Differential energy saving algorithms in a distributed router architecture

Differential Energy Saving Algorithms in a Distributed Router Architecture

Andrea Bianco[a,*], Fikru Getachew Debele[a], Luca Giraudo[b]

[a] *Dipartimento di Elettronica e Telecomunicazioni, Politecnico di Torino*
Corso Duca degli Abruzzi, 24 - 10129 Turin, Italy
[b] *Google, 1667 Plymouth Street, Mountain View CA 94043, USA*

Abstract

A distributed multi-stage software router (MSSR) is composed by several interconnected software routers running on personal computers (PCs). The MSSR architecture overcomes scalability and performance issues of single software router by providing parallel forwarding paths. Like many networking devices, a MSSR must be sized for peak traffic load, which implies energy inefficiency at low loads. Thus, we focus on energy saving schemes to improve the router energy efficiency by dynamically adapting the MSSR architecture to the currently offered load. We first introduce an optimal energy saving algorithm defined as a mixed integer linear programming (MILP) optimization model. Then, heuristic solutions, named differential algorithms are discussed. While the optimal approach provides higher energy savings, the heuristics avoid the complete MSSR reconfiguration, thus reducing forwarding delays and minimizing service interruption. The performance evaluation shows that the proposed heuristic algorithms, that gracefully modifies the internal MSSR configuration, preserve the load proportional energy demand characteristics of the optimal algorithm, with a minimal loss of efficiency, largely compensated by algorithm simplicity.

Keywords: energy efficiency, distributed router, heuristic algorithms

1. Introduction

Although ICT can make a major contribution to support energy savings and to face climate changes, it is responsible for roughly 2% of global carbon emissions - a figure close to the worldwide airline industry consumption. Since the ICT sector is growing at a faster rate [1], the CO_2 emission by ICT industry will reach an estimated of 6% by 2020 [2, 3]. Telecom infrastructures and devices contribute to about 25% of the total ICT consumption.

Today the energy consumption in networking devices is proportional to the installed capacity rather than to traffic demands. Thus, dynamically resizing the system capacity to match traffic demands may significantly enhance network energy efficiency. Distributed router architectures, being composed by many independently powered components, are perfectly suited to provide the flexibility needed to dynamically resize the router architecture to match input traffic demands.

This research work focuses on the problem of reducing the energy consumption of distributed router architectures, with emphasis on a multistage software router (MSSR)architecture [4] shown in Fig. 1. The MSSR architecture is a distributed software router architecture proposed to overcome single PC-based software router performance limitations. The main benefits of the MSSR architecture, discussed in [4], include scalability, flexibility, programmability, enhanced performance and low cost.

The architecture exploits classical PCs as elementary switching elements to build a high-performance software router (SR). The MSSR architecture is organized in three stages: i) a front-end stage exploiting layer-2 load balancers (LBs), either open-software or open-hardware based [5], that act as interfaces to the external networks and distribute IP packets to ii) back-end personal computers (BEPCs), also named *back-end routers*, that provide IP routing functionality, and iii) an interconnection network, based on Ethernet switches, that connects the two stages. A control entity, named *Virtual Control Processor* (virtualCP), running on a selected back-end router, controls and manages the overall architecture through a DIST protocol [4]. The virtualCP hides the internal details of the MSSR architecture to external network devices.

State-of-the-art PC-based routers can route few Gbps if packet processing is performed by the CPU [6][7] or few tens of Gbps if a specialized packet processing is implemented [8][9]. Therefore, the MSSR architecture might require several tens of BEPCs to build a high performance routing capability. This performance scale implies a high redundancy level at the back-end stage, which may translate into a source of energy waste during low traffic periods. Indeed, like most networking devices, the MSSR is typically designed to sustain peak traffic, thus dissipating a constant amount of power regardless of the actual traffic load. However, during low traffic periods, the routing task could be transferred to a subset of back-end routers, by setting all un-needed back-end routers in low power state by switching them off to save energy.

As an example, let us examine the design of a MSSR architecture comparable to a Juniper T320 core router that supports

*Corresponding author
Email addresses: andrea.bianco@polito.it (Andrea Bianco), fikru.debele@polito.it (Fikru Getachew Debele), lucagiraudo@google.com (Luca Giraudo)

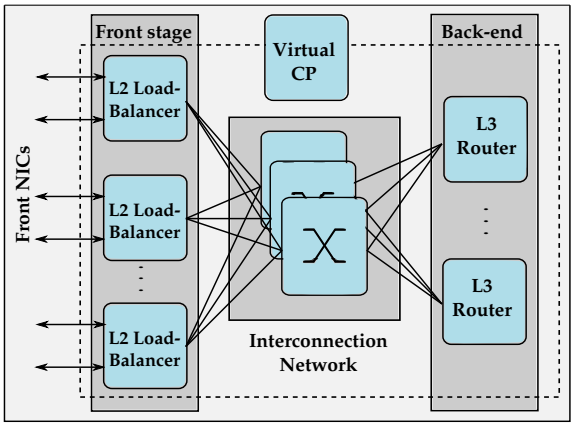


Figure 1: Multistage software router architecture

up to sixteen 10 Gbps ports with a 160 Gbps forwarding capacity. The T320 router has a nominal power consumption of 2.8 kW [10]. Suppose the following internal components are available to build the MSSR architecture:

- back-end routers with 5.5 Gbps forwarding capacity and equipped with a single 10 Gbps interface;
- LBs with two (one internal and one external) 10 Gbps interfaces;
- a commercial Ethernet switch (or a set of interconnected switches) with enough capacity to interconnect LBs and back-end routers.

To design a 160 Gbps capable MSSR, 16 LBs, 1 switch and 29 back-end routers are needed. Assuming that PCs used as LBs and back-end routers have a power consumption of 80W [1] and that the interconnecting switch consumes 200W, the MSSR architecture dissipates roughly 3.8 kW at maximum load. However, during very low traffic periods, one (or few) back-end router(s) may be enough to provide the required service, thanks to input traffic consolidation to few BEPCs while setting all other BEPCs to low power state. Thus, it would be possible to dramatically reduce the MSSR power consumption to 1.5 kW, including the power consumption of LBs and the switch. Thus, the MSSR architecture is competitive, in terms of power consumption, with commercially available routers also in this worst case scenario. However, the real advantage of the MSSR architecture relies on the ability to scale the router configuration to adapt to traffic fluctuations. A commercial router is sized for peak traffic and, today, it is not easily re-sizeable. Since the average traffic load on 24 hours is much less than the peak load, having a scalable architecture leads to significant energy savings, as demonstrated in this paper.

In the above example, the LBs and the switch contribute to about 34% of the total energy consumption. We do not consider energy saving features for these two stages because they act respectively as external interfaces (which must stay active to guarantee MSSR external connectivity) and internal interconnection network (which must be active to guarantee internal connectivity). As such, saving energy by switching off LBs is only possible when operating at the network level, as in [11], where the whole network power consumption is optimized by redirecting the traffic over a subset of routers. Therefore, in this paper we propose algorithms that resize the number of active BEPCs, on the basis of their power consumption and routing capacity, to adapt the overall MSSR capacity to the incoming traffic demand so as to minimize the architecture power consumption. The proposed algorithms gracefully reconfigure the back-end stage when needed without causing service disruption or reconfiguration delays. Preliminary results were presented in [12]. The main novel contributions, besides more extensive results, are: i) the extension of previously proposed algorithms with other heuristic solutions and ii) the analysis under unsplittable traffic assumption.

The remainder of the paper is organized as follows: related works in energy savings are presented in Sec. 2. In Sec. 3 we present the MSSR energy saving problem and give the formal problem formulation. Proposed heuristic solutions are discussed in Sec. 4. Sec. 5 compares the performance of the proposed heuristics with respect to the optimal solution. Finally, Sec. 6 concludes the paper.

2. Related work

Rising energy cost and increasing environmental standards urged researchers and industries to draw attention to energy footprint aspect of data networking. Starting from a position paper by Gupta et al. [13], IT researchers started focusing on energy saving issues also on data network. In this section, we focus on research efforts related to energy optimization in electronic devices, not limiting our attention to networking devices but including also data centers and server clusters.

Chase et al. [14] proposed an energy-conscious request switching paradigm to reduce energy usage for server cluster during low load periods. The switch monitors cluster load and concentrates traffic on the minimal set of servers that can satisfy the load with a specified utilization and latency levels. The remaining idle servers are put to a low-power state. The proposal basically extends the load-balancing switches with an energy-conscious routing policy that leverages the power management features of the back-end servers.

A similar approach is proposed by Pinheiro E. et al. [15]. In this case a systems that dynamically turns cluster nodes either on to be able to handle a load increase or off to save energy during low load periods is proposed. A control-theoretic and load distribution algorithm makes cluster reconfiguration decision by considering the cluster total load, the power and performance implication of changing the current configuration. The technique saves up to 38% of energy.

Power-aware request distribution [16] is a method of scheduling service requests among servers in a cluster so that energy consumption is minimized, while maintaining a particular level of performance. Energy efficiency is obtained by pow-

[1] This value could be further reduced for LBs if they are designed in open hardware exploiting field programmable gate array (FPGA).

ering down some servers when the desired quality of service can be met with fewer servers.

All the above mentioned schemes [14–16] operate only at the coarse-granularity of the entire server and/or only homogeneous servers are considered. However, in the MSSR architecture back-end routers are heterogeneous both in capacity and power consumption. Furthermore, managing one or more Network Interface Cards (NICs) in the routers adds another dimension to the optimization problem. Observe that a 10 Gbps NIC may consume a non negligible amount of power, roughly 20 W [17], one fourth of a standard PC consumption.

Heath T. et al. [18] designed a cooperative Web server for a heterogeneous cluster exploiting an optimization model to minimize the energy per request. The approach saves 45% more energy than an energy-conscious server that was proposed for homogeneous clusters. Their approach is similar to the optimal solution of the MSSR design as defined in this paper. However, both solutions imply service disruption during the MSSR reconfiguration phase, a negative feature overcame by our proposed differential algorithm.

The authors in [19] propose an energy-efficient resource allocation mechanism called ECO-ALOC for cluster-based software routers. ECO-ALOC tunes CPU states and consolidates virtual routers to save energy at a server and cluster level. The packing granularity is the (large) size of a virtual machine.

Differently from ECO-ALOC, our proposed algorithms focus on energy savings considering also NIC consumption. As reported in [20], device energy consumption increases with the number of active ports. Thus, taking into account the number of NICs becomes a crucial issue in the MSSR architecture design, because, for a fixed needed capacity, configurations requiring less NICs may perform better from an energy perspective. On the contrary, considering the CPU consumption is less important in our scenario, because the algorithms try to exploit active PCs as close as possible to full capacity before turning on additional PCs. Indeed, a new BEPC is turned on only if all the already active PCs are used to their full capacity. Obviously, the incoming traffic may not exactly fit the total capacity of the active PCs or traffic may fluctuate while a given MSSR configuration is active. In this case, CPU tuning would further reduce energy consumption.

A white paper from Juniper Networks, Inc. [21] reports an energy criteria used to compare energy consumption of different network devices. The normalized energy consumption rating (ECR), measured in W/Gbps, is defined as

$$ECR = \frac{E}{T} \tag{1}$$

where E is energy consumption of the device and T is the effective full-duplex throughput. Both values may come from either internal testing or the vendor's data sheet. ECR is a peak metric that reflects the highest performance capacity of the device. In our proposed energy saving algorithms for MSSR design we use a similar criteria as one of the PC selection criteria when designing a new back-end router configuration.

Finally, a MSSR energy saving scheme has been proposed in [22]. Since the optimal problem is not solvable for large scale routers, a two-step heuristic approach has been proposed to split the problem into link and router optimization problems. The results show that the two-step algorithm scales to a large size MSSR and its solution is within 10% relative error of the optimal solution. In [12], the authors proposed a differential power saving scheme for the same architecture. The proposed algorithm preserves the load proportional power demand characteristics of the optimal solutions. Furthermore, the capability of the differential heuristic to define the new MSSR configuration as a slight modification of an existing solution permits to minimize the potential service interruption that a full reconfiguration, required by the optimal approaches, would create. In both papers only splittable traffic is considered.

3. System Model

In this section we present the MSSR architecture energy saving model that optimally adapts the number of BEPCs to the currently offered traffic load. First, we discuss the assumptions used to formalize the problem and later we describe the optimization model in detail.

3.1. Assumptions

The MSSR architecture energy saving model is based on the following assumptions:

1) Input traffic: traffic is assumed to be known through predictions, estimates measurements or historic traffic profiles. Two traffic models are considered:

   i) *unsplittable input traffic*: Flow based input traffic information is assumed, to ensure in order delivery of packets at the output interface. A flow-based load balancing technique among BEPCs may be used at LBs to guarantee ordered packet delivery. A flow is defined as all the packets with the same values on a specific subset of header fields. Input traffic belonging to the same flow will be directed by the LB to the same BEPC for routing operation. The name *unsplittable input traffic* is used because packet belonging to the same flow are not split among BEPCs, i.e., they are routed by the same BEPC. Flow based routing may also be beneficial to guarantee a given target QoS level.

   ii) *splittable (aggregated) input traffic*: If ordered delivery and QoS provisioning are not mandatory, the input traffic is characterized simply by the amount of aggregated data that should be processed by the MSSR architecture in a given time frame. LBs split, according to a proper packet based load balancing scheme, the aggregate traffic among all available BEPCs for routing operation.

2) PC and NIC power consumption: PCs' and NICs' power consumption are independently optimized. Indeed, we assume that it is possible to turn off NICs on an active BEPCs when NICs are not needed. We consider a single link per card scenario. Therefore, we assume that turning off a link means turning off the NIC itself.

3) on/off power model: to keep the problem formulation simple, we chose an on/off energy model [23] both for the BEPCs and their links, i.e., the power consumption does not depend on the actual resource load, but it is either zero when the resource is off or equal to a fixed value when the resource is on. This assumption well matches NIC behavior where the load dependent power consumption is negligible [20]. For what concern PCs, the energy consumption ratio between idle and full load state can be as low as 20% [24, 25]. However, in the MSSR architecture, all activated PCs work almost at full load as described in Sec. 4.3. Thus, we may rely on the on/off assumption also for PCs composing the back-end stage.

3.2. Problem formulation

Based on the above assumptions, the MSSR architecture energy saving scheme is formalized as follows:
GIVEN

i) a set of BEPCs B; each PC $b \in B$ characterized by a power consumption (excluding NICs) $P_b \in \mathbb{R}$ and a routing capacity[2] $C_b \in \mathbb{R}$,

ii) a set of links L_b connected to each PC $b \in B$; each link $l \in L_b$ is characterized by power consumption $P_{bl} \in \mathbb{R}$ and link capacity $C_{bl} \in \mathbb{R}$, and

iii) a set of input traffic demands T, where T_k is a flow, $k = 1, 2, ..., |T|$,

SELECT PCs and links required to route the traffic demand such that the architecture power consumption is minimized, SUBJECT TO link rate and router capacity.

Let α_b be a PC selection binary variable (1 if PC $b \in B$ is activated, 0 otherwise), and β_{bl} the link selection binary variable (1 if link $l \in L_b$ connected to PC $b \in B$ is used, 0 otherwise). Furthermore, let δ_{blk} be a flow selection:

- binary variable, $\delta_{blk} \in \{0, 1\}$, for unsplittable input traffic, set to 1 if a flow $T_k \in T$ is forwarded on link l of router b and to 0 otherwise, or

- real variable, representing the portion of splittable input traffic to be forwarded by router b on link l; i.e., $\delta_{blk} \in [0, 1]$. Note that for splittable traffic, the set T has only one element which represents the aggregate input traffic, i.e., $k = 1$.

The MSSR energy saving problem is formalized as a mixed integer linear programming (MILP):

minimize:

$$P = \sum_b \left(P_b \alpha_b + \sum_l P_{bl} \beta_{bl} \right) \tag{2}$$

subject to:

$$\sum_b \sum_l \delta_{blk} = 1, \quad \forall k \tag{3}$$

$$\sum_l \sum_k \delta_{blk} S_k \leq C_b \alpha_b, \quad \forall b \in B, \forall k \tag{4}$$

$$\sum_k \delta_{blk} S_k \leq C_{bl} \beta_{bl}, \quad \forall b \in B, \forall l \in L_b, \forall k \tag{5}$$

$$\alpha_b \geq \beta_{bl}, \quad \forall b \in B, \forall l \in L_b \tag{6}$$

$$\alpha_b, \beta_{bl} \in \{0, 1\}$$

$$\delta_{blk} \in \{0, 1\} \vee [0, 1]$$

where S_k is the rate of flow T_k measured in bits/s. In the formulation, (3) ensures that input traffic T is served, while (4) and (5), make sure the capacity constraints of router (C_b) and link (C_{bl}) are not violated. (6) ensures that router b is active if at least one of its links is chosen to serve some traffic.

Eqn. (2) - (6) define a MILP problem that optimizes the MSSR architecture power consumption, considering both PCs and NICs, for both unsplittable and splittable input traffic. The solution to the above defined problem is a BEPCs configuration capable of routing input traffic T satisfying devices capacity constraints while minimizing the power consumption. The problem is NP-hard, as demonstrated in Appendix A. Thus, exact methods can only be used to solve small size cases. Therefore we define heuristics to solve the problem. However, we use the MILP results as a reference to measure the performance of the proposed heuristic algorithms.

4. Heuristic Algorithms

Besides the NP-hardness of the MILP solution, there is another important reason that discourages the use of the optimal algorithm. When solving the MILP at different times to track traffic changes, the optimal solutions may required a complete reconfiguration of BEPCs, because no correlation exists among consecutive configurations. This may causes temporary service disruption or large forwarding delays due to the PC reconfiguration time. One solution could be to keep the previous configuration until the new one is setup, but this compromises the optimality of the solution during the reconfiguration phase. In the following, the proposed heuristics permit to overcome the pitfalls of independently running optimal solutions at different times.

4.1. Algorithm Description

The proposed energy saving heuristic is a differential algorithm that defines a new back-end stage configuration needed to satisfy the current traffic demand by modifying an existing MSSR configuration to avoid service interruption. The algorithm is a modified first-fit-decreasing bin-packing algorithm [26] with bins (PCs) having different size (actual routing capacity C_b^{PC}) and different usage cost (power dissipation) and splittable or unsplittable items (input traffic). Furthermore, the algorithm must consider bins (links) within a bin (PC) in the actual capacity computation and packing phases.

[2] Routing capacity for a PC is defined as the amount of traffic the PC can forward, measured in bits/s

During the initialization phase, the algorithm computes the actual routing capacity of each PC and sorts the devices (both PCs and links). The actual routing capacity (C_b^{PC}) of a PC $b \in B$ is limited either by the CPU packet processing capacity or by the sum of the link rates on the PC [6]. The sorting phase enhances packing efficiency [27], because the devices have different capacity and power dissipation. Sorting criteria are discussed in Sec. 4.2). Furthermore, for unsplittable input traffic, flows are also sorted in descending order of size, again to increase the packing efficiency.

After the initialization phase, the algorithm keeps monitoring input traffic changes. If an input traffic increase is detected, the algorithm keeps the current configuration and augments the current capacity by turning on additional devices. Otherwise, if traffic decreases, some active devices are turned off to down size the current active configuration to match traffic demand (more details in Sec. 4.3).

In the following subsections, we refer to the pseudo-code as reported in Algorithm 1-3 to describe the algorithms. Algorithm 1 describes the sorting algorithm, while Algorithms 2 and 3 present the packing schemes for unsplittable and splittable traffic respectively.

4.2. Initialization Phase

At algorithm start up, the available devices of MSSR configuration, i.e., the set of PCs and their NICs available for the back-end stage, is analyzed to compute the actual routing capacity (C_b^{PC}) of each PC and, as a consequence, of the whole architecture (C_{MSSR}) as:

$$C_b^{PC} = \min(C_b, \sum_{l \in L_b} C_{bl}) \tag{7}$$

$$C_{MSSR} = \sum_{b \in B} C_b^{PC} \tag{8}$$

Eqn. (7) defines the capacity of a PC as the minimum between a PC CPU packet processing capacity (C_b) and the sum of all the link rates on that PC. The MSSR architecture capacity is defined as the sum of all BEPCs actual routing capacity.

Then, PCs and links are sorted according to one of the following two schemes (Algorithm 1):

i) device efficiency: in descending order of efficiency (lines 1 - 9)

ii) device power dissipation: in ascending order of power dissipation (lines 11 - 18)

The device efficiency is defined as the amount of traffic routed per watt:

$$\eta = \frac{\text{actual capacity}}{\text{power}} \tag{9}$$

When sorting PCs according to their efficiency, two slight variations are considered. The version of the algorithm denoted as η_{NIC-} does not consider NIC power consumption when evaluating PCs' efficiency.

$$\eta_{NIC-} = \frac{C_b^{PC}}{P_b} \tag{10}$$

Instead, the version of the algorithm that also takes into account the link power consumption during the PCs sorting stage is named η_{NIC+}. In this case, Eqn. (10) can be rewritten as:

$$\eta_{NIC+} = \frac{C_b^{PC}}{P_b + \sum_{l \in L_b} P_{bl}} \tag{11}$$

where N is the number of links connected to back-end PC b.

Similarly, also links in PC b are sorted according to their efficiency η_{bl}:

$$\eta_{bl} = \frac{C_{bl}}{P_{bl}} \tag{12}$$

The second sorting scheme sorts devices by power dissipation, where less dissipating devices come first in the list. Also the sorting according to PCs power consumption comes in two variations P_{NIC-} and P_{NIC+}. P_{NIC+} sorts PCs considering links power consumption as:

$$P_{NIC+} = P_b + \sum_{l \in L_b} P_{bl} \tag{13}$$

P_{NIC-} considers only the PCs power consumption for sorting purposes. Links on a PC are also sorted according to their power consumption if the second sorting scheme is deployed.

4.3. Packing Algorithm and BEPCs reconfiguration schemes

The goal of the algorithm is to minimize the power consumed by the BEPCs that serve the requested traffic demand. After the device sorting phase (Algorithm 1), both Algorithm 2 & Algorithm 3 follow a greedy approach in packing the incoming traffic to BEPCs. When using the efficiency sorting scheme in Algorithm 1, Algorithm 2 & Algorithm 3 start activating (i.e., setting in the on state) the available most efficient PC b according to Eqn. (10) or (11) and the most efficient links l on that PC b according to Eqn. (12). The number of active PCs is stored in variable A. When the first PC's actual capacity (C_b^{PC}) is fully utilized, both algorithms consider packing the residual incoming traffic to the next available most efficient PC. This procedure is iterated until all the incoming traffic is served (Algorithm 2, lines 5-16; Algorithm 3 lines 4 - 13). If the incoming traffic exceeds the maximum MSSR architecture capacity (C_{MSSR}), the extra amount of traffic is discarded (Algorithm 2, line 22; Algorithm 3 line 19). Similarly, if the sorting in Algorithm 1 is based on device power consumption, both algorithms starts packing traffic to the least power consuming PCs and the least consuming links on those PCs, until all the incoming traffic is served.

Once a BEPCs configuration has been defined, the algorithm continues to monitor the incoming traffic demand to identify a traffic modification worth of a MSSR reconfiguration phase startup, exploiting either a threshold based or a sampling based reconfiguration triggering mechanism, discussed in Sec.4.4.

When a reconfiguration request is triggered, if the traffic demand has increased the algorithm computes the extra traffic demand and turns on additional resources, i.e., inactive links on already active PCs and new PCs currently in low power state, to satisfy the increased traffic demand (Algorithm 2, lines 17 -

20; Algorithm 3 lines 14 - 17). As previously described, links and PCs to be activated are considered in order of increasing efficiency or decreasing power consumption. On the other hand, if traffic decreases, the algorithm adjusts the running configuration by turning off extra links and PCs to down-size the back-end configuration (Algorithm 2, lines 26 - 28; Algorithm 3 lines 25 - 27). Under decreasing input traffic, the algorithm turns off devices in reverse order: less efficient or more energy consuming PCs and links first.

The block diagram depicted in Fig. 2 shows how the proposed energy saving scheme fits into the MSSR architecture. The energy saving algorithms run on the *control PC* where also the virtualCP is running, to ease their interaction. Two events may trigger a MSSR reconfiguration: First, the virtualCP monitors, through the DIST protocol, any change in the current MSSR configuration, to detect any modification in link and PCs configuration that may be caused by device faults, upgrade or device addition/removal for management purposes. Second, the traffic statistic module that collects traffic information detects either a modification in the traffic request above a predefined threshold or a new input traffic sampling event. Once the new MSSR configuration has been defined by the energy saving algorithms, the virtualCP switches on and off the proper set of PCs and links exploiting the DIST protocol features. Furthermore, load balancing tables of front-end stage LBs are modified accordingly, to ensure that the incoming traffic is forwarded only to currently active PCs.

4.4. Reconfiguration triggering mechanism

In threshold based reconfiguration, a specific threshold is defined. If the traffic change exceeds the threshold, then a new BEPCs configuration is defined to handle the new traffic demand. We assume an input traffic change of ±10% to trigger a back-end stage reconfiguration. Thus, any change in input traffic below ±10% does not activate a reconfiguration step. Note that the current MSSR configuration could be defined by augmenting the traffic demand by a given amount, e.g., 10%, to make the architecture more tolerant to traffic fluctuations below the threshold and to reduce the risk of packet losses. We do not consider the capacity augmentation in this paper neither for the proposed algorithms nor for the optimal solution. The threshold value can be estimated from the variability observed in a traffic profile to balance PCs turning on/off frequency with energy savings. Indeed, the threshold must be set such that the time between two consecutive reconfiguration requests is significantly larger than the time needed to activate a device.

In the sampling based reconfiguration, a traffic sampling interval is defined a priori and the input traffic is sampled accordingly. Every time a new sampled input traffic is available, the algorithm resize the BEPCs configuration to match the traffic demand, i.e. the algorithm enters the reconfiguration phase at regularly spaced intervals. Similarly to the threshold selection, the definition of a proper sampling time should take into account the time required to turn on/off PCs to make on/off energy overhead negligible.

Algorithm 1 Sorting algorithm - Pseudo-code description

Require: BEPCs configuration;
Ensure: Sorts PCs and NICs according to Eqn. (9) - (13);

```
1: if sort by efficiency then
2:     if NIC+ then
3:         sort(list of η_NIC+)
4:     else
5:         sort(list of η_NIC−)
6:     end if
7:     for all PCs do
8:         sort(list of η_bl);
9:     end for
10: else
11:    if NIC+ then
12:        sort(list of P_NIC+)
13:    else
14:        sort(list of P_NIC−)
15:    end if
16:    for all PCs do
17:        sort(list of P_bl);
18:    end for
19: end if
```

Algorithm 2 Unsplittable traffic routing - Pseudo-code description

Require: Set of flows (T) and BEPCs configuration. Note that at least 1 active router ($A = 1$) must be available.
Ensure: Turn on/off PCs and NICs;

```
1: sort(list of flows)
2: sort(list of device)
3: loop
4:     if reconfiguration required then
5:         for all flows f = 1, 2, ..., |T| do
6:             for all PCs b = 1, 2, ..., |B| do
7:                 for all links l = 1, 2, ... |L_b| do
8:                     if flow f fits in link l then
9:                         pack flow f in link l;
10:                        break;
11:                    end if
12:                end for
13:                if flow f packed then
14:                    break;
15:                end if
16:            end for
17:            if flow f did not fit in any A PCs link then
18:                if b is less than |B| then
19:                    turn on next PC in the list
20:                    decrement f;
21:                else
22:                    drop flow f;
23:                end if
24:            end if
25:        end for
26:        if b less than A then
27:            extra PCs = A - b;
28:            switch off extra PCs;
29:        end if
30:        A = b;
31:    end if
32: end loop
```

Algorithm 3 Splittable traffic routing - Pseudo-code description

Require: BEPCs configuration. Note that at least 1 active router (A = 1) must be available;
Ensure: Turn on/off PCs and NICs;

```
sort(list of device)
loop
  if reconfiguration required then
    for All PCs b = 1, ..., |B| do
      for All links l ∈ L_b on PC b do
        if l has residual capacity then
          assign portion of input traffic to l;
          input traffic -= l residual capacity;
        end if
        if All packed then
          break;
        end if
      end for
      if All not packed in A PCs then
        if b less than |B| then
          turn on next PC in the list;
          assign portion of input traffic to the activated PC;
        else
          drop extra input traffic;
        end if
      else
        break;
      end if
    end for
    if b less than A then
      extra PCs = A - b;
      switch off extra PCs;
    end if
    A = b;
  end if
end loop
```

Regardless of the reconfiguration triggering mechanism used, traffic estimation for splittable traffic can be obtained by collecting traffic information through standard procedures already available on devices, e.g., exploiting SNMP. Unsplittable traffic implies flow awareness: The flow definition depends on how the provider wishes to manage the network. The finer the granularity in flow definition the more complex will be the collection of measurements. PC-based LBs can easily handle per flow-measurements with no additional HW requirements. Furthermore, the increasing availability of SDN based networking devices suggest that flow-based traffic statistics will be available in the near future to support the unsplittable traffic case. Thus, we can assume that no additional hardware components are needed to obtain the needed traffic estimates.

Finally, as previously observed, threshold and sampling time must be set to ensure that BEPCs reconfigurations are triggered on a relatively long time scale, i.e. much larger than PC on/off switching times. This would make the on/off overhead energy negligible with respect to the savings ensured by a reconfiguration. Parameter set up can be done off-line exploiting historical data; real time parameter tuning can also be envisioned to better follow traffic fluctuations. Details on traffic sampling frequency and on threshold setting to trigger a reconfiguration are not examined in the paper, but standard techniques derived from traffic monitoring and analysis can be exploited to solve this issue.

4.5. Computational complexity

Setting up a new configuration involves the following steps: i) computing the actual capacity of each PC, ii) sorting devices and (unsplittable) input traffic, and iii) packing the input traffic to the new configuration. Computing the actual capacity of each PCs requires examining PCs and links in each PCs, which can be done in $O(mn)$ time, where n is the number of PCs and m is the number of links on each PC. The number of NICs per PC is usually limited to a small number and can be considered as a constant under the asymptotic assumption of large size MSSR architectures. Thus, the actual capacity computation can be performed in $O(n)$. The sorting phase can be executed in $O(nlogn)$ [28]. The implemented first-fit-decreasing algorithm for packing has a worst case running time of $O(nlogn)$ [29]. Thus, in summary, a new MSSR configuration can be designed in $O(n + nlogn + nlogn) = O(nlogn)$ time, i.e., logarithmically in the number of PCs.

5. Performance evaluation

In this section we evaluate the proposed differential algorithms with respect to the optimal solution for unsplittable and splittable input traffic. First, we discuss the simulation setup and the traffic traces. Then, in the following subsections we present the main performance results.

5.1. Simulation setup

To asses the energy saving achieved by the proposed algorithms, a back-end stage configuration exploiting three groups of PCs is used for both the optimal and the differential algorithms. Each group consists of five PCs and each PC in each group has the following specification [6, 17, 30]:

Group I

- Back-end PC routing capacity C_b = 4 Gbps;

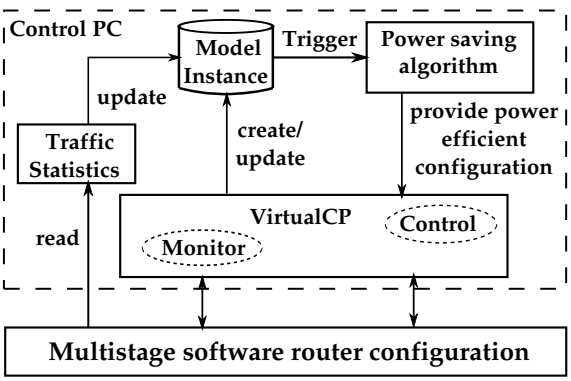


Figure 2: MSSR architecture power saving scheme block diagram

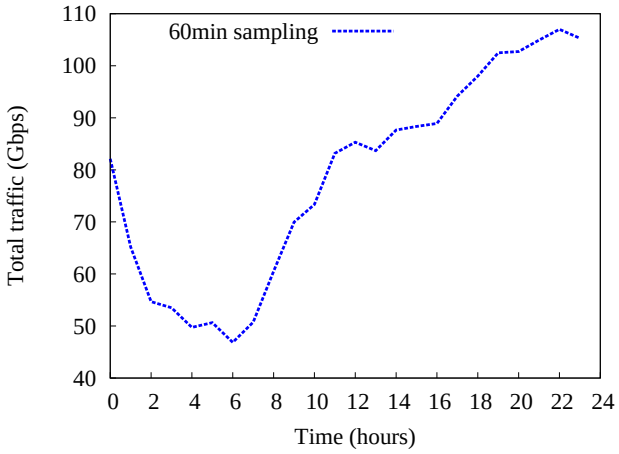


Figure 3: Input traffic trace used in the simulation

- PC power consumption P_b = 60 W;
- Link capacity C_{bl} = 1 Gbps (4 links per PC);
- Link power consumption P_{bl} = 4 W;

Group II

- Back-end PC routing capacity C_b = 8.7 Gbps;
- PC power consumption P_b = 100 W;
- Link capacity C_{bl} = 10 Gbps (1 link per PC);
- Link power consumption P_{bl} = 20 W;

Group III

- Back-end PC routing capacity C_b = 8.7 Gbps;
- PC power consumption P_b = 80 W;
- Link capacity C_{bl} = 1 Gbps (9 links per PC);
- Link power consumption P_{bl} = 6 W;

For the second and third groups, the actual routing capacity (computed according to Eqn. (7)) is limited by the PC routing capacity being the PC total link capacity larger than its routing capacity. Instead, for the first group, the router capacity and the total link capacity are the same. According to (8), the MSSR has a maximum routing capacity of 107 Gbps. To fully utilize this capacity, 11 LBs each with two (one internal and one external) 10 Gbps link are needed. Assuming 80 W of power consumption for each LB and 200 W for the switch, without any power saving scheme the architecture consumes 2.73 kW (1.65 kW by the back-end routers).

5.2. Traffic Traces

In the simulation, we used two different traffic traces to derive the input traffic load. The first one is based on a captured traffic scenario from a university router in Twente. To analyze a large MSSR architecture, the traffic absolute values were scaled up while keeping constant the relative traffic amount at each sampling instant. Traces were aggregated to create samples of 60 min duration, and the traffic volume was averaged over a week to get a per day volume statistics. Fig. 3 shows the volume trace.

This trace was used to generate unsplittable input traffic and for sampling based reconfiguration initialization of the BEPCs. The unsplittable traffic is mainly composed by small and large size flows. A parameter α (and $\beta = 1-\alpha$) defines the proportion of small to large size flows. To assess the impact of flow size, we defined three scenarios: One dominated by large flows, one for small flows and one mixed scenario. Thus, the following α were used:

- α = 0.2: predominant large size flows
- α = 0.5: same proportion of small and large size flows
- α = 0.8: predominant small size flows

For each α, at each sampling time, flow size is randomly generated until the summation of the traffic generated by the flows is equal to the total traffic volume. Small flows size are uniformly generated between 1 Mbits and 10 Mbits, large flows range between 50 Mbits and 100 Mbits, according to the flow size observed in the Internet. The first scenario is more difficult to manage, because packing larger size flows results in reduced efficiency.

We present results for unsplittable input traffic generated for variable α. Note that at full load all flows might not be served even under the optimal packing because of some capacity wastage in each PCs due to quantization.

The second traffic trace is a synthetically generated traffic trace constantly increasing from 10% to 100% of the maximum architecture capacity (C_{MSSR}), used for the splittable traffic scenario.

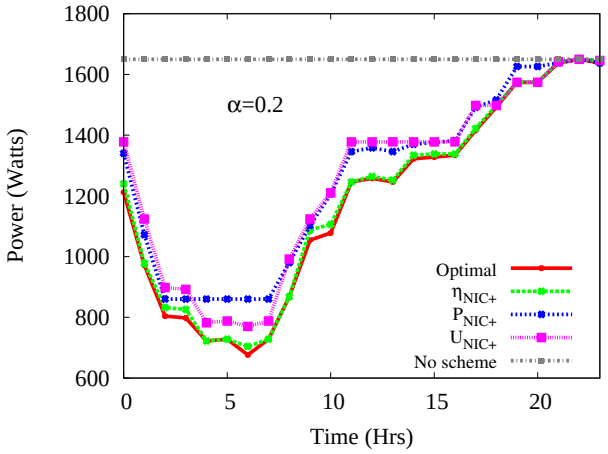
5.3. Evaluation metrics

Based on the above simulation scenario, we compare the proposed energy saving differential algorithms with respect to the optimal algorithm and with a heuristic version not exploiting the sorting phase. The energy consumption of the back-end routers over a given period is obtained from the power dissipation curves as:

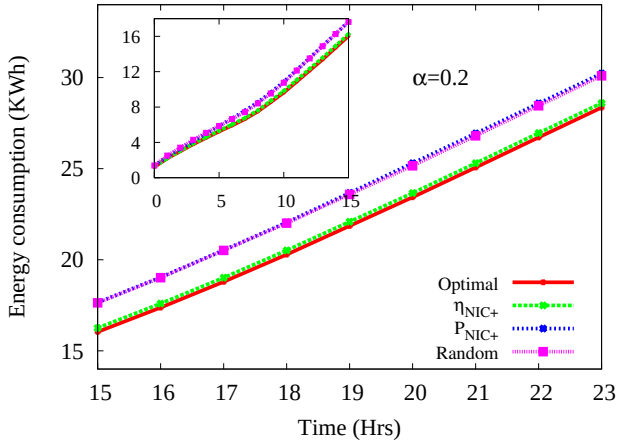
$$\text{Energy} = \text{power} \times \text{time} \tag{14}$$

Both the power dissipation and the derived energy consumption of the BEPCs configuration defined by each algorithm are reported. We also focus on the difference among configurations, expressed as the different number of PCs activated in two consecutive BEPCs configurations, as an indirect measure of service interruption or increased delays. Finally we consider the two router sorting policies NIC+ and NIC- to highlight the impact of considering the link power consumption when sorting PCs. In summary, results from the following algorithms are presented for both splittable and unsplittable input traffic:

- Optimal: optimally solved BEPCs configuration. We used CPLEX [31] to solve Eqn (2) - (6)



(a) Load proportional power dissipation



(b) Energy consumption over 24 Hrs

Figure 4: Performance for unsplittable traffic input (NIC+ sorting policy)

- η_{NIC+}: differential heuristic that sorts devices according to their efficiency and considers link consumption in PC sorting
- η_{NIC-}: same as η_{NIC+} but without considering link consumption when sorting PCs
- P_{NIC+}: differential heuristic sorting devices according to their power dissipation and considering also link consumption in PC sorting
- P_{NIC-}: same as P_{NIC+} but without considering link consumption when sorting PCs
- Random: differential heuristics with no device sorting scheme.

The reported results are averaged over 5 runs for unsplittable input traffic, due to the long running time of the optimal solution. For splittable input traffic, results were instead averaged over 10 runs thanks to the relaxed timing constraints for the optimal algorithm.

Results: Unsplittable input traffic

In the unsplittable input traffic scenario, we consider the measured traffic trace depicted in Fig. 3 and the sampling based MSSR reconfiguration, as discussed in Sec. 4.3. Each MSSR configuration is kept for the whole sampling interval.

Fig. 4(a) reports the power dissipation of the BEPCs configurations designed by the various energy saving algorithms for $\alpha = 0.2$; i.e., the large size flow scenario. The differential algorithms (labeled η_{NIC+} and P_{NIC+}) sort BEPCs considering their link consumption as discussed in Sec. 4.2. The curve labeled "No scheme" refer to the dissipation of the MSSR when no energy saving algorithm is applied. The power dissipation of the BEPCs configurations defined by all proposed algorithms is proportional to the load demand. However, different power savings are achieved by the different algorithms. Fig. 4(b) depicts the energy consumption of the BEPCs configuration over

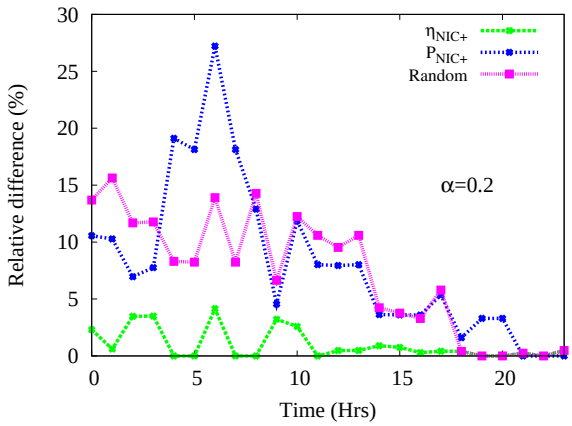
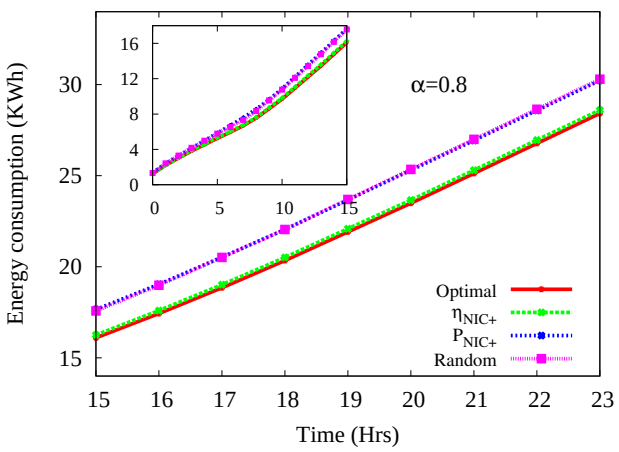


Figure 5: Relative difference in the number of activated BEPCs for the differential vs the optimal algorithm (NIC+ sorting policy)

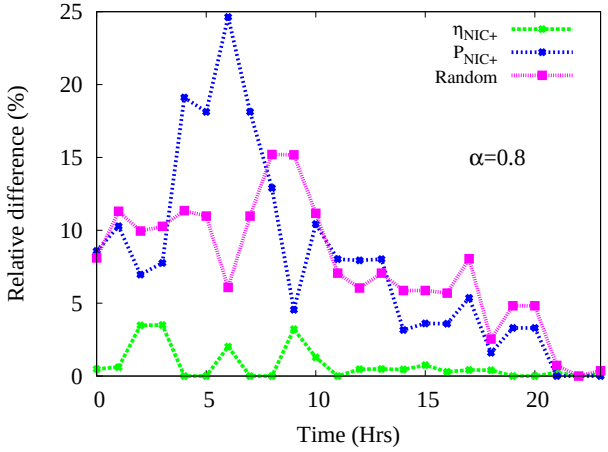
24 hours, as given by Eqn. (14). The figure is magnified over the time range 15 to 23 while the smaller inset in Fig. 4(b) shows the remaining part, for sampling ranging from 0 to 15. The savings that can be achieved by the η_{NIC+} are very close to those of the optimal case, and P_{NIC+} is slightly worse that the unsorted case. The η_{NIC+} presents clear advantages being almost optimal if not for the choice of the last PC for a given configuration. Indeed, when choosing the last PC the packing algorithm may be forced to select the most efficient PC even if the installed unused capacity translates to energy inefficiency.[3]

For what concern the P_{NIC+} algorithm, choosing the PC with smaller power consumption does not directly translate to improved efficiency. Indeed, in the simulated setting, the least consuming PCs are in **group I**, but they show also the smallest capacity. If the traffic demand exceeds the PC's capacity, an-

[3]Comparing the 24 hours energy consumption of this MSSR architecture with the consumption of a commercial router with similar forwarding capacity such as Juniper E120 (39 kWh per day), we observe that the MSSR architecture is more efficient when coupled with the proposed algorithms.

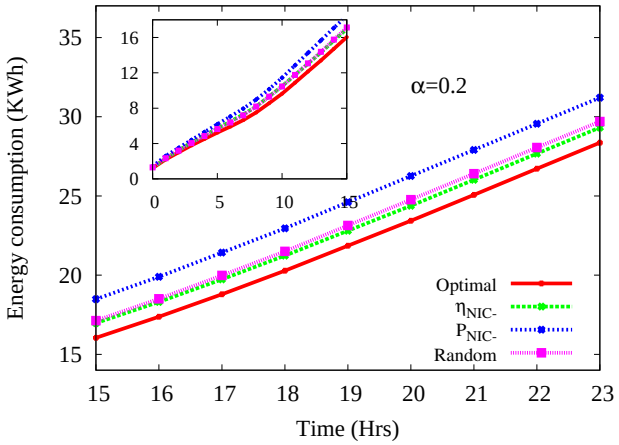


(a) Energy consumption

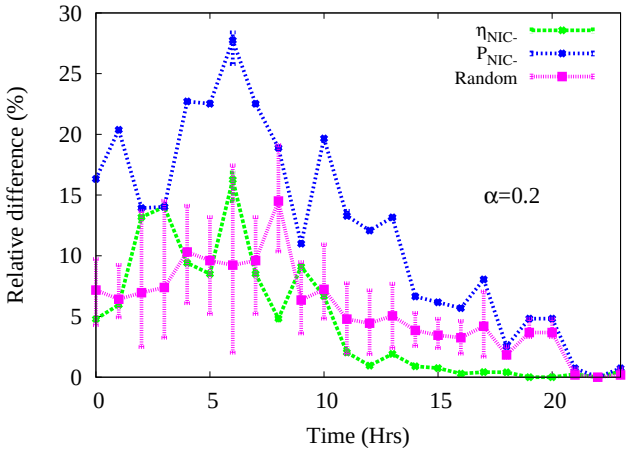


(b) Differential algorithms relative difference with respect to optimal algorithm

Figure 6: Algorithm comparison for unsplittable traffic input (NIC+ sorting policy & $\alpha = 0.8$)



(a) Energy consumption



(b) Differential algorithms relative difference with respect to optimal algorithm

Figure 7: Algorithm comparison for unsplittable traffic input (NIC- sorting policy & $\alpha = 0.2$)

other PC of the same group will be turned on instead of one PC of larger capacity from another group. This results in higher losses because more PCs with smaller capacity are selected.

The performance difference between the differential algorithms and the optimal solution is depicted in Fig. 5, where the relative difference is defined as:

$$\text{Relative difference} = \frac{P_{ALGO} - P_{OPT}}{P_{OPT}} \qquad (15)$$

where P_{ALGO} and P_{OPT} are the power dissipation of the BEPCs configuration defined by the proposed differential algorithms and the optimal algorithm respectively. P_{NIC+} algorithm perform worse at low loads, as shown in the time interval 4am and 8am. For increasing load, the inefficiency of the differential algorithms decreases given the more constrained scenario.

A similar behavior is shown as α moves to 0.5 and 0.8. As depicted in Fig. 6 for $\alpha = 0.8$, the packing efficiency only slightly varies for increasing α. Indeed, the average flow size is very small if compared to the device capacity: The smaller link ca-

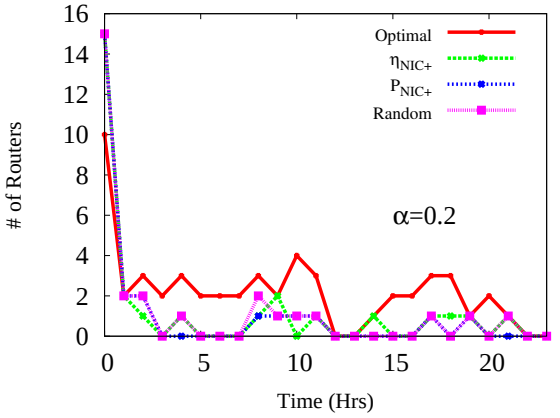
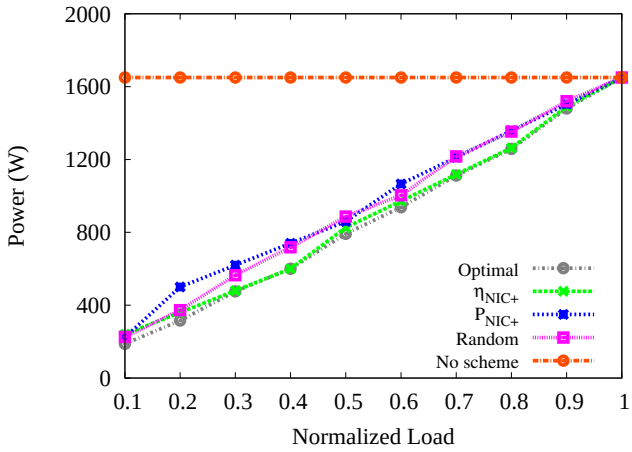
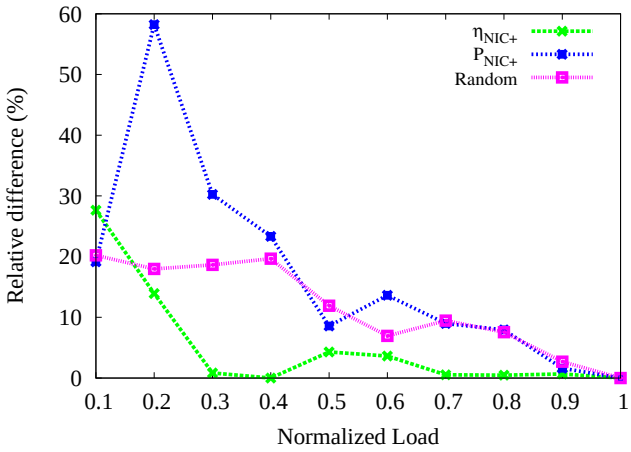


Figure 8: Configuration difference between two consecutive solutions (unsplittable traffic)



(a) Load proportional power dissipation



(b) Differential algorithms relative difference with respect to optimal algorithm

Figure 9: Comparison of different algorithms for splittable traffic input (NIC+ policy)

pacity in the simulation scenario is 1 Gbps, 10 times larger that the largest flow size, with a slight effect of variable α on the packing efficiency. Thus, in the remainder of the paper, we only report results based on $\alpha = 0.2$.

The energy saving capability of both sorting algorithms drops for the NIC- scenario. Fig. 7(a) shows that algorithms based on efficiency sorting are performing better (see label η_{NIC-}) when compared to the other differential algorithms but far from the optimal solution. The inefficiency is due to the fact that most efficient PCs show also high link energy consumptions, which are not taken into account, thus decreasing the packing efficiency.

While the η_{NIC-} sorting based differential algorithm still performs fairly well, the performance drop is larger than in the unsorted scenario due to the missing information on link consumption. The effect is more visible at low loads, because according to the sorting scheme, **group III** PCs are given priority. These devices have higher total link consumption if compared to the other groups and thus their selection worsen performance. The algorithm that sorts devices according to their power consumption performs even worse under NIC- (see label P_{NIC-} in Fig. 7) for reasons similar to those of the NIC+ case. The reported min-max values show negligible variability over the 5 runs both for η_{NIC-} and P_{NIC-} for most loads. The uncertainty level is much less in η_{NIC+} and P_{NIC+} as well as for splittable algorithms because of much better packing. Hence, we do not report the min-max bars in the plots. The higher variability in the Random algorithm is due to the fact that the routers are shuffled at each run.

Although the energy saving achieved by the optimal approach is superior, results demonstrate that the differential algorithms have comparable energy efficiency. Indeed, the energy savings achieved by differential algorithms (η_{NIC+} for instance) reach 11 kWh per day for a single MSSR device, a relative saving of 27% if compared to the no energy saving scheme. Furthermore, the differential algorithms gracefully modify the current MSSR configuration minimizing service interruptions. Fig. 8 supports the claim that the differential algorithms are less disruptive compared to the optimal reconfiguration. The plot reports the difference in the number of PCs between two consecutive configurations for the optimal and differential algorithms based on efficiency sorting. Similar results hold for power based sorting algorithms. The maximum configuration difference is at simulation startup because the initial (zero time) configuration refer to a scenario where only one PC is switched on. Disregarding the initial setup, the proposed differential algorithms show reconfiguration including at most 2 PC, largely less than the optimal solution, as expected.

Results: splittable input traffic

For splittable input traffic scenario we consider a synthetic normalized input load increasing from 0.1 to 1. We apply the threshold based reconfiguration initialization described in Sec. 4.3.

Fig. 9(a) compares the energy savings that can be achieved by the NIC+ policy for splittable input traffic. A large saving up to 1.53kW is obtained at low loads by the heuristic, with respect to the curve labeled ”No scheme” that refers to the maximum capacity MSSR configuration.

Similarly to the unsplittable input traffic case, the differential algorithms make the energy consumption of the BEPCs load proportional. Fig. 9(b) reports the relative difference between the proposed differential heuristics and the optimal algorithm under splittable input traffic scenario. The η_{NIC+} heuristic well approximates the optimal solution, as in case of unsplittable traffic, with a worst case error of about 30%. The worst performance happens at low load for the same reasons described in the case of unsplittable input traffic scenario. The efficiency based sorting algorithm significantly enhances the heuristic performance both in the unsplittable and splittable input traffic, as shown by the worse performance of the random heuristic. On the other hand, sorting via power consumption does not provide significant energy benefits.

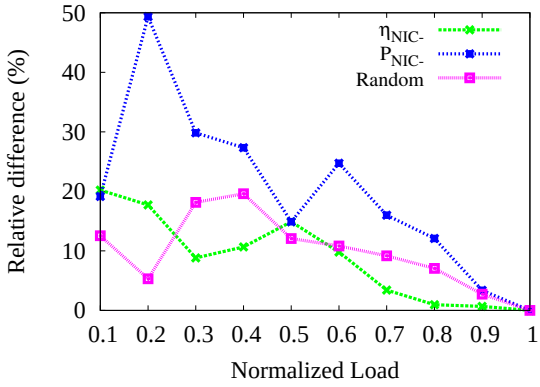


Figure 10: Relative difference of differential algorithms with respect to optimal algorithm (NIC- policy, splittable input traffic)

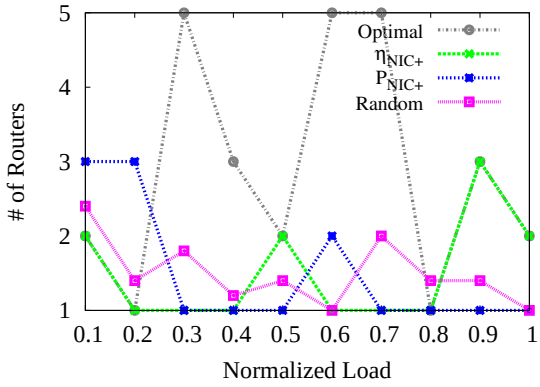


Figure 11: Configuration difference between two consecutive solutions (splittable input traffic)

As in the unsplittable input traffic case, a comparison of Fig. 9(b) and Fig. 10 highlights the importance of taking into account link power in sorting BEPCs, because NIC+ policy largely outperforms the NIC- sorting policy.

Finally, the configuration difference between two consecutive solutions under splittable input traffic scenario is depicted in Fig. 11. The number of newly introduced PCs from one configuration to the next is minimal in the proposed differential algorithms if compared to the optimal solution.

6. Conclusions

In this paper we proposed differential algorithms that, besides having obvious scalability advantage, reduces service disruption and minimize delay as opposed to the optimal algorithm. We also compared the performance of the proposed algorithms with respect to the optimal algorithm under two different traffic scenario.

The differential algorithms result in a load proportional energy consumption with savings comparable to those of the optimal algorithm. Indeed, energy saving can reach 27% and 57% with respect to the case when no energy saving scheme is deployed for unsplittable and splittable input traffic respectively. The simulation results show that the differential algorithm that sorts devices according to their efficiency outperforms the other heuristics. On the other hand, the intuitive approach of sorting devices according to their power consumption to save energy results in reduced saving.

Although the energy saving schemes are defined for a specific MSSR architecture, the proposed algorithms could easily be adapted to other distributed architectures such as data centers or distributed routers with multiple line cards.

Appendix A. Proof of NP-hardness

The combined problem that considers PCs and links together to optimize the energy consumption of an MSSR architecture is not easily mappable to any of the well-known NP-hard problems. Thus, we consider a simplified version of the combined problem where the links are not considered, assuming that they are consuming a negligible amount of energy. In this case the router selection is sufficient to optimize the overall router consumption.

The simplified version of our problem is as follows:

$$\textbf{min} \quad \sum_b P_b \alpha_b \tag{A.1}$$

$$\textbf{s.t.} \quad \sum_b \delta_{bk} = 1, \quad \forall k \tag{A.2}$$

$$\sum_k \delta_{bk} S_k \leq C_b \alpha_b, \quad \forall b \in B, \forall k \tag{A.3}$$

$$\alpha_b \in \{0,1\}, \quad \delta_{bk} \in \{0,1\} \vee [0,1]$$

If flow $T_k \in T$ is unsplittable, this problem can be directly mapped to the Generalized Cost Variable Sized Bin-Packing problem [32], where the items are represented by the flows and the bins by the routers. Since that problem is NP-hard, then the unsplittable version of our simplified problem is NP-hard as well as the complete form which introduces the link optimization.

On the other side, if T_k is splittable, then the problem can be mapped to the Knapsack problem [29], but looking at the allocation problem from a different perspective: the items to be packed are the routers, meanwhile the knapsack is the traffic (which we consider as a single aggregated entity). But in this case the mapping is not direct as in the previous case, since major differences are present in the formulation of the problems. Indeed, the Knapsack problem is as follows:

$$\textbf{max.} \quad \sum_b v_b \alpha_b \tag{A.4}$$

$$\textbf{s.t.} \quad \sum_b w_b \alpha_b \leq W \tag{A.5}$$

$$\alpha_b \in \{0,1\}$$

where v_b and w_b are respectively the *value* (later defined) and the capacity of the router b, and W is the aggregated router capacity needed to satisfy traffic T_k. The main differences of our problem with respect to the standard knapsack problem are:

1. the Knapsack is a maximization problem, while our problem is a minimization problem, due to the usage of values instead of costs.

2. in the KNAPSACK the total size of selected items must not exceed W, meanwhile in our case it must be at least equal to W to allocate enough capacity to forward input traffic T_k of size S_k.

The first issue is solved by defining a correct transformation from energy costs P_b to values v_b exists (e.g. $v_b = \frac{1}{P_b}$). The second issue is solved by searching for a minimum $W \geq S_k$ such that $\sum_b w_b \alpha_b \leq W$ and $\sum_b w_b \alpha_b \geq S_k$. If an algorithm to select the optimal W exists, then our problem can be mapped directly to the KNAPSACK problem which is known to be NP-hard [29]. Thus, the splittable version of our simplified problem is NP-hard as well.

Finally, a simple algorithm to select the optimal W is based on the iterative solution of a sequence of KNAPSACK problems, starting with $W = S_k$ we increase W by a small amount at every step until the condition in Eqn. A.6 is verified:

$$\sum_b w_b \alpha_b \leq W \wedge \sum_b w_b \alpha_b \geq S_k \tag{A.6}$$

At this stage, the optimal W is obtained.

Acknowledgments

This research was funded by the Italian Ministry of Research and Education through the PRIN SFINGI (SoFtware routers to Improve Next Generation Internet) project. This work was developed while Luca Giraudo was holding a Post Doc position at Politecnico di Torino.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 2011–2016." [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html [Accessed: May, 2013]

[2] M. Webb *et al.*, "SMART 2020: Enabling the low carbon economy in the information age," *The Climate Group. London*, vol. 1, no. 1, p. 1, 2008.

[3] S. Mingay, "Green IT: the new industry shock wave," *Gartner RAS Core Research Note G*, vol. 153703, p. 2, 2007.

[4] A. Bianco, J. M. Finochietto, M. Mellia, F. Neri, and G. Galante, "Multistage switching architectures for software routers," *Network, IEEE*, vol. 21, no. 4, pp. 15–21, 2007.

[5] M. Petracca, R. Birke, and A. Bianco, "HERO: High-speed enhanced routing operation in Ethernet NICs for software routers," *Computer Networks*, vol. 53, no. 2, pp. 168–179, 2009.

[6] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: exploiting parallelism to scale software routers," in *ACM SOSP*, vol. 9. Citeseer, 2009.

[7] L. Rizzo, "Netmap: a novel framework for fast packet I/O," in *Proceedings of the 2012 USENIX conference on Annual Technical Conference*. USENIX Association, 2012, pp. 9–9.

[8] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: a GPU-accelerated software router," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 195–206, 2010.

[9] "Intel DPDK Overview." [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/presentation/dpdk-packet-processing-ia-overview-presentation.pdf

[10] "T320 Power System Electrical Specifications." [Online]. Available: http://www.juniper.net [Accessed: May, 2013]

[11] L. Chiaraviglio, M. Mellia, and F. Neri, "Energy-aware backbone networks: a case study," in *Communications Workshops (ICC), IEEE International Conference on*. IEEE, 2009, pp. 1–5.

[12] A. Bianco, F. G. Debele, and L. Giraudo, "On-line power savings in a distributed multi-stage router architecture," in *Global Telecommunications Conference (GLOBECOM), IEEE*, December 2012.

[13] M. Gupta and S. Singh, "Greening of the Internet," in *Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 19–26.

[14] J. Chase and R. Doyle, "Balance of power: Energy management for server clusters," in *Workshop on Hot Topics in Operating Systems (HotOS)*, vol. 200, no. 1, 2001, pp. 163–165.

[15] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Dynamic cluster reconfiguration for power and performance," in *Compilers and operating systems for low power*. Kluwer Academic Publishers, 2003, pp. 75–93.

[16] K. Rajamani and C. Lefurgy, "On evaluating request-distribution schemes for saving energy in server clusters," in *Performance Analysis of Systems and Software (ISPASS), IEEE International Symposium on*. IEEE, 2003, pp. 111–122.

[17] "Broadcom NetXtreme II network adapter user guide." [Online]. Available: http://www.broadcom.com/docs/support/ethernet_nic/Broadcom_NetXtremeII_Server_T7.8.pdf [Accessed: May, 2013]

[18] T. Heath, B. Diniz, E. V. Carrera, W. Meira Jr, and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM, 2005, pp. 186–195.

[19] C. Fragni and L. H. M. K. Costa, "ECO-ALOC: Energy-efficient resource allocation for cluster-based software routers," *Computer Networks*, vol. 56, no. 9, pp. 2249 – 2261, 2012.

[20] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," *NETWORKING 2009*, pp. 795–808, 2009.

[21] "Energy efficiency for network equipment: two steps beyond greenwashing," white paper. [Online]. Available: http://www.juniper.net/us/en/local/pdf/whitepapers/2000284-en.pdf [Accessed: May, 2013]

[22] A. Bianco, F. G. Debele, and L. Giraudo, "Energy saving in distributed router architectures," in *Communications (ICC), IEEE International Conference on*, IEEE, 2012, pp. 2951–2955.

[23] J. C. C. Restrepo, C. G. Gruber, and C. M. Machuca, "Energy profile aware routing," in *Communications Workshops (ICC), IEEE International Conference on*. IEEE, 2009, pp. 1–5.

[24] "SPECpower_ssj2008 Results." [Online]. Available: http://www.spec.org/power_ssj2008/results/power_ssj2008.html

[25] L. D. Gray, A. Kumar, and H. H. Li, "Workload characterization of the specpower_ssj2008 benchmark," in *Performance Evaluation: Metrics, Models and Benchmarks*. Springer, 2008, pp. 262–282.

[26] E. G. Coffman Jr, M. R. Garey, and D. S. Johnson, *Approximation algorithms for bin packing: a survey*. PWS Publishing Co., 1997, pp. 46–93.

[27] J. Kang and S. Park, "Algorithms for the variable sized bin packing problem," *European Journal of Operational Research*, vol. 147, no. 2, pp. 365–372, 2003.

[28] S. S. Skiena, *The algorithm design manual*. Springer, 1998, vol. 1.

[29] S. Martello and P. Toth, *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[30] "Approximate desktop, notebook, & netbook power usage." [Online]. Available: http://www.upenn.edu/computing/provider/docs/hardware/powerusage.html [Accessed: May, 2013]

[31] "IBM ILOG CPLEX optimization studio." [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/ [Accessed: May, 2013]

[32] L. Epstein and A. Levin, "An APTAS for generalized cost variable-sized bin packing," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 411–428, 2008.