Politecnico di Milano

Corso di Laurea Magistrale in Ingegneria delle Telecomunicazioni

Scuola di Ingegneria Industriale e dell'Informazione

Dynamic Routing and Bandwidth Assignment for Live Virtual Machines Migrations

Relatore: Prof. Massimo **TORNATORE**

Correlatore: Dr. Francesco **MUSUMECI**

Autore: Luca **PACE**

Matr. 804011

Anno Accademico 2014/2015

Contents

Abstract

In recent years, virtualization is playing an increasingly important role in a cloud environment, as it allows service providers to create an abstraction of the physical servers, contained within the data centers. In this way services can be directly hosted on these abstract entities, which are then called virtual machines (VM). The direct result of service virtualization is the ability to move the service hosted on a virtual machine, from a data center to another; hence, the virtual machines migration. However, the migration of a virtual machine within a Wide Area Network (WAN), should be carried out efficiently since it requires a considerable use of network resources (available bandwidth, especially).

In this thesis we proposed a model for the live migration of virtual machines, focusing our attention on the Routing and Bandwidth Assignment problem. Then, we present four algorithms for the optimal choice of the bandwidth to be assigned to every migration request of a virtual machine that needs to be moved from one data center to another.Each one of the illustrated algorithms has been implemented within an optical network discrete event based simulator, in order to evaluate its performance: in particular we studied the behavior of both the blocking probability and the network resource consumption.

Sommario

Negli ultimi anni la virtualizzazione sta ricoprendo un ruolo sempre piú importante all'interno dell'ambiente Cloud, in quanto permette ai service provider di creare un'astrazione dei server fisici, contenuti all'interno dei data center. In questo modo i servizi possono essere ospitati direttamente su queste entitá astratte, che prendono il nome di macchine virtuali (VM). La diretta conseguenza della virtualizzazione dei servizi é la possibilitá di spostare il servizio ospitato su una macchina virtuale, da una data center all'altro: nasce cosíla migrazione di macchine virtuali. Tuttavia, la migrazione di una macchina virtuale all'interno di una rete geograficamente estesa (WAN), deve essere effettuata in maniera efficiente poiché essa richiede un utilizzo considerevole delle risorse di rete (di banda specialmente). Pertanto la migrazione deve essere effettuata nel rispetto dei livelli di qualitá del servizio prestabiliti garantendo, tuttavia, un limitato consumo delle risorse di rete.

In questo lavoro di tesi abbiamo proposto un modello per la migrazione in tempo reale di macchine virtuali, focalizzando la nostra attenzione sul problema di Routing and Bandwidth Assignment. Vengono, quindi, presentati quattro algoritmi per la scelta ottimale della banda da assegnare ad ogni richiesta di migrazione di una macchina virtuale da un data center all'altro. Ciascuno degli algoritmi illustrati ato implementato all'interno di un simulatore di rete ottica ad eventi discreti allo scopo di valutarne le prestazioni: in particolare viene studiato il comportamento della probabilitá di blocco e dell'occupazione delle risorse di rete.

Chapter 1

Introduction

1.1 Cloud Computing

Nowadays, cloud computing is playing an increasing important role as cloud infrastructures are used to host both enterprise and public internet services. It represents a new paradigm of computing as it provides a shared pool of configurable computing resources such that client requests can be served on demand. It is able to provide both users and enterprises with high capabilities to process and store data in a separated location (the so called data centers). This approach helps maximize the use of computing power while reducing the overall cost of resources by using less power to maintain the system. Besides, it allow users to access a server to retrieve their data without purchasing licenses for several applications. It also benefits enterprises as it allows to reduce infrastructures costs and it allows them to get applications up and running faster. There are many other factors that have led the growth of cloud computing in recent years, such as: the availability of high-capacity networks, storage devices as well as the ever increasingly adoption of hardware virtualization. Cloud computing providers offer their "services" according to three different models:

- Iaas (Infrastructure as a Service): it is about offer computers and other resources (they can be both physical and virtual machines); in this way, users are not involved into the detail of infrastructures like physical computing

resources management, security, data back-up. They can easily deploy applications by installing operating-system images and their application software on the cloud infrastructure. On the other hand, a cloud hypervisor can support a large numbers of virtual machines and it is able to scale services up and down according to customers requirements. Moreover, IaaS cloud providers supply this great amount of resources from their large pools of equipment installed in data centers. Usually, in this model, providers bill services in according to the computing resources usage: cost reflects the amount of resources allocated.

- PaaS (Platform as a Service): providers offer a development environment to users who want to develop an application; morover the provider can develop toolkit and standards for development and channels for distribution and payment. Typically they deliver a computing platform, including operating system, programming-language execution environment, database, and web server; while clients can develop and run their software solutions on a cloud platform without the cost of buying and managing the underlying hardware and software layers.

- SaaS (software as a Service): users can access to application software and databases while cloud providers manage the infrastructure and platforms that run the applications. Providers install and operate application software in the cloud and users access the software from cloud clients, so they do not have to deal with infrastructure and platform where the application runs. The pricing model is typically a monthly or yearly flat fee per user, so prices become scalable and adjustable if users are added or removed at any point.

A cloud environment is called 'public cloud' when the services are provisioned over a network that is open for public use, while a 'private cloud' is a cloud infrastructure operated only for a single organization. However both public cloud service providers own and operate the infrastructure at their data center and access is generally via the Internet. Thanks to the rapid development of the cloud environment, an ever increasing number of applications are surfacing, increasing the cloud in-

frastructures demand. Consequently, increasing demand brings to an increasing energy consumption, so Cloud infrastructure providers have to virtualize physical servers to achieve a better management of resources in a multi data center environment. Services are then hosted over virtual machines (VMs) with the benefit that a virtual machine can be migrate from a data center to another in order to achieve cloud bursting, load balancing, fault recovery and so on. However, the migration process consumes a large amount of energy both at source and destination location, as well as in the network as it generate a large amount of traffic. From this background,where Cloud resources are placed over geographically-distributed data centers, arises the need of an underlying high-performance network, typically based on high-speed technology, which can be able to meet the requirements of overlaid applications, such as dynamic bandwidth allocation scheme, efficient grooming and using a low-cost networking technology platform. Typically cloud computing services are offered over a metropolitan network which is based on an optical networking paradigm. In facts, optical networks are considered as a valuable solution to meet such requirements and implement cloud computing applications. Indeed, optical transmission technology can achieve higher capacity connections with a cost-effective way. This is due to its ability to transfer huge volumes of data with very low latency. Consequently, optical networking technology is considered the best way to connect data centers providing computing and storage services in the cloud computing environment.

1.2 Virtualization In Cloud Environment

Operating system virtualization has attracted considerable interest in recent years, particularly from the data center and cluster computing communities. The adoption of virtualization in data centers creates the need for a new class of networking designed to support elastic resource allocation, increasingly mobile workloads, and maximum availability under production loads. Virtualization technologies encapsulate existing applications, and abstract them from the physical hardware.

Unlike physical machines, virtual machines are represented by a portable soft-

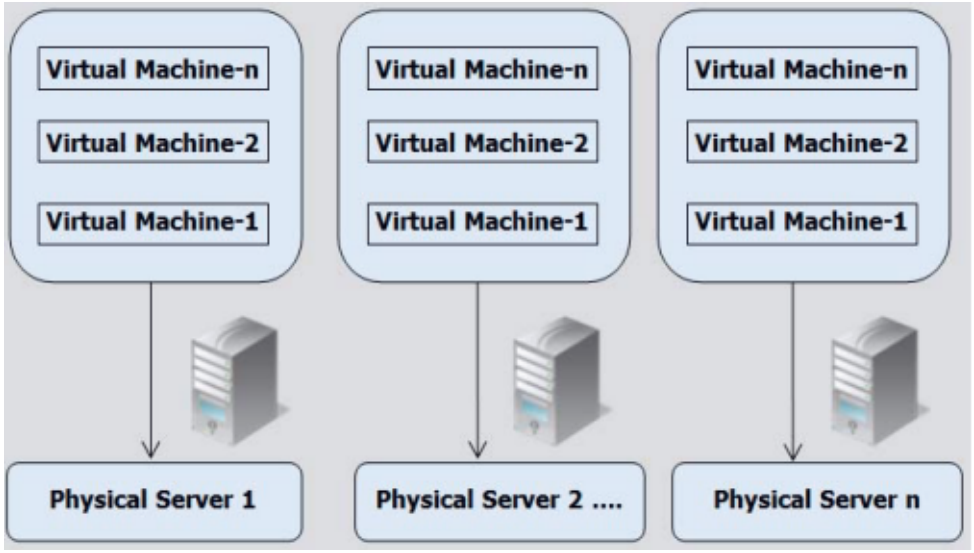

Figure 1.1: Server with virtual machines

ware image, which can be instantiated on physical hardware in few seconds. With virtualization comes elasticity where compute capacity can be scaled up or down, on demand, by adjusting the number of virtual machines running on a given physical server. Cloud infrastructure providers virtualize physical servers for easy and secure resource allocation in a multitenant data center environment: in this way services can be hosted over these virtual machines (VMs); this technology can increase hardware utilization by allowing multiple isolated Operating System (OS) to run in the same host. Besides, there is a further benefit allowed by virtualization: that of migrate a virtual machine. Host virtulization offers the benefit for data centers to live migrate an entire virtual Machine from a single physical host to another, enabling virtual machines to become portable across larger geographies.

The combination of virtualization and migration, VM migration, is increasingly utilized in today's multi-tenant cloud data centers. VM migration brings lots of benefits: it facilitates load balance and consolidate servers by migrating VMs out of overloaded or underloaded physical hosts. New multi-core processing architectures running 10-20 or more virtual machines per server results in a significant increase in the number of elements to be managed. Since many virtual machines can be instantiated on a physical server, bandwidth utilization increases proportionally.

This implies that traditional RBA (Routing and Bandwidth Assignment) schemes and network topologies must be re-architected for virtualization and private clouds. Migrating an entire OS and all of its applications as one unit allows us to avoid many of the difficulties faced by process-level migration approaches. With virtual machine migration, on the other hand, the original host may be decommissioned once migration has completed, allowing providers to save energy, in this way. Also, once a server requires hardware maintenance or software update, the services running on it can be migrated to others. In the multi-tenant clouds, resource can be redistributed through live migration to allow more tenants. There are two main methods of migrating a VM: offline migration and online/live migration. If the offline migration is conducted, the services running on the migrated VM will terminate during the whole migration process, while live migration mechanisms can keep the services on the migrated VM alive during most of the migration process. The main advantages of live migration are that it not only endows the benefit of VM migration, but also imposes little interruption on the running services. An other important aspect is that live migration of virtual machines allows a separation of concerns between the users and operator of a data center or cluster. Users have 'carte blanche' regarding the software and services they run within their virtual machine, and need not provide the operator with any OS-level access at all (e.g. a root login to quiesce processes or I/O prior to migration). Similarly the operator need not be concerned with the details of what is occurring within the virtual machine; instead, they can simply migrate the entire operating system and its attendant processes as a single unit. Overall, live OS migration is a extremely powerful tool for cluster administrators, allowing separation of hardware and software considerations, and consolidating clustered hardware into a single coherent management domain.

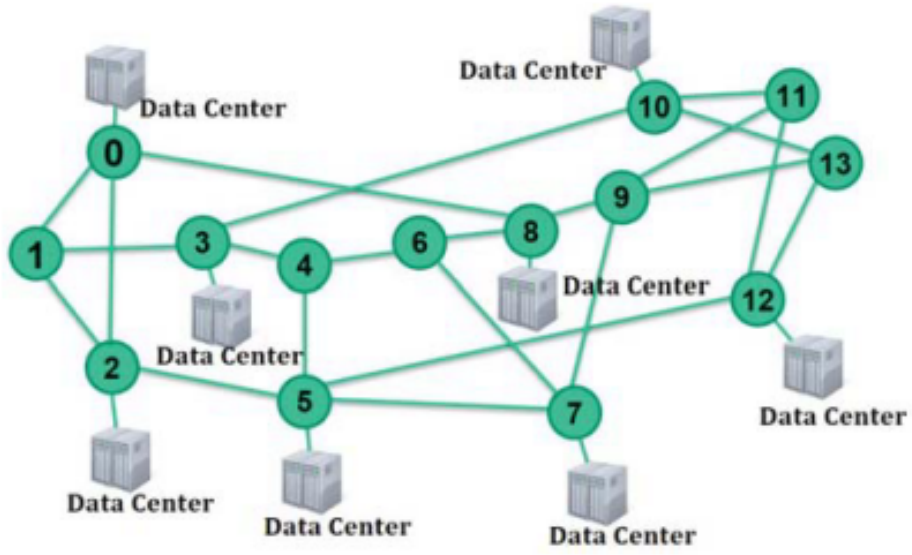

Figure 1.2: Network topology with data centers

If a physical machine needs to be removed from service, an administrator may migrate OS instances including the applications that they are running to alternative machine(s), freeing the original machine for maintenance. One of the concerns with live VM migration is the negative impact on active services in the network. Live VM Migration, depending on the memory sizes and running applications may consume nearly the entire bandwidth which impacts the performance of competing flows in the network. Therefore, it is imperative to explore the impact of wide-area migration of virtual machines and services between data center networks. Data center network topology and traffic dynamics can affect the performance of VM migration, so VM migration should be conducted such that network resources can be allocated intelligently to minimize degradation of network performance.

1.3 Work Organization

The rest of the work is organized as follows. In chapter 2 we present the migration model adopted in our work: showing as a live pre-copy approach for virtual machine migration works and presenting its main involved parameters. We focus on how such parameters (dirtying rate, migration bandwidth, total migration dura-

tion, downtime) can influence the whole migration process; then we introduce the main trade-off characterizing the virtual machines migration through the issue of the function points curve. After describing the state of the art about VM migration, we dedicate chapter 3 to introduce basic Routing and Bandwidth Assignment scheme. Then we illustrate our contribution to this topic: the aim of our thesis is to propose algorithms for the Routing and Bandwidth Assignment problem in the new scenario of the migration of virtual machines within a cloud environment. Hence, in each section of the chapter we present one of the proposed algorithms. In chapter 4 we report the simulative and illustrative numerical results aiming to show how the choice of the adopted algorithm can influence the blocking probability. Moreover, we compare the results of our proposed algorithms and we evaluated their behavior according to the variations of simulation parameters, such as dirtying rate and arrival rate. In chapter 5 we focus on a more general problem as can be the migration of multiple virtual machines, instead of migrating a single virtual machine per time. In particular we try to analyze two cases: the serial migration and the parallel migration of a set of virtual machines. Finally in chapter 6, we conclude this work with some significant observations on the performance of the different proposed algorithms, and also possible future works are discussed.

Chapter 2

Live Virtual Machines Migration

2.1 Virtual Machines

Virtualization technologies are widely adopted by companies to manage flexible computation environment and to run isolated virtual environments for each customer. As virtual machines can provide desirable features like flexibility, better protection and hardware independence, they are applied in various research areas and have great potential, also because providers can freely migrate them from a server to another without loss of quality of service. Standard computer systems are hierarchically constructed from three components: hardware, operating system, and application software. The standard architecture has many advantages, since the interfaces are nicely defined, and the hardware and software components can be decoupled. Nevertheless in this architecture, hardware, operating system, and application software are fixed, and they are not interchangeable. So software is restricted by the operating system: it cannot move freely among all computers connected by a network, because usually these computers all vary in hardware and operating systems. So this architecture loses flexibility, and this becomes a critical issue when the internet becomes larger and application software becomes more complicated. In a scenario as the one just descripted, the idea of virtualization takes place, with the new concern of virtual machine.

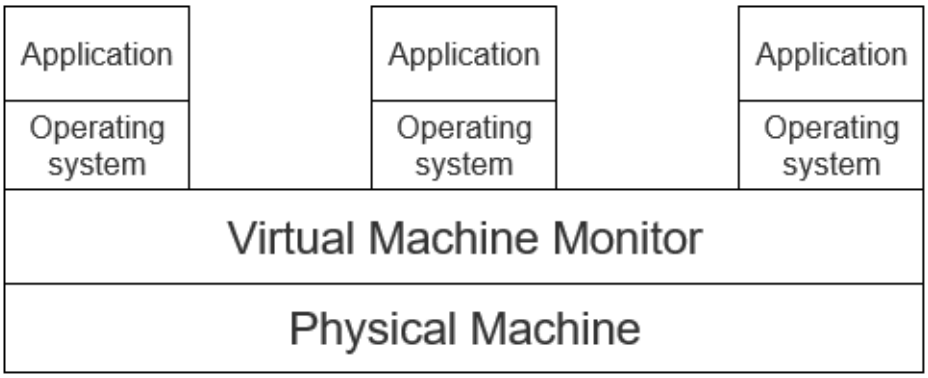

Figure 2.1: Simple Virtual Machine Scheme

A virtual machine provides an abstraction of the underlying physical system to the guest operating system running on it. Virtual machine migration is, then, the task of moving a virtual machine from one physical hardware entity to another one. As a virtual machine provides a fully protected and isolated replica of the underlying physical system, we need a new layer above the original bare systems to abstract the physical resources and provide interface to operating systems running on it. This layer is called the Virtual Machine Monitor (VMM). The VMM is the essential part of the virtual machine implementation, because it performs the translation between the hardware and virtualized underlying platform: providing virtual processors, memory, and virtualized I/O devices. Since all the virtual machines share the same hardware component, the Virtual Machine Monitor should also provide appropriate protection so that each virtual machine is an isolated replica. The basic scheme of a virtual machine should be like in Figure 1.1, where the virtual machine monitor sits between the system hardware and operating systems.

2.2 Virtual Machines Migration

The procedure of migrating the operating system and applications from a physical machine to another physical machine is an important issue in a virtualized environment. VMs migration consists in multiple main resource transferring: pro-

cessor, memory, network and storage. During the migration process, the VM is paused at the source host and is resumed on the destination host only when all resources are already been migrated and reconfigured into the new host. The period of time in which VM stays offline is called downtime, and it lasts from the time instant in which the VM is paused until it is resumed at the destination. The downtime period varies according to the resources available on the VM, to the workload submitted to the VM, and to the migration technique: offline or online (live) migration.

2.2.1 Offline Virtual Machines Migration

Offline migration technique allows to transfer the VM to the destination physical host while the VM is off. The offline migration introduces a great delay for the application running on the virtual machine, but it is the easiest to accomplish because it does not require the VM preservation. As the VM is off, there is no need to preserve network connections, and it is neither necessary to transfer the processor state nor the RAM content. The offline migration procedure just comprises shutting down the virtual machines and restarting it into another location. The storage migration, or disk migration, is performed by standard data transfer tools and is the only network traffic generated. It takes a long time and it needs great amount of network bandwidth to transfer a whole disk.

2.2.2 Live Virtual Machines Migration

On the other hand, live migration technique transfers the virtual machine while it still runs. The live migration should not cause a perceptible downtime to the VM user. Online migration technique allows administrators to move running virtual machines (VMs) among different physical hosts without service interruptions. This is a very useful tool for service providers hosting a high number of applications. The quality of service that a service provider can guarantee when hosting and running an application is described by a Service Level Agreement (SLA) and is typically expressed in terms of application availability. The majority of policies permits

very low value of downtime (in order of seconds); so routine activities such as restarting a machine for hardware maintenance are extremely difficult under these conditions. Live migration mitigates this problem by allowing administrators to move VMs with little interruption, and avoiding them to invest considerable resources in fault-tolerant systems. This new approach permits regular maintenance of the physical hardware, supports dynamic reconfiguration and shifts workload to another host to better manage the allocation of resources. However, short interruptions of service are still unavoidable during live migration due to the overheads of moving the running virtual machine. Recent studies have demonstrated that service interruption duration can vary considerably depending on which application is running on the machine, due to the different memory usage requirements. Migrating an entire virtual machine as a single entity, means that active memory and execution state are transferred from the source to the destination. This allows seamless movement of online services without requiring clients to reconnect or login. On migration completion, virtual I/O devices are disconnected form the source and re-connected on the destination physical host, so migration means to copy in-memory state and CPU registers.

There are more than one technique for live migrate a virtual machine, but all of them have to take into account the importance of two parameters: total migration time and downtime. Total migration time refers to the total time required to move the machine from the source host to the destination host; while downtime is the period of time in which the virtual machine is not running (it is shut down). The first migration approach plans to halt the original virtual machine at the source and copy its entire memory to the destination. This technique minimizes total migration time but incurs high values of downtime because the machine is suspended during the entire transfer process. A second procedure operates by stopping the VM to copy only the necessary information and data to the destination while the rest of the memory can be transferred when it is accessed at the destination. This second technique has a very short downtime, but it suffers from high total migration time. It has previously been shown that both presented migration approaches have poor performance.

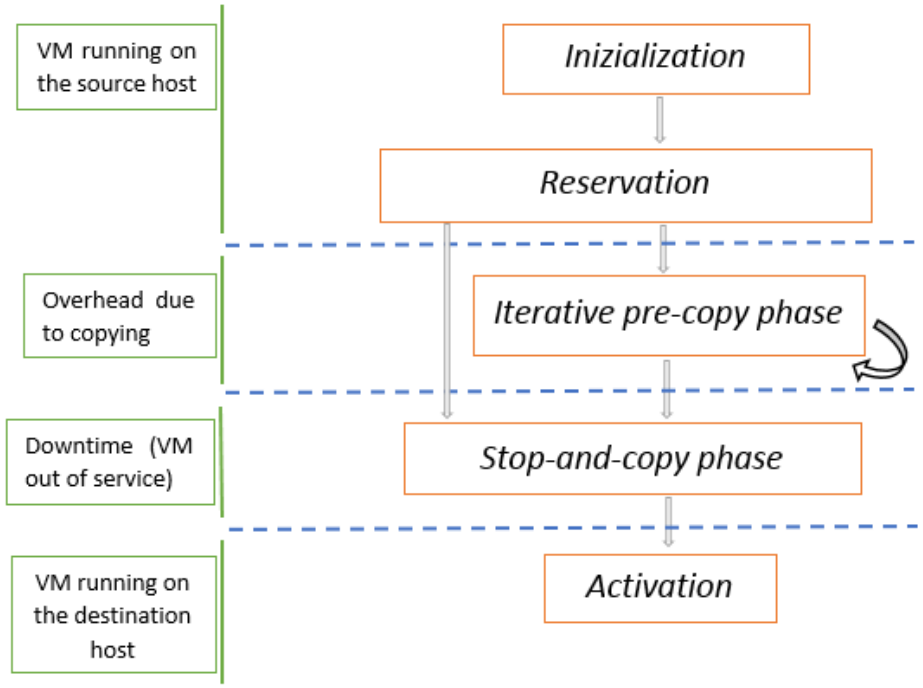

Figure 2.2: Virtual Machine Migration Timeline

The former may lead to significant service interruption especially if lots of applications are running on the virtual machine, while the latter incurs a longer total migration time. The better strategy to address the problems associate with both the previous proposed procedures can be the pre-copy migration, because it try to combine an iterative step approach with a final and typically very short stop-and-copy phase. The basic idea is to perform multiple iterative rounds of copying: in each round the VM memory pages that have been modified since the previous iteration are resent to the destination until a stopping condition is met. This can be a situation in which the number of remaining modified pages will be small enough (less than a prefixed threshold) to halt the VM for a short period of time, in which these remaining pages are copied and the virtual machine is restarted on the destination host. A migration scheme like this one, is able to minimize both total migration time and downtime. Pre-copy migration can be divided into multiple stages (as shown in Figure 1.2), namely:

- *Initialization*: a host is pre-selected as the target of the migration.

- *Reservation*: resources at the destination host are reserved.
- *Iterative pre-copy phase*: the entire memory is sent in the first iteration, and in each subsequent iteration pages modified during the previous iteration are transferred to the destination.
- *Stop-and-copy phase*: the virtual machine is halted at the source host for a final transfer round.
- *Activation*: the virtual machine is resumed at the destination host.

The iterative pre-copy phase may continue indefinitely, thus, it is imperative to define a stop condition; usually the condition can be defined depending on the downtime or the number of iteration of the iterative pre-copy phase of the migration process. We can evaluate the migration performance by monitoring already presented metrics: total migration time and total downtime. We can consider the migration duration as the period in which both virtual machines are active while during the downtime the virtual machine is suspended and so, customers see it as out of service. Adopting the pre-copy approach, total migration duration may be defined as the sum of the time spent on all migration stages, from the initialization phase at the source host to the activation stage at the destination host. However, downtime is the time required for the last two stages: stop-and-copy phase and activation. We expect that iterative pre-copy stage will be the dominant factor in the migration process, but we found that, also other stages may require a significant amount of time, so we classify the initialization and reservation phases as pre-migration phase and activation as post-migration phase. In order to describe a realistic and accurate model for the migration, we have to take into account two important factor that affect the entire process: migration bandwidth and dirtying rate. Bandwidth assigned to the migration is strictly related to link capacity (more precisely to residual bandwidth on links) and it is inversely proportional to total migration time and downtime. Higher values of bandwidth allow faster transfers and thus it requires less time to complete the migration. Dirty rate (or page dirty rate) is the rate at which virtual machine memory pages are modified, thus, it

directly affect affects the number of pages that are transferred in each iteration of the pre-copy phase. Higher values of dirtying rates result in more data being sent per iteration which leads to longer total migration time. Furthermore, higher page dirty rates may result in longer downtime because there are more pages that need to be sent in the final transfer round, in which the virtual machine is suspended. The relationship between dirtying rate and migration performance is not linear because of the stop conditions. If the dirtying rate is lower than link capacity, the migration process is able to transfer all modified pages in a very quickly way, resulting in a low total migration time and downtime. On the other hand, if the dirtying rate takes values very close to the link capacity, migration performance degrades significantly. At lower dirtying rates downtime can be considered a quite constant value, because network links have enough capacity to transfer dirty memory pages in subsequent iterations leading to a very short stop-and-copy phase. However when dirtying rate increases to the point that a stop condition is reached (when a prefixed number of iteration is reached or when the number of memory pages to be transferred is greater than a predetermined threshold value), migration process is forced to enter the final stage with a large number of dirty pages yet to be sent. Therefore, downtime starts to increase proportionally to the growth of the number of modified memory pages that need to be transferred in the stop-and-copy stage. Total migration time also increases with an increasing dirtying rate, due to the fact that more pages have to be transferred in each pre-copy round and then you need more iterations to achieve a short final stop-and-copy round. In the first pre-copy iteration the entire allocated memory of the virtual machine is copied to the destination host.The duration of this first iteration is thus directly proportional to the virtual machine memory size and, obviously, it impacts total migration time. On average, total migration time can be assumed linearly increasing with memory size of the virtual machine. On the other hand, for low dirtying rates downtime is almost the same regardless of the memory size as the migration procedure copies all dirtied pages in the successive iterations resulting in a short stop-and-copy phase. If free capacity of network link is unable to support the dirtying rate, then larger VMs suffer longer downtime (linearly proportional to the VM size) because there

are more physical pages that require copying in the stop-and-copy stage. Also pre and post migration can generate overheads, such as required time for operations that are not part of the transfer process. These operations are related to the initialization phase on the destination host, maintaining free resources and resume the virtual machine. As these procedures require constant amount of time, they are significant especially with higher link speeds.

2.3 Live Migration Model

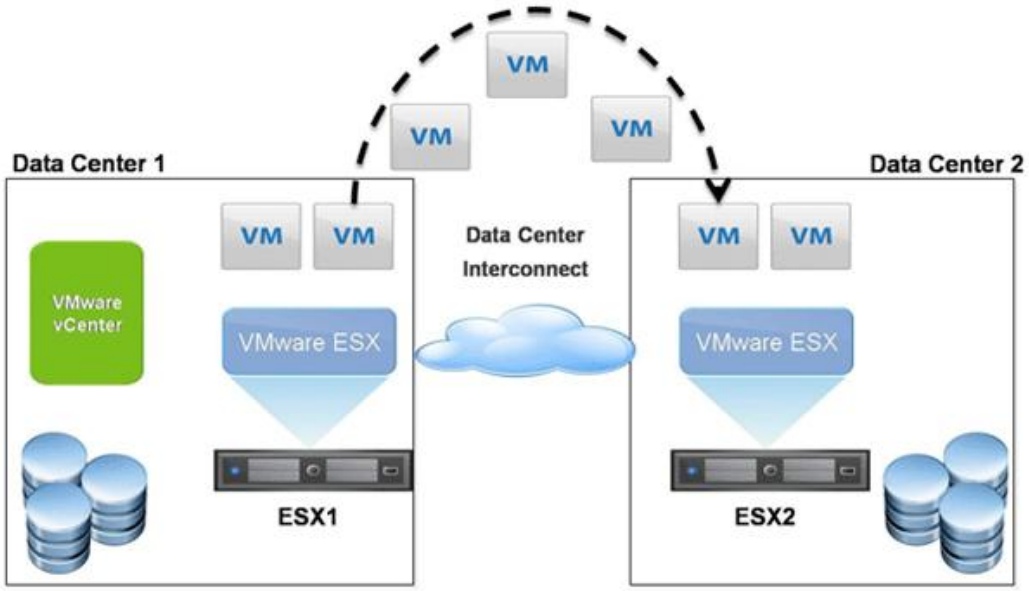

Figure 2.3: Virtual Machine migration between two data center

A virtual migration over a Wide Area Network is carried out by first establishing a network connection between source and destination locations, then transferring the memory and disk storage states, and finally by re-configuring the virtual machine and resuming the service at the destination location. In our study we consider the iterative pre-copy-based live migration approach of virtual machines. In Figure 2.4 there is an example timing diagram of this process. In the first iteration, the entire virtual machine memory is copied to the destination host. In subsequent iterations is copy only modified or 'dirtied' memory from previous

iteration. We, then, consider a small delay inter-iteration τ, given by end-to-end network delay (also known as Round-Trip Time), processing delay at both source and destination location, or a combination of both. The duration of each iteration, therefore, can be evaluated as the sum of memory transmission time and the inter-iteration delay. The iterative copy phase is stopped when a predefined number of iterations has been performed or the amount of dirtied memory is low enough to achieve a downtime constraint. A final stop-and-copy phase is then started, in which, the virtual machine is suspended at its source, dirtied memory is copied to the destination, and the network is reconfigured before resuming the virtual machine at the destination location. Obviously, this last stop-and-copy phase causes virtual machines and hosted applications/services to be unavailable. The amount of dirtied memory transferred in each iteration is related to the characteristics of the virtual machine and its hosted applications. While this amount of data can vary for different iterations, an average dirtying rate can be assumed for a particular VM type. Thus, we can assume dirtying rate constant and indicate it by D; we also denote the number of iterations in the pre-copy phase by n, the amount of memory transferred in each iteration i by V_i, and its duration (in seconds) by T_i. Remember that T_i consists both of the memory transmission time and the inter-iteration delay, τ. Memory copied during the first iteration, V_1, is equal to the size of the memory of the virtual machine, denoted V^M. As bandwidth provisioned for migration is R, the amount of data transmitted in round i can be represented as:

$$V_i = \begin{cases} V^M, & if\ i = 1 \\ D * T_{i-1}, & otherwise \end{cases} \tag{2.1}$$

The duration of the iteration i can be calculated as:

$$\begin{aligned} T_i =& \frac{V_i}{R} + \tau = \begin{cases} \frac{V^M}{R} + \tau, & if\ i = 1 \\ \frac{(D * Ti-1)}{R} + \tau, & otherwise \end{cases} \\ =& \frac{V^M}{R}\lambda^{i-1} + \tau\frac{1-\lambda^i}{1-\lambda} \end{aligned} \tag{2.2}$$

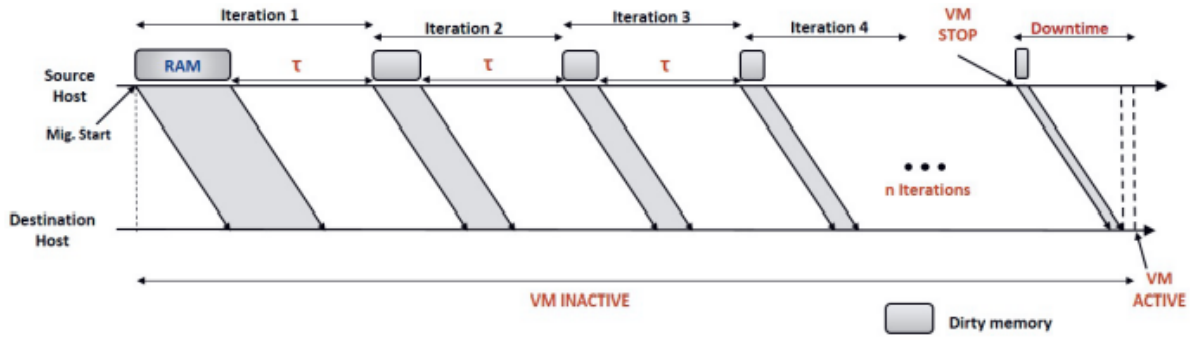

Figure 2.4: Timing diagram of iterative pre-copy-based VM migration technique

In these formulas we note the presence of the parameter λ: it is the ratio between the dirtying rate and the provisioned bandwidth for the migration (λ = D/R). We can easily say that λ has a key role in the virtual machine migration process because it is used to determine a reasonable value for the bandwidth assigned to the migration request. We can easily note that this is a closed form representation of the iteration time t_i but it is valid for any value if i from 1 to the maximum number of iteration n. Once we have defined the duration of each iteration of the iterative phase, we can define the total migration time ($Tmig$) as:

$$\begin{aligned} T_{mig} &= \sum_{i=i}^{n} T_i + t_{down} \\ &= \frac{V^M}{R} * \frac{1-\lambda^{n+1}}{1-\lambda} + \tau \frac{n(1-\lambda) - \lambda(1-\lambda^{n+1})}{(1-\lambda)^2} + t_{down}, \end{aligned} \tag{2.3}$$

where t_{down} represents the duration of the stop-and-copy phase, in which the virtual machine is suspended, and then, users see it as unavailable. Knowing that the iterative phase performs n iterations, the amount of data (dirtied memory phase) to be transferred after the virtual machine is stopped, is given by V^S:

$$\begin{aligned} V^S &= T_n * D \\ &= V^M \lambda^n + \tau \frac{1-\lambda^i}{1-\lambda} D \end{aligned} \tag{2.4}$$

Remembering that downtime includes both final memory copy and network re-

configuration time, the duration for which the VM is down is given by t_{down}, as:

$$\begin{aligned} t_{down} &= \frac{V^S}{R} + t_g \\ &= \frac{V^M}{R}\lambda^n + \lambda\tau\frac{1-\lambda^i}{1-\lambda} + t_g \end{aligned} \tag{2.5}$$

where t_g is the network re-configuration time. In conclusion, we can say that the total time required to migrate a virtual machine and the total amount of data to be transferred depend on the memory modification (or dirtying) rate, provisioned network bandwidth, and inter-iteration delay (which contributes to the duration of a single iteration). In the next session we will try to explain how these factors can influence the entire process of the live migration of a virtual machine and why a further investigation on their behavior can help in discovering and proposing new bandwidth allocation methods to reduce network resource consumption for virtual machine migration.

2.4 Migration Parameters

After having presented a model for live migration of virtual machines, we use Figure 2.5 and Figure 2.6 [6] to show an example of the behavior of total migration time and downtime in function of different values of dirtying rate for a virtual machine with memory size equal to 4 GB (inter-iteration delay, τ, and network reconfiguration time, t_g, are on the order of few hundred milliseconds). In Figure 2.5 the y-axes represent migration duration (in seconds) while in Figure 2.6 they represent migration downtime (in seconds); the x-axis, in both graphs, represents range of λ; remembering that λ is the ratio between dirtying rate and bandwidth, it is easy to understand that the x-axis can represents a range of values for the bandwidth provisioned for the migration.

From model equations we can assume that there are two parameter we can use to monitor the migration process and its performance: provisioned bandwidth R and number of iteration n of the iterative phase. Even though provisioning bandwidth is important, we note that the ratio between dirtying rate and bandwidth

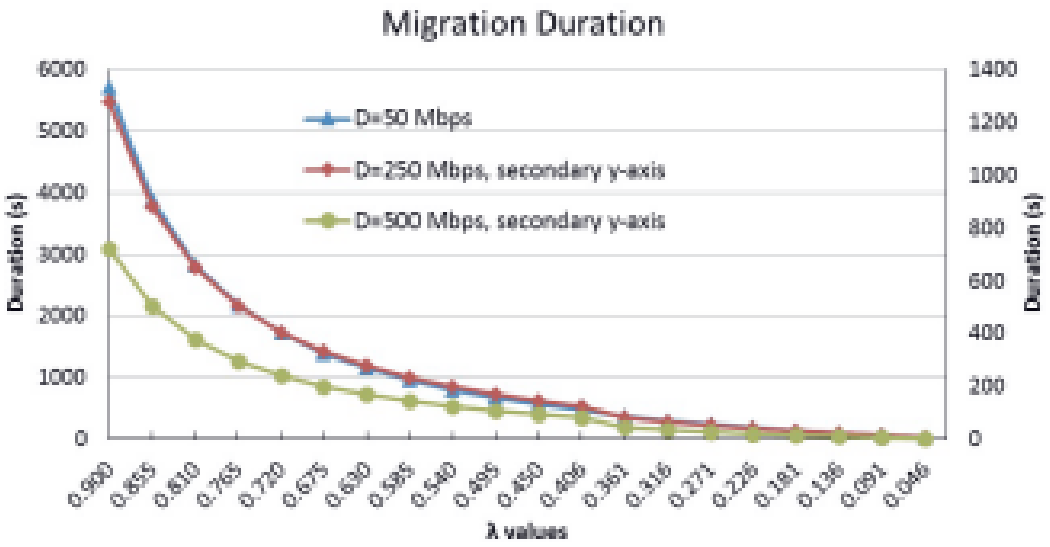

Figure 2.5: Total migration duration for different D

can be more important, how you can see from model equation. We make an example to clarify this assumption. If λ is greater than one (it means that dirtying rate is greater than bandwidth), then the migration bandwidth will be not sufficient because the VM's memory will be modified (or dirtied) faster than it can be transmitted, and so the iterative phase can be very long and the whole memory has to be transmitted in the last stop-and-copy phase, but in this way you get a high downtime. On the other hand, if λ is lower than one (it means that bandwidth is greater than dirtying rate), you can transmit memory faster than it can be modified, and thus it is possible to achieve reasonable values of downtime and total migration duration. Assuming $\lambda \leq 1$, we know that assign higher bandwidth means lower migration duration, but we want a value of provisioned bandwidth such that it is possible to ensure sufficient bandwidth for all the migration requests we have to manage. So, identifying an optimum value for the parameter λ is a challenge. Once we have discovered the importance of the λ parameter and, consequently, the provisioned bandwidth, we can focus our attention on the other parameter used to derive this ratio: the dirtying rate. Depending on the workload running on the migrated virtual machine, which it cannot be changed, dirtying rate has a

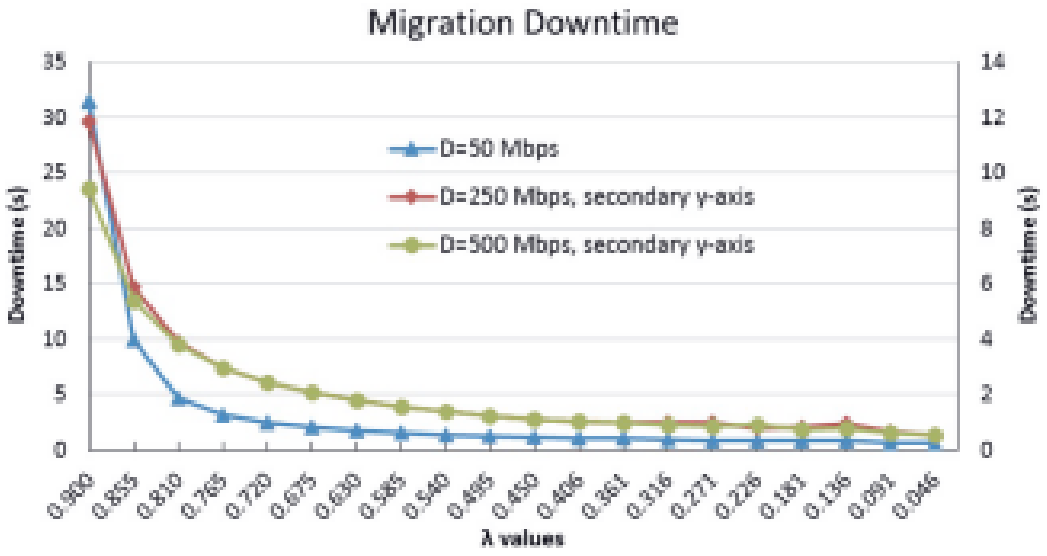

Figure 2.6: Migration downtime for different D

key role in determine how to allocate a reasonable value for the provisioned bandwidth to improve the performance of the live migration model. For example, in multi-tenant clouds, it could happen that a virtual machine allocated to a tenant (data center) needs to be migrated to another data center to make a better use of physical and hardware resources. If the total migration duration is large it means the waiting time of a new tenant becomes too long; but also a large downtime can create problems as it can violate the Service Level Agreement of the service. In addition, the required bandwidth for the transmission can change in each iteration on the pre-copy approach. Therefore, to address all these drawbacks, the dirtying rate should be taken into account to determine a constant bandwidth value during the pre-copy phase. The goal of optimizing total migration duration, downtime together with network resource consumption is a complex problem. In fact, high bandwidth will reduce migration duration, but at the same time it may increase network resource utilization. By the other side, low bandwidth can reduce resource consumption, but it also increase total migration time as well as downtime. Same considerations can be made for the number of iterations in the pre-copy stage.

Analyzing an other time equations for the calculation of migration duration and

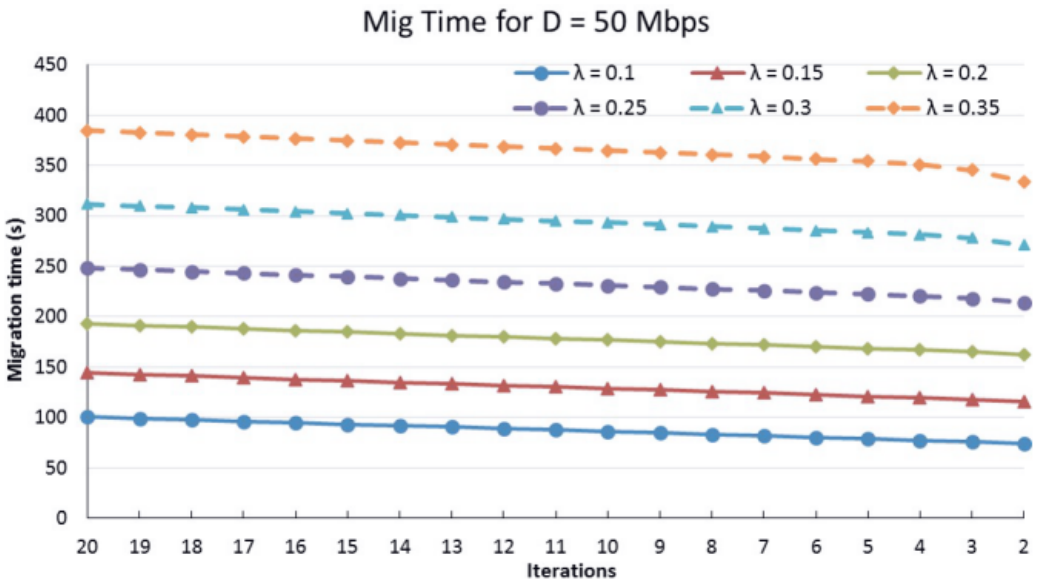

Figure 2.7: Migration duration for different iteration

downtime of the migration model, it is easy to observe that exist a relationship between migration time and number of iteration. In Figure 2.7 [6] the y-coordinate represents total migration time (in seconds) while in the x-coordinate there is the number of iterations. You immediately notice that total migration duration regularly decreases with decreasing number of iterations, for each one of the considered λ values. Even if it is simple to understand that this behavior can be explained thinking that more iterations require longer duration, it allow us to formulate an important observation: if the goal of a migration is to reduce total migration time, it is desirable to perform as few iterations as possible, and provision as high bandwidth (thus lower λ) as possible. In fact, higher values of λ parameter involve bandwidth values very close to the dirtying rate: it means that in each iteration transferred amount of data (memory pages) is low and and it requires more time to complete the transmission. In addition, more iteration are needed, so total migration duration is larger and network resources utilization increases as they are allocated for longer time.

2.5 Function Points Curves

In the most of the live migration techniques, performance is usually measured in terms of total migration time and system downtime. All existing systems try to control migration time by limiting the rate at which memory pages can be transferred while downtime is determined by evaluating for how much time the service (the application running on the virtual machine that has to be migrated) is found unavailable from users. Minimizing both of these metrics is the key to obtain optimal performance for a live virtual machine migration system and it is usually achieved using a control system as the following. The virtual machines administrator sets configuration parameters for the migration process, hoping that these conditions can be met. These input parameters are a limit to the network bandwidth for the migration procedure and an acceptable downtime for the last iteration of the pre-copy phase of the migration. But imposing a limit for the bandwidth that can be assigned to a migration request, without taking into account the important role played the dirtying rate, can bring some troubles: such as it can result in a backlog of pages to migrate or in a greater total migration duration. A first widely used idea can be to set a high bandwidth limit, but it can affect the performance of running applications. Another kind of solution is to determine a limit on the number of iteration that can be performed or to increase the allowed downtime; but both these idea can make live migration equivalent to pure stop-and-copy migration.

So we are looking for a method that allow to assign a reasonable bandwidth value for the migration. We want a model that take into account the trade-off existing between bandwidth, migration time and resource occupation: higher bandwidth involves shorter total migration duration but it consumes more resources (even if for a short period), while lower bandwidth means longer time required to perform migration but it consumes resources for a longer period (even if the amount of allocated resources is not a great value). In addition to this, also the dirtying rate has to be considered in the estimation of the correct value of bandwidth: in this way we want our model to provide a reasonable bandwidth for each virtual machine that needs to be migrated. To reach this purpose, we can use the

equations from the model proposed in the previous sections to provide a relationship between total migration duration and provisioned bandwidth. In particular, we can determine the value of the metric T_{mig} for several bandwidth values. These bandwidth values were derived from the definition of the λ parameter in the following manner: once we'we fixed the virtual machine's dirtying rate, we evaluate the bandwidth by varying the value of λ (obviously it must be less than 1 otherwise memory is dirtied faster than it can be transferred to the destination data center). In this way, we are able to establish the sought relationship by building a curve from the points just calculated. Moreover we have to specify all the other involved input parameters, such as:

- V^M: virtual machine memory size
- R: provisioned bandwidth
- D: virtual machine dirtying rate
- λ: ratio between dirtying rate and bandwidth
- n: number of iteration of the iterative pre-copy phase
- τ: inter-iteration delay
- t_g: network reconfiguration time

In order to take correct values for these parameters we have made an exhaustive research in the literature to better understand how they are involved in a multi data center cloud environment (and which values they assume). In particular, we took inspiration from the study of Uttam Mandal and its research group in the essay 'Bandwidth Provisioning for Virtual Machine Migration in Cloud: Strategy and Application'. Thus, in our calculations, we have considered the following assumptions:

- we have considered three different values for the dirtying rate: 50 Mb/s, 100 Mb/s and 500 Mb/s

- for each dirtying rate we have examined three different virtual machine memory size: 100 MB, 1 GB, 10 GB
- we have imposed n=20 as maximum number of iteration for the iterative pre-copy phase
- network reconfiguration time t_g is equal to 1 second
- inter-iteration time τ is equal to 100 ms
- λ takes values in the range (0,1)

As you can note by the figures below, there exists an inverse proportionality relation between migration duration and bandwidth. In particular, you can observe that, it is not convenient to choose too high bandwidth values to perform migration because it non brings benefits in terms of total migration time. So it is useful to choose a bandwidth value such that select a higher bandwidth does not perform a significant reduction of the migration duration. Such a value is named 'function point' (hence the name of 'function points curve' for the just build graph). In conclusion, the function point curve represent a set of possible bandwidth value to assign to a migration request together with the correspondent total migration duration for each one of the considered bandwidth values. In the next pages we show the function points curves for each one of the virtual machine topologies examined in this study.

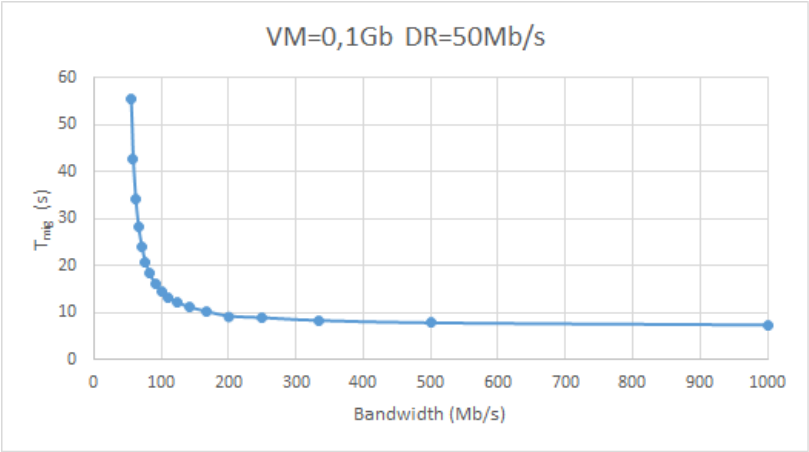

Figure 2.8: Function points for D=50Mb/s and V^M=100MB

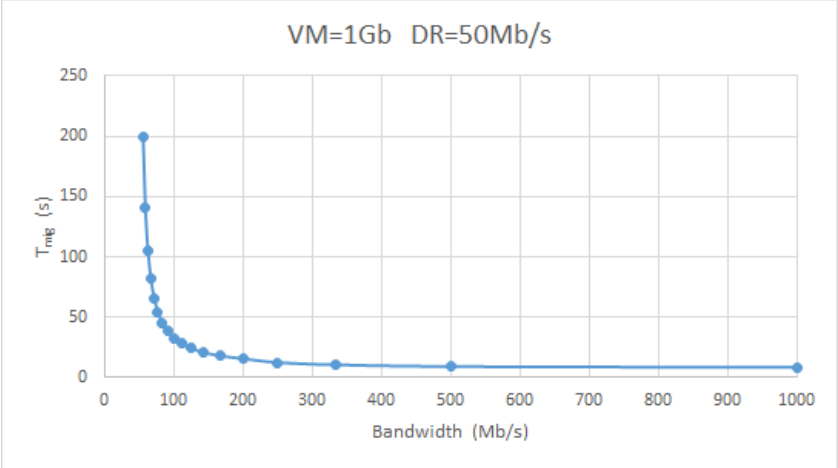

Figure 2.9: Function points for D=50Mb/s and V^M=1GB

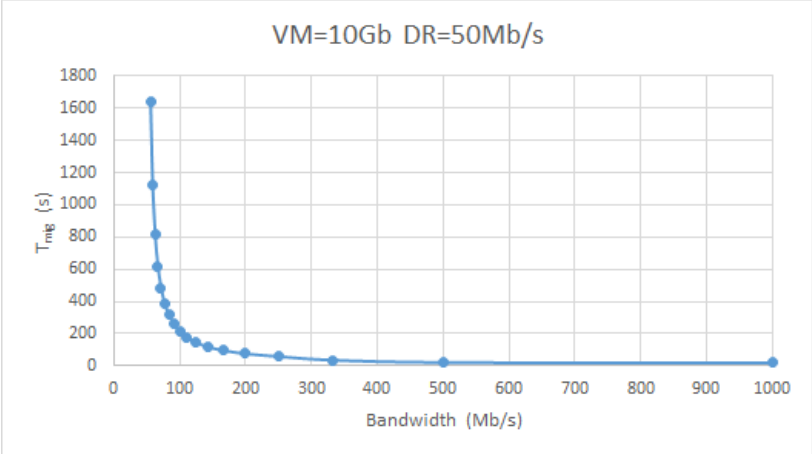

Figure 2.10: Function points for D=50Mb/s and V^M=10GB

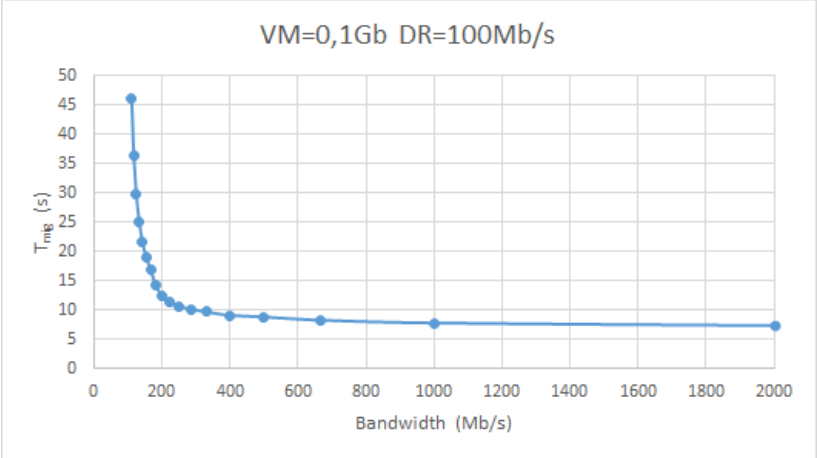

Figure 2.11: Function points for D=100Mb/s and V^M=100MB

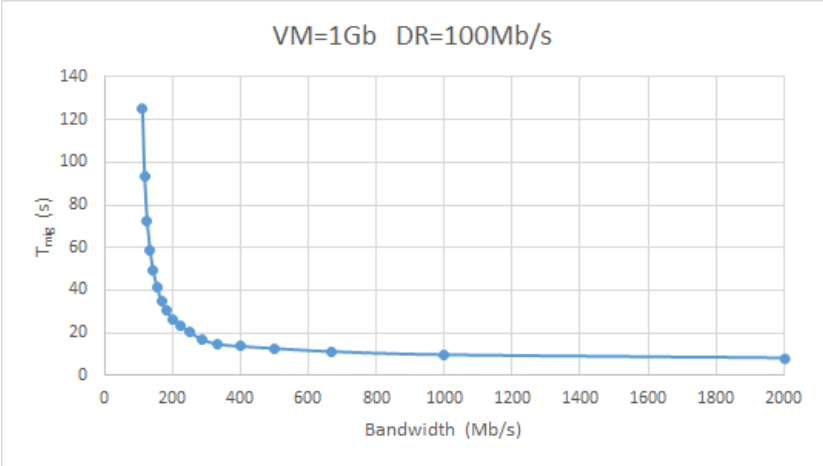

Figure 2.12: Function points for D=100Mb/s and V^M=1GB

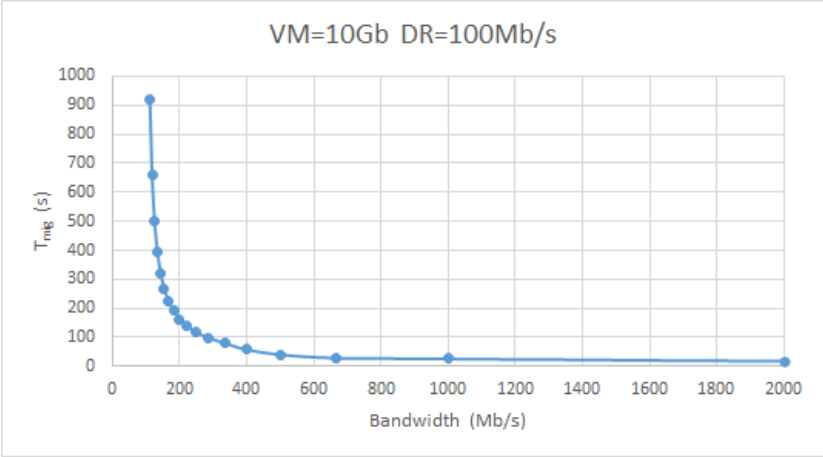

Figure 2.13: Function points for D=100Mb/s and V^M=10GB

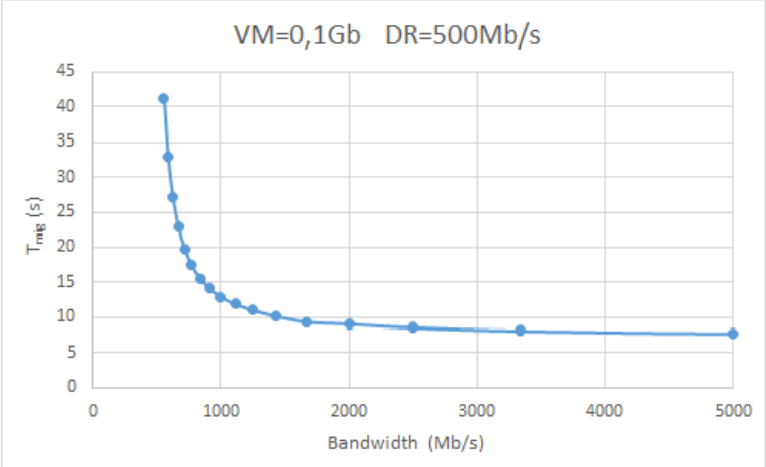

Figure 2.14: Function points for D=500Mb/s and V^M=100MB

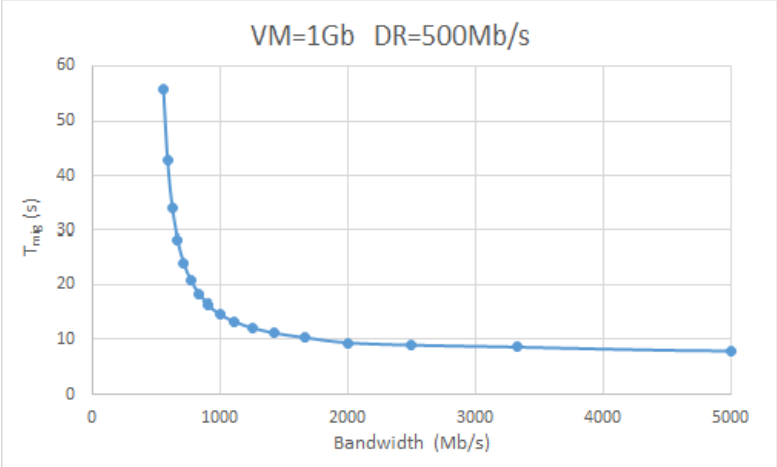

Figure 2.15: Function points for D=500Mb/s and V^M=1GB

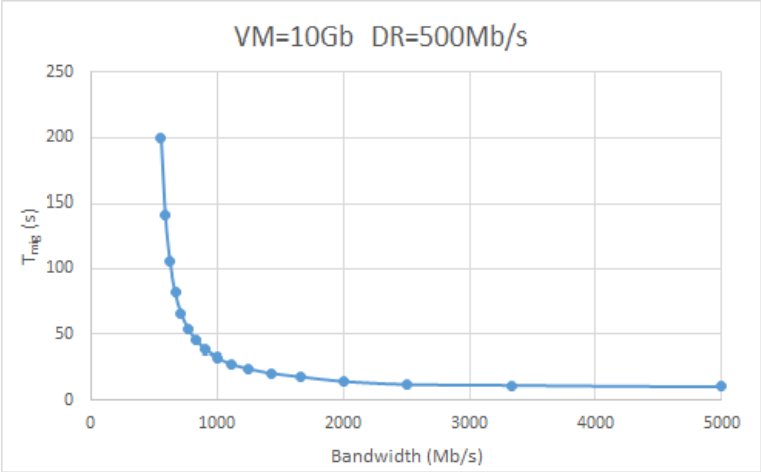

Figure 2.16: Function points for D=500Mb/s and V^M=10GB

Chapter 3

Routing and Bandwidth Assignment Algorithms For Live Virtual Machine Migration

In recent years cloud computing is covering a fundamental role in developing and running virtualized applications and Internet services. Cloud infrastructures usually consist of geographically-distributed public data centers, connected together by wide area networks by means of high-capacity optical channels. As a consequence, the optical network has to support increasing traffic demands, due to users requiring cloud service, as well as the one generated by the communication between data centers. Hence, it is necessary to look for an efficient bandwidth provisioning scheme, for a better resource allocation in the network. As in a cloud environment physical servers, placed into data centers, are virtualized, you can expect that migrating virtual machines over the network can give an important contribution to the growth of network traffic. In addition, virtual machines migration requires a significant amount of bandwidth between the source and destination data centers. This scenario tell us that it can be very useful to study new techniques that allow to assign enough bandwidth to migration request while avoiding a waste of resource. Indeed, even if assigning high bandwidth is important to achieve good performance, the network may not be able to support future capacity requirements due to high network utilization and capacity exhaustion.

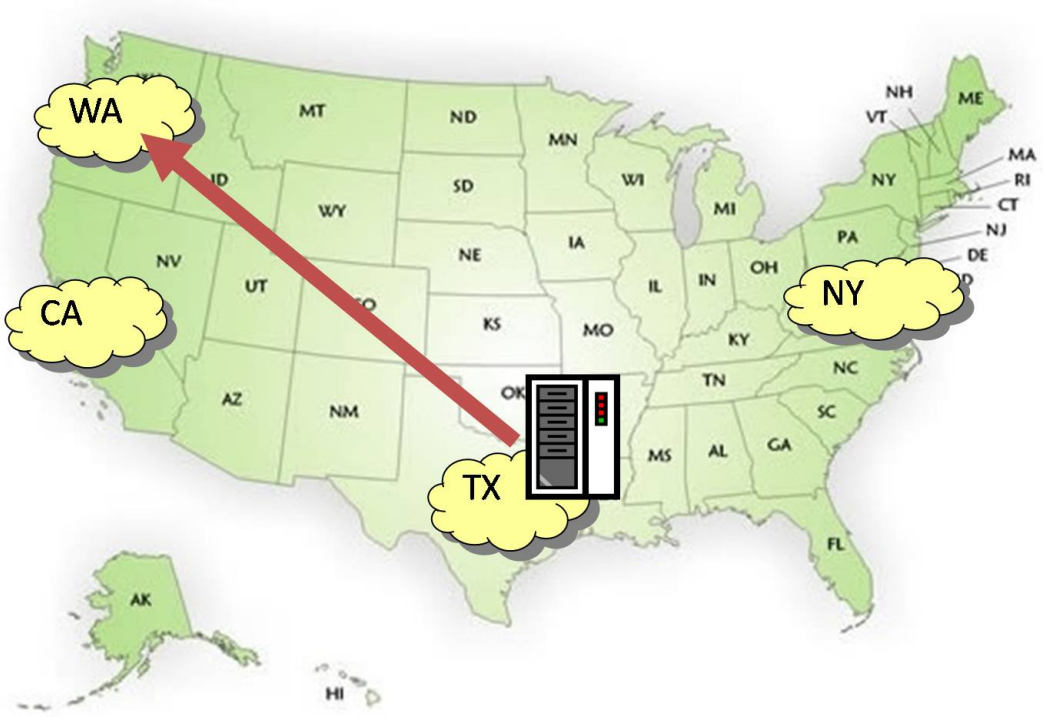

Figure 3.1: Virtual Machine Migration over a WAN

By the other side, assign lower bandwidth is useful to avoid high level of network utilization but it can result in worse performance. Hence, you have to take care of this tradeoff among performance and network resource utilization in evaluating the bandwidth required to migrate a virtual machine. These are the reasons that led us to investigate new algorithms for the Routing and Bandwidth Allocation problem for live migration of virtual machines. In this chapter,after presenting some works related to these topics, we first give a general description of the general problem of allocating bandwidth in a wide-area network and then, we present some RBA algorithms that can be adopted in case of virtual machines migration.

3.1 State of the Art

Numerous publications in recent years have proposed solutions for virtual machines migrations in a Cloud environment. We present in this section some of the most interesting works on this subject, in order to present the problem in all its aspects.

From reading [1] we can learn how, in this last few years, Cloud Computing (CC) services are increasing; as a consequence, also the the energy consumption of the network and of the computing resources supporting cloud systems is growing, causing the emission of enormous quantities of CO_2. As a solution new routing algorithms are presented, in order to route user requests from data centers powered by renewable energy sources (RES). However, due to its intermittency and volatility, renewable energy cannot be used to its full potential. In facts, in [2] a new solution is proposed to address the problem of high energy consumption in a cloud environment: virtual machines migration can be used to relocate energy demands and to achieve a better network resources utilization. Results show that implementing migration techniques in the US cloud network scenario can bring to a significant reduction of non-renewable energy consumption (up to 30 percent) while consuming a small amount of extra resources. Nevertheless significant bandwidth is consumed during this process. Even if VM migration over a local-area network (LAN), within a single data center, causes not relevant performance degradation and downtime, migration over a wide-area network (WAN) with significant end-to-end delay may cause undesirable performance effects. High network bandwidth can reduce these impacts. In [6] authors propose quantitative models for migration duration and downtime over a WAN. Based on these models, they devise a strategy to determine appropriate migration bandwidth. In [5] there is a study demonstrating that heterogeneous bandwidth (instead of homogeneous bandwidth) for migration reduces significant resource consumption in SDN-enabled optical networks. In [8] is presented a model for migrating Operative Systems running services with liveness constraints, achieving impressive performance with minimal service downtime. In [9] a cloud computing platforms linked with a virtual private network (VPN)-based network infrastructure is presented, aiming to provide connectivity between enterprise and cloud data center sites. As ICT constitutes a significant portion of the worldwide energy consumption, reduction in operating energy expenditures is a high priority for cloud-service providers. Authors in [7] analyze the key parameters that affect the migration cost by implementing a model for the cost prediction by using learned knowledge about the workloads at the hypervisor level. It is the first kind of work to estimate VM live migration cost in terms of both performance and energy in a quantitative ap-

proach. Another approach is to exploit live VM migration for moving virtualized workloads towards those data centers with cheaper energy price: [4] illustrates how a cost-efficient VM migration based on varying electricity prices, significantly reduces the energy cost to operate cloud services. In [11] authors explore the security issues involved in live migration of VMs and demonstrate the importance of security during the migration process. A model which demonstrates the cost incurred in reconfiguring a cloud-based environment in response to the workload variations is then studied. Live VM Migration can consume nearly the entire bandwidth which impacts the performance of competing flows in the network so data center administrators are looking to intelligently reserve minimum bandwidth required to ensure a network-aware VM migration. In [10] are evaluated the performances of Remedy, a cost estimation model to calculate total traffic generated by the migration of virtual machines.

As we found out by analyzing the literature, many studies have been conducted concerning the migration of virtual machines; in particular they show what the benefits it brings (load balancing, fault recovery, cloud bursting and so on). However we must also analyze any problems that may arise migrating a virtual machine. The aspect that seems to require further studies is the very high level of resources consumption, that migration requires, bandwidth requirements in particular. In [3] researchers formulate the bandwidth and routing assignment problem for virtual machines migration with duration and downtime constraints in photonic cloud networks, showing that optimal bandwidth and routing assignment can reduce network resource utilization. However,there is still not a solution to the problem of bandwidth allocation for a network scenario that involves migrations of virtual machines among data centers. Therefore, in this work, we propose Routing and Bandwidth Assignment algorithms for a network scenario where the offered traffic is e one generated by the migrations of virtual machines between data centers. Hence, we aim to propose a solution to the RBA problem so that migrations can be carried out efficiently while ensuring an intelligent use of network resources.

3.2 Routing and Bandwidth Assignment Problem

Unlike in simple file transfers, virtual machine total migration duration does not depend linearly on data size (memory pages) and assigned bandwidth.

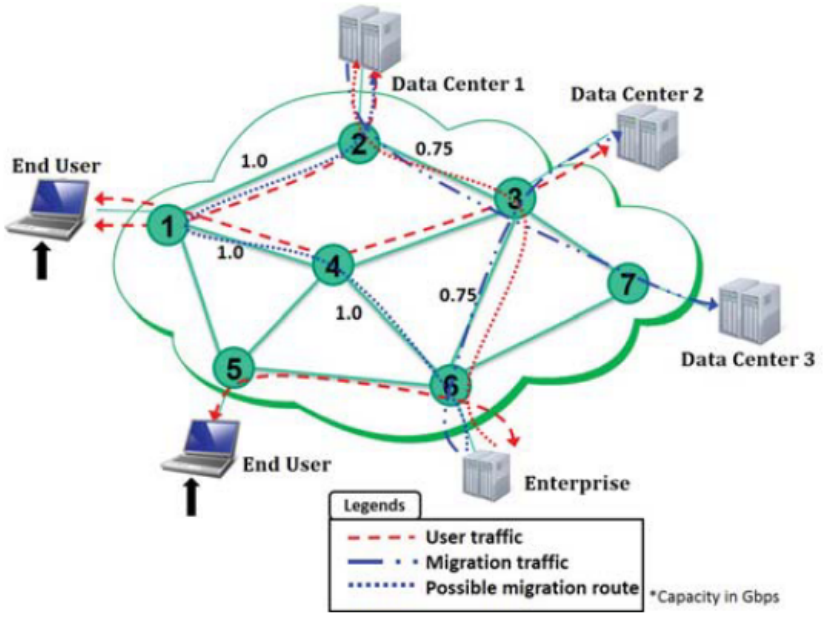

Figure 3.2: Cloud architecture network with different connections

It also depends on virtual machine's features and on application characteristics as well as stop condition of the iterative pre-copy phase in a live migration approach. In fact, usually, migration requests have to achieve latency and downtime requirements: in other words, there is a maximum threshold value for the total migration time until which migration has to be performed, guaranteeing that downtime is less of equal a prefixed value. Therefore, enough bandwidth must be allocated to the migration request to respect these constraints while aiming to an intelligent allocation of network resources in order to reduce the degradation of network performance. For this purpose, we took care of propose some algorithms for the Routing and Bandwidth Allocation problem for this new scenario: a network topology in which some nodes represent the location of data

centers, and the offered traffic is the one generated by the migration of virtual machines (i.e. large amount of data to be transferred and routed between data centers geographically-distributed in a wide-area network). Figure 3.2 [3] shows an example cloud architecture connected via an IP-over-WDM photonic network, where you can see both user and VM migration traffic. Routing and Bandwidth Assignment problem involves looking for an algorithm that allow to calculate a path between the source and the destination nodes (data centers, in this case) of the connection and to assign it enough bandwidth such that all requests can be provisioned. Such an algorithm takes as input the set of connection requests; in our case each connection is a virtual machine's migration request. In particular a request is characterized by several parameters:

- *source node*: the data center from which the virtual machine has to be migrated,
- *destination node*: the data center where the virtual machine has to be migrated,
- *virtual machine memory size*: the amount of data (memory pages) to transfer,
- *dirtying rate*: rate at which data is modified during the iterative pre-copy phase while the application hosted on the virtual machine is still running.

Our goal is not only to develop these algorithms but we also want to evaluating their performances both in terms of blocking probability and in respect to migration metrics such as total migration duration, downtime and number of iteration of the iterative phase. In this chapter we will present four algorithms: Min and Max provide bandwidth in a static manner, than, with the third algorithm, Congestion-Aware, we try to calculate bandwidth taking into account the network state and finally, we propose a more intelligent algorithm that took inspiration from previous results. Analyzing the whole problem, we can say that we focus on a scenario of migration request only, taking as input the network topology, the traffic (migration requests) distribution and its arrival rate. Then we proceed going through the Routing and Bandwidth Assignment problem (choosing one of

the proposed algorithms), after which a connection can be provisioned or dropped. In a final phase resources are released and performance of algorithms evaluated. Going into detail of the RBA problem, we can say that each algorithm presents the same scheme and it can be divded in three phases:

- *Routing phase*: k shortest path calculation
- *Provisioning phase*: Bandwidth Assignment problem and resource allocation
- *Deprovisioning phase*: release resources

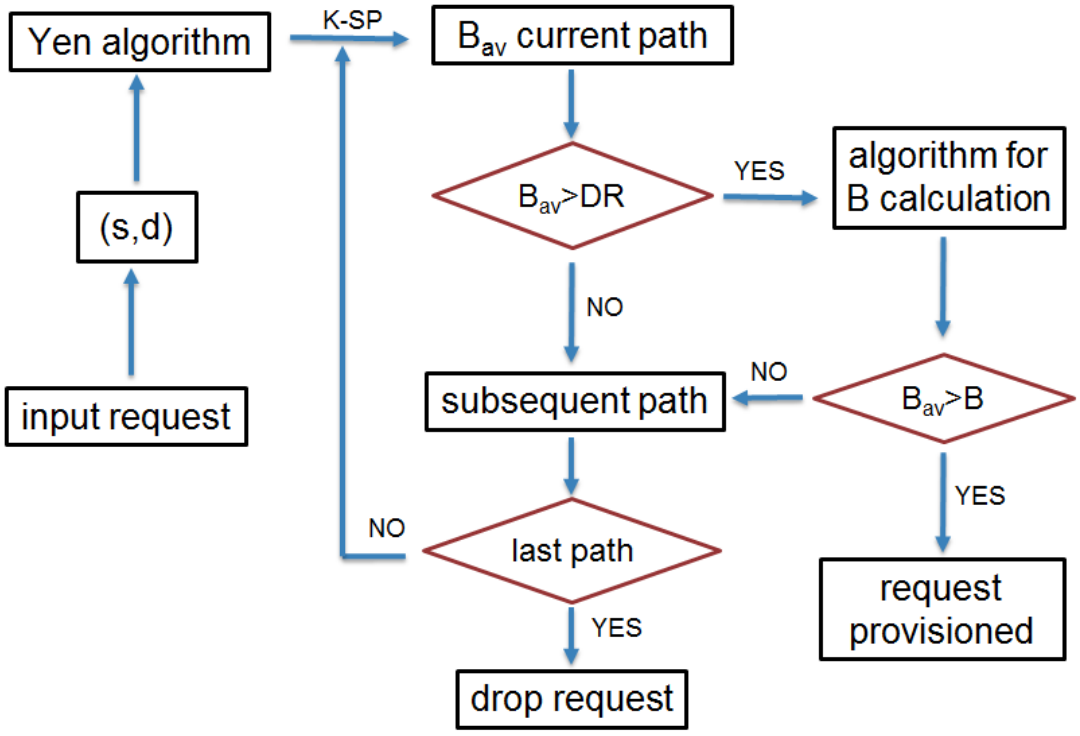

Figure 3.3: Flow chart of Routing and Bandwidth Assignment algorithm

In figure 3.3 there is a flow chart that clarifies how is the general scheme of each one of our algorithms and how it works. For each migration request the source and destination data centers are extracted and the Yen algorithm is applied for each one of these couple of nodes, in order to calculate k shortest path for the migration

request. Starting from the first, for each one of the paths, the available bandwidth (B_{av}) on the path is evaluated, then there is a first control: we check if the free bandwidth B_{av} is greater than the dirtying rate (D). If yes, it proceeds with the calculation of the migration bandwidth B using one of the proposed algorithms; if free bandwidth is grater than B, than the migration request is routed, resources are allocated and migration is performed. If B is less than B_{av}, then, next path is analyzed and the same procedure is applied for all the k shortest paths, unless a path with enough free bandwidth to perform migration is found. If such a path is not found, the migration request is dropped. In the next sections we will explain in details how the four algorithms work.

3.3 RBA Algorithm - Min

As we are considering a live pre-copy based model for virtual machines migration, we have to take into account the key role played by the dirtying rate within this process when we have to decide the bandwidth value to assign to a migration request. We have to remind that assigning to a request a bandwidth lower than the dirtying rate implies that virtual machines memory is transferred, from source to destination data centers, less quickly than it can be dirtied and it will be a problem. Therefore we have to find an algorithm that allow to select a bandwidth greater than dirtying rate for each migration request in order to provide enough bandwidth to complete all migration requests. For this reason, we named our first algorithm 'Min', as it try to provide a bandwidth value that is the lowest possible such that migration can be performed: i.e. it assigns a bandwidth slightly higher than the dirtying rate. In particular, we can calculate the bandwidth as:

$$B = D * (1 + \alpha) \qquad a \in (0, 1) \tag{3.1}$$

where D is the dirtying rate of the virtual machine and α is a positive parameter in (0,1) that allow us to evaluate the bandwidth to allocate to the migration request, such that it is greater that the dirtying rate itself. By definition, α can assume any value greater than 0 and less than 1, so it is important to establish which is the best choice for this parameter. In order to achieve this purpose, we

did some simulations (obviously, selecting Min as the algorithm chosen for the bandwidth calculation) aiming to observe how the blocking probability changes in relation to the arrival rate Ar (in particular we did simulations for different value of arrival rate, i.e. $Ar \in [0,001 - 1]$). We repeat simulations for several values of α, obtaining the following curves:

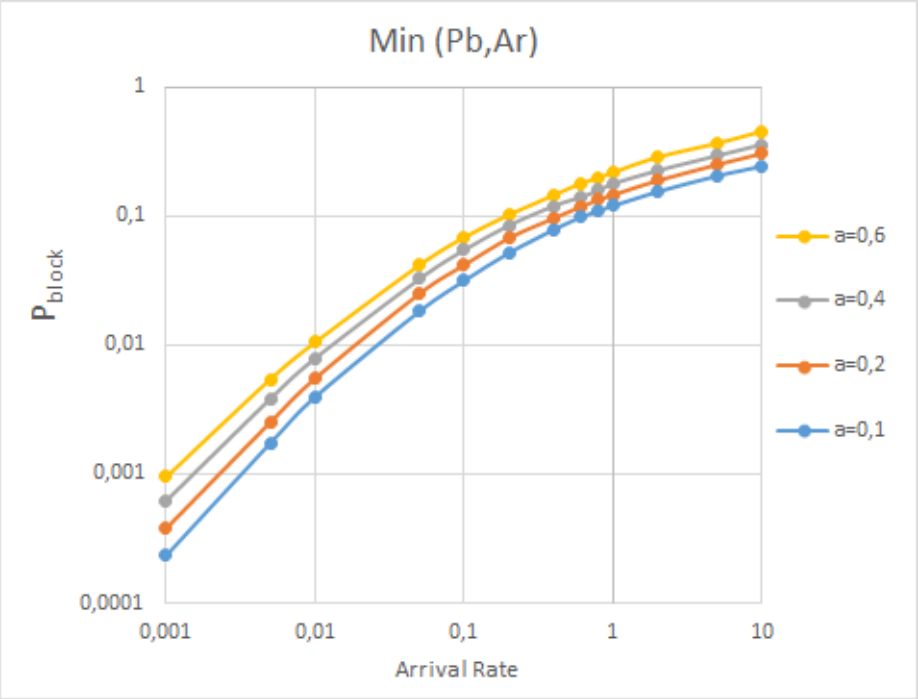

Figure 3.4: Blocking probability curves for algorithm Min

In figure 3.4 we can easily observe that blocking probability increases as an higher value is chosen for the parameter α. This can be explained thinking that having higher values of α means assigning higher bandwidths, then more network resources are used and consequently blocking probability increases. Now the challenge is to choose which value of parameter α allow to provide a bandwidth than provides that migration is performed successfully without requiring too much time to complete it. To address this new problem we did other simulations to monitor how blocking probability behaves by varying the parameter α. As we're mainly interested on the relationship between blocking probability and α we use a fixed

value for the arrival rate for all these simulation, in particular it is equal to 0,1. Plotting results obtained in this way, we have:

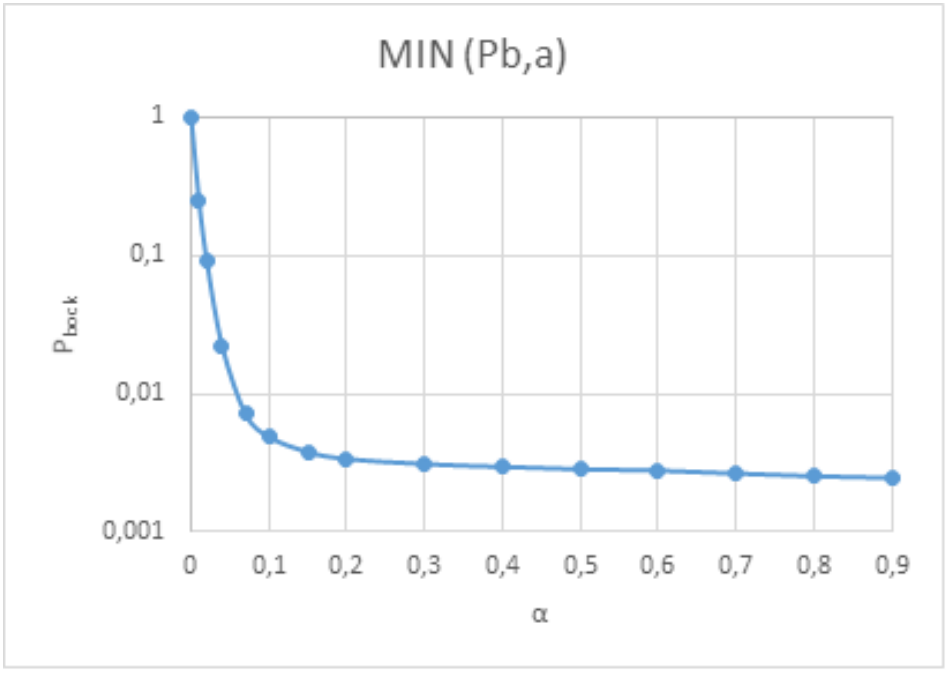

Figure 3.5: Blocking probability curves for algorithm Min with fixed α

In figure 3.5 we immediately note that there exists an inverse proportionality relation between blocking probability P_{block} and the parameter α. It means that blocking probability decreases for growing values of α. But we further observe that for high value of α blocking probability does not decrease in a significant way, so we can select a fixed value for α such that choosing a value greater than it, it will have not considerable benefits in reducing probability. In order to find out which can be the more appropriated value, we decided to adopt an analytical approach looking at the incremental ratio of the curve. We, then, choose the value of α for which the incremental ratio is less than a percentage:

$$\frac{\Delta P_{block}}{\Delta \alpha} \leq \epsilon_1 \tag{3.2}$$

where we have chosen $\epsilon_1 = 0.01$ as it is the most significant value for the considered ratio. In facts, as the blocking probability curve is a decreasing one, also

the incremental ratio decreases. In particular, for higher value of α, it decreases very slowly and it take values that are lower than 0.01 but always greater than 0.001: so the value $\epsilon_1 = 0.01$ can be seen as an upper bound.

In this work, the better choice for the α parameter is α=0,2 as it provides this ratio equal to 0.08.

Until now, we have provided a description of how bandwidth is selected. Now we want to focus on the general procedure of how algorithm Min works, step by step, for each incoming migration request:

Input: Topology network graph $G = (N, E)$, migration request R, dirtying rate D.
Output: The path with the minimum value of available bandwidth. The bandwidth value chosen to provision the migration of the virtual machine.

1. K shortest path calculation between the source s and destination d nodes of the migration request through the Yen algorithm.

2. For each one of the k shortest path the available bandwidth $B_{p,av}$ is evaluated as:

$$B_{p,av} = \min_{e} B_{e,av} \qquad \forall\ e \in L$$

   where L is the set (subset of E) of the links belonging to the considered path.

3. For each one of the k shortest path:

   a. If $B_{p,av} > D$: calculate B in according to the algorithm's formula.

      - If $B_{p,av} \geq B$ the migration request is provisioned with bandwidth B.
      - If $B_{p,av} < B$ check the next path.

   b. If $B_{p,av} \leq D$: the current path is dropped and the next one is analyzed.

4. If, after having analyzed all the k paths, a path with enough bandwidth is not found, migration request is dropped.

5. If a valid path is found, resources are allocated, migration is performed and it takes note of the provided bandwidth value to estimate total migration duration and downtime metrics in order to evaluate performance of the algorithm.

6. Once migration is completed, allocated resources are released.

3.4 RBA Algorithm - Max

As in min we provided a kind of lower bound for the bandwidth to assign to a migration request such that it can be preformed, we now want to inspect how can be estimated an upper bound for the same process. In particular, we want to assign to a migration request a high value of bandwidth, but, at the same time, we have to avoid that it reaches too large values as it can results in a useless waste of resource. So, we can use the function point curves defined in the previous chapter, in order to find a maximum value of bandwidth for each virtual machine type. In formulas we can express the issue of finding this 'maximum' bandwidth as:

$$B = \min(B_{av}, B_{max}) \tag{3.3}$$

In equation 3.3 B_{av} represents the available bandwidth on the selected path while B_{max} is the maximum bandwidth provided by the function points curve of the virtual machine to be migrated. We know that B_{max}, by definition, is the value of bandwidth such that choosing a value greater it will not provides a significant reduction of the total migration duration T_{mig}. So, by choosing the minimum between B_{av} and B_{max} means to allocate a bandwidth that is equal to that available on the path if it is lower than the one indicated by the function point curve or, in the other case, allocate exactly the function point bandwidth.

After having decided the criteria used to assign the bandwidth to an input migration request, we have to specify which are the function points bandwidth values for each one of the virtual machines analyzed in this study. As we consider three value both for the dirtying rate and the virtual machine's memory size, we have to work with nine different types of virtual machines. To find out which can be the best choice for the maximum value of bandwidth that allow to complete

migration in respect of the function point purpose (selecting a bandwidth that guarantees a total migration time not too large without wasting network resources), we adopt an analytical strategy for each virtual machine. More in details, for each virtual machine type, we choose the bandwidth such that the incremental ratio of the function points curve is more ϵ_2 (this method is similar to the one used to choice the better value of α for the algorithm Min):

$$\frac{\Delta T_{mig}}{\Delta B} \geq \epsilon_2 \tag{3.4}$$

As stated in the last equation, we select the first value of bandwidth that provide this ratio greater than ϵ_2 (when considering bandwidth values from the lowest to the greatest). In particular we chosen $\epsilon_2 = 0.001$ as selecting higher value of bandwidth doesn't imply any benefit in total migration time reduction. In the table below, values of B_{max} are reported, for each type of virtual machines considered in this work:

Dirtying rate	*VM size*	B_{max}
50 Mb/s	100 Mb/s	200 Mb/s
50 Mb/s	1 Gb/s	333 Mb/s
50 Mb/s	10 Gb/s	500 Mb/s
100 Mb/s	100 Mb/s	286 Mb/s
100 Mb/s	1 Gb/s	400 Mb/s
100 Mb/s	10 Gb/s	667 Mb/s
500 Mb/s	100 Mb/s	1666 Mb/s
500 Mb/s	1 Gb/s	2000 Mb/s
500 Mb/s	10 Gb/s	2500 Mb/s

Table 3.1: B_{max} *values for different virtual machines*

Now we want to focus on how algorithm Max works, step by step, for each incoming migration request:

Input: Topology network graph $G = (N, E)$, migration request R, dirtying rate D.

Output: The path with the minimum value of available bandwidth. The bandwidth value chosen to provision the migration of the virtual machine.

1. K shortest path calculation between the source s and destination d nodes of the migration request through the Yen algorithm.
2. For each one of the k shortest path the available bandwidth $B_{p,av}$ is evaluated as:
$$B_{p,av} = \min_{e} B_{e,av} \qquad \forall\ e \in L$$
where L is the set (subset of E) of the links belonging to the considered path.
3. For each one of the k shortest path:
   a. If $B_{p,av} > D$: calculate B in according to the algorithm's formula.
      - If $B_{p,av} \geq B$ the migration request is provisioned with bandwidth B.
      - If $B_{p,av} < B$ check the next path.
   b. If $B_{p,av} \leq D$: the current path is dropped and the next one is analyzed.
4. If, after having analyzed all the k paths, a path with enough bandwidth is not found, migration request is dropped.
5. If a valid path is found, resources are allocated, migration is performed and it takes note of the provided bandwidth value to estimate total migration duration and downtime metrics in order to evaluate performance of the algorithm.
6. Once migration is completed, allocated resources are released.

3.5 RBA Algorithm - Congestion Aware

As shown in previous paragraphs, in algorithms Min and Max the value of bandwidth to assign to a migration is chosen in a static way, such that, once the path is selected, then bandwidth is given by a formula ((2.1) for Min and (2.3) for Max)

that not involves parameters that can tell us something about the network congestion level. So, with this third algorithm, Congestion-Aware, we want to take into account the network state while provisioning bandwidth for the migration. As migrations are performed, resources are allocated: a portion of bandwidth over a link is reserved and it cannot be used for other requests until it is released; moreover more than one link are used to route each request from its source data center to its destination data center. Relying on these considerations, we can say that network resources usage can be represented through two parameters: the assigned bandwidth for the migration (B_{av}) and the number of hops belonging to the path used to route the migration request (H). For this reason, in this algorithm the routing phase is slightly different from the one seen in Min and Max, as we want to select the less loaded path, and not only the shortest one as the route along which perform migration. For the bandwidth assignment problem, however, we decided to assign a bandwidth value equal to the one suggested by the function points curve of the considered virtual machine. Below there is the a detailed overview of all the step of the algorithm Congestion-Aware (for each incoming migration request):

Input: Topology network graph $G = (N, E)$, migration request R, dirtying rate D.
Output: The path with the minimum value of available bandwidth. The bandwidth value chosen to provision the migration of the virtual machine.

1. K shortest path calculation between the source s and destination d nodes of the migration request through the Yen algorithm.

2. For each one of the k shortest path two parameters are evaluated: the available bandwidth $B_{p,av}$ and the number of hops H belonging to the path (it is the path length):

$$B_{p,av} = \min_{e} B_{e,av} \qquad \forall\ e \in L$$

   where L is the set (subset of E) of the links belonging to the considered path.

3. For each path the following product is computed:

$$B_{p,av} \cdot H$$

4. Paths are sorted in decreasing order of this product.

5. The path with the minimum value of the product $B_{av} \cdot H$ is selected, then:

   a. if $B_{p,av} > D$: a bandwidth value B equal to the one suggested by the function points curve is assigned to the migration request.

      - If $B_{p,av} \geq B$ the migration request is provisioned with bandwidth B.
      - If $B_{p,av} < B$ check the next path.

   b. If $B_{p,av} \leq D$: the current path is dropped and the next one is analyzed.

6. If, after having analyzed all the k paths, a path with enough bandwidth is not found, migration request is dropped.

7. If a valid path is found, resources are allocated, migration is performed and it takes note of the provided bandwidth value to estimate total migration duration and downtime metrics in order to evaluate performance of the algorithm.

8. Once migration is completed, allocated resources are released.

3.6 RBA Algorithm - Range

As will be explained in the next chapter, analyzing the results obtained from the simulations for the algorithms Min and Max we found that the number of hop of the path, on which the migration request is routed, can have great impact on resources occupation. But, influencing resources occupation means that path's length (measured in hops) can be a key factor also in evaluating blocking probability. In fact, if the chosen path presents a high number of hops, it means that on network's links the load is high and free bandwidth is low, therefore, in order

to serve a request you must assign less bandwidth. Hence, we tried to develop an algorithm which can be able to provide bandwidth in according to the path's length. So the general idea of this algorithm, called Range, is to build a kind of correspondence between the number of hops of the selected path and the bandwidth required to complete a virtual machine migration. Let's, then, introduce the equation that allows us to achieve this goal:

$$B = B_{MAX} - (B_{MAX} - B_{min}) * \frac{h - H_{min}}{H_{MAX} - Hmin} \tag{3.5}$$

It is easy to recognize that 3.5 is a straight-line equation where h is the independent variable and it takes values in the range (H_{min}, H_{max}) while the dependent variable B represents the provisioned bandwidth for migration and it takes values in the range (B_{min}, B_{max}). More in details, the involved parameters are the following:

- h: it is number of hops of the current path we're analyzing;
- H_{min}: minimum value of hop number, it is equal to the length of the first shortest path;
- H_{max}: maximum value of hop number, it is equal to the topology network diameter;
- B_{min}: minimum value of bandwidth required to perform migration, it is calculated through the same formula used in algorithm Min;
- B_{max}: maximum bandwidth that can be allocated to perform a migration, it is given by the function point value of the considered virtual machine.

As in this study we are analyzing migration of several types of virtual machines, we have to define these parameters presented above, for each one of the considered dirtying rate and for each one of the VM memory size. So we have to specify the bounds of intervals in which the variables may take values: we show them in the following table:

Dirtying rate	*VM memory size*	H_{min}	H_{max}	B_{min}	B_{max}
50 Mb/s	100 MB	2	8	60 Mb/s	500 Mb/s
50 Mb/s	1 GB	2	8	60 Mb/s	500 Mb/s
50 Mb/s	10 GB	2	8	60 Mb/s	500 Mb/s
100 Mb/s	100 MB	2	8	120 Mb/s	667 Mb/s
100 Mb/s	1 GB	2	8	120 Mb/s	667 Mb/s
100 Mb/s	10 GB	2	8	120 Mb/s	667 Mb/s
500 Mb/s	100 MB	2	8	600 Mb/s	2500 Mb/s
500 Mb/s	1 GB	2	8	600 Mb/s	2500 Mb/s
500 Mb/s	10 GB	2	8	600 Mb/s	2500 Mb/s

Table 3.2: *Used parameters for algorithm Range*

In table 3.2, we used the following equation to evaluate the value of H_{max}:

$$d = 2 * (\sqrt{n} - 1) \tag{3.6}$$

For a graph with n nodes and m edges, the diameter is the length of the longest shortest-path between any two vertices. There exist two main variations on the diameter, depending on the metric used to construct and evaluate the shortest paths. The hop-diameter is the length of the longest shortest-path between any two vertices, where the shortest paths are computed and evaluated using hop count as the metric, while in the length-diameter, the shortest paths are computed and evaluated using Euclidean length (i.e. physical path length) as the metric. We consider the hop-diameter in our formulation. In equation 3.6 d represents the network topology diameter while n is the number of nodes in the network. As we are considering the NSFNet with 24 nodes, from the calculations the network diameter d results to be 7.79, but it should be rounded up so we consider it equal to 8.

Now we want to focus on how algorithm Range works, step by step, for each incoming migration request:

Input: Topology network graph $G = (N, E)$, migration request R, dirtying

rate D.
Output: The path with the minimum value of available bandwidth. The bandwidth value chosen to provision the migration of the virtual machine.

1. K shortest path calculation between the source s and destination d nodes of the migration request through the Yen algorithm.

2. For each one of the k shortest path two parameters are evaluated: the available bandwidth $B_{p,av}$ and the number of hops h belonging to the path (it is the path length):
$$B_{p,av} = \min_{e} B_{e,av} \qquad \forall\ e \in L$$
where L is the set (subset of E) of the links belonging to the considered path.

3. For each one of the k shortest path:

   a. if $B_{p,av} > D$: a bandwidth value B is calculated according to equation 3.5:

      - If $B_{p,av} \geq B$ the migration request is provisioned with bandwidth B.
      - If $B_{p,av} < B$ check the next path.

   b. If $B_{p,av} \leq D$: the current path is dropped and the next one is analyzed.

4. If, after having analyzed all the k paths, a path with enough bandwidth is not found, migration request is dropped.

5. If a valid path is found, resources are allocated, migration is performed and it takes note of the provided bandwidth value to estimate total migration duration and downtime metrics in order to evaluate performance of the algorithm.

6. Once migration is completed, allocated resources are released.

In the following graphs we plotted the behavior of equation 3.5 for the three values of dirtying rate:

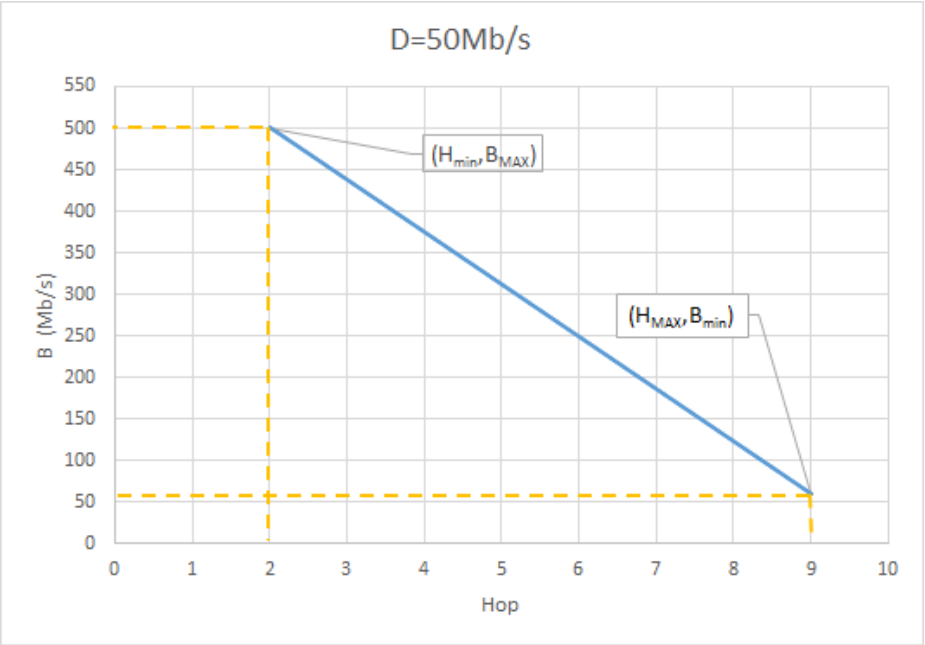

Figure 3.6: Range algorithm's scheme for dirtying rate equal to 50Mb/s

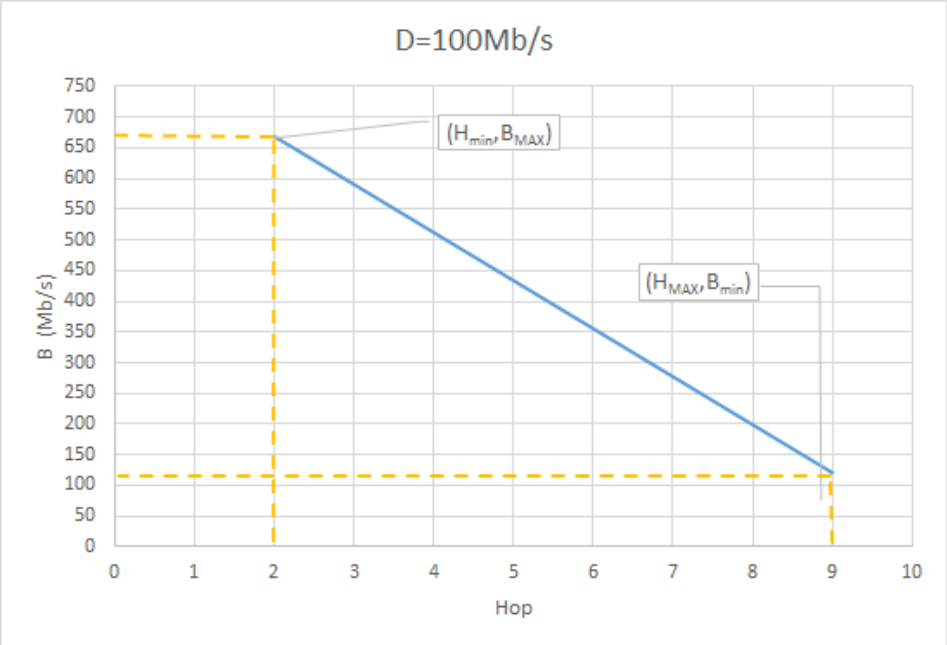

Figure 3.7: Range algorithm's scheme for dirtying rate equal to 100Mb/s

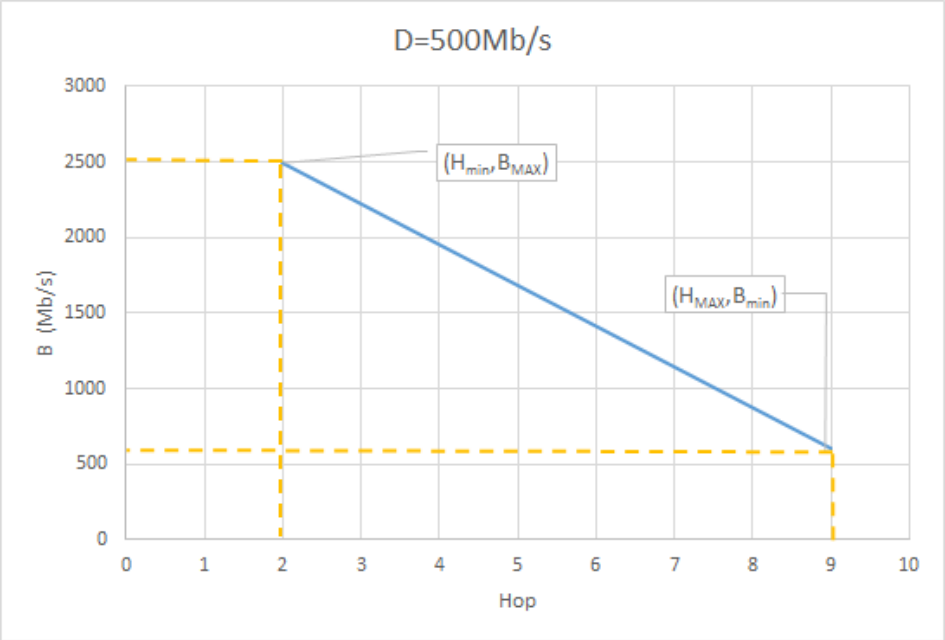

Figure 3.8: Range algorithm's scheme for dirtying rate equal to 500Mb/s

Chapter 4

Simulative Results

To evaluate the performance of the proposed Routing and Bandwidth Assignment algorithms, we have developed a C++ discrete-events simulator, which is able to simulate the behavior of an Wide Area Network with a distributed Cloud system on top of it. Connections are originated from a set of data centers, which represent the source and destination nodes of each request. Offered traffic is uniformly distributed from all the couples of data centers; in particular it consists of connection requests with Poissonian inter-arrival time, which arrive with frequency (arrival rate) λ. Our goal is to evaluate the performances of some algorithms for the Routing and Bandwidth Assignment problem for a virtual machines migration scenario, so there isn't an a priori fixed value for the connection duration as each connection request represent a migration request. Then, the duration of each request depends of all the parameters involved in the migration process and it can't be known in advance. Another consequence of solving the Routing and Bandwidth Allocation problem is that we can't adopt the typical wavelength-based approach of a wavelength-division-multiplexing optical network. In facts, as the problem expected to choose the best value of bandwidth to provision to each migration, we can't consider use channels of fixed bandwidth. Consequently, we only consider each link provisioned with a maximum valued of capacity and we then, add a constraint which assures that the overall sum of assigned bandwidths on a link not exceeds the link capacity.

In this chapter, we first show some simulation settings, then we pass to illus-

trate results obtained from simulations for each one of the previously proposed algorithms.

4.1 Simulation Settings

Before showing the graphs relating to the results, we have to explain which are the metrics and the parameters used to compare of our algorithms. In particular we can first present the input parameters used in the simulator:

- *Arrival rate*: it is the arrival frequency of migration requests and it takes values in (0,001 – 10) because migration requests are not so frequent as user's connections requiring internet services; number of migrations is not very large but each migration carry a great amount of data. We refer to this parameter as A_r. It is an input parameter of this simulator, hence we did many simulations for several different values of arrival rate (typically it compares in the x–axis of graphs).

- *Dirtying rate*: is the rate at which memory pages are modified while the virtual machine (and the application hosted on it) is till running. We refer to this metric with D in all the graphs of this work.

- *VM size*: it is the size of the virtual machine memory, i.e. the amount of data that needs to be transferred in the first round of the iterative pre-copy phase. We refer to this metric with V^M in all the graphs of this work.

Now we list all the metrics used to evaluate the performances of the just presented Routing and Bandwidth Assignment algorithms:

- *Blocking probability*: it is the metric used to evaluate the performance of algorithms and it is measured as the ratio between provisioned byte (i.e. amount of bytes transferred by migration requests successfully provisioned) and total offered bytes (i.e. amount of data of both provisioned and blocked migration requests, in byte). We refer to this metric with P_{block} in all the graphs of this work.

- *Resource Occupation*: it measures the amount of network resources allocated to provisioned migration requests and it is measured in [byte*hop]; we refer to resource occupation as RO in graphs.

- *Carried Load*: it represents the level of network resources utilization and it is the ratio between allocated resources for provisioned requests and the total resources of the network; obviously it is a pure number, we refer to carried load as CL.

- *Total Migration Time*: it is the average total time required to migrate a virtual machine; it is used to evaluate the resource occupation metric and it is measure in seconds; we refer to migration duration as T_{mig}.

- *Bandwidth*: it is the average bandwidth assigned by an algorithm to migrate a virtual machine; it is used to evaluate the resource occupation metric and it is measure in [Mb/s]; we refer to migration duration as B.

- *Hop* it is the average number of hops belonging the a path over which a virtual machine is migrated; it is used to evaluate the resource occupation metric and it is measured in [hop]; we refer to migration duration as H.

The network topology adopted in the simulator is shown in figure 5.1. This is the network topology called USA24 and it consists of 24 nodes and 43 bidirectional links. Each link has a capacity equal to 100 Gb/s. In the network were placed eight data centers, which constitute the only possible sources and destinations for each migration request.

In this study are considered several types of virtual machines: they differ in dirtying rate D and memory size V^M. In particular V^M can assume three values: 100 MB, 1GB and 10GB as in [2], [3], [17] and [8]. Moreover, each migration request is generated with an uniform distribution of memory size.

Another important parameter need a better description from a numerical point of view: the dirtying rate. In literature, usually, you can find the page dirty rate (also named dirty frequency) which represents the rate at which memory pages are dirtied (modified) during the iterative pre-copy phase of the live migration process. As we know that memory page size is equal to 4 kB, we can easily found

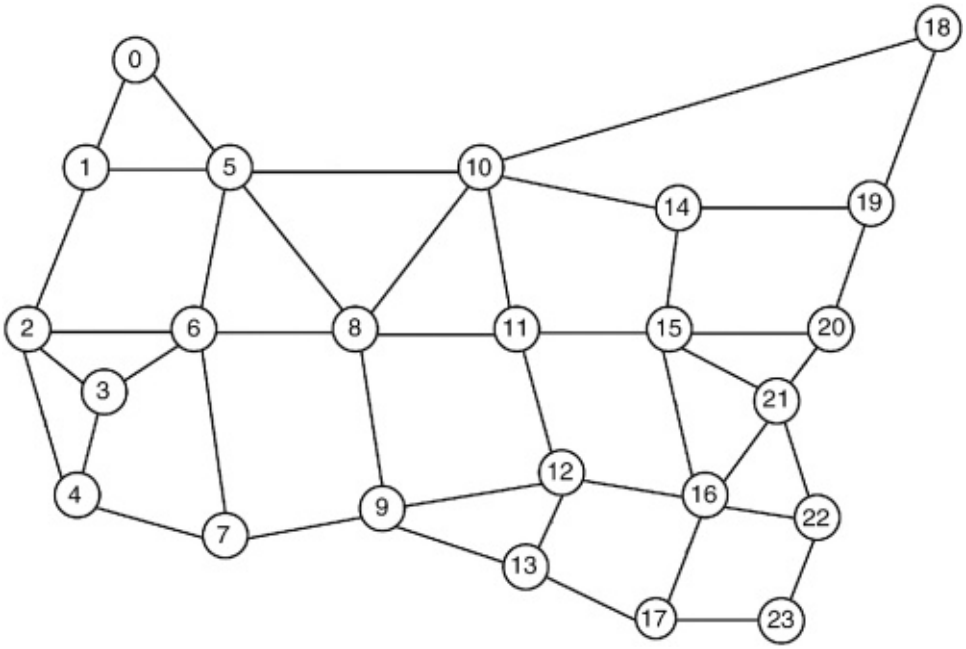

Figure 4.1: Network topology USANet24

the correspondent values for the dirtying rate expressed in [Mb/s]. In the following table some typical values are reported.

In our simulations we assumed three possible values for the dirtying rate: 50 Mb/s, 100 Mb/s and 500 Mb/s.

4.2 Illustrative Numerical Results

The first algorithms we have analyzed are Min and Max. We did some simulations for several different values of arrival rate in order to evaluate the blocking probability for both algorithms. The following pictures report the blocking probability curves for both algorithms Min and Max. In particular:

- figure 4.2 shows blocking probability curves for dirtying rate equal to 50 Mb/s;
- figure 4.3 shows blocking probability curves for dirtying rate equal to 100 Mb/s;

Dirty Frequency	*Page Size*	*DirtyingRate*
500 pages/s		16 Mb/s
1000 pages/s		32 Mb/s
2000 pages/s		64 Mb/s
3000 pages/s	4 kB	96 Mb/s
4000 pages/s		128 Mb/s
5000 pages/s		160 Mb/s
10000 pages/s		320 Mb/s

Table 4.1: *Dirtying rates values for different dirty frequencies*

- figure 4.4 shows blocking probability curves for dirtying rate equal to 500 Mb/s.

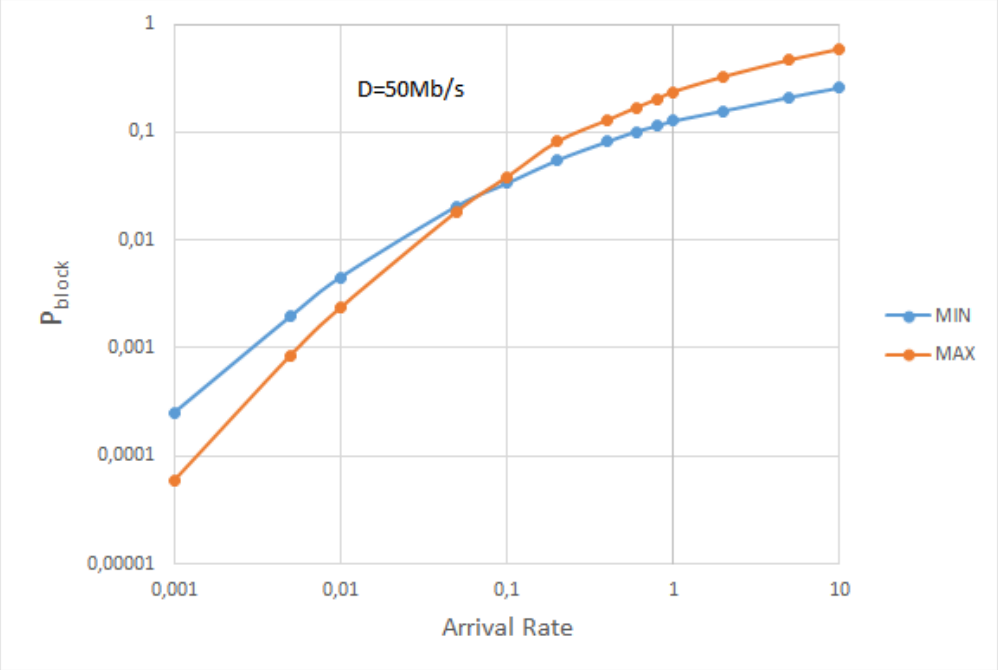

Figure 4.2: Blocking Probability for algorithms Min and Max for D=50Mb/s

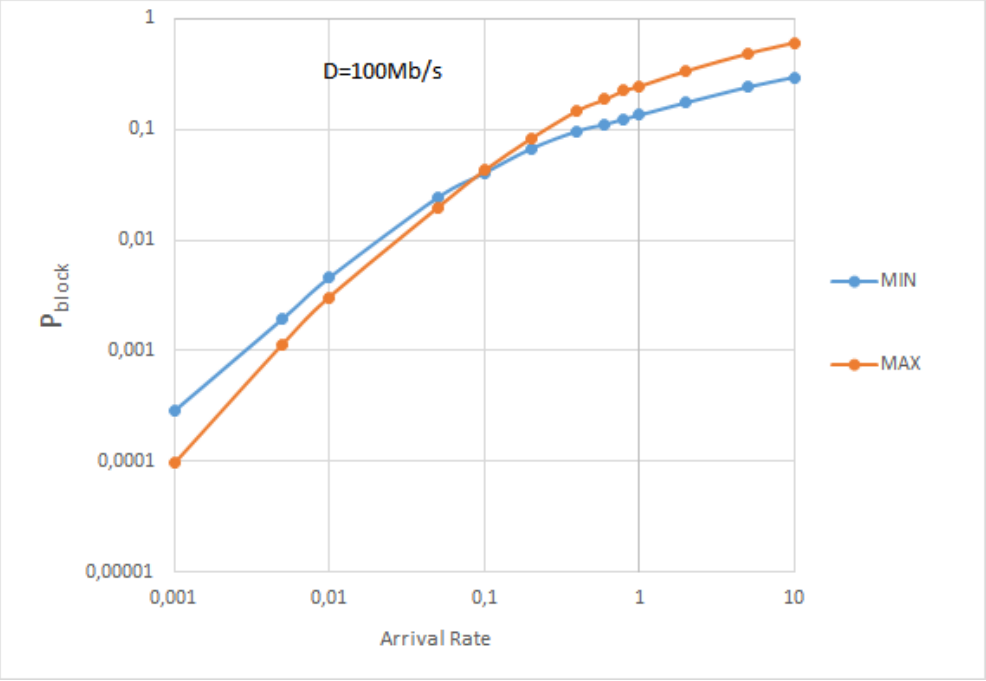

Figure 4.3: Blocking Probability for algorithms Min and Max for D=100Mb/s

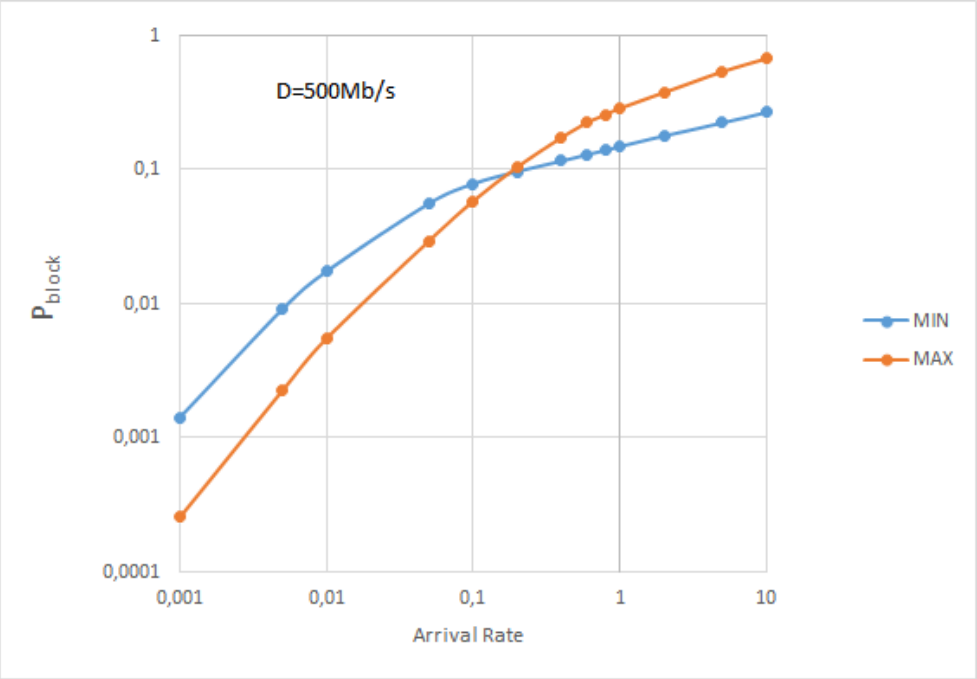

Figure 4.4: Blocking Probability for algorithms Min and Max for D=500Mb/s

From a first look at the previous graphs, you can immediately notice that there are very low values of blocking probability for very small Arrival Rate (less than unity), while for Arrival Rate of some order of magnitude higher (1-10) blocking probability strongly grows (it arrives almost to the total block). Also from the results obtained, we note the existence of an intersection point between the two curves of probability, which can be attributed to a change in behavior of the two algorithms depending on the assumed value by the arrival rate. In particular we observe that Min behaves better than Max for high arrival rate while for arrival rate very low, less than unity, we have the opposite situation (namely Max shows blocking probability values lower than those of Min) So, for high values of arrival rate ($Ar > 1$) we see that Min behaves better than Max (i.e. Min presents lower blocking probability than Max): this behavior can be understood if we think that an incoming connection is blocked if it doesn't find sufficient resources (e.g. if there is not enough bandwidth available on any of the k shortest path previously calculated). Therefore, if more migration requests are served simultaneously, the probability that the resources to allocate for a possible incoming connection are few can be very high: thus, an algorithm which assigns high values of bandwidth (such as Max) leads to a higher blocking probability, compared to an algorithm that allocates lower values of bandwidth.

However, For small values of the arrival rate ($Ar < 1$), the most influential factor in determining blocking probability is no longer the arrival rate because, now, the arrival time instants of the migration request are not so close in time. Nevertheless you can easily notice that the scarcity of available resources needed to route a new connection may depend on the time taken to transfer the memory of the virtual machine. Migration requests of different virtual machines may require different total migration times, therefore resources can be employed for different time periods, whose overlap can determine the absence of available resources for an arriving migration request. So it can happen that, when a new migration request arrives, there are no resources available because already assigned to the transfer of other virtual machines, whose migration requests were accepted earlier. Therefore, an algorithm which assigns low values of bandwidth (such as Min), implies high values of total migration time (T_{mig}) and, consequently, higher blocking probability compared to an algorithm that assigns upper values of bandwidth (as Max).

Moreover, in each graph you observe that the intersection point between the two curves moves to the right for increasing values of dirtying rate. In order to understand the reason of this behavior, we decided to analyze another parameter: the Resource Occupation, calculated as:

$$RO = B \cdot T_{mig} \cdot H \tag{4.1}$$

In equation 4.1 B is the average provisioned bandwidth for the migration, T_{mig} is the average total migration time and H is the average number of hops (i.e. the average path's length on which migrations are performed). Below we report the trend of Resource Occupation in function of the arrival rate for the two algorithms Min and Max for each value of dirtying rate. Besides, are also reported the trends of the components of the Resource Occupation such as: average total migration time, average provisioned bandwidth and average path's length (for each valued of dirtying rate). You immediately note that the curves of Resource Occupation of the two algorithms follow the same trend as those of blocking probability: namely, for high arrival rates Min behaves better than Max and viceversa for low arrival rates. You can observe that B and T_{mig} have a slightly constant trend by varying the arrival rate, due to the fact that they are averages values for provisioned bandwidth and total migration time. But H shows a very interesting behavior: it results to have an increasing trend for both algorithms, however Max grows faster than Min. Indeed, as Max assigns higher bandwidth values than Min, in Min links capacity may expire in shorter times than in Max and so, as a consequence, Max shows an average path's length higher than Min.

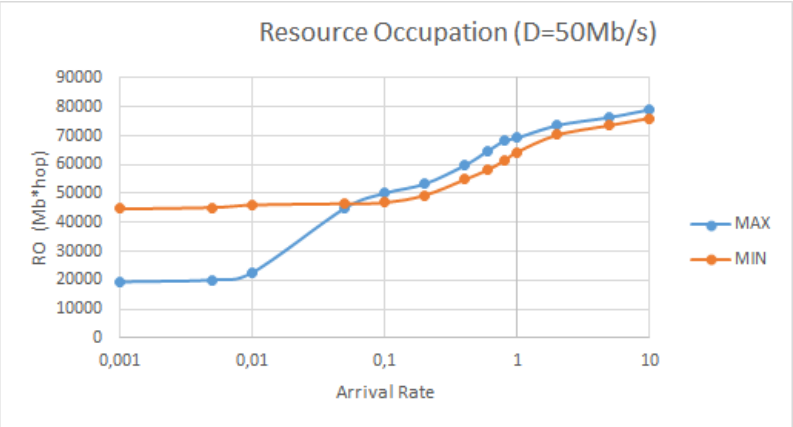

Figure 4.5: Resource Occupation for Min and Max for D=50Mb/s

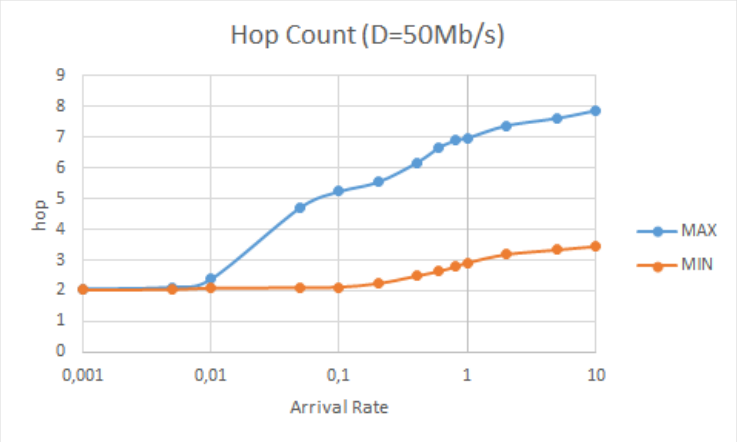

Figure 4.6: Hop Count for Min and Max for D=50Mb/s

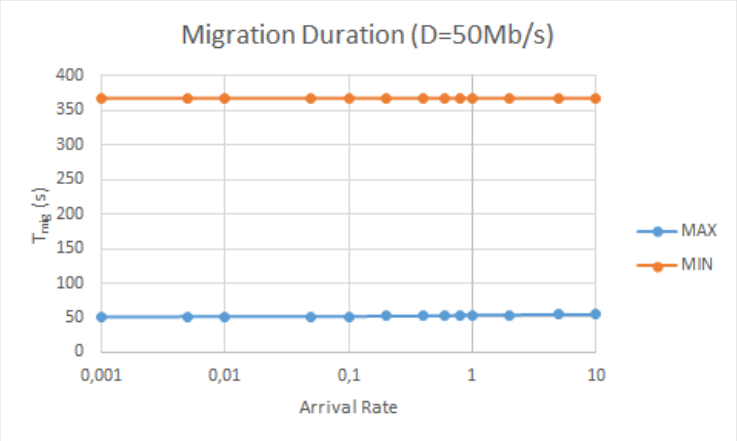

Figure 4.7: Migration Duration for Min and Max for D=50Mb/s

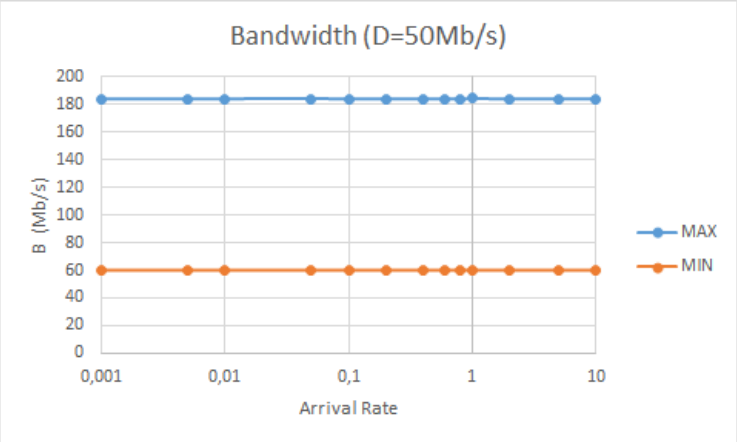

Figure 4.8: Provisioned bandwidth for Min and Max for D=50Mb/s

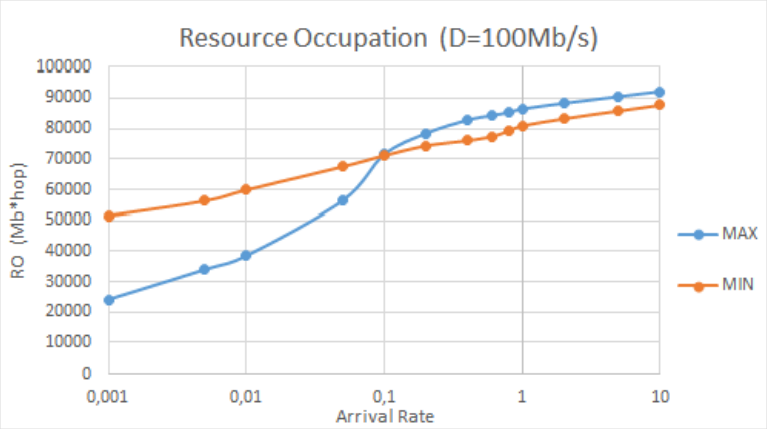

Figure 4.9: Resource Occupation for Min and Max for D=100Mb/s

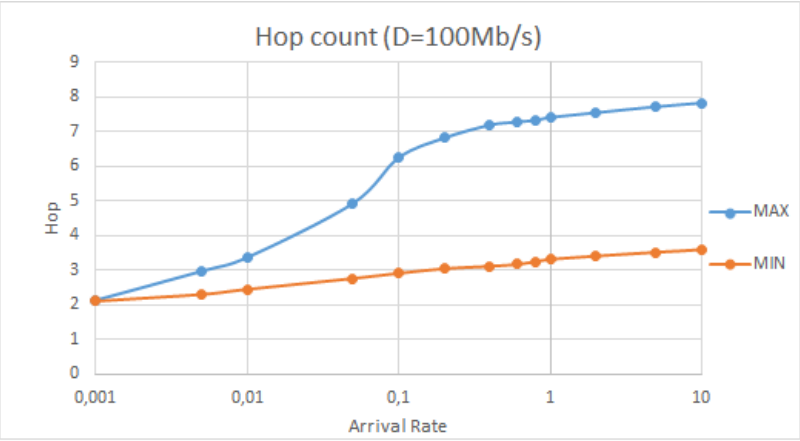

Figure 4.10: Hop Count for algorithms Min and Max for D=100Mb/s

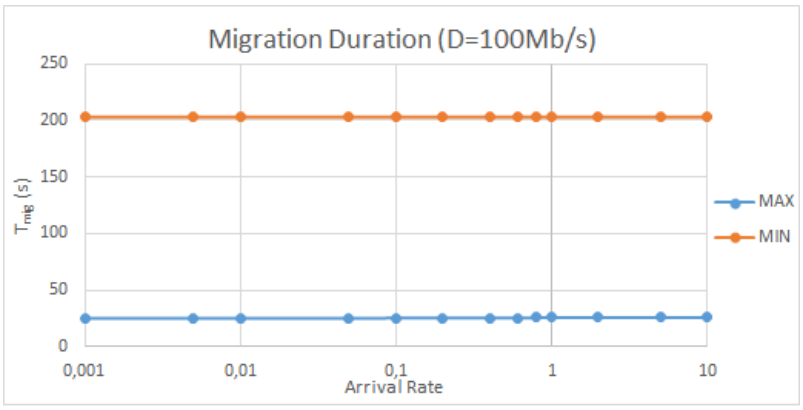

Figure 4.11: Migration Duration for Min and Max for D=100Mb/s

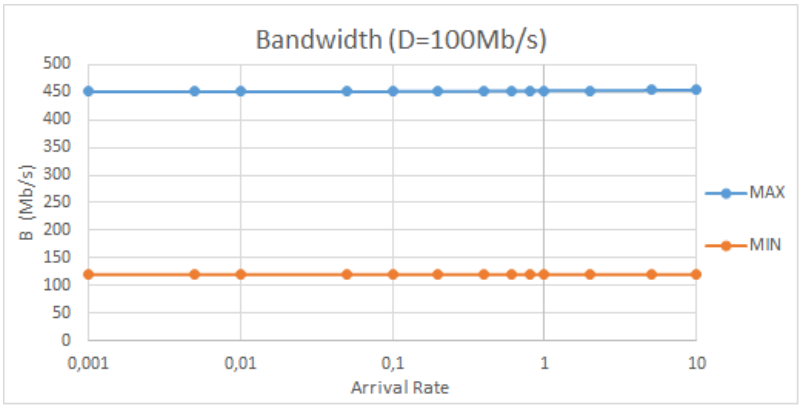

Figure 4.12: Provisioned Bandwidth for Min and Max for D=100Mb/s

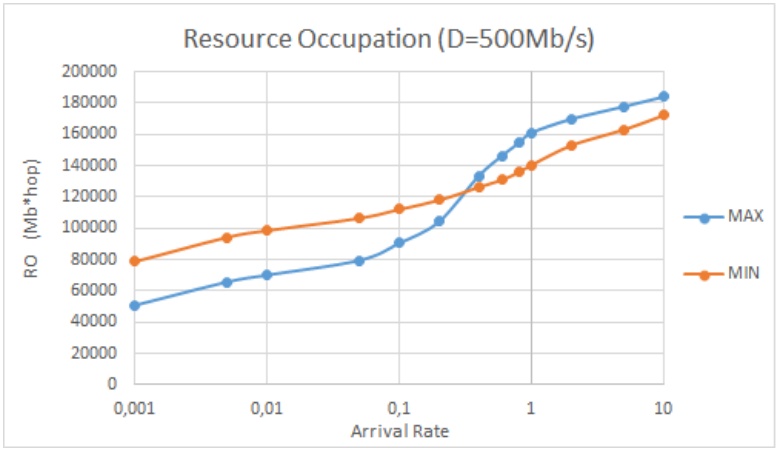

Figure 4.13: Resource Occupation for algorithms Min and Max for D=500Mb/s

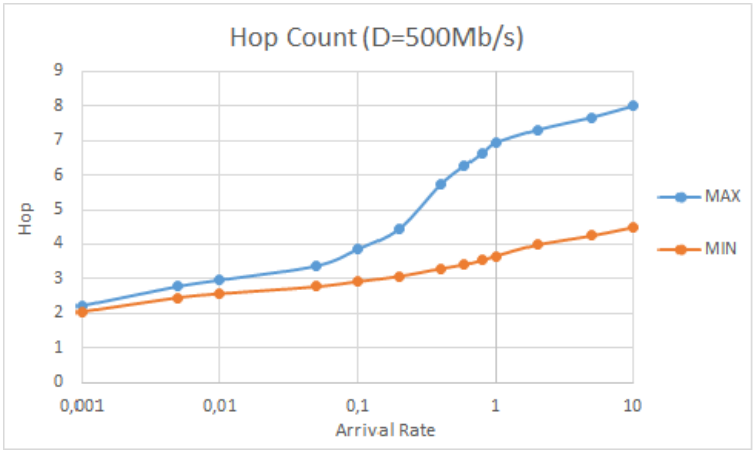

Figure 4.14: Hop Count for Min and Max for D=500Mb/s

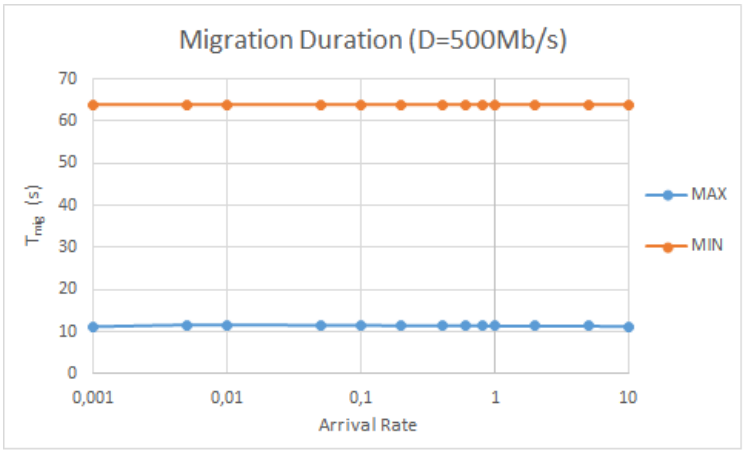

Figure 4.15: Migration Duration for Min and Max for D=500Mb/s

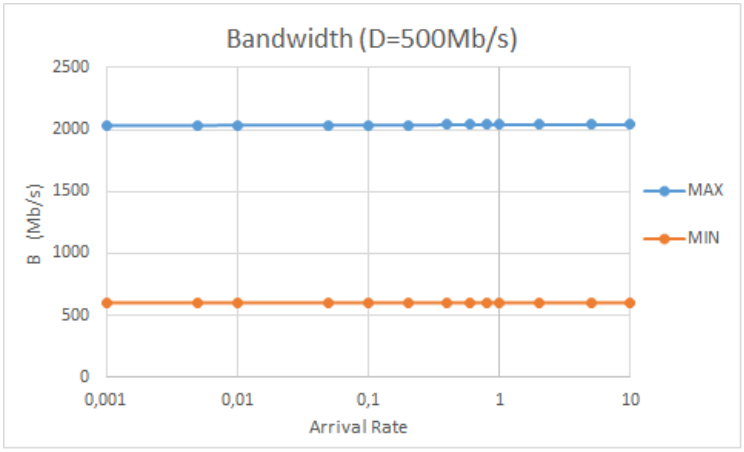

Figure 4.16: Provisioned Bandwidth for algorithms Min and Max for D=500Mb/s

Looking at the previous graphs, we can say that Resource Occupation presents two min factors: the product $T_{mig} \cdot H$ which influences its amplitude and the H parameter which determines its trend (as we have already said, for high arrival rate in Max path's length grows faster than in Min resulting in larger blocking probability while for low arrival rate there is the opposite situation).

Until now, all the presented graphs in this chapter show the arrival rate (A_r) on the x-axis, both for blocking probability and resource occupation. But arrival rate is an input parameter in the discrete-events based simulator, so it is not an adaptive metric. In other words, it can not say anything about the actual behavior of the network, nor the offered load or the served traffic. For this reason, in order to better understand the real behavior of the network, another metric is evaluated: the Carried Load. It represents the amount of served traffic in relation to the offered one. In particular, it is evaluated as shown in the formula below.

$$C_L = \frac{RO \cdot A_r}{H_{tot} \cdot C} \tag{4.2}$$

As you can note from equation 4.2, the Carried Load is a pure number both numerator and denominator are expressed in $[hop \cdot \frac{bit}{s}]$: the former (product between resource occupation RO and arrival rate Ar) represents the total amount of resources allocate for the provisioned migration requests while the latter (product between the total number of hop in the network H_{tot} and the capacity of each link C) represents the total amount of resources available on the network. So this ratio evaluates the relative amount of resources used for migration provisioning. In the following figures are reported the blocking probability curves of the two algorithms Min and Max, for each one of the considered values of dirtying rate.

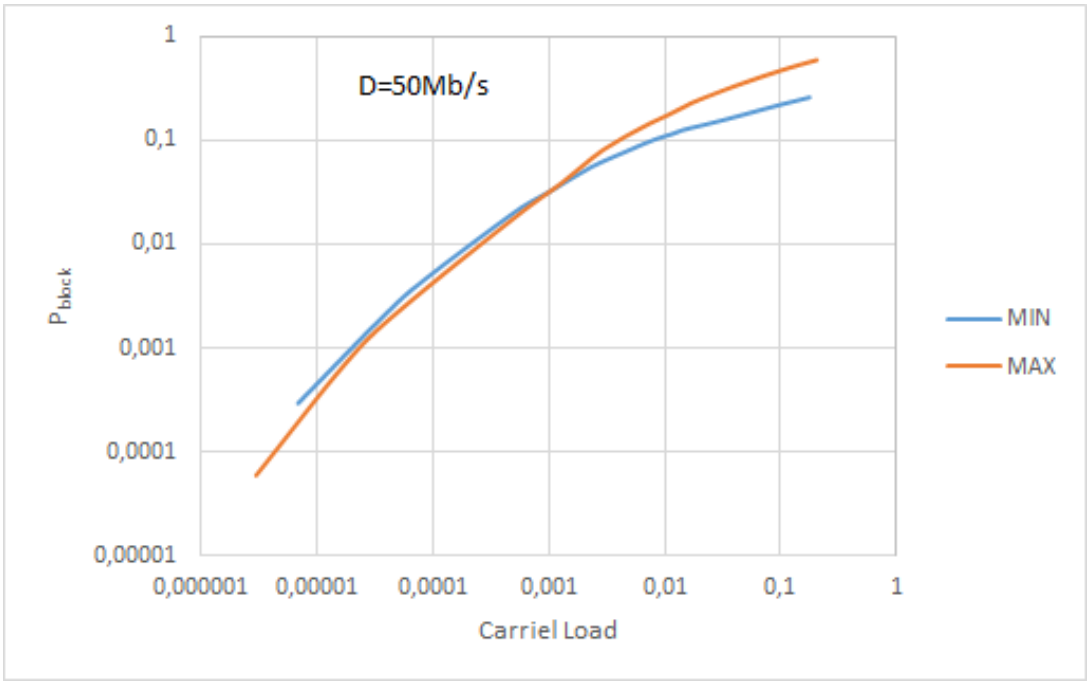

Figure 4.17: Blocking Probability - Carried Load for D=50Mb/s

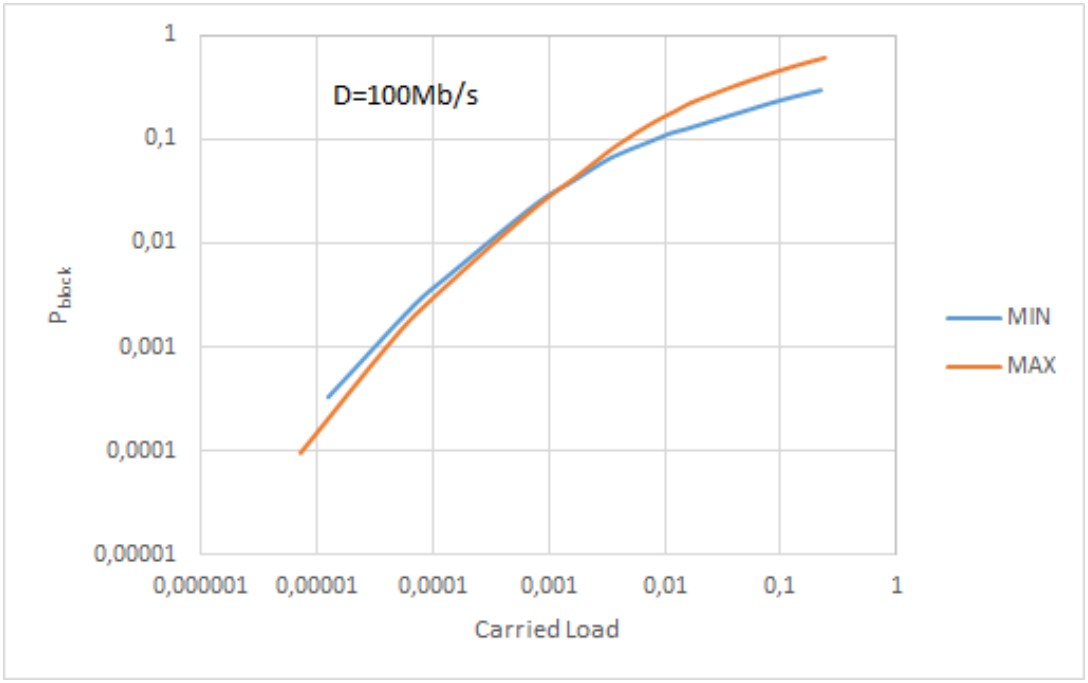

Figure 4.18: Blocking Probability - Carried Load for D=100Mb/s

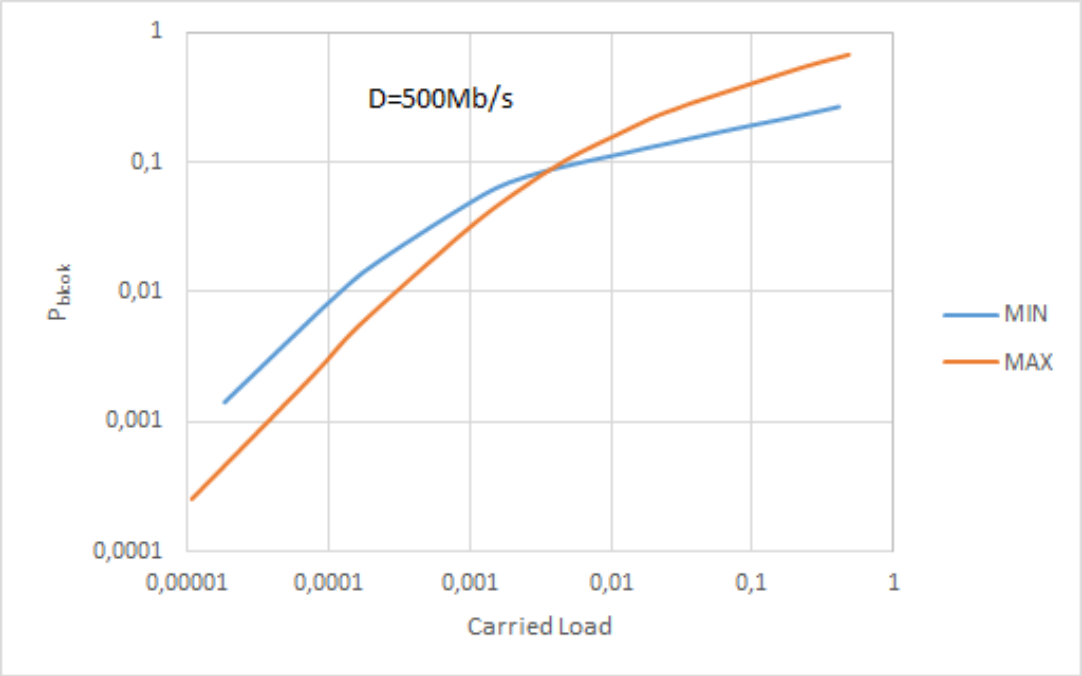

Figure 4.19: Blocking Probability - Carried Load for D=500Mb/s

Analyzing the results obtained from the simulations performed so far, we find out a change in behavior for both the considered algorithms. Furthermore, Min and Max assign bandwidth in a static way: by means of a simple formula or of a priori fixed values. Therefore we decided to introduce a third algorithm that chooses the bandwidth to be allocated to migration requests in a dynamic way, i.e., taking into account the actual network state. It is the Congestion-Aware algorithm, already described in the previous chapter. So now we want to present and analyze results obtained from its simulations.

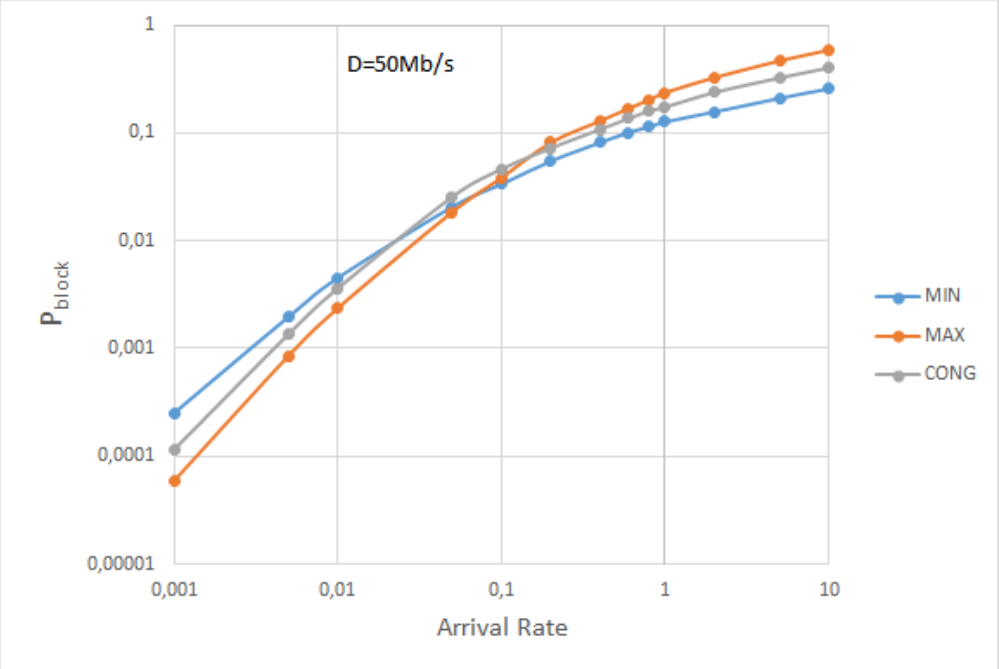

Figure 4.20: Blocking Probability for algorithm Cong for D=50Mb/s

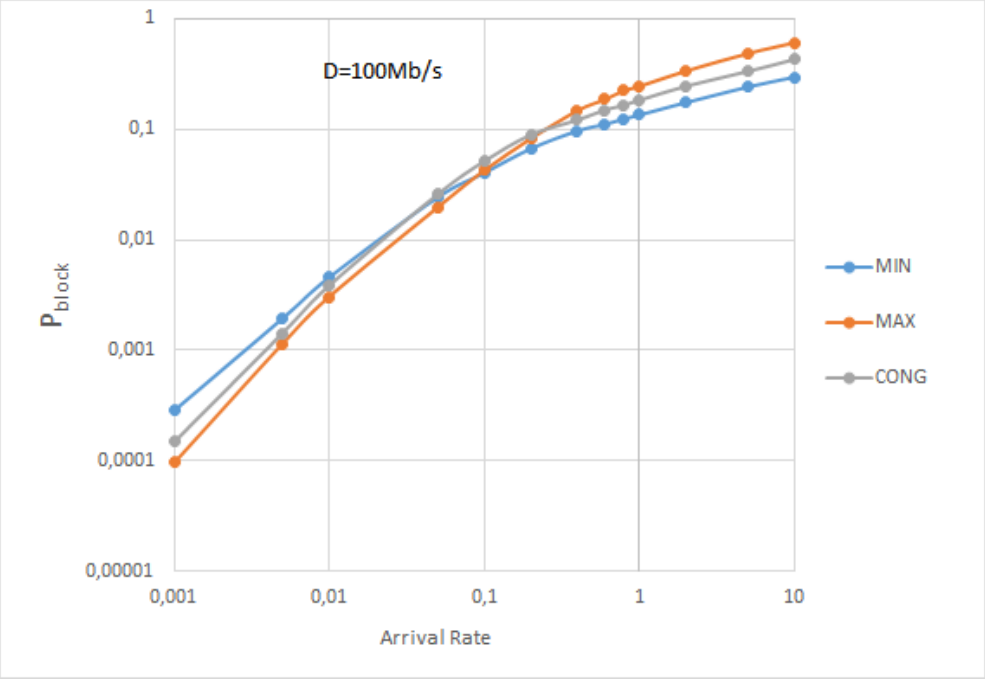

Figure 4.21: Blocking Probability for algorithm Cong for D=100Mb/s

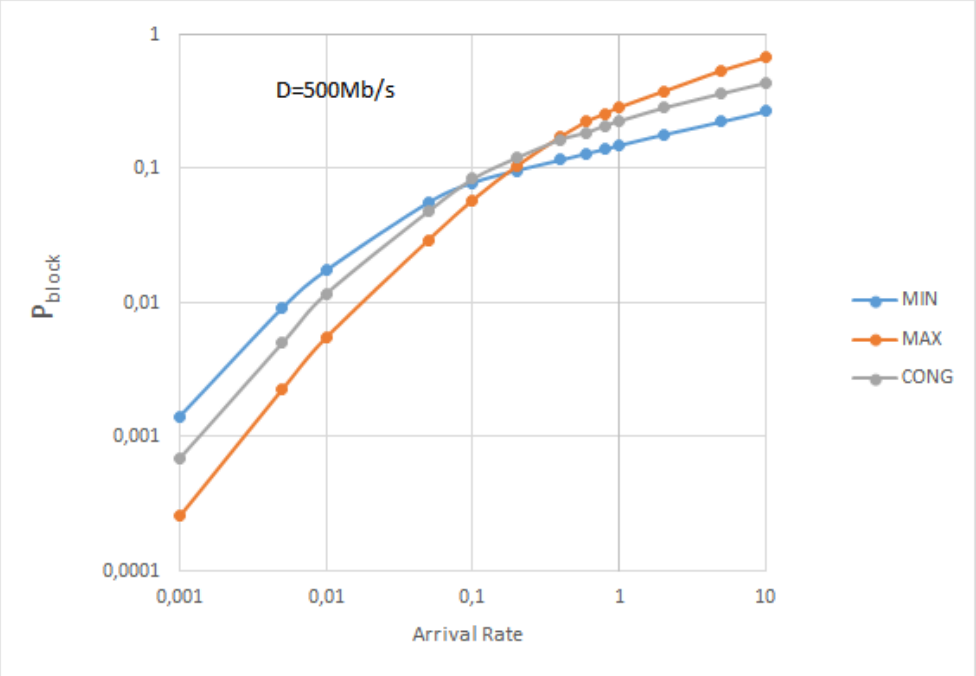

Figure 4.22: Blocking Probability for algorithm Cong for D=500Mb/s

Figures 4.20, 4.21 and 4.22 show the blocking probability for all the three algorithms Min, Max and Congestion-Aware for each value of dirtying rate. You can observe that Congestion-Aware presents a similar trend to other algorithms, but in particular its blocking probability curve is comprised between those of Min and Max. Despite being an adaptive algorithm, we can see how the blocking probability has a trend similar to that of the algorithms that assign the band in a static manner. In facts, Congestion-Aware presents a double behavior depending on the number of incoming migration requests. For high arrival rates it is better than Max but worse than Min while for low arrival rates it is better than Min but worse than Max.

Given the progress achieved, we are now trying to think of an algorithm that is able to grasp this double behavior: the trend of Max for small arrival rate and the one of Min for high arrival rate. However, it is not easy, as in the first case the most influential factor in determining blocking probability is the total dura-

tion of the migration while in the second case it is the number of simultaneously served migration requests. Therefore we want to find a common factor that affects both algorithms to emulate (Max for lower arrival rates and Min for higher arrival rates). So we can take inspiration from the study previously conducted on the Resource Occupation. We know that it is evaluated as the product of three factors: (average) total migration time, (average) provisioned bandwidth and (average) path's length. In particular we note that the latter is a very influential factor in both algorithms in order to determine the trend of the curve of the Resource Occupation. Then we can choose to use the number of hops as a parameter according to which assign the provisioned bandwidth for migration. But, influencing resources occupation means that path's length (measured in hops) can be a key factor also in evaluating blocking probability. In fact, if the chosen path presents a high number of hops, it means that on network's links the load is high and free bandwidth is low, therefore, in order to serve a request you must assign less bandwidth. So the general idea of the fourth algorithm, called Range, is to build a kind of correspondence between the number of hop of the selected path (h) and the bandwidth required to complete a virtual machine migration (B). In particular this relationship is given by a straight-line equation in which the independent variable represents the (average) number of hops belonging to the selected path for the migration. In this way the algorithm is able to provide a bandwidth to each migration request having a path's length h. In according to our consideration we expect that for higher values of h lower bandwidth are assigned while, in the opposite situation, higher bandwidths are associated to lower number of hops. In facts, equation (3.5) represents a straight-line with a negative angular coefficient, which means that higher bandwidths are assigned to shorter paths and viceversa. After having explained the reasons that led us to develop algorithm Range, we are now interested in some results. In next figures are presented the blocking probability curves of all four proposed algorithms in function of the arrival rate, for each one of the considered values of dirtying rates.

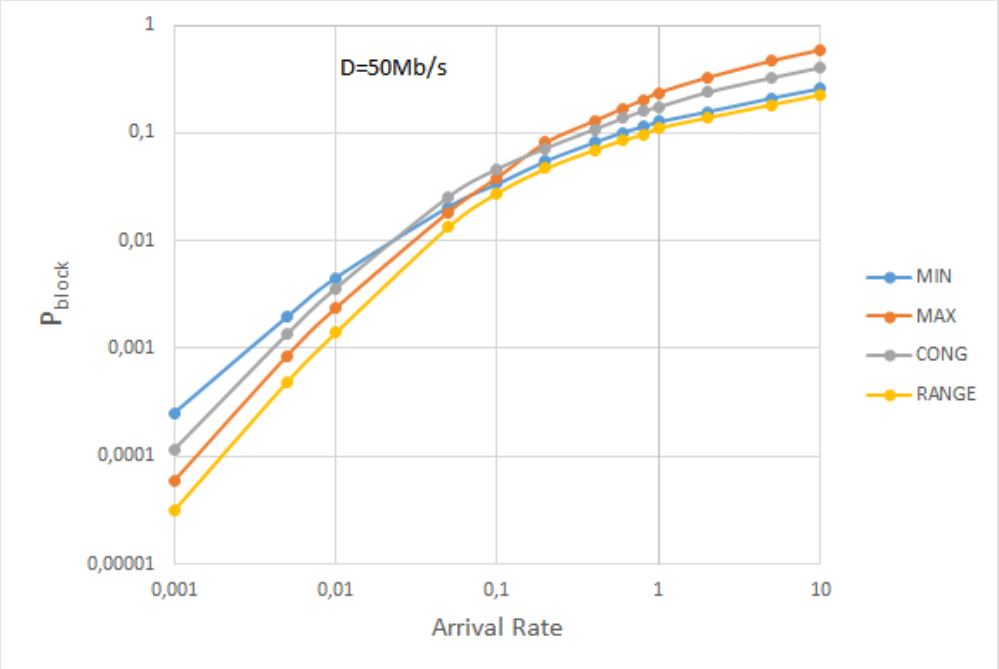

Figure 4.23: Blocking Probability for algorithm Range for D=50Mb/s

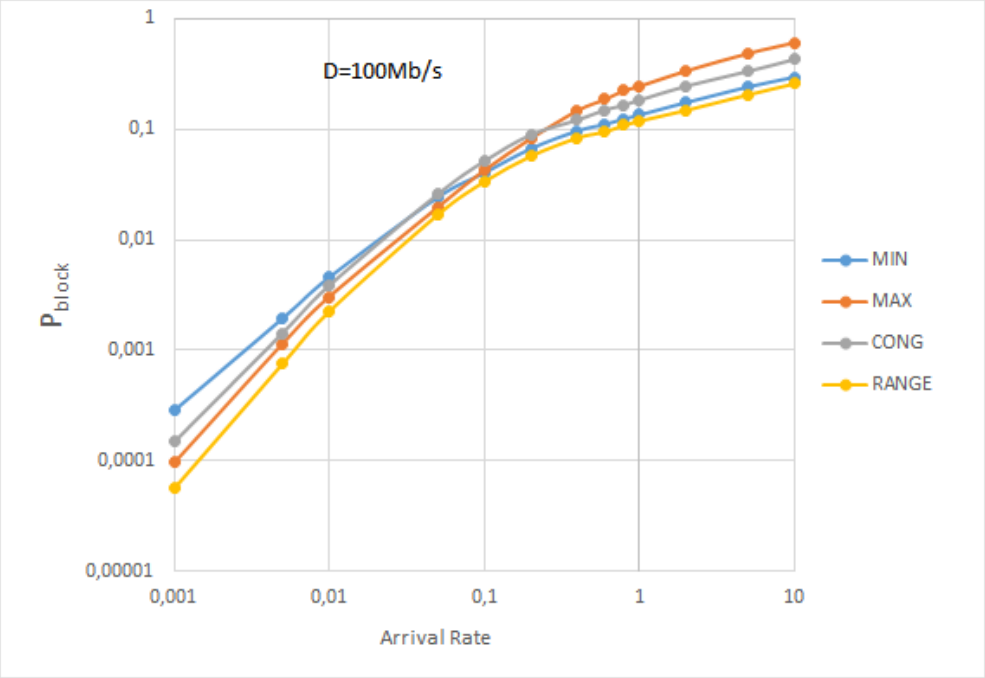

Figure 4.24: Blocking Probability for algorithm Range for D=100Mb/s

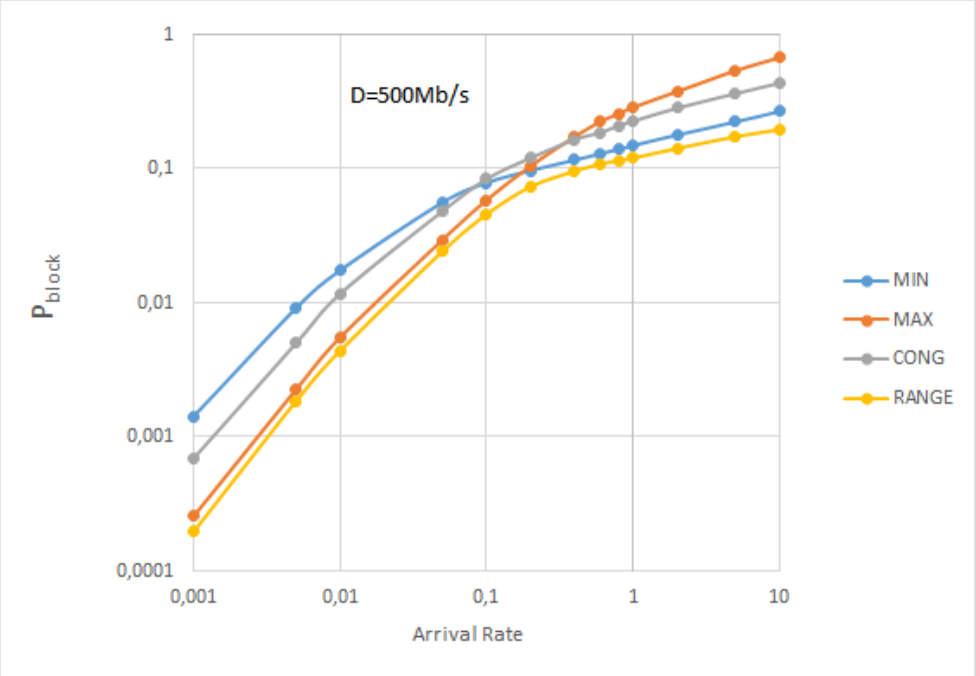

Figure 4.25: Blocking Probability for algorithm Range for D=500Mb/s

Looking at figures 4.23, 4.24 and 4.25, you can observe a very interesting trend for the blocking probability of algorithm Range (it is represented by the yellow curve). It reaches values of probability both less than those of algorithm Min for high arrival rates and less than those of algorithm Max for low arrival rates. Hence data from simulations have proved what we formulated as a hypothesis earlier: namely path's length is a good metric for provisioning bandwidth for migrations and, consequently, it is a key factor in determining blocking probability.

Chapter 5

Multiple Virtual Machines Migration

As it can be easily found in literature, many live migration methods are proposed to improve the migration performance and efficiency. But, due to the high power consumption, the idea of increasing the data center processing power may not always be the optimal solution to the issue of the increasing growth of ICT energy consumption. Since live migration technology is widely used in modern cloud computing data centers, live migration of multiple virtual machines is becoming even more frequent. For these reasons, the new idea of federated cloud computing model is arising. It is based on an intelligent sharing of the workloads across the DC resources of multiple cloud providers. Obviously several issues may be solved, such as the design of inter-data center connection network, building suitable communication infrastructure to achieve the required quality of service. One of the fundamental ideas that makes this new model possible, is the use of virtual machines, as they make the decoupling of services and applications from the underlying hardware environment possible. Moreover, in a cloud federation, virtual machines can be easily moved from a data center to another one while maintaining the network state consistency. This feature can be very useful when considering sets of correlated virtual machines that must be migrated together, in a live way, while maintaining their interconnections. In facts, many applications are often executed over multiple virtual machines and, then relative services are

available to users only if all the VMs are active and interconnected.

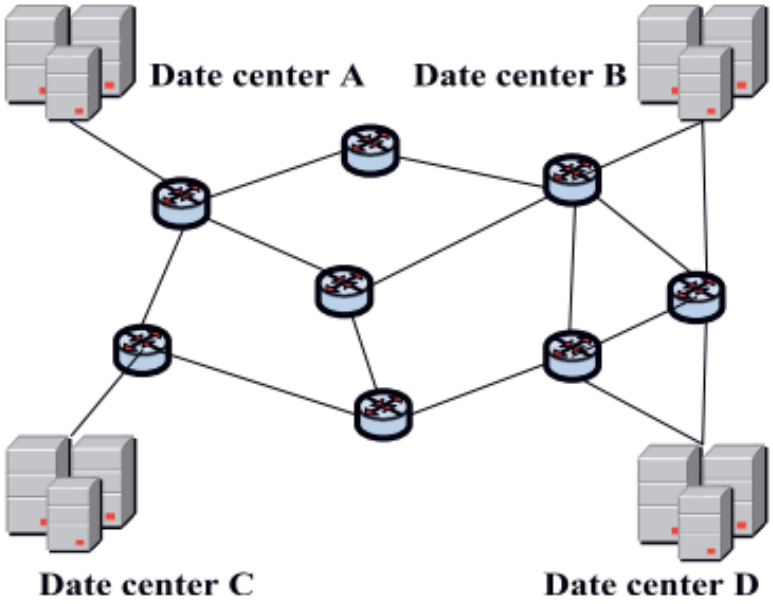

Figure 5.1: Data centers network example

Different from the single virtual machine migration, the live migration of multiple virtual machines is able to face many new problems, such as: migration failures due to the insufficient resources in target machine, conflicts due to the concurrent migrations, and troubles due to the dynamic changes of virtual machine workloads. All the above issues should be overcome to maximize the migration efficiency in virtualized cloud data center environments. Hence, new methods of multiple virtual machines migration need to be developed. While the issue of single VM live migration has been extensively studied, the case of multiple VM migration is still to be investigated in details. In this chapter, we will present two methodologies for the migration of a set of multiple virtual machines: serial migration and parallel migration. After that, we will try to build a model for this new topic: an integer linear programming formulation for the serial/parallel migration. In this thesis work, we have proposed and evaluated new algorithms for the routing and bandwidth assignment problem for virtual machines migration, aiming to solve the trade-off between migration performance and resource utilization. Hence, in this chapter we would like to understand how the migration of multiple virtual

machines can affect the network resources consumption.

5.1 Model Parameters

The problem of migrating a set of correlated virtual machines is more complex than the one of migration a single machine per time. For this reason, the metrics used to evaluate performance of the migration will depend on the migration of several machines and not just one as well as the way in which migrations are performed. So, we can use the model earlier adopted in the simulative section, to model the migration process, but is has to be expanded because we are now considering a set of virtual machines, and it is not trivial. Hence, a new definition of downtime and migration duration is required. We consider the general case of a set of M virtual machines with the following parameters:

- V_j^M is the memory size of the j-th virtual machine;
- D_j is the page dirtying rate of the j-th virtual machine;
- P_j is the memory page size of the j-th virtual machine.

According to the pre-copy based migration approach, when the j-th VM memory copy begins at time $t_{0,j}$, the entire memory V_j^M is copied to the destination location, while the virtual machine is still running at the source location. Then the iterative phase can start: modified memory pages are transferred in each iteration, which starts at times $t_{1,j}, \ldots, t_{n_j,j}$. The iterative phase goes on until the amount of dirty memory is below a predefined threshold V_{th} or a maximum number n_{max} of iterations is performed. If R_j is the bandwidth value selected for migrate the virtual machine j, we can assume that both R_j and D_j are constant rates during the entire migration process (this assumption is reasonable as the memory page dirtying rate depends on the application running on the machine). Moreover we assume that the memory allocated to each one of the M virtual machines is the same. Based on these considerations, we have:

$$D_j = D, \qquad V_j^M = V^M, \qquad P_j = P, \qquad \forall\ j \in M$$

So the duration of the i-th iteration of memory transfer of virtual machine j is:

$$T_{i,j} = V^M \frac{(P \cdot D)^i}{R_j^{i+1}} \qquad i = 0, \ldots, n_j \tag{5.1}$$

As explained in previous chapters, the migration process of the j-th virtual machines can reach good performance if the average dirtying rate is smaller than the transfer rate, which means:

$$\lambda_j = \frac{(P \cdot D)}{R_j} < 1 \tag{5.2}$$

The last round (the stop-and-copy phase) can start either for the smaller iteration index i such that:

$$V_{i,j} = \lambda_j^i \cdot V^M \leq V_{th} \tag{5.3}$$

or when n_{max} iterations are reached. Therefore the number of iterations required to migrate the virtual machine j is:

$$N_j = \min\left(\left\lceil \log_{\lambda_j} \frac{V_{th}}{V^M} \right\rceil, n_{max}\right) \tag{5.4}$$

The total time required to migrate the j-th VM is given by:

$$T_{mig,j} = \sum_{i=0}^{n_j} T_{i,j} = \frac{V^M}{R_j} \frac{1 - \lambda_j^{n_j+1}}{1 - \lambda_j} \tag{5.5}$$

However the total time required to migrate a set of virtual machines (and so also the total downtime) is strictly relate on how the VM are migrated. In particular it is a very important issue the order in which they are moved, i.e. when they are scheduled to be migrated. Here we adopt two procedures: the case in which the M virtual machines are migrated one by one (i.e. one at time) and the case when they are all migrated at the same time. We call the former case "'serial migration"' and the latter "'parallel migration"'.

5.2 Serial Migration of Virtual Machines

Serial migration means that virtual machines have to be migrated one by one, one at time and that each migration is performed at full transmission rate. This means that, among the links of the selected path for the migration, the whole capacity is used to migrate the VM. Moreover, as we are considering a set of M virtual machines, the transmission rate is equal for all the migration requests: i.e.

$$R_j = R, \qquad D_j = D$$

where D_j is the dirtying rate and it is constant for all the virtual machines. Hence, we can define the ratio:

$$\lambda = \frac{P \cdot D}{R}$$

Besides, we need to introduce another parameter which is the number of iterations needed to migrate a VM in the pre-copy approach, that has been already defined by n_j, but from now on we will refer to it as $n^{(s)}$. Therefore the total time required for migrating M virtual machines in a serial manner is:

$$T_{mig}^{(s)} = \sum_{j=1}^{M} T_{mig,j} = M \cdot \frac{V^M}{R} \frac{1 - \lambda^{n^{(s)}+1}}{1 - \lambda} \tag{5.6}$$

While total migration time can be evaluated as the sum of the migration time of the single virtual machine for all the machines belonging to the set, the downtime calculation needs some more considerations. In particular, the downtime of the entire set of VM starts when the first virtual machine is stopped at the source host (which means when the last iteration, the stop-and-copy phase, of the first VM begins) and it ends when the last machine is resumed at the destination location. We call T_{res}, the time required to restart a virtual machine at the destination host. So the downtime for the serial migration can be defined as:

$$T_{down}^{(s)} = \frac{V^M}{R} \lambda^{n^{(s)}} + (M-1) \frac{V^M}{R} \frac{1 - \lambda^{n^{(s)}+1}}{1 - \lambda} + T_{res} \tag{5.7}$$

5.3 Parallel Migration of Virtual Machines

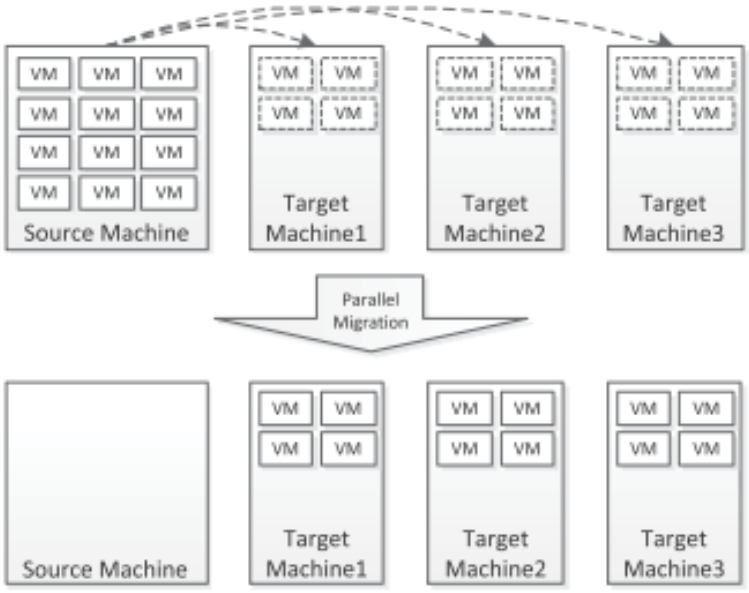

Figure 5.2: Parallel virtual machines migration example

When the M virtual machines are migrated in a parallel way (it means, simultaneously), each machine is transferred at a rate which is lower than the link capacity. In fact, being migrated at the same time the whole available bandwidth is shared among the served connections. Here, we assume that all the M virtual machine of the set have the same memory size, which means that the same amount of bandwidth is assigned to each of them. Besides, all the virtual machines start, and also end, the migration at the same time. Hence, we can assume that there are always M machines migrated at the same time and each transfer occurs at rate:

$$R_j = \frac{R}{M} \qquad \forall\ j$$

But if R_j is calculated in this way, it means that:

$$\lambda_j = M\lambda$$

While the number of iterations needed to complete the migration of each virtual machines is:

$$n_j = n^{(p)}$$

The total migration time required to perform the parallel migration of the whole set of virtual machines is equivalent to the total migration time of any single virtual machines; so it is calculated as:

$$T_{mig}^{(p)} = T_{mig,j} = M\frac{V^M}{R}\frac{1-(M\lambda)^{n^{(p)}+1}}{1-M\lambda} \tag{5.8}$$

The total downtime of the parallel migration if a set of M virtual machines corresponds to the last iteration of any single virtual machines (i.e. the stop-and-copy phase), and it can be evaluated as:

$$T_{down}^{(p)} = M\frac{V^M}{R}(M\lambda)^{n^{(p)}} + T_{res} \tag{5.9}$$

5.4 Comparison of the two methods

Observing the reported formulas in the previous sections, we can draw some conclusions about the two presented methods, by evaluating the two metrics, total migration time and downtime. We first focus on the total time required to migrate a set of M virtual machines. From equations (5.6) and (5.8) we immediately note that the rate at which migrations are performed is not the same. For the serial case it is the same for all the machines of the set and it is equal to the full transfer rate while in the parallel one each machine is migrated at a rate which is a portion of the full rate as bandwidth is shared among several migration requests. As a consequence migrating a sequence of M VMs at rate R does not require the same amount of time as migrating M VMs in parallel (i.e. each one at rate R/M). This occurs because in the iterative phase the fixed memory page dirtying rate generates different amounts of data to be transferred in the two cases. As in the parallel case migrations are performed at a lower rate than in the serial case, it can be proved that the parallel total migration time is always greater than the serial total migration time. In particular we can prove this result by showing some

calculation. If we subtract equation (5.6) from the (5.8), we have:

$$T_{mig}^{(p)} - T_{mig}^{(s)} = \frac{M\lambda(M-1)(V^M - V_{th})}{R(1-\lambda)(1-M\lambda)} \tag{5.10}$$

we note that, under assumption earlier declared, this quantity is always non negative. We can repeat a similar evaluation also for the downtime, by considering the equations (5.7) and (5.9), which represent the downtime for serial and parallel migration, respectively.

$$T_{down}^{(s)} - T_{down}^{(p)} = \frac{(M-1)(V^M - V_{th})}{R(1-\lambda)} \tag{5.11}$$

in this way we proved that the serial migration downtime is larger than the parallel migration downtime.

From the above presented results it isn't possible to select one of the two proposed multiple virtual machines migration method as the best one in any case. On the other hand, they suggest that choosing the best solution may depends on the service you want to provide. More in details, parallel migration offers better performance in terms of quality of service since it guarantees a smaller downtime. The serial migration, instead, presenting a lower total migration time, can achieve better results in terms of total network resources consumption and transmission overhead. So in the next section, we propose a mathematical model for the multiple virtual machines migration process, in which the model has to decide whether perform parallel or serial migration in according to several linear constraints.

5.5 Integer Linear Programming Model

In this section, we present the Integer Linear Programming (ILP) model built to solve the live migration problem of a set of virtual machines.

5.5.1 Model description

The optimization problem we are dealing with consists of minimizing the amount of network resource used for the migration of multiple virtual machines. The model gives as output the sum of the resource occupation on each link of the network for both the serial and parallel migration. The goal of our problem is to decide, for all the requests, whether a migration has to be provisioned in a serial or parallel way while selecting a function point (i.e. the couple of bandwidth provided for the migration and the corresponding migration time) for each virtual machine that needs to be migrated. The total resource occupation is composed of the network resources used for the serial migration and the resources used for the parallel migration (the resource consumption is evaluated as the product between provisioned bandwidth for the migration, total migration time and number of hops of the path selected for the routing of the migration request). The ILP problem is stated as follows. Given a physical fixed network topology and a set of virtual machines, we decide the optimal function point selection as well as the migration method (serial or parallel) in order to minimize the overall network resources consumption. In this optimization problem we want to minimize the amount of used resource in the network in order to satisfy all the migration requests, i.e., to migrate each virtual machines from a source data center to a destination one. We assume that all the links in the network can be utilized for the routing of the requests while only a set of nodes can be the destination of the migration requests.

5.5.2 Sets and parameters

- $G = (N, E)$ is the graph used to model the physical network topology, where N represents the set of nodes and E the set of bidirectional links.

- D is a subset of the set N and it is the set of the data centers, the only nodes that can be source and destination of the migration requests.
- V is the set of the virtual machines to be migrated.
- F represents the set of the function points: each pair (together with a virtual machine $v \in V$) has a bandwidth and a total migration time as parameters.
- $C_{i,j}$ is the link capacity, for each link (i, j) in the set E.
- $B^{p,v}$ is a bandwidth value of the function point p of the virtual machines v.
- $T^{p,v}$ is a total migration time of the function point p of the virtual machines v.
- V^v is the memory size of the virtual machine v.
- $M_d^{f,v}$ represents the migration request of virtual machine v from a node d of the network to a destination data center node f.

5.5.3 Decision variables

- $x_{i,j}$ (binary) is used to indicate if the link (i, j) is used ($x_{i,j}$=1) or not ($x_{i,j}$=0).
- $y_{i,j}^{d,f,v}$ (binary) is used to indicate if the link (i, j) is used to migrate the virtual machine v ($x_{i,j}$=1) from node d to node f or not ($x_{i,j}$=0).
- $z_{i,j}$ (integer) represents the total amount of resources utilized on link (i, j).
- $d_{i,j}^{f,d,v}$ (integer) represents the amount of data transported on link (i, j) from a node f to destination node d for the virtual machine v.
- $w_{d,e}^{v,p}$ (binary) is used to indicate if the bandwidth of the function point p is used to migrate the virtual machine v from node d to destination node e ($w_{d,e}^{v,p}$=1) or not ($w_{d,e}^{v,p}$=0).
- $u_{d,e}^{v,p}$ (binary) is used to indicate if the migration time of the function point p is used to migrate the virtual machine v from node d to destination node e ($u_{d,e}^{v,p}$=1) or not ($u_{d,e}^{v,p}$=0).

- $p_{i,j}^{d,f}$ (binary) is used to indicate if a request from node f to the destination node d is migrated parallel on link (i,j) ($p_{i,j}^{d,f}$=1) or not ($p_{i,j}^{d,f}$=0).
- $pv_{i,j}^{d,f,v}$ (binary) is used to indicate if the virtual machine v is migrated from a node f to the destination node d in a parallel way ($pv_{i,j}^{d,f,v}$=1) or not ($pv_{i,j}^{d,f,v}$=0).
- $s_{i,j}^{d,f}$ (binary) is used to indicate if a request from node f to the destination node d is migrated serial on link (i,j) ($p_{i,j}^{d,f}$=1) or not ($p_{i,j}^{d,f}$=0).
- $sv_{i,j}^{d,f,v}$ (binary) is used to indicate if the virtual machine v is migrated from a node f to the destination node d in a serial way ($pv_{i,j}^{d,f,v}$=1) or not ($pv_{i,j}^{d,f,v}$=0).
- $rp_{i,j}^{d,f}$ (integer) represents the total amount of resources consumed on link (i,j) for a parallel migration from a node f the destination node d ($rp_{i,j}^{d,f}$=1) or not ($rp_{i,j}^{d,f}$=0).
- $rs_{i,j}^{d,f}$ (integer) represents the total amount of resources consumed on link (i,j) for a serial migration from a node f to the destination node d ($rs_{i,j}^{d,f}$=1) or not ($rs_{i,j}^{d,f}$=0).
- $rsv_{i,j}^{d,f,v}$ (integer) represents the amount of resources consumed on link (i,j) for the serial migration of the virtual machine v from a node f to the destination node d ($rsv_{i,j}^{d,f}$=1) or not ($rsv_{i,j}^{d,f}$=0).

5.5.4 Objective function

Minimize :

$$\sum_{(i,j)\in E}\sum_{d\in N}\sum_{f\in D}\left(rp_{i,j}^{d,f}+rs_{i,j}^{d,f}\right) \tag{5.12}$$

The first contribution of the objective function accounts for the overall resource consumption due to the parallel migrated virtual machines while the second contribution represents the one due to the serial migrated virtual machines. The objective of the optimization is to minimize the overall network resources consumption. Note that the summations are evaluated for each destination node, for each node and for each link of the network.

5.5.5 Constraints

$$\sum_{(i,d)\in E}\sum_{e\in N} d_{i,d}^{e,f,v} + M_d^{f,v}\cdot V^v = \sum_{(d,j)\in E}\sum_{e\in N} d_{d,j}^{e,f,v} \qquad \forall\, d\in N, \forall\, f\in D, \forall\, v\in V \tag{5.13}$$

$$\sum_{(i,d)\in E}\sum_{e\in N} d_{i,d}^{e,f,v} = \sum_{e\in N} M_e^{d,v}\cdot V^v \qquad \forall\, d\in D, \forall\, v\in V \tag{5.14}$$

$$\sum_{v\in V}\sum_{d\in N}\sum_{f\in D}\sum_{p\in P} w_{d,f}^{v,p}\cdot y_{i,j}^{d,f,v} \le C_{i,j} \qquad \forall\,(i,j)\in E \tag{5.15}$$

$$d_{i,j}^{d,f,v} \le y_{i,j}^{d,f,v}\cdot C_{i,j} \qquad \forall\,(i,j)\in E, \forall\, d\in N, \forall\, f\in D, \forall\, v\in V \tag{5.16}$$

$$x_{i,j} = y_{i,j}^{d,f,v} \qquad \forall\,(i,j)\in E, \forall\, d\in N, \forall\, f\in D, \forall\, v\in V \tag{5.17}$$

$$\sum_{p\in F} w_{d,e}^{v,p} \ge y_{i,j}^{d,e,v}\cdot M_d^{e,v} \qquad \forall\,(i,j)\in E, \forall\, d\in N, \forall\, e\in D, \forall\, v\in V \tag{5.18}$$

$$\sum_{p\in F} u_{d,e}^{v,p} \ge y_{i,j}^{d,e,v}\cdot M_d^{e,v} \qquad \forall\,(i,j)\in E, \forall\, d\in N, \forall\, e\in D, \forall\, v\in V \tag{5.19}$$

$$w_{d,e}^{v,p} \le u_{d,e}^{v,p}\cdot M_d^{e,v} \qquad \forall\, d\in N, \forall\, e\in D, \forall\, v\in V, \forall\, p\in F \tag{5.20}$$

$$z_{i,j} \ge \sum_{d\in N}\sum_{e\in D}\sum_{v\in V}\sum_{p\in F} w_{d,e}^{v,p}\cdot B^{p,v}\cdot T^{p,v} \qquad \forall\,(i,j)\in E \tag{5.21}$$

$$p_{i,j}^{d,f} + s_{i,j}^{d,f} = M_d^{f,v}\cdot y_{i,j}^{d,f,v} \qquad \forall\,(i,j)\in E, \forall\, d\in N, \forall\, f\in D, \forall\, v\in V \tag{5.22}$$

$$pv_{i,j}^{d,f,v} \leq p_{i,j}^{d,f} \cdot y_{i,j}^{d,f,v} \qquad \forall\ (i,j) \in E, \forall\ d \in N, \forall\ f \in D, \forall\ v \in V \tag{5.23}$$

$$sv_{i,j}^{d,f,v} \leq s_{i,j}^{d,f} \cdot y_{i,j}^{d,f,v} \qquad \forall\ (i,j) \in E, \forall\ d \in N, \forall\ f \in D, \forall\ v \in V \tag{5.24}$$

$$pv_{i,j}^{d,f,v} + sv_{i,j}^{d,f,v} = M_{d}^{f,v} \cdot y_{i,j}^{d,f,v} \qquad \forall\ (i,j) \in E, \forall\ d \in N, \forall\ f \in D, \forall\ v \in V \tag{5.25}$$

$$rp_{i,j}^{d,f} \geq \sum_{v \in V} \sum_{p \in F} y_{i,j}^{d,f,v} \cdot w_{d,f}^{v,p} \cdot B^{p,v} \cdot T^{p,v} \qquad \forall\ (i,j) \in E, \forall\ d \in N, \forall\ f \in D \tag{5.26}$$

$$rsv_{i,j}^{d,f,v} \geq \sum_{p \in F} V^{v} \cdot sv_{i,j}^{d,f,v} \cdot B^{p,v} \cdot T^{p,v} \qquad \forall\ (i,j) \in E, \forall\ d \in N, \forall\ f \in D, \forall\ v \in V \tag{5.27}$$

$$rs_{i,j}^{d,f} \geq \sum_{v \in V} rsv_{i,j}^{d,f,v} \qquad \forall\ (i,j) \in E, \forall\ d \in N, \forall\ f \in D \tag{5.28}$$

Equations (5.13) and (5.14) are the balance constraints for each node and for each virtual machine. In particular the latter is referred to each destination node while the former is referred to each other node in the network topology. (5.15) is the link capacity constraint and it is evaluated for each link of the network. (5.16) and (5.17) are the link consistency constraints: the total amount of data transferred in each link must not exceed the link capacity. The next three relation are consistency constraints due to the function points: (5.18) implies that a bandwidth should be chosen if there is a migration request, while (5.19) implies that a total migration time has to be selected if there is a migration request. (5.20) guarantees that each

virtual machine is migrated using the values of bandwidth and migration time of the same function point. (5.21) calculates the total resources used on each link for all the requests that pass through it. (5.22) states that the request has to be migrated in only one of the two method: parallel or serial. (5.23) states that if the request from source node d to destination node f is parallel migrated, then all the virtual machines from source d to destination f are parallel migrated. (5.24) states that if the request from source node d to destination node f is serial migrated, then all the virtual machines from source d to destination f are serial migrated. (5.25) states that the migration of the virtual machine v from node d to destination f can be either serial or parallel. (5.26) evaluates the total amount of resources used of link (i, j) for the parallel migrations. (5.27) evaluate the consumed resources on link (i, j) for each virtual machine serial migrated while (5.28) calculate the overall resource used,on link (i, j) for serial migration.

Chapter 6

Conclusions

In recent years there has been an evolution of the network traffic from a mostly static paradigm to a more dynamic one. New opportunities are then arising, aiming to efficiently employ the large capacity of transport networks, provided by the fiber optics and their great reliability. Among all the new services that are rapidly evolving, Cloud Computing assumes considerable importance. It appears as a new hybrid model of resource exploitation, where resources can be represented by computer networks or more in general the Internet. In this new architecture, the basic idea is that services and applications should reside mainly on web servers (the so called "'Cloud"') instead of on personal computers interconnected by a network. In this way, each user having a device, can access the "'Cloud"', which is then able to deliver the services and the data that the user is requesting. As these web servers are increasingly used, also their energy consumption level is growing. Among the various solutions that the sector ICT is developing to address this issue, we have focused our attention on the benefits offered by host virtualization. In particular it allows data center providers to make an abstraction of their services over these virtual machines. The main advantage of creating virtual machines is that you can migrate them from a data center to another one, in order to perform load balancing, cloud bursting, fault recovery, energy consumption reduction. In this thesis, we first proposed a model for the live virtual machines migration giving a full explanation of all the involved parameters. Then we discussed about the trade off between network resource consumption and migration performance and, for this

reason, we decided to focus our attention on the bandwidth provisioning scheme. In particular, we proposed four algorithms to solve the Routing and bandwidth Assignment problem for a network scenario in which the traffic is only represented by the migration requests from a source data center to a destination one. The virtual machines migration topic is widely discussed in the literature, but there are still no studies on the bandwidth assignment in a network of data centers. So this thesis work is devoted to the study and implementation of algorithms on a discrete event-based C++ simulator in order to evaluate their performances, in terms of blocking probability.

From the study of the results obtained for the first two implemented algorithms, Min and Max, we found out that assigning always a minimum (or maximum) value of bandwidth can achieve different performances, depending on the arrival rates of the migration requests. In particular we observed that allocating lower bandwidth allows to achieve good performance only for high arrival rate while assigning greater bandwidth results in lower blocking probability for lower arrival rates. So we note a double behavior for these two algorithms as the most influential factor in determining the blocking probability changes from low arrival rates (migration requests duration) to high rate (number of simultaneous served request). After these first remarks, we asked ourselves what results could lead assigning the bandwidth in a more adaptive way. As algorithms Min and Max evaluate the provisioned bandwidth for the migration in a static manner, the new algorithm, Congestion-Aware, takes into account the network state in this calculation. From results, for this third algorithm, we observed a blocking probability behavior similar to that of the previous ones. In particular, the probability curve of Congestion-Aware is placed between the curves of Min and Max, which means that: for high arrival rates it behaves better than Max but worse than Min while for small arrival rate it achieves lower probability than Min but greater than Max.

As a further study, we investigate also the Resource Occupation, in order to better understand the behavior of these algorithms. We found out that the path's length (i.e. the number of links belonging to the chosen path for the routing of the input migration request, measured in hop) plays a key role in determining both network resource consumption and blocking probability. This discovery constitutes the basis on which build the fourth algorithm, Range. The main idea is to

assign bandwidth to a migration request depending on the length of the selected path. After some simulations, results have confirmed our assumptions showing that a relationship can be established between the number of hops of the path and the blocking probability. Moreover, Range is able to capture both the behaviors presented by previous algorithms: it achieves values of blocking probability lower than Min for high arrival rates and lower than Max for small arrival rates.

After illustrating the results obtained from a study carried out in a purely simulative way, we focused our attention on the issue of migrating multiple virtual machines. In chapter 5, we aim to observe our main topic by a different point of view, presenting two model for the migration of a set of virtual machines: one for the serial and one for the parallel migration. Besides, is proposed an integer linear programming formulation whose objective function tries to minimize the network resource utilization while deciding when to perform the serial migration or the parallel one.

In conclusion, the goal of this thesis is to formulate a new analytical approach that is able to get many different results from the ones you can find in the literature. The work we have proposed can form the basis for a new approach to the trade-off that the Cloud technology has resulted in the ICT sector: achieve optimal performance while guaranteeing a network resource consumption as low as possible. Despite the good results, we think that the performance and the accuracy of these algorithms can still be improved. In particular we identified several issues that can be implemented in a future study:

- separation of intra data center traffic and inter data center one;
- introduction of a high number of data centers in the network;
- usage of a different network topology in the simulation tool;
- introduction of traffic grooming;
- dirtying rate daily variation;
- daily variation of traffic distribution;
- consider a higher number of different types of virtual machines;

- introduce the multiple virtual machines migration model into the simulation tool: both the serial and the parallel migration;
- usage of a Wavelength-Division-Multiplexing based scheme for the bandwidth provisioning;

List of Figures

List of Tables

Bibliography

[1] *Low-Emissions Routing for Cloud Computing in IP-over-WDM Networks with Data Centers*
Mirko Gattulli, Massimo Tornatore, Riccardo Fiandra, and Achille Pattavina
IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 32, NO. 1, JANUARY 2014

[2] *Greening the Cloud Using Renewable-Energy-Aware Service Migration*
Uttam Mandal, M. Farhan Habib, Shuqiang Zhang, and Biswanath Mukherjee, University of California, Davis Massimo Tornatore, University of California, Davis and Politecnico di Milano
IEEE Network November/December 2013

[3] *Bandwidth and Routing Assignment for Virtual Machine Migration in Photonic Cloud Networks*
Uttam Mandal, M. Farhan Habib, Shuqiang Zhang, Massimo Tornatore,Biswanath Mukherjee

[4] *Cost-Efficient Live VM Migration Based on Varying Electricity Cost in Optical Cloud Networks*
Abhishek Gupta, Uttam Mandal, Pulak Chowdhury, Massimo Tornatore and Biswanath Mukherjee University of California, Davis, USA Politecnico di Milano, Italy
This work was funded under NSF award number 1217978

[5] *Heterogeneous Bandwidth Provisioning for Virtual Machine Migration over SDN-Enabled Optical Networks*

Uttam Mandal, M. Farhan Habib, Shuqiang Zhang, Pulak Chowdhury, Massimo Tornatore, and Biswanath Mukherjee

[6] *Bandwidth Provisioning for Virtual Machine Migration in Cloud: Strategy and Application*
Uttam Mandal, Pulak Chowdhury, Massimo Tornatore, Charles U. Martel, and Biswanath Mukherjee University of California, Davis; Also with Politecnico di Milano, Italy
This work has been supported by NSF Grant No. CNS-1217978

[7] *Performance and Energy Modeling for Live Migration of Virtual Machines*
Haikun Liu, Cheng-Zhong Xu, Hai Jin, Jiayu Gong, Xiaofei Liao
This work is supported by National 973 Basic Research Program of China under grant No. 2007CB310900, the MoE-Intel Information Technology Special Research Foundation under grant No. MOE-INTEL-10-05, and U.S. NSF under grants CRI-0708232, CNS-0702488, CNS-0914330, CCF-1016966

[8] *Live Migration of Virtual Machines*
Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield
USENIX Association NSDI '05: 2nd Symposium on Networked Systems Design and Implementation pages 273-286

[9] *CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines*
Timothy Wood, K. K. Ramakrishnan, Fellow, IEEE, Prashant Shenoy, Fellow, IEEE, Senior Member, ACM, Jacobus Van der Merwe, Jinho Hwang, Guyue Liu, and Lucas Chaufournier
IEEE/ACM TRANSACTIONS ON NETWORKING

[10] *Towards a Network Aware VM Migration: Evaluating the Cost of VM Migration in Cloud Data Centers*
Hellen Maziku and Sachin Shetty College of Engineering Tennessee State University, Nashville, TN, USA
2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)

[11] *Resource Allocation using Virtual Machine Migration: A Survey*
Ts'epo Mofolo, R. Suchithra, N. Rajkumar MS (IT) Department Jain University, Bangalore, India
Poster Paper Proc. of Int. Conf. on Advances in Information Technology and Mobile Communication 2013

[12] *Live Virtual Machine Migration Techniques in Cloud Computing: A Survey*
Pradip D. Patel, Miren Karamta, M. D. Bhavsar, M. B. Potdar
International Journal of Computer Applications (0975 - 8887) Volume 86 - No 16, January 2014

[13] *Energy-efficient Management of Virtual Machines in Eucalyptus*
Pablo Graubner, Matthias Schmidt, Bernd Freisleben Department of Mathematics and Computer Science, University of Marburg Hans-Meerwein-Str. 3, D-35032 Marburg, Germany
2011 IEEE 4th International Conference on Cloud Computing

[14] *Power Consumption of Virtual Machine Live Migration in Clouds*
Qiang Huang, Fengqian Gao, Rui Wang, Zhengwei Qi School of Software, Shanghai Jiaotong University, Shanghai, China 200240
2011 Third International Conference on Communications and Mobile Computing

[15] *Does Live Migration of Virtual Machines cost Energy?*
Anja Strunk and Waltenegus Dargie Chair of Computer Networks Faculty of Computer Science Technical University of Dresden 01062 Dresden, Germany
2013 IEEE 27th International Conference on Advanced Information Networking and Applications

[16] *On the Efficiency of Dynamic Routing of Connections with Known Duration*
Diego Lucerna, Andrea Baruffaldi, Massimo Tornatore, Achille Pattavina

[17] *Delay Guaranteed Live Migration of Virtual Machines*
Jiao Zhang, Fengyuan Ren and Chuang Lin Dept. of Computer Science and Technology, Tsinghua University, Beijing, China Tsinghua National Labora-

tory for Information Science and Technology, Beijing, China
IEEE INFOCOM 2014 - IEEE Conference on Computer Communications

[18] *Efficient Pre-Copy Live Migration with Memory Compaction and Adaptive VM Downtime Control*
Guangyong Piao, Youngsup Oh, Baegjae Sung, and Chanik Park Department of Computer Science and Engineering Pohang University of Science and Engineering (POSTECH) Pohang, South Korea
2014 IEEE Fourth International Conference on Big Data and Cloud Computing

[19] *Predicting the Performance of Virtual Machine Migration*
Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W. Moore and Andy Hopper University of Cambridge Computer Laboratory

[20] *Dynamic Bandwidth Allocation Schemes to Improve Utilization under Non-Uniform Traf?c in Ethernet Passive Optical Networks*
Kyuho Son, Hyungkeun Ryu, Song Chong and Taewhan Yoo
IEEE Communications Society 0-7803-8533-0/04/20.00 (c) 2004IEEE

[21] *Traffic Aware Cross-Site Virtual Machine Migration in Future Mobile Cloud Computing*
Jiaqiang Liu, Yong Li, Depeng Jin, Li Su, Lieguang Zeng
Published online: 27 September 2014 Springer Science+Business Media New York 2014

[22] *Optimizing Live Migration of Virtual Machines with Context Based Prediction Algorithm*
Yong Cui, Yusong Lin, Yi Guo, Runzhi Li, Zongmin Wang
International Workshop on Cloud Computing and Information Security (CCIS 2013)

[23] *Network Aware VM Migration in Cloud Data Centers*
Hellen Maziku and Sachin Shetty College of Engineering Tennessee State University, Nashville, TN, USA
2014 Third GENI Research and Educational Experiment Workshop

[24] *A Network-aware Virtual Machine Placement and Migration Approach in Cloud Computing*
Jing Tai Piao, Jun Yan
2010 Ninth International Conference on Grid and Cloud Computing

[25] *Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments*
Kejiang Ye, Xiaohong Jiang, Dawei Huang, Jianhai Chen, Bei Wang
2011 IEEE 4th International Conference on Cloud Computing

[26] *Optimized Pre-Copy Live Migration for Memory Intensive Applications*
Khaled Z. Ibrahim, Steven Hofmeyr, Costin Iancu, Eric Roman

[27] *A Review of Routing and Wavelength Assignment Approaches for Wavelength-Routed Optical WDM-Networks*
Hui Zang, Jason P. Jue and Biswanath Mukherjee

[28] *Renewable Energy-Aware Grooming in IP-over-WDM Networks*
Thilo Schöndienst and Vinod M. Vokkarane
2014 International Conference on Computing, Networking and Communications, Green Computing, Networking and Communications Symposium

[29] *Provisioning of Deadline-Driven Requests With Flexible Transmission Rates in WDM Mesh Networks*
ragos Andrei, Student Member, IEEE, Massimo Tornatore, Member, IEEE, Marwan Batayneh, Student Member, IEEE, Charles U. Martel, and Biswanath Mukherjee, Fellow, IEEE
IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 18, NO. 2, APRIL 2010

[30] *IP Over WDM Networks Employing Renewable Energy Sources*
Xiaowen Dong, Taisir El-Gorashi and Jaafar M. H. Elmirghani

[31] *Live Migration of Virtualized Edge Networks: Analytical Modeling and Performance Evaluation*
Franco Callegati, Walter Cerroni

[32] *Energy-Information Transmission Tradeoff in Green Cloud Computing*
Amir-Hamed Mohsenian-Rad and Alberto Leon-Garcia
$978 - 1 - 4244 - 5637 - 6/104/26.00$ ĺ2010 IEEE

[33] *Traffic Grooming in an Optical WDM Mesh Network*
Keyao Zhu, Student Member, IEEE, and Biswanath Mukherjee, Member, IEEE
IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, VOL. 20, NO. 1, JANUARY 2002

[34] *Dynamic Traffic Grooming in WDM Mesh Networks Using a Novel Graph Model*
Hongyue Zhu, Hui Zang, Keyao Zhu, and Biswanath Mukherjee
This work has been supported by the National Science Foundation (NSF) under Grant Nos. NCR-9508239 and ANI-9805285, and by Sprint Advanced Technology Laboratories.

[35] *A Novel Generic Graph Model for Traffic Grooming in Heterogeneous WDM Mesh Networks*
Hongyue Zhu, Student Member, IEEE, Hui Zang, Member, IEEE, Keyao Zhu, Student Member, IEEE, and Biswanath Mukherjee, Member, IEEE
IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 11, NO. 2, APRIL 2003

[36] *Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments*
Kejiang Ye, Xiaohong Jiang, Dawei Huang, Jianhai Chen, Bei Wang
2011 IEEE 4th International Conference on Cloud Computing

[37] *CQNCR: Optimal VM Migration Planning in Cloud Data Centers*
Md. Faizul Bari, Mohamed Faten Zhani, Qi Zhang, Reaz Ahmed, and Raouf Boutaba David R. Cheriton School of Computer Science University of Waterloo, Ontario, Canada

[38] *Multiple Virtual Machine Live Migration in Federated Cloud Systems*
Walter Cerroni Department of Electrical, Electronic and Information Engi-

neering - University of Bologna
2014 IEEE INFOCOM Workshop on Cross-Cloud Systems