POLITECNICO DI TORINO

SCUOLA DI DOTTORATO
Dottorato in Ingegneria Elettronica e delle Comunicazioni
XXIV ciclo

Tesi di Dottorato

Distributed Software Router Management

Fikru Getachew Debele

Tutore
prof. Andrea Bianco

Coordinatore del corso di dottorato
prof. Ivo Montrosset

February 28, 2013

Distributed Software Router Management

by
Fikru Getachew Debele

Submitted to the Department of Electronics and Telecommunication
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
Politecnico di Torino

February 28, 2013

Acknowledgements

Dear God, thank you so much that you have been walking along my side during the tough times I passed through during this research work, and indeed, throughout my life.

One of the joys of completion is to look over the journey past and remember all who have been with me along this long but fulfilling road.

First and foremost, I would like to express my heartfelt gratitude to Prof. Andrea Bianco for his constant and professional guidance. You showed me how to do research and your support uphold me to finish my study. I am also grateful to all other people at TLC that I have learned to know over the years and who have let me take part of their professional experience during seminars and discussions.

I have been very fortunate to get involved in a research group that beleives in teamwork. I want therefore to thank all the people I have collaborated with in these years. Among them, special thanks go to Robert Birke, who really took the time to familiarize with my research and come up with new and inspiring research directions during early days of my Ph.D. study. Many thanks also to Luca Giraudo and Nanfang Li with whom I happily collaborate during some of my research activities.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. My research work was supported by the Italian Government for the first 3 years and then by SFINGI project.

Last, but not least, I would like to thank my family for their love and understanding. More than ever, I would like to express my love to the most important persons in my life: my lovely wife, Tigist Seyum, and my wonderful child Yegeta Fikru.

Abstract

With the stunning success of the Internet, information and communication technologies diffused increasingly attracting more uses to join the the Internet arsenal which in turn accelerates the traffic growth. This growth rate does not seem to slow down in near future. Networking devices support these traffic growth by offering an ever increasing transmission and switching speed, mostly due to the technological advancement of microelectronics granted by Moore's Law. However, the comparable growth rate of the Internet and electronic devices suggest that capacity of systems will become a crucial factor in the years ahead.

Besides the growth rate challenge that electronic devices face with respect to traffic growth, networking devices have always been characterized by the development of proprietary architectures. This means that incompatible equipment and architectures, especially in terms of configuration and management procedures. The major drawback of such industrial practice, however, is that the devices lack flexibility and programmability which is one of the source of ossification for today's Internet.

Thus scaling or modifying networking devices, particularly routers, for a desired function requires a flexible and programmable devices. Software routers (SRs) based on personal computers (PCs) are among these devices that satisfy the flexibility and programmability criteria. Furthermore, the availability of large number of open-source software for networking applications both for data as well as control plane and the low cost PCs driven by PC-market economy scale make software routers appealing alternative to expensive proprietary networking devices. That is, while software routers have the advantage of being flexible, programmable and low cost, proprietary networking equipments are usually expensive, difficult to extend, program, or otherwise experiment with because they rely on specialized and closed hardware and software.

Despite their advantages, however, software routers are not without limitation. The objections to software routers include limited performance, scalability problems and lack of advanced functionality. These limitations arose from the fact that a single server limited by PCI bus width and CPU is given a responsibility to process large amount of packets. Offloading some packet processing tasks performed by the CPU to other processors, such as GPUs of the same PC or external CPUs, is a viable approach to overcome some of these limitations.

In line with this, a distributed Multi-Stage Software Router (MSSR) architecture has been proposed in order to overcome both the performance and scalability issues of single PC based software routers. The architecture has three stages: i) a front-end layer-2 load balancers (LBs), open-software or open-hardware based, that act as interfaces to the external networks and distribute IP packets to ii) back-end personal computers (BEPCs), also named *back-end routers* in this thesis, that

provide IP routing functionality, and iii) an interconnection network, based on Ethernet switches, that connects the two stages. Performance scaling of the architecture is achieved by increasing the redundancy of the routing functionality stage where multiple servers are given a coordinated task of routing packets. The scalability problem related to number of interfaces per PC is also tackled in MSSR by bundling two or more PCs' interfaces through a switch at the front-end stage. The overall architecture is controlled and managed by a control entity named *Virtual Control Processor* (virtualCP), which runs on a selected back-end router, through a DIST protocol. This entity is also responsible to hide the internal details of the multistage software router architecture such that the whole architecture appear to external network devices as a single device.

However, building a flexible and scalable high-performance MSSR architecture requires large number of independently, but coordinately, running internal components. As the number of internal devices increase so does the architecture control and management complexity. In addition, redundant components to scale performance means power wastage at low loads. These challenges have to be addressed in making the multistage software router a functional and competent network device. Consequently, the contribution of this thesis is to develop an MSSR centralized management system that deals with these challenges. The management system has two broadly classified sub-systems:

i) power management: a module responsible to address the energy inefficiency in multistage software router architecture

ii) unified information management: a module responsible to create a unified management information base such that the distributed multistage router architecture appears as a single device to external network from management information perspective.

The distributed multistage router power management module tries to minimize the energy consumption of the architecture by resizing the architecture to the traffic demand. During low load periods only few components, especially that of routing functionality stage, are required to readily give a service. Thus it is wise to device a mechanism that puts idle components to low power mode to save energy during low load periods. In this thesis an optimal and two heuristic algorithms, namely on-line and off-line, are proposed to adapt the architecture to an input load demand. We demonstrate that the optimal algorithm, besides having scalability issue, is an off-line approach that introduce service disruption and delay during the architecture reconfiguration period. In solving these issues, heuristic solutions are proposed and their performance is measured against the optimal solution. Results show that the algorithms fairly approximate the optimal solution and use of these algorithms save up to 57.44% of the total architecture energy consumption during low load periods.

The on-line algorithms are superior among the heuristic solutions as it has the advantage of being less disruptive and has minimal service delay.

Furthermore, the thesis shows that the proposed algorithms will be more efficient if the architecture is designed keeping in mind energy as one of the design parameter. In achieving this goal three different approaches to design an MSSR architecture are proposed and their energy saving efficient is evaluated both with respect to the optimal solution and other similar cluster design approaches.

The multistage software router is unique from a single device as it is composed of independently running components. This means that the MSSR management information is distributed in the architecture since individual components register their own management information. It is said, however, that the MSSR internal devices work cooperatively to appear as a single network device to the external network. The MSSR architecture, as a single device, therefore requires its own management information base which is built from the management information bases dispersed among internal components. This thesis proposes a mechanism to collect and organize this distributed management information and create a single management information base representing the whole architecture. Accordingly existing SNMP management communication model has been modified to fit to distributed multistage router architecture and a possible management architecture is proposed. In compiling the management information, different schemes has been adopted to deal with different SNMP management information variables. Scalability analysis shows that proposed management system scales well and does not pose a threat to the overall architecture scalability.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Internet traffic is growing at faster rate as more and more bandwidth hungry applications and services such as audio and video streaming are deployed. The proliferation of tablets, mobile phones, and other smart devices are driving up the demand for connectivity as well. In addition, with the stunning success of the Internet, information and communication technologies diffused increasingly attracting more uses to join the the Internet arsenal which in turn accelerates the traffic growth. This growth rate does not seem to slow down in near future [1, 2].

Networking devices support these traffic growth by offering an ever increasing transmission and switching speed, mostly due to the technological advancement of microelectronics granted by Moore's Law [3]. However, the comparable growth rate of the Internet and electronic devices suggest that capacity of systems will become a crucial factor in the years ahead. Given the current trend of increase in system capacity and traffic growth, the former will lag by a factor of 10 over the same time period [4].

Besides the growth rate challenge that networking devices face with respect to traffic growth, they have always been characterized by the development of proprietary architectures. This means that incompatible equipments and architectures, especially in terms of configuration and management procedures. The major drawback of such industrial practice, however, is that the devices lack flexibility and programmability which is one of the source of ossification in the Internet [5]. For instance networking devices, more importantly those deployed at the core of the Internet, shows such inflexibility that deploying simple changes to a network to adapt to a traffic demand incurs huge cost due to required upgrade or even equipment replacement. Fixing problems and introducing new features is almost difficult because of unavailability of devices' software source codes.

Thus scaling or modifying networking devices, particularly routers, for a desired function requires flexible and programmable devices. Software routers (SRs) [1] are among the devices that satisfy the flexibility and programmability criteria. Because of the aforementioned challenges related to real hardware routers, the technique of modifying PCs and using them as a router is an appealing alternative. Modifications to a PC include but not limited to attaching two or more network interface cards (NICs) that connect different networks, upgrading the current hardware to improve performance, installing networking applications, etc.

The flexibility and programmability feature of software router comes from the fact that the hardware is available from multi-vendor at low cost and the large availability of open-source software for networking application, such as Linux [6] and Click Modular Router [7] for the data plane, as well as eXtensible Open Router Platform (XORP) [8] and Quagga [9] for control plan. However software routers are not without limitation. Most criticisms to single PC based software routers include limited performance such as routing capability, lack of scalability such as number of interface and lack of advanced functionalities.

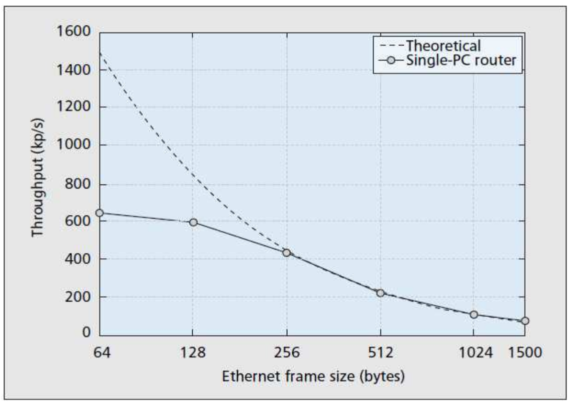
In the following a multistage software router architecture, proposed to address single PC based software router limitations, will be introduced (Section 1.1), and then we point out what is considered the main open issues in multistage architecture (Section 1.2). The analysis of such open issues leads to present the goal and the research questions addressed in this thesis (Section 1.3). Finally, the outline of this thesis is presented (Section 1.4).

1.1 Multistage software router architecture

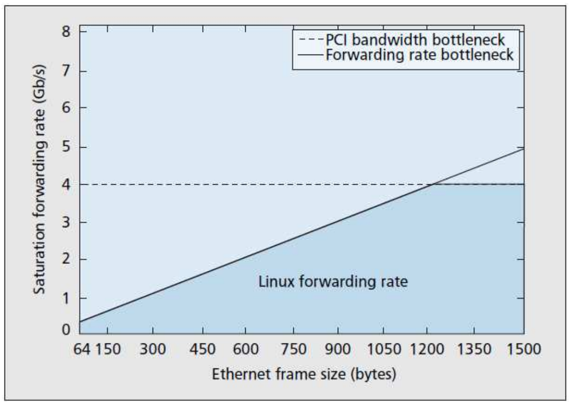
High-end performance can not be obtained easily today with routers based on a single PC architecture. State-of-the-art PC based routers and switches have a potential for switching up to few Gbps if the packet processing is performed by the CPU [10–12] or few tens of Gbps if a specialized packet processing is implemented [13]. While such capacity is more than enough for a large number of applications, by no means comparable to carrier-grade equipments that scales as high as 92 Tbps [14].

The scalability and performance limitation of single PC based software routers arose from the fact that PCs are limited by Peripheral Component Interconnect (PCI) bus and processing capacity as shown in Figure 1.1. Figure 1.1(a) shows throughput for single PC software router having the following specification: PCI - X Intel PRO/1000 Gigabit Ethernet line cards, a single Intel Xeon CPU running at 2.6 GHz, equipped with 1 Gigabyte 128-bit wide, 200 MHz double data rate (DDR) RAM, and PCI-X bus running at 133 MHz, that is, with 8 Gbps bandwidth as

[1]Software router is a term denoting a personal computer, equipped with two or more network cards, designated to do the task of routing packets between networks.



(a) thoughput



(b) saturation forwarding rate

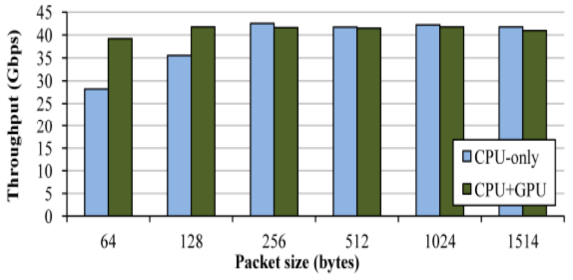
Figure 1.1. Single PC software router performance

the baseline system [10, 15]. The plot clearly depicts the impact of packet size on performance, showing that a single PC can only reach about 640 kpps (kilo packet per second), considering the minimum-size Ethernet frames. For smaller Ethernet frames the bottleneck stem from the maximum packet rate that the PC architecture can forward because of CPU availability and memory-read-latency constraint.

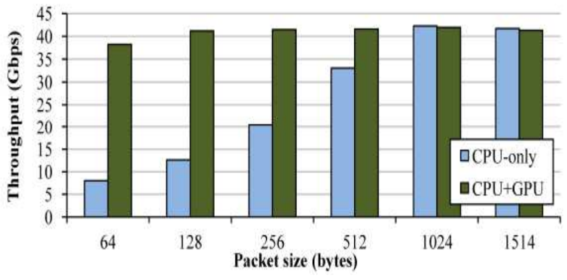
On the other hand, Figure 1.1(b) shows the source of the bottleneck as the packet size increases. This time the number of interfaces increase to eight 1 Gigabyte cards to evaluate the routers performance under multiple flows simultaneously crossing the router. The maximum throughput is limited to about 4 Gbps, which corresponds to the PCI bus bandwidth that must be crossed twice by each packet to be stored into the RAM, processed, and then transmitted [10, 15].

It is possible that the maximum throughput a single PC can sustain is further reduced if more complex operations that increase per-packet processing time must be performed by the CPU; for example, imposing access control list (ACL) rules, network address translation (NAT) operations, and so on.

Optimizing Linux network stack to eliminate per-packet memory management overhead and to process packets in batch and/or offloading core packet processing operations (such as IP table lookup or IPsec encryption) to Graphics Processing Units (GPUs) enhances software routers performance by many folds as shown in Figure 1.2. The platform is called PacketShader [13]. Figures 1.2(a) and 1.2(b) show IP packet forwarding performance of PacketShader for all packet sizes. The CPU+GPU mode reaches close to the maximum throughput of 40 Gbps. While this requires modification in kernel source codes, it demonstrates that software routers have capacity to route few tens of Gbps. Despite this progress, however, software routers based on single PC still lag behind their specialized hardware based routers counterpart.



(a) IPv4 forwarding



(b) IPv6 forwarding

Figure 1.2. Performance measurement of PacketShader

Given these single PC based router bottlenecks, a foreseeable approach to attain high-end performance in software routers is to offload packet processing tasks to external CPUs. That is, to scale software routers to larger size and to achieve higher performance, a distributed architectures composed by several PCs should be sought for [11, 13, 15–17]. In line with this, a Multi-Stage Software Router (MSSR) architecture depicted in Figure 1.3 has been proposed [15]. In building a high performance software router, the multistage architecture exploits classical PCs as elementary switching elements. It is an attempt to scale the capacity of networking devices, routers in particular, in an incremental way. Besides convenient scaling, multistage software router has the advantage of being flexibility and reconfigurability as it implements open-source networking application software both in its control and data plane. The architecture has the following three stages:

i) the layer-2 front-end *Load Balancers (LBs)* acting as the interfaces to the external networks. LBs are also responsible to distribute IP packets to back-end routers. Several algorithms such as simple round-robin scheme, more complex algorithms that, for example, guarantee in sequence routing of packets [18] or balance packets to a particular back-end router based on quality of service (QoS) parameters can be implemented in distributing packets to back-end stage. Load balancing is obtained simply by setting the destination MAC address of the Ethernet frame, so that the correct back-end router Ethernet NIC is addressed. Load balancers implementation can be either hardware-based (such as Field-programmable gate array (FPGA) based) or software-based (PCs running Click Modular Router [7]).

ii) the *back-end PCs* (also referred as *back-end routers*) providing layer-3 routing functionality. Back-end routers are PCs running Linux as the data plane and XORP or Quagga as the control plane and share the same routing table providing several parallel forwarding paths in the architecture.

iii) an interconnection network based on Ethernet switches to interconnect the two

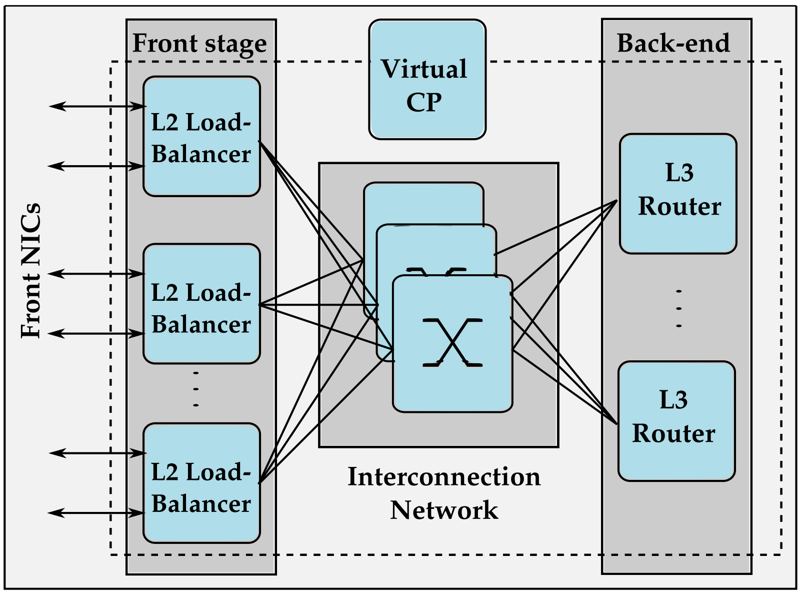


Figure 1.3. MSSR Architecture: the load balancers (first stage), the switch (second stage) and the back-end routers (third stage)

stages: the back-end PCs and the load balancer stages

The data plane operation in the three stages can be visualized as follows: When packet arrives at the router input port, it is

- received by an LB front-NIC, processed by the balancer CPU to perform simple and fast load balancing among back-end routers, and then transmitted by the LBs back-NIC toward the interconnection network;
- switched by the interconnection network to an appropriate back-end router NIC;
- received by a back-end router and processed by its CPU to perform the required packet operations, then transmitted toward the interconnection network;
- switched by the interconnection network to a proper LB back-NIC;
- received by a LB back-NIC, processed by the balancer CPU to switch the packet toward the appropriate front-NIC, then transmitted toward the next-hop node.

From control plane perspective, the multistage architecture comprises an entity named *virtual Control Processor* (virtualCP) that manages, controls and configures the whole architecture [19]. The virtualCP is implemented by choosing one of the back-end routers as a master node and this node runs all the routing protocols. LBs redirect all the routing protocol traffic to this master node. An internal control protocol named DIST, that cooperates strictly with the routing software, has been developed to:

(i) coordinate the routing process among back-end routers and the load balancing function among LBs;

(ii) configure the architecture; and

(iii) provide automatic fault recovery mechanisms.

The virtualCP is also responsible to present the MSSR architecture to the external network as a single, large router by hiding the internal architectural detail.

At the cost of control and management complexity, the multistage software router architecture is able to:

- overcome the performance limitation of single PC based routers by offering multiple, parallel forwarding paths;
- scale the total number of interfaces an MSSR can host, and as a consequence, the router capacity;
- improve router performance by incrementally adding/upgrading internal elements seamlessly;
- recover from faults through automatic reconfiguration of the internal elements;
- provide functional distribution, to overcome single PC CPU limitations, for example, allowing the offloading of CPU intensive tasks such as filtering/cryptography to dedicated PC

Note that routing capacity and number of interfaces scaling simply involves increasing the number and/or capacity of the back-end routers and the front stage PCs respectively. The interconnecting switch can also be duplicated for capacity scaling or if redundancy is required.

1.2 Open issues in multistage software router

Like many networking devices, the MSSR is typically designed for the peak load. Therefore, a high-end MSSR architecture might require tens or hundreds of PCs. Let's demonstrate this through a practical example. Suppose we want to design a MSSR equivalent to a Juniper T320 core router that supports up to sixteen 10 Gbps ports and has 160 Gbps forwarding capacity [20]. The following internal components are available:

- back-end routers with 5.5 Gbps forwarding capacity and equipped with single 10 Gbps interface;
- LBs with two (one internal and one external) 10 Gbps interface;
- a hardware switch with enough capacity to interconnect LBs and routers.

As per this specification, we need 16 LBs, 1 switch and 29 back-end routers to design a 160 Gbps capable MSSR equivalent to T320 router. That is the architecture requires a total of 45 PCs and 1 network switch. It is easy to see that this number increases with performance. As the number of internal devices increase, the multistage software router architecture faces two challenges.

First, control and management complexity increases with the number of internal devices. The virtualCP has to communicate to each device to infer their operation status, update them with control messages, detect any abnormalities and take measure, etc. Furthermore, individual internal devices register their own management information system. Thus, the virtualCP has to collect this information and build a management information system that represents the whole architecture. All these tasks create complexity in controlling and managing the architecture. This complexity problem has partially been solved from the control plane perspective [19].

Second, performance scaling implies a high level of redundancy at the back-end stage which translates into a source of energy wastage during low traffic periods. Thus, at low load MSSR architecture is not efficient from energy cost perspective. For example the T320 equivalent MSSR has 20 back-end routers but only few of them are needed at low loads. From high load perspective, energy consumption could threaten the scalability feature of the MSSR architecture. That is as more and more performance is needed, it might not be realistic to increase the number of internal devices from energy consumption perspective.

Dealing with some of these challenges that the multistage architecture faces is the task of this thesis as detailed in the following subsection.

1.3 Goal, research questions and approach

In light of the reasoning so far, the goal of this thesis is to develop a centralized MSSR management system that (i) collects and compiles a unified management information system, and (ii) resize the MSSR architecture to the input traffic in an efficient way such that the power wastage is minimized during low load periods. In achieving this goal, an answer for the following research questions will be sought-after:

Research Question 1: What is the state-of-the-art in distributed architecture information management and energy saving?

Research Question 2: What are the possible approaches that make MSSR architecture energy consumption proportional to the input load demand?

Research Question 3: Which MSSR configuration best suits for resizing the architecture such that the energy wastage is minimized over a specified period?

Research Question 4: How to build a unified management information base (MIB) that represents the MSSR architecture?

The objective of Research Question 1 is to identify the main contributions and research trends in distributed information management and energy saving techniques in a network so far. A literature study is performed to present a structured overview of the research field.

Research Question 2 tries to look for the right approaches to resize the MSSR architecture to the load demand such that the energy saving achieved will be as close as the optimal solution. The optimal algorithm is based on an optimization problem and two other viable energy saving heuristics are identified: an off-line algorithm named two-step algorithm and an on-line algorithm called a differential on-line algorithm. The two-step algorithm solves the energy minimization problem by splitting the optimal problem into two steps: router and link power optimization steps. While the two steps are optimal individually, the combined solution, however, is not. This divide-and-conquer approach reduces the optimal problem complexity at the cost of solution quality but scales well to a practical multistage router architecture size. The differential on-line algorithm; being a heuristic solution, it has the obvious scalability advantage and also, unlike the optimal and two-step algorithm, it reduces service disruption and/or minimize delay. This is because it builds a new solution on top of existing one which is not the case for the other two algorithms. The energy saving achieved by two-step and differential on-line algorithms are compared to the optimal solution to measure their performance. Results show that the proposed algorithms result in load proportional MSSR energy consumption and fairly approximate the optimal solution.

However, the performance of the proposed algorithms to Research Question 2 is different for different input MSSR configurations. Therefore Research Question 3 is meant to design the MSSR configuration such that the operation of energy saving algorithms will be more efficient. Given a set of group of PCs to be used for MSSR configuration and an input traffic profile over a specified period, the problem is to choose an MSSR configuration that is more flexible to tune such that minimization of the power wastage is maximized during low load periods. The energy saving is computed over the input traffic sampling periods. We propose three MSSR design approaches; namely goal programming, performance-power ratio heuristic and locally optimal design approaches. The proposed design approaches energy efficiency is then compared with other existing cluster design approaches and the optimal solution. It is shown that our design approaches permit up to 10% of energy saving compared to existing design approaches for similar MSSR architecture cost.

Finally Research Question 4 deals with distributed management information gathering and compilation mechanisms. As stated earlier the multistage is composed of large number of autonomous systems that individually register their own management information. As a single device, however, the MSSR requires a management information base that represents the whole architecture. This information base is built from internally dispersed management information bases located in each internal component which requires collection and compilation. This research question, therefore, seeks a mechanism to build a unified management system such that the architecture appear as a single device from management perspective. In the thesis, after identifying MSSR internal network management requirements, an SNMP manager-agent communication model is extended to fit the multistage architecture. The model defines mechanisms to collect data from distributed elements in a reliable way and aggregate the data to a unified view.

1.4 Thesis outline

The thesis structure closely follows the Research Questions. It is organized as follows:

- **Chapter 2: State-of-the-art: Distributed software routers and management** presents a structured overview of the main contributions so far in the field of distributed architecture management. It divides the literature survey into three main categories: software routers, energy saving in networks and distributed information management.

- **Chapter 3: Multistage architecture energy management** presents the different energy saving algorithms. Performance comparison is supported by simulation results [21,22]. **Chapter 4: Energy efficient multistage architecture design**, which closely related to energy saving algorithms, focuses on designing an energy efficient MSSR architecture [23].

- **Chapter 5: Multistage architecture network management** investigates how to create a unified management information base that represents the whole architecture from internally dispersed management information bases local to each internal devices [24].

- **Chapter 6: Conclusions** finally closes the thesis by drawing conclusions and identifying directions for future work.

Chapter 2

State-of-the-art: Distributed software routers and management

As described in Section 1.1 a multistage software router is a distributed architecture that is composed of many internal components. The large number of autonomously running devices could translate into energy consumption and management information is distributed among internal components. This requires additional effort to make the architecture energy efficient and a mechanism to manage the distributed information system.

In the following, different efforts made by the research community to address the issue of energy consumption in a network as well as management of distributed information in similar architecture to multistage will be presented. Software router related research activities so far will be detailed in Section 2.1. Section 2.2 focuses on main contributions in power management in distributed architecture while Section 2.3 presents different schemes used to manage distributed information systems.

2.1 Software routers

Over the years, viability of software routers as an alternative to expensive and proprietary networking devices have been studied extensively. Different researchers also tried to solve the scalability and performance related issue to software routers to make them practical to deploy in large networks.

Andrea et al. [10, 25] assessed the feasibility of building a high-performance IP router out of a common PC hardware and open source operating system. Fig. 2.1 shows the saturation transfer rate for Linux and Click, when the routers are crossed by four traffic flows, either unidirectional (UNI) or bidirectional (BI) for different Ethernet payload sizes. The results are based on a high-end PC with a Super-Micro X5DPE-G2 mainboard equipped with one 2.8 GHz Intel Xeon processor and 1 Gbyte

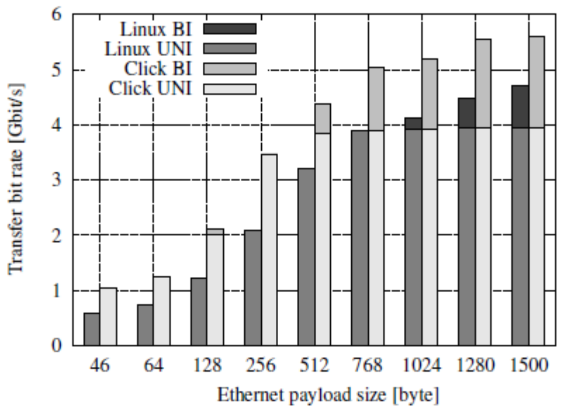


Figure 2.1. Router transfer rate comparison for different packet sizes

of PC2100 DDR RAM consisting of two interleaved banks, so as to bring the memory bus transfer rate to 4.2 Gbyte/s. Eight 1 Gbps Intel PRO 1000 NICs are installed on the router to generate the required number of flows. The result shows that the software router can transfer up to 500 Mbps when handling minimum size packets and up to 5.5 Gbps when handling 1518 byte packets. For minimum-size Ethernet frame, it is not possible to route a single 1 Gbps traffic flow even if the PCI bus bandwidth is 8 Gbps. The main limitation stem from central processing unit (CPU) overloading and from large host-memory-read latency.

A similar work has been presented in [26, 27] but this time the authors used FPGA-enhanced NICs to offload the CPU from performing IP routing and directly transfer packets across the PCI bus, completely bypassing the standard Linux IP stack. Thus the NIC operation is in fast-path routing mode. Packet transfers on the PCI-bus are regulated by a distributed (asynchronous) scheduling algorithm based upon in-band messages exchange among the FPGA-enhanced NICs. This approach

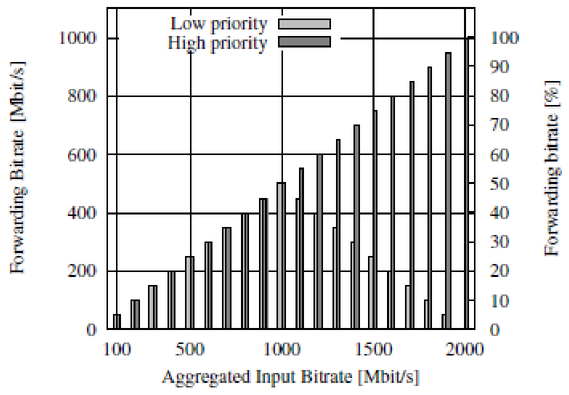


Figure 2.2. Performance of FPGA-enhanced NICs with two-priority traffic

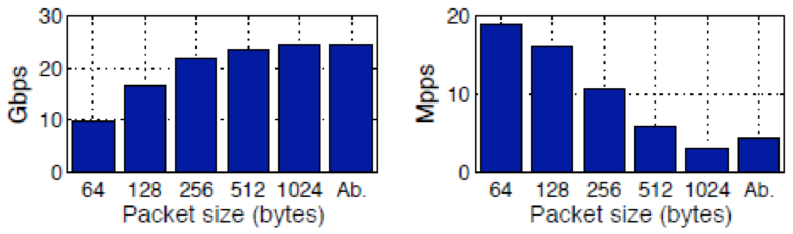


Figure 2.3. Server forwarding rate for minimal-forwarding application as a function of different packet-size distribution

also enables a more sophisticated quality-of-service (QoS) oriented classification and scheduling algorithms to substitute the classical first-in first-out (FIFO) service discipline available on commercial NICs. Fig. 2.2 shows packet classification used jointly with fast-path routing where high-priority packets receive a better services whereas only low-priority packets experience losses.

Multi-core servers performance potential is best exploited by parallelizing packet-processing within a server which enhance single PC based routers [11,17]. The parallelization involves CPU accompanied by memory access (through dedicated memory controllers and buses) and NICs parallelization (through multiple queues) and lower level of software stack are built to leverage packet processing parallelization (through batching). Fig. 2.3 shows the maximum loss-free forwarding rate achieved through this custom server parallelization when running the minimal-forwarding application (a traffic arriving at port i is just forwarded to port j with no additional operation). A 24.6 Gbps forwarding rate is sustained for larger packets without hitting any bottleneck inside the server. That is the performance is limited by the number of NICs installed on the server. In contrast, the server saturates at 9.7 Gbps or 18.96 Mpps for the minimum packet size.

Another customized approach to scale monolithic router performance is a platform called PacketShader [13]. PacketShader optimizes Linux network stack to eliminate per-packet memory management overhead and to process packets in batch and/or offloading core packet processing operations (such as IP table lookup or IPsec encryption) to Graphics Processing Units (GPUs) to enhance single PC based software router performance as shown in Figure 1.2.

Despite the above mentioned efforts, admittedly, monolithic software routers do not scale beyond the 10 Gpbs range if packet processing is performed by the CPU [10–12] or 40 Gbps range if a custom packet processing is implemented [13]. This performance is 2-3 orders of magnitude away from carrier-grade routers that switch up to 92 Tbps [14]. Thus in scaling single PC software router performance a different approach based on router functionality distribution to multiple external CPUs has been proposed recently [11, 15].

Partridge et al. [28] used a combination of multiple line cards (each supporting one or more network interfaces) and forwarding engine cards all plugged into a switched backplane to build a 50 Gbps software router in a custom configuration. It was the first multigigabit router using conventional CPUs that demonstrate routers can continue to serve as a key component in high-speed networks.

Recently Bianco et al. proposed one of the possible distributed software router architecture introduced in Section 1.1 that distribute router functionality into multiple off-the-shelf commodity PCs [15, 16]. The architecture is named multistage software router for the reason that the architecture is composed of three stages: Load balancers, interconnection network and back-end routers (See Section 1.1 for detail). The authors demonstrate that the performance of a multistage software router formed by N_1 load balancers, each equipped with at least three back-end NICs, and N_2 back-end routers scales as:

$$\mathrm{P}_{\min} = \min(N_1 \times 1488, N_2 \times 640)\mathrm{kpps} \tag{2.1}$$

considering a minimum sized packet. The theoretical maximum of 1488 kpps and the single router forwarding limit of 640 kpps when IP routing is adopted are reported in Fig. 1.1(a).

Due to the need of high number of ports and performance, the number of PCs in a multistage architecture could be large. Thus, many internal devices need to be controlled and configured. In addition, the internal interconnection must appear to external network as a single entity which again requires a special coordination. In addressing this issue Bianco et al. [19] proposed a control protocol named DIST that directly interact with software routing suites such as Quagga and XORP to be operating system independent. The protocol is responsible to:

(i) coordinate the routing process among back-end routers and the load balancing function among LBs;

(ii) configure the architecture; and

(iii) provide automatic fault recovery mechanisms.

The multistage software router has also been extended to virtualized environment [29] in looking for increased flexibility (scalability, maintenance and consolidation) and easier introduction of new features such as energy saving mechanism. The authors measured the performance of multistage software router and demonstrated its feasibility in a virtualized environment.

RouteBricks [11, 17] is based on parallelizing router functionality across multiple servers and within each server. Fig. 2.4 depicts a high-level view of a traditional router and RouteBricks architecture. In the architecture there is no centralized components and no component in the architecture need to operate at a rate greater than cR, where c ranges typically from 2 to 3 and R is the servers' packet processing

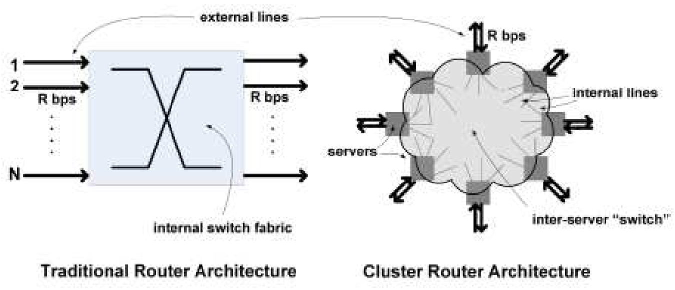


Figure 2.4. High-level view of a traditional router and a server cluster-based router

capability. The number of ports can increase simply by adding servers to the cluster which makes the architecture incrementally extensible in terms of number of ports as well as switching capacity.

A distributed software router architecture named DROP (Distributed SW ROuter Project) is also proposed in [30]. The architecture is partially based on the main guidelines of the IETF ForCES (Forwarding and Control Element Separation) standard [31]. DROP allows building logical network nodes through the aggregation of multiple SRs, which can be devoted to packet forwarding or to control operation. The authors also provided some performance evaluation using different control plane tests as defined in RFC 4062 [32].

2.2 Power management in distributed architecture

The global Internet and its thousands of equipments consume an enormous energy amount and have an impact on global warming. While ICT can make a major contribution to the global response to climate change, by itself however, is responsible for 2% of global carbon emissions - a similar figure to the global airline industry. With the ICT sector growing at faster rate, it is imminent that the CO_2 emission by ICT industry grows as well - estimated to be 6% by 2020 [33, 34].

Telecoms infrastructure and devices contribute about 25% of the 2020 ICT sector carbon footprint. However the ICT technology has a silver lining - the substantial inefficiency in the technology that can be readily addressed. In regard to networking devices, for example, the energy consumption is proportional to the installed capacity rather than the traffic demand. Thus, devices energy consumption is constant irrespective of the variability in the load which obviously raise a question on how to resize the device capacity to a load demand.

While modern hardware and operating systems devised methods at different levels to save energy in electronic devices, the ever increasing capacity demand is generating an increase in power consumption both at device and network level. For instance, while desktop computers produced in 2003 consume roughly 100-120 W when used moderately, the same is also true for today's desktops [35]. On the infrastructure side, the power dissipation of routers has grown with a 1.4 times increase in power dissipation for every doubling of capacity [36]. Therefore a common opinion among network researchers is that the sole introduction of low consumption silicon technologies may not be enough to curb energy requirements, thus the focus is shifting towards different areas. For instance, there is an ample opportunity of saving energy by controlling the devices as well as network topology during low traffic periods, considering that the networks are typically oversized [37].

Therefore, beginning with a position paper by Gupta et al. [38], researchers in IT focused on how to save energy in data network holistically. In this section we discuss state-of-the-art energy saving techniques in a distributed architecture, particularly clusters similar to the multistage software router architecture.

In line with this, Chase et al. [39] proposed an energy-conscious request switching paradigm to reduce energy usage for server cluster during low traffic periods. The switch monitors cluster load and concentrates traffic on the minimal set of servers that can serve it at a specified utilization and latency. This induces the remaining idle servers to step down 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. [40]. In this case a system that dynamically turns cluster nodes on to be able to handle increase in load and off to save energy during low load periods is proposed. A control-theoretic and load distribution algorithm makes the cluster reconfiguration decision by considering the cluster total load and the power and performance implication of changing the current configuration. The technique saves up to 38% in energy.

Power-aware request distribution [41] 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 powering down some servers when the desired quality of service can be met with fewer servers.

The schemes in [39–41] allow energy saving only at the coarse-granularity of the entire server and/or only homogeneous servers are considered. In multistage architecture however, besides the back-end routers are heterogeneous in capacity as well as power consumption, they could have one or more network cards adding another *layer* to an optimization problem. Indeed, the minimization of active internal NICs and/or interfaces on NICs can lead to large energy saving especially in high rated network cards. For instance a 10 Gbps link consumes in the order of 20 watts [42]. If we install four of such NIC per PC, then the consumption is proportional to the PC itself. Therefore the wastage in the network cards needs to be addressed. Moreover, the heterogeneity requires a mechanism to prioritize the back-end routers according to their energy efficiency.

In reducing the granularity, Heath T. et al. [43] designed a cooperative Web server for a heterogeneous cluster that uses modeling and optimization to minimize the energy consumed per request. The approach conserves 45% more energy than an energy-conscious server that was proposed for homogeneous clusters. While their approach is similar to the optimization problem defined in this thesis, we show that optimal solution is an off-line solution that results in service disruption. Going beyond the off-line solution, we propose an on-line heuristic energy saving schemes in a heterogeneous back-end router cluster.

Two forms of energy saving in a network; namely rate adaptation and sleeping of network devices, has been proposed as an appealing alternatives [44]. Researchers exploit these options to save energy in a network through smart topology reconfiguration. Chiaraviglio et al. [45] addressed a network design problem by considering the minimization of the total power consumed by the network. They proposed heuristics to select a minimum set of routers and links to be used in order to support a given traffic demand. The main idea is to power off links and even full routers while guaranteeing QoS constraints such as maximum link utilization. A novel energy reduction approach at the network level by considering nodes capable of adapting their performance to the actual load has also been proposed in [46]. While those researchers focus on energy-aware routing, this thesis focuses on energy-aware load-balancing.

A white paper from Juniper Networks, Inc. [47] reports an energy criteria used to compare energy consumption of different network devices. The normalized energy consumption rating (ECR), measured in watts/Gbps, is defined as

$$ECR = \frac{E}{T} \tag{2.2}$$

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 energy saving algorithms and MSSR design we used similar criteria as one of device selection parameter in setting up a new back-end router configuration.

Finally, Bolla et al. gives a detailed survey [48] on emerging technologies, projects, and work-in-progress standards which can be adopted in networks and related infrastructures in order to reduce their energy and carbon footprint. The authors categorized current approaches to save energy in fixed network infrastructure into three groups:

- Re-engineering approaches - aim at introducing and designing more energy-efficient elements for network device architectures;

- Dynamic adaptation of network/device resources - focuses on adapting packet processing engines and of network interfaces to actual traffic load;
- Sleeping/standby approaches - used to smartly and selectively force idle network/device portion to low power mode.

In this thesis the proposed energy saving techniques fall under sleep/standby category where unused elements (PCs or network cards) set to sleep mode and wake up only if needed.

2.2.1 Energy efficient cluster design

A cluster is a group of stand alone computational resources that work cooperatively together to achieve a single purpose. It is a viable approach to cope with the fast growth of the Internet [1]. Clusters can serve purposes such as infrastructure scalability, high availability, increased computational resources and load balancing. Data centers as well as networking devices such as routers that already use cluster approach are presented in [15, 49, 50].

Server cluster design focuses on server types that currently give the best performance per unit of price, or the types that give the best absolute performance (i.e, highest computational capacity). The Google cluster architecture [49] considers best performance per unit of price as PCs selection criterion to setup a cluster. This approach prioritize commodity-class PCs to high-end multi-processor servers because of their cost advantage. Software reliability, instead of hardware, is the prerequisite for this design principle. The choice of the commodity-class PCs implies larger number of PCs in a cluster to handle the peak load, which results in a high power consumption.

Designers also estimate cluster capacity based on the applications running on it where capacity could mean bandwidth, CPU, RAM, storage etc needed as a server configuration. The aim is to use servers with best absolute performance such that the cluster handles peak load [51, 52] with fewer reliable computers. However, such cluster is far more expensive than the cluster based on commodity-class PCs for the same performance, due to the higher interconnect bandwidth and reliability of the servers [49]. Furthermore, the high capacity granularity implies power inefficiency despite power saving schemes deployment similar to the one proposed in [21, 22]. This is because even for small traffic demand we have to turn on high capacity server which consumes high power.

In this thesis we propose three different multistage software router back-end cluster design approaches with main focus on energy efficiency. The first approach, given a pool of different types of computers, searches for an optimal back-end router configuration that minimize power consumption over a specified period for a given traffic profile. The problem is formulated as a preemptive goal programming. In the second approach we consider performance-power ratio as PCs' selection criteria in building a cluster. In this case the cluster is composed of homogeneous PCs which has the advantage of simple management. The last approach sizes the back-end router configuration based on local optimization. This approach optimize the configuration for the peak load and resizes this configuration for the other load scenarios.

2.3 Distributed information management system

Multistage architecture is build on the principle that the whole architecture will be viewed as a single device to external network. This principle applies for the management information as well. Hence the multistage architecture is required to have a single management information base (MIB), which could be virtual or accessible locally to the internal manager, representing the whole architecture. Management information unification is necessary because each component in the multistage architecture, as an independent element, stores its own management information. That is, the management information is dispersed internally and there is no single management information base (MIB) representing the whole multistage architecture. For example consider the sysUpTime[1]. Each internal component in MSSR has its own sysUpTime but which one represents the up time of the multistage architecture as a whole? Or do we need to generate a new sysUpTime? Thus creating a MIB for multistage software router requires additional complexity to compile a response to external manager request. Hence building a multistage software router MIB is one of the main task of this thesis.

Since the inception of network management in late 1980s, there have been several approaches to manage networks, systems and services. Initial solutions were protocol-based management approaches that adopted the manager-agent paradigm. Manageable resources are modeled through "cluster" of managed objects at different levels of abstraction. Managed objects encapsulate the underlying resource and offer an abstract access interface at the object boundary [54, Ch. 2. p. 63ff]. An application in an agent role accesses the managed objects via the management interfaces. The access interface is defined through formal specification of the relevant managed object classes and the associated access mechanism. Figure 2.5 depicts the management interaction of protocol-based network management approaches. OSI systems managment (OSI-SM) [55] and Internet Simple Network Management Protocol (SNMP) [56] are the two prominent technologies that adopted the manager-agent paradigm.

[1]sysUpTime is the time (in hundredths of a second) since the network management portion of the MSSR architecture was last re-initialized [53].

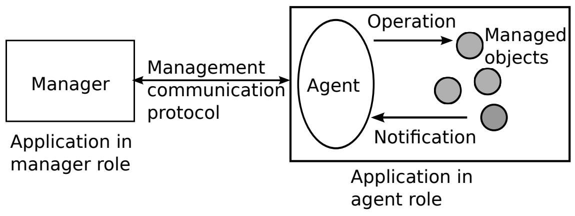

Manager
Application in manager role
Management communication protocol
Agent
Operation
Managed objects
Notification
Application in agent role

Figure 2.5. Protocol-based network management interaction

Other management technologies include CORBA (based on distributed objects technology), eXtensible Markup Language (XML) based network management (recently proposed as an alternative to existing network management solutions) [57–59]. In this thesis however we implement an SNMP based network management for two reasons:

- it is the most widely deployed management protocol and hence easy interoperability
- availability of open source code for our implementation [60]

SNMP is accompanied by a standard describing how to specify managed objects - the Structure of Management Information (SMI) - and a document defining standard managed objects organized in a tree structure - the Management Information Base (MIB-II). It also uses object identifiers (OIDs) suffices for naming to address objects.

An SNMP entity acting as manager role sends request to an entity acting as an agent, which support a different SNMP version than the manager. For interoperability among the different SNMP versions a third network entity called *proxy agent* is required in between the manager and the agent. Similarly, *proxy agent* intervention is also required in converting responses received from an entity acting in an agent role into responses sent to an entity acting in a manager role but supports a different SNMP version. The *proxy agent*, therefore, performs SNMP message header and PDU formats conversion to suite to the message format of the recipient, i.e. the entity beyond the *proxy agent*. Unlike the *proxy agent*, however, the message modification proposed in this thesis is to perform different variable computations and aggregation to create a single-entity image of the multistage architecture which requires the message header and PDU format conversion necessary for its operation before responding to the original request. Our purpose is not dealing with message handling among different SNMP version. Rather it is to address the issue of a single-entity information view of the multistage software router to the external network manager which involves different SNMP variables computation and aggregation. Moreover, our approach deals with many internal agents for a single variable request.

2.4 Conclusion

In this chapter, we provided a survey of the state-of-the-art in distributed architecture management spanning the period 2005 - 2010. Section 2.1 summarizes the main effort made by researchers in scaling software routers. Both single PC based router scaling and distributed router architecture based on multiple CPUs to achieve high-end router performance have been discussed.

For distributed router solution, the energy consumption of the architecture increases with performance since performance scaling is proportional to the number of interconnected devices. Thus Section 2.2 details the energy saving mechanisms adopted in networks in general, and distributed architecture similar to MSSR in particular. The Subsection 2.2.1 presents energy efficient cluster design approaches which contributes to energy efficiency of distributed architectures.

Finally, a distributed architecture information management schemes have been discussed with main focus on SNMP management.

Chapter 3

Multistage architecture energy management

Like many networking devices, the multistage software router is typically sized for peak traffic. State-of-the-art PC-based routers can route only few Gbps [10, 11] if the packet processing is performed by the CPU or few tens of Gbps if a specialized packet processing is implemented [13]. Therefore, the multistage architecture might require tens or hundreds of back-end routers to achieve high-end performance. This performance scaling implies a high level redundancy at the back-end stage which translates to a source of energy wastage at low loads. To reduce such a wastage during low traffic periods, the routing task can be transferred to a subset of back-end routers and all other routers are put in low power state to save energy until they will be required again to manage higher traffic.

Let us consider a realistic scenario: we want to realize a router with 16 interfaces at 10 Gbps (i.e. equivalent to a Juniper T series (T320) core router with 160 Gbps forwarding capacity and up to 2.8 kW power consumption [20]) using a multistage software router with the following internal components:

- back-end routers with 5.5 Gbps forwarding capacity [10] and each equipped with single 10 Gbps interface;
- LBs each with two 10 Gbps interfaces (one connect to external and the other to internal network);
- a hardware switch with enough capacity to interconnect LBs and routers.

Then the architecture will be composed of 16 LBs, 1 switch and 29 back-end routers. If we assume LBs and routers are running on PC with idle power consumption equal to 60 Watts (approximate lower side idle power consumption for today computers), then this architecture consumes 2.7 kW in idle state excluding the switch and the

NICs on back-end routers. However, the energy saving mechanism proposed in this chapter can reduced the architecture consumption down to 1 kW in idle state, which is the equivalent consumption of 16 LBs and 1 back-end router.

While back-end routers and switch are redundant during low traffic periods, LBs are not because they act as external interfaces (which must stay active to guarantee external connectivity). Energy saving in LBs could be achieved if operating at network level, where the whole network energy consumption is optimized by redirecting the traffic over a subset of routers, not when operating at the device level [45]. Therefore saving in LBs is not discussed in this thesis.

The main contributions of this chapter are the MILP formulation, proposal of an off-line algorithm called two-step algorithm and an on-line differential algorithm to solve MSSR energy wastage problem. The analysis of the proposed algorithms are also presented. Thus the chapter is divided into two major topics: Off-line algorithm and on-line algorithm to save energy in MSSR. In Section 3.1 we describe the energy saving problem in multistage architecture and give a detailed formulation of the problem followed by an off-line energy saving algorithms. Performance evaluation of the two-step algorithm will be discussed in Subsection 3.1.3. Highlighting the advantages and disadvantages of off-line approaches, Section 3.2 presents an alternative: an on-line algorithm. Computational complexity and implementation issues are also briefly described. Subsection 3.2.4 discusses simulation results of on-line algorithm and finally conclude the chapter in Section 3.3.

3.1 Off-line algorithms

In light of the Research Question 2 discussed in Chapter 1, we propose a mixed integer linear programming model (MILP) to select a subset of back-end routers, characterized by different power consumption and routing capacity,[1] so as to minimize overall power consumption while satisfying the traffic demand. The problem is a typical resource allocation problem: objects (the traffic) have to be stored in a set of bins (the routers) while minimizing a total cost (the power consumption). Thus, while our research focuses on multistage software router, the proposed technique can be applied to many load distribution scenarios such as a computational clusters, where is it possible to identify a set of resources and a set of containers with an associated cost and capacity (e.g. jobs to workers or tasks to processors).

[1]Routing capacity for a PC is defined as the amount of traffic it can process and forward per second, measured in bits/sec

3.1.1 Problem definition and MILP formulation

The redundancy in the back-end routers designed to handle the peak hour traffic may result in energy waste under low traffic periods. We design mechanisms to reduce energy consumption by adapting the number of back-end routers to the currently offered traffic load.

The proposed saving mechanism is based on an optimization technique which requires the virtualCP to collect periodically the information about the components of the architecture and the incoming traffic. In this section we use different parameters and variables to describe the system as a *Mixed Integer Linear Programming* (MILP) model and we introduce them here to clarify the notation:

- **Input**: the total input traffic to the multistage router, $T \in \mathbb{R}$
- **Routers**: B is the set of back-end routers. $\forall r \in B$, $P_r \in \mathbb{R}$ is the router power consumption (excluding line cards) and $C_r \in \mathbb{R}$ is the routing capacity
- **Links**: L is the set of all internal links. $L_r \subseteq L$ is the set of internal links connected to router r. $\forall r \in B$, $\forall l \in L_r$, $P_{rl} \in \mathbb{R}$ is the link power consumption and $C_{rl} \in \mathbb{R}$ is the link capacity
- **MILP variables**: $\forall r \in B$, α_r is a router selection variable (equal to 1 if the router is activated, 0 otherwise). $\forall r \in B$, $\forall l \in L_r$, β_{rl} is the link selection variable (equal to 1 if the link is used, 0 otherwise). Finally, t_{rl} is a portion of traffic T to be forwarded by router r on its link l.

In the problem formulation we have to take care of different details and different assumption about the system, ranging from the input traffic to the power consumption of a link. The key points considered in our work are as follows:

1. *Traffic measurements are available at the virtualCP*: we do not focus on the issue of measuring or estimating the incoming traffic and collecting measurements. We assume that the virtualCP is able to obtain reliable aggregate measurements on the input traffic.
2. *The input traffic T is splittable among the back-end routers*: every packet is managed independently by LBs and sent to a different back-end router. This may create out-of-sequence delivery of packets belonging to the same flow, unless reordering is envisioned at LBs.
3. *Routers and NICs energy consumption*: the router and its cards are optimized separately. That is while a router is used in forwarding traffic on some of its interfaces during low load periods, the remaining network cards can be turned OFF to contribute to energy saving.

4. *Single link per card scenario*: we focus on single link per card scenario. However, a back-end router can be connected to the switch with more than one link. We further assume that techniques such as smart power down [61] will put a card to low power mode upon a decision taken by the controller to turn off the link on that card. Therefore, we represent the combined power consumption of the card and the link l on a given router r by $P_{rl} \in \mathbb{R}$

5. *ON-OFF power model for the back-end routers and links*: measurements show that power consumption of network devices and links can be approximated by a linear model [46, 62] with an initial step. Indeed, most of the energy consumption is due to the activation of the resource, whereas the remaining part depends linearly on the actual load. To keep the problem formulation as simple as possible, we chose the ON-OFF model: the energy consumption does not depend on the actual resource load, but it is either zero or a constant value.

6. *Off-line solution*: the energy saving problem is solved using an *off-line* algorithm: the problem is not taking into account the evolution of the input traffic, but it is designed to give the best solution given a specific input traffic. Solutions obtained with different input traffic loads are independent, i.e., the sets of activated routers under similar loads may be completely disjoint. As a result, under frequently changing input traffic, the multistage router configuration may be *"unstable"*: the set of active routers may change completely even for small variations of the input traffic load. Furthermore, the virtualCP must manage state transitions (e.g. turning on and off back-end routers due to a new solution) in a non-disruptive way. A feasible mechanism would be to turn on all PCs in current and in future solutions, then to redirect the traffic to the subset of routers belonging to future solution only and finally to turn off the idle routers belonging exclusively to the first solution. This solution compromises the optimal solution. Another alternative could be an on-line differential algorithm that builds a new configuration solution on top of an existing one (See section 3.2 for detail)

Based on the fourth and fifth assumptions mentioned above, the maximum power consumption of a back-end router r is given by:

$$P_r + \sum_{l} P_{rl} \tag{3.1}$$

Thus, the multistage software router energy saving scheme can be formalized as a

MILP problem as follows:

$$\min \quad P_{combined} = \textstyle\sum_r (P_r \alpha_r + \sum_l P_{rl}\beta_{rl}) \tag{3.2}$$
$$\text{s.t.} \quad \textstyle\sum_r \sum_l t_{rl} = 1 \tag{3.3}$$
$$\textstyle\sum_l t_{rl} T \leq C_r \alpha_r \quad \forall r \in B \tag{3.4}$$
$$t_{rl} T \leq C_{rl}\beta_{rl} \quad \forall r \in B, \forall l \in L_r \tag{3.5}$$
$$\alpha_r \geq \beta_{rl} \quad \forall r \in B, \forall l \in L_r \tag{3.6}$$
$$\alpha_r, \beta_{rl} \in \{0,1\}, t_{rl} \in [0,1] \tag{3.7}$$

In the MILP formulation, (3.3) ensures that all the input traffic T is served, while (3.4) and (3.5) make sure the capacity constraints of each router (C_r) and link (C_{rl}) are not violated. (3.6) ensures that router r is active if at least one of its links is chosen to carry some traffic.

Equations (3.2)–(3.7) define a MILP problem that optimizes the multistage architecture power consumption, considering both routers and NICs. Hence, we refer to it as the *combined problem* in the next sections. The problem is NP-hard, and it cannot be mapped easily in its complete form to a well-known problem as demonstrated in Appendix A. Thus, exact methods can only be used to solve small size cases.

3.1.2 Two-step approach

The combined problem cannot be mapped directly to problems with well known solutions, although it is similar to some classical problems in the area of resource allocation (e.g. Knapsack and Bin-Packing [63]). Being NP-hard, the combined problem is complex because it jointly optimizes routers and links and it is unsolvable for large size. Indeed, in Table 3.1 we report the maximum size of the multistage router as the number of back-end routers, for a given number of interfaces per router, for which we were able to obtain a solution in reasonable time (e.g. 5 minutes, the default SNMP collection interval time which can be the basis for an estimation of the current traffic load). It is clear from Table 3.1 that as soon as the number of interfaces per router grows the maximum number of routers drops quickly in the combined problem case. Thus, the combined problem is seriously limited in scalability by the number of interfaces per router; since the number of interfaces should be easily around 10 in a normal PC, then the size of the multistage router would be limited among 12 to 20 routers, which may not be enough even in the Juniper T320 example introduced earlier in this chapter.

Since the combined problem is not scalable and cannot be used even for small size multistage routers with few links per back-end router, we focus on heuristics solutions to improve scalability. We have two elements (i.e. routers and links) involved in the optimization and hence we took advantage of the problem structure to split the combined problem in two parts, using the *divide and conquer* approach. We name this solution the *two-step* problem: in the first step, the *router optimization* problem, we focus on routers only, whereas in the second step, the *link optimization* problem, we optimize the number of links on each router selected in the first step.

This approach is much more scalable than the combined problem, since the single steps are smaller in size and easier to solve than their parent problem. As shown in Table 3.1, the two-step problem is two order of magnitude more scalable than the combined problem. We didn't try to solve the two-step for more than 716 back-end routers, the maximum size we were able to solve in the case of combined problem. Indeed, as detailed in Appendix A, the single steps are still NP-hard, but they are easily mappable to well-known problems. Thus, we can take advantage of the existing heuristics and approximation algorithms available in the literature to solve them.

On the other side, the two-step approach is not optimal since the divide and conquer approach implies that we are using a greedy approach. Even though the single steps are optimal, the output of the first step is obtained without considering the links thus it may be different from the optimal solution of the combined problem. Furthermore, the second step algorithm cannot send feedbacks to the first step in case of bad inputs. In the next subsections we present in detail the two steps and we evaluate the quality of the two-step solutions compared to combined problem solutions to determine the impact of the two-parts splitting on the energy saving problem. We will not consider approximation algorithms to solve the single steps but we rely on optimal solutions given by CPLEX [64] solver. Since we focus mainly on the approximations introduced by the splitting itself we avoid the additional approximations that would be introduced by heuristics.

Router optimization

The router optimization step is the first step of the two-step approach. At this stage, routers to be involved in traffic forwarding are optimally chosen without considering

	Maximum number of back-end routers	
Interfaces	**combined**	**two-step**
1	716	more than 716
2	134	more than 716
4	30	more than 716
8	20	more than 716
16	12	more than 716

Table 3.1. Maximum size of the MILPs in terms of number of routers and interfaces to obtain a solution in reasonable time

the NICs. Thus the router optimization step is formulated as follows:

$$\begin{aligned}
\min \quad & P^{R\_OPT} = \textstyle\sum_r P_r \alpha_r && (3.8)\\
\text{s.t.} \quad & \textstyle\sum_r t_r = 1 && (3.9)\\
& t_r T \leq C_r^* \alpha_r, \quad \forall r \in B && (3.10)\\
& \alpha_r \in \{0,1\} && (3.11)\\
& t_r \in [0,1] && (3.12)
\end{aligned}$$

where all variables, constraints and terms have the same meaning as in (3.2)–(3.7) with the following modification: t_{rl} which is redefined as t_r because links are not considered here and $C_r^* = \min(C_r, C_{rl})$ which defines the actual routing capacity of a router. This problem is a variation of the well known BIN-PACKING problem with splittable item (traffic T) where the bins (routers) have different size (routing capacity C_r). Unlike the classical BIN-PACKING problem where the cost of using a bin is the same as the size of the bins, in this scheme the cost (power consumption P_r) of the routers is different from the size (routing capacity C_r). Therefore, the above problem is a BIN-PACKING PROBLEM WITH GENERALIZED COST AND VARIABLE SIZED BINS [65] with splittable items. (3.8) – (3.12) are used to optimally select the routers to be used in packing the splittable traffic T while minimizing the power consumption.

As previously mentioned, the two-step approach is based on the *divide and conquer* paradigm. Therefore the information required to globally optimize the system is partitioned among the two steps making them less optimal from a global point of view. To assess the impact of information partitioning, we introduce two different schemes to configure the first step:

- **Router-Power scheme (NIC^-)**: no link information is made available to the first step. In this scheme we use (3.8)–(3.12) with no modification.
- **Router+Link-Power scheme (NIC^+)**: NIC power consumption is considered in the first step.

In the NIC^+ scheme the cost of using a router is defined as the power consumption of the router plus the sum of the power consumption of all of its network cards. Hence the P_r parameter in (3.8) is replaced by

$$P_r^{new} = P_r + \sum_l P_{rl} \qquad (3.13)$$

P_r^{new} is the same as in (3.1) and it represents the maximum power consumption of router r including its NICs.

Link optimization

The link optimization problem makes use of the solution of the router optimization step, so it is always the second step in two-step approach. Once the routers are chosen by the first step and the amount of traffic T_r (such that $\sum_r T_r = T$) sent to router r is determined, then the links on each router r are chosen according to the following optimization problem to be solved *independently on each router* $r \in B$:

$$\begin{aligned}
\min \quad & P_r^{L\_OPT} = \textstyle\sum_l P_{rl}\beta_{rl} && (3.14)\\
\text{s.t.} \quad & \textstyle\sum_l t_{rl} = 1 && (3.15)\\
& t_{rl}T_r \leq C_{rl}\beta_{rl}, \quad \forall l \in L_r && (3.16)\\
& \beta_{rl} \in \{0,1\}, t_{rl} \in [0,1] && (3.17)
\end{aligned}$$

where T_r is the portion of the traffic T to be routed by router r, as provided by the router optimization step. t_{rl} is a portion of T_r sent on link l to router r and it is an optimization variable for this step. The other variables and parameters have the same meaning as before. Note that the link optimization is performed independently for each active router $r \in B$. As in the first step, this formulation is a GENERALIZED COST VARIABLE SIZED BIN-PACKING problem, where the links are the bins with cost P_{rl} and size C_{rl} and the traffic is the splittable item T_r.

After solving the second step on each router, the total power consumption of the multistage architecture due to the two-step approach is given by:

$$P_{twostep} = P_{R\_OPT} + \sum_r P_r^{L\_OPT} \tag{3.18}$$

3.1.3 Performance evaluation

In this section we present results obtained using CPLEX [64] to implement the optimization models and we compare the combined and two-step solutions against the scenario where no energy saving mechanism is implemented. This approach gives us more opportunity to control single parameters and to focus on algorithm behaviors without introducing additional complexity, implementation issues and approximations (e.g. two-step scheme is based on BIN PACKING problem which is typically solved by heuristics).

In the comparison, we fixed our attention to a realistic scenario, i.e. a multistage router architecture with features comparable to those of a Juniper T320 router [20]. The main parameters [11, 35, 42] used in the simulation are:

- LBs and router power consumption $P_{LB} = P_r = 80$ W;
- back-end router routing capacity $C_r = 8000$ Mbps;

- link capacity $C_{rl} = 1000$ Mbps;
- link power consumption $P_{rl} = 2$ W;

According to this profile, the 160 Gbps forwarding capability of Juniper T320 can be realized using 20 back-end routers interconnected through a switch, each back-end router being equipped with eight 1 Gbps single-port network cards. Without energy saving scheme, this network consumes about $20 * 80 + 2 * 8 * 20 + 16 * 80 = 3200$ W (1920 W by back-end routers) without considering the internal switch and assuming to have 16 LBs each with one 10 Gbps link. However, the energy-saving scheme proposed can save up to 1838 W when a minimal traffic is offered at inputs (i.e. only one back-end router and one of its link are active to guarantee minimal functionalities). This saving accounts for 57.44% of the total energy consumption of the MSSR architecture.

Besides the analysis performed with the above described multistage router configuration, we also evaluate independently the impact of the variability of four configuration parameters (i.e. C_r, P_r, C_{rl} and P_{rl}) on the solution quality (measured as the difference between the combined solution and the two-step solution). More precisely, we fix three of the parameters to the above given default values, while we vary the fourth parameter as follows:

- C_r: uniformly distributed in the range 5000 Mbps $\leq C_r \leq$ 10000 Mbps. Ten links (at 1 Gbps) for each router are required to avoid bottleneck.
- P_r: uniformly distributed in the range 60 W $\leq P_r \leq$ 120 W.
- C_{rl}: one of the standard link rates with priority to 1 Gbps links (100 Mpbs with probability $p = 0.25$, 1 Gbps with probability $p = 0.75$).
- P_{rl}: randomly chosen from {2,3,4} W.

For each of the four above described scenarios, we run the combined model and the two-step schemes for traffic loads ranging from 10% to 100% of the total routing capacity of the multistage architecture. For each of those load values, 20 random instances were generated and results were averaged over the instances. Furthermore, in the case of the two-step scheme, we evaluate both the router-optimization schemes described in subsection 3.1.2 (i.e. NIC$^-$ and NIC$^+$ schemes) under the same configurations. The comparison metric is the total power consumption of the back-end stage (minimum, maximum and average over the 20 instances). The relative difference between the optimal and the proposed algorithm is also reported to clarify the comparison metric where relative difference is defined as:

$$\text{Relative difference} = \frac{P_{twostep} - P_{combined}}{P_{combined}} \tag{3.19}$$

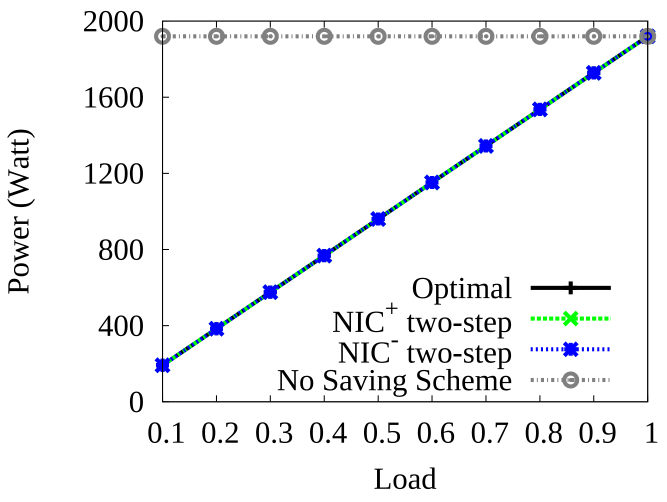


Figure 3.1. Load proportional energy saving scheme in back-end routers

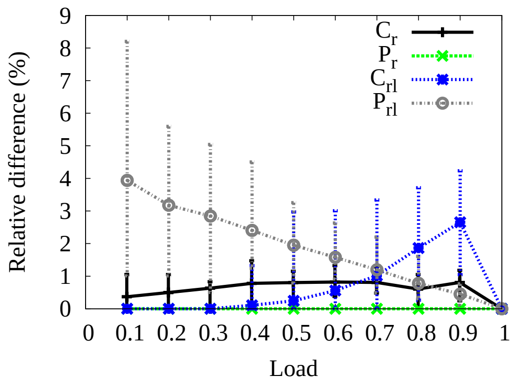
where $P_{twostep}$ and $P_{combined}$ are the two-step and optimal algorithm solution respectively.

Main results

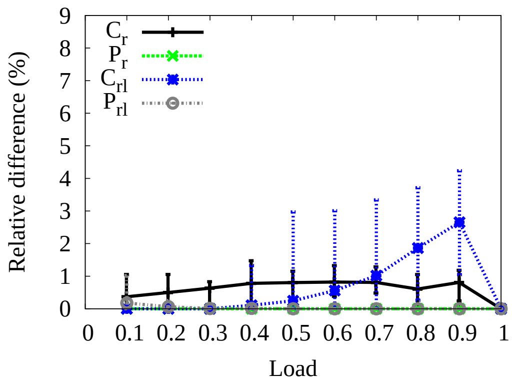
First of all, the most important result is that the usage of off-line energy saving schemes proposed makes the energy consumption of the back-end stage of the multistage router proportional to input load allowing a huge energy saving when the input load is not maximum, as reported in Fig. 3.1. For the above given experimental setup the energy saving ranges from 0 W in the case of maximum input load to 1838 W (equivalent to 95.73% of maximum power consumption of the back-end router) when input load is smaller than one link capacity and one router only is needed.

Secondly, the solutions of combined and two-step approaches are very similar in terms of power consumption. In the worst case considered in our simulations the maximum power difference between the two-step approach and the optimal solution is less than 9% as reported in Fig. 3.2, where all the considered scenarios are compared. In the next section we explain more in detail the differences among the solutions

Finally, the NIC^{-} and NIC^{+} schemes obtain the same results in most of the scenarios, but the NIC^{+} has a huge impact on the P_{rl} scenario where link power is randomized. In this scenario our heuristic allows to improve the two-step solution and to make it equivalent to the combined solution as reported in Fig. 3.2(a) and Fig.3.2(b), meanwhile it has no negative impact on the other scenarios. Thus the NIC^{+} heuristic is useful to reduce the negative impact due to the variability of link



(a) NIC- two-step heuristic



(b) NIC+ two-step heuristic

Figure 3.2. Relative difference due to variability in router and link parameters

features and to improve the greedy approach followed by the two-step schemes.

Detailed results

In this section we explain more in detail the impact of each parameter on the proposed algorithms.

C_r scenario

The effect of C_r variability on the two-step approach is minimal as reported in Fig. 3.2. The difference is due to the greedy nature of the two-step: In the first step there is no knowledge of the available links, so it happens that the amount of traffic sent to routers by the first step cannot be managed efficiently by the second optimization step on each of the routers. For instance, consider a simple scenario involving two back-end routers:

	C_r (Mbps)	P_r (W)	No. of links	C_{rl} (Gbps)	P_{rl} (W)
R1	9100	80	10	1	4
R2	7300	80	10	1	4

Table 3.2. C_r scenario description

Observe that the routers are equivalent from the point of view of the objective function of the first step (i.e. P_r) and the difference is only in the routing capacity as required by C_r scenario. Let us assume an input load $T = 11$ Gbps. A possible optimal solution is to route 9 Gbps to R1 and 2 Gbps to R2 using nine links on R1 and two links on R2 at full capacity and consuming $2 * 80 + 9 * 4 + 2 * 4 = 204$ W.

However, in the two-step approach the final solution is typically different from the optimal one. Since there is no knowledge of links in the first step, the first solution usually maximizes the usage of one of the routers. For instance, one possible choice is to forward 9100 Mbps to R1 and 1900 Mbps to R2 using ten links on R1 and two links on R2 (two links, one on R1 and another on R2, are not used at full load) consuming $2 * 80 + 10 * 4 + 2 * 4 = 208$ W, 4 W worse than the optimal power consumption.

Thus, the solution of the first step is not always well-suited to efficiently load the links, leading to higher energy consumption. However, the amount of additional energy required is generally small because only few additional links are involved (e.g. up to ten links at load equal to 0.9 in our case). Finally, there is no difference among NIC$^-$ and NIC$^+$, because the routers support the same set of links. In the case of NIC$^+$ the same amount of power is added to all the routers, thus the difference among routers remain the same.

P_r scenario

There is no difference among the two-step and the combined approaches, because the variability is introduced in the optimization parameter included in the objective function. Since the first step is the reduced version of the combined problem while considering routers only, then the optimal and two-step approaches are equivalent in this case. Indeed, there are no issues related to the inefficiency in link utilization in the second step as in the previous case, because the total available capacity C_r (8 Gbps) is exactly the sum of the available links (eight links at 1 Gbps). This means that the same routers will be chosen by both schemes, showing no difference in Fig. 3.2. Furthermore, NIC$^-$ and NIC$^+$ are equivalent for the same reasons explained in previous scenario.

C_{rl} scenario

This scenario is introducing variability in link features, thus it highlights the weaknesses of the two-step approach especially at high loads (See Fig. 3.2). In this case the links have the same power consumption, but they may have a different capacity (e.g. 100 Mbps or 1 Gbps). Since all the routers have the same power consumption, it is important to be able to use efficiently the links by sending the right amount of load to all the routers as in C_r scenario; this is done efficiently by the combined scheme, but it cannot be done by the two-step scheme where all the routers are equivalent in the router optimization step regardless of their links. Furthermore, the difference among optimal and two-step schemes increases with the load, because an increasing number of routers are activated and more routers will receive a wrong portion of the load. Finally, the NIC$^+$ heuristic cannot improve the solution since it is focused on link power consumption only, which is the same for all the links.

P_{rl} scenario

As in the previous scenario, the variability in links highlights the weaknesses of two-step scheme as shown in Fig. 3.2. In this case, all the routers are the same from the point of view of power consumption and capacity thus they are equally likely to be included in the solution by the two-step scheme. The relative difference reported in Fig. 3.2(a) is decreasing with load because at small loads it is less likely to activate the best routers (which are *randomly* chosen because they are equivalent). However, at high loads, most of the routers are activated; thus, it is less likely to exclude the best routers from the solution.

Finally, in this scenario the NIC$^+$ heuristic is very effective reducing the error in the considered configurations almost to zero because the aggregated power consumption of links and routers allows to choose more efficiently the best routers: the set of power-hungry links influences the choice of routers giving more priority in the first step to routers hosting power-efficient links. As presented in Fig. 3.2(b), there are still some small differences at low load, but this is due to the fact that the heuristic aggregates all links in a single bunch. Thus, the router optimization step is choosing on the aggregate power and not on the single link power. As a consequence, some errors are more likely when few links will be used, as in the low load case.

3.2 On-line algorithm

In the previous section we show that the two-step algorithm closely approximate the optimal solution considering different scenarios under the assumption that the input traffic is splittable (at a packet level) among the back-end routers. Given this assumption, the two-step algorithm also scales to a practical size of a MSSR architecture. However, if flow based routing is required, the two-step problem maps to a generalized cost variable sized bin packing problem [66] and its scalability deteriorates. In addition, as we describe later in this section, the off-line algorithms (both the optimal and two-step) are more service disruptive and introduce more delay unless the solution quality is compromised.

This section provides an alternative to off-line approaches; namely on-line differential energy saving algorithms. Unlike the optimal and two-step algorithms, the on-line algorithms set up a new back-end router configuration by modifying an existing configuration. It is less optimal compared to the off-line algorithms but has the advantage of being less disruptive and scales well to solve large configuration size both under splittable and flow based routing.

3.2.1 System modeling

While the assumptions presented in subsection 3.1.1 still holds, in this section we consider an on-line solution for the following two traffic scenarios:

- **unsplittable input traffic**: Flow based input traffic information is necessary if ordered delivery at the output interface is important to minimize out of order delivery and delay needed to order the flows. The ordered deliver can be performed by load balancing the incoming traffic based on flows, where a flow is defined as packets belonging to the same source and destination. Input traffic belonging to same flow will be directed by load balancers to the same back-end PCs for routing operation. This input traffic is referred to as *unsplittable input traffic* since a flow does not split into different back-end PCs but forwarded only by a single PC. QoS provision is another perspective to see the need for flow based routing.

- **splittable input traffic**: If ordered deliver and QoS provision is not mandatory, the input traffic can be present to the optimizer as an aggregate input. The LBs split the aggregate traffic among available back-end PCs for routing operation as in the case of off-line algorithms. We refer to this traffic as *splittable input traffic.*

Consequently, the multistage architecture energy saving model presented in Sec. 3.1 requires a slight modification to fit both splittable and unsplittable input traffic scenario. In the following, we discuss this modification in detail.

Problem formulation

Based on the the above stated assumptions, the multistage architecture energy saving scheme can be stated and formalized as follows: **Given** i) a set of back-end PCs B; each PC $b \in B$ characterized by power consumption (excluding network cards) $P_b \in \mathbb{R}$ and routing capacity $C_b \in \mathbb{R}$, ii) a set of links L_b connected to each PC $b \in B$; each link $l \in L_b$ characterized by power consumption $P_{bl} \in \mathbb{R}$ and link capacity $C_{bl} \in \mathbb{R}$, and iii) a set of input traffic demand $T \in \mathbb{R}$, **Select** PCs and links required in routing the traffic demand such that the energy consumption of the architecture is minimized, **Subject to** link rate and router capacity.

In formalizing the problem definition, let α_b be a PC selection binary variable (equal to 1 if a PC $b \in B$ is activated, 0 otherwise), and β_{bl} be the link selection binary variable (equal to 1 if the link $l \in L_b$ connected to PC $b \in B$ is used, 0 otherwise). And let δ_{blk} be

- a flow selection binary variable, $\delta_{blk} \in \{0,1\}$, for unsplittable input traffic; 1 if a flow $t_k \in T$ is forwarded on link l of router b , 0 otherwise, OR

- a 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 is the aggregate input traffic. Thus $k = 1$ under splittable traffic scenario and δ_{blk} represents a portion of an input traffic T.

Given the above definitions and assumptions, the MSSR energy saving problem can be formulated as a mixed integer linear programming (MILP) as follows:

$$
\begin{aligned}
\textbf{min.} \quad & P = \textstyle\sum_b (P_b \alpha_b + \sum_l P_{bl} \beta_{bl}) && (3.20)\\
\textbf{s.t.} \quad & \textstyle\sum_b \sum_l \delta_{blk} = 1 \quad \forall k \in T && (3.21)\\
& \textstyle\sum_l \sum_k \delta_{blk} S_k \le C_b \alpha_b \quad \forall b \in B, \forall k \in T && (3.22)\\
& \textstyle\sum_k \delta_{blk} S_k \le C_{bl} \beta_{bl} \quad \forall b \in B, \forall l \in L_b, \forall k \in T && (3.23)\\
& \alpha_b \ge \beta_{bl} \quad \forall b \in B, \forall l \in L_b && (3.24)\\
& \alpha_b, \beta_{bl} \in \{0,1\} \\
& \delta_{blk} \in \{0,1\} \vee [0,1]
\end{aligned}
$$

where S_k is the size of flow k. (3.21) ensures that all the input traffic T is served while (3.22) and (3.23), respectively, make sure the capacity constraints of each router (C_b) and link (C_{bl}) are not violated. (3.24) ensures that router b is active if at least one of its links is chosen to carry some traffic.

Equations (3.20) – (3.24) define a MILP problem that optimizes the MSSR architecture power consumption, considering both PCs and NICs simultaneously for both unsplittable and splittable input traffic. The solution to the above defined problem is a back-end PCs configuration capable of routing an input traffic T under devices capacity constraint while minimizing the energy consumption. The problem is categorized as NP-hard problems as demonstrated in appendix A. Thus, exact methods can only be used to solve small size cases and therefore we opt for a heuristic method. We use the results, however, as a reference to measure the performance of our proposed heuristic algorithms.

3.2.2 Proposed heuristic algorithm

Besides the optimal problem is NP-hard, reconfiguration of back-end PCs to handle changes in input traffic usually causes service disruption or forwarding delay unless an optimal solution is compromised. This is because a solution which is optimal in one input traffic scenario will not be optimal in another. Thus the change in an input traffic triggers re-computation for optimal solution under current traffic demand. Most frequently, the new solution differs from the previous solution which means turning off some already active devices and turning on some others. This transition requires some time resulting in a reduced routing capacity temporary. That is some input traffic are delayed or not served at all during reconfiguration phase. One solution could be to keep the previous configuration until a new one is setup but this compromises the optimality of the solution during the reconfiguration phase since the previous configuration is not optimal under current traffic scenario.

Algorithm description

The proposed energy saving heuristic is an on-line 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 or delay. The algorithm is a modified first-fit-decreasing bin-packing algorithm [63] with bins (PCs) having different size (actual routing capacity C_a) and different usage cost (power dissipation) where the items (input traffic) can be splittable or unsplittable. The algorithm also has to consider bins (links) within a bin (PC) scenario during packing. More specifically, the algorithm tries to pack the links with portion of the input traffic taking heed of the routing capacity of the PC being not exceeded.

During the initialization phase, the algorithm performs two tasks: computes the actual routing capacity of each PCs and sorts the devices (both PCs and links). The actual routing capacity of a PC acting as a router is limited either by the CPU packet processing capacity or by the sum of link rates on the PC [10] [11]. For example, if a PC has 4 Gbps routing capacity but has only single 1 Gbps link, then the actual routing capacity is limited to the link capacity; i.e. 1 Gbps. Since the devices have different capacity and power dissipation, sorting devices is necessary in order to increases the packing efficiency [66]. The initialization phase of the algorithm performs these tasks according to a predefined criteria (detail in section 3.2.2). Furthermore, for unsplittable input traffic the flows will be sorted in descending order - again to increase the efficiency of packing.

After the initialization phase, the algorithm continues to monitor changes in the input traffic. If the input traffic increases, the algorithm will retain the current configuration and augments the current capacity by turning on additional devices. Otherwise, if the input traffic decreases, the algorithm turns off some devices to down size the current active configuration to the traffic demand (detail in section 3.2.2).

Algorithm initialization

At start up, the algorithm analyze the MSSR available configuration, i.e. the set of PCs used as the back-end stage and of their NIC cards, to determine the actual routing capacity (C_a) of each PC and therefore of the whole architecture (C_{MSSR}).

The algorithm computes these capacities as:

$$C_a = \min(C_b, \sum_l C_{bl}) \tag{3.25}$$

$$C_{MSSR} = \sum_b C_a \tag{3.26}$$

Equation (3.25) defines actual 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. Then the sum of all back-end PCs actual routing capacity defines the MSSR architecture capacity (C_{MSSR}).

Then the algorithms sort devices, both PCs and links, according to the following two schemes:

i) device efficiency - devices are ordered in descending order of efficiency OR

ii) device power dissipation - devices are ordered in ascending power dissipation

The sorted devices are used in the packing algorithm (described in Sec. 3.2.2) that prioritize the first device in the sorted list which has a capacity to handle a traffic demand.

In the first sorting scheme, efficiency is defined, in general, as the amount of traffic routed per watt:

$$\eta = \frac{\text{actual capacity}}{\text{power}} \tag{3.27}$$

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

$$\eta_{NIC-} = \frac{C_a}{P_b} \tag{3.28}$$

Instead, the version of the algorithm that takes into account also the link power consumption during PCs sorting stage is named η_{NIC+}. In this case, (3.28) can be rewritten as:

$$\eta_{NIC+} = \frac{C_a}{P_b + \sum_l P_{bl}} \tag{3.29}$$

where N is the number of links connected to back-end PC b. Similarly, links in a PC b are also sorted according to their efficiency η_{bl}:

$$\eta_{bl} = \frac{C_{bl}}{P_{bl}} \tag{3.30}$$

The second sorting scheme sorts devices by power dissipation where less dissipating devices come first in the list. That is less energy consuming devices are preferred if they have a residual capacity to route incoming traffic. PCs sorted according to their power consumption also has $P_{_{NIC-}}$ and $P_{_{NIC+}}$ versions. Accordingly, $P_{_{NIC+}}$ sorts PCs considering links power consumption as:

$$P_{_{NIC+}} = P_b + \sum_l P_{bl} \tag{3.31}$$

On the other hand, $P_{_{NIC-}}$ considers only the PCs power consumption for sorting. Links on a PC also will be sorted according to their power consumption in case the second sorting scheme is deployed.

Packing algorithm

The goal of the algorithm is to minimize energy consumed by a subset of back-end PCs that serves the requested traffic demand. Therefore, after the devices sorting phase, the algorithm follows a greedy approach in packing the incoming traffic to the back-end PCs. For efficiency sorting scheme, the algorithm starts activating (i.e., setting in the on state) the available most efficient PC b according to (3.28) or (3.29) and the most efficient links l on that PC b according to (3.30). When the first PC's actual capacity (C_a) is fully utilized, the algorithm considers packing residual incoming traffic to the next available most efficient PC. This procedure is iterated until all the incoming traffic is served. If the incoming traffic exceeds the MSSR architecture capacity (C_{MSSr}), the extra amount of traffic is discarded. Similarly, if the sorting is based on device power consumption, the algorithm starts packing the least power consuming PCs and least consuming links on those PCs until all incoming traffic is served.

After each configuration setup, the algorithm continues to monitor the incoming traffic change to identify a traffic modification worth of a reconfiguration phase. A reconfiguration phase initialized in following two ways:

i) threshold based or

ii) sampling based

In threshold based reconfiguration initialization, a specified threshold is defined. If the traffic change exceeds this threshold, then a new back-end PCs configuration will be defined by the algorithm to handle the new traffic demand. We assumed an input traffic change by ±5% of total input traffic to initiate reconfiguration of the back-end PCs. Any smaller change in input traffic that does not reach ±5% of total traffic will not activate a reconfiguration step. To serve such a small variation in the input traffic, however, the current configuration can be augmented by 5% of total

MSSR capacity. We did not consider the capacity augmentation in these work to be fair in comparing the proposed algorithm with the optimal solution which does not consider this extra capacity either.

Sampling based reconfiguration initialization defines a traffic sampling interval a priori and the input traffic is sampled accordingly. Every time there is a new sampled input traffic, the algorithm resize the back-end PCs configuration to the load demand, i.e. the algorithm enters the reconfiguration phase on regular interval according to the input traffic sampling time.

When a reconfiguration request is triggered, if the traffic demand increases, the algorithm computes the extra traffic demand and turns on additional resources, i.e., links on already active PCs and new PCs currently in off state, needed to route the increased demand. As discussed earlier, links and PCs to be activated are considered in order of increasing efficiency or decreasing power consumption. If the traffic decreases, on the other hand, the algorithm adjusts the running configuration by turning off extra links and PCs to down size the back-end configuration to the traffic demand. Under decreasing input traffic, however, we consider turning off devices in reverse order - turn off less efficient or more energy consuming PCs and links first depending on the sorting scheme used.

The pseudo-code description of the proposed energy saving heuristic is reported in Algorithms 1 - 3. Algorithm 1 presents the sorting algorithm and Algorithms 2 and 3 details packing schemes deployed. The packing schema for unsplittable and splittable traffic is slightly different as highlighted by the corresponding pseudo code.

The block diagram depicted in Fig. 3.3 shows how the proposed energy saving scheme fits into the MSSR architecture. The energy saving scheme runs on the same back-end PC where the virtualCP is running to ease their interaction. Any change in the model instance triggers the power saving algorithm to recompute back-end configuration that saves energy under the current configuration. The model instance is modified for two reasons: First the virtualCP monitors, through the DIST protocol, any change in the MSSR configuration, to identify any modification in link and PCs configuration, that may be caused by device faults, upgrade or addition/removal for management purposes. These modifications trigger create/update actions on the system model instance. Second it is modified by the traffic statistic module that collects traffic information. Thus any change in the input traffic above a predefined threshold or new input traffic sampling or any modification in PC configuration triggers the energy saving algorithm to define a new energy efficient MSSR configuration.

Once the new MSSR configuration has been defined by the energy saving algorithm, 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.

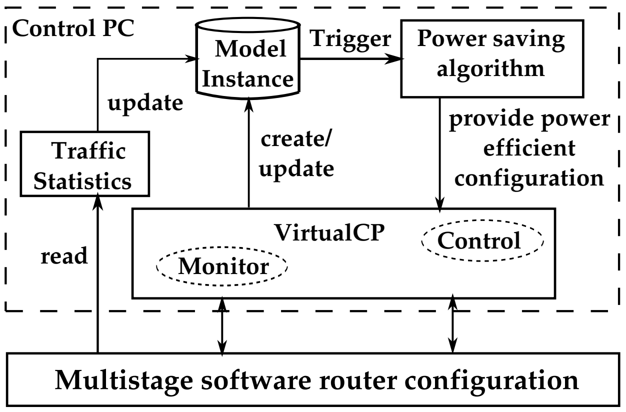


Figure 3.3. MSSR architecture power saving scheme block diagram

3.2.3 Computational complexity

Setting up a new configuration involves the following steps: (i) computing the actual capacity (C_a) of each PCs, (ii) sorting of devices and input traffic and (iii) packing the input traffic to the new configuration. Computing the actual capacity of each PCs requires looping through the PCs and links in those PCs which can be done in time $O(mn)$ where n is the number of PCs and m is the number of links on each PC. The number of links per PC usually limited to few number and can be

Algorithm 1 Sorting algorithm - Pseudo-code description

```
if sort by efficiency then
    if $NIC+$ then
        sort(list of $\eta_{NIC+}$)
    else
        sort(list of $\eta_{NIC-}$)
    end if
    for all PCs do
        sort(list of $\eta_{bl}$);
    end for
else
    if $NIC+$ then
        sort(list of $P_{NIC+}$)
    else
        sort(list of $P_{NIC-}$)
    end if
    for all PCs do
        sort(list of $P_{bl}$);
    end for
end if
```

Algorithm 2 Unsplittable traffic routing - Pseudo-code description

```
sort(list of flows)
sort(list of device)
$A$ = 1; //number of active PCs, at least one device to readily give service
loop
  if reconfiguration required then
    for All flows $f$ = 1, 2, ..., M do
      for All PCs $b$ = 1, 2, ..., N do
        for All links $l$ = 1, 2, ... L do
          if Flow $f$ fits in link l then
            Pack flow $f$ in link $l$;
            break;
          end if
        end for
        if Flow $f$ packed then
          break;
        end if
      end for
      if Flow $f$ did not fit in any $A$ PCs link then
        if $b$ is less than N then
          Turn on next PC in the list
          decrement $f$;
        else
          drop flow $f$;
        end if
      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
```

considered as a constant. Thus actual capacity computation can be performed in a time $O(n)$. Considering merge sort, sorting devices and input traffic (in case of flow based routing), can be done in worst case in a time $O(nlogn)$ [67]. We implemented a first-fit-decreasing algorithms for packing which has a worst case running time of $O(nlogn)$ [68]. Summarizing the three steps, a configuration choice can be performed in a time $O(n + nlogn + nlogn) = O(nlogn)$ which scales logarithmically in the number of flows and PCs.

Algorithm 3 Splittable traffic routing - Pseudo-code description

```
sort(list of device);
$A$ = 1; //number of active PCs, at least one device to readily give service
loop
  if reconfiguration required then
    for All PCs $b$ = 1, ..., N do
      for All links l on PC $b$ do
        if $l$ has residual capacity then
          pack portion of input traffic;
          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 N then
          Turn on next PC in the list;
        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
```

3.2.4 Performance evaluation

In this section we evaluate the proposed on-line differential algorithm with respect to the optimal solution for unsplittable and splittable input traffic. First we discuss the experimental scenario and in the following subsections we present the main results.

Experimental setup

To asses the energy saving achieved by the proposed algorithms, a back-end stage base configuration with three groups of PCs are used as an input both to the optimal and to the on-line algorithm. Each group consists of five PCs and each PC in each group has the following specification [10,11,35,42]:

	C_r **(Gbps)**	P_r **(W)**	**No. of links**	C_{rl} **(Gbps)**	P_{rl} **(W)**
Group I	4	60	4	1	4
Group II	8.7	100	1	10	20
Group III	8.7	80	9	1	6

Table 3.3. Simulation parameters for on-line algorithm performance evaluation

For the second and third group the total link capacity of each router (10000 Mbps for the second and 9000 Mbps for the third group) is more than the corresponding routing capacity which is 8700 Mbps for both groups. Therefore the actual routing capacity (computed according to (3.25)) for both groups is limited by capacity of the routers. Instead, for the first group the router capacity and the total link capacity are the same and hence the actual capacity is 4000 Mbps. Overall, the multistage software router has a maximum routing capacity of $5 * 4000 + 5 * 8700 + 5 * 8700 =$ 107 Gbps. To fully utilize this capacity we assumed 11 LBs each with one 10 Gbps link external and at least three back-end links with total capacity of 10 Gbps for internal connection [15]. Assuming 80 W power consumption for each LB, without any energy saving scheme, the architecture consumes 2.53 kW (1.65 kW by the back-end routers) excluding the interconnecting switch.

Traffic traces

In the simulation, we used two different traffic traces. The first one is based on a realistic traffic scenario. We captured a traffic from a university core router in Twente to derive the input traffic load. To build large MSSR architecture, the traffic was scaled up while keeping the ratio the same among traffic loads at each sampling. We aggregated the traces into 60min time interval. Then, we averaged the traffic volume over a week to get a per day volume statistics. Fig. 3.4 shows the volume trace. We used this trace to generate unsplittable input traffic and is suitable for sampling based reconfiguration initialization of the back-end PCs.

The second traffic trace is a synthetic traffic trace used in splittable input traffic scenario where the traffic is assumed to change by $\pm 5\%$ of the total architecture capacity (C_{MSSR}). We applied this scheme to compare the on-line algorithm with the two-step algorithm discussed in subsection 3.1.2.

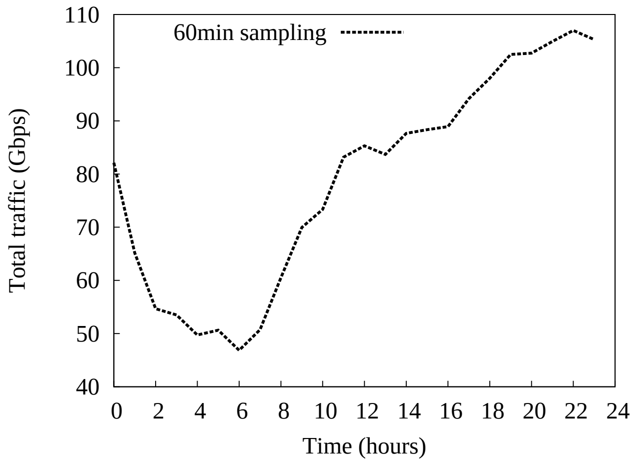


Figure 3.4. Input traffic trace used in the experiment

Evaluation metrics

Based on the experimental scenario and the input traffic traces, we compare the proposed energy saving on-line algorithm with respect to the optimal off-line algorithm and with an on-line heuristic version that is not sorting PCs in terms of energy consumption. We also compare the different algorithms in terms of the difference between two consecutive configurations.

An energy consumption of the back-end routers over a given period is obtained from power dissipation curves as:

$$\text{Energy} = \text{power} \times \text{time} \tag{3.32}$$

This is equivalent to compute the area under each power curve. Both the power dissipation and a derivate energy consumption of the back-end PCs configuration defined by each algorithms will be reported.

The configuration difference, expressed as the difference between two consecutive back-end PCs configuration in terms of number of PCs, is a measure of service interruption or delay. Turning off some devices in the current configuration during the reconfiguration step, means that the capacity of the MSSR decreases by the number of devices turned off until the new configuration come on-line resulting is service interruption temporarily.

We also consider the two router sorting policies NIC+ and NIC- to highlight the impact of link power when sorting PCs. Accordingly, results from the following algorithms will be presented for both splittable and unsplittable input traffic:

- Optimal: back-end PCs configuration solved optimally. We used CPLEX [64] to solve Equations (3.20) - (3.24)

- η_{NIC+}: an on-line heuristic that sort devices according to their efficiency and considers link consumption in PC sorting
- η_{NIC-}: same as η_{NIC+} but does not consider link consumption in PC sorting
- P_{NIC+}: an on-line heuristic that sort devices according to their power dissipation and considers link consumption in PC sorting
- P_{NIC-}: same as P_{NIC+} but does not consider link consumption in PC sorting
- Random: on-line heuristics without device sorting scheme

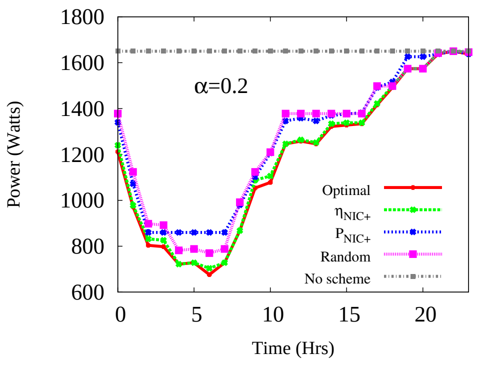
Recall that saving for LBs are not considered, as discussed in Sec. 3. Therefore, the results presented include only the back-end PCs configuration. The results reported are average over 5 runs for each scenario under unsplittable input traffic considering the amount of time it took us to solve the optimal problem. For splittable input traffic we averaged the results over 10 runs since it is much faster to solve the optimal problem under this scenario.

Results: Unsplittable input traffic

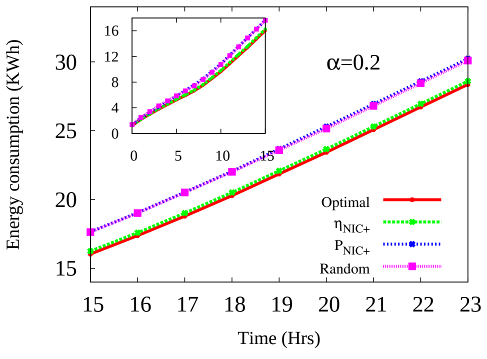
In unsplittable input traffic scenario, we consider sampling based MSSR reconfiguration initialization as discussed in Sec. 3.2.2 and Sec. 3.2.4. That is we sample the input traffic every given sampling interval and reconfigure the back-end PCs to handle the sampled traffic. This configuration lasts for the sampling interval. For this simulation we choose the realistic traffic trace depicted in Fig. 3.4 but it is equally applicable for synthetic traffic scenario. The unsplittable traffic is composed of mainly smaller and larger flows. A parameter α (and $\beta = 1 - \alpha$) defines the proportion of smaller and larger flows to be used in a given simulation. In one extreme we choose large amount of smaller flows and in the other large amount of larger flows to see the impact of flow sizes on the proposed algorithm. Thus the following α are defined:

- $\alpha = 0.2$,$\beta = 0.8$ - more number of larger flows
- $\alpha = 0.5$,$\beta = 0.5$ - same proportion of smaller and larger flows
- $\alpha = 0.8$,$\beta = 0.2$ - more number of smaller flows

At each sampling time, for each defined α, we generate flows such that the sum equal to a traffic volume (See Fig. 3.4) at that sampling time. Smaller flows are uniformly generated from 1 Mbits to 10 Mbits and larger flows from 50 Mbits to 100 Mbits. The first scenario is used to evaluate the algorithms under worst case since packing larger flows results in less efficiency. In the following we present detail



(a) Load proportional power dissipation



(b) Energy consumption over 24 hours

Figure 3.5. Comparison of different algorithms for unsplittable input traffic

results for unsplittable input traffic generated as per the defined α and β. Note that at full load it is possible all the flows might not be served even under optimal packing because of some capacity wastage in each PCs during packing. Fig. 3.5(a) reports the power dissipation of the back-end PCs configuration set up by different energy saving algorithms for $\alpha = 0.2$. The on-line algorithms (labeled η_{NIC+} and P_{NIC+}) sorts the back-end PCs considering their link consumption as discussed in Sec. 3.2.2. The figure shows that the power dissipation of the back-end PCs configurations defined by different algorithms is proportional to the load demand. However the power saving that can be achieved by the different algorithms differs. The power saving by each algorithm can be computed as the gap between the power dissipation of the architecture if no power saving scheme is deployed, the curve labeled ”No scheme”, and the power dissipation of the architecture defined by each algorithm. Fig. 3.5(b) and Fig. 3.6 confirms the claim that each algorithm saves different amount of power. Fig. 3.5(b) depicts the energy consumption of the back-end PCs configuration over 24 hours as given by (3.32). The figure is magnified over the time range 15 to 23 while the smaller inset in Fig. 3.5(b) shows the remaining part for the time range 0 to 15.

It clearly shows that the saving by the η_{NIC+} algorithm is as good as the optimal and that of P_{NIC+} is just as worse as the unsorted case. η_{NIC+} on-line algorithm is successful because in sorting the PCs it can globally see which PCs are more efficient than the others. This algorithm losses against the optimal because of the last PC choice for a given configuration. The choice of the PCs in setting up a configuration for η_{NIC+} is influenced by the sorting algorithm. That is the algorithm does not have a freedom to choose the last device in a configuration, even if it is less loaded. Therefore the capacity wastage is translated into energy inefficiency. While the global view of devices is also true for P_{NIC+} algorithm, it does so from power consumption point of view. But least power consumption does not, in general,

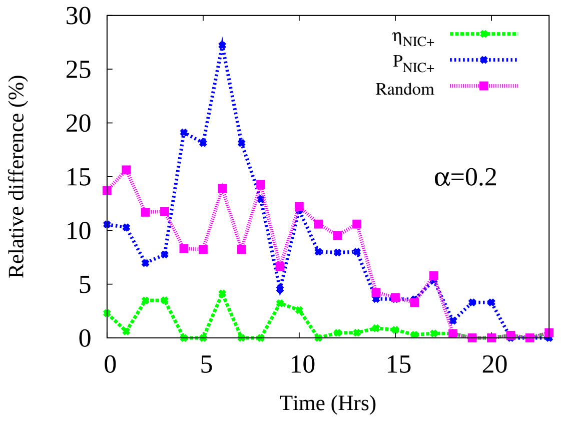


Figure 3.6. On-line algorithms relative difference with respect to optimal algorithm

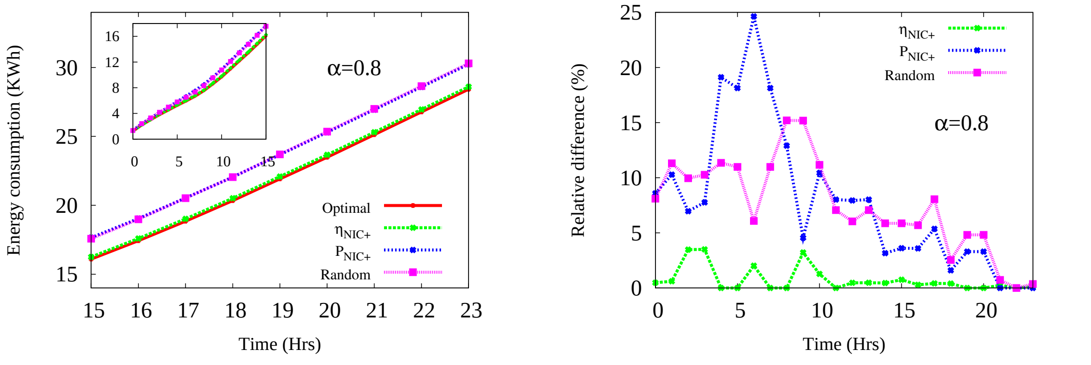
mean energy efficiency. In this simulation scenario the least consuming PCs are those in **group I** which has also the least capacity. If a traffic demand is beyond one of the PC's capacity, another one of the same group will be turned on instead of turning on one bigger capacity PC from the other groups. This results in higher loss because higher loads demand more number of less capable PCs and signifies that what is important is not only the power consumption but the amount of task a device performs per watt.

The relative difference between the on-line algorithms and the optimal is depicted in Fig. 3.6 where relative difference is defined as:

$$\text{Relative difference} = \frac{P_{ALGO} - P_{OPT}}{P_{OPT}} \tag{3.33}$$

In (3.33), P_{ALGO} is the power dissipation of the back-end PCs configuration at a given sampling time and P_{OPT} is that of the optimal configuration. Fig. 3.6 justify that the P_{NIC+} algorithm loses more against the optimal at low loads as shown in the time interval 4am and 8am. As the load is increasing the inefficiency of the on-line algorithms decreases because more and more of wrongly included devices in back-end router configuration becomes appropriate to include.

Similar behavior is seen as α changed to 0.5 and 0.8 as well. The results for $\alpha = 0.8$ is plotted in Fig. 3.7. As α changes from 0.2 to 0.8, the figures clearly show that packing efficiency slightly varies among the different αs. This is explainable because the flow sizes are very small compared to the capacity of the devices. The smaller link capacity in the simulation scenario is 1 Gbps which is 10 times that of the largest flow size. Thus packing can be done efficiently under all considered α cases. In the following we present results based on $\alpha = 0.2$ only as the results are similar for the other αs. The energy saving capability of both algorithms based on efficiency sorting and power sorting drops under NIC- scenario. Fig. 3.8(a) shows that still algorithms



(a) Energy consumption

(b) On-line algorithms relative difference with respect to optimal algorithm

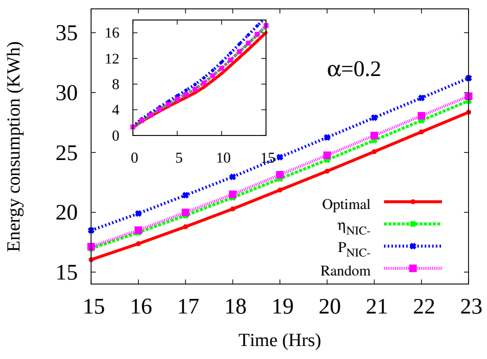
Figure 3.7. Comparison of different algorithms for unsplittable input traffic

based on efficiency sorting are performing better (see label η_{NIC-}) compared to the other on-line algorithms but it loses much against the optimal in comparison to η_{NIC+} sorting based algorithms. This is because the sorting algorithm lacks global view of devices' efficient which directly influence the packing algorithm. The inefficiency arises from the fact that those PCs which are efficient according to η_{NIC-} sorting criteria have high energy consumption in their links. Thus the overall efficiency decreases.

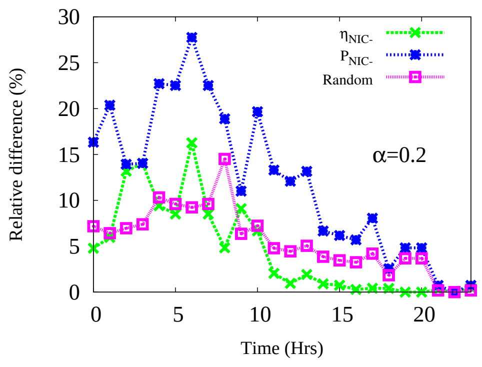
While the η_{NIC-} sorting based on-line algorithm still performs well, the lack of global view of efficient PCs costs the algorithm much more than even the unsorted scenario under some load situation. This can be seen from the relative difference plot shown in Fig. 3.8(b). This usually happens at low loads because according to this sorting scheme **group III** PCs are given priority. These devices has higher total link consumption compared to the others groups. Consequently prioritizing these devices is the source of inefficiency. At about 40% of the total load, however, the inefficiency introduced by these devices reverts as more appropriate devices are included in the solution.

The algorithm that sorts devices according to their power consumption, which was not performing well in NIC+ scenario, is even worse under NIC- (see label P_{NIC-} in Fig. 3.8). The inefficiency reasoning for P_{NIC+} also holds for P_{NIC-}. In addition, in P_{NIC-} the inefficiency is exacerbated by lack of global view of less consuming PCs.

Although the power saving achieved by the off-line optimal approach is larger, the on-line algorithms have an obvious scalability benefits. Furthermore, it gracefully modifies the current MSSR configuration minimizing service interruptions. Fig. 3.9 supports the claim that the on-line algorithms are less disruptive compared to the off-line optimal reconfiguration. The plot reports the difference in the number of PCs between two consecutive configurations for the optimal off-line and on-line



(a) Energy consumption



(b) On-line algorithms relative difference with respect to optimal algorithm

Figure 3.8. Comparison of different algorithms for unsplittable input traffic

algorithm based on efficiency sorting but the result is similar for the algorithm based on power sorting as well. The peak configuration difference is at the start of the simulation. This is because the algorithms, before receiving the first traffic input, turn on only one PC to save energy while readily giving service as traffic arrives. In our simulation, when the algorithms reconfigure the architecture for the first time, they receive a large amount of traffic (see Fig. 3.4 at 12am). Thus the number of routers change from one to more than 10 in both the optimal and on-line algorithms.

After this initial setup, however, the on-line algorithms require reconfiguration of at most 2 PCs while it is usually more than 2 PCs for the optimal. In the optimal case the change in reconfiguration includes PCs in the active configuration. This means that as the input traffic changes, the optimal off-line algorithm recomputes the back-end PCs configuration from scratch since the current configuration is not optimal under the new scenario. That is it is possible that the optimal off-line algorithm could turn off PCs in the active configuration if they are not optimal under the changed load and turns on other PCs which are already off to compensate the reduced capacity. This results in capacity deficiency during the reconfiguration period and therefore service disruption or delay is inevitable. The on-line algorithms on the other hand just turn on required number of additional dvices that augment the capacity of existing configuration to balance the increase in the load (see Algorithm 2) or turn off any extra devices if the traffic is decreasing.

Results: splittable input traffic

In this section we present performance evaluation of the proposed on-line algorithm that sorts devices by efficiency under splittable input traffic. We do not consider on-line algorithm that sorts devices based on their power dissipation since it is less

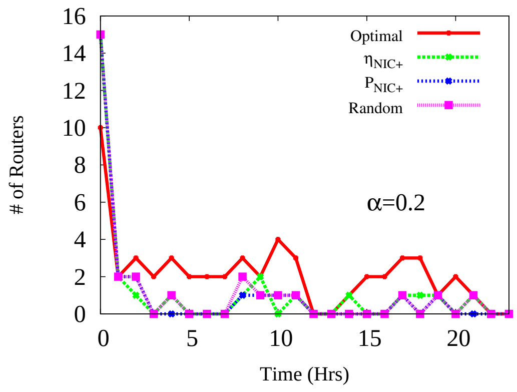
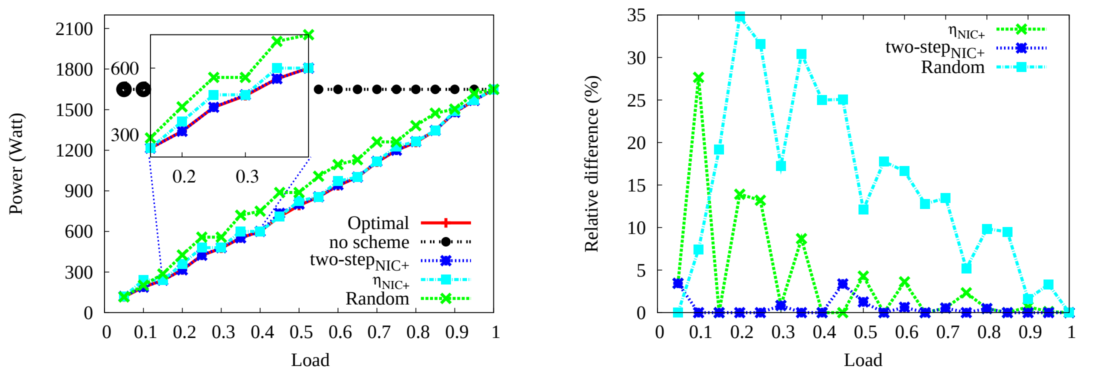


Figure 3.9. Configuration difference between two consecutive solutions (unsplittable traffic)

efficient. We compare the on-line algorithm with the optimal, two-step and an on-line algorithm that randomizes the list of back-end PCs. The experiment is based on the MSSR architecture described in Sec. 3.2.4. We used a synthetic traffic trace as detailed in Sec. 3.2.4 where a traffic change by ±5% of the total architecture capacity (C_{MSSR} = 107 Gbps) initiate the the back-end PCs reconfiguration phase. We run all algorithms for a load ranging from 5% to 100%. Note that the off-line algorithms (i.e. the optimal and two-step) are the same as the one presented in Section 3.1. We consider both router sorting policy given by equations (3.28) and (3.29) to highlight the impact of link power consideration in sorting the back-end routers.

Fig. 3.10(a) and Fig. 3.11(a) compare the saving that can be achieved by different schemes for splittable input traffic. Compared to the no scheme scenario, all the three schemes; i.e., the optimal, two-step and the on-line differential algorithms, save a lot of energy especially at low loads. In no load condition, for instance, the proposed algorithm saves up to 1.53 kW at the back-end stage under the worst scenario (i.e. only one back-end router - from the second group with its link - is needed to guarantee minimal functionality). In general, all the proposed algorithms tune the MSSR to the traffic demand resulting in a load proportional energy consumption architecture. The small figure inside Fig. 3.10(a) magnifies the detail of the three schemes in a load range between 0.15 to 0.4. To be more precise we present also the relative difference of the proposed heuristic and two-step with respect to the optimal as shown in Fig. 3.10(b) and 3.11(b).

From these results it is evident that the heuristics are less efficient compared with the optimal solution and among the heuristics the two-step shows better performance for both NIC$^+$ and NIC$^-$ efficiency sorting and the unsorted heuristic scenario is the worst. On-line heuristic is less efficient in comparison with the two-step algorithm with worst case performance gap of 27.7% and 20.2% with respect to the optimal algorithm for NIC$^+$ and NIC$^-$ router sorting respectively. Two-step has rather



(a) Load proportional power dissipation

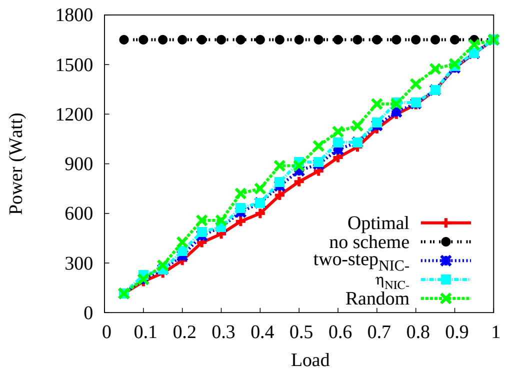
(b) On-line algorithms relative difference with respect to optimal algorithm

Figure 3.10. Comparison of different algorithms for splittable input traffic

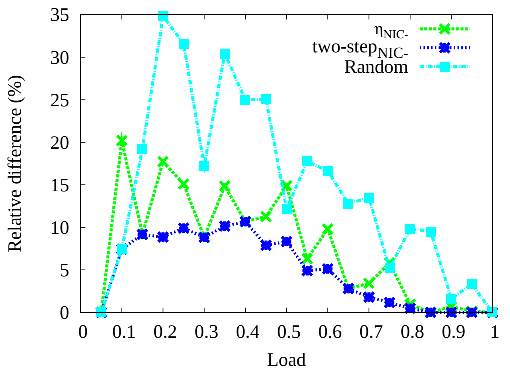
lower performance gap; i.e. 3.45% and 10.67% respectively for NIC$^+$ and NIC$^-$ router sorting respectively.

In both NIC$^+$ and NIC$^-$ back-end PCs sorting schemes the on-line heuristics worst performance gap occurs at low loads. This is because the second and the third group of PCs are given priority according to NIC$^+$ and NIC$^-$ efficiency sorting respectively. However the load demand is much less than the capacity provided by the configuration made up of these group of PCs and hence the decision made by the on-line heuristic translate into less efficiency in terms of energy consumption. For instance at load 0.1 (which is equivalent to 10.7 Gbps input traffic), we need two routers from the second group with a total capacity of 17.4 Gbps according to NIC$^+$ efficiency sorting. This is in excess of $\simeq$ 7 Gbps and the total consumption is $2*80+2*20 = 200$ W. A better choice would have been one router from the second group and one router from the first group with a total power consumption of $80+20+60+4*4 = 168$ W which is the right choice by the two-step algorithm. Thus the excess capacity by the on-line algorithm translates into energy inefficiency. This is significant at low loads relatively. As the load is increasing, the heuristic solutions are relatively comparable to the optimal and the performance gap between on-line and two-step also closes. This is because more and more devices wrongly included by the on-line heuristic in previous solution becomes the right choices thus reducing the performance gap with respect to both the two step and the optimal solution.

The better performance of unsorted heuristic scenario at low loads is because, for this simulation scenario, it happened that the routers at the front end in the unsorted list is a better combination compared to the heuristic solutions. We present here to show that this scenario is possible but running the simulation multiple times indicates that this is not generally true.



(a) Load proportional power dissipation



(b) On-line algorithms relative difference with respect to optimal algorithm

Figure 3.11. Comparison of different algorithms for splittable input traffic

It is said that the on-line differential algorithm is less efficient compared to the two-step and the optimal algorithms. Why are we then still interested in an on-line differential algorithm? The answer lies in its obvious scalability benefits and its capability to build a new solution on top of an existing one. While two-step scales well to practical MSSR size, the differential solution is lacking in both two-step and optimal algorithms because they took an off-line approach to solve the energy saving problem. That is, for every change in input traffic demand we need to solve the problem from scratch to scale the architecture to the traffic change since previous load optimal solution will not be any more optimal in the current traffic condition in off-line approaches. This could result in a completely different solution from the previous one which requires turning off an already active devices and turning on non active ones. This is undesirable since it results in service disruption until the new configuration is ready.

The proposed on-line heuristic, however, continually checks for a change in traffic demand and builds upon an existing solution; reducing possibility of service interruption while keeping the energy consumption of the new configuration as minimal as possible. More precisely, whenever there is a reconfiguration request, the heuristic start to pack the first efficient devices. At the end of packing it computes the difference between the number of devices in the previous configuration and the current one. If the difference is positive, then it turns on required additional energy efficient resources among non active configuration to serve the difference. Otherwise turns off the less efficient resources equivalent to the computed difference (see Algorithm 3).

Fig. 3.12 supports the claim that the heuristic is less disruptive compared to the optimal. It shows the configuration difference between two consecutive solution both for optimal and the on-line differential heuristic. The results is similar for two-step and the on-line differential heuristic as well. At any instance the on-line

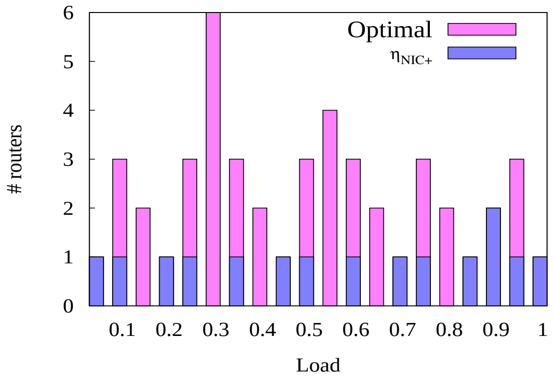


Figure 3.12. Configuration difference between two consecutive solution

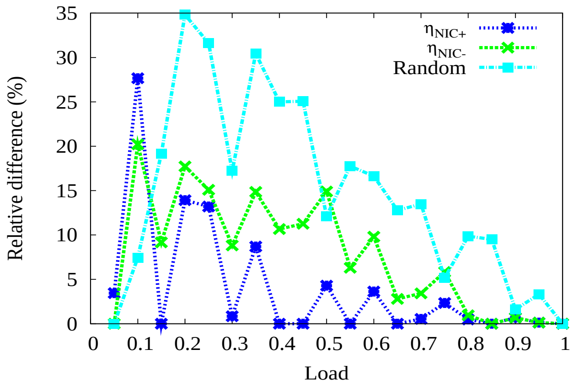


Figure 3.13. Comparison of on-line algorithms based on NIC^+ and NIC^- efficiency sorting

algorithm has less or equal configuration difference compared to the optimal solution. For instance a load transition from 0.25 to 0.3 results in turning on and/or off 6 routers in case of optimal solution but the number of routers don't change in the heuristic solution. As previously mentioned, the optimal algorithm computers for a new configuration whenever there is a reconfiguration request and usually the solution is different from the previous solution. The best it can do, for instance for increase in traffic, is to turn on additional efficient resources without turning off the previously running devices given that the previously running devices are among the optimal under the current condition. And this is how the heuristic works; i.e., it keeps the previous solution and builds up on it. Therefore the heuristic configuration difference is a lower limit for the optimal configuration difference in a general sense as shown in Fig. 3.12.

Finally, Fig. 3.13 highlights the importance of link power consideration in sorting the back-end routers. It shows that considering link power consumption in sorting routers improves the efficiency of the on-line heuristic except the unique behavior seen at very low loads. At this load, among the randomized routers list, routers with better efficiency have been at the front of the unsorted list favoring unsorted scenario but this is not true in general.

3.3 Conclusions

In this chapter, we presented energy saving algorithms that minimize the energy consumption of a multistage software router by resizing the architecture to the traffic demand through proper choose of subset of the back-end routers. We defined an optimal problem and proposed a *two-step* heuristic to overcome the scalability issue related to the optimal algorithm. The simulation results obtained on a realistic scenario show that the solution quality of the proposed scheme is within 9% of the optimal solution in the worst case considered.

We also presented an energy saving on-line differential algorithm to solve the same problem. We evaluated the algorithm in relation to the optimal solution and compared its performance with respect to the two-step approach as well. The results shows that, even though the on-line differential heuristic is less optimal compared to the off-line approaches (both the optimal and two-step algorithms), the saving is still significant in comparison to the no saving scheme. The main advantage of on-line heuristic lies in its capability to build a new solution upon existing one. This makes the algorithm non disruptive and therefore superior to the off-line approaches in terms of QoS provision. Moreover, it scales well to solve large size MSSR configuration.

Finally, all algorithms make MSSR architecture to be energy efficient. That is the architecture consumes energy proportional to the input traffic load demand. The saving achieved by the proposed algorithms reaches up to 57.44% of the total architecture energy consumption at low loads.

Though the energy saving schemes are defined for multistage software router, the schemes could easily be adapted to other distributed architectures composed of different parts (e.g. a proprietary router with multiple line cards) where energy efficiency of the parts need to be addressed independently.

Chapter 4

Energy efficient multistage architecture design

In chapter 3 we proposed energy saving algorithms that adapt a MSSR architecture capacity to a given known traffic demand by tuning the back-end stage. The off-line algorithms optimize the MSSR architecture at a given time instant for a known traffic demand. If running this algorithm at different times for a variable traffic profile, the MSSR architecture configurations obtained may be composed by a different set of PCs. In the on-line case a differential approach is sought for: a new configuration is obtained by updating the configuration defined at the previous time by minimizing the number of PCs that should be switched on/off. In both approaches, no attempt is made to globally optimize the MSSR architecture taking into account the long-term (e.g., 24 hours) traffic profile.

It is also shown that the proposed algorithms can save up to 57.44% of energy when compared to architectures that does not implement energy saving scheme [21]. However, the achievable savings depend on the back-end routers configuration. For instance, if the back-end stage is built using PCs having coarse capacity granularity, it is difficult to resize the configuration in period of low traffic load, and the installed unused capacity translates into energy wastage. On the other hand, a back-end stage composed of PCs with smaller capacity granularity is more flexible for reconfiguration. But, the smaller capacity means a larger number of PCs to handle a given traffic demand, which requires more energy.

The goal of this chapter is, therefore, to define the back-end routers configuration that minimizes the energy consumption over a given period, under the assumption that once the MSSR configuration has been optimally defined the energy saving algorithms similar to those presented in Chapter 3 are used to adapt the back-end stage configuration to the input traffic demand.

The remainder of this chapter is organized as follows. The proposed energy efficient back-end routers design approaches are detailed in Sec. 4.1. Sec. 4.2 presents the energy consumption and back-end routers cost comparison of proposed design approaches with the optimal and existing design methods. Finally we conclude this chapter in Sec. 4.3.

4.1 Energy efficient back-end routers design

Capacity to handle traffic is the basic requirement when designing a high performance MSSR to satisfy the peak load demand. However, the approach of sizing back-end routers on the peak demand does not translate into energy efficient configuration. Given the increasing importance of energy saving in networks, back-end router design should consider energy consumption in addition to peak load capacity requirement. To achieve this goal, we propose three different back-end routers design approaches: a goal programming based methodology, a heuristic and a locally optimal approach, described in detail in the subsections 4.1.1, 4.1.2, and 4.1.3 respectively.

All design approaches assumes the following input parameters:

- splittable input traffic $T_t \in \mathbb{R}$: an average traffic profile, derived by estimates or measures, and sampled every time t;

- set of available PCs to be used as back-end routers. Let S be a set of groups of PCs of different types. Each PC in the same group is characterized by the same power consumption $P_k \in \mathbb{R}$, routing capacity $C_k \in \mathbb{R}$, and hardware cost $W_k \in \mathbb{R}$. In the design phase, in each group $k \in S$, PCs are assumed to be available in infinite number.

4.1.1 Goal programming design approach

The goal programming design approach models the energy efficient MSSR design as a preemptive goal programming problem. The model has two objectives: energy minimization and cost minimization. The primary objective is to minimize the energy consumption of the back-end routers over the traffic sampling duration. This defines the the number of PCs, N_k, from each group k to be used as a back-end router in the MSSR architecture. The problem is constrained by each router capacity and a maximum admissible budget I. The budget constraint is used to keep the MSSR cost under control. The problem is formulated as follows:

$$\begin{aligned}
&\textbf{minimize} \\
&\qquad O(t,k) = \textstyle\sum_t \sum_k P_k N_k \alpha_t && (4.1)\\
&\textbf{subject to} \\
&\qquad \textstyle\sum_k C_k N_k \alpha_t \geq T_t \quad \forall t && (4.2)\\
&\qquad \textstyle\sum_k W_k N_k \leq I \quad \forall k \in S && (4.3)\\
&\qquad 0 \leq N_k \alpha_t \leq N_k \quad \forall k \in S, \forall t, && (4.4)\\
&\qquad N_k, N_k \alpha_t \in \mathbb{N} \\
&\qquad \alpha_t \in [0,1]
\end{aligned}$$

The solution to the optimization problem (4.1) – (4.4) is the number of PCs N_k from each group k to be used to build the back-end routers configuration. (4.2) adapts N_k by $\alpha_t \in [0,1]$ defining the suitable N_k composition that satisfies T_t at each sampling instance t. It takes into account the energy saving algorithms running after the initial design phase to adapt the designed configuration to the input traffic demand T_t. (4.3) ensures that the hardware cost of the selected PCs should not exceed the maximum cost I and (4.4) bounds the number of PCs needed at each sampling time t within N_k.

The objective function, $O(t,k)$, minimizes the sum of each sampling instance active configuration power dissipation. This is equivalent to minimizing the energy consumption of MSSR architecture over a specified period. Note that the active configuration at time instance t consists of only $N_k \alpha_t$ PCs from each group k. Hence, the power dissipation of a MSSR configuration varies for each sampling time t.

If the maximum admissible budget I, which is difficult to set up a priori, is too large with respect to traffic needs, the first design phase could result in an over-sized configuration because the optimization target is the energy consumption. This situation may incur unnecessary cost to set up the back-end router configuration. Thus, we define a second step that optimize the cost of the back-end routers by tailoring any possibly over-sized configuration obtained in the first step. To maintain the primary objective, we enforce the energy cost obtained from the energy optimization problem as an equality constraint and build the budget optimization model as follows:

minimize

$$O(k) = \textstyle\sum_k W_k N_k \tag{4.5}$$

subject to

$$\textstyle\sum_k C_k N_k \alpha_t \geq T_t \quad \forall t \tag{4.6}$$

$$0 \leq N_k \alpha_t \leq N_k \quad \forall t, \forall k \in S \tag{4.7}$$

$$\textstyle\sum_t \sum_k P_k N_k \alpha_t = O(t,k) \tag{4.8}$$

$$N_k, N_k \alpha_t \in \mathbb{N}$$

$$\alpha_t \in [0,1]$$

Constraints (4.6) and (4.7) have the same meaning as in (4.2) and (4.4) respectively. (4.8) guarantees that the new back-end routers configuration dissipates the same power as in the previous phase. As pointed out earlier the primary objective represents the sum of each sampling instance active devices power dissipation. Thus the cost minimization step, except trimming any excess devices from the configuration, should not affect the primary objective. This is achieved through minimization of the total cost while maintaining the primary objective and satisfying the traffic demand at each time t which is a problem defined through (4.5) – (4.8).

In summary, equations (4.1) – (4.8) define a preemptive goal programming model that minimizes the aggregate energy consumption of back-end routers over a specified period with right cluster cost.

4.1.2 Heuristic design approach

The heuristic approach defines the back-end routers configuration by greedily choosing the most efficient PCs until the traffic requirement is satisfied. The PCs efficiency is defined as performance per unit watt.

$$\eta_k = \frac{C_k}{P_k} \tag{4.9}$$

Since we assume infinite PCs for each group, only the most efficient group k is used to build the back-end routers cluster. Thus, the number of PCs, N_k, required to handle a peak load T_{max} is simply computed as:

$$N_k = \left\lceil \frac{T_{max}}{C_k} \right\rceil \tag{4.10}$$

where

$$T_{max} = \max(T_t) \tag{4.11}$$

As discussed in Chapter 2, similar cluster design approaches exist in the literature. The Google cluster architecture [49] considers performance per unit of price as PCs selection criteria to setup a cluster configuration. This approach gives priority to commodity-class PCs to high-end multi-processor servers because of their cost advantage. The other and most common way of resizing a cluster capacity is to use servers with best absolute performance such that the cluster handles peak load with fewer reliable computers [51, 52]. In both cases the clusters have not been analyzed for power consumption. In the design validation section we will compare the energy cost of the clusters designed by these design schemes with those of our proposed design techniques.

4.1.3 Locally optimal design approach

This optimal design approach defines the optimal back-end routers configuration considering only the peak load. The problem is therefore defined as: given a peak load T_{max} as define in (4.11) and a set of group of PCs, S, where each PC in group $k \in \mathbb{R}$ is characterized by the same power consumption $P_k \in \mathbb{R}$ and routing capacity $C_k \in \mathbb{R}$, select a subset of S within a budget constraint that minimize the power consumption of the back-end router configuration and satisfy the peak load demand. It is formulated as follows:

minimize

$$\textstyle\sum_k P_k N_k \tag{4.12}$$

subject to

$$\textstyle\sum_k C_k N_k \geq T_{max} \tag{4.13}$$

$$\textstyle\sum_k W_k N_k \leq I \quad \forall k \in S \tag{4.14}$$

$$N_k \in \mathbb{N}$$

(4.12) – (4.14) is a simplified form of (4.1) – (4.4) with a single sampling time, i.e. the peak load sampling time. Thus the decision variable α_t has no meaning and T_t is replaced by T_{max}. Once the back-end router configuration is set up, i.e. N_k is known from each group k, then this configuration is used as a basis for the other sampling time t adjusted (by energy saving algorithms deployed after the design stage) to the traffic demand T_t at each sampling instance. Of course this design is not globally optimal since an optimal configuration in one load condition will not be optimal (even after adjustment by energy saving algorithms) under another load.

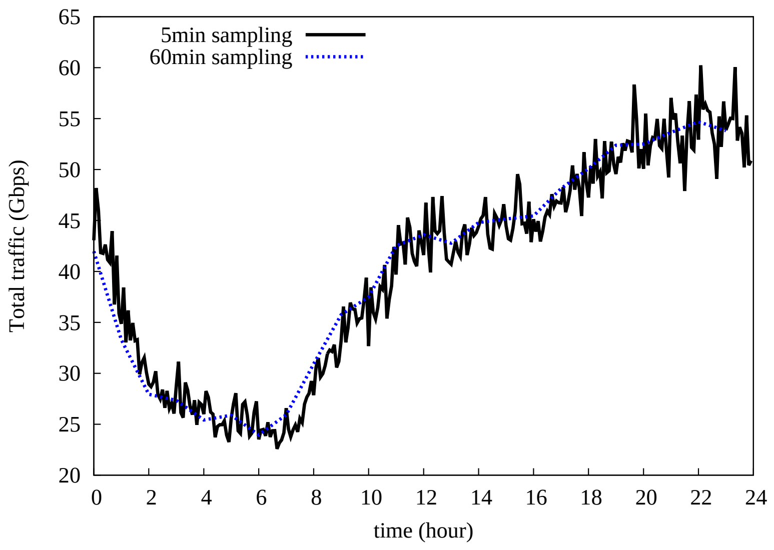


Figure 4.1. Input traffic trace used in the experiment

4.2 Design validation

In this section we discuss the performance of the proposed approaches through experimental results. First we discuss inputs to the design problem: the traffic traces and the experimental scenario. Then, we present the main results and discuss the results in detail in the following subsections.

4.2.1 Traffic traces

Instead of defining a synthetic traffic load with a typical day and night behavior, we captured traffic from Twente university core router to derive the input traffic load. To build large MSSR, traffic was scaled up while keeping the ratio among traffic loads at different sampling time. We used Perl scripts to aggregate the traces into 5min, 15min, 30min and 60min time interval. Then, we averaged the traffic volume over a week to get a per day volume statistics. Fig. 4.1 shows the 5min and 60min volume traces. We did not include the 15 and 30min curves for the sake of clarity, but they show a similar behavior.

4.2.2 Experimental setup

The following four groups of PCs are used as input to the model [10, 11, 35]:
An infinite number of PCs in each group is assumed in the design phase. Based on this setup, we compare the different design approaches in terms of energy consumption and cost of back-end routers configuration.

- *Design-I*: builds a back-end routers cluster by choosing PCs with highest performance/price ratio [49]. As discussed in Chapter 2, this approach minimize the cost of the cluster which we verify through our simulation. It achieves this goal by giving high priority to commodity PCs to set up the cluster because of their cost advantage.

- *Design-II*: builds a back-end routers cluster by choosing PCs with the highest performance (i.e, routing capacity) [51, 52]. This approach is the most widely used cluster sizing technique. In this case, the design approach prioritize high-end PCs to take advantage of the absolute performance they provide and to ensure hardware reliability.

- *Design-III*: builds a back-end routers cluster by choosing PCs with the highest performance/power ratio. This is the heuristic cluster design scheme proposed in this chapter. It orders the group of PCs according to their efficiency (performance/power ratio) and selects the most efficient group to design the cluster.

- *Design-IV*: This approach is another design scheme proposed. It designs the back-end routers cluster by locally optimizing the configuration at the peak load. Then this configuration output will be used as a basis for other traffic input sampled at a different time.

- *Design-V,I_x*: This is our third back-end routers cluster design scheme. It solves the optimization problem defined in (4.1) – (4.8) for different budget constraints I_x where x = 1,2,...n and $I_1 < I_2 < ... < I_n$ to set up the cluster.

In comparing these design approaches, we derive the energy consumption over a period of 24 hours. The optimal solution which is obtained by solving the following

	Group Parameters		
Group	C_k **(Mbps)**	P_k **(W)**	W_k **(Units)**
I	6500	75	250
II	8700	95	400
III	10000	120	500
IV	8500	92	400

Table 4.1. Group of PCs and corresponding parameters used in the back-end router cluster design

Integer Linear programming (ILP) model will be used as a reference for comparison:

minimize

$$O_{optimal} = \sum_k P_k N_k \tag{4.15}$$

subject to

$$\sum_k C_k N_k \geq T_t \tag{4.16}$$

$$N_k \in \mathbb{N}$$

Note that we do not have any additional constraint in this model, except to guarantee the capacity for the traffic load and solved for each sampling time t independently.

We also assume that algorithms similar to those presented in Chapter 3 will be used to turn on/off PCs at each traffic sampling time to adapt a configuration setup by the different cluster design schemes to the traffic demand. To avoid any approximations, however, we rely on CPLEX [64] to collect the results.

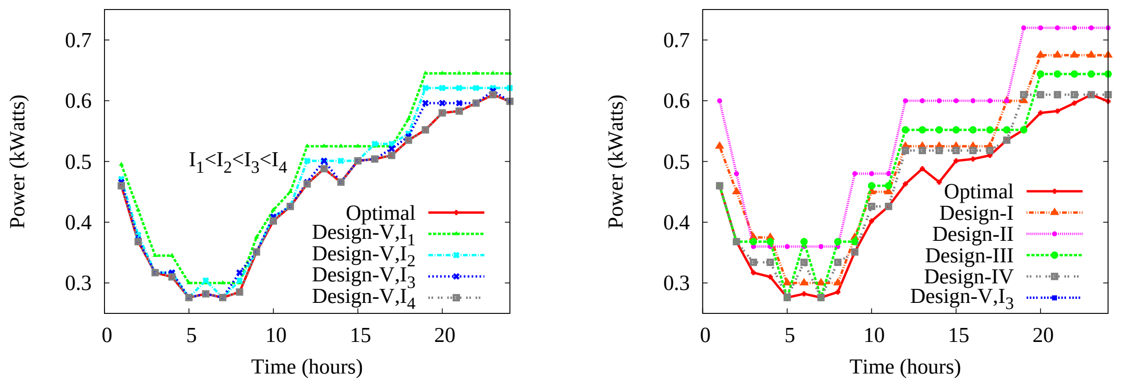
4.2.3 Results

In this subsection we compare the back-end routers selected by the different design approaches from the energy consumption and cost perspective. We also discuss the impact of traffic sampling interval on each design approaches.

Energy consumption

Fig. 4.2 compares the power dissipation of the different configurations. The result is based on the 60 minutes input traffic sampling. The figure is split into two figures for clarity. The optimal solution (labeled as "Optimal") obtained by solving the ILP model (4.15) – (4.16) is included in both graphs as a reference. It is the lower limit to the energy consumption of back-end routers defined by the other design approaches. One can also observe from both figures that all the design approaches result in an energy proportional cluster configuration except that the efficiency of the energy saving algorithms differ under different design approaches.

Fig. 4.2(a) presents power dissipation of a cluster designed by goal programming approach for different initial investment I_x. It is easy to see that the most constrained design (*Design-V,*I_1) is the least energy efficient. This is because, constrained by budget, the optimization problem has no option but to choose PCs with least cost though they are not energy efficient. However, as the budget is relaxed, the back-end routers energy efficiency improves where the least constrained design (*Design-V,*I_4) is the most efficient. *Design-V,*I_4 has the same performance as the optimal solution because extreme relaxation of the initial investment is the same as no constraint at all which is the case of the optimal solution.



(a) Power dissipation: Design-V for different budget constraints, I_x

(b) Power dissipation: Design-I, Design-II, Design-III, Design-IV and Design-V,I_3

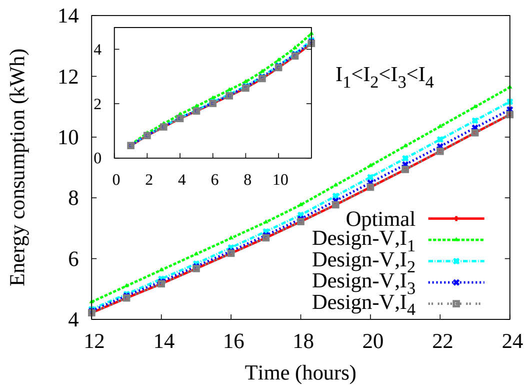
Figure 4.2. Power dissipation of back-end PCs defined by different design approaches (based on 60min traffic sampling)

Fig. 4.2(b) compares the power dissipation of *Design-I*, *Design-II*, *Design-III*, *Design-IV* and *Design-V*. Only of the goal programming approach, i.e *Design-V,I_3*, is reported in this plot for comparison with the other design approaches. The reason of choosing *Design-V,I_3* will be clear soon while we discuss about the cost of the back-end router cluster cost in Subsection 4.2.3. But this result shows that *Design-V,I_3* is performing better than the others and *Design-II* is the worst cluster design approach from energy cost perspective. To distinguish among the other design approaches' performance, we report the energy consumption of the back-end routers computed using (3.32) which is equivalent to computing the area under each power curve. The result is shown for a 24 hrs traffic pattern in Fig. 4.3(a) and Fig. 4.3(b).

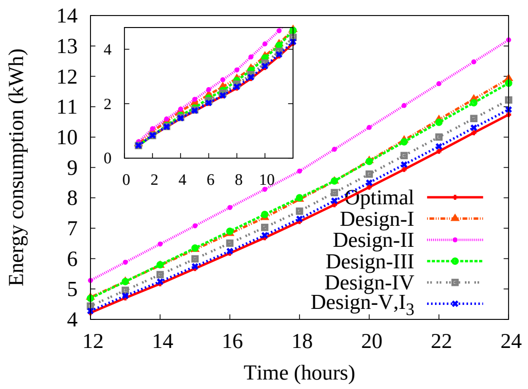
Fig. 4.3(b) confirms that *Design-II* consumes the highest energy in the specified period. This design approach gives priority to high performance devices to setup the back-end routers cluster. Since the high-end servers have large capacity granularity, resizing the configuration to the input traffic mostly ends up in wasting some capacity. Thus, less loaded PCs in the back-end are sources of energy inefficiency and the energy required by this approach is roughly 2kWh larger than the one required by the optimal algorithm per day.

The design principle that configures back-end routers based on performance/price ratio, *Design-I*, has better tailoring properties as it give priority to mid-range PCs that have relatively smaller capacity and consume less energy, making it easier to match input traffic. This yields roughly 1kWh (per day) less energy consumption than *Design-II*, although it is fairly far from the optimal.

Design-IV is comparable to the goal programming approach, being roughly 0.25kWh less efficient than the optimal solution over 24 hours. Since this design



(a) Energy consumption: Design-V



(b) Energy consumption: Design-I, Design-II, Design-III, Design-IV and Design-V,I_3

Figure 4.3. Energy consumption of back-end PCs defined by different design approaches (based on 60min traffic sampling)

approach is based on optimization problem, the cluster is composed of heterogeneous PCs from different groups giving more flexibility to resize the back-end PCs to match the load demand. The design scheme is only locally optimal to the peak load, and hence less efficient compared to the goal programming (Design-V,I_3) or the optimal approach.

Design-III, on the other hand, performs worse than the two other proposed design schemes. In this design approach, even though best efficient group is chosen to set up the cluster, the cluster is composed of homogeneous PCs making it less efficient to tune the cluster to the input traffic load.

The curves labeled *Design-III*, *Design-IV*, and *Design-V,I_x* are the energy consumption of back-end routers cluster design approaches proposed in this thesis. Goal programming technique outperforms the other design approaches if not under very tight budget constraint (see label *Design-V,I_1* in Fig. 4.3(a)) in which case the energy consumption is similar to *Design-III* and *Design-I* (See Fig. 4.3(b)). However, with less strict budget requirements, goal programming performance is very close to the optimal solution (see label *Design-V,I_3* and *Design-V,I_4*), saving up to 1-2 kWh every day if compared to other design approaches. The goal programming approach is more efficient because it selects PCs from heterogeneous groups giving more flexibility to resize the back-end PCs to match input traffic.

The proposed design approaches save roughly 10% of energy when compared to existing design techniques. This figure even rises to 20% for the goal programming design approach if the budget constraint is further relaxed. Therefore these design approaches supplement the energy saving algorithms proposed in Chapter 3 to achieve saving higher than the reported 57.44% saving.

Back-end router cluster cost

Since cost is always a key consideration, it is important to compare the cost of back-end routers defined by different design approaches, as shown in Fig. 4.4. As expected, *Design-I* has the least price because it is based on the performance/price ratio selection criteria which tries to minimize price while increasing performance. *Design-V,I_1* has similar costs mainly because of the tight initial investment constraint. It is also interesting to note that the two approaches have similar energy efficiency, as shown in Fig. 4.3(b). However, the cost of *Design-V,I_x* increases as the problem is less and less constrained by initial investment while the energy consumption decreases. The goal programming solution comparable to the optimal solution, i.e. *Design-V,I_4*, costs $\simeq$1.5-2 times larger than any of the other approaches. From all possible *Design-V* constrained by different initial investment I_x, *Design-V,I_3* has similar costs with the other design approaches but it achieves near optimal energy efficiency. That is why we considered *Design-V,I_3* to compare its energy consumption with the other design approaches in the previous results. *Design-V,I_3* is preferable for the reason that we are comparing design approaches with similar cost and in addition we considered a design output which has energy efficiency as close as the optimal while minimizing the cost. But, in general, there is a trade-off between energy efficiency and cost in goal programming back-end routers design approach which can be controlled by selecting a proper budget constraint.

Except the best goal programming approach, all other design approaches have similar cost. At least with the considered set of PCs, which are those available on the market today, energy efficiency seems to be one of the most important metric in designing back-end routers cluster.

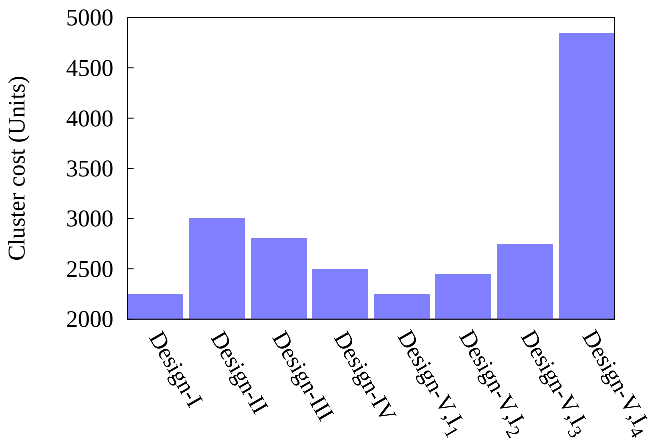
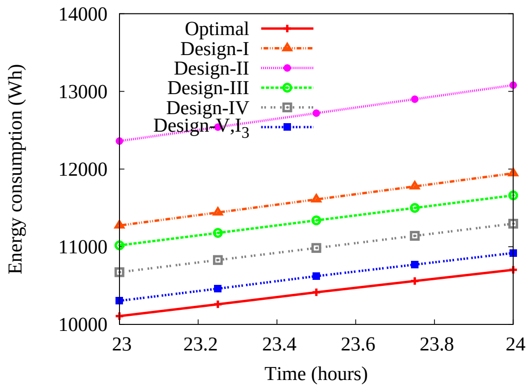
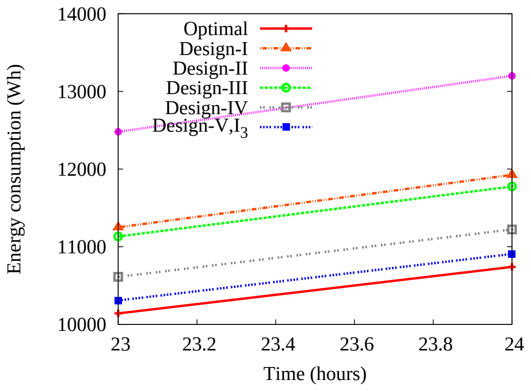


Figure 4.4. Cluster cost for different design approaches



(a) 15min traffic sampling



(b) 60min traffic sampling

Figure 4.5. Energy consumption: Design-I, Design-II, Design-III, Design-IV, and Design-V,I_3 (based on 15min and 60min traffic sampling)

Sampling interval impact on design appraoches

We also analyzed the impact of traffic sampling intervals on energy efficiency. We analyzed the energy consumption of the back-end PCs using traffic traces sampled every 5min, 15min, 30min and 60min. Fig. 4.5(a) and Fig. 4.5(b) present energy consumption of back-end router cluster defined by the different design approaches under 15min and 60min traffic sampling respectively. Comparing Fig. 4.5(a) and Fig. 4.5(b) reveals that energy consumption depends only slightly on traffic sampling interval. For instance, if we consider goal programming design approach, the difference in energy consumption is only 44 Wh over 24 hour period if the cluster is designed based on 15min and 60min. We present here the 15min and 60min sampling but the trend is similar for 5min and 30min sampling as well.

However, if the back-end routers cluster is analyzed under a sampling interval for which it was not designed, the results are different as shown in Fig. 4.6. We considered a cluster designed based on 60min sampling, later analyzed for its energy consumption under the same traffic trace but for a 5min sampling interval. The curve labeled "Design-X/Sampling-Y" represents a back-end PCs designed under "X min" traffic sampling but analyzed for energy consumption under "Y min" sampling. The figure shows the difference in energy consumption for a MSSR designed for 60min and analyzed for 60min and 5min traffic sampling time. The mismatch accounts for 0.8 kWh per day. This implies that algorithms responsible to resize the back-end routers configuration after initial deployment should stick to the sampling time used during the design phase to reduce energy inefficiency.

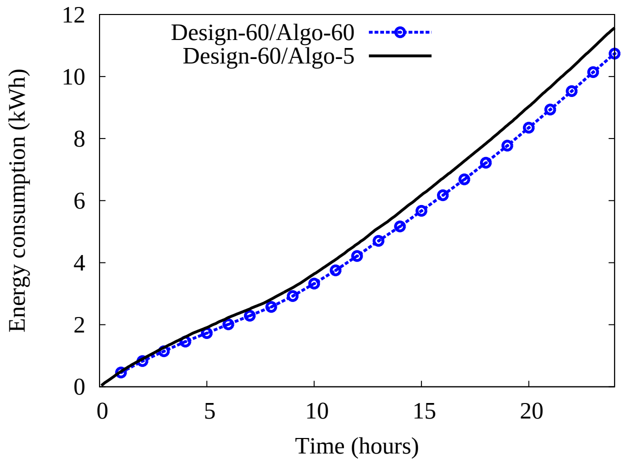


Figure 4.6. Design and sampling mismatch effect

4.3 Conclusions

We proposed three different energy efficient back-end routers design approaches that make energy saving algorithms more efficient. They permit to save up to 10% of energy with respect to existing design approaches for similar budget. Savings could raise to 20% for the goal programming design approach if the budget constraint is further relaxed.

The traffic sampling interval is not so fundamental in designing energy efficient back-end routers. But, if back-end PCs are designed for a given sampling time, algorithms responsible to resize the configuration for variable traffic load should stick to the sampling time used during the design stage for better efficiency.

Although we considered the specific design scenario of a MSSR architecture, the proposed design approaches can be used in any cluster design that involve PCs as the basic computational resources.

Chapter 5

Multistage architecture network management

In Chapter 1 we discussed that the MSSR advantages come at the cost of control and management complexity. This complexity problem has partially been addressed from the control plane perspective [19]: An internal control protocol named DIST has been developed to: i) configure the architecture; ii) coordinate the routing process among back-end routers and the load balancing function among LBs; and iii) provide automatic fault recovery mechanisms. Furthermore, DIST dynamically maps the virtualCP on one of the back-end routers.

In Chapter 3 and 4 we discussed MSSR energy management in detail. In this chapter we tackle the information management issue in MSSR architecture. After briefly introducing management information organization, we discuss MSSR architecture management complexity problems (Section 5.1) and set management requirements to build a management system suitable for MSSR (Sec. 5.2). Then in Sections 5.3.1, 5.3.2 and 5.3.3 we describe the solution to the MSSR internal network management problem detailing required communication model modification and methods on how to create a unified information management system that represents the MSSR architecture. Scalability analysis of proposed solution is detailed in Section 5.3.4. Finally in Section 5.3.5 we briefly discuss the proposed solution implementation and conclude the chapter in Section 5.4.

5.1 Problem description

Networks require management procedures to monitor and control their operation. To reach this goal, network devices keep track of management information, such as cross-traffic, interface status and general system statistics. This information is organized into a management information base (MIB) [53] accessible through different management protocols such as SNMP, NetConf and NetFlow [56, 69, 70].

The MIB is organized in a tree structure for individual network devices: objects, ranging from scalar objects such as the system up time to tabular objects, such as the routing table, are represented as tree leaves. A MIB object has the following specifications: a unique name, attributes and valid operations that can be performed on the objects (read and write). A numeric tag, called object identifier (OID), is used to navigate through the tree to uniquely identify each MIB object.

Individual devices in a network has its own MIB and hence MIB variables are directly accessible. The MSSR architecture, however, has a unique characteristics. It is different from local area network in that the internal network has to be hidden from external network mimicking a single device. It is different as well from a single device because internally it is composed of devices having independent computing capability. From management perspective this means that the MIB information is distributed among different internal elements and none of them represents the multistage architecture. For example consider the sysUpTime[1]. Each internal component in MSSR has its own sysUpTime but which one represents the up time of the multistage architecture as a whole? Or do we need to generate a new sysUpTime? Analyzing such management related questions to create a unique MIB for multistage software router requires additional complexity.

In this chapter we discuss building a multistage software router MIB from the internally distributed management information among internal components. Therefore the problem of MSSR information management can be seen as the problem of mapping and/or combining the distributed information into an aggregated view. This problem is twofold: i) definition of a communication model to collect the distributed data and ii) mapping the various data to create a single-entity view. We define an internal network manager that coordinate the internal elements and operate on the MIB information distributed among the internal elements to create such a single-entity view.

In addition the internal network manager has to work in coordination with other modules, such as the multistage energy saving module described in Chapter 3, that affects management operation. Turned off devices by the energy saving algorithms are not available to respond to a management request. Consequently a multistage management system need to track such unaccessible information for consistent management.

Access to management information is further complicated if any of the internal device is not management capable. For instance LBs implemented in FPGA [26] are not management capable because these devices do not support management agents

[1]sysUpTime is the time (in hundredths of a second) since the network management portion of the MSSR architecture was last re-initialized [53].

which are responsible to respond to management requests. Therefore it is not possible to request for management information at all if the devices are not management capable. Whenever it is possible to request for management information, these devices track only limited number of the vast majority of management information. The obvious question to answer is that how does the MSSR build its management information from internal devices under this condition?

Addressing these and similar challenges require either modification of existing functional areas of network management [71, 72] or definition of new design requirements and new approach to manage MSSR architectures. In this thesis we took the later road and present our proposal to solve the above described management complication.

5.2 Multistage architecture internal network management requirements

In this section we discuss multistage network management design requirements which guides us in proposing an MSSR management solution and evaluating its feasibility. We identified the following four management requirements and a fifth one is included as a future work.

1) Unified management information: MSSR is build on the principle that the whole architecture will be viewed as a single device to external network. This principle applies for the management information as well. Hence the multistage architecture should have a single MIB, which could be virtual or accessible locally to the internal manager, representing the whole architecture. Management information unification is necessary because each component in the multistage architecture, as an independent element, stores its own management information. That is, the management information is dispersed internally and there is no single management information base (MIB) representing the whole multistage architecture. Therefore building such a unified MIB is one of the fundamental requirements.

2) Management information coherence: The internal network is prone to failure. In such situation management information of the failed device is not accessible. Besides internal component failures, other functional modules of the architecture controls the internal devices in achieving some objective. For instance, the energy module proposed in Chapter 3 turns off back-end routers to save energy consumption of the architecture. During this off period, the MIB of the turned off device is not available to the internal manager. Those internal fluctuations result in management information inconsistency when observed by the external manager unless the internal manager copes up with such internal effects.

3) Scalability: The internal manager runs on one of the back-end router;

namely the virtualCP. The amount of management information to be gathered from internal devices at any time should not impede the performance of the virtualCP as it is responsible for internal architecture coordination. Furthermore excessive management information (if any) should not affect internal link bandwidth especially that of the VirtualCP. Thus, in designing a multistage management system, scalability should have to be taken into account as well.

4) Predict non-management capable devices management information: Some internal devices, for instance LBs implemented on NetFPGA, of the MSSR architecture do not have management capability. In other words, the standard management agents do not run in these devices to collect management information. The internal manager, therefore, has to predict information related to devices that do not have management capability.

5.3 Proposed management architecture

Given the MSSR management system requirements, we present our solution to the management challenges an MSSR architecture faces in the following sections. The proposal is based on building a virtual MIB. That is the virtualCP does not have most of required management information locally rather collects the information from internal devices and compiles it at the time an external manager requests for it. This design approach has the advantage of being scalable as will be discussed later in this chapter. The other design approach could be to build a MIB local to the virtualCP and regularly update the MIB with the information collected from internal devices irrespective of a request from external manager. This could results in management information flooding in the internal network and could threat the management system as well as the whole architecture scalability. We postponed this research as a future activity.

As detailed in Section 2.2, SNMP is successful in network management and it is a widely deployed protocol. And hence we consider SNMP as preferred management protocol for MSSR network management as well. We also adopted the hierarchical architecture management approach shown in Fig. 5.1 as a network management systems (NMS). It uses the concept of manager of managers sitting at a higher level and requesting information from smaller domain managers in a tree like fashion.

We propose an SNMP dual-role entity named *aggregator* that coordinates the internal, independently running, SNMP *agents* and interacts with the external managers issuing SNMP requests. The aggregator is logically one of the modules of the virtualCP, the entity in charge of architecture control and coordination. However, it may run independently of other modules, possibly in a different back-end router.

Load balancers redirect any external SNMP requests to the aggregator. When an SNMP request is received, the aggregator queries the internal SNMP agents to

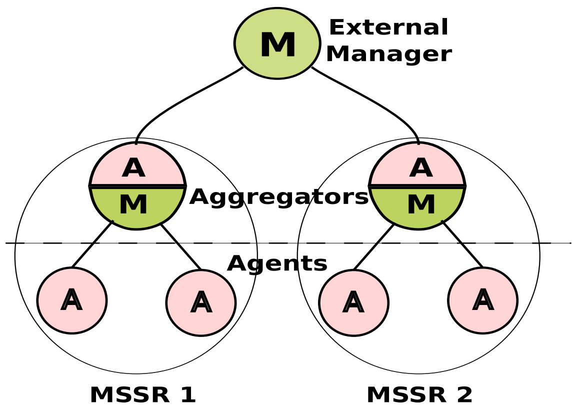


Figure 5.1. Management System used in multistage Software Router: logical architecture

obtain the required MIB information and aggregates the information representing the whole multistage architecture. In other words, the aggregator is a dual-role entity located at the mid-level of the hierarchy: for external managers it acts as an agent and for internal agents it acts as a manager.

In terms of real implementation, whereas LBs and the switch (may) run an SNMP agent, back-end routers host both the *aggregator* and the *agent* functionalities. More precisely, each back-end router runs two instances of an SNMP process: an aggregator (listening on the standard SNMP port to be reachable from external hosts) and a standard agent listening on a different (configurable) port, used for internal communication only. Even if all the back-end routers are listening on the standard SNMP port, only one aggregator handles the external requests, because LBs forward SNMP request to the *active* aggregator only which resides on the back-end router designated as the one hosting the virtualCP by the DIST protocol. The standard SNMP agent instance permits to have the same interface towards all elements of the architecture and is used to collect the local information.

All back-end routers run both SNMP instances for resilience purpose. Indeed, if the currently active aggregator fails, another one can quickly take over, avoiding management failures. The take over procedure is taken care by the DIST protocol. When DIST detects a failure of the current aggregator, it elects a new one and reconfigures the LBs to properly redirect SNMP traffic.

Observe that not all the internal elements may be SNMP-capable. Whereas back-end routers can be assumed to be SNMP-capable because they are based on Linux PCs, LBs, especially if hardware based, might not run an SNMP agent. Therefore, we assume two classes of load balancers: SNMP-capable and SNMP-incapable. This assumption affects the way in which the aggregator collects and computes MIB variables. In the first case, the aggregator can directly collect the proper information from the LBs agents. In the second case, either some MIB variables are approximated on the basis of the data available at the back-end routers or an alternative collection mechanism is deployed, as detailed in Sec. 5.3.3. This is in line with one of the management requirements listed in Section 5.2, i.e. predicting non-management capable devices management information.

To ease information sharing among aggregators (which is needed for a quick takeover in case of failure) and communication with the DIST protocol entity, the architecture also comprises a database. Among others, the database stores configuration information of the internal elements and most MIB counters. If the LB is not SNMP-capable, a minimal set of interface statistics, namely the received/transmitted bytes/packets and the link speed information are also saved in the database. The database is important, in general, to keep management information coherence. Database resilience issues are not discussed because standard techniques can be adopted.

The proposed *aggregator* module shares some features with the SNMP proxy agent, documented in RFC 2576 [73]. However the purpose of a proxy agent is message translation between the different SNMP versions, while the work presented here focuses on creating a single-entity view of the multistage architecture to be presented to external SNMP-capable peers.

5.3.1 Manager-agent communication model

The standard communication scenario used in SNMP [74] works for a single device which has all the information in the local MIB. However, in the studied multistage architecture, as already stated, the aggregator does not have the whole information locally available. This requires a modification to the standard SNMP manager-agent communication model.

Fig. 5.2 shows the modified manager-agent communication model. The steps highlighted in the dashed box are the required extension to deal with the multistage architecture. Upon a request reception, the aggregator agent decodes the request to extract the object identifiers (OIDs) and checks the variable availability. If the variable is locally available, the aggregator manager responds reporting the current variable value. Otherwise, the aggregator sends SNMP requests to the appropriate internal element(s), collects the response(s) received within a given timeout [2] and, if required, aggregates the data to create the single-entity MIB variable. Finally, the aggregator agent answers to the original external SNMP request.

If multiple responses are expected from the internal elements, a response to the external manager's request is sent on the basis of the available information at a given time, even if some responses from internal agents are not available yet. For those

[2]In our implementation, the internal timeout was set to one second

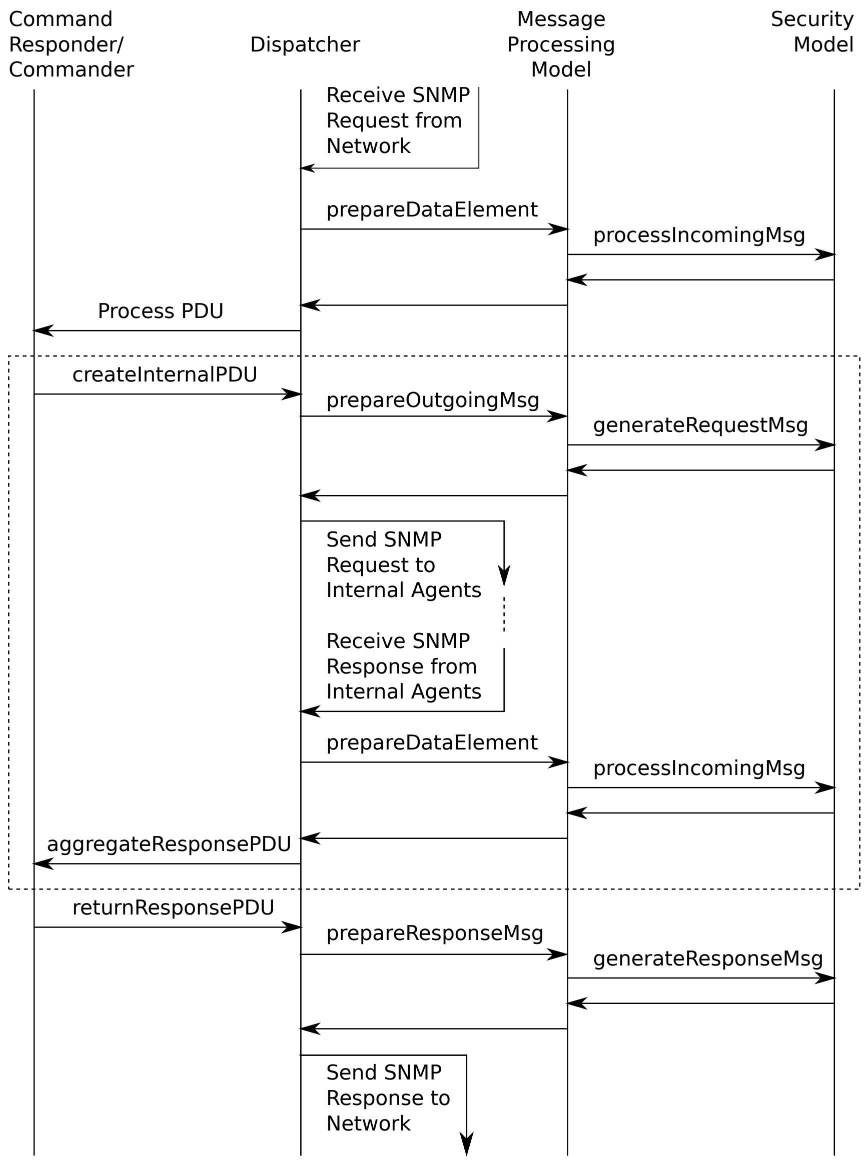


Figure 5.2. Modified manager-agent communication model for the multistage software router

elements which did not respond for whatever reason, the aggregator uses, if available, the corresponding variable value saved in the database at the previous successful request. On reception of a new request for a counter type MIB variable, if the agent comes back to service, the aggregator checks to detect any variable re-initialization: if found, the old value contained in the database and the newly available counter value are summed up to mask the discontinuity. This compensation guarantees that counters are kept monotonically increasing. Violating the monotonicity behavior of counters would be disturbing for the external management software, because these values are typically used to compute temporal trends.

5.3.2 multistage router MIB

In the MIB definition of our multistage software router architecture, we mainly consider, among all variables defined in the MIB tree, the system, the interface (IF) and the IP group objects and demonstrate on how to build a unified management information for MSSR architecture. This is one of the main MSSR management requirements that we want to achieve (See Section 5.2 for detail).

The MIB variables can be grouped into two main categories, based on how the aggregator computes the response:

Global Variables This category contains variables which are global for the multistage software router, e.g., the routing table (*ipRouteTable*), the system up time (*sysUpTime*) or the system name (*sysName*). These variables do not depend on a specific internal element; hence, they are stored in the database to ease information sharing among all aggregators. A response to an external SNMP request for this type of variables translates simply into a query to the database. The database might be populated by the aggregator itself or by the DIST daemon depending on the specific information. For example, the system name is provided by the aggregator, while the routing information is updated by DIST.

Collected Variables This category comprises all the variables requiring collection of data from one or more internal agents, e.g., interface information. A further division is between *specific* and *aggregated* variables. *Specific* variables can be fetched through a single request to a specific internal element. This group comprises all the variables containing specific properties of an internal element, e.g., the link status (*ifOperStatus*) or the link speed (*ifSpeed*). Instead, *aggregate* variables need multiple queries to different internal agents and require some kind of data aggregation. For instance, the total number of received packets at the IP layer (*ipInReceives*) or the discarded packets at the interface (*ifInDiscards*) are computed using counters from several internal elements.

5.3.3 Single-entity management information view: the case of *aggregate* variables

Global and *collected* variables are easy to handle: a simple request forwarding either to the database manager or to an internal agent is needed. Thus, we focus on how to compute the more complex *aggregate* variables. These variables mainly comprise *IF* and *IP* counters.

Fig. 5.3 shows the main counters involved during packet forwarding, both in a single device router and in the multistage software router. The challenge is to define a mapping between the counters on the left, representing the single-entity view we want to achieve, and the counters on the right distributed over the three stages of the multistage architecture. In the following subsections we present such mapping. Where ambiguity might exist, we use an over-line to indicate the mapped computed variables and the superscript LB and BR for counters at LB or back-end router interface respectively. Furthermore, for simplicity, we define $ifInPkts$ as the sum of both unicast ($ifInUcastPkts$) and non-unicast ($ifInNUcastPkts$) packets.

$ifInOctets$, $ifInErrors$ and $ifInUnknownProtos$

$ifInOctets$, $ifInErrors$ and $ifInUnknownProtos$ count respectively the number of received bytes, the number of discarded packets due to errors and to unknown/unsupported protocols. These counters are interface specific and, therefore, simply treated as collected variables.

As already introduced earlier, the computation of MIB variables may be difficult because some elements may be non SNMP-capable. For this reason, we consider three different cases:

- SNMP-capable LBs: SNMP messages are used;
- SNMP-incapable, DIST-capable LBs: The existing control plane is extended to transport minimal traffic statistics (e.g. packet and byte counters);
- SNMP-incapable and DIST-incapable LBs: Data collection is not possible. Counters are approximated using the information available at the back-end routers (which are SNMP-capable), assuming a uniform distribution of traffic among front-end interfaces. Unfortunately, $ifInErrors$ and $ifInUnknownProtos$ variables are not available in this case, because these events occur at LBs' interfaces only.

Similar considerations apply to the outgoing counterparts: $ifOutOctets$, $ifOutErrors$ and $ifOutUnknownProtos$.

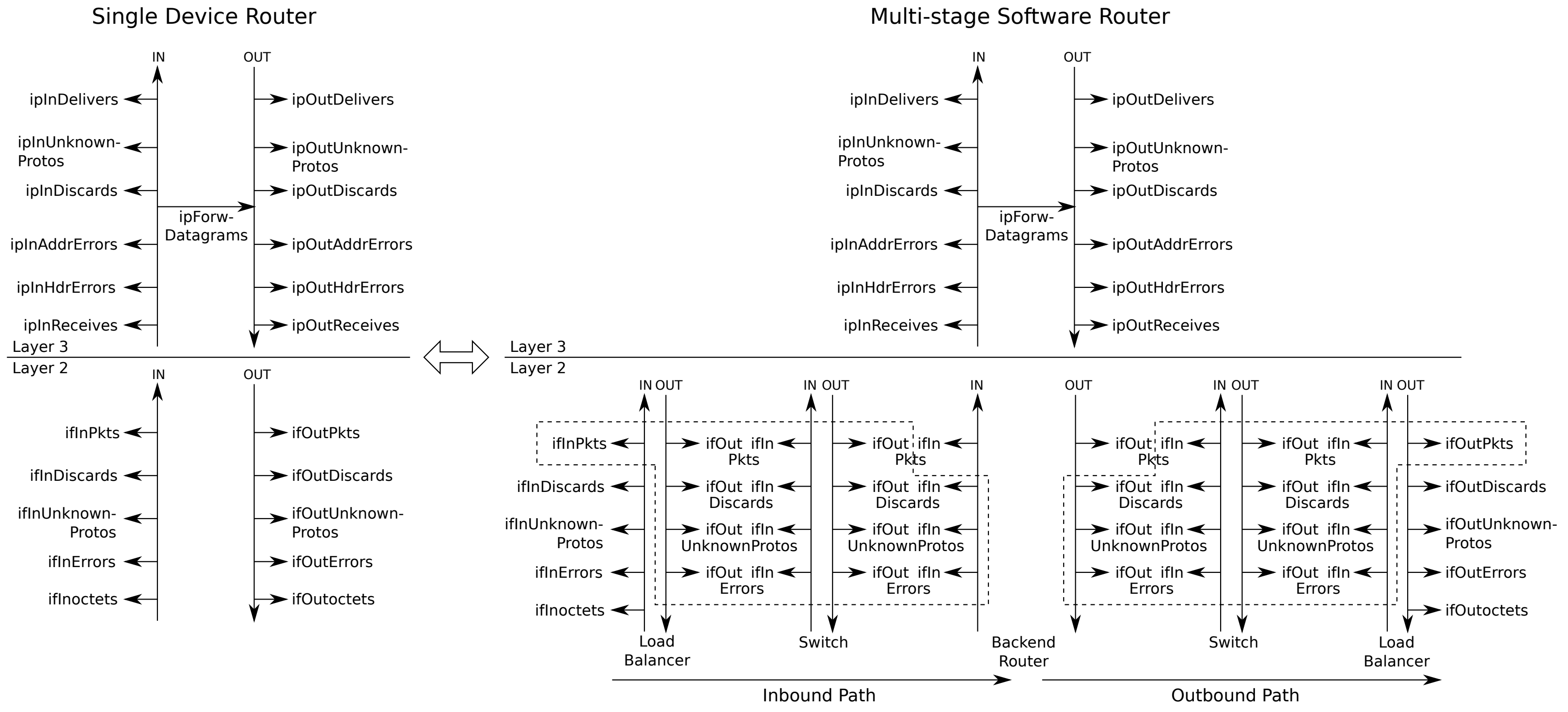


Figure 5.3. Main IF and IP counters involved in packet forwarding for a single-stage router (right) and the multistage software router (left)

$\overline{ifInDiscards}$

As defined in the RFC 1213 [53], the variable $ifInDiscards$ counts the packets which, even if correct at reception time, are discarded by the device for any reason. We use this definition to compute all packets lost while traversing the internal network of the multistage architecture. However, it is not possible to track the exact path (and thereof the exact counters involved) of each packet within the multistage architecture, due to the unpredictable decision of the load balancing scheduler. Hence, we define D_i as the share of packets internally discarded for interface i.

D_i is computed as the difference of the correctly received packets at the input interface of the LB ($ifInPkts_i^{LB}$) and the sum of correctly received packets at all the interfaces of the back-end routers R^{BR}, weighted by w_i, the percentage of traffic received at interface i. R^{BR} and w_i are computed as:

$$R^{BR} = \sum_{j=1}^{M}(ifInPkts_j^{BR}) \tag{5.1}$$

$$w_i = \frac{ifInOctets_i^{LB}}{\sum_{k=1}^{N} ifInOctets_k^{LB}} \tag{5.2}$$

where M is the total number of back-end router interfaces, and N the total number of external LBs interfaces. Thus,

$$D_i = ifInPkts_i^{LB} - w_i R^{BR} \tag{5.3}$$

$$\overline{ifInDiscards_i} = ifInDiscards_i^{LB} + D_i \tag{5.4}$$

The above formulas make the implicit assumption that the loss probability is the same on all internal paths, but it has the nice property of being completely independent of the internal load balancing scheme adopted. Thanks to this property, the same procedure can also be applied on the reverse path to compute $\overline{ifOutDiscard_i}$ without knowing the result of the routing operation.

In case of SNMP-incapable and DIST-incapable LB, $\overline{ifInDiscards_i}$ is directly approximated by D_i, replacing $ifInOctets$ and $ifInPkts$ in Eq. (5.1)-(5.4) with the received bytes $rxBytes$ and received packets $rxPkts$ statistics, respectively, stored in the database by DIST.

$\overline{ifInPkts}$

$ifInPkts$ is the sum of all the corresponding counters at the back-end routers weighted by w_i.

$$\overline{ifInPkts_i^{LB}} = w_i(\sum_{j=1}^{M} ifInPkts_j^{BR}) \tag{5.5}$$

For SNMP-incapable and DIST-incapable LB the same substitution as for $ifInDiscards$ apply.

IP counters

The IP counters are located only at the back-end routers. The mapping consists of the sum of all the corresponding IP counters at the back-end routers. For instance, $\overline{ipInReceives}$ is computed as:

$$\overline{ipInReceives} \;=\; \sum_{j=1}^{M} ipInReceives_j^{BR} \tag{5.6}$$

sysUpTime

In addition to the above counters, a special mention is needed for the $sysUpTime$, a global variable used to store the elapsed time since the management system was running. This information is used as a time reference for the other variables by the external management software, to plot temporal graphs. Given that the aggregator can run on different back-end routers at different time, it is important that the $sysUpTime$ is not related to a specific instance of the aggregator, but rather tied to the up time of the whole architecture. To achieve this, the first aggregator stores the reference startup time into the database. When an aggregator fails and another takes over, the start up information remains the same. The $sysUpTime$ is re-initialized if and only if all the back-end routers fail.

5.3.4 Scalability analysis

The use of a centralized aggregator has the advantage of reduced management complexity. However, scalability issue might arise due to the concentration of SNMP traffic. We said in Section 5.2 scalability should be considered as one of the management system requirement. Thus, we try to estimate the amount of SNMP traffic internally generated to process an external SNMP request. The worst case scenario is a request for $IfInDiscards$, because it implies the collection of the largest number of variables from the multistage architecture (see Eq. (5.1)-(5.4)).

As reported in Section 5.3.3, M is the total number of back-end router interfaces, meanwhile N is the total number of external LB interfaces. Eq. (5.1) requires to collect $2M$ variables, because $IfInPkts$ is the sum of two variables and Eq. (5.2) requires N variables. Furthermore, three more variables are needed for Eq. (5.3) and (5.4). In the worst case, for each variable two SNMP messages (request and response) are required. Typically, the management station repeats the requests in time to plot temporal graphs and keep device history. Therefore, the amount of management traffic can be computed as:

$$\begin{aligned} \text{total\_traffic} &= \frac{2(2M+N+3)S}{T} \\ &\approx \frac{2(2M+N)S}{T} \end{aligned} \tag{5.7}$$

where S is the SNMP message size, typically about 100 bytes for SNMP response [75], and T is the update period, typically set to about 5 minutes.

Let us now consider two scenarios: i) a medium range edge router with 360 interfaces at 1Gbps (i.e. a mid-range 7600 series Cisco router [76]) and ii) a core router with 16 interfaces at 10Gbps (i.e. a high-end Juniper T series router [20]). Assuming back-end PCs with 1Gbps routing capability and one LB per interface (worst case in terms of generated messages), we have that for i) $M = 360$,$N = 360$ and for (ii) $M = 160$,$N = 16$. Even assuming a very aggressive update period of $1s$, the management traffic would be equal to 216 KBytes/s and 67 Kbytes/s respectively for one MIB variable. Even considering tens of MIB variables traced by the management station, the management traffic is negligible with respect to the total forwarding routing capacity, posing no threat to the overall architecture.

The above example considers the worst case scenario where one request is sent to collect one variable from internal device. In our implementation, however, the *aggregator* generates a request containing a list of varbinds to a device to collect more than one variable instances in one response to reduce SNMP traffic generated internally. For each additional variable to the list we have an average 18bytes payload. For one variable the SNMP packet average packet size is 100bytes and for two variables it will be 118bytes and so on. That is, if k is the number of variables in the varbinds list and l the payload due to one variable in the list, then the total SNMP traffic generated will be

$$Total\_bytes \cong SNMP\_packet\_size_{avg} + k \times l \tag{5.8}$$

Thus instead of sending individual request to internal devices, request to multiple variables are bundled together into a single SNMP message, which would allow to significantly reduce the number of messages and increase transmission efficiency. Furthermore, (5.7) overestimates the real internal management traffic, because an SNMP request is smaller than an SNMP response message.

From these arguments we conclude that the proposed management system scales well for large size of MSSR and does not pose a threat to the overall architecture scalability in general.

5.3.5 Software implementation

The management architecture proposed in this chapter was implemented and verified in a test-bed, similar to the figure shown in Fig. 1.3. More precisely, the prototype is based on a customized version of Net-SNMP (ver. 5.4.2.1) [60] and MySQL DBMS in addition to the software required to implement the multistage architecture.

5.4 Conclusions

The multistage software router, being a distributed router architecture, requires a coordinated information management to mask the internal structure and to present the architecture to external managers (e.g. Cacti and MRTG) as a single device.

We defined a hierarchical management system based on three elements (a *manager*, *aggregators* and *agents*) as an extension of the standard manager-agent communication model. The new system consists of a multistage distributed MIB and an extended communication model, which define the mechanisms to collect data from distributed elements in a reliable way and to aggregate the data in an unified view. The net-SNMP 5.4.2.1 [60] implementation has been modified and tested in a small scale test-bed and its scalability was assessed through simple load computations for two classical high-end router configurations.

Chapter 6

Conclusions

In this chapter overall conclusions to the research topics discussed in this thesis in accordance with the research question identified in Chapter 1 are presented (Section 6.1) and finally conclude the thesis by pointing out possible extension and future research directions in Section 6.2

6.1 Overall conclusions

A multistage software router architecture has the advantage of low cost, flexibility and programmability over their proprietary networking devices and overcomes a single PC based software routers performance limitation by offering multiple, parallel forwarding paths. Scalability issues of single PC based software routers has also been addressed in MSSR architecture since the number of interfaces as well as performance can be increased/improved by incrementally adding/upgrading internal elements seamlessly.

Like many networking devices, the MSSR is typically designed for the peak traffic and hence a high-end MSSR architecture might require tens or hundreds of PCs. As the number of internal devices increases, however, MSSR faces two challenges: control and management complexity and energy wastage during low traffic periods. The large amount of energy consumption at high load could also threaten the scalability feature of the MSSR architecture. In addressing these issues, this thesis dare to develop a centralized MSSR management system that (i) collects and compiles a unified information management system, and (ii) resize the MSSR architecture to the input traffic in an efficient way such that the power wastage is minimized during low load periods.

To solve energy related issues of MSSR architecture, three algorithms, namely optimal, two-step and on-line differential algorithms, with their respective pros and cons have been proposed in Chapter 3 to address the Research Question 2. The algorithms adapt the energy consumption of multistage software router to a traffic load by properly choosing a set of active back-end routers to match incoming load. Results show that the proposed algorithms save up to 57.44% of the architecture energy consumption compared to the no saving scheme at low loads.

The two-step algorithm takes a divide-and-conquer approach to overcome the scalability issue of the optimal algorithm. Even if the single steps in the two-step approach are optimal, the combined solution however is not. This is because the information required to globally optimize the system is partitioned among the two steps making them less optimal from a global point of view. The two-step scales well to a practical size of MSSR architecture and closely approximate the optimal with a solution quality within 9% difference in the worst case considered.

Both the optimal and two-step algorithms face service disruption and/or increased delay during reconfiguration period. The on-line differential algorithm, even though less optimal compared to the off-line approaches (both the optimal and two-step algorithms), has the advantage of being non disruptive and therefore superior to the off-line approaches in terms of QoS provision. Moreover, it scales well to solve large size MSSR configuration. The worst case performance gap of on-line algorithm with with respect to the optimal algorithm is about 27.7% .

In Chapter 4 an energy efficient back-end routers design approaches that supplement the efficiency of energy saving algorithms have been proposed. The back-end PCs cluster design approaches permit to save up to 10% of energy with respect to existing design approaches for similar budget constraint. The design schemes in combination with the energy saving algorithms makes the MSSR implementation practical by reducing energy wastage in MSSR architecture that could threaten its scalability feature.

The answer to the Research Question 4 has been provided in Chapter 5. Multistage architecture is build on the principle that the whole architecture will be viewed as a single device to external network. This principle applies for the management information as well. Therefore, a mechanism that coordinated information management to mask the internal MSSR structure and present the architecture to external managers as a single device has been proposed as a solution to information management complexity.

The MSSR management system is designed to satisfy requirements: unified management view, management information coherence, scalability and capability to interact with management incapable internal devices. The system consists of an extended communication model, which defines mechanisms to collect data from distributed internal elements in a reliable way and aggregate the data to create a unified management view. A database local to the virtualCP provides mechanism to mask information inconsistency and also use to store management incapable devices management information. From scalability test, we argued that the proposed management system scales well and does not pose scalability threat to the overall architecture.

6.2 Future research directions

In this section we present some future research directions in MSSR management:

- In Chapter 3 we considered a single link NIC scenario where turning off a link implied turning off the network card itself. Both the energy saving problem modeling and simulation can be extended to include a multi-link network card scenario where saving in the links and NICs are performed separately.

- Our MSSR back-end router cluster design discussed in Chapter 4 does not consider the QoS issues. Inclusion of such parameter increases the complexity of the problem but also results in a more complete work. In addition capacity scaling of an MSSR architecture is performed by adding one or more PCs to the back-end stage. However, the PCs to be added to the architecture has to be carefully selected in such a way that the energy efficiency of the architecture is not compromised. Hence inclusion of quality of service issues as an additional design requirement and energy efficient capacity scaling could be potential research activities in future.

- In extending Research Question 4 one could ask if it is possible to develop a single MIB local to the virtualCP. In Chapter 5 we considered a virtual MIB - the internal manager has to collect and compile SNMP variables from internal components upon arrival of a request from external manager. The time need to collect and compile the variables may increase with the number of internal devices which could result in response delay. This is undesirable for some time critical applications. Therefore, compiling the information ahead before the external manager request for it could be an alternative to reduce dissatisfaction by the external manager operator [77, 78]. This requires building a MIB - a data base that stores compiled management information - local to the virtualCP and updating frequently whether external managers ask for management information or not. Thus, responding to an external manger request just involves querying the local data base which would be much faster than the method proposed in this thesis. In addition the management system could include the remaining SNMP features not considered in this thesis, namely support to SNMP *set* and *trap* messages. How to set a MIB variable which is located in scattered manner in the software router and get trap information about the software router are other challenges for future work.

Appendix A

Splittable item with variable bin size and cost - mapping

The combined problem considering router and link optimization is too complex and it is not easily mappable to any of the well-known NP-hard problems when considered in its full form. 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 energy consumption.

The simplified version of our problem is as follows:

$$\begin{aligned}
\text{minimize} \quad & \textstyle\sum_r P_r \alpha_r & & \text{(A.1)}\\
\text{s.t.} \quad & \textstyle\sum_r t_{ir} = 1 \quad \forall i \in I & & \text{(A.2)}\\
& \textstyle\sum_i T t_{ir} \le C_r \alpha_r \quad \forall r \in B & & \text{(A.3)}\\
& \alpha_r \in \{0,1\} & & \text{(A.4)}\\
& t_{ir} \in \{0,1\} \vee [0,1] & & \text{(A.5)}
\end{aligned}$$

where I is the set of all items to be packed. This formulation is equivalent to the formulation of the single steps described in Sec. 3.1.2, except for the fact that here we consider a more general formulation with many items which may be splittable (e.g. $t_{ir} \in [0,1]$) or unsplittable (e.g. $t_{ir} \in \{0,1\}$).

If the items are unsplittable, the problem can be directly mapped to the Generalized Cost Variable Sized Bin-Packing problem [65], where the items are represented by traffic 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 too.

On the other side, the splittable problem can be mapped to the Knapsack problem [68], 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 without considering network flows). 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. In fact, the KNAPSACK problem is as follows:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_i v_i x_i && \text{(A.6)} \\ \text{s.t.} \quad & \textstyle\sum_i w_i x_i \leq W && \text{(A.7)} \\ & x_i \in \{0,1\} && \text{(A.8)} \end{aligned}$$

where x_i is the selection variable for router i, v_i and w_i are respectively the *value* and the capacity of the router i and W is the size of the traffic. The two main differences in the formulations are:

1. the KNAPSACK problem is a maximization problem, while our problem is a minimization problem, due to the usage of values instead of costs.
2. in the KNAPSACK problem the total size of selected items must not exceed W, meanwhile in our case it must be at least equal to W in order to allocate enough capacity to forward the input traffic T.

The first issue is negligible according that a correct transformation from energy costs P_i to values v_i exists (e.g. $v_i = \frac{1}{P_i}$). The second issue is also negligible according that it is possible to obtain the minimum $W \geq T$ such that $\sum_i w_i x_i \leq W$ and $\sum_i w_i x_i \geq T$. If such 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 [68], thus the splittable version of our simplified problem is NP-hard as well as its complete form.

Finally, a simple algorithm to select the optimal W is based on the iterative solution of a sequence of KNAPSACK problems, starting with $W = T$ then increasing W by the minimum unit at every step until the condition in Eq. A.9 is verified:

$$\sum_i w_i x_i \leq W \wedge \sum_i w_i x_i \geq T \tag{A.9}$$

When the condition in Eq. A.9 is verified the optimal W is obtained as well as the optimal solution for the simplified energy saving problem.

Bibliography

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011–2016," white paper. [Online]. Available: http://www.cisco.com/en/US/netsol/ns827/networking_solutions_sub_solution.html [Accessed: Dec., 2012]

[2] "Internet usage statistics." [Online]. Available: http://www.internetworldstats.com/stats.htm [Accessed: Nov., 2012]

[3] R. Schaller, "Moore's law: past, present and future," *Spectrum, IEEE*, vol. 34, no. 6, pp. 52–59, 1997.

[4] R. Tkach, "Scaling optical communications for the next decade and beyond," *Bell Labs Technical Journal*, vol. 14, no. 4, pp. 3–9, 2010.

[5] J. Turner and D. Taylor, "Diversifying the internet," in *Global Telecommunications Conference, 2005. GLOBECOM'05. IEEE*, vol. 2. IEEE, 2005, pp. 6–pp.

[6] "Linux." [Online]. Available: http://www.kernel.org/ [Accessed: Dec., 2012]

[7] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click Modular Router," *ACM Trans. Comput. Syst.*, vol. 18, no. 3, pp. 263–297, 2000.

[8] M. Handley, O. Hodson, and E. Kohler, "XORP: an Open Platform for Network Research," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 53–57, 2003.

[9] "Quagga routing suite." [Online]. Available: http://www.nongnu.org/quagga/ [Accessed: Dec., 2012]

[10] A. Bianco, R. Birke, D. Bolognesi, J. Finochietto, G. Galante, M. Mellia, M. Prashant, and F. Neri, "Click vs. Linux: two efficient open-source IP network stacks for software routers," in *High Performance Switching and Routing, 2005. HPSR. 2005 Workshop on*, May 2005, pp. 18–23.

[11] M. Dobrescu, N. Egi, K. Argyraki, B. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: Exploiting parallelism to scale software routers," in *ACM SOSP*, 2009, pp. 15–28.

[12] "Vyatta Series 2500." [Online]. Available: http://www.vyatta.com/ [Accessed: July, 2011]

[13] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: a GPU-accelerated software router," *SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 195–206,

August 2010.

[14] "Cisco carrier routing system." [Online]. Available: http://www.cisco.com/en/US/products/ps5763/index.html [Accessed: Dec., 2012]

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

[16] A. Bianco, J. Finochietto, G. Galante, M. Mellia, D. Mazzucchi, and F. Neri, "Scalable layer-2/layer-3 multistage switching architectures for software routers," *GLOBECOM 2006*, 2006.

[17] K. Argyraki, S. Baset, B. Chun, K. Fall, G. Iannaccone, A. Knies, E. Kohler, M. Manesh, S. Nedevschi, and S. Ratnasamy, "Can software routers scale?" in *Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow.* ACM, 2008, pp. 21–26.

[18] I. Keslassy and N. McKeown, "Maintaining packet order in two-stage switches," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2. IEEE, 2002, pp. 1032–1041.

[19] A. Bianco, R. Birke, J. Finochietto, L. Giraudo, F. Marenco, M. Mellia, A. Khan, and D. Manjunath, "Control and management plane in a multistage software router architecture," in *High Performance Switching and Routing, 2008. HSPR 2008. International Conference on*, May 2008, pp. 235–240.

[20] "T Series Core Routers." [Online]. Available: www.juniper.net/us/en/local/pdf/datasheets/1000051-en.pdf [Accessed: Dec., 2012]

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

[22] ——, "On-line energy saving in a distributed multistage router architecture," pp. 1–6, December 2012.

[23] A. Bianco, F. G. Debele, and N. Li, "Energy efficient distributed router design," *Communications (ICC), 2013 IEEE International Conference on*, June 2013.

[24] A. Bianco, R. Birke, F. G. Debele, and L. Giraudo, "SNMP Management in a Distributed Software Router Architecture," in *Communications (ICC), 2011 IEEE International Conference on.* IEEE, 2011, pp. 1–5.

[25] A. Bianco, J. M. Finochietto, G. Galante, M. Mellia, and F. Neri, "Open-Source PC-based software souters: A viable approach to high-performance packet switching," in *QoS-IP (Third international workshop on QoS in multiservice IP networks).* Springer Berlin / Heidelberg, 2005, pp. 353–366.

[26] A. Bianco, R. Birke, G. Botto, M. Chiaberge, J. Finochietto, G. Galante, M. Mellia, F. Neri, and M. Petracca, "Boosting the performance of PC-based

software routers with FPGA-enhanced network interface cards," in *High Performance Switching and Routing, 2006 Workshop on.* IEEE, 2006, pp. 6–pp.

[27] 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.

[28] C. Partridge, S. Member, P. P. Carvey, I. Castineyra, T. Clarke, J. Rokosz, J. Seeger, M. Sollins, S. Starch, B. Tober, G. D. Troxel, D. Waitzman, and S. Winterble, "A 50-Gb/s IP router," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 237–248, 1998.

[29] A. Bianco, R. Birke, L. Giraudo, and N. Li, "Multistage software routers in a virtual environment," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE.* IEEE, 2010, pp. 1–5.

[30] R. Bolla, R. Bruschi, G. Lamanna, and A. Ranieri, "DROP: An open-source project towards distributed SW router architectures," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE.* IEEE, 2009, pp. 1–6.

[31] L. Yang, R. Dantu, T. Anderson, and R. Gopal, "Forwarding and control element separation (forces) framework," *RFC3746*, pp. 5–30, 2004.

[32] V. Manral, R. White, and A. Shaikh, "Benchmarking basic OSPF single router control plane," *IETF Request for Comments*, vol. 4061, 2005.

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

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

[35] "Approximate desktop, notebook, & netbook power usage." [Online]. Available: http://www.upenn.edu/computing/provider/docs/hardware/powerusage.html [Accessed: March, 2012]

[36] D. Neilson, "Photonics for switching and routing," *Selected topics in quantum electronics, IEEE Journal of*, vol. 12, no. 4, pp. 669–678, July–Aug. 2006.

[37] "ARBOR Networks." [Online]. Available: http://asert.arbornetworks.com/2009/08/the-internet-after-dark/ [Accessed: Dec., 2012]

[38] M. Gupta and S. Singh, "Greening of the Internet," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '03. ACM, 2003, pp. 19–26.

[39] J. Chase and R. Doyle, "Balance of power: Energy management for server clusters," in *Proceedings of the 8th Workshop on Hot Topics in Operating Systems (HotOS)*, 2001, pp. 163–165.

[40] E. Pinheiro, R. Bianchini, E. 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.

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

[42] Broadcom Corporation, "Broadcom NetXtreme II network adapter user guide." [Online]. Available: http://www.broadcom.com/docs/support/ethernet_nic/Broadcom_NetXtremeII_Server_T7.4.pdf [Accessed: Dec. 2012]

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

[44] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. USENIX Association, 2008, pp. 323–336.

[45] L. Chiaraviglio, M. Mellia, and F. Neri, "Energy-aware backbone networks: A case study," in *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*, June '09, pp. 1–5.

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

[47] "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: Dec., 2012]

[48] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future Internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *Communications Surveys Tutorials, IEEE*, vol. 13, no. 2, pp. 223–244, quarter 2011.

[49] L. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The Google cluster architecture," *Micro, IEEE*, vol. 23, no. 2, pp. 22–28, March-April 2003.

[50] Q. Ye and M. MacGregor, "Cluster-based IP router: Implementation and evaluation," in *Cluster Computing, 2006 IEEE International Conference on.* IEEE, 2006, pp. 1–10.

[51] "Windows clustering." [Online]. Available: http://technet.microsoft.com/en-us/library/cc757731(v=ws.10) [Accessed: Dec., 2012]

[52] "Domino 8.0 administration." [Online]. Available: http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp [Accessed: Dec., 2012]

[53] K. McCloghrie and M. Rose, "RFC 1213 Management Information Base for

Network Management of TCP/IP-based internets: MIB-II," 1991. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1213.txt [Accessed: Nov., 2012]

[54] S. Aidarous and T. Plevyak, *Telecommunications Network Management:Technologies and Implementations.* Wiley-IEEE Press, 1998.

[55] "Information technology - open systems interconnection - systems management overview," 1992. [Online]. Available: www.itu.int/rec/T-REC-X.701-199201-S/en

[56] J. Case, and M. Fedor, and M. Schoffstall, and J. Davin, "RFC 1157 A Simple Network Management Protocol (SNMP)," 1990. [Online]. Available: http://tools.ietf.org/html/rfc1157 [Accessed: Nov., 2012]

[57] Aiko Pras and Thomas Drevers and Remco van de Meent and Dick A. C. Quartel, "Comparing the performance of SNMP and Web services-based management," *IEEE Transactions on Network and Service Management*, pp. 72–82, 2004.

[58] A. Pras and J. Martin-Flatin, "What can Web Services bring to integrated management?" *Handbook of network and system administration*, p. 241, 2007.

[59] J. Schonwalder, A. Pras, and J. Martin-Flatin, "On the future of Internet management technologies," *Communications Magazine, IEEE*, vol. 41, no. 10, pp. 90–97, 2003.

[60] "Net-SNMP." [Online]. Available: http://net-snmp.sourceforge.net/ [Accessed: Dec., 2012]

[61] Intel, "Intel network adapters user guide." [Online]. Available: http://www.intel.com/support/network/sb/cs-009715.htm [Accessed: Dec., 2012]

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

[63] 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.

[64] "IBM ILOG CPLEX Optimization Studio." [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/ [Accessed: Dec., 2012]

[65] L. Epstein and A. Levin, "An APTAS for generalized cost variable-sized bin packing," *SIAM J. Comput.*, vol. 38, pp. 411–428, April 2008.

[66] 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.

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

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

[69] E. R. Enns, "RFC 4741 NETCONF Configuration Protocol," 2006. [Online]. Available: http://tools.ietf.org/search/rfc4741 [Accessed: Nov., 2012]

[70] E. B. Claise, "RFC 3954 Cisco Systems NetFlow Services Export Version 9," 2004. [Online]. Available: http://tools.ietf.org/html/rfc3954 [Accessed: Nov., 2012]

[71] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2.* Addison-Wesley Longman Publishing Co., Inc., 1998.

[72] A. Clemm, *Network Management Fundamentals.* Cisco Press, 2007.

[73] R. Frye, and D. Levi, and S. Routhier, and B. Wijnen, "RFC 2576 Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework," 2000. [Online]. Available: http://tools.ietf.org/html/rfc2576 [Accessed: Nov., 2012]

[74] D. Harrington, and R. Presuhn, and B. Wijnen, "RFC 3411 An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," 2002. [Online]. Available: http://tools.ietf.org/html/rfc3411 [Accessed: Nov., 2012]

[75] C. Pattinson, "A Study of the behaviour of the Simple Network Management Protocol," *In Proc. of 12th International Workshop on Distributed Systems, Nancy, France*, 2001.

[76] "Cisco 7600 Series Routers." [Online]. Available: http://www.cisco.com/en/US/products/hw/routers/ps368/index.html [Accessed: Dec., 2012]

[77] J. Hoxmeier and C. DiCesare, "System response time and user satisfaction: An experimental study of browser-based applications," in *Proceedings of the Association of Information Systems Americas Conference.* Citeseer, 2000, pp. 140–145.

[78] B. Shneiderman, "Response time and display rate in human performance with computers," *ACM Computing Surveys (CSUR)*, vol. 16, no. 3, pp. 265–285, 1984.

List of Abbreviations

ACL	Access Control List
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
DBMS	Data base management system
DROP	Distributed SW ROuter Project
ECR	energy consumption ratio
FIFO	First In, First Out
ForCES	Forwarding and Control Element Separation
FPGA	Field-Programmable Gate Array
GPUs	Graphics Processing Units
ILP	Integer linear programming
LBs	Load Balancers
MIB	Management Information Base
MILP	Mixed Integer Linear programming
MRTG	Multi Router Traffic Grapher
MSSR	Multi-Stage Software Router
NAT	Network Address Translation
NIC	Network Interface Card
NMS	Network Management Systems

OID	object Identifiers
OSI SM	Open Systems Interconnection - systems managment
PCI	Peripheral Component Interconnect
PDU	Protocol data unit
pps	packets per second
QoS	Quality-of-Service
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SR	Software Router
VirtualCP	Virtual Control Processor
XML	Extensible Markup Language
XORP	eXtensible Open Router Platform

Index

About the author

Fikru Getachew was born in Fitche, Ethiopia, on Setember 19th, 1974. He studied Electrical and Electronics Technology, Nazareth, Ethiopia and graduated in 1998 (Batchelor of Electrical/Electronics Technology). After four years of service in govenmental university as an assistance graduate, he started to pursue his study and obtained Masters of Technology(MTech) from Indian institute of Technology, Kanpur, India in 2004. Until he joined Politecnico di Torino as PhD student in 2009, he served at Arba Minch University, Ethiopia as lecturer. In those years he also worked as an ICT coordinator (2006 - 2009) and Departement head (2004 - 2006). During his office he successfully coordinated and follow up implementation of World Bank financed projects which he won based on competition among other universities in the country.

Since Jan. 2009 he has been with Department of Electronics and Telecommunications (DET) of Politecnico di Torino performing research activity mainly related to distributed software routers and management. His main topics of interest include distributed systems, network management, network virtualization and cloud computing.

A list of his publications in reverse chronological order are:

- Andrea Bianco, Fikru Getachew Debele, Nanfang Li, Energy Efficient Distributed Router Design, IEEE International Conference on Communications, Budapest, Hangary, June 2013
- Andrea Bianco, Fikru Getachew Debele, Luca Giraudo, On-line Differential Energy Saving in a Distributed Router Architecture, IEEE Global Communication conference, California, USA, December 2012
- Andrea Bianco, Fikru Getachew Debele, Luca Giraudo, Energy Saving in Distributed Router Architecture, IEEE International Conference on Communications, Ottawa, Canada, June 2012
- Andrea Bianco, Robert Birke, Fikru Getachew Debele, Luca Giraudo, SNMP Management in a Distributed Software Router Architecture, IEEE International Conference on Communications, Kyoto, Japan, June 2011