# Autonomic Computing to Manage Green Core Networks with Quality of Service

Remi Sharrock[1(✉)], Thierry Monteil[2,4], Patricia Stolf[3,4], and Olivier Brun[2]

[1] Institut Mines-Telecom, Telecom ParisTech, CNRS LTCI UMR 5141, Paris, France
remi.sharrock@telecom-paristech.fr
[2] CNRS, LAAS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4, France
[3] IRIT, 118 Route de Narbonne, 31062 Toulouse Cedex 9, France
[4] UPS, INSA, INP, ISAE; UT1, UTM, LAAS, Université de Toulouse, 31077 Toulouse Cedex 4, France

**Abstract.** In a context where data and computing services are moving to external specialized datacenters the manual management of these systems is becoming an issue. Human administrators have to deal with hardware resources optimization while meeting the users' needs. In our approach we propose to reconfigure both a set of applications deployed in a datacenter (by adapting their behaviors using autonomic computing) and the wired network (by switching on and off its equipments like routers, modules). We take into account both the energetic costs with the network equipments and the quality of service provided to the end user by the deployed applications. The main contribution of the proposed model is to consider a compromise between the total power consumption of the network equipments and the application quality of service. We validated our approach by simulating deployed applications on the Grid'Mip infrastructure similar to a small core network made up of Cisco routers (part of Grid'5000 project).

## 1 Introduction

Data and computing services outsourcing (Web sites, Databases, Distributed applications) towards specialized datacenters is increasingly selected as a way to delegate complex infrastructure management to experts. Some of these datacenters tend to offer cloud computing services which allow access to pre-configured platforms or software environments, sometimes in a matter of seconds. From the datacenter customer point of view, these services allow to adapt the application architecture dynamically, for example according to the fluctuating incoming request flows. From the datacenter's administrator point of view, the mix of customers' needs and deployed application's needs may require an optimization of the hardware resources usage, for example to reduce the electrical power consumption of the datacenter. Thus, there is a compromise to be found between, on one hand, the quantity of resources (power) that are needed for a given service and on the other hand the quality (performance) of this service.

In our approach, we tend to introduce the general problem of self-optimization applied at both application (performance optimization) and network infrastructure levels (power optimization). We introduce a criterion which is a function of the global electric power consumed by the network and of a parameter representing the loss of service quality for the applications. The problem of optimization consists in minimizing this loss and/or the power consumption depending on the administrator's choice. This minimization takes into account a number of constraints such as the dynamic topology (links shutdowns, network equipments or nodes shutdowns), the network links capacities and the routing policies.
The rest of the paper is organized as follows. Related work is first discussed in Sect. 1.1. Then the environment considered by our approach is presented in Sect. 1.2. Section 2 presents the model proposed including the network, the application, the routing protocol and the criterion used. Results of our experimental evaluation are presented in Sect. 3 , then finally, we conclude in Sect. 4.

### 1.1 Related Work

**Quality of Service characteristics** The QoS characteristics are measurable values and a set of constraints on these values. They are mainly classified in the following groups [1]:

- **Temporal**: transit time, response time, time of establishment, jit
- **Capacity**: entrance/exit flow rate, user data rate, data control rate
- **Integrity**: connection breakdown probability, loss probability
- **Security**: protection level, authentication

**Difference between QoS characteristic, profile and needs** It is important to first differentiate the **QoS profile** of an application (its multiple behaviors) and the **QoS characteristics**. Indeed, the first one has an influence on the other. The QoS profile is a set of internal values for the application which influences the QoS characteristics. For example, a video application may have two possible types of compression for the images [2], one using the BZIP2 algorithm and the other Lempel-Ziv. The internal parameter is the compression algorithm which influences the transmission time because both algorithms have different compression times. We can thus say that the QoS profile of the application is described by the compression algorithm (two behaviors possible) and the influenced QoS characteristics is the transmission time. In this case, by using the BZIP2 algorithm, the application has a transmission time of 20 ms, whereas with the algorithm Lempel-Ziv the transmission time is 40 ms. Finally, **QoS needs** are a set of predefined QoS characteristics needed by the application. Depending on its different QoS profiles, the application may have different QoS needs.

A self-adaptive application would choose automatically its QoS profile so as to satisfy the QoS needs, which is a complex optimization problem by itself. In our approach, we introduce a mathematical model which takes into account those QoS needs. One part of the optimization consists in selecting which QoS needs will satisfy at best the global minimization problem.

**Applications' QoS management capacity** We consider two cases for the applications' QoS management capacity: either the application has multiple QoS profiles (multiple behaviors) or it has absolutely no notion of QoS (no QoS profile at all and only one behavior).

**Application without notion of QoS:** for an application without notion of QoS, it cannot change its functioning neither at the beginning, nor during the execution.

**Applications with multiple QoS profiles:** the application has the capacity to modify its behavior and is able to send/receive different traffics (for example, compression of different data). In this case, either the application is self-adaptive and has internal mechanisms to choose its QoS profiles or the application is external-adaptive and rely on external mechanisms to choose its QoS profile.

For our approach, we consider external-adaptive applications with multiple QoS profiles. Indeed, the change of QoS profile (the reconfiguration) is made by an autonomic tool according to the results of the optimization. This allows to make reconfiguration decisions at the global level, by taking into account all the applications and the datacenter network infrastructure.

### 1.2 Environment Considered by the Approach

**The DiffServ domain** Our approach considers one DiffServ domain[3]. The DiffServ model is implemented in a DiffServ domain (DS Domain) which corresponds to a zone having a common QoS policy, usually true for datacenter network. Indeed, a datacenter is for the most part managed by a single administrative entity which integrates the network into a DS domain belonging to the entity. Our approach proposes to use a single optimization tool for the DS Domain.

**The routing of the network** Usually, the machines of a datacenter are grouped in clusters. A cluster can be connected physically to several routers. This means that all the machines of the cluster are connected with several routers and thus have several network cards. Our approach considers only the case of the dynamic routing tables and does not take into account the case of Channel Bonding[4] (aggregation of several network interfaces in a logical interface).

Also, for the border routers and core routers, the routing is either static, or it depends on the routing protocol used inside the DS domain, meaning that it depends on the IGP (Interior Gateway protocol). The roles of an IGP are:

- to establish the optimal routes between a point of the network and all the destinations available of a bounded domain;
- to avoid buckles;
- in case of modification of the topology (disconnection of a physical link, a router's shutdown), to guarantee the convergence of the network, that is the restoring of it's optimal connectivity without buckle as soon as possible.

We distinguish usually:

- link state protocols which establish neighborhood tables and use the Dijks algorithm to calculate the best routes. Two examples of such protocols are *IS* (Intermediate system to intermediate system) [5], *OSPF* (Open Shortest Path First) [6];
- distance vector protocols: *RIP* (Routing Information Protocol), *IGRP* (Interior Gateway Routing Protocol) [7];
- The hybrid protocols, which have characteristics of both first ones: *EIGRP* (Enhanced Interior Gateway Routing Protocol) [8].

The inclusion of QoS and network energy consumption can be done at the protocol level. [9] proposes the use of energy Efficient Ethernet standard (IEEE 803.3az) by adding a prioritization of streams impacted by the saving energy mechanisms (sleeping mode, coalescing mechanism) to ensure the desired level of QoS. In [10], is proposed a change in the OSPF that allows (depending on the communication links usage) to reconfigure the routing tables and to turn off routers. In [11], the authors present some detailed router consumptions and a generic router consumption model. Studies specifically on datacenter and core networks have been made in [12]. They are interested in the location of datacenters from the data access point of view under network energy consumption. The authors also define classes of popularity data leading them to address the problem of data replication. There is a linear programming formulation to find a solution.

Our approach considers the case of various routing protocols. We give a first example of heuristics with a dynamic routing table generated by the OSPF routing protocol. The choice of OSPF as routing protocol has for consequence to create routing tables following a metric defined on the links. For example, the cost of routes can be calculated according to the capacities of all the links of the route. A load balancing is made on routes having the same cost and routes having a higher cost are not used. It is then possible to switch off some router links or some routers.

## 2 Model

### 2.1 Network Model

**Variables and functions for the network representation** The topology of the network is expressed with an oriented graph $G = \{\mathbf{N}, \mathbf{E}\}$ where $\mathbf{N}$ is the set of nodes of one network domain and $\mathbf{E}$ is the set of edges. A couple $(i, j)$ represents the edge between the node $i$ and the node $j$. A node can be a border router, a core router or a host. A router has different modules plugged into his frame. Each module contains several network ports that can be used to create links between the nodes. We consider that the hosts have only one module but possibly multiple network ports.

- $\mid \mathbf{N} \mid = n_N$ and $\mid \mathbf{E} \mid = n_E$
- $c^e_{i,j}$ is the capacity of the edge $(i, j)$

- $p^e_{i,j}$ is the electric power consumption of the port in the node $i$ used to create the link between the node $i$ and the node $j$
- $z^e_{i,j}$ defines the state of the port in the node $i$ used for the link $(i,j)$, 1 means the port is switched on and 0 is switched off
- $p^m_{i,l}$ is the electric power consumption of the module $l$ of the node $i$
- $z^m_{i,l}$ defines the state of the module $l$ of the node $i$ (same convention as for $z^e_{i,j}$)
- $p^c_i$ is the electric power consumption of the node $i$ when all modules and ports are switched off (also called the frame consumption)
- $z^c_i$ defines the state of the node $i$ (same convention as for $z^e_{i,j}$)
- $\mathbf{M_i}$ is the set of the modules in the node $i$ :
  $\mathbf{M_i} = \{m_1, \dots m_l \dots, m_{n_{M_i}}\}, \mid \mathbf{M_i} \mid = n_{M_i}$
- $\mathbf{E_{i,l}}$ is the set of ports of the module $l$ of the node $i$
- $\mathbf{E_i}$ is the set of all the ports of the node $i$, $\forall i \in \mathbf{N}$ :
  $\mathbf{E_i} = \bigcup_{l \in [1, n_{M_i}]} \mathbf{E_{i,l}}$

**Relations and constraints for the network** The total electric power consumption of the network $P_{total}$ is:

$$P_{total} = \sum_{i \in \mathbf{N}} \{p^c_i.z^c_i + \sum_{l \in [1, n_{M_i}]} [p^m_{i,l}.z^m_{i,l} + \sum_{j \in \mathbf{E_{i,l}}} p^e_{i,j}.z^e_{i,j}]\} \tag{1}$$

$$= \sum_{i \in \mathbf{N}} p^c_i.z^c_i + \sum_{i \in \mathbf{N}, l \in [1, n_{M_i}]} p^m_{i,l}.z^m_{i,l} + \sum_{i \in \mathbf{N}, j \in \mathbf{E_i}} p^e_{i,j}.z^e_{i,j} \tag{2}$$

If all ports of a module are switched off the module can also be switched off:

$$\forall i \in \mathbf{N}, \forall l \in [1, n_{M_i}] : \sum_{(i,j) \in E_{i,l}} z^e_{i,j} = 0 \Rightarrow z^m_{i,l} = 0 \tag{3}$$

Because it is easier to solve a linear problem the relation (3) could be expressed as two linear constraints:

$$\forall i \in \mathbf{N}, \forall l \in [1, n_{M_i}] : z^m_{i,l} - \sum_{j \in \mathbf{E_{i,l}}} z^e_{i,j} \leq 0 \tag{4}$$

$$\forall i \in \mathbf{N}, \forall l \in [1, n_{M_i}], j \in \mathbf{E_{i,l}} : z^e_{i,j} - z^m_{i,l} \leq 0 \tag{5}$$

The same linear constraints can also be written for the nodes. If all modules of a node are switched off the node can also be switched off:

$$\forall i \in \mathbf{N} : z^c_i - \sum_{l \in [1, n_{M_i}]} z^m_{i,l} \leq 0 \tag{6}$$

$$\forall i \in \mathbf{N}, l \in [1, n_{M_i}] : z^m_{i,l} - z^c_i \leq 0 \tag{7}$$

There is a symmetry when two ports are connected, when one is switched off then the connected one is also switched off:

$$\forall i, j \in \mathbf{N} : z^e_{i,j} - z^e_{j,i} = 0 \tag{8}$$

### 2.2 Application Model

**Variables and functions for the applications** We consider one-to-one applications that are composed of one sender and one receiver. This will simplify the traffic matrix, the final notation and the number of unknown values. Yet the generalization of more complex applications is possible for the model used. The following variables are defined:

- $A$ is the set of applications on the datacenter:
  $\mathbf{A} = \{a_1, \dots a_k \dots, a_{n_A}\}, \mid \mathbf{A} \mid = n_A$
- $a_k^s$ is the sender process of the application $a_k$ and $a_k^r$ the receiver (generalisation with several receiver is possible). We suppose that the communication is one-way and we neglected back traffic (signaling, acknowledgements, etc)
- $N_s^k$ is the host of $a_k^s$ and $N_r^k$ the host of $a_k^r$
- $\mathbf{NE}$ is the set of QoS needs for all applications. Every element of this set is composed of a set of values expressing an elementary QoS characteristic asked by the application (flow, jit, response time, ...) which are grouped together in a tuple. $\mathbf{NE}$ is composed of numerical values, interval or other representations allowing to characterize an elementary need in QoS $\mathbf{NE} = \{n_1, \dots n_s \dots, n_{n_{NE}}\}, \mid \mathbf{NE} \mid = n_{NE}$
- $\mathbf{NE^k}$ allows to specify for an application $a_k$ the various possible needs for this application.
- $\mathbf{B^k}$ is a set which has the same size of $\mathbf{NE^k}$:
  $\mathbf{B^k} = \{b_1^k, \dots b_n^k \dots, b_{n_{B^k}}^k\}, \mid \mathbf{B^k} \mid = n_B^k$. It is composed of binary variables $b_n^k \in \mathbf{B}^k$ :
  $$b_n^k = \begin{cases} 1 \text{ if the need } n_s^k (s = n) \text{ is chosen for the} \\ \quad \text{application } a_k, \\ 0 \text{ otherwise} \end{cases}$$
- We suppose that there is a metric function called $M$ allowing to measure the quality of a need. This last one allows to define a total order relation noted $<$ between the various needs of an application. To simplify afterward the notation, we suppose that needs are altogether tidied up $\mathbf{NE^k}$ following this order $<$ using the metric $M$ (this is always achievable with an index permutation):
  $$M(n_1^k) < M(n_2^k) < \ldots < M(n_{n_{NE^k}}^k)$$
- $x_{i,j}^k$ is the network data flow for the application $a_k$ on the edge $(i, j)$
- $AF$ gives for an application $a_k$ and a chosen need $n_m^k$, the average flow produced by this application. Our approach supposes that the average flow produced by the application according to its needs can be estimated.
- The function RoutingNodes for the network takes a sender node and a receiver node and creates the set of nodes used to go from the sender node to the receiver node depending on the routing policy (OSPF, RIP, RIPv2, etc). This function takes into account the unusable switched off nodes. For an application $a_k$:
  $$RN^k = \text{RoutingNodes}(N_s^k, N_r^k)$$

– for a node $i$, the set $\mathbf{RN^{k,i}_{input}}$ defines the set of nodes connected to node $i$ and sending data for the application $a_k$ and $\mathbf{RN^{k,i}_{output}}$ the set of nodes connected to node $i$ and receiving data for the application $a_k$:
$j \in \mathbf{RN^{k,i}_{input}}$ if $j \in RN^k$ and $\exists e_{i,j} \in \mathbf{E}$
$j \in \mathbf{RN^{k,i}_{output}}$ if $j \in RN^k$ and $\exists e_{j,i} \in \mathbf{E}$

**Relations and constraints for the applications** For an application, only one single need can be chosen:

$$\forall k \in [1, n_A] : \sum_{n=1}^{n_{B^k}} b_n^k - 1 = 0 \tag{9}$$

The flow going out of the sending node is equal to the average flow produced by the application and follows the routing policy :

$$\begin{aligned} &\forall k \in [1, n_A], j \in \mathbf{E} \text{ so that } \exists e_{N_s^k, j} \in \mathbf{E} : \\ &x^k_{N_s^k, j} - \sum_{n=1}^{n_{B^k}} (AF(n_n^k).b_n^k) = 0 \end{aligned} \tag{10}$$

For an application $a_k$ the flow which goes out of the sending node is equal to the flow which goes in the receiving node. We suppose that hosts are connected to border routers with only one link:

$$\begin{aligned} &\forall k \in [1, n_A], \forall i \in \mathbf{E} \text{ so that } \exists e_{N_s^k, i} \in \mathbf{E}, \\ &\forall l \in \mathbf{E} \text{ so that } \exists e_{l, N_r^k} \in \mathbf{E} : \\ &x^k_{N_s^k, i} - x^k_{l, N_r^k} = 0 \end{aligned} \tag{11}$$

The conservation of the flows across the core network is expressed as follow. There is no lost of information, and all information that enter in a core router should exit :

$$\forall k \in [1, n_A], \forall i \in \mathbf{E} : \sum_{\substack{j \in \mathbf{E}, \exists e_{i,j} \\ j \neq N_s^k, N_r^k}} x^k_{i,j} - \sum_{\substack{u \in \mathbf{E}, \exists e_{u,i} \\ u \neq N_s^k, N_r^k}} x^k_{u,i} = 0 \tag{12}$$

The capacity of the switched on links must be respected:

$$\forall (i,j) \in \mathbf{E} : \sum_{k=1}^{n_A} (x^k_{i,j} - c^e_{i,j}.z^e_{i,j}) \leq 0 \tag{13}$$

$$\forall (i,j) \in \mathbf{E}, \forall k \in [1, n_A] : -x^k_{i,j} \leq 0 \tag{14}$$

We define a quality loss of service $QL$ for the application $a_k$ with the chosen quality need of service $c$ as being:

$$\forall k \in [1, n_A] : QL_{a_k} = M(n^k_{n_{NE_k}}) - \sum_{c=1}^{n_{B^k}} M(n_c^k).b_c^k$$

The total quality loss for all applications on the datacenter is:

$$QL_{Total} = \sum_{a_k \in \mathbf{A}} QL_{a_k}$$

### 2.3 The Routing Protocol

Two cases have to be discussed. In the first one, the routing policy is not constrained (called optimal policy). All routes can be used and the optimal solution represents the optimal propagation flow on the network. In the second one, the OSPF policy is used. In that case, it is necessary to add constraints which specify that data flows are fairly divided on the routes having the same cost (the cost being calculated by the OSPF-specific metrics, usually the links capacities):

$$\begin{aligned} &\forall k \in [1, n_A]; \forall i \in \mathbf{RN^k}, Cardinal(\mathbf{RN^{k,i}_{output}}) > 1, \\ &i \neq N_s^k; \forall j \in \mathbf{RN^{k,i}_{output}} : \\ &x_{i,j}^k = \frac{1}{Cardinal(\mathbf{RN^{k,i}_{output}})} \sum_{l \in \mathbf{RN^{k,i}_{input}}} x_{l,i}^k \end{aligned} \tag{15}$$

### 2.4 Global Criterion

The problem can be written as the minimization of a criterion, by going through the possibilities for the various variables $z = [0; 1]$ (the switched on/off elements of the managed network), $b = [0; 1]$ (the chosen QoS needs for the applications) and $x \in \mathbb{R}^+$ (the flow values of the network):

$$\underset{x \in \mathbb{R}^+; z=[0;1]; b=[0;1]}{\text{Min}} \alpha.P_{total} + \beta.(1-\alpha).QL_{total} \tag{16}$$

We introduce $\alpha \in [0; 1]$ which represents the compromise between the total power of the managed network and the service quality loss. It is the duty of the datacenter's administrator to define $\alpha$.

$\beta$ allows a normalization to return both criteria on a comparable scale. This is made using the minimal and maximal borders of $P_{Total}$ and $QL_{Total}$. These borders can be easily calculated for $P_{max}$ and $QL_{max}$ by setting all the power variables and quality of service variable to the maximum. To calculate $P_{min}$ and $QL_{min}$ we use a pre-optimization by setting $\alpha = 0$ and $\alpha = 1$. $\beta$ is calculated by means of an average arithmetic on these borders:

$$\beta = (P_{min} + P_{max})/(QL_{min} + QL_{max}) \tag{17}$$

## 3 Experiments

### 3.1 Context

**Resolution context** To validate our approach, we chose to use the Grid'MIP topology (a part of the grid'5000 project [13]) described by Fig. 1. The three

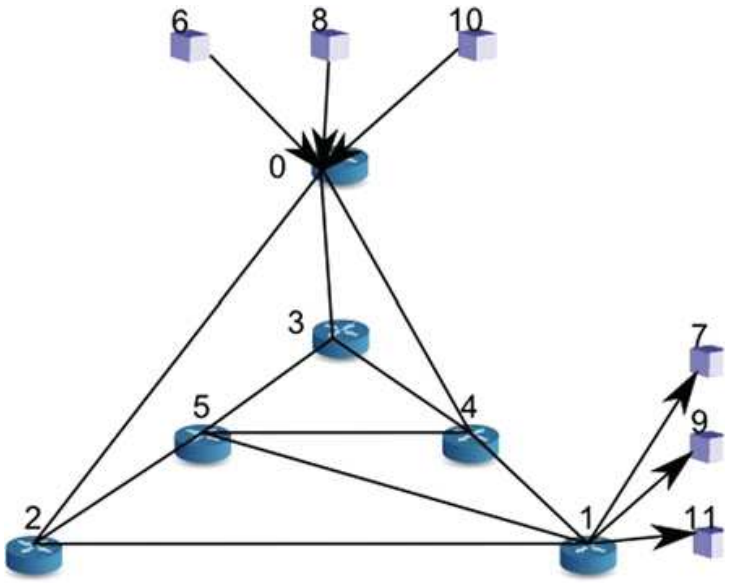



**Fig. 1.** Topology of the Grid'MIP architecture

border routers are numbered 0, 1, 2 and have two modules: one has 48 Gigabit-ethernet ports (each link connected has a capacity of $c_{i,j} = 1000$ Mbits/s) and the other has four Fiber Channel 10-Gigabit ports ($c_{i,j} = 10000$ Mbits/s). The first module is linked to hosts that run applications and the second module to core routers. The three core routers are numbered 3, 4, 5 and have one Fiber Channel module with four 10-Gigabit ports. For the initialization of the power constants ($p_i^c$, $p_{i,l}^m$, $p_{i,j}^e$) we used values measured on the Grid'MIP platform using specific bluetooth wattmeters called Plogg.

Regarding the application placement, we decided to use a simple layout that allows to highlight the switching on/off difference of the network equipments in two precise cases: the optimal-policy case and the OSPF-policy case. That's why we chose to place all the sending applications on hosts linked to the border router 0 and all the receiving applications on hosts linked to the border router 1 ($\forall k \in [1, n_A] : N_s^k = 0, N_s^k = 1$). Thus, the host 6 sends traffic to host 7, host 8 to host 9, etc. If we consider the optimal case there are 3 possible routes from router 0 to router 1: 0-2-1, 0-4-1 and 0-3-5-1. All links of these three routers being different and having a 10-Gbit/s capacity, the total possible bandwidth from router 0 to router 1 is 30 Gbit/s. If we then consider the OSPF case, only 2 routes remain because of the OSPF metric based on capacity: 0-2-1 and 0-4-1. The total bandwidth is therefore 20 Gbit/s. For the average flow constants $AF(n_n^k)$ we consider that all applications are consistent for a better result readability. We associate 5 basic QoS needs to these applications ($| \mathbf{B^k} | = n_{B^k} = 5$) and for simplification purposes we consider the metric $M$ defining the quality of each need to be equal to the average flow resulting from the chosen need, i.e. $M(n_n^k) = AF(n_n^k)$. The 5 average flows resulting of the 5 needs are fixed in Mbits/s to $AF(n_0^k) = 200$, $AF(n_1^k) = 400$, $AF(n_2^k) = 600$, $AF(n_3^k) = 800$ and $AF(n_4^k) = 1000$. Finally, for our experiments, we vary the number of applications between 1 to 60 ($n_A \in [1..60]$) and $\alpha$ by steps of 0.5.

### 3.2 Simple Configuration

**Resolution for the optimal case** The variables of the global model are never multiplied together, so this is a linear programming (**LP**) problem. However, it is necessary to use discrete variables when modeling the problem, for example for the values associated with binary variables of the problem. In this particular case, the model adds integrity constraints and the problem is known as integer linear (**LP**) programming.

For the first optimal calculation we use JOpt [14], an open-source java tool that encapsulates **LP**. JOpt adds a generic layer to linear solvers by using java objects. This allows to access distant solvers like CPLEX [15]. JOpt manages the calculation with an internal load-balancing policy over multiple solvers.

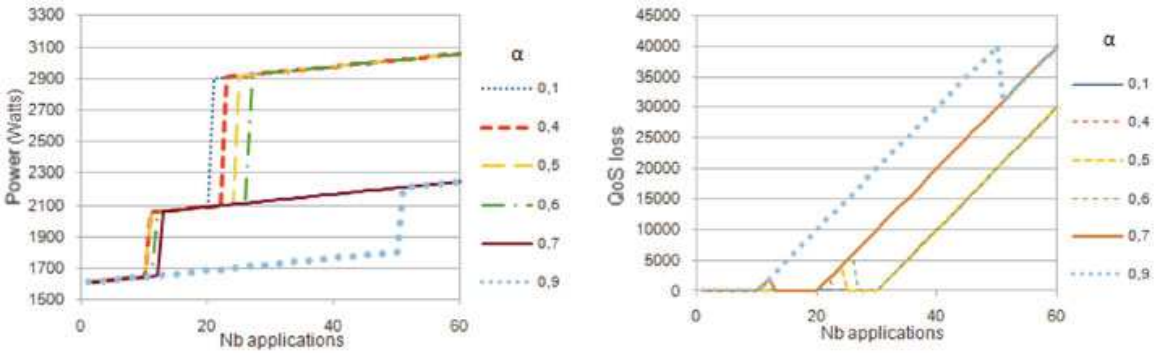
For the criterion resolution, four steps are needed for the optimal case:

- **Step 1 Initialization:** The first step initializes the problem inputs: network graph (routers, modules, ports and links), the placement of the applications on the graph (for each $a_k$ creation of the sending and receipting nodes, of the application needs and calculation of the average flows) and the constants.
- **Step 2 Preparation:** The second step prepares the criterion for the final objective function and the constraints. In fact, this step calculates the coefficients and constructs the JOpt java objects: variables, terms, criteria, constraints and objective function.
- **Step 3 Normalization:** The third step allows to calculate the normalization needed for the objective function. Two calls on the distant solver are needed for this step for the calculation of $P_{min}$ and $QL_{min}$. $P_{max}$ and $QL_{max}$ are also calculated but do not need a solver.
- **Step 4 Minimization:** The forth step consists in launching the minimization of the distant solver and getting the results back.

The calculation of the coefficients and the construction of the JOpt java objects (variables, terms, criteria, constraints and objective function) are being made on a JOpt client coded in java that transfers these objects to the distant linear solver. In our case, we use a CPLEX solver on a distant server with four processors dual-core Intel Xeon 3.2 GHz. In our experiments we distinguish between the **preparation_time** needed to prepare the calculation and the **network_time** needed to transfer the JOpt java objects and get the results back.

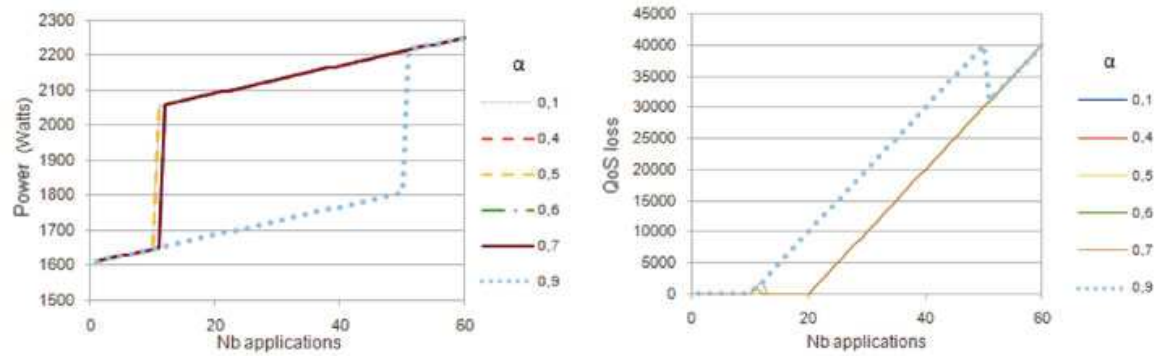
Figures 2a and 3a show the electric power needed by the network as a function of the number of applications and some selected $\alpha$ values.

In the optimal case, (Fig. 2a), between 0 and 10 applications for $\alpha < 0.8$ there is only one route switched on (on Fig. 1 the route 0-2-1 or 0-4-1) and the power consumption increases slowly from 1600 to 1700 Watts as a function of the number of ports (approximately 4 ports per application: 2 for the sender and 2 for the receiver, so 4 Watts per application). For $\alpha = 0,9$ the consumption increases following this scheme until 50 applications. Between 11 and 13 applications for $\alpha < 0,8$, we can see a jump in the power consumption indicating that a router (frame, modules and ports concerned) is switched on. We have to wait until 50




(a) Power consumption (b) Quality loss of service

**Fig. 2.** Optimal self-optimization case




(a) Power consumption (b) Quality loss of service

**Fig. 3.** Heuristic with OSPF routing protocol for self-optimization case

applications to observe this jump for $\alpha = 0,9$. Indeed, after 50 applications, the average flow resulting from the minimum needs is fixed to 200 Mbits/s and the capacity of this unique route (10 Gbit/s) is exceeded which forces the switch on of a second route.

We observe that the more $\alpha$ increases the more power is taken into account in the minimization, delaying the switch on of new routers to the detriment of quality of service, as can be seen on Fig. 2b. Between this first jump and the second after 20 applications we have routers 0, 1, 2 and 4 switched on and the routes 0-2-1 and 0-4-1 used. Starting at 20 applications and for $\alpha < 0,7$, we see a second power jump more important than the first one because it concerns the switch on of routers 3 and 5. This jump is moving from 21 applications for $\alpha = 0,1$ to 27 applications for $\alpha = 0,6$. We see experimentally that for $\alpha = 0,7$ this jump isn't happening (until 60 applications). With the routers 3 and 5 switched on, the network reaches a power consumption of more than 2900 Watts and 3 routes are used (0-4-1, 0-2-1 and 0-3-5-1).

Regarding the QoS (Figs. 2b and 3b), we see that globally the more $\alpha$ is increasing, the more quality loss of service the user gets. We also see that the

quality losses coincide with the power jumps of Figures 2a and 3a. For example for $\alpha = 0,9$, the power is predominant in the objective function. In this case we see that the quality loss of service starts after 11 applications. Indeed, until 10 applications, the total average flow cannot exceed the capacity of the only route switched on (either 0-2-1 or 0-4-1). Thus these 10 applications use the needs whose quality measured by $M$ is maximum and have a resulting average flow $AF(n_4^k) = 1000$ Mbits/s. After 11 applications, the QoS is degraded for at least one application. This phenomenon is visible just before new routers are switched on. We can indeed observe a slight increase of quality loss, visible from 20 to 27 applications for $\alpha < 0,7$ in the optimal case (Fig. 2b).

**Algorithm 1** Heuristic used with OSPF as the routing function

```
Initialize constants, constraints and execute algorithm OSPF
Solve P_min (LP with α = 1), QL_min (LP with α = 0)
Solve directly P_max, QL_max
β = (P_min + P_max)/(QL_min + QL_max)
Create Solution best_solution ← Solve LP
Create List explored_solutions ← best_solution
while nb_iterations < max_iter & moving_objective do
  N^trie ← sort N by inverse number of applications a_k using them
  for i ∈ N^trie do
    M_i^trie ← sort M_i by inverse number of applications a_k using them
    for l ∈ M_i^trie do
      E_{i,l}^trie ← sort E_{i,l} by inverse number of applications a_k using them
      for e_{i,j} ∈ E_{i,l}^trie do
        execute algorithm 2 with z^e_{i,j} = 0
        nb_iterations ← nb_iterations + 1
      end for
      execute algorithm 2 with z^m_{i,l} = 0
      nb_iterations ← nb_iterations + 1
    end for
    execute algorithm 2 with z^c_i = 0
    nb_iterations ← nb_iterations + 1
  end for
end while
return best_solution
```

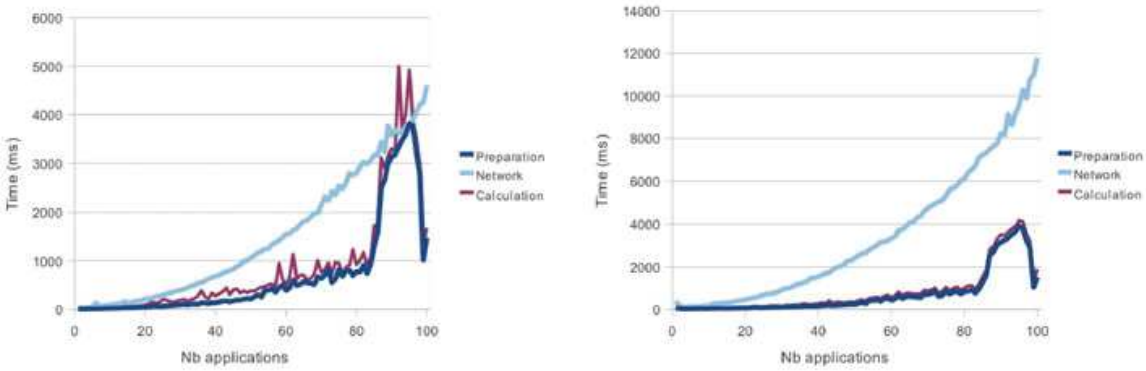
**Resolution when using a heuristic** The use of a heuristic when the network routing policy is OSPF is mandatory because when minimizing the global objective function, for each change of the value of variable $z$, the routing function changes. Because the **Constraint** 15 becomes dependent on a variable to minimize, we introduce a heuristic proposed by algorithm 1. The idea consists in calculating a first solution that uses all OSPF routes for all applications. This is done at the beginning of the heuristic with "algorithm OSPF". This algorithm adds constraints that force the flows using non-OSPF routes to be null. Before starting the iteration loop of the heuristic, a first solution (saved in variable

**Algorithm 2** Verifying function for the heuristic

**Require:** $z$ as input; **explored_solutions** & **best_solution** as input/output
  **if** $z = 0 \notin$ **explored_solutions then**
    **if** $\forall k \in [1, nA]$ & $z = 0$ & $\exists OSPF\_route(N_s^k, N_r^k)$ **then**
      Solution **s** $\leftarrow$ execute algorithm OSPF and Solve **LP** with **best_solution** and $z = 0$
      **if** $\exists$**s then**
        **explored_solutions** $\leftarrow$ **s**
        **if s** is best **then**
          **best_solutions** $\leftarrow$ **s**
          $nb\_tries \leftarrow 0$
        **else if** $nb\_tries < max\_tries$ **then**
          $nb\_tries \leftarrow nb\_tries + 1$
          **explored_solutions** $\leftarrow$ **s**
        **else**
          $moving\_objective =$ **false**
        **end if**
      **else**
        **explored_solutions** $\leftarrow$ **s**
        stop the for loop
      **end if**
    **else**
      **explored_solutions** $\leftarrow$ **s**
      stop the for loop
    **end if**
  **end if**

**best_solution**) is calculated taking into account these constraints. A history saved in variable **explored_solutions** allows to memorize an association between the set of solutions for the variables and the set of constraints added by the heuristic to avoid recalculating a solution with a topology that has already been explored.

The heuristic is stopped if the maximum number of iterations is reached ($nb\_iterations < max\_iter$) or if the best solution hasn't changed since a number of iterations $moving\_objective$). We sort the frames (nodes), the modules and the ports by number of flows using them (smaller number first). We try to switch off first the ports then the modules and finally the frames (nodes) using the sorted list. Indeed, this minimizes the number of applications impacted when switching off the equipment. For each switching off we verify if one OSPF route still exists for all applications and if one solution exists using algorithm 2. This algorithm is the same for the ports ($z_{i,j}^e = 0$), the modules ($z_{i,l}^m = 0$) or the frames (nodes) ($z_i^c = 0$) and we describe it in a generalized way with variable $z$. If this algorithm finds an OSPF route for all applications and a better solution, it is saved in **best_solution**, otherwise it continues exploring the solutions for $max\_tries$ iterations. Every time a better solution is found $nb\_tries = 0$. If we




(a) optimal case (b) heuristic case

**Fig. 4.** Resolution time

reach *max_tries* without finding a better solution then we stop the heuristic *moving_objective* = **false**.

When using the heuristic (Figs. 3a and b) we observe that the results follow the optimal ones but never switch on routers 3 and 5. As explained in Sect. 3.1, when using an OSPF-policy, the route 0-3-5-1 cannot be used for load balancing. Indeed, when using the link capacities as the OSPF metric the cost of route 0-3-5-1 is more important and will never be used for the routes from router 0 to router 1. Regarding the QoS, it is therefore globally more degraded than in the optimal case because the total bandwidth is lower without this third route.

### 3.3 Comparison of the Optimal and Heuristic Cases

Figures 4a and b show the resolution times of the objective function in the optimal and heuristic cases. We varied the number of applications to 100 to show the impact on the different times:

- Preparation time: calculation of the objective, variables, terms and JOpt constraints(developed form).
- Network time: transfer time for the set of optimization parameters to the CPLEX server and transfer time to get the results back.
- Calculation time: the raw and only calculation time for the criterion by the CPLEX server.

Each resolution makes three calls to the distant CPLEX server. Indeed, two calls are needed for the calculation of $\beta$ (for $P_{min}$ and $QL_{min}$) and a call for the resolution of the criterion. The preparation and calculation times include the times for $P_{min}$, $P_{max}$, $QL_{min}$, $QL_{max}$ and the final problem. The network transfer times include the three distant CPLEX calls.

We observe that the preparation and calculation time are substantially the same in the optimal and heuristic cases. These times vary between 20 ms for one application to one second for 80 applications. After 80 applications, these times increase to 4 seconds because it is more difficult to find a solution because all

network links considered in the experiment are saturated. For 100 applications there is no possible solution so the times decrease to one second.

Regarding the network transfer times for the JOpt data to the CPLEX server, we see that when we use the heuristic it is more important than in the optimal case. Indeed the heuristic adds additional constraints related to the routing policy. Adding constraints increases the number of variables and terms to be transmitted to the server for the resolution. In the heuristic case, this time varies exponentially from 360 ms for one application to 12 seconds for 100 applications.

Given the results, we can conclude that using a heuristic doesn't generate an overhead for the calculation time or preparation time. However, the network transfer time is multiplied by 2.6. Globally, these times are still reasonable in the case of dynamic reconfiguration of network devices like routers. Therefore, the optimization of the network has to be planned with a granularity of about an hour, which makes the resolution time negligible.

## 4 Conclusion

Human administrators cannot face the complexity of management of the IT infrastructure and the deployed application on datacenters anymore. Whether it concerns hardware or software issues, the optimization process is a tricky and costly task. We introduced the "self-optimizing" autonomic property at the hardware level by applying it to the optimization of datacenters energy costs.

We introduced an approach to describe a compromise between, on one hand, the power consumption of the network infrastructure and on the other hand, the deterioration of the QoS for the applications using this network. Being able to control the dynamic reconfiguration at two levels: at the application level by dynamically reconfiguring QoS profiles and at the hardware level by switching on and off links, modules or routers allows to have a global management of the datacenter. Indeed, the use of an autonomic manager allows the administrator to control the energy costs or the performance by varying only one parameter that handles a high level management policy.

Regarding the limits of our approach, we suppose that a relation of order exists between the QoS needs by using the function $M$. This order is not to be confused with the final user "quality of experience" (QoE) [16].

The goal was to deal with the performance/electric consumption dilemma for the network part. This is a challenge for years to come as the perfect system must take into account the energy consumption of the machines and also the network equipments, the QoS and financial costs for example.

**Acknowledgment.** This work was partially supported by the COST (European Cooperation in Science and Technology) framework, under Action IC0804. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).

## References


1. Skene, J., Lamanna, D.D., Emmerich, W.: Precise service level agreements. In: 26th International Conference on Software Engineering (ICSE'04), Edinburgh, Scotland, United Kingdom (2004)
2. Chang, F., Karamcheti, V.: Automatic configuration and run-time adaptation of distributed applications. In: High Performance Data, Computing, p. 11 (2000)
3. Grossman, D.: New terminology and clarifications for diffserv. Technical report, RFC 3260 (2002)
4. Hsueh, C., Lin, H., Huang, G.C.: Channel bonding in linux ethernet environment using regular switching hub. Syst. Cybern. Inform. **2**(3), 35–38 (2004)
5. Callon, R.W.: Use of OSI IS-IS for routing in TCP/IP and dual environments. Technical report, RFC 1195 (1990)
6. Moy, J.: Ospf version 2. Technical report, RFC 2328 (1998)
7. Zinin, A.: Cisco IP routing: packet forwarding and intra-domain routing protocols. Addison-Wesley, Boston (2002)
8. Albrightson, R., Garcia-Luna-Aceves, J.J., Boyle, J.: EIGRP-a fast routing protocol based on distance vectors. In: Proceedings of the Networld/Interop, vol. 94 (1994)
9. Liu, X., Ghazisaidi, N., Ivanescu, L., Kang, R., Maier, M.: On the tradeoff between energy saving and qos support for video delivery in eee-based fiwi networks using real world traffic traces. Lightwave Technol. J. **29**(18), 2670–2676 (2011)
10. Arai, D., Yoshihara, K.: Eco-friendly distributed routing protocol for reducing network energy consumption. In: International Conference on Network and, Service Management, October 2010
11. Chabarek, J., Sommers, J., Barford, P., Estan, C., Tsiang, D., Wrigh, S.: Power awareness in network design and routing. In: Proceedings of the IEEE INFOCOM (2008)
12. Dong, X.W., El-Gorashi, T., Elmirghani, J.M.H.: Green ip over wdm networks with data centers. Lightwave Technol. J. **29**(12), 1861–1880 (2011)
13. Capello, F., Caron, E., Dayde, M., Jegou, Y., Desprez. F, Primet, P., Jeannot, E., Lanteri, S., Leduc, J., Melab, N.: Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In: 6th IEEE/ACM International Workshop on Grid Computing, Seattle, Washington, USA, pp. 99–106 (2005)
14. Shneidman, J.: JOpt, a simplified java wrapper for linear and mixed integer programming. Technical report, http://www.eecs.harvard.edu/econcs/jopt/ (2005)
15. IBM: Mathematical programs - IBM ILOG CPLEX optimizer - software. Technical report. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ (2009)
16. Jain, R.: Quality of experience. IEEE Multimed. **11**(1), 96–97 (2004)