SAPIENZA
UNIVERSITÀ DI ROMA

Bandwidth Management in Live Virtual Machine Migration

Department of Information Engineering, Electronic and Telecommunication,
Sapienza University of Rome

Doctorate of Philosophy in Information and
Communications Technologies Engineering – XXIX Cycle

Candidate

Danilo Amendola
ID number 1596823

Thesis Advisors

Prof. Enzo Baccarelli
Ph.Dr. Nicola Cordeschi

Thesis reviewers:
Prof. Stefano Buzzi
Prof. Romano Fantacci

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Information and Communications
Technologies Engineering

15 December 2016

Thesis defended on 27 February 2017
in front of a Board of Examiners composed by:

Prof. Piero Tognolatti

Prof. Fortunato Santucci

Prof. Eugenio Martinelli

Prof.ssa Elena Pettinelli

Prof. Elio Di Claudio

Bandwidth Management in Live Virtual Machine Migration

Ph.D. thesis. Sapienza – University of Rome
ISBN: 000000000-0

This thesis has been typeset by LaTeX.

Version: January 29, 2017

Author's email: danilo.amendola@gmail.com

Abstract

In this thesis I investigated the bandwidth management problem on live migration of virtual machine in different environment. First part of the thesis is dedicated to intra-data-center bandwidth optimization problem, while in the second part of the document I present the solution for wireless live migration in 5G and edge computing emerging technologies.

Live virtual machine migration aims at enabling the dynamic balanced use of the networking/computing physical resources of virtualized data centers, so to lead to reduced energy consumption and improve data centers' flexibility. However, the bandwidth consumption and latency of current state-of-the-art live VM migration techniques still reduce the experienced benefits to much less than their potential. Motivated by this consideration I analytically characterize and test the optimal bandwidth manager for intra-data-center live migration of VMs. The goal is to minimize the migration-induced communication energy consumption under service level agreement (SLA)-induced *hard* constraints on the total migration time, downtime, slowdown of the migrating applications and overall available bandwidth.

For this purpose, after recognizing that the resulting (non-convex) optimization problem is an instance of Geometric Programming, I solve it by resorting to an "ad hoc" developed adaptive version of the so-called primal-dual gradient-based iterations and, then, I analytically characterize its feasibility conditions. The carried out simulations point out that:

(i) the energy savings attained by the proposed bandwidth manager over the state-of-the-art ones currently utilized by Xen, KVM and VMware hypervisors are over 40% and approach 66% under strict QoS constraints;

(ii) the proposed bandwidth manager is capable to quickly adapt to the abrupt changes possibly experienced by the dirty rates of the running applications and/or the round trip times of the utilized (possibly, congested) TCP/IP connections; and,

(iii) its actual implementation may be carried out in a distributed and scalable way, and it consumes less than 1.5% of the CPU computing power per migrated VM.

Second part of this work regards live virtual machine migration into a wireless channel environment, to reduced energy consumption in future live migration context for 5G and Fog computing (also know as edge computing). Then, I analytically characterize and test an optimal tunable-complexity bandwidth manager for live

migration of virtual machines in wireless channel. I present the optimal tunable-complexity bandwidth manager for the QoS live migration of virtual machines under a wireless channel from mobile device to Fog site/access point. The goal is the minimization of the migration-induced communication energy under service level agreement *hard* constrains on the total migration time, downtime and overall available bandwidth. I solved the non-convex problem by resorting to suitably developed adaptive version of the so-called primal-dual gradient-based iteration and, then, I analytically characterize its feasibility conditions. Hence, I develop and simulate the resulting bandwidth manager on a wireless migration environment between a mobile device and a Fog site in scenarios 3G, 4G and WiFi, then, I test and compare its energy performance through extensive simulations.

The thesis is organized in seven chapters, firstly three chapters are dedicated to introduce the context and related work. Therefore, the core of the thesis is spread into two distinguished parts. Finally, I conclude in Chapter 7 that is followed by Appendix and Bibliography. Chapter 1 provide an introduction to virtualization and data-center, in Chapter 2 I discus about the tackled problem and reference technology, then, in Chapter 3 I describe related work for live migration of virtual machine and an in deep presentation of current approaches to live migration is provided (images have been crafted on purpose to improve understandability). Part I, Chapter 4 is dedicated to intra-data-center bandwidth problem. Instead, Part II, Chapter 5 and Chapter 6, concern live virtual machine migration in wireless channel for 5G technology and Chapter 6 provide a brief discussion on the goodness of multi-path TCP compared to single-path TCP. Finally, Chapter 7 contains the conclusion of the thesis.

CONTENTS

CHAPTER 1

INTRODUCTION

The first computer virtualization was done in 1960s on IBM mainframe by G.J. Popek and R.P. Goldberg: "Formal Requirements for Virtualizable Third Generation Architectures" and in [1] 1974 they describes the roles and properties of virtual machines and virtual machine monitors that we still use today.

- ,

Data centers structures are facilities to house server and network components under an engineered solutions. Data centers provide optimized solutions with intensive high performance, high reliability, and optimized energy consumption.

Data centers have their root in huge computer rooms of big industries. Nowadays data centers are located all around the work and new interesting locations are proposed recently (e.g. desert locations, deep sea, etc.), to take advantage of environmental feature. Technology evolution changed the simple computer room into evolved network systems capable to provide fast Internet connectivity and non-stop operation to deploy systems and to establish always-on presence on the Internet. This very large facilities, called Internet Data Centers (IDCs), or similarly Cloud Data Centers (CDCs), that I call in a term: *data center*. They have enabled businesses to do much more with much less, both in terms of physical space and time required to create and maintain data.

During the last two decades data centers' infrastructures quickly growing in terms of number of devices, energy requirement and network complexity, therefore new solutions was provided to better perform services with less energy consumption and more reliability. Traditional data center, it is defined by the physical infrastructure, which is dedicated to a singular purpose and determines the amount of data that

can be stored and handled by the data center as a whole. Early data centers offer sufficient performance and reliability, in particular considering there was little point of reference. But slow and inefficient delivery was a prominent challenge, and utilization was astonishingly low in relation to the total resource capacity. A milestone evolution of data center is the Cloud, between years 2003 and 2010, virtualized data centers produced a big revolution, made it possible to pool the resources of computing, network and storage, using that resources to create a central, with more flexible resource that could be reallocated based on customer's needs.

We are entering in the era of cloud computing. Forecasting of mobile data traffic increase thousand times in 2020 with respect to 2010, and doubling of mobile data traffic every year [2].

With the deployment of 5G technologies the world will be fully interconnected and those enable many challenging applications. With the mobile cloud computing (MCC), more personalized and interactive services will be available with resource-limited mobile terminals. Fifth-generation cellular networks aims to change the world by connecting anything to anything.

Mobile cloud computing emerging in the context of 5G has the potential to overcome resource limitation in the mobile devices (that appear as a bottleneck in 5G applications), which enables many resource-intensive services for mobile users with the support of mobile big data delivery and cloud-assisted computing [3]. With this considerations I write Part II of my thesis on live migration over wireless 5G context application. New generation of wireless networks will combine very interactive and more responsive applications with powerful computing and high storage capacity data centers. For this features it is essential to have computation and storage capacity at the edge of the network.

Amazon's history is the emblem of *cloud computing* evolution. The company understand that their internal data center infrastructures was overestimated for the most part of the year, they had very powerful and high capable storage and computing capacity but low data center's throughput utilization during the rest of the year. Infrastructure's network was not capable to fit the changing of requested resources, then the next step was the evolution in an architecture with the capability to sell unused resources and create the first world cloud company, today know as Amazon Web Services (AWS).

1.1 Data Center Virtualization

Virtualization in data centers have received significant attention as a cost-effective solution in infrastructure for storing large amount of data and housing large-scale

service applications. Big companies like Amazon, Google, Yahoo! and Facebook are using data centers for their core services [4, 5, 6] or to sell their resources to external customers.

The architectures data center are far from being ideal, data centers use dedicate servers to run applications, resulting in poor server utilization and high operational cost. The situation improved with the emergence of server virtualization technologies (e.g.: VMware, Xen, KVM, etc.). Use of virtualization technologies allow multiple virtual machines (VMs) to be co-located in the same physical server, furthermore, these technologies improve performance isolation between co-located VMs, improving performance and preventing interference attacks.

As in server virtualization also network virtualization aims at creating virtual networks (VNs) on top of a shared physical network substrate allowing each virtual network. The separation between logical networks and physical networks permit to introduce customized network protocols and management policies.

I briefly introduce some concept and a short overview of recent literatures on virtualization data center networks and architectures. In the work [7] Bari et al. their contribution are three fold: (i) first, they provide a summary of the recent work on data center network virtualization; (ii) second, they compared these architecture and highlight their design trade-offs; (iii) third, they pointed out the key future research directions for data center network virtualization. For the growing interest on energy consumption many paper was written in the last years about green data center networks (DCNs), in [8] Bila et al. present a taxonomy survey on DCNs, with an overview of the research and the recent state-of-art energy efficiency techniques DCNs.

1.2 Data center architectures

This section provide an overview of data center architecture, I won't be exhaustive about that argument, essentially I am looking on an overview about milestone on data center architectures.

Looking at conventional data center network topology [7] as in Fig. 1.1 I can distinguish different layers. The Top-of-Rack (ToR) switch in the access layer that provides connectivity to the servers mounted on every rack. The aggregation switch (AS) in the aggregation layer (or distribution layer) forwards traffic from multiple access layer (ToR) switches to the core layer. Core layer provides secure connectivity between aggregation switches and core routers (CR) connected to the external network, the Internet.

Traditional data center networks which organize switches in a simple tree topology

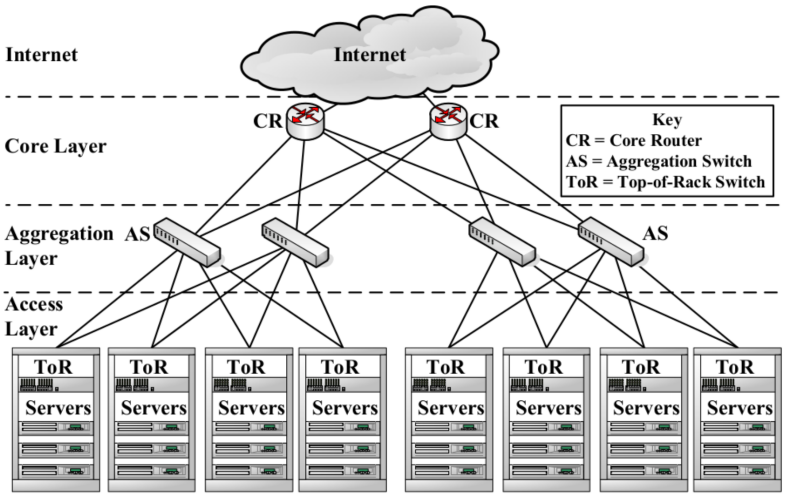


Figure 1.1. Conventional data center network topology. From [7]

fall under this category as proposals such as VL2 [9] or more complicated topology such as fat trees [10]. A second category is called *hybrid architectures* [11], include designs in which packets are forwarded using a combination of switches and servers as in solutions such as BCube [12] and DCell [13]. Third, we can consider *server-only data center* architectures which do not rely on switches for packet forwarding, each server plays two role, running regular applications and also relaying traffic between servers. Every server is directly connected to a few other servers to form a data center-wide interconnected in a three dimensional torus topology [11].

A Conventional topology is *flat layer 2 topology*, which uses only layer 2 switches. *Clos* is a topologies family built up from multiple stages of switches, in which each ones in a stage is connected to all switches in the next stage, which provides extensive path diversity.

A special type of *Clos* is the *Fat-tree* topology that is organized in a tree-like structure, as shown in Fig. 1.2. This topology, introduced by Leiserson in [10], have the properties that make it suitable for data center networks. It is built of k-port switches contains k *pods*; each *pod* has two layers (*aggregation* and *edge*) of $k/2$ switches. It contains $(k/2)^2$ *core* switches, each ones of them has one poret connected to each of k pods. The ith port of any core switch is connected to pod i so that consecutive ports in the aggregation layer of each pod switch are connected to core switches on $k/2$ strides. Each edge switch is directly connected to $k/2$ end-host; each of the remaining $k/2$ ports of an edge switch is connected to $k/2$ of an aggregation switch [14, 7].

Microsoft researchers presented in the work [9] VL2 an evolution for data center network topology, based on Clos topology. The objective of VL2 is a practical

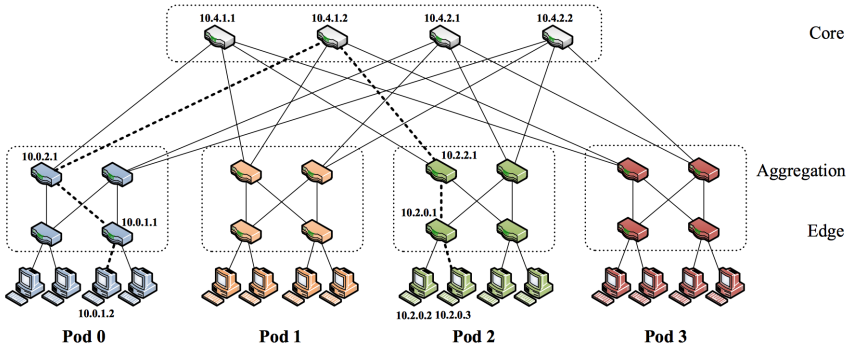


Figure 1.2. Simple *Fat-tree* topology. Using the two-level routing. From [14]

network architecture that scales to support huge data centers with uniform high capacity between servers and enable any server to be assigned to any service.

In [11] Popa et al. compared four different possible topologies for data centers: *Fat tree* switch-only networks, *de bruijn-based* server-only networks, *BCube* and *de Bruijn-based* hybrid networks.

An open problem in data centers networks is the routing of packets into the data center architecture considering the path between source and destination with load-balancing across the set of paths to make fall use of the available network capacity.

However, this is not the focus topic of my research.

In the next chapter I will illustrate the basic knowledge of my work, in particular what is the *live migration* of virtual machine in data centers and how does it works.

CHAPTER 2
BACKGROUND

> *"We are in the midst of a substantial change in the way computing services are provided. As a consumer, you surf the web on your cell phone, get directions from a GPS device, and stream movies and music from the cloud. At the heart of these services is virtualization — the ability to abstract a physical server into a virtual machine."*
>
> - Matthew Portnoy, *Virtualization Essentials[15]*

Virtualization is an emerging technique that allows running multiple operating systems (OSs) simultaneously on a single server. For this purpose, a special middleware layer, the virtual machine manager (VMM) or hypervisor, abstracts from physical computing/networking resources and provides the so-called virtual machines (VMs), which act like real networked computers with their own virtual resources [15].

In the last decade there was fundamental changes in the way computing services are provided. At the beginning was mainframe, and personal computer changes the rule of the game through *digitization* of the physical desktop, and client/server technology. The Internet, boom and bubble, spanned the last and current centuries and continue today. We are, today, in the midst of another of those model-changing trends: *virtualization*.

In modern virtualized networked data centers (VNetDCs), live migration allows to move a continuously running VM from one server to another, so to attain multiple goals, including failure tolerance and energy-saving through server consolidation/load balancing [16]. Although live migration is becoming a service primitive function for the resource management of VNetDCs, it may induce slowdown of the application run by the migrating VM, as well as not negligible increments of the networking traffic and the computing-plus-networking energy consumption. Several position papers point out that the transmission rates of Web servers reduce by 15% to 20%

and the energy consumption may increase up to 15% during live VM migration [16], [17], [18]. This means that, if not carefully optimized, live VM migration may waste more energy as being saved afterwards as shown by Xu et al. [16].

Therefore, since effective energy-saving techniques for computing servers have quickly evolved during the last ten years [19], energy consumption of the network devices supporting intra-data-center live VM migration has emerged as a substantial issue [16]. At this regard, the work in [20] shows that, in a typical VNetDC from Google, the migration-induced network energy consumption approaches 50% of the overall ICT consumption when the server utilization is around 15% of total, which is quite typical in production data centers. This means, in turn, that, assuming an average industrial electricity cost of $ 0.07 per kW-hour, minimizing the networking-induced energy consumption of medium/large VNetDCs equipped with more $32k$ servers may translate to a saving of about $ 3.8 M over a typical four-year service life of a VNetDC [20].

2.1 Reference scenario and tackled problem

Motivated by these techno-economic considerations, I focus on the optimal adapted and distributed management of the network bandwidth consumed by intra-data-center live VM migration. A (simplified) sketch of the reference VNetDC platform is reported in Fig. 2.1 (see, for example, Figs. 10.3 and 13.4 of [15]).

Under dynamic workload, VMs may experience "hot spots" (inadequate network/-computing/memory resources to fulfill the QoS requirements) and "cold spots"(over provisioned resources with low server utilizations). Migrating VMs in order to alleviate hot (resp., cold) spots through load balancing (resp., server consolidation) needs to address at run-time three basic questions, namely:

(i) *When* to migrate;

(ii) *Which* VMs to migrate and *where* to migrate them; and,

(iii) *How* to manage the bandwidths of the end-to-end connections which support the planned VM migration.

Task of the *Resource Profiler* of Fig. 2.1 is to periodically measure the resource requirements of the instantiated VMs and the spare resources of the active servers, in order to allow the *Hot/Cold-spot Detector* to individuate over/under-loaded servers and, then, trigger VM migrations. After receiving a migration signaling, the *Migration Planner* of Fig. 2.1 proceeds to individuate both the VMs to be migrated and the corresponding list of source-destination servers.

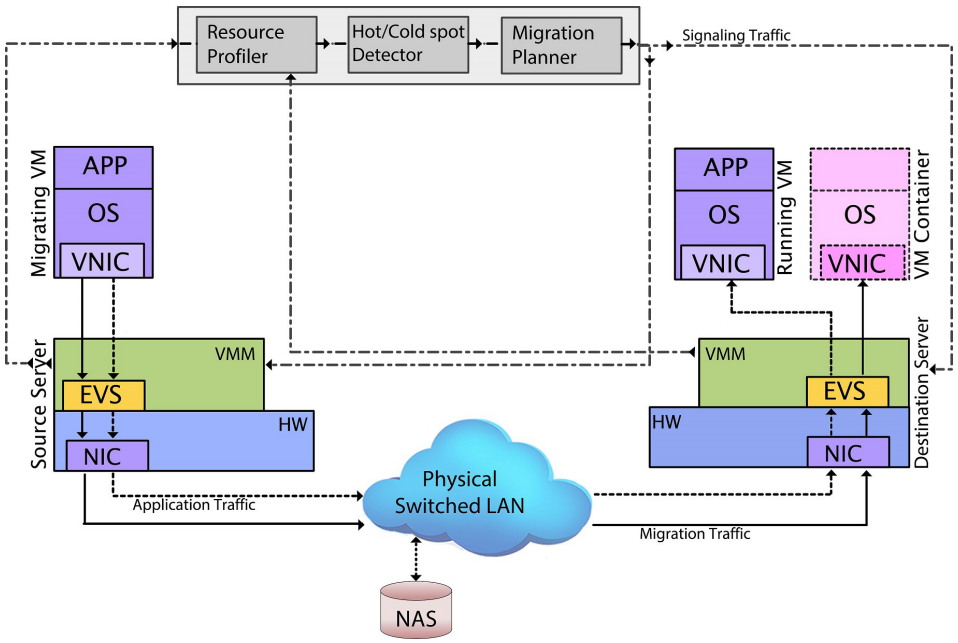


Figure 2.1. Reference VNetDC architecture for intra-data-center live VM migration. Continue (resp., dotted) arrowed lines denote end-to-end TCP connections conveying migration (resp., application) traffic. Dashed-dotted arrowed lines denote signalling flows for implementing the resource profiling and migration plan. APP: application; VMM: virtual machine manager; OS: operating system; HW: networking/computing hardware; EVS: external virtual switch; VNIC: virtual network interface card; NIC: physical network interface card; NAS: network-attached storage.

Determining a new mapping of VMs to physical servers that mitigates Hot/Cold spots while meeting the QoS requirements of the migrating applications is an NP-hard vector bin packing problem. In general, it is approximatively solved by selecting a suitable scalar [21],[22], [23] or vector [24] metric which adequately captures the migration cost and, then, by implementing a greedy-type heuristic (such as, for example, the Worst-Fit [22], Best-Fit Decreasing [23], First-Fit Decreasing [21] or modified Best-Fit Decreasing [24] heuristic) which provides a "good" approximate solution of the bin packing problem (see Chapter 6 of [19] and Chapter 14 of [25] for updated overviews on the overall topic of cost-effective constrained VM placement).

Finally, after receiving the list of the VMs to be migrated and the corresponding source-destination servers from the Migration Planner, task of the VMMs of Fig. 2.1 is to build up the corresponding end-to-end connections and manage the corresponding bandwidths. In current production VNetDCs, VM placement is actually performed by various capacity planning tools, such as VMware Capacity Planner, IBM WebSphere CloudBurst and Lanamark Suite [15], [19]. These tools seek to consolidate VMs for CPU, memory and computing power savings, yet without considering consumption of network resources.

Motivated by these considerations, the QoS energy-efficient dynamic and distributed management of the migration bandwidths is, indeed, the specific topic of this contribution. Our target is to minimize the communication energy wasted by live VM migration under five QoS hard constraints, which capture the typically considered SLA-induced performance metrics [16], [19]. Specifically, the first two constraints upper limit the total migration time and downtime (e.g., the service interruption time). In order to avoid (possible) migration-induced traffic congestion phenomena, the third constraint limits the maximum bandwidth available for the migration, while the fourth constraint upper bounds the maximum tolerated slowdown of the migrated application. Finally, the last constraint enforces the convergence of the overall migration process by lower limiting the corresponding speed-up factor, that is, the minimum ratio between the volumes of data migrated over two consecutive rounds.

2.2 Reference technology live migration: Xen Migration

Xen is an abstraction, built atop other abstractions, wrapped around other abstractions. In the virtualization Xen is the reference technology for academic world. In this section I explain general aspects about live migration, but keep in mind that the documentation is based mainly on Xen's papers and books. The goal of virtualization technologies are the target to uncouple the software from messy, noisy, fallible hardware. One of the benefit of them is to offer a sort of total hardware independence, but for this purpose another feature is essential: *migration*.

Migration, as already introduced, is a core function for virtualizzed data center to provide flexibility and reliability. Migration transfers the entire virtual machine (in-memory state of the kernel, all processes and all applications states) on a different physical server with a short downtime for the user's perspective. Migration may be *live* or *cold* (sometimes called also *hot* and *dead*, but less-commonly used), distinguishing them on the basis of whether the instance is running at the time of migration. In the *live* migration, the virtual machine is paused and downtime is kept minimum. In *cold* migration, the virtual machine is paused, saved and sent to another physical machine. In both cases the saved machine will expect receive its IP address and ARP cache to work in the new network. This is no obvious cause the in-memory state of network stack persist unchanged in both cases *live* and *cold* migration, the first method introduced in Xen to migrate a machine was the command *xmsave* and *xmrestore*. With this command the administrator can hibernate a machine, saving its entire memory image to disk and power off the

physical machine. Hence, using *restore* command to start again the machines after restarting the server.

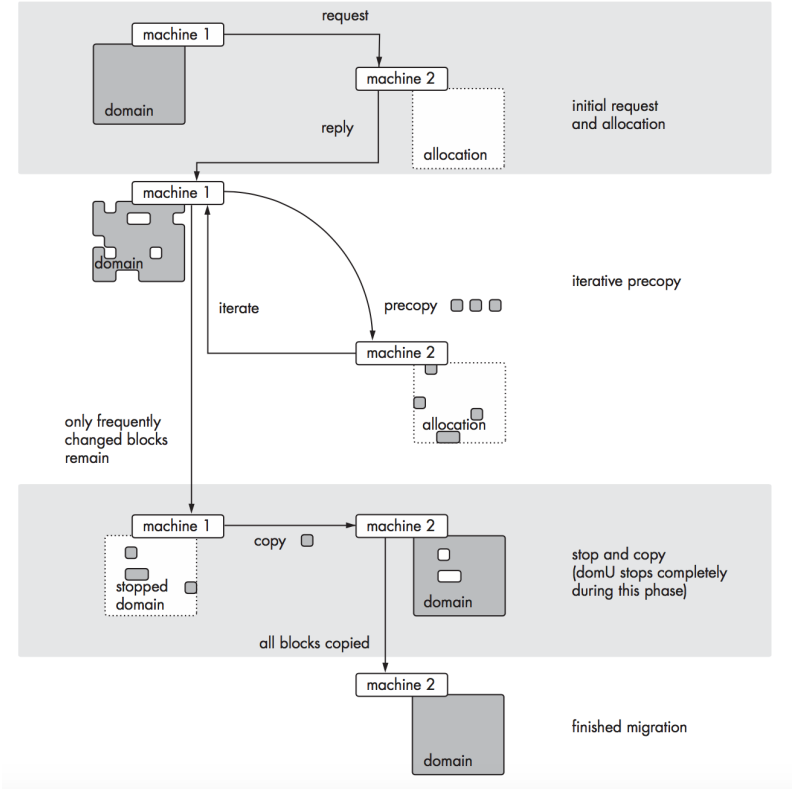


Figure 2.2. Overview of live migration process in Xen. From [26].

Live migration move a domain from a physical machine to another transparently, imperceptibly to the customer's world. Xen transfers the domain's configuration with the state of the machine and it does not require the administrator any manually operation. As you can see in Fig. 2.2, and in [26], it is not based only on the basic idea of save and restore the machine's state. Source machine does not hibernate until the very last phase of migration, and it is the time of out-of-service is very short. In the following Chapter 3 and Section 3.1 I explain how live migration works, with detailed descriptions of different techniques.

In Xen, see Fig. 2.2, live migration begin with request or *reservation*, to the target machine containing the necessary resources. If the target accepts the request, the source begins the iterative pre-copy phase of live migration. In this phase, Xen copies pages of memory over a TCP connection to the destination server. At the same time, pages that change are marked as *dirty* and then recopied iteratively in the next round of iterative pre-copy. The server iterates this until only frequently changed pages remain, then only few data need to be copy to the destination. Hence, it begins the stop-and-copy phase. Now Xen stops the VM and copies over any

pages remains. Xen will iterate up to 29 times and stop if the number of dirty pages falls below a certain threshold. Administrator can specify the maximum number of iterations and the dirty pages threshold at compile time. Finally the VM *starts* executing on the destination server [26].

CHAPTER 3
RELATED WORK

"We are like dwarfs perched on the shoulders of giants, and thus I are able to see more and farther than the latter. And this is not at all because of the acuteness of our sight or the stature of our body, but because I are carried aloft and elevated by the magnitude of the giants."

- Bernard of Chartres, *Riemen, prologue to Steiner 2006, p. 23*

During the last years, the problem of the VM placement has been largely investigated by both academy and industry (see, for example, the recent overviews in [16], and Chapter 14 of [25]) and several software environments have been recently developed for supporting the VM placement operation (see, for example, Chapter 6 of [19] for an updated overview). However, at the best of the authors' knowledge, the bandwidth management problem tackled by this thesis deals with a *still* unexplored research topic, as also confirmed by several recent position papers [16], [27], [28], [29].

In the following an overview on the related work to my thesis. Firstly, I review some recent advances on the basic techniques adopted for implementing intra-data-center live VM migration. Afterwards, I consider some contributions which specifically deal with the energy consumption on virtual machine migration.

Basically, live migration consists in the copy of all necessary data from physical server to the destination server, as sowed in the Fig. 2.1. The simplest way to provide this functionality is the *stop-and-copy technique*, that you can see in Fig. 3.3. Stop-and-copy entail that we must stop the VM on the source server, then start copying the entire memory, registry, etc. to the destination server, hence, reactivate the VM in new location.

The first research line mainly focuses on the volume of data migrated by three basic live migration techniques, namely,

- the *pre-copy*,
- *post-copy*
- and *hybrid* techniques.

Specifically, *pre-copy based VM migration* (see Fig. 3.2) has been firstly investigated in [30]. This work proposes a *heuristic* bandwidth manager for the Xen hypervisor, which dynamically increases the migration bandwidth over consecutive migration rounds. The goal is to reduce the downtime due to the memory dirty rate of the migrating VM, while increasing the utilization of the overall available bandwidth. This bandwidth management policy is the *de facto* state-of-the-art one and it is currently implemented by a number of commercial VMMs, such as, Xen, VMware and KVM hypervisors [19]. However, this policy:

(i) does not minimize the communication energy wasted by live VM migration; and,

(ii) does not enforce any constraint on the total migration time and/or downtime.

Subsequent works on the *pre-copy based live VM migration* leveraged [30] and aimed at reducing the migrated data by performing data compression [31], [32], [33], and/or by exploiting the memory change probabilities of the running applications [34]. Specifically, in [31], the incremental changes in the dirtied pages are computed and sequentially migrated, after performing run-length encoding. The authors of [32] propose a check-pointing mechanism, in order to trace the execution of the VM at the source server. The cloned VM on the destination server is synchronized with the VM on the source server by iteratively transferring suitable log messages. Since the total size of the log files is (substantially) less than of the dirtied pages, a reduction of the volume of the migrated data is attained. The work in [33] proposes to reduce the total migrated data by using hash-based fingerprints to find similar memory pages. The approach pursued in [34] for reducing the volume of the migrated data relies on the fact that the pre-copy technique migrates memory pages over consecutive rounds and the data migrated at each round are those which have been dirtied during the previous round. Hence, in order to avoid multiple migrations of a same memory page, the scheduler proposed in [34] ranks the memory pages for increasing values of their dirty rates and, then, migrates the pages with the highest dirty rates at the end of the migration process (that is, during the last round). Overall, these contributions:

(i) do not consider the energy aspects of the migration process; and,

(ii) do not enforce QoS constraints on the performance metrics.

Post-copy techniques (see Fig. 3.3) for live VM migration are proposed in [35], [36] and [37]. At this regard, [35] proposes a dynamic combined utilization of self-ballooning and pre-fetching techniques, in order to reduce as much as possible the migrated data, as well as the occurrence of network faults. The authors of [36] introduce the concept of VM fork (e.g., a form of process forking), in order to quickly instantiate VM clones on-the-fly. Similarly, [37] develops a post-copy based framework for the (quasi) instantaneous consolidation of VMs which process fast time-varying workloads.

Lastly, *hybrid live migration* (see Fig. 3.4) techniques are presented in [38] and [39], which attempt to combine the positive aspects of the post-copy and pre-copy paradigms. Specifically, [38] proposes a pre-paging technique, which speculates on the memory locations of the application run by the migrating VM. The authors of [39] present a technique that suitable combines delta compression and run-length coding, while migrating the data during the post-copy phase. Overall, the papers in [35]-[39] do not afford the bandwidth management problem and/or the related energy aspects. They subsume, indeed, that the migration bandwidth is assigned at run-time by the VMM on a best effort basis, that is, by accounting for the currently available residual bandwidth.

The (aforementioned) second research direction focuses on the modeling analysis and measurement of the CPU and network energies consumed by pre-copy based live VM migration techniques [18], [28], [40]. Specifically, [18] and [28] present quantitative approaches for modeling the time performance and energy consumption of the migration processes, which exploit the dynamic bandwidth management proposed in [30].

In [21], the authors investigate the optimized VM placement in VNetDCs by considering both the migration and server-induced costs. However, the migration cost considered in [21] accounts only for the slow-down of the migrating application. The work in [22] proposes both gray and black box approaches for performing load balancing through live VM migration. However, this work focuses on VM placement strategies and does not consider the migration costs. The authors of [40] develop a framework for the minimum-energy consolidation of VMs in VNetDCs. Although the approach in [40] exploits live migration for moving the VMs to be consolidated, it does not consider the migration-induced performance penalty and energy consumption. Overall, all the works in [18], [28] and [40] rely on the state-of-the-art bandwidth management policy developed in [30] and do not attempt to optimize it.

3.1 Current approach to live migration of VMs

This section provides a short review on the main aspects of live VM migration which are directly involved by the management of the migration bandwidth. Updated overviews on the general topic of live VM migration are provided, for example, in [16], Chapter 3 of [19] and Chapter 17 of [25].

Live VM migration allows a running VM to be transferred from a physical source server to a destination one by exploiting the end-to-end (typically, TCP/IP-based) connection built up atop the underlying intra-data-center LAN [19]. Since current VNetDCs are equipped with network-attached storage (NAS) which is uniformly accessible from the servers (see Fig. 2.1), intra-data-center VM migration reduces to copy the in-memory state and the CPU registers of the migrated VM.

There are four main techniques for VM migration, namely, stop-and-copy migration (SaCM), pre-copy migration (PeCM), post-copy migration (PoCM) and hybrid migration (HyBM). They trade-off the total migration time and downtime. These techniques rely on the implementation of at least one of the following three phases [30]:

1. *Push phase:* the source server transfers to the destination server the memory image (e.g., the RAM content) of the migrating VM over consecutive rounds. To ensure consistency, the memory pages modified (e.g., dirtied) during this phase are re-sent over multiple rounds;

2. *Stop-and-Copy phase:* the VM at the source server is halted and lastly modified memory pages and device states are transferred to the destination server;

3. *Pull phase:* the migrated VM begins to run on the destination server. The access to memory pages still residing on the source server is accomplished by issuing page-fault interrupts.

Hence, the *techniques* available for live migration are four:

- stop-and-copy migration (SaCM),
- pre-copy migration (PeCM),
- post-copy migration (PoCM),
- hybrid copy migration (HyBM).

SaCM technique utilizes only the Stop-and-Copy phase. This guarantees that the volume of the migrated data equates the memory size of the migrated VM, but it generally induces long downtimes [17].

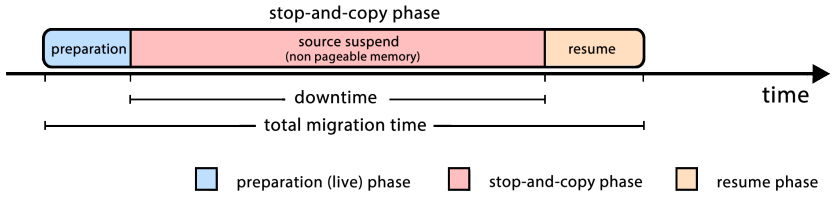


Figure 3.1. Time-line for stop-and-copy live migration.

PeCM technique implements the push and stop-and-copy phases of the migration process. It guarantees finite migration times, tolerable downtimes and robustness against the (possible) failures of the destination server (see Section 4.3). However, it induces overhead in the total volume of the migrated data, which may be substantial under write-intensive applications [17].

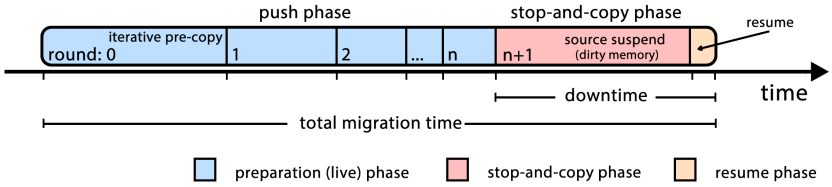


Figure 3.2. Time-line for pre-copy based live migration.

PoCM technique is composed by the stop-and-copy and pull phases of the migration process. Since only the I/O and CPU device states are transferred to the destination server during the stop-and-copy phase, the experienced downtime is limited and no data overhead is induced. However, the resulting total migration time is, in principle, undefined and, due to the page-fault interrupts issued during the pull phase, the slowdown experienced by the migrated application may be substantial.

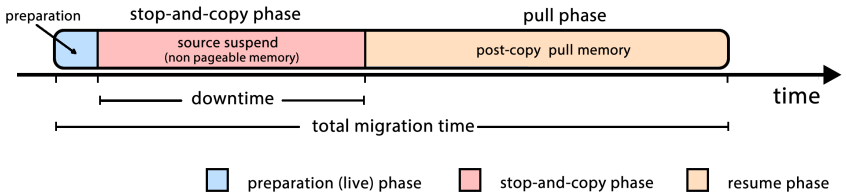


Figure 3.3. Time-line for post-copy based live migration.

Then HyBM technique incorporates all three phases of the migration process, in order to trade-off the (aforementioned) pros and cons of the PeCM and PoCM techniques [17]. For this purpose, the read-only (resp., write-intensive) memory pages of the migrating VM are transferred during the push (resp., pull) phase, while the content of the I/O and CPU registers are quickly migrated during the stop-and-copy phase. So doing, the total migration time of the HyBM technique is lower (resp.,

larger) than the corresponding one of the PoCM (resp., PeCM) technique, while the opposite conclusion holds for the resulting downtime [17].

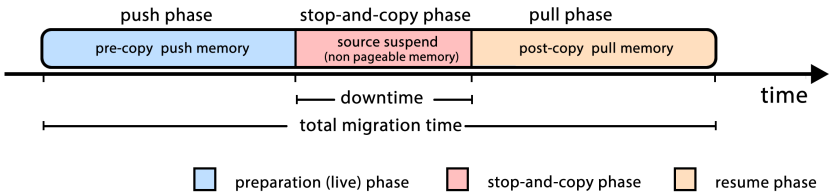


Figure 3.4. Time-line for hybrid live migration.

Approaches	Objective	Optimization	Complexity	Techniques	Effectiveness
MECOM	Live migration of a VM	Migration time, application downtime, network traffic	Medium	Compression	Migration time reduction: 32%
Koto *et al.* Chiang *et al.*			Low	Filter un-useful memory pages	Migration time reduction: 34 – 68.3%
Svard *et al.*			Medium	Compression, page delta transfer	Migration time reduction: 40%
CR/RT -Motion			Medium	Transfer and replay of execution logs	Migration time reduction: 31.5%
Jo *et al.*			Low	Transfer storage locations of memory	Migration time reduction: over 30%
PMigrate		Migration time, application downtime	Low	Parallelizing migration	Migration speedup factor: 2.49 – 9.88
Liu *et al.*		Migration performance, energy	Medium, $\mathcal{O}(n)$	Migrate the VM with minimum cost	Migration cost reduction: 72.9%
Breitgand *et al.*		Performance of migration and the migrating VM	Low	Migration bandwidth allocation	minimize SLO violations
CloudScale	Resource conflict handling	Performance of co-located VMs	Low, $\mathcal{O}(m{\cdot}n)$	Migrate the VM with minimum cost on VM performance	SLO violation reduction: 83%
iAware	Load balancing, power saving	Performance of VMs on source and destination	Low, $\mathcal{O}(m{\cdot}n)$		Performance improved: 16 – 65%
Deshpande *et al.*	Concurrent migration of multiple VMs	Concurrent migration time, network traffic	Medium	De-duplication	Network traffic reduction: 75%
VDN	Concurrent deployment and snapshotting of multiple VMs	Startup delay of multiple VMs	High	De-duplication, chunking	Speedup factor: 30 – 80
Razavi *et al.*		Scalability, VM booting delay	Low	VM image caching	Speedup factor: 8
Nicolae *et al.*		Performance of VM snapshotting and deployment	Medium	Lazy VM deployment, store incremental update	Speedup factor: 2 – 25, bandwidth reduction: 90%
VMTorrent		Scalability, VM booting delay	Medium	VM image profile, block pre-fetching	Speedup factor: 30

Figure 3.5. Comparison of approach to manage VM performance overhead caused live migration, deployment, and snapshotting of multiple VMs with a single data center. From [16]

I anticipate that the optimal bandwidth manager developed in this thesis may be applied under *all* the mentioned migration techniques. However, in order to speed up its presentation, in the sequel I focus on the PeCM as reference technique. The main reasons behind this choice are that:

(i) PeCM is the migration technique currently implemented by a number of commercial VMMs, such as, Xen, VMware and KVM [15], [19]; and,

(ii) the bandwidth management framework provided by the PeCM technique is general enough to embrace those featured by the SaCM, PoCM and HyBM techniques.

In the schema in Fig. 3.5 from [16] the authors provided a survey of different approaches to manage VM performance overhead caused live migration, deployment, and snapshotting of multiple VMs with a single data center.

3.2 Pre-copy live migration (PeCM)

The PeCM technique involves six stages [30], as you can see in the Fig. 3.6, namely:

1. *Pre-migration:* resource profiling, detection of the (possibly present) hot/cold spots and planning of the VMs to be migrated are performed. At the end of this stage, the Migration Planner communicates to the VMMs the VMs to be migrated and the selected destination servers (see the dashed-dotted signaling paths of Fig. 2.1). This stage spans T_{PM} seconds, e.g, T_{PM} (s), where s means seconds;

2. *Reservation:* the computing/communication/storage/memory physical resources are reserved at the destination server by instantiating a large enough VM container (see the dotted box of Fig. 2.1). T_{RE} (s) is the duration (in seconds) of this stage;

3. *Iterative pre-copy:* this stage is composed by $(I_{MAX} + 1)$ rounds and spans T_{IP} seconds. During the initial round (e.g., at $round\#0$), the entire memory content of the migrating VM is sent to the destination server. During the subsequent I_{MAX} rounds (e.g., from $round\#1$ to $round\#I_{MAX}$), the memory pages modified during the previous round are re-transferred to the destination server (see Fig. 3.7);

4. *Stop-and-copy:* the migrating VM is halted and a final memory-copy round (e.g., $round\#(I_{MAX} + 1)$) is performed (see Fig. 3.7). This last round spans T_{SC} seconds;

5. *Commitment:* the destination server notifies that it has received successfully a consistent copy of the migrated VM. T_{CM} (s) is the duration of this stage;

6. *Re-activation:* the I/O resources and IP address are re-attached to the migrated VM on the destination server. T_{AT} (s) is the needed time.

Before proceeding, I remark that a task of the *Pre-migration* phase is to guarantee that the designed migration plan is stable, i.e., it is not affected by ping-pong phenomena. I anticipate that, for this purpose, the approach developed in *Section 4* of [22] has been implemented in the carried out tests. Specifically, as in [22], the

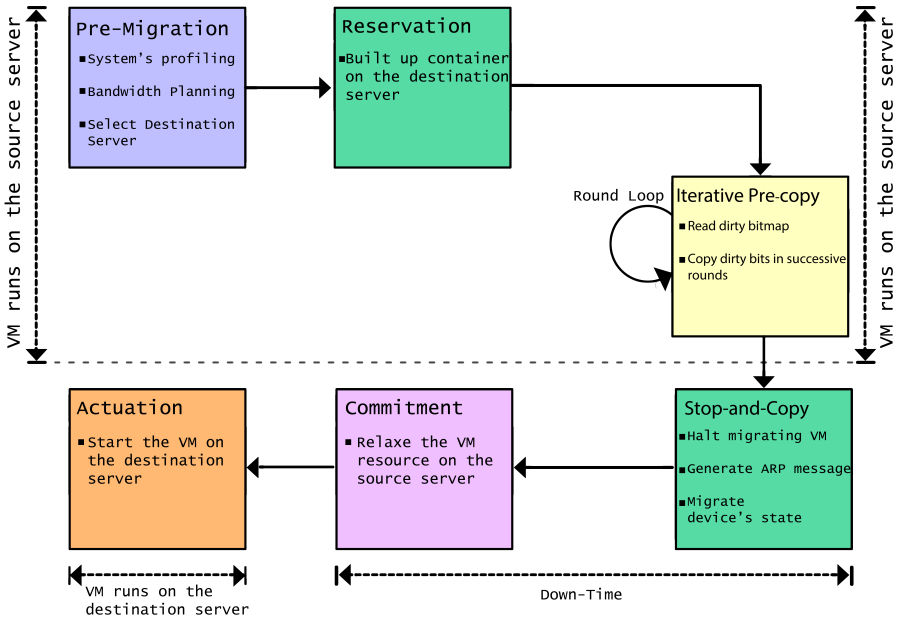


Figure 3.6. Pre-copy live migration stages (six stages).

Hot/Cold-spot Detector of Fig.2.1 periodically measures the average aggregate utilization of the computing-plus-bandwidth resources of the currently running physical servers. To ensure that small transient fluctuations of the measured utilization do not trigger needless migrations, a migration request is signaled by the *Detector* of Fig. 2.1 only when at least k out the m most recent measurements together with the next predicted value of the servers' utilization fall out a target (e.g., desired) utilization interval U (see *Eq. (1)* of [22] and the related text). It has been experienced that this approach is capable to effectively filter out transient fluctuations of the average resource utilization and avoid needless migrations [22]. As in *Section 7* of [22], I anticipate that, in the carried out tests, I posed: $k = 3$, $m = 5$ and $U = [0.4, 0.75]$, while the measuring period has been set to 10 seconds.

3.3 Migration times and network energy

From a formal point of view, the total migration time T_{TOT} (s) is the overall duration:

$$T_{TOT} \triangleq T_{PM} + T_{RE} + T_{IP} + T_{SC} + T_{CM} + T_{AT}, \tag{3.1}$$

of the (aforementioned) six stages, while the downtime:

$$T_{DT} \triangleq T_{SC} + T_{CM} + T_{AT}, \tag{3.2}$$

is the time required for the execution of the last three stages. From a practical point of view, T_{TOT} in (3.1) is the period when the states of the source and destination servers must be synchronized which may also affect the reliability of the migration process (see Section 4.3 in the sequel), while T_{DT} in (3.2) is the period in which the migrating VM is halted and the clients experience a service outage [16].

Let R (Mb/s) be the transmission rate used during the third and fourth stages for migrating the VM, that is, the migration bandwidth. Since, by definition, only T_{IP} and T_{SC} depend on R, while all the remaining migration times in Eqs. (3.1) and

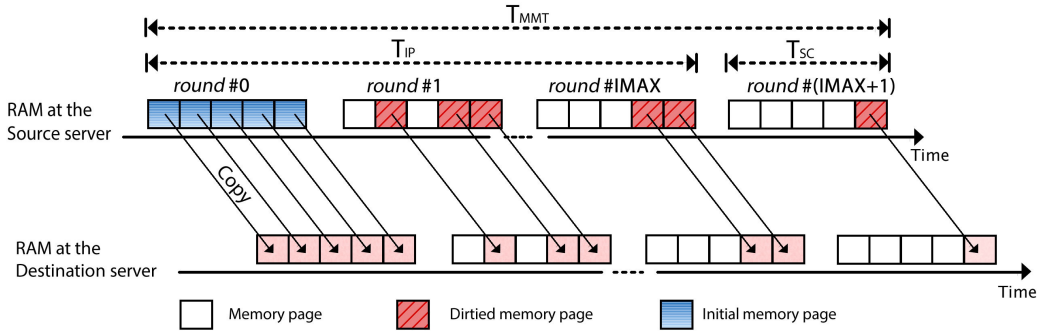


Figure 3.7. Time-chart of the PeCM technique.

(3.2) play the role of constant parameters, in the sequel, I focus on the evaluation of the (already defined) stop-and-copy time T_{SC} and the resulting memory migration time T_{MMT}, which is defined as in:

$$T_{MMT} \equiv T_{MMT}(R) \triangleq T_{IP}(R) + T_{SC}(R). \tag{3.3}$$

Hence, T_{MMT} is the time needed for completing the memory transferring of the migrating VM, e.g., the duration of the performed $(I_{MAX} + 2)$ memory-copy rounds of Fig. 3.7.

3.3.1 Modeling the bandwidth-dependent migration times

Table 3.1 reports the definitions of the key parameters used in the thesis. Since the PeCM technique performs the iterative pre-copy of dirtied memory bits over consecutive rounds (see Fig. 3.7), let V_i (Mb) and T_i (s), $i = 0, \ldots, (I_{MAX} + 1)$, be the volume of the migrated data and the time duration of the ith round, respectively. By definition, V_0 and T_0 are the memory size M_0 (Mb) of the migrating VM and the time needed for migrating it during the $0th$ round, respectively (see the leftmost part of Fig. 3.7).

Hence, after indicating by $\overline{w}$ (Mb/s) the (average) memory dirty rate of the

Symbol	Meaning/Role
I_{MAX}	Number of migration pre−copy rounds
i	Round index, i=0,. . .,$(I_{MAX}+1)$
$\overline{w}(Mb/s)$	Memory dirty rate of the migrated VM
R (Mb/s)	Migration bandwidth
$P(R)$ (W)	Communication power at the migration bandwidth R
$\hat{R}$ (Mb/s)	Maximum available migration bandwidth
M_0 (Mb)	Memory size of the migrated VM
$\mathcal{E}_{TOT}$ (J)	Total consumed communication energy
Δ_{MMT} (s)	Maximum tolerated memory migration time
Δ_{SC} (s)	Maximum tolerated stop−and−copy time
β	Migration speed−up factor
n	Integer−valued iteration index

Table 3.1. Main taxonomy of the thesis.

migrating application (e.g., the average number of bits per−second which are modified by the application), directly from the reported definitions I have:

$$V_i \triangleq \overline{w}\, T_{i-1} = M_0(\overline{w}/R)^i, i = 1, \ldots, (I_{MAX}+1), \tag{3.4}$$

with $V_0 \equiv M_0$, and

$$T_i \triangleq V_i/R = (M_0/R)(\overline{w}/R)^i, i = 0, \ldots, (I_{MAX}+1), \tag{3.5}$$

so that I also have (see (3.3))

$$T_{MMT}(R) \equiv \sum_{i=0}^{I_{MAX}+1} T_i = (M_0/R)(\sum_{i=0}^{I_{MAX}+1} (\overline{w}/R)^i), \tag{3.6}$$

and (see Eq. (3.5))

$$T_{SC}(R) \equiv T_{I_{MAX}+1} = (M_0/R)(\overline{w}/R)^{I_{MAX}+1}. \tag{3.7}$$

3.3.2 Network energy consumption

The power (measured in *Watt* (W)) drawn by a physical network interface card (NIC) consists of a static (e.g., setup) portion and a dynamic portion [41, 42, 43]. The static portion P_{SETUP} (W) does not depend on the transmission rate R and accounts for the power needed for the setup of the server-to-server network connection. The dynamic portion: $P_{DYN} \equiv P_{DYN}(R)$ accounts for the additional rate-depending power consumed by both the transmit and receive physical NICs, when they work on

behalf of the migrating VM. In order to limit the implementation cost, current data-centers use off-the-shelf rack-mount physical servers interconnected by commodity Ethernet switches [44], [45]. Furthermore, they typically implement *TCPNewReno*-based protocol for performing congestion control and attaining server-to-server reliable communication [45, 46]. This means, in turn, that the dynamic power wasted by the transmit/receive NICs may be modeled as in [47, 41, 48]

$$P_{DYN}(R) = K_0(R)^{\alpha}, \tag{3.8}$$

where

$$K_0 \triangleq (1/g)(RTT/1.22\ MSS)^{\alpha}, ((W) \times (s/Mb)^{\alpha}). \tag{3.9}$$

In Eq. (3.9), RTT (s) is the average round-trip-time of the available end-to-end connection, $g((W)^{-1})$ is the coding gain-to-receive noise power ratio, $\alpha > 1$ is a (dimension-less) shaping factor and MSS (Mb) is the maximum size of the utilized TCP segments [45], [47, 41].

In order to utilize a single unified framework for modeling the energy performance of the (aforementioned) migration techniques, let

$$\theta \triangleq \begin{cases} 1, \text{ for } PeCM \\ 0, \text{ for } SaCM/PoCM, \end{cases} \tag{3.10}$$

be an auxiliary binary variable which, by definition, is unit-valued (resp., vanishes) under the PeCM (resp., SaCM and PoCM) technique[1]. Therefore, since the dynamic energy $\mathcal{E}_i$ (J) consumed during the ith round equates the product: $\mathcal{E}_i \equiv \mathcal{E}_i(R) \triangleq P_{DYN}(R)\ T_i,\ i = 0, \ldots, (I_{MAX} + 1)$, by summing these products over the round index, I obtain the following expression for the total communication energy $\mathcal{E}_{TOT}$ (J) consumed during the migration process:

$$\mathcal{E}_{TOT} \equiv \mathcal{E}_{TOT}(R) = \left\{ K_0 M_0 R^{\alpha-1} \left[1 + \theta \left(\sum_{i=1}^{I_{MAX}+1} (\overline{w}/R)^i \right) \right] \right\} + \mathcal{E}_{SETUP}, \tag{3.11}$$

with $\mathcal{E}_{SETUP}$ (J) which accounts for the static portion of $\mathcal{E}_{TOT}$. Likewise, by summing the expressions in (3.4) over the round index, I arrive at the following closed-form formula for the total volume V_{TOT} (Mb) of the migrated data:

$$V_{TOT} \triangleq \sum_{i=0}^{I_{MAX}+1} V_i = M_0[1 + \theta(\sum_{i=1}^{I_{MAX}+1} (\overline{w}/R)^i)]. \tag{3.12}$$

[1] According to the definition of Section 3.1, the HyBM technique utilizes $\theta = 1$ (resp., $\theta = 0$) during the push phase (resp., the stop-and-copy and pull phases).

Before proceeding, two main remarks are in order. First, the sum-expressions in (3.11), (3.12) hold for every value of the ratio $(\overline{w}/R)$. In the specific cases of $(\overline{w}/R) = 1$ and $(\overline{w}/R) \neq 1$, these summations may be calculated in closed-form as reported in (A.1) of the final Appendices. However, in order to speed up the presentation, in the sequel, I adopt the sum-expressions in (3.11), (3.12). Second, according to (3.10), the terms proportional to θ in (3.11) and (3.12) account for the energy and data overheads induced by the iterative pre-copy stage of Fig. 3.7. Hence, by definition, these terms vanish under the SaCM and PoCM techniques. However, since these techniques may induce large migration times and downtimes (see Section 3.1), the attainment of a balanced consumed energy-vs.-time performance trade-off is the reason for introducing the bandwidth manager optimization problem of the following Chapter 4.

Remark 1. *On the validity limits of the adopted network performance model*

Regarding the validity limits of the network performance model of Eqs. (3.8) and (3.9), three remarks are in order. First, since Eqs. (3.8) and (3.9) model the power-vs.-rate relationship of *end-to-end* (possibly, multi-hop) TCP connections, they hold regardless of the physical topology of the intra-data-center network. These formulas assume that the considered TCP connections work in the Congestion Avoidance state [45], [47], [49]. Since current intra-data-center networks adopt topologies which maximize the resulting bisection bandwidths (such as, for example, the fat tree topology; see [25]), this assumption is typically met in practical application scenarios (see, for example, the test results reported in [16], [25] and [45]). Second, actual (single-path multi-hop) end-to-end migration routes are selected by the IP-based Network layer of the implemented Internet protocol stack during the *Pre-migration* phase of Section 3.1. For this purpose, the (usual) shortest-path (e.g., minimum-cost) criterion is typically pursued [16], [27]. Third, the power consumption-vs.-delay performance of the selected *end-to-end* migration route is measured by the resulting values of the parameters g and RTT of Eq. (3.9). Specifically, since longer routes consume more power and induce larger end-to-end delays, I expect that the value assumed by g (resp., RTT) in Eq. (3.9) decreases (resp., increases) for increasing number of the hops of the designed route. In agreement with these considerations, I anticipate that the test scenarios of Section 4.9 with larger values of K_0 could be refer, indeed, to longer end-to-end routes.

Part I

Part 1: Bandwidth manager in intra-data-center networks

CHAPTER 4

MINIMUM ENERGY BANDWIDTH MANAGER IN INTRA-DATA-CENTER NETWORKS

"In these situations the combination of virtualization and migration significantly improves manageability."

- Clark et al., *Live Migration of Virtual Machines [30]*

In this chapter I illustrate the bandwidth management optimization problem (BMOP) for live virtual machine migration intra-data-center networks, the major contributions of my work may be summarized with the following points:

(i) by referring to intra-data-center live VM migration (see Fig. 2.1), I cast the contrasting targets of low communication energy consumption and limited migration time/downtime in the form of a suitable (non-convex) optimization problem, namely, the bandwidth management problem. The BMOP formulation is general enough to embrace all the (aforementioned) pre-copy, post-copy and hybrid live VM migration techniques. Furthermore, it may be applied to both in-bound and out-bound live VM migration over wired/wireless virtualized data centers;

(ii) I develop closed-form analytical conditions, which are necessary and sufficient for the feasibility of the BMOP. Interestingly enough, a suitable exploitation of these conditions leads to a closed-form analytical formula for the optimized setting of the number of pre-copy rounds. This is, indeed, still an open problem, even under the state-of-the-art bandwidth management policy of [30] (see, for example, [50] and references therein);

(iii) I develop an *adaptive* version of the resulting optimal bandwidth manager, which is capable to quickly track in a fully autonomic way the (possibly, unpredictable) time-variations of both memory dirty rate of the migrating application and congestion level of the used end-to-end network connection. The proposed bandwidth manager may be implemented in a *distributed* and *scalable* way, so that its per-migration implementation complexity is constant and does *not* depend on the (possibly large) size of the considered data center. Furthermore, it may be utilized in tandem with every heuristic adopted by the Migration Planner of Fig. 2.1 for the dynamic VM placement;

(iv) lastly, I present the results of extensive field trials carried out by prototyping in software the proposed bandwidth manager through a (partial) modification of the legacy Xen hypervisor [51]. The carried out field trials support three main conclusions. First, the proposed adaptive bandwidth manager quickly converges to the optimal migration bandwidth, typically within 5-6 iterations with a final accuracy around 6%-7%. Second, the *per-VM* computing resource required for running the proposed bandwidth manager is limited up to 1-1.5% of the CPU computing power, regardless of the size of the considered data center. Third, in all carried out trials, the energy reduction of the proposed bandwidth manager over the state-of-the-art one of [30] is over 40% and it approaches 66% under strict QoS constraints. The corresponding stretching of the execution times of the migrated applications remains limited up to 20%, even for write-intensive programs [52].

4.1 QoS bandwidth management optimization problem

Four constraints are considered in the formulation of the BMOP, which capture, in turn, the metrics currently adopted for measuring the performance of live migration techniques [16], [18].

The first two constraints upper limit the tolerated memory migration and stop-and-copy times of Eqs. (3.6) and (3.7) and they read as in:

$$\Psi_1(R) \triangleq \theta[(T_{MMT}(R)/\Delta_{MMT}) - 1] \leq 0, \tag{4.1}$$

and

$$\Psi_2(R) \triangleq [(T_{SC}(R)/\Delta_{SC}) - 1] \leq 0. \tag{4.2}$$

In Eqs. (4.1) and (4.2), Δ_{MMT} (s) and Δ_{SC} (s) are the tolerated maximum memory migration and stop-and-copy delays (see Eqs. (3.6) and (3.7)). Furthermore, the θ

parameter in (4.1) accounts for the fact that, by definition, the memory migration and stop-and-copy times coincide under the SaCM and PoCM techniques (see Eq. (3.3) at vanishing T_{IP}).

A further constraint arises from the consideration that, without any stop condition, the iterative pre-copy stage of the PeCM technique may run indefinitely (see Fig. 3.7). Although the stop conditions depend highly on the design of the considered VMM, in [50] it is pointed out that they should account for the following two thresholds: (i) the number of the performed rounds exceeds a pre-defined threshold I_{MAX}; and, (ii) the ratio (V_i/V_{i+1}) of the volumes of data migrated over two consecutive rounds falls below a predefined speed-factor $\beta > 1$. Hence, since the constraints in (4.1) and (4.2) and the energy function in (3.11) already account for I_{MAX}, an exploitation of (3.4) allows us to formulate the β-related constraint as in:

$$\Psi_3(R) \triangleq \theta[(\beta\overline{w}/R) - 1] \leq 0. \tag{4.3}$$

By definition, this constraint is present only when the PeCM technique is considered and this motivates the presence of the θ parameter (see Eq. (3.10)).

The last constraint accounts for the maximum bandwidth assigned to the migration process and it is fixed in order to avoid (or, at least, mitigate) migration-induced traffic congestion phenomena [16]. In principle, depending on the considered VMM, the available bandwidth may be exclusively dedicated to the migration process (e.g., out-band migration) or may be shared with the application running on the migrating VM (e.g., in-band migration) [49]. In the first case, the constraint reads as in: $R \leq R_{MAX}$, where R_{MAX} (Mb/s) is the maximum bandwidth exclusively dedicated to the migration process. In the second case, a total bandwidth R_{TOT} (Mb/s) is assigned to the migrating VM and shared with the hosted application (see the continue and dotted arrowed paths of Fig. 2.1). Hence, in order to upper limit the slowdown (e.g., the stretching of the execution time) suffered by the migrating application, as detailed in Section 4.2.1, I limit the fraction of the overall available bandwidth R_{TOT} used by the migration process up to $\rho_{MAX} \in (0, 1)$, e.g., I enforce the constraint: $R \leq \rho_{MAX} R_{TOT}$. Hence, after introducing the dummy variable:

$$\widehat{R} \triangleq \{R_{MAX};\ \rho_{MAX} R_{TOT}\}, \tag{4.4}$$

the following constraint:

$$(R/\widehat{R}) - 1 \leq 0, \tag{4.5}$$

applies both cases of in/out-band migration, provided that I pose $R_{MAX} = \infty$ (resp., $\rho_{MAX} R_{TOT} = \infty$) under the in-band (resp., out-band) migration.

Overall, the considered QoS BMOP is formally defined as in:

$$\min_{R\geq 0} \ \mathcal{E}_{TOT}(R), \tag{4.6}$$

$$\textit{s.t.: constraints in } (4.1),\ (4.2),\ (4.3) \textit{ and } (4.5)\ . \tag{4.7}$$

4.2 Generalization of the problem

The reported formulation of the BMOP may be directly generalized along three main directions of potential interest.

First, in the case in which the application run by the migrating VM presents varied traffics over the time, the resulting dirty rate may change during the migration process. In order to deal with this case, let $\overline{w}_i$ (Mb/s) be the dirty rate during the ith round of Fig. 3.7, and let us introduce the following two dummy positions:

$$\overline{w}_{MAX} \triangleq \max_{1\leq i\leq I_{MAX}+1}\{\overline{w}_i\}, \tag{4.8}$$

and

$$C_i \triangleq \begin{cases} 1, \text{ for } i=0, \\ \prod_{m=1}^{i}\ \overline{w}_m, \text{ for } i\geq 1. \end{cases} \tag{4.9}$$

Hence, after replacing: (i) $(\overline{w}/R)^i$ by $C_i/(R)^i$ into Eqs. (3.6) and (3.11); (ii) $(\overline{w}/R)^{I_{MAX}+1}$ by $C_{I_{MAX}+1}/(R)^{I_{MAX}+1}$ into Eq. (3.7); and, (iii) $\overline{w}$ by $\overline{w}_{MAX}$ into (4.3), the problem formulation in (4.6), (4.7) still applies verbatim. At this regard, I anticipate that, since $\overline{w}_{MAX}$ in (4.8) and C_i in (4.9) play the role of (positive) constants, the solving approach of Section 4.7 directly generalizes to the case of time-varying dirty rate. The adaptive capacity of the resulting bandwidth manager to (possibly, unpredictable) variations of the dirty rates is tested in Section 4.9.3.

Second, in order to reduce the size of the migrated VM and/or protect the migrated data, compression coding (such as, for example, run-length, delta or ballooning-based coding) and/or error-protection coding (such as, for example, FEC coding and ARQ mechanisms) may be applied to the memory image of the migrating VM during the Pre-migration stage [16], [18], [19]. Hence, after indicating by S (bit) the size of the uncompressed VM, the size M_0 of the resulting compressed and/or coded version to be actually migrated may be expressed as in: $M_0 = (cop)(red)S$, where $0 \leq cop \leq 1$ is the utilized compression ratio, and $red \geq 1$ is the inverse of the adopted coding rate (e.g., $red = 1$ for not coded migration). Hence, without loss of generality, in the sequel, I directly consider M_0 as the actual size of the migrated VM.

Third, from the outset it follows that the BMOP applies, by design, to the PecM,

SaCM and PoCM techniques, by posing, respectively:

$$\theta = 1,\ I_{MAX} \geq 1, \qquad \text{(PeCM)} \tag{4.10}$$

$$\theta = 0,\ I_{MAX} = -1,\ \rho_{MAX} R_{TOT} = \infty, \quad \text{(SaCM)} \tag{4.11}$$

and

$$\theta = 0,\ I_{MAX} = -1, \qquad \text{(PoCM)} \tag{4.12}$$

in Eqs. (3.11)-(4.4). The application of the BMOP to the HyBM technique follows the guidelines of the footnote 1.

4.2.1 Limiting the tolerated migration-induced slowdown

Let $\overline{T}_{EXE}^{MIG}$ (resp., $\overline{T}_{EXE}$) be the average execution time of the application run by the VM in the presence (resp., absence) of migration. After modeling the migrating VM as a M/M/1 queue with First-In First-Out (FIFO) service discipline, the analysis carried out in [49] leads to the conclusion that the migration-induced slowdown: $SD^{MIG} \triangleq \overline{T}_{EXE}^{MIG} / \overline{T}_{EXE}$ equates:

$$SD^{MIG} = \frac{1}{1 - \rho_{MIG}}, \tag{4.13}$$

where $\rho_{MIG} \in [0, 1]$ is the utilization factor of the NICs of Fig. 2.1 by the migrated data (see the dotted lines of Fig. 2.1). The same conclusion holds when the M/G/1 queue model with Processor Sharing (PS) service disciple is adopted[1]. Furthermore, since the average queue delay of the (more general) G/G/1/FIFO queue systems still scales up as: $(1 - \rho_{MIG})^{-1}$ at medium/large values of ρ_{MIG} (I say, for $\rho_{MIG} > 0.1$; see, for example, Chapter 24 of [44]), I conclude that (4.13) still captures the asymptotic behavior of the slowdown of G/G/1/FIFO queues, while it is somewhat conservative (i.e., it acts as an upper bound) at low values of ρ_{MIG} (I say, at $\rho_{MIG} \leq 0.1$).

Overall, the net conclusion is that imposing an upper bound ρ_{MAX} on ρ_{MIG} in (4.13) leads to fix an upper bound on the tolerated application slowdown, regardless of the queue model adopted for the migrating VM. This is the reason, indeed, for including the constraint in (4.5) into the formulation of the tackled QoS migration problem. Actually measured values of slowdown for a spectrum of test applications are reported in Section 4.9.6.

[1] This is a direct consequence of the fact that the M/M/1/FIFO and the M/G/1/PS queue systems share the same formula for the corresponding average queue delays (see, for example, Chapter 22 of [44]). At this regard, I point out that the credit-based service discipline implemented by the state-of-the-art Xen scheduler is, indeed, an instance of the PS service discipline (see [51]).

4.3 Migration failure-vs.- memory migration time

Due to the (possible) failure of the involved servers, VM migration may halt in the middle of memory transfer and this leads to energy wasting [53]. Motivated by this consideration, in this section, I point out how the constraint in (4.1) on the allowed memory migration time limits the maximum tolerated migration-failure probability and the corresponding wasted energy. Towards this end, let $Pr_F(\Delta_{MMT})$ be the (possibly, profiled) server failure probability (SFP) over a time-window of Δ_{MMT} seconds. The actual behavior of this probability is application-dependent and it may be influenced by specific working metrics, such as, for example, server age, server utilization, volume of the performed I/O operations and maintenance level [53]. However, due to the decreasing reliability of the hardware components with the age, it is reasonable to expect that the SFP increases (or, at least, does not decrease) for increasing values of Δ_{MMT}. This is confirmed by the analysis of [54], which leads to the following (quite general) power-like parametric model for the SFP:

$$Pr_F(\Delta_{MMT}) = \begin{cases} (1+\omega)(\frac{\Delta_{MMT}}{T_F}) - \omega(\frac{\Delta_{MMT}}{T_F})^3, \text{ for } \Delta_{MMT} \leq T_F, \\ \\ 1, \quad \text{for } \Delta_{MMT} > T_F \end{cases} \tag{4.14}$$

In (4.14), $\omega \in [0, 0.5]$ is a dimension-less (possibly, profiled) shaping factor which fixes the heavy-tail behavior of the SFP, while T_F (s) is the (server age-dependent) maximum expected inter-failure time. Hence, fixing a maximum value: Δ_{MMT} on the tolerated memory migration time is equivalent to upper-bound the corresponding SFP. This leads to two main conclusions. First, the constraint in (4.1) on the allowed memory migration time plays the role of a reliability constraint. Second, under the (somewhat conservative) assumption that all the already transferred data are lost when a migration failure happens, the corresponding average energy wasted over a time-window of Δ_{MMT} seconds reads as in:

$$\overline{N}\ Pr_F(\Delta_{MMT})\ \overline{\mathcal{E}}_{TOT}. \tag{4.15}$$

In (4.15), $\overline{N}$ is the (possibly, profiled) average total number of VM migrations which are attempted over Δ_{MMT} seconds, while $\overline{\mathcal{E}}_{TOT} \triangleq E\{\mathcal{E}_{TOT}\}$ is the (possibly, profiled) per-migration average consumed energy.

4.4 VM migration-vs.-VM replication

VM migration may be also employed to attain high availability (HA) by providing fault-tolerance and/or supporting on-line server maintenance [53]. In principle, HA may be achieved either by live migration or whole-system replication (WSR) of the VMs [32], [55]. Goal of this section is to give insight about the communication-vs.-computing energy trade-off dictated by the migration-vs.-replication dichotomy. Towards this end, I (shortly) point out that WSR is a HA-oriented technique which simultaneously runs in parallel a same VM on (at least) two different physical servers. By leveraging the deterministic replay of the input commands and the periodic exchange of synchronization interrupts, WSR guarantees that the primary and replicated servers perform the same sequence of state transitions and, then, produce the same output sequence [55]. In order to evaluate the overall energy: $\mathcal{E}_{WSR}$ (J) required by the replication of the same VM on the source and destination servers, let L_{TOT} (Mb) be the overall workload offered by the VM and let ps_S (resp., ps_D) be the average processing speed (in (Mb/s)) at which this workload is processed by the source (resp., destination) server[2]. Furthermore, let P_S (resp., P_D) the average computing power consumed by running the VM on the source (resp., destination) server, and let SD_S^{WSR} (resp., SD_D^{WSR}) be the replication-induced slowdown at the source (resp., destination) server[3]. Hence, the overall average energy consumed by the WSR equates:

$$\mathcal{E}_{WSR} = \left(\frac{L_{TOT}}{ps_S}\right) P_S \; SD_S^{WSR} + \left(\frac{L_{TOT}}{ps_D}\right) P_D \; SD_D^{WSR}. \tag{4.16}$$

The corresponding overall computing-plus-communication energy: $\mathcal{E}_{MIG}(J)$ wasted by the VM migration reads as in:

$$\mathcal{E}_{MIG} = \left(\frac{L_{TOT}}{ps_S}\right) P_S \; SD_S^{MIG} f + \left(\frac{L_{TOT}}{ps_D}\right) P_D \; SD_D^{MIG}(1-f) + \mathcal{E}_{TOT}. \tag{4.17}$$

In Eq. (4.17), I have that: (i) SD_S^{MIG} (resp., SD_D^{MIG}) is the migration-induced slowdown at the source (resp., destination) server (see Eq. (4.13)); (ii) f (resp., $(1-f)$) is the fraction of the overall workload L_{TOT} processed by the source (resp., destination) server; and, (iii) $\mathcal{E}_{TOT}$ is the communication energy in (3.11) consumed by the VM migration. Hence, from an energy point of view, VM migration

[2]These processing speeds are proportional to the fractions of CPU cycles that the VMMs of Fig. 2.1 assign to the VM on the source and destination servers [51].

[3]The replication-induced slowdown arises from the interrupts which are needed to guarantee server synchronization. Its typical value is around 2 [32], [55]. Since the migration-induced slowdown arises from bandwidth-contention phenomena (see Eq. (4.13)), the replication and migration-induced slowdowns generally assume different values.

outperforms VM replication when the following inequality holds:

$$\mathcal{E}_{MIG} < \mathcal{E}_{WSR}. \tag{4.18}$$

Interestingly enough, in the case in which the source and destination servers are homogeneous computing nodes (that is, when I have: $ps_S \equiv ps_D \triangleq ps_0$; $P_S \equiv P_D \triangleq P_0$; $SD_S^{WSR} \equiv SD_D^{WSR} \triangleq SD_0^{WSR}$, and $SD_S^{MIG} \equiv SD_D^{MIG} \triangleq SD_0^{MIG}$), the inequality in (4.18) reduces to the following one:

$$\mathcal{E}_{TOT} < \left(\frac{L_{TOT}}{ps_0}\right) P_0 \left[2SD_0^{WSR} - SD_0^{MIG}\right], \tag{4.19}$$

which holds regardless of the value assumed by f in (4.17). Eq. (4.19) confirms that, in order to guarantee energy-efficient VM migrations, I must minimize the communication energy $\mathcal{E}_{TOT}$ wasted by the memory migration and simultaneously limit the migration-induced slowdown. This is, indeed, the target of the BMOP in (4.6), (4.7).

4.5 Feasibility conditions of the BMOP

The following *Proposition 1* formalizes the necessary and sufficient conditions for the feasibility of the BMOP in (4.6), (4.7) (see the Appendix A for the proof).

Proposition 1. *The BMOP in* (4.6), (4.7) *is feasible if and only if the following three conditions are simultaneously met:*

$$\left\{\left(\frac{M_0}{\Delta_{MMT}}\right)\left[\left(\frac{I_{MAX}+2}{\widehat{R}}\right)\delta\left(\frac{\overline{w}}{\widehat{R}}-1\right)+ \left(\frac{1-(\overline{w}/\widehat{R})^{I_{MAX}+2}}{\widehat{R}-\overline{w}}\right)\left(1-\delta\left(\frac{\overline{w}}{\widehat{R}}-1\right)\right)\right]\right\} \leq 1, \tag{4.20}$$

$$(M_0/\Delta_{SC})(1/\widehat{R})(\overline{w}/\widehat{R})^{I_{MAX}+1} \leq 1, \tag{4.21}$$

$$\theta\ \beta\ (\overline{w}/\widehat{R}) \leq 1. \tag{4.22}$$

Regarding the practical utility of the conditions (4.20)-(4.22), I point out that, when (and only when) these conditions are met, I are guaranteed that there exists at least one value of R that satisfies all the constraints in (4.1), (4.2), (4.3), (4.4) and (4.5). Interestingly, the effects of $\widehat{R}$ and I_{MAX} on the reported feasibility conditions are formally stressed by the following *Proposition 2*.

Proposition 2. (a)) *At fixed I_{MAX}, all the functions at the left-hand-side (l.h.s.) of Eqs.* (4.20) *unused internals-* (4.22) *strictly decrease (resp., strictly increase) for increasing values of $\widehat{R}$ (resp., $\overline{w}$).*

(a)) *At fixed $\widehat{R}$, I have that:*

(b.1)) *the feasibility condition in* (4.22) *may be met only if $(\overline{w}/\widehat{R}) \leq 1$ at $\theta = 1$;*

(b.1)) *for increasing values of I_{MAX}, the function at the l.h.s. of Eq.* (4.21)*: (i) does not vary; (ii) is strictly decreasing; and, (iii) is strictly increasing, at $(\overline{w}/\widehat{R}) = 1$, $(\overline{w}/\widehat{R}) < 1$, and $(\overline{w}/\widehat{R}) > 1$, respectively;*

(b.1)) *for increasing values of I_{MAX}, the function at the l.h.s. of Eq.* (4.20) *strictly increases, regardless of the values assumed by the ratio $(\overline{w}/\widehat{R})$.*

Proof. A direct inspection of Eqs. (4.20)-(4.22) leads to the stated conclusions. □

Overall, *Proposition 2* points out that increasing values of $\widehat{R}$ always reduce the memory migration and stop-and-copy times T_{MMT} and T_{SC}, while the corresponding effects of I_{MAX} are more questionable. In fact, T_{MMT} always increases for increasing I_{MAX} (see *Proposition 2.b.3*), while larger values of I_{MAX} lead to reduced values of T_{SC} only at: $(\overline{w}/\widehat{R}) < 1$ (see *Proposition 2.b.2*).

4.6 On the optimized setting of I_{MAX}

The previously reported conclusion leads, in turn, to two main insights of practical interest. First, at $(\overline{w}/\widehat{R}) \geq 1$, the migration technique that minimizes the stop-and-copy time in (3.7) is the SaCM one (see Eq. (4.11)). Second, at $(\overline{w}/\widehat{R}) < 1$, I claim that an optimized setting: $\widetilde{I}_{MAX}$ of I_{MAX} is obtained by computing the value of I_{MAX} that meets the constraint in (4.21) with the equality, that is,

$$\widetilde{I}_{MAX} \equiv \left\lceil \frac{\log(M_0/\Delta_{SC}\widehat{R})}{\log(\widehat{R}/\overline{w})} - 1 \right\rceil, \text{ for } (\widehat{R}/\overline{w}) > 1, \tag{4.23}$$

where $\lceil . \rceil$ is the ceiling function. In order to support this claim, let us consider the cases of: (i) $\widetilde{I}_{MAX} \leq 0$; (ii) $\widetilde{I}_{MAX} \geq 1$ and the feasibility condition in (4.20) failing at $I_{MAX} = \widetilde{I}_{MAX}$; and, (iii) $\widetilde{I}_{MAX} \geq 1$ and the BMOP feasible at $I_{MAX} = \widetilde{I}_{MAX}$. The first case occurs when the tolerated stop-and-copy time Δ_{SC} in (4.2) is so high that the PeCM technique is useless. In this case, the SaCM technique should be applied, in order to minimize both the volume of the migrated data and

the corresponding consumed communication energy (see (4.11)). When the second case happens, the BMOP is infeasible and I are forced to increase Δ_{MMT} up till the condition in (4.20) is met at $I_{MAX} = \widetilde{I}_{MAX}$. Finally, when the third case occurs, the value of Δ_{SC} is so low that the SaCM technique fails to meet the feasibility condition in (4.21) (see Eq.(4.11)). Hence, in this case, the key question concerns the evaluation of the optimal setting of I_{MAX} that minimizes the consumed energy $\mathcal{E}_{TOT}$ in (4.6). Unfortunately, this is an outstanding question which is *still unresolved*, even in the (more investigated) case of state-of-the-art hypervisors. In fact, it has been tested that the application-oblivious default setting: $I_{MAX} = 29$ currently implemented by state-of-the-art hypervisors [19] does not guarantee, indeed, the on-line convergence of the iterative pre-copy process, especially for values of $(\overline{w}/\widehat{R})$ approaching the unit [16], [50]. Interestingly enough, I anticipate that, in all the carried out field trials, I have ascertained that the application-aware setting in (4.23) *minimizes* also the consumed energy in (4.6) (see Section 4.9.5).

4.7 Optimal bandwidth management

The objective function in (3.11) is not convex for $1 < \alpha < 2$, so that the resulting BMOP is not a convex optimization problem. However, since the objective function and the constraints in (4.6), (4.7) are posynomial functions (see Section 11.5 of [56]), and, then, the BMOP is an instance of Geometric Programming, it may be recast in a convex form by applying the log-transformation: $\widetilde{R} \triangleq log(R)$. Hence, after introducing the following dummy positions (see Eqs. (3.11), (4.1)), (4.2) and (4.3)): $\mathcal{E}_{TOT}(\widetilde{R}) \triangleq \mathcal{E}_{TOT}(R = e^{\widetilde{R}})$, and $\Psi_i(\widetilde{R}) \triangleq \Psi_i(R = e^{\widetilde{R}})$, $i = 1, 2, 3$, the BMOP may be equivalently re-formulated as in:

$$\min_{\widetilde{R}} \ \mathcal{E}_{TOT}(\widetilde{R}), \tag{4.24}$$

$$s.t.: \ \Psi_i(\widetilde{R}) \leq 0, \ i = 1, 2, 3, \ \text{and:} \ \widetilde{R} - \log \widehat{R} \leq 0. \tag{4.25}$$

Since the above problem is strictly convex, it admits an unique solution whenever the feasibility conditions in (4.20)-(4.22) are met. The solution may be computed, in turn, through an application of the Karush-Kuhn-Tucker (KKT) optimality conditions, provided that the Slater's qualification is also met (see Chapter 4 of [56]). At this regard, the following (sufficient) condition may be proved (see the Appendix B for the proof).

Proposition 3. Let all the feasibility conditions in (4.20)-(4.22) be met with the

strict inequality. Then, the Slater's qualification holds for the optimization problem in (4.24), (4.25).

Since the problem in (4.24), (4.25) is convex, the box constraint on $\widetilde{R}$ in (4.25) may be managed as an implicit one. Hence, the resulting Lagrangian function reads as in:

$$L(\widetilde{R}, \overrightarrow{\lambda}) = \mathcal{E}_{TOT}(\widetilde{R}) + \sum_{i=1}^{3} \lambda_i \Psi_i(\widetilde{R}), \tag{4.26}$$

where $\overrightarrow{\lambda} \triangleq [\lambda_1 \lambda_2 \lambda_3]^T$ is the (column) vector of the (nonnegative) Lagrange multipliers of the convex constraints in (4.25). Furthermore, since strong duality and Lagrangian min-max equality hold (see Chapter 6 of [56]), the (unique) solution of (4.24), (4.25) is the saddle point:

$$\max_{\overrightarrow{\lambda} \geq \overrightarrow{0}} \left\{ \min_{\widetilde{R} \leq \log \widehat{R}} \left\{ L(\widetilde{R}, \overrightarrow{\lambda}) \right\} \right\}, \tag{4.27}$$

of the Lagrangian function in (4.26). This is, in turn, the orthogonal projection onto the box-type sets: $\widetilde{R} \leq log\widehat{R}$, and $\overrightarrow{\lambda} \geq \overrightarrow{0}$ of the solution of the following algebraic equation:

$$\overrightarrow{\nabla} L(\widetilde{R}, \overrightarrow{\lambda}) = \overrightarrow{0}, \tag{4.28}$$

where $\overrightarrow{\nabla}(.)$ is the four-dimensional vector gradient of $L(.)$ in (4.26) performed with respect to the primal and dual (scalar) variables $\widetilde{R}$ and λ_i, i=1, 2, 3. The final Appendix C details the analytical expressions of the four partial derivatives $\nabla_{\widetilde{R}} L(.)$ and $\nabla_{\lambda_i} L(.)$, i=1, 2, 3.

4.7.1 Adaptive primal-dual iterations

Due to the presence of the exponential terms, the solution:

$$\{\widetilde{R}^*, \lambda_1^*, \lambda_2^*, \lambda_3^*\}, \tag{4.29}$$

of Eq. (4.28) resists closed-form computation. However, it may be iteratively computed by implementing on-line a suitable set of gradient-based projected primal-dual iterations. At this regard, I note that, as pointed out in [57] and [58], the primal-dual algorithm is an iterative procedure for solving convex optimization problems, which applies quasi-Newton methods for updating the primal-dual variables simultaneously and moving towards the saddle-point of the underlying Lagrangian function at each iteration. In our framework, four scalar iterations must be carried out at the *nth* step, namely (see Eqs. (4.27) and (4.28)):

$$\widetilde{R}^{(n+1)} = \min\left\{\log \widehat{R};\ \widetilde{R}^{(n)} - \zeta_0^{(n)} \nabla_{\widetilde{R}} L(\widetilde{R}^{(n)}, \overrightarrow{\lambda}^{(n)})\right\}, \tag{4.30}$$

and

$$\lambda_j^{(n+1)} = \max\left\{0;\ \lambda_j^{(n)} + \zeta_j^{(n)} \nabla_{\lambda j} L(\widetilde{R}^{(n)}, \overrightarrow{\lambda}^{(n)})\right\},\ j = 1, 2, 3, \tag{4.31}$$

where $n = 0, 1, 2, \ldots$, is a discrete iteration index and $\left\{\zeta_j^{(n)}\right\}$, $j = 0, 1, 2, 3$, is a (suitable) sequence of nonnegative step-sizes.

Regarding the convergence to the global minimum of the primal-dual iterations of Eqs. (4.30) and (4.31), three main remarks are in order. First, under the feasibility conditions of *Proposition 1*, the global minimum of Eqs. (4.24) and (4.25) exists. Second, due to the strict convexity of the problem in (4.24) and (4.25), the global minimum is unique (see *Theorem 3.4.2* of [56]). Furthermore, the convergence of the primal-dual iterations of Eqs. (4.30) and (4.31) to the global minimum is guaranteed, regardless of the adopted starting point and the size of the considered instance of the optimization problem. Formal proofs of this property of global convergence may be found, for example, in [56], [57] and [59]. Third, in practical application scenarios, the average memory dirty rate $\overline{w}$ and/or the round-trip-time dependent K_0 parameter in (3.11) may exhibit *unpredictable* (possibly, abrupt) time-variations over a same migration session and/or consecutive migration sessions. As detailed in Section 4.9.3, $\overline{w}$ may vary due to workload fluctuations experienced by the migrating VM [50], [49], while congestion-induced jitters of the round-trip-time RTT of the utilized TCP connection may give arise to (unpredictable) changes of K_0 in (3.9). An effective means for tracking the unpredictable time-fluctuations of K_0 and/or $\overline{w}$ in an *adaptive* way is provided by the gradient-descendant algorithm in [60] for the adaptive updating of (scalar) step-size sequences. In my framework, these updating iterations read as in:

$$\zeta_0^{(n+1)} = \max\left\{0;\ \min\left\{a_{MAX}; \zeta_0^{(n)} - \gamma B_0^{(n)} \nabla_{\widetilde{R}} L(\widetilde{R}^{(n)}, \overrightarrow{\lambda}^{(n)})\right\}\right\}, \tag{4.32}$$

and

$$\zeta_j^{(n+1)} = \max\left\{0;\ \min\left\{a_{MAX}; \zeta_j^{(n)} + \gamma B_j^{(n)} \nabla_{\lambda_j} L(\widetilde{R}^{(n)}, \overrightarrow{\lambda}^{(n)})\right\}\right\},\ j = 1, 2, 3, \tag{4.33}$$

where γ and a_{MAX} are positive constants to be suitably tuned [60] (see Section 4.9.3). Furthermore, the scalar $B_j^{(n)}$ in (4.32) and (4.33) is the derivative of the variable at the l.h.s. of Eqs. (4.30) and (4.31) with respect to the corresponding step-size $\zeta_j^{(n)}$ [60], and it may be iteratively updated as in (see Eq. (2.5) of [60]):

$$B_0^{(n+1)} = \left(1 - \zeta_0^{(n)}\right) B_0^{(n)} - \nabla_{\widetilde{R}} L(\widetilde{R}^{(n)}, \overrightarrow{\lambda}^{(n)}), \tag{4.34}$$

and

$$B_j^{(n+1)} = \left(1 - \zeta_j^{(n)}\right) B_j^{(n)} + \nabla_{\lambda_j} L(\widetilde{R}^{(n)}, \overrightarrow{\lambda}^{(n)}),\ j = 1, 2, 3, \tag{4.35}$$

with $B_j^{(0)} = 0$, for $j = 0, 1, 2, 3$.

4.8 Implementation aspects: profiling tasks and implementation scalability

The implementation of the proposed bandwidth manager requires a priori information about the power-vs.-rate relationship in (3.8) and the memory size and dirty rate in (3.5). As detailed in the sequel, this information may be acquired on-line during the first part of the Pre-migration stage by exploiting some commands and profiling tools already done available by current VMMs [19], [51].

Profiling the network connection and the migrating application

The average round-trip-time RTT in (3.9) and the maximum throughput $\widehat{R}$ in (4.4) of the available TCP connection may be directly measured at the Transport layer by using, for example, the (standard) Linux *iperf* command [51]. In order to profile at runtime the parameters $\mathcal{E}_{SETUP}$, K_0 and α in (3.11), I may use the Xen *ifconfig* command [51]. It reports the power state of the physical NIC which is used by the migrating VM (see Fig. 2.1). Hence, by issuing the *ifconfig* command at R=0 and R=1 (Mb/s), I directly measure $\mathcal{E}_{SETUP}$ and K_0, respectively (see Eq. (3.11)). Afterwards, by issuing the *ifconfig* command at R=$\widehat{R}$, I measure the total (e.g., static-plus-dynamic) power: $P_{TOT}(\widehat{R})$ consumed by the TCP connection at R=$\widehat{R}$. Hence, since, by definition, I have that:

$$P_{DYN}(\widehat{R}) \triangleq \widehat{R}\mathcal{E}_{DYN}(\widehat{R}) \equiv P_{TOT}(\widehat{R}) - \widehat{R}\mathcal{E}_{SETUP}, \tag{4.36}$$

directly from the relationship in (3.8), I obtain the following closed-form expression for the α exponent:

$$\alpha = \frac{\log(\frac{P_{DYN}(\widehat{R})}{K_0})}{\log \widehat{R}} \equiv \frac{\log(\frac{P_{TOT}(\widehat{R}) - \widehat{R}\mathcal{E}_{SETUP}}{K_0})}{\log \widehat{R}}. \tag{4.37}$$

I ascertained through the carried out field trials that the profiled energy consumptions in (3.11) of the implemented connections differ from the actual ones measured through an (external) Watts Up Pro power-meter *less* than *2%*. This confirms that both the performed profiling operations and the adopted energy and traffic models of Section 3.3 are, indeed, accurate enough.

Regarding the profiling of the migrating application, I note that the memory size M_0 of the migrating VM may be measured during the Pre-migration stage through the *xenstore* command [51]. Afterwards, in order to profile at run-time the corresponding average dirty memory rate $\overline{w}$, let $\widetilde{M_0}$ be the (integer-valued) number of memory pages of the migrating VM and let $m = 0, 1, \ldots, (\widetilde{M_0} - 1)$, be the (possibly, relative) corresponding memory address index. Furthermore, let $\chi(m, i) \in \{0, 1\}$, $m = 0, 1, \ldots, (\widetilde{M_0} - 1)$, $i = 0, \ldots, I_{MAX}$, be the binary function which marks the dirtied/not dirtied state of the mth memory page at the end of the ith round. Interestingly enough, the spectrum $\{\chi(m, i)\}$ of the dirtied memory pages may be directly acquired at run-time from the dirty bitmap which the Xen hypervisor makes periodically available [19], [51]. Therefore, the resulting dirty memory rate $\overline{w}$ averaged over the duration T_{IP} of the iterative pre-copy stage may be directly profiled on-line through the following relationship:

$$\overline{w} = \left(\frac{M_0}{T_{IP} \widetilde{M_0}} \right) \left(\sum_{m=0}^{\widetilde{M_0}-1} \sum_{i=0}^{I_{MAX}} \chi(m, i) \right) (Mb/s). \tag{4.38}$$

Implementation complexity and scalability

From an implementation point of view, the four primal-dual iterations in (4.30), (4.31) are carried out by the source server during the last part of the Pre-migration stage. Although the duration of each $n-$indexed iteration in (4.30), (4.31) may depend on the adopted VMM, it should be small enough to allow the iterations to converge to the global optimum within a limited fraction of the overall Pre-migration stage. On the basis of this consideration, I anticipate that, in the carried out field trials, the time duration: $T_I(s)$ of each $n-$indexed iteration is set to ten times the inverse of the maximum clock's frequency of the utilized CPU.

Regarding the implementation complexity and scalability of the proposed bandwidth manager, I point out that each VMM *locally* runs the iterations in (4.30), (4.31) and *only* manages the migration of the hosted VMs (see Fig. 2.1). Furthermore, the implementation complexity of the iterations in (4.30), (4.31) does *not* depend on the actual setting of I_{MAX}. These structural properties of the proposed bandwidth manager lead, in turn, to three main conclusions about the resulting implementation complexity. First, the implementation of the manager may be carried out on a per-VM basis, that is, in a *distributed way*. Second, the resulting per-migration implementation complexity does not depend neither on the total number of (possibly, simultaneous) performed migrations nor on the (possibly, large) size of the considered data center. This is due to the fact that, in our framework, the overall effect on the performed migration of the aggregate traffic supported by the data center is fully

summarized by the corresponding value assumed by the (scalar) K_0 parameter of Eq. (3.9). Third, the implementation complexity of Eqs. (4.30) and (4.31) increases, by design, in a *linear way* with the average number $\overline{M}_I$ of the (n-indexed) iterations requested by the convergence to the global minimum. Hence, the *per-migration* implementation complexity of the proposed bandwidth manager scales up as $O(\overline{M}_I)$, with $\overline{M}_I$ typically limited up to 10 iterations (see Section 4.9.3). From the outset, I conclude that the problem in Eqs. (4.6) and (4.7) is a non-convex optimization problem but it is not NP-Hard. This property is retained by the overall class of the Geometric Programming problems and the tackled problem is, indeed, an instance of Geometric Programming (see, for example, Section 11.5 of [56]). At this regard, I point out that rich lists of non-convex polynomial-complexity optimization problems are provided, for example, in [61] and [62], where the non-convexity-vs.-NP-Hardness dichotomy is also discussed.

4.9 Simulations result

In order to actually test and compare the performance of the proposed bandwidth manager, I have implemented a wired test-bed. In-band PeCM is the utilized migration technique and the architecture of the implemented test-bed is the one reported in Fig. 2.1.

At this regard, I stress that, since our focus is on the management of the migration bandwidth, the placement of the migrating VM is assumed to be already decided by the Migration Planner of Fig. 2.1 during the Pre-migration stage of Section 3.2. Hence, according to Section 2.1, testing the placement performance of the migrated VMs is out of the scope of the carried out field trials. Furthermore, since the *distributed* nature of the proposed bandwidth manager guarantees that its per-migration implementation complexity does *not* depend on the size of the considered data center (see the last part of the previous Section 4.8), as, for example, in [18], [28], [30], [50] and [49], it suffices to consider a test-bed which is constituted by two LAN-interconnected virtualized physical servers. Specifically, the implemented test-bed consists of two identical Dell Power Edge servers equipped with 3.06 GHz Intel Xeon dual-core CPU and 4 GB of RAM. They alternate the roles of source/destination servers for the migrating VMs. A Gigabit Ethernet LAN is implemented through a Cisco Nexus 55548P commodity switch. The NAS of Fig. 2.1 is configured by using an IBM server xSeries 336 having Intel Xeon X5470 3.00 GHz CPU, 2 GB fully buffered DIMM modules, integrated Gigabit Ethernet NICs, and an Ultra320 SCSI controller.

All servers use the paravirtualized Xen 3.3 hypervisor as VMM [51]. We imple-

mented in software the proposed bandwidth manager at the driver domain (e.g., *Dom0*) of the legacy Xen 3.3 protocol hosted by the servers. Interestingly, out of approximatively 1600 lines of code needed for implementing the proposed bandwidth manager, 40% is directly reused from existing Xen/Linux code. The reused code includes part of the Linux's TCPNewReno congestion control protocol, and Xen's I/O buffer management, shadow page tables, dirty bitmaps, *iperf* and *ifconfig* command tools [51]. Furthermore, in the carried out field trials, the Xen-driven CPU scheduler is set to operate under the no work conserving mode [51], in order to avoid resource contention among concurrent VMs. Specifically, I enforce *Dom0* to use a single physical core, in order to isolate it and avoid performance interference. The migrating VM uses the remaining physical core.

4.9.1 Test applications and test-bed profiling

Multiple heterogeneous synthetic and real-world applications have been selected, in order to carry out comparative field trials. Specifically, as a synthetic benchmark, I used the *memtester* application [63], which is a highly write-intensive program currently used to find faults in RAM. Interestingly, it allows us to set the value of the average memory dirty rate $\overline{w}$ through an input program parameter. From the SPEC CINT2000 benchmark tool [64], the sub-programs *gap, vortex* and *eon* have been selected as first instances of real-world applications with (very) different spectra of the resulting memory change probabilities (see Section 4.9.7). The *gap, vortex* and *eon* applications are three real-world programs which mainly stress the processor, memory and compiler components of the host server, respectively [64].

As second set of real-world applications, from SPEC CPU2006 [65], the *401.bzip2* and *429.mcf* programs have been chosen. The first one is a read-intensive (e.g., CPU intensive) application, which exhibits a quite low memory dirty rate. The second one is a more memory intensive application, which presents a balanced mix of read and write memory accesses. Finally, as third instance of real-world application, I have also selected the *memcached* program [52]. This is a (very) write-intensive program, which caches multiple key/value pairs in the main memory. In this case, I used *memaslap* as load generator (e.g., client program) [66] and I have configured it, so to randomly generate *set* and *get* operations at an $1:10$ ratio.

In the carried out field trials, TCPNewReno-over-IP connections are built up for migrating the tested VMs. Both the *MSS* in (3.9) of the utilized TCP connections and the capacity of the transmit and receive buffers of the implemented Xen hypervisor are set to the size of the memory pages of the migrated VMs (see Table 4.1). So doing, if a TCP segment is lost and/or a part of a memory page is dirtied, the whole TCP segment is marked as lost/dirtied and re-migrated later.

Each migration trial has been repeated six times and the average results are reported in the sequel. Furthermore, unless otherwise stated, it is understood that a random ordering is adopted for migrating the dirtied memory pages over the pre-copy rounds. The effect of specific migration orderings will be examined in the last Section 4.9.7. Finally, in all carried out field trials, the migration of the VM from the source server starts after 15 seconds of pre-running. During this initial time interval, all the profiling tasks of Section 4.8 are carried out. Table 4.1 reports the profiled parameters of the implemented test-bed.

$\alpha = 1.31$	$\mathcal{E}_{SETUP} = 3 \times 10^{-4}\ (J)$
$K_0 = 1.8 \times 10^{-3}\ ((W) \times (s/Mb)^{\alpha})$	$RTT = 4 \times 10^{-4}\ (s)$
$MSS = 12\ (Kb)$	$Buffers' size\ of\ the\ VMM = 12\ (Kb)$

Table 4.1. Profiled parameters of the implemented test-bed.

4.9.2 The benchmark Xen bandwidth management

The currently implemented Xen hypervisor adopts a pre-copy heuristic bandwidth management policy, which operates on a *best effort* basis, while attempting to shorten the final stop-and-copy time [19], [30]. The rationale behind this Xen policy is that, in principle, the stop-and-copy time may be reduced by monotonically increasing the migration bandwidth over consecutive rounds [30]. For this purpose, the Xen hypervisor uses pre-assigned minimum: $R_{MIN}^{XEN}\ (Mb/s)$, and maximum: $R_{MAX}^{XEN}\ (Mb/s)$ bandwidth thresholds[4], in order to bound the migration bandwidth during the pre-copy stage (see Section 5.3 of [30]). Specifically, the Xen migration bandwidth R^{XEN} equates: $R_{MIN}^{XEN}\ (Mb/s)$ at $round\#0$, and, then, it increases in each subsequent round by a constant term: $\Delta R^{XEN}\ (Mb/s)$, so to reach the maximum value: $R^{XEN} = R_{MAX}^{XEN}$ at the last round: $round\#(I_{MAX}^{XEN} + 1)$ (see Section 5.3 of [30]). In the carried out field trials, I have implemented this benchmark policy by setting:

$$\Delta R^{XEN} = (R_{MAX}^{XEN} - \overline{w})/(I_{MAX}^{XEN} + 1), \tag{4.39}$$

and

$$R_i^{XEN} = \overline{w} + i\Delta R^{XEN}, i = 0, \ldots, (I_{MAX}^{XEN} + 1). \tag{4.40}$$

We point out that, on the basis of the (recent) surveys in [16], Chapter 3 of [19] and Chapter 17 of [25], as well as at best of the authors' knowledge, this is the only bandwidth management policy currently considered by both academy and industry

[4] In the sequel, I denote by the upper-script: ^{XEN} the Xen's performance metrics and working parameters, while I mark by the asterisk: * the performance metrics and working parameters of the proposed bandwidth manager.

for VM migration. This is also the bandwidth policy currently implemented by Xen, KVM and VMware commercial hypervisors [19].

4.9.3 Tests on the tracking capabilities under contention phenomena

Real-world applications may vary the produced traffics over the time [47] and, then, it may be of interest to test how the proposed bandwidth manager reacts when the workload offered by the migrating VM changes unexpectedly. As pointed out in [16], memory contention phenomena and/or network congestions may produce abrupt (typically, unpredictable) time-variations of the parameters $\overline{w}$ and/or K_0 present in the energy function of Eq. (3.11). Hence, in order to evaluate the tracking capabilities of the proposed adaptive bandwidth manager in (4.30), (4.31) and its sensitivity to the parameters a_{MAX} and γ in (4.32), (4.33), in Fig. 4.1(a) I report the measured behaviors of the energy sequence: $\{\mathcal{E}_{TOT}^{*(n)},\ n \geq 0\}$ when, due to memory contention phenomena, the memory dirty rate of the running *memtester* application abruptly passes from: $\overline{w} = 225$ (Mb/s) to: $\overline{w} = 675$ (Mb/s) at $n = 30$ and, then, it falls out to: $\overline{w} = 225$ (Mb/s) at $n = 60$. Fig. 4.1(b) reports the corresponding energy behaviors when, due to congestion-induced fluctuations of the round-trip-time RTT, the K_0 parameter in (3.9) passes from: 1.8×10^{-3} $((W) \times (s/Mb)^{\alpha})$ to: 1.8×10^{-2} $((W) \times (s/Mb)^{\alpha})$ at $n = 30$ and, then, it falls out to: $1.8 \times 10^{-3}((W) \times (s/Mb)^{\alpha})$ at $n = 60$.

An examination of the plots of Fig. 4.1 supports four main conclusions. First, according to the fact that the energy function in Eq. (3.11) increases for increasing $\overline{w}$ and/or K_0, all the plots of Figs. 4.1(a) and 4.1(b) scale up at $n = 30$ and, then, scale down at $n = 60$. Second, the proposed bandwidth manager quickly reacts to abrupt unpredicted time variations of the migrating application and/or underlying network connection. Specifically, I have numerically ascertained that it is capable to converge to the steady-state optimum within 25-30 iterations with a final accuracy less than 1%, while only 5-6 iterations suffice to attain the convergence with an accuracy around 6%-7% (see Fig. 4.1). Third, virtually indistinguishable plots are obtained for γ ranging over the interval $[10, 10^3]$, so that Fig. 4.1(a) reports the time trajectories measured at $\gamma = 100$ and $a_{MAX} = 10^{-3}, 5 \times 10^{-3}$ and 10^{-1}. Fourth, the plots of Figs. 4.1(a) and 4.1(b) at $a_{MAX} = 0.1$ (resp., $a_{MAX} = 0.001$) present the shortest (resp., longest) durations of the transient states, but they exhibit the largest (resp., smallest) oscillations in the steady-states. The plots at $a_{MAX} = 0.005$ show, indeed, intermediate behaviors. Specifically, in Fig. 4.1(a), the plot at $a_{MAX} = 0.005$ approaches the plot at $a_{MAX} = 0.1$, while, in Fig. 4.1(b), the curve at $a_{MAX} = 0.005$ approaches the corresponding curve at $a_{MAX} = 0.001$. We believe that this is due

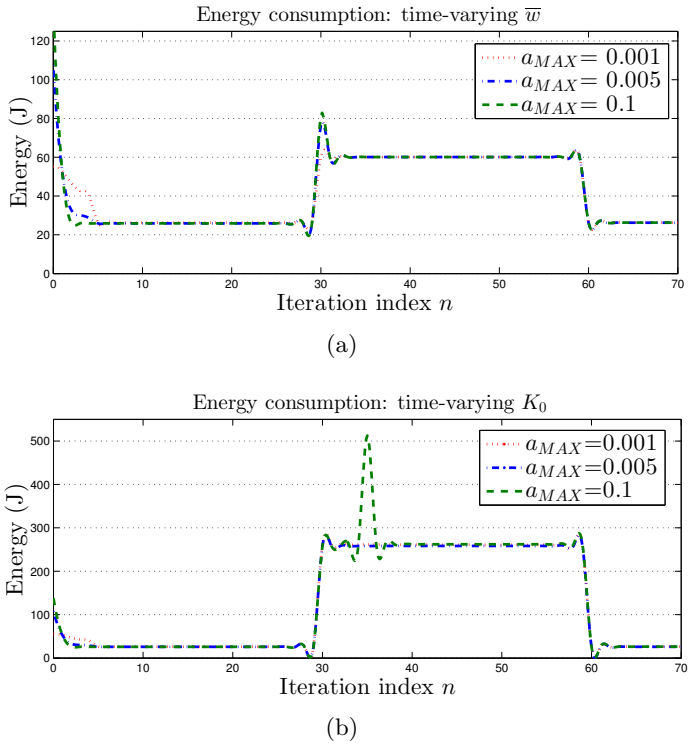


Figure 4.1. Time evolutions (in the n index) of the energy consumption of the proposed bandwidth manager at: $\widehat{R} = 900$ (Mb/s), $M_0 = 512$ (Mb), $\beta = 1.15$, $\Delta_{MMT} = 13.5$ (s), $\Delta_{SC} = 0.6$ (s), $\widetilde{I}_{MAX} = 3$, and $\gamma = 100$ for the application scenario of Section 4.9.3. (a) Case of time-varying $\overline{w}$; (b) Case of time-varying K_0.

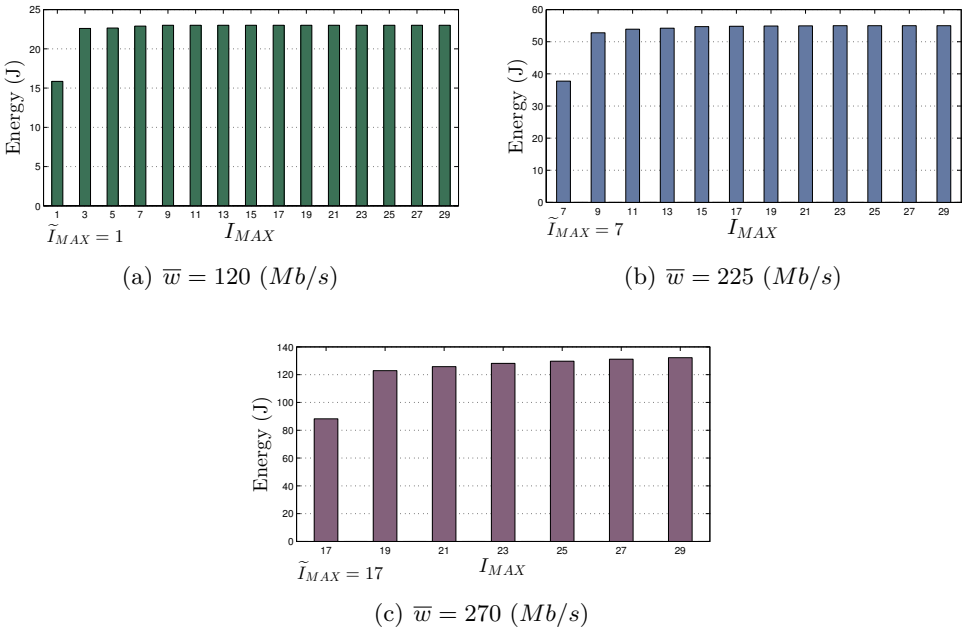
to the fact that the scaling behavior of the energy function in Eq. (3.11) is different when $\overline{w}$ or K_0 is varied.

Overall, from the outset, I conclude that the proposed adaptive bandwidth manager is robust with respect to the actual tuning of γ and a_{MAX}, at least for values of γ and a_{MAX} ranging over the intervals $[10, 10^3]$ and $[10^{-3}, 10^{-1}]$, respectively. However, at least in the carried out field trials, the setting: $\gamma = 100$ and $a_{MAX} = 5 \times 10^{-3}$ exhibits the best trade-off among the contrasting requirements of short transient-states and stable steady-states, and it will be adopted in the sequel.

4.9.4 Validation tests on $\widetilde{I}_{MAX}$

By referring to the test-bed setting of Table 4.1, the bar plots of Figs. 4.2 and 4.3 report the measured energy consumption: $\mathcal{E}^*_{TOT}$ of the proposed bandwidth manager for increasing values of I_{MAX} at $\widehat{R} = 300$ (Mb/s) and $\widehat{R} = 100$ (Mb/s), respectively. The corresponding values of $\widetilde{I}_{MAX}$ in (4.23) are marked on the x-axis of the reported Figures.

An examination of these plots leads to three main conclusions. First, in all carried



(a) $\overline{w} = 120$ (Mb/s)

(b) $\overline{w} = 225$ (Mb/s)

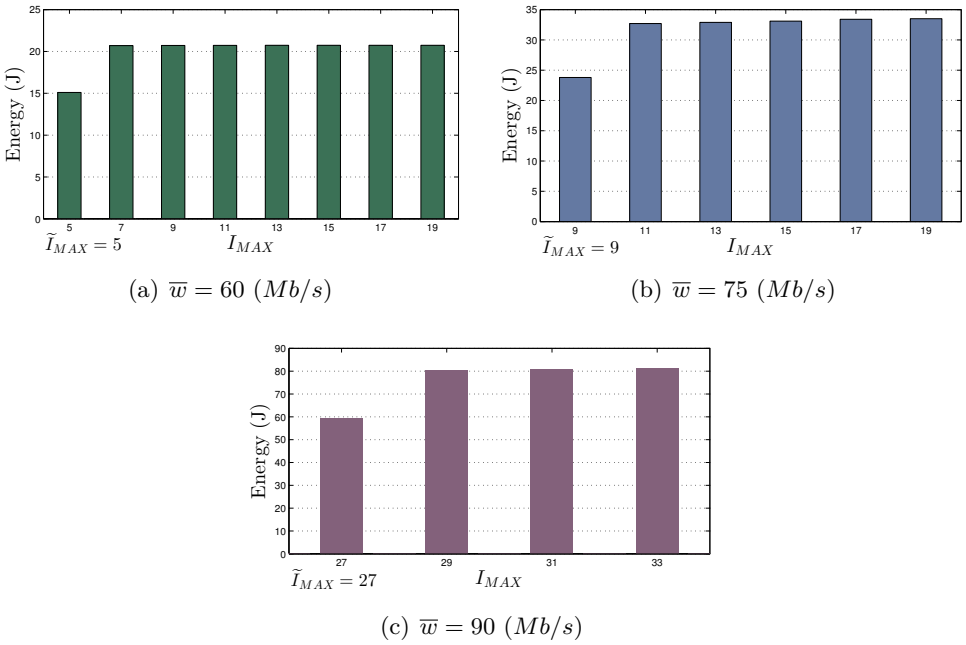
(c) $\overline{w} = 270$ (Mb/s)

Figure 4.2. $\mathcal{E}^*_{TOT}$ - vs. - I_{MAX} behavior for the proposed bandwidth manager at: $\widehat{R} = 300$ (Mb/s), $M_0 = 512$ (Mb), $\beta = 1.10$, $\Delta_{MMT} = 50$ (s), $\Delta_{SC} = 0.3$ (s). The application scenario of Section 4.9.4 is considered at: (a) $\overline{w} = 120$ (Mb/s); (b) $\overline{w} = 225$ (Mb/s); and, (c) $\overline{w} = 270$ (Mb/s).

out tests, $\widetilde{I}_{MAX}$ in (4.23) coincides with the value of I_{MAX} at which $\mathcal{E}^*_{TOT}$ attains its global minimum. Second, in all tested cases, I have experienced that $\widetilde{I}_{MAX}$ is the smallest value of I_{MAX} which makes the BMOP in (4.6), (4.7) to be feasible. Third, the measured increasing behavior of $\mathcal{E}^*_{TOT} \equiv \mathcal{E}^*_{TOT}(I_{MAX})$ for $I_{MAX} \geq \widetilde{I}_{MAX}$ is quite slow, and the corresponding energy gap: $|\mathcal{E}^*_{TOT}(I_{MAX}) - \mathcal{E}^*_{TOT}(\widetilde{I}_{MAX})|$ stays below 8%, even for values of the difference: $|I_{MAX} - \widetilde{I}_{MAX}|$ as high as 10. Overall, these observations confirm that the setting in (4.23) effectively reduces the consumed energy and it is also robust against measurement errors possibly affecting the performed profiling operations.

4.9.5 Comparative energy tests under random migration ordering and synthetic workload

The benchmark bandwidth management policy of the Xen hypervisor of Section 4.9.2 *does not* guarantee, by design, minimum energy consumptions and *does not* enforce QoS constraints on the resulting memory migration and stop-and-copy times. Furthermore, differently from $\widetilde{I}_{MAX}$ in (4.23), the maximum number of allowed rounds: I^{XEN}_{MAX} is fixed by the Xen hypervisor in an application-oblivious way (typically, $I^{XEN}_{MAX} \leq 29$; see [19], [51]). Hence, in order to carry out *fair* energy



(a) $\overline{w} = 60$ (Mb/s) (b) $\overline{w} = 75$ (Mb/s)

(c) $\overline{w} = 90$ (Mb/s)

Figure 4.3. $\mathcal{E}^*_{TOT}$ - vs. - I_{MAX} behavior for the proposed bandwidth manager at: $\widehat{R} = 100$ (Mb/s), $M_0 = 512$ (Mb), $\beta = 1.10$, $\Delta_{MMT} = 50$ (s), $\Delta_{SC} = 0.3$ (s). The application scenario of Section 4.9.4 is considered at: (a) $\overline{w} = 60$ (Mb/s); (b) $\overline{w} = 75$ (Mb/s); and, (c) $\overline{w} = 90$ (Mb/s).

comparisons, in the carried out field trials, I proceed as follows: (i) set I^{XEN}_{MAX} and R^{XEN}_{MAX}; (ii) measure the resulting Xen energy consumption $\mathcal{E}^{XEN}_{TOT}(J)$, speed-up factor β^{XEN}, memory migration time T^{XEN}_{MMT}, and stop-and-copy time T^{XEN}_{SC}; (iii) enforce $\widehat{R} \equiv R^{XEN}_{MAX}$, together with the QoS constraints: $\Delta_{MMT} \equiv T^{XEN}_{MMT}$, $\Delta_{SC} \equiv T^{XEN}_{SC}$, and $\beta \equiv \beta^{XEN}$; and, finally, (iv) measure the resulting energy consumption $\mathcal{E}^*_{TOT}$ of the proposed bandwidth manager at $I_{MAX} = \widetilde{I}_{MAX}$ (see Eq. (4.23)). The (aforementioned) *memtester* in [63] is the application considered in this section and the implemented migration ordering of the dirtied memory pages is the random one.

The numerical results measured through a campaign of field trials are reported by Tables 4.2 and 4.3. They embrace a spectrum of application scenarios, which are characterized by different network bandwidths, memory dirty rates and pre-copy rounds.

An examination of the results of Tables 4.2 and 4.3 leads to four main conclusions. First, in all the carried out field trials, the per-cent energy saving: $(1 - (\mathcal{E}^*_{TOT}/\mathcal{E}^{XEN}_{TOT}))\%$ of the proposed bandwidth manager over the Xen one is over 45% and approach 66% under large values of I_{MAX} (see the last rows of Tables 4.2 and 4.3). These noticeable energy gains support the conclusion that the Xen (heuristic) bandwidth management policy in (4.40) is definitely *energy suboptimal*.

On the contrary, the bandwidth management policy developed in this thesis is the optimal one and, by design, it minimizes the migration-induced energy consumption. Second, at assigned memory migration and stop-and-copy times, the number of pre-copy rounds: I_{MAX}^{XEN} required by Xen manager to meet the assigned migration times is, in average, about *two times* larger than the corresponding $\widetilde{I}_{MAX}$ one in (4.23). At this regard, I have numerically tested that about 20% the reported energy gains are induced by the optimized setting of $\widetilde{I}_{MAX}$ in (4.23). Third, at fixed $\overline{w}$ and $\widehat{R} \equiv R_{MAX}^{XEN}$, increasing values of I_{MAX}^{XEN} and $\widetilde{I}_{MAX}$ lead to increasing values of the measured energy gains (see the last rows of Tables 4.2 and 4.3). Since larger values of the pre-copy rounds lead, in turn, to lower values of the resulting stop-and-copy times (see the second rows of Tables 4.2 and 4.3), I conclude that more noticeable energy savings are provided by the proposed bandwidth manager under stricter downtimes. Fourth, the values of the measured energy gains mainly depend on the considered ratio: $(\overline{w}/\widehat{R})$ (compare the last rows of Tables 4.2 and 4.3). In the carried out tests, these gains attain their maxima for values of $(\overline{w}/\widehat{R})$ ranging over the interval $[0.4, 0.75]$ (see the last rows of Tables 4.2(a)-4.3(a) and 4.2(b)-4.3(b)). In fact, an examination of Tables 4.2 and 4.3 points out, that, although *both* the energies $\mathcal{E}_{TOT}^{XEN}$ and $\mathcal{E}_{TOT}^{*}$ consumed by the Xen and the proposed bandwidth managers increase for increasing $(\overline{w}/\widehat{R})$, the rate of the energy increment of the Xen bandwidth manager maximally exceeds the corresponding one of the proposed manager at values of $(\overline{w}/\widehat{R})$ around 0.5. However, the attained energy savings maintain larger than 45% and approach 53%, even for values of $(\overline{w}/\widehat{R})$ as high as 0.95 (see the last rows of Tables 4.2(c) and 4.3(c)).

I_{MAX}^{XEN}	7	15	23
$T_{SC}^{XEN} = \Delta_{SC}(s)$	$1.2\text{x}10^{-1}$	$1.8\text{x}10^{-3}$	$2.6\text{x}10^{-5}$
$T_{MMT}^{XEN} = \Delta_{MMT}(s)$	40.4	57.1	68.1
$\varepsilon_{TOT}^{XEN}(J)$	23.7	29.3	33.6
$\tilde{I}_{MAX}$	4	8	13
$\varepsilon_{TOT}^{*}(J)$	12.2	12.4	12.6
Energy saving(%)	48.7%	58.3%	63.7%

(a)

I_{MAX}^{XEN}	7	15	19
$T_{SC}^{XEN} = \Delta_{SC}(s)$	1.7	$2.9\text{x}10^{-1}$	$1.6\text{x}10^{-1}$
$T_{MMT}^{XEN} = \Delta_{MMT}(s)$	41.9	67.9	74.5
$\varepsilon_{TOT}^{XEN}(J)$	45.3	73.2	79.6
$\tilde{I}_{MAX}$	3	9	12
$\varepsilon_{TOT}^{*}(J)$	22.3	28.0	28.8
Energy saving(%)	50.7%	61.7%	63.9%

(b)

I_{MAX}^{XEN}	7	15	23
$T_{SC}^{XEN} = \Delta_{SC}(s)$	4.3	3.5	2.8
$T_{MMT}^{XEN} = \Delta_{MMT}(s)$	44.6	79.1	110
$\varepsilon_{TOT}^{XEN}(J)$	61.4	109.0	151
$\tilde{I}_{MAX}$	3	7	11
$\varepsilon_{TOT}^{*}(J)$	33.1	54.1	71.3
Energy saving(%)	46.1%	50.2%	52.7%

(c)

Table 4.2. Simulations results comparison for energy saved: $\widehat{R} = R_{MAX}^{XEN} = 100$ (Mb/s) and $M_0 = 512$ (Mb); (a) $\overline{w} = 40$ (Mb/s); (b) $\overline{w} = 75$ (Mb/s); (c) $\overline{w} = 95$ (Mb/s).

I_{MAX}^{XEN}	7	15	23
$T_{SC}^{XEN} = \Delta_{SC}(s)$	$1.3\text{x}10^{-2}$	$1.9\text{x}10^{-4}$	$2.9\text{x}10^{-6}$
$T_{MMT}^{XEN} = \Delta_{MMT}(s)$	4.56	6.35	7.64
$\varepsilon_{TOT}^{XEN}(J)$	63.6	78.6	90.2
$\tilde{I}_{MAX}$	4	8	13
$\varepsilon_{TOT}^{*}(J)$	32.7	32.9	33.2
Energy saving(%)	48.7%	58.3%	63.7%

(a)

I_{MAX}^{XEN}	7	15	27
$T_{SC}^{XEN} = \Delta_{SC}(s)$	$1.9\text{x}10^{-1}$	$5.9\text{x}10^{-2}$	$9.6\text{x}10^{-3}$
$T_{MMT}^{XEN} = \Delta_{MMT}(s)$	4.7	6.7	8.9
$\varepsilon_{TOT}^{XEN}(J)$	122	177	229
$\tilde{I}_{MAX}$	3	7	14
$\varepsilon_{TOT}^{*}(J)$	60.0	72.8	77.9
Energy saving(%)	50.7%	58.8%	66%

(b)

I_{MAX}^{XEN}	7	15	27
$T_{SC}^{XEN} = \Delta_{SC}(s)$	$4.7\text{x}10^{-1}$	$3.9\text{x}10^{-1}$	$2.8\text{x}10^{-1}$
$T_{MMT}^{XEN} = \Delta_{MMT}(s)$	4.9	8.8	13.7
$\varepsilon_{TOT}^{XEN}(J)$	165	292	456
$\tilde{I}_{MAX}$	3	7	13
$\varepsilon_{TOT}^{*}(J)$	89	146	211
Energy saving(%)	46.0%	50.2%	53.7%

(c)

Table 4.3. Simulations results comparison for energy saved: $\widehat{R} = R_{MAX}^{XEN} = 900$ (Mb/s) and $M_0 = 512(Mb)$; (a) $\overline{w} = 360$ (Mb/s); (b) $\overline{w} = 675$ (Mb/s); (c) $\overline{w} = 855$ (Mb/s).

4.9.6 Comparative tests under random migration ordering and real-world workloads

In order to further validate and refine the above conclusions by considering also real-world applications, in this section, I report and compare the migration-induced energy consumptions and stretching of the execution times which are suffered by the (aforementioned) *bzip2*, *mcf* and *memcached* programs. The test parameters are those of Table 4.1 and the migration ordering is the random one. Furthermore, all the reported performance results have been obtained at: $M_0 = 512\ (Mb)$, $\widehat{R} \equiv R_{MAX}^{XEN} = 900\ (Mb/s)$, $I_{MAX}^{XEN} = 29$ and $\widetilde{I}_{MAX} = 14$.

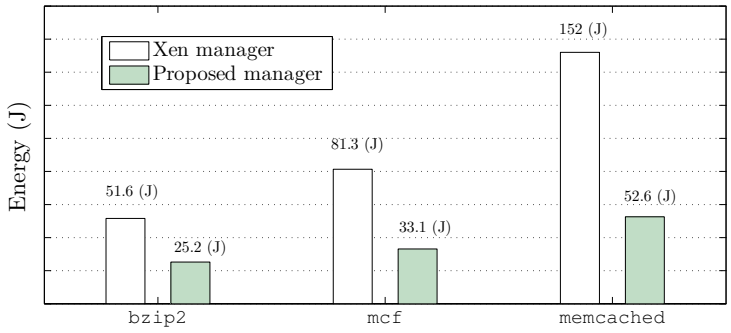


Figure 4.4. Energy consumptions for the application scenario of Section 4.9.6.

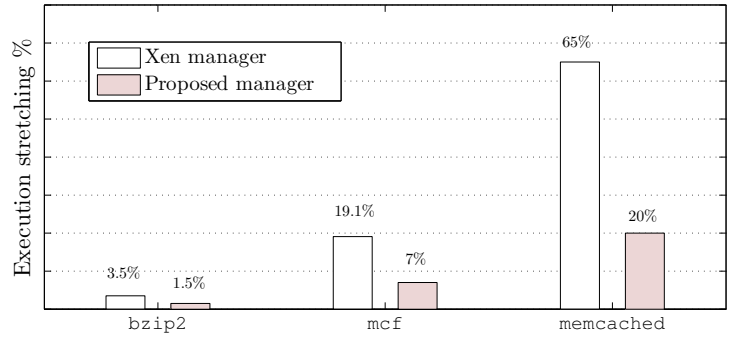


Figure 4.5. Migration-induced average stretching of the execution times(in per-cent) for the application scenario of Section 4.9.6.

Fig. 4.4 reports the measured average energy consumptions, while Fig. 4.5 shows the corresponding migration-induced per-cent stretching of the average execution times, which is formally defined as: $((SD^{MIG} - 1) \times 100)\ (\%)$. An examination of the bar plots Figs. 4.4 and 4.5 leads to three main conclusions. First, since the dirty rate increases by passing from the (read-intensive) *bzip2* program to the (write-intensive) *memcached* one, the corresponding energy consumptions and execution stretching also exhibit increasing trends under both the Xen and proposed bandwidth managers

(see Figs. 4.4 and 4.5). Second, an examination of the bar plots of Fig. 4.4 shows that the per-cent energy savings of the proposed manager over the Xen one equate 51.1%, 59.2% and 65.4% under the *bzip2*, *mcf* and *memcached* applications, respectively. This confirms the trend of the previous Section 4.9.5 about the larger energy-gains offered by the proposed manager under write-intensive applications. Third, the novel insight which stems from the examination of Fig. 4.5 is that the gaps between the stretching of the execution times of the Xen and proposed bandwidth managers are around 2%, 12%, and 45% under the *bzip2*, *mcf* and *memcached* applications, respectively. This supports the further conclusion that the proposed manager is capable to significantly reduce the (typically, noticeable) stretching of the execution times which is usually experienced when write-intensive applications are migrated.

4.9.7 Comparative tests under ordered migration and real-world trace workloads

Recent contributions point out that, depending on the application to be migrated, some memory pages may be dirtied more frequently than other ones (see [34] and references therein).

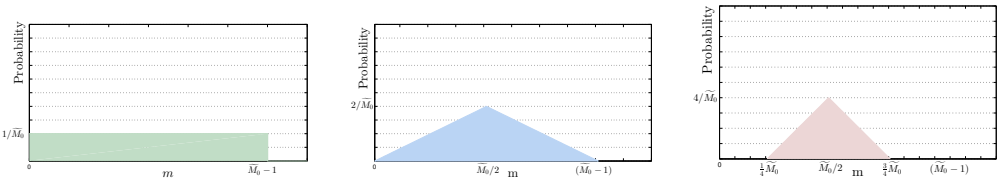
Hence, the goal of a last set of tests is two-fold. First, I investigate how the (possibly, available) knowledge of the spectrum: $\{Pr(m), 0 \leq m \leq \widetilde{M_0} - 1\}$ of the memory change probabilities (MCPs) of the migrating application could be exploited by the proposed bandwidth manager, in order to further reduce the resulting energy consumption. Second, I give insight about the stretching of the total migration time which is induced by the more or less scattered allocation of the memory of the migrating VM over the available address space. For this purpose, both random and ordered migration orderings are tested.

Towards this end, I begin to observe, that the mth MCP: $Pr(m)$ may be profiled at run-time on the basis of the (previously defined) content $\{\chi(m,i)\}$ of the available dirty bitmap as in:

$$Pr(m) = \left(\sum_{i=0}^{I_{MAX}} \chi(m,i) \right) / \left(\sum_{l=0}^{\widetilde{M_0}-1} \sum_{k=0}^{I_{MAX}} \chi(l,k) \right), \; 0 \leq m \leq \widetilde{M_0} - 1. \qquad (4.41)$$

In is enough to implementing Eq. (4.41) at *Dom0* of the test Xen hypervisor, to measured the envelopes of the MCP spectra sketched as in Fig. 4.6 for the (aforementioned) *gap, vortex* and *eon* real-world trace workloads in [64] [5].

[5] I point out that Fig. 4.6 reports the *envelopes* of the (profiled) set of *probabilities* in (4.41) for the tested applications. Hence, according to (4.41), the memory index m in Fig. 4.6 runs over the (integer-valued) interval: $m = 0, 1, \ldots, (\widetilde{M}_0 - 1)$, in order to guarantee that the corresponding summation of the probabilities in (4.41) is unit.



(a) *gap* spectra (b) *envelope* spectra (c) *eon* spectra

Figure 4.6. Envelopes of the profiled MCP spectra of the: (a) *gap*; (b) *vortex*; and, (c) *eon* programs. In all cases, $\widetilde{M}_0 = 42,667$ (*memory page*) and $\overline{w} = 720$ (Mb/s).

Interestingly, the flat envelop of Fig. 4.6(a) reflects the fact that the *gap* application writes at random over the allotted memory space, while the more sharped envelopes of Figs. 4.6(b) and 4.6(c) point out that both the *vortex* and *eon* applications use the central memory space for supporting write-intensive routines. Hence, as pointed out, for example, in [34], I may exploit the knowledge of the (profiled) MCPs in (4.41), in order to migrate the dirtied memory pages with the highest MCPs at the end of the overall migration process.

In order to measure the actually attained energy reduction, I have simulated like at the *Dom0* of the test Xen hypervisor the random migration (RM) and ordered migration (OM) schedulers of Tables 4.3(a) and 4.3(b), respectively. In both cases, the maximum number of migrated memory pages at *round#i* is limited up to (see Eq. (3.4)): $MAX_i \triangleq \lfloor V_i/MSS \rfloor$, and the (possibly) exceeding dirtied memory pages are migrated later [34]. The bar plots of Fig. 4.7 report the measured energy consumptions of the proposed and Xen bandwidth managers under both the RM and OM schedulers.

An examination of these bar plots leads to three main conclusions.

- First, since the envelop of the MCP spectrum in Fig. 4.6(a) of the *gap* program is flat, the resulting energy consumptions of the proposed and Xen bandwidth managers coincide under both RM and OM schedulers (see the pairs of leftmost and rightmost bars marked as *gap* in Fig. 4.7). However, the energy consumption of the proposed bandwidth manager is 65% less than the corresponding one of the Xen manager.

- Second, under the *vortex* application, the energy consumption of the proposed bandwidth manager equipped with the OM scheduler is about 15% lower than the corresponding one of the RM scheduler (compare the two leftmost bars marked as *vortex* in Fig. 4.7). However, the corresponding energy saving attained by the Xen manager is limited up to 10% (compare the two rightmost bars marked as *vortex* in Fig. 4.7).

(a) RM scheduler

$Set: \sum_{m=1}^{\widetilde{M}_0} \chi(m,-1) := \widetilde{M}_0$;
$For\ i = 0\ to\ i = \ (I_{MAX}+1)$
{
$If\ ((\sum_{m=1}^{\widetilde{M}_0} \chi(m,i-1)) > MAX_i)$;
then { *Pick up at random* MAX_i *dirtied pages and migrate them at the planned migration bandwidth* };
else { *Migrate all the dirtied pages at the planned migration bandwidth* };
}.

(b) OM scheduler

$Set: \sum_{m=1}^{\widetilde{M}_0} \chi(m,-1) := \widetilde{M}_0$;
$For\ i = 0\ to\ i = \ (I_{MAX}+1)$
{
$If\ ((\sum_{m=1}^{\widetilde{M}_0} \chi(m,i-1)) > MAX_i)$
then { *Select the* MAX_i *dirtied pages with the lowest MCPs and migrate them at the planned migration bandwidth* };
else { *Migrate all the dirtied pages at the planned migration bandwidth* };
}.

Table 4.4. Pseudo codes of the implemented: (a) RM and (b) OM schedulers.

- Third, under the *eon* application, the energy reduction of the proposed bandwidth manager equipped with the OM scheduler over the corresponding one equipped with the RM scheduler approaches 25% (compare the two rightmost bars marked as *eon* in Fig. 4.7), while the Xen manager limits its energy reduction up to 16% (compare the two leftmost bars marked as *eon* in Fig. 4.7).

This trend is confirmed by the bar plots of Fig. 4.8. They report the corresponding measured total migration times of the proposed bandwidth manager under the RM and OM scheduling disciplines.

An examination of the bars marked as RM in Fig. 4.8 points out, under the RM scheduling, the total migration time (somewhat) decreases by passing from the *gap* application to the *eon* one. This behavior is aligned with the fact that the memory allocation of the *gap* application is sparser than the corresponding one of the *eon* application (see Fig. 4.6). However, the resulting gap among the leftmost

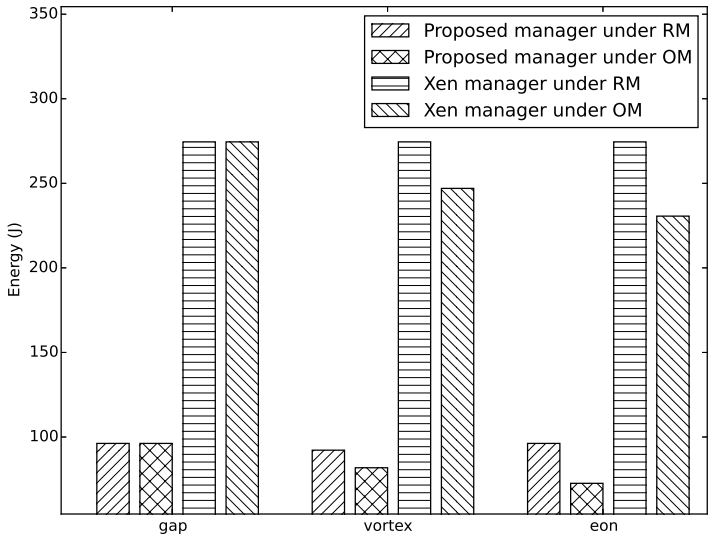


Figure 4.7. Energy consumptions for the application scenarios of Section 4.9.7 at: $M_0 = 512\ (Mb)$, $\overline{w} = 720\ (Mb/s)$, $\widehat{R} \equiv R_{MAX}^{XEN} = 900\ (Mb/s)$, $I_{MAX}^{XEN} = 29$.

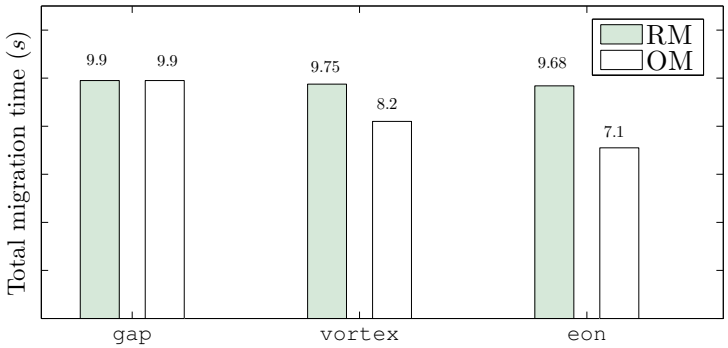


Figure 4.8. Total migration times of the proposed bandwidth manager for application scenario of Section 4.9.7 at $\Delta_{SC} = 0.02\ (s)$ and $\mathcal{E}_{TOT}^{*} = 97\ (J)$.

and rightmost bars marked as RM of the Fig. 4.8 is not so substantial and it is limited up to 2.2%. This conclusion must be, indeed, reconsidered when the OM scheduling discipline is applied. In fact, a comparison of the leftmost and rightmost bars marked as OM in Fig. 4.8 unveils that gap among the corresponding total migration times substantially increases and approaches 28%.

Overall, the net conclusion which stems from the plots of Figs. 4.7 and 4.8 is that the proposed bandwidth manager is capable to effectively exploit the (possibly, available) *memory change probabilities*, in order to reduce both the *consumed energy* and the *total migration time* of applications with "picked" memory allocations.

4.10 Conclusion to Part I

In this chapter I developed the optimal bandwidth manager for intra-data-center live VM migration. It minimizes at run-time the communication energy wasted by the migration of the VM memory under hard QoS constraints on both the migration time and downtime. A complete conclusion resume for this is provided in Chapter 7.

Hence, the following Part II shows an implementation of my bandwidth manager for wireless live migration in 5G context application.

Part II

Part 2: Bandwidth manager for wireless 5G networks

CHAPTER 5

LIVE MIGRATION IN WIRELESS FOG COMPUTING FOR 5G NETWORKS

"5G is considered key to the Internet of Things. Billions of sensors will be built into appliances, security systems, health monitors, door locks, cars and wearables – from smart-watches to dog collars. Analyst firm Gartner predicts the number of networked devices will skyrocket from about 5 billion in 2015 to 25 billion by 2020.
All those sensors producing mountains of data should, in turn, spur carriers to spend billions upgrading their networks for 5G."

- Stephen Shankland, *CNET*

In the above chapters, Part I, I investigated main aspects of live VMs migration which are directly involved in management of migration bandwidth, then I provided a solution and demonstrate how my manager guarantee optimal performances for intra data center live migration. Here, in Part II, I want to show how my approach could be applied in wireless live migration, in particular in 5G technologies, to improve significantly the performances. Hence, I show the performance of my approach and I compare it with some of the main solution in literature.

Live virtual machine migration aims at enabling the dynamic balanced use of the networking/computing physical resources, so to lead to reduced energy consumption. In this chapter I analytically characterize and test an optimal bandwidth manager for live migration of VMs in wireless channel. In the following Section 5.3, I present the optimal tunable-complexity bandwidth manager (TCBM) for the QoS live migration of VMs under a wireless channel from smart phone to access point. The goal of my approach is to minimize the migration-induced communication energy under service level agreement *hard* constrains on the *total migration time*, *downtime* and *overall available bandwidth*.

In the Chapter 3 I did not introduced the Part II related work to distinguish between the two application's scenarios. Hence, in the following, Section 5.1, I provide the related work for wireless channel live migration bandwidth manager. In Section 5.1 I introduce the state-of-art for technologies in fifth-generation (5G) mobile network systems and Fog computing (also called Edge Computing) which allow application's migration via wireless channel.

In Fig. 5.1 I illustrate the working application of TCBM wireless live migration bandwidth manager. In 5G networks a fundamental issue is to provide services with low latency (1 ms) and high bandwidth capacity, then Fog computing, also termed *edge computing*, can address those problems by providing elastic resources and services to end users at the edge of network.

Fog computing and a new air interface are the key factor to enable low latency in future 5G systems. Therefore, I deign a three tiers architecture, as show Fig. 5.1, Mobile VMs tier, Fog sites tier and Cloud Data Center tier. The difference between Fog computing and cloud computing are that the first enables computation capability to the edge of the network, instead, cloud computing provides resources, with high computation capabilities, distributed on the core network.

The most important Internet companies in the world (e.g. Amazon, Google, Microsoft, etc.) are investing on huge data centers infrastructure all over the world with incredible capacity, but at the same time most important networks operator (e.g. TIM, AT&T, and so on) are building network infrastructures with growing computation and storage capabilities in middle of their networks infrastructure.

Fig. 5.1 show the designed architecture for this chapter, the mobile physical hosts usually is a smart phone, but could be considered a sensor or a plethora of innovative devices growing in Internet of Things (IoT), or a more powerful hardware devices like laptop, smart TV, and so on. However, all of them are wireless connected devices with growing computation requirements.

There are three layers,

- the first, yellow domain, for mobile devices which have limited computing resources and available energy;
- the second is the Fog sites, in green color, with high *responsiveness* but limited *computing/storage* capacity: base-station, access-point or road-side-unit (RSU) if we consider vehicular networks;
- and finally, the red one, are the data centers infrastructures with remarkable computation/storage and energy capability.

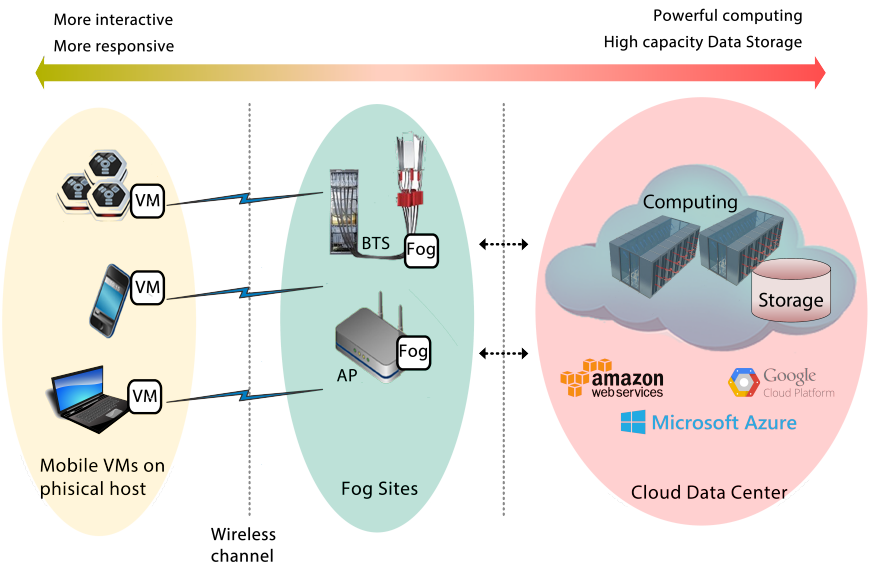


Figure 5.1. Prototype architecture. Mobile virtualized devices, Fog sites and cloud data centers.

5.1 Related work and reference architecture

Mobile cloud computing (MCC) emerging in the context of 5G has the potential to overcome resource limitation in the mobile devices (appear as a bottleneck in 5G applications), which enables many resource-intensive services for mobile users with the support of mobile big data delivery and cloud-assisted computing [3]. In 5G a fundamental issue is to provide services with low latency, and Fog computing (FC) can address those problems by providing elastic resources and services to end users at the edge of the network.

In the following I illustrate the main technologies that allow the application's migration or parts of them from mobile device to a remote site, in which their capabilities increase. Usually, they use this feature in order to reduce energy consumption and take advantage of a computing power of remote resource, greater than that it has available locally.

Here a list of solutions investigated, followed by a description for each of them:

- CloneCloud [67, 68]: is a system that has the ability to automatically transform mobile device application in such a way that they can run into the cloud;
- VOLARE [69]: is a middelware-based solution which allows context-aware adaptive cloud service discovery for the mobile devices.

- Cuckoo [70]: is a computational offloading framework for mobile devices;
- Cloudlet [71]: is a set of widely dispersed and decentralized Internet infrastructure components, with non-trivial characteristic to make available for the nearby mobile devices computing resource and storage resources;
- MAUI [72]: is a system that is able to minimize the energy due to the VM migration by means of fine-grained offloading.

CloneCloud

CloneCloud [67] is a system that has the ability to automatically transform mobile applications in such a way that they can turn on the cloud. Even if mobile applications are design to be performed on the smart phone, or as a client-server model, CloneCloud is designed so that it can offload part of the execution on cloud resources. In Fig. 5.2 from authors paper [67] is shown the CloneCloud prototype architecture.

The feature that this system automatically transforms applications and optimizes the distributed execution, according to the capacity of the device and the cloud, ensures that developers can design their applications without be aware if they can be migrated or not.

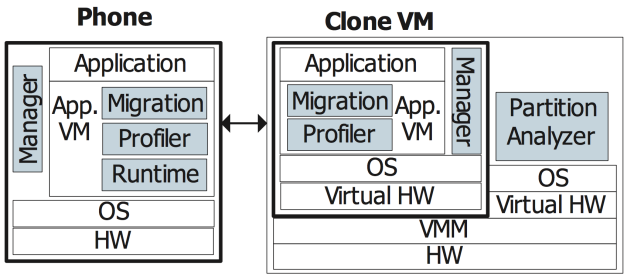


Figure 5.2. The CloneCloud prototype architecture [67].

To split parts of an application to run locally and remotely, the system uses an off-line partitioning mechanism, which works without any source code or any special characters from the local application.

The Fig. 5.3 shows that the partitioning framework is composed by:

- *Static Analyzer*, which analyzes the application taking into account a variety of constraints, results in a list of the parts of the application that can be migrated.

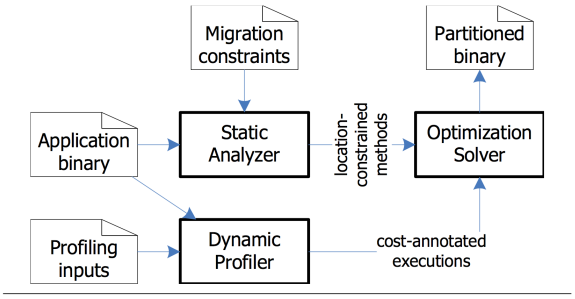


Figure 5.3. Structure of the partitioning framework of CloneCloud, as in paper [67].

- *Dynamic Profiler*, which acquires information about the execution time and the energy cost, in order to giving rise to a cost model.
- *Optimization Solver*, it receives as input both the result of the *static analyzer* that of the *dynamic profiler*, to choose which parts must be performed locally and which in remote.

Fig. 5.4 shows how it works the the migration process for a *thread* through CloneCloud.

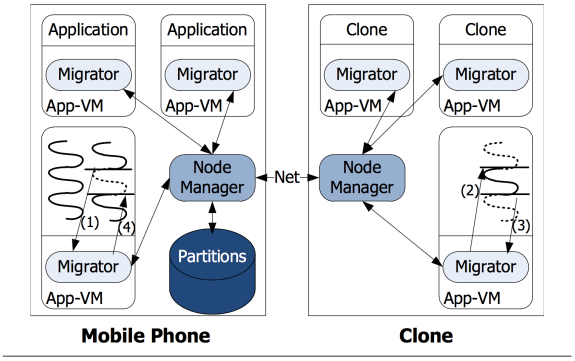


Figure 5.4. Example of migration process in CloneCloud, from [67].

VOLARE

VOLARE [69] is a middleware-based solution which allows context-aware adaptive cloud service discovery for smart phones. In this scenario, cloud service requests are dynamically adapts going to monitor resources and the status of the devices.

The middlware of VOLARE is composed by two level,

- *service discovery time* level

- and *runtime* level.

In particular we have that, while at *service discovery time* VOLARE is responsible to intercept requests from mobile devices, at *runtime* it continue to control the cloud bindings and the context of the device, so that if there are any kind of change, VOLARE reacts going to look for that service which coincides with the requirements and starts rebinding.

VOLARE's architecture is composed by five modules:

1. Context monitoring module,
2. Adaptation module,
3. Service request module,
4. QoS monitoring module,
5. Service binding module.

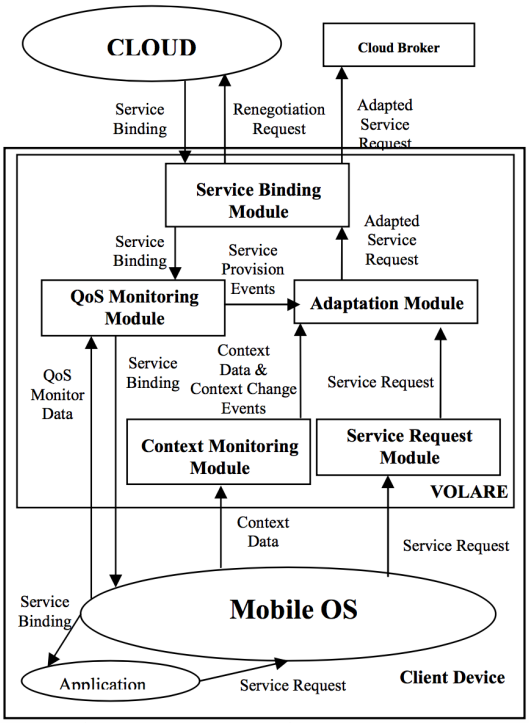


Figure 5.5. Architecture of VOLARE, from [69].

Cuckoo

Cuckoo [70] is a computation offloading framework for mobile devices, currently implemented only for Android OS. Given that the computing resources of the cloud

are not always available when required, Cuckoo has been designed specifically to solve this problem. At this regard it is had that this framework has the ability to support both local and remote execution of applications, in such a way that even when you can not make use of the cloud, applications continue to work. Fig. 5.6 show an overview of Cuckoo's Android IPC mechanism.

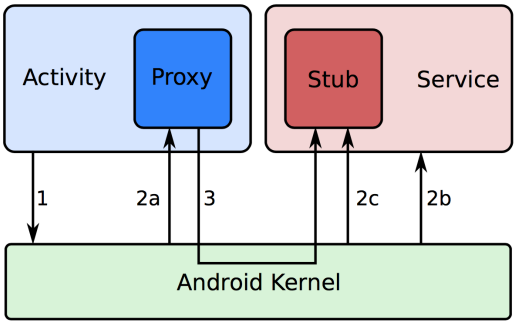


Figure 5.6. An overview of the Android IPC mechanism of Cuckoo. An activity binds to a service *(1)*, then it gets a proxy object back from the kernel *(2a)*, while the kernel sets up the service *(2b)* containing the stub of the service *(2c)*. Subsequent method invocations by the activity on the proxy *(3)* will be routed by the kernel to the stub, which contains the actual implementation of the methods [70].

Cuckoo was implemented based on the server-client model. The server has the ability to be able to run on every possible resource that has a JVM (Java Virtual Machine) installed. The services provided by the Client can be uploaded on the server so that later they can be run remotely. Once the service is loaded on the server, this can be used indefinitely.

Cloudlet

Cloudlet is an architecture for mobility-enhanced using small cloud data center that is located at the edge of the Internet. Fig. 5.7 represents a possible realization of Cloudlet [71], which can be defined as a set of widely dispersed and decentralized Internet infrastructure components, with non-trivial characteristic to make available to its computing resources and storage at the service of nearby devices. Looking more specifically, a Cloudlet can be seen as a *data-center in a box*, able to manage themselves, require little power, access control for setup and Internet connectivity. Its easy implementation allows its use in different environments, like the office of a doctor or any commercial space. Observing the Cloudlet inside, you can see that is very similar to a cluster of multi-core computers.

In order to integrate Cloudlet with WiFi, is presented a solution called *transient customization of Cloudlet infrastructure*, which makes use of hardware VM

Figure 5.7. Cloudlet concept, discussed from Satyanarayanan et al. in [71].

technology.

MAUI

MAUI [72] is a system that enables fine-grained energy-aware offload of mobile code to the infrastructure. This solution could be used in order to minimize the energy consumption of mobile devices. In particular, this is a system that is able to minimize the energy due to the VM migration by means of fine-grained offloading. This program is made so as to only decide at runtime what methods run remotely; in this operation takes into account the constraints imposed by the connectivity of smart phones in order to have the greatest possible energy savings.

Thanks to the portability of the code, MAUI made two versions of a smart phone application, one of which is run locally on the smart phone, while the other is processed in the remote infrastructure.

Fig. 5.8 show the high level architecture of MAUI system. Regarding the smart phone side, MAUI is composed by three modules, namely *Solver*, *Client Proxy* and *Profiler*; while at server side MAUI contains four modules that are *Solver*, *Server Proxy*, *Profiler* and *Controller*.

My TCBM System's architecture

In Fig. 5.9 I show a schema of system's architecture in which I apply tunable complexity bandwidth manger. In the schema are distinguished three domains: the mobile devices, Fog sites, and cloud data centers domains. Hence, very reactive-constrained applications could be retrieved in near-domain Fog site, but to obtain high performance in terms of computation or storage it is necessary to reach a cloud

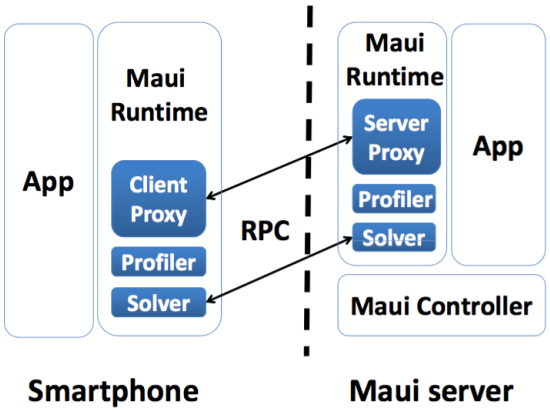


Figure 5.8. High level view of the architecture of Maui, illustrated by Cuervo et al. in [72].

data center structure. A common solution to provide high computation capabilities for applications/VMs on mobile devices is to be migrated from the mobile domain to the Fog site. For instance, in Fig. 5.9 I show a virtual machine that is cloned inside the Fog site through the wireless interface, this will enable the cloned virtual machine to use Fog site's resources without drain battery energy of mobile device.

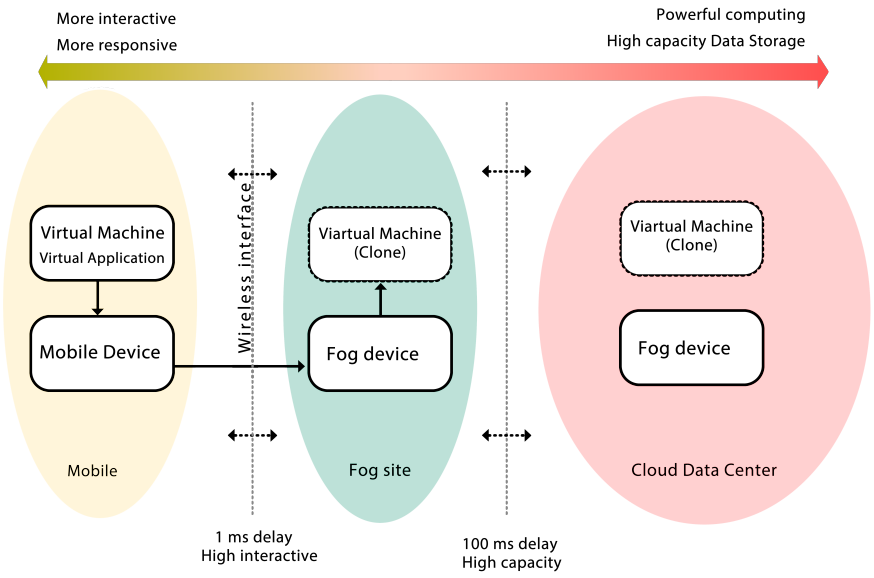


Figure 5.9. An overview of architectural structure of the system.

I already introduced live migration in the above pages, see Chapter 2, hence, this chapter is organized as follows: in Section 5.3 introduces our tunable-complexity

bandwidth manager, formulation and solution of the non-convex optimization problem. Section 5.7 shows our tests and results on this approach. Finally, I present my conclusion to the Part II of thesis in Section 5.8.

5.1.1 Current approach for bandwidth migration and future applications

In [30] Clark et al. wrote important consideration in literature about live migration of virtual machines. They demonstrate that the migration of entire OS instances on a commodity cluster work with impressive performance and minimal downtimes (as low as $60ms$). And they demonstrate that live migration performances are sufficient to make it a practical tool for servers running interactive loads. The challenge for the next few years will be directed to the new technologies oriented to the wireless connections with low latency (1 ms) with high capacity.

In the following, for clarity, I need to use some recalled equations and figures from previous chapters in which I considered the intra-data-center channel optimization bandwidth problem.

As already shown in Chapter 2, in literature there are four main techniques for VM migration, namely, stop-and-copy migration (SaCM), pre-copy migration (PeCM), post-copy migration (PoCM) and hybrid migration (HyBM). They trade-off the total migration time and downtime. In my work I used a pre-copy (*push*) approach, and on the basis literature references I eschew a post-copy (*pull*) approach which faults in missing pages across the network since this adds a residual dependency of arbitrarily long duration, as well as providing in general rather poor performance.

Considering the related work, at this time there are not works considering the bandwidth management during the VMs live migration for wireless channel. This thesis is the first which considers the bandwidth management both in wired network environment and wireless networks (in fifth generation network). As described in Chapter 4, this approach is capable to effectively filter out transient fluctuations of the average resource utilization and avoid needless migrations [22].

In the following Section 5.3 I describe the implementation of a tunable-complexity bandwidth manager for live migration in wireless channel.

5.2 Trade-off between migrate and not migrate to the Fog site

In Section 3.3 I already defined a model for times, network energy and data, in the following, after a short recall of some formula I model and compare the trade-of between run the VM application locally and migrate the VM to a remote Fog site.

In formula, as already shown, the total migration time $T_{TOT}(s)$ is the overall duration:

$$T_{TOT} \triangleq T_{PM} + T_{RE} + T_{IP} + T_{SC} + T_{CM} + T_{AT}, \tag{5.1}$$

of the six stages, while the downtime:

$$T_{DT} \triangleq T_{SC} + T_{CM} + T_{AT}, \tag{5.2}$$

is the time required for the execution of the last three stages.

Memory migration time T_{MMT}, which is defined as in:

$$T_{MMT} \equiv T_{MMT}(R) \triangleq T_{IP}(R) + T_{SC}(R). \tag{5.3}$$

Hence, T_{MMT} is the time needed for completing the memory transferring of the migrating VM.

Directly from the reported definitions I have:

$$V_i \triangleq \overline{w} T_{i-1} \qquad i = 1, ..., (I_{MAX} + 1), \tag{5.4}$$

with $V_0 \equiv M_0$, and

$$T_i \triangleq \frac{V_i}{R_i} \equiv \frac{\overline{w}}{R_i} T_{i-1} = M_0 \overline{w}^i \prod_{m=0}^{i} R_m^{-1} \quad i = 0, ..., (I_{MAX} + 1), \tag{5.5}$$

with $T_0 \equiv \dfrac{M_0}{R_0}$, so that I also have

$$T_{MMT}(R) \equiv \sum_{i=0}^{I_{MAX}+1} T_i = M_0 \Big[\sum_{i=0}^{I_{MAX}+1} \overline{w}^i \Big(\prod_{l=0}^{i} R_l^{-1} \Big) \Big], \tag{5.6}$$

and (see (Eq. 5.5))

$$T_{SC}(R) \equiv T_{I_{MAX}+1} = M_0 \overline{w}^{I_{MAX}+1} \prod_{m=0}^{I_{MAX}+1} R_m^{-1}. \tag{5.7}$$

The power (measured in Watt (W)) drawn by a physical network interface card (NIC) consists of a static (e.g., setup) portion and a dynamic portion, see Section 3.3.2 for other consideration on network energy consumption.

$$P_{DYN}(R) = K_0 (R)^{\alpha}, \tag{5.8}$$

where

$$K_0 \triangleq (1/g)(RTT/1.22\, MSS)^{\alpha}\ ,\ ((W)\ \times\ (s/Mb)^{\alpha}). \tag{5.9}$$

In Eq. (5.9), RTT (s) is the average round-trip-time of the available end-to-end connection, $g((W)^{-1})$ is the coding gain-to-receive noise power ratio, $\alpha > 1$ is a (dimension-less) shaping factor and MSS (Mb) is the maximum size of the utilized TCP segments [45].

Therefore, since the dynamic energy $\mathcal{E}_i(J)$ consumed during the ith round equates the product: $\mathcal{E}_i \equiv \mathcal{E}_i(R) \triangleq P_{DYN}(R)\, T_i, i = 0, ..., (IMAX + 1)$, by summing these products over the round index, I obtain the following expression for the total communication energy $\mathcal{E}_{TOT}(J)$ consumed during the migration process:

$$\mathcal{E}_{TOT} \equiv \mathcal{E}_{SETUP} + K_0 M_0 R_0^{\alpha-1} + \theta\Bigg\{ K_0 M_0 (\overline{w})^{I_{MAX}+1} (R_{I_{MAX}+1})^{\alpha-1} \Bigg[\prod_{k=0}^{I_{MAX}} (R_K)^{-1} \Bigg] + (1 - \delta(I_{MAX})) \Bigg\{ \sum_{l=1}^{I_{MAX}} K_0 M_0 (\overline{w})^l (R_l)^{\alpha-1} \Bigg[\prod_{m=0}^{l-1} (R_m)^{-1} \Bigg] \Bigg\} \Bigg\} \tag{5.10}$$

with $\mathcal{E}_{SETUP}$ (J) which accounts for the static portion of $\mathcal{E}_{TOT}$.

Likewise, by summing the expressions in (5.4) over the round index, and taking into account the equation (5.5), I obtain the following closed-form formula for the total volume V_{TOT} (Mb) of the migrated data:

$$V_{TOT} \triangleq \sum_{i=0}^{I_{MAX}} V_i = M_0 \Bigg[1 + \theta \Bigg(\sum_{i=1}^{I_{MAX}} \overline{w}^i \prod_{m=0}^{i} R_m^{-1} \Bigg) \Bigg] \tag{5.11}$$

Following to the aforementioned network energy model, and looking to the architectural overview as in Fig. 5.9, which one is the best choose between migrate and not migrate the VM to the Fog site?

Purpose of this section is to illustrate in which cases is convenient to run execution on mobile devices (e.g. smart phone) and in which case it is more appropriate offloading part of application from mobile device to Fog site (or cloud), in order to improving performance and saving energy of the mobile device. In particular migration does not appears to be always the best choice [73], as will be show below.

Without loss of generality, an application can be divided into two parts, one of which is performed on the mobile system, while the other may be offloaded to be executed in the remote domain (Fog site/cloud). I denote by s_m the speed of the mobile device and with w the amount of computation for the second part. The amount of time employed for the execution of w is equal to:

$$T_1 = \frac{w}{s_m}. \tag{5.12}$$

In case of that the second part is offloaded to the cloud, the input data d_d is

sent to the remote domain at bandwidth B in $\frac{d_d}{B}$ (s) seconds. To simplify the calculations, I avoid to consider both the time of initial network's setup and the time needed to transfer the program to the server.

Indicating with s_s the speed of the server, the time for offloading and execute the second part of the main program is equal to:

$$T_2 = \frac{d_d}{B} + \frac{w}{s_s}. \tag{5.13}$$

Observing the formulas (5.12) and (5.13) leads to the first conclusion that the use of virtualization is convenient for the purposes of increasing performance in case occurs that $T_1 > T_2$:

$$\frac{w}{s_m} > \frac{d_d}{B} + \frac{w}{s_s} \Longrightarrow w \times \left(\frac{1}{s_m} - \frac{1}{s_s}\right) > \frac{d_d}{B} \tag{5.14}$$

This formula lends itself to some considerations:

- if w is big, the program requires hard computation;
- if s_s is large, the server is very fast;
- if d_d is small, a small amount of data is considered;
- if B is large, the bandwidth is large.

In summary, it is that even if the server is very fast, does not seem to be convenient to migrate in terms of performance until $\frac{w}{s_m} < \frac{d_d}{B}$.

Furthermore to increasing the performances, use the technique of migration could lead to significant energy savings, because it could save you a lot of mobile device energy battery going to run programs with high energy consumption directly to the cloud domain. Obviously, also in this case, it is not always convenient to migrate from mobile system to server.

Let p_m be the power of the mobile device. Modifying Eq. (5.12), it can be obtained the energy to perform task, equal to:

$$E_1 = p_m \frac{w}{s_m} \tag{5.15}$$

Suppose p_c is the power used to transfer data on the network. After sending the data, the system listen on the network interface and stay in a state of waiting until the server returns the result of the offloaded computation. In this *idle* state, the mobile device consumes a power p_i, and then a power consumption equal to:

$$E_2 = p_i \times \frac{d_d}{B} + p_c \times \frac{w}{s_s} \tag{5.16}$$

Comparing the Eq. (5.15) and Eq. (5.16), it is concluded that migration allows energy savings if $E_1 > E_2$:

$$p_m \frac{w}{s_m} > p_i \times \frac{d_d}{B} + p_c \times \frac{w}{s_s} \Longrightarrow w \times \left(\frac{p_m}{s_m} - \frac{p_i}{s_s}\right) > p_i \times \frac{d_d}{B}. \tag{5.17}$$

The Eqs. (5.14) and (5.17), respectively used to ensure high performance and energy saving during the phase of offloading, are very similar. In particular, use of offloading to save energy is useful in case of large w (heavy computation) and small d_d (light communication).

In both cases (Eq. (5.14) and Eq. (5.17)), have bandwidth B very higher ease the satisfaction of the constraints in such a way as to make convenient migrate to improve performance and save energy.

Table 5.1. Measurements of energy consumption for different part of the device Nokia N95 smart-phone, for wireless data, see [74] by Perrucci et al..

Technology	**Action**	**Power [mW]**	**Energy [J]**
Bluetooth	BT off	12	–
	BT on	15	–
	BT connected and idle	67	–
	BT discovery	223	–
	BT receiving	425	–
	BT sending	432	–
WiFi (infrastructure mode)	In connection	868	8.2
	In disconnection	135	0.4
	Idle	58	–
	Idle in power save mode	26	–
	Downloading@4.5Mbps	1450	–
WiFi (ad hoc)	Sending@700 kB/s	1629	–
	Receiving	1375	–
	Idle	979	–
2G	Downloading@44Kbps	500	–
	Handover 2G $--> $ 3G	1389	2.4
3G	Downloading@1Mbps	1400	–
	Handover 3G $--> $ 2G	591	2.5

The values of power and energy shown in the Table 5.1 are average values, since they were obtained by repeating several times the experiments. In particular, for Bluetooth they have high power consumption in event that the smart-phone trying to discover another device, or if try to send or receive data. For Bluetooth sending data is the more expensive operation, in therms of power consumption. With regard to the WiFi (in infrastructure mode), the operations that imply the greatest waste of power are obtained in case of connection (turn on WiFi and connect to AP) and

download, while there is a small waste of energy in case of connection, and a lower waste of energy in case of disconnection (turned off WiFi and disconnect from AP). In case of WiFi in ad-hoc mode, two equals smart phone *NOKIA N95* are used. In this case, there is not none energy consumption but the power consumption increases significantly with respect the scenario WiFi in infrastructure mode. For other results and further considerations see [74].

5.3 Tunable complexity bandwidth manager, definition and properties

In this section I introduce the tunable complexity bandwidth management (TCBM). Let I_{max} be the number of performed pre-copy rounds, as described in Section 3.2.

A primary goal of our work is to formal define a model overview of how wireless live migration works. Here I recall some of the formula and concept already defined in the Chapter 2 about how live migration does works.

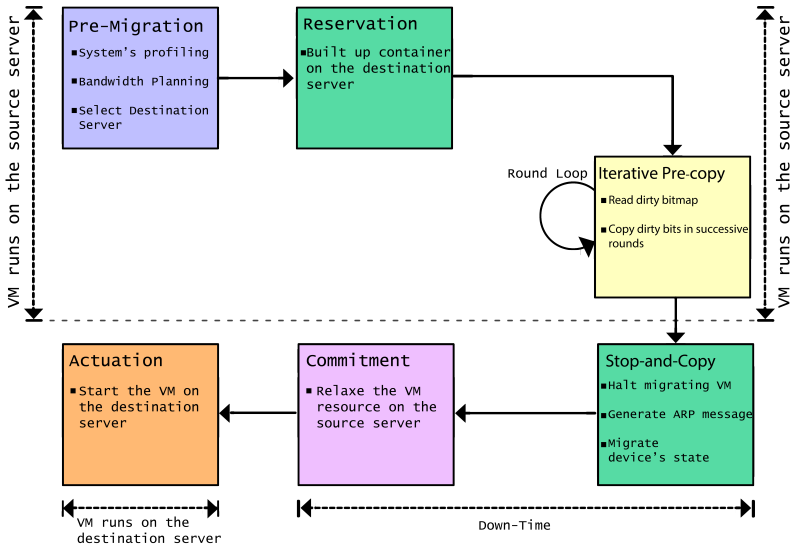


Figure 5.10. Pre-copy live migration stages (six stages). Recalled from Chapter 3.

Most important variables are the total migration time T_{TOT}, T_{DT} and T_{TMT}, as recalled in the previous section and explained in Chapter 2, see also Fig. 5.10 (already shown in Section 3.2 as Fig. 3.6) in which I provide a graphical illustration of live migration stages.

$$\{ R_i,\ 0 \leq i \leq I_{MAX} + 1 \} \tag{5.18}$$

Let R_i (Mb/s) be the transmission rate used during the third and fourth stages at the ith round for migrating the VM, that is, the migration bandwidth. However, I presented a first formulation of the problem considering an optimized but constant

rate for all the rounds, $R_i = R\,\forall i$ in Part I. Since, by definition, only T_{IP} and T_{SC} depend on R, while all the remaining migration times in T_{TOT} and T_{DT} play the role of constant parameters, in the sequel, I focus on the evaluation of the (already defined) stop-and-copy time T_{SC} and the resulting memory migration time T_{MMT}.

Since the PeCM technique performs the iterative pre-copy of dirtied memory bits over consecutive rounds, let V_i (Mb) and T_i (s), $i = 0, \ldots, (I_{MAX} + 1)$, be the volume of the migrated data and the time duration of the ith round, respectively. By definition, V_0 and T_0 are the memory size M_0 (Mb) of the migrating VM and the time needed for migrating it during the $0th$ round, respectively.

Hence, I formalize the tackled tunable-complexity bandwidth manager, as you can see in Fig. 5.11. In addition to R_0 and $R_{I_{MAX}+1}$ I have Q updated rounds, which is the number of updated rates with my manager. Then I update Q out of I_{MAX} rates of the pre-copy rounds evenly spaced by $S \triangleq \dfrac{I_{MAX}}{Q}$ over the round-index set $\{1, 2, 3, \ldots, I_{MAX}\}$.

For this purpose, I perform the partition of the round index set $\{1, 2, 3, \ldots, I_{MAX}\}$ into Q not overlapping contiguous subsets of size S.

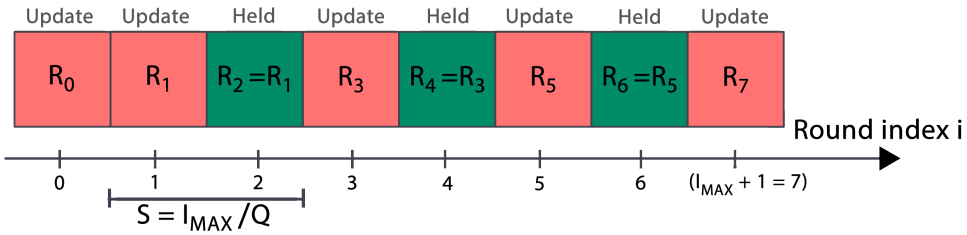


Figure 5.11. Reference framework for the tunable-complexity bandwidth manager. Case of $I_{MAX} = 6$, $Q = 3$. The rates to be uploaded are: R_0, R_1, R_3, R_5 and R_7. The rates to be held are: $R_2 \equiv R_1$; $R_4 \equiv R_3$; $R_6 \equiv R_5$.

The fist rate R_{jS+1} , $j = 0, \ldots, (Q-1)$ of each subset is updated, while the remaining $(S-1)$ rates are set to R_{jS+1}, that is $R_i \equiv R_{jS+1}$, for $i = jS + 2$, $jS + 3, \ldots, (j+1)S$.

Fig. 5.11 illustrates the framework of the updated and held migration rates for the dummy case of $Q = 3$ and $I_{MAX} = 6$. In this case, the rates R_0, R_1, R_3, R_5 and the rate R_7 are the $Q + 2 = 5$ migration rates to be updated, while rates R_2, R_4 and rate R_6 are the $(I_{MAX} - Q) = 3$ migration rates which are not updated and, by definition, they equate: $R_2 \equiv R_1$; $R_4 \equiv R_3$; $R_6 \equiv R_5$.

5.4 Definition and expression of TCBM for QoS live migration of VMs

In order to formally introduce the TCBM, let I_{MAX} be the number of performed pre-copy rounds, so that the overall set of $(I_{MAX}+2)$ migration rates is the (usual) one (see Fig. 5.11)

Let Q be the integer-valued number of the pre-copy migration rates we decide to update and let $S \triangleq \frac{I_{MAX}}{Q}$ the resulting integer-valued size of the rate clusters which assume the same values (see Fig. 5.11).

Formally speaking, Q and S are to be selected according to the following formal rules:

(i) if $I_{MAX} \geq 1$, Q must be integer-valued and falling into the interval:

$$1 \leq Q \leq I_{MAX} \tag{5.19}$$

Furthermore, Q must be selected so that the resulting ratio:

$$S \triangleq \frac{I_{MAX}}{Q} \tag{5.20}$$

is also integer-valued;

(ii) if $I_{MAX} = 0$, the set $\{ R_1, R_2, \ldots, R_{I_{MAX}} \}$ of the pre-copy rates is the empty one, so that we must pose:

$$Q = 1 \; and \; S = 0; \tag{5.21}$$

(iii) if $I_{MAX} = -1$ *and* $\theta = 0$ *(case of Stop-and-Copy only)*, we must optimize only R_0, so that we must still pose:

$$Q = 1 \; and \; S = 0; \tag{5.22}$$

If the above assumptions (5.19)-(5.22) be met, the TCBM is formally defined as follows:

(i) it updates the following set of $(Q+2)$ migration rates:

$$\Xi \triangleq \{ R_0; R_{jS+1}, \; j = 0, 1, \ldots, (Q-1); \; R_{I_{MAX}+1} \}; \tag{5.23}$$

(ii) it sets the remaining $(I_{MAX} - Q)$ migration rates as in:

$$R_{jS+1}, \; for \; i = (jS+2), \; (jS+3), \; \ldots, \; (j+1)S, \tag{5.24}$$

for any assigned $j = 0, 1, 2, \ldots, (Q-1)$.

Hence, the TCBM partition the set:

$$\{1,\ 2,\ \ldots,\ I_{MAX}\} \tag{5.25}$$

of the migration rates which compose the pre-copy phase into Q not overlapping clusters of S contiguous rates.

Afterwards, the TCBM updates the first rate of each cluster (e.g., the cluster header: R_{jS+1}; see Eq. (5.23)), and sets all the remaining $(S-1)$ rates of each cluster to the cluster header (see Eq. (5.24)). Finally, the TCBM updates R_0 and $R_{I_{MAX}+1}$, that is, the first and last migration rates (see Eq. (5.23)). Overall, the TCBM updates only $(Q+2)$ migration rates out of the available $(I_{MAX}+2)$ and, the furthermore, the updated intermediate Q rates are evenly spaced apart of $S \triangleq \dfrac{I_{MAX}}{Q}$ steps over the set in (5.25) of the pre-copy round indexes.

5.4.1 Expression of the downtime for the TCBM

The downtime is the time required for the execution of the last three stages of pre-copy live VM migration widely discussed in subsection 3.2, that are **Stop-and-copy, Commitment** and **Re-activation**. In particular, downtime is the time that elapses between the stop of the service on the mobile device and the reactivation of the VM at the destination host.

In the stop-and-copy case (e.g., $I_{MAX} = -1\ and\ \theta = 0$) we still have $T_{DT} = \dfrac{M_0}{R_0}$.

However, at $I_{MAX} \geq 0$ and $\theta = 1$, by introducing the definition positions (5.23)-(5.24) in the general expression of the downtime

$$\begin{aligned}
//alsoeq : n28 T_{DT} \triangleq T_{I_{MAX}+1} &\equiv M_0\, C_{I_{MAX}+1} \left(\prod_{k=0}^{(I_{MAX}+1)} R_k^{-1} \right) = \\
&= M_0\, (\overline{w})^{I_{MAX}+1}\, \frac{1}{R_0\, R_{I_{MAX}+1}} \left(\prod_{k=1}^{I_{MAX}} R_k^{-1} \right) = \\
&= M_0\, (\overline{w})^{I_{MAX}+1}\, \frac{1}{R_0\, R_{I_{MAX}+1}} \left[\prod_{j=0}^{Q-1} \left(\prod_{k=jS+1}^{(j+1)S} (R_{jS+1})^{-1} \right) \right] = \\
&= M_0\, (\overline{w})^{I_{MAX}+1}\, \frac{1}{R_0\, R_{I_{MAX}+1}} \left[\prod_{k=0}^{Q-1} \left(\frac{1}{R_{kS+1}} \right)^{S} \right].
\end{aligned} \tag{5.26}$$

Overall I have:

$$T_{DT} = M_0 \Bigg\{ \frac{1}{R_0}\delta(1+I_{MAX}) + \Bigg\{ (\overline{w})^{(1+I_{MAX})} \Bigg(\frac{1}{R_0\, R_{I_{MAX}+1}} \Bigg) \Bigg[\prod_{k=0}^{Q-1} \Bigg(\frac{1}{R_{kS+1}} \Bigg)^{S} \Bigg] \Bigg\} * \\ * \Big(1 - \delta(1+I_{MAX}) \Big) \Bigg\}, \tag{5.27}$$

where δ is the delta of Dirac function, that is equal to one only when the argument is zero, while is equal to zero in the other cases.

The above expression depends only on the $(Q+2)$ cluster headers in Eq. (5.23).

5.4.2 Expression of the total migration time for the TCBM

The total migration time represent the sum of all the six stages of the PeCM technique (see subsection 3.2). In particular it is obtained by adding the duration of all $(I_{MAX}+2)$ rounds (as you can see in Fig. 3.7). Since the constraint on the total migration time is present only for $\theta = 1$, I may directly consider the case of $I_{MAX} \geq 0$.

The general expression is:

$$T_{TM} \triangleq M_0 \sum_{l=0}^{I_{MAX}+1} (\overline{w})^l \Bigg(\prod_{m=0}^{l} \Bigg(\frac{1}{R_m} \Bigg) \Bigg), \tag{5.28}$$

while the expression used for the TCBM is

$$T_{TM} = M_0 \Bigg\{ \frac{1}{R_0} + (\overline{w})^{1+I_{MAX}}\, (R_0\, R_{I_{MAX}+1})^{-1} \Bigg[\prod_{m=0}^{Q-1} (R_{mS+1})^{-S} \Bigg] + \\ + (1-\delta(I_{MAX})) \frac{1}{R_0} \Bigg\{ \sum_{k=0}^{Q-1} \sum_{l=kS+1}^{(k+1)S} (\overline{w})^l \Bigg\{ \delta(k)(R_1)^{-l} + \\ + (1-\delta(k)) \Bigg[\prod_{p=0}^{k-1} (R_{pS+1})^{-S} \Bigg] (R_{kS+1})^{-l+kS} \Bigg\} \Bigg\} \Bigg\} \tag{5.29}$$

Also in this case, the above expression depends only on the $(Q+2)$ cluster headers in Eq. (5.23) to be optimized. The steps that lead to the formula Eq. (5.29) are shown in Appendix A.4.

5.4.3 Expression of the energy wasted by the TCBM

In order to express the energy $\mathcal{E}_{TOT}$ wasted by the TCBM as a function of the $(Q+2)$ cluster headers in Eq. (5.23), we begin to rewrite the general expression

$$\mathcal{E}_{TOT} = \mathcal{E}_{SETUP} + K_0 M_0 R_0^{\alpha-1} + \theta\Big[\sum_{i=1}^{I_{MAX}+1} K_0 M_0 C_i R_i^{\alpha-1}\Big(\prod_{l=0}^{i-1} R_l^{-1}\Big)\Big] \tag{5.30}$$

in

$$\mathcal{E}_{TOT} \equiv \mathcal{E}_{SETUP} + K_0 M_0 R_0^{\alpha-1} + \theta\Big\{K_0 M_0 \left(\overline{w}\right)^{I_{MAX}+1} \left(R_{I_{MAX}+1}\right)^{\alpha-1} \Big[\prod_{k=0}^{I_{MAX}} \left(R_k\right)^{-1}\Big] + \\ + \left(1-\delta(I_{MAX})\right)\Big\{\sum_{l=1}^{I_{MAX}} K_0 M_0 \left(\overline{w}\right)^{l} \left(R_l\right)^{\alpha-1}\Big[\prod_{m=0}^{l-1} \left(R_m\right)^{-1}\Big]\Big\}\Big\}, \tag{5.31}$$

where the term $(1-\delta(I_{MAX}))$ accounts for the fact that the corresponding term $\Big\{\sum_{l=1}^{I_{MAX}} K_0 M_0 \left(\overline{w}\right)^{l} \left(R_l\right)^{\alpha-1}\Big[\prod_{m=0}^{l-1} \left(R_m\right)^{-1}\Big]\Big\}$ does not vanish only for $I_{MAX} \geq 1$.

The final expression for the energy wasted by the TCBM it is shown below

$$\mathcal{E}_{TOT} = \mathcal{E}_{SETUP} + K_0 M_0 (R_0)^{\alpha-1} + \theta K_0 M_0 (R_0)^{-1}\Big\{\left(\overline{w}\right)^{1+I_{MAX}} \left(R_{I_{MAX}+1}\right)^{\alpha-1} * \\ * \Big[\prod_{k=0}^{Q-1} \left(R_{kS+1}\right)^{-S}\Big] + (1-\delta(I_{MAX}))\Big\{\sum_{m=0}^{Q-1}\Big\{\sum_{l=mS+1}^{(m+1)S} \left(\overline{w}\right)^{l}\Big\{\delta(m)\left(R_1\right)^{\alpha-l} + \\ + \left(1-\delta(m)\right)\Big[\prod_{p=0}^{m-1} \left(R_{pS+1}\right)^{-S}\Big]\left(R_{mS+1}\right)^{\alpha+mS-l}\Big\}\Big\}\Big\}\Big\}, \tag{5.32}$$

while in Appendix A.4 we show all the steps to get the above equation.

5.4.4 Expression of the constraints on the slowdown and maximum rate for the TCBM

Then I define here the expression of the constraints on the slowdown and maximum rate for the manager TCBM. The $(Q+1)$ constraints on the slowdown read as in

$$\begin{aligned} &\beta\,\overline{w}(R_i)^{-1} - 1 \leq 0, \\ &for\ i = 0\ and\ i = jS+1\ ,\ j = 0, 1, \ldots, (Q-1); \end{aligned} \tag{5.33}$$

while the $(Q+2)$ constraints on the allowed maximum rate are

$$\begin{aligned} &R_i \leq \widehat{R}\ ,\ i=0\,;\ i=jS+1, \\ &for\ j=0,1,\ldots,(Q-1);\ \ i=(I_{MAX}+1), \end{aligned} \tag{5.34}$$

where $\widehat{R}$ is still given by

$$\widehat{R} \triangleq min\{R_{MAX}\ ;\ \rho_{MAX}\ R_{TOT}\}. \tag{5.35}$$

In particular, the constraint (5.33) there is only in the case of PeCM and it makes sure that the pre-copy phase is not carried out to infinity. On the contrary, the second constraint (5.34) is on the maximum bandwidth allocated to the migration process at each round.

5.5 Formulation of TCBM non-convex optimization problem

The TCBM, according to the Eqs. (5.23), (5.27), (5.29), (5.32), (5.33), (5.34) and (5.35), is the solution of the following non-convex optimization problem to be solved by our manager (the TCBM). It could be solved as an instance of geometric problem (solution formulation as an instance of Geometric program is provided in the following Section 5.5.1):

$$\min_{\{R_0, R_{jS+1},\, j=0,1,\ldots,(Q-1);\, R_{I_{MAX}+1}\}} \mathcal{E}_{TOT} \tag{5.36}$$

s.t.

$$\Psi_1 \triangleq \theta\left\{\left(\frac{1}{\Delta}_{TM} T_{TM}\right)-1\right\}\leq 0; \tag{5.37}$$

$$\Psi_2 \triangleq \left(\frac{1}{\Delta}_{DT} T_{DT}\right)-1 \leq 0; \tag{5.38}$$

$$\Psi_3 \triangleq \theta\left\{\beta\, \overline{w}\, R_i^{-1}-1\right\}\leq 0, \tag{5.39}$$

$$for\ i=0;\ i=jS+1;\ j=0,\ldots,(Q-1);$$

$$R_i \leq \widehat{R}, \tag{5.40}$$

$$for\ i=0;\ i=jS+1;\ j=0,\ \ldots,(Q-1);\ i=I_{MAX}+1;$$

where $\mathcal{E}_{TOT}$, T_{TM} and T_{DT} are given by Eqs. (5.32), (5.29) and (5.27), respectively.

Four constraints are considered in the formulation of the TCBM, which capture,

in turn, the metrics currently adopted for measuring the performance of live migration techniques [16, 18]. The first two constraints (Eq. (5.37) and Eq. (5.38)) upper limit the tolerated total migration time and downtime. Constrain (5.39) account the ratio of the volumes of data migrated over two consecutive rounds falls below a predefined speed-factor $\beta > 1$. It arises from the consideration that, without any stop condition, the iterative pre-copy stage of the PeCM technique may run indefinitely (see Fig. 3.7 in Section 3.2). Although the stop conditions depend highly on the design of the considered VMM, in [50] it is pointed out that they should account for the following two thresholds:

(i) the number of the performed rounds exceeds a pre-defined threshold I_{MAX}; and,

(ii) the ratio $\left(\frac{V_i}{V_{i+1}}\right)$ of the volumes of data migrated over two consecutive rounds falls below a predefined speed factor $\beta > 1$.

Hence, since the constraints in (5.37) and (5.38) and the energy function in Eq. (5.32) already account for I_{MAX}, an exploitation of Eq. (3.4) allows us to formulate the β-related constraint as in (5.39). By definition, this third constraint is present only when the PeCM technique is considered and this motivates the presence of the θ parameter (see Eq. (3.10)).

Constrain (5.40) upper limit the maximum available rate. Furthermore, the θ parameter in (5.37) accounts for the fact that, by definition, the total migration and stop-and-copy times coincide under the SaCM and PoCM techniques. The fourth constraint accounts for the maximum bandwidth assigned at each round for the migration process and it is fixed in order to avoid migration-induced traffic congestion phenomena [16].

The primal variables of the above non-convex problem are the $(Q + 2)$ cluster headers in Eq. (5.23). Furthermore, the problem presents $(2Q + 5)$ constraints, but the last $(Q + 2)$ constraints in Eq. (5.40) are of box-type and then they may be managed as implicit constraints. The utilization of the $\delta(\circ)$ function in Eqs. (5.32), (5.29) and (5.27) allows the problem formulation in (5.36)-(5.40) to cover also the 2 limit case of $I_{MAX} = -1$ and $I_{MAX} = 0$.

Specifically, the stop-and-copy (SoC) case is still obtained by posing in (5.36)-(5.40):

$$\theta = 0,\ I_{MAX} = -1,\ \rho_{MAX}\, R_{TOT} = +\infty,\ Q = 1\ and\ S = 0. \tag{5.41}$$

Furthermore, at $I_{MAX} = 0$, we must also pose:

$$\theta = 1,\ I_{MAX} = 0,\ Q = 1\ and\ S = 0. \tag{5.42}$$

5.5.1 Feasibility conditions for TCBM

The following *Proposition 4* formalizes the necessary and sufficient conditions for the feasibility of the TCBM in (5.36)-(5.40) (see Appendix A.4 for the proof).

Proposition 4. *TCBM feasibility conditions*
The TCBM is feasible if and only if the following three conditions are simultaneously met:

$$\theta\Bigg\{ \frac{M_0}{\Delta_{TM}} \left[\frac{1}{\widehat{R}} \left(I_{MAX}+2\right) \delta\left(\frac{\overline{w}}{\widehat{R}} - 1\right) + \left(\frac{1-(\overline{w}/\widehat{R})^{I_{MAX}+2}}{\widehat{R}-\overline{w}} \right) \left(1 - \delta\left(\frac{\overline{w}}{\widehat{R}} - 1\right)\right) \right] \Bigg\} \leq 1; \tag{5.43}$$

$$\frac{M_0}{\Delta_{DT}}\ \frac{1}{\widehat{R}}\ \left(\frac{\overline{w}}{\widehat{R}}\right)^{I_{MAX}+1} \leq 1; \tag{5.44}$$

$$\theta\beta \left(\frac{\overline{w}}{\widehat{R}}\right) \leq 1,\ \beta \geq 1. \tag{5.45}$$

Regarding the practical utility of the conditions (5.43)–(5.45), we point out that, when (and only when) these conditions are met, we are guaranteed that there exists at least one vector of values of R that satisfies all the constraints in (5.36)–(5.40). Interestingly, the effects of $\widehat{R}$ and I_{MAX} on the reported feasibility conditions are formally stressed by the following *Proposition 5.* □

Proposition 5. *I_{MAX} and $\widehat{R}$ effects on the feasibility conditions*
The effects of $\widehat{R}$ and I_{MAX} on the reported feasibility conditions are formally stressed by the following:

(i) At fixed I_{MAX}, all the functions at the left-hand-side (l.h.s) of Eqs. (5.43)-(5.45) strictly decrease (resp. strictly increase) for increasing values of $\widehat{R}$ (resp. $\overline{w}$).

(ii) At fixed $\widehat{R}$,we have that:

  (a) the feasibility condition in (5.45) may be met only if $(\overline{w}/\widehat{R}) \leq 1$ *at* $\theta = 1$;

  (b) for increasing values of I_{MAX}, the function at the l.h.s. of Eq. (5.44): (i) does not vary; (ii) is strictly decreasing; and, (iii) is strictly increasing, at $(\overline{w}/\widehat{R}) = 1$, $(\overline{w}/\widehat{R}) < 1$ and $(\overline{w}/\widehat{R}) > 1$, respectively;

  (c) for increasing values of I_{MAX}, the function at the l.h.s. of Eq. (5.43) strictly increases, regardless of the values assumed by the ratio $(\overline{w}/\widehat{R})$. □

Proof. A direct inspection of Eqs. (5.43)–(5.45) leads to the stated conclusions. □

In summary, while increasing values of $\widehat{R}$ always lead to reduced T_{TM} and T_{DT}, the effect of I_{MAX} is more questionable. Specifically, T_{TM} always increases for increasing I_{MAX}, while it is guaranteed that larger values of I_{MAX} lead to lower downtimes at $(\overline{w}/\widehat{R}) < 1$. This conclusion leads to three main consequences.

First, in the carried out numerical tests, we limit to the case: $(\overline{w}/\widehat{R}) \leq 1$.

Second, at $(\overline{w}/\widehat{R}) > 1$, it is guaranteed that the downtime increases for increasing I_{MAX}. Hence, in this case, the stop-and-copy solution (if feasible) is the best one.

Third, at $(\overline{w}/\widehat{R}) < 1$ we claim that an optimized setting: $\widetilde{I}_{MAX}$ of I_{MAX} is obtained by computing the value of I_{MAX} that meets the constraint in (5.44) with the equality, that is

$$\widetilde{I}_{MAX} \equiv \left\lceil \frac{\log\left(\dfrac{M_0}{\Delta_{DT}\,\widehat{R}}\right)}{\log\left(\dfrac{\widehat{R}}{\overline{w}}\right)} - 1 \right\rceil, \quad for \quad (\overline{w}/\widehat{R}) < 1, \tag{5.46}$$

where $\lceil \circ \rceil$ is the ceiling function and log is the natural logarithmic. Afterwards, if $\tilde{I}_{MAX} \geq 1$, I test the feasibility of the constraint (5.43) at $I_{MAX} = \tilde{I}_{MAX}$. If the constraint (5.43) is met, $\tilde{I}_{MAX}$ is a feasible setting. Otherwise, I select a larger value of Δ_{TM} and, then, we repeat the calculation of (5.46). Algorithm 1 reports the flow chart of the described procedure for the optimized setting of I_{MAX}.

Algorithm 1: Pseudo code for the optimized setting of I_{MAX}.

Data: $\widehat{R}, \overline{w}$
Result: $\widetilde{I}_{MAX}$
initialization of $\overline{w}/\hat{R} < 1$;
repeat
  compute $\widetilde{I}_{MAX}$ as in (5.46);
  if $I_{MAX} \equiv \widetilde{I}_{MAX}$ *meet the feasibility conditions in* (5.43) **then**
    return $\widetilde{I}_{MAX}$;
  else
    select a lower value for $\overline{w}/\hat{R} < 1$;
  end
until $I_{MAX} \equiv \widetilde{I}_{MAX}$ *not meet the feasibility conditions in* (5.43);
return $\widetilde{I}_{MAX}$;

5.5.2 A convex form as an instance of Geometric program for TCBM optimization problem

Since all the $(Q+2)$ rate variables in (5.23) are non-negative, we introduce the dummy positions:

$$\tilde{R}_i \triangleq \log(R_i), \quad i=0;\ i=jS+1;\ j=0,\ldots,(Q-1);\ i=(I_{MAX}+1); \tag{5.47}$$

so that

$$R_i = e^{\tilde{R}_i}, \quad i=0;\ i=jS+1;\ j=0,\ldots,(Q-1);\ i=(I_{MAX}+1). \tag{5.48}$$

Furthermore, we collect the $(Q+2)$ log-rates in (5.47) in the following $(Q+2)$-dimensional (column) vector:

$$\overrightarrow{\tilde{R}} \triangleq [\tilde{R}_0,\ \tilde{R}_1,\ \tilde{R}_{S+1},\ldots,\ \tilde{R}_{(Q-1)S+1},\ \tilde{R}_{I_{MAX}+1}] \ \in (IR)^{Q+2} \tag{5.49}$$

and, then, we indicate as:

$$\begin{aligned}\mathcal{E}_{TOT}(\overrightarrow{\tilde{R}}) = \mathcal{E}_{SETUP} + K_0 M_0\, e^{[(\alpha-1)\tilde{R}_0]} + \theta K_0 M_0\, e^{[-\tilde{R}_0]} \Bigg\{ (\overline{w})^{I_{MAX}+1} \ * \\ * \ e^{\left[+(\alpha-1)\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]} + \\ + (1-\delta(I_{MAX}))\Bigg\{\sum_{m=0}^{Q-1}\Bigg\{\sum_{l=mS+1}^{(m+1)S}(\overline{w})^l\Bigg\{\delta(m)e^{[(\alpha-l)\tilde{R}_1]} + \\ + \ (1-\delta(m))\ e^{\left[(\alpha+mS-l)\tilde{R}_{mS+1} - S\tilde{R}_1 - (1-\delta(m-1))\left(\sum_{p=1}^{m-1}\tilde{R}_{pS+1}\right)\right]}\Bigg\}\Bigg\}\Bigg\}\Bigg\},\end{aligned} \tag{5.50}$$

$$\begin{aligned}T_{DT}(\overrightarrow{\tilde{R}}) = M_0\, e^{[-\tilde{R}_0]}\Bigg\{\delta(1+I_{MAX}) + (1-\delta(1+I_{MAX}))(\overline{w})^{1+I_{MAX}} * \\ * \ e^{\left[-\tilde{R}_{I_{MAX}+1} - SR_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]}\Bigg\},\end{aligned} \tag{5.51}$$

$$T_{TM}(\overrightarrow{\tilde{R}}) = M_0\, e^{[-\tilde{R}_0]} \Bigg\{ 1 + (\overline{w})^{1+I_{MAX}}\, e^{\Bigg[-\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S* \\ * e^{\left(\sum_{k=1}^{Q-1} \tilde{R}_{kS+1}\right)}\Bigg]} + (1-\delta(I_{MAX}))\Bigg\{ \sum_{k=0}^{Q-1} \Bigg\{ \sum_{l=kS+1}^{(k+1)S} (\overline{w})^l \Bigg\{ \delta(k)\, e^{[-l\tilde{R}_1]} + \\ + (1-\delta(k)) * e^{\Bigg[(kS-l)\tilde{R}_{kS+1} - S\tilde{R}_1 - (1-\delta(k-1))S\left(\sum_{p=1}^{k-1} \tilde{R}_{pS+1}\right)\Bigg]} \Bigg\}\Bigg\}\Bigg\}\Bigg\}, \tag{5.52}$$

the resulting expressions of the total energy, downtime and total migration time which are obtained by introducing the positions in (5.48) into the formulas of Eqs. (5.32), (5.27) and (5.29), respectively.

Hence, the TC problem in (5.36)-(5.40) may be recast in the following equivalent form:

$$\min_{\overrightarrow{\tilde{R}}} \ \mathcal{E}_{TOT}(\overrightarrow{\tilde{R}}) \tag{5.53}$$

$$s.t.$$

$$\Psi_1(\overrightarrow{\tilde{R}}) \triangleq \theta\Bigg\{ \left[\frac{1}{\Delta_{TM}} T_{TM}(\overrightarrow{\tilde{R}}) \right] - 1 \Bigg\} \leq 0; \tag{5.54}$$

$$\Psi_2(\overrightarrow{\tilde{R}}) \triangleq \left[\frac{1}{\Delta_{DT}} T_{DT}(\overrightarrow{\tilde{R}}) \right] - 1 \leq 0; \tag{5.55}$$

$$\Psi_3(\overrightarrow{\tilde{R}}) \triangleq \theta \left[\beta\, \overline{w}\, e^{-\tilde{R}_i} - 1 \right] \leq 0, \tag{5.56}$$

$$for\ i = 0;\ i = jS+1;\ j = 0, \ldots, (Q-1);$$

$$\tilde{R}_i \leq \log(\widehat{R}), \tag{5.57}$$

$$for\ i = 0;\ i = jS+1;\ j = 0,\ \ldots, (Q-1);\ i = I_{MAX}+1.$$

The above problem is convex (but, generally, not strictly convex) in the $(Q+2)$ log-rate variables $\overrightarrow{\tilde{R}}$ in (5.49), so that the objective function in (5.53) admits an unique global minimum value.

Let

$$\overrightarrow{\tilde{R}} \in (IR)^{Q+2}, \tag{5.58}$$

the $(Q+2)$-dimensional (possibly, not unique) vector solution of the problem (5.53)-(5.57). Since, the following *Proposition 6* hold:

Proposition 6. Let the feasibility conditions in Eqs. (5.43)-(5.45) be met with the strict inequality. Hence, the (convex Geometric programming) problem of Eqs. (5.53)-(5.57) meets the Slater's conditions.

Due to its convex form, the box constraints in (5.57) of the TC problem may be managed as implicit ones through orthogonal projection. Hence, let:

$$\overrightarrow{\lambda} \triangleq [\, \lambda_1 \, \lambda_2 \, \lambda_{3,0} \, \lambda_{3,1} \, \lambda_{3,(S+1)} \, \lambda_{3,(2S+1)} \, \cdots \, \lambda_{3,((Q-1)S+1)}] \; \in (IR)^{Q+3} \tag{5.59}$$

be the nonnegative $(Q+3)$-dimensional vector of the Lagrange multipliers associated to the $(Q+3)$ inequality constraints in (5.54)-(5.56). Hence, the Lagrangian function $L(\overrightarrow{\tilde{R}}, \overrightarrow{\lambda})$ of the convex problem in (5.53)-(5.57) reads as in:

$$\begin{aligned} L(\overrightarrow{\tilde{R}}, \overrightarrow{\lambda}) \triangleq \; & \mathcal{E}_{TOT}(\overrightarrow{\tilde{R}}) \; + \; \lambda_1 \theta \Bigg\{ \Bigg[\frac{1}{\Delta_{TM}} \, T_{TM}(\overrightarrow{\tilde{R}}) \Bigg] - 1 \Bigg\} + \\ & + \; \lambda_2 \Bigg\{ \Bigg[\frac{1}{\Delta_{DT}} \, T_{DT}(\overrightarrow{\tilde{R}}) \Bigg] - 1 \Bigg\} + \theta \Bigg\{ \; \lambda_{3,0} \left[\beta \, \overline{w} \, e^{-\tilde{R}_0} \; - \; 1 \right] + \\ & + \; \Bigg\{ \sum_{m=0}^{Q-1} \lambda_{3,(mS+1)} \Big[\beta \, \overline{w} \, e^{-\tilde{R}_{Sm+1}} \; - \; 1 \Big] \Bigg\} \Bigg\}. \end{aligned} \tag{5.60}$$

Therefore, the minimization to be carried out is the following one:

$$\max_{\overrightarrow{\lambda} \geq \overrightarrow{0}} \; \Bigg\{ \min_{\{ \overrightarrow{\tilde{R}}_i \leq \log \widehat{R}, \; i = 0, \, (S+1), \, ..., \, (Q-1)S+1, \, (I_{MAX}+1) \}} \; \Big\{ L(\overrightarrow{\tilde{R}}, \overrightarrow{\lambda}) \Big\} \Bigg\}. \tag{5.61}$$

The latter may be, in turn, computed by performing the orthogonal projection on the box sets in (5.61) of the vector which solves the $(2Q+5)$-dimensional vector gradient of the Lagrangian function in (5.61) performed with respect to both the primal $\overrightarrow{\tilde{R}}$ and dual $\overrightarrow{\lambda}$ variables in (5.49) and (5.59) respectively, that is:

$$\overrightarrow{\nabla} L(\overrightarrow{\tilde{R}}, \overrightarrow{\lambda}) = \overrightarrow{0}_{(2Q+5)x1}. \tag{5.62}$$

In this sequel, I indicate as

$$\Big\{ \overrightarrow{\tilde{R}}^* \, , \; \overrightarrow{\lambda}^* \Big\} \tag{5.63}$$

the $(2Q+5)$-dimensional (possible, not unique) projection solution of the algebric equations system in (5.62). This solution in (5.63) may be, in turn, iteratively computed through a suitable set of projected gradient-based primal-dual scalar iterates. At this regard, we note that, as pointed out in [57, 58], the primal-dual algorithm is an iterative procedure for solving convex optimization problems, which applies quasi-Newton methods for updating the primal-dual variables simultaneously and moving toward the saddle-point of the underlying Lagrangian function at each iteration. In our framework, the

$$2Q \; + \; 5 \, (\textit{Computation complexity of TCBM}) \tag{5.64}$$

scalar updating to be carried out at the n-th iterate reads in:

$$\overrightarrow{\tilde{R}}_i^{(n+1)} = \left[\overrightarrow{\tilde{R}}_i^{(n)} \; - \; \omega_i^{(n)} \, \nabla_{\tilde{R}_i} \, L(\overrightarrow{\tilde{R}}^{(n)} \, , \; \overrightarrow{\lambda}^{(n)})\right]^{\log(\tilde{R})}, \quad n \geq 0; \;\; \tilde{R}_i^{(0)} = \log(\overline{w}); \\ for \;\; i = 0; \;\; i = jS + 1; \;\; j = 0, \ldots, (Q-1); \;\; i = (I_{MAX} + 1); \tag{5.65}$$

$$\lambda_i^{(n+1)} = \left[\lambda_i^{(n)} \; + \; \tau_i^{(n)} \, \nabla_{\lambda_i} \, L(\overrightarrow{\tilde{R}}^{(n)} \, , \; \overrightarrow{\lambda}^{(n)})\right]_+, \quad n \geq 0, \;\; \lambda_i^{(0)} = 0; \quad i = 1, 2; \tag{5.66}$$

$$\lambda_{3,i}^{(n+1)} = \left[\lambda_{3,i}^{(n)} \; + \; \Psi_{3i}^{(n)} \, \nabla_{\lambda_{3,i}} \, L(\overrightarrow{\tilde{R}}^{(n)} \, , \; \overrightarrow{\lambda}^{(n)})\right]_+, \quad n \geq 0, \;\; \lambda_{3,i}^{(0)} = 0; \\ for \;\; i = 0; \;\; i = jS + 1; \;\; j = 0, 1, \ldots, (Q-1); \tag{5.67}$$

where $\{\omega_i^{(n)}, \;\; n \geq 0\}$, $\{\tau_i^n, \;\; m \geq 0\}$ and $\{\Psi_{3i}^{(n)}, \;\; n \geq 0\}$ are $(2Q + 5)$ suitable nonnegative sequences of n-varying step-sizes. The projections in (5.65), (5.66) and (5.67) account for the box-type constraints in (5.61). The Appendix A details the analytical expressions of the partial derivatives $\nabla_{\tilde{R}} L(.)$ and $\nabla_{\lambda_i} L(.)$.

Regarding the convergence to the global minimum of the primal-dual iterations of Eqs. (5.65), (5.66) and (5.67), three main remarks are in order.

First, under the feasibility conditions of *Proposition 4*, the global minimum of Eqs. (5.53)-(5.57) exist.

Second, due to the strict convexity of the problem in (5.53)-(5.57), the global minimum is unique (see Theorem 3.4.2 of [49]). Furthermore, the convergence of the primal-dual iterations of Eqs. (5.65)-(5.67) to the global minimum is guaranteed, regardless of the adopted starting point and the size of the considered instance of the optimization problem. Formal proofs of this property of global convergence may be found, for example, in [57, 49, 59].

Third, in practical application scenarios, the average memory dirty rate $\overline{w}$ and/or the round-trip-time dependent K_0 parameter in (5.10) may exhibit unpredictable (possibly, abrupt) time-variations over a same migration session and/or consecutive migration sessions. As detailed in Section 5.7, $\overline{w}$ may vary due to workload fluctuations experienced by the migrating VM [50, 49], while congestion-induced jitters of the round-trip-time RTT of the utilized TCP connection may give arise to (unpredictable) changes of K_0 in (3.9). □

5.5.3 Expressions for the gain sequences of the TCBM

Here I present a new approach for calculating the gain sequences $\{\omega_i^{(n)}\}$, $\{\tau_i^{(n)}\}$ and $\{\Psi_i^{(n)}\}$ present in Eqs. (5.65), (5.66) and (5.67). These expressions follow the approach very recently presented in [75] for the computation of the gain sequences of

the Congestion Control algorithm implemented by the new paradigm of Multi-path TCP. Shortly, in [75] the Optimal Congestion control problem is formulated as a primal-dual optimization problem and then solved by a implementing a suitable set of projected gradient-based iterations (see *Eqs. (3),(4) and (6), (7)* of [75]). Hence, the formal approach followed in [75] in the same adopted in my Eqs. (5.61), (5.62), (5.65)-(5.67). This is the reason why the formal results about the convergence property and the optimized setting of the gain sequences reported in [75] apply verbatim to our TCBM optimization problem. Specifically, two main formal results are relevant in our context.

First, the *Theorem 3.3* of [75] proves that is sufficient that each gain sequence $k_y(n)$ for updating the generic variable $y(n)$ is positive and depend only on $y(n)$ (e.g. it does not depend on the remaining variables to be optimized) in order to guarantee that the optimal solution of the constrained optimization problem is globally asymptotically stable, that is, it is reached from any starting point. Proof of Th. 3.3 is reported in the *Appendix C* of [75] and relies on the Lypunov function of Eq. (21) of [75].

Second, a good choice for updating $k_y(n)$ is to set it proportional to $(y(n))^2$, that is (see section II.B and Eq. (12) of [75]):

$$k_y(n) \propto \frac{1}{2}\Big(y(n)\Big)^2, \;\; n \geq 1. \tag{5.68}$$

Hence, according to these results, I must investigate and implement the following $(2Q+5)$ formulas for the updating of the gain sequences like:

$$\begin{aligned} \omega_i^{(n)} = max\Big\{\frac{a_{MAX}}{10};\, min\Big\{a_{MAX};\;\frac{1}{2}\Big(\tilde{R}_i^{(n)}\Big)^2\Big\}\Big\}, \quad n \geq 1; \;\; \omega_i^{(0)} = a_{MAX}, \\ for \;\; i = 0;\;\; i = jS+1;\;\; j = 0,\ldots,(Q-1); \quad i = (I_{MAX}+1); \end{aligned} \tag{5.69}$$

$$\begin{aligned} \tau_i^{(n)} = max\Big\{\frac{a_{MAX}}{10};\, min\Big\{a_{MAX};\;\frac{1}{2}\Big(\lambda_i^{(n)}\Big)^2\Big\}\Big\}, \quad n \geq 1; \;\; \tau_i^{(0)} = a_{MAX}, \\ for \;\; i = 1,2; \end{aligned} \tag{5.70}$$

$$\begin{aligned} \Psi_{3,i}^{(n)} = max\Big\{\frac{a_{MAX}}{10};\, min\Big\{a_{MAX};\;\frac{1}{2}\Big(\lambda_{3,i}^{(n)}\Big)^2\Big\}\Big\}, \quad n \geq 1; \;\; \Psi_{3,i}^{(0)} = a_{MAX}; \\ for \;\; i = 0;\;\; i = jS+1;\;\; j = 0,\ldots,(Q-1). \end{aligned} \tag{5.71}$$

I expect that the optimized value to be used in (5.69)-(5.71) for a_{MAX} is very larger. However, the tuning of a_{MAX} in (5.69))-(5.71) must be carried out through

numerical trials.

5.6 Profiling network connection and migrating application

In case of implementation my proposed bandwidth manager requires a priori information about the power-vs-rate relationship in (3.8) and the memory size and dirty rate in (3.5). As detailed in the sequel, this information may be acquired on-line during the first part of the Pre-migration stage by exploiting some commands and profiling tools already done available by current VMMs [51, 19].

The average round-trip-time (RTT) in (3.9) and the maximum throughput $\widehat{R}$ in (5.35) of the available TCP connection may be directly measured at the Transport layer by using, for example, the (standard) Linux *iperf* command [51]. In order to profile at runtime the parameters $\mathcal{E}_{SETUP}$, K_0 and α in (5.32), we may use the Xen *ifconfig* command [51]. It reports the power state of the physical *NIC* which is used by the migrating VM. Hence, by issuing the *ifconfig* command at $R = 0$ and $R = 1$ (Mb/s), may be directly measure $\mathcal{E}_{SETUP}$ and K_0 respectively. Afterward, by issuing the *ifconfig* command at $R = \widehat{R}$, it is possible to measure the total (e.g., static-plus-dynamic) power: $P_{TOT}(\widehat{R})$ consumed by the TCP connection at $R = \widehat{R}$. Hence, since, by definition, we have that:

$$P_{DYN}(\widehat{R}) \triangleq \widehat{R}\mathcal{E}_{DYN}(\widehat{R}) \equiv P_{TOT}(\widehat{R}) - \widehat{R}\mathcal{E}_{SETUP}, \tag{5.72}$$

directly from the relationship in (3.8), I obtain the following closed-form expression for the α exponent:

$$\alpha = \frac{\log\left(\dfrac{P_{DYN}(\widehat{R})}{K_0}\right)}{\log\ \widehat{R}} \equiv \frac{\log\left(\dfrac{P_{TOT}(\widehat{R}) - \widehat{R}\mathcal{E}_{SETUP}}{K_0}\right)}{\log\ \widehat{R}}. \tag{5.73}$$

Regarding the profiling of the migrating application, I note that the memory size M_0 of the migrating VM may be measured during the pre-migration stage through the xen-store command [51]. Afterward, in order to profile at runtime the corresponding average dirty memory rate $\overline{w}$, let $\widetilde{M}_0$ be the (integer-valued) number of memory pages of the migrating VM and let $m = 0, 1, \ldots, (\widetilde{M}_0 - 1)$, be the (possibly, relative) corresponding memory address index. Furthermore, let $\chi(m, i) \in \{0, 1\}$, $m = 0, 1, \ldots, (\widetilde{M}_0 - 1)$, $i = 0, \ldots, I_{MAX}$, be the binary function which marks the dirtied/not dirtied state of the mth memory page at the end of the ith round. Interestingly enough, the spectrum $\{\chi(m, i)\}$ of the dirtied memory pages may be directly acquired at run-time from the dirty bitmap which the Xen hypervisor makes

periodically available [51, 19]. Therefore, the resulting dirty memory rate $\overline{w}$ averaged over the duration T_{IP} of the iterative pre-copy stage may be directly profiled on-line through the following relationship:

$$\overline{w} = \left(\frac{M_0}{T_{IP}\ \widetilde{M}_0} \right) \left(\sum_{m=0}^{\widetilde{M}_0 - 1} \sum_{i=0}^{I_{MAX}} \chi(m\ ,\ i) \right)\ (Mb/s). \tag{5.74}$$

5.7 Simulation results on TCBM

In this section I provide the results of comparative tests and simulations of the proposed bandwidth manager TCBM. In the following I discuss some simulations' results that show the goodness of our TCBM, comparing with the results obtained from Xen solution and the BMOP (Bandwidth Management Optimization Problem) manager, see Part I of this thesis, in which, unlike in TCBM software, the initial rate is held for the entire duration of the VM migration.

Therefore, goal of this paragraphs is to provide range of values of the input data in order to perform evaluation and performance comparisons on wireless TCPNewReno-over-IP based application scenario.

Specifically, the reported data refer to the average parameters of typical wireless IEEE 802.11b, 3G-UTRAN and 4G-LTE connections. I anticipated that the reported data are in agreement with most relevant literature [74] for 3G-UTRAN and [76] for 4G-LTE.

After noting that $\tilde{I}_{MAX}$ refers to our optimized setting of the allowed pre-copy rounds, typically values for the tested VMs are:

$$1\ \leq\ \tilde{I}_{MAX}\ \leq\ 29, \tag{5.75}$$

where $\tilde{I}_{MAX} = 29$ is the Xen's default setting.

All simulations have been carried out in three different application scenarios, i.e., the scenario in which the mobile device migrates to the access point by 3G; the scenario in which the mobile device migrates with the use of the 4G; and finally the scenario where migration is performed by making use of WiFi. In the following I present the parameter's values used in the three application scenarios.

Maximum throughput at the Transport Layer rate R_{MAX} is:

$$R_{MAX} \in \begin{cases} 0.9\ \times\ 2\,(Mb/s) & for\ \ 3G\,cellular, \\ 0.9\ \times\ 11\,(Mb/s) & for\ \ IEEE\,802.11.b, \\ 0.9\ \times\ 50\,(Mb/s) & for\ \ 4G-LTE, \end{cases} \tag{5.76}$$

where R_{MAX} (Mb/s) is the maximum throughput at the Transport Layer.

The static (i.e., rate independent) part of the overall energy consumption of the considered connection $\mathcal{E}_{SETUP}$ is:

$$\mathcal{E}_{SETUP} \in \begin{cases} 3.25\,(J) & for\ \ 3G\,cellular; \\ 5.9\,(J) & for\ \ IEEE\,802.11.b; \\ 5.1\,(J) & for\ \ 4G-LTE. \end{cases} \tag{5.77}$$

About the dynamic power-vs-rate relationship, I note that this relationship for TCP-based end-to-end connections working in the Congestion Avoidance state (e.g., in the steady-state) is well captured by the following α-power formula:

$$P_{DYN}(R) \;=\; K_0\,(R)^{\alpha},\ \ (Watt) \tag{5.78}$$

where R (in (Mb/s)) is the TCP throughput and

$$K_0 \;=\; \frac{1}{g}\Big(\frac{RTT}{1.22\,MSS}\Big)^{\alpha},\ \ \Big(Watt \times \Big(\frac{s}{Mb}\Big)^{\alpha}\Big). \tag{5.79}$$

The corresponding dynamic part $\mathcal{E}_{DYN}$ (J) of the energy consumed by the connection is given by $\mathcal{E}_{DYN} \triangleq (P_{DYN}(R)\,/\,R) \equiv K_0\,(R)^{\alpha-1}$. Hence, the case $\alpha = 2$ corresponds to the linear energy-vs-rate relationship.

In (5.79) RTT (s) is the average round-trip-time of the TCP connection, g $((Watt)^{-1})$ is the coding gain-to-receive noise power ratio, and MSS is the corresponding Maximum Segment Size.

In my simulations MSS equates the size of an overall missed memory page, that is

$$MSS = 1000(Byte) \times 8 = 0.008(Mb). \tag{5.80}$$

Furthermore, RTT in (5.79) depends on the spatial coverage and traffic congestion of the considered TCP connection. In our tests, I have:

$$RTT \in \begin{cases} 250 \times 10^{-3}\,(s) & for\ \ 3G\,cellular; \\ 25 \times 10^{-3}\,(s) & for\ \ IEEE\,802.11.b; \\ 35 \times 10^{-3}\,(s) & for\ \ 4G-LTE. \end{cases} \tag{5.81}$$

Finally, the actual value of K_0 in (5.78) depends on the considered transmission medium and technology adopted at the *Physical Layer* through the gain g in (5.79).

In my simulations I have experienced the following (average) values in upload:

$$K_0\left(Watt \times \left(\frac{s}{Mb}\right)^{\alpha}\right) \in \begin{cases} 1.8 \times 10^{-1}\,(s) & for\ 3G\,cellular; \\ 5 \times 10^{-2}\,(s) & for\ IEEE\,802.11.b; \\ 9 \times 10^{-2}\,(s) & for\ 4G-LTE. \end{cases} \tag{5.82}$$

The values in (5.82) are the ones utilized in the numerical tests. These values are in agreement with the following experiences, as you can see in [76] and [77]:

(i) IEEE 802.11b NICs typically consume 0.1 (μJ) per bit, i.e., 0.1 (J) per Mb of transferred bulk data;

(ii) 3G connections are 3.65 times more energy expensive than WiFi ones in uploading [77];

(iii) uploading of bulk data over 4G-LTE connections is about 1.8 times more energy expensive than the WiFi case [76].

5.7.1 Benchmark Xen bandwidth management

The currently implemented Xen hypervisor adopts a pre-copy heuristic bandwidth management policy, which operates on a best effort basis, while attempting to shorten the final stop-and-copy time [19, 30]. The rationale behind this Xen policy is that, in principle, the stop-and-copy time may be reduced by monotonically increasing the migration bandwidth over consecutive rounds [30]. For this purpose, the Xen hypervisor uses pre-assigned minimum: R_{MIN}^{XEN} (Mb/s), and maximum: R_{MAX}^{XEN} (Mb/s) bandwidth thresholds, in order to bound the migration bandwidth during the pre-copy stage (see Section 5.3 of [30]). Specifically, the Xen migration bandwidth R^{XEN} equates: R_{MIN}^{XEN} (Mb/s) at $round\#0$, and, then, it increases in each subsequent round by a constant term: ΔR^{XEN} (Mb/s), so to reach the maximum value: $R^{XEN} = R_{MAX}^{XEN}$ at the last round: $round\#(I_{MAX}+1)$ (see *Section 5.3* of [30]).

In the carried out tests, I have implemented this benchmark policy by setting:

$$\Delta R^{XEN} = \frac{(R_{MAX}^{XEN} - \overline{w})}{(I_{MAX}^{XEN} + 1)}, \tag{5.83}$$

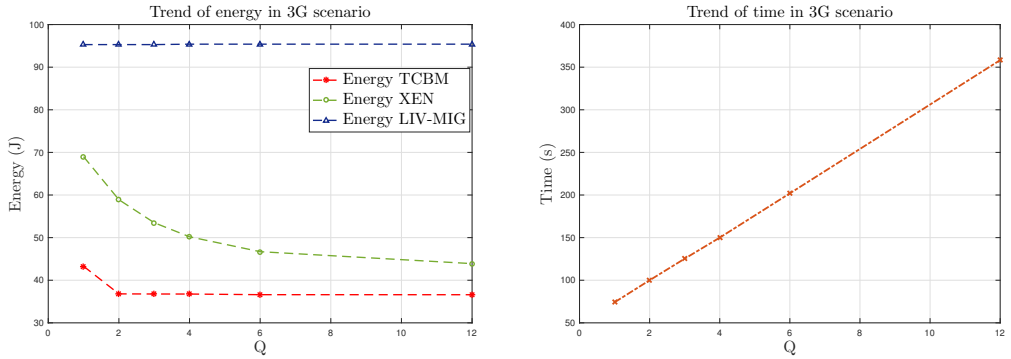
and

$$\begin{aligned} R_i^{XEN} &= \overline{w} + i\Delta R^{XEN}, \\ i &= 0, \ldots, (I_{MAX}^{XEN} + 1). \end{aligned} \tag{5.84}$$

Finally, I point out that, on the basis of the recent surveys in [16], *Chapter 3* of [19] and *Chapter 17* of [25], this is the only bandwidth management policy currently considered by both academy and industry for VM migration. And, on basis of my knowledge, this is also the bandwidth policy currently implemented by Xen, KVM and VMware commercial hypervisors [19].

5.7.2 How to choose the best value of Q parameter

In this section, I describe the results and considerations relating to the implementation of bandwidth manager for the QoS of the live migration of VMs. In particular, TCBM is a solution which represents a good compromise between low energy consumption and low implementation complexity. With LIV-MIG I refer to the bandwidth manager BMOP as in Chapter 4. It is recalled that, unlike the method implemented in I (where one optimized rate is calculated and in each round is used the same rate), the TCBM manager is characterized from the fact that on $I_{MAX}+2$ slots available, in addition to the rate R_0 and $R_{I_{MAX}+1}$, are updated only Q other rate. I have to take into account the fact that Q must be a divisor of I_{MAX}. For this reason is of crucial importance, for my purposes, to choose accurately the integer value to give to the variable Q.

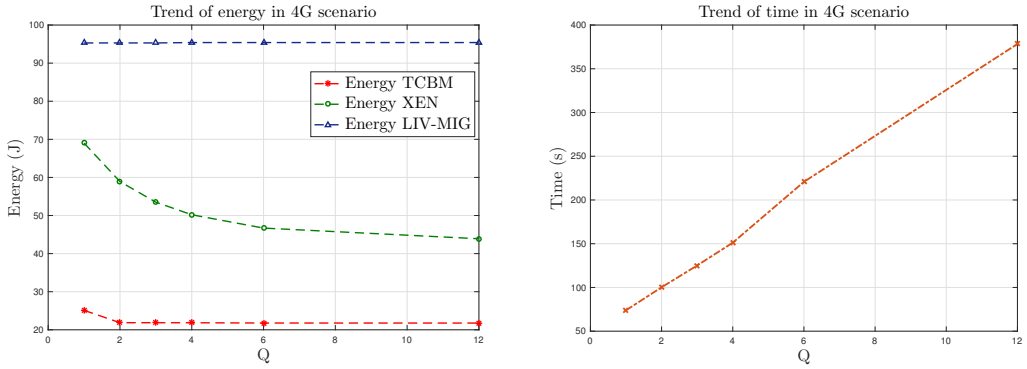


(a) Trend of enegy in 3G scenario (b) Trend of time in 3G scenario

Figure 5.12. (a) Trend of energy in 3G scenario for Xen manager, LIV-MIG manager and our bandwidth manager TCBM, with $I_{MAX} = 12$ and $Q = 1, 2, 3, 4, 6, 12$. The parameters used in this case are: $K_0^{3G} = 0.18((W) \times (s/Mb)^{\alpha})$; $\widehat{R} \equiv R_{MAX}^{3G} = 1.8(Mb/s)$; $\alpha = 2$; $\mathcal{E}_{SETUP}^{3G} = 3.25(J)$; $a_{MAX} = 0.0001$; $M_0 = 256(Mb)$; $\overline{w} = 0.18(Mb/s)$; $\Delta_{TM} = 2770(s)$; $\Delta_{DT} = 7.53 \times 10^{-7}(s)$. (b) Trend of the time due to the variation of Q for Tunable complexity bandwidth manager.

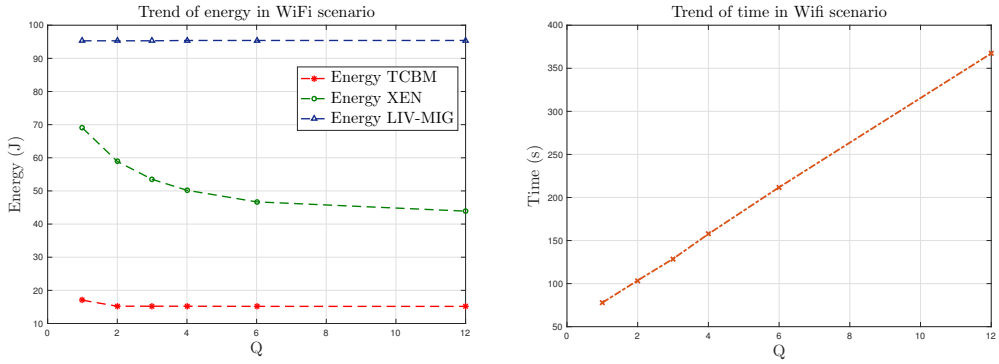
An examination of the Figs. 5.12, 5.13 and 5.14 leads to four main conclusion.

First, stands out immediately evident that in both three scenarios, migrate through the TCBM makes sure that the energy curve turns out to be always below the curves obtained through Xen and LIV-MIG (see Chapter 4) manager. In



(a) Trend of energy in 4G scenario (b) Trend of time in 4G scenario

Figure 5.13. (a) Trend of energy in 4G scenario for Xen manager, LIV-MIG manager and our bandwidth manager TCBM, with $I_{MAX} = 12$ and $Q = 1, 2, 3, 4, 6, 12$. The parameters used in this case are: $K_0^{4G} = 0.09((W) \times (s/Mb)^{\alpha})$; $\widehat{R} \equiv R_{MAX}^{3G} = 1.8(Mb/s)$; $\alpha = 2$; $\mathcal{E}_{SETUP}^{4G} = 5.1(J)$; $a_{MAX} = 0.0001$; $M_0 = 256(Mb)$; $\overline{w} = 0.18(Mb/s)$; $\Delta_{TM} = 2770(s)$; $\Delta_{DT} = 7.53 \times 10^{-7}(s)$. (b) Trend of the time due to the variation of Q for our Tunable complexity bandwidth manager.



(a) Trend of energy in WiFi scenario (b) Trend of time in WiFI scenario

Figure 5.14. (a) Trend of energy in WiFi scenario for Xen manager, LIV-MIG manager and our bandwidth manager TCBM, with $I_{MAX} = 12$ and $Q = 1, 2, 3, 4, 6, 12$. The parameters used in this case are: $K_0^{WiFi} = 0.05((W) \times (s/Mb)^{\alpha})$; $\widehat{R} \equiv R_{MAX}^{3G} = 1.8(Mb/s)$; $\alpha = 2$; $\mathcal{E}_{SETUP}^{WiFi} = 5.9(J)$; $a_{MAX} = 0.0001$; $M_0 = 256(Mb)$; $\overline{w} = 0.18(Mb/s)$; $\Delta_{TM} = 2770(s)$; $\Delta_{DT} = 7.53 \times 10^{-7}(s)$. (b) Trend of the time due to the variation of Q for Tunable complexity bandwidth manager.

particular, TCBM has been developed to improve the performance of LIV-MIG in the case where $\alpha = 2$, since in these conditions the bandwidth manager implemented in [78] assumes a constant behavior and energy consumption much higher than Xen. Regarding the constraints on downtime, total migration time, maximum available migration bandwidth $\widehat{R}$ and the value of the memory dirty rate of the migrated VM $\overline{w}$, in all three scenarios were taken equal to those of 3G, so as to achieve comparable results (seen that the 3G appears to be the worst case scenario with respect 4G and

WiFi). Regarding the other parameters such as K_0 and $\mathcal{E}_{SETUP}$, each scenario uses its own values, which are listed at the beginning of previous Section 5.7.

Second, looking the Figs. 5.12(a), 5.13(a), 5.14(a) and 5.15, we can see that to the growth of Q, the energy (except in LIV-MIG,where always remained constant) tends to decrease to stabilize since the early values of Q. For this reason, I decided to choose values of Q very low (i.e., $Q = 1, 2$). In particular, in the next tests I decided to work with $Q = 1$. Do not forget that working with $Q = 1$ does not mean to have a single update rate, but rather we must remember that my method updates $Q + 2$ rates, because R_0 and $R_{I_{MAX}+1}$ are always updated in case of migration.

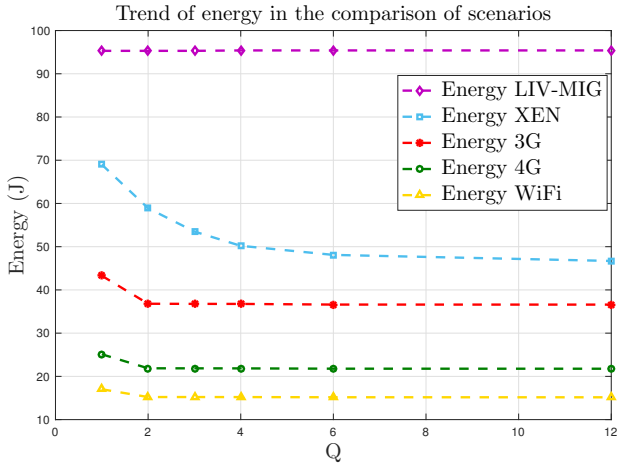
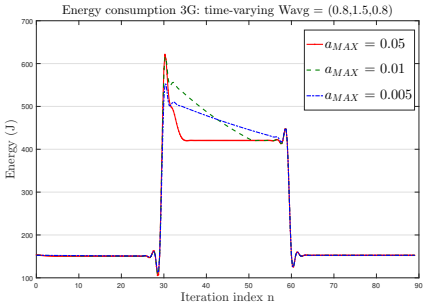


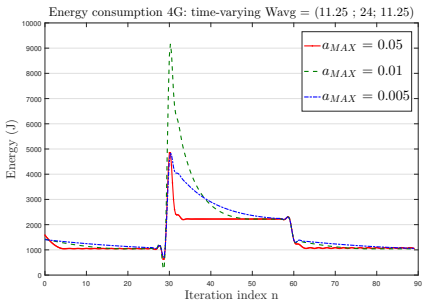
Figure 5.15. Trend of energy for the three scenarios 3G, 4G and WiFi, implemented by the TCBM; with $I_{MAX} = 12$ and $Q = 1, 2, 3, 4, 6, 12$ and with respect the results obtained using Xen and LIV-MIG manager.

Third, in Figs. 5.12(b), 5.13(b) and 5.14(b) are shown the timing of the tests performed on the three application scenarios, with respect to all values of Q. In all three cases, I have that to growing of Q, increases the time that my bandwidth manager uses to migrate from mobile devices to base station, but times continue to grow linearly.

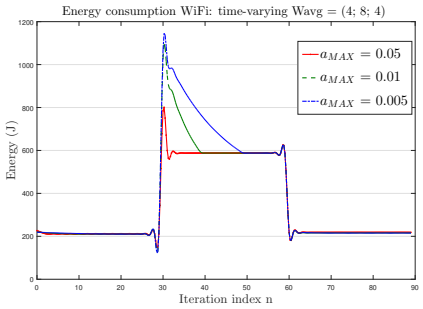
Fourth, even if the energy setup of WiFi results to be higher than 4G and in turn the energy setup of 4G turns out to be greater than that of 3G, given the energy formula (5.50) of TCBM, I have that the is energy consumptions of WiFi have to be the best, followed by the 4G and ultimately 3G. This result is due to the fact that as regards the energy calculation, come into play the parameter K_0 (the effects of which were discussed in Section 5.3); then it appears that $K_0^{WiFi} \geq K_0^{4G} \geq K_0^{3G}$. The following is a summary graph of the curves of energy of all three scenarios compared.



(a) $\overline{w} = [0.8, 1.5, 0.8]$



(b) $\overline{w} = [11.25, 24, 11.25]$



(c) $\overline{w} = [4, 8, 4]$

Figure 5.16. Time evolutions (in the n index) of the energy consumption of the proposed bandwidth manager, case of time-varying $\overline{w}$, at: (a) $\widehat{R} = 1.8(Mb/s)$, $M_0 = 256(Mb)$, $\beta = 2$, $\Delta_{TM} = 1460\ (s)$, $\Delta_{DT} = 0.14\ (s)$, for 3G scenario; (b) $\widehat{R} = 45\ (Mb/s)$, $M_0 = 256\ (Mb)$, $\beta = 2.33$, $\Delta_{TM} = 58.6\ (s)$, $\Delta_{DT} = 5.61 \times 10^{-3}\ (s)$, for 4G scenario; (c) $\widehat{R} = 9.9\ (Mb/s)$, $M_0 = 256\ (Mb)$, $\beta = 2.33$, $\Delta_{TM} = 266\ (s)$, $\Delta_{DT} = 2.55 \times 10^{-2}\ (s)$, for WiFi scenario.

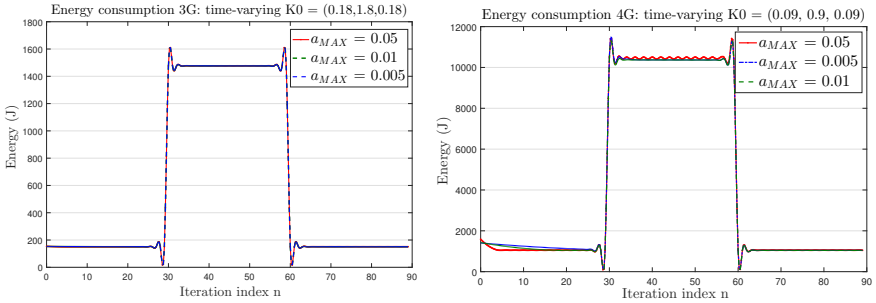
5.7.3 Tracking capabilities under contention phenomena

Real-world applications may vary the produced traffics over the time [47] and, then, it may be of interest to test how the proposed bandwidth manager reacts when the workload offered by the migrating VM changes unexpectedly.

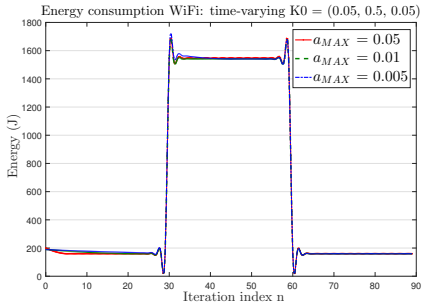
As pointed out in [16], memory contention phenomena and/or network congestions may produce abrupt (typically, unpredictable) time-variations of the parameters $\overline{w}$ (dirtied memory) and or K_0 (network status)

Hence, in order to evaluate the tracking capabilities of the proposed adaptive bandwidth manager in Eqs. (5.65)-(5.67) and its sensitivity to the parameters a_{MAX} in Eqs. (5.69)-(5.71), in Fig. 5.16, I report the measured behaviors of the energy sequence: $\{\mathcal{E}_{TOT}^{*(n)},\ n \geq 0\}$ when, due to memory contention phenomena, the memory dirty rate $\overline{w}$ of the running *memtester* application abruptly varies.

An examination to the plots of Fig. 5.16 and Fig. 5.17 supports the three following main conclusions.



(a) $K_0 = [0.18, 18, 0.18]$ (b) $K_0 = [0.09, 0.9, 0.09]$



(c) $K_0 = [0.05, 0.5, 0.05]$

Figure 5.17. Time evolutions (in the n index) of the energy consumption of the proposed bandwidth manager, case of time-varying K_0, at: (a) $\widehat{R} = 1.8\ (Mb/s)$, $M_0 = 256\ (Mb)$, $\beta = 2$, $\Delta_{TM} = 1460\ (s)$, $\Delta_{DT} = 0.14\ (s)$, for 3G scenario; (b) $\widehat{R} = 45\ (Mb/s)$, $M_0 = 256\ (Mb)$, $\beta = 2.33$, $\Delta_{TM} = 58.6\ (s)$, $\Delta_{DT} = 5.61 \times 10^{-3}\ (s)$, for 4G scenario; (c) $\widehat{R} = 9.9\ (Mb/s)$, $M_0 = 256\ (Mb)$, $\beta = 2.33$, $\Delta_{TM} = 266\ (s)$, $\Delta_{DT} = 2.55 \times 10^{-2}\ (s)$, for WiFi scenario.

- First, according to the fact that the energy function increases for increasing $\overline{w}$ and/or K_0, all the plots of Fig. 5.16 and Fig. 5.17 scale up at n = 30 and, then, scale down at n = 60.

- Second, the proposed bandwidth manager quickly reacts to abrupt unpredicted time variations of the migrating application and/or underlying network connections.

- Third, while virtually indistinguishable plots are obtained for a_{MAX} ranging over the interval $[5 \times 10^{-2}, 5 \times 10^{-3}]$ in case of time-varying K_0 (see. Fig. 5.17), the same results is not obtained in case of time-varying $\overline{w}$ (see. Fig. 5.17). This phenomenon is due to the fact that while K_0 is a multiplicative constant in the formula of the energy, $\overline{w}$, in addition to a profound impact on energy, causes that our TCBM uses more iterations to go from transient-states to the steady-states. Precisely, it is showed that, the decrease of a_{MAX} increases the number of iterations that are used by the software to return to the equilibrium

state.

For this reason I prefer to work with a_{MAX} high, over the interval $[0.5, 0.05]$, in such a way that (in a maximum of six or seven iterations), the software reacts well to variations of $\overline{w}$.

Overall, from the outset, I conclude that the proposed adaptive bandwidth manager is robust with respect to the actual tuning of a_{MAX}, at least for values of a_{MAX} ranging over the the interval $[0.5, 0.05]$, in order to exhibits the best trade-off among the contrasting requirements of short transient-states and stable steady-states.

5.7.4 Comparative energy simulations under random migration ordering and synthetic workload

The benchmark bandwidth management policy of the Xen hypervisor of Section 5.7.2 does not guarantee, by design, minimum energy consumptions and does not enforce QoS constraints on the resulting memory migration and stop-and-copy times. Furthermore, differently from $\tilde{I}_{MAX}$, in (5.46), the maximum number of allowed rounds: I_{MAX}^{XEN} is fixed by the Xen hypervisor in an application-oblivious way (typically, $I_{MAX}^{XEN} \leq 29$; see [51, 19]). Hence, in order to carry out *fair* energy comparisons, in the carried out field trials, I proceed as follows:

(i) set I_{MAX}^{XEN} and R_{MAX}^{XEN};

(ii) measure the resulting Xen energy consumption $\mathcal{E}_{TOT}^{XEN}$, speed-up factor β^{XEN}, total migration time T_{TM}^{XEN}, downtime T_{DT}^{XEN};

(iii) enforce $\widehat{R} \equiv R_{MAX}^{XEN}$, together with the QoS constraints: $\Delta_{TM} \equiv T_{TM}^{XEN}$, $\Delta_{DT} \equiv T_{DT}^{XEN}$, and $\beta \equiv \beta^{XEN}$;

(iv) measure the resulting energy consumption $\mathcal{E}_{TOT}^{*}$ of the proposed bandwidth manager at $I_{MAX} = \tilde{I}_{MAX}$. (see Eq. (5.46)).

The *memtester* [63] is the application, on the basis of which, I implement my simulations in this section and the implemented migration ordering of the dirtied memory pages is the random one.

The numerical results measured through a campaign of simulations developed for the three considered scenarios (3G, 4G and WiFi) as reported in Table 5.2, Table 5.3, Table 5.4, Table 5.5, Table 5.6 and Table 5.7. This application scenarios are characterized by different initialization values, but from the same ratio $(\overline{w}/\widehat{R})$ and from the same values of α and M_0. The other parameters are characteristics of each mobile scenario.

Table 5.2. Scenario 3G with $M_0 = 256(Mb)$; $\alpha = 2$; $K_0 = 0.18$; $E_{SETUP} = 3.25(J)$; and (a) $\left(\frac{\overline{w}}{\widehat{R}}\right) = 0.33$ and $\widehat{R} = 0.33 \times R_{MAX}^{XEN} = 0.55(Mb/s)$;

I_{MAX}^{XEN}	6	14	25
$T_{DT}^{XEN} = \Delta_{DT}$(s)	1.85	6.41×10^{-3}	2.67×10^{-6}
$T_{TM}^{XEN} = \Delta_{TM}$(s)	1210	1670	2140
β	1.97	2.06	2.09
$\mathcal{E}_{TOT}^{XEN}$(J)	138	156	178
$\mathcal{E}_{TOT}^{LIV\_MIG}$(J)	122	123	123
Q	1	1	1
$\mathcal{E}_{TOT}^{TCBM}$(J)	104.22	104.62	104.62
Energy save respect XEN (%)	24.5	32.9	41.2
Energy save respect LIV_MIG (%)	14.57	14.9	14.9

Table 5.3. Scenario 3G with $M_0 = 256(Mb)$; $\alpha = 2$; $K_0 = 0.18$; $E_{SETUP} = 3.25(J)$; and (b) $\left(\frac{\overline{w}}{\widehat{R}}\right) = 0.11$ and $\widehat{R} = 0.11 \times R_{MAX}^{XEN} = 0.2(Mb/s)$;

I_{MAX}^{XEN}	6	14	25
$T_{DT}^{XEN} = \Delta_{DT}$(s)	0.0155	1.14×10^{-7}	1.05×10^{-14}
$T_{TM}^{XEN} = \Delta_{TM}$(s)	2110	2740	3400
β	4.43	4.69	4.85
$\mathcal{E}_{TOT}^{XEN}$(J)	51	50.5	48.7
$\mathcal{E}_{TOT}^{LIV\_MIG}$(J)	96.6	96.6	96.6
Q	1	1	1
$\mathcal{E}_{TOT}^{TCBM}$(J)	45.69	46.76	46.51
Energy saved vs. XEN (%)	10.4	7.4	4.5
Energy saved vs. LIV_MIG (%)	52.7	51.6	51.8

The tables below cited show the energy values obtained through simulations for Xen, the bandwidth management policy developed in the Part I of this thesis, and the Tunable-complexity bandwidth manager, widely discussed in Section 5.3.

An examination of the results of these simulations data leads to two main conclusion.

First, in all the simulations the percent energy saving:

- $(1 - (\mathcal{E}_{TOT}^{*}/\mathcal{E}_{TOT}^{XEN}))\%$ of the proposed bandwidth manager over the Xen one is between 3% (minimum value of energy saving) for $(\overline{w}/\widehat{R}) = 0.11$ and $I_{MAX} = 25$, to 44.4% (maximum value of energy saving) for $(\overline{w}/\widehat{R}) = 0.33$ and $I_{MAX} = 25$ (see Tables 5.2, 5.4, 5.6);
- $(1 - (\mathcal{E}_{TOT}^{*}/\mathcal{E}_{TOT}^{LIV_MIG}))\%$ of the proposed bandwidth manager over the BMOP (Bandwidth Management Optimization Problem, see Part I and paper [78])

Table 5.4. Scenario 4G with parameters $M_0 = 256(Mb)$; $\alpha = 2$; $K_0 = 0.09$; $E_{SETUP} = 5.1(J)$; and (a) $\left(\frac{\overline{w}}{\widehat{R}}\right) = 0.33$ and $\widehat{R} = 0.33 \times R_{MAX}^{XEN} = 14.85(Mb/s)$;

I_{MAX}^{XEN}	6	14	25
$T_{DT}^{XEN} = \Delta_{DT}$(s)	0.103	5.42×10^{-4}	4.03×10^{-7}
$T_{TM}^{XEN} = \Delta_{TM}$(s)	46.9	65.2	83.6
β	1.87	1.95	1.98
$\mathcal{E}_{TOT}^{XEN}$(J)	1880	2150	2470
$\mathcal{E}_{TOT}^{LIV\_MIG}$(J)	1550	1550	1550
Q	1	1	1
$\mathcal{E}_{TOT}^{TCBM}$(J)	1366	1373	1373
Energy saved vs. XEN (%)	27.3	36.1	44.4
Energy saved vs. LIV_MIG (%)	11.8	11.4	11.4

Table 5.5. Scenario 4G (b) with parameters: $M_0 = 256(Mb)$; $\alpha = 2$; $K_0 = 0.09$; $E_{SETUP} = 5.1(J)$; and (b) $\left(\frac{\overline{w}}{\widehat{R}}\right) = 0.11$ and $\widehat{R} = 0.11 \times R_{MAX}^{XEN} = 4.95(Mb/s)$;

I_{MAX}^{XEN}	6	14	25
$T_{DT}^{XEN} = \Delta_{DT}$(s)	5.9×10^{-4}	4.07×10^{-9}	3.4×10^{-16}
$T_{TM}^{XEN} = \Delta_{TM}$(s)	84.9	110	137
β	4.47	4.78	4.89
$\mathcal{E}_{TOT}^{XEN}$(J)	632	624	602
$\mathcal{E}_{TOT}^{LIV\_MIG}$(J)	1170	1170	1170
Q	1	1	1
$\mathcal{E}_{TOT}^{TCBM}$(J)	531.7	545.25	541.8
Energy saved vs. XEN (%)	15.8	12.6	10
Energy saved vs. LIV_MIG (%)	54.5	53.4	53.7

is between 11.2% (minimum value of energy saving) for $(\overline{w}/\widehat{R}) = 0.33$ and $I_{MAX} = 6$, to 54.5% (maximum value of energy saving) for $(\overline{w}/\widehat{R}) = 0.11$ and $I_{MAX} = 6$ (see Tables 5.3, 5.5, 5.7).

In all scenarios, TCBM appears to be the best one from the point of view of energy saving. These noticeable energy gains support the conclusion that the bandwidth management policy developed in this thesis is the optimal one, and, by design, it minimizes the migration-induced energy consumption.

Second, the values of the measured energy gains mainly depend on the considered ratio: $(\overline{w}/\widehat{R})$. In particular, in these tests only values of $(\overline{w}/\widehat{R}) \leq 0.33$ are considered, because, if and only if this constraint is satisfies, the Xen (heuristic) bandwidth management policy presents decreasing values of energy for increasing values of I_{MAX}. Hence, under this condition, it make sense to compare our bandwidth

Table 5.6. Scenario WiFi (a) with $M_0 = 256(Mb)$; $\alpha = 2$; $K_0 = 0.05$; $E_{SETUP} = 5.9(J)$; (a) $\left(\frac{\overline{w}}{\widehat{R}}\right) = 0.33$ and $\widehat{R} = 0.33 \times R_{MAX}^{XEN} = 3.267(Mb/s)$;

I_{MAX}^{XEN}	6	14	25
$T_{DT}^{XEN} = \Delta_{DT}$(s)	0.468	2.46×10^{-3}	1.83×10^{-6}
$T_{TM}^{XEN} = \Delta_{TM}$(s)	213	296	380
β	1.87	1.95	1.98
$\mathcal{E}_{TOT}^{XEN}$(J)	229	267	302
$\mathcal{E}_{TOT}^{LIV\_MIG}$(J)	194	195	195
Q	1	1	1
$\mathcal{E}_{TOT}^{TCBM}$(J)	172.3	173.2	173.2
Energy saved vs. XEN (%)	24.7	35.1	42.6
Energy saved vs. LIV_MIG (%)	11.2	11.2	11.2

Table 5.7. Scenario WiFi (b) with $M_0 = 256(Mb)$; $\alpha = 2$; $K_0 = 0.05$; $E_{SETUP} = 5.9(J)$; (b) $\left(\frac{\overline{w}}{\widehat{R}}\right) = 0.11$ and $\widehat{R} = 0.11 \times R_{MAX}^{XEN} = 1.089(Mb/s)$;

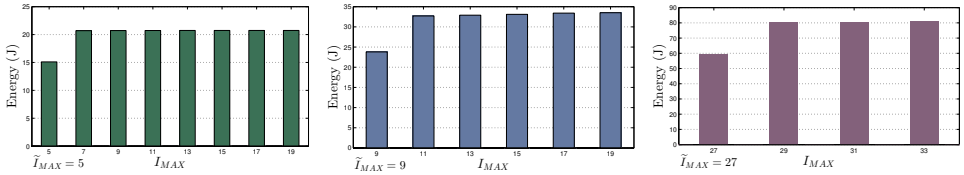
I_{MAX}^{XEN}	6	14	25
$T_{DT}^{XEN} = \Delta_{DT}$(s)	2.68×10^{-3}	1.85×10^{-8}	1.55×10^{-15}
$T_{TM}^{XEN} = \Delta_{TM}$(s)	386	501	622
β	4.47	4.78	4.89
$\mathcal{E}_{TOT}^{XEN}$(J)	77.2	76.3	73.6
$\mathcal{E}_{TOT}^{LIV\_MIG}$(J)	148	148	148
Q	1	1	1
$\mathcal{E}_{TOT}^{TCBM}$(J)	70.26	71.9	71.5
Energy saved vs. XEN (%)	9	5.7	3
Energy saved vs. LIV_MIG (%)	52.5	51.4	51.7

manager with Xen and BMOP.

In the carried out tests, is reported that, while the TCBM in each scenario presents a constant gain with respect to the optimization method described in Chapter 4 and [78], from the comparison with Xen comes out that the percentage of energy saving tends to decrease (for increase of I_{MAX}) when the ratio $(\overline{w}/\widehat{R}) < 0.33$; on the contrary the percentage of energy saving tends to increase when the ratio $(\overline{w}/\widehat{R}) = 0.33$.

5.7.5 Comparative simulations under random migration ordering and real-world workloads

In order to further validate and refine the above conclusions by considering also real-world applications, in this section we report and compare the migration-induced



(a) 3G Scenario (b) 4G Scenario (c) WiFi Scenario

Figure 5.18. Energy consumptions obtained by simulating *bzip2*, *mcf* and *memcached* in : (a) 3G scenario; (b) 4G scenario; (c) WiFi scenario.

energy consumptions which are suffered by the *bzip2*, *mcf* and *memcached* programs, also in this case each simulation is carried out for each of the three scenarios (3G, 4G and WiFi) by which the mobile device can migrate to the base station or Fog site. The test parameters are reported, in Appendix A.8, in Tables A.1, A.1 and Table A.1.

In all the experiments, R_{MAX} was chosen equal to the value of R_{MAX} of 3G (which turns out to be smaller, than those in the 4G and WiFi), in such a way to have the comparisons in a consistent manner.

The Figures 5.18 show the results of the tests.

An examination of the bar plots of Fig.5.18 leads to two main conclusion.

First, since the dirty rate increases by passing from the (read-intensive) *bzip2* program to the (write-intensive) *memcached* one, the corresponding energy consumptions also exhibit increasing trends under both the Xen, LIV-MIG Chapter 4 and proposed bandwidth managers.

Second, in all cases, the energy consumption relating to the migration by applying our method appears to be lower than both Xen and LIV-MIG manager. In particular, the percent energy saved of the proposed manager over the Xen and the LIV-MIG under the *bzip2*, *mcf* and *memcached*, for each application scenarios are reported in Table 5.8.

Table 5.8. Percent energy saved of the TCBM manager over Xen and LIV-MIG managers.

	Parameter description	***bzip2***	***mcf***	***memc.***
3G	Energy saving resp. Xen(%)	28.1	41.92	44.74
	En. saving resp. LIV-MIG(%)	32.8	11.15	6.67
4G	Energy saving resp. Xen(%)	60.5	69.21	70.84
	En. saving respect LIV-MIG(%)	63.17	52.86	50.76
WiFi	Energy saving resp. Xen(%)	75.04	81.33	82.46
	En. saving resp. LIV-MIG(%)	76.67	71.41	70.37

This confirms the trend of the previous Section 5.7.4 about the large energy-gains offered by the proposed manager under write-intensive applications.

5.8 Conclusion to Part II

In this chapter I presented a novel approach for bandwidth management in live migration virtual machine on wireless application context which is an extension of my manger LIV-MIG and outperform the others considered approaches. My results show a significant improvement with respect to the currently used approach in most relevant implementation architecture for live virtual machines migration.

CHAPTER 6
CRITICAL ASSESSMENT AND FUTURE WORK

Unlike in past years, nowadays we have smart phones and mobile devices in general (e.g. laptop) that have multiple radio interfaces, such as WiFi and 4G and/or 3G. In the event that these NICs allow the connection to the Internet, then the mobile device is defined to be multi-homed, this means that the device has different IP addresses and can connect simultaneously to the Internet in different location. Although at first glance multi-homing seems to be very interesting, we can not disregard the fact that neither TCP nor UDP support its implementation in an optimal manner. The situation can become destructive in the case of mobile device, in which the battery life is a key parameter for the client that wants to buy a new smart phone. Multi-path TCP is proposed as solution to the problem of shift session active on different path (see. [79, 80, 75]).

6.1 VM migration under single-path TCP and multi-path EWTCP

The main objective of MPTCP is to allow the simultaneous use of different network paths to a single TCP connection, but it can be further used for shift a connection from one location to another. Consider the following energy model of the interfaces of your device. It can be considered three states: *transmission energy*, if the data are transmitted; *high stand-by energy*, following the end of data transmission, and the state of *low energy*, which occurs when the stand-by phase ends [81].

Let us suppose we have a WiFi interface and 3G interface, and suppose further that they have the same consumption of idle phase. In the case of continuous shift from one interface to another (due to changes in throughput), when the device steps from 3G to WiFi, the stand-by energy of 3G is added to the energy that WiFi uses

for the transmission phase, and vice versa. Therefore, in the case of very frequent fluctuations of the throughput, even if each of them were chosen the best path, you would have a huge waste of energy relative to the ever more frequent phases of stand-by of the two interfaces. On the basis of these observations, we illustrate a case in which to turn on simultaneously the two NICs of a smart phone could be more efficient from the point of view of energy saving compared to using only one. In particular, the purpose of this chapter is to evaluate $\mathcal{E}_{TOT}$ of migration of our TCBM and Xen under single-path TCP and under multi-path EWTCP [79], so that we can numerically prove the goodness of the previous assumption.

The reference scenario is that of a radio connection 5G from a mobile device to your AP (Access Point), which acts like Fog site. The mobile device is equipped with two NICs (Network Interface Cards), the WiFi 802.11b and the 4G.

In the case of single-path TCP, the two NICs are turned on separately and used separately, while in the case of EWTCP the two NICs are turned on simultaneously and used simultaneously. Hence, the solutions single-path have a lower $\mathcal{E}_{SETUP}$ (which depend on the single NIC used), while the solution EWTCP has $\mathcal{E}_{SETUP}$ which is the sum of the energies of setup: $\mathcal{E}^{4G}_{SETUP} + \mathcal{E}^{WiFi}_{SETUP}$ of the two NICs used simultaneously.

6.2 Goodput-vs-Power in multi-path TCP working in the Congestion Avoidance state

The key criterion for calculating the average goodput of a TCP connection (or sub-connection) in the state of Congestion Avoidance is the following: at full performance, the size $\overline{w}$ (measured in segments size (MSS (byte)) of the congestion window must not vary. Hence, $(1 - p^{LOSS}) \times \overline{I}_{\overline{w}} = p^{LOSS} \times \overline{D}_{\overline{w}}$, where:

- p^{LOSS} is the average rate of the lost segments of the considered flow, in steady state;
- $\overline{I}_{\overline{w}}$ is the average rate of increase of the events;
- $\overline{D}_{\overline{w}}$ is the average decrease of $\overline{w}$.

Considering that $p^{LOSS} \leq 1$ in the state of Congestion Avoidance, the previous becomes:

$$\overline{I}_{\overline{w}} = p^{LOSS} \times \overline{D}_{\overline{w}}. \tag{6.1}$$

With reference to a multi-path TCP connection, we place:

- R $\triangleq$ set of routes that use the connection, $|R| \geq 1$;

- $\overline{w}^{TOT} \triangleq \sum_{r \in R} \overline{w}_r$ (segments);
- $\overline{RTT} \triangleq$ average round-trip-time of r-th subflow;
- $\overline{R}^{TOT} \triangleq \sum_{r \in R} \overline{R}r =$ total goodput of the multi-path TCP connection (Mbit/s);
- $\overline{R}_r(Mbit/s) \triangleq$ goodput of r-th subflow;
- $P_r(Watt) \triangleq$ power required to transfer/receive the r-th subflow;
- $P^{TOT} \triangleq \sum_{r \in R} P_r =$ total power needed to support the entire connection multi-path (Watt).

6.2.1 Single-path TCP Reno

In this case we have that:

$$P^{TOT} \equiv P = K_0(\overline{R})^{\alpha}, \ \alpha \geq 1 \, (Watt); \tag{6.2}$$

$$R^{TOT} \equiv \overline{R} \, (Mbit/s). \tag{6.3}$$

6.2.2 Multi-path EWTCP

This version uses simultaneously all paths available. In particular, we obtained

$$P^{TOT} \triangleq \sum_{r \in R} P_r \equiv a(\overline{R}_1, \ldots, \overline{R}_N) \left\{ \sum_{r \in R} K_r (\overline{R}_r)^c \right\}, \quad (Watt); \tag{6.4}$$

where we have that:

$$a(\overline{R}_1, \ldots, \overline{R}_N) \equiv 1; \tag{6.5}$$

$$K_r \triangleq [(\overline{RTT}_r \sqrt{\Omega_r})/(MSS\sqrt{2a})]^{\alpha}, \quad where \, MSS \, is \, measured \, in \, (Mbit); \tag{6.6}$$

$$c \triangleq \alpha = \frac{2}{m}; \tag{6.7}$$

$$R^{TOT} \triangleq \sum_{s \in R} \overline{R}_s, \quad (Mbit/s). \tag{6.8}$$

The relationship in Eq.(6.4) between P^{tot} and $\{\overline{R}_s, \, s \in R\}$ is not monomial neither posynomial so that, the problem of VM migration is no longer Geometric.

However, it is not convex and the type of convexity depends on the type of multi-path TCP considered, in our case multi-path EWTCP.

At this regard, we note specifically that in the problem of the migration of VM we have that:

- the power P is the total power on $N \geq 1$ paths, that is

$$P \equiv P^{TOT} \triangleq \sum_{r \in R} P_r \ (Watt); \tag{6.9}$$

- the migration rate R_i at i-th round, $i = 0, \ldots, (I_{MAX}+1)$ should be considered as the total rate on N paths of migration to the i-th round, i.e.

$$R_i \equiv R_i^{TOT} \triangleq \sum_{r \in R} \overline{R}_r^{(i)}, \quad i = 0, \ldots, (I_{MAX}+1). \tag{6.10}$$

So, if we want to continue to use the framework already developed for the single-path TCP, we have to find a case in which, even in case of multipath TCP, the relationship from $P^{TOT} \triangleq \sum_{r \in R} P_r$ and $R^{TOT} \triangleq \sum_{r \in R} R_r$ is monomial:

$$P^{TOT} = K_0 (R^{TOT})^{\alpha}, \quad \alpha \geq 1, (Watt) \tag{6.11}$$

with R^{TOT} measured in (Mb/s) and K_0 measured in $\left(\frac{Watt}{(Mbit/s)^{\alpha}} \right)$.

This is what we will do in the next section.

6.3 Total power-vs-total throughput monomial relationship in the case of Equal-Balanced multi-path TCP

Given Eqs. (6.9) and (6.10); is R the set of end-to-end routes that make up the considered multi-path TCP connection; and is N the number of routes that compose the TCP connection, with $N \geq 1$.

Then, the following result is worth.

Proposition 7. We assume that all $N \geq 1$ routes available carry the same throughput, i.e. we assume that:

$$\overline{R}_1 = \overline{R}_2 = \overline{R}_3 = \cdots = \overline{R}_N \equiv \overline{R} \quad (Mbit/s) \tag{6.12}$$

where $\overline{R}\,(Mbit/s)$ is the throughput for-path.

Under the assumption (6.12), we have that:

a) R^{TOT} is equal to:

$$R^{TOT} \triangleq \sum_{r \in R} \overline{R}_r \equiv N\overline{R} \quad (Mbit/s); \tag{6.13}$$

b) in all the considered cases of single-path TCP and multi-path EWTCP, the relation from total power (Eq. (6.9)) and total throughput (Eq. (6.13)) is monomial and takes the common form of Eq. (6.11);

c) the value of K_0 depends on the type of single-path or multi-path TCP considered. Directly placing (6.12) in formal expressions (6.4) and (6.5), we get the following explicit expressions of Ko at Eq. (6.11) for the two considered cases:

$$K_0 = \left(\frac{\overline{RTT}}{MSS}\sqrt{\frac{\Omega}{2}}\right)^{\alpha}, \quad \alpha \triangleq 2/m \geq 1, \qquad for\ single-path\ TCP\ Reno; \tag{6.14}$$

$$K_0 = \left(\frac{1}{MSS\sqrt{2a}\,N}\right)^{\alpha}\left[\sum_{r=1}^{N}(\overline{RTT}_r\sqrt{\Omega_r})^{\alpha}\right], \quad \alpha \triangleq 2/m \geq 1, \quad a \geq 1, \qquad for\ EWTCP; \tag{6.15}$$

with R^{TOT} (Mb/s), K_0 $\left(\frac{Watt}{(Mbit/s)^{\alpha}}\right)$, P^{TOT} $(Watt)$, $\overline{RTT}$ (s), MSS $(Mbit)$, Ω $((Watt)^{\frac{2}{\alpha}})$.

Because the Eqs. (6.11) and (6.15) are apply only under the assumption (6.13), we ask what is the sufficient conditions so that the considered multi-path EWTCP, actually satisfy the condition (6.13). At this regard, the following condition apply sufficient.

Proposition 8. The sufficient condition that ensure that multi-path EWTCP meets the condition (6.13), is:

$$\overline{RTT}_r\sqrt{\Omega_r} \equiv cost, \qquad \forall\, r = 1, 2, \ldots, N. \tag{6.16}$$

6.4 Additional considerations on multi-path EWTCP

The solution multi-path TCP equal-balanced considered, has a diversity gain due to multi-path, and is reflected on the fact that K_0 in Eq. (6.15) decreases like $\frac{1}{N^{\alpha-1}}$ to growing of N. It is worth the following proposition.

Proposition 9. Suppose to put a = 1 in Eq. (6.15). Then, the corresponding value of K_0 decreases as $\frac{1}{N^{\alpha-1}}$ for N growing, i.e.:

$$K_0 \propto \theta \left(\frac{1}{N^{\alpha-1}} \right). \tag{6.17}$$

Proof. The proof is obtained for inspection of the corresponding expression of K_0 given in Eq. (6.15) □

The effect of increase (decrease) a is, in all cases, to increase (decrease) the additive increment of each congestion windows $\{\overline{w}_r, \quad r \in R\}$ to the reception of an ACK on route r-th. That is, for every ACK received on the route r-th, $\{\overline{w}_r$ increases additively by an amount that depends on the type of TCP, but considered that, in any case is proportional to a. This can lead to multi-path versions of TCP that are "not friendly" with the single-path TCP but, as we shall see, provides a gain of multi-path that can more than offset the overhead power due to the use interfaces simultaneously. In the following, we focus on EWTCP with a = 1 in Eq. (6.15), in order to reduce to Eq. (6.14) when $N = 1$.

In addition, we will choose the parameters $\overline{RTT}$ and Ω of the two network WiFi and 4G so that the condition is satisfied:

$$\overline{RTT}_{WiFi} \sqrt{\Omega_{WiFi}} \equiv \overline{RTT}_{4G} \sqrt{\Omega_{4G}} \tag{6.18}$$

so that the EWTCP considered actually fairly balanced R^{TOT} on the two paths available. The condition (6.18) is almost always satisfied in practice (see [79]).

The choice of EWTCP to represent the multi-path TCP is due to the fact that the evidence given in [79], have already shown that, among the various multi-path TCP considered, the EWTCP is the one that best leads in the mobile-wireless environment.The following table shows the parameters for connections WiFi and 4G to simulate. I note explicitly that the parameters of Table 6.1 meet Eq. (6.18) and

Table 6.1. Parameters of the two radio connections of tests. In particular, 0.9 is the maximum efficiency (i.e., minimum overhead) due to the TCP+IP+MAC header's lengths.

	IEEE 802.11b (WiFi)	4G
α	2	2
MSS (Mbit)	0.008	0.008
R_{MAX} (Mbit/s)	$0.9 \times 11(Mb/s)$	$0.9 \times 50(Mb/s)$
$\overline{RTT}$(s)	25×10^{-3}	35×10^{-3}
Ω(Watt)	10^{-2}	5.1×10^{-3}
E_{SETUP}(J)	5.9	5.1

then, by construction, the corresponding EWTCP balance equally the total flow R^{TOT} which gives rise, on both WiFi and 4G available connections.

6.5 Definition of the simulation setup for comparisons

The purpose of this set of simulations is to compare the energy consumption of $\mathcal{E}_{\mathcal{TOT}}$ (J) of our TCBM to vary of M_0, Q, Δ_{TM}, Δ_{DT} and I_{MAX} on the three types of connections that now we will define.

a) **Single-path TCP New Reno connection under WiFi:** On the basis of the values of Tab. 6.1 and using the Eq. (6.14) for the calculation of the corresponding K_0, this test scenario is characterized by the following parameters of the TCP connection:

$$\begin{cases} K_0^{WiFi} = 5 \times 10^{-2} \left(\dfrac{Watt}{(Mb/s)^2} \right); \\ R_{MAX}^{WiFi} = 0.9 \times 11(Mb/s); \\ \alpha = 2; \\ E_{SETUP}^{WiFi} = 5.9(J). \end{cases} \tag{6.19}$$

b) **Single-path TCP New Reno connection under 4G:**

On the basis of the values of Tab. 6.1 and using the Eq. (6.14) for the calculation of the corresponding K_0, this test scenario is characterized by the following parameters of the TCP connection:

$$\begin{cases} K_0^{4G} = 5 \times 10^{-2} \left(\dfrac{Watt}{(Mb/s)^2} \right); \\ R_{MAX}^{4G} = 0.9 \times 50(Mb/s); \\ \alpha = 2; \\ E_{SETUP}^{4G} = 5.1(J). \end{cases} \tag{6.20}$$

c) **Multi-path EWTCP connection that uses simultaneously and in a balanced way both links 4G and WiFi:**

On the basis of the values of Tab. 6.1 and using the Eq. (6.15) with $N = 2$ and $a = 1$ for the calculation of the corresponding K_0, this test scenario is

characterized by the following parameters of dual-path TCP connection:

$$\begin{cases} K_0^{EWTCP} = 25 \times 10^{-3}\left(\dfrac{Watt}{(Mb/s)^2}\right); \\ R_{MAX}^{EWTCP} = 2min\left\{R_{MAX}^{WiFi}\ ;\ R_{MAX}^{4G}\right\} = 2 \times 9.9(Mb/s); \\ \alpha = 2; \\ E_{SETUP}^{EWTCP} = E_{SETUP}^{WiFi} + E_{SETUP}^{4G} = 11(J). \end{cases} \tag{6.21}$$

About the three scenarios now defined, note the following.

First,the purpose of the comparisons is to ascertain if it is energetically more convenient migrate using only the WiFi (case a) or only the 4G connection (case b) or both connections simultaneously, multi-path EWTCP (case c).

Second, because EWTCP uses both connections in parallel, E_{SETUP}^{EWTCP} is the summation (and therefore larger) of E_{SETUP}^{WiFi} and E_{SETUP}^{4G}. On the other hand, because EWTCP can balance the total flow on the two links available thus reducing the level of congestion, K_0^{EWTCP} is half of the corresponding K_0^{WiFi} and K_0^{4G}. Third, because EWTCP distributes evenly the total flow R^{TOT} between the two links available, the maximum value of R^{TOT} of EWTCP can not be greater than twice the smallest of the maximum flows of the two links.

6.6 Simulation results and conclusions

The results shown below are made by implementing in all three cases (a, b, c of Section 6.5) our TCBM, for various values of I_{MAX}, M_0, Q, Δ_{TM}, Δ_{DT}, obviously after having checked the feasibility of all three cases (Eqs. (5.43)-(5.45)). Even if E_{SETUP}^{EWTCP} is equal to the sum of E_{SETUP}^{WiFi} and E_{SETUP}^{4G}, while K_0^{EWTCP} is equal to half of the corresponding K_0^{WiFi} and K_0^{4G}, I expect that the energy saving due to the smaller K_0^{EWTCP} is greater then the energy increase induced by E_{SETUP}^{EWTCP}.

So, I expect that the simulated results of TCBM confirm the following hierarchy,

$$E_{SETUP}^{EWTCP} \ < \ min\left\{E_{TOT}^{WiFi}\ ;\ E_{TOT}^{4G}\right\} \tag{6.22}$$

with the inequality more closer for greater M_0 and smaller values of Δ_{TM} and Δ_{TM} (always ensuring the feasibility of the cases a, b, c).

The Tables 6.2, 6.3, 6.4 show the results obtained from TCBM, in the case where migration is performed by using only WiFi connection, or only 4G, or in the case in which are used simultaneously both connections WiFi and 4G. Now we can make some observations. First of all, is immediately evident that the energy consumed during the migration phase, except in the last case of table 6.2, appears to be always

Table 6.2. Results of simulation of scenarios with: (a) $M_0 = 64(Mb)$ and $\Delta_{DT} = 0.2(s)$.

$\frac{\overline{w}}{R_{MAX}}$	$\Delta_{TM}(s)$	I_{MAX}	$\mathcal{E}_{TOT}^{4G}(J)$	$\mathcal{E}_{TOT}^{WiFi}(J)$	$\mathcal{E}_{TOT}^{EWTCP}(J)$	$Gain_{4G}(\%)$	$Gain_{WiFi}(\%)$
0.1	7.2	1	73.95	40.93	32	56.76	21.82
0.2	8.1	2	119.11	45.3	39.01	67.25	13.89
0.3	9.17	2	171.43	50.68	49.88	70.9	1.6
0.35	9.89	3	197.64	54.29	54.48	72	-0.53

Table 6.3. Results of simulation of scenarios with: (b) $M_0 = 128(Mb)$ and $\Delta_{DT} = 0.2(s)$.

$\frac{\overline{w}}{R_{MAX}}$	$\Delta_{TM}(s)$	I_{MAX}	$\mathcal{E}_{TOT}^{4G}(J)$	$\mathcal{E}_{TOT}^{WiFi}(J)$	$\mathcal{E}_{TOT}^{EWTCP}(J)$	$Gain_{4G}(\%)$	$Gain_{WiFi}(\%)$
0.1	14.36	1	157.2	76.13	53.87	65.73	29.24
0.2	16.14	2	244.5	84.86	68.34	72.05	19.47
0.3	18.43	3	344.27	96.03	88.06	74.42	8.3
0.35	19.8	3	398.75	102.38	100.58	74.7	1.75

Table 6.4. Results of simulation of scenarios with: (c) $M_0 = 256(Mb)$ and $\Delta_{DT} = 0.2(s)$.

$\frac{\overline{w}}{R_{MAX}}$	$\Delta_{TM}(s)$	I_{MAX}	$\mathcal{E}_{TOT}^{4G}(J)$	$\mathcal{E}_{TOT}^{WiFi}(J)$	$\mathcal{E}_{TOT}^{EWTCP}(J)$	$Gain_{4G}(\%)$	$Gain_{WiFi}(\%)$
0.1	29	2	271.55	145.52	95.45	64.85	34
0.2	32.4	3	475.75	163.88	124.53	73.82	24
0.3	37	4	689.96	186.51	164.47	76.16	11.82
0.35	39.71	4	801.99	200.28	189.73	76.3	5.26
0.4	43.1	5	915.75	216.48	212.55	76.79	1.8

less than energy consumed which is obtained by using separately or WiFi, or 4G, in agreement with the Eq. (6.22), as we hoped.

Second, the table shows that while the increasing of I_{MAX}, the gain of the EWTCP increases with respect to 4G, in the same time decreases compared to the WiFi. Because R_{MAX}^{4G} turns out to be much greater than R_{MAX}^{WiFi}, you get a result that the multi-path EWTCP has an $R_{MAX}^{EWTCP} = 2 \times R_{MAX}^{WiFi}$. It appears that 4G go worse in all cases, but this is not really true, since the value of energy is distorted by the fact that, unlike the WiFi and EWTCP (until $\overline{w}/R_{MAX}$ remains low, equal to 0.4 in the best case of tables below), 4G satisfies the constraint on the total migration, but uses it for a period of time so short that accordingly to meet the other constraints of feasibility, waste a lot of energy during the migration.

In conclusion, the results show that for low values of the ratio $\overline{w}/R_{MAX}$ (at the most equal to $\overline{w}/R_{MAX} = 0.4$ in case of $M_0 = 256(Mb)$) there are cases where it is more convenient to use together both NICs WiFi and 4G instead of keeping turned on only one of the two NICs. This is the most important result from the application of multi-path EWTCP to the problem of migration between smart phone and AP, because it could seem absurd to think that keeping alight both the WiFi adapter that 4G, the smart phone consume less than if it is on one or the other NIC. Obviously these results are obtained in particular operating conditions, such

as to have the same rate on both paths (see Eq. (6.18)), in addition to all the other assumptions made in the preceding Sections 6.3 - 6.5.

CHAPTER 7

SUMMARY CONCLUSION

In this thesis I developed an optimal bandwidth manager for live migration of virtual machines (LIV-MIG), in Part I I presented the intra-data-center network bandwidth manager for live migration of virtual machine, hence in Part II I provided an extension of my bandwidth manager for wireless live migration, with tunable-complexity capabilities, for future's 5G technologies.

In Part I I developed the optimal bandwidth manager for intra-data-center live VM migration. It minimizes at run-time the communication energy wasted by the migration of the VM memory under *hard* QoS constraints on both the migration time and downtime. After implementing it atop a wired test-bed, I measured and compared its energy performance through extensive simulations and tests by considering synthetic and real-world workloads, as well as random and ordered migration scheduling disciplines. The carried out simulations and tests highlight that the average energy saving of the proposed bandwidth manager over the corresponding state-of-the-art Xen one is over 40% and approaches 66% under strict constraints on the tolerated downtimes. Interestingly, the measured per-migration CPU slow-down induced by its implementation is, in average, limited up to $1.5-2\%$, while the measured average stretching of the execution times of the migrated applications is under 20%.

In second part of this thesis I developed the optimal tunable-complexity bandwidth manager (TCBM) for live VM migration from mobile device to access point (AP)/Fog site.

First because right now there are no mobile devices capable to migrate, and second, because APs are usually only used as a point to access to Internet, and they do not have any kind of computing capability. Which it is assumed that will be introduced only with the birth of the new mobile technology 5G.

In particular, my TCBM minimizes at run-time the communication energy wasted by the migration of the VM memory under hard QoS constraints on both the migration time and downtime. After implementing it atop a wireless test-bed, I measured its energy performance compared to Xen and LIV-IMG manager, through extensive simulations by considering synthetic and real-world workloads, in the three scenarios of 3G, 4G and WiFi. As reported in Chapter 5, all the tests carried out show the goodness of our bandwidth manager compared to XEN and LIV-MIG, in any scenario in which they were made the simulations. Hence we obtained a more efficient use of the available bandwidth and, at the same time, a considerable energy saving.

Therefore, in the following I considered the critical assessment and possible future work about my thesis distinguishing the two part of this work.

Regarding hints for future research in Part I, I point out that the reported BMOP formulation of Eqs. (4.6), (4.7) may apply verbatim to live VM migration over WAN connections, provided that the adopted power-rate function in (3.8), setup energy in (3.11) and round-trip-time in (3.9) are suitable modeled. In fact, I stress that, from a networking perspective, guaranteeing seamless migration of IP-addressed VMs over WANs poses quite specific challenges. First, in NAS-equipped LAN environments, it suffices that, during the stop-and-copy phase, the destination server broadcasts an unsolicited ARP reply message, in order to advertise all the LAN interfaces that the IP address of the migrated VM has been moved to a new server. However, more challenging is to attain seamless traffic redirection when the VM migrates over different subnets [27]. Therefore, extending the actual implementation of the proposed bandwidth manager to WAN environments for the support of inter-data-center VM migration is a research topic currently investigated by the authors.

Second, the network topologies of WAN environments are not controlled by the data-center's designer, so that high bisection bandwidths are no longer guaranteed. Hence, in WAN environments, the performance effects of TCP re-transmissions cannot be longer neglected and multi-path TCP may become an appealing option, in order to attain end-to-end load balancing. As a consequence, the generalization of the power model of Eqs. (3.8) and (3.9) and the resolution of the resulting bandwidth management problem under multi-path TCP over WAN connections are additional hints for future research. Towards this end, the performance results recently presented in [82] and [83] may provide a good starting point.

Hence, concerning the future work about Part II, relative to bandwidth management for wireless channel, I firstly analyzed the phenomenon of multi-path TCP. Later, I realized a version of my TCBM able to work even in case the multi-path

TCP, and more particularly in its version EWTCP, that is best suited to work in a wireless-mobile environment. The comparison of the case Single-path with multi-path EWTCP (both implemented according to our TCBM) get a truly amazing result, but this case should be extended in future work. We have that to migrate from mobile devices to Fog site, under appropriate operating conditions (first of all that both paths have the same rate), is more advantageous in terms of energy saving keep lit simultaneously both network cards (and hence two energies of idle etc.) than take turned on only one of the two.

Regarding hints for future research, I point out that with the advent of 5G technology, the implementation of the software on mobile devices could be done in order to make them aware to decide whether to use a single or multiple network interface to migrate, depending on the conditions of the surrounding environment, so as to obtain significant energy savings.

APPENDIX A

APPENDIX

A.1 Proof of *Proposition 1*

After observing that the condition: $R \leq \widehat{R}$ is necessary and sufficient for meeting the constraint in (4.5), I begin to prove the sufficiency of the conditions in (4.20)-(4.22). Specifically, since the functions: $\Psi_i(R)$, $i = 1, 2, 3$, in (4.1)-(4.3) decrease for increasing R, it suffices that the corresponding constraints in (4.1)-(4.3) are met at $R = \widehat{R}$. Hence, posing $R = \widehat{R}$ in (4.2) and (4.3) directly leads to Eqs. (4.21) and (4.22), respectively. Furthermore, after observing that:

$$\sum_{i=0}^{I_{MAX}+1} (\overline{w}/\widehat{R})^i = \begin{cases} I_{MAX} + 2, \ for \ (\overline{w}/\widehat{R}) = 1, \\ \\ \left[\frac{1-(\overline{w}/\widehat{R})^{I_{MAX}+2}}{1-(\overline{w}/\widehat{R})}\right], \ for \ (\overline{w}/\widehat{R}) \neq 1, \end{cases} \tag{A.1}$$

I directly arrive at the condition in Eq. (4.20).

The proof of the necessary part is by contradiction. Hence, let us assume that at least one of the conditions in (4.1)-(4.3) fails at $R = \widehat{R}$. So, being the Ψ-functions in (4.1)-(4.3) decreasing for increasing R, in order to meet the failing constraint, I would set: $R > \widehat{R}$, that, in turn, would lead to violate the constraint in (4.5). This proves that the conditions in (4.20)-(4.22) are also necessary for the feasibility of the BMOP.

A.2 Proof of *Proposition 2*

In order to test that the convex problem in (4.24), (4.25) meets the Slater's qualification, it suffices to prove that there exists a feasible solution $\widetilde{R}^*$ at which the

box constraint in (4.25) is met with the equality, while the remaining nonlinear constraints: $\Psi_i(\widetilde{R}) \leq 0,\ i = 1, 2, 3$, are fulfilled with the strict inequality [56]. To this end, let us assume that all the feasibility conditions in (4.20)-(4.22) are met with the strict inequality. Hence, $R = \widetilde{R}$ in (4.4) is a feasible solution of the problem in (4.6), (4.7). Furthermore, since the conditions in (4.20), (4.21) and (4.22) are the constraints in (4.1), (4.2) and (4.3) evaluated at $R = \widehat{R}$, I deduce that all the constraints in (4.1)-(4.3) are met with the strict inequality at $R = \widehat{R}$. Therefore, since the exponential transformation is strictly increasing and, then, inequality preserving, I conclude that: $\widetilde{R}^* \triangleq \log \widehat{R}$ is a feasible solution of the problem in (4.24), (4.25) which meets all the nonlinear constraints: $\Psi_i(\widetilde{R}) \leq 0,\ i = 1, 2, 3$, in (4.25) with the *strict* inequality.

A.3 Expressions of the gradients of the Lagrangian function

The scalar gradients of the Lagrangian function in Eq. (4.26) assume the following expressions:

$$\begin{aligned}\nabla_{\widetilde{R}} L =& K_0 M_0 e^{(\alpha-1)\widetilde{R}} \left\{ (\alpha - 1) + \theta \left[\sum_{i=1}^{I_{MAX}+1} (\alpha - 1 - i) \left(\frac{\overline{w}}{e^{\widetilde{R}}} \right)^i \right] \right\} \\ & - \theta\lambda_1 \left(\frac{M_0}{\Delta_{MMT}} \right) e^{-\widetilde{R}} \left[\sum_{i=0}^{I_{MAX}+1} (1 + i) \left(\frac{\overline{w}}{e^{\widetilde{R}}} \right)^i \right] \\ & - \lambda_2 \left(\frac{M_0}{\Delta_{SC}} \right) (I_{MAX} + 2) e^{-\widetilde{R}} \left(\frac{\overline{w}}{e^{\widetilde{R}}} \right)^{I_{MAX}+1} - \theta\lambda_3\beta \left(\frac{\overline{w}}{e^{\widetilde{R}}} \right); \end{aligned} \tag{A.2}$$

$$\nabla_{\lambda_1} L = \theta \left[e^{-\widetilde{R}} \left(\frac{M_0}{\Delta_{MMT}} \right) \left(\sum_{i=0}^{I_{MAX}+1} \left(\frac{\overline{w}}{e^{\widetilde{R}}} \right)^i \right) - 1 \right]; \tag{A.3}$$

$$\nabla_{\lambda_2} L = \left[e^{-\widetilde{R}} \left(\frac{M_0}{\Delta_{SC}} \right) \left(\frac{\overline{w}}{e^{\widetilde{R}}} \right)^{I_{MAX}+1} \right] - 1; \tag{A.4}$$

$$\nabla_{\lambda_3} L = \theta \left[\beta\ \overline{w}\ e^{-\widetilde{R}} - 1 \right]; \tag{A.5}$$

A.4 Development of the final expression of the T_{MT} for the TCBM

In the following I develop the final expression of the total migration time (TMT) for the tunable complexity bandwidth manager (TCBM) developed in Section 5.3.

Let's start from

$$T_{TM} = M_0\Big\{\frac{1}{R_0} + \frac{1}{M_0}T_{DT} + (1-\delta(I_{MAX}))\Big\{\sum_{l=1}^{I_{MAX}}(\overline{w})^l\Big[\prod_{m=0}^{l}\Big(\frac{1}{R_m}\Big)\Big]\Big\}\Big\}, \quad \text{(A.6)}$$

with T_{DT} given by eq. (5.27) at $I_{MAX} \neq -1$.

Now, for each $l = 1, \ldots, I_{MAX}$, with $I_{MAX} \geq 1$, I have that:

$$\prod_{m=0}^{l}\Big(\frac{1}{R_m}\Big) = \frac{1}{R_0\,(R_1R_2\ldots R_S)\,(R_{S+1}R_{S+2}\ldots R_{2S})(R_{2S+1}R_{2S+2}\ldots R_{3S})\ldots R_{l-1}R_l} = \\ = \frac{1}{R_0\,(R_1)^S\,(R_{S+1})^S\,(R_{S+2})^S\,R_{l-1}\,R_l}, \quad l = 1, \ldots, I_{MAX}. \quad \text{(A.7)}$$

The number of factors in eq (A.7) which are raised to the S exponent depends on the values actually assumed by $l \in \{1, \ldots, I_{MAX}\}$ and S. Hence, in order to properly express the rate product at the denominator of eq. (A.7) in terms of the corresponding cluster-header rates $\{R_{jS+1}\ ,\ j = 0, \ldots, (Q-1)\}$ (see (5.23)), I develop the summation in eq. (A.6) as in:

$$\sum_{l=1}^{I_{MAX}}(\overline{w})^l\Big[\prod_{m=0}^{l}(R_m)^{-1}\Big] = \sum_{k=0}^{Q-1}\sum_{l=kS+1}^{(k+1)S}(\overline{w})^l\Big[\prod_{m=0}^{l}(R_m)^{-1}\Big]. \quad \text{(A.8)}$$

Now, for $l \in [kS+1, (k+1)S]$, the inner product may be expressed in terms of the cluster headers in eq. (5.23) as in:

$$\prod_{m=0}^{l}(R_m)^{-1} \equiv \begin{cases} (R_0\,R_1^l)^{-1} \quad , \quad for\ l \in [1, S];\ k = 0 \\ \dfrac{1}{R_0}\Big[\prod_{p=0}^{k-1}(R_{pS+1})^{-S}\Big]\Big(R_{kS+1}\Big)^{-l+kS}, for\ l \in [kS+1, (k+1)S];\ k \geq 0 \end{cases} \quad \text{(A.9)}$$

so that

$$\prod_{m=0}^{l}(R_m)^{-1} = \frac{1}{R_0}\Big\{\delta(k)(R_1)^{-l} + (1-\delta(k))\Big[\prod_{p=0}^{k-1}(R_{pS+1})^{-S}\Big](R_{kS+1})^{-l+kS}\Big\}, \\ for \quad l \in [kS+1, (k+1)S];\ k = 0, 1, \ldots, (Q-1). \quad \text{(A.10)}$$

Hence, by inserting eq. (A.10) into eq. (A.8) I obtain:

$$\sum_{l=1}^{I_{MAX}} (\overline{w})^l \Big[\prod_{m=0}^{l} (R_m)^{-1}\Big] = \frac{1}{R_0} \sum_{k=0}^{Q-1} \sum_{l=kS+1}^{(k+1)S} (\overline{w})^l \Big\{\delta(k)(R_1)^{-l} + (1-\delta(k))\Big[\prod_{p=0}^{k-1}(R_{pS+1})^{-S}\Big] * (R_{kS+1})^{-l+kS}\Big\} \tag{A.11}$$

Finally, after introducing eq. (A.11) and eq. (5.27) into eq. (A.6) I obtain the following final expression for T_{TM}:

$$\begin{aligned} T_{TM} = M_0 \Big\{ \frac{1}{R_0} + \overline{w}^{I_{MAX}+1} (R_0\, R_{I_{MAX}+1})^{-1} \Big[\prod_{m=0}^{Q-1} (R_{mS+1})^{-S}\Big] + \\ + (1-\delta(I_{MAX})) \frac{1}{R_0} \Big\{ \sum_{k=0}^{Q-1} \sum_{l=kS+1}^{(k+1)S} (\overline{w})^l \Big\{ \delta(k)(R_1^{-l}) + \\ + (1-\delta(k)) \Big[\prod_{p=0}^{k-1}(R_{pS+1})^{-S}\Big] (R_{kS+1})^{-l+kS} \Big\}\Big\}\Big\} \end{aligned} \tag{A.12}$$

A.5 Development of the expression of the energy wasted by the TCBM

In the following I develop the final expression for the energy wasted by the tunable complexity bandwidth manager.

According to eq. (5.26), the first rate-product in eq. (5.31) equates:

$$\prod_{k=0}^{I_{MAX}} (R_k)^{-1} = \frac{1}{R_0} \Big[\prod_{k=0}^{Q-1} (R_{kS+1})^{-S}\Big]. \tag{A.13}$$

Furthermore, the single summation over the l-index may be (once a time) equivalently rewritten as the following nested double summation:

$$\begin{aligned} \sum_{l=1}^{I_{MAX}} K_0 M_0\, (\overline{w})^l\, (R_l)^{\alpha-1} \Big[\prod_{m=0}^{l-1} (R_m)^{-1}\Big] \equiv \sum_{m=0}^{Q-1} \sum_{l=mS+1}^{(m+1)S} K_0 M_0 (\overline{w})^l (R_l)^{\alpha-1} * \Big[\prod_{k=0}^{l-1} (R_k)^{-1}\Big] = \\ = (\text{since all rates } \{R_l, mS+1 \le l \le (m+1)S\} \\ \text{are mapped into: } R_{mS+1}; \text{ see eq. ((5.24)))} = \\ = \sum_{m=0}^{Q-1} (R_{mS+1})^{\alpha-1} K_0 M_0 \Big\{ \sum_{l=mS+1}^{(m+1)S} (\overline{w})^l \Big[\prod_{k=0}^{l-1} (R_k)^{-1}\Big]\Big\} \end{aligned} \tag{A.14}$$

Now, as in eq. (A.10) (with l replaced by $(l-1)$) I have that:

$$\prod_{k=0}^{l-1}(R_k)^{-1} = \begin{cases} (R_0\, R_1^{l-1})^{-1} \quad , \quad for\, l \in [1,S] \\ \dfrac{1}{R_0}\left[\prod_{p=0}^{m-1}(R_{pS+1})^{-S}\right](R_{mS+1})^{mS+1-l}, for\, l \in [mS+1;(m+1)S]\; ;\; m \geq 1 \end{cases} \tag{A.15}$$

so that

$$\prod_{k=0}^{l-1}(R_k)^{-1} = \frac{1}{R_0}\left\{\delta(m)(R_1)^{1-l} + (1-\delta(m))\left[\prod_{p=0}^{m-1}(R_{pS+1})^{-S}\right](R_{mS+1})^{mS+1-l}\right\}, \quad for\;\; l \in [mS+1;(m+1)S]\;\; ;\;\; m = 0,1,\ldots,(Q-1). \tag{A.16}$$

Hence, after inserting eq. (A.16) in (A.14) I obtain:

$$\sum_{l=1}^{I_{MAX}} K_0 M_0\,(\overline{w})^l\,(R_l)^{\alpha-1}\left[\prod_{m=0}^{l-1}(R_m)^{-1}\right] \equiv K_0 M_0 (R_0)^{-1}\left\{\sum_{m=0}^{Q-1}\sum_{l=mS+1}^{(m+1)S}(\overline{w})^l\{\delta(m)(R_1)^{\alpha-l} + \right.$$
$$\left. +(1-\delta(m))\left[\prod_{p=0}^{m-1}(R_{pS+1})^{-S}\right](R_{mS+1})^{\alpha+mS-l}\right\}\Bigg\}\Bigg\} \tag{A.17}$$

Finally, after inserting eq. (A.17) into eq.(5.31), I arrive at the final expression for the energy wasted by the TCBM (see eq. (5.32))

A.6 Proof of Proposition 4

After observing that the condition: $R_i \leq \widehat{R}$ is necessary and sufficient for meeting the constraint in (5.40), I begin to prove the sufficiency of the conditions in (5.43)-(5.45). Specifically, since the functions: Ψ_i , $i = 1,2,3$, in (5.37)-(5.39) decrease when at least one of the bandwidths $\{R_i,\;\; i = 0,1,\ldots,I_{MAX}+1\}$ increase, it suffices that the corresponding constraints in (5.37)-(5.39) are met when all the bandwidths simultaneously assume the maximum allowed value, that is, at $R_i \equiv \widehat{R}$, $i = 0,1,\ldots,I_{MAX}+1$. Hence, posing $R_i = \widehat{R}$ in (5.38) and (5.39) directly leads to Eqs. (5.44) and (5.45), respectively.

Furthermore, after observing that:

$$\sum_{i=0}^{I_{MAX}+1}\left(\frac{\overline{w}}{\widehat{R}}\right)^i \equiv \begin{cases} I_{MAX}+2\;\; , \quad for \;\; (\overline{w}/\widehat{R}) = 1, \\ \dfrac{1-(\overline{w}/\widehat{R})^{I_{MAX}+2}}{1-(\overline{w}/\widehat{R})} \;\; , \quad for \;\; (\overline{w}/\widehat{R}) \neq 1, \end{cases} \tag{A.18}$$

I directly arrive at the condition in Eq. (5.43).

The proof of the necessary part is by contradiction. Hence, let us assume that at least one of the conditions in (5.37)-(5.39) fails at $R_i = \widehat{R}$. So, being the Ψ-functions in (5.37)-(5.39) decreasing for increasing R_i, in order to meet the failing constraint, I would set: $R_i > \widehat{R}$, that, in turn, would lead to violate the constraint in (5.40). This proves that the conditions in (5.43)-(5.45) are also necessary for the feasibility of the TCBM.

A.7 Expressions of gradients in Lagrangian function

The scalar gradients of the Lagrangian function in Eq. (5.60) assume the following expressions:

$$\nabla_{\tilde{R}_0} L = \left(\frac{\partial \mathcal{E}_{TOT}}{\partial \tilde{R}_0}\right) - \theta\,\lambda_1 \frac{1}{\Delta_{TM}} T_{TM} \;-\; \lambda_2 \frac{1}{\Delta_{DT}} T_{DT} \;-\; \theta\,\lambda_{3,0}\,\beta\,\overline{w}\,e^{[-\tilde{R}_0]}; \quad \text{(A.19)}$$

where:

$$\begin{aligned}\frac{\partial \mathcal{E}_{TOT}}{\partial \tilde{R}_0} =& (\alpha-1) K_0 M_0\, e^{[(\alpha-1)\tilde{R}_0]} - \theta K_0 M_0\, e^{[-\tilde{R}_0]} \Bigg\{ \\ & (\overline{w})^{1+I_{MAX}}\, e^{\left[(\alpha-1)\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]} + \\ & +(1-\delta(I_{MAX}))\left\{\sum_{m=0}^{Q-1}\left\{\sum_{l=mS+1}^{(m+1)S}(\overline{w})^l\left\{\delta(m)\,e^{[(\alpha-l)\tilde{R}_1]} + \right.\right.\right. \\ & \left.\left.\left. +(1-\delta(m))\, e^{\left[(\alpha+mS-l)\tilde{R}_{mS+1} - S\tilde{R}_1 - (1-\delta(m-1))\left(\sum_{p=1}^{m-1}\tilde{R}_{pS+1}\right)\right]}\right\}\right\}\right\}\Bigg\}. \end{aligned} \quad \text{(A.20)}$$

$$\nabla_{\tilde{R}_1} L = \left(\frac{\partial \mathcal{E}_{TOT}}{\partial \tilde{R}_1}\right) + \frac{\theta\,\lambda_1}{\Delta_{TM}}\left(\frac{\partial T_{TM}}{\partial \tilde{R}_1}\right) + \frac{\lambda_2}{\Delta_{DT}}\left(\frac{\partial T_{DT}}{\partial \tilde{R}_1}\right) - \lambda_{3,1}\,\beta\,\overline{w}\,e^{[-\tilde{R}_1]}; \quad \text{(A.21)}$$

where:

$$\frac{\partial T_{TM}}{\partial \tilde{R}_1} = - M_0\, e^{[-\tilde{R}_0]}\Bigg\{ + S(\overline{w})^{1+I_{MAX}}\; e^{\left[-\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]} + \\ + (1-\delta(I_{MAX}))\Bigg\{\sum_{k=0}^{Q-1}\Bigg\{\sum_{l=kS+1}^{(k+1)S}(\overline{w})^l\Bigg\{\delta(k)\; l\; e^{[-l\tilde{R}_1]} \; + \; (1-\delta(k))S* \\ * e^{\left[(kS-l)\tilde{R}_{kS+1} - S\tilde{R}_1 - (1-\delta(k-1))S\left(\sum_{p=1}^{k-1}\tilde{R}_{pS+1}\right)\right]}\Bigg\}\Bigg\}\Bigg\}\Bigg\}; \tag{A.22}$$

$$\frac{\partial T_{DT}}{\partial \tilde{R}_1} = -SM_0 e^{[-\tilde{R}_0]}(\overline{w})^{1+I_{MAX}}(1-\delta(1+I_{MAX}))e^{\left[-\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]}. \tag{A.23}$$

$$\frac{\partial \mathcal{E}_{TOT}}{\partial \tilde{R}_1} = - \; \theta K_0 M_0 e^{[-\tilde{R}_0]}\Bigg\{S(\overline{w})^{1+I_{MAX}}\; e^{\left[(\alpha-1)\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))\; S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]} + \\ + (1-\delta(I_{MAX}))\Bigg\{\sum_{k=0}^{Q-1}\Bigg\{\sum_{l=mS+1}^{(m+1)S}(\overline{w})^l\Bigg\{(l-\alpha)\;\delta(m)\; e^{[(\alpha-l)\tilde{R}_1]} \; + \; S(1-\delta(m))* \\ * e^{\left[(\alpha+mS-l)\tilde{R}_{mS+1} - S\tilde{R}_1 - (1-\delta(m-1))\left(\sum_{p=1}^{m-1}\tilde{R}_{pS+1}\right)\right]}\Bigg\}\Bigg\}\Bigg\}\Bigg\}. \tag{A.24}$$

$$\nabla_{\tilde{R}_{jS+1}} L = \left(\frac{\partial \mathcal{E}_{TOT}}{\partial \tilde{R}_{jS+1}}\right) + \frac{\theta\,\lambda_1}{\Delta_{TM}}\left(\frac{\partial T_{TM}}{\partial \tilde{R}_{jS+1}}\right) + \frac{\lambda_2}{\Delta_{DT}}\left(\frac{\partial T_{DT}}{\partial \tilde{R}_{jS+1}}\right) - \lambda_{3,(jS+1)}\;\beta\;\overline{w}\; e^{[-\tilde{R}_{jS+1}]}\,, \\ for\;\; j = 1, 2, \ldots, (Q-1); \tag{A.25}$$

where:

$$\frac{\partial T_{DT}}{\partial \tilde{R}_{jS+1}} = - \; S(1-\delta(Q-1))M_0\; e^{[-\tilde{R}_0]}\;(\overline{w})^{1+I_{MAX}}\;(1-\delta(1+I_{MAX}))* \\ * e^{\left[-\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]}, for j = 1, 2, \ldots, (Q-1); \tag{A.26}$$

$$\frac{\partial T_{TM}}{\partial \tilde{R}_{jS+1}} = M_0 e^{[\tilde{R}_0]} \Bigg\{ - (\overline{w})^{1+I_{MAX}} \; S(1-\delta(Q-1)) * \\ * \, e^{\left[-\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1} \tilde{R}_{kS+1} \right) \right]} + \\ + (1-\delta(I_{MAX})) \Bigg\{ \sum_{k=j}^{Q-1} \Bigg\{ \sum_{l=kS+1}^{(k+1)S} (\overline{w})^l \Bigg\{ \delta(k-j)(jS-l) * \\ * \, e^{\left[(jS-l)\tilde{R}_{jS+1} - S\tilde{R}_1 - (1-\delta(k-1))S\left(\sum_{p=1}^{j-1} \tilde{R}_{pS+1} \right) \right]} - \\ - (1-\delta(k-j))S(1-\delta(k-1)) \, e^{\left[(kS-l)\tilde{R}_{kS+1} - S\tilde{R}_1 - (1-\delta(k-1))S\left(\sum_{p=1}^{k-1} \tilde{R}_{pS+1} \right) \right]} \Bigg\} \Bigg\} \Bigg\} \Bigg\}, \\ for \;\; j = 1, 2, \ldots, (Q-1); \tag{A.27}$$

$$\frac{\partial \mathcal{E}_{TOT}}{\partial \tilde{R}_{jS+1}} = \theta K_0 M_0 \, e^{[\tilde{R}_0]} \Bigg\{ \\ - (1-\delta(Q-1)) \; S * \; e^{\left[(\alpha-1)\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1} \tilde{R}_{kS+1} \right) \right]} \; + \; (1-\delta(I_{MAX})) * \\ * \Bigg\{ \sum_{m=j}^{Q-1} \Bigg\{ \sum_{l=mS+1}^{(m+1)S} (\overline{w})^l (1-\delta(m)) \Bigg\{ \delta(m-j)(\alpha + jS - l) * \\ * \; e^{\left[(\alpha+jS-l)\tilde{R}_{jS+1} - S\tilde{R}_1 - (1-\delta(m-1))\left(\sum_{p=1}^{j-1} \tilde{R}_{pS+1} \right) \right]} \; - \; (1-\delta(m-j))(1-\delta(m-1)) * \\ * \, e^{\left[(\alpha+mS-l)\tilde{R}_{mS+1} - S\tilde{R}_1 - (1-\delta(m-1))S\left(\sum_{p=1}^{m-1} \tilde{R}_{pS+1} \right) \right]} \Bigg\} \Bigg\} \Bigg\} \Bigg\}, \\ for \;\; j = 1, 2, \ldots, (Q-1). \tag{A.28}$$

$$\nabla_{\tilde{R}_{I_{MAX}+1}} L = \left(\frac{\partial \mathcal{E}_{TOT}}{\partial \tilde{R}_{I_{MAX}+1}} \right) + \frac{\theta \, \lambda_1}{\Delta_{TM}} \left(\frac{\partial T_{TM}}{\partial \tilde{R}_{I_{MAX}+1}} \right) + \frac{\lambda_2}{\Delta_{DT}} \left(\frac{\partial T_{DT}}{\partial \tilde{R}_{I_{MAX}+1}} \right); \tag{A.29}$$

where:

$$\frac{\partial T_{TM}}{\partial \tilde{R}_{I_{MAX}+1}} = -M_0 (\overline{w})^{1+I_{MAX}} \; e^{[-\tilde{R}_0]} \, e^{\left[-\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1} \tilde{R}_{kS+1} \right) \right]}; \tag{A.30}$$

$$\frac{\partial T_{DT}}{\partial \tilde{R}_{I_{MAX}+1}} = -M_0\, e^{[-\tilde{R}_0]}\, (\overline{w})^{1+I_{MAX}}\, (1-\delta(1+I_{MAX})) * \\ * e^{\left[-\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]}; \tag{A.31}$$

$$\frac{\partial \mathcal{E}_{TOT}}{\partial \tilde{R}_{I_{MAX}+1}} = \theta K_0\, (\overline{w})^{1+I_{MAX}}\, M_0\, e^{[-\tilde{R}_0]}\, (\alpha-1) * \\ * e^{\left[-(\alpha-1)\tilde{R}_{I_{MAX}+1} - S\tilde{R}_1 - (1-\delta(Q-1))S\left(\sum_{k=1}^{Q-1}\tilde{R}_{kS+1}\right)\right]}. \tag{A.32}$$

$$\nabla_{\lambda_1} L = \theta \left\{ \left(\frac{T_{TM}}{\Delta_{TM}} \right) - 1 \right\}; \tag{A.33}$$

$$\nabla_{\lambda_2} L = \left\{ \left(\frac{T_{DT}}{\Delta_{DT}} \right) - 1 \right\}; \tag{A.34}$$

$$\nabla_{\lambda_{3,0}} L = \theta \left\{ \beta \overline{w}\, e^{[-\tilde{R}_0]} - 1 \right\}; \tag{A.35}$$

$$\nabla_{\lambda_{3,1}} L = \theta \left\{ \beta \overline{w}\, e^{[-\tilde{R}_1]} - 1 \right\}; \tag{A.36}$$

$$\nabla_{\lambda_{3,(jS+1)}} L = \theta \left\{ \beta \overline{w}\, e^{[-\tilde{R}_{3(jS+1)}]} - 1 \right\}; \quad for \quad j = 1, 2, \ldots, (Q-1). \tag{A.37}$$

A.8 Profiled parameter for simulation scenarios

In the following the tables with profiled parameter for simulation scenario in:

- Table A.1 for 3G scenario,
- Table A.2 for 4G scenario,
- Table A.3 for WiFi scenario.

Table A.1. Profiled parameter for simulations in 3G scenario

3G SCENARIO		
Program	Parameter	Value
bzip2	$(\overline{w}/\widehat{R})$	0.2
	$\overline{w}$	0.36 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{XEN}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.18 $((W) \times (s/Mb)^{\alpha})$
	$\mathcal{E}_{SETUP}$	3.25 (J)
mcf	$(\overline{w}/\widehat{R})$	0.333
	$\overline{w}$	0.5994 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{XEN}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.18 $((W) \times (s/Mb)^{\alpha})$
	Q	1
	$\mathcal{E}_{SETUP}$	3.25 (J)
memcached	$(\overline{w}/\widehat{R})$	0.37
	$\overline{w}$	0.666 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{XEN}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.18 $((W) \times (s/Mb)^{\alpha})$
	Q	1
	$\mathcal{E}_{SETUP}$	3.25 (J)

Table A.2. Profiled parameter for simulations in 4G scenario

4G SCENARIO		
Program	Parameter	Value
bzip2	$(\overline{w}/\widehat{R})$	0.2
	$\overline{w}$	0.36 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{3G}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.09 $((W) \times (s/Mb)^{\alpha})$
	$\mathcal{E}_{SETUP}$	5.1 (J)
mcf	$(\overline{w}/\widehat{R})$	0.333
	$\overline{w}$	0.5994 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{3G}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.09 $((W) \times (s/Mb)^{\alpha})$
	Q	1
	$\mathcal{E}_{SETUP}$	5.1 (J)
memcached	$(\overline{w}/\widehat{R})$	0.37
	$\overline{w}$	0.666 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{3G}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.09 $((W) \times (s/Mb)^{\alpha})$
	Q	1
	$\mathcal{E}_{SETUP}$	5.1 (J)

Table A.3. Profiled parameter for simulations in WiFi scenario

WiFi SCENARIO		
Program	Parameter	Value
bzip2	$(\overline{w}/\widehat{R})$	0.2
	$\overline{w}$	0.36 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{3G}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.05 $((W) \times (s/Mb)^{\alpha})$
	$\mathcal{E}_{SETUP}$	5.9 (J)
mcf	$(\overline{w}/\widehat{R})$	0.333
	$\overline{w}$	0.5994 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{XEN}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.05 $((W) \times (s/Mb)^{\alpha})$
	Q	1
	$\mathcal{E}_{SETUP}$	5.9 (J)
memcached	$(\overline{w}/\widehat{R})$	0.37
	$\overline{w}$	0.666 (Mb/s)
	$\widehat{R} \equiv R_{MAX}^{XEN}$	1.8 (Mb/s)
	M_0	256 (Mb)
	I_{MAX}^{XEN}	23
	α	2
	K_0	0.05 $((W) \times (s/Mb)^{\alpha})$
	Q	1
	$\mathcal{E}_{SETUP}$	5.9 (J)

BIBLIOGRAPHY

[1] G. J. Popek, R. P. Goldberg, Formal requirements for virtualizable third generation architectures, Communications of the ACM 17 (7) (1974) 412–421. doi:10.1145/957195.808061.
URL http://doi.acm.org/10.1145/361011.361073

[2] S. Barbarossa, S. Sardellitti, P. Di Lorenzo, Communicating While Computing: Distributed mobile cloud computing over 5G heterogeneous networks, IEEE Signal Processing Magazine 31 (6) (2014) 45–55. arXiv:arXiv:1105.3232v1, doi:10.1109/MSP.2014.2334709.
URL http://ieeexplore.ieee.org/document/6923537/

[3] M. Chen, Y. Zhang, Y. Li, S. Mao, V. C. Leung, EMC: Emotion-aware mobile cloud computing in 5G, IEEE Network 29 (2) (2015) 32–38. doi:10.1109/MNET.2015.7064900.
URL http://ieeexplore.ieee.org/document/7064900/

[4] Amazon elastic compute cloud (amazon ec2).
URL http://aws.amazon.com/ec2/

[5] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107. arXiv:10.1.1.163.5292, doi:10.1145/1327452.1327492.
URL http://portal.acm.org/citation.cfm?doid=1327452.1327492

[6] P. T. Metaxas, P. T. Metaxas, How Google Works , World 6 (6) (2006) 9–10.
URL http://www.howgoogleworks.net

[7] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, M. F. Zhani, Data Center Network Virtualization: A

Survey, IEEE Communications Surveys & Tutorials 15 (2) (2013) 909–928. doi:10.1109/SURV.2012.090512.00043.
URL http://ieeexplore.ieee.org/document/6308765/

[8] K. Bilal, S. U. R. Malik, O. Khalid, A. Hameed, E. Alvarez, V. Wijaysekara, R. Irfan, S. Shrestha, D. Dwivedy, M. Ali, U. Shahid Khan, A. Abbas, N. Jalil, S. U. Khan, A taxonomy and survey on Green Data Center Networks, Future Generation Computer Systems 36 (2014) 189–208. doi:10.1016/j.future.2013.07.006.
URL http://www.sciencedirect.com/science/article/pii/S0167739X13001519

[9] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, S. Sengupta, VL2, ACM SIGCOMM Computer Communication Review 39 (4) (2009) 51. doi:10.1145/1594977.1592576.
URL http://portal.acm.org/citation.cfm?doid=1594977.1592576

[10] C. E. Leiserson, Fat-trees: Universal networks for hardware-efficient supercomputing, IEEE Transactions on Computers C-34 (10) (1985) 892–901. doi:10.1109/TC.1985.6312192.
URL http://ieeexplore.ieee.org/document/6312192/

[11] L. Popa, S. Ratnasamy, G. Iannaccone, A. Krishnamurthy, I. Stoica, A cost comparison of datacenter network architectures, in: Proceedings of the 6th International COnference, Vol. 6 of Co-NEXT '10, ACM, New York, NY, USA, 2010, pp. 1–12. doi:10.1145/1921168.1921189.
URL http://doi.acm.org/10.1145/1921168.1921189

[12] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, BCube, ACM SIGCOMM Computer Communication Review 39 (4) (2009) 63. doi:10.1145/1594977.1592577.
URL http://portal.acm.org/citation.cfm?doid=1594977.1592577

[13] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, S. Lu, Dcell, in: Proceedings of the ACM SIGCOMM 2008 conference on Data communication - SIGCOMM '08, SIGCOMM '08, ACM Press, New York, New York, USA, 2008, p. 75. doi:10.1145/1402958.1402968.
URL http://portal.acm.org/citation.cfm?doid=1402958.1402968

[14] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: ACM SIGCOMM Computer Communication Review,

Vol. 38, ACM, 2008, p. 63. doi:10.1145/1402946.1402967.
URL http://portal.acm.org/citation.cfm?doid=1402946.1402967

[15] M. Portnoy, Virtualization essentials, John Wiley & Sons, 2012.

[16] F. Xu, F. Liu, H. Jin, A. V. Vasilakos, Managing Performance Overhead of Virtual Machines in Cloud Computing: A Survey, State of the Art, and Future Directions, Proceedings of the IEEE 102 (1) (2014) 11–31. doi:10.1109/JPROC.2013.2287711.

[17] W. Voorsluys, J. Broberg, S. Venugopal, R. Buyya, Cost of virtual machine live migration in clouds: A performance evaluation, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 5931 LNCS, Cloud Computing, Beijing, China, 2009, pp. 254–265. arXiv:1109.4974, doi:10.1007/978-3-642-10665-1_23.

[18] Y. Wu, M. Zhao, Performance Modeling of Virtual Machine Live Migration, in: IEEE International Conference on Cloud Computing (CLOUD), IEEE, 2011, pp. 492–499. doi:10.1109/CLOUD.2011.109.

[19] S. Froberg, Distributed and cloud computing from parallel processing to the internet of things by Kai Hwang, Geoffry C. Fox, and Jack J. Dongarra, Vol. 38, Morgan Kaufmann, 2013. doi:10.1145/2439976.2439991.
URL http://dl.acm.org/citation.cfm?doid=2439976.2439991

[20] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, H. Liu, Energy proportional datacenter networks, in: ACM SIGARCH Computer Architecture News, Vol. 38, ACM SIGARCH Computer Architecture News, 2010, p. 338. doi:10.1145/1816038.1816004.
URL http://dl.acm.org/citation.cfm?doid=1816038.1816004

[21] A. Verma, P. Ahuja, A. Neogi, p{M}apper: Power and Migration Cost Aware Application Placement in Virtualized Systems, in: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, New York, NY, USA, 2008, pp. 243–264.

[22] T. Wood, P. J. Shenoy, A. Venkataramani, M. S. Yousif, Black-box and gray-box strategies for virtual machine migration, in: NSDI, 4th USENIX Symposium on Networked Systems Design and Implementation, Vol. 7, Cambridge, USA, 2007, pp. 229–242.

[23] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, Future

Generation Computer Systems 28 (5) (2012) 755–768. doi:10.1016/j.future.2011.04.017.
URL http://dx.doi.org/10.1016/j.future.2011.04.017

[24] M. Mishra, A. Sahoo, On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach, in: Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011, 2011, pp. 275–282. doi:10.1109/CLOUD.2011.38.

[25] R. Boutaba, Q. Zhang, M. F. Zhani, Virtual Machine Migration in Cloud Computing Environments, Vol. i, IGI Global, 2014. doi:10.4018/978-1-4666-4522-6.ch017.
URL http://dx.doi.org/10.4018/978-1-4666-4522-6.ch017

[26] C. Takemura, L. S. Crawford, The book of Xen: a practical guide for the system administrator, 1st Edition, No Starch Press, San Francisco, CA, USA, 2010.

[27] M. Mishra, A. Das, P. Kulkarni, A. Sahoo, Dynamic resource management using virtual machine migrations, IEEE Communications Magazine 50 (9) (2012) 34–40. doi:10.1109/MCOM.2012.6295709.
URL http://dx.doi.org/10.1109/MCOM.2012.6295709

[28] H. Liu, H. Jin, C.-Z. Xu, X. Liao, Performance and energy modeling for live migration of virtual machines, Hpdc'11 16 (2) (2011) 249–264. doi:10.1007/s10586-011-0194-3.
URL http://link.springer.com/10.1007/s10586-011-0194-3

[29] G. Soni, M. Kalra, Comparative Study of Live Virtual Machine Migration Techniques in Cloud, International Journal of Computer Applications 84 (14) (2013) 19–25. doi:10.5120/14643-2919.

[30] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, A. Warfield, Live migration of virtual machines, in: NSDI'05 Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, Vol. 2, USENIX Association, 2005, pp. 273–286. arXiv:1601.03854, doi:10.1145/1251203.1251223.
URL http://dl.acm.org/citation.cfm?id=1251203.1251223

[31] P. Svärd, B. Hudzia, J. Tordsson, E. Elmroth, Evaluation of delta compression techniques for efficient live migration of large virtual machines, ACM SIGPLAN Notices 46 (7) (2011) 111. doi:10.1145/2007477.1952698.

[32] H. Liu, H. Jin, X. Liao, L. Hu, C. Yu, Live migration of virtual machine based on full system trace and replay, in: Proceedings of the 18th {ACM} international symposium on High performance distributed computing, New York, USA, 2009, pp. 101–110. doi:10.1145/1551609.1551630.

[33] X. Zhang, Z. Huo, J. Ma, D. Meng, Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration, in: 2010 IEEE International Conference on Cluster Computing, IEEE, Crete, Greece, 2010, pp. 88–96. doi:10.1109/CLUSTER.2010.17.
URL http://ieeexplore.ieee.org/document/5600319/

[34] F. F. Moghaddam, M. Cheriet, Decreasing live virtual machine migration down-time using a memory page selection based on memory change PDF, in: 2010 International Conference on Networking, Sensing and Control, ICNSC 2010, Chicago, USA, 2010, pp. 355–359. doi:10.1109/ICNSC.2010.5461517.

[35] M. R. Hines, K. Gopalan, Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning, in: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments - VEE '09, ACM Press, New York, New York, USA, 2009, p. 51. doi:10.1145/1508293.1508301.
URL http://portal.acm.org/citation.cfm?doid=1508293.1508301

[36] H. Liu, H. Jin, X. Liao, L. Hu, C. Yu, Live migration of virtual machine based on full system trace and replay, in: Proceedings of the 18th {ACM} international symposium on High performance distributed computing, ACM HPDC'09, New York, USA, 2009, pp. 101–110. doi:10.1145/1551609.1551630.

[37] T. Hirofuchi, H. Nakada, S. Itoh, S. Sekiguchi, Reactive consolidation of virtual machines enabled by postcopy live migration, in: Proceedings of the 5th international workshop on Virtualization technologies in distributed computing - VTDC '11, ACM Press, New York, New York, USA, 2011, p. 11. doi:10.1145/1996121.1996125.
URL http://portal.acm.org/citation.cfm?doid=1996121.1996125

[38] P. Lu, A. Barbalace, B. Ravindran, HSG-LM: hybrid-copy speculative guest OS live migration without hypervisor, in: Proceedings of the 6th International Systems and Storage Conference on - SYSTOR '13, New York, USA, 2013, p. 1. doi:10.1145/2485732.2485736.
URL http://dl.acm.org/citation.cfm?doid=2485732.2485736

[39] L. Hu, J. Zhao, G. Xu, Y. Ding, J. Chu, HMDC: Live virtual machine migration based on hybrid memory copy and delta compression, Applied Mathematics and Information Sciences 7 (2 L) (2013) 639–646. doi:10.12785/amis/072L38.

[40] Y. L. Min, F. Rawson, T. Bletsch, V. W. Freeh, PADD: Power-aware domain distribution, in: Proceedings - International Conference on Distributed Computing Systems, Montreal, Canada, 2009, pp. 239–247. doi:10.1109/ICDCS.2009.47.

[41] E. Baccarelli, M. Biagi, Power-Allocation Policy and Optimized Design of Multiple-Antenna Systems With Imperfect Channel Estimation, IEEE Transactions on Vehicular Technology 53 (1) (2004) 136–145. doi:10.1109/TVT.2003.822025.
URL http://ieeexplore.ieee.org/document/1262136/

[42] E. Baccarelli, M. Biagi, R. Bruno, M. Conti, E. Gregori, Broadband Wireless Access Networks: A Roadmap on Emerging Trends and Standards, John Wiley & Sons, Ltd, Chichester, UK, 2005. doi:10.1002/0470022515.ch14.
URL http://doi.wiley.com/10.1002/0470022515.ch14

[43] E. Baccarelli, M. Biagi, A Novel Self-Pilot-Based Transmit-Receive Architecture for Multipath-Impaired UWB Systems, IEEE Transactions on Communications 52 (6) (2004) 891–895. doi:10.1109/TCOMM.2004.829525.
URL http://ieeexplore.ieee.org/document/1306613/

[44] M. Harchol-Balter, Performance Modeling and Design of Computer Systems Queuing Theory in Action, Cambridge University Press, 2013.

[45] L. Wang, F. Zhang, J. Arjona Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, Z. Liu, GreenDCN: a general framework for achieving energy efficiency in data center networks, Selected Areas in Communications, IEEE Journal on 32 (1) (2014) 4–15.

[46] E. Baccarelli, M. Biagi, C. Pelizzoni, N. Cordeschi, F. Garzia, When does interference not reduce capacity in multi-antenna networks?, in: IEEE 6th Workshop on Signal Processing Advances in Wireless Communications, 2005., IEEE, IEEE, 2005, pp. 298–302. doi:10.1109/SPAWC.2005.1505920.
URL http://ieeexplore.ieee.org/document/1505920/

[47] N. Cordeschi, M. Shojafar, D. Amendola, E. Baccarelli, Energy-efficient adaptive networked datacenters for the QoS support of real-time applications, The Journal of Supercomputing 71 (2) (2015) 448–478. doi:10.1007/s11227-014-1305-8.
URL http://link.springer.com/10.1007/s11227-014-1305-8

[48] E. Baccarelli, M. Biagi, C. Pelizzoni, N. Cordeschi, Maximum-Rate Node Selection for Power-Limited Multiantenna Relay Backbones, IEEE Transactions on Mobile Computing 8 (6) (2009) 807–820. doi:10.1109/TMC.2008.171.
URL http://ieeexplore.ieee.org/document/4731258/

[49] D. Breitgand, G. Kutiel, D. Raz, Cost-aware live migration of services in the cloud, in: Proceedings of the 3rd Annual Haifa Experimental Systems Conference on - SYSTOR '10, New York, USA, 2010, p. 1. doi:10.1145/1815695.1815709.
URL http://dl.acm.org/citation.cfm?id=1815695.1815709

[50] K. Z. Ibrahim, S. Hofmeyr, C. Iancu, E. Roman, Optimized pre-copy live migration for memory intensive applications, in: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '11, ACM Press, New York, New York, USA, 2011, p. 1. doi:10.1145/2063384.2063437.
URL http://dl.acm.org/citation.cfm?doid=2063384.2063437

[51] D. Chisnall, The Definitive Guide to the Xen Hypervisor, first ed., Prentice Hall, 2007.

[52] B. Fitzpatrick, Distributed caching with memcached, in: Linux Journal, Vol. 2004, Linux Journal, 2004, p. 5.

[53] S. Fu, Failure-aware resource management for high-availability computing clusters with distributed virtual machines, Journal of Parallel and Distributed Computing 70 (4) (2010) 384–393. doi:10.1016/j.jpdc.2010.01.002.
URL http://linkinghub.elsevier.com/retrieve/pii/S0743731510000031

[54] S. Fu, C.-Z. Xu, Exploring event correlation for failure prediction in coalitions of clusters, in: Proceedings of the 2007 ACM/IEEE conference on Supercomputing - SC '07, ACM Press, New York, New York, USA, 2007, p. 1. doi:10.1145/1362622.1362678.
URL http://doi.acm.org/10.1145/1362622.1362678http://portal.acm.org/citation.cfm?doid=1362622.1362678

[55] T. C. Bressoud, F. B. Schneider, Hypervisor-based fault tolerance, ACM Transactions on Computer Systems 14 (1) (1996) 80–107. doi:10.1145/225535.225538.
URL http://portal.acm.org/citation.cfm?doid=225535.225538

[56] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, Nonlinear Programming, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006. doi:10.1002/0471787779.
URL http://doi.wiley.com/10.1002/0471787779

[57] D. S. Lun, N. Ratnakar, M. M??dard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, F. Zhao, Minimum-cost multicast over coded packet networks, IEEE Transactions on Information Theory 52 (6) (2006) 2608–2623. arXiv:0503064, doi:10.1109/TIT.2006.874523.

[58] R. Srikant, The Mathematics of Internet Congestion Control, Vol. 50, Springer Science & Business Media, 2004. doi:10.1109/TAC.2004.841398.

[59] J. Zhang, D. Zheng, M. Chiang, The Impact of Stochastic Noisy Feedback on Distributed Network Utility Maximization, IEEE Transactions on Information Theory 54 (2) (2008) 645–665. doi:10.1109/TIT.2007.913572.

[60] H. Kushner, J. Yang, Analysis of adaptive step-size SA algorithms for parameter tracking, IEEE Transactions on Automatic Control 40 (8) (1995) 1403–1410. doi:10.1109/9.402231.
URL http://ieeexplore.ieee.org/document/402231/

[61] A. Motivation, Exploiting Hidden Convexity For Flexible And Robust Resource Allocation In Cellular Networks, in: INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, IEEE, 2007, pp. 964–972.

[62] D. Henrion, J.-B. Lasserre, Solving nonconvex optimization problems, IEEE Control Systems Magazine 24 (3) (2004) 72–83. doi:10.1109/MCS.2004.1299534.
URL http://dx.doi.org/10.1109/MCS.2004.1299534

[63] Memtester - memory diagnostic tool.
URL http://pyropus.ca/software/memtester.

[64] Spec cpu2000. standard performance evaluation corporation.
URL http://www.spec.org/cpu2000/CINT2000

[65] J. L. Henning, SPEC CPU2006 benchmark descriptions, ACM SIGARCH Computer Architecture News 34 (4) (2006) 1–17. arXiv:arXiv:1011.1669v3, doi:10.1145/1186736.1186737.
URL http://portal.acm.org/citation.cfm?doid=1186736.1186737

[66] Memaslap.
URL http://docs.libmemcached.org/bin/memaslap

[67] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, CloneCloud: Elastic execution between mobile device and cloud, in: Proceedings of the sixth conference on Computer systems - EuroSys '11, ACM, ACM Press, New York, New York, USA, 2011, p. 301. doi:10.1145/1966445.1966473.
URL http://portal.acm.org/citation.cfm?doid=1966445.1966473

[68] M. Schüring, Mobile cloud computing – open issues and solutions, in: 15th Twente Student Conference on IT, 2011.

[69] P. Papakos, L. Capra, D. S. Rosenblum, Volare: context-aware adaptive cloud service discovery for mobile systems, in: Proceedings of the 9th International Workshop on Adaptive and Reflective Middleware - ARM '10, ACM, 2010, pp. 32–38. doi:10.1145/1891701.1891706.
URL http://portal.acm.org/citation.cfm?doid=1891701.1891706

[70] R. Kemp, N. Palmer, T. Kielmann, H. Bal, Cuckoo: A Computation Offloading Framework for Smartphones, in: Mobile Computing, Applications, . . . , Springer, 2012, pp. 59–79. doi:10.1007/978-3-642-29336-8_4.
URL http://dx.doi.org/10.1007/978-3-642-29336-8_4

[71] M. Satyanarayanan, P. Bahl, R. Cáceres, N. Davies, The case for VM-based cloudlets in mobile computing, IEEE Pervasive Computing 8 (4) (2009) 14–23. doi:10.1109/MPRV.2009.82.

[72] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: Making smartphones last longer with code offload, in: Proceedings of the 8th international conference on Mobile systems, applications, and services - MobiSys '10, Vol. 17, ACM, ACM Press, New York, New York, USA, 2010, p. 49. doi:10.1145/1814433.1814441.
URL http://portal.acm.org/citation.cfm?doid=1814433.1814441

[73] K. Kumar, J. Liu, Y.-H. Lu, B. Bhargava, A Survey of Computation Offloading for Mobile Systems, Mobile Networks and Applications 18 (1) (2013) 129–140. doi:10.1007/s11036-012-0368-0.
URL http://dx.doi.org/10.1007/s11036-012-0368-0http://link.springer.com/10.1007/s11036-012-0368-0

[74] G. P. Perrucci, F. H. P. Fitzek, J. Widmer, Q. Wei, W. Kellerer, Survey on Energy Consumption Entities on Mobile Phone Platform, in: Mobile Applications and Services, IEEE Vehicular Technology Conference (VTC) - Spring. IEEE, Budapest, Hungary., IEEE, 2011, pp. 1–6.

[75] Q. Peng, A. Walid, J. Hwang, S. H. Low, Multipath TCP: Analysis, Design, and Implementation, IEEE/ACM Transactions on Networking 24 (1) (2016) 596–609. arXiv:1308.3119, doi:10.1109/TNET.2014.2379698.
URL http://ieeexplore.ieee.org/document/7000573/

[76] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, O. Spatscheck, A close examination of performance and power characteristics of 4G LTE networks, in: Proceedings of the 10th international conference on Mobile systems, applications, and services - MobiSys '12, ACM, ACM Press, New York, New York, USA, 2012, p. 225. doi:10.1145/2307636.2307658.
URL http://dl.acm.org/citation.cfm?doid=2307636.2307658

[77] N. Balasubramanian, A. Balasubramanian, A. Venkataramani, Energy consumption in mobile phones, in: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference - IMC '09, ACM, ACM Press, New York, New York, USA, 2009, p. 280. doi:10.1145/1644893.1644927.
URL http://dx.doi.org/10.1145/1644893.1644927

[78] E. Baccarelli, D. Amendola, N. Cordeschi, Minimum-energy bandwidth management for QoS live migration of virtual machines, Computer Networks 93 (2015) 1–22. doi:10.1016/j.comnet.2015.10.006.
URL http://dx.doi.org/10.1016/j.comnet.2015.10.006

[79] D. Wischik, C. Raiciu, Design, implementation and evaluation of congestion control for multipath TCP, in: ... and Implementation, Vol. 11, 2011, pp. pp. 1260–1271.
URL http://static.usenix.org/event/nsdi11/tech/full{_}papers/Wischik.pdf

[80] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, M. Handley, How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP., in: NSDI'12 Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, no. 1, USENIX, San Jose, CA, 2012, pp. 29–42.
URL http://elf.cs.pub.ro/soa/res/lectures/mptcp-nsdi12.pdf

[81] C. Pluntke, L. Eggert, N. Kiukkonen, Saving mobile device energy with multipath TCP, in: ACM Proceedings of the sixth international workshop on MobiArch - MobiArch '11, ACM, 2011, pp. 1–6. doi:10.1145/1999916.1999918.

[82] N. Cordeschi, V. Polli, E. Baccarelli, Interference Management for Multiple Multicasts with Joint Distributed Source/Channel/Network Coding, IEEE

Transactions on Communications 61 (12) (2013) 5176–5183. doi:10.1109/TCOMM.2013.111113.120904.
URL http://ieeexplore.ieee.org/document/6668863/

[83] Y. Xu, B. Leong, D. Seah, A. Razeen, mPath: High-Bandwidth Data Transfers with Massively Multipath Source Routing, IEEE Transactions on Parallel and Distributed Systems 24 (10) (2013) 2046–2059. doi:10.1109/TPDS.2012.298.
URL http://ieeexplore.ieee.org/document/6336741/