Performance and Security Provisioning for Mobile Telecom Cloud

by

Seyed Yahya Vaezpour
B.Sc., Sharif University of Technology, 2008
M.Sc., Amirkabir University of Technology, 2011

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science
University of Victoria
Victoria, BC Canada

Performance and Security Provisioning for Mobile Telecom Cloud

by

Seyed Yahya Vaezpour
B.Sc., Sharif University of Technology, 2008
M.Sc., Amirkabir University of Technology, 2011

Supervisory Committee

Dr. Kui Wu, Co-Supervisor
(Department of Computer Science)

Dr. Gholamali C Shoja, Co-Supervisor
(Department of Computer Science)

Dr. Wu-Sheng Lu, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Kui Wu, Co-Supervisor
(Department of Computer Science)

Dr. Gholamali C Shoja, Co-Supervisor
(Department of Computer Science)

Dr. Wu-Sheng Lu, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Mobile Telecom Cloud (MTC) refers to cloud services provided by mobile telecommunication companies. Since mobile network operators support the last-mile Internet access to users, they have advantages over other cloud providers by providing users with better mobile connectivity and required quality of service (QoS). The dilemma in meeting higher QoS demands while saving cost poses a big challenge to MTC providers. We tackle this challenge by strategically placing users' data in distributed switching centres to minimize the total system cost and maximize users' satisfaction. We formulate and solve the optimization problems using linear programming (LP) based branch-and-bound and LP with rounding.

Furthermore, we discuss MTC brokerage which allows MTC providers to act as a brokerage to broker third-party cloud providers' (TPC) cloud resources and integrate the resources reserved from TPC with those of their own MTC. We address the technical challenges of optimally allocating users' cloud requests to MTC and TPC data centres to meet users' QoS requirement with minimum cost. We also study the price range that can be profitable to a MTC brokerage. We then investigate the resource reservation problem with dynamic request changes. We evaluate our solution using real Google traces collected over a 29-day period from a Google cluster.

We also address security provisioning in MTC. Mobile cloud allows users to offload computational intensive applications to a mobile phone's agent in the cloud, which could be implemented as a thin virtual machine (VM), also termed as phone clone. Due to shared hardware components among co-resident VMs, a VM is subject to covert channel attacks and may potentially leak information to other VMs located in the same physical host. We design SWAP: a security aware provisioning and migration scheme for phone clones. We evaluate our solution using the Reality Mining and the Nodobo dataset. Experimental results indicate that our algorithms are nearly optimal for phone clone allocation and are effective in maintaining low risk and minimizing the number of phone clone migrations.

Contents

List of Tables

List of Figures

List of Abbreviations

MTC Mobile Telecom Cloud

TPC Third Party Cloud

MS Mobile Station

BTS Base Station Subsystem

DSC Distributed Switching Centre

DC Data Centre

QoS Quality of Service

VM Virtual Machine

SLA Service Level Agreement

LP Linear Programming

QP Quadratic Programming

SWAP Security Aware Provisioning and Migration Scheme for Phone Clones

MCF Maximum Conflict First

HDF Highest Degree First

CHDF Constrained Highest Degree First

ACKNOWLEDGEMENTS

I would like to thank:

My Father, Mother, Sister, and Brother, for supporting me through my education.

Professor Kui Wu and Professor Gholamali C Shoja, for mentoring, support, encouragement, and patience.

University of Victoria, NSERC, and Ericsson, for fellowship award and financial support.

Professor Jianping Wang and Rui Zhang, for the collaborative research work.

DEDICATION

I dedicate this thesis to my parents for their love and support throughout my life.

Chapter 1

Introduction

1.1 Motivation

Cloud computing provides mobile telecommunication companies with new opportunities to augment their services to mobile users with mobile telecom cloud (MTC), which refers to cloud services supported by mobile telecommunication companies. Since mobile network operators own the network infrastructure and support the last-mile Internet access to users, they have advantages over other cloud providers in offering cloud services with better quality of service (QoS).

MTC can play different roles in providing users with cloud services. In this thesis, we consider two main types of MTC cloud services. First, MTC can provide its own cloud services to the users. Second, MTC can integrate its own services with third-party cloud providers to provide users with higher QoS and cheaper cloud services. In this case, a MTC provider group buys TPC services with discounts and jointly schedules the reserved TPC resources and its own MTC resources. Figure 1.1 shows the main roles through which telecom companies can provide users with cloud services.

In the first case, MTC works independently to provide cloud services to the users. The main problem among many others is the network planning problem. In this thesis, we focus on the data placement problem. Because mobile communications companies already have the required physical system and networking infrastructure in place and also because they provide the last-mile Internet access to mobile users, they have direct knowledge of mobile users' mobility patterns, data access, and delay requirements. With this information, MTC can strategically distribute users' data in Distributed Switching Centres (DSCs). This grants MTC an unique advantage over

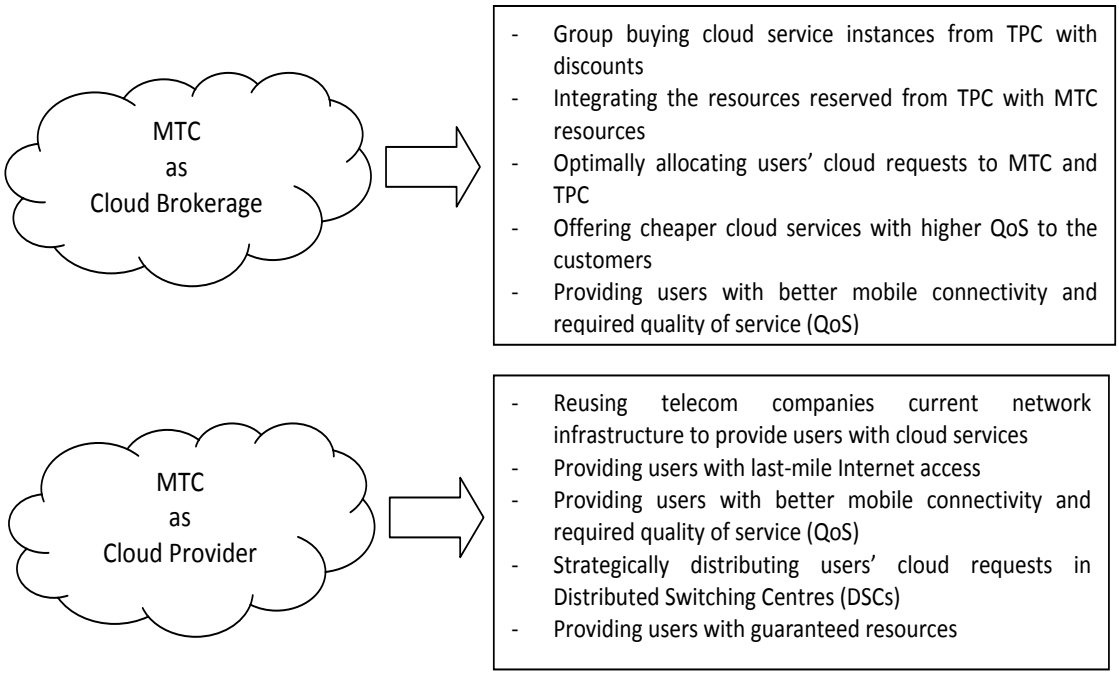

Figure 1.1: Mobile telecom companies as cloud brokerage and cloud provider

third-party mobile cloud.

Intuitively, a customer's data should be placed at locations where the telecom company has guaranteed resources and can directly control them to meet the QoS requirements. DSCs are the best place to host data centres (DCs) and users' data. At the first look, the idea of localizing a mobile user's data in a DSC seems trivial. As the number of users grows, however, this simple idea raises several challenges. In this thesis, we present a mathematical model for data placement problem in MTC that can be used to minimize the system cost and maximize users' satisfaction.

In the second case, MTC teams up with Third-Party Cloud (TPC) providers. In this scheme, the key challenge is to strategically integrate resources from different tiers to serve the customers better. Therefore, in this thesis, we focus on orchestrated multi-tier resource pooling. The main idea is to let a MTC provider group buy TPC services with discounts and jointly schedule the reserved TPC resources and its own MTC resources. Generally speaking, MTC can provide users with cloud services of higher bandwidth and lower delay, but the cost of MTC services might be high; On the other hand, TPC can offer users cheaper cloud services, but mobile users may suffer from higher delay and lower bandwidth. Our method takes advantage of both services and creates a win-win solution for both the MTC provider and its customers.

The system architecture of our solution is illustrated in Figure 1.2. The MTC provider acts as a brokerage to group buy a large chunk of cloud resources at a discounted price offered by TPC services such as Amazon EC2 [1]. The problem is to integrate services of MTC and TPC to benefit both the MTC provider and its customers. In addition, TPC providers may potentially benefit from the solution, since they can group sell their services to more users. The above problem involves higher QoS services with a higher cost in MTC and lower QoS services with a cheaper price in TPC. In this thesis, we aim at developing a mathematical model for MTC brokerage, to reflect the cloud brokerage scheme in which a telecom company can provide its own cloud services as well as third-party cloud services reserved with group buying.

One barrier to push MTC into market is security concern, particularly when different users' data and VMs are deployed on the same physical machine. To address this concern, we focus on security aware provisioning and migration of phone clones over MTC. One type of MTC services is to build phone clones with cloud computing. The concept of phone clones [15] is to build software clones of smart phones on the cloud and enable mobile users to offload computation intensive tasks and backup data

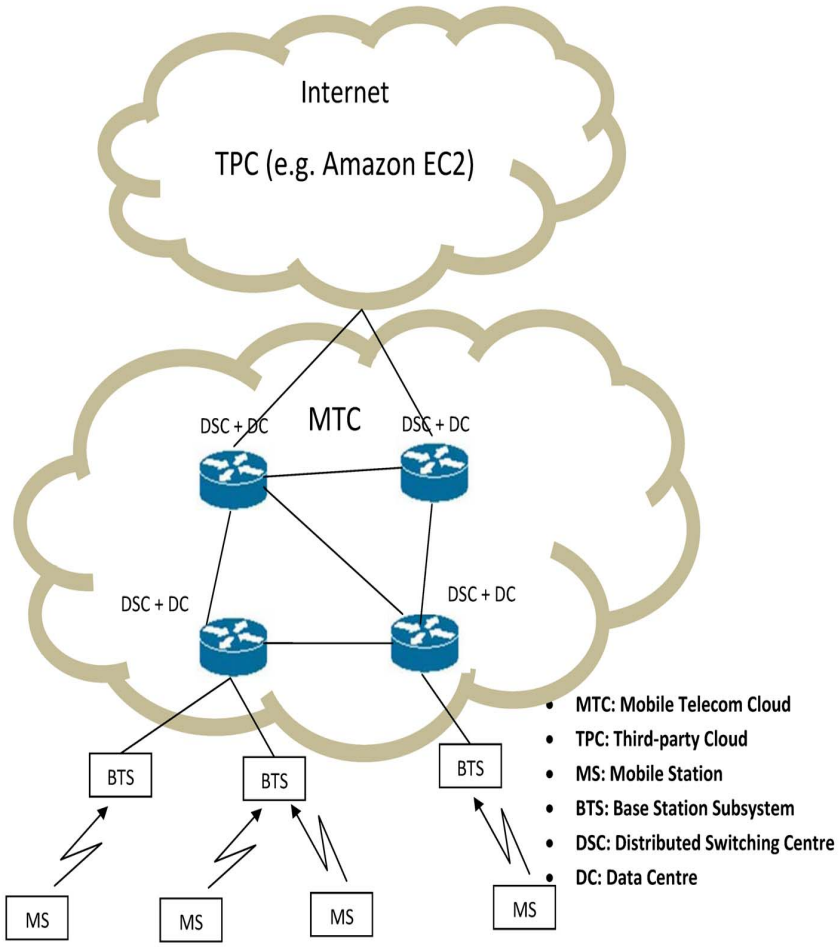

Figure 1.2: System architecture

to the cloud. Depending on different functionalities supported by the phone clones, they could be implemented as a process or a *thin* virtual machine (VM) [15,39,46] on a cloud host. Due to the ease of management and the richer functionalities of VM, we assume the VM version of phone clones in this thesis. To allow resource multiplexing, multiple VMs are usually allocated and managed with a hypervisor, such as KVM [5] or Xen [7], in one physical machine. Due to the large number of mobile users, it is not surprising if one hypervisor hosts hundreds of phone clones. Under such circumstances, provisioning and migration strategies for phone clones become critical to the success of mobile telecom cloud.

It has been shown that a VM can attack another VM on the same host via covert channel attacks [43, 58]. Such attacks exploit the CPU cache or the memory bus in a virtualized environment to steal information from other VMs. It has been demonstrated in [43,58] that covert channel can be effective and very hard to detect. We need to tackle two constraints in the allocation and migration of phone clones: security and resource. To give users' a reasonable sense of security, phone clones should be physically isolated. For example, users should feel more comfortable if their phone clones are collocated with those of their friends rather than strangers. Nevertheless, due to the limited number of physical hosts and the large number of mobile users, it may not be possible to find a good isolation for all phone clones. As a result, a phone clone may have to live together with other strangers' phone clones on the same host. In this thesis, we propose and evaluate SWAP: a security aware provisioning and migration scheme for phone clones. For the provisioning of new phone clones, we take advantage of the mobile telecom cloud where it is easy to build a communication graph based on mobile users' communication history.

1.2 Contributions

The main contributions of this thesis are threefold.

Contribution 1: We solve the data planning problem in MTC:

- We present a mathematical model for data placement problem in MTC that can be used to minimize the system cost and maximize users' satisfaction. With this analytical model, we formulate two optimization problems and present two algorithms, LP based branch-and-bound and LP with rounding, to solve the optimization problems.

- We propose a novel approach by grouping the users based on their similarity in mobility pattern and QoS requirements. We then apply the same analytical model with groups of users as the basic unit. In this way, our solution is capable of handling millions of mobile users.

The above contribution has been published in [49].

Contribution 2: We solve the resource planning in MTC brokerage with orchestrated multi-tier resource pooling:

- To facilitate resource planning, we present a mathematical model for MTC brokerage. The model reflects the cloud brokerage scheme in which a telecom company can provide its own cloud services as well as third-party cloud services reserved with group buying.

- We formulate and solve the optimization problem of minimizing the costs of MTC brokerage. In addition, the solution provides an insight on MTC's profitable price range that the MTC brokerage could possibly charge to its customers.

- To alleviate the (negative) impact of dynamic changes of customers' requests, we present a contract update scheme for the MTC brokerage to make wise decisions based on predicted customers' requests.

- We demonstrate the benefits of our solution through tests using real Google traces collected over 29-day period from a Google cluster containing over 12500 physical machines [42].

The above contribution is currently under submission [50].

Contribution 3: We address security aware provisioning and migration of phone clones in MTC:

- We propose a system model that captures the mobile users' communication relationship and the potential risks when collocating phone clones, and solve the optimization problem that minimizes the risks in the provisioning of new phone clones. The optimization problem requires intensive computations due to the large number of phone clones. To avoid this problem, we present a clique-covering method to pre-process the communication graph and significantly speed up the optimization algorithm.

- We propose a phone clone migration strategy to mitigate the impact of potential covert channels. We introduce a decay function to model the time-varying feature of covert channels, and migrate some phone clones whenever the risk among the phone clones in a host becomes high. In this context, we minimize the total number of migrations to meet a given security requirement.

- We evaluate our solution using Reality Mining dataset collected from 100 mobile phones over the course of 9 months in the Reality Mining project undertaken at the MIT Media Laboratory [20] and Nodobo dataset collected from a group of 27 students at a high school [14].

The above contribution has been published in [51].

1.3 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we review the related work on mobile telecom cloud, cloud brokerage, and security provisioning in mobile cloud. In Chapter 3, we consider the scenario where a MTC is providing only its own resources to the users. Chapter 4 discusses the scenario where a MTC provider acts as a brokerage, called MTC brokerage, to broker TPC cloud resources and integrate the resources reserved from TPC with those of its own. In Chapter 5, we address security aware provisioning and migration of phone clones in MTC. Finally, Chapter 6 concludes the thesis and discusses future work.

Chapter 2

Background and Related Work

In this chapter, we present an overview of related work. First, we discuss mobile cloud computing and previous works in this area. Next, we review the works about cloud brokerage. Finally, we discuss the works that have been done recently in the area of security in mobile cloud.

2.1 Mobile Cloud Computing

Smartphones are becoming more and more powerful every day. We use many applications running on smartphones such as web browsing, gaming, GPS, organizing emails, looking up songs by audio samples, and capturing and editing videos. As the number of running applications increases on a smartphone, execution speed, memory space, and the lifetime of the battery become problematic. Researchers have been investigating mobile cloud as a solution for offloading computation-intensive tasks to the cloud [13, 15, 17].

Considerable research has been conducted on mobile cloud, and prominent among those are the MAUI [17] and the CloneCloud [15] projects. Recent research on mobile cloud have focused on two approaches for remote execution. The first approach partitions a program and offloads part of a code to the cloud [11, 24, 25]. This approach results in energy saving because it offloads CPU-intensive part of the code to the cloud. In the second approach, the entire application will be migrated to the cloud [16, 46]. This approach is easier for the programmer since the application doesn't need to be partitioned to offload to the cloud.

MAUI [17] creates two version of a program on a smartphone. The first version

runs locally on the smartphone, and the other is executed remotely on the cloud. CloneCloud [15] offloads the right portion of the code to the cloud. The goal of CloneCloud is to make the job of programmer easier by making application partitioning automatic and seamless. CloneCloud rewrites an unmodified application, and in the modified version, at chosen points individual threads migrate from the mobile device to a clone in a cloud. Remaining part of the code will be executed in the smartphone. After finishing the execution of thread on the cloud, the state of the remote code returns to the phone, and it merges with the original process.

Satyanarayanan et al. [46] proposes an architecture for augmenting smartphone capabilities. The mobile software can be instantiated on a nearby cloudlet, and mobile users can use the virtual machine (VM) technology and use that service over a wireless LAN. A cloudlet is a trusted computer or cluster of computers, and it is available to nearby mobile users connected to the Internet. The quality of service is increased by using cloudlet because of the cloudlets physical proximity and one-hop network latency. The difference between a cloudlet and cloud is that a cloudlet has decentralized ownership by a local business while cloud has centralized ownership by a company such as Amazon or Google. Another difference is that a cloudlet has LAN latency while the cloud has Internet latency [46].

Gordon et al. [29] presents a method based on replication rather than partitioning which runs the code on both the client and the server. Therefore, it reduces the latency by predicting which replica will be faster. Balan et al. [10] proposes cyber foraging which dynamically augment the computing resources of a wireless mobile device by exploiting nearby compute and data staging servers. For example, public spaces such as airport lounges and coffee shops will be equipped with computing and data servers, and they can be used to improve the performance of applications on mobile clients.

2.1.1 Mobile Telecom Cloud (MTC)

Mobile cloud services can be provided either by third-party cloud providers such as Amazon EC2 [1] or by the wireless service providers such as telecom companies. Cloud computing provides mobile telecommunication companies with new opportunities to augment their services to mobile users with Mobile Telecom Cloud (MTC), which refers to cloud services supported by mobile telecommunication companies. Because mobile communications companies already have the required physical system

and networking infrastructure in place, and also because they provide the last-mile Internet access to mobile users, they have advantages over other cloud providers in offering cloud services with better quality of service (QoS).

In this thesis, we focus on mobile telecom cloud as telecom companies can take advantage of their existing infrastructures for a quick start and compete effectively with other third party cloud service providers [18, 34]. Moreover, telecom companies are in a better position to manage and upgrade the network and provide high quality of service as they own the transmission network. Thus, telecom companies are able to provide service level agreements with higher reliability [38] and configure networks with end-to-end bandwidth and latency guarantees for cloud services [53].

Telecom companies can play different roles in offering cloud services to the mobile users. One of the main roles of telecom companies is to provide users with their required connectivity and quality of service. Operators can provide connectivity between mobile users and third-party cloud providers. Another role of telecom companies can be delivering and reselling third-party cloud to the users. In this scenario, telecom company can burst the cloud onto a third party with growth of demand [22].

2.2 Cloud Brokerage

Cloud brokerage [26, 30, 47, 55, 62] offers great opportunities for consumers to find the best services and best price, which however raises new challenges on how to select the best service out of the huge pool. Sundareswaran et al. [47] propose a novel brokerage-based architecture in the cloud, where the cloud brokers are responsible for the service selection. Cloud brokerage can greatly benefit both service providers and customers, since it is hard for customers to collect necessary information and analyze all service providers in order to make a right decision. As such, cloud service brokerage has been identified as a key component for future cloud technology development and research [26, 47, 55].

Number of cloud users has grown significantly while they have different service level requests. With the growth of cloud computing, many cloud service providers companies are offering cloud services to cloud users. Therefore, there is a need to develop a mechanism for managing the cloud resources to meet the users' cloud demands. Cloud brokerage companies can manage users' cloud requests and providers' cloud services. Cloud service brokers play different roles in managing the cloud requests such as Aggregation, Integration, and Customization [54]. First, a cloud aggre-

gator is a type of cloud broker that packages multiple cloud computing services into one composite service. Therefore, the client is using multiple services from different cloud providers through a cloud broker and pays the bill to the cloud broker. Second, a cloud integrator merges different services from different cloud providers and provide a new workflow. Third, a cloud customizer changes the existing cloud services and create an extension to them [54].

The number of cloud brokerage companies is increasing everyday. For example, CloudMore [4] manages multiple cloud services from multiple providers and offers cloud service aggregation through partners. The company works in different countries such as USA, UK, Sweden, Finland, and Norway. Main partners of CloudMore include IBM, Microsoft, and VMWare. CloudMore claims to serve users in over 9 countries and work with 1,000 resellers. Appirio [2] helps companies power their business with the cloud. Appirio has partnerships with different cloud providers and provides cloud service from companies such as salesforce.com, Google, Workday and Cornerstone OnDemand. AWS Marketplace [3] is an online store where users can sell or buy software that runs on Amazon Web Services (AWS). As subscribers, users can buy software through AWS Marketplace, and as sellers, users can sell Amazon EC2 instances on AWS Marketplace.

Most of third-party cloud providers offer different pricing options to the users such as accessing the cloud with on-demand pricing or reserved instances. With the option of reserved instances, users are offered different discount levels. Cloud users usually pay the cloud provider with a pay-per-usage scheme, however the option of cloud reserved instances can provide the users with a huge discount. By reserving an instance from the cloud provider, users have access to the cloud during the term of contract. During this time period, the users have access to the cloud for a long period with a significant discount. The benefit of a cloud user from buying a reserved instance depends on the usage pattern. Due to the prepayment of reservation fees, the more a cloud user uses the reserved instance the more the benefits. On the other hand, on-demand instances are inefficient for the users because of its higher rates. For example, Amazon EC2 [1] charges the users based on running hours. It means that if a user with on-demand pricing runs an instance for less than an hour, the user will be charged for a full hour.

Cloud brokerage allows users with arbitrary demand pattern to benefit from cloud reserved instances even if users do not fully utilize the cloud. Therefore, a cloud broker can reserve a huge number of instances from a third-party cloud provider

with a discount. In this scheme, the users will buy the cloud from the cloud broker instead of buying directly from a cloud provider. Therefore, users with on-demand requests can benefit from reserved instance discounts without worrying about long time period reserved instances. The cloud broker gathers users cloud requests and aggregates them in one reserved instance. Therefore, many cloud users can benefit from a broker's reserved instance. Another benefit of the users from cloud brokerage is that cloud providers offer volume discounts to cloud brokers who buy large number of reserved instances [1]. Therefore, with the volume discount offered to a cloud broker, the users' cloud plan prices are reduced significantly [55].

Wang et al. [55,56] propose a new cloud brokerage service that reserves a large pool of instances from cloud providers and serves users with price discounts. The broker exploits both pricing benefits of long-term instance reservations and multiplexing gains. Most IaaS (Infrastructure as a Service) clouds offer significant volume discounts to those who have purchased a large number of instances. By aggregating the users' cloud demands, the cloud broker can easily qualify for such discounts, which in turn reduces the cost for the users. Nevertheless, MTC brokerage differs from traditional ones in that a MTC brokerage has its own cloud resources, which should be jointly considered to maximize the brokerage's profit.

The problem studied in the paper is similar to that in cloud bursting, a service deployment method in which an application runs in private cloud and bursts into a public cloud when the resource requests spike [32]. Nevertheless, our problem involves tiered discounts of TPC. In particular, a higher discount is given to a larger commitment. Resource scheduling in this context is different from that in cloud bursting.

2.3 Security in Mobile Cloud

It has been shown that a VM can attach another VM on the same host via covert channel attacks [43, 58]. Such attacks exploit the CPU cache or the memory bus in a virtualized environment to steal information from other VMs. In this section, we review previous works about threats of cross-VM covert channel attacks in the cloud. Then, we discuss the differences between our method and recent works about cross-VM covert channel attacks.

Cloud computing provides users with abundant computing resources, however, the drawback is its security concerns. The confidentiality of client data in the cloud

is of utmost concern. Tysowski [48] discusses the security challenge in mobile cloud computing, and Khan et al. [35,36] propose a security scheme for mobile users in cloud environment to protect mobile users' identity. A mobile user's identity is identified in the cloud through different methods such as password protection. Once an adversary hacks in the system, he can steal information from users' cloud data by stealing their identities. Virtualization technologies, such as Xen [7], Linux KVM [5] and VMWare [6], multiplex the physical resources of cloud servers between different users, and physical isolation between the mobile devices may be breached.

Cloud customers need to trust cloud providers when offloading task from mobile phones to the cloud. However, in the cloud infrastructure, there is a threat from other customers because many resources are shared among customers' VMs. For example, in a multi-tenancy cloud environment, on a single physical server, many VMs may be collocated. Therefore, an adversary may penetrate the target VM via side channels between VMs and steal information. Ristenpart et al. [43] explores the possibility of cross-VM attacks in existing third-party clouds such as Amazon EC2 [1]. Many recent research works [57, 58, 60] have discussed different types of cross-VM covert channel attacks. It has been shown that sensitive information such as cryptographic keys can be leaked out through shared caches. This will result in leakage of users' sensitive information, and users do not have any control over it [57].

Covert channel attacks are discussed in classic security research articles such as U.S. NCSC report [28]. Cross-VM covert channel attacks are concerns for cloud users. There are different methods against Cross-VM covert channel attacks that can be categorized into architectural, monitoring, or fuzzy time based methods. In architectural methods, system components are modified to reduce the security risks. For example, Kadloor et al. [33] modifies the resource scheduler and designs a scheduling policy which guarantees a desired degree of privacy. Monitoring based methods insert additional components into the system to monitor malicious activity to detect covert channel attacks [45, 63]. Fuzzy time methods weaken malicious virtual machines' ability to receive the signal by eliminating fine-grained timers [52].

Xu et al. [59] introduces controlled-channel attacks, a new type of side-channel attack and implements these attacks. In these attacks, the untrusted operating system uses its control over the platform to construct powerful side channel attacks. Zhang et al. [64] proposes an access-driven side channel attack by which a malicious virtual machine (VM) extracts fine-grained information from a victim VM running on the same host. They address these challenges and demonstrate the attack in a lab setting.

Kim et al. [37] presents a system-level protection mechanism against cache-based side channel attacks in the cloud.The hypervisor provides each VM with small amounts of memory that is largely free from cache-based side channels. Therefore, each VM can use its own special pages to store sensitive data without revealing its usage patterns. Xu et al. [61] explores L2 cache covert channels and demonstrates a covert channel with considerably higher bit rate.

Jaeger et al. [31] discusses the covert channel attacks and propose a method for managing the risk of such information flows using the risk flow policy, and Zhang et al [65] propose a game strategy by periodically migrating VMs, making it difficult for adversary to locate the target VMs. Most of the recent researches about covert channel attacks induce deployment cost, and they are specific to one type of covert channel attacks. However, in this thesis, we use the knowledge about cloud tenants to propose a general method for security in mobile cloud.

Chapter 3

Optimal Data Placement in Mobile Telecom Cloud

3.1 Motivation

Smart phones have become a part of our daily life. While smart phones become more and more powerful, their processing speed and storage capacity are still limited, compared to servers and desktop computers. Cloud computing eliminates this limitation and presents unprecedented opportunities to enhance mobile users' computing capability and support ubiquitous access to a large amount of data. We call cloud services dedicated to mobile users as mobile cloud.

Mobile cloud services could be provided by third-party cloud providers such as Apple's iCloud, yet mobile communications companies should fit the role better. When mobile cloud services are provided and supported by mobile communications companies, we call this type of cloud computing mobile telecom cloud (MTC). Because mobile communications companies already have the required physical system and networking infrastructure in place and also because they provide the last-mile Internet access to mobile users, they have direct knowledge of mobile users' mobility patterns, data access, and delay requirements. With this information, MTC can strategically distribute users' data in Distributed Switching Centres (DSCs). This grants MTC a unique advantage over third-party mobile cloud.

MTC poses new opportunities as well as challenges to mobile communications companies. On one hand, mobile users want to fully utilize mobile cloud to store and process a large volume of data. This trend results in a substantial traffic load

which may interfere with other existing voice and data services. On the other hand, mobile communication companies need to protect their network infrastructure from potential congestion caused by cloud traffic. One way to solve this dilemma is to become location aware, i.e., MTC providers should try to constrain cloud traffic locally to avoid congestion in their backbone network. This motivates our research in this chapter.

Telecom companies can provide users with mobile connectivity, and users can thus ubiquitously access the cloud data centres. Intuitively, a customer's data should be placed at the locations where the telecom company has guaranteed resources and can directly control them to meet the QoS requirements. DSCs are the best place to host data centres (DCs) and users' data. At the first look, the idea of localizing a mobile users' data in a DSC seems trivial. As the number of users grows, however, this simple idea raises several challenges.

First, with a large number of users, using a simple greedy algorithms tend to place a heavy load on certain data centres and cause congestion in these data centres. We hence need to formally formulate and solve the optimal data placement problem in a systematic manner. Second, we need to consider the benefits to both the service provider and the customers. To be specific, we need to minimize the system cost of service provider while maximizing customers' satisfaction. Third, data placement is a type of network planning problem, and when the number of users is large (e.g., millions), deriving a globally optimal solution may be computationally intractable. We thus need to find a scalable solution.

In this chapter, we address the above challenges and make the following contributions:

- We present a mathematical model for data placement problem in MTC that can be used to minimize the system cost and maximize users' satisfaction. With this analytical model, we formulate two optimization problems and present two algorithms, LP based branch-and-bound and LP with rounding, to solve the optimization problems.

- We propose a novel approach by grouping the users based on their similarity in mobility pattern and QoS requirements. We then apply the same analytical model with groups of users as the basic unit. In this way, our solution is capable of handling millions of mobile users.

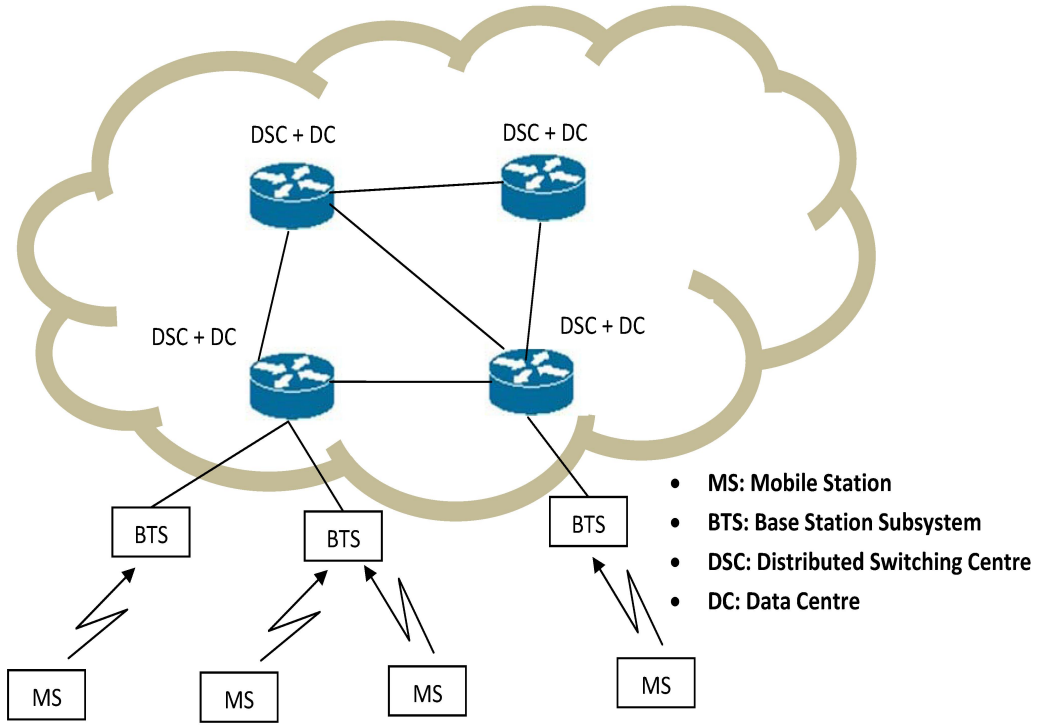

Figure 3.1: Telecom Distributed Cloud-Based Data Centres

The rest of this chapter is organized as follows. Subsection 3.2 describes the system model of MTC, including three components: network model, user model, and costs model. In Subsection 3.3, we formulate the optimization problem of minimizing the operating cost and maximizing users' satisfaction in MTC and present two solutions. Subsection 3.4 presents a cluster-based method to group users into clusters. In Subsection 3.5, we evaluate the efficacy of our design.

3.2 System Model of Mobile Cloud

3.2.1 Network Model

Figure 3.1 shows the architecture of network and switching subsystems of distributed data centres in MTC. In this model, users are mobile, and they can connect to the cloud as long as they can communicate with the telecom service provider. The telecom company provides users with access to the data centres (DC) through a location-independent process, i.e., the users just access the service without need to

know where the data is located. In this model, we assume that data centres are co-located with the Distributed Switching Centres (DSC). In the rest of the chapter, we thus use DSC and DC *interchangeably*.

We assume that there are m users and n DSCs in the system. We model the network as a graph $G =< V, E >$, where V is the set of nodes representing the DSCs and E is the set of edges representing the links between DSCs. For each link, we consider its delay and bandwidth. When there is a link between DSC i and DSC j, we set its bandwidth as b_{ij} and its delay as d_{ij}. For any DSC i, we assume $b_{ii} = \infty$ and $d_{ii} = 0$. We use a matrix $B = [b_{ij}]_{n\times n}$ to record the bandwidth values and a matrix $D = [d_{ij}]_{n\times n}$ to record the delay values. For ease of reference, notations used in this chapter are listed in Table 3.1.

Remark 1. *We treat the data placement problem as a network planning problem, and as such, we do not consider the dynamic changes in network delay and network bandwidth in this chapter. Such obliviousness makes sense and has been common in network planning, and this is because the static network delay and bandwidth are considered as the approximate estimation of network condition in the long term.*

3.2.2 User Model

Service Level Agreements

Telecom companies can provide users with required QoS, usually defined in SLAs. In this chapter, we consider the SLAs that contain user's bandwidth and delay requirements.

Assume that we record users' required bandwidth in an $m \times 1$ vector, denoted by SLA_b, where $SLA\_b[i]$ contains the bandwidth requirement by user i. To find the suitable placement for a user's data, we use the matrix B and the vector SLA_b to filter the links in the network that cannot be used by the user. For this purpose, we define a bandwidth filter function, denoted by $BF(x)$, where x is the required bandwidth. The function $BF(x)$ returns a $n \times n$ matrix of binary values, in which 1 indicates that the corresponding link has bandwidth no less than x and 0 otherwise. For example, as shown in Figure 3.2, if the bandwidth requirement of user i is 3 Mbps, the function $BF(3)$ returns a $n \times n$ binary matrix, where 1 indicates that the corresponding link could be used by user i and 0 otherwise.

Assume that we record the maximum delay that users can tolerate in an $m \times 1$ vector SLA_d, where $SLA\_d[i]$ contains the maximum delay that user i can tolerate.

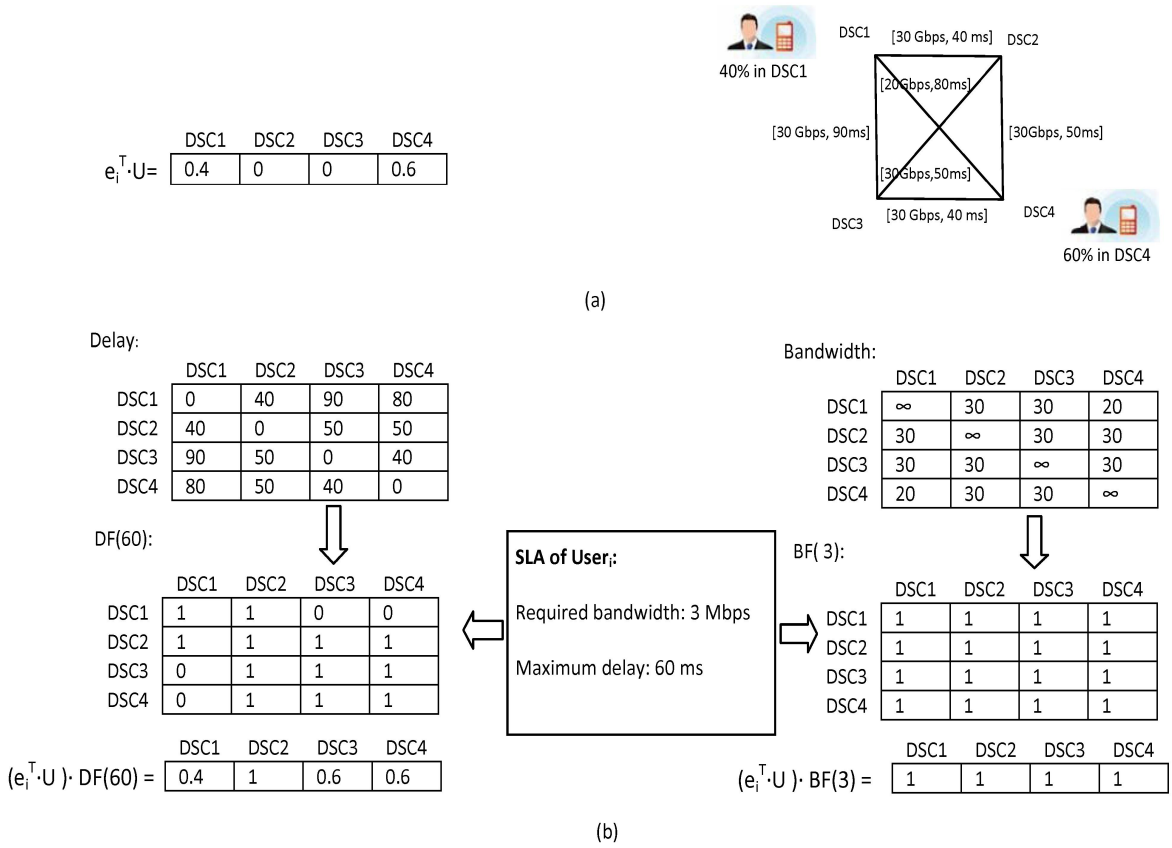

Figure 3.2: (a) An example of network model and user model (b) Calculation for finding the optimal data placement based on user's SLA

Similar to the pre-processing of bandwidth requirement, we use the matrix D and the vector SLA_d to filter the links in the network that cannot be used by the user. We define a delay filter function, denoted by $DF(x)$, where x is the delay that the user can tolerate. The function $DF(x)$ returns a $n \times n$ matrix of binary values, in which 1 indicates that the corresponding link has a delay no larger than x and 0 otherwise. As an example shown in Figure 3.2, if the delay tolerance of user i is 60 ms, the function $DF(60)$ returns a $n \times n$ binary matrix, where 1 indicates that the corresponding link could be used by user i and 0 otherwise.

Remark 2. *In a cellular network, users' SLA contain information about the external delay and bandwidth (the delay and the bandwidth between the users' mobile device and the nearest DSC) and internal delay and bandwidth (the delay and the bandwidth*

between the nearest DSC and the DSC that contains users' data). The telecom service provider can control the external bandwidth by setting the bandwidth of the link between the mobile users and the DSC nearest to the users. Therefore, in this chapter, we focus on the internal SLA which has a higher impact on users' QoS satisfaction.

Data Access Profile

As part of their core services, telecom companies can provide users with mobile connectivity. Users can thus ubiquitously access the cloud data centres. As shown in Figure 3.1, mobile users connect to their nearest DSC. After connecting to the nearest DSC, their request is routed to the data centre that contains their data. Usually, as users move, they connect to different DSCs; however, providing the required QoS regulated in SLAs is of crucial importance. To search for the optimal placement of a user's data, we need to know the pattern that a user connects to DSCs. In the long run, such a pattern could be obtained with statistical test over the user's historical access times.

We use an $m \times n$ matrix $U = [u_{ij}]$ to record the users' long-term data access pattern, where u_{ij} represents the percentage of time that user i connects to DSC j. Note that for any $i = 1, 2, \ldots, m, \sum_{j=1}^{n} u_{ij} = 1$. Suppose e_i is a $m \times 1$ column vector that has a 1 for its i-th component and zero everywhere else. Therefore the vector $e_i^T \cdot U$ denotes the i-th row vector in matrix U. We use an $m \times 1$ vector a to record users' maximum data amounts, where $a[i]$ represents the amount of data of user i.

Following the common practice, we assume that the arrivals of users' data access requests follow a Poisson process with mean rate λ. To ease analysis, we normalize the value of λ by the total number of users in the system so that $\lambda \in (0, 1]$.

3.2.3 Operating Costs of MTC

Mobil users connect to cloud as they connect to their nearest DSC. A telecom company makes decision about best user data placement based on many parameters such as SLAs and different costs. We only consider the maintenance cost and electricity cost in different DSCs. Referring to Remark 1, we assume a static cost model, which makes statistical sense in the long term.

We denote the average maintenance cost per data unit in DSC i as $c_m[i]$, and denote the average electricity cost per data unit in DSC i as $c_e[i]$. The costs could be calculated as dollars per Gbps per month with historical operating data. Therefore,

Table 3.1: Notations

Notation	**Definition**
m	Number of users
n	Number of DSCs
b_{ij}	Bandwidth between DSCs i and j
d_{ij}	Delay between DSCs i and j
u_{ij}	Percentage of time that user i connects to DSC j
x_{ij}	A binary variable (1 if user i's data is put in DSC j, and 0 otherwise)
B	$n \times n$ bandwidth matrix $[b_{ij}]$
D	$n \times n$ delay matrix $[d_{ij}]$
U	$m \times n$ matrix $[u_{ij}]$
X	$m \times n$ matrix $[x_{ij}]$
a	Column vector where $a[i]$ represents the amount of data of user i
$SLA\_b[i]$	Bandwidth required by user i
$SLA\_d[i]$	Maximum delay that user i can tolerate
e_i	$m \times 1$ column vector that has a 1 for its i-th component and zero everywhere else.
λ	Normalized Poisson arrival rate
c_m	Column vector where $c_m[i]$ represents the avg. cost of maintenance per data unit in DSC i
c_e	Column vector where $c_e[i]$ represents the avg. cost of electricity per data unit in DSC i
c	Column vector where $c[i]$ represents the avg. total cost per data unit in DSC i
$M[i]$	Maximum storage capacity of DSC i
α	The parameter used for making a trade off between costs and users' satisfaction
BT	Users' satisfaction threshold in bandwidth requirements
DT	Users' satisfaction threshold in delay requirements
C	Maximum operating cost of all data centres

the average overall cost per data unit in DSC i could be calculated as $c[i] = c_m[i]+c_e[i]$. We use column vectors c, c_m, c_e to store the values of $c[i], c_m[i], c_e[i]$, respectively.

3.3 Optimal Data Placement in MTC

3.3.1 Minimizing the Operating Costs in MTC

In this section, we formulate the optimal data placement problem in MTC for minimizing the operating costs: how to deploy users' data in the best locations such that the users' SLAs can be satisfied and the overall operating cost of MTC is minimized?

We consider the following optimization problem:

$$\min_{X=[x_{ij}]_{m\times n}} \quad a^T X c \tag{3.1}$$

subject to

$$x_{ij} \in \{0,1\} \tag{3.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \text{ for } i = 1,2,\ldots,m \tag{3.3}$$

$$\sum_{i=1}^{m} x_{ij} a[i] \leq M[j], \text{ for } j = 1,2,\ldots,n \tag{3.4}$$

$$(e_i^T \cdot U) \cdot BF(SLA\_b[i]) \cdot (X^T \cdot e_i) = 1, \text{ for } i = 1,2,\ldots,m \tag{3.5}$$

$$(e_i^T \cdot U) \cdot DF(SLA\_d[i]) \cdot (X^T \cdot e_i) = 1, \text{ for } i = 1,2,\ldots,m \tag{3.6}$$

$$\lambda(U^T X + X^T U) \leq B \tag{3.7}$$

Equation (3.1) is the objective function. In this equation, a^T is a $1 \times m$ row vector with its i-th item recording the data amount of user i; $X = [x_{ij}]$ is an $m \times n$ allocation matrix where $x_{ij} = 1$ indicates that the data of user i is assigned to DSC j; c is a $n \times 1$ column vector with its i-th item recording the operating cost per data unit in data centre i. The objective is thus to minimize the total operating cost of all data centres.

Equations (3.2) and (3.3) together assume that a user's data is allocated to one DSC and is not split among multiple DSCs. We will discuss the relaxation of this requirement later. Inequality (3.4) indicates that the total amount of data assigned to a DSC should be less than its maximum capacity.

Equation (3.5) means that users' bandwidth requirements are satisfied. As shown in Figure 3.2, $e_i^T \cdot U$ is a $1 \times n$ row vector with its j-th value representing the proportion of time that user i accesses DSC j (in a long term), e.g., user i accesses DSC 1 for 40% of times and DSC 4 for 60% of times. $SLA\_b[i]$ records the bandwidth requirement of user i (3 Mbps in the example), and the function $BF(SLA\_b[i])$ returns a $n \times n$ matrix of binary values, in which 1 indicates that the corresponding link has bandwidth no less than $SLA\_b[i]$ and 0 otherwise. Therefore, the product, $(e_i^T \cdot U) \cdot BF(SLA\_b[i])$, returns a $1 \times n$ row vector with its j-th value indicating the percentage of time that user i's bandwidth requirement is satisfied if its data were assigned to DSC j. Since $X^T \cdot e_i$ is a $n \times 1$ binary column vector representing the data allocation of user i, the constraint $(e_i^T \cdot U) \cdot BF(SLA\_b[i]) \cdot (X^T \cdot e_i) = 1$ means that the bandwidth requirement of user i is always satisfied. Similar to Equation (3.5), Equation (3.6) means that users' delay requirements are satisfied.

Inequality (3.7) ensures that after data placement, the expected total bandwidth requirement on each link is less than the capacity of the link. Note that $\leq$ means element-wise comparison between two $n \times n$ matrices.

3.3.2 Maximizing Users' Satisfaction in MTC

In this section, we try to maximize users' QoS satisfaction within cost constraints in MTC. For this purpose, we consider the following optimization problem:

$$\underset{X=[x_{ij}]_{m\times n}}{\text{Min}} \quad -\frac{1}{2m}(\sum_{i=1}^{m}(e_i^T \cdot U) \cdot BF(SLA\_b[i]) \cdot (X^T \cdot e_i) + \sum_{i=1}^{m}(e_i^T \cdot U) \cdot DF(SLA\_d[i]) \cdot (X^T \cdot e_i)) \tag{3.8}$$

subject to

$$x_{ij} \in \{0, 1\} \tag{3.9}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \text{ for } i = 1, 2, \ldots, m \tag{3.10}$$

$$\sum_{i=1}^{m} x_{ij} a[i] \leq M[j], \text{ for } j = 1, 2, \ldots, n \tag{3.11}$$

$$(e_i^T \cdot U) \cdot BF(SLA\_b[i]) \cdot (X^T \cdot e_i) \geq BT, \text{ for } i = 1, 2, \ldots, m \tag{3.12}$$

$$(e_i^T \cdot U) \cdot DF(SLA\_d[i]) \cdot (X^T \cdot e_i) \geq DT, \text{ for } i = 1, 2, \ldots, m \tag{3.13}$$

$$a^T X c \leq C, \tag{3.14}$$

$$\lambda(U^T X + X^T U) \leq B \tag{3.15}$$

where $X = [x_{ij}]$ is an $m \times n$ allocation matrix where $x_{ij} = 1$ indicates the data of user i is assigned to DSC j and 0 otherwise. Equation (3.8) is the objective function to measure the overall QoS satisfaction in the network. Since we want to maximize users' satisfaction, the equation is multiplied by -1 to turn the problem into a minimization problem.

Equation (3.9) and Equation (3.10) together assume that a user's data is allocated to one DSC and is not split among multiple DSCs. Inequality (3.11) indicates that the total amount of data assigned to a DSC should be less than its maximum capacity. Inequality (3.12) means that users' satisfaction in bandwidth requirements should be more than the threshold BT. Similarly, inequality (3.13) means that users' satisfaction in delay requirements should be more than the threshold DT. Inequality (3.14) indicates that the total operating cost of all data centres should be less than the maximum value C, and inequality (3.15) ensures that after data placement, the expected total bandwidth requirement on each link is less than the capacity of the

link.

Remark 3. *We assume that a user's data is allocated to only one DSC and is not split among multiple DSCs. Of course, we can relax this requirement to allow x_{ij} to be real numbers. The optimization problem would be easy to solve with linear programming. Nevertheless, we would need more information regarding users' data access pattern, including, for example, which part of the data a user needs to access at what time. This implies a large overhead in data collection and analysis. We could also allow different copies of a user's data in multiple DSCs. This makes the network planning easy, but the cost on data storage and maintaining the data consistence would be non-trivial. To summarize, there is no data placement solution that fit all situations. As such we place the constraints of Equation (3.9) and Equation (3.10) to ease network planning in practice.*

3.3.3 Two algorithms for Solving the Optimization Problem

In this section, we present two algorithms, linear programming (LP) based branch-and-bound and LP with rounding, for solving the optimization problems mentioned in Sections 3.3.1 and 3.3.2. The above optimization problems can be readily solved with the LP based branch-and-bound algorithm, which sets the values of all variables to 0 or 1. This method, however, does not scale to large networks as its memory usage and running speed increase drastically with the number of variables.

To avoid this problem, we introduce an approximation algorithm based on linear programming combined with the rounding technique. The basic idea is to relax the requirement (3.9) to allow x_{ij} being real numbers; we then solve the relaxed problem using linear programming. For the x_{ij} values that are either close to 1 or close to 0, we round them up to 1 or down to 0, respectively. In the next round of iteration, we fix those rounded values, and perform the same procedure again until the convergence condition is met.

Figure 3.3 shows the algorithm for solving the optimization problem by using LP with rounding. In this algorithm, first we set the value of ϵ to a small value. Then, we run the optimization problem and solve the relaxed optimization problem for unresolved x_{ij} using linear programming. For each x_{ij}, if x_{ij} is less than ϵ and no constraint would be violated if x_{ij} was changed to 0, we fix the value of $x_{ij} = 0$. If $x_{ij} \geq 1 - \epsilon$ and no constraint would be violated if x_{ij} was changed to 1, we fix the

```
1: ε = 0.01
2: run_optimization = true
3: solved_num = 0
4: while true do
5:     if run_optimization == true then
6:         Solve the relaxed optimization problem (i.e., the problem permitting x_ij
   to be real numbers) for unresolved x_ij using linear programming
7:     end if
8:     previous_solved_num = solved_num
9:     for i ← 1 to m do
10:        for j ← 1 to n do
11:            if x_ij is not fixed then
12:                if (x_ij ≤ ε) and (change in x_ij = 0 does not violate any constraints)
   then
13:                    Fix the value of x_ij = 0
14:                    solved_num = solved_num + 1
15:                else if (x_ij ≥ 1 − ε) and (change in x_ij = 1 does not violate any
   constraints) then
16:                    Fix the value of x_ij = 1
17:                    solved_num = solved_num + 1
18:                end if
19:            end if
20:        end for
21:    end for
22:    if solved_num == m ∗ n then
23:        Exit
24:    end if
25:    if previous_solved_num == solved_num then
26:        run_optimization = false
27:        ε = ε + 0.001
28:    else
29:        run_optimization = true
30:    end if
31: end while
```

Figure 3.3: Solving the optimization problem by using LP with rounding

value of $x_{ij} = 1$. If in one iteration, the algorithm does not fix the value of any x_{ij}, we increase the value of ϵ and perform the rounding with the value of new ϵ. Otherwise, we perform the optimization again until the value of all variables are set.

3.3.4 Further Discussion

While providing users with their required QoS is of utmost importance, cloud providers should strive for reducing costs. There is a clear trade off between enhancing users' QoS and reducing the costs of the cloud provider. In this section, we introduce a mechanism so that such a trade off could be easily determined.

To measure users' satisfaction, we considered users' required bandwidth and delay during their access time. The average users' satisfaction can be measured with Equation (3.8). In the meantime, the cost constraint is given by inequality (3.14). If the telecom cloud provider wants to investigate the trade off between users' satisfaction and the corresponding minimal costs incurred, it can use a weighted sum of users' satisfaction and costs by setting the parameter α in the following objective function:

$$\begin{aligned}
&\text{Min}\, \alpha(-\frac{1}{2m}(\sum_{i=1}^{m}(e_i^T \cdot U) \cdot BF(SLA_b[i]) \cdot \\
&(X^T \cdot e_i) + \sum_{i=1}^{m}(e_i^T \cdot U) \cdot DF(SLA_d[i]) \cdot (X^T \cdot e_i))) \\
&+ (1-\alpha)\frac{a^T X c}{C} \qquad (3.16) \\
&\text{subject to Equations (3.9)-(3.15)}
\end{aligned}$$

By setting the parameter α in the above objective functions, cloud provider can easily make a trade off between users' satisfaction and costs.

In previous sections, we have discussed data placement of individual mobile users. However, in a real MTC scenario, there may be millions of mobile users. Therefore, there is a need for data placement algorithms that are scalable to millions of mobile users. In the next section, we introduce an algorithm for data placement that uses the similarity between users' data access patterns and clusters the users in order to make the algorithm scalable in MTC.

3.4 A Cluster-Based Solution

In this section, we introduce an algorithm for managing the data for a large number of mobile users in MTC. Since there are similarities between users' data access profiles, we can cluster the users with similar access patterns and treat the users in the same cluster the same way. For this purpose, we first define the similarity distance between the data access pattern of user i and user j as $dist(i,j) = \left\|(e_i^T \cdot U) - (e_j^T \cdot U)\right\|_2$, where $\|\cdot\|_2$ is the vector 2-norm. Then, we model the similarity between users' data access pattern as a graph $H =< V, E >$, where V is the set of all users, and E is the set of edges representing the similarity between data access pattern of two users. We assume that there is an edge between user i and user j if $dist(i,j)$ is less than a given threshold.

In the rest of the chapter, we use cliques and clusters interchangeably. We partition the graph H into cliques. Since the problem of covering a graph with the minimum number of cliques is proven NP-complete and even does not allow constant approximation [66], we adopt the following well-known heuristic method to obtain an approximate solution: We iteratively search for cliques that cover more nodes that have not been covered so far. Heuristically, the nodes with larger degrees may have a better chance of appearing in larger cliques. Thus, the search starts from the node with the highest degree, until all nodes are covered. The use of the above procedure may result in some clusters which are too big for a data centre. Since the users in the same cluster have similar data access profile, we split the large clusters, if any, into smaller ones by random partition.

Suppose that in the end, we obtain p clusters, denoted as H_1, H_2, ..., H_p. We can then apply the same analytical framework developed in previous sections, with clusters as the basic allocation unit, for data placement. For this purpose, we need to estimate the QoS requirement for the group of users in a cluster. Since we assume that the arrivals of requests for mobile users follow a Poisson process, the expected bandwidth requirement of the cluster H_i is $\lambda \sum_{j \in H_i} SLA\_b[j]$. The delay requirement of the group of users can be approximated as the average delay requirements of users in the cluster.

Remark 4. *We point out that there are potentially many other ways to cluster the users, e.g., by considering the bandwidth and delay requirements in the clustering criteria. In addition, there are other ways to decide the QoS requirements for a group of users. For example, we can use the minimum delay requirement in a group, i.e., we*

could over-engineer the system to meet all the delay requirements in the group. Nevertheless, the main purpose of this section is to demonstrate the potential of cluster-based solution to handle the optimal network planning problem for a large number of users. To save space, we leave further investigation of various clustering methods and their impact as future work.

3.5 Performance Evaluation

In this section, we test the two solutions, i.e., LP based branch-and-bound and LP with rounding, to the problem of minimizing the operating costs and maximizing users' satisfaction. To make the algorithm scalable, we use the cluster-based solution introduced in Section 3.4. Different parameters have been used to test the solutions. To evaluate the performance, we test the algorithms with a simulated network. The simulation parameters are listed in Table 3.2.

For each simulation scenario, twenty runs with different random seeds were conducted and the results were averaged. We calculated the confidence interval if large variance among multiple runs was observed. When confidence intervals were calculated, the confidence levels were set to 95%. In the following figures, confidence intervals of simulation results are shown by the vertical bars.

3.5.1 Performance Evaluation of Minimizing Operating Costs

In this section, we solve the optimization problem of minimizing costs using the LP-based branch-and-bound algorithm and LP with rounding. Since the results are similar for these two LP methods, we just show the result for one of them with the name 'Optimization method' in the diagrams. For comparison, we implement and test other two naive algorithms:

1. Probabilistic method: based on a user's data access profile, this method allocates the user's data to a DSC with the probability proportional to the frequency that she/he visits the DSC.

2. Most frequent method: based on a user's data access profile, this algorithm allocates a user's data to the DSC that she/he visits most frequently.

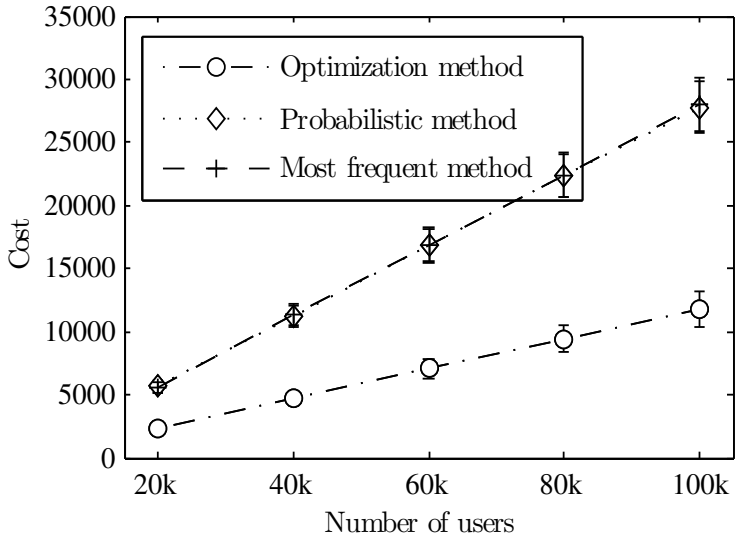

Figure 3.4: Comparison of costs of different methods

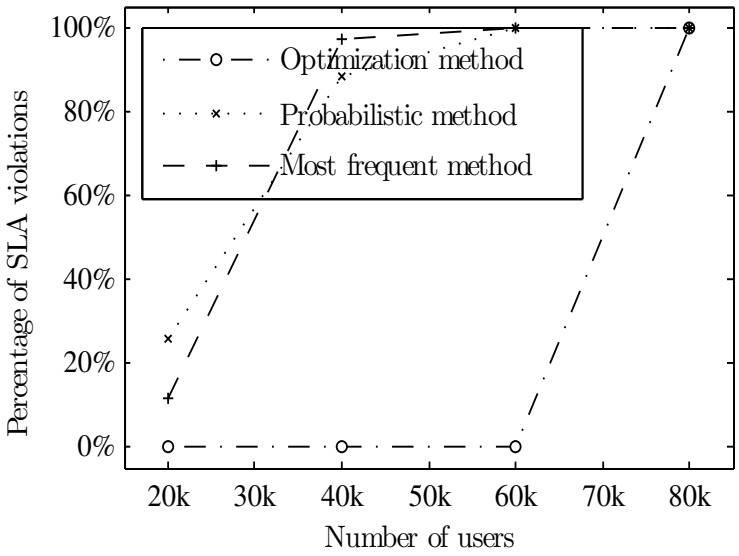

Figure 3.5: Comparison of percentage of SLA violations in different methods

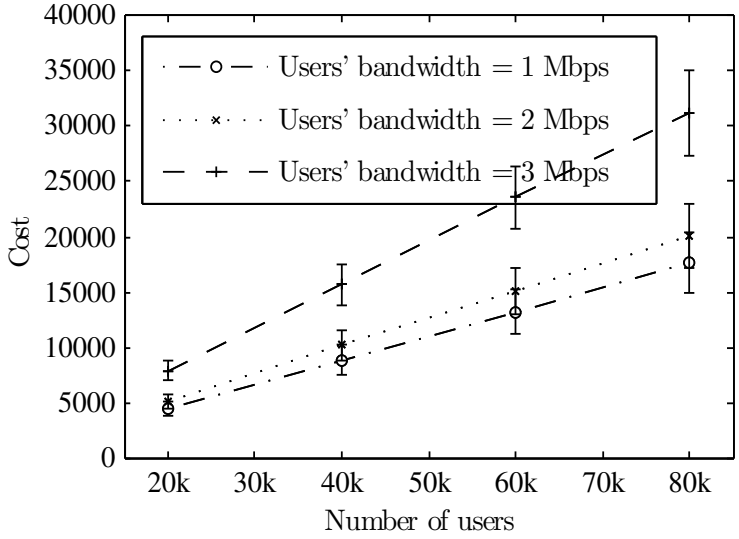

Figure 3.6: Comparison of costs with different bandwidths in users' SLA, users' average delay in SLA=80 ms

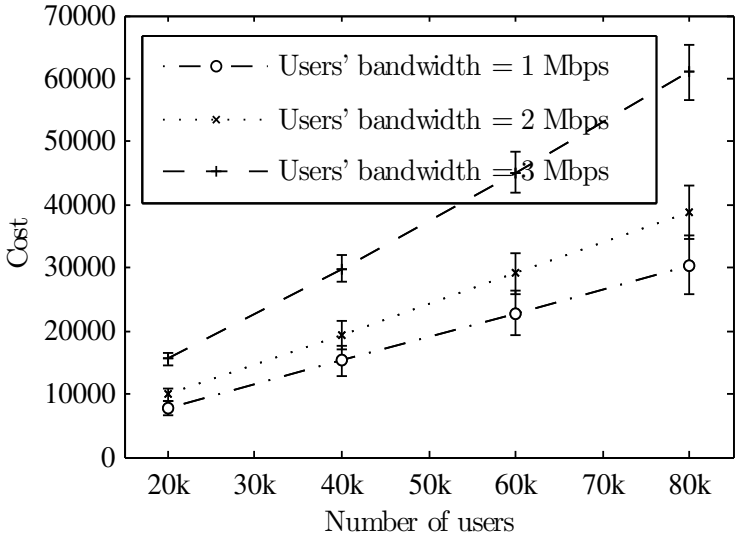

Figure 3.7: Comparison of costs with different bandwidths in users' SLA, users' average delay in SLA=60 ms

Total Operating Cost with Different Data Placement Algorithms

In this section, we changed the number of users from $20k$ to $100k$, and assigned their data to 25 DSCs. The users are clustered in groups, and the size of each group is about 1000 users. We compared the total operating cost with different data placement algorithms. Figure 3.4 shows the cost with the three different algorithms: the optimization method proposed in Section 3.3, the probabilistic method and the most frequent method described above. We can see that the total operating cost for the optimization method is about 40% of that of other methods. Since this method solves the optimization problem and finds the best possible solution, the total operating cost is significantly reduced. We can also see that the total operating cost of the most frequent method and that of the probabilistic method are very close. This is reasonable because the probabilistic method allocates a user's data to a DSC with the probability proportional to the frequency that he visits the DSC, that is, the more frequent a user visits a DSC, the higher chance that his data is allocated to the DSC.

Comparison of SLA Violations in Different Methods

In this section, we compare percentage of SLA violations in different methods. In this experiment, we changed the maximum capacity of DSCs to a low value 300000, users' required bandwidth to higher value (randomly chosen to 2 or 3 Mbps), and users' delay requirement to a lower value (randomly chosen to 60 or 80 ms). In this experiment, we set the users' data amount to 100 units.

Figure 3.5 shows the percentage of SLA violations in different methods. From this figure, we have the following observations: First, the percentage of SLA violations in the probabilistic method and the most frequent method is much higher than that in the optimization method. Second, when the number of users is 80k, the total amount of users' data is more than total capacity of DSCs (i.e., $80000 \times 100 > 25 \times 300000$). Therefore, all methods will lead to SLA violations.

Total Operating Cost with Different Bandwidth Requirements

Figure 3.6 shows the comparison of total operating cost with different bandwidth requirement in users' SLA. In this experiment, we changed the number of users from $20k$ to $80k$, and assigned their data to 25 DSCs. We considered that users require the delay less than 80 ms and changed the users bandwidth requirement in SLA from 1

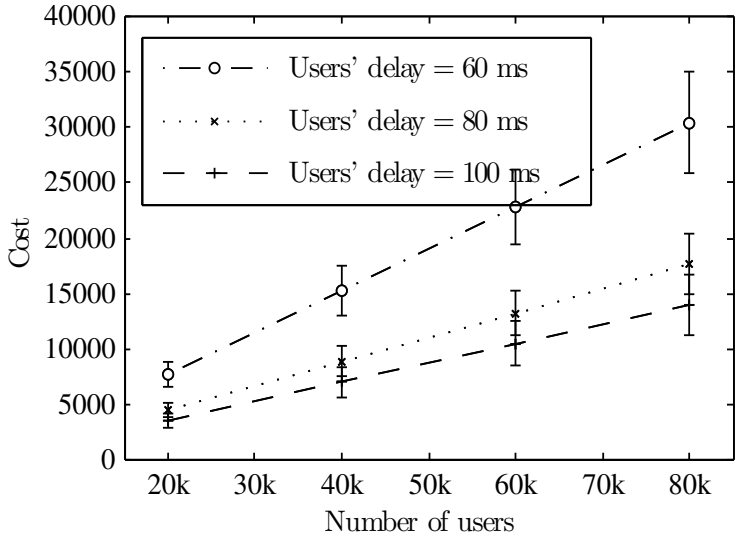

Figure 3.8: Comparison of costs with different delays in users' SLA, Users' average bandwidth in SLA=1 Mbps

Mbps to 3 Mbps. To save space, we only illustrate the results with the optimization method, because the performance trend similar to that in Figure 3.4 could be observed for the probabilistic method and the most frequent method.

From Figure 3.6, we can see that users' SLAs have a significant impact on the total operating cost. As users require a higher bandwidth, the total operating cost increases. This is because when the required bandwidth becomes high, the feasible number of links between DSCs that meet the bandwidth requirement becomes smaller, resulting in a smaller number of options to place users' data. As a result, some users' data may have to move to DSCs that have a higher operating cost to meet their bandwidth requirement.

In Figure 3.7, we assumed that users' delay requirement is less than 60 ms and changed the users bandwidth requirements from 1 Mbps to 3 Mbps. We observed the same trend of cost increase as in Figure 3.6. Comparing Figure 3.6 and Figure 3.7, we can see that the total operating cost with the delay requirement of 60 ms is higher than that with the delay requirement of 80 ms. This is reasonable because a tight requirement on delay further reduces the number of feasible solutions and leads to a higher total operating cost.

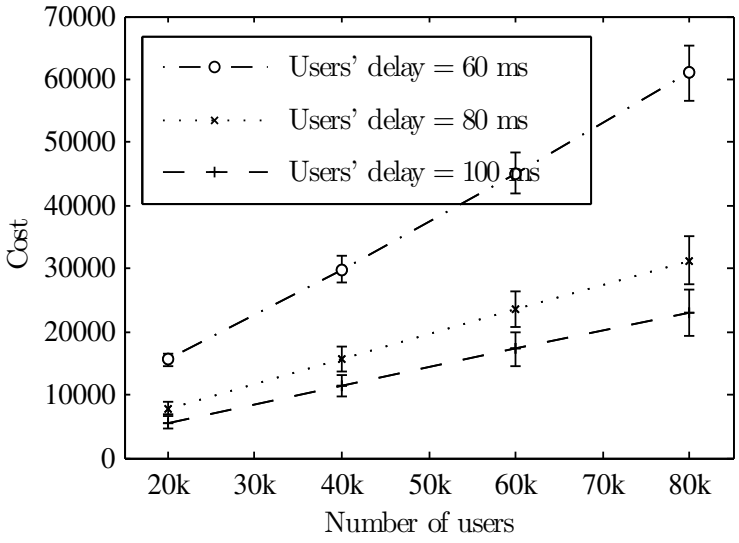

Figure 3.9: Comparison of costs with different delays in users' SLA, Users' average bandwidth in SLA=3 Mbps

Total Operating Cost with Different Delay Requirements

In this experiment, we changed the number of users from $20k$ to $80k$, and assigned their data to 25 DSCs. The users are clustered in groups, and the size of each group is about 1000 users. We assumed that users' required bandwidth is 1 Mbps and changed the users delay requirements from 60 ms to 100 ms. We only show the results obtained with the optimization method, since the results obtained with the other two methods follow a similar trend.

Figure 3.8 shows the comparison of operating costs of the optimal data placement method with varying delays in users' SLA. We can see that as users require a lower delay, the total operating cost increases. Figure 3.8 shows that the total operating cost with users delay of 60 ms is higher than that with users' delay of 100 ms, due to the same reason explained in the previous section. In addition, the difference between the operating cost with users delays of 60 ms and that with the delay of 80 ms is more than the difference between the operating cost with delay of 80 ms and that with the delay of 100 ms. As users require a lower delay, they cannot use many links of the network and the operating cost increases significantly. Figure 3.9 shows the comparison of operating cost with users bandwidth of 3 Mbps. As users require a bandwidth of 3 Mbps, the operating cost increases compared to that when users require a bandwidth of 1 Mbps.

3.5.2 Performance Evaluation of Maximizing Users' Satisfaction

In this section, we test the two solutions, i.e., LP based branch-and-bound and LP with rounding, to the problem of maximizing users' satisfaction.

Performance Comparison of The Two Solutions

We compare the execution time of LP based branch-and-bound and LP with rounding. Figure 3.10 shows the execution time of these algorithms (over a desktop computer with Core i7 CPU and 10 GB RAM). From this figure, we can see that the running time of LP based branch-and-bound is less than that of LP with rounding. This is caused by the fact that LP based branch-and-bound finds the solution in one iteration. LP with rounding, however, runs over several iterations. Nevertheless, LP with rounding needs much less amount of memory. We can also observe that the

Table 3.2: Simulation Parameters

Parameter	**Value**
No. of DSCs (n)	25
No. of users (m)	Varying from $20k$ to $100k$
Normalized Poisson arrival rate	0.1
Maximal capacity of DSCs ($M[i]$)	5M
Each user's data amount ($a[i]$)	100
Network bandwidth (b_{ij})	Randomly chosen between 10 Gbps and 60 Gbps
Network delay (d_{ij})	Randomly chosen between 20 ms and 120 ms
Users' access pattern (u_{ij})	Random values in $[0, 1]$
Users' bandwidth requirement ($SLA\_b[i]$)	Varying from 1 Mbps and 5 Mbps
Users' delay requirement ($SLA\_d[i]$)	Varying from 20 ms and 100 ms
Costs of DSCs per data unit ($c[i]$)	Randomly chosen between 0.001 and 0.02
Users' satisfaction threshold in delay requirements	0.5
Users' satisfaction threshold in bandwidth requirements	0.5
Maximum operating cost of all data centres (C)	1000000

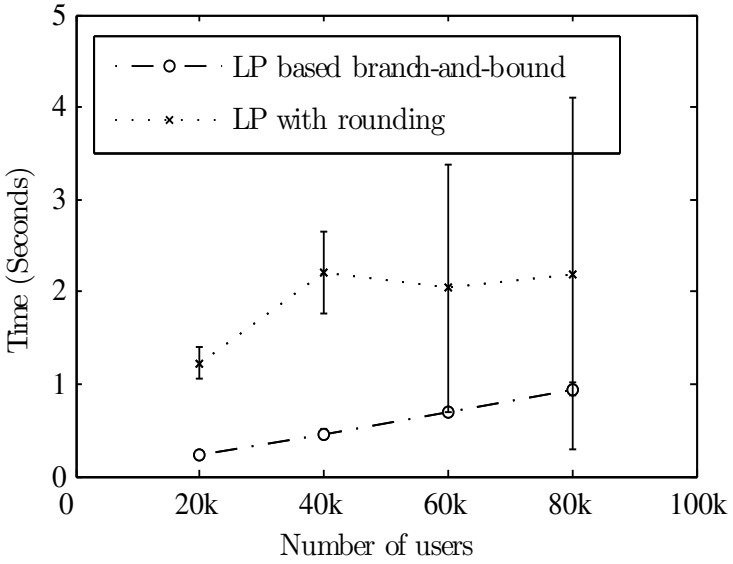

Figure 3.10: Comparison of execution time

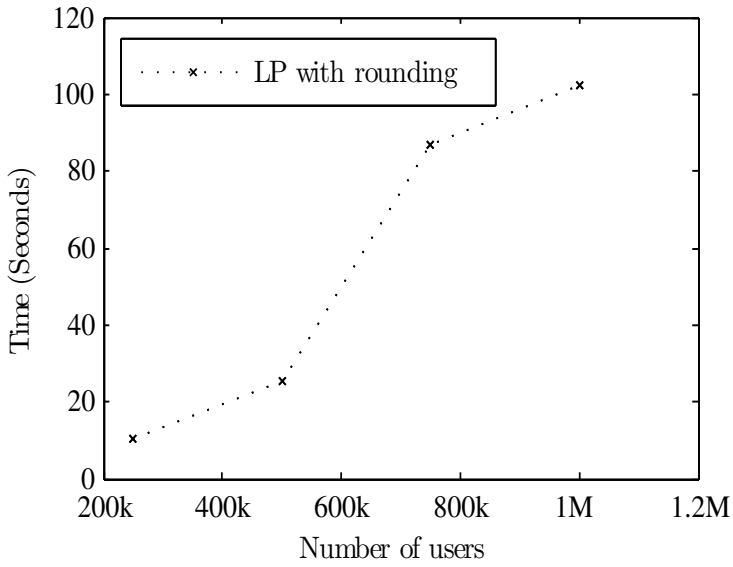

Figure 3.11: Execution time of LP with rounding method

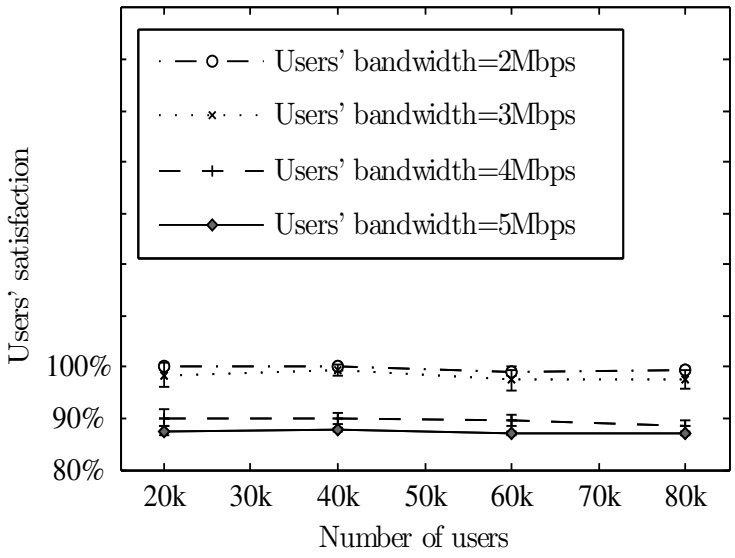

Figure 3.12: Comparison of users' satisfaction with different bandwidths in users' SLA, users' delay in SLA=100 ms

variation of the execution time using LP with rounding is higher than the LP based branch-and-bound method. This is caused by the fact that the rounding method may lead to different number of iterations for different scenarios.

Nevertheless, the LP based branch-and-bound method does not scale well. When we change the number of users to $1000k$ and the number of DSCc to 100, the LP based branch-and-bound method cannot be executed on regular desktop computers due to its large memory overhead. There is on going research on how to improve LP based branch-and-bound method including for example using parallel programming [27].

In contrast, we do not have such a problem using LP with rounding. Figure 3.11 shows the running time of this method for network scenarios of 100 DSCs and up to $1M$ users. This figure indicates that the LP with rounding method is applicable to the cloud scenarios with a high number of users and DSCs.

Users' Satisfaction with Different Bandwidth Requirements

In this test scenario, we set the users' delay requirement to 100 ms, and change their bandwidth requirement from 2 Mbps to 5 Mbps. Figure 3.12 shows users' satisfaction with different users' bandwidth requirement.

From this experiment, we first observe that users' satisfaction for bandwidth re-

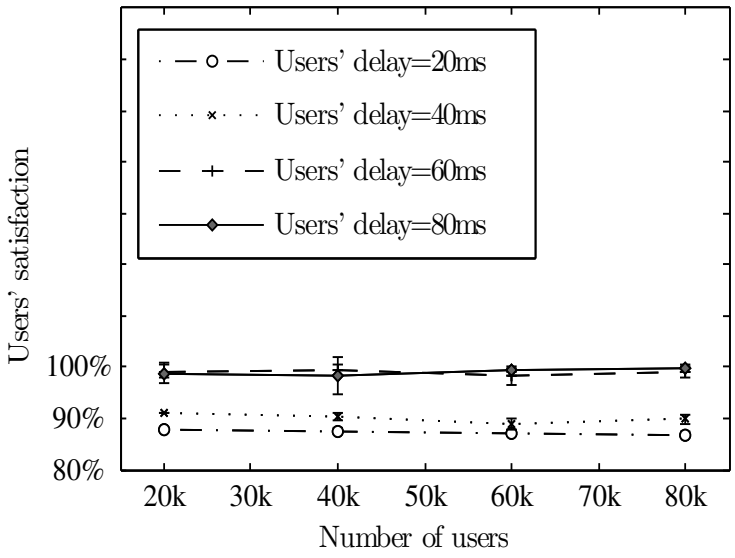

Figure 3.13: Comparison of users' satisfaction with different delays in users' SLA, users' bandwidth in SLA=1 Mbps

quirement of 2, 3, 4, and 5 Mbps is about 99.5%, 98%, 89%, and 87%, respectively. When users require a low bandwidth (e.g. 2 or 3 Mbps), their satisfaction is very high (e.g. 99.5% or 98%). We can also see that as users require higher bandwidth, it might not be possible to locate their data on a DSC that can satisfy their SLA.

Users' Satisfaction with Different Delay Requirements

In this test scenario, users' bandwidth requirement is set to 1 Mbps, and their delay requirement is changed from 20 to 80 ms. Figure 3.13 shows users' satisfaction with different users' delay requirement. It is observed that users' satisfaction for delay requirement of 20, 40, 60, 80 ms is about 87%, 88%, 99%, and 99.5%, respectively. If users require lower delay, it might not be possible to locate their data on a DSC that can satisfy their requirement, and thus the users' satisfaction drops to a lower value.

3.5.3 Tradeoff Between Operating Costs and Users' Satisfaction

Figures 3.14 and 3.15 show the effect of value of α on users' satisfaction and cloud provider's operating costs, respectively, as discussed in Section 3.3.4. From these

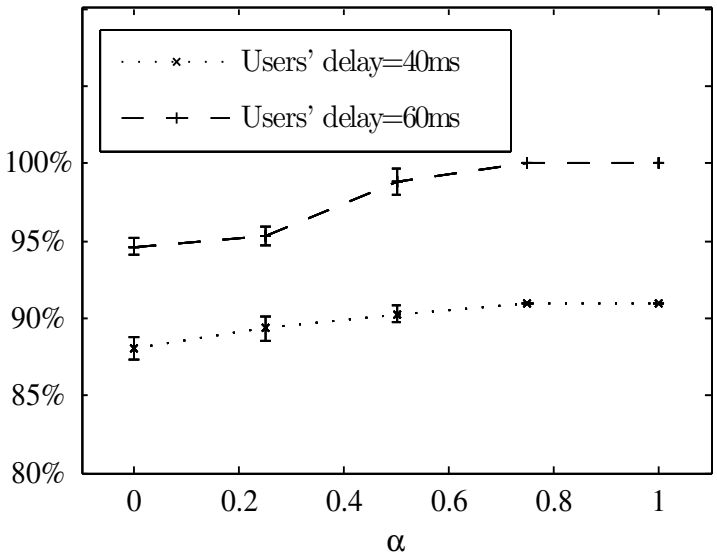

Figure 3.14: Users' satisfaction with different values of α, users' bandwidth in SLA=1 Mbps

figures, we observe the following results:

First, when the value of α is small, the operating costs decrease, but users' satisfaction drops to a lower level. For example, when the value of α is 0, the objective function in Equation (3.16) only cares about costs. Therefore, users' satisfaction level drops, but it cannot drop to a too small value due to the constraints on minimal QoS requirements regulated by inequalities (3.12) and (3.13).

Second, when the value of α is large, users' satisfaction increases, and the operating costs raise. When the value of α is 1, the objective function in Equation(3.16) only consider users' satisfaction. Therefore, operating costs increase to a high value, within the constraint defined by inequality (3.14).

3.6 Summary

Mobile Telecom Cloud (MTC) can provide users with high quality of cloud services as telecom companies own the network infrastructure. We study the problem of minimizing the system costs and maximizing users' satisfaction in MTC. The first studied problem is how to deploy users' data in the best locations such that the users SLAs can be satisfied, and the overall operating cost of MTC is minimized.

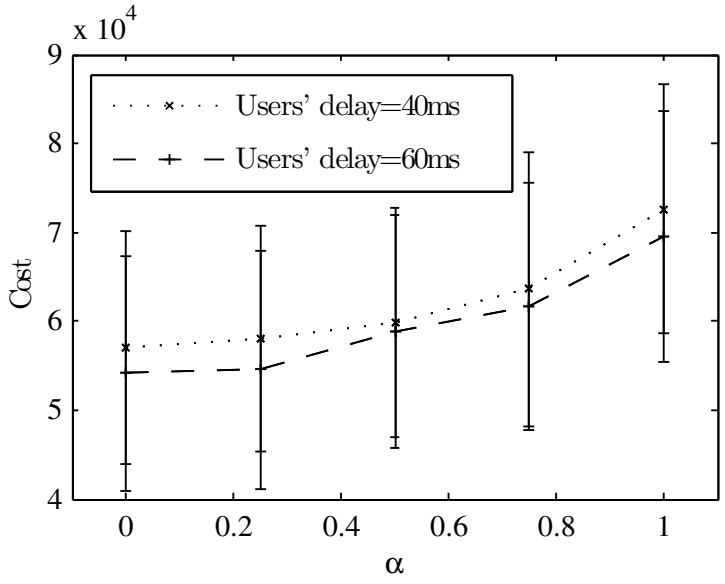

Figure 3.15: Costs with different values of α, users' bandwidth in SLA=1 Mbps

The second studied problem is how to maximize users' QoS satisfaction within cost constraints in MTC.

In this chapter, we have presented an analytical framework which consists of a network model, a user model (access profile and SLAs), and the cost model. In the network model, we consider that users are mobile, and they can connect to the cloud as long as they can communicate with the telecom service provider. The network is modeled as a graph representing network links' delay and bandwidths. The user model consist of service level agreements (users' required bandwidth and delay), and data access profile representing the pattern that a user connects to DSCs. The cost model represent operating costs of MTC such as maintenance cost and electricity cost in different DSCs.

To solve the optimization problem, we present two algorithms, LP based branch-and-bound and LP with rounding. By clustering the users with similar data access pattern, we can manage a system of up to millions of mobile users in MTC. We further investigate how to make trade off between users' satisfaction and cloud provider's operating costs. Simulation results show that our method can scale well to large networks. For our future work, we plan to study various clustering methods as mentioned in Section 3.4. In the next chapter, we discuss the scenario where a MTC provider acts as a brokerage, called MTC brokerage, to broker TPC cloud resources and integrate the resources reserved from TPC with those of its own. This will provide better service at lower costs to the customers while increasing MTC's profits.

Chapter 4

Mobile Telecom Cloud Brokerage with Orchestrated Multi-Tier Resource Pooling

4.1 Motivation

Cloud computing provides mobile telecommunication companies with new opportunities to augment their services to mobile users with mobile telecom cloud (MTC), which refers to cloud services supported by mobile telecommunication companies. For example, MTC can store its customers' data into the cloud storage owned by other cloud providers. By investing in the existing network infrastructure, MTC can offer cloud services with higher Quality of Service (QoS). However, large cloud providers such as Google and Amazon, which are called third-party cloud (TPC) providers in this paper, normally have much larger computing resources and can provide users with cheaper cloud services. An interesting question is: can MTC offer better and cheaper cloud services than TPC?

This chapter provides a positive answer to the above question with an orchestrated multi-tier resource pooling method. The main idea is to let a MTC provider group buy TPC services with discounts and jointly schedule the reserved TPC resources and its own MTC resources. Generally speaking, MTC can provide users with cloud services of higher bandwidth and lower delay, but the cost of MTC services might be high; On the other hand, TPC can offer users cheaper cloud services, but mobile users may suffer from higher delay and lower bandwidth. Our method takes advantage

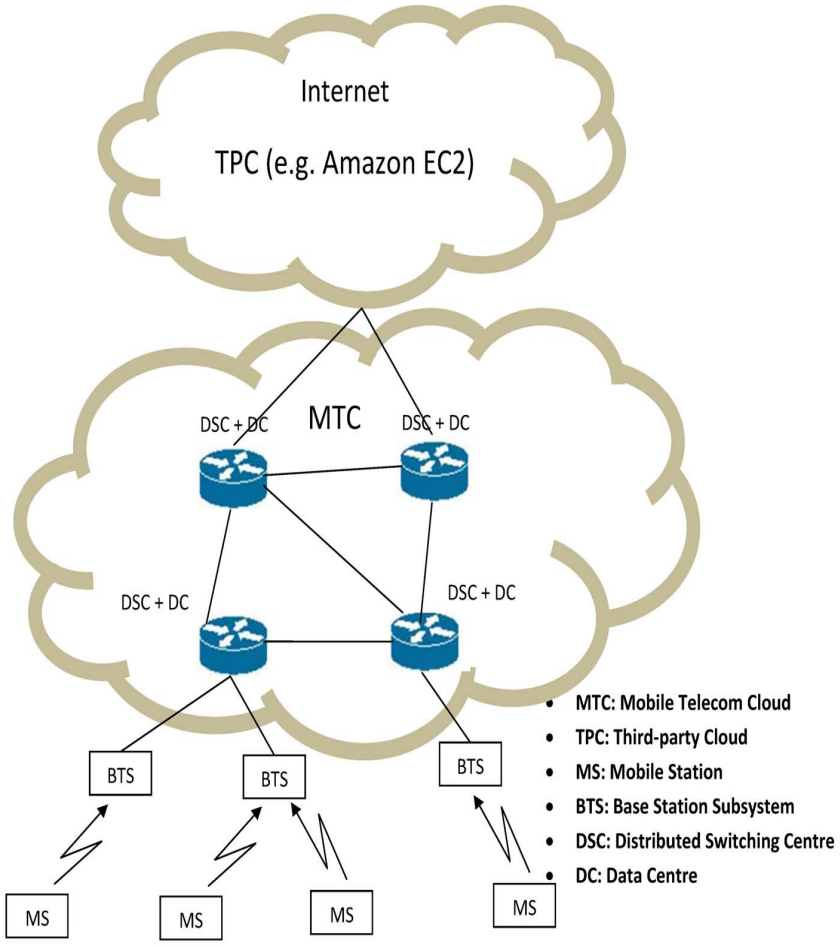

Figure 4.1: System architecture of MTC brokerage

of both services and creates a win-win solution for both the MTC provider and its customers without causing any negative impact on TPC providers.

The system architecture of our solution is illustrated in Figure 4.1. The MTC provider acts as a brokerage and group buys a large chunk of cloud resources with a discount price offered by TPC services such as Amazon EC2 [1]. Mobile users connect to the nearest Distributed Switching Centre (DSC), and their requests are routed to the DSC that contain their cloud services. If users require a lower QoS, their services may be dispatched to TPC using the reserved resources. The problem is to integrate services of MTC and TPC to benefit both the MTC provider and its customers. In addition, TPC providers may potentially benefit from the solution, since they can group sell their services to more users. Note that in this chapter, we

do not consider the scenario that MTC migrates its network elements using network function virtualization (NFV) to TPC.

The above problem involves higher QoS services with a higher cost in MTC and lower QoS services with a cheaper price in TPC. It seems very similar to the scheduling problem for multi-tiered resources, such as memory/cache management in operating systems, which has been broadly investigated before.

Nevertheless, a close investigation discloses that our problem poses special complexity and challenges that demand a different solution. *First*, the discounts offered by TPC depends on the amount of reserved resources, because higher commitments of future resource usages are the main motivation for TPC to offer discounts. As such, major TPC providers such as Amazon EC2 adopt different discount levels according to users' reserved volume of resources. This layered pricing strategy poses a challenge for the MTC brokerage to make proper decision on resource reservation, since higher discounts imply higher commitments that may not be fulfilled by its customers. *Second*, the MTC provider must guarantee that with the same price paid, its customers obtain better QoS than that of TPC, or otherwise the customers are not motivated to use its services. This poses another challenge for the MTC provider to discover the profitable price range and carefully design its charging strategy. *Third*, when the number of customers is large, finding the best match between users' cloud requests and suitable resource supplies introduces a complexity to the optimal resource planning.

In this chapter, we address the above challenges and make the following contributions:

1. To facilitate resource planning, we present a mathematical model for MTC brokerage. The model reflects the cloud brokerage scheme in which a telecom company can provide its own cloud services as well as third-party cloud services reserved with group buying.

2. We formulate and solve the optimization problem of minimizing the costs of MTC brokerage. In addition, the solution provides an insight on MTC's profitable price range that the MTC brokerage could possibly charge to its customers.

3. To alleviate the (negative) impact of dynamic changes of customers' requests, we present a contract update scheme for the MTC brokerage to make wise decisions based on predicted customers' requests.

4. We demonstrate the benefits of our solution through tests using real Google traces collected over 29-day period from a Google cluster containing over 12500 physical machines [42].

The rest of this chapter is organized as follows. Subsection 4.2 describes the system model of MTC brokerage, including three components: cloud service model, TPC pricing model, and quality of service model. In Subsection 4.3, we formulate the optimal resource scheduling for MTC brokerage and present two solutions. In Subsection 4.4, we discuss profitable price range for MTC brokerage. Subsection 4.5 discusses the adaptive resource planning in MTC brokerage. In Subsection 4.6, we evaluate the efficacy of our design.

4.2 System Model of MTC Brokerage

For ease of reference, notations used in this chapter are listed in Table 4.2.

4.2.1 Cloud Service Model

Figure 4.2 shows an example of the cloud service model. The telecom company acts as a MTC brokerage and can group buy cloud service instances from TPC with discounts. Mobile users connect to telecom network infrastructure, and their cloud services can be allocated to MTC or TPC based on their QoS requirements.

Aligning with the tiered QoS levels, there are tiered service plans and the corresponding prices. We assume that the system consists of n different cloud service plans offered by MTC and TPC, where MTC offers n_1 cloud plans, and TPC offers $n_2(= n - n_1)$ cloud plans. Accordingly, a service plan is associated with a QoS level, and we assume that the QoS levels can be ranked. Note that this assumption is common and has been adopted in existing QoS models such as DiffServ [23]. We use a $n \times 1$ column vector, Q_p, to denote the QoS levels, where $Q_p[i]$ represents the QoS level of cloud plan i.

We assume that MTC has limited computing/storage resources but TPC has resources abundant enough to meet all computing requests. In this sense, we denote the total capacity of cloud plans with $n \times 1$ column vector M where $M[i]$ represents capacity of plan i, and we set the total capacity of TPC plans to infinity. Note that we use the term "capacity" to simplify problem modeling. The practical meaning of

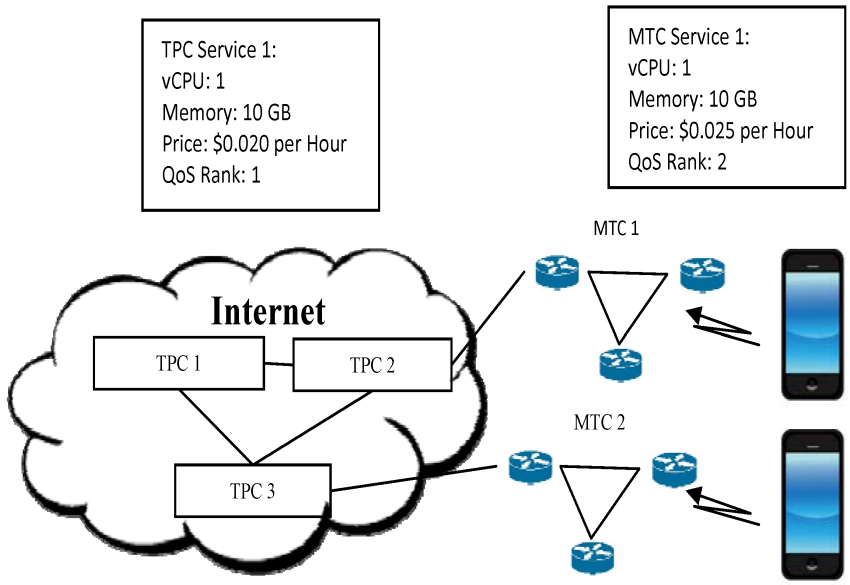

Figure 4.2: An example of TPC and MTC cloud services

Table 4.1: An Example Reserved Instance Volume Discounts

Total Reserved Instances	**Discount**
$250,000 to $500,000	10%
$500,000 to $750,000	20%
$750,000 to $1,000,000	30%
More than $1,000,000	40%

M should be concretized to specific computing or storage resources, depending on the users' requirements.

4.2.2 TPC Pricing Model

A TPC such as Amazon EC2 offers discounts if the users' commitments meet the given threshold values. As an example, Table 4.1 shows a tiered pricing structure for buyers of large service volume. For different TPC providers the discount value for different amount of reserved instances may be different from this table, however, the tiered structure is the same. Taking advantage of the discounts, MTC brokerage can reserve a sufficient number of reserved instances from TPC and serve its customers with a price cheaper than what TPC charges directly to individual customers. We

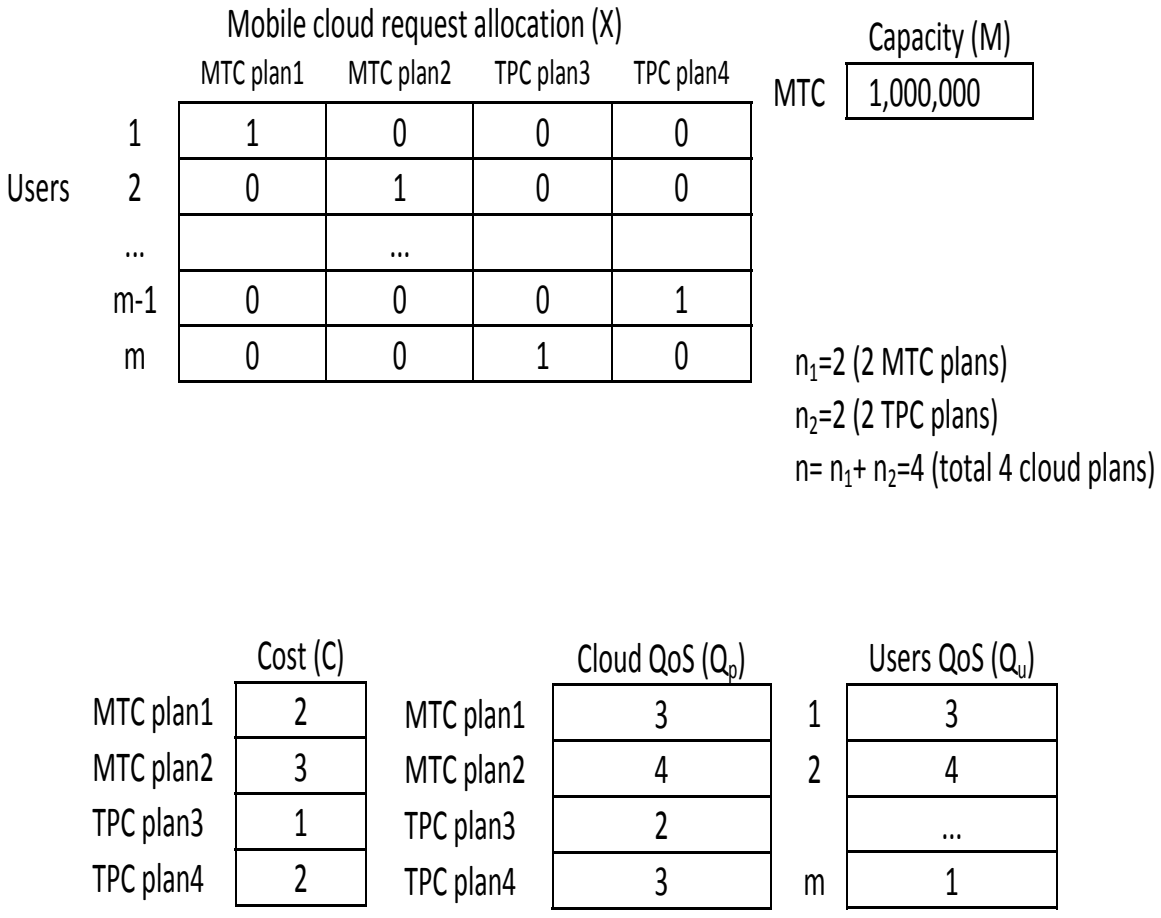

Figure 4.3: An example of users and cloud provider inputs

assume TPC is unaware of individual MTC customers.

Using the discounts structure in Table 4.1 as an example, assume that a customer asks for $m3.medium$ service from Amazon EC2, which will be charged for \$50 per month. If the MTC brokerage aggregates $100,000$ such customers, the MTC could reserve $100,000$ $m3.medium$ instances with the cost of $\$3,000,000$ per month. Assuming the cost for MTC brokerage to operate is $\$500,000$ per month, then the customers can get a price of less than \$50 per month for the same service from the MTC brokerage.

Table 4.2: Notations

Notations for optimal resource scheduling (Section 4.3)	
Notation	**Meaning**
m	Number of users
n	Total number of MTC and TPC plans ($n = n_1 + n_2$)
n_1	number of MTC cloud plans
n_2	number of TPC cloud plans
$F_{n_1}(X)$	Filter function that changes the last $n-n_1$ columns to zero, where X is $m \times n$ matrix
$F_{n_2}(X)$	Filter function that changes the first $n-n_2$ columns to zero, where X is $m \times n$ matrix
C	$n \times 1$ column vector where $C[i]$ represents the cost of cloud plan i
P	$m \times 1$ column vector where $P[i]$ represents the charging price offered to user i
Q_u	$m \times 1$ column vector where $Q_u[i]$ represents user i QoS level requirement
Q_p	$n \times 1$ column vector where $Q_p[i]$ represents QoS level achievable by cloud plan i
h	TPC discount threshold
d	TPC discount value for discount threshold h
M	$n \times 1$ vector representing capacity of cloud plans
X	$m \times n$ matrix where x_{ij} is a binary variable (1 if user i's cloud request is allocated to cloud plan j, and 0 otherwise)
Notations for handling changes in service request (Section 4.5)	
c_e	Cost of buying extra resources from TPC without discount during the smallest time unit
c_r	Cost of buying one reserved instance during the term of contract from TPC
$P_B(t)$	Blocking probability at time t

4.2.3 Quality of Services

To facilitate customers' service order, we assume that the MTC brokerage packages resources as service plans. Each service plan is bound with a service level agreement that specifies the QoS requirements including detailed technical components such as number of vCPUs, size of memory, delay, and bandwidth. In this chapter, we use a tiered numbering to denote the level of QoS. In general, a service plan with a higher price offers better QoS (in terms of each technical component) than that with a lower price. Users can select a service plan that matches best with their service preferences. Associated with each service plan is its price, based upon which it can be ranked. As a reasonable assumption, if a customer pays for a given service plan, her service can be allocated to any plan with an equal or higher rank.

Assuming there are m users in the system, we use a $m \times 1$ column vector, Q_u, to denote users' required service plans, where $Q_u[i]$ represents the QoS requirement of user i. Following Section 4.2.1, we assume that the cloud services are packaged into n plans, which are ranked and recorded in a $n \times 1$ column vector, Q_p.

4.2.4 Objective

The objective of optimal resource scheduling is to maximize MTC brokerage's profit. Since a MTC brokerage's profit depends on the amount of reserved resources, the uncertainty in future customers' requests makes accurate reservation difficult. To better investigate the impact of such uncertainty, we adopt a two-step solution. In the first step, we solve the optimal resource scheduling problem using all information known at hand (Section 4.3). The solution can give us insight into the profitable price range that a MTC brokerage could charge to its customers (Section 4.4). Based on this, in the next step, we study the strategy for the MTC brokerage to tame the negative impact of dynamic changes in service requests (Section 4.5).

4.3 Optimal Resource Scheduling for MTC brokerage

4.3.1 Problem Formulation

We consider the following optimization problem. Given the users' QoS requirements, how should the MTC brokerage group buy cloud instances from TPC such that the overall service costs are minimized? The problem can be formally formulated as follows:

$$\min_{X=[x_{ij}]_{m\times n}} 1^T_{m\times 1} \cdot (F_{n_1}(X) \cdot C) + (1-d)(1^T_{m\times 1} \cdot (F_{n_2}(X) \cdot C)) \tag{4.1}$$

subject to

$$x_{ij} \in \{0, 1\} \tag{4.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \text{ for } i = 1, 2, \ldots, m \tag{4.3}$$

$$1^T_{m\times 1} \cdot (F_{n_2}(X) \cdot C) \geq h \tag{4.4}$$

$$X \cdot Q_p \geq Q_u \tag{4.5}$$

$$X^T \cdot 1_{m\times 1} \leq M \tag{4.6}$$

Equation (4.1) is the cost function that measures total cost of cloud services. X is a $m \times n$ matrix where $x_{ij} = 1$ indicates that user i's cloud request is allocated to cloud plan j. T denotes the transpose of a matrix. We define the filter function $F_{n_1}(X)$. The input of $F_{n_1}(X)$ is a $m \times n$ matrix. The output of $F_{n_1}(X)$ is a $m \times n$ matrix where the first n_1 columns are the same as input matrix X, and the second n_2 columns are all zeros. We also define the filter function $F_{n_2}(X)$. The input of $F_{n_2}(X)$ is a $m \times n$ matrix. The output of the function is a $m \times n$ matrix where the first n_1 columns are all zeros, and the second n_2 columns are the same as input matrix X. We record the costs of cloud plans offered by MTC or TPC with the $n \times 1$ column vector C.

In this problem formulation, we assume that the MTC brokerage group-buys more than discount threshold h from TPC and gets the discount d. Equations (4.2) and (4.3) together assume that a user's cloud request is allocated to only one cloud

```
1: while The given volume discount threshold is not reached do
2:     Select the TPC plan i with lowest cost;
3:     Assign user j to plan i if the user's QoS is satisfied with plan i;
4: end while
5: for each remaining user do
6:     Find the set of cloud plans that meet the user's QoS constraint;
7:     Assign the user to the plan that has the lowest cost in the above set;
8: end for
```

Figure 4.4: The MinCost greedy algorithm

provider plan. Inequality (4.4) ensures that the total amount of reserved instances from TPC is greater than the discount threshold. Inequality (4.5) ensures that users' QoS requirements are satisfied. Inequality (4.6) ensures that the total amount of cloud services allocated to MTC plans is less than the total capacity of MTC cloud services. Figure 4.3 shows an example of users and cloud provider inputs.

4.3.2 Algorithms for Solving the Optimization Problem

First, we set the discount threshold h from TPC and the corresponding discount d in the above optimization problem. We present two algorithms, linear programming (LP) with rounding and MinCost Greedy, to solve the problem. In the first algorithm, we use an approximation algorithm based on linear programming combined with the rounding technique. The basic idea is to relax the requirement in Equation (4.2) to allow x_{ij} being real numbers; we then solve the relaxed problem using linear programming. For the x_{ij} values that are either close to 1 or close to 0, we round them up to 1 or down to 0, respectively. In the next round of iteration, we fix those rounded values, and perform the same procedure again until the convergence condition is met (i.e., the change of the objective function is smaller than a small threshold in two consecutive iterations).

In the MinCost Greedy algorithm, we first meet the given discount threshold. While the given volume discount threshold is not reached, we select the TPC plan i with lowest cost and assign user j to plan i if the user's QoS is satisfied with plan i. Then for the remaining requests, we allocate them to the cheapest plan that can meet their QoS. The MinCost Greedy algorithm is shown in Figure 4.4.

Note that in the above algorithm, we assume that the discount threshold h from TPC and the corresponding discount d are given. In practice, however, there are

different discount levels such as the reserved instance tiers shown in Table 4.1, and we actually do not know which TPC discounts plan leads to the minimum overall cost. The key idea to address in the above problem is that if the MTC brokerage group-buys a TPC plan leading to the minimum overall cost, it has no reason to reserve resources in other discount levels. Using Table 4.1 as an example, if the MTC brokerage can reach the $\$500,000$ volume, it has no reason to split the large volume to a smaller scale for less discounts. As such, starting from lowest discount threshold, we solve the above optimization problems (with either LP with rounding or MinCost greedy) for each discount tier with the discount threshold h and corresponding discount d. The solution having the smallest cost is the final solution.

4.3.3 A Scalable Solution

With millions of mobile users, the number of variables in the above optimization problem will be too big to handle. To make the solution scalable, we cluster the users based on their QoS requirements. Users having similar QoS requirements are grouped together, and we treat a group of users as a "virtual" user in the above optimization problem.

4.4 Profitable Price Range for MTC Brokerage

The solution of the above cost minimization problem also provides an insight into the price range that is both attractive to customers and profitable for the MTC brokerage. After we obtain the optimal solution $X = [x_{ij}]_{m \times n}$, a profitable charging scheme P must meet the following two conditions:

$$P \leq X \cdot C \tag{4.7}$$

$$1^T_{m \times 1} \cdot (F_{n_1}(X) \cdot C) + (1-d)(1^T_{m \times 1} \cdot (F_{n_2}(X) \cdot C)) \leq \sum_{i=1}^{m} P[i] \tag{4.8}$$

P is a $m \times 1$ column vector where $P[i]$ represents the charging price offered to user i. Inequality (4.7) ensures that for each user, the charged price is less than the price without discount. This is because individual user cannot get volume discount from

TPC. Inequality (4.8) ensures that the total cost of MTC brokerage is less than the sum of users' prices. The former condition is to attract customers to use the services from the MTC brokerage; the latter condition is to guarantee the profit of the MTC brokerage.

4.5 Adaptive Resource Planning

In the previous sections, we studied the optimal resource allocation problem with the assumption that users' requests do not change. Nevertheless, in practice, users may drop their service plans, upgrade their plans, or new users' may arrive in the system. Simply put, the impact of dynamic requests on the MTC brokerage is as follows: when a user drops its plan which was allocated to TPC, the MTC brokerage loses profit since it has contract commitment to the TPC. Also, when a new user joins the system, new discount threshold of TPC may be reached. In both cases, the existing resource allocation may not be optimal anymore.

Nevertheless, it is impractical for the MTC brokerage to sign a new contract with the TPC for each customer-initiated change. This task is normally done on monthly or even a longer basis, as the existing business model for reserved instances is term based. For this reason, depending on the time period for the termed contract, the time horizon in our study is between time $t-1$ when the last contract was signed and the current time t when the contract needs update. *The adaptive resource planning problem is to decide at current time how the MTC brokerage should update its contract with the TPC (i.e., what amount of resources should be reserved).*

For adaptive resource planning, we assume the following resource allocation scheme during time $t-1$ and t.

- The users might not be bound to termed contracts with the MTC brokerage, so they have the freedom to change or drop their plans. This assumption is to support the maximum flexibility to attract customers. As a return, the MTC brokerage has the right to allocate the unused resources to new users.

- When users drop their plans, no action needs to be taken. This is because the MTC brokerage has made commitment to the TPC already at time $t-1$, no matter whether or not the reserved resources will be used in the termed period.

- When a new user arrives, the MTC brokerage allocates the cheapest plan that

meets the user's QoS requirement. If no resource is available at the chosen plan, the MTC brokerage records a "request overflow," for which the MTC brokerage could either reject the request or purchase extra resource from TPC without any discounts.

- For users who want to upgrade or downgrade their plan, the MTC brokerage processes their request as dropping their current plan and adding them as arrivals with a new QoS requirement.

To make an effective decision, the MTC brokerage needs to accurately monitor the resource usage during time interval between $t-1$ and time t. Since cloud monitoring is beyond the scope of this thesis, we assume that the cloud platform can provide the MTC with the up-to-date information of the resources owned by the MTC or reserved from the TPC. In particular, the cloud platform could provide the information regarding the resource utilization at each QoS level and cloud request arrival rate $\lambda(t)$.

By monitoring $\lambda(t)$ during time $t-1$ and time t in each QoS level, we use the historical information from the time period $t-1$ to t to estimate future cloud request arrivals in the time period from t to $t+1$. This estimation can then be used to calculate the number of instances that should be reserved from TPC. To be specific, we build a performance model and use the historical information for estimating model parameters. After that, we predict the request blocking probability and determine the proper amount of instances in each QoS level that should be reserved from TPC.

We model this system as a non-stationary Erlang loss system $(M(t)/M/s/0)$ [9]. The requests arrive at the servers following a non-stationary Poisson process with rate $\lambda(t)$. The servers' service time follows an exponential process with service rate μ. In this model, there are s servers, and the waiting queuing space is 0. With this model, if all the servers are busy a new request will be blocked and dropped out of the system. Translating the above performance model to our system, a blocking means that all reserved TPC instances are busy, and the new request should be served either with a MTC plan, if available, or with a TPC plan that is temporarily purchased by the brokerage without having a term-discount from the TPC.

Clearly, the blocking probability is a key parameter for making proper decision on reserved resources from TPC. If the blocking probability is too low, the number of servers may be too large, indicating that the brokerage may have over reserved TPC resources; If the blocking probability is too high, the number of servers may

```
1: for each QoS level i do
2:     Calculate arrival rate λ(t) based on a cloud monitoring technique;
3:     s_opt = ∞;
4:     for each discount threshold h shown in Table 4.1 do
5:         Calculate min_c and min_s using the algorithm shown in Figure 4.6;
6:         if min_s < s_opt then
7:             s_opt = min_s;
8:         end if
9:     end for
10:    Use s_opt as optimal number of reserved instances for QoS level i;
11: end for
```

Figure 4.5: The algorithm for calculating optimal number of reserved instances for each QoS level

not be enough to meet requests. In the former case, the MTC brokerage has been over committed to TPC; in the latter case, the MTC brokerage has to temporarily purchase new instances from TPC, and thus cannot benefit from TPC discounts. With historical data from time $t-1$ to time t, we can model the arrival processes and the service rate and assume that they remain the same in the next contract term. Nevertheless, we need to determine the suitable number of servers that minimizes the total cost for the next contract term, i.e.,

$$\min_{s}\left(\int_{0}^{1} \lambda(t) \cdot P_B(t) \cdot c_e \cdot dt\right) + s \cdot c_r \cdot (1-d) \tag{4.9}$$

subject to

$$s \cdot c_r \geq h \tag{4.10}$$

where $P_B(t)$ is the blocking probability at time t, h is a given TPC discount threshold, and d is the corresponding TPC discount value. c_e denotes the cost of buying extra resources from TPC without discount during the smallest time unit, and c_r denotes the cost of buying one reserved instance during the term of contract from TPC.

Calculating the blocking probability at each time step requires that we solve a set of differential equations which is extremely complex even for a small system. Therefore, we use the Fixed Point Approximation (FPA) method [9] to find the average number of busy servers, $E[Q(t)]$, and blocking probability $P_B(t)$, at each time step.

We try to find the optimal number of reserved instances for each QoS level. There-

```
1: s = smallest value that satisfies Inequality (4.10);
2: Calculate P_B(t) using FPA algorithm;
3: min_s = s;
4: min_c = Objective function (4.9) with s reserved instances;
5: while P_B(t) > ϵ do
6:     s = s + 1;
7:     Calculate P_B(t) using FPA algorithm;
8:     c_temp = Objective function (4.9) with s reserved instances;
9:     if c_temp < min_c then
10:        min_c=c_temp;
11:        min_s=s;
12:    end if
13: end while
```

Figure 4.6: The algorithm for finding optimal number of reserved instances by reaching discount threshold h

fore, for each QoS level i, we consider different discount thresholds shown in Table 4.1, and for each discount threshold, we calculate the optimal number of reserved instances. Therefore, we can find which discount threshold results in the minimum number of TPC reserved instances and minimum costs.

Through numerical calculation, it's easy to see that the objective function (4.9) is not convex, and thus the optimization problem is hard to solve. With a simple example, we show that objective function (4.9) is not convex. In an experiment, we set the μ to 5 and $\lambda(t) \in \{10, 10, 10, 10, 5, 5, 5, 5, 1, 1, 1, 1\}$, and we changed the number of reserved instances, s, from 1 to 50. Figure 4.7 shows how the first term in objective function (4.9), $(\int_0^1 \lambda(t) \cdot P_B(t) \cdot c_e \cdot dt)$, changes with different number of reserved instances. Figure 4.8 shows the value of objective function (4.9) with different number of reserved instances. From Figure 4.8, we can observe two local minimum at $s = 2$ and $s = 13$. Therefore, this example shows that objective function (4.9) is not convex.

Nevertheless, when $P_B(t)$ gets too small, the value of the objective function mainly depends on the second term $s \cdot c_r \cdot (1 - d)$, which increases linearly with s. Due to this reason, for a given discount threshold h, we can estimate the optimal number of reserved instances using the following algorithm. First, we set the variable s to the smallest value that satisfies inequality (4.10) and calculate $P_B(t)$ using the FPA algorithm. Then, we repeatedly increase s and re-calculate $P_B(t)$ and the objective function (4.9) with s reserved instances, until $\max_{t\in[0,1]}\{P_B(t)\}$ becomes smaller than a threshold. The algorithm for calculating optimal number of reserved instances for each QoS level is shown in Figure 4.5. For a given discount threshold h, we can find the optimal number of reserved instances using algorithm shown in Figure 4.6.

4.6 Performance Evaluation

4.6.1 Optimal Resource Scheduling

In this section, we evaluate the performance of optimal resource scheduling by assuming that users' requests are known and remain unchanged. This study is helpful for initial service planning of MTC brokerage, when the brokerage has collected users' service requests but has not enough historical system running logs to predict future changes.

We use simulated data for this study. We first evaluate the performance of two

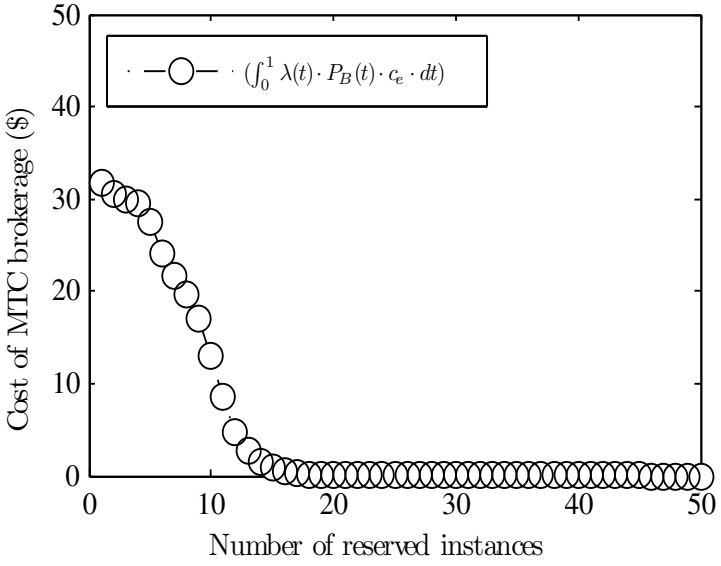

Figure 4.7: Changes of the first term in objective function (4.9), $(\int_0^1 \lambda(t)\cdot P_B(t)\cdot c_e\cdot dt)$, with different number of reserved instances

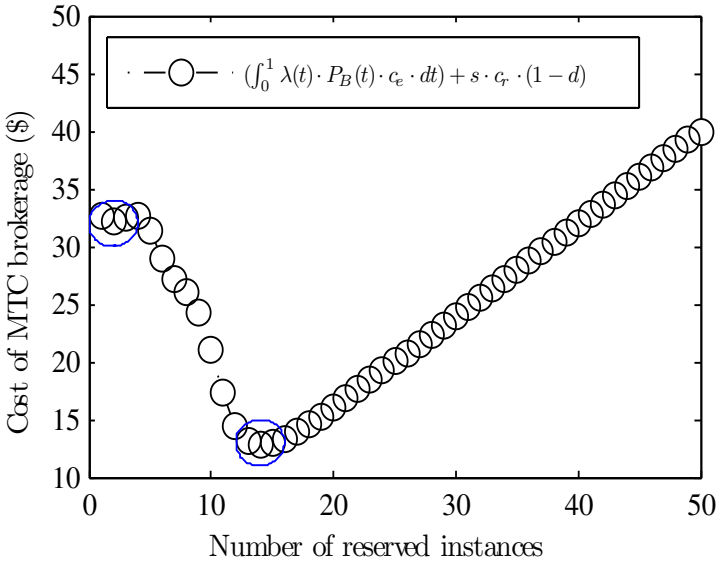

Figure 4.8: Changes of the objective function (4.9) with different number of reserved instances; two local minimums are observed.

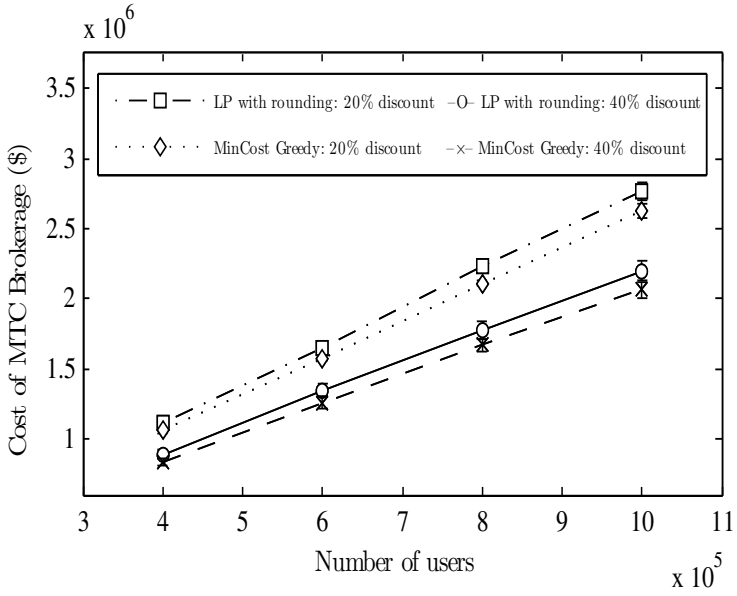

Figure 4.9: Comparison of cost of MTC brokerage using LP with rounding and MinCost Greedy algorithm with different TPC discount values

algorithms: LP with rounding and MinCost Greedy, in solving the optimization problem (4.1). Then, we illustrate the applicable price ranges that the MTC brokerage could charge to users, and the benefits of multi-tier resource pooling to both MTC brokerage and end users. The simulation parameters are listed in Table 4.3. The discount value is the discount of TPC plans as shown in Table 4.1.

Performance Comparison of The Two Algorithms

In this test, we changed the number of users from 400,000 to 1,000,000 and assigned their requests to 4 cloud plans. Figure 4.9 shows the comparison of cost of MTC brokerage between LP with rounding and MinCost Greedy algorithm with different TPC discount values. In the upper two lines, TPC discount threshold is set to $\$500,000$, and in this discount tier, the TPC discount value is 20%. In lower two lines, TPC discount threshold is set to $\$1,000,000$, and in this discount tier, the TPC discount value is 40%.

It can be observed that the cost of LP with rounding and MinCost Greedy algorithm are very close with the cost of LP with rounding slightly higher.

We compare the execution time of LP with rounding and MinCost Greedy. Figure 4.10 shows the execution time of the two algorithms (on a desktop computer with

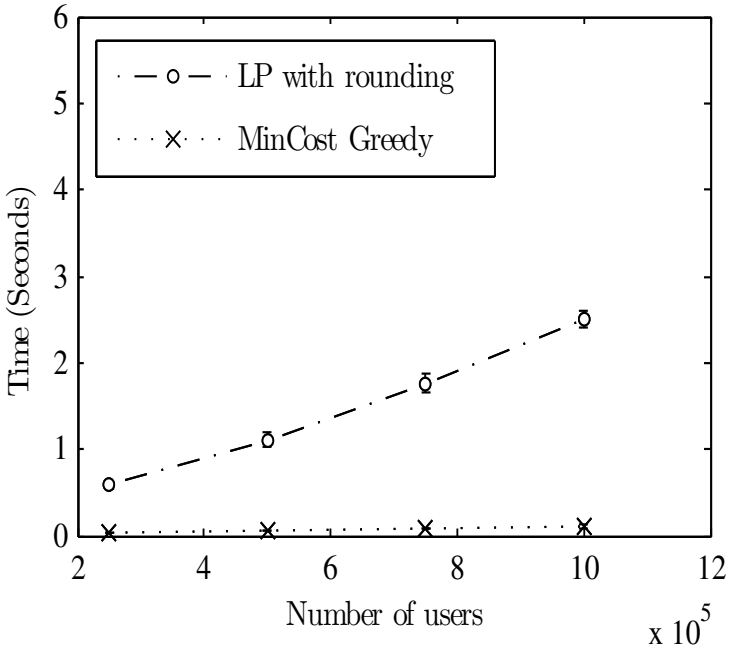

Figure 4.10: Comparison of the execution time of LP with rounding and MinCost Greedy

Table 4.3: Simulation Parameters

Parameter	**Value**
No. of MTC plans ($n1$)	2
No. of TPC plans ($n2$)	2
No. of users (m)	Varying up to $1,000,000$ users in different scenarios
Size of each group	1000
Maximal capacity of MTC plans ($M[i]$)	500,000 users
Costs of MTC and TPC plans ($C[i]$)	Randomly chosen between \$2 and \$8
Reserved Instance volume discounts threshold (h) and discount value (d)	Shown in Table 4.1
User i QoS requirements ($Q_u[i]$) and QoS achievable by cloud plan i ($Q_p[i]$)	Varying between 1 and 4
Cost of buying extra resources from TPC (c_e)	\$0.18 per hour
Cost of buying one reserved instance during one year from TPC (c_r)	\$1000

Core i7 CPU and 10 GB RAM). From this figure, we can see that MinCost Greedy runs much faster than LP with rounding. We can also observe that the execution time of LP with rounding increases as the number of users increases. This is because LP with rounding requires multiple iterations to converge, and the higher the number of users, the higher the number of iterations.

In summary, MinCost Greedy algorithm can efficiently solve the optimal resource scheduling problem.

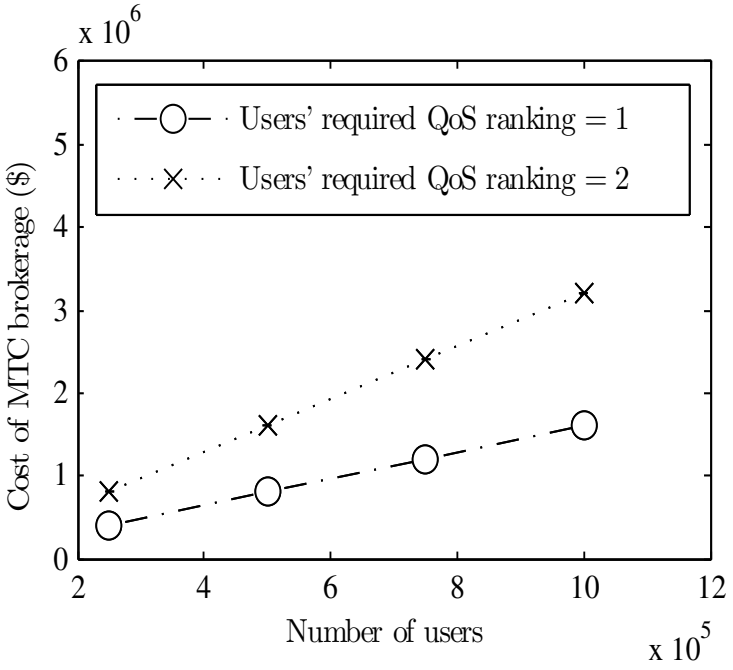

Figure 4.11: Comparison of cost of MTC brokerage with TPC discount 20% and different users' QoS ranking

Total cost with different users' required QoS ranking

In this experiment, we changed the number of users from 200,000 to 1,000,000 and assigned their requests to 4 cloud plans. TPC discount threshold is set to $\$500,000$, and in this discount tier, the TPC discount value is 20%. Figure 4.11 shows the comparison of cost of MTC brokerage with different users' required QoS ranking.

In another experiment, TPC discount threshold is set to $\$1,000,000$, and in this discount tier, the TPC discount value is 40%. Figure 4.12 shows the comparison of cost of MTC brokerage with different users' required QoS ranking.

Region of applicable charging scheme

After finding the optimal resource allocation, MTC brokerage offers cloud plans to mobile users with a price that benefits both users and MTC. To show the applicable charging ranges, we changed the number of users from $200,000$ to $1,000,000$ and assigned their requests to 4 cloud plans. TPC discount threshold is set to $\$500,000$, and in this discount tier, the TPC discount value is 20%. In another test, TPC discount threshold is set to $\$1,000,000$, and in this discount tier, the TPC discount

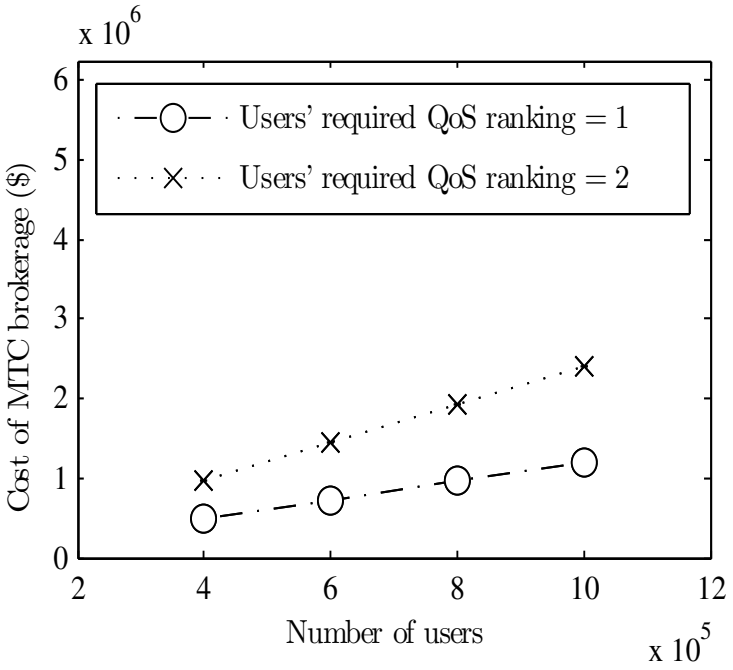

Figure 4.12: Comparison of cost of MTC brokerage with TPC discount 40% and different users' QoS ranking

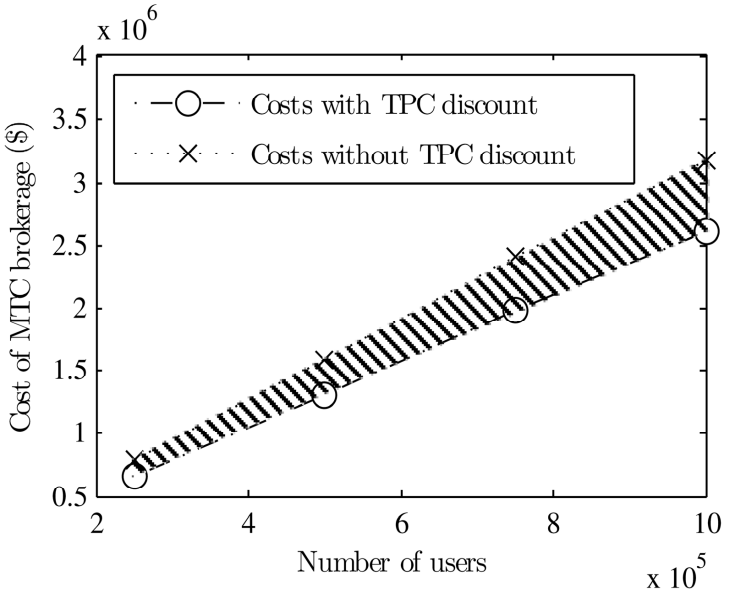

Figure 4.13: Region of applicable charging scheme with TPC discount 20%

value is 40%. The shaded regions in Figure 4.13 and Figure 4.14 shows the region of applicable charging scheme with different TCP discount values. As long as the MTC brokerage offers the users a price in the shaded region, both users and the MTC brokerage can benefit from the offered price.

In addition, by tuning the charging price in the applicable range, the MTC brokerage can actually make adjustment between its profit and the users' savings. Figure 4.15 shows the tradeoffs.

4.6.2 Adaptive Resource Planning in MTC brokerage:

In this section, we show the benefit of adaptive resource planning algorithm. For this study, we use real Google trace data [42] that was released in November 2011 and consists of a 29-day request trace collected from a cluster that contains more than $12,500$ machines. We utilized the data provided in the task event table, where each VM request is called a task, and each task contains information about CPU and memory requirements. In our framework, the input to the algorithm is users' required QoS levels. Therefore, we defined 4 different QoS levels and assigned the tasks' CPU and memory requirements to the QoS levels.

Due to the high number of machines in the trace data, the value of objective function (4.9) is extremely hard to calculate. We thus scale down the system by sampling

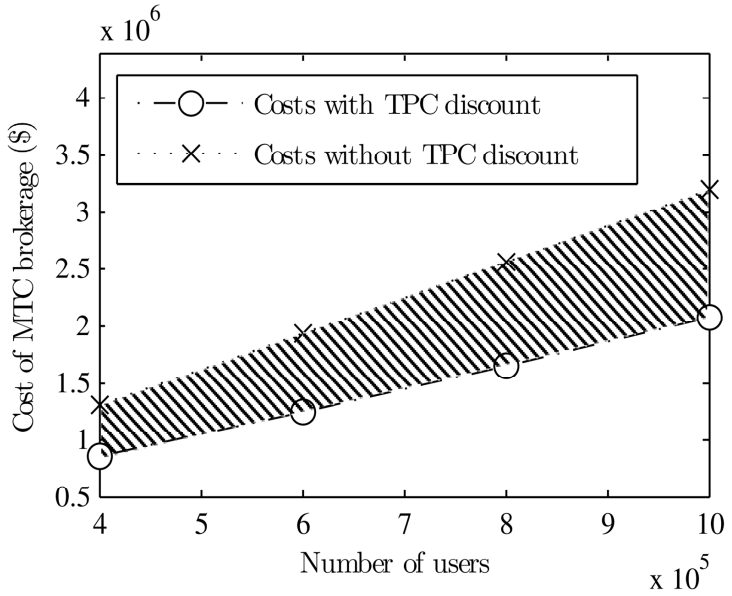

Figure 4.14: Region of applicable charging scheme with TPC discount 40%

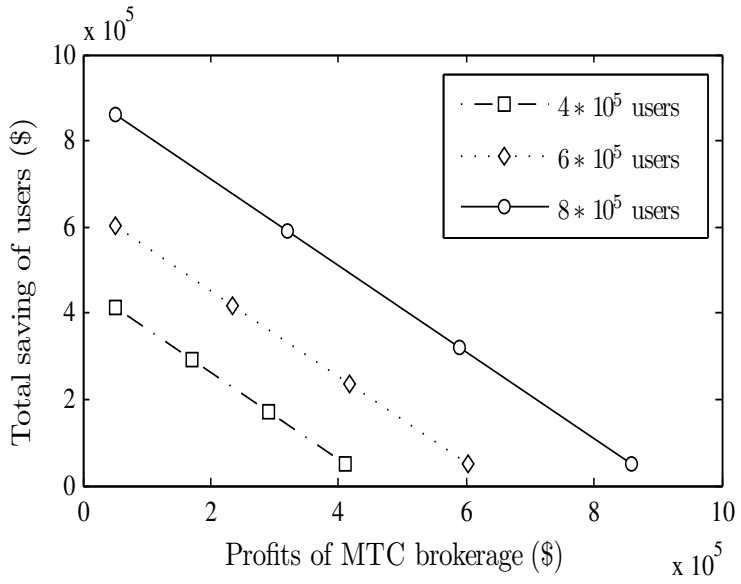

Figure 4.15: Comparison of profits of MTC brokerage and total saving of users with TPC discount 40%

the data uniformly random to obtain a smaller dataset. With the sampled Google trace data, we calculated the number of user request arrivals during different times of the day and built a non-stationary Poisson arrival model to model the stochastic request arrivals. We assume that the service time of each instance follows exponential time with rate 50 requests per hour, and each reserved instance costs $\$1,000$ per year before discount.

Driven by the above model, we test our adaptive resource scheduling strategy. Figure 4.16 shows the comparison of total cost of adaptive resource planning algorithm with different estimations of number of reserved instances. At the beginning of the contract, the MTC brokerage pays the initial payment to TPC for sufficient number of reserved instances for the term of contract, e.g. one year, and whenever there are extra cloud requests that cannot be served with the reserved instances, they will be served with a temporary cloud plan that needs to be purchased without a discount. In this diagram, we changed the number of reserved instances from 700 to 1500, and we set the discount value to 40%. Therefore, according to the number of users' cloud request arrivals, with $s = 700$, there will be an underestimation of number of reserved instances and with $s = 1500$, there will be an overestimation of number of reserved instances. It can be observed that in the long term, by using adaptive resource planning, total cost of MTC brokerage decreases significantly.

4.7 Summary

Mobile Telecom Cloud (MTC) can provide users with high quality of cloud services as telecom companies own the network infrastructure. Therefore, MTC can offer their own telecom cloud services as well as third-party cloud services and provide users with a variety of different cloud services with minimum cost and highest QoS. We study the problem of minimizing the costs and providing the users with their desired QoS in MTC and TPC where telecom companies play the role of cloud brokerage, called the MTC brokerage. We present the system model of MTC brokerage. The cloud service model consist of different cloud service plans offered by MTC and TPC, and each service plan is associated with a QoS level. The TPC pricing model represents the discounts that a TPC such as Amazon EC2 offers if the users' commitments meet the given threshold values. The quality of service model represents users' QoS requirement and QoS level achievable by a cloud plan.

We discuss optimal resource scheduling for MTC brokerage. The problem studies

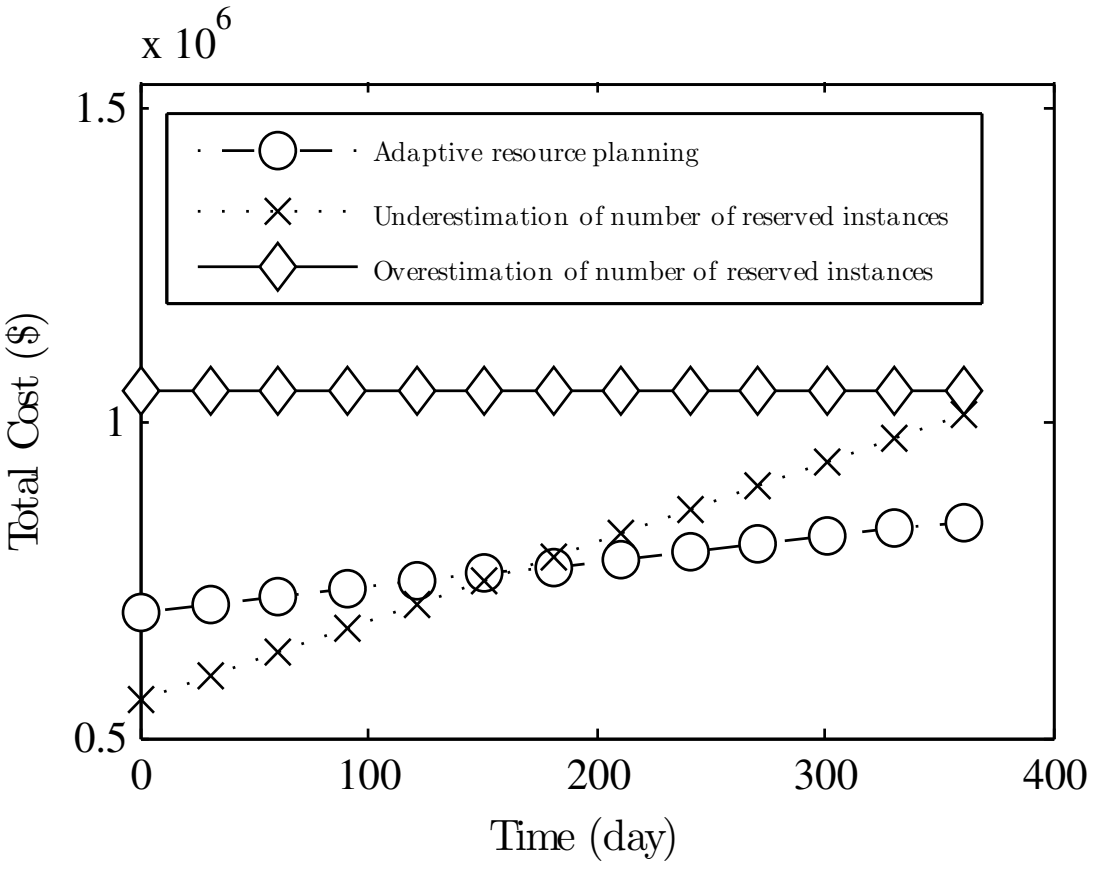

Figure 4.16: Total cost with different estimations of number of reserved instances

how should the MTC brokerage group buy cloud instances from TPC such that the overall service costs are minimized. To solve the optimization problem, we present two algorithms, LP with rounding and MinCost greedy. We also study profitable price range for MTC brokerage. This profitable price range is both attractive to customers and profitable for the MTC brokerage. Therefore, we ensure that for each user, the charged price is less than the price without discount, and the total cost of MTC brokerage is less than the sum of users' prices.

Since there are dynamic changes in users' cloud requests, we study the problem of adaptive resource planning in MTC brokerage. Users may drop their service plans, upgrade their plans, or new users may arrive in the system. However, it is impractical for the MTC brokerage to sign a new contract with the TPC for each customer-initiated change. The adaptive resource planning problem is to decide at current time how the MTC brokerage should update its contract with the TPC. We model this system as a non-stationary Erlang loss system and present a solution for finding the suitable number of reserved instances that minimizes the total cost for the next contract term. Using both simulated data and real Google traces, we show that our proposed resource allocation algorithms achieve considerable cost savings for both the users and the MTC brokerage. One barrier to push MTC into market is security concern, particularly when different users' data and VMs are deployed on the same physical machine. Therefore, in the next chapter, we address security aware provisioning and migration of phone clones in MTC.

Chapter 5

SWAP: Security Aware Provisioning and Migration of Phone Clones Over Mobile Clouds

5.1 Motivation

A telecom company can leverage its advantages indicated earlier to provide its customers with high-quality and more attractive cloud computing services. One type of such services is to create phone clones within cloud computing to help mobile users augment the functionality of their smart phones and increase the lifetime of the battery.

The concept of phone clones [15] is to build software clones of smart phones on the cloud and enable mobile users to offload computation intensive tasks and backup data to the cloud. Current smartphone devices are normally installed with many useful applications to make users' daily life much easier and more efficient than before. Some applications, however, may require intensive calculation and consume a large amount of energy. In this case, the smartphone could offload the tasks to its clone in the cloud [13, 19, 39].

Depending on different functionalities supported by the phone clones, they could be implemented as a process or a *thin* virtual machine (VM) [15, 39, 46] on a cloud host. Due to the ease of management and the richer functionalities of VM, we assume the VM version of phone clones in this chapter. Fig. 5.1 shows an example architecture of the system. To allow resource multiplexing, multiple VMs are usually allocated

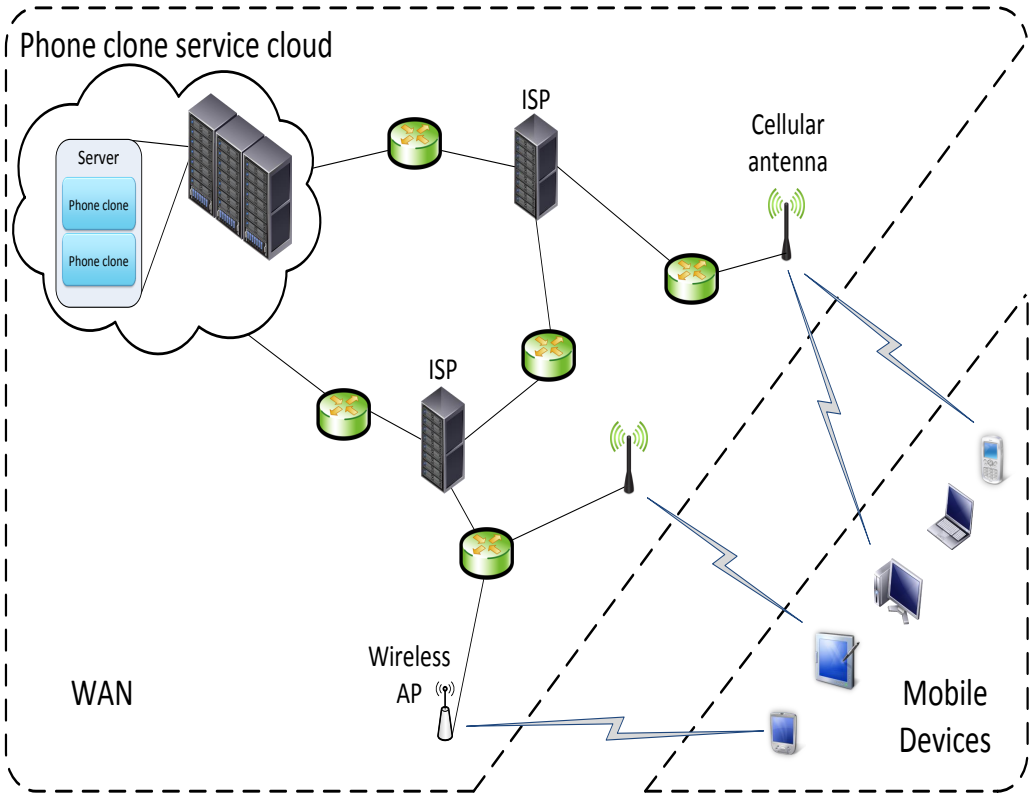

Figure 5.1: System architecture of phone clones in mobile cloud

and managed with a hypervisor, such as KVM [5] or Xen [7], in one physical machine. Due to the large number of mobile users, it is not surprising if one hypervisor hosts hundreds of phone clones. Under such circumstances, provisioning and migration strategies for phone clones become critical to the success of mobile telecom cloud.

We need to tackle two constraints in the allocation and migration of phone clones: security and resource. To give users' a reasonable sense of security, phone clones should be physically isolated. For example, users should feel more comfortable if their phone clones are co-located with those of their friends rather than strangers. Nevertheless, due to the limited number of physical hosts and the large number of mobile users, it may not be possible to find a good isolation for all phone clones. As a result, a phone clone may have to live together with other strangers' phone clones on the same host.

It has been shown that a VM can attack another VM on the same host via covert channel attacks [43, 58]. Such attacks exploit the CPU cache or the memory

bus in a virtualized environment to steal information from other VMs. It has been demonstrated in [43, 58] that covert channel can be effective and very hard to detect. The big challenge we are faced with in mobile cloud is: how to *mitigate* the risk of covert channel attacks when we do not have enough resource to physically isolate strangers' phone clones?

In this chapter, we present our strategy to tackle this problem. We propose and evaluate SWAP: a security aware provisioning and migration scheme for phone clones. For the provisioning of new phone clones, we take advantage of the mobile telecom cloud where it is easy to build a communication graph based on mobile users' communication history. The communication graph reflects the relationship between mobile users, and it should be safe to allocate together the phone clones that have a direct communication link, since they need to communicate anyway. Whenever this requirement cannot be met due to resource constraint, we then solve the optimization problem that minimizes the risk posed by potential covert channels. Our way of mitigating the negative impact of covert channels comes from the observation that covert channels normally need time to build [28, 65]. By constraining the co-locating time duration of two strangers' phone clones on the same host, we can counter the host limitation problem by migrating strategically selected VMs.

The contributions of this chapter are three-fold.

- First, we propose a system model that captures the mobile users' communication relationship and the potential risks when co-locating phone clones, and solve the optimization problem that minimizes the risks in the provisioning of new phone clones. The optimization problem requires intensive computations due to the large number of phone clones. To avoid this problem, we present a clique-covering method to pre-process the communication graph and significantly speed up the optimization algorithm.

- Second, we propose a phone clone migration strategy to mitigate the impact of potential covert channels. We introduce a decay function to model the time-varying feature of covert channels, and migrate some phone clones whenever the risk among the phone clones in a host becomes high. In this context, we minimize the total number of migrations to meet a given security requirement.

- Third, we evaluate our solution using Reality Mining dataset collected from 100 mobile phones over the course of 9 months in the Reality Mining project

undertaken at the MIT Media Laboratory [20] and Nodobo dataset collected from a group of 27 students at a high school [14].

The rest of the chapter is organized as follows. We explain the system model in Section 5.2. We solve the optimal phone clone provisioning problem in Section 5.3 and present a dynamic phone clone migration strategy and the related algorithms in Section 5.4. We perform comprehensive performance evaluation of SWAP in Section 5.5.

5.2 System Model

5.2.1 Communication Model

We assume that the system has m phone clones and n hosts. Based on the communication history of phone clones, we can build a communication graph $G =< V, E >$, where V denotes the set of nodes representing the phone clones and E denotes the set of edges representing the communication between phone clones. When two phone clones i and j communicate with each other, we establish a link between them. The communication graph can be represented with an adjacency matrix $A = [a_{ij}]_{m \times m}$, where $a_{ij} = 1$ indicates that there is a communication link between phone clones i and j and $a_{ij} = 0$ otherwise. To facilitate later calculation, we assume that $a_{ii} = 1$ for any phone clone i.

We use the above simple communication graph for ease of illustration of problem formulation. The model can be easily extended by introducing trust weights for communication links and by introducing the dynamic changes of communication graphs. Nevertheless, the core problems addressed in this chapter remain the same, and all the algorithms proposed in the chapter are applicable with slight modifications.

5.2.2 Potential Risk and Its Calculation

Definition 1. *(Potential Risk): When two phone clones that do not have a direct communication link between them are allocated to the same physical host, the allocation scheme may introduce a potential breach. Given a communication graph $G =< V, E >$, the potential risk of a phone clone allocation scheme, $\mathcal{I}$, is the total number of potential breaches introduced by the scheme.*

Remark 5. *In this thesis, we assumed that two phone clones with a communication link don't attack each other when they are collocated in the same host. We can relax this assumption and add a weight to communication links to address different scenarios such as that two phone clones having a communication link are possible to attack each other when collocated in the same host.*

We use an allocation matrix $X = [x_{ij}]_{m\times n}$ to represent a phone clone allocation scheme, where $x_{ij} = 1$ indicates that phone clone i is allocated to host j and $x_{ij} = 0$ otherwise. Given the adjacency matrix A and the allocation matrix X, we wish to mathematically calculate the potential risk of the phone clone allocation scheme.

It turns out that the calculation could be well formulated with matrix trace. We use $\bar{A}$ to denote the complementary matrix of A, i.e., $\bar{A} = [\bar{a}_{ij}]_{m\times m} = [1 - a_{ij}]_{m\times m}$. We also define an $n \times n$ risk indicator matrix, $I = X^T\bar{A}X$, where X^T represents the transpose of matrix X. Then, the potential risk of the phone clone allocation scheme can be calculated with

$$\mathcal{I} = \frac{1}{2}tr(I) = \frac{1}{2}tr(X^T\bar{A}X), \tag{5.1}$$

where $tr(\cdot)$ represents the trace of a matrix.

As a simple example to explain Equation (5.1), Fig. 5.2(a) shows the communication graph of 4 phone clones. In this example, phone clone 1 and phone clone 4 are placed on host 1, and phone clone 2 and phone clone 3 are placed on host 2. Since there is no communication edge between phone clones 1 and 4 in the communication graph, and they are placed on the same host, the potential risk of this allocation scheme is 1 for this example. Fig. 5.2(b) illustrates the calculation of potential risk with matrices A and X.

For ease of reference, the main notations used in the chapter are listed in Tables 5.1.

5.3 Minimizing The Potential Risk in Phone Clone Allocation

5.3.1 Problem Formulation

For provisioning a new system, we first need to solve the following problem to minimize the risk of a phone clone allocation scheme.

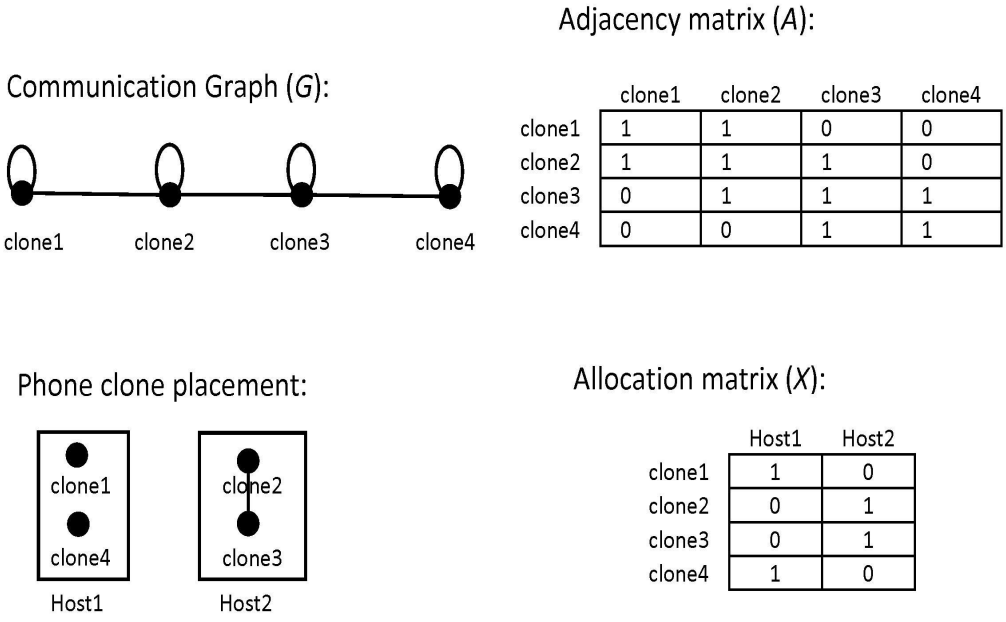

Figure 5.2: (a) An example of system model and phone clone placement (b) The calculation of potential risk

Problem 1.

$$\underset{X=[x_{ij}]_{m\times n}}{\text{Min}} \quad \mathcal{I} = \frac{1}{2} tr(X^T \bar{A} X) \tag{5.2}$$

subject to

$$x_{ij} \in \{0, 1\} \tag{5.3}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \ \textit{for } i = 1, 2, \ldots, m \tag{5.4}$$

$$\sum_{i=1}^{m} x_{ij} \leq M[j], \ \textit{for } j = 1, 2, \ldots, n \tag{5.5}$$

Equations (5.3) and (5.4) together assume that a phone clone is allocated to only one host. Inequality (5.5) indicates that the total number of phone clones assigned to a host should be less than its maximum capacity.

The potential risk is tightly associated with the sparsity of the communication graph. We disclose this relationship with the following lemma and theorem, the proofs of which are provided in Subsection 5.3.2.

Lemma 1. *Assume that m phone clones are allocated to n hosts, and that the communication graph is (k, h)-sparse*[1]*. The potential risk of any reasonable phone clone allocation scheme*[2] *satisfies the following condition:*

$$\mathcal{I} \geq \frac{m^2}{2n} - \frac{m}{2} - (km - nh). \tag{5.6}$$

Theorem 1. *Assume that m phone clones are allocated to n hosts, and that the communication graph is a random graph with link probability of p_l (i.e., each pair of nodes has a link with probability of p_l). For $\forall h > 0$, the potential risk of any phone clone allocation scheme satisfies the following probabilistic lower bound:*

$$Pr\{\mathcal{I} \geq (1 - p_l)(\frac{m^2}{2n} - \frac{m}{2}) - p_l h \frac{m - n}{2}\} \geq 1 - e^{-p_l h^2 (\frac{1}{2} - \frac{1}{2m})/3}. \tag{5.7}$$

Remark 6. *With slight changes, Problem 1 can be used to address more complicated problem settings. For instance, it can be easily extended to study the case where*

[1] A graph $G = (V, E)$ with m vertices is called (k, h)-sparse if every subset of $m' \leq m$ vertices spans at most $km' - h$ edges.

[2] An allocation scheme is called reasonable if no host is left empty.

Table 5.1: List of Notations

Notations for Phone Clone Allocation (Section 5.3)	
Notation	**Definition**
m	Number of phone clones
n	Number of hosts
p	Number of cliques
$A = [a_{ij}]$	Adjacency matrix of communication graph
$X = [x_{ij}]$	Allocation matrix
I	Indicator matrix
$\mathcal{I} = tr(I)$	Potential risk
$R = [r_{ij}]$	Risk matrix
$M[i]$	Maximum capacity of host i
s	Column vector where s[i] represents the number of phone clones in clique i
Notations for Phone Clone Migration (Section 5.4)	
$f(t)$	Decay function
$D(t) = [d_{ij}(t)]$	Decay matrix at time t
$X(t) = [x_{ij}(t)]$	Allocation matrix at time t
$I(t)$	Risk indicator matrix at time t
$\mathcal{I}_v(t)$	Time-variant risk indicator vector
δ	Risk threshold
q	Total number of risky phone clones
l	Number of risky phone clones on the host under investigation
Z	A $l \times 1$ vector where $Z[i, 1] = 0$ indicates phone clone i is selected for migration and 1 otherwise.
R_1	$q \times n$ matrix representing the potential risk between a risky phone clone and a host
R_2	$q \times q$ matrix representing the potential risk between two risky phone clones
U	$q \times n$ reallocation matrix where $u_{ij} = 1$ indicates risky phone clone i is assigned to host j and 0 otherwise.
$C[i]$	Remaining capacity of host i

different phone clones demand different computing/memory resources, or the case where a user has different levels of trust with respect to other users.

Remark 7. *Problem 1 is a quadratic integer programming problem. It is NP-hard [44], since $\bar{A}$ could be indefinite for a general communication graph. The common way to obtain a lower bound is to use Linear Programming (LP) relaxations. The problem, however, translates to the problem of relaxing Quadratic programming with LP involving indefinite constraints, which is still an open issue [41].*

In a real system, the number of phone clones is usually large. To speed up the processing, we propose another heuristic in the next section.

5.3.2 Proofs

Proof of Lemma 1

Proof. Given a phone clone allocation scheme $m_1, m_2, \ldots, m_n$, where $m_i (i = 1, 2, \ldots, n)$ is the number of phone clones allocated to host i, we have $m_i > 0$ and $m = \sum_{i=1}^{n} m_i$. Since the communication graph is (k, h)-sparse, every subset of $m' \leq m$ vertices spans at most $km' - h$ edges. Because the total number of possible links among the phone clones on host i is $\frac{m_i(m_i-1)}{2}$, the potential risk at host i, denoted by $\mathcal{I}_i$, satisfies:

$$\mathcal{I}_i \geq \frac{m_i(m_i - 1)}{2} - (km_i - h).$$

We thus obtain the potential risk of the allocation scheme:

$$\begin{aligned}
\mathcal{I} &= \sum_{i=1}^{n} \mathcal{I}_i \\
&\geq \sum_{i=1}^{n} \frac{m_i^2}{2} - \sum_{i=1}^{n} \frac{m_i}{2} - \sum_{i=1}^{n} (km_i - h) \\
&\geq \frac{m^2}{2n} - \frac{m}{2} - (km - nh)
\end{aligned}$$

□

Proof of Theorem 1

Proof. Given a phone clone allocation scheme $m_1, m_2, \ldots, m_n$, where $m_i (i = 1, 2, \ldots, n)$ is the number of phone clones allocated to host i, we have $m_i > 0$ and $m = \sum_{i=1}^{n} m_i$.

Denote the number of links among the phone clones on host i as $\mathcal{L}_i$. Since the communication graph is a random graph with link probability of p_l,

$$E[\mathcal{L}_i] \equiv \mu = p_l \frac{m_i(m_i-1)}{2}. \tag{5.8}$$

Since the links in a random graph are independent, based on Chernoff bound [40], $\forall \epsilon_i h > 0 (\epsilon > 0)$,

$$Pr\{\mathcal{L}_i \geq \mu(1+\epsilon_i h)\} < (\frac{e^{\epsilon_i h}}{(1+\epsilon_i h)^{(1+\epsilon_i h)}})^{\mu}$$

Since $\epsilon_i h > 0$, we can use $e^{-\mu(\epsilon_i h)^2/3}$ to approximate $(\frac{e^{\epsilon_i h}}{(1+\epsilon_i h)^{(1+\epsilon_i h)}})^{\mu}$. When $0 < \epsilon_i h \leq 1$, we have $(\frac{e^{\epsilon_i h}}{(1+\epsilon_i h)^{(1+\epsilon_i h)}})^{\mu} \leq e^{-\mu(\epsilon_i h)^2/3}$ (Refer to Theorem 4.4 in [40]). When $1 < \epsilon_i h$, the maximum gap between $e^{-\mu(\epsilon_i h)^2/3}$ and $(\frac{e^{\epsilon_i h}}{(1+\epsilon_i h)^{(1+\epsilon_i h)}})^{\mu}$ is less than 3%. Therefore, it is appropriate to use $e^{-\mu(\epsilon_i h)^2/3}$ to approximate $(\frac{e^{\epsilon_i h}}{(1+\epsilon_i h)^{(1+\epsilon_i h)}})^{\mu}$. We have

$$Pr\{\mathcal{L}_i \geq \mu(1+\epsilon_i h)\} \leq e^{-\mu(\epsilon_i h)^2/3}$$

Equivalently,

$$Pr\{\mathcal{L}_i < \mu(1+\epsilon_i h)\} \geq 1 - e^{-\mu(\epsilon_i h)^2/3}$$

Following the same reasoning in Lemma 1, the potential risk at host i, denoted by $\mathcal{I}_i$, satisfies:

$$Pr\{\mathcal{I}_i \geq \frac{m_i(m_i-1)}{2} - \mu(1+\epsilon_i h)\} \geq 1 - e^{-\mu(\epsilon_i h)^2/3} \tag{5.9}$$

Since Inequality (5.9) holds for all $\epsilon_i > 0$, we set

$$\epsilon_i = \frac{1}{m_i} \tag{5.10}$$

Replacing (5.8) and (5.10) into (5.9), we have

$$\begin{aligned} Pr\{\mathcal{I}_i &\geq \frac{m_i^2 - m_i}{2}(1 - p_l(1 + h\frac{1}{m_i})\} \geq 1 - e^{-\mu(\epsilon_i h)^2/3} \\ Pr\{\mathcal{I}_i &\geq \frac{m_i^2 - m_i}{2}(1-p_l) - \frac{m_i - 1}{2} p_l h\} \geq 1 - e^{-p_l h^2(\frac{1}{2} - \frac{1}{2m})/3} \end{aligned} \tag{5.11}$$

Note that the right-hand side is due to the fact that $\frac{1}{m} \leq \frac{1}{m_i}$. Since $\mathcal{I} = \sum_{i=1}^{n} \mathcal{I}_i$, we

have

$$Pr\{\mathcal{I} \geq \sum_{i=1}^{n} \frac{m_i^2 - m_i}{2}(1-p_l) - \sum_{i=1}^{n} \frac{m_i - 1}{2} p_l h\}$$
$$\geq 1 - e^{-p_l h^2(\frac{1}{2} - \frac{1}{2m})/3} \tag{5.12}$$

That is,

$$Pr\{\mathcal{I} \geq (1-p_l)(\frac{m^2}{2n} - \frac{m}{2}) - p_l h \frac{m-n}{2}\} \geq 1 - e^{-p_l h^2(\frac{1}{2} - \frac{1}{2m})/3}.$$

□

5.3.3 Reducing Model Complexity with Cliques

Since phone clones that already have communication links are unlikely to attack each other with covert channels, we could consider them as a group and allocate them to the same host. Mathematically, we need to find a clique cover of the communication graph. Since the problem of covering a graph with the minimum number of cliques is proven NP-complete and even does not allow constant approximation [66], we adopt the following well-known heuristic method to obtain an approximate solution:

We iteratively search for cliques that cover more vertices that have not been covered so far. Heuristically, the vertices with larger degrees may have a better chance of appearing in larger cliques. Thus, the search starts from the vertex with the largest degree, until all vertices are covered.

After partitioning the graph into p cliques, we treat the cliques as the basic allocation units, and adjust the measure of potential risk accordingly. If there is no link between a node i in one clique and a node j in another clique, we count a missing edge between the two cliques. We introduce a risk matrix $R = [r_{ij}]_{p \times p}$ where p is the number of cliques after the partition and r_{ij} is the total number of missing edges between clique i and clique j. If we allocate phone clones with cliques as the basic allocation unit, the indicator matrix, $I = X^T R X$. Here we slightly abuse the notation of X by reusing $X = [x_{ij}]_{p \times m}$ to denote the allocation matrix with cliques as the basic allocation unit, because it is easy to figure out the dimension of X from the context.

Therefore, the potential risk of an allocation scheme, $\mathcal{I}$, could be calculated with

$$\mathcal{I} = \frac{1}{2} tr(I) = \frac{1}{2} tr(X^T R X). \tag{5.13}$$

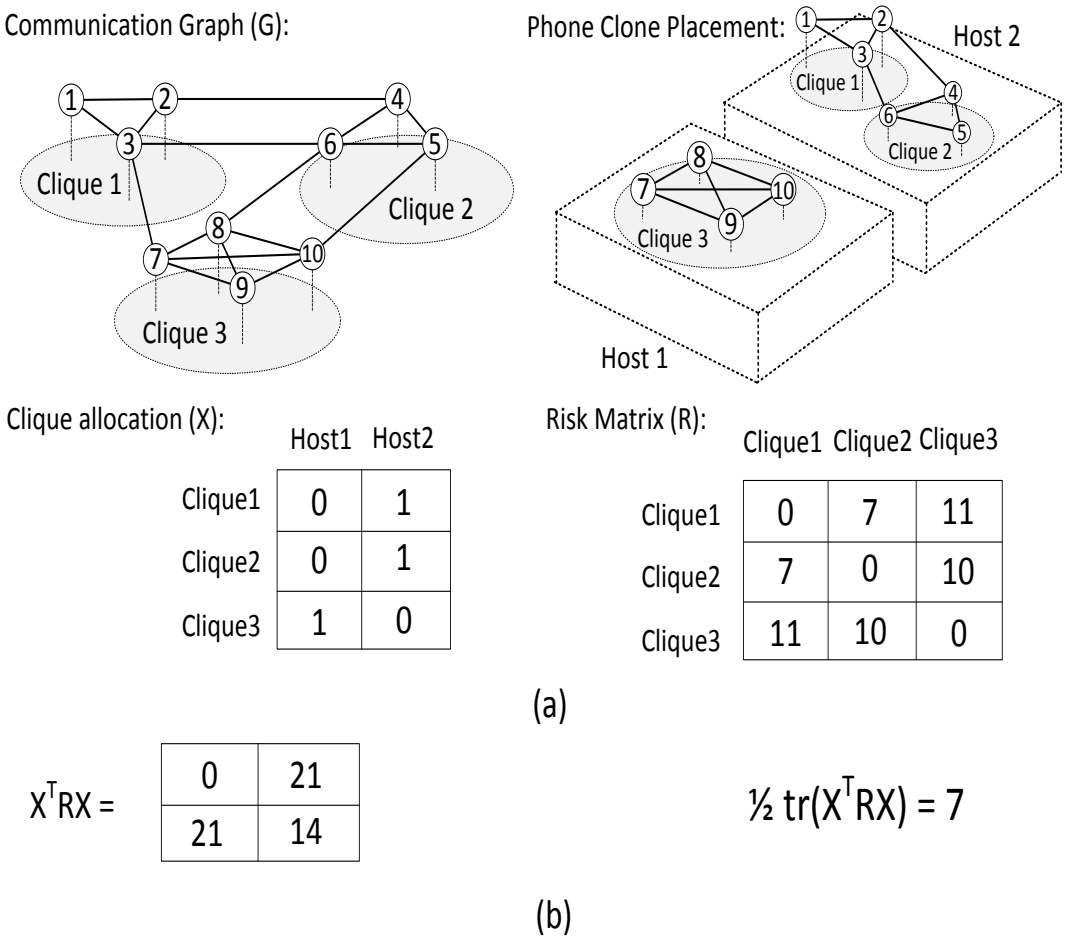

Figure 5.3: (a) An example clique-based phone clone placement (b) The calculation of potential risk

For example, Figure 5.3 shows the communication graph of 10 phone clones. In this example, the graph is partitioned into clique 1, clique 2, and clique 3. After partitioning, clique 3 is placed on host 1, and clique 1 and clique 2 are placed on host 2. The potential risk of this allocation scheme is 7.

We thus consider the following optimization problem:

Problem 2.

$$\underset{X=[x_{ij}]_{p\times n}}{\text{Min}} \quad \mathcal{I} = \frac{1}{2} tr(X^T R X) \tag{5.14}$$

subject to

$$x_{ij} \in \{0, 1\} \tag{5.15}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \ \textit{for } i = 1, 2, \ldots, p \tag{5.16}$$

$$\sum_{i=1}^{p} x_{ij} s[i] \leq M[j], \ \textit{for } j = 1, 2, \ldots, n \tag{5.17}$$

where $X = [x_{ij}]$ is an $p \times n$ allocation matrix where $x_{ij} = 1$ indicates clique i is assigned to host j and 0 otherwise. Equations (5.15) and (5.16) together assume that a clique is allocated to only one host. Inequality (5.17) indicates that the total number of phone clones assigned to a host should be less than its maximum capacity.

Problem 2 has the same structure as Problem 1 but has less number of parameters. Since it is also NP-hard [44], we propose a heuristic algorithm, called *QP with Rounding*, as follows:

QP with Rounding: We relax the requirement (5.15) to allow x_{ij} as real numbers between 0 and 1; we then solve the relaxed problem using quadratic programming. For the x_{ij} values that are either close to 1 or close to 0, we round them up or down to 1 and 0, respectively. In the next round of iteration, we fix those rounded values, and perform the same procedure again until the convergence condition is met.

5.3.4 Proposed Heuristic Algorithms

Since QP with rounding may be slow when number of phone clones are large, we introduce various heuristic algorithms to solve the problem.

maximum-conflict-first (MCF): We propose a greedy algorithm which is shown in Fig. 5.4. The main idea is to handle the "hard" ones first, i.e., the phone clones that

1: Sort phone clones in the ascending order of their node degree in the communication graph;
2: **for** each phone clone i in the sorted list **do**
3: Find the host that has the minimum number of phone clones in conflict with phone clone i;
4: Allocate phone clone i to this host;
5: **end for**

Figure 5.4: The maximum-conflict-first (MCF) algorithm

1: Sort phone clones in the descending order of their node degree in the communication graph;
2: **for** each phone clone i which has the highest degree and has not been allocated **do**
3: Find host j that has the maximum number of common edges between phone clone i and the phone clones in host j and the minimum number of missing edges between phone clone i and the phone clones in host j;
4: Allocate phone clone i to host j.
5: **end for**

Figure 5.5: The highest-degree-first (HDF) algorithm

have more conflicts with others are allocated first.

highest-degree-first (HDF): This greedy algorithm allocate the phone clones in three steps:

Step 1: Sort the phone clones in the graph in decreasing order of nodes degree.

Step 2: Select the phone clone i which has the highest degree and has not been allocated.

Step 3: Assign the selected phone clone i to the host that has the highest number of common edges between the phone clone i and the phone clones in that host.

Assume phone clone i is selected in Step 2. For implementing the Step 3, we use the following heuristic value: For each host j, we calculate the difference of the following two values: (number of edges between phone clone i and the phone clones in host j) minus (number of missing edges between phone clone i and the phone clones in host j). Then, we assign the phone clone i to the host which has the highest value. The main idea of the HDF algorithm is shown in Fig. 5.5.

constrained-highest-degree-first (CHDF): CHDF is similar to HDF, except that it keeps the number of phone clones in a host less than the threshold (m/n). In other words, it tries to spread the phone clones as evenly as possible among the hosts. Therefore, when the number of assigned phone clones in a host reaches the threshold (m/n), it does not assign any more phone clones to that host.

5.4 Dynamic Migration of Phone Clones

5.4.1 The Problem and The Basic Idea

In the previous section, we considered the *static* phone clone allocation problem. In practice, covert channels need time to build [65] [28]. As such, phone clones, even those belonging to strangers, can be safely allocated together during a certain time period. In other words, the risk of covert channels is time dependent.

To address dynamic risk management over time, we consider the fact that the risk of covert channel attacks becomes higher when two phone clones that have no communication link stay too long on the same host. We introduce a *decay function* $f(t)$ and a decay matrix $D(t) = [d_{ij}]_{m\times m}$ for this purpose. If phone clones i and j have a communication link, then $d_{ij}(t) = 1$, otherwise $d_{ij}(t) = f(t)$. The function $f(t) \in [0, 1]$ is the decay function where t indicates the time duration when phone clones i and j stay on the same host. It is a non-increasing function with $f(0) = 1$.

Accordingly, we use $\bar{D}(t)$ to denote the complementary matrix of $D(t)$, i.e., $\bar{D}(t) = [\bar{d}_{ij}(t)]_{m\times m} = [1 - d_{ij}(t)]_{m\times m}$, and $X(t)$ to denote the allocation matrix at time t. The $n \times n$ indicator matrix also becomes time variant and is calculated as $I(t) = X^T(t)\bar{D}(t)X(t)$.

We introduce a time-variant risk indicator vector to model the dynamic risk states of the system:

Definition 2. *(Time-variant risk indicator vector): Assume that at time t, the indicator matrix of the system is $I(t)$. The time-variant risk indicator vector of the system at time t, $\mathcal{I}_v(t) = \frac{1}{2}diag(I(t))$, where $diag(.)$ returns a vector with values as the diagonal elements of a matrix. The i-th value in $\mathcal{I}_v(t)$ reflects the risk value on host i at time t.*

Note that unlike potential risk, which is a constant for a given allocation scheme, the values in $\mathcal{I}_v(t)$ changes over time.

We call the system is resilient to covert channel attacks if the following condition holds:

$$max(\mathcal{I}_v(t)) \leq \delta, \tag{5.18}$$

where $max(.)$ returns the largest element in a vector. When this condition is violated, we need to migrate some phone clones and answer the following problem: *how can we maintain the resilience of the system with the minimum number of phone clone migrations?*

To solve the problem, we propose an event-driven heuristics method where phone clone migration is triggered by the violation of condition (5.18). Once the phone clone migration is required, the process to follow consists of three steps:

1. Identify hosts with the risk value higher than the threshold. Theses hosts are called risky hosts.

2. Select phone clones from the risky hosts for migration. The selected phone clones are called risky phone clones.

3. Reallocate the risky phone clones.

5.4.2 Step 1: Selection of Risky Hosts

After obtaining the time-variant risk vector $\mathcal{I}_v(t)$, the risky hosts can be identified easily by checking the i-th value ($1 \leq i \leq n$) in the vector. Note that the i-th value in the time-variant risk indicator vector represents the risk value of host i.

5.4.3 Step 2: Selection of Risky Phone Clones

In this step, we determine the risky phone clones that need to migrate to other hosts.

For each risky host, we calculate the minimum number of phone clones for migration such that the risk value on the host remains below the threshold δ. Assume that there are l phone clones in a chosen risky host k. We construct a $l \times l$ submatrix of $D(t)$, denoted as $D_s(t)$, which is obtained by selecting the l rows and l columns from $D(t)$ corresponding to the phone clones in risky host k. We consider the complementary matrix $\overline{D}_s(t) = \mathbf{1}_{\boldsymbol{l}\times\boldsymbol{l}} - D_s(t)$, where $\mathbf{1}_{l\times l}$ is a $l \times l$ matrix with all 1s.

We try to minimize the number of migrations such that the risk indicator value on host k remains below the threshold δ. We thus consider the following optimization problem:

Problem 3.

$$\underset{Z_{l\times 1}}{\text{Min}} \ -\mathbf{1}_{1\times l} Z \tag{5.19}$$

subject to

$$z_i \in \{0, 1\} \tag{5.20}$$

$$\frac{1}{2} Z^T \ \overline{D}_s(t) \ Z \leq \delta, \tag{5.21}$$

where Z is a $l \times 1$ vector where $Z[i, 1] = z_i = 0$ indicates phone clone i is selected for migration and 1 otherwise. Equation (5.19) tries to minimize the number of phone clone migrations. Inequality (5.21) ensures that the risk of the remaining phone clones on the host is no larger than the threshold δ.

Problem 3 belongs to the category of integer linear programming with quadratic constraints, which is hard to solve. We propose a greedy algorithm as follows: At each step, the greedy algorithm selects a phone clone with the maximum risk value, and removes it from the host until the risk value on the host drops below the threshold δ. We run the above algorithm for each risky host separately and select the risky phone clones from each risky host. In the next step, we try to reallocate these risky phone clones to minimize the total risk in the system.

5.4.4 Step 3: Migration of Risky Phone Clones

In this step, we reallocate the risky phone clones to the other hosts to minimize the potential risk. Suppose, in the previous step, we have selected a total of q risky phone clones from risky hosts. To migrate the risky phone clones, we need to consider the potential risk of allocating phone clones to hosts and the potential risk among these q risky phone clones.

The potential risk between a risky phone clone and a host can be measured by the matrix $R_1 = [r^1_{ij}]_{q\times n}$, where r^1_{ij} is the total number of missing edges in the communication graph between the risky phone clone i and the phone clones that are currently residing on host j.

There is another potential risk when reallocating two risky phone clones to the same host. The potential risk between two risky phone clones is measured by the matrix $R_2 = [r^2_{ij}]_{q\times q}$, such that $r^2_{ij} = 0$ if the risky phone clone i and the risky phone clone j have a communication link, and 1 otherwise.

We need to solve the following problem:

Problem 4.

$$\underset{U=[u_{ij}]_{q\times n}}{\text{Min}} \quad \frac{1}{2}tr(U^T R_2 U) + \frac{1}{2}tr(R_1^T U) \tag{5.22}$$

subject to

$$u_{ij} \in \{0, 1\} \tag{5.23}$$

$$\sum_{j=1}^{n} u_{ij} = 1, \ \textit{for } i = 1, 2, \ldots, q \tag{5.24}$$

$$\sum_{i=1}^{q} u_{ij} \leq C[j], \ \textit{for } j = 1, 2, \ldots, n \tag{5.25}$$

where $U = [u_{ij}]$ is a $q \times n$ reallocation matrix where $u_{ij} = 1$ indicates risky phone clone i is assigned to host j and 0 otherwise.

Equation (5.22) is the objective function to measure the risk of reallocation scheme, in which the first term $\frac{1}{2}tr(U^T R_2 U)$ measures the potential risk of reallocating two risky phone clones to the same host, and the second term $\frac{1}{2}tr(R_1^T U)$ measures the potential risk in reallocating a risky phone clone to a host. Equations (5.23) and (5.24) assume that a risky phone clone is reallocated to only one host. Inequality (5.25) indicates that the total number of risky phone clones assigned to a host should be less than its remaining capacity.

Mathematically, Problem 2 and Problem 4 belong to the same category of integer quadratic programming, and thus can be solved with the same heuristic algorithm introduced in Section 5.3.

Remark 8. *One point of concern is the convergence problem in the phone clone migration: the migration of a risky phone to a new host might trigger more migrations in the new host, and as such the migration process may create a cascading effect for phone clone migration in a short time. This problem, however, will not happen, because phone clone migration is triggered by the violation of condition (5.18). By default, the decay function $f(0) = 1$. That means, when a risky phone clone is reallocated to a new host, it will not trigger further migration in a short period of time. Further migration is required in the future only when strangers' phone clones stay too long in the same host. By minimizing the potential risk calculated with (5.22), we minimize the chance for future migration.*

5.5 Performance Evaluation

5.5.1 Simulation Model

We perform comprehensive simulation to evaluate the performance of our phone clone allocation and migration algorithms.

We need to select a decay function $f(t)$ to model the risk of co-placing a pair of phone clones that do not have mutual trust for a continuous time interval t. Zhang et al. [65] and Gligor et al. [28] pointed out that time is needed to establish cross-VM covert channels, including the time spent on compromising the victim VM which leaks data. The time required to establish a covert channel between an arbitrary pair of co-resident VMs is random. Without pre-knowledge of such a random variable, we resort to the common practice that assumes the random variable to be normally distributed. Therefore, the decay function can be modeled using the cumulative distribution function of normal distribution. To simplify the calculation, we estimate this decay function using a Sigmoid function as follows:

$$f(t) = 1 - \frac{1}{1 + \theta \cdot e^{-b(t-a)}} \ , \tag{5.26}$$

where θ is a fixed large positive number to bring $f(0)$ close to 1. Then, a and b are adjustable parameters, which control the shape of the decay function. Note that the proposed decay function is just a good example, and other types of decay functions could be used. In our simulation, we set $\theta = 100000$, $a = 0$, and $b = 1$. Note that we only use the above function as an example, and other functions could be applied within our framework. The "right" selection of decay function depends on the statistical analysis and modelling of covert channel attacks and is beyond the scope of this chapter.

We test our algorithms with different communication graph models: the random graph model, the Barabási-Albert (BA) graph model [8], MIT Reality Mining dataset [20, 21], and Nodobo dataset [14]. In the random graph model, each pair of phone clones are assigned a communication link with a given probability. In the Barabási-Albert graph model, the graph begins with an initial connected network of n_0 phone clones. New phone clones are added one at a time, and each new phone clone is connected to n_e, ($n_e \leq n_0$), existing phone clones with a probability that is proportional to the number of links that the existing phone clones already have.

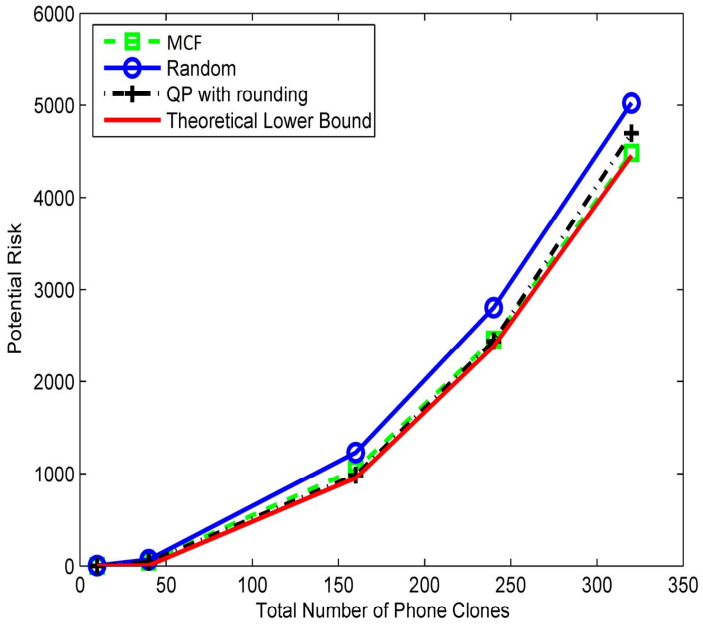

Figure 5.6: Potential risk of phone clones with the random communication graph; no. of hosts=5; maximum capacity of each host= 100 phone clones

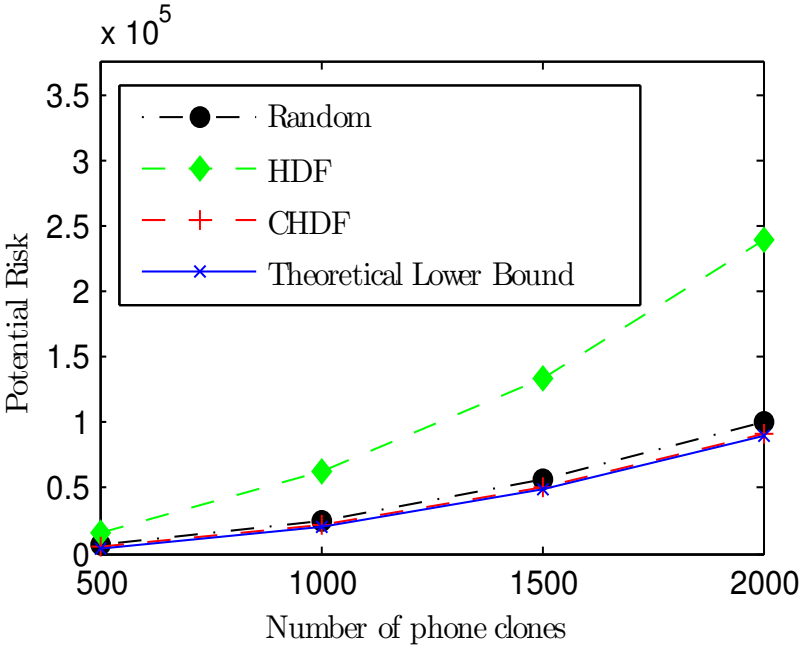

Figure 5.7: Potential risk of phone clones with random communication graph; no. of hosts=10

5.5.2 Performance of Phone Clone Allocation

We compare our phone clone allocation algorithms with a naïve random method. In the random method, each phone clone randomly selects a host that has free space to join.

Fig. 5.6 shows the potential risk with a random communication graph generated with the link probability of 50%. The theoretical lower bounds are obtained using Theorem 1 with the probability no less than 98%. When the number of phone clones is small (say smaller than 50), all methods have similar performance. This is because the system has enough capacity to avoid risky allocation. When the number of phone clones becomes large, the *MCF* algorithm works the best and its performance is very close to the theoretical lower bound. This implies that (1) Theorem 1 provides very tight probabilistic lower bound, and (2) our *MCF* and *QP with rounding* algorithms are nearly optimal.

We tested the algorithms with higher number of phone clones. Figure 5.7 shows the potential risk of random allocation, *HDF*, and *CHDF* algorithms. The performance of *CHDF* is better than *HDF* and Random allocation because *CHDF* algorithm spreads the phone clones as evenly as possible to avoid potential risks. From this figure, we can observe that with high number of phone clones, performance of *CHDF* matches the theoretical lower bound with confidence interval $1 - 10^{-12}$.

5.5.3 Performance of Phone Clone Migration with Random Graph

We test if our phone clone migration scheme can maintain a low risk level with a small number of phone clone migrations. In the first test, we use the random communication graph generated with the link probability of 50%. The number of hosts is set to 10; each host has a maximum capacity of 50 phone clones. We change the number of phone clones from 20 to 200 and change the risk threshold in each host from 2 to 30. We measure the normalized number of phone clone migrations, which is calculated as the total number of migrations divided by the number of phone clones.

From the results shown in Fig. 5.8, we can see that although a smaller δ value or a larger number of phone clones results in more phone clone migrations, the normalized number of phone clone migrations is low, e.g., in the worst case (200 phone clones and $\delta = 2$), the average number of phone clone migrations is small in the long term. We also test "heavier" cases by reducing the number of hosts and increasing the number

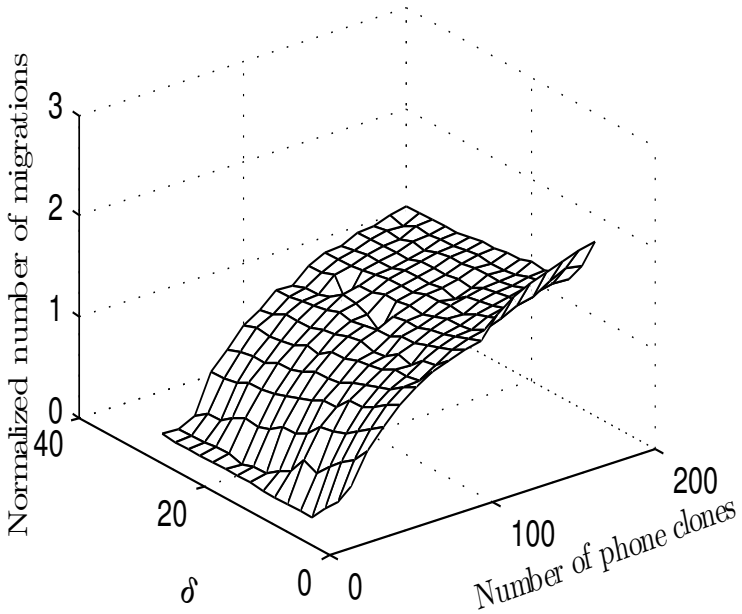

Figure 5.8: Normalized number of phone clone migration

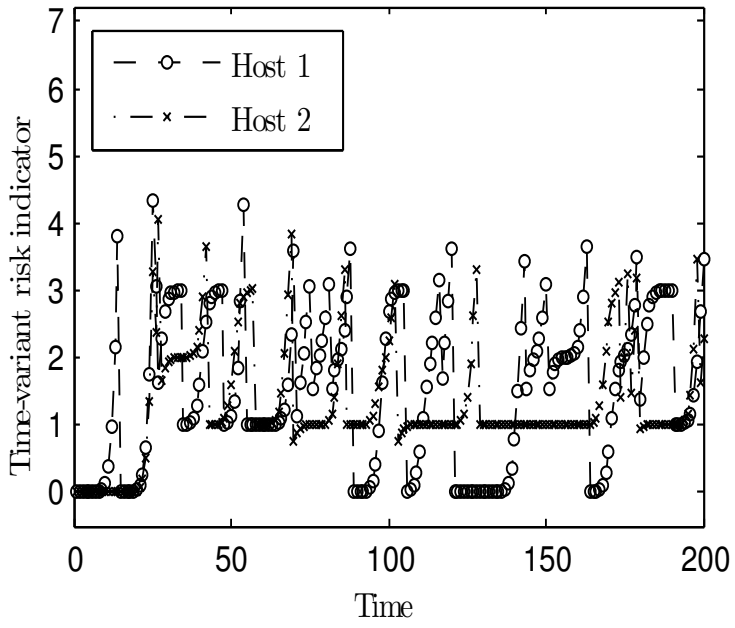

Figure 5.9: Time-variant risk indicator on two randomly selected hosts with the random communication graph

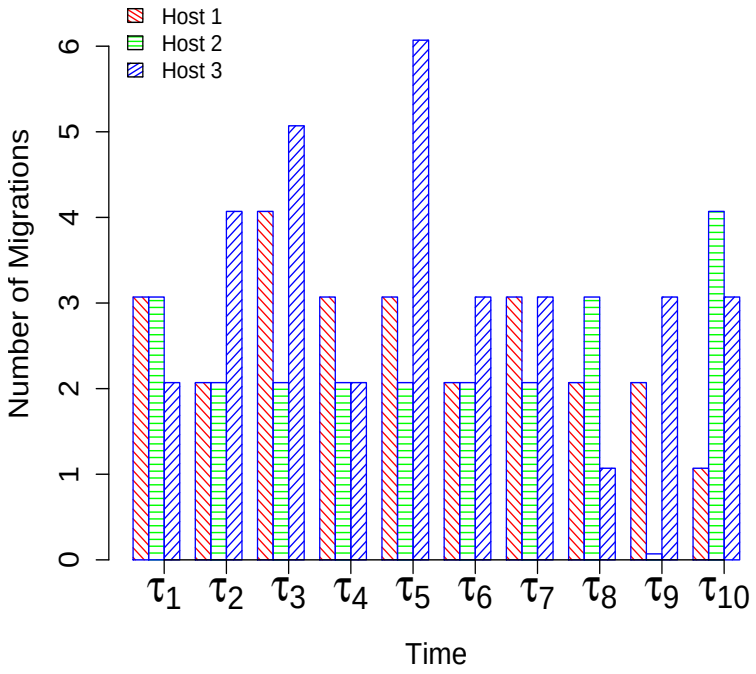

Figure 5.10: Number of migrations on three randomly selected hosts (each $\tau_i = 20$ time units) with the random communication graph

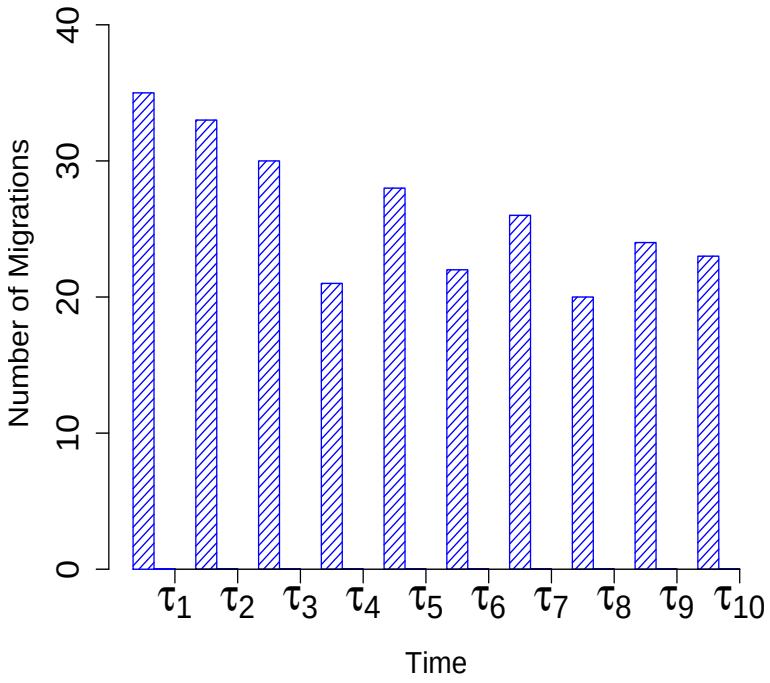

Figure 5.11: Total number of migrations in the system (each $\tau_i = 20$ time units) with the random communication graph

of phone clones, and we observe the similar phenomenon.

To further illustrate the dynamic behavior of phone clone migration, we set the number of phone clones to 100 and randomly select 2 hosts and show the dynamic change of time-variant risk indicator in the two hosts when the risk threshold δ is set to 3. It is clear from Fig. 5.9 that when the time-variant risk indicator in a host increases above the threshold, the migration algorithm kicks in to trigger phone clone migration so that the risk is reduced below the threshold value. Fig. 5.10 shows the number of migrations on three randomly selected hosts. Since phone clone migration is triggered only when the time-variant risk indicator in a host becomes higher than the threshold, we measure the number of migrations every 20 unit time. We have observed that our phone clone migration scheme can maintain a low risk level in all hosts. In addition, we have observed that no host or phone clone behaves significantly different from others, which suggests that our scheme does not penalize a particular host or a particular phone clone with more migrations.

Fig. 5.11 shows the total number of migrations in all hosts. We can easily see that the system is pretty stable in the sense that no sudden large number of phone clone migrations is required. This type of stability is important for ease of system management.

5.5.4 Phone Clone Migration with the BA Graph Model

We also test the scenarios where the communication graph of phone clones follows the BA model, by setting the model parameters $n_0 = 5$ and $n_e = 4$. Other system settings are the same as in the previous section. Fig. 5.12 shows the dynamic changes of time-variant risk indicator on two randomly selected hosts; Fig. 5.13 shows the number of migrations of three randomly selected hosts; Fig. 5.14 shows the total number of phone clone migrations in all hosts. From these figures, we observe the phenomena similar to those obtained with the random graph model.

5.5.5 Phone Clone Migration with the Reality Mining Dataset

We test the migration algorithm with the data collected from 100 mobile phones over the course of 9 months in the Reality Mining project undertaken at the MIT Media Laboratory [20, 21]. In this project, one hundred smart phones were deployed over a time period of an academic year. The Reality Mining dataset contains different information including call logs, Bluetooth proximity data, cell tower IDs of all par-

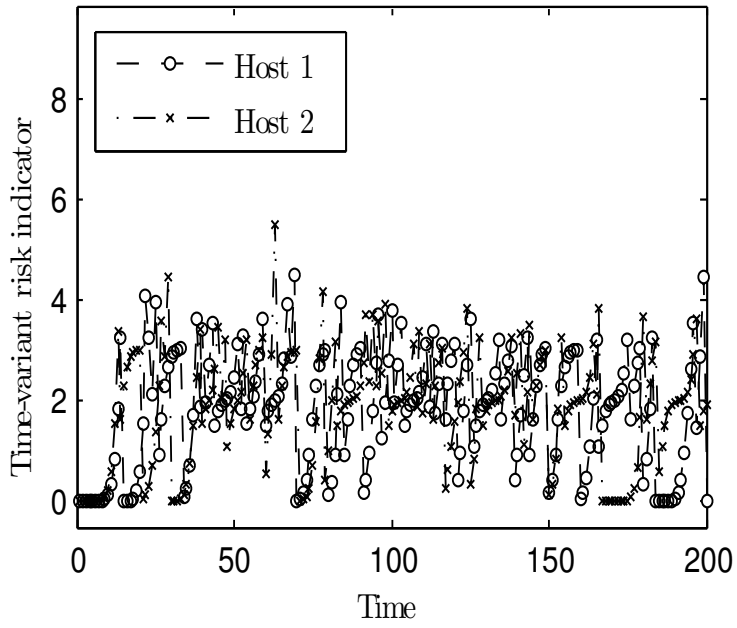

Figure 5.12: Time-variant risk indicator on two randomly selected hosts with the BA communication graph

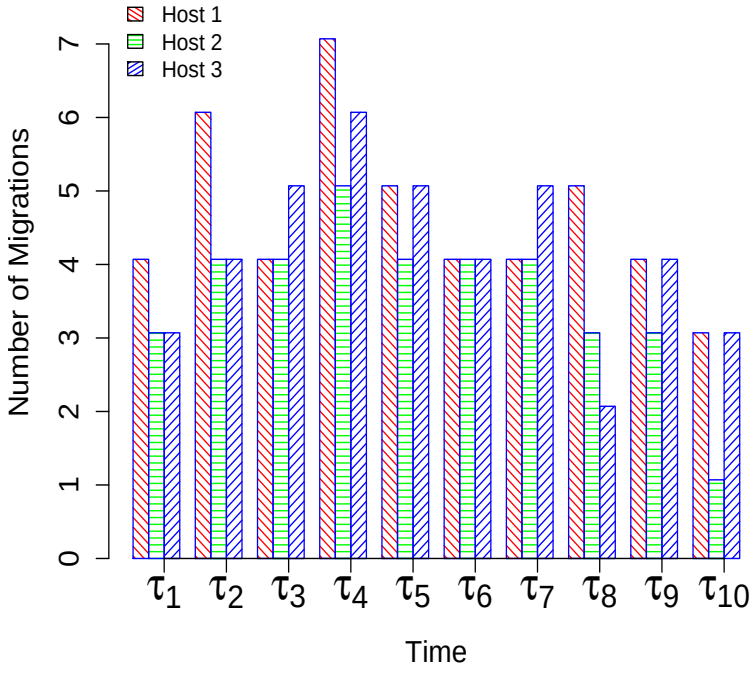

Figure 5.13: Number of migrations on three randomly selected hosts (each $\tau_i = 20$ time units) with the BA communication graph

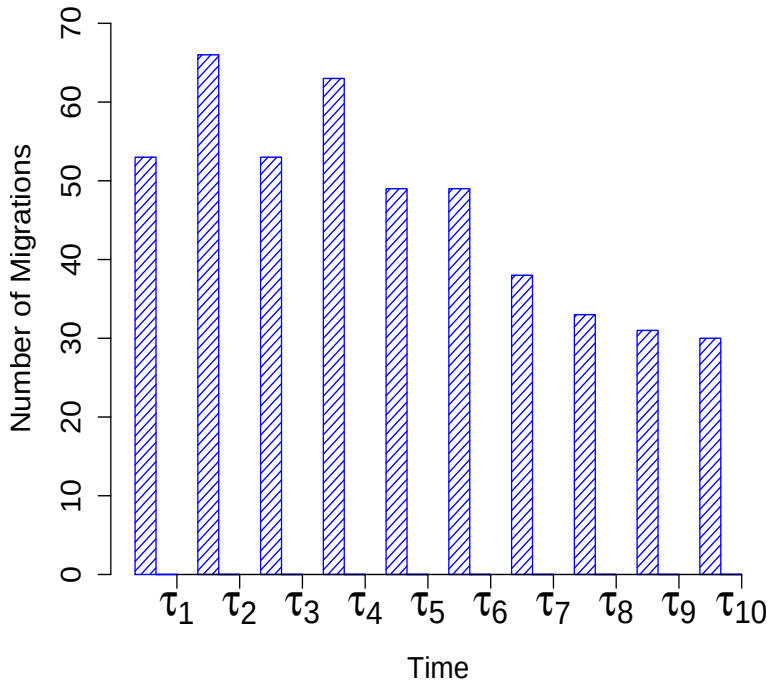

Figure 5.14: Total number of migrations in the system (each $\tau_i = 20$ time units) with the BA communication graph

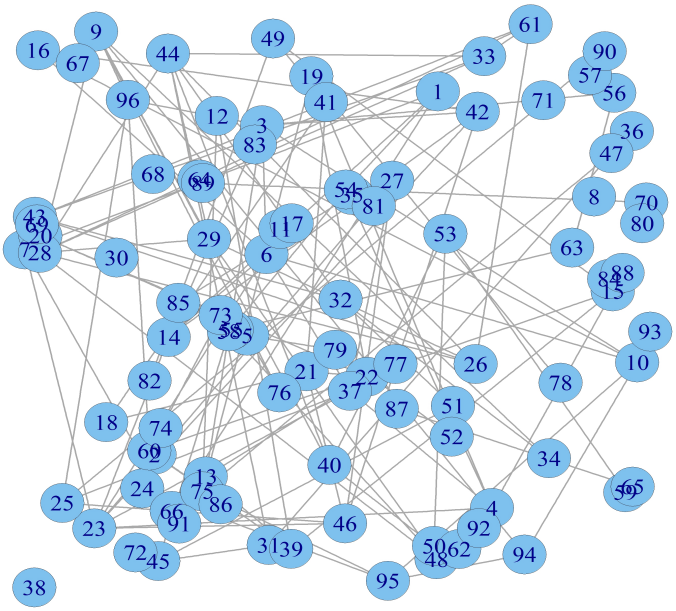

Figure 5.15: The estimated Reality Mining social graph [21]

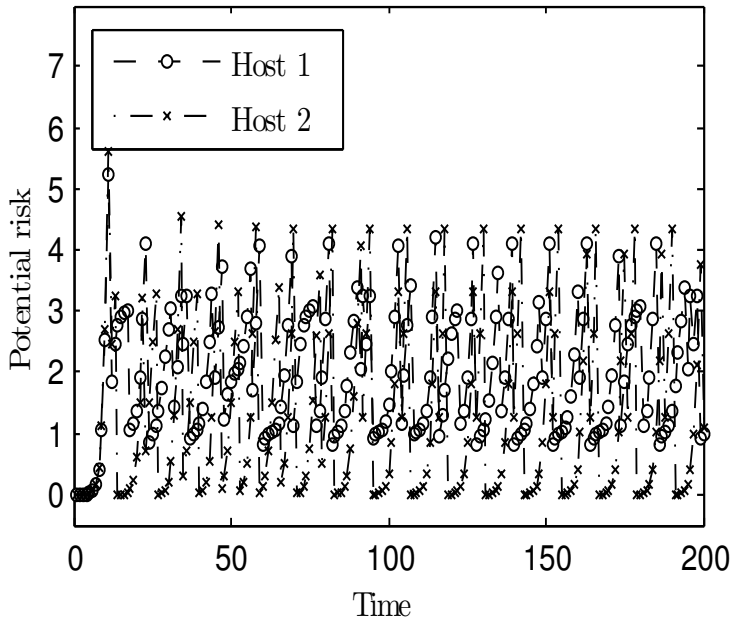

Figure 5.16: Time-variant risk indicator on two randomly selected hosts with the Reality Mining dataset

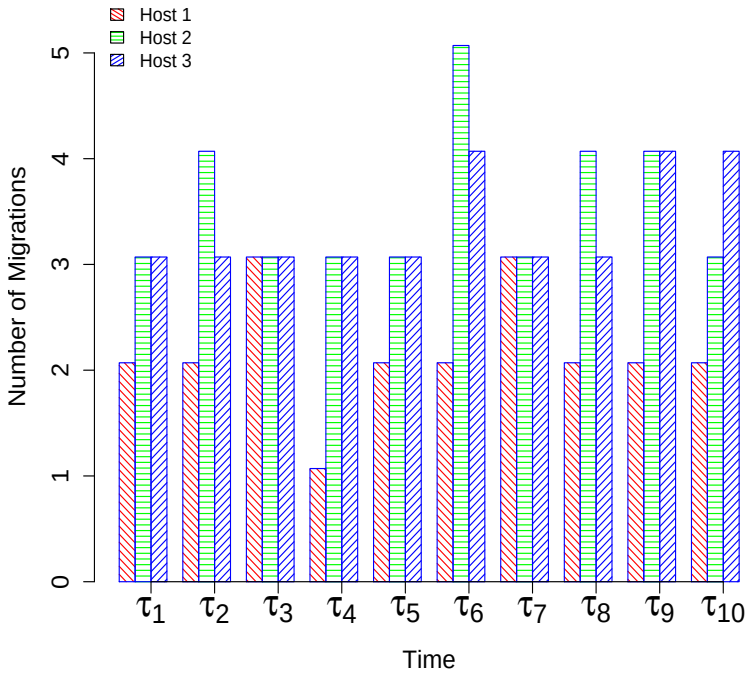

Figure 5.17: Number of migrations on three randomly selected hosts (each $\tau_i = 20$ time units) with the Reality Mining dataset

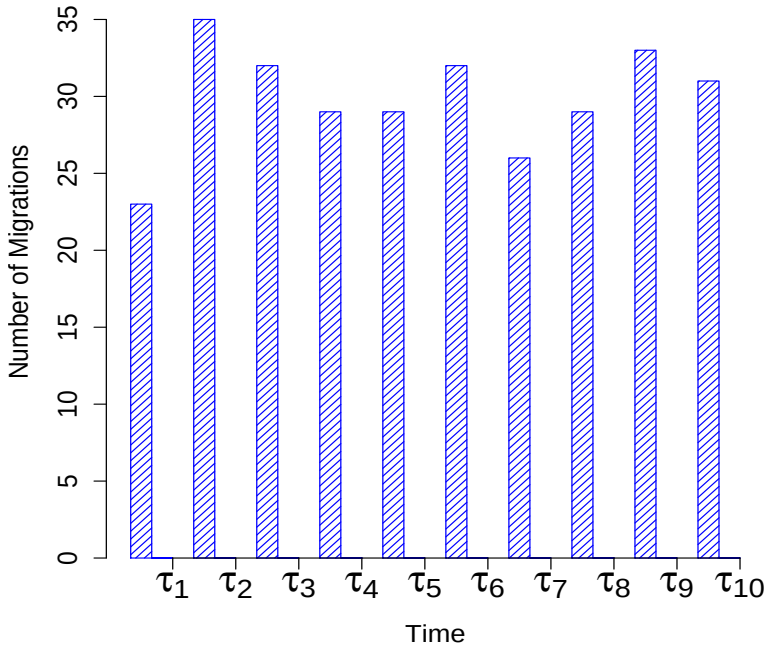

Figure 5.18: Total number of migrations in the system (each $\tau_i = 20$ time units) with the Reality Mining dataset

ticipants. During a 9 month period, a total of approximately 450,000 hours of information about users' location, communication and device usage behavior are recorded [20]. The Reality Mining dataset is available at http://reality.media.mit.edu/.

The participants of Reality Mining project consisted of students and staff at MIT university, and the data is collected between September 2004 and June 2005. This dataset was collected from 100 participants that were given Nokia 6600 smart phones. From these 100 participants, 75 were students or faculty in the MIT Media Laboratory (90% graduate students, 10% staff), and the remaining participants were students at the MIT Sloan business school [20].

In the Reality Mining project, Nokia 6600 phones were programmed to run the Context Log application and record the information during the phone usage. These smart phones continuously logged different information such as location (from cell tower ids) and other proximity information (recorded from Bluetooth devices). The running application on the phones also recorded the phone's activity (e.g. voice calls and text messages) and other information such as active applications. Two methods were used to collect data from phones. In the first method, participants were provided with data plans (GPRS) on their smart phones, and their phones connected to the data server and uploaded the logged data to the server. For the remaining participants, data were stored on each phone's internal memory card [20].

Fig. 5.15 shows the estimated Reality Mining social graph. We test the migration algorithm using the estimated communication graph. In this experiment, the number of hosts is set to 10. Fig. 5.16 shows the dynamic changes of time-variant risk indicator on two randomly selected hosts; Fig. 5.17 shows the number of migrations of three randomly selected hosts; Fig. 5.18 shows the total number of phone clone migrations in all hosts. From these figures, we conclude that risk indicator is controlled in a given threshold with a stable number of phone clone migrations over time.

5.5.6 Phone Clone Migration with the Nodobo Dataset

We also test the migration algorithm with the data collected from a group of 27 students at a high school in Glasgow [12, 14]. The Nodobo dataset is available online at http://nodobo.com/. In this project, each of the participants was given a Google Nexus smart phone, and each smart phone collected communications and social information including logs of phone calls, SMS messages, Bluetooth proximity detection, WiFi access point, and cell tower ID. Data was stored in phones' internal memory

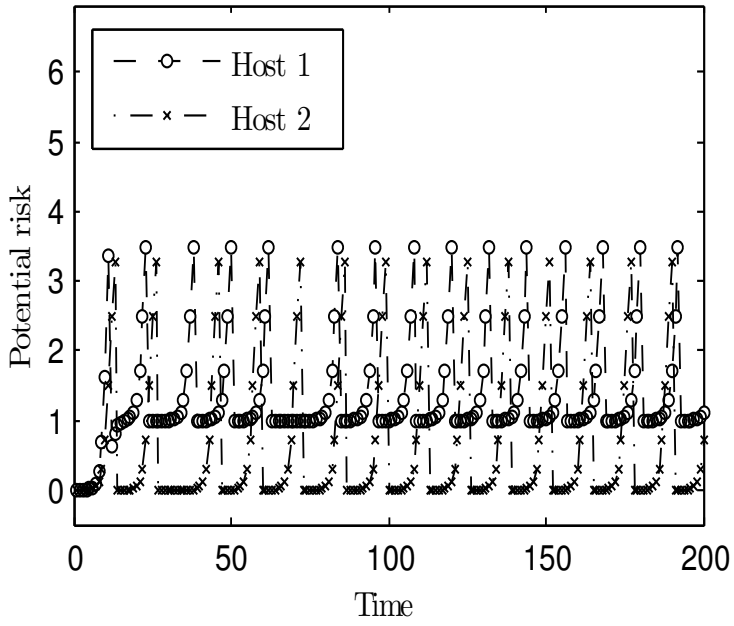

Figure 5.19: Time-variant risk indicator on two randomly selected hosts with the Nodobo dataset

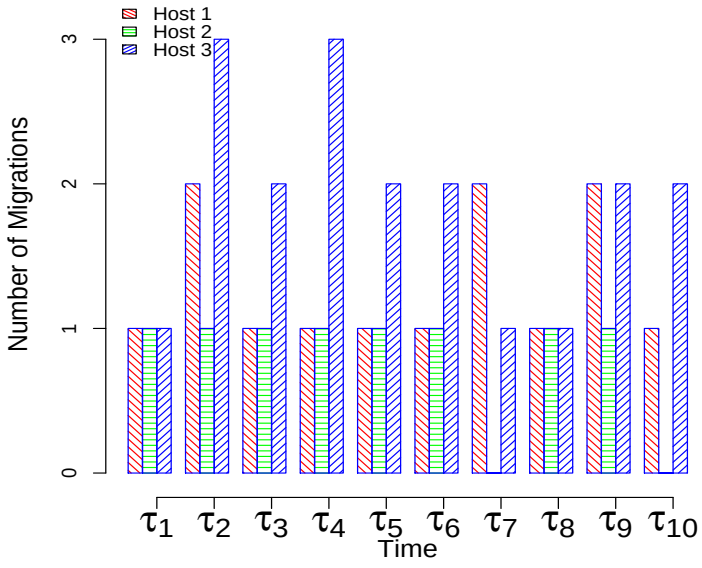

Figure 5.20: Number of migrations on three randomly selected hosts (each $\tau_i = 20$ time units) with the Nodobo dataset

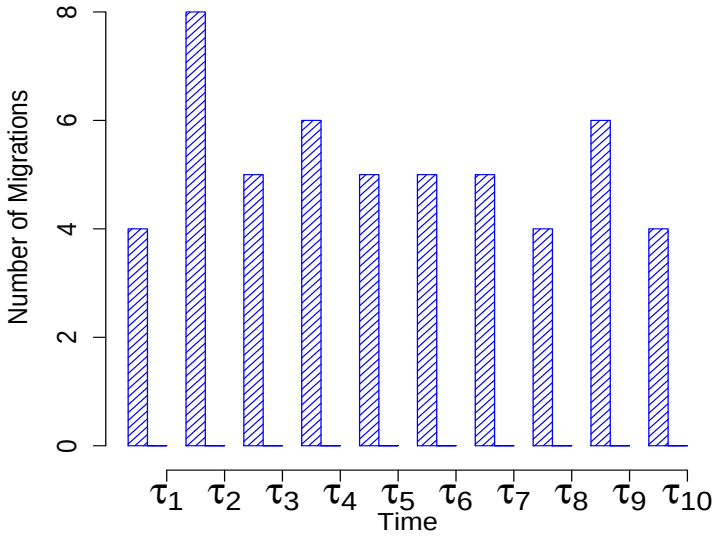

Figure 5.21: Total number of migrations in the system (each $\tau_i = 20$ time units) with the Nodobo dataset

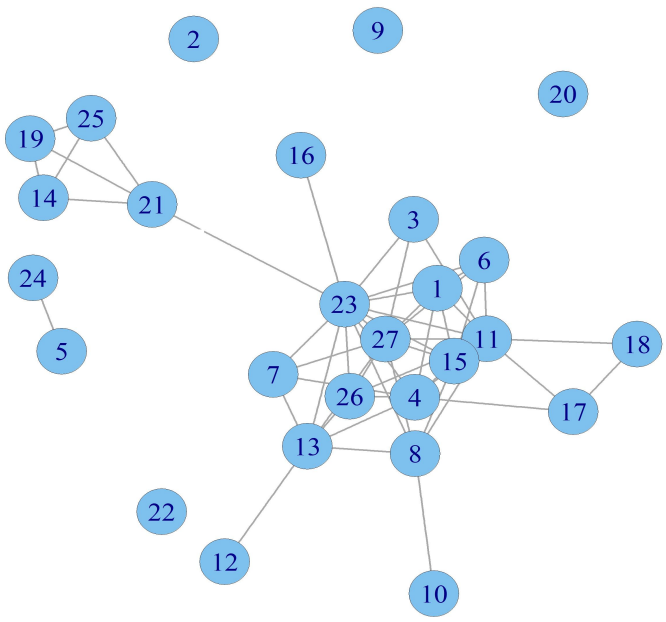

Figure 5.22: The estimated Nodobo social graph [12]

and transferred to a central server. Different types of data were recorded including the duration of the calls, associated phone numbers, or length of the message [14].

On each of the smart phones, a modified Android platform was installed which allowed the researchers to record information about phone usage without affecting users' experience. During the time period from September 2010 to the end of January 2011, the Nodobo study recorded 13035 call records, 83542 SMS records, and 5.2 million proximity records. It was observed that the majority of calls occurred during the school day, from 8am to 3pm [14]. Most of phone activities occurred during this period, but the activity continued after school hours and throughout the evening. The number of phone calls decreased around 10 pm, but SMS messages continued to be sent until 2 am.

Fig. 5.22 shows the estimated Nodobo social graph. In this experiment, the number of hosts is set to 5. Fig. 5.19 shows the dynamic changes of time-variant risk indicator on two randomly selected hosts; Fig. 5.20 shows the number of migrations of three randomly selected hosts; Fig. 5.21 shows the total number of phone clone migrations in all hosts. From these figures, we can also conclude that risk indicator is controlled in a given threshold with a stable number of phone clone migrations over time.

5.6 Summary

Cloud computing opens a new era for mobile communications industry. Mobile users are presented with the great opportunity of utilizing the abundant computing resources in cloud to execute powerful applications. One way to harness this opportunity is to build phone clones in cloud and allow users to offload computation intensive tasks to the phone clones. Associated with the enhanced services to users, however, is the potential risk of information leakage over the cloud. In this chapter, we presented a system model for potential risks in mobile cloud. The system model consists of communication model, potential risk, and its calculation. Based on the communication history of phone clones, we can build a communication graph representing phone clones and communication between phone clones. We consider that when two phone clones that do not have a direct communication link between them are allocated to the same physical host, the allocation scheme may introduce a potential breach.

In this chapter, we systematically studied this problem and developed SWAP: a security aware provisioning and migration scheme for phone clones. First, we formulate the problem of minimizing the potential risks in phone clone allocation. We reduce the model complexity with cliques. For solving the problem, we present different algorithms including QP with Rounding and different heuristic algorithms such as MCF, HDF, and CHDF. Second, we study dynamic migration of phone clones in mobile cloud. It has been shown that covert channels need time to build. Therefore, we try to migrate phone clones when the risks become higher than a given threshold. We propose an event-driven heuristics method for phone clone migration which consists of three steps: 1) Identifying hosts with the risk value higher than the threshold 2) Selecting phone clones from the risky hosts for migration 3) Reallocating the risky phone clones.

We solve the core technical challenges in SWAP and develop algorithms that have been demonstrated to work effectively for phone clone allocation and migration. We evaluate our solution using Reality Mining and Nodobo dataset. Experimental results show that our algorithms maintain low risk with small number of phone clone migrations. This chapter addressed a new set of problems that call for further research, including resource multiplexing and task scheduling algorithms for phone clones. Improving performance of phone clones poses another challenge and demands innovative solutions that optimize both security and system performance. These will be part of our future work.

Chapter 6

Conclusions and Future Work

Mobile Telecom Cloud (MTC) can provide users with high quality of cloud services as telecom companies own the network infrastructure. In the first part of this thesis, we study the problem of minimizing the system costs and maximizing users' satisfaction in MTC and present an analytical framework, which consists of a network model, a user model (access profile and SLAs), and the cost model. To solve the optimization problem, we present two algorithms, LP based branch-and-bound and LP with rounding. By clustering the users with similar data access pattern, we can manage a system of up to millions of mobile users in MTC. We further investigate how to make trade offs between users' satisfaction and cloud provider's operating costs. Simulation results show that our method can scale well to large networks.

Next, we discuss the MTC brokerage. MTC can offer their own telecom cloud services as well as third-party cloud services and provide users with a variety of different cloud services with minimum cost and highest QoS. We study the problem of minimizing the costs and providing the users with their desired QoS in MTC and TPC where telecom companies play the role of cloud brokerage, called the MTC brokerage. To solve the optimization problem, we present two algorithms, LP with rounding and MinCost greedy. Since there are dynamic changes in users' cloud requests, we study the problem of adaptive resource planning in MTC brokerage. Using both simulated data and real Google traces, we show that our proposed resource allocation algorithms achieve considerable cost savings for both the users and the MTC brokerage.

Finally, we discuss security in mobile cloud. Cloud computing opens a new era for mobile communications industry. Mobile users are presented with the great opportunity of utilizing the abundant computing resources in cloud to execute powerful applications. One way to harness this opportunity is to build phone clones in cloud

and allow users to offload computation intensive tasks to the phone clones. Associated with the enhanced services to users, however, is the potential risk of information leakage over the cloud. In this thesis, we systematically study this problem and develop SWAP: a security aware provisioning and migration scheme for phone clones. We solve the core technical challenges in SWAP and develop algorithms that have been demonstrated to work effectively for phone clone allocation and migration. We evaluate our solution using Reality Mining and Nodobo dataset. Experimental results show that our algorithms are effective in maintaining low risk and minimizing the number of phone clone migrations.

6.1 Future Work

6.1.1 Migration of Cloud Services and Resource Multiplexing in MTC Brokerage

In this thesis, we considered dynamic changes in users' cloud requests, and we studied the problem of adaptive resource planning in MTC brokerage. However, we didn't consider details of migration of users' cloud services from one cloud plan to another cloud plan. This thesis opens a new set of problems that call for further research, including resource multiplexing and migration of users' cloud services in MTC brokerage.

6.1.2 Location-Aware On-Demand QoS Provisioning in Mobile Telecom Cloud (LOQ)

While mobile telecom cloud enhances mobile users' QoS experiment, different parameters may affect the QoS that is provided by cloud services. Such parameters including network congestion, out-of-signal connection problems, and non-heterogeneity of mobile users may reduce the users' QoS significantly. Therefore, proper management of mobile telecom cloud network and users' QoS experience increases the users satisfaction and will result in cloud services with higher bandwidth and lower delay.

Different areas can be investigated to increase mobile users' QoS satisfaction in mobile telecom cloud:

1. Monitoring user' QoS experience: Telecom companies are the last-mile internet providers and have direct knowledge of mobile users. Therefore, by measuring

the users' QoS experience, they can locate and migrate cloud services in mobile telecom cloud to enhance users' satisfaction.

2. Heterogeneity of the network: Different mobile users may connect to mobile network with different devices and different wireless technologies. Therefore, they may have different QoS experience resulting in users' dissatisfaction. Since mobile telecom cloud have direct knowledge of users connectivity, they can better provide users with their required QoS.

3. Network management: Different mobile users may connect to the network during different times of a day or different locations. Therefore, the network may be congested during certain period of time or certain areas of the network. Thus, mobile telecom cloud can manage the network and users' cloud service location to ensure users' QoS satisfaction.

In our future work, we can design a system for increasing mobile users' QoS satisfaction in mobile telecom cloud. The inputs to the system are users' QoS experience information, users' information (e.g. device type and connectivity), and network information (e.g. congestion, links states). Based on these inputs, we can locate cloud services in mobile telecom cloud and outputs the optimized location of cloud services.

The LOQ on-demand QoS provisioning system consist of following components:

1. QoS measurement metric: A metric for measuring users' QoS experience in accessing telecom cloud can be defined. This metric will be used for analysis of telecom cloud and suggestions for future improvements.

2. Modeling QoS as a function: For precise analysis of the telecom cloud, we can model the users' experienced QoS as a function. This function measures QoS based on the defined QoS measurement metric.

Bibliography

[1] Amazon elastic compute cloud (EC2), 2015. http://aws.amazon.com/ec2.

[2] Appirio, 2015. http://appirio.com.

[3] Aws marketplace, 2015. https://aws.amazon.com/marketplace.

[4] Cloudmore, 2015. http://web.cloudmore.com.

[5] Linux kvm, 2015. http://www.linux-kvm.org/.

[6] Vmware, 2015. http://www.vmware.com/.

[7] Xen, 2015. http://www.xenproject.org/.

[8] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47–97, 2002.

[9] Khalid Abdulaziz Alnowibet. *Nonstationary Erlang Loss Queues and Networks*. PhD thesis, 2004.

[10] Rajesh Balan, Jason Flinn, Mahadev Satyanarayanan, Shafeeq Sinnamohideen, and Hen-I Yang. The case for cyber foraging. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, pages 87–92, 2002.

[11] Rajesh Krishna Balan, Mahadev Satyanarayanan, So Young Park, and Tadashi Okoshi. Tactics-based remote execution for mobile computing. In *Proceedings of the 3rd international conference on Mobile systems, applications and services*, pages 273–286, 2003.

[12] Jamie Banford and James Irvine. Estimating social graphs in an education environment: Using mobile communication devices. *IEEE Vehicular Technology Magazine*, 7(1):31–37, 2012.

[13] Marco V Barbera, Sokol Kosta, Alessandro Mei, and Julinda Stefa. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. In *Proc. of IEEE INFOCOM*, pages 1285–1293, 2013.

[14] Stephen Bell, Alisdair McDiarmid, and James Irvine. Nodobo: Mobile phone as a software sensor for social network research. In *73rd IEEE Vehicular Technology Conference (VTC Spring)*, pages 1–5, 2011.

[15] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, pages 301–314. ACM, 2011.

[16] Byung-Gon Chun and Petros Maniatis. Augmented smartphone applications through clone cloud execution. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, pages 8–11, 2009.

[17] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62. ACM, 2010.

[18] C. Deng, L. Qian, M. Xu, Y. Du, Z. Luo, and S. Sun. Federated cloud-based big data platform in telecommunications. In *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*, pages 44–48, 2012.

[19] Niu Di. Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications. *IEEE Wireless Communications*, page 3, 2013.

[20] Nathan Eagle and Alex Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268, 2006.

[21] Nathan Eagle, Alex Sandy Pentland, and David Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, 2009.

[22] Ericsson. The telecom cloud opportunity. *Ericsson Discussion Paper*, March 2012.

[23] John Evans and Clarence Filsfils. *Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice*. Morgan Kaufmann, 2007.

[24] Jason Flinn, Dushyanth Narayanan, and Mahadev Satyanarayanan. Self-tuned remote execution for pervasive computing. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, pages 61–66, 2001.

[25] Jason Flinn, SoYoung Park, and Mahadev Satyanarayanan. Balancing performance, energy, and quality in pervasive computing. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 217–226, 2002.

[26] Frank Fowley, Claus Pahl, and Li Zhang. A comparison framework and review of service brokerage solutions for cloud architectures. In *ICSOC 2013 Workshops on Service-Oriented Computing*, pages 137–149, 2014.

[27] B. Gendron and T. G. Crainic. Parallel branch-and-branch algorithms: Survey and synthesis. *Operations Research*, 42(6):1042–1066, 1994.

[28] V Gligor. *A Guide To Understanding Covert Channel Analysis of Trusted Systems*. National Computer Security Center, U.S., 1993.

[29] Mark S Gordon, David Ke Hong, Peter M Chen, Jason Flinn, Scott Mahlke, and Zhuoqing Morley Mao. Accelerating mobile applications through flip-flop replication. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 137–150, 2015.

[30] Ines Houidi, Marouen Mechtri, Wajdi Louati, and Djamal Zeghlache. Cloud service delivery across multiple cloud platforms. In *2011 IEEE International Conference on Services Computing (SCC)*, pages 741–742, 2011.

[31] Trent Jaeger, Reiner Sailer, and Yogesh Sreenivasan. Managing the risk of covert information flows in virtual machine systems. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, SACMAT '07, pages 81–90, 2007.

[32] Pritesh Jain, Dheeraj Rane, and Shyam Patidar. A novel cloud bursting brokerage and aggregation (CBBA) algorithm for multi cloud environment. In *2012 Second International Conference on Advanced Computing & Communication Technologies (ACCT)*, pages 383–387, 2012.

[33] Sachin Kadloor, Negar Kiyavash, and Parv Venkitasubramaniam. Mitigating timing based information leakage in shared schedulers. In *Proc. of IEEE INFOCOM*, pages 1044–1052, 2012.

[34] N. Katica and A. Tahirovic. Opportunities for telecom operators in cloud computing business. In *Proceedings of the 35th International Convention on Information and Communication Technology, Electronics and Microelectronics*, pages 495–500, 2012.

[35] Abdul Nasir Khan, ML Mat Kiah, Sajjad A Madani, Mazhar Ali, et al. Enhanced dynamic credential generation scheme for protection of user identity in mobile-cloud computing. *The Journal of Supercomputing*, pages 1–20, 2013.

[36] Abdul Nasir Khan, M. L. Mat Kiah, Samee U. Khan, and Sajjad A. Madani. Towards secure mobile cloud computing: A survey. *Future Gener. Comput. Syst.*, 29(5):1278–1299, July 2013.

[37] Taesoo Kim, Marcus Peinado, and Gloria Mainar-Ruiz. Stealthmem: System-level protection against cache-based side channel attacks in the cloud. In *USENIX Security Symposium*, pages 189–204, 2012.

[38] P. Koehler, J. Kraemer, and A. Anandasivam. Cloud computing: New business opportunities for telecommunications companies? In *21st European Regional ITS Conference*, pages 1–4, 2010.

[39] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, and Xinwen Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *Proc. of IEEE INFOCOM*, pages 945–953, 2012.

[40] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithm and Probabilistic Analysis*. Cambridge University Press, 2005.

[41] Andrea Qualizza, Pietro Belotti, and Francois MargotKosta. Linear programming relaxations of quadratically constrained quadratic programs. *The IMA Volumes in Mathematics and its Applications*, 154:407–426, 2012.

[42] Charles Reiss, John Wilkes, and Joseph L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, November 2011.

[43] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 199–212, 2009.

[44] S. Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4):262–279, December 1974.

[45] Brendan Saltaformaggio, Dongyan Xu, and Xiangyu Zhang. Busmonitor: A hypervisor-based solution for memory bus covert channels. In *Proceedings of the 6th European Workshop on Systems Security*, EuroSec'13, pages 1–6, 2013.

[46] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, 2009.

[47] Smitha Sundareswaran, Anna Squicciarini, and Dan Lin. A brokerage-based approach for cloud service selection. In *5th IEEE International Conference on Cloud Computing (CLOUD)*, pages 558–565, 2012.

[48] Piotr Konrad Tysowski. *Highly Scalable and Secure Mobile Applications in Cloud Computing Systems*. PhD thesis, University of Waterloo, 2013.

[49] Seyed Yahya Vaezpour, Kui Wu, and Gholamali Shoja. Location matters: optimal data placement in mobile telecom cloud. In *Proceedings of the Second IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 176–185, 2014.

[50] Seyed Yahya Vaezpour, Kui Wu, and Gholamali Shoja. Mobile telecom cloud brokerage with orchestrated multi-tier resource pooling. In *4th IEEE International Conference on Cloud Networking (CloudNet'15)*, 2015.

[51] Seyed Yahya Vaezpour, Rui Zhang, Kui Wu, Jianping Wang, and Gholamali Shoja. Swap: Security aware provisioning and migration of phone clones over mobile clouds. In *IFIP Networking 2014 Conference*, pages 1–9, 2014.

[52] Bhanu C. Vattikonda, Sambit Das, and Hovav Shacham. Eliminating fine grained timers in xen. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 41–46, 2011.

[53] D. Verchere. Cloud computing over telecom network. In *Optical Fiber Communication Conference and Exposition (OFC/NFOEC)*, pages 1–3, 2011.

[54] Bimlesh Wadhwa, Aditi Jaitly, and Bharti Suri. Cloud service brokers: an emerging trend in cloud adoption and migration. In *Software Engineering Conference (APSEC, 2013 20th Asia-Pacific*, pages 140–145, 2013.

[55] Wei Wang, Di Niu, Baochun Li, and Ben Liang. Dynamic cloud resource reservation via cloud brokerage. In *2013 IEEE 33rd International Conference on Distributed Computing Systems (ICDCS)*, pages 400–409, 2013.

[56] Wei Wang, Di Niu, Ben Liang, and Baochun Li. Dynamic cloud instance acquisition via IaaS cloud brokerage. *IEEE Transactions on Parallel and Distributed Systems*, pages 1580–1593, 2015.

[57] Zhenghong Wang. *Information Leakage Due to Cache and Processor Architectures*. PhD thesis, Princeton University, 2012.

[58] Zhenyu Wu, Zhang Xu, and Haining Wang. Whispers in the hyper-space: high-speed covert channel attacks in the cloud. In *Proceedings of the 21st USENIX conference on Security symposium*, Security'12, pages 9–9, 2012.

[59] Yuanzhong Xu, Weidong Cui, and Marcus Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, 2015.

[60] Yunjing Xu, Michael Bailey, Farnam Jahanian, Kaustubh Joshi, Matti Hiltunen, and Richard Schlichting. An exploration of l2 cache covert channels in virtualized environments. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 29–40, 2011.

[61] Yunjing Xu, Michael Bailey, Farnam Jahanian, Kaustubh Joshi, Matti Hiltunen, and Richard Schlichting. An exploration of l2 cache covert channels in virtualized environments. In *Proceedings of the 3rd ACM workshop on Cloud Computing Security Workshop*, pages 29–40, 2011.

[62] Rui Zhang, Kui Wu, and Jianping Wang. Online resource scheduling under concave pricing for cloud computing. In *IEEE 22nd International Symposium of Quality of Service (IWQoS)*, pages 51–60, 2014.

[63] Yinqian Zhang, Ari Juels, Alina Oprea, and Michael K. Reiter. Homealone: Co-residency detection in the cloud via side-channel analysis. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, pages 313–328, 2011.

[64] Yinqian Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Cross-vm side channels and their use to extract private keys. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pages 305–316, 2012.

[65] Yulong Zhang, Min Li, Kun Bai, Meng Yu, and Wanyu Zang. Incentive compatible moving target defense against vm-colocation attacks in clouds. In *SEC*, volume 376 of *IFIP Advances in Information and Communication Technology*, pages 388–399, 2012.

[66] D. Zuckerman. Np-complete problems have a version that's hard to approximate. In *Proceedings of the 8th IEEE Annual Structure in Complexity Theory Conference*, May 1993.