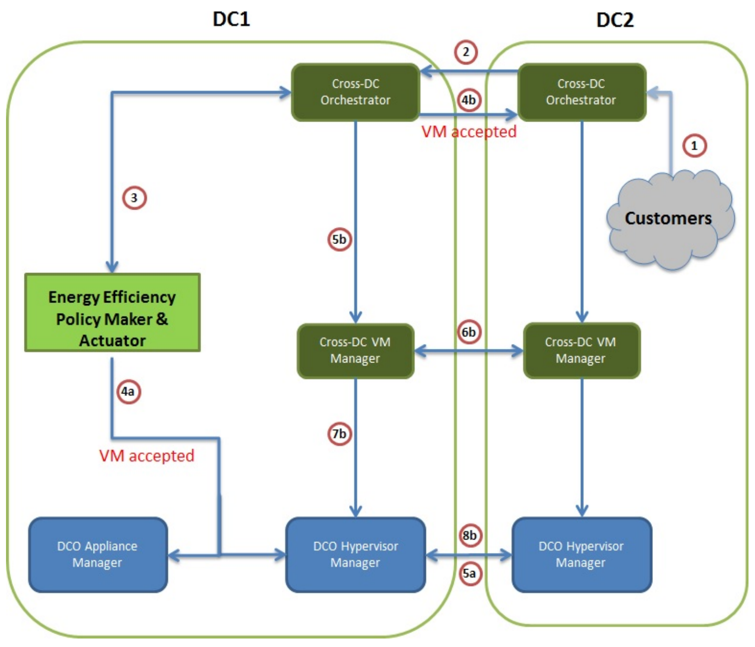


Figure 2-2: Control flow of the DOLFIN Platform response to an inter-DC workload accommodation request

2.3 Explicit DC state change request from the Smart Grid or the local DC Prediction Engine

In case a revision of the current (already optimized) DC operational state is mandated, either by the Prediction Engine or the Smart Grid Controller, the Policy maker evaluates the urgency of the request and subsequently, either rejects the request maintaining the current DC status or decides to proceed with the optimization, based on the policies that govern the current DC operation. If the request for a state change is selected, the control is passed to the Optimizer, which devises a new optimization plan also considering the migration of VMs to synergetic DCs, if necessary. The implementation of the optimization process and the migration of VMs accommodating the possible surplus load to synergetic DCs is performed through the control flow of Figure 2-3.

To envisage the actual control flow between the eCOP and SDC platforms and elaborate on the occurring control handovers between the constituent subcomponents, the major possible control flows are presented hereafter. In order to conceptualize the functionality and interworking of the two DOLFIN-enabling platforms and the way different events are handled, three different events need to be examined. These events are:

- the arrival of a new VM request from a domestic DC customer
- the arrival of a new VM request from a synergetic DC customer
- the arrival of Smart Grid or Prediction Engine event, mandating the devise of a new optimization plan.

2.4 New VM request from a domestic DC customer

In this case, the VM request arrives at the Policy Maker subcomponent of the Energy Efficiency Policy Makes & Actuator. Initially, the Policy Maker accepts or rejects the VM creation after consulting the Policy Repository on the Policy pertaining to the specific VM and after receiving input from the ICT Performance and Energy Supervisor on the current DC status.

Following the acceptance of a VM creation, the control is handed over to the Optimizer, residing within the Energy Efficiency Policy Maker & Actuator, to optimize the VM allocation. At the end of the optimization process the control is handed back to the Policy Maker to confirm the applicability of the optimization plan and subsequently the control is handed over to the Actuator. The Actuator then begins issuing commands to the DCO managers, implementing the accepted optimization plan.

As opposed to the above process, the rejection of a VM request by the Policy Maker is followed by a request to the Cross-DC Orchestrator to relocate the VM to a synergetic DC. As a result, the Cross-DC Orchestrator receives the control and subsequently queries the Cross-DC Orchestrators of synergetic DCs to pinpoint a DC able to host the VM in hand. After the acceptance of the VM by a synergetic DC the control is passed to the Cross-DC VM Manager which interacts with the corresponding Cross-DC VM Manager to organize the transfer of the VM. Last, the Cross-DC VM Manager issues commands to the DCO Hypervisor Manager to interact with the corresponding DCO Hypervisor Manager of the synergetic DC and complete the VM transfer. The above process is depicted in Figure 2-3.

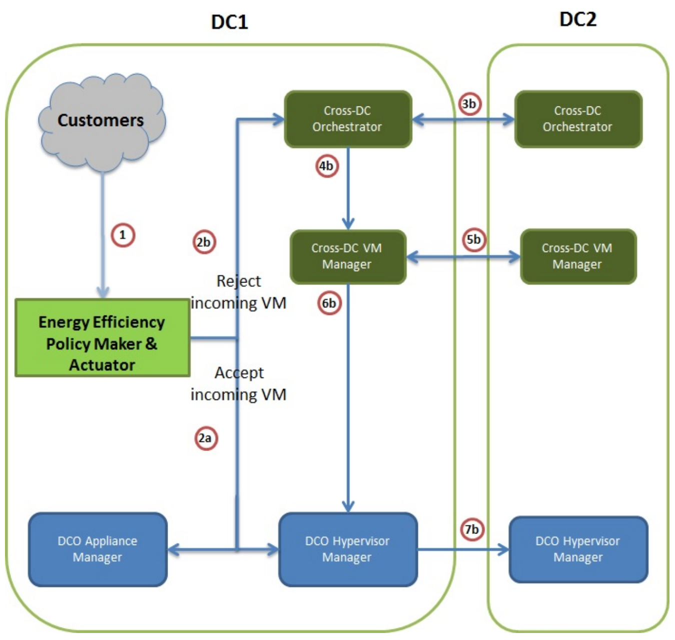


Figure 2-3: Intra-DC VM request; Control Flow

There is however the possibility that the Policy Maker can neither accept nor reject an incoming VM request, in the borderline case, where the DC has almost reached its maximum capacity. In this case, the Policy Maker hands the control to the Optimizer, which devises an optimization plan of the current VM allocation and based on the outcome of the optimization process, the Optimizer reports to the Policy Maker, whether the VM request should be accepted or not. This process might introduce a non-ignorable delay to the VM acceptance process, since the optimization plan needs to be performed prior to the acceptance. Therefore, simultaneously to this process, a request is sent to the Cross-DC Orchestrator to relocate the VM to a synergetic DC. As a result a rejection of the VM by the Optimizer can be handled in a timely manner, through the already initiated VM migration.

2.5 New VM request from a synergetic DC customer

In this case, the VM creation request arrives at the Cross-DC Orchestrator. Subsequently, the request is forwarded to the Policy Maker, which accepts or rejects the VM creation following the process detailed in the previous paragraph. If the VM creation is rejected, the Cross-DC Orchestrator informs the respective DC that the VM request has been rejected. However, if a VM is accepted by the Policy Maker, the Cross-DC Orchestrator is informed to accept the VM from the synergetic DC and the Cross-DC VM managers interact to organize the transfer of the VM. Subsequently, the respective DCO Hypervisor Managers interact to complete the VM transfer. At the same time, the Policy Maker passes the control to the Optimizer to optimize the allocation of the VMs, once the Cross-DC Orchestrator is informed. At the end of the optimization process the control is handed back to the Policy Maker to confirm the applicability of the optimization plan and if confirmed the control is passed to the Actuator.

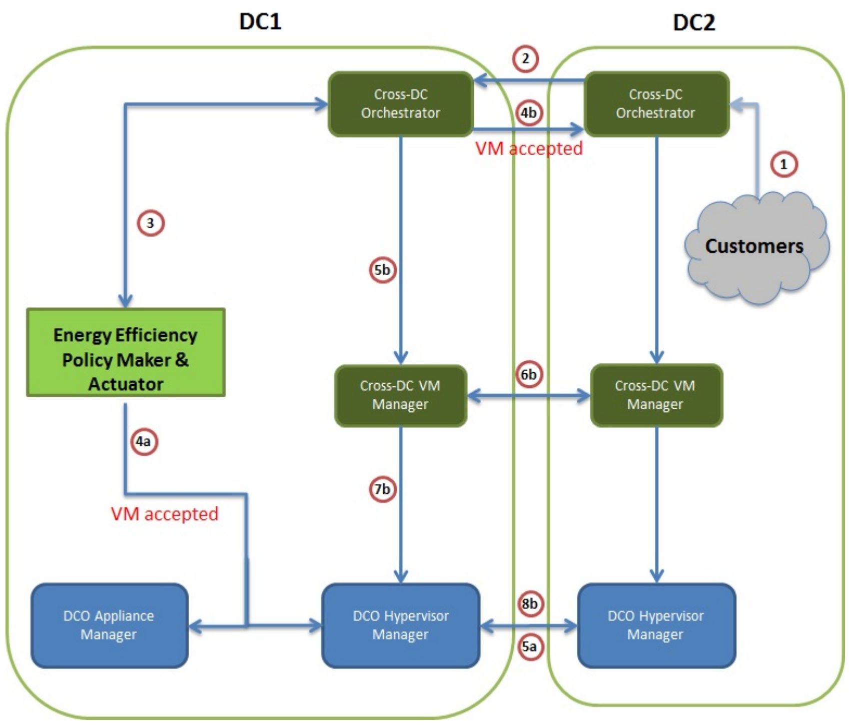


Figure 2-4: Synergetic VM request; Control Flow

The Actuator then begins issuing commands to the DCO Hypervisor Manager, in order to implement the optimized plan, once the accepted VM is transferred from the DCO Hypervisor Manager of the synergetic DC. As a result, the time consuming processes of optimization and Cross-DC VM migration run in parallel, allowing the timely service of the synergetic DC. However, as already stated above in borderline cases of maximum DC utilization the Policy Maker has to consult the Optimizer on the acceptance of a synergetic VM. However, this process does not necessarily introduce a delay, since other synergetic DCs can accept the request in a timely manner. The above process is depicted in Figure 2-4.

2.6 New Smart Grid or Prediction Engine event

In case a revision of the current optimization plan is mandated, either by the Prediction Engine or the Smart Grid Controller, the Policy maker evaluates the urgency of the request and subsequently, either rejects the request maintaining the current DC status or decides to proceed with the optimization. If the latter is selected, the control is passed to the Optimizer, which devises a new optimization plan – allowing also the migration of VMs to synergetic DCs if necessary. The implementation of the optimization process and the migration of VMs accommodating surplus load to synergetic DCs is performed through the control flow of Figure 2-3.

3 Energy Models for DC Software Platforms

3.1 Overview

In DOLFIN, we manage and manipulate Virtual Machines (VMs) in order to optimize the energy consumption of either one DC or a group of DCs. DOLFIN determines any possible energy saving in DCs through careful placement of these VMs. In essence, *VMs and VM characteristics are the basic unit of management decisions related to ICT*. Although energy is consumed by servers and other equipment, it is the management of VMs that affects the outcome of any decisions related to energy usage, as VMs migration between servers, racks and clusters of the same or synergetic DCs and their association SLA play a key role in the DC energy management from the ICT point of view.

In order to make a decision regarding any optimization, we need to determine the current energy consumption within a DC, so that decisions about placement of virtual machines can be made. The current energy consumption is calculated from the resources within the DC, and this includes both the physical resources and the virtual resources. In order to bring about energy savings, the VMs must be placed in servers, racks, clusters and DCs following an optimization strategy. Such an optimization process is a continuous loop of energy monitoring, decision making, and plan actuation.

It is not effective enough to have the power consumption of either a rack or each of the hosts to make these decisions. The energy consumption of each VM is needed. However, there are currently no mechanisms for determining the power consumption of each virtual machine. In reality, it is unlikely that any hardware monitor can be built to determine VM power consumption, and there is currently no support in operating systems to support such functionality. The only solution is to observe the actual dynamics of VMs on a host and evaluate the energy consumption through the use of an energy model.

With respect to monitoring the energy consumption of a host, many machines are not able to return their current energy usage. External devices are available which plug into the mains power supply and can return a data stream of energy consumption values via a digital interface. To build a full energy monitoring solution is relatively expensive in both parts and also the time required to integrate these into a management system. Some DCs do such a process at the level of a rack. However, buying and integrating on a per-host level requests considerably more effort.

As an alternative to the approach of hardware energy monitors, we can attempt to calculate the energy consumption from the data and metrics we do have available, and feed this data into an energy model. If we can make the energy model accurate and representative enough then we can get a very close approximation to actual consumption. Although such a model may not be able to calculate the *exact* energy consumption at any one time, if the variation between the model and the actual consumption is small enough and insignificant for the decision making process, then such a model will be effective and usable.

There is clearly a dilemma whereby we need to manage individual VMs to optimize energy consumption, however there is no way to monitor the energy consumption. To do the job properly we need to know the energy used by each VM, as VMs are the basic unit of management. The approach of using an energy model is a good step forward, and by creating an accurate model for hosts we can extrapolate and so we can use the same modelling approach for VMs as we can for hosts. The main issue is having a model that is accurate enough to determine VM energy consumption.

3.2 Energy Consumption Models

A DC or a group of synergetic DCs will have a number of physical and virtual IT resources, e.g. physical servers, virtual routers, virtual machines, switches, server racks etc. which are consumers of energy. There may be different measurement hardware or energy models that may co-exist in order to quantify the energy consumption of a particular resource. The sum of all of the per-resource energy consumption values is used to calculate the energy used by a DC. Subsets of this data can be used to calculate the energy used by a rack, or by an individual host.

The task of this section is to generalize an energy model, while simplifying some aspects of it, so that an energy model can be utilized by an energy decision engine. As stated, our goal is to define an energy model for hosts, given a set of monitorable and observable attributes, and then once this has been verified as accurate enough, to extrapolate this model to virtual machines.

3.2.1 Definitions

We define as ***Energy Consumers***, all hardware consuming energy and being shared among many resources. Examples being: *CPU, Memory, Network Card, Network Switch,* etc.

The Energy Consumers may be further characterized by ***multiple states***, such as: running, idle, sleep, suspend, transmit, receive, store, retrieve, etc. where each consumer in a state consumes a different amount of energy.

We define as ***Resources*** all logical groupings of resources that co-operate to realize services. These logical groupings depend (and consume their energy) on a number of Energy Consumers. Examples are: *Physical Server, Physical Server Rack, Virtual Machine, Virtual Application etc*. All Resources require all of or a proportion of the energy being consumed by a number of these Energy Consumers.

From the collection of Energy Consumers we need to determine the energy used by each of the resources in a DC from the set of monitorable and observable attributes. First we can calculate the energy used by a host, then by a rack, and then by a whole DC. Secondly we can determine the energy used by a VM. For all of these consumption figures, the values are factored into a set of KPIs for energy consumption that inform us of the overall energy picture.

The KPIs for energy consumption of both the physical, and more importantly the virtual resources are utilized to make management decisions. These management decisions may reflect trade-offs that can be fine-tuned from a management facility to save energy, i.e., trade processing cost for communication, trade memory for processing, trade memory for communication etc… in the process of determining the placement of a VM.

3.2.2 Proposed KPIs for Energy Consumption

The ***Key Performance Indicators (KPIs)*** use the energy consumers plus various coefficients that can be adapted to the hardware used, i.e., whether server processors, network cards and memories are energy efficient or not, or whether they support the multiple states. Although the energy savings are actually hardware-related, the monitoring, the modelling, and the decisions are virtual machine related.

Some of the proposed KPI functions are:

$$\begin{aligned}\text{Physical Server Energy Consumption} = {} & a + b * server\ processing\ load \\ & + c * server\ memory\ utilization \\ & + d * server\ network\ traffic\end{aligned}$$

$$\textbf{Average VM Energy Consumption} = \frac{\boldsymbol{Physical\ Server\ Energy\ Consumption - a}}{\boldsymbol{Number\ of\ VMs\_Hosted\_In\_this\_Server}}$$

$$\begin{aligned}\text{VM Energy Consumption} = {} & b * VM\ processing\ Load \\ & + c * VM\ memory\ consumption \\ & + d * VM\ network\ traffic\end{aligned}$$

$$\textbf{Average Application Energy Consumption} = \frac{\boldsymbol{VM\ Energy\ Consumption}}{\boldsymbol{Number\_of\_applications\_running\_in\_this\_VM}}$$

$$\begin{aligned}\text{Application Energy Consumption} = {} & b * Application\ processing\ load \\ & + c * Application\ memory\ consumption \\ & + d * Application\ network\ traffic\end{aligned}$$

Of course, many more KPIs can be derived for these resources, and we expect to see more over time as more knowledge is acquired w.r.t modelling of virtual resource energy consumption.

3.2.3 Coefficients

The coefficients used in the above KPIs are highlighted here. They represent multipliers of various consumptions. In the following a, b, c, d are hardware related:

a = baseline energy consumption (consumption of server with 0 virtual machines).

b = energy consumption per percentage of cpu load

c = consumption per byte of memory used

d = consumption per byte communicated

When using the KPIs to determine energy usage, we expect to adjust the values to the underlying real hardware.

3.3 Evaluations

As stated, the values for the coefficients of the KPIs need to be adjusted to reflect the actual energy consumption of the real hardware being used. Ideally the function for each KPI and the determined coefficients will produce values for energy consumption which would match the real energy consumption very closely. We expect that a DC will measure the energy consumed by the real resources and then create the coefficients for the energy model functions.

As the mapping from real hardware to the energy model is in its early stages, and due to there being little documentation or literature on energy consumption in the face of dynamic behaviour of real resources, we expect the creation of these coefficients to be a trial-and-error process in the near future.

For our initial experiments we start by being simple. Although we may have a simple starting point, we can define a path towards being more accurate (but more complicated) model as new knowledge is derived.

In order to become more accurate we can embed resource-dependent energy models or real monitoring probes (e.g., regarding CPU usage, memory, network card etc) whenever possible. Along these lines we devised a KPI which factored in two CPU states, namely busy and idle, and two memory states, namely allocated and free, and also factored in network traffic that was incoming and outgoing.

So taking our KPI for Physical Server Energy Consumption, and extending it to include the extra states, we define this new Physical Server Energy Consumption KPI as:

$$\begin{aligned} &a + b1 * server\_processing\_load + b2 * server\_cpu\_idle\ + \\ &c1 * server\_memory\_utilization + c2 * server\_memory\_free + \\ &d1 * server\_network\_traffic + d2 * server\_network\_traffic \end{aligned}$$

This KPI takes into account more of the states of the host.

3.4 Preliminary Results

As an initial starting point we collected some online server power usage figures [2][3][4]. As servers can range from a basic 1U single core device, up to 8U multi-core, large memory device, getting global encompassing figure is not possible. However, we estimated some baseline values that we can evaluate against. In this simple case of a basic 1U server we defined:

30 Watts - baseline energy per Host
10 Watts - average energy per virtual machine
5 Watts - average per application running on each VM

All of these values can be used with the hierarchy of resources in order to determine the energy values per server, per rack and per DC.As an example we can calculate that a host with 3 virtual machines will consume 60 Watts (namely 30 W + 3 * 10 W).The 5 Watts, per application, is included within the per virtual machine figure, but can be useful separately from other management decisions.

[2]http://www.zdnet.com/article/toolkit-calculate-datacenter-server-power-usage/
[3]http://www.infoq.com/articles/power-consumption-servers
[4]http://www.kingston.com/en/business/server_solutions/power_benchmark

To calculate the energy usage of a rack we determined that a common size of rack hardware is of size 42U.This can theoretically hold 42 of 1U machines, but it is common to only fit 36 machines into a rack with some room remaining for the required networking hardware that connects the machines.[5]Therefore, on average, a rack will consume 2160 Watts (60W * 36).

For a whole Data Center we need the number of racks in that Data Center. Below is a table of documented information for the number of racks in some Amazon DCs. By multiplying the nominal rack consumption by the number of racks we will see the amount of power.

data center size	# of server racks	power (watts)
US East (Virginia)	5030	10.864.800
US West (Oregon)	41	88.560
US West (N. California)	630	1.360.800
EU West (Ireland)	814	1.758.240
AP Northeast (Japan)	314	678.240
AP Southeast (Singapore)	246	531.360
SA East (Sao Paulo)	25	54.000
Total	**7.100**	**15.336.000**

In order to evaluate the energy model itself we have done some runs on a testbed and we get some values for energy consumption. We collect CPU, memory, and network monitoring values for the host that runs the UCL VLSP testbed platform (see Annex 2). These monitoring values are then pushed into the Energy Model to calculate the number of Watts used.

By taking the new Physical Server Energy Consumption KPI, and setting the following coefficients, we setup the energy model. We set the baseline energy consumption to be the same as the initial estimates of 30 Watts. For the CPU states we set a coefficient of 50 when the CPU is loaded and a coefficient of 0.2 for when the CPU is idle. For the memory states we set a coefficient of 0.04 for memory that is allocated (and we assume is utilized) and a coefficient of 0.02 for memory this is free (and not utilized). Finally, we set a coefficient of 0.0005 for networking traffic, both inbound and outbound. (We assumed that there is little power difference for each one in a server environment, but on a mobile device this assumption may not hold).

```
"energyFactors": {
   "baseLineEnergyConsumption": 30,
   "cpuIdleCoefficient": 0.2,
   "cpuLoadCoefficient": 50,
   "freeMemoryCoefficient": 0.02,
   "memoryAllocationCoefficient": 0.04,
   "networkIncomingBytesCoefficient": 0.00005,
   "networkOutboundBytesCoefficient": 0.00005
}
```

On a host that runs the VLSP testbed platform we setup an experiment that included 3 virtual machines running small applications. We then gathered data from the monitoring system, and fed this data into the

[5]https://huanliu.wordpress.com/2012/03/13/amazon-data-center-size/

energy model. The execution was undertaken for a 1 hour period. In the figure below we can see the collected data: CPU in blue, memory in yellow, and network in red. The derived energy data is presented in green.

From this initial experiment, with a baseline value of 30 Watts, we can see in the following figure that the power usage is in the region of 60 Watts (as in our initial estimated), but the average over time is closer to 40 Watts.

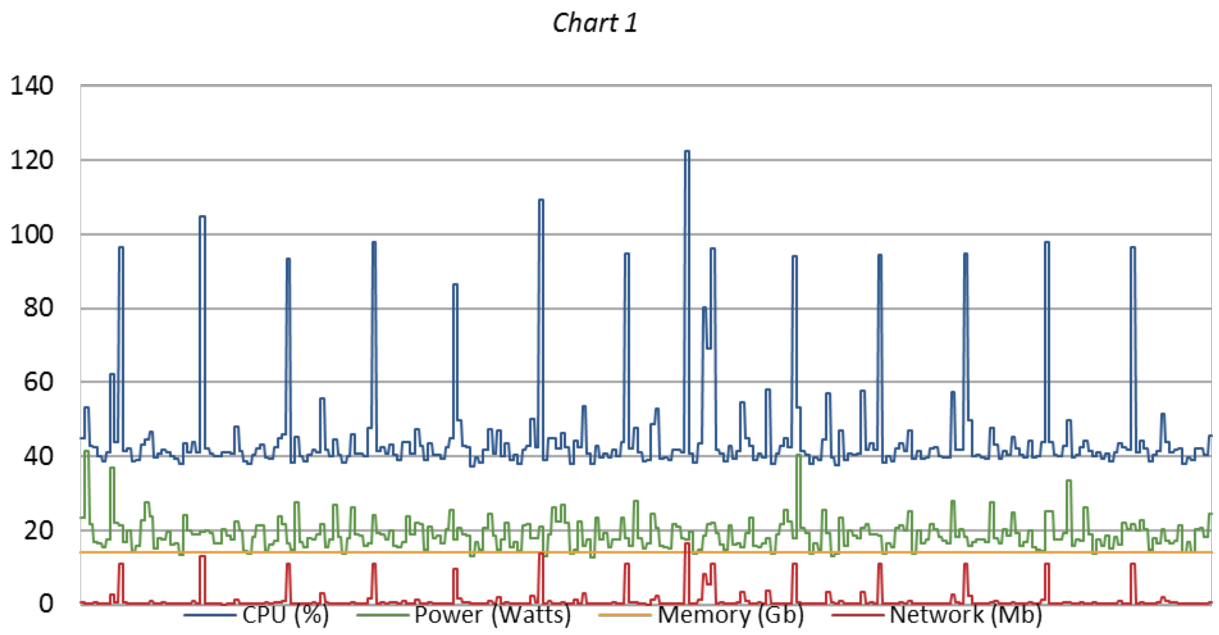


Figure 3-1: Power consumption during the indicative experiment presenting the energy models to be used by DOLFIN.

4 Synergetic Data Centres (SDC) Functionality

4.1 SDC Modules functional description

The SDC (Synergetic DCs) part of the DOLFIN architecture provides a dynamic and energy-efficient allocation of workload demands, across a distributed network of co-operating DCs. The SDC provides the necessary functionality to enable the DC to participate along with other DCs in the framework of a DOLFIN-enabled energy efficient DC cluster. Whilst the eCOP contains the core optimization features and facilities for each Data Centre, the SDC has the following functionality:

- It responds to requests from the eCOP for the remote placement of VMs,
- It interacts with other Data Centres in order to orchestrate the placement of VMs,
- It interacts with other Data Centres so as to actually perform the VM relocation,
- It collects monitoring data from remote DCs regarding VMs that have been placed remotely,
- It performs the storage and manipulation of information required to support the cross Data Centre operations, gathering data from the local DC and some data from remote DCs,
- It monitors, processes, and maintains SLAs for customer services based on monitored key performance parameters,
- It integrates with the Smart Grid environment that is responsible for the energy stabilisation through the inter-connection with a smart grid network, informing the eCOP of the changing demands for energy.

The following table provides an overview of the functions of each component of the SDC.

Table 4-1: Tabulated overview of the SDC components functionality

Module	Function
Cross-DC Monitoring Data Collector	It collects knowledge not only from the Synergetic Data Centre resources, but also from the network routers and in general from the network resources point of view. This is an important prerequisite to achieve energy efficiency across DCs.
Cross-DC Workload Orchestrator	It is a distributed software element that gets the SDC decisions and does most of the different types of resource optimisation including energy optimisation and management of their trade-offs in a cross DC optimization scenario. It 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-DCVM Manager	It realises a DC Interconnect (DCI) interface and performs the actual migration of VMs cross DCs. It will typically apply a set of standard alternatives for coping with high/peak workloads, more precisely with allocation of VMs, data, services and tasks. The Cross DC Monitoring Collector, Workload Orchestrator and VM Manager collaborate closely in order to offer a flexible, reliable, and fast communications solution to handle the increased network traffic arising from the DCs operation.
DOLFIN Information Data Base	It offers abstracted and logically-centralised information manipulation (including information collection, aggregation/processing, storage/indexing and distribution) across all DOLFIN architectural components. Also, it includes information on the local energy requirements e.g. Demand/Response requests from the local Smart Grid operator. The Information Base uses two separate interfaces for communication with the DOLFIN software entities: (i) the Information Management Interface is used for information manipulation configuration, including the information sources/sinks registration to the Information Base, the management of internal Information Base, information manipulation functions and the establishment, operation and optimisation of information flows; and (ii) the Information Exchange Interface that offers the actual information exchange capability to the DOLFIN components. These two interfaces allow a unified yet abstracted information handling for all types of participating entities. The DOLFIN expert system needs to have the topology data of infrastructures in DCs, the electricity consumption, and infrastructure performance data of hardware infrastructures of the DCs constantly available in order to manage systems in the DCs in a smart optimized, and automated way and as much as possible in "real time". This is necessary to achieve energy savings and the resulting C02 emission reductions and to meet the different demands from EPs in terms of optimization or different distribution of the power consumption of the smart grids. This info is also stored in the DOLFIN Energy Information Base.
SLA Renegotiation Controller	The SLA Renegotiation Controller guarantees to steer the DC operation towards environmentally sound behaviour. This module will take into account the existing approaches in modelling SLA criteria and augment these approaches with the use of energy related criteria. This module will also handle the negotiation process between the DC and the end-users, allowing the DC to request from an end-user to decrease the performance metrics of their SLA, in exchange of lower service costs. This component will also consider the SLA negotiation process between co-operating DCs for the SLA that govern the services that are being transferred between them.
Customer profile management	This component is meant to be in close cooperation with the SLA Renegotiation Controller, enabling the full functionality of the latter. Specifically, it acts as a persistence and management element managing the customer profiles that are characterized by a number of SLAs (either current or past) enriched with aggregated information related to costs, actual performance, QoS, security and environmental friendliness.
Smart Grid Controller	The Smart Grid Controller interconnects the DC with the Smart city electricity network. This most critical part of the system acts as a broker between the core DOLFIN infrastructure and subsystems, represented by the Energy Efficiency Policy Maker & Actuator, and the outside world, namely the Smart Grid (SG) and the Energy Providers (EPs).

In the following paragraphs, a detailed description of SDC modules and their interaction with other modules (outside of the SDC scope) is given.

4.2 SDC Modules interfaces

The elements of the SDC create a distributed infrastructure that interacts with and is reliant on the elements of the eCOP system within each Data Centre. The energy-conscious SDC provide a dynamic, service-effective and energy-efficient allocation of load demands, across a distributed network of co-operating DCs. The three main elements of the SDC within a DC are:

1. The Cross DC Workload Orchestrator,
2. The Cross DC Monitoring Data Collector, and
3. The Cross DC VM Manager

These three elements interact with identical components within each Data Centre, finally forming sets of distributed systems allowing the various DOLFIN-enabled DCs to seamlessly interoperate.

In addition the, SDC provides the Smart Grid Controller for the integration with the Smart Grid environment, being responsible for assisting the Grid towards achieving energy stabilisation, providing responses on the changing demands for energy. This element is not part of the distributed infrastructure as it is per-DC, and interacts with the Smart Grid only.

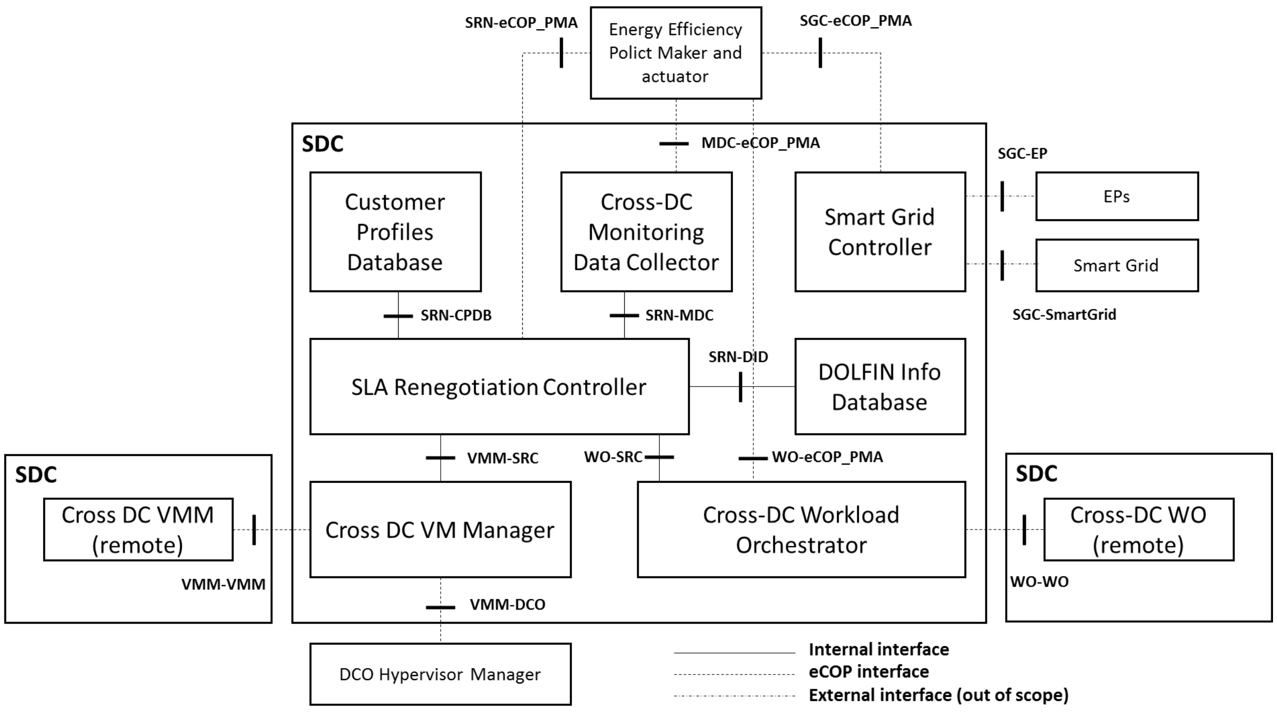


Figure 4-1: SDC modules and interfaces

Figure 4-1 offers a high-level view of cross DC interactions, showing the distributed interaction of the Cross-DC elements. Of particular note is the Cross DC Workload Orchestrator that is itself a fully distributed system. Any local Energy Actuator can interact with its local Cross DC Orchestrator in order to determine if another DC is able to extra VM load from another DC. The set of Cross DC Orchestrators interact amongst themselves in order to determine if a VM, or set of VMs, can be placed in a different DC. The Energy Actuator does not need to interact with other DCs itself.

Furthermore, the Cross DC VM Managers in the various DCs interact in order to negotiate the high level details of a VM migration process. Moreover, the Cross DC VM Managers interact with their local Hypervisor Managers in order to undertake the actual migration. Both the Hypervisor Managers and the migration process are existing technology that the SDC relies on, however the Cross DC VM Managers provide the lifecycle management and negotiation for setting up the migration process and also to provide the status details to other SDC components once the migration has been effected.

When a VM is relocated to another DC, the Cross DC Monitoring Data Collector of the remote DC will collect details for the VM and transmit those back to the originating DC, via its Cross DC Monitoring Data Collector. Such details will include live VM info that can be used for billing purposes. Referring to Figure 4-1, the following table describes the role of each interface. Interface details and APIs are specified in the Annex 2 where applicable.

Table 4-2: Tabulated overview of the basic interfaces of the SDC modules

Interface tag	Right module	Left Module	Direction	Role and description
VMM-VMM	Cross-DC VMM	Cross-DC VMM (other DC)	<->	This interface is engaged at VM migration phase between two separate DCs. The two modules Exchange the necessary information to evaluate the migration feasibility and issue the actual migration via the VMM-DCO interface
VMM-DCO	Cross-DC VMM	DCO Hypervisor Mgr (eCop)	<->	This interface is engaged once the feasibility of the VMM movement is ensured at the VMM level. The interface is used to issue the actual VM movement commands to the hypervisors
WO-eCOP_PMA	Cross-DC WO	eCOP - Policy Maker and Actuator	<->	This interface is engaged when a cross-DC VM relocation is needed. The Policy Maker asks the Cross DC Workload Orchestrator to request a VM transfer to an external DC or the Cross DC Workload Orchestrator receives a VM relocation request through the remote Cross DC Workload Orchestrator and queries the Policy Maker in order to proceed with the relocation or not.

4.3 Internal modules design

4.3.1 Cross-DC Workload Orchestrator

The Cross DC Workload Orchestration is a software element which does most of the different types of resource optimisation including energy optimisation and management of their trade-offs at cross DC and network level. It 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. Moreover, it closely collaborates with the Monitoring Data Collector, which collects and manages all of the monitoring data received from the DC to DC network, including virtual and physical resources.

It is also responsible of coordinating the VM migration process between federated DCs.

4.3.1.1 Module detailed description

The Cross-DC Workload Orchestrator is triggered by the local Policy Maker, which asks the Cross-DC Orchestrator if it can place a certain Virtual Machine (VM) with some specified attributes. Such a process is undertaken only if the Energy Actuator is unable to place a VM on the local DC. The Cross-DC Workload Orchestrator acts as a fully distributed sub-system, whereby all of the Cross-DC Orchestrators in all of the DCs in a group can interact with one another. They interact in order to determine if a VM can potentially be moved.

Figure 4-2 shows the interactions that happen whenever the Cross DC Workload Orchestrator is triggered:

1. The local Policy Maker starts the process to request a migration to a federated DC invoking the Cross DC Workload Orchestrator
2. The Cross DC Workload Orchestrator interact with the other Cross DC Workload Orchestrators, located in different federated DC, in order to determine if a VM can potentially be moved there.
3. Each invoked Cross DC Workload Orchestrator forward the request to its local Policy Maker
4. The Policy Maker accepts or rejects the request
5. The invoked Cross DC Workload Orchestrator forwards the answer to the Cross DC Workload Orchestrator which initiated the process
6. When a federated DC willing to take the VM is found, the local Cross DC Virtual Machine Manager is instructed to perform the migration.

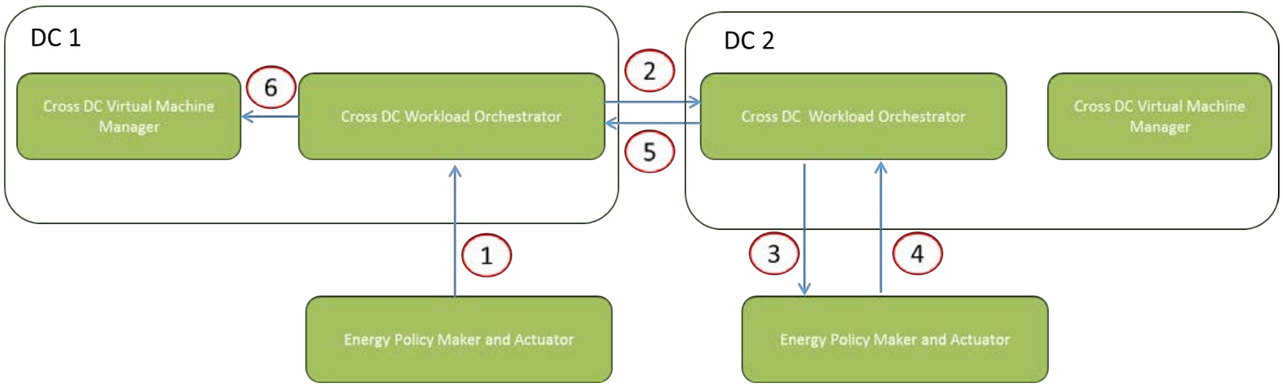

DC 1
DC 2
Cross DC Virtual Machine Manager
Cross DC Workload Orchestrator
Cross DC Workload Orchestrator
Cross DC Virtual Machine Manager
Energy Policy Maker and Actuator
Energy Policy Maker and Actuator
1
2
3
4
5
6

Figure 4-2: Cross DC Workload Orchestrator module in the VM migration phase

The Cross DC Workload Orchestration component is also responsible for the placement of the virtual routers, the allocation of service components and functions running on the routers as applications. In particular, it is responsible for instantiating the virtual routers as it is the software element which does most of the management and orchestration and is in charge of managing the full lifecycle of the virtual routers.

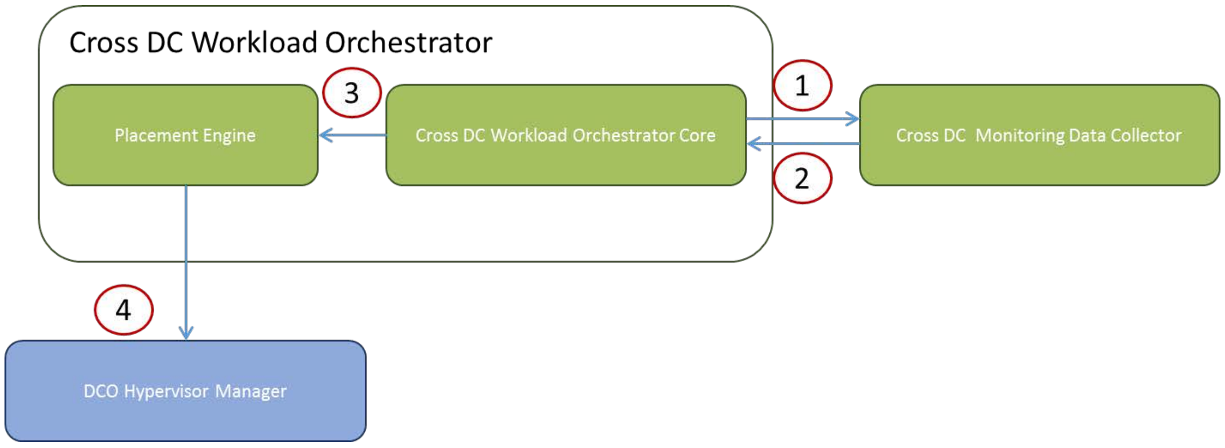

Cross DC Workload Orchestrator
Placement Engine
Cross DC Workload Orchestrator Core
Cross DC Monitoring Data Collector
DCO Hypervisor Manager
1
2
3
4

Figure 4-3: Cross DC Workload Orchestrator module in the network management phase

As shown in Figure 4-3, the Cross DC Workload Orchestrator periodically polls the Cross DC Monitoring Data Collector to get the network assets status and metrics. If the need arises to increase the number of virtual routers, the Placement Engine will decide where to locate them, and will communicate the decision to the DCO Hypervisor Manager, which will translate the decision to commands executed by the local Hypervisors.

4.3.1.2 Module decomposition

4.3.1.2.1 Cross DC Workload Orchestrator Core

The Cross DC Workload Orchestrator Core is the main component of the Cross DC Workload Orchestrator. It acts as a control point for the platform by sending out the command and acts as a management element for the platform by collecting monitoring data and enabling reactive behaviour. It also provides the placement mechanism with all necessary information to perform the steps needed to prepare the DC in order to be able to accommodate load from other DCs.

4.3.1.3 *Cross-DC VM Movement workflow*

This section provides an indicative scenario where two DCs, say DC1 and DC2, interact in order to perform a relocation of a VM (representing a general workload). Figure 4-4 depicts the relevant interaction steps. We assume that there is an ***Energy Actuator***, as an abstract initiator of the activity such as the smart grid controller or the policy maker.

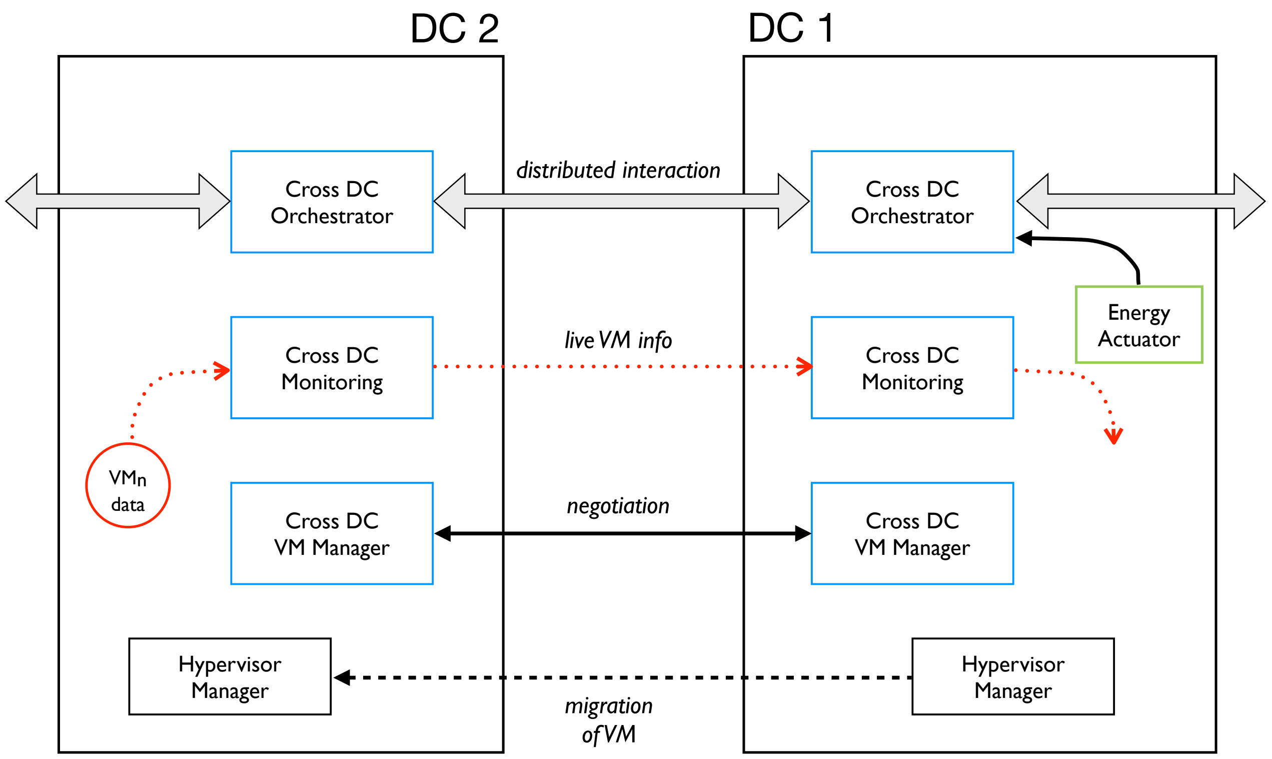


Figure 4-4: High-level view of the cross-DC workload relocation interactions

Suppose that DC1 needs to transfer a VM, and DC2 will become the destination, after some negotiation. The process is broken down into 15 steps, which are described in the following.

1. The Energy Actuator asks the Cross-DC Orchestrator if it can place a certain Virtual Machine (VM) with some specified attributes. Such a process is undertaken only if the Energy Actuator is unable to place a VM on the local DC (DC1 in this case) in which case it triggers the Cross-DC Orchestrator.

2. The Cross DC Orchestrator of DC1 will interact with Cross DC Orchestrators of the DOLFIN-enabled Synergetic DCs. It will make a request to place or relocate a VM with the same specified VM attributes that the Energy Actuator provided. In this instance the Cross DC Orchestrator of DC2 is messaged.

3. The Cross DC Orchestrator of DC2 provides the response to the Cross DC Orchestrator of DC1. The response can be one of two outcomes:

    a. NO. DC2 is unable to accept the VM. i.e. placement is not possible on DC2.

    b. YES. DC2 can accept the VM. i.e. placement is possible on DC2.

4. The Cross DC Orchestrator of DC1 will inform the Energy Actuator that made the original request with the outcome. This will result in one of two situations:
    a. On NO, the Energy Actuator is unable to relocate a VM so it will have to leave the VM at DC1. No further steps with other DCs are taken at this time.
    b. On YES, the Energy Actuator will initiate the necessary actions to export the VM, and will move onto step 5.
5. The Policy Maker will trigger a request to the Cross DC Orchestrator of DC1 to initiate a cross DC VM transfer.
6. The Cross DC Orchestrator of DC1 will interact with the Cross DC VM Manager of DC1 to tell it that the VM needs to be placed in DC2.
7. The Cross DC VM Manager of DC1 will tell the Hypervisor Manager of DC1 to start preparing the VM for migration.
8. The Hypervisor Manager notifies the Cross DC VM Manager.
9. The Cross DC VM Manager of DC1 will then interact with the Cross DC VM Manager of DC2 in order to manage the lifecycle management and negotiation for setting up the migration process.
10. Following this interaction, the Cross DC VM Manager of DC2 will inform the Hypervisor Manager at DC2 to prepare for the migration of the VM from DC1.
11. At this stage the high level negotiation is complete, and both Hypervisor Managers interact directly in order to do the actual VM migration.
12. Once the VM migration is complete, the Hypervisor Manager of DC2 will inform its Cross DC VM Manager that the VM has been transferred.
13. The Cross DC VM Manager of DC2 will then respond to the Cross DC VM Manager of DC1 verifying the transfer of the VM from DC1 to DC2.
14. The Cross DC VM Manager of DC1 then verifies the transfer to the Cross DC Orchestrator of DC1.
15. Finally, the Cross DC Orchestrator of DC1 informs the Energy Actuator at DC1 that the process is complete.

The aforementioned steps present the full round trip of requests and responses using the SDC components to organise a migration of a VM from one DC to another DC. The following figure depicts this interaction.

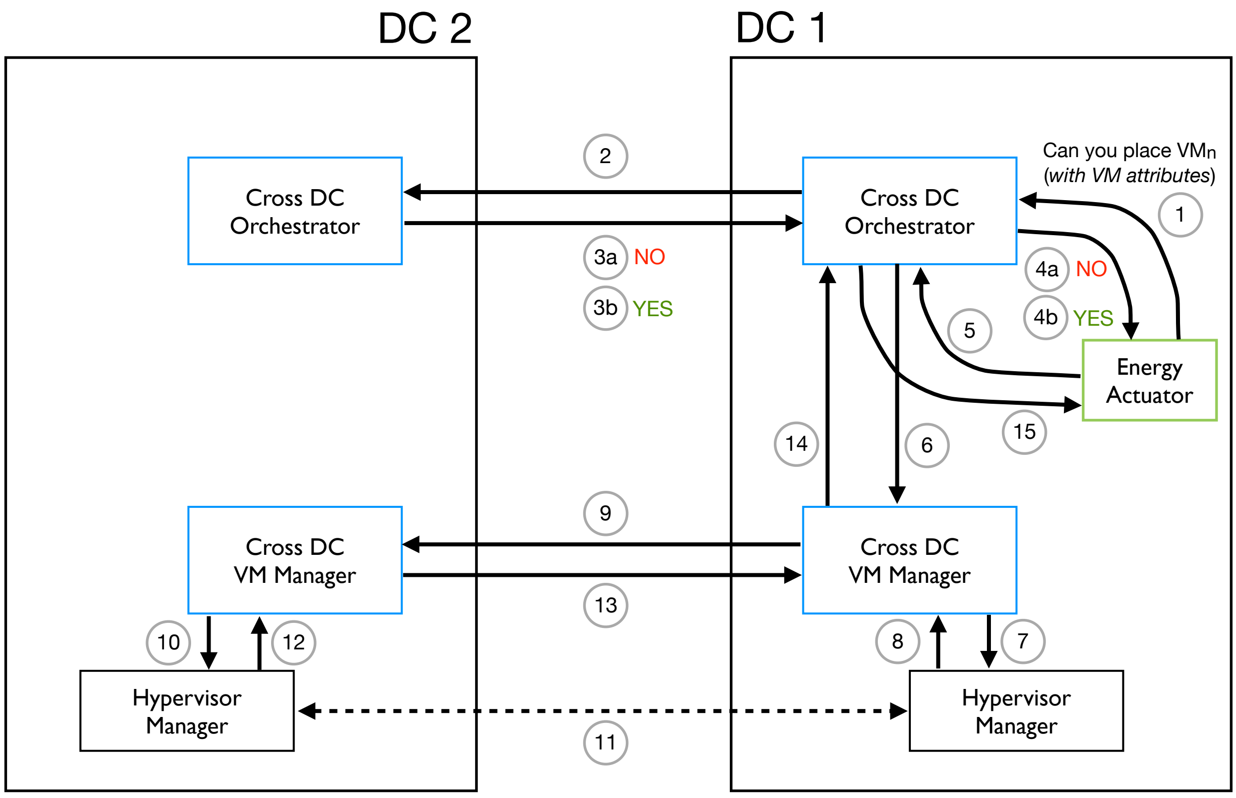


Figure 4-5: Detailed view of the cross-DC VM relocation interactions

An alternate view of the interaction, as a sequence diagram, showing the same set of requests and responses is provided in Figure 4-6.

4.3.1.4 References to requirements

This section reports the list of requirements satisfied by this module. The complete list of identified requirements is described in the deliverable D2.2 (D2.2 - DOLFIN requirements and system architecture).

Table 4-3: References to requirements (Cross-DC Workload Orchestrator)

Requirement ID & Name	**Requirement description**
Q13DCManagement	The system shall include a management entity for orchestrating the whole process.
Q14 DC2DCInterface	The system shall be able to interface with other federated DCs.
Q56 DC VM Handling (VM params)	In accordance with Ask for VM Params Adjustment" this requirement implies that the Workload and VM Manager should be able to fulfil a request for changing the parameters of a certain set of VMs".

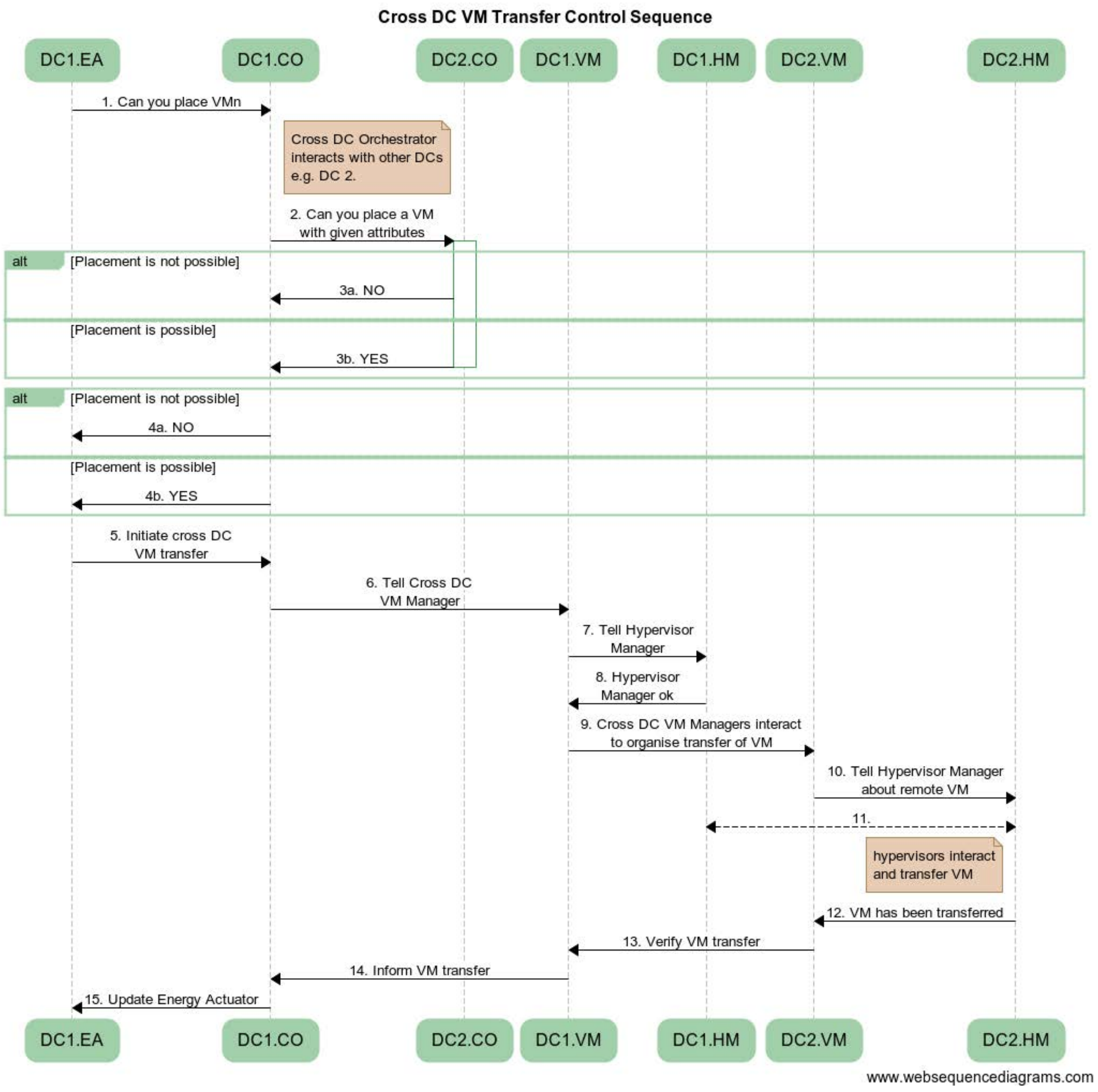

Cross DC VM Transfer Control Sequence
DC1.EA
DC1.CO
DC2.CO
DC1.VM
DC1.HM
DC2.VM
DC2.HM
1. Can you place VMn
Cross DC Orchestrator interacts with other DCs e.g. DC 2.
2. Can you place a VM with given attributes
alt
[Placement is not possible]
3a. NO
[Placement is possible]
3b. YES
alt
[Placement is not possible]
4a. NO
[Placement is possible]
4b. YES
5. Initiate cross DC VM transfer
6. Tell Cross DC VM Manager
7. Tell Hypervisor Manager
8. Hypervisor Manager ok
9. Cross DC VM Managers interact to organise transfer of VM
10. Tell Hypervisor Manager about remote VM
11.
hypervisors interact and transfer VM
12. VM has been transferred
13. Verify VM transfer
14. Inform VM transfer
15. Update Energy Actuator
www.websequencediagrams.com

Figure 4-6: Detailed sequential view of the cross-DC VM relocation interactions

4.3.2 Cross-DC VM Manager

The Cross-DC VM Manager (CDC VMM) is responsible for coordination with the local DC components on each site in order to allocate, migrate and de-allocate VMs to actual computing, storage or network resources. These resources are exposed from the DCO Hypervisor directly, but are managed in a higher level by the ICT and EPM&A DOLFIN components that reside in each DC. The component uses a RESTful API to interact with the various components of the architecture, whose calls are defined later in this document. It receives directives from the Cross-DC Workload Orchestrator, polls the DOLFIN Information DataBase for data and reports to the Cross-DC Monitoring Data Collector. It also coordinates with the DCO Hypervisor Manager and if necessary with other components of the DC. Thus, part of the API has to interface with the respective DCO Manager, and will therefore be DC specific.

4.3.2.1 Module detailed description

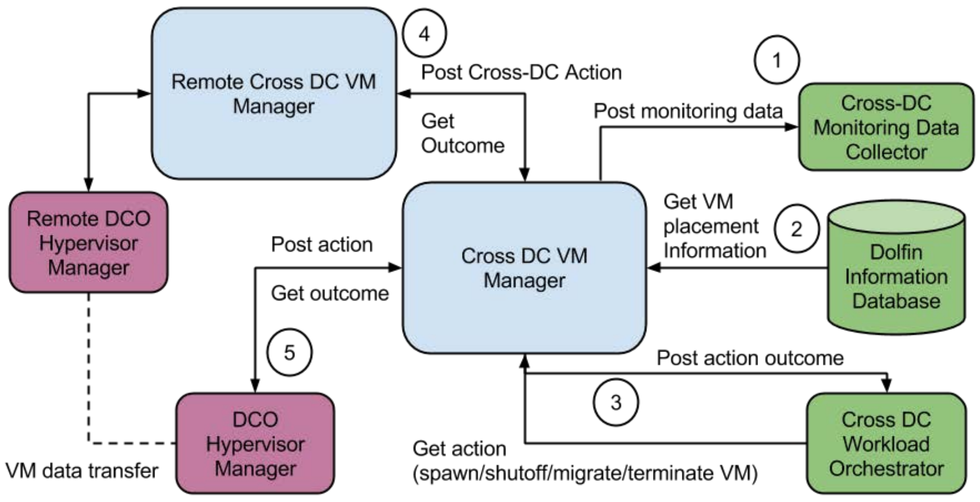

Remote Cross DC VM Manager
4
Post Cross-DC Action
Get Outcome
1
Post monitoring data
Cross-DC Monitoring Data Collector
Remote DCO Hypervisor Manager
Post action
Get outcome
Cross DC VM Manager
Get VM placement Information
2
Dolfin Information Database
5
Post action outcome
3
DCO Hypervisor Manager
VM data transfer
Get action (spawn/shutoff/migrate/terminate VM)
Cross DC Workload Orchestrator

Figure 4-7: Overview of CDC VM Manager's interaction with other DOLFIN modules.

The CDC VMM's main responsibility is to manage a VM's lifecycle across DOLFIN-enabled DCs. Although it is primarily responsible for delegating such actions to the relevant DC on behalf of DOLFIN, it also has to coordinate complicated actions (such as cross-DC migration of VMs) with other CDC VMMs, as per the CDC WO wishes. In Figure 4-7, a birds-eye view of such interactions is outlined. In essence, CDC VMM has to coordinate with:

1. ***CDC Monitoring Data Collector:*** as the Monitoring Data Collector is responsible for monitoring all actions, the CDC VMM has to provide a number of statistics, preferably in an asynchronous manner, so as not to saturate the collector needlessly. Statistics reported include: VM actions performed (new, started, shutdown, terminated, migrated VMs), VM actions failed or cancelled (along with error codes) and total resources allocated for running the CDC VMM (CPU, MEM, Disk IO, Net IO etc.).

2. ***DOLFIN Information Database:*** In order for the CDC VMM to correctly perform the requested actions it needs information for each DC, contained in the DOLFIN Info DB. Mostly, a description of the DC or DCs that partake in a specific operation needs to be known in advance, so that the CDC VMM can alter its action plan or cancel the operation altogether. Of interest therefore is the topology of the DC, the Hypervisor Manager (and hypervisor) used by each DC, the storage transport scheme preferred and relevant network setup (e.g. firewalls between DCs).

3. ***Cross DC Workload Orchestrator:*** Given that the CDC VMM is a delegation module for actions decided by the CDC WO, the relevant interactions include requests for local VM actions (spawn, start, shutdown, terminate) that are delegated to the relevant DCO HM and requests for migration to other DCs. The latter case is handled by the CDC WO with regards to the negotiation (i.e. finding a DC willing to receive the migrated VM) and the CDC VMM is thus notified only to coordinate the migration. It will however respond when an error occurs or a migration cannot be coordinated for some reason (e.g. hypervisor incompatibility).

4. ***Remote CDC VMM:*** Communication with a remote CDC VMM is only necessary when a migration is being negotiated. At such time, technical info about the VM under migration need to be pushed to the receiving side (to decide placement and compatibility) and a migration (destination) URI needs to be propagated to the sender. Once a connection is established, the CDC VMM delegates the migration to the DCO HM and, ultimately, the hypervisor.

5. ***DCO Hypervisor Manager:*** The CDC VMM will need to delegate most actions to the DCO HM, and will therefore need to use APIs that are common among most DC installations. Although the first target is the OpenStack API, it would be desirable to be able to communicate with DCO HMs that provide the AWS EC2 or VMWare vSphere API.

4.3.2.2 Module Interactions

In this section, the Cross-DC VM Manager REST API is defined in detail. API Calls are organized by their intended user or receiver with respect to the CDC VMM in the following Figures and presented in detail in Annex 1.

1. Cross-DC Monitoring Data Collector

This diagram presents the cross-DC VM Manager interaction with the Cross-Dc Monitoring Data Collector. The interaction is quite simple having a post action message request and an ACK reply. Messages details are presented in Annex 1.

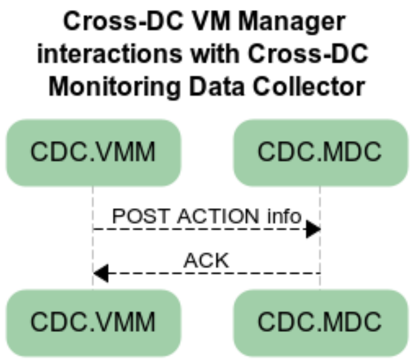

Cross-DC VM Manager interactions with Cross-DC Monitoring Data Collector
CDC.VMM
CDC.MDC
POST ACTION info
ACK
CDC.VMM
CDC.MDC

Figure 4-8: CDC VMM POSTs statistics of actions undertaken to the MDC

2. DOLFIN Information Database

This diagram presents the cross-DC VM Manager interaction with the DOLFIN Information Data Base. The interaction is quite simple having a get action message request and an ACK reply. Messages details are presented in Annex 1.

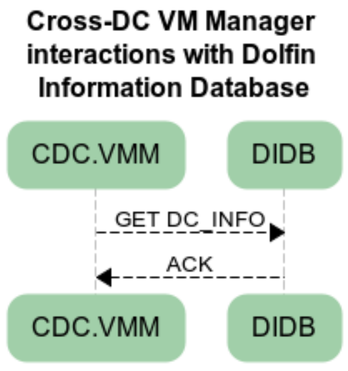

Cross-DC VM Manager interactions with Dolfin Information Database
CDC.VMM
DIDB
GET DC_INFO
ACK
CDC.VMM
DIDB

Figure 4-9: Polling the DIDB for info.

3. Cross-DC Workload Orchestrator

This diagram presents the cross-DC VM Manager interaction with the Cross-DC Workload orchestrator. The interaction is initiated by the Workload Orchestrator (CDC.WO) posting a message such as CREATE, START, SHUTDOWN, RESTART, TERMINATE or MIGRATE a VM, along with additional data and parameters. The Cross-DC VM Manager initially acknowledges the request and replies the outcome of the action. Messages details are presented in Annex 1.

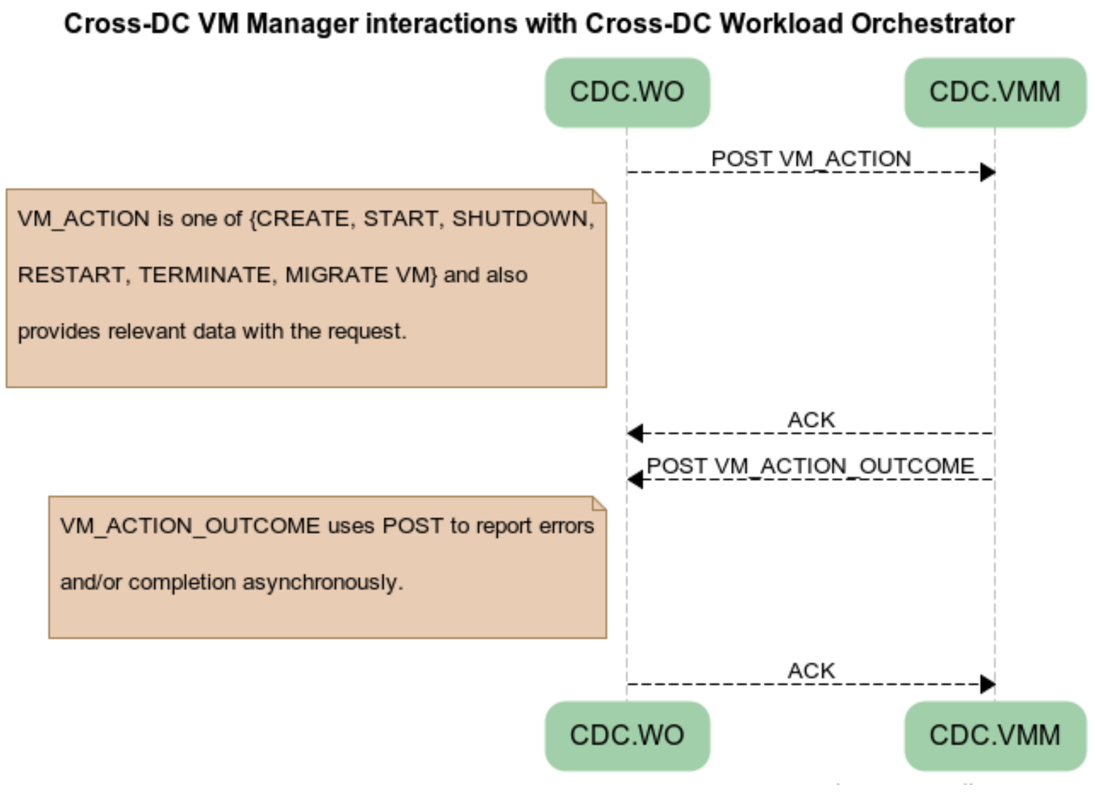

Cross-DC VM Manager interactions with Cross-DC Workload Orchestrator
CDC.WO
CDC.VMM
POST VM_ACTION
VM_ACTION is one of {CREATE, START, SHUTDOWN, RESTART, TERMINATE, MIGRATE VM} and also provides relevant data with the request.
ACK
POST VM_ACTION_OUTCOME
VM_ACTION_OUTCOME uses POST to report errors and/or completion asynchronously.
ACK
CDC.WO
CDC.VMM

Figure 4-10: Receiving request for VM actions from the Workload Orchestrator.

4. Remote Cross-DC VM Manager

This diagram presents the interaction between two cross-DC VM Managers. Each VM Manager (VMM) may request to a) get information of a specific VM, b) migrate live or offline a VM and c) get information of the migration request. Messages details are presented in Annex 1.

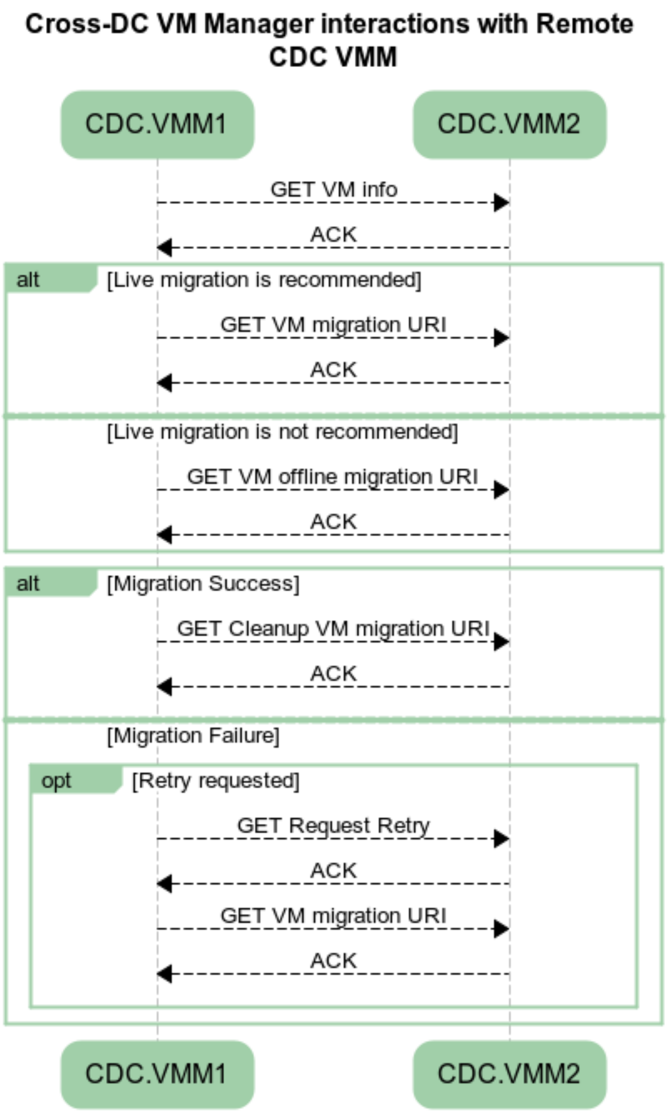


Figure 4-11: Coordinating with a remote instance of the VMM. Although the module operation is the same in either side, the flow is depicted for clarity.

5. DCO Hypervisor Manager

This diagram presents the interaction between a cross-DC VM Manager and two CDO Hypervisor Managers. Although most actions are straightforward and implemented in the relevant DCO HM APIs, the case for cross-DC migration is highlighted. Simple acknowledgements to requests are omitted for clarity. Messages details are presented in Annex 1.

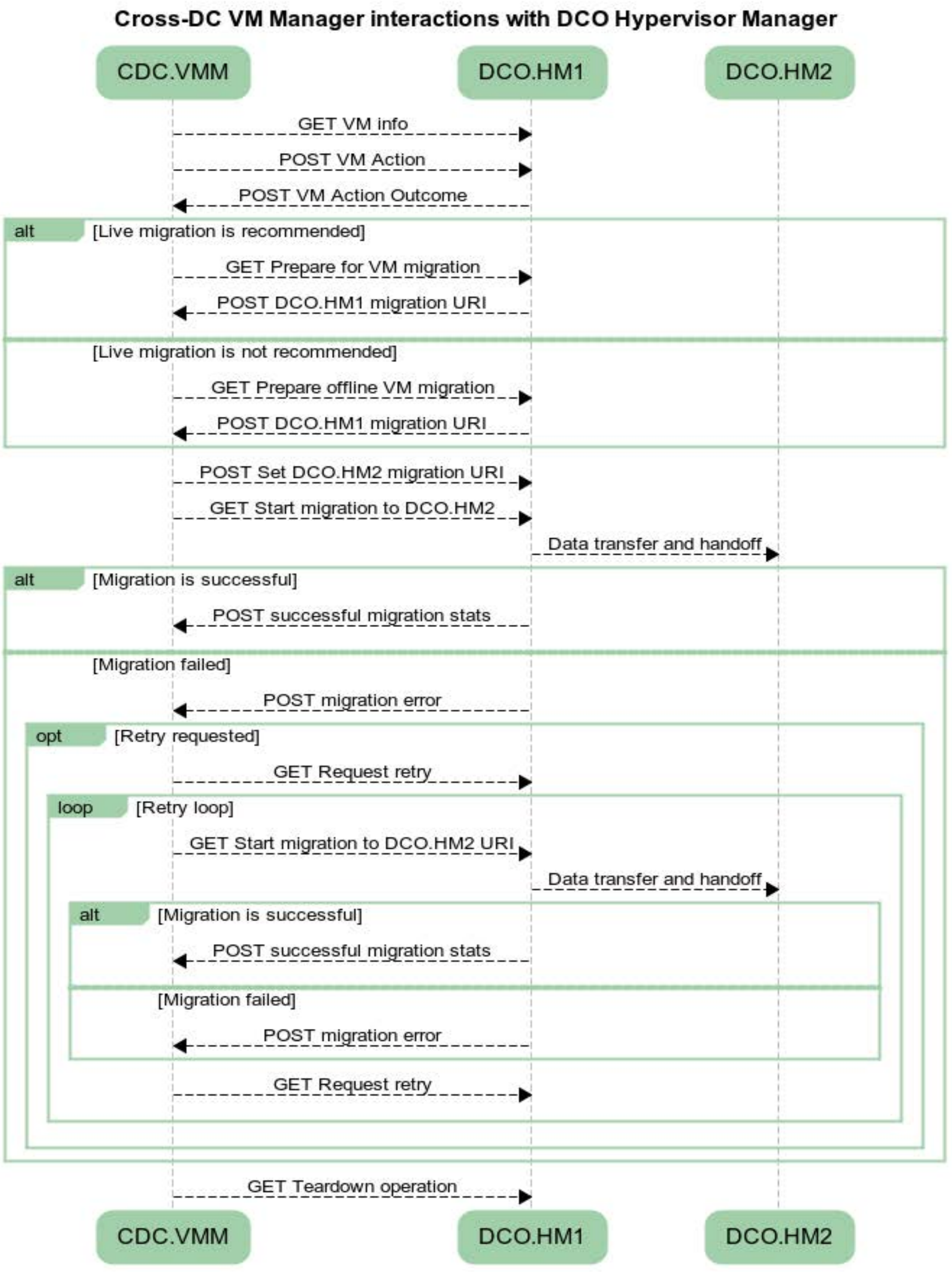

Cross-DC VM Manager interactions with DCO Hypervisor Manager
CDC.VMM
DCO.HM1
DCO.HM2
GET VM info
POST VM Action
POST VM Action Outcome
alt
[Live migration is recommended]
GET Prepare for VM migration
POST DCO.HM1 migration URI
[Live migration is not recommended]
GET Prepare offline VM migration
POST DCO.HM1 migration URI
POST Set DCO.HM2 migration URI
GET Start migration to DCO.HM2
Data transfer and handoff
alt
[Migration is successful]
POST successful migration stats
[Migration failed]
POST migration error
opt
[Retry requested]
GET Request retry
loop
[Retry loop]
GET Start migration to DCO.HM2 URI
Data transfer and handoff
alt
[Migration is successful]
POST successful migration stats
[Migration failed]
POST migration error
GET Request retry
GET Teardown operation
CDC.VMM
DCO.HM1
DCO.HM2

Figure 4-12: Interactions with the DCO HM module.

4.3.2.3 References to requirements

This section reports the list of requirements satisfied by this module. The complete list of identified requirements is described in the deliverable D2.2 [1].

Table 4-4: References to requirements (Cross-DC VM Manager)

Requirement ID & Name	Requirement description
Q15 DC2DCControlInterface	The DC2DC subsystem shall implements CONTROL interfaces to coordinate the VM migration between DCs.
Q16DC2DCDataInterface	The DC2DC subsystem shall implements DATA interfaces to perform the VM migration between DCs.
Q56 DCVMHandling (VM params)	In accordance with Ask for VMParamsAdjustment" this requirement implies that the Workload and VM Manager should be able to fulfil a request for changing the parameters of a certain set of VMs".
Q73NegotiateVMTransfer	Since DC to DC transfer may involve a multi-part balancing the way of what and how to transfer should be negotiated at VM level.

4.3.3 Cross-DC Monitoring Data Collector

The main purpose of this component is to collect knowledge not only from the Synergetic DC resources, but also from the network routers and in general from the network resources point of view. The data collected will be stored into the DOLFIN Information Data Base. When needed, the Cross DC Monitoring Data Collector will implement algorithms to produce relevant metrics and translate raw data into meaningful metrics useful to the Cross DC Workload Orchestrator to better managing the network.

The role of Cross DC Monitoring Data Collector is summarized in the following key points, related to the wider cross-DC scope:

- It receives energy related DC-level monitoring information via:
  - the DOLFIN Info DB (fed by the eCOP Policy Maker and Actuator for the local DC)
  - the other SDS components and resources
- Receives monitoring information from network elements interconnecting DCs to track resources availability, aggregating this information with the energy-related figures
- Implements aggregation algorithms to compute an overall status information describing the degree of possibility to move workload across SDCs
- Updates this information in the DOLFIN Info DB, which is then used by other SDC components to take decision on workload movement feasibility

Figure 4-13 presents a high level description of the internal structure of the Cross DC Monitoring Data Collector module. This module interacts with the DOLFIN Info DB to get architectural information and relations between cross-DC components and DCs.

The component is intended to provide the SDC-level abstraction of what the ICT Performance and Energy Supervisor does at eCOP layer, oriented towards networked DC interactions. The federation-oriented architecture of the Cross DC Monitoring Data Collector will be detailed in the next paragraphs in order to clarify the functionalities of each depicted element.

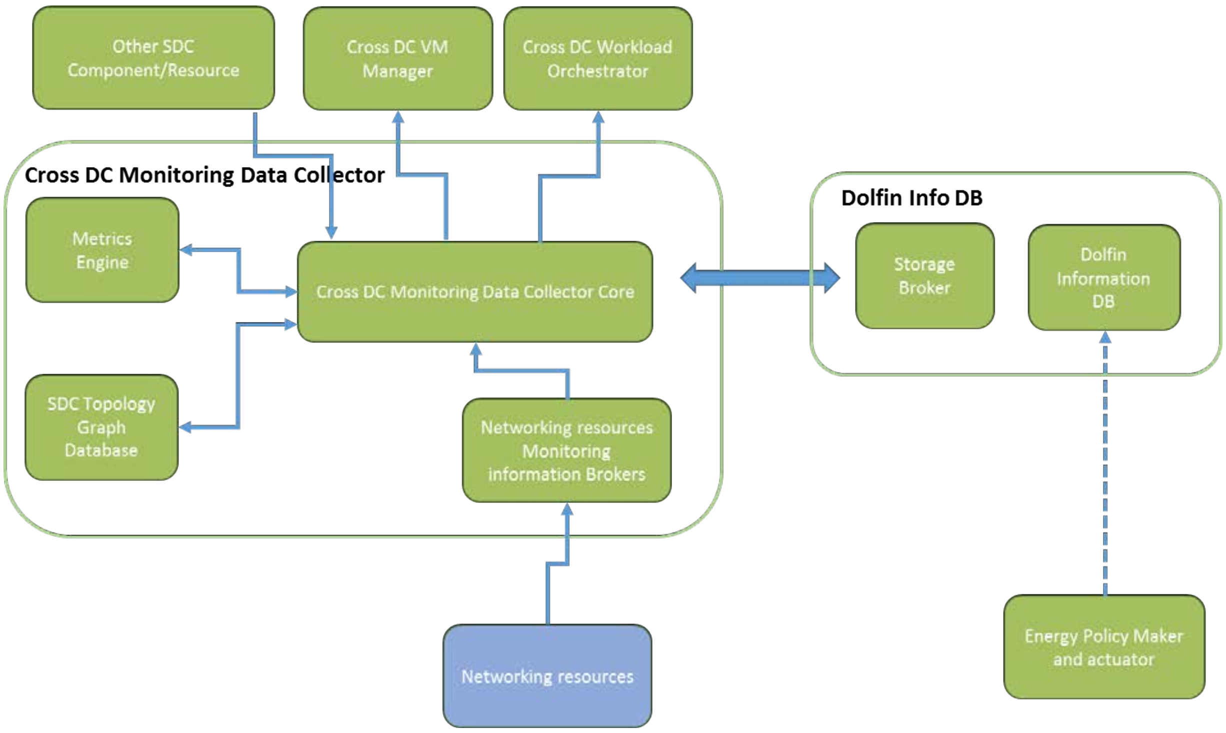

Other SDC Component/Resource
Cross DC VM Manager
Cross DC Workload Orchestrator
Cross DC Monitoring Data Collector
Metrics Engine
Cross DC Monitoring Data Collector Core
SDC Topology Graph Database
Networking resources Monitoring information Brokers
Dolfin Info DB
Storage Broker
Dolfin Information DB
Networking resources
Energy Policy Maker and actuator

Figure 4-13: Cross DC Monitoring Data Collector module

4.3.3.1 Module detail description

The basic objective of Cross DC Monitoring Data Collectors is to provide a set of functionalities aiming at receiving monitoring data from the dolphin Information DB (in turn sent by the eCOP Energy Policy Maker and Actuator), from the other SDS components resources, and from the networking resources. This information will be aggregated and a real time status picture of the energy consumption and efficiency degree at each DC will be stored and updated in the DOLFIN Info DB.

Each inbound data event is correlated to its physical source, matching this information with the higher-level topology information stored in the DOLFIN DB. Once the relationship of an event with its corresponding physical source has been sorted out, each data update event is stored into the DOLFIN Information DB component, whose purpose is, among others, to keep the overall status of the DOLFIN SDC system over time.

Moreover, the Cross DC Monitoring Data Collector makes use of the network status and these streams of data in order to perform series of data aggregating procedures, leading to the generation of compound metrics.

The Cross DC Monitoring Data Collectors are envisioned to comprise the following sub-modules:

- Cross DC Monitoring Data Collector core
- Metrics Engine
- SDC Topology Graph Database, a temporary runtime structure derived from information stored in the DOLFIN Info DB

4.3.3.2 Module decomposition

4.3.3.2.1 Cross DC Monitoring Data Collector core

This sub-component coordinates the communications between the other sub-modules and exposes multiple interfaces used by other higher-level DOLFIN components such as the Cross DC Workload Orchestrator and Cross DC VM Manager.

This component is essentially a stateful data hub, continuously interfacing with a low number of other actors, and making low utilization of computational power. An asynchronous event-driven development framework will be used for the implementation of this component, using a reactive high level language.

- **The Event Receiver** collects information from the eCOP layer (accessing the DOLFIN Information DB, where they are stored), from SDC components or from networking resources and places the events into the Queue. Depending on its type, this sub-component assigns a priority level to each event, so that the events are organized in the queue based on their priority. This way, if a high priority event arrives to the Event Receiver, it will be pushed to the first position of the queue and it will be marked as the next event to be processed. Moreover, the events will be characterized an associated timestamp. The timestamp provides information to the Metrics Engine about the time that the event has been generated.
- **The Queue** stores the events in priority order, as explained in the previous paragraph. If two elements have the same priority, they are served according to their arriving order in the queue in a First In First Out mode.
- **The Event Process Engine** classifies the event and executes the appropriate actions, such as metrics recalculation or SDC Topology Graph DB update, taking into consideration recent relevant data saved in the memory of the component, exposing basic caching capabilities; in case an external entity requests for metrics information, it can perform a quick search in its cache memory for a faster response. If the information is not found there, the module forwards the request to the DOLFIN Information DB.

4.3.3.2.2 Metrics Engine

This sub-component aggregates the DOLFIN Information Database data by using patterns obtained by the SDC Topology Graph Database and produces updated metric values which are then made available to the Cross DC Monitoring Data Collector core sub-component and, finally, to the other DOLFIN modules. This engine is not self-invoked; its function is triggered by the Cross DC Monitoring Data Collector core sub-component when the chosen criteria are met. It is therefore designed to run on-demand, exhibiting computationally-intensive behaviour, when active.

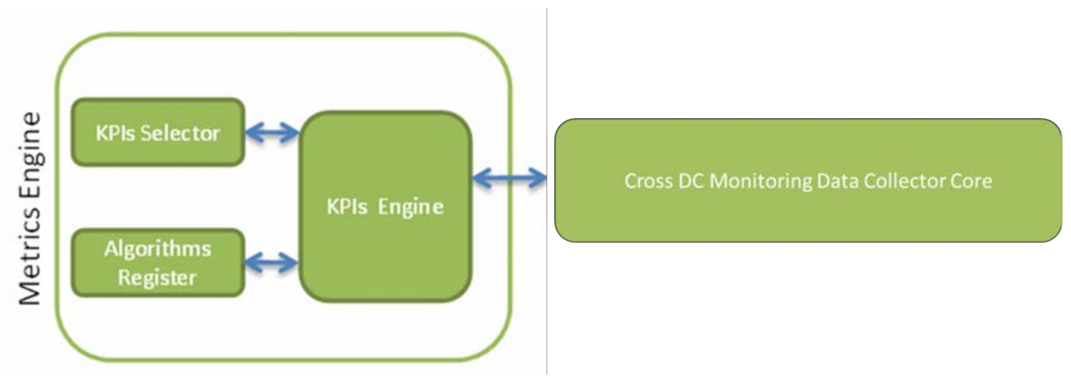


Figure 4-14: Metrics Engine

The above figure depicts the Metrics Engine. The KPIs Selector stores the different metrics the Metrics Engine can produce. When a new event triggers the Metrics Engine, it requests the KPIs to the KPIs Selector. The KPIs provided depend on several factors, for example the type of the event or the KPIs activated by the administrator. The Algorithms Register is a database hosting the implementation details of the algorithms used for calculating the requested KPIs. The Algorithms Register together with the KPIs Selector makes the system more flexible, allowing for easy addition and update of KPIs and of the algorithms for calculating them.

The Metrics Engine deals with large sets of data; computes the metrics in one run, processing big amounts of data. Its architecture is fairly simple but tricky to scale. For this reason a development pattern suitable for intensive data computation of sparse data set will be used.

4.3.3.2.3 Networking resource monitoring information broker

The Cross DC Monitoring Data Collector comprises one sub-components specialized in handling raw data, producing a stream of data events suitable for persistence within the DOLFIN Information Database. Its primary objective is to form relationships between a data event and its corresponding record in the SDC Topology Graph Database, representing the physical asset which originated the data event.

The DCO Hypervisor Broker collects data coming from the federated DC's network resources. It is basically an event handlers. Similarly to the Cross DC Monitoring Data Collector core, an event-driven asynchronous development pattern, capable of aggressive data caching, is best suited for this component and will be adopted throughout the implementation phase.

To facilitate the communication to and from the Broker, a proper data handling protocol needs to be adopted, allowing the data producers (publishers), the consumers (for example the Cross DC Monitoring Data Collector core) and the broker to be distributed on different physical locations/servers. The functional module is schematized as follow:

- The producer publishes the messages to an Exchanger entity that resides inside the broker.
- The Exchanger replicates the messages and forwards them to message queues. Each queue provides two main functions: (1) data caching, (2) spike rate mitigation. Moreover, more message queues might be implemented to perform optimal load balancing.
- The consumer proceeds with the subscription to the broker to obtain the delivery of the message in the queue. The delivery model might support both an asynchronous notification and on demand, using a request/reply model.

To increase reliability, the broker should support acknowledgment mechanisms, counteracting the effect of networking problems or application fails. In this course, when a message is sent to the consumer, the consumer has to send an explicit notification to the broker, saying that the data has been correctly received and processed. After having received this acknowledgement, the broker will remove the data from the queue.

4.3.3.2.4 SDC Topology Graph Database

The SDC Topology Graph Database contains a runtime version of the data describing the physical distribution and topology of the federated DC network assets. The persistent information about federated DC topology is kept in the DOLFIN Info DB. This sub-component is used at runtime to allow a higher degree of freedom in computing the optimization algorithms. This database of intertwined data elements must reflect the actual allocation of the federated DC resources. In this framework, this database acts as a source of sensible data in the context of metrics production, so that the SDC Monitoring Data Collector has local knowledge of the originating context of all data events.

4.3.3.2.5 References to requirements

This section reports the list of requirements satisfied by this module. The complete list of identified requirements is described in the deliverable D2.2 [1].

Table 4-5: References to requirements (Cross-DC Monitoring Data Collector)

Requirement ID & Name	Requirement description
Q9 MonitorInterface	The monitoring subsystem shall be able to present metrics-information to the management subsystem. This interface should expose a number of metrics and their frequency in a normalized form. It could receive the metrics to retrieve their frequency and response format (e.g. publish to a message broker). The system shall monitor the DC infrastructure: IT devices, power units, other DC facilities that might be involved in the process of energy optimization.
Q12 MonitorOpenDataFmt	The monitoring subsystem should utilize open data formats available.
Q7 MonitorParams	The monitoring subsystem shall always monitor DC parameters. This requirement necessitates the formal definition of all parameters that have to be monitored to quantify KPIs and evaluate the DC performance. This allows the implementation of the appropriate actions and policies. The list of parameters to be published should be configurable in terms of frequency of measurements and way to deliver. Also, the access to this functions should be controlled.
Q11 MonitorResilience	The monitoring subsystem should be as much resilient as possible (i.e. implement heartbeat, watchdog functions).
Q8 MonitorTranslate	The monitoring subsystem shall always translate parameters to metrics (KPI).
Q10 MonitorVariousData	The monitoring subsystem shall handle many kind of data of different nature (i.e. performance information, energy consumption, power dissipation, VM status and deployment, semi-static parameters, etc). This function will complement in terms of Data the other peer Monitoring interfaces.

4.3.4 DOLFIN Information Data Base

4.3.4.1 Module detail description

The DOLFIN Information Data Base (IDB) offers abstracted and logically-centralised information manipulation (including information collection, aggregation / processing, storage / indexing and distribution) across all DOLFIN architectural components. Moreover, it includes information of the local energy requirements e.g. Demand/Response requests from the local Smart Grid operator.

It uses two separate interfaces for communication with the DOLFIN software entities: (1) the Information Management Interface is used for information manipulation configuration, including the information sources/sinks registration to the Information Base, the management of internal Information Base, information manipulation functions and the establishment, operation and optimisation of information flows; and (2) the Information Exchange Interface that offers the actual information exchange capability to the DOLFIN components. These two interfaces allow a unified yet abstracted information handling in all types of participating entities.

The DOLFIN IDB provides storage and indexing, along with processing functionalities. It maintains a registry, storing specifications for the available information to be collected, retrieved, or disseminated. Moreover, it may apply aggregation functions (e.g. MAX, MIN, AVERAGE) to the collected data before they are stored or disseminated or filter data at the aggregation level for optimisation purposes.

Last but not least, in order to manage systems in the DCs in a smart optimized, and automated way and as much as possible in "real time" and in order to achieve energy savings and the resulting C02 emission reductions or to meet the different demands from EPs in terms of optimization or different distribution of the power consumption of the smart grids, the DOLFIN expert system needs to have the topology data of infrastructures in DCs, the electricity consumption and infrastructure performance data of hardware infrastructures of the DCs constantly available. This info is also stored in the DOLFIN Energy Information Base.

4.3.4.1.1 Energy Information Base

The Energy Information Base is a logical database of the status of energy consumption of each individual DOLFIN-enabled DC. It stores performance metrics that describe the utilisation and energy consumption of each DC at the granularity provided by each individual DCs Energy Policy Maker & Actuator component. This way DOLFIN maintains a log of each DC's relevant energy metrics, in a manner similar with the eCOP DB. Thus, energy consumption trends over different time and date periods can be identified for each individual DC and also as a whole. In addition, the EIB allows processing and data transformations as well as information collection, aggregation, indexing and distribution, through its Storage Broker component.

The structure used for storing that data will be efficient enough to support fast search, insertion and retrieval operations. Well-designed databases are easy to enhance, simpler to work with, and has fewer problems when you need to extract information. For this reason the database will meet the following characteristics:

- ***Self-describing nature of database system***: the data base has to contain a complete definition of the structure and constraints, not only the database itself. The information is saved in the DBMS. It contains information about the structure of each file, the type and storage format of each data item, constrains on the data and the relationship between the data.
- ***Concurrency and consistency***: The control of data concurrency and data consistency is essential. Data concurrency involves that many users can access data at the same time and data consistency implies that each user sees a consistent view of the data. In a multiuser database, the statements within multiple simultaneous transactions can update the same data and need to produce consistent results. Concurrent and consistent use of data increases the economy of a system. Data capturing and data storage is not redundant, the system can be operated from a central control and the data can be updated more efficiently.
- ***Data Integrity***: The concept of data integrity guarantees that all data in a database can be traced and connected to other data. Enforcing data integrity ensures the quality and reliability of the data in the database. It includes the protection of the database from unauthorized access and unauthorized changes. Having a single, well defined and well controlled data integrity system increases stability, performance, reusability and maintainability
- ***Data Persistence***: Data persistence means that the data is maintained as long as it is not deleted explicitly. The life span of data needs to be determined directly or indirectly be the user and must not be dependent on system features. Additionally data once stored in a database must not be lost. Changes of a database, which are done by a transaction, are persistent. When a transaction is finished even a system crash cannot put the data in danger.

- ***Data abstraction***: It will offer an abstracted and logically-centralised information manipulation across all DOLFIN architectural components.

The Energy Information Base uses two separate interfaces for communication with the DOLFIN software entities:

- The Information Management Interface: is used for information manipulation configuration, including the information sources/sinks registration to the Information Base, the management of internal Information Base, information manipulation functions and the establishment, operation and optimisation of information flows
- The Information Exchange Interface that offers the actual information exchange capability to the DOLFIN components.

4.3.4.1.2 Energy Information Systems

DOLFIN keeps the DC performance monitored at all times. Hardware and software probes are deployed across the DC infrastructure and provide a continuous flow of measurement data which tracks the functioning of the services and ICT hardware at the highest possible level of detail.

DOLFIN uses this flow of data to produce and update various measurement metrics which can be better used to perform energy efficiency evaluations and to screen the DC in search of sub-optimal energy exploitation conditions. In order to produce the needed metrics, the ICT Performance and Energy Supervisor component perform what effectively is an operation of normalization and aggregation of the data stored in the DCO Monitor database. The task of the ICT Performance and Energy Supervisor is to extract all the relevant data from the DCO Monitor database, process it with various aggregation patterns and produce aggregated readings which briefly describe the overall DC functionality in simple form, at low detail level.

The extracted readings have multiple purposes:

- they are used as inputs for metric computation and updating. The various metrics produced by the ICT Performance and Energy Supervisor are the indexes which express in very clear, numeric form, various aspects of the current energy efficiency and effectiveness of the DC, and are the primary input factored in by the Energy Policy Maker component
- they can be stored for later reuse. These are not only useful to produce statistics and for historical reporting of the DC performance, as the Energy Policy Maker may also access this historic data to extract trend features, or evaluate potential plan-ahead policies.

The ICT Performance and Energy Supervisor exposes a query interface which allows easy and searchable access to various quantities. In particular, it exposes:

- access to the current computed metrics, and their historical data;
- access to the aggregated values which are inputs for the metrics evaluation, and their historical data;
- access to the un-aggregated data being reported by the monitored appliances, in a clear and normalized format.

4.3.4.2 Module decomposition

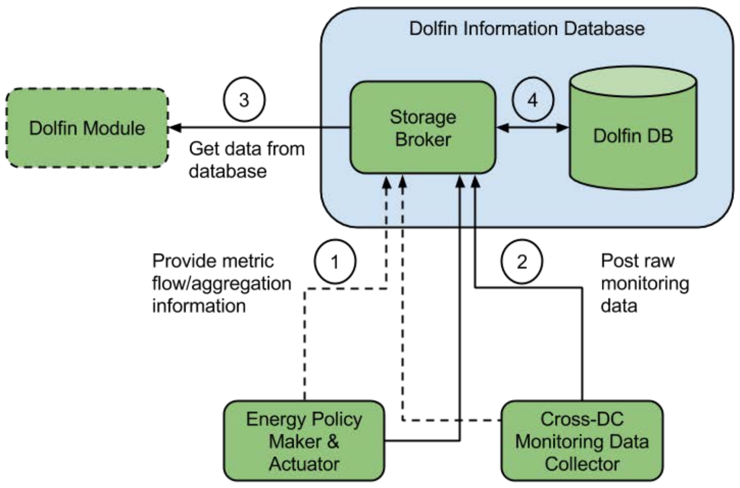

Dolfin Information Database
Dolfin Module
3
Get data from database
Storage Broker
4
Dolfin DB
Provide metric flow/aggregation information
1
2
Post raw monitoring data
Energy Policy Maker & Actuator
Cross-DC Monitoring Data Collector

Figure 4-15: High level depiction of DOLFIN Information DB's interactions.

As previously outlined, the DOLFIN Information Database module needs to store and manipulate a wide range of data provided and required by the various DOLFIN components. As such, the DOLFIN Information Database needs to provide:

1. An API to define raw data storage parameters, aggregated metrics computed from raw data and the retention and access schemes for the relevant data. This API is in line with the Information Management Interface defined above, and exposed to the CDC Monitoring Data Collector module.
2. An API to receive raw and aggregated monitoring data from CDC MDC. These data are stored for reference in the DOLFIN DB and are potentially processed further to derive new metrics/KPIs as needed. This API forms part of the Information Exchange Interface described in the previous section.
3. A way to query the database for information. This functionality is available to any DOLFIN module, and makes the DOLFIN DB the primary information repository for all participating DCs.
4. A way to extract relevant KPIs from stored data. As most metrics are defined as aggregated flows of raw monitoring data flows, this function should be satisfied with simple retrieval operations to the DB as the CDC MDC has performed online aggregation. However, the modular design of the DB enables the Storage Broker to compute metrics through offline analysis of the historical data stored in the DB. Thus, a module can extract higher level information from the data stored in the DB.

The Storage Broker module of the DB provides a simple interface that abstracts the peculiarities of the internal storage architecture. The Storage Broker can be extended to interface with different datastore through different backend-specific plugins. Thus, a large number of backends can be used for information storage.

4.3.4.3 References to requirements

The list of requirements satisfied by this module follows. The complete list of identified requirements is described in the deliverable D2.2[1].

Table 4-6: References to requirements (DOLFIN Information Database)

Requirement ID & Name	Requirement description
Q7MonitorParams	The monitoring subsystem shall always monitor DC parameters. These requirements identify the necessity to make a formal collection for all parameters that have to be monitored to build KPI and other triggers levels used to evaluate the DC performance and schedule the appropriate policies' actions. The list of params to be published should be configurable in terms of frequency of measurements and way to deliver. Also, the access to these functions should be controlled.
Q8MonitorTranslate	The monitoring subsystem shall always translate params to metrics (KPI).
Q9MonitorInterface	The monitoring subsystem shall be able to present metric information to the management subsystem. This interface is considered to expose in a normalized form a number of metrics and their frequency. It could receive the metrics to retrieve, their frequency and response format (e.g. publish to a message broker). The system shall monitor the DC infrastructure: IT devices, power units, other DC facilities that might be involved in the process of energy optimization.
Q10MonitorVariousData	The monitoring subsystem shall handle many kind of data and with different nature (i.e. performance information, energy consumption, power dissipation, VM status and deployment, semi-static parameters, etc). This function will complement in terms of Data the other peer Monitoring interfaces.
Q11MonitorResilience	The monitoring subsystem should be as much resilient as possible (i.e. implement heartbeat, watchdog functions).

4.3.5 Smart Grid Controller

The Smart Grid Controller will act as the component that integrates the DOLFIN platform with the Smart Grid network. The purpose of this component is to be able to interact with the energy related requests from the Smart Grid side.

DOLFIN is targeting the Smart Grid demand/response scenario, so as to maintain the electricity balance in the distribution network. To this respect, the Smart Grid, acting as the controlling entity, will identify that an electricity stabilisation issue is about to occur. In order to avoid such a situation, the Smart Grid network will issue a request, towards the DC, for lowering the power consumption. This could be expressed as an absolute figure, determining the exact level of power consumption that should be followed, or it could also be expressed as a percentage of decrease to the current levels of power consumption. The role of the Smart Grid Controller is to intercept and perform an initial examination of such message requests from the Smart Grid.

To this respect, the Smart Grid Controller is required to exchange messages with the Smart Grid network and be able to parse them, so as to create an internal representation of the message that would be understandable by the DOLFIN platform. Furthermore, this component will have to incorporate the knowledge to communicate with possibly different protocols with the Smart Grid, thus acting as a kind of gateway for interfacing with Smart Grid networks.

4.3.5.1 Module detail description

The Smart Grid Controller (SGC) acts as a broker between the core DOLFIN infrastructure and subsystems, represented by the Energy Efficiency Policy Maker & Actuator, and the outside world, namely the Smart Grid (SG) and the Energy Providers (EPs).

The SGC undertakes to support the DOLFIN System awareness of its energy surroundings, namely energy prices and SG status in terms of energy availability and (smart) energy grid stability. In this framework, it is responsible for assuring that the current and near future energy consumption profile of the DC are known to the SG and the EPs at all times. Last, it is responsible of informing the SG and the EPs of the availability of the DC's own energy resources, if any (e.g. in the case the DC is supported by one or more RES). The information from the DC side is provided by the Energy Efficiency Policy Maker & Actuator. The same holds for the information originating from the SG/EPs side; all is redirected to the Energy Efficiency Policy Maker & Actuator.

The data that are relayed to the Energy Efficiency Policy Maker and Actuator and the SG/EPs are lightly processed by the SGC in order to detect possible suboptimal situations that could lead to either less cost- and energy- efficient operation of the DC, or less stable SG operation, respectively. In that case, the interested party is responsible of issuing a proper action plan to optimize its place in the energy consumption/production chain. The SGC is not responsible for issuing such a plan; instead, it is responsible for communicating this information between the interested entities.

In short, the main tasks of the SGC are to:

- Handle information related to the DC energy consumption and relay it to the SG/EPs;
- Communicate to the SG/EPs information related to the DC's own energy sources, if available;
- Handle information related to the energy tariffs from the SG/EPs side.

It should be noted that the SGC should be in place to support more than one communication protocols, in order to guarantee reliable communication with the set of cooperating SG Operators and the EPs. From an architectural perspective and to fulfil all the requirements accompanying the aforementioned desired functionality, the SGC could be modelled as depicted in Figure 4-16.

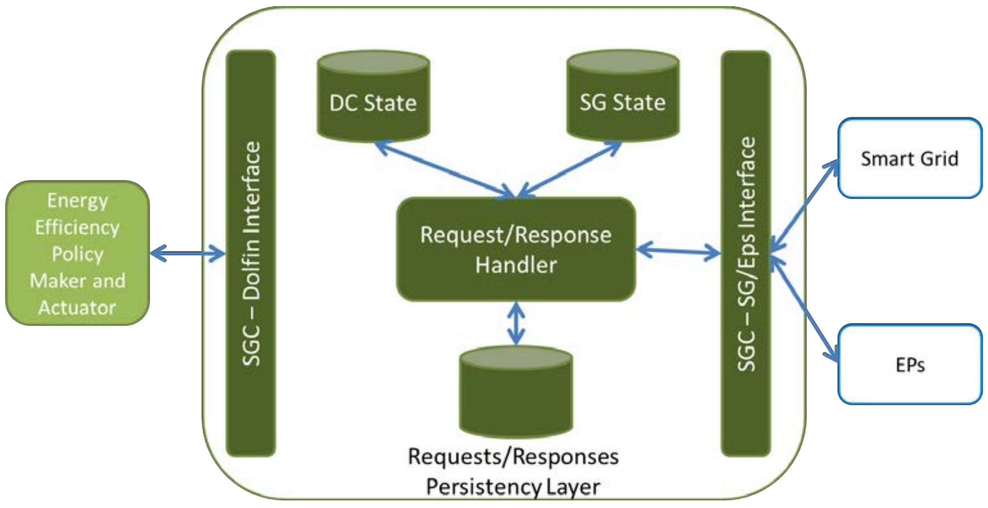


Figure 4-16: The SGC high level architecture

Every time a request is received by the SGC, it is translated in an internal SGC format (through the SGC-DOLFIN/SG/EPs interfaces) and is stored into the Requests/Responses Persistency Layer. Then, the request is retranslated and forwarded to the interested entity. Upon reception of an answer, any information related to the DC/SG state, if any, is stored in the appropriate registers. Note that the DC state may comprise information related to its current and near future computing load, the availability of its own energy sources (if any), whereas the SG state may include data relate to the current energy tariffs or their trend, the availability of energy in the SG along with some grid stability metrics indicating possible instability in the near future. Granted the previous DC/SG states, the SGC will evaluate whether an alert should be sent to any interested entity (DC/SG/EPs) along with the set of appropriate state information in order to render the receiver capable of performing sets of corrective actions to improve the DC cost- and energy-efficiency (from the DC side) or the SG stability (from the SG/EPs side). In any case, the SGC acts as a mediation layer between the Energy Efficiency Policy Maker & Actuator and the SG/EPs side, also providing minor caching services (regarding the DC/SG states) in order to limit overhead, unnecessary traffic to/from the DOLFIN core internal ecosystem.

4.3.5.2 Energy Provider Interaction

Demand/response refers to mechanisms used to encourage consumers to reduce their demand and subsequently reduce also the peak demand for electricity. Electrical generation and transmission systems are sized to correspond to a predetermined peak demand, plus a margin for unforeseen events. By lowering the peak demand, we actually reduce the overall plant and capital cost requirements. Electricity Grid operators generally plan to use the least expensive generating plant (in terms of marginal cost) at any given period and use additional capacity from more expensive plants, as demand increases. Demand/response in most cases is targeted at reducing peak demand to reduce the risk of potential disturbances, avoid additional capital cost requirements for additional plant and avoid the use of more expensive and/or less efficient operating plant. On the other hand, consumers of electricity will also benefit by paying lower prices. To automate the process of demand response, technologies are employed to detect the need for load shedding, communicate the demand to participating consumers, automate load shedding and verify compliance with demand/response programs. To this respect and considering also that DCs are big consumers of electricity, it is reasonable that Smart Grid operators would like to apply demand/response mechanisms across DCs.

Our approach for interfacing with the Smart Grid is to utilize Open Automated Demand Response (OpenADR), which is a research and standards development effort for energy management. OpenADR is a communications data model designed to facilitate sending and receiving demand/response signals from a utility or independent system operator to electric customers. The intention of the data model is to interact with building and industrial control systems that are pre ‐ programmed to take action based on a demand/response signal, enabling a demand/response event to be fully automated. The OpenADR specification designed to facilitate common information exchange between a utility and their end ‐ use participants. During a demand/response event, the utility provides information regarding what has changed and on what schedule, such as start and stop times. The OpenADR Specification defines the interface to the functions and features of a Demand Response Automation Server (DRAS) that is used to facilitate the automation of customer response to various demand/response programs and dynamic pricing, through a communicating client.

Such an approach fits with the DOLFIN way, and the Smart Grid Controller will act as a gateway which will receive Demand/Response Events from a Demand Response Automation Server and will convert them into information for the Energy Efficiency Policy Maker and Actuator.

4.3.5.2.1 Demand/Response Event Information

Demand/Response Events are typically designed to carry a variety of information to cause reactions by customers of a utility's request. In some cases prices are used to trigger responses by a customer to the DR events, while in other cases it might be a shift level event. In general there can be a wide range of different types of information associated with a DR event depending upon how the DR strategy of a utility is designed. This information is represented by EventInfoInstance entities. Indicative data held by one of these Demand/Response EventInfoInstance entities includes:

- name – This is the name of the type. Analogous to a variable name.
- typeID – This identifies the type of information and may take on one of the following values:
    - PRICE_ABSOLUTE - Price number, e.g. €0.25
    - PRICE_RELATIVE - Change in price, e.g. €0.05
    - PRICE_MULTIPLE - Multiple of current price, e.g. €1.5
    - LOAD_LEVEL - Amount of load based on an enumeration, e.g. moderate, high, etc.
    - LOAD_AMOUNT - Fixed amount of load to shed or shift, e.g. 5MW
    - LOAD_PERCENTAGE - Percentage of load to shed or shift, e.g. 10%
    - GRID_RELIABILITY - Number signifying the reliability of the grid
- scheduleType – This specifies how a schedule may be associated with the DR Event information is defined and may take on the following values:
    - NONE–There is no schedule and thus the EventInfo does not change values during the entire DR event ACTIVE state.
    - DYNAMIC–The time schedule is not fixed during configuration, but can be set when the DR event is issued.
    - STATIC–The schedule is fixed when the DR program is configured within the DRAS
- Schedule – If the scheduleType is STATIC then this is the configured schedule. A schedule is a sequence of time slots that are valid over the entire ACTIVE period of a DR event. Each time slot may take on a different value in the EventInfoInstance
- enumerations – This is a list that defines a fixed set of values that the EventInfo instance may take. If defined, the EventInfoInstance is an enumeration and can take on any of the values in the list. If left undefined, the EventInfo instance can take on any contiguous value between the minValue and maxValue.
- minValue – Minimum possible value of an EventInfoInstance.
- maxValue – Maximum possible value of an EventInfoInstance.

These Demand/Response Events, after going through the Smart Grid Controller and being passed to the Energy Efficiency Policy Maker and Actuator, may affect how DOLFIN places it's VMs.

4.3.5.3 References to requirements

This section reports the list of requirements satisfied by this module. The complete list of identified requirements is described in the deliverable D2.2 [1].

Table 4-7: References to requirements (Smart Grid Controller)

Requirement ID & Name	Requirement description
Q53: DC2 SmartGridInterface	In order for the Smart Grid Controller to be able to communicate with both the Smart Grid/EPs and the DOLFIN-enabled DCs, a proper interface serving the relevant requests/responses of such communication is needed.
Q54: DCHeatControl	In case the Smart Grid Controller retrieves a demand for a heat exchange state change from the DC part, the Energy Eff. Policy Maker and Actuator should be able to control the heat exchange towards the Smart City (turn it off or on).
Q62: Energy Provider Cost profile	The Aggregators should be able to request for the tariff schemes of a certain set of EPs, through the Smart Grid Controller.

4.3.6 SLA Renegotiation Controller

4.3.6.1 Module detail description

The SLA Renegotiation refers to the process of automatically negotiating and agreeing with DC customers in lowering 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. Thus, the SLA renegotiation process is of paramount importance for the DOLFIN ecosystem, facilitating the DOLFIN operation at a synergetic DC level.

In order to capitalize on the merits offered by the SLA Renegotiation, the DOLFIN architecture comprises the SLA Renegotiation Controller, a component dedicated to the monitoring of the SLA parameters, the enforcement of the SLAs and the automated renegotiation of the SLAs, based on the predefined customer profiles. In this course, 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.[2]. In particular, the involved stages are:

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

In order to monitor/enforce the SLA terms, the expected service guarantees are defined in the form of a SLA Template. Specifically, the defined SLA template links different levels of SLA guarantees (i.e. different combinations of SLA parameters) to different combinations of virtual resources [2]. Thus, prior to the instantiation of each VM by a DC customer, the SLA of the customer is instantiated, with the DC customer reserving a format of virtual resources, while consenting to a certain level of service.

4.3.6.1.1 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. The basic idea is that the actions taken by DOLFIN should aim to maximize efficiency while still observing the SLA negotiated with the customers. Thus before we analyse the SLA issues, we shortly analyse the issue of User Profiles, as they are closely interrelated. The following service characteristics and their specific prioritization for each individual 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 a low as possible. In several cases, when the rest of the characteristics are of low importance, cost can be a dominant factor in a customer’s profile.
- **Performance:** In performance critical applications like HPC computing or low-latency computing 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 for utmost importance for several classes of users and applications that need to have strict guarantees for business critical or personal information.
- **Environmental friendliness:** In several 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 parameters 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 take role in 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 point establishes what are the necessary steps for testing whether or not the SLAs are compliant on with what it customer agreement requires. The SLA is generally expressed in “textual form”, for example:

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

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

- ***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 time may be dependant by the category of the fault (i.e. Critical, High, Medium, Low).

- **Performance.** The performance parameters are (1) dependent from the service and (2) are 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 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)

- **Environmental friendliness:** This parameter quantifies the "sensitivity" of the user to the DC awareness of the energy and CO2 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) 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 following diagram is provides a schematic model for the Customer Profiles DB.

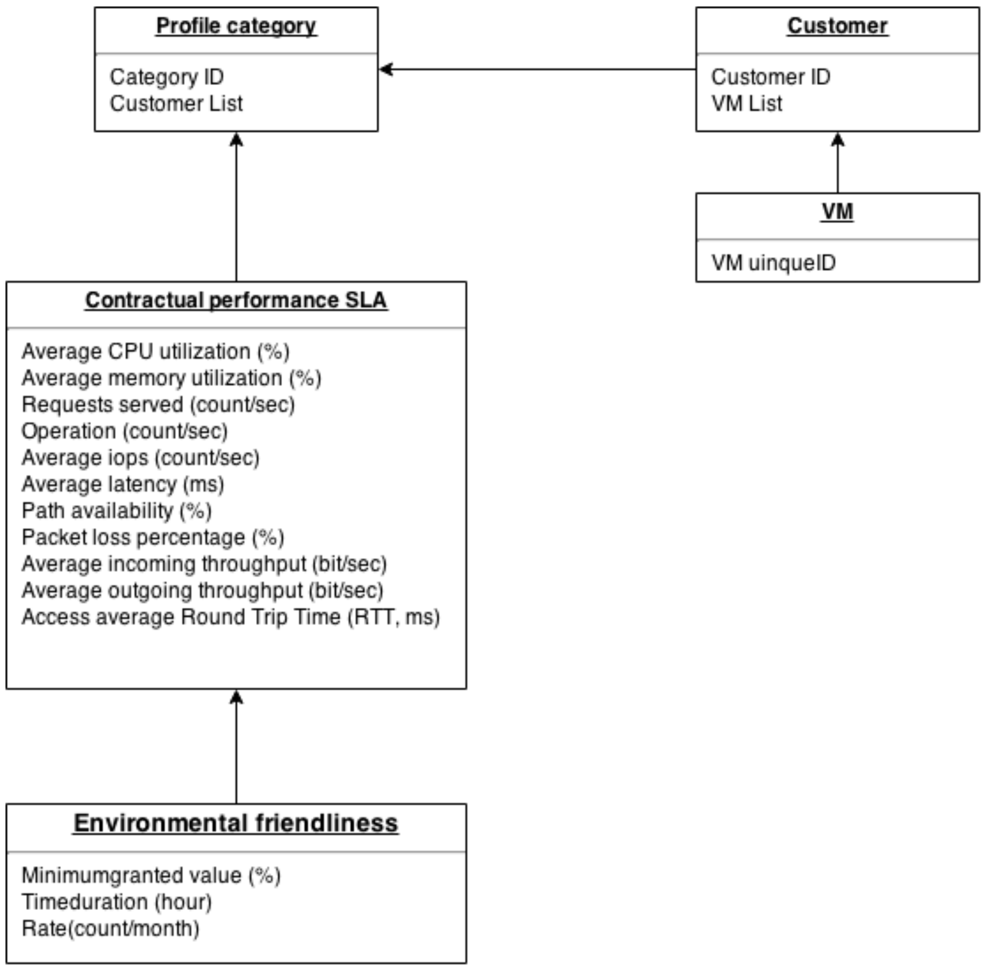

Profile category
Category ID
Customer List
Customer
Customer ID
VM List
VM
VM uinqueID
Contractual performance SLA
Average CPU utilization (%)
Average memory utilization (%)
Requests served (count/sec)
Operation (count/sec)
Average iops (count/sec)
Average latency (ms)
Path availability (%)
Packet loss percentage (%)
Average incoming throughput (bit/sec)
Average outgoing throughput (bit/sec)
Access average Round Trip Time (RTT, ms)
Environmental friendliness
Minimumgranted value (%)
Timeduration (hour)
Rate(count/month)

Figure 4-17: Schematic model for the Customer Profiles DB

4.3.6.1.2 SLA Templates

During the SLA instantiation, the DC 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 the 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 allotted in advance to computational resources such as the number of reserved CPU cores, the type of reserved CPU cores, the RAM, the HDD and the HDD throughput.

Moreover, Green VM features which are not amenable to renegotiation, but are 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.

Having defined the virtual resources and Green VM features as part of the SLA templates, 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[3].

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 live VM, taking a snapshot of it and relocating it to a synergetic DC. The VM CPU speed, indicates a threshold value above which the VM CPU speed needs to be, allowing the DC operator to migrate VMs to servers of reduced CPU speed and therefore of reduced energy consumption (through DVFS), whereas a renegotiation of the contractual VM CPU speed would allow the VM migration to a DC of obsolete or lower level resources, providing 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. Concluding, 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 disrespect the contractual defined SLA parameters.

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

4.3.6.1.3 SLA Instantiation

Subsequently to the definition of the contractual SLA templates defined by the above parameters follows 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 the form 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 the SLAs and also exploit the flexibility provided by the acceptable SLA downgrade toward energy efficient DC operation at a synergetic DC level.

4.3.6.1.4 SLA Monitor and Enforcement

In order 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 monitors continuously the VM SLAs and to do so the Controller 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, to obtain information on the current status of the VM SLAs.

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.

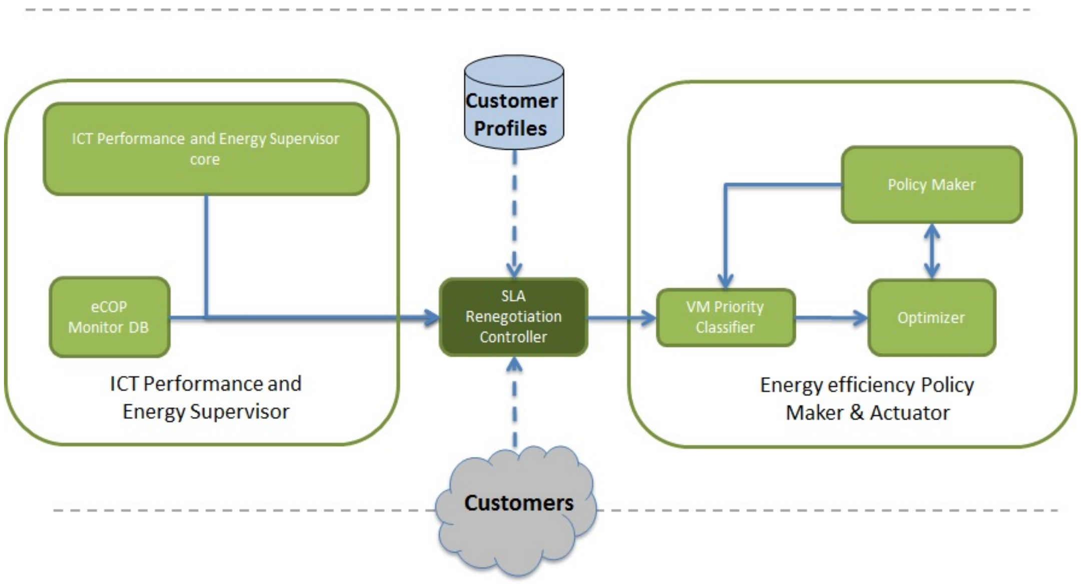

Customer Profiles
ICT Performance and Energy Supervisor core
eCOP Monitor DB
ICT Performance and Energy Supervisor
SLA Renegotiation Controller
Policy Maker
VM Priority Classifier
Optimizer
Energy efficiency Policy Maker & Actuator
Customers

Figure 4-18: The SLA Renegotiation Controller Interactions

Interacting with the above components the SLA Renegotiation Controller is able to continuously monitor the SLAs, whereas it also interacts also the Customer Profiles DB, which maintains info from the SLA instantiation stage on the DC customer contracts. Having collected all necessary 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. These information will be exploited later on by the Optimizer during the implementation of the revised optimization plan. All of the above interactions, during the SLA monitoring, the SLA instantiation and the SLA renegotiation process are detailed in Figure 4-18. This activity will be implemented by employing a Complex Event Processing 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. To elaborate, the SLA renegotiation Controller pinpoints 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, these information are 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.

4.3.6.2 Module decomposition

In order to support all stages of the SLA lifecycle described above, the SLA Renegotiation Controller comprises a number of subcomponents each 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 constituent elements being:

- SLA Template Repository
- SLA Monitor
- SLA Renegotiation Core

Whenever DC customers request the deployment of a new VM the SLA Renegotiation Core will be accessed, which subsequently will interact with the SLA Template Repository to provide information on the available SLA levels (defined as monolithic combination of the SLA parameters described in 4.3.6.1.1), the available virtual resources and the implicit cost of the selected service. The service charges are also determined by 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 registers at one of the provided SLA levels for a certain percentage of time and for a combination of reserved resources at a given price charge, whereas 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. In this direction, 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, to update the SLA parameters. 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, the updated 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 VM requests arrive at the SLA Renegotiation Core, which subsequently interacts with the SLA Template Repository as described above. The internal module architecture is presented in Figure 4-19.

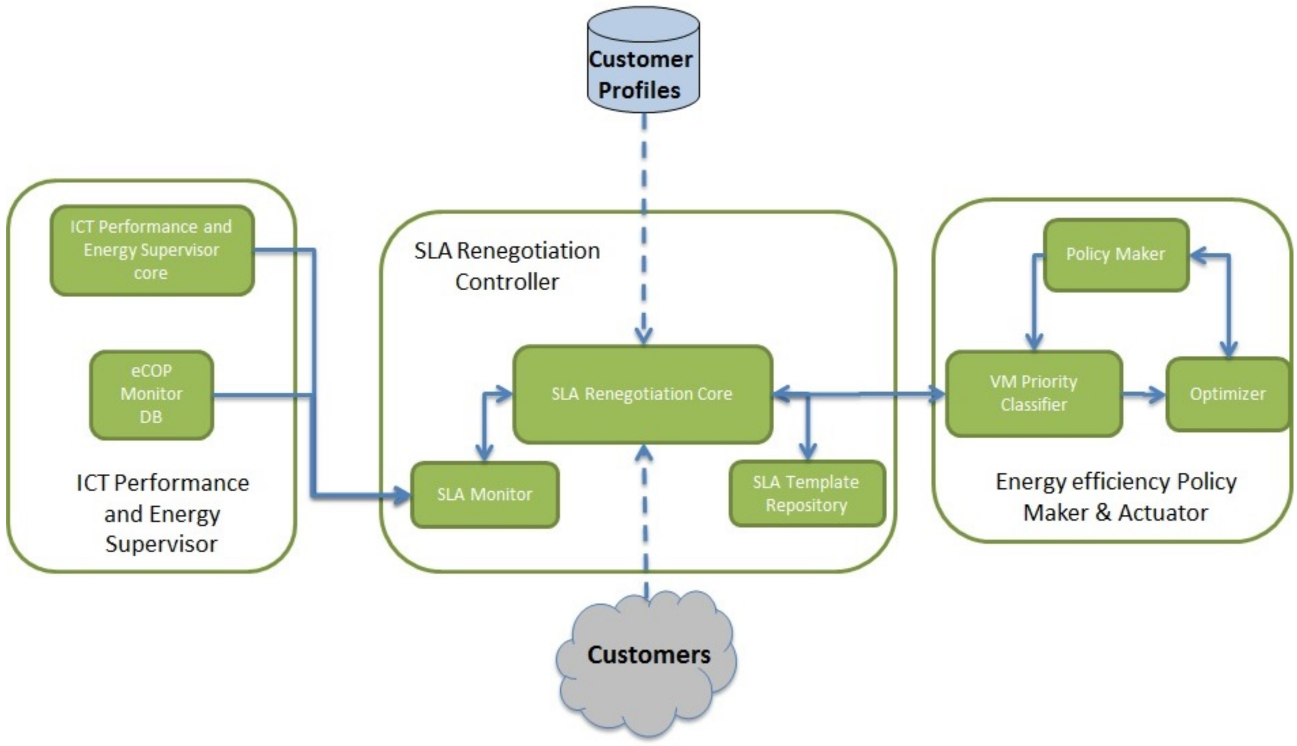

Customer Profiles
ICT Performance and Energy Supervisor core
eCOP Monitor DB
ICT Performance and Energy Supervisor
SLA Renegotiation Controller
SLA Renegotiation Core
SLA Monitor
SLA Template Repository
Policy Maker
VM Priority Classifier
Optimizer
Energy efficiency Policy Maker & Actuator
Customers

Figure 4-19 The SLA Renegotiation Controller Internal Architecture

WINDs' SLA renegotiation process is also presented in Annex 3 as an indicative example, demonstrating the convergence with the DOLFIN renegotiation process described above.

4.3.6.3 References to requirements

This section reports the list of requirements satisfied by this module. The complete list of identified requirements is described in the deliverable D2.2 [1].

Table 4-8: 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).
Q8 MonitorTranslate	The monitoring subsystem shall always translate parameters to metrics (KPI).
Q50 Ask for PostponeExecutionPolicy	The Energy Eff. Policy Maker and Actuator should be able issue a request of the Workload and VM Manager to shift computing load (postpone it for later) when such a need arises, e.g. 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.
Q51 Ask for RenegotiatioPolicy	The Energy Eff. Policy Maker and Actuator should be able issue a request of the Renegotiation Policy (e.g. renegotiation of Workload and VM Manager to shift computing load (postpone it for later) 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 to do so (e.g. by the MgmntSLARenegotiation requirement), the SLA Renegotiation Controller should be able to actually perform the renegotiation

5 Conclusions

This document elaborates on the technical aspects of a synergetic DC architecture and the interoperability of the constituent DCs. It provides the design specification of the DOLFIN DC Software Platform at a Synergetic DC level in the framework of the WP4 “Synergetic DCs for energy efficiency”.

In this course, the document details the architectural design, functionality and the interconnections emerging in an agglomeration of synergetic DCs as well as the inner architecture of the constituent WP4 components. Moreover, the document elaborates on the following pillars of the SDC subsystem:

- architectural design of the SDC DOLFIN software platform,
- interworking interfaces design,
- control flow between the eCOP and SDC subsystems and
- energy models for the DC Software Platform.

In particular, building upon deliverable D3.1 “Data Centre energy consumption optimization platform (eCOP) (Design)”, the control flow between the eCOP and SDC subsystems is presented, demonstrating the interoperability of both subsystems, based on indicative scenarios. Furthermore, the presented control flow between the synergetic DCs - via the cornerstone of the DOLFIN platform, i.e. the eCOP - outlines the actual interconnections at a Synergetic DC level.

Next, the interconnections and interoperability of DCs at SDC level are explicitly described, presenting also the interworking interface design between the SDC components. Moreover, the architectural design and interface design of the constituent SDC subcomponents are described in detail, providing an in depth design specification of the SDC.

Specifically, the SDC - Energy-conscious Synergetic DCs (SDC) subsystem provides a dynamic, service-effective, and energy-efficient allocation of demands, across a distributed network of co-operating DCs. SDC design specification includes:

- SDC functional description - it provides a functional overview of the WP4 components which are enabling a distributed network of co-operating DCs.
- Modules interfaces - The elements of the Synergetic DCs create a distributed infrastructure, and interact with each other. This document provides a high-level view of cross DC interactions, showing the distributed interaction of the Cross-DC.

Internal design for WP4 specific modules: Cross-DC Workload Orchestrator, Cross-DC VM Manager, Cross-DC Monitoring Data Collector, DOLFIN Information Data Base, Smart Grid Controller, SLA Renegotiation Controller and Customer profile management

6 References

[1] 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.

[2] 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.

[3] 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.

[4] Cluster of FP7 Projects Proposes New Environmental Efficiency Metrics for Data Centres, 10 09 2014. [Online]. Available: http://ec.europa.eu/digital-agenda/en/news/cluster-fp7-projects-proposes-new-environmental-efficiency-metrics-data-centres.

[5] D2.1 – Business scenarios and use case analysis, [Online]. Available: http://www.dolfin-fp7.eu/wp-content/uploads/2014/01/DOLFIN_D2.1_IRT_FF_20140207.pdf.

[6] O. J. Kephard and M. D. Chess, "The vision of autonomic computing," *Computer,* vol. 36, no. 1, pp. 41-50, 2003.

[7] M. C. Huebscher and J. A. McCann, "A Survey of Autonomic Computing - Degrees, Models, and Applications," *ACM Comput. Surv.,* vol. 40, no. 3, pp. 7:1-7:28, 2008.

7 Annex 1 – SDC APIs Description

		Class: Server_measurement
Description	This class represents a measurement of a server.	
Attribute name	**Attribute type**	**Description**
value	Double	The measurement value
time	Datetime	The time of the record, in ISO8601 format
serial_number	String	The serial number of the monitored server
type	String	The name of the measurement type

		Class: Metric
Description	This class represents a measurement of a server.	
Attribute name	**Attribute type**	**Description**
name	String	The abbreviated term of the metric name
description	String	An optional description of the metric

				CDC Monitoring Data Collector
Description	The API to send stats to the CDC MDC.			
Endpoint Name	CDC_VMM_Stats			
RPC Call name	CDC_VMM_Stats_update			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Stat_Name	N	String	Name of the measurement.
	Stat_Value	N	Double	Measurement value.
Provides response	200 OK			
Response Parameters	None			
Requester	"ANY"			
Managed errors	**ServerError**	Some wrong parameter was sent.		

	DOLFIN Information DB
Description	The API to request info from the DOLFIN Info DB.
Endpoint Name	InfoDB
RPC Call name	InfoDB_query

Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	InfoDB_Table	N	String	Table to query.
	InfoDB_Query	N	String	Escaped query as a single string.
Provides response	200 OK			
Response Parameters	String [] table rows			
Requester	"ANY"			
Managed errors	**ServerError**	Some wrong parameter was sent.		
	NoData	Query returned no data.		

				CDC Workload Orchestrator Action Response
Description	The API exposed to the Workload Orchestrator to receive commands.			
Endpoint Name	CDC_VMM			
RPC Call name	CDC_VMM_ACTION_RESPONSE			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID.
	Action_ID	N	Intege r	The action to which this response refers.
	ActionError	Y	String	Optional Action Error (null is OK)
Provides response	200 OK			
Response Parameters	None			
Requester	CDC.WO			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				CDC VMM VM Info
Description	The API to receive info relevant to the VMM for a particular VM.			
Endpoint Name	CDC_VMM			
RPC Call name	CDC_VMM_VM_INFO			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	Y	String	The VM ID for some operations.
Provides response	200 OK			
Response Parameters	{String param: String value,}			
Requester	ANY			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				CDC VMM VM Migration Setup
Description	The API exposed to setup the migration URI.			
Endpoint Name	CDC_VMM_Migration			
RPC Call name	CDC_VMM_VM_Migration_URI			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.

	VM_ID	N	String	The VM ID under migration.
	Tunnel_setup	N	String	If necessary, tunnel setup info.
	Migration_Type	N	String	Live or not.
Provides response	200 OK			
Response Parameters	String URI			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				CDC VMM VM Migration Cleanup
Description	The call to finalize the migration process.			
Endpoint Name	CDC_VMM_Migration			
RPC Call name	CDC_VMM_VM_Migration_URI_Cleanup			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID under migration.
	Migration_URI	N	String	The relevant URI to cleanup.
Provides response	200 OK			
Response Parameters	None			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				CDC VMM VM Migration Retry
Description	In case of failure, try to reset the URI.			
Endpoint Name	CDC_VMM_Migration			
RPC Call name	CDC_VMM_VM_Migration_Retry			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID under migration.
Provides response	200 OK			
Response Parameters	String Retry_Possible (if the error is transient)			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				DCO HM VM Info
Description	Get relevant info about the VM.			
Endpoint Name	DCO_HM			
RPC Call name	DCO_HM_VM_Info			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID.
Provides response	200 OK			
Response Parameters	{String param: String value,}			

Requester	ANY	
Managed errors	**ServerError**	Some wrong parameter was sent.

				DCO HM Action Request
Description	The API exposed to the CDC VMM to receive commands.			
Endpoint Name	DCO_HM			
RPC Call name	DCO_HM_ACTION			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	Y	String	The VM ID for some operations.
	Action	N	Integer	The action to be performed.
	ActionParams	Y	String	Optional action parameters.
Provides response	200 OK			
Response Parameters	Action_ID (To Identify the action response)			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				DCO HM Action Response
Description	The API exposed to the Workload Orchestrator to receive commands.			
Endpoint Name	DCO_HM			
RPC Call name	DCO_HM_ACTION_RESPONSE			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID.
	Action_ID	N	Integer	The action to which this response refers.
	ActionError	Y	String	Optional Action Error (null is OK)
Provides response	200 OK			
Response Parameters	None			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				DCO HM VM Migration Setup
Description	The API to initiate setup of the migration.			
Endpoint Name	DCO_HM_Migration			
RPC Call name	DCO_HM_VM_Migration_Setup			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID under migration.
	Tunnel_setup	Y	String	If necessary, tunnel setup info.
	Migration_Type	N	String	Live or not.
Provides response	200 OK			

Response Parameters	None			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				CDC VMM VM Get Migration URI
Description	The API exposed to return the migration URI.			
Endpoint Name	CDC_VMM_Migration			
RPC Call name	CDC_VMM_VM_Migration_Setup			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID under migration.
	Tunnel_setup	Y	String	If necessary, tunnel setup info.
	Migration_URI	N	String	The migration URI.
Provides response	200 OK			
Response Parameters	None			
Requester	DCO.HM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				DCO HM VM Migration Remote Setup
Description	The API to post the remote setup details.			
Endpoint Name	DCO_HM_Migration			
RPC Call name	DCO_HM_VM_Migration_Remote_Setup			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID under migration.
	Tunnel_setup	Y	String	If necessary, tunnel setup info.
	Remote_Migration_URI	N	String	The remote URI.
Provides response	200 OK			
Response Parameters	None			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				DCO HM VM Migration Start
Description	The API to post the remote setup details.			
Endpoint Name	DCO_HM_Migration			
RPC Call name	DCO_HM_VM_Migration_Start			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID to migrate.
Provides response	200 OK			
Response Parameters	String MIGRATION_ACTION_ID			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

	CDC VMM VM Migration Outcome			
Description	The API exposed to return the migration request outcome.			
Endpoint Name	CDC_VMM_Migration			
RPC Call name	CDC_VMM_VM_Migration_Outcome			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID under migration.
	Migration_Outcome	N	String	The error or success.
	Migration_Action_ID	N	String	The attempt to which this outcome refers.
Provides response	200 OK			
Response Parameters	None			
Requester	DCO.HM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

	DCO HM VM Migration Retry			
Description	The API to check if retry is possible (i.e. there is a transient error).			
Endpoint Name	DCO_HM_Migration			
RPC Call name	DCO_HM_VM_Migration_Retry			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID under migration.
	Migration_Action_ID	N	String	The migration attempt ID.
	Migration_Outcome	Y	String	Optionally, the error.
Provides response	200 OK			
Response Parameters	String RETRY_POSSIBLE			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

	DCO HM VM Migration Teardown			
Description	The API to clean up after the operation or cancel it.			
Endpoint Name	DCO_HM_Migration			
RPC Call name	DCO_HM_VM_Migration_Teardown			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	Tenant_ID	N	String	The requesting user.
	VM_ID	N	String	The VM ID under migration.
	Migration_Action_ID	N	String	The migration attempt ID.
Provides response	200 OK			
Response Parameters	None			
Requester	CDC.VMM			
Managed errors	**ServerError**	Some wrong parameter was sent.		

				DOLFIN Info DB IEI
Description	The API to provide metrics to the DOLFIN Info DB.			
Endpoint Name	InfoDB_data_exchange			
RPC Call name	InfoDB_ data_exchange			
Request Parameters	Parameter	Optional (Y/N)	Type	Description
	InfoDB_Metric	N	String	String representation of metric.
	InfoDB_Timestamp	N	String	The time of the metric, in ISO8601 format
	InfoDB_Value	N	Double	The metric's value.
Provides response	200 OK			
Response Parameters	None			
Requester	CDC MDC			
Managed errors	ServerError	Some wrong parameter was sent.		

				DOLFIN Info DB IMI
Description	The API to change the DOLFIN Info DB parameters.			
Endpoint Name	InfoDB_management			
RPC Call name	InfoDB_ management			
Request Parameters	**Parameter**	**Optional (Y/N)**	**Type**	**Description**
	InfoDB_mgmt_query	N	String	String representation of the management query.
Provides response	200 OK			
Response Parameters	None			
Requester	CDC MDC			
Managed errors	**ServerError**	Some wrong parameter was sent.		

8 Annex 2 – VLSP APIs Description

This section presents an architectural view of the Very Lightweight Software Driven Network and Services Platform (VLSP). It is often combined with the VIS to form the VLSI - the Very Lightweight Software-Defined Infrastructure. We describe the main components of the VLSP, and also give a description of the all of the relevant interfaces which are used to interact with the VLSP.

All of the API calls of the interface are RESTful, passing in a resource id if required, and arguments if required. All responses are in JSON format. Each call is shown together with the HTTP method required to activate the relevant functionality. If a URI is passed in, but the HTTP method is incorrect, then an error status will be returned.

8.1 Component Description

Firstly, there is the Component Description which describes the elements of VLSP.

8.1.1 Component Architecture

The VLSI - the Very Lightweight Software-Defined Infrastructure is made up from the VLSP and the VIS.

The Very Lightweight Software Driven Network and Services Platform (VLSP) includes a new lightweight network hypervisor, a novel infrastructure for the management and manipulation of virtual networks on top of the hypervisor and facilities for handling information / knowledge in the SDN environment.

The VLSI software stack consists of three layers, and n architectural overview of the VLSI software stack is shown is shown in Figure 8-1. Two layers are from the VLSP, namely

- the Lightweight Network Hypervisor (in green),
- the Virtual Infrastructure Management (in orange), and
- the Virtual Infrastructure Information Service (VIS) (in blue).

This section focuses on the VLSP, which is a software test-bed that consists of over 700 java classes and more than 150 k-lines of code. We detail the two main layers of the VLSP below.

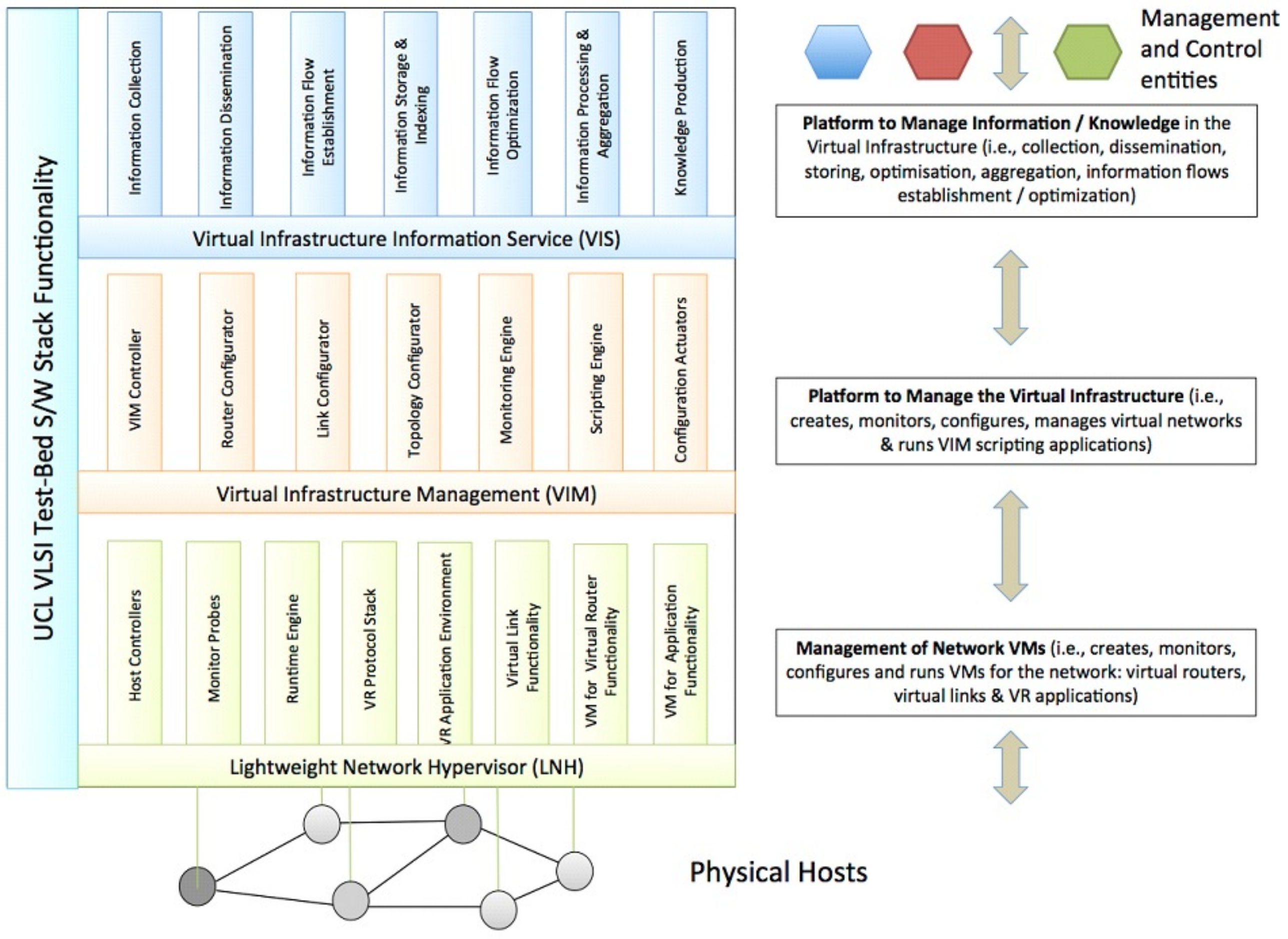

UCL VLSI Test-Bed S/W Stack Functionality
Information Collection
Information Dissemination
Information Flow Establishment
Information Storage & Indexing
Information Flow Optimization
Information Processing & Aggregation
Knowledge Production
Virtual Infrastructure Information Service (VIS)
VIM Controller
Router Configurator
Link Configurator
Topology Configurator
Monitoring Engine
Scripting Engine
Configuration Actuators
Virtual Infrastructure Management (VIM)
Host Controllers
Monitor Probes
Runtime Engine
VR Protocol Stack
VR Application Environment
Virtual Link Functionality
VM for Virtual Router Functionality
VM for Application Functionality
Lightweight Network Hypervisor (LNH)
Management and Control entities
Platform to Manage Information / Knowledge in the Virtual Infrastructure (i.e., collection, dissemination, storing, optimisation, aggregation, information flows establishment / optimization)
Platform to Manage the Virtual Infrastructure (i.e., creates, monitors, configures, manages virtual networks & runs VIM scripting applications)
Management of Network VMs (i.e., creates, monitors, configures and runs VMs for the network: virtual routers, virtual links & VR applications)
Physical Hosts

Figure 8-1: Overview of VLSI Software Stack

8.1.2 The Lightweight Network Hypervisor (LNH)

The Lightweight Network Hypervisor includes a fully operational lightweight virtual router (VR) combined with virtual network connectivity. These elements can be combined in order to build any network topology required. The created virtual network is designed with the goal of transmitting and routing datagrams from any source to any destination. It behaves like a lightweight virtual network, but it has management facilities to start and stop virtual routers on-the-fly, together with the ability to create and tear down network connections between virtual routers dynamically. Furthermore, these lightweight routers have an application layer interface that provides the capability to start and stop Java software applications. These applications use a transport protocol API which can send and receive datagrams or packets, and thus act as the service elements within the platform.

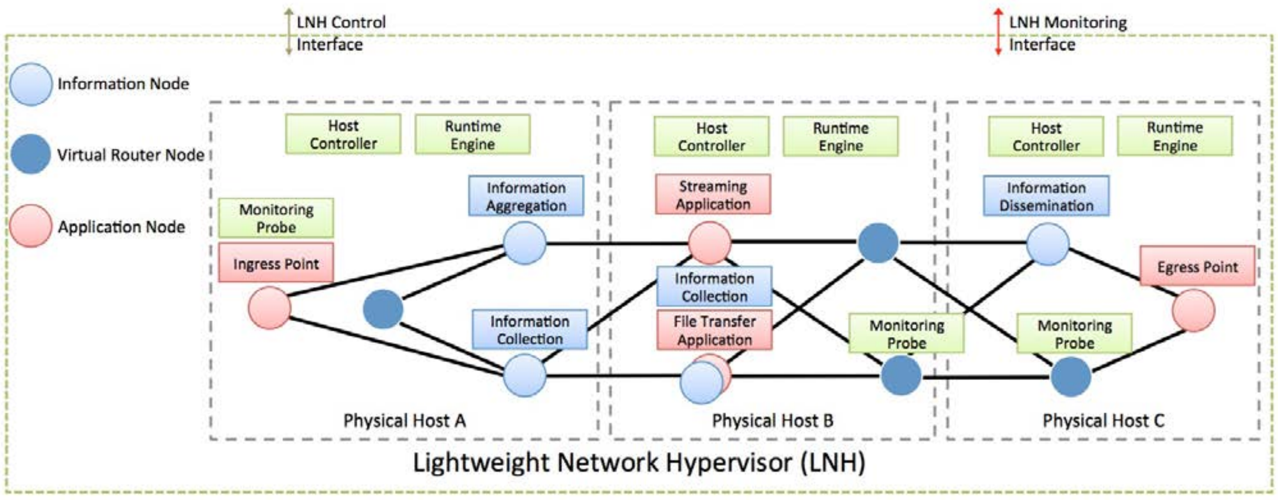


Figure 8-2: The Lightweight Network Hypervisor

The Host Controllers (shown in green in Figure 8-2 are executed on every machine that can host virtual routers. Their main job is to actually start a virtual router, stop a virtual router, start a virtual link, and stop a virtual link. Other tasks undertaken by the Host Controllers are to configure the routers once they have started, or to pass on commands to specific virtual routers, as needed.

The virtual network topology consists of virtual routers (shown as coloured circles) and virtual links (show in black). Each virtual router is instrumented with the VLSP monitoring system in order to gather data on each of the network interfaces of each virtual router. The data includes information on traffic volumes coming in and going out of each interface. The monitoring system collects the raw data and passes in onto the Monitoring Engine function of the above layer.

The main LNH elements can be found in the following table.

Table 8-1: The main LNH elements

Name	Description
Host Controllers	The host controllers execute on every physical machine and manipulate & configure virtual routers, links and virtual router applications.
Monitor Probes	The monitor probes are tiny configurable applications probing the software or hardware for monitoring data.
Runtime Engine	It is responsible for the runtime operation of the LNH, including support for event-based notifications and time scheduling.
Virtual Router Protocol Stack	The lightweight network protocol stack of the VRs.
Virtual Router Application Environment	The application environment that hosts VR applications.
Virtual Link Functionality	The functionality of the virtual links, including link weighting and other configuration options.
Virtual Machine for Virtual Router & Application Functionality	A virtual machine with the virtual router and the relevant applications functionality.

8.1.3 The Virtual Infrastructure Management (VIM)

In this section we describe the Virtual Infrastructure Management component, highlighting its purpose and its architecture. The Virtual Infrastructure Management (VIM) component is responsible for the management and lifecycle of the virtualized elements that will be used within a network, particularly virtual network elements. As the virtual elements are not physical themselves, but exist on top of physical elements, their lifecycle and their management needs to be approached carefully to ensure continued operation and consistency. An overview of the VIM architecture on top of the LNH is shown in Figure 8-3.

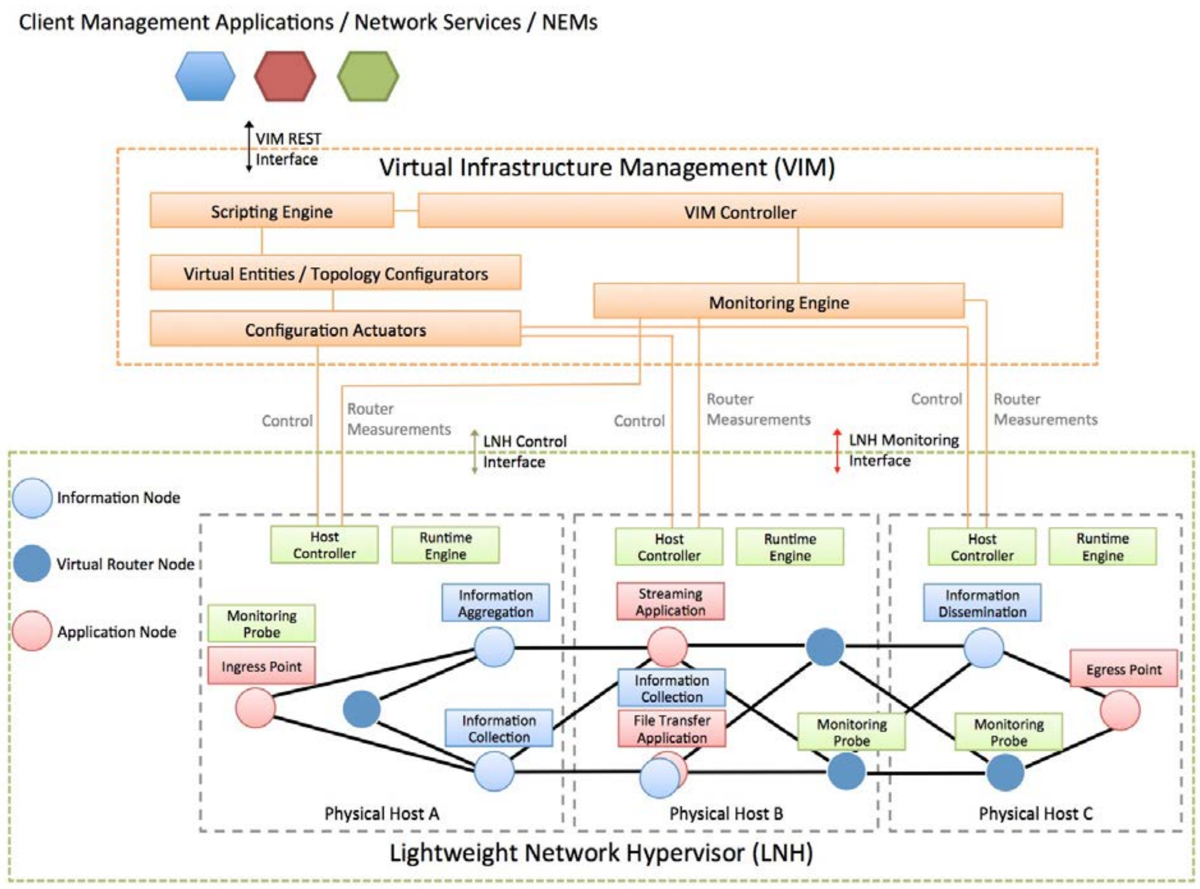


Figure 8-3: The Virtual Infrastructure Management Component

The virtual network elements, which exist on top of physical networks, can be setup with arbitrary topologies and with an arbitrary number of end-points. The virtual links in a virtual topology are eventually mapped onto physical links in the underlying network. A virtual link may span multiple physical links, and cross many physical routers, or it may span a single physical network link. New virtual links can be added or can be removed from a virtual network dynamically at run-time.

The virtual networks are very flexible and adaptable, and generally have few limitations, except that a virtual link cannot support more traffic and higher-data rates than the underlying physical links. Furthermore, a whole virtual network can be shutdown as needed, if the applications that use it no longer need the network. Such a shutdown frees resources from the underlying physical network.

The full management of virtual networks on physical networks requires the matching and analysis of the flow rates on the virtual links to the flow rates of the underlying physical links. It is important to ensure that the physical links are not congested with too many virtual links. Also, the allocation and mapping of virtual links must take into account the current state of the physical network and the current virtual networks.

However, if a situation arises where a re-configuration is required, the virtual network management should be capable of mapping a virtual link across different physical links at run-time, but leave the virtual topology intact.

The VIM component has a seemingly simple task, but in reality the management requires continual monitoring, analysis, and adaption of the virtual elements to the physical elements. As all of these virtual elements are distributed, the management is a complex task.

The diagram in Figure 8-3 shows how the VIM component interacts with the virtual network elements that will be present in a running virtual network. All of the elements of the VIM component constitute a fully distributed system, whereby an element or node can reside on any host. A full virtual network can be instantiated on a single machine, for demonstration or testing purposes, or instantiated across multiple servers, in a full deployment situation.

The VIM directly controls the lifecycle of each virtual element, by collecting knowledge on the status of physical resources in order to determine where a virtual element can be created. The virtual network element will be created, managed, and shutdown by lifecycle phase of this component.

Due to the dynamic nature of virtual elements and because they can be disassociated from the physical elements they are mapped to, it is possible to do a live adaption of a virtual element from one physical host to another physical one, at run-time.

The VIM controller acts as a control point for managing the virtual elements. This block accepts all its input via the VIM REST interface from other management applications / network services. The monitoring engine acts a collection point for the monitoring data needed to keep the management functions running. Control commands are being sent to the VIM and they are either acted upon immediately or are passed to the corresponding Host Controllers of the LNH.

The main VIM elements can be found in the following table.

Name	Description
VIM Controller	It is the heart of the component, providing the central control of the VIM operations.
Scripting Engine	VIM can be configured using Closure scripting.
Monitoring Engine	It is the main monitoring component of the infrastructure, i.e., collecting & manipulating measurements from the monitoring probes residing at the LNH.
Virtual Entities / Topology Configurators	These functions are responsible for the configuration of virtual routers, links and topologies, supporting different levels of abstraction.
Configuration Actuators	The Virtual Entities / Topology configurators communicate with the configuration actuators which in turn enforce the configuration changes through the LNH's host controllers.

Table 2 - The main VIM elements

8.2 API Calls

The Virtual Infrastructure Manager is the element that has access from other components in the DOLFIN. The Lightweight Network Hypervisor is not publically accesssible, and is there to support the Virtual Infrastructure Manager in its role of managing the lifecycle of Virtual Routers and Virtual Compute nodes. As such, we will only document the interface of the Virtual Infrastructure Manager, as it is the only element that can be addressed directly.

The resources that are addressable via the Virtual Infrastructure Manager are:

- Routers
- Links
- Applications

The API interface is grouped into API calls for:

- **Routers:** This group of REST calls handles the Router resources
- **Links:** This group of REST calls for Link resources in a managed network.
- **Links on a Router:** This group of REST calls handles the Link resources attached to a specified Router.
- **Applications on a Router:** This group of REST calls handles the Application resources on a specific router
- **Router Link Stats:** This group of REST calls handles the link stats for Link resources attached to a specified Router.
- **Shutdown for Routers:** This group of REST calls getting info about Routers that have shut down.

The next table summarizes the REST calls.

Process	HTTP Method	URI	Args	Response	Example
Router					
Create Router	POST	http://host:10000/router/?args	[name] [address]	routerID name address mgmtPort r2rPort time	{"routerID":1, "address":"1", "name":"Router-1", "mgmtPort":11001, "r2rPort":11002, "time":1362054061000}
Delete Router	DELETE	http://host:10000/router/2	*router ID*	routerID status	{"routerID": 10, "status": "deleted"}
List All Routers	GET	http://host:10000/router/	[detail] = id | all	type: “router” list: [1, 2, 4, 7]	{ “type”: “router”, “list”: [1, 2, 4, 7]}
Get Router Info	GET	http://host:10000/router/2	*router ID*	routerID name address links mgmtPort r2rPort time	{"routerID": 2, "address": "2", "name":"Router-2", "links": [1], "mgmtPort": 11003, "r2rPort": 11004, "time": 1361817254727}
Get No Of Routers	GET	http://host:10000/router/count		Value	{ “value”: 5 }
Get Max Router ID	GET	http://host:10000/router/maxid		Value	{ “value”: 42 }
Link					
Create Link	POST	http://host:10000/link/?args	router1 router2 [weight] [linkName]	linked linkName weight nodes time	{"linkID": 196612, "linkName": "Router-1.Connection-0", "nodes": [1,2], "time":1362079103160,"weight":10}

Process	HTTP Method	URI	Args	Response	Example
Delete Link	DELETE	http://host:10000/link/7		linkID status	{"linkID": 196612, "status": "deleted"}

List All Links	GET	http://host:10000/link/	[detail] = id | all	type: “link”
				list: [8, 12, 14, 27]
					{ “type”: “link”, “list”: [1, 2, 5]}
					
					Where ‘detail=all’
					
					{"type":"link", "list":[2293796,196612,1835044,1179673,655376,262153], "detail": [{"id":2293796,"name":"Router-5.Connection-0","nodes":[5,6], "time":1361989254649,"weight":10}, {"id":196612,"name":"Router-1.Connection-0","nodes":[1,2], "time":1361989254649,"weight":10}, {"id":1835044,"name":"Router-4.Connection-0","nodes":[4,6], "time":1361989254649,"weight":10}, {"id":1179673,"name":"Router-3.Connection-0","nodes":[3,5], "time":1361989254649,"weight":10}, {"id":655376,"name":"Router-2.Connection-0","nodes":[2,4], "time":1361989254649,"weight":10}, {"id":262153,"name":"Router-1.Connection-1","nodes":[1,3], "time":1361989254649,"weight":10}}],
					}

Get Link Info	GET	http://host:10000/link/8	*link ID*	linkID linkName weight nodes time	{"linkID":8, "linkName":"Router-5.Connection-0", "weight":10, "nodes":[5,6], "time":1362079709109}
Set Link Weight	PUT	http://host:10000/link/8?args	weight	inkID linkName weight nodes time	{"linkID":8, "linkName":"Router-5.Connection-0", "weight":30, "nodes":[5,6], "time":1362079709109}

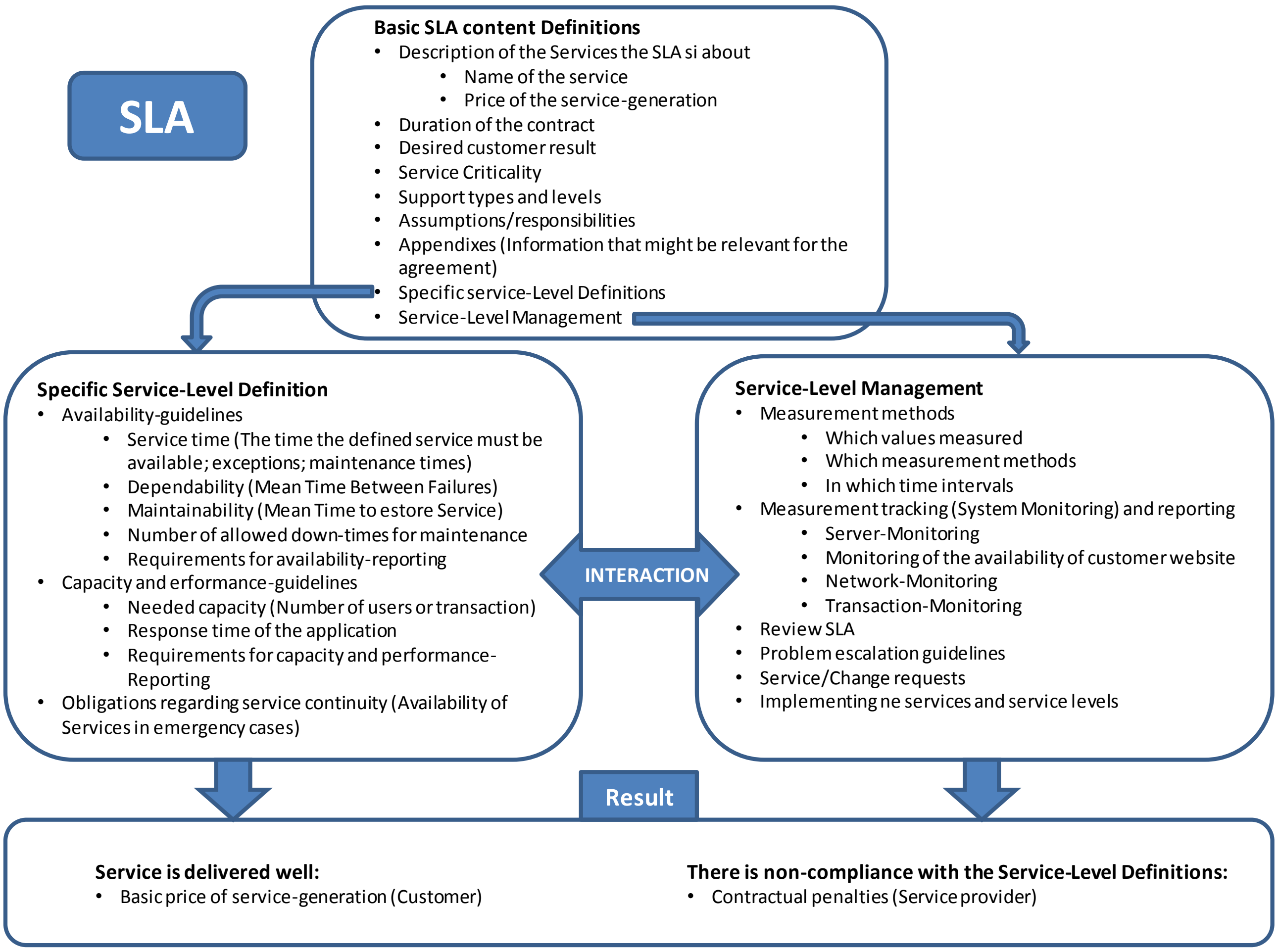
Process	HTTP Method	URI	Args	Response	Example
Get No Of Links	GET	http://host:10000/link/count		value	{ “value”: 17 }
Links on a Router					
List Router Links	GET	http://host:10000/router/9/link/	[attr] = id | name | weight | connected	routerID type list	routerID: 9 type: “link” list: [12, 14]
Get Router Link Info	GET	http://host:10000/router/9/link/12	*router ID*	linkID linkName weight nodes time	{"linkID":12, "linkName":"Router-9.Connection-0","weight":10 "nodes":[9,10],"time":1362080281512}
Applications on a Router					
Create App	POST	http://host:10000/router/9/app/?args	routerID className args	appID appName classname args routerID starttime runtime	{"appID":2, "appName": "/Router-7/App/usr.applications.Send/1", "classname": "usr.applications.Send", "args": "[8, 4000, 2500000, -s, 1024]", "routerID":7, "starttime":1362090555686,"runtime":0}
List Apps	GET	http://host:10000/router/9/app/		type: “app” list: [2, 3]	{ “type”: “app”, “list”: [2, 3]}
Get App Info	GET	http://host:10000/router/9/app/2	*app ID*	appID appName classname args routerID starttime runtime	{"appID":2, "appName":"/Router-7/App/usr.applications.Send/1", "classname": "usr.applications.Send", "args": "[8, 4000, 2500000, -s, 1024]", "routerID": 7, "starttime": 1362090555686, "runtime": 42854}

Process	HTTP Method	URI	Args	Response	Example
Router Link Stats					
Get Router Link Stats	GET	http://host:10000/router/9/link_stats	*Router id and 'link_stats'.* *None.*	*value*	{"type": "link_stats", "links": [12 30 104], "link_stats": [["Router-12 Router-7.Connection-2" 358050 1714 0 21 337824 1656 228431 1078 0 0 205428 1007 0 1 0 1], ["Router-30 Router-7.Connection-7" 155976 715 0 5 136884 671 73858 302 0 0 53040 260 0 1 0 2], ["Router-104 Router-7.Connection-19" 50203 232 0 0 45900 225 2067 3 0 0 0 0 0 1 0 1]], "routerID": 7}
Get Router Link Stats	GET	http://host:10000/router/9/link_stats/12	*Router id 'link_stats'] dst router id*	*value*	{"type": "link_stats", "links": [12], "link_stats": [["Router-12 Router-7.Connection-2" 358050 1714 0 21 337824 1656 228431 1078 0 0 205428 1007 0 1 0 1]], "routerID": 7}
Shutdown for Routers					
List All Shutdown Routers	GET	http://host:10000/removed/	*None*	*type list*	{"type":"shutdown", "list": [{"routerID":1,"time":1362133396696}, {"routerID":2,"time":1362133396696}, {"routerID":3,"time":1362133396696}]}

9 Annex 3 – 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 1980ies 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 is shown in the following picture:



Basic SLA content Definitions
- Description of the Services the SLA si about
    - Name of the service
    - Price of the service-generation
- Duration of the contract
- Desired customer result
- Service Criticality
- Support types and levels
- Assumptions/responsibilities
- Appendixes (Information that might be relevant for the agreement)
- Specific service-Level Definitions
- Service-Level Management

Specific Service-Level Definition
- Availability-guidelines
    - Service time (The time the defined service must be available; exceptions; maintenance times)
    - Dependability (Mean Time Between Failures)
    - Maintainability (Mean Time to estore Service)
    - Number of allowed down-times for maintenance
    - Requirements for availability-reporting
- Capacity and erformance-guidelines
    - Needed capacity (Number of users or transaction)
    - Response time of the application
    - Requirements for capacity and performance-Reporting
- Obligations regarding service continuity (Availability of Services in emergency cases)

Service-Level Management
- Measurement methods
    - Which values measured
    - Which measurement methods
    - In which time intervals
- Measurement tracking (System Monitoring) and reporting
    - Server-Monitoring
    - Monitoring of the availability of customer website
    - Network-Monitoring
    - Transaction-Monitoring
- Review SLA
- Problem escalation guidelines
- Service/Change requests
- Implementing ne services and service levels

Service is delivered well:
- Basic price of service-generation (Customer)

There is non-compliance with the Service-Level Definitions:
- Contractual penalties (Service provider)

Figure 9-1: SLA Components

1. Description of the Services the SLA is about: 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

9.1 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,
- and 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.

9.2 Vision for a WIND Data Centre Federation

9.2.1 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.

9.2.2 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.

9.2.3 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 lesser 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 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.

9.3 Wind Data Centre Service Level Agreement

Below a typical Wind Datacenter Service Level Agreement is exemplified.

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.