Seamless Live Virtual Machine Migration for Cloudlet Users with Multipath TCP

By

Fikirte Abebe Teka

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science
In
Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

Abstract

Virtual machine (VM) based cloudlets are used in a cloud computing system to enhance the performance of resource-intensive applications. After a mobile device (MD) discovers a cloudlet in a vicinity, it takes a service initiation time (Ts) to setup a VM inside the cloudlet before data offloading from the MD to the VM starts. When more than one cloudlets are presented in a nearby geographical location, initiating a service with each cloudlets may be frustrating for the cloudlet users who change location frequently. In order to eliminate the delay caused by the Ts after the first cloudlet, this thesis proposes a seamless live VM migration between the neighbour cloudlet where the seamlessness is achieved by multipath TCP (MPTCP). In addition to MPTCP, the prior network configuration of the migrating VM with the destination network information helps to achieve a zero network downtime at the destination cloudlet after migration is completed.

Acknowledgements

I am extremely grateful to go through a research experience which would not be possible without my principal supervisor Dr. Chung-Horng Lung who has given me the opportunity with lots of inspirational ideas, encouragement and patience throughout the course of this thesis. I would also like to express my sincere thanks to my co-supervisor, Dr. Samuel A.Ajila, for his invaluable comments and ideas.

I take this opportunity to express a deep sense of gratitude to our department computer network administrator, Jerry Buburuz who provided resources and resolved all the technical errors I faced during the lab environment setup. My appreciation also goes to the author of [CaCh2013], Catalin Nicutar who has answered all my questions and added a knowledge about MPTCP.

Lastly, I thank St. Virgin Mary who has given me strength and heard all my prayers during the time of stress, my mother, Aynalem, who always believes in me even when I am falling apart, my beloved husband, Eyob, for his love, patience, and support, my daughter, sisters, brothers and friends for their love and encouragement. My love and respect to my families and friends are endless and immense.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

AP	Access Point
BW	Bandwidth
IETF	Internet Engineering Task Force
IP	Internet Protocol
Kbps	Kilo Bits per Second
KBps	Kilo Byte per Second
KVM	Kernel-based Virtual Machine
LAN	Local Area Network
Mbps	Mega Bits per Second
MBps	Mega Byte per Second
MCC	Mobile Cloud Computing
MD	Mobile Device
MPLS	Multi-Protocol Label Switching
MPTCP	Multipath TCP
NFS	Network File Sharing
QoE	Quality of Experience
RST	Reset
RTO	Retransmission Timeout
RTT	Round Trip Time
TCP	Transport Control Protocol
VM	Virtual Machine
VPLS	Virtual Private LAN Service
VPN	Virtual Private Network
WAN	Wide Area Network

Chapter 1 Introduction

The current mobile devices (MDs) have a better computational capacity than a typical server in the mid-1990s. "A high computational performance" is a phrase relative to the application executing on the device. The increase in the demand for mobile applications encourages the developers to bring the desktop level applications to the MDs. Even if the computational, storage and battery life time of MDs have improved, compared to the resource-intensive applications developed for MDs, the MDs remain to be resource-poor devices.

Cloud computing has become the common choice for offloading computation-intensive applications such as real-time face recognition, human language translation, and augmented reality, from local resource-poor devices to a highly efficient remote server or cluster. The local device sends an instruction and displays results obtained from the remote server.

Most of the applications executing on MDs are real-time and interactive applications. As it has been mentioned in [MaBa2009], offloading real-time application to a distant remote cloud through the Internet reduces the performance of response time due to Wide Area Network (WAN) RTT latency. RTT latency mainly affects two factors for the MD users. The first factor is the quality of experience (QoE). As RTT latency increases, the QoE degrades. The second factor is that high RTT latency drains the MDs battery faster [EdAr2010]. Energy efficiency is also an important performance parameter for MDs.

Maximizing the benefit of offloading applications for MDs can be achieved by minimizing the RTT latency between the MDs and the servers. To minimize RTT latency, researchers have proposed resource-rich nearby servers which are connected to or integrated with wireless access points (APs) to be used for offloading such as CLONECLOUD [BySu2011], MOCHA [SoMu2012], MAUI [EdAr2010] and virtual machine (VM) based cloudlet [MaBa2009]. In this way, the need for improving QoE and energy efficiency can be accomplished by low RTT latency, one-hop, high bandwidth (BW) wireless access to the servers.

There are basically two types of offloading mechanism. One is by partitioning applications and sending only resource-intensive instructions to servers and executing the rest of the instructions on the MD itself. CLONECLOUD [BySu2011], MOCHA [SoMu2012] and MAUI [EdAr2010] used this approach. The second offloading mechanism is to send all the instructions to the server. This mechanism needs to create a VM instance on the server for each MD to serve as a shared infrastructure and to provide a strong isolation between computations from different MDs. VM based cloudlet [MaBa2009] uses the second approach where the MD mostly acts as a sensing, user interaction, and data storage device.

The focus of this research is on the VM based cloudlets.

1.1 Motivation

The motivation of this research stems from the fact that MD users are not stationary, although they are getting benefit from the nearby static cloudlet servers.

A frequent movement is common to a MD user but changing network location causes the following two problems:

- The first problem is the IP address of the MD changes which will close the established transport control protocol (TCP) connection between the cloudlet and the MD.
- The second problem is the MD needs to wait for another service initiation time before offloading starts, if there is available possible nearby cloudlet in the new location. Service initiation time is the time from discovering the cloudlet to the time offloading starts.

The proposed solution for the first problem is to use MPTCP [MPTCP] instead of regular TCP which keeps the established connection open even if the original IP address is changed. The solution for the second problem is to migrate the VM instance from the first cloudlet to the next cloudlet which eliminates the extra service initiation time. This Section explains problems and technologies that are related to the two research areas.

Service initiation time with the VM based cloudlet servers is noticeable. After a MD discovered a nearby cloudlet, the cloudlet synthesizes a VM to the user. VM creation on demand is a time consuming processes that may frustrate the MD users. Dynamic VM synthesis is a method proposed in [MaBa2009] and further developed in [KiPa2013] to minimize the service initiation time (see Section 2.2.1). Although dynamic VM synthesis minimizes the time of VM customization up to 1 minute

[KiPa2013], the delay is still noticeable. Especially mobile users who rely on cloudlet services may find the extended delays for service initiation at new location to be unacceptable.

In a scenario where more than one cloudlets are present behind every wireless AP, such as in a city where libraries, coffee shops, shopping mall and stadiums with access to cloudlet servers are located geographically nearby, user should initiate service only once with the first cloudlet. Then the current state of the VM instance has to be migrated to the next cloudlet as the user changes location. This could be achieved with a seamless live VM migration which avoids service disruption and extra service initiation time.

Since each cloudlet may be located in different local area network (LAN), a live VM migration has to be performed over the WAN. In order to make the live VM migration seamless from the application layer point of view, all the network connections have to remain established after the VM is migrated. Changing location of both the MD and the VM changes the original IP address they acquired during the service initiation. Using a regular TCP as a transport layer protocol, seamless live VM migration and MD handover is impossible to achieve. The reason is that TCP identifies each connection uniquely with the source and destination IP addresses and the port numbers. If any one of the tuple changes, the connection would be closed abruptly.

In this research, the possibility of live VM migration with minimal downtime from cloudlet to cloudlet is studied to increase the QoE. The new IETF standard called

MPTCP [MPTCP] is used as a transport layer protocol to achieve a seamless MD handover and live VM migration.

1.2 Goals

MPTCP has proved its capability of a seamless MD handover from Wi-Fi to 3G/4G network [ChGr2012]. In this research, from Wi-Fi to Wi-Fi handover of MDs and a live VM migration over the WAN is studied with the use of MPTCP feature. The main goals of the research are summarized below:

- By creating a network collaboration between geographically nearby cloudlets, we aim to achieve a seamless live VM migration triggered by the MD location with the help of the MPTCP protocol.
- To investigate the performance of live VM migration for different values of network resources such as bandwidth and RTT latency between cloudlets.

1.3 Contributions

The contributions of the thesis are summarized clearly below:

- This research demonstrates the capability of MPTCP to support network migration during live server VM migration over a WAN. Without any code modification of MPTCP, a server VM is migrated live seamlessly without interrupting the application running on the VM. The original IP address used to establish the TCP subflow is changed after migration but the MPTCP connection with the MD remains established.

- This thesis proposes two interfaces to be used by the VM and to configure the destination network information before the VM is migrated. This approach has two advantages: seamless connection migration and zero network VM downtime after the migration is completed. Once the VM is completely migrated at the destination cloudlet, there is no time the VM wait for an IP address to be assigned since all the network information are pre-configured at the source cloudlet.

1.4 Thesis Organization

The remainder of the thesis is organized as follows.

Chapter 2 delivers the background knowledge of cloudlets, live VM migration and MPTCP. Related works which consider the MD users movement in a multiple cloudlets scenario and connection migration are reviewed in this chapter.

Chapter 3 presents the proposed approaches to overcome the limitation of live VM migration over the WAN with the help of MPTCP.

Chapter 4 provides the experiment results and the performance analysis of live VM migration. An algorithm is formulated for a better performance based on the lab results.

Chapter 5 concludes the thesis and points out the future works to be done.

Chapter 2 Background and Related Works

This chapter introduces the basic concept of three important topics for this research: Mobile Cloud Computing (MCC) with the involvement of cloudlets, live VM migration, and MPTCP. Section 2.1 describes the general concept of MCC and offloading mechanisms. Dynamic VM synthesis operation which creates a VM instance in cloudlets on demand is presented in Section 2.2. In order to understand how the live VM migration from cloudlet to cloudlet works, the general concept of live VM migration is presented in Section 2.3. One of the requirements of live VM migration is network migration. MPTCP protocol is proposed to support the network migration part. The operation of MPTCP is described in Section 2.4. Lastly, other researches which considers multiple cloudlets in a nearby location and connection migration are reviewed in Section 2.5.

2.1 Mobile Cloud Computing

MCC overcomes the resource limitation of wireless MDs by leveraging fixed infrastructure. Resources such as CPU, RAM, data storage, and battery energy are limited to MDs due to their portability and light weight feature. Compared to the static computing devices, the MDs remain to be resource-poor devices regardless of the improvement they have shown through time. The comparison between static servers and MDs in terms of processor speed is shown in table 2.1 which is adopted from [Ja2012].

Resource intensive applications such as speech recognition, natural language processing, computer vision and graphics, machine learning, augmented reality, planning, and decision-making are mostly used in resource-poor MDs. In order to enhance the performance of these resource-intensive applications, offloading from the MDs to resource-rich servers in a vicinity or to a distant server is a common solution.

Table 2.1: A comparison between static servers and MDs [Ja2012]

Year	Static Server		MDs	
	Processor	**Speed**	**Processor**	**Speed**
1997	Pentium II	266MHz	PalmPilot	16 MHz
2002	Itanium	1GHz	Blackberry 5810	133 MHz
2007	Core 2	9.6 GHz (4 cores)	Apple iPhone	412 MHz
2012	Xeon E3	14 GHz (2*4 cores)	Samsung Galaxy 3S	3.2 GHz(2 cores)

Researchers have demonstrated that offloading applications to remote server is not always the optimal solution for the MDs [EdAr2010] [BySu2011] [MaBa2009]. The RTT latency between the MD and the server is a parameter that governs the performance of the system in terms of energy consumption and QoE. As RTT latency increases the energy consumption of the MD increases and the QoE degrades. Using a nearby server instead of a remote server for offloading applications minimizes the network RTT latency between the cloudlet and the MD.

The following subsection describes an application offloading mechanisms to nearby or remote servers.

2.1.1 Methods of offloading

There are basically two types of offloading mechanisms. Executing resource-intensive application remotely by partitioning the application is the first method. This approach relies on the programmers to specify how to partition the program and to identify the instructions to be sent to the remote. The partitioning scheme has to adapt the network condition and the availability of resources on real-time basis. For example, if a smartphone can access a remote server with a good wireless network connection, resource-intensive tasks will be executed remotely. If the energy cost to send the tasks remotely is higher due to poor wireless network connection, the instructions will be executed locally on the smartphone.

The second offloading mechanism is by sending the whole application to a VM instance found in a remote server. No smart decision is required for this method. This approach reduces the burden on the application programmers because the applications do not need to be modified to take advantage of remote execution. If a remote server is available, the whole application is offloaded to the server, otherwise the device executes the whole program locally.

An overview of MOCHA and MAUI are discussed here which uses the program partitioning mechanism. Dynamic VM synthesis which is the main approach considered in this research is discussed in detail in Section 2.1.2.

1. MOCHA [SoMu2012]

MOCHA (MObile Cloud Hybrid Architecture) is 3-tier architecture which is proposed for real-time face recognition application. The authors argued how a high RTT latency between the MDs and the cloud is a challenge for real-time applications. A nearby cloudlet server is proposed for offloading the application.

MDs such as smartphones, tablets, and laptops are connected to the cloud via a nearby cloudlet. The cloudlet is assumed to support multiple network connections such as Bluetooth and Wi-Fi and the cloudlet is connected to high-speed Internet.

The system is based on partitioning multiple independent computations among available cloud servers and cloudlets. The cloudlet is the one responsible for partitioning the computation among itself and multiple servers in the cloud. Fixed or greedy algorithms are used to distribute the tasks. In fixed algorithm, the tasks are equally distributed among the available resource (cloud or cloudlet).Where as in a greedy algorithm, tasks are distributed to which ever server that can complete the tasks within a short response time. The total response time is the time that it takes for the last response to be returned.

According to their result, whether cloudlet is available or not, the greedy algorithm outperforms. On the other hand, the fixed and greedy approaches have similar performance when both the cloud and cloudlet servers have the same processing times and communication latencies.

2. MAUI [EdAr2010]

MAUI approach is also based on application partitioning. The Primary goal of MAUI design is to save the smartphones energy by remote execution of programs. From the experiment, the authors concluded that as RTT increases, the energy cost increases almost linearly for even the same network type. Low RTT latency and high BW of Wi-Fi technology make it more preferable over 3G to offload a code to a remote server. Even offloading codes to a remote cloud with Wi-Fi consumes more energy than offloading to a nearby server. The authors made interesting research to decide to use a local server than remote cloud for resource-intensive mobile applications.

The MAUI architecture physically consists a smartphone and Wi-Fi accessible MAUI server. The applications running on the MAUI servers are modified. Initially, the mobile application developers need to identify the part of the code that could be executed remotely and part of the code that do not benefit from remote execution. From the energy consumption point of view, it is not always beneficial to offload the remote annotated programs. So each time when offloading is invoked and if remote server is available, an optimization framework decides whether the program should be offloaded or not. Once an offloaded method terminates, MAUI gathers profiling information that is used to better predict whether future invocations should be offloaded. The CPU cycle saved from offloading, and the estimated BW and RTT latency from the real-time network connectivity determines the cost of offloading for MAUI approach. The cost formulates an optimization problem whose solution

dictates which program should be offloaded to the server and which should continue to execute locally on the smartphone.

2.2 VM based Cloudlet

"A cloudlet is a trusted, resource-rich computer or cluster of computers that's well-connected to the Internet and available for use by nearby MDs." [MaBa2009].

A cloudlet is a new architectural element for MCC that represents the middle tier of a 3-tier hierarchy: MD – cloudlet – cloud. As public clouds, such as Amazon EC2, VM abstraction is used in a cloudlet to serve as a trusted infrastructure which isolates different MDs application. Cloudlets maintains only soft state that is cached or otherwise re-creatable whereas public clouds maintain both soft and hard state.

Cloudlets are proposed to be self-managing. The self-management feature makes the widespread deployment of cloudlets more feasible. Also, pre-use customization and post-use cleanup ensures that cloudlets infrastructure is restored to its pristine software state after each use, without manual intervention.

There are two different approaches to deliver VM state to a cloudlet infrastructure. One is VM migration, in which an already executing VM is suspended, its processor, disk, and memory state are transferred, and finally VM execution is resumed at the destination from the exact point of suspension. The other approach is called dynamic VM synthesis which customizes a VM on demand.

Section 2.2.1 describes dynamic VM synthesis in detail which is originally proposed in [MaBa2009] and further modified in [KiPa2013].

2.2.1 Dynamic VM synthesis [KiPa2013]

Pre-use customization in a cloudlet infrastructure means customizing a VM on demand for users upon a request. VM customization in ordinary way consumes a lot of time. An ordinary way of VM customization time includes the time to create VM disc space, install operating system on it and install a particular application. Delivering a service after all the above steps consumes a lot of time. Minimizing VM customization time is the main objective of dynamic VM synthesis.

The intuition behind the dynamic VM synthesis approach is that although each VM customization is unique, it is typically derived from a small set of common base systems such as a freshly-installed Windows 8 guest or Linux guest.

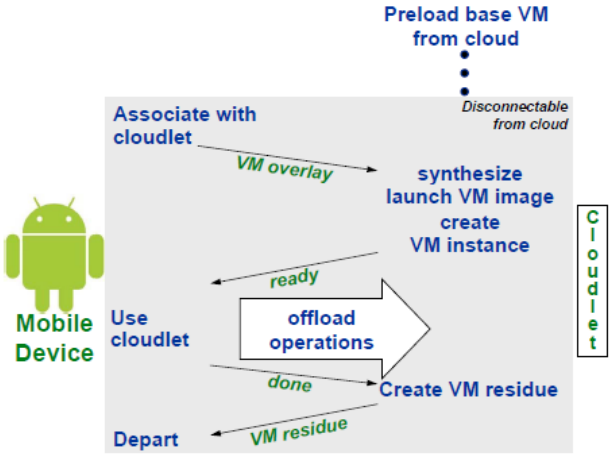


Fig. 2.1: Dynamic VM synthesis [KiPa2013]

Fig. 2.1 illustrates the dynamic VM synthesis approach. Offline preparation of base VM and overlay VM is the first step. A base VM is a VM with just an operating system installed on it. When a relevant software installed on the base VM, it is called launch VM. A launch VM is the VM image used for offloading by the MD. Once the user

starts offloading data to the launch VM, the VM become a VM instance for the MD. A VM overlay is the compressed binary difference between the base VM image and the launch VM image.

Dynamic VM synthesis is the reverse process of overlay creation. A MD delivers the VM overlay to a cloudlet that already possesses the base VM from which this overlay was derived. The cloudlet decompresses the overlay, applies it to the base to derive the launch VM, and then creates a VM instance from it. The MD starts offloading data to the VM instance. The instance will be destroyed after the session is over but the launch VM can be cached for future use.

Overlay can also be delivered from the remote cloud. Downloading VM overlay from remote cloud saves the energy consumption the MD which is used for data transmission. However, downloading the overlay from remote cloud may take longer time if the WAN BW is worse than the Wi-Fi BW.

2.2.2 Service Initiation Time

The service initiation time is the time from the MD discovers the cloudlet to the time the MD starts offloading data. The four main processes, binding the MD with the cloudlet, transfer the VM overlay, decompress the VM overlay, and apply the VM overlay on the base VM to create the launch VM, determine the duration of the service initiation time.

Binding the MD with the cloudlet is the first step. The binding sequence is the establishment of a secure TCP tunnel using Secure Sockets Layer (SSL) between the MD and the cloudlet. This step may also involve user authentication and billing

interaction. After successful authentication, VM overlay is transferred from the MD to the cloudlet. The transferred VM overlay is decompressed to be applied on the base VM. After applying the VM overlay, the launch VM in the cloudlet is ready to receive data from the MD.

The four applications shown in Fig. 2.2 are described below:

***FACE**: detects and attempts to identify faces in an image from a prepopulated database which runs on a Microsoft Windows environment.*

***SPEECH**: performs speech-to-text conversion of spoken English sentences. The Java-based application backend runs on Linux.*

***AR:** is an augmented reality application that identifies buildings and landmarks in a scene captured by a phone's camera, and labels them precisely in the live view. An 80 GB database constructed from over 1000 images of 200 buildings is used to perform identification. It runs on Microsoft Windows.*

***FLUID:** is an interactive fluid dynamics simulation that renders a liquid slopping in a container on the screen of a phone based on accelerometer inputs. The application backend runs on Linux. The structure of this application is representative of real-time games.*

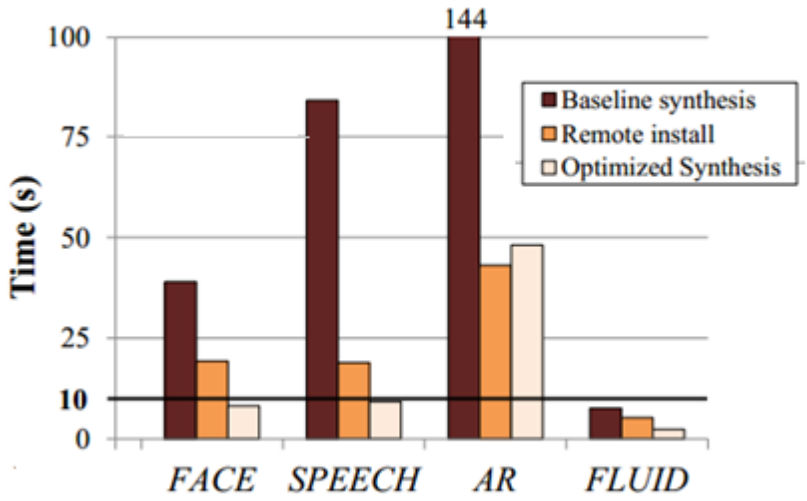


Fig. 2.2: Service initiation time [KiPa2013].

In order to minimize the service initiation time, Kiryong et al. [KiPa2013] applied four optimization techniques (Deduplication, Bridging the Semantic Gap, Pipelining and Early Start). This interesting research minimizes the VM synthesis time significantly. Fig. 2.2 shows the performance for the baseline synthesis, the optimized synthesis, and remote installation for four sample real-time applications. Remote installation approach runs a custom VM image. This involves launching a standard VM, uploading and installing the application packages, and then executing the custom applications. Due to frequent errors and varying performance, remote installation is claimed to be unacceptable method of launching a VM by the author even if it shows a better performance than the baseline.

2.3 Live VM Migration

Live VM migration is the key technology in the virtualization world which becomes an essential operation in a cloud resource management. Live VM migration is the process of moving a running VM from one physical host to another with a minimal downtime. This operation can be performed within a data center from one server to another or it can be performed across geographically distributed data centers. Some of the key reasons for VM migrations on enterprise level are listed below:

- **Cloud Bursting**: when an enterprise local servers are overloaded, migrating the highly stressed VMs to a cloud mitigates the problem. Once the workload reduces, the VMs are reallocated to the enterprise servers.
- **"Follow the sun":** enabling users to access application with low-RTT latency is the main reason for the "Follow the sun" strategy. For a scenario where multiple groups located in different continents that are collaborating on a common project, each group requires low-RTT latency access to the project applications and data during normal business hours. Migrating the VM in evening time from one site to other is one solution. For instance, teams involved in a project from China and Ottawa can benefit from this strategy. Since night time in Ottawa is a day time for China.

- **Green:** VMs may be migrated to consolidate VMs onto a smaller number of physical servers, allowing under-loaded servers to be shutdown completely to save power.

There are three key aspects to be considered in a live VM migration; RAM state, storage and network migration. The following three subsections discuss these aspects in detail.

2.3.1 RAM State Migration

Basically, there are two methods of RAM state migration; pre-copy and post-copy.

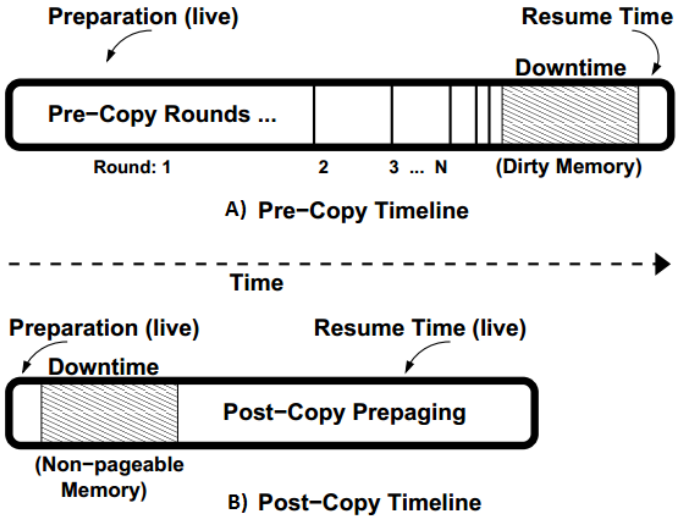


Fig. 2.3: Timeline for pre-copy vs. post-copy [MiUm2009]

A. Pre-copy Migration

Pre-copy is the famous approach than the post-copy. Different vendors such as (Kernel-based Virtual Machine) KVM, XEN and VMware adopts this mechanism. Pre-

copy migration first copies the memory state to the destination iteratively, with only the modified pages being sent during each iteration. (See Fig. 2.3 A)

Pre-copy migration involves the next three phases:

Push phase: While a VM is running in the source host, all memory pages are copied to destination in a first iteration. Modified pages from the previous round are copied iteratively to destination.

Stop-and-copy phase: After the source VM is paused, the remaining dirty pages are copied to the destination.

Pull phase: The destination VM is started. If the VM accesses a page that has not yet been copied, the page is pulled across the network from the source VM.

The disadvantage of pre-copy migration is that VM migration may not be successful if the RAM state is changing faster than page transmission rate to the destination. Also, if RAM write intensive application is running on the VM, sending the dirty pages for each iteration consumes a lot of BW; hence it degrades the performance for other active services.

B. Post-copy Migration

Post-copy migration transfers a VM's memory content after its processor state has been sent to the target host.

Post-copy migration follows the next two phases:

Stop-and-copy phase: A VM at the source node is suspended. Minimal processor state are copied to the destination node.

Pull phase: The VM at the destination is resumed without any memory content the VM at the destination begins fetching memory pages from the source through network.

The drawback of post-copy is when there is a network disconnection between the source node and the destination node during VM migration. Once after migration is started but not completed, the VM at the destination starts without any memory content and memory pages are fetched from the source node while the VM at the source node is down. Network disconnection at this time may result unsuccessful live VM migration.

2.3.2 Network Migration

Once memory state have been migrated, the VM active network connections has to remain open. In LAN migration, the VM can retain its IP and MAC address after migration. Then the destination host transmit an unsolicited ARP message that advertises the VM's MAC and IP address. This causes the local Ethernet switch to adjust the mapping for the VM's MAC address to its new switch port [ChKe2005].

Over a WAN, retaining the same IP address before and after VM migration is a challenge. Extending the layer 2 connectivity over WAN is one solution. For instance, CloudNet [TiRa2011] uses VPLS technology to extend LAN networks over the WAN such that the ARP message can be forwarded over the Internet to update the Ethernet

switch mappings at the source and destination sites. This allows open network connections to be seamlessly redirected to the VM's new location.

Mobile IP has been also used for network connection redirection [ToRe2014] without the constraint of retaining the original IP address. The VM used two IP addresses; one is called the Home Address (HoA) and the other one is called Mobile Address (MoA). The HoA is an IPv6 address used as a permanent identifier, and the MoA is a locator that is the temporary IPv6 address allocated every time when the VM moves. Every time the VM changes location, the VM updates an agent where the new location is. Traffics for the VM always goes through the HoA then all the traffics forwarded to the HoA will be redirected to MoA. This causes additional path for the traffics.

Multipath TCP is a transport layer solution which has been also used for VM migration to support network connection handover seamlessly. Even if the original IP address of the VM has changed, the connections remain established [CaCh2013]. It was implemented to migrate a VM running a client application. This is also the approach used in this thesis to migrate a VM running a server application (see Section 3.2.1). The detail about multipath TCP is presented in Section 2.4.

2.3.3 Storage Migration

Storage migration is one of the requirements for live VM migration. LAN based live VM migration considers a shared storage between source and destination host, thus eliminates the need to migrate the disk state.

Over WAN, storage migration can be the dominant one during a live VM migration in terms of both time and BW consumption. Disk migration is dependent on the disc reading speed, and the RTT latency and BW between the source and the destination host. For instance, DRDB storage migration system used in [TiRa2011] reading speed is 4MB/s. Even if a high BW is available, the speed of the storage migration is limited to the hard disc reading speed.

2.4 Multipath TCP (MPTCP)

Protocols are what makes it possible for different vendors computing devices to communicate with each other. TCP/IP is a protocol developed by United States Department of Defense for its research network - ARPANET during the mid-1970s. TCP/IP is a five layer protocol which uses TCP or UDP as a transport layer protocol. TCP provides a connection-oriented, reliable, byte stream service between source and destination host applications. Each established connections are identified by five tuples (Source and destination IP address, source and destination port number and the underlying protocol which is IP most of the time). Even if computing devices such as smartphones are integrated with multiple interfaces, TCP allows only a single interface to be used with the destination TCP connection at a time. If any of the five tuples changes within the life time of the TCP connection, TCP closes the connection abruptly. This feature of the TCP protocol limits the performance that can be achieved with available resources today. Smartphones can get a maximum throughput if they are able to use both their Wi-Fi and 3G/4G interfaces at the same time for streaming data from a single application. Multipath TCP (MPTCP) is an IETF protocol recently

developed which allows multiple interfaces to be used for a single application and socket interface [MPTCP] unlike the regular TCP/IP protocol.

2.4.1 Regular TCP Operation Overview

Before two end hosts start sending data reliably, they must establish a session. If the transport layer protocol is TCP, the three-way handshake starts the session. The client sends a SYN (for "synchronize") packet to the port on which the server is listening. The server replies with a SYN+ACK packet, acknowledging the SYN requested from the client. Then the client acknowledges with ACK to inform the server that the connection has established.

The first SYN packet sent by the client contains the source port and the initial sequence number chosen by the client, and it may also contain TCP options that are used to negotiate the use of TCP extensions. The same way as the client, the SYN+ACK packet replied by the server contains the server initial sequence number and the options that it supports. Only a single TCP connection is created per each session. Each TCP connections from session to session are uniquely identified by the IP addresses and the port numbers used in the initial three-way handshake.

After the connection is established between the client and the server, the client starts the data flow. The TCP layer segmented the data coming from the application layer. A sequence number is used to set the data in different segments, which is also used to reorder segments at the receiver side, and detect losses. After segments are received, the receiver acknowledges with cumulative acknowledgment. A cumulative

acknowledgement tells the sender which Byte of the segment the receiver is expecting next. This is one way of detecting loss.

A FIN packet is used usually to close a TCP connection. A FIN packet indicates the sequence number of the last Byte sent. The connection is terminated after the FIN segments have been acknowledged in both directions. If one of the hosts detect unusual situation such as if a segment is sent from different source IP or source port for the established connection, it sends Reset (RST) packet to close the TCP connection abruptly.

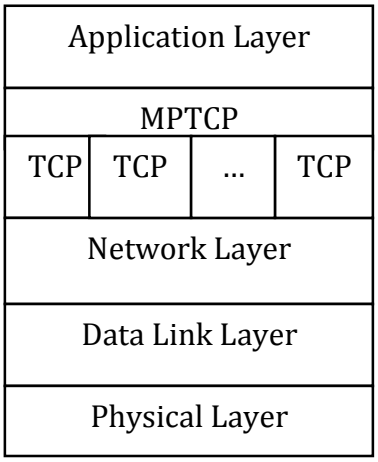


Fig. 2.4: MPTCP protocol stack

2.4.2 MPTCP Operation

MPTCP connection provides a bidirectional Byte stream between two hosts as a regular TCP. From the application layer point of view, both MPCTP and TCP appears to be the same. However, MPTCP supports more than one TCP connection with different IP addresses to stream data from a single socket interface. In general, MPTCP is a shim layer between an application layer and one or more TCP connections

(see Fig. 2.5). MPTCP is responsible for connection setup, transferring data fairly between TCP connections, adding subflows, removing subflows and tearing down the session for one or more than one TCP connections.

In order to make some terms used in the context of MPTCP protocol clear, they are defined as follows:

Path: A sequence of links between a sender and receiver defined by 4-tuple of source and destination address/port pairs.

Subflow: A flow of TCP segments operating over an individual path. Subflow is same as a regular TCP connection.

(MPTCP) Connection: There is only one to one mapping between a connection and an application socket. A set of one or more subflows can operate under a connection.

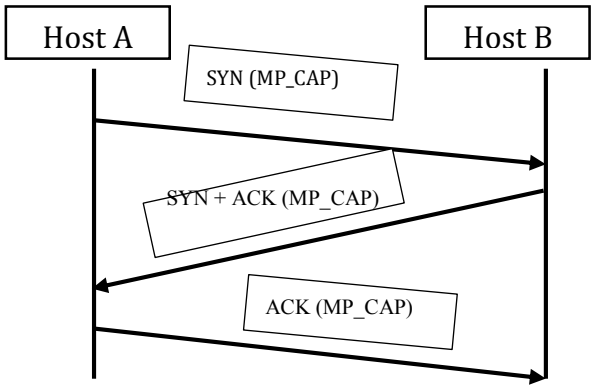


Fig. 2.5: MPTCP three-way handshake

1. Connection Setup

Initiating a MPTCP connection is the same as a regular TCP three-way handshake connection setup except that the SYN (synchronize), SYN+ACK (synchronize + acknowledgment), and ACK (only acknowledgement) packets carry the MP_CAPABLE (MPTCP Capable) option as shown in Fig. 2.5. One of the purposes of exchanging MP_CAPABLE option is to make sure that the remote host supports MPTCP before the connection is established. This option also enables the hosts to exchange a 64-bit random key which is used to verify the validity of adding a new subflows later. If the MP_CAPABLE option is dropped, MPTCP will gracefully fall back to a regular TCP connection.

2. Adding New Subflows

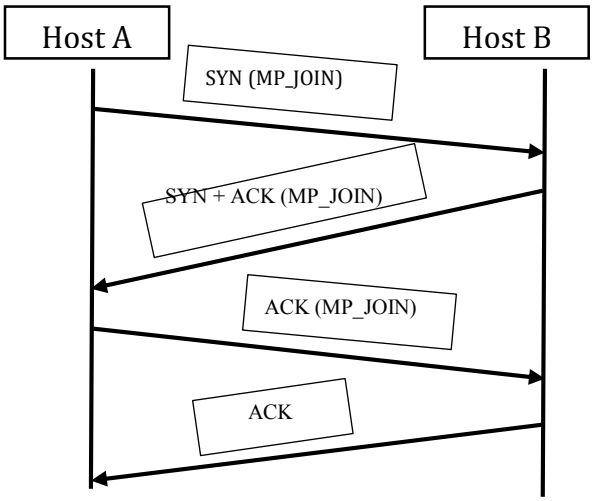


Fig. 2.6: Adding a subflow using MP_JOIN option

New subflows can be added after MPTCP connection is established. A host knows its own IP addresses and teaches the paired end host about its available IP addresses through signalling. Using the knowledge of IP addresses, a host can initiate a new subflow over unused pair of addresses.

Adding a new TCP subflow to the existing connection also uses a three-way handshake. During exchanging the SYN, SYN/ACK and ACK, the segments carry MP_JOIN option (see Fig. 2.6). The MP_JOIN option with keys from the original subflow indicates the endpoint that the TCP session is joining an already established particular MPTCP connection. Unless ACK packet is sent from server side after the handshake, the subflow remains in pre-established state. It is not permitted to send data on a pre-established state subflow. This is for the purpose security which is not the scope of this research. In general, MP_JOIN takes two RTT time to be established.

Subflows can be added with equal priority or as a backup from the existing subflows. The MP_PRIOR option from the sender notifies the receiver the status of the subflow. Data is never transmitted on the backup subflow unless the current functioning subflows fails. As soon as the active subflow fails, the backup subflow takes over.

3. Address Advertisement

A host advertises its available additional IP addresses using the ADD_ADDR22 MPTCP signalling. This option can be send at any time during a connection when paths become available. The ADD_ADDR22 signal receiver synchronizes a subflow

using the additional IP address after checking whether the signal is coming from the right host. The MP_JOIN option is used to establish TCP subflow with the IP address advertised with the ADD_ADDR22.

4. **Remove Address**

Interfaces may disappear during the lifetime of MPTCP connection. The affected host should send REMOVE_ADDR option to inform the peer that the interface is no longer available. Before the REMOVE_ADDR signal receiver acts on it, it checks the unavailability of the path by sending signals. If a response is received, the receipt of REMOVE_ADDR ignores removing the subflow. If there is no response is received, it sends RST signal on the affected subflow and then it removes the IP address.

5. **Closing a connection**

A connection may be closed abruptly or normally in a subflow level or in a MPTCP connection level. A FIN packet in MPTCP only affects the subflow on which it is sent. When a sender finishes sending data, it closes the MPTCP connection with DATA_FIN packet. With all subflows closed, MPTCP connection remains open to enable *break-before-make* handover between subflows.

A subflow is abruptly closed if a host sends RST signal whereas MP_FASTCLOSE option is signaled in order to close the MPTCP connection abruptly.

6. **Flow Control**

During TCP connection set up at the beginning, for flow control purpose, hosts exchange window size indicating the number of Bytes receiver can buffer. The sender could not send more than the receiver buffer amount before receiving acknowledgment. MPTCP maintains a single receive buffer pool for all subflows per MPTCP connection. The receiver window indicates the maximum data sequence number that can be sent rather than the maximum subflow sequence number.

Table 2.2: Summary of MPTCP signals

Signalling	Name	Function
MP_CAPABLE	Multipath TCP Capable	Checks the capability of the end host on establishing a MPTCP connection
MP_JOIN	Join Connection	Adds additional subflow to existing MPTCP connection
REMOVE_ADDR	Remove Address	Removes failed subflow
MP_PRIO	Multipath Priority	Inform subflow priority
MP_FASTCLOSE	Fast Close	Closes MPTCP connection abruptly.
ADD_ADDR22	Add Address	Informs the availability of additional IP address to the paired host

2.5 Related Works

MCC enables MDs to offload resource-intensive applications and data to clouds or cloudlets to save energy and leverage the capacity of computation. In a scenario where there are one or more cloudlets in a nearby location, mobile user could decide the well suited cloudlet to offload resource-intensive application. Researchers have studied the selection of the best cloudlet to the MDs based on the wireless network quality, the BW, the RTT, and the load on the cloudlet server.

[JiKa2013] Jiwei et al. have discussed the lack of consistent network performance for mobile user while offloading; the optimal offloading decision becomes a suboptimal due to the MD user movement. The authors proposed, a three-tier (Smartphone <-> cloudlet <-> cloud) architecture called ENDA (Embracing Network inconsistency for Dynamic Application offloading in MCC) that track users location using GPS and save it in centralized database found in a remote cloud. The location story helps to predict user's movement and to identify the energy efficient Wi-Fi AP. The real-time network performance between smartphone and AP and server-side load are considered to make optimized offloading decisions.

ENDA has considered multiple cloudlet on specified area and proposed a centralized cloudlet system which totally differs from the decentralized VM based cloudlet [MaBa2009]. The research has focused only on energy saving but nothing has been mentioned how the state of the user's application could be transferred from one cloudlet to the other.

[KoDe2014] also considers more than one cloudlet in a nearby geographical location. The authors have claimed that in order to make the best cloudlet selection, the smartphone has to consider the processing speed and the available memory size of a cloudlet. Also the wireless network RTT latency between smartphone and AP and the BW of fixed network from the cloudlet to the remote server has been considered as a crucial metrics while making cloudlet selection at the first time. Unlike the ENDA architecture, the real-time network performance is not considered in this paper.

[JaTa2013] has studied how using cloudlet through Wi-Fi can save energy of the smartphones than accessing remote cloud through 3G/4G. Even if this paper has mentioned about transferring the state of the offloaded application from one cloudlet to the other while the user is moving, the offloading mechanism and the networking collaboration between cloudlets is not mentioned.

This thesis clearly mentioned how the current state of the MD transferred from one cloudlet to the other. A VM based cloudlets are assumed to exist in a geographical vicinity. The network collaboration between the cloudlets assists the seamless live VM migration between cloudlets with the help of MPTCP. This thesis does not consider the real-time Wi-Fi network quality and the work load of the cloudlets.

2.5.1 Connection migration over WAN

One of the challenges of live VM migration over WAN is the migration of the network or the connection migration. A server VM connected to multiple clients need to keep the connection after it changes its original location. In order to achieve seamless connection migration, the TCP layer put its limitation to the system. The IP address of the client and the server identifies the TCP connection. In order to keep the connections established, the VM needs to retain its original IP address in the new location. Designing a system to keep the IP address the same does not give flexibility. The VM can only be migrated to a pre-know locations. One of the technologies used to achieve this is VPLS by extending the LAN over the WAN. The limitation of this

approach is that the gateway remains in remote location which makes it hard for efficient routing.

LISP (Locator/ID Separation Protocol) [FaMe2013] is a routing architecture that separates the location from the identity of hosts, and enables to provide mobility and multihoming. Even if no modification is required to the VM, it has a weakness against the routing convergence when the VM has moved.

Mobile IP is another approach that separates the location from the identifier which adopts the call forwarding approach. Fig. 2.7 shows a mobile IP diagram. The network the MD registered initially is called the Home Network and the new location the MD has moved is called the Foreign Network. Whenever the MD changes location, it notifies the Home Agent about the new location (Foreign Network). The traffics forwarded to the MD goes directly to the Home Agent who knows about the location of the MD. The Home Agent changes the destination IP address of the incoming packets to the Foreign Network address of the MD (care-of-address).

Fig. 2.7 shows the flow of traffic from Server X to MD A after the MD changes it location to the Foreign Network. The incoming traffic from Server X goes through 1->2->3. But when packets are destined to Server X from MD A, the traffic follows the best route based on the routing table (4->5).

Mobile IP has been also used for network connection redirection [ToRe2014] without the constraint of retaining the original IP address. This approach causes additional path for the traffics and it hides the congested link from TCP congestion control mechanism. The authors admitted that using their approach adds about 1 sec

in the total VM migration and 2 sec in the VM downtime from the ordinary live VM migration.

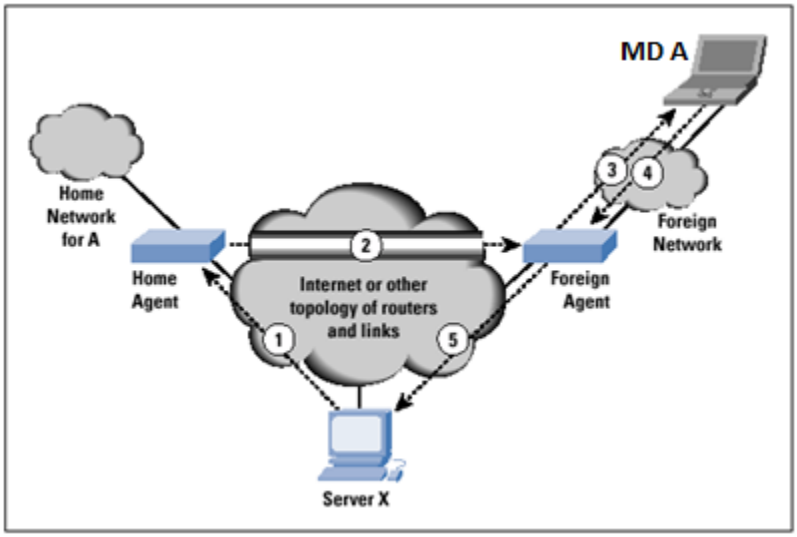


Fig. 2.7: Mobile IP

All the above approaches: VPLS, LISP and Mobile IP are proposed to overcome the limitation of the TCP protocol. If the transport layer can be decoupled from the network layer, all the problems can be solved without any modification of the existing network.

The approach used in this thesis is by implementing the MPTCP protocol in each node which does not require network modification and it also decouples the TCP layer from the IP layer. MPTCP also converges fast to a routing change as a regular TCP.

Chapter 3 Seamless Live VM Migration between Cloudlets

This chapter describes a seamless live VM migration between two cloudlets, where such an event is triggered by the location of a MD. Section 3.1 discusses the problem and the proposed solution using a multiple cloudlets scenario. The approach used to accomplish the live VM migration with the support of MPTCP protocol is described in Section 3.2. In addition to a seamless live VM migration, a smooth MD handover mechanism from one cloudlet to the next is studied. Two types of handover scenarios are covered in Section 3.3. To accomplish all the above, MPLS based VPN is proposed to create a collaboration between cloudlets which is discussed in Section 3.4.

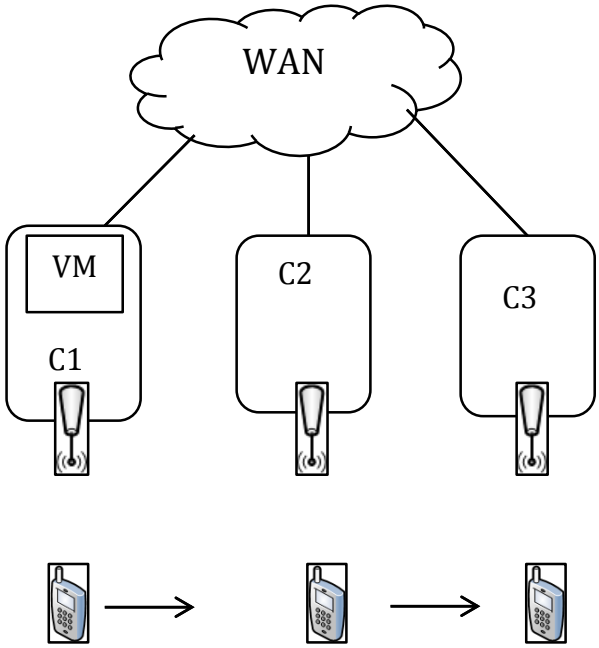
3.1 The Problem and the Proposed Solution

A nearby VM based cloudlets have been proposed to improve MCC system. Proximity of servers to MDs minimizes RTT latency. Offloading to a remote server through minimum RTT latency not only improves QoE but also saves the energy of the MDs.

Dynamic VM synthesis [KiPa2013] (see Section 2.2.1), customizes VM on demand for cloudlet users. After service initiation time, user starts offloading data to the launch VM to enhance the applications performance. Since the cloudlet is one hop

away from the user, only a minimum delay will be experienced as long as the user remains in the communication range of the cloudlet.

If a MD user changes location and relies on the cloudlet service, the MD needs to wait for service re-initiation time to start offloading. During the service re-initiation time, the MD's service is temporarily disconnected. Users may find the extended delays for service re-initiation at a new location unacceptable from the QoE perspective.



Legend: C1, C2, and C3 are cloudlets while VM is a virtual machine instance.

Fig. 3.1: A scenario with multiple cloudlets located in a nearby geographical location

In the scenario shown in Fig. 3.1, the cloudlets are assumed to be integrated with wireless AP. The client, MD, uses the VM instance in a cloudlet as a server.

Analysing the Scenario:

The following steps explain the existing possible operation based on the scenario illustrated in Fig. 3.1.

1 The MD initiates a service with the nearby cloudlet (C1) within a service initiation time and starts offloading to the launch VM.

2 After some time, the MD user walks away from the communication range of C1 and enters the communication range of C2.

3 The MD requests for a new IP address from C2.

4 If C1 permits to be accessed from remote, C2 forwards traffic from MD to C1 through the WAN after the MD initiates a new TCP connection with the VM instance. Otherwise the MD launches a new VM with C2.

The first problem from the above operation starts at step 3 when the MD's IP address is changed. A new IP address for the MD closes the established TCP connection with the VM instance which compels the MD to start a new TCP connection.

The second problem occurs at step 4. At step 4, both cases have their merits and demerits. The first case is accessing C1 from C2 which involves the WAN RTT latency and this contradicts the purpose of using a cloudlet even if it eliminates extra service re-initiation time with C2. The second case is initiating a new service with C2

which eliminates the WAN RTT latency with a cost of delayed extra service re-initiation time before offloading starts.

The Proposed solution: The main idea of the proposed solution is to combine the advantages of the above two cases involved at step 4. These are eliminating extra service re-initiation time and reducing the duration that the MD accesses the VM through the WAN. In order to achieve this, the following three points describe the proposed solutions.

1 To avoid re-establishment of TCP connections, the proposed solution is to use MPTCP (see Section 2.4) transport layer protocol. With MPTCP, it is possible for the connection to remain established after the MD changes its IP address, which can significantly reduce the delay.
2 In order to make sure that traffic is forwarded between the cloudlets, creating a secure and a trusted network collaboration using Virtual Private Network (VPN) between geographically nearby cloudlets is one solution.
3 To eliminate extra service re-initiation time, migrating the existing state of the VM instance from the first cloudlet (C1) to the next cloudlet (C2) is proposed.

The third proposed solution has its own problem. Changing location of VM instance from one LAN to another forces the VM instance to change its IP address. As the solution provided to the MD, MPTCP protocol assists the live migration of the VM instance without closing the established connections. The MD runs a client application while the VM instance runs a server application. Additional mechanism is

needed to accomplish a live server VM migration without any connection interruption which is described in detail in Section 3.2.1.

Expected Performance of the Proposed Solution

The main objective of migrating the VM instance is to make the RTT latency between the MD and the VM as minimum as possible. This could be achieved after time T which can be expressed as:-

$$T = T_{CIP} + T_{CVM} + T_{MVM} + T_{VMIP} \quad (3.1)$$

where

T_{CIP}: the time for the MD to be connected with the neighboring destination cloudlet and to acquire a new IP address.

T_{CVM}: the time the source cloudlet takes to make a decision on VM migration.

T_{MVM}: the time VM needs to be migrated from the source cloudlet to the neighboring cloudlet.

T_{VMIP}: the time the destination cloudlet takes to assign IP address to the VM.

The overall performance is dependent on the time duration T and the quality of the service provided to the MD during this duration. Fig. 3.2 shows the ideal expected performance in terms of RTT latency before, after and during live VM migration.

Let:

$RTT_{VM\text{-}MD}$ is the network round trip time (RTT) between the MD and the VM.

RTTc-c is the network RTT between the source and the destination cloudlet.

RTT_{MD}-c is the network RTT between the nearby cloudlet and the MD. Assume the same load in each cloudlet, the same network latency between the MD and the cloudlet, and the same Wi-Fi technology for each cloudlet.

The network RTT between the VM and the MD before the MD changes location is expressed as:

$$RTT_{VM\text{-}MD} = RTT_{MD\text{-}C}. \quad (3.2)$$

The RTT between the MD and the VM after the MD changes location and before the VM is migrated to the destination cloudlet involves the RTT latency between the source and destination cloudlet in addition to the RTT latency between the nearby cloudlet and the MD. This can be expressed as:

$$RTT_{VM\text{-}MD} = RTT_{MD\text{-}C} + RTTc\text{-}c + RTT_{delta} \quad (3.3)$$

RTT_{delta} is the amount of the additional RTT caused by the VM migration traffic and the value depends on the available BW and the RTT between the source and destination cloudlets.

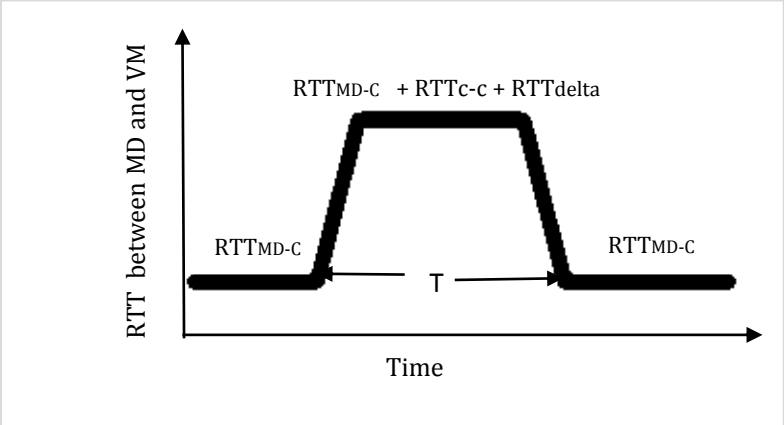


Fig. 3.2: Expected latency performance before and after the MD changes location.

After the VM migration is completed to the destination cloudlet, the RTT latency between the MD and the VM drops down to the RTT latency between the nearby cloudlet and the MD. This can be expressed as:

$$RTT_{VM\text{-}MD} = RTT_{MD\text{-}C}. \quad (3.4)$$

Reducing time T gives a better overall performance of the system. T_{VMIP} is equal to zero with the proposed approach described in Section 3.2.1. From all the tasks, VM migration elapses for longer duration time than the others. The VM migration time mainly depends on the following parameters: the RTT latency and BW between source and destination cloudlets, the workload of the source cloudlet (C1) and the destination cloudlet (C2), the application running on the VM, and the VM monitor (or expressed as hypervisor) used. The effect of the first parameters, the RTT latency and BW are explained and empirically evaluated in Chapter 4.

3.2 MPTCP for Seamless Live VM Migration

Live VM migration is the process of moving a running VM from one physical host to another one while the application is running and without interrupting the active connections. Regarding about the connection migration along with the VM instance, the underlying transport protocol plays the role. The most commonly used transport layer protocol is TCP. TCP uses five tuples, *the source IP address, the destination IP address, the source port, the destination port and the protocol* to identify a connection between a server and a client. If any of the five tuples change during the lifetime of the TCP connection, the established connection will be closed abruptly. In order to continue the connection with the server, the client needs to initiate a new connection.

Over the WAN, VM migration is performed from one network to another one which forces the IP address of the VM to be changed. This causes the VM to restart all the established connections. In previous researches, to retain the original IP address of the VM after migration, a layer 2 network extension over the WAN has been proposed with the help of tunneling and virtual private LAN service (VPLS) technology.

In this research, the use of MPTCP protocol in both the client and the server is proposed. Unlike the traditional TCP, MPTCP as a transport layer protocol resolved the problem stated above. However, MPTCP works only if the migrating VM runs a client application as only the client can send SYN packet with the new IP address

acquired after migration. Even if a server supports MPTCP, the server does not synchronize automatically after the IP address is changed. Since, the main objective of this research is to migrate a server VM instance to the nearest possible cloudlet; the existing MPTCP does not meet the entire requirement unless the kernel code is modified. To resolve this limitation without modification of MPTCP code, this research proposed another solution to support a live VM migration without having to re-establish a new TCP connection.

The following subsection discusses how MPTCP meets the requirement of this research objective.

3.2.1 Migrating a Server VM with MPTCP Protocol

If a VM server has only one virtual interface and if it is restarted, all the connections the server has with its clients will be lost regardless of the transport layer protocol used. This is how both MPTCP and TCP are designed purposely to be compatible with the Internet standards. Server will never initiate a connection with a client. Modifying the kernel implementation to meet this requirement is one solution, but this solution will cause a complication if a server initiates a connection with a client. The reason is that the client IP address may not be always the original IP address. For instance, in a network path that involves network address translation (NAT) box, the original IP address of the client is hidden.

This thesis proposed two additional approaches with a collaboration of MPTCP protocol. The first proposed approach is for the VM instance to have two

virtual interfaces and the second one is to construct a way the VM instance to know its future IP address. The following subsection describes how these two solutions will work together with MPTCP protocol support.

3.2.1.1 Prior Knowledge of the Next IP Address

Before we proceed to the approach used in this research, an analogy of job relocation with two phone numbers is given below.

"*George was told that his job location would be changed from Ottawa to Calgary but the exact time was unknown. Since he knew what his phone number would be in Calgary, he started giving his number to his friends. He told his friends to use the Calgary phone number in case he is unreachable with the Ottawa number. The time came and he moved; his friends called him with the Ottawa phone number, but they received no response. They immediately call him with the Calgary phone number. This way, George would never be disconnected from his friends."*

From the above analogy, a prior knowledge of the new location phone number is important to keep the connection established. How the VM would know what the IP address will be in the new location is the problem. Setting up a network administration strategy can resolve the issue. One of the possible network administration strategies is given below:

LAN IP address: 10.4.0.0/24

Broadcast IP: 10.4.0.255/24

MD IP address: The Evens (10.4.0.2/24, 10.4.0.4/24 ...).

VM IP address: The Odds (10.4.0.3/24, 10.4.0.5/24 ...).

Cloudlet IP address: 10.4.0.1/24

Reserved IP address: 10.4.0.224/24

The VM IP address is the next odd IP address of the paired MD IP address. From the scenario shown in Fig.3.1, assume that the LAN IP address of C1 is 10.4.0.0/24 and C2 is 10.5.0.0/24. The service was initially initiated at C1 with IP addresses of the MD 10.4.0.2/24 and of the VM 10.4.0.3/24. After the user changes location, the MD is assigned with IP address of 10.5.0.8/24. As soon as the VM is accessed with this new source IP address, the VM knows that the new location IP address would be the next odd IP address which is 10.5.0.9/24.

Querying the remote DHCP server for unused IP address may also be one solution to know the IP address of the migrating VM instance in the destination host. Once after the DHCP replies with one unused IP address, the DHCP labels that IP address as a reserved IP address. The duration that the provided IP address remains to be reserved depends on the local policy. If in case the VM migration is unsuccessful, the IP address has to be released. The DHCP can track the reachability of the VM instance with the IP address it provides. After some time if the VM is reachable, the IP address can be considered as an assigned IP address.

3.2.1.2 Advertising the New IP Address

The ADD_ADDR22 (see Section 2.4.2) option from MPTCP protocol is used by the VM to inform the client that there is a new or additional IP address. In order to send the ADD_ADDR22 option, there has to be at least one active subflow between the VM and the MD.

This thesis proposes the VMs to have two virtual interfaces inside a cloudlet. One interface is to operate in the regular VM operation mode, while the other one works only after the VM is migrated. The proposed approach is described in detail as follows.

As we can see Fig.3.3, the VMs are configured to have two virtual Ethernet interfaces. Both Eth0 and Eth1 are added to a bridge that connects to the wireless LAN (WLAN) or to the cloudlet Ethernet port which is connected with wireless AP.

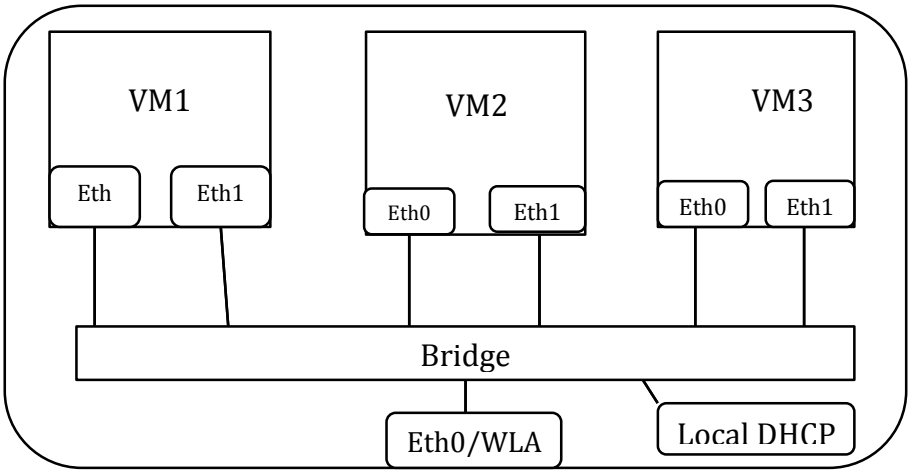


Fig. 3.3: Bridged VM networking inside a cloudlet

Usually, either Eth0 or Eth1 is used to communicate with the MD at a time. If Eth0 is up, Eth1 is down. After a single VM migration the state of the two interfaces will exchange. Always, the down state interface is intended to serve as the interface during migration. When a VM is accessed from a different source IP address than the existing LAN IP address of the VM, the VM knows that the paired MD changes location. The VM makes itself ready to be migrated by turning on the down interface. The IP address to be assigned to this interface is determined by the paired MD source IP address.

Although, ADD_ADDR22 advertises the existence of the additional IP address to the MD, no subflow can be started with the additional interface before the VM is migrated. Since the IP address assigned to the additional interface is from a different LAN, the VM is unreachable through this interface. But the connection from the MD side will keep trying to create a new subflow with the new IP address of the VM.

The VM will be available through the additional interface right after the migration is completed. At this time the previous working interface becomes inactive. Since the MD knows the additional IP address of the VM which was advertised before migration, the MD initiates a connection through that interface. As soon as the new subflow is started, the previous interface will be taken down to make it ready for the next potential migration. Table 3.1 summarizes the operation mode of the VM and the interfaces status.

Table 3.1: Number of subflows and operation status of the server VM

Operation Status of VM	Interface Status		Number of Subflow
	Eth1	Eth0	
Normal operation time	Down	Up	1
After VM is accessed with different LAN IP address	Up (inactive)	Up	1
Right after migration is completed	Up	Up (inactive)	1
Normal Operation time	Up	Down	1

3.3 MD Handover with MPTCP

A computing end device installed with MPTCP protocol stack can be configured with three different operational modes based on the requirements. Each operational mode identifies the handover mode as well. Handover is the way how the existing connection of a client is transferred from one available network to the other. The following subsection describes the three types of handover modes with MPTCP. Then selection of the right handover mode for the MDs and the VM for this research is presented.

3.3.1 MPTCP Handover Modes:

The following are the three general handover modes of MPTCP. A computing device with one or more interfaces can select one of the handover modes based on the need.

1 **Full-MPTCP Mode:** This is a MPTCP mode that allows creating TCP subflow with all active existing IP addresses. For instance, a MD integrated with both Wi-Fi and 3G/4G interfaces can benefit from the full-MPTCP mode using both interfaces at the same time to achieve the maximum possible throughput. If one of the interfaces goes down, the other active interface keeps working without any disruption.

2 **Backup Mode:** If backup mode is enabled, MPTCP opens TCP subflows with all available interfaces just like full-MPTCP mode. However, only a subset of interfaces is active at normal data transfer based on the priority of the interfaces. The MP_PRIO option sent by the peer is used to identify which interfaces are used as a backup. As the other peer received the MP_PRIO signal, MPTCP keeps the subflow open but never sends data unless the current active interface goes down.

3 **Single-Path Mode:** Only a single subflow is active with this mode. Since MPTCP protocol has the *break-before-make* feature, it is able to keep the established connection open for retransmission time out (RTO) time after the active subflow is lost. This feature enables peer devices to continue data transfer with the open subflow after a new IP address is acquired for the same interface or from different interface. A handover for a smartphone can be from Wi-Fi to 3G/4G network after it is totally disconnected from the Wi-Fi network. Compared to the Backup mode, this waits for two more round-trip times before the new MPTCP subflow is established and data can be sent.

Both full-MPTCP and backup modes act the same as the single-path mode if only one interface could be active at a time. Consider a scenario where a MD user walks from one Wi-Fi AP to the other Wi-Fi AP, only a single subflow can be active with a single Wi-Fi interface. The former Wi-Fi connection has to be disconnected to be re-connected with the later Wi-Fi AP. This approach does not generate any application disruption, as the MPTCP protocol establishes a new subflow with the existing connection using the new IP address. This is possible because of the *break-before-make* feature of MPTCP protocol.

3.3.2 Handover Scenarios for MDs

In this research, two MD handover scenarios are considered. The objective of the following analysis is to perform a handover before the MPTCP connection is closed between the VM and the MD.

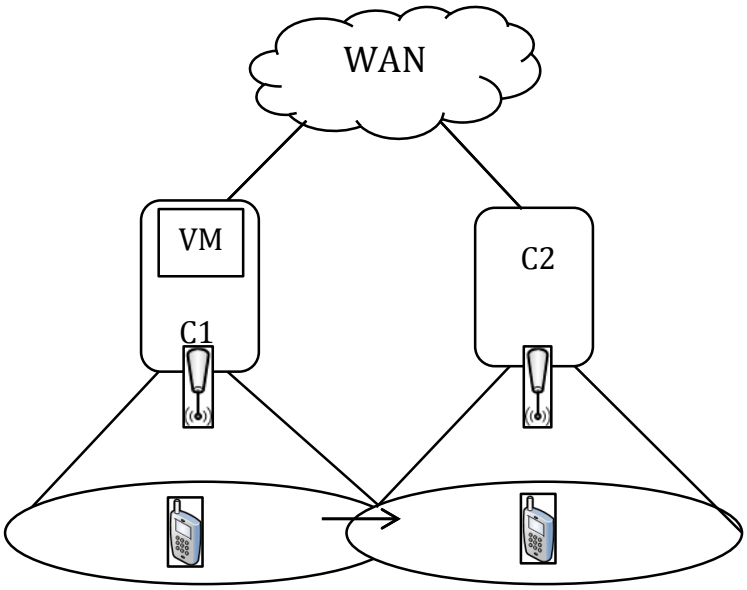


Fig. 3.4: Overlapping Wi-Fi network region

I. Directly from Wi-Fi to Wi-Fi

Two geographically close Wi-Fi networks may overlap or may have a distance between each networking area. The distance between the two Wi-Fi network coverage may enable the MD to handover without any interruption if the MPTCP protocol is installed in the MD and the server the MD is connected to. The RTO object in the MPTCP protocol gives enough time for the MD to acquire a new IP address and to resume the data transfer from where it has stopped without reestablishment of a new TCP connection with the server (if the application session timeout is longer).

If the two network regions overlap as shown in Fig. 3.4, for a MD moving from C1 to C2 region, the time needed to create a new subflow with the open MPTCP connection is the time the MD assigned an IP address from C2.

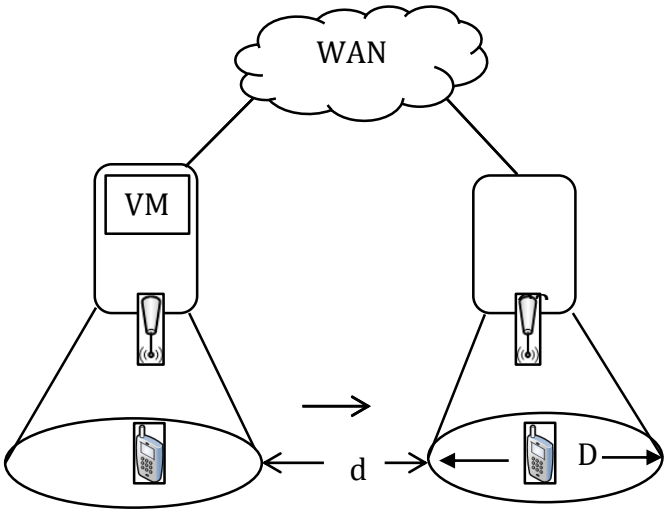


Fig. 3.5: Two Wi-Fi network regions separated with distance d

In a case where there is no overlap between two cloudlets network regions, as shown in Fig. 3.5, the maximum distance *d* that allows a MD to catch up on the established MPTCP connection is calculated below.

$$d <= (RTO - RTT/2 - T_{IP}) * V_{hum} \quad (3.5)$$

where d is the distance between the two cloudlet network regions, RTO is the retransmission timeout for the MPTCP connection, RTT is the round trip time between the APs through WAN, V_{hum} is the average walking speed of human being and T_{IP} is the time that the MD obtain a new IP address from the local DHCP .

The RTO for the MPTCP protocol varies from 13 to 30 minutes [ChGr2012]. The actual RTO value depends on the RTT latency between the client and the server. In order to guarantee for the new subflow to be created before MPTCP connection timeouts, the minimum RTO value which is 13 minutes is used to calculate the maximum distance between two cloudlet network regions.

The average human walking speed is assumed to be 1.2 meter per second [JoTh2004]. However, a runner may take a very short time to run from one network region to the other, the thesis assumes that the user walks with the average human walking speed, i.e.,

$V_{hum} = 1.2m/sec$

RT0 = 13min = 780 sec

The value of the RTT and T_{IP} depends on the real-time network condition. In order to calculate the ideal maximum distance d, this thesis neglected the value of RTT and T_{IP}. Hence,

$$d <= RTO * Vhum = 780 \text{ sec} * 1.2 \text{ m/sec} = 936\text{m} \quad (3.6)$$

If D/2 is the Wi-Fi AP communication range in meters, D + d will give the maximum distance between the two cloudlets which allows a handover of MDs before the established MPTCP connection timeouts. The Wi-Fi communication range distance, D, depends on the channel, the modulation, the antenna type, the multiplexing and the location of the cloudlet used by the wireless local area network (WLAN) technology.

This calculation for maximum distance d gives the ideal distance based on the transport layer protocol. However, applications have their own session timeout values. Before the MPTCP connection gives up on waiting for a response, application session may timeout.

The MPTCP handover mode used for this scenario could be full-MPTCP or single-path MPTCP. Both handover modes have the same performance since only one subflow can be active at a time. Fig. 3.6 shows the sequence diagram for the handover that takes place between the MD and the VM. From Fig. 3.5, first the MD user was in C1 network region. The VM instance is launched in the nearest cloudlet server (C1). As the user walks out from the network region of C1 and is connected to C2, MPTCP protocol handovers the established connection seamlessly.

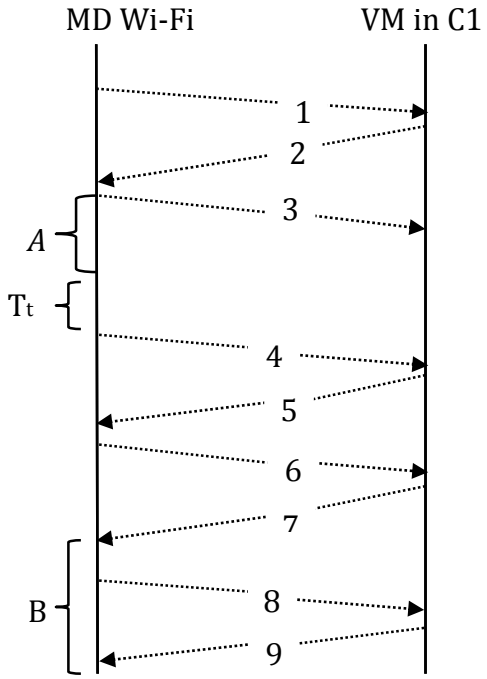


Fig. 3.6: The handover sequence diagram between a MD and a VM

The sequence diagram shown in Fig. 3.6 is described below:

1 Steps 1, 2, and 3 are the three-way handshake of the MPTCP between the VM and the MD with the MP_CAPABLE option.

2 *A* represents the duration that data has been transferred between the MD and the VM while the MD remains in C1 network region. T_t is the time that the MD spent without any network coverage which is less than the actual RTO of the connection.

3 After the MD is disconnected from C1 and connected with C2, C2 assigns a new IP address to the MD. The three-way handshake starts at step 4 which is the

SYN packet with the JOIN option. Step 7 indicates that an acknowledgement for the JOIN subflow. B stands for the data transfer using the new subflow.The REMOVE_ADDR option is sent from the MD to the VM in C1 to inform the previous address of the MD is no longer available (step 8). An acknowledgement is sent in a response (step 9).

II. From Wi-Fi to 3G/4G to Wi-Fi

The ubiquitous nature of 3G/4G is one advantage of cellular network over Wi-Fi network. Smartphones and tablets integrated with the 3G/4G capability can take the advantage of this feature to be transferred between Wi-Fi networks seamlessly with the help of the MPTCP protocol.

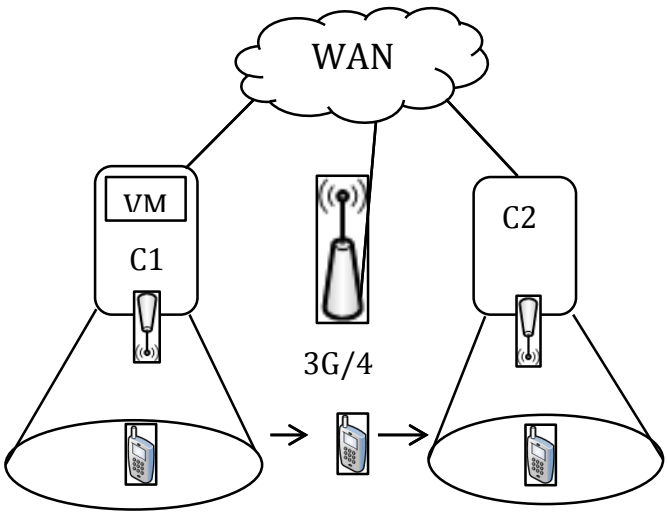


Fig. 3.7: Scenario with Wi-Fi and 3G/4G networks

The existence of a cellular network between two Wi-Fi networks does not change the maximum distance that should be between two cloudlets that potentially would perform live VM migration (see equation 3.5). The reason is that, for this scenario, only MDs that have 3G/4G capability can benefit.

The handover mechanism is the same as that of the Wi-Fi to Wi-Fi network handover. But for a MD connected with the cellular network, all the three handover modes, full-MPTCP, backup, and single-path, can be used depending on the required performance by the user. For instance, if energy efficiency is more important than the throughput, the backup or the single-path MPTCP handover mode can be used.

Full-MPTCP handover mode provides a better throughput since both the Wi-Fi and 3G/4G interfaces can be used simultaneously.

The sequence diagram for backup handover mode which is illustrated in Fig. 3.8 is described as follows.

1 Steps 1, 2 and 3 are the three-way handshake of the MPTCP protocol between the VM and the MD with the MP_CAPABLE option. This subflow is started with the Wi-Fi interface of the MD.Data transfer starts with the Wi-Fi subflow (*A* shows the data transfer) while the JOIN option of MPTCP protocol is sent through 3G/4G interface (steps 4, 5, 6, and 7). The JOIN option includes information which indicates that this particular subflow is a backup subflow. No data will be sent through this subflow unless the Wi-Fi subflow fails.

2 As soon as the MD is disconnected from the Wi-Fi network, the data transfer shifts to the backup subflow (*B* denotes the data transfer). The REMOVE_ADDR (step 8) option is sent through the 3G/4G subflow to inform the VM that the previous IP address is no longer available. At step 9, an acknowledgment is sent in a response from the VM to the MD.

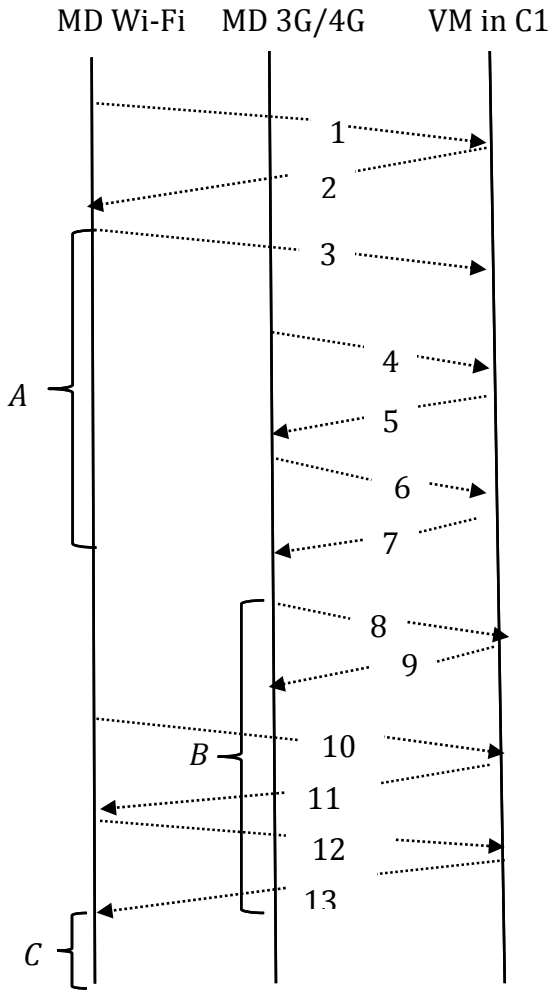


Fig. 3.8: Sequence diagram for MPTCP backup handover mode for Wi-Fi to 3G/4G to Wi-Fi

3 After the second Wi-Fi AP assigns an IP address to the MD, the JOIN option is sent through the Wi-Fi interface of the MD with the new IP address (steps 9, 10, 11 and 12). Immediately after the Wi-Fi subflow is formed, the data transfer shifts from the 3G/4G subflow to the Wi-Fi subflow (*C* denotes the data transfer) then the 3G/4G subflow remains as a backup.

3.4 Networking Collaboration between Cloudlets

The main advantage of MDs is their light weight with a mobility feature. Users can take this advantage and perform tasks while they are moving. A frequent movement of the MD users is common but the performance delivered to the users varies. The service availability, real-time network condition and the nature of radio channels limits the service providers to deliver the same performance while users move from one location to the other.

If a service is available in different locations and if there is no collaboration between the two service providers, it is impossible to transfer the user state from one service location to the other seamlessly. The user can notice the service disruption and also involves in the service initiation to the new location. For example, cellular (3G/4G) network transfers open connections from one antenna to the next seamlessly, because there is a networking collaboration between the sites.

For a MCC, Satyanarayanan et al. proposed cloudlet servers to be decentralized and to be managed only by the local businesses [MaBa2009]. In a scenario where multiple cloudlets are in a nearby geographical location (scenario 3.1), it is impossible

to make a seamless live VM migration from one cloudlet server to the next if there is no collaboration between those cloudlets.

In this research, a VPN connection between cloudlets is proposed to allow a seamless live VM migration between two cloudlets. This is discussed in the following subsection.

3.4.1 VPN for Cloudlets

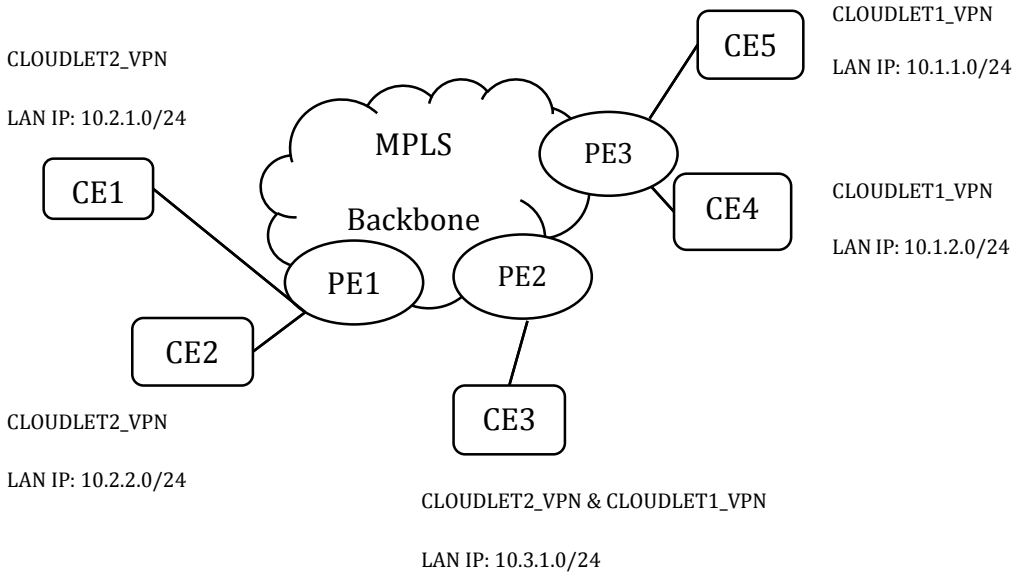


Fig. 3.9: VPN connection for cloudlets

Accessing a host remotely over the public network needs administrative permission and a dedicated software that allows the remote user to access the host. In this research, one of the objectives is to migrate a VM between two cloudlets over the WAN. To meet this purpose, each cloudlet must trust each other and there must

be an administrative permission in each hosts to allow a VM migration from one cloudlet to the other.In order to create a secure networking collaboration between different sites of cloudlets, VPN is the most appropriate approach. Adoption of Multi-protocol label switching (MPLS) based VPN is proposed to meet the requirements.

3.4.2 Algorithm to form a VPN between Cloudlets

Cloudlets may be connected directly to the customer end (CE) routers or may be connected after one or multiple routers from the CE. Involving networking devices in between cloudlets and provider edge (PE) routers increases the RTT latency between different sites of cloudlets. Since the objective in this research is to minimize RTT latency, we assume that cloudlets are capable of performing the functionality of a CE. In other words, cloudlets are connected directly to the PEs. Hence, cloudlet and CE are used interchangeably afterwards in the context of VPNs.

A VPN is formed based on the geographical location of the CEs. If CEs are close enough for a walking distance, the CEs would form a VPN. From Fig. 3.9, CLOUDLET1_VPN and CLOUDLET2_VPN are established when a proximity cloudlet servers are found. CE1, and CE2 belongs to CLOUDLET2_VPN and CE5, and CE4 belongs to CLOUDLET1_VPN. CE3 is a member of both CLOUDLET1_VPN and CLOUDLET2_VPN. The LAN IP address for each CEs in the same VPN has to be different.

A cloudlet will be added to a VPN if and only if it is near to at least one cloudlet from the VPN. The maximum distance a cloudlet can have with at least one cloudlet

from a VPN is equal to the distance calculated in equation 3.5. The network coverage of the wireless network in meters plus 936 meters (equation 3.6) is the maximum distance one cloudlet should have with at least one cloudlet from a VPN.

Add a Cloudlet to a VPN

The algorithm to add a cloudlet in a VPN is defined below. D_{max} is the maximum distance between two cloudlets to make seamless handover. The existed cloudlet VPNs are represented by VPN_c. C stands for new cloudlet to be added to a VPN. D_{ck} is the distance between cloudlet C and cloudlet K.

Algorithm Add a cloudlet to VPN

Input: D_{max}, VPN_c, C

Output: VPNs where C is a member

Step 1: For each VPN, V Ɛ VPN_c

Step2: For each cloudlet K Ɛ V

Step 3: Add C to V, If $D_{ck} <= D_{max}$

Step 4: Update V

Step 5: End

The algorithm finds VPNs that a new cloudlet can be added. It checks the distance (D_{ck}) between the new cloudlet (C) with each other cloudlet (K) from each

existing cloudlet VPN (VPNc). If the distance is less than or equal to D_{max}, the cloudlet will be added to the VPN where the neighbour cloudlet is found and it updates the member of that VPN.

Once the VPNs are created, a new cloudlet candidate can be added to the existing VPNs. Once the IP address space is assigned to VPN sites, merging a VPN will cause IP address overlaps or IP address change to each sites. If a cloudlet candidate is in between two VPNs, it would be a member of both VPNs but merging would not be performed. CE3 from Fig. 3.9 can be an example.

Removing cloudlet from a VPN will cause complication, if the cloudlet being removed is a neighbour for cloudlets far apart with a distance longer than D_{max}. One VPN may be divided in two VPNs because of removal of one cloudlet.

3.4.3 Location Identifier

MPLS VPN allows identifying the location of the MDs with their IP address. Each site has different LAN IP address (see Fig.ure 3.9) so as the MDs based on their location. Packets coming to a cloudlet are filtered based on the source IP address. When a cloudlet is accessed from different site, it is initiated to perform the VM migration. For the following two main reasons VM migration is triggered when mobile IP address is changed:

1 A cloudlet can be accessed with different LAN IP address if there is a cloudlet in the new location or if it is from a cellular network. After the original IP address of the MD is changed, the first cloudlet could not give the optimal

performance anymore since WAN RTT latency is involved between the MD and the first cloudlet.

2 Since cloudlets are assumed to be owned by local businesses, the MD would benefit only if the user remains in the particular business area. For example, a coffee shop would allow users to access the cloudlet if users remain in the shop. Although users will not be disconnected as soon as they move out from the business area, there would be a time limit to allow the VM instance to be migrated. After all, the VM instance consumed a resource in a cloudlet and it has to be freed for the use of another customer.

3.4.4 Possible Neighbour Database

Each cloudlet maintains a possible neighbour database which helps to make decision on the VM migration. When a new cloudlet joins a VPN, the new cloudlet multicasts its presence to all VPN members. The other cloudlets receive and check the RTT with the new cloudlet. If RTT is in acceptable range, the cloudlet is added to the possible neighbour database with the current RTT value. Since RTT is dependent on the real-time network condition, each cloudlet compute the RTT with their neighbour cloudlets. Every minute, the RTT value is computed and the database is updated. The following are the two main reasons for making the RTT an important parameters in forming the neighbour database.

Table 3.2: RTT latency compared with subjective impression [MaBa2009]

Response time	Subjective Impression
< 150ms	Crisp
150ms – 1s	Noticeable to Annoying
1s – 2s	Annoying
2s -5s	Unacceptable
> 5s	Unusable

The first reason is the QoE. After the MD user changes location, the MD will access the previous cloudlet through the WAN. If the RTT between the previous and the current cloudlets is unacceptable there is no need to provide the service. If the RTT is unacceptable, a new VM will be synthesised in the new location. Table 3.2 shows the subjective impression of a user as the response time increases. Neglecting the RTT latency difference between network RTT and response time, the response time is considered as network RTT. This thesis considers VM migration between cloudlets only if the RTT between the cloudlets is less than 150msec.

The second reason is that RTT determines the TCP throughput. Ideally the maximum TCP throughput achievable is:

$$\text{TCP Throughput} = \frac{\text{Receiver window size}}{\text{RTT}} \qquad (3.7)$$

In the lab experiment, Iperf [IPERF] application is used to measure the TCP throughput between two cloudlets as RTT increases. By default Iperf application sets

the maximum server window size to 83.5KByte. The result is shown below in Fig. 3.10.

The throughput of TCP is inversely related with the VM migration time. As RTT increases between the cloudlets, the duration of VM migration would be unacceptable.

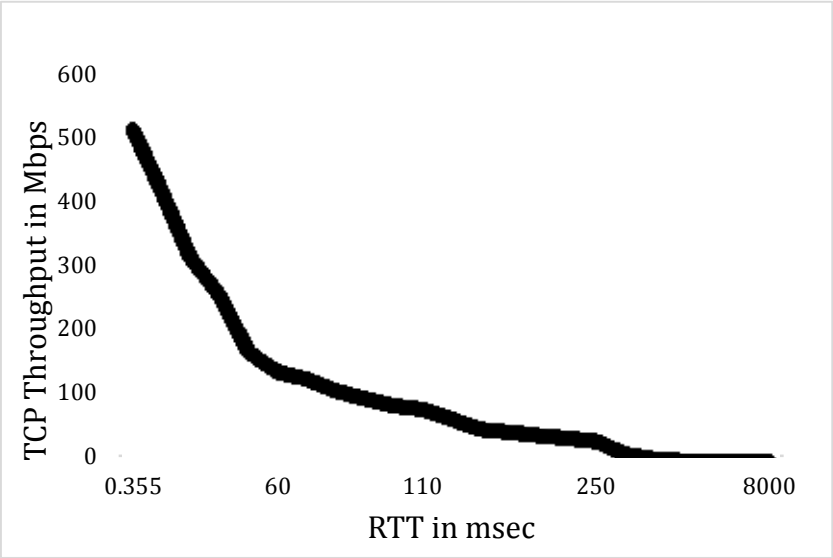


Fig. 3.10 TCP throughput vs RTT

The VM migration decision is based on the neighbour database. The cloudlet IP address tells the network address. When a cloudlet receives a packet from remote LAN, it checks whether the network address matches any of the cloudlets in the possible neighbour database. The VM migration starts if and only if the cloudlet is found in the neighbour database.

Chapter 4 Experiment and Performance Results

This chapter investigates the possibility of live VM migration with IP address change where MPTCP feature is used to support the network connection migration. The experiment setup is presented in Section 4.1. Performance metrics are given in Section 4.2. The assumptions for the cloudlets networking is described in Section 4.3. Section 4.4 tested the baseline performance of the experiment environment. Experiment results in Section 4.5 proves the feasibility of MPTCP for seamless live VM migration. Section 4.6 further investigates the performance of the system with different network resources. Based on the experiment results, Section 4.7 proposes a VM migration decision maker algorithm. Lastly, the limitation of the experiment is presented in Section 4.8.

4.1 Experiment Environment

The experiments have been conducted using 16GB RAM, core i7 Linux 3.11.10+ (Ubuntu 12.04 LTS) host. The cloudlet servers are emulated using VMs inside the Linux host. The VM monitor (or hypervisor) used is KVM with QEMU emulator which uses a pre-copy RAM state migration mechanism [KVMQEMU].

In total, four VMs inside the Linux host are used to set up the experiment environment (see Fig. 4.1). The two VMs, VM1 and VM2, represent cloudlet 1 (C1) and cloudlet 2 (C2), respectively. The nested VM (VM3) inside cloudlet 1 denotes the server VM instance used by the MD, and the fourth VM (VM4) represents the MD. The specifications used for the VMs are listed in Table 4.1. The nested VM instance is similar to "small" VM instance on Amazon EC2.

Table 4.1: VM specifications inside the Linux host

VM	Virtual CPU (vCPU)	RAM Size	Virtual Disk	Virtual Interface	Operating System
Cloudlets (VM1 & VM2)	4	8GB	40GB	2	Linux 3.11.10+ (64 bit)/ Ubuntu 12.04 LTS
MD (VM4)	2	4GB	40GB	2	Linux 3.11.10+(64 bit)/ Ubuntu 12.04 LTS
Nested VM (VM3)	2	2GB	10GB	2	Linux 3.11.10.squeezemptcp (32 bit) / Debian 7.3.0

The RAM size of VM4 is less than the RAM size of the MD. The QEMU emulator used in this experiment doesn't allow to have more than 2GB RAM as a nested VM inside the 8GB RAM VM. Since the intention of the experiment is to create a server client connection between the MD and the VM instance, the difference of the RAM size does not create any problem.

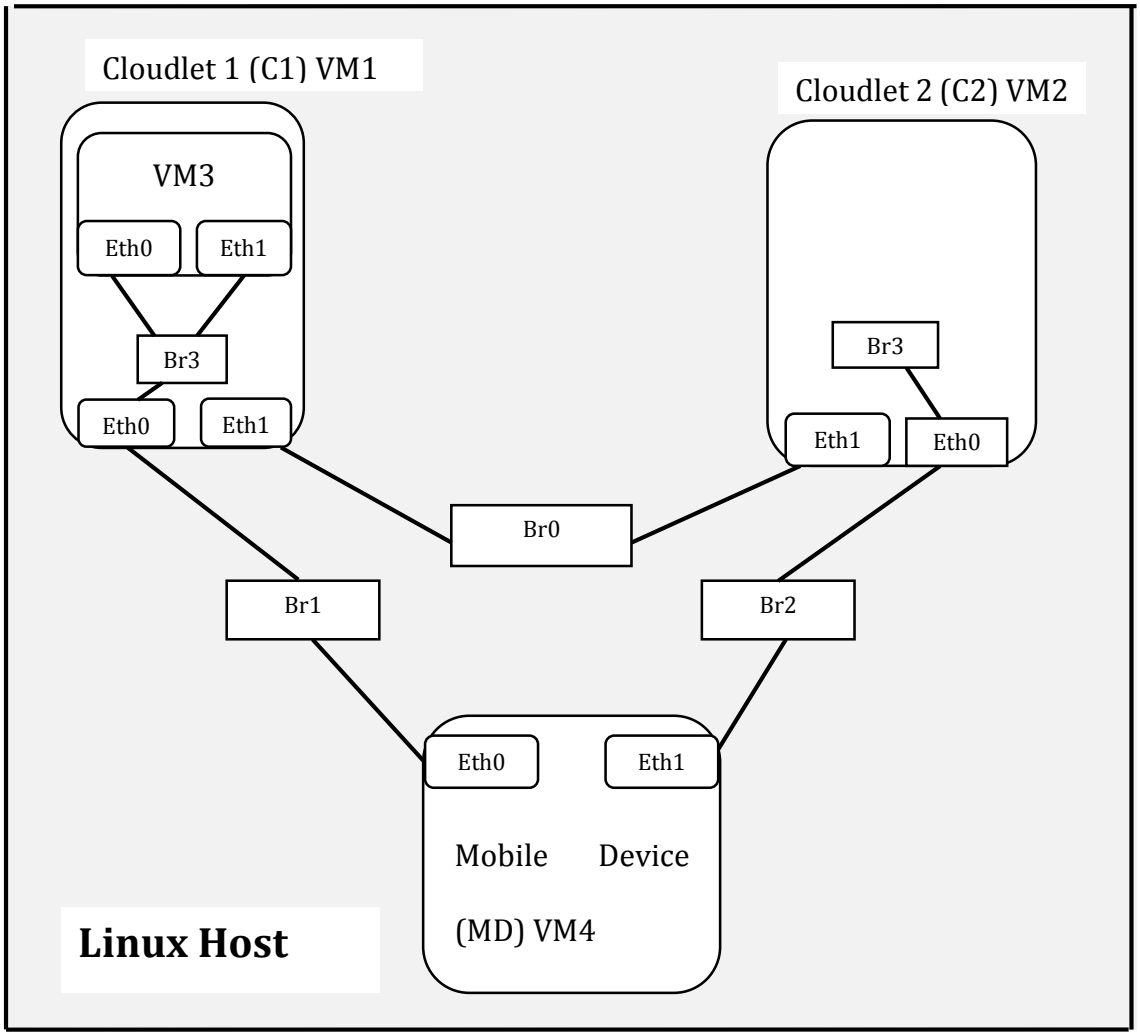


Fig. 4.1: Experiment setup using VMs and bridged networking inside a Linux host

Both wireless and wired transmission medium technologies are characterized by a VM bridged networking. The VMs networking in Fig. 4.1 emulates the cloudlets and the MD networking shown in Fig. 3.4.

In Fig. 4.1, VM3 inside VM1 stands for the VM instance server used by the MD. VM4's Eth0 and Eth1 are used to emulate a single Wi-Fi interface of a MD. VM4 accesses VM3 through one interface at a time (Eth0 or Eth1). The movement of the MD is emulated by configuring VM4's interfaces to UP or DOWN state.

For instance, if the MD is assumed to be in the communication range of C1 initially, only VM4's Eth0 will be at the ON state. The movement of the MD from C1 to C2 is imitated by configuring first VM4's Eth0 to the DOWN state, then Eth1 to the UP state. Since a Wi-Fi interface can be connected to one AP at a time, the above mentioned method is suitable to represent a Wi-Fi interface for this research objective.

MPTCP protocol is installed in all the VMs inside the Linux hosts in Fig.4.1. A static routing protocol is used to forward traffic in between the VMs. The LAN IP address connected to Br1 and Br2 of the Linux host is 10.1.1.0/24 and 10.1.2.0/24, respectively. Br0 of the Linux host associates the two cloudlets with 134.117.64.0/24 public IP address.

The static routing on the VM4 changes automatically as the MD changes location. The following example explains the scenario. Assume that the MPTCP connection is established with the VM4's Eth0 interface and the VM3's Eth0 interface initially. The static routing forwards traffic as follows:

Eth0 of the VM4 -> Br1 of the Linux Host -> Br3 of VM1 -> Eth0 of VM3.

After some time the VM4's Eth0 interface is taken down and then the Eth1 interface is turned on to imitate the movement of the MD. This situation changes the static routing as follows:

> Eth1 of the VM4 -> Br2 of the Linux host -> Br3 of VM2 -> Br0 of the Linux host through Eth1 of VM2 -> Br3 of VM1 through Eth1 of VM1 -> Eth0 of VM3.

High BW availability and low RTT latency between the MD and the cloudlet through Br1 and Br2 emulates a LAN. By varying the available BW and by introducing RTT latency between the two cloudlets, Br0 network emulates the WAN. Emulating WAN is achieved using the traffic control (tc) command found in Linux distributions [TC]. Using this application a queuing delay can be introduced in outgoing packets from the host. Token Bucket Filter (tbf) queuing discipline provided by Linux is used to set the upload BW of an interface.

4.2 Performance Metrics and Measurement Mechanisms

QoE can be affected by many system factors. This thesis adopts four of the most important performance metrics that influence the QoE for the system deployed.

1. **Throughput:** Changing location from C1 to C2 exposed the MD to low throughput due to the WAN bottleneck. If VM migration and the MD traffic share the same path, the throughput of the MD drops for the duration of the VM migration. Returning to the previous high throughput for the MD is the performance measurement for the system.

Iperf [IPERF] tool is used to measure throughput between the VM and the MD.

2. **RTT latency**: As in the throughput, the movement of the MD introduces a high RTT latency between the VM instance and the MD. Minimizing the RTT latency after the live VM migration is taken to be one of the performance metrics.

   The *ping* command is used to measure the network RTT with 1 sec interval.

3. **Total VM migration time:** The VM is migrated if the MD changes location. The duration the MD accesses the VM instance with high RTT latency and low throughput through WAN has to be as short as possible. This duration is determined by the VM migration process. The longer the VM migration time, the poorer the QoE.

   The experiment collects the total VM migration time from the command line execution time. The VM migration is initiated using a command utility, the time before and after the migration command completes is considered as the total VM migration time.

4. **VM down time**: The VM downtime is part of the total VM migration time. A downtime is the time the VM is not accessible. The RAM state migration mechanisms, (both post-copy and pre-copy, see Section 2.3.1), have a VM downtime. Unacceptable VM downtime may be experienced based on the network condition between the source and destination hosts.

Network layer unavailability during VM migration is also investigated using the *ping* command. During migration, the VM is pinged with the original IP address and also with the expected next IP address at the same time. The timestamps with *ping* responses are saved in a file. Later on, the VM downtime is calculated using the difference between the two timestamps, the timestamp the VM at the source host stops replying and the timestamp the VM at the destination host starts replying to the *ping* requests. In other words, the VM downtime is the time the VM from the source is paused to the time the VM from the destination host starts running. The downtime of the VM is measured with an interval of 0.01sec (10msec). It is possible to minimize the interval to 1msec but flooding of packets to a VM may influence the performance. For this reason, the VM downtime is known if only it is greater than 10msec. Downtime definitely happens during VM migration process, but it can be recorded only if it is greater than 10msec. Otherwise the down time is expressed as less than 10msec.

4.3 Cloudlets Networking Assumptions

This thesis assumes that cloudlets are connected to a high-speed Internet connection. The availability of high BW is considered throughout the experiment.

To further enhance the performance of the cloudlets system, low RTT latency between cloudlets in a vicinity is assumed. Before we proceed to the experiment, the

causes for high end-to-end delay are explained in the following subsection. The indications that supports the assumption for low RTT latency are also explained.

4.3.1 End-to-end Delay

End-to-end delay is the accumulation of the processing and queuing delay in routers, propagation delay, and the end host processing delay.

1. **Queuing delay:** if the packets arrival rate exceeds the transmission rate, packets waits in a buffer to be transmitted. The time from the packets arrival to the time the packets are transmitted refers to as a queuing delay.
2. **Propagation delay:** The nature of physical medium (i.e., multimode fiber, twisted-pair copper wire, wireless, etc.) determines the propagation speed. The propagation delay is the distance between two routers divided by the propagation speed.
3. **Transmission delay:** The transmission delay is the amount of time required for the router to completely push out the packet. Transmission delay is a function of the packet's length in bits divided by the transmission rate of the link.
4. **Processing delay:** The time required to examine the packet's header and determine where to forward the packet is part of the processing delay.

The following three situations support the assumption for low RTT latency between geographically nearby cloudlets.

1. **Geographical proximity:** Geographical proximity between hosts minimizes propagation delay since propagation delay is inversely proportional to the physical length of a link.
2. **Minimum Hops:** As explained in Section 3.4.2, cloudlets are assumed to be capable of performing as a CE; which makes the cloudlets to be found one hop away from the PE router. The possibility of cloudlets in a vicinity aggregating to one PE is high. In such cases, the total number of hops between the cloudlets become two, i.e., CE1-PE1-CE2 (see Fig. 3.9). This minimizes the total queuing and processing delay in each router.
3. **High data transmission rate:** High-speed Internet connection to the cloudlet is also one assumption made in this research. High-speed Internet services found from Bell Canada and Rogers Internet Company are taken as references. The *Bell Fibe Internet 175* service with download and upload speed of 175Mbps from Bell Canada and *Ultimate Fiber* service with download and upload speed of 350Mbps from Rogers are the high speed Internet connection products found in the market by the time this research is completed. The upload and download speeds are the maximum available BW but not the actual available BW at a given time. However, high data transmission rate guarantees for low transmission delay.

How real the above assumptions are in the real world is a question that needs to be evaluated. In order to assess the feasibility of low RTT latency and high throughput assumption, public servers which provide the Iperf application are used

to measure the network layer RTT and the maximum throughput. By running Iperf client on a host in our department, the results were collected. The locations of the Iperf servers and the results gathered from the Iperf client are presented in Table 4.2.

Table 4.2: RTT and throughput result from public Iperf servers

No.	Location	Public Iperf Servers	RTT In msec	TCP Throughput In Mbps
1	Russian Federation / Saint Petersburg City	iperf.eltel.net	143	12
2	USA / California State	iperf.scottlinux.com	71	68.8
3	Belgium / Brabant Wallon City	multipath-tcp.org	119	4.46
4	Russian Federation / Saratov City	iperf.saratov.ertelecom.ru	175	19
5	Russian Federation / Nizhegoroo City	st2.nn.ertelecom.ru	209.477	20.3

All the above servers were found more than 15 hops away from the host on which the experiment was tested. From the above results, it is fair to assume the network RTT between the cloudlets can be less than 150msec. The experiment in the following sections investigate the system performance when RTT vary from 20msec to 150msec.

The other reason to consider the maximum RTT latency to be 150msec is the QoE explained in Section 3.4.4. A cloudlet will be added to a possible neighbour database if the RTT is less than 150msec.

4.4 Baseline Performance

The baseline performance is the performance of the experiment setup shown in Fig. 4.1 without introducing any additional RTT latency and without throttling the BW between VM1 and VM2. The average measured network RTT latency between the VMs and the maximum throughput achieved for the original experiment setup is shown in Table 4.2. Iperf application is used to measure the throughput and the *ping* command is used to measure the average RTT. The receiver window size is set to the default TCP window size which is 64KByte.

Table 4.3: Original RTT and throughput measurement

	RTT	**Maximum Throughput** (Receiver window size = 64KByte)
Between VM1 and VM2	0.382msec	627Mbps
Between VM3 and VM4	0.392msec	160Mbps

Without any application running on VM1, VM2, and VM3, migrating VM3 from VM1 to VM2 takes 10.67sec and 243.46msec VM downtime on average. The total VM migration time and the VM downtime gives similar result with other research [WeAn2013].

In order to measure VM3 migration effect on the throughput performance of other active applications, VM1 launches Iperf server and a client Iperf application runs on VM2. The Iperf client is the one that floods packets to the Iperf server. The Iperf server acknowledges the received packets then it drops the packets. For this scenario, the upload BW of VM2 is used to upload data while the download BW is used

for only TCP acknowledgements. On the other hand, VM3 migration takes place from VM1 to VM2. In this case, VM1's upload BW is used to migrate VM3 while the download BW is used only for TCP acknowledgements. Although the direction of the VM migration and Iperf application data flooding is opposite, the acknowledgement signalling affects the performance. The baseline performance while another active application is running between VM1 and VM2 gives a total VM3 migration time and VM3 downtime of 11.16sec and 244.8msec, respectively.

Fig. 4.2 shows the effect of VM3 migration on the performance of throughput between VM1 and VM2. The period T in Fig. 4.2 represents the total VM migration time. During period T, the result illustrates how the maximum throughput fluctuates by dropping once the VM migration is started.

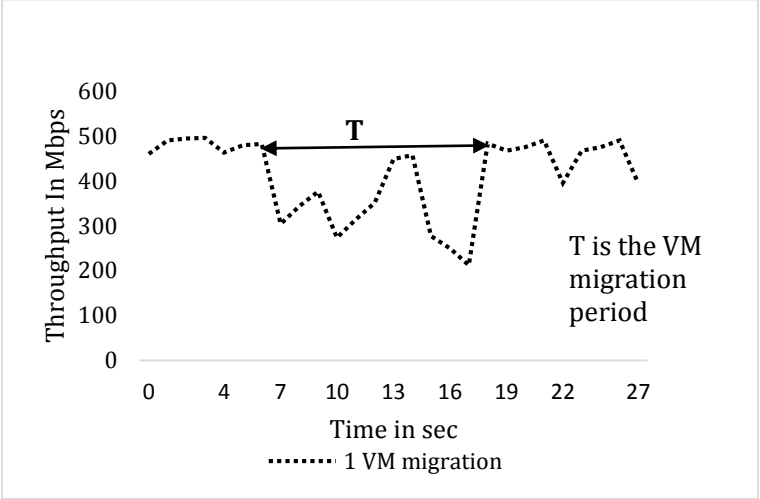


Fig. 4.2: Throughput performance between VM1 and VM2 when VM3 is migrated from VM1 to VM2

4.5 Live VM Migration with MPTCP and Performance Results

The experiments that are described in this Section are intended to demonstrate a seamless live VM migration using MPTCP from the connection migration perspective. The primary focus is to show how TCP connection remains open after both VM3 and VM4, as depicted in Fig. 4.1, change their original IP addresses.

As it is presented in Section 3.2.1.1, the prior knowledge of the IP address to be assigned for the migrating VM in the destination host enables a live server VM migration seamlessly with the MPTCP protocol. The connection migration is achieved with the change of the VM IP address. In addition, the results from the following experiment also proves the seamless handover of the MD from C1 to C2 network.

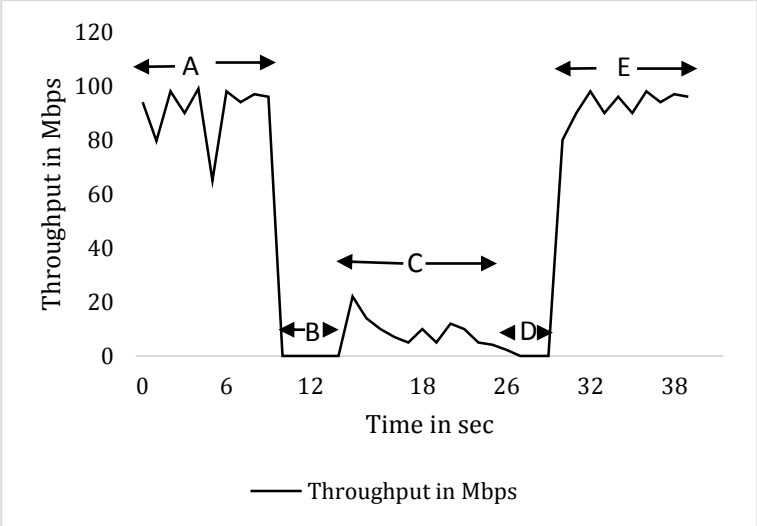


Fig. 4.3: Throughput performance between VM3 and VM4 before and after migration of VM3 from VM1 to VM2

Fig. 4.3 illustrates the maximum throughput obtained between VM3 and VM4. The results are gathered from VM4, with Iperf server and Iperf client running on VM3 and VM4, respectively. Table 4.4 shows the configured values of the maximum available BW and the RTT between the VMs for this particular experiment.

Table 4.4: BW and RTT setup between VMs

	Maximum available BW in Mbps	**Network RTT in msec**
Between VM1 and VM2 through Br0	350	20
Between VM4 and VM1 through Br1	100	2
Between VM4 and VM2 through Br2	100	2

Table 4.5: Initial interfaces and IP addresses for VM3 and VM4

VM	**Interface**	**Interface state**	**IP address**
VM3	Eth0	UP	10.1.1.15/24
	Eth1	DOWN	NONE
VM4	Eth0	UP	10.1.1.14/24
	Eth1	DOWN	NONE

In Fig. 4.3, the time is slotted in to A, B, C, D, and E. The following paragraphs explain each time slot and the connection during the time slots. An example is used to illustrate the steps of migration and the duration of each time slot based on measurements. More detailed performance results are presented in Section 4.6.

Time slot A is the time where VM4 accesses VM3 through VM1. During this time slot, VM3 resides inside VM1 and VM4 is directly connected to VM1. The MPTCP

connection is established with the UP state interfaces and IP addresses given in Table 4.5. The traffic between VM4 and VM3 flows as:

Eth0 of the VM4 -> Br1 of the Linux Host -> Br3 of VM1 -> Eth0 of VM3.

Table 4.6: Interfaces and IP addresses after VM4 changes the initial interface

VM	Interface	Interface state	IP address
VM3	Eth0	UP	10.1.1.15/24
	Eth1	DOWN	NONE
VM4	Eth0	DOWN	NONE
	Eth1	UP	10.1.2.18/24

After some time, the Eth0 interface of VM4 is taken down. Time slot B denotes the time where both the Eth0 and the Eth1 interfaces of VM4 are down (5 sec). This is to imitate the time a MD user walks from one Wi-Fi AP to another.

After 5sec downtime, the Eth1 interface of VM4 is changed to the UP state. During this time, VM4 is connected back to VM3 through VM2. The existing MPTCP connection adds a new subflow with the UP state interfaces, and IP addresses as presented in Table 4.6. As we can see from Fig. 4.3 time slot C, the application keeps running even after VM4 IP address is changed. The traffic between VM4 and VM3 flows as:

Eth1 of the VM4 -> Br2 of the Linux host -> Br3 of VM2 -> Br0 of the Linux host through Eth1 of VM2 -> Br3 of VM1 through Eth1 of VM1 -> Eth0 of VM3.

During time slot C, because the communication between VM3 and VM4 goes through the WAN, the throughput drops down.

As soon as VM3 is accessed with the IP address of 10.1.2.18/24, VM3 recognizes that VM4 has changed its original location (interface), which is the beginning of time slot C. The above phenomena (MD moving to new location or VM4 changing the initial IP address) triggers the migration of VM3 from VM1 to VM2 in order to minimize the RTT latency and maximize the throughput between VM3 and VM4 by avoiding the traffic flow through the WAN.

Table 4.7: IP addresses and state of interfaces for VM3 and VM4 before VM3 is migrated to VM2.

VM	Interface	Interface state	IP address
VM3	Eth0	UP	10.1.1.15/24
	Eth1	UP(inactive)	10.1.2.19/24
VM4	Eth0	DOWN	NONE
	Eth1	UP	10.1.2.18/24

As the approach mentioned in Section 3.2.1.1, the Eth1 interface of VM3 will be changed to the UP state and configured with the next odd IP address (i.e., 10.1.2.19/24) of the current VM4 IP address (10.1.2.18/24). Table 4.7 shows the state of the interfaces and the IP addresses assigned at this situation. Even if the Eth1 interface of VM3 is inactive for this time slot, VM3 advertises 10.1.2.19/24 IP address to VM4 as an additional IP address. The MPTCP connection from VM4 tries to add new subflow with the Eth1 interface of VM3, but the subflow will be created once VM3 migration is completed.

Table 4.8: IP addresses and state of interfaces for VM3 and VM4 after VM3 is totally migrated to VM2.

VM	Interface	Interface state	IP address
VM3	Eth0	DOWN	NONE
	Eth1	UP	10.1.2.19/24
VM4	Eth0	DOWN	NONE
	Eth1	UP	10.1.2.18/24

Time slot C represents part of VM3 migration time through the WAN. However, VM3 migration process is completed after time slot D. Time slot D starts from the time VM3 is inaccessible at the application level to the time VM3 become available in VM2. The total VM migration time (C+D) is 14.00 sec. VM3 application level downtime (D) is 3 sec, whereas the network layer downtime is only 204.3msec for this particular experiment.

Time slot E shows the time after VM3 migration is completed. The MPTCP connection adds a new subflow with the UP state interfaces and IP addresses shown in Table 4.8. At this time, the traffic between VM4 and VM3 follows as:

Eth1 of the VM4 -> Br2 of the Linux Host -> Br3 of VM2 -> Eth1 of VM3.

Fig. 4.4 shows how the high throughput between VM4 and VM3 is achieved after VM3 migration is completed. The objective of the VM migration to the nearest cloudlet is to keep the MD's throughput to the highest possible value.

Time slots B, C, and D affect the performance of the entire system. Time slot B is determined by the geographical location of the cloudlets. If the network coverage of

the cloudlets overlaps, minimum time slot B can be achieved. Time slots C and D depend on the VM migration process. The shorter these time slots are, the better the QoE.

Section 4.6 further investigates the effect of RTT latency and maximum available BW between VM1 and VM2 for time slots C and D.

4.6 Performance Analysis of VM Migration

This Section further investigates the performance of VM migration for different values of the maximum available BW and network RTT time between VM1 and VM2. The results presented here were based on the KVM hypervisor. Different hypervisors may generate different results from the one presented in this thesis. Without any application running on the migrating VM, KVM is known for its shorter VM downtime than that of Xen Server, Hyper V and VMware [WeAn2013].

4.6.1 Total VM Migration Time and VM Downtime

This experiment measures the total VM migration time (C+D) and VM network layer downtime using the approach mentioned in Section 4.2. The objective is to investigate the total VM migration time and the VM downtime when the network RTT and BW are varied between VM1 and VM2. The network RTT values used for the experiments are 20msec, 50mesc, 100msec, and 150msec and the BW values used for the experiments are 350Mbps, 175Mbps, and 60Mbps.

Table 4.9: Total VM migration time in seconds.

	RTT=20ms	**RTT=50ms**	**RTT=100ms**	**RTT=150ms**
BW=350Mbps	14.00	16.62	29.44	42.02
BW=175Mbps	19.93	20.55	35.27	48.77
BW=60Mbps	43.74	46.79	48.35	50.13

Table 4.10: Network layer VM downtime in milliseconds.

	RTT=20ms	**RTT=50ms**	**RTT=100ms**	**RTT=150ms**
BW=350Mbps	204.53	860.81	1878.25	2811.18
BW=175Mbps	709.30	1273.41	2367.87	3564.01
BW=60Mbps	2483.76	3011.99	3547.25	3687.74

Results in terms of the total VM migration time in seconds and network layer VM downtime in milliseconds are shown in Table 4.9 and Table 4.10, respectively. Fig. 4.4 and Fig. 4.5 demonstrate the pictorial results.

The results show that the total VM migration time and the VM downtime increase as the RTT latency increases with a fixed available BW between VM1 and VM2. It can also be observed that as the available BW decreases, the migration time and VM downtime increases. The RTT effect is more noticeable for high BW, e.g., 175Mbps and 350Mbps than the low BW, e.g., 60Mbps. The following points investigate the reasons for the results shown in Table 4.9, Table 4.10, Fig. 4.4, and Fig. 4.5.

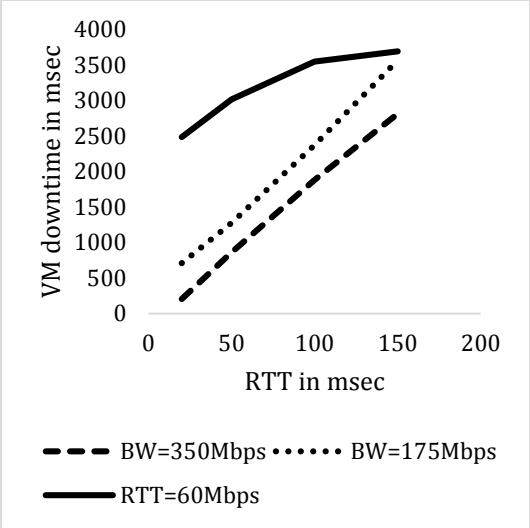


Fig. 4.4: Network layer VM downtime vs RTT

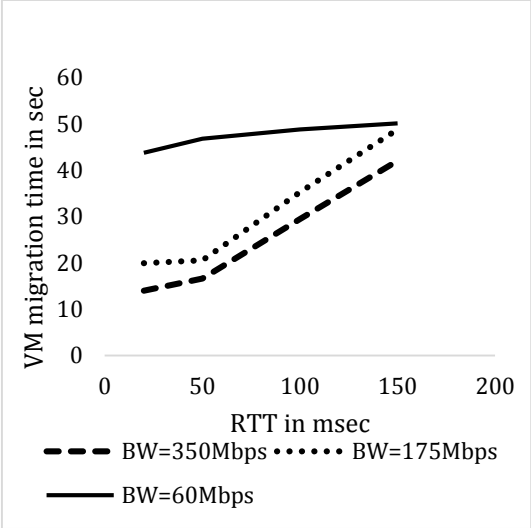


Fig. 4.5: VM migration time vs RTT

- The total amount of memory configured for VM3 is 2GByte while the used amount of the memory is only 263MByte. The hypervisor used for this experiment is KVM. KVM only transfers the used memory and it has a way to abstract the free memory for the migrating VM at the destination host. Recall from the baseline performance result, migrating more than 263MByte takes only 10.67 seconds. The estimated data transfer speed of KVM will be the used memory divided by the migration time which is equal to (263MByte × 8 bits/Byte / 10.67 sec) 197.19Mbps. Note that 197.19Mbps is the estimated speed, not the exact value. To calculate the exact data transfer speed we need to know how many iteration KVM goes through and the data transferred in each iteration. During the baseline experiment, no application was running on

the migrating VM; hence the estimated data transfer speed is close to the actual data transmission speed of KVM.

If the available BW is less than 197.19Mbps, queuing delay will be introduced in addition to the RTT values between VM1 and VM2. As the total delay increases between VM1 and VM2, the data transmission speed decreases which subsequently increases the total VM migration time.

- The throughput for VM3 migration does not stay the same during the migration process. The TCP auto tuning was turned on, both on VM1 and VM2 and also the TCP stack on VM1 and VM2 was configured with minimum of 4KByte, maximum of 5394KByte, and 85KByte default receiver window size. When VM3 is migrated from VM1 to VM2, VM2 increases or decreases its window size to control the data flow between VM1 and VM2.

The VM migration is performed over the TCP protocol. The efficiency of TCP is determined by the BW delay product (BDP). If the BDP is greater than the receiver window size, TCP data transmission will not be efficient. To identify the scenarios which do not use the available BW efficiently, the average receiver window size has to be known. In order to identify the estimated average receiver window size, the following calculation is performed.

Table 4.11: The BDP results in KByte.

	20msec	**50msec**	**100msec**	**150msec**
350Mbps	854	2734	4272	6409
175Mbps	427	1068	2136	3204
60Mbps	146	366	732	1098

Table 4.11 shows the BDP results in KByte for the corresponding RTT and BW values. From the BDP results, when BW is 350Mbps and RTT is 150msec, the data transmission is not efficient even when the receiver window size is in its maximum value, i.e. 6409Kbyte ((350Mbps × 150msec)/ (8 bits/Byte × 1024)) which is greater than 5394Kbyte (maximum system configured window size). For this particular case the data transmission speed is governed by the receiver window size and the RTT instead of the available BW. If BDP is greater than the receiver window size, the ideal maximum TCP throughput can be calculated as:

$$\text{Maximum TCP throughput} = \frac{\text{Receiver Window Size}}{\text{RTT}} \qquad (4.1)$$

From equation 4.1, the average receiver window size can be evaluated. From Table 4.9, the total VM3 migration time is 42.02 sec. and the estimated TCP data transmission speed can be calculated as:

$$\text{Estimated data transmission speed} = \frac{\text{VM3 Used Memory}}{\text{VM3 total Migration time}} \qquad (4.2)$$

Equation 4.2 gives the estimated data transmission speed as 50.07Mbps when BW is 350Mbps and RTT is 150msec. Substituting 50.07Mbps ((50.07Mbps/8bits/Byte) = 6.26MBps) in equation 4.1, the average receiver window size will be (6.26MBps × 150msec) 916KByte.

In Table 4.11, with the average receiver window size of 916KByte, the data transmission for the corresponding scenario of the shaded values are not efficiently using the available BW. During these scenarios, the data transmission is determined by the RTT and receiver window size rather than the available BW. This is also the reason for reflecting similar total VM migration time when RTT 150msec regardless of the available BW values (see Fig. 4.4).

Section 4.6.2 shows all the results in terms of RTT latency and throughput performance.

4.6.2 Throughput and RTT latency

In the previous subsection, we have seen how the total VM migration time and VM downtime is affected by the available BW and the RTT latency between the source and the destination cloudlets. In addition to the total VM migration time and the VM downtime, the QoS provided to the user during this period is also very crucial.

This subsection investigates how the RTT and the throughput of the MD are affected during the VM migration time when the values for the available BW and the RTT vary between the source and destination cloudlets.

1. RTT latency

The experiments show that during the VM migration process the RTT latency between VM3 and VM4 increases from its original value.

The RTT values initially used in this experiment are 20msec, 50mesc, 100msec, and 150msec and the maximum available BW used in this experiment are 350Mbps and 60Mbps.

Fig. 4.6 to Fig. 4.9 show the actual RTT during VM migration for initial RTT of 20msec, 50msec, 100mesec and 150msec respectively. The RTT was measured by using the *ping* command with a 1 second interval. The objective is to show how much the RTT latency increases when the VM starts to migrate.

The figures, Fig. 4.6 to Fig. 4.9 show the results from the time when VM4 starts to access VM3 through the WAN. At the start, the low RTT latency shows the RTT between VM4 and VM3 before VM3 migration process is started. The VM migration is started manually to observe the effect of the VM migration. The time durations noted as T1, T2, T3, T4, T5, T6, T7 and T8 in the figures show the total VM migration time for the correspondence scenario.

Before analysing the results shown from Fig. 4.6 to Fig. 4.11, some terms are defined below for clarity.

Initial RTT: The average RTT between the migrating VM and the MD through the WAN before the migration process is started. For this experiment, it would be the

average RTT between VM3 and VM4 through Br0 before VM3 starts to migrate from VM1 to VM2.

Actual average RTT: is the average RTT between VM3 and VM4 during VM3 migration process from VM1 to VM2.

Delta RTT: is the difference between the actual average RTT and the initial RTT.

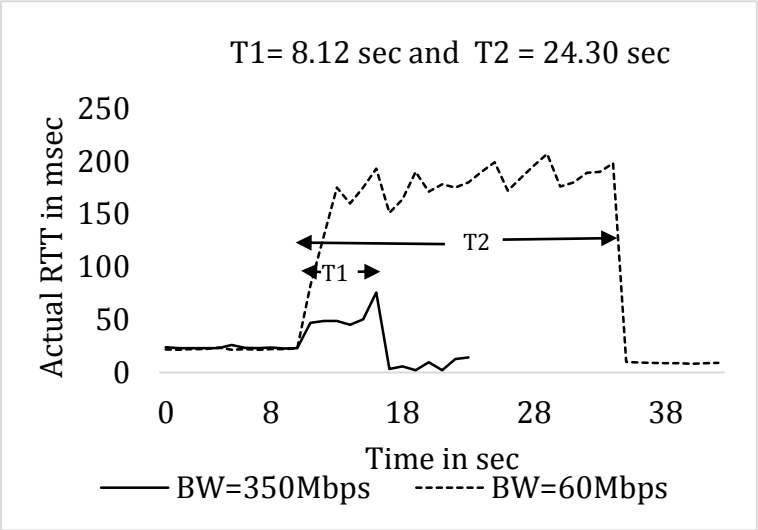


Fig. 4.6: Actual RTT latency during VM migration for initial RTT=20msec.

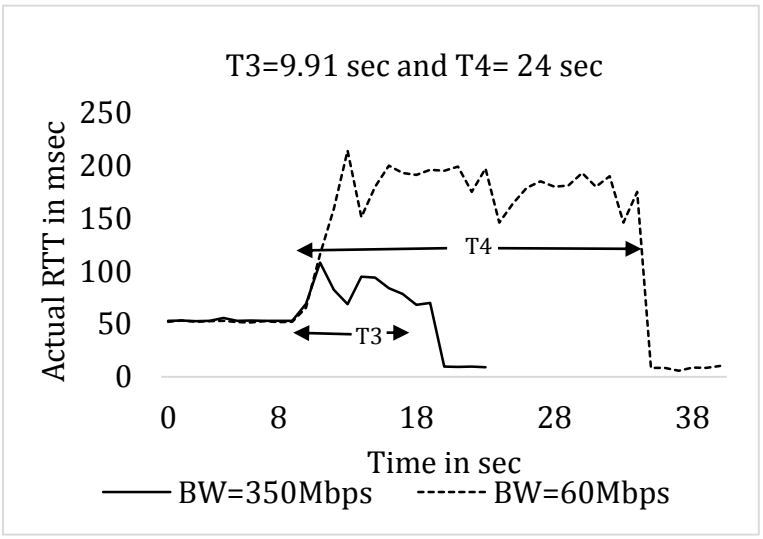


Fig. 4.7: Actual RTT latency during VM migration for initial RTT=50msec.

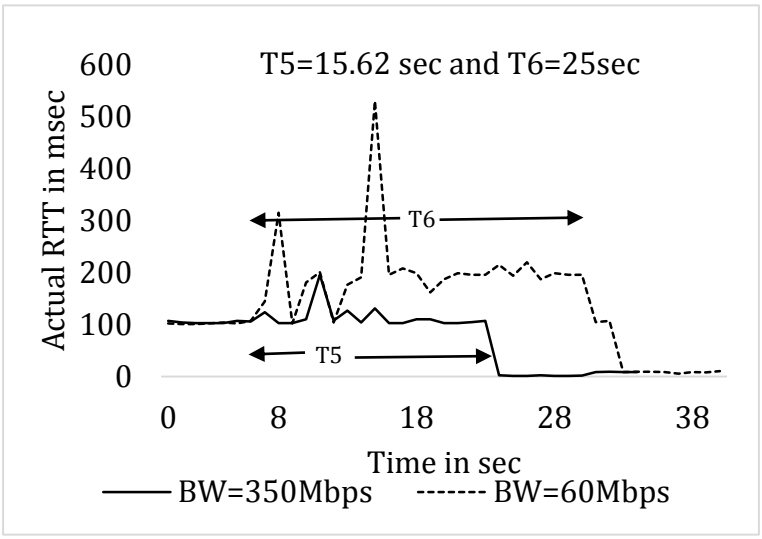


Fig. 4.8: Actual RTT latency during VM migration for initial RTT=100msec.

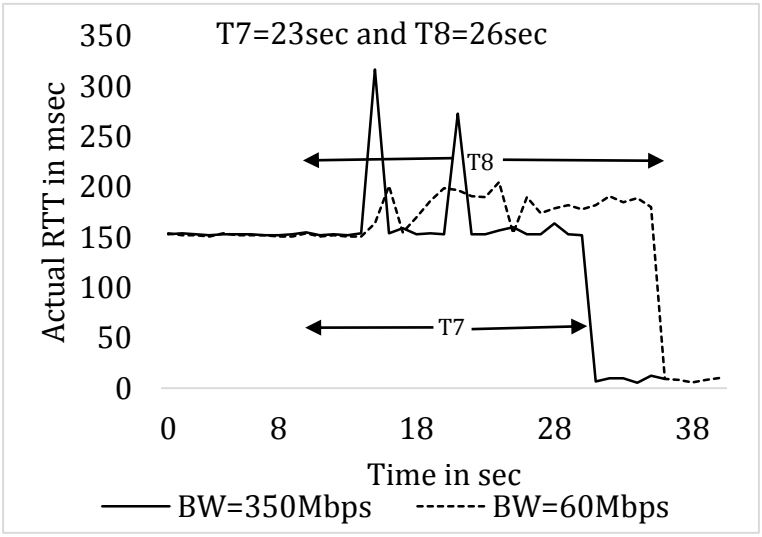


Fig. 4.9: Actual RTT latency during VM migration for initial RTT=150msec.

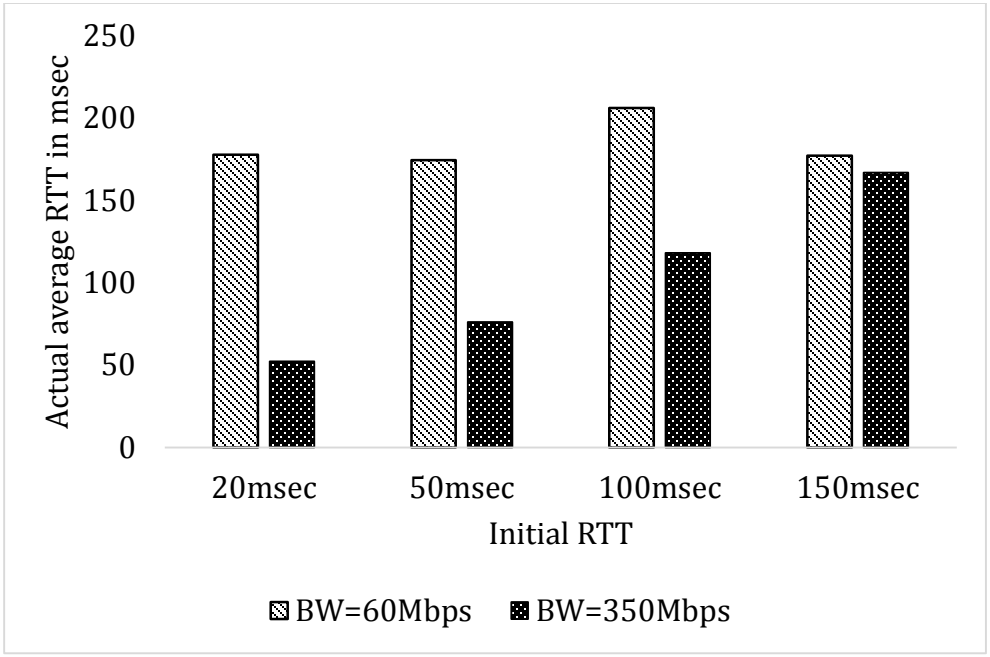


Fig. 4.10: Average RTT latency during migration process vs initial RTT

From Fig. 4.6 to Fig. 4.9, it is clearly shown that the value difference between the initial RTT and the RTT during the VM migration. Fig. 4.10 summarizes the results in terms of the actual average RTT. For a fixed RTT latency, minimizing the available BW increases the VM migration time also the actual average RTT significantly. The significance is more noticeable for a low RTT latency value. For low BW, e.g., 60Mbps, the actual average RTT is independent on the initial RTT. Regardless of the initial RTT, the actual average RTT reflects the same amount. For a high BW, e.g., 350 Mbps, the actual average RTT increases as the initial RTT increases. However, the amount of the RTT added to the initial RTT is not the same.

Fig. 4.11 illustrates the delta RTT as the initial RTT increases for the BW of 60Mbps and 350 Mbps. Delta RTT decreases as the initial RTT increases regardless of the available BW.

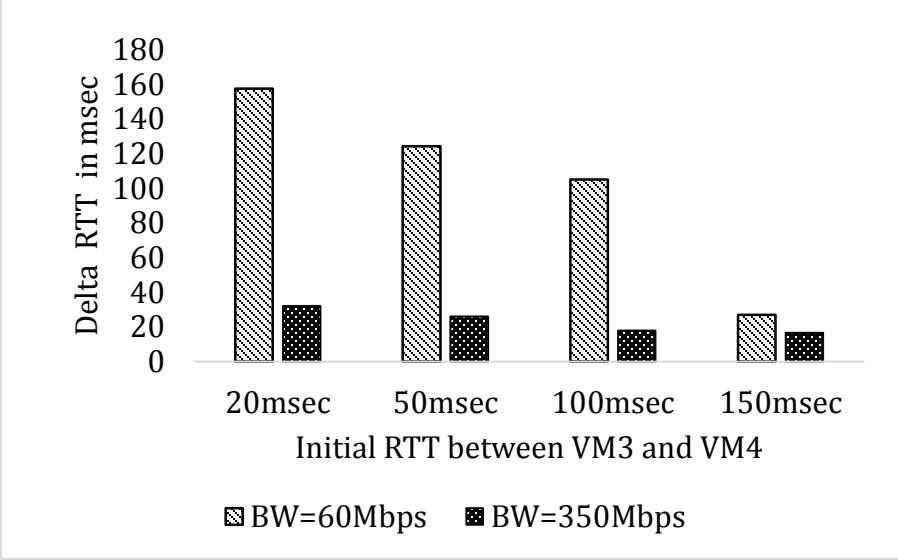


Fig. 4.11: Delta RTT vs the initial RTT

Therefore, having low initial RTT before VM migration does not guarantee for low RTT during migration. From the results, it is concluded that as long as there is a high available BW, the actual average RTT does not show much difference from the initial RTT.

1. Throughput

One of the main advantages of using a nearby cloudlet to offload application is for the opportunity of achieving a high throughput. The throughput that is hard to achieve through WAN can be achieved from a nearby server.

The result shown in Fig. 4.12 is the throughput achieved between VM3 and VM4. The figure shows the throughput only after VM4 is directly connected to VM2, i.e., VM4 and VM3 are communicating through the WAN.

For the sake of clear observation, VM3 migration from VM1 to VM2 is started manually. The time before the 10th second is the time when VM4 is connected to VM3 through the WAN. At the 10th second, the migration process is started. The completion of the VM migration for each RTT values can be observed when a high throughput is achieved.

As it can be clearly observed, the throughput of VM4 drops down after the 10th second. Specially for the 20msec RTT the throughput decreases significantly. For low initial RTT values, delta RTT is more noticeable as shown in Fig. 4.11. As the RTT increases significantly, the throughput decreases accordingly.

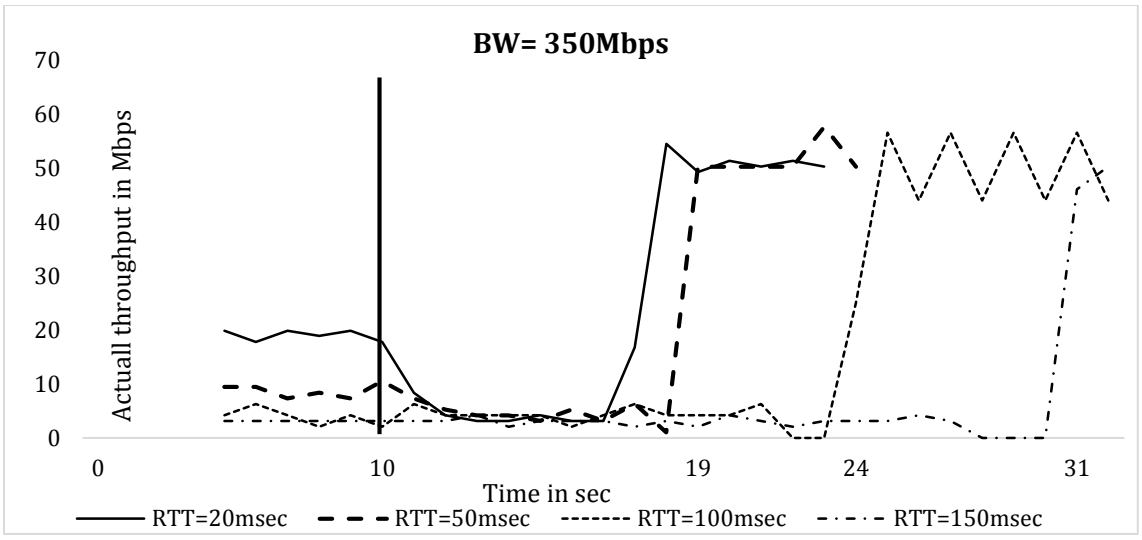


Fig. 4.12: Actual throughput vs time during VM3 migration for BW=350Mbps

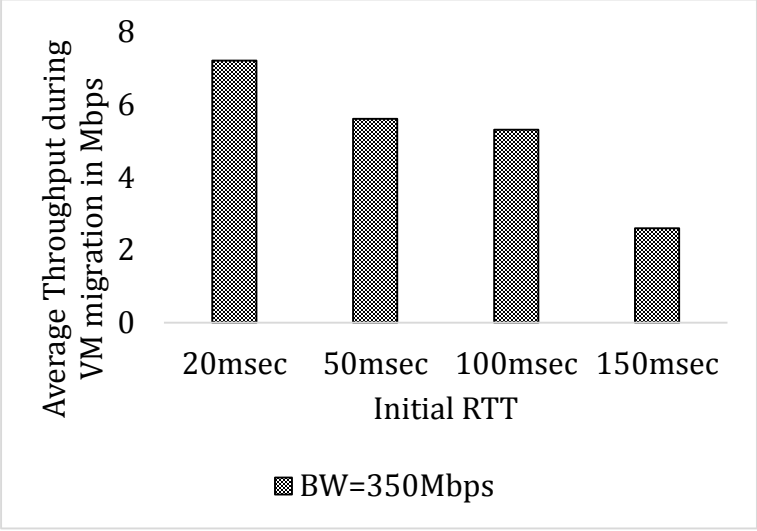


Fig. 4.13: Average throughput during VM3 migration

From Fig. 4.13, it is observed that a five times increase in the initial RTT, decreases the average throughput only by 26.4 % (initial RTT increase from 20msec to 100msec and the throughput decreases from 7.2Mbps to 5.3 Mbps).

However, 20msec initial RTT shows a better performance than all the others RTT values in terms of VM migration time (See Fig. 4.6 T1=8.12sec).

4.7 VM Migration Decision Algorithm

From the experimental results, VM migration with only the bare hypervisor is impossible to guarantee fast VM migration over the WAN. Optimization techniques for VM migration over WAN mentioned in other research efforts, such as [TiRa2011], have to be adopted for VM migration to make the system more efficient.

In addition to the optimization techniques, a smart VM migration decision maker is appropriate for a better QoE. As the main objective is eliminating the service re-initiation time for the MD, a better service has to be provided during the migration process. All the performance metrics need to be considered before the VM starts to migrate. If the VM migration is predicted to be worse performance than service re-initiation time, it is better for the MD to launch a new VM at the new location.

The type of applications running on the VM also determines the VM migration time. In this thesis, only Iperf application runs on the migrating VM. Iperf is a type of send and receive application which does not consume much of the CPU (consumes maximum of 15% of VM3 CPU). A VM migration decision algorithm needs to consider the type of the application running on the VM in terms of resource (RAM, CPU, hard disc, and network) consumption and usage. For instance, RAM write intensive applications generate dirty memory pages frequently. A pre-copy migration works best only when memory pages can be copied to the destination host faster than they

are dirtied. For instance, KVM will never finish live migration if the RAM page is dirtied with a speed of 32MBps [WeAn2013].

Therefore, to be beneficial from the live VM migration, a decision algorithm plays an important role in determining if the live VM migration should proceed. The decision algorithm needs to consider the main factors presented in this chapter, e.g., RTT latency and available BW. The proposed VM migration decision algorithm is presented as follows:

Service Initiation Time: – the time the MD discovers the nearby cloudlet to the time data offloading starts.

The algorithm is as follows:

1. Estimate VM migration time based on the three factors:
    a. Size of the VM
    b. Application running on the VM
    c. The network resources (RTT and available BW between the source and destination hosts)
2. Estimate Delta RTT.
3. If the initial RTT + Delta RTT <=150msec, migrate the VM.
4. If 150msec < Initial RTT + delta RTT <= 1sec and if the estimated VM migration time is less than the service initiation time, migrate the VM.
5. Otherwise, ignore VM migration and notify user to start a new service. Destroy the VM instance.

The algorithm considers three main parameters in making the decision on the VM migration: the service initiation time, the VM migration time and the RTT during VM migration. The relation between the QoE and the RTT is considered from Table 3.2.

If the RTT during migration is less than 150msec, it means that the user is having a reasonably good QoE even through the WAN. Regardless of the total estimated VM migration time, the VM migration can be performed.

The VM migration is triggered only if the initial RTT is less than or equal to 150msec. From the previous experimental results, we have seen that the RTT during migration is higher than the initial RTT. Step 4 of the decision making algorithm makes sure that if the RTT during migration is less than or equal to 1sec. If RTT is between 150msec and 1 sec the user can notice the RTT latency. The user can benefit from the VM migration only if the migration time is less that the service initiation time.

There is no need to trigger VM migration, if the actual RTT during VM migration is greater than 1 sec since QoE becomes low for the user.

The decision making algorithm should be implemented in each cloudlet. The estimation of the total VM migration time and delta RTT can be achieved through profiling. The estimation starts with prediction based on resources and the resource consumption. Applications and network resources profiler need to be deployed in to the system which keeps tracking of all VM migration results. The profiling results can be stored in a performance knowledge base [MuGr2007]. Through time, the profiler

provides reasonably accurate information. Each cloudlet also can share their profiles to learn a wide varieties of situations within short time.

4.8 Limitation of the Experiment

The experiment does not consider storage migration. VM1 and VM2, in Fig. 4.1, shared a storage through network file sharing (NFS) which is mounted in the Linux host. In real world, shared storage through WAN is not an effective way of implementation. A hard disc read and write intensive applications suffer from the RTT latency between the CPU location and the storage location.

The storage migration needs an optimization technique. Storage migration is influenced by the network resources (BW and RTT) only if the throughput between the source and the destination cloudlets is less than the reading speed of the hard disc. For example, DRDB reading speed is 4MBps. In order to migrate a 10GB storage, it takes 41.67 minutes. If the available BW is less than the reading speed, it will take longer time to migrate.

Migrating storage as the VM is triggered to be migrated would not give a feasible scenario. This thesis considers other optimization efforts to make the whole live VM migration possible. Two optimization techniques are described briefly here. The first one is duplicating and updating the storage between neighbour cloudlets before the VM migration is triggered. This method has been used in [TiRa2011]. The second one is a fully distributed architecture for a single, shared, synchronized and scale-out POSIX file system which is proposed by [HiYo2014].

Chapter 5 Conclusions and Future Work

Live VM migration as a means of providing a minimum delay service between a client and a server has been studied in this thesis. The idea is to move a server VM closer to the client as the client changes location. In this research, the client is considered to be a MD and the server is a VM instance launched in a cloudlet.

Live VM migration over the WAN includes migration of the RAM state, network, and storage. This thesis focused on the network migration with the help of MPTCP protocol. The MPTCP protocol has proved its capability for seamless MD handover from a Wi-Fi to a 3G/4G network [ChGr2012] and seamless live VM migration [CaCh2013]. Both the above situation were possible because the MD and the VM are running a client application. There is a difference between migrating a VM running a server application and migrating a VM running a client application.

In this thesis, a prior knowledge of what the IP address will be assigned for the migrating VM at the destination cloudlets makes a seamless live server VM migration possible with the help of MPTCP protocol. In addition to the prior knowledge of the IP address, two virtual interfaces are proposed for the migrating VM. One interface to operate during normal operation and the other interface to operate when a VM migration is triggered. Configuring the VM with the destination network information before the VM is migrated helps to minimize the downtime caused by configuring the VM after the VM has moved. Unlike the mobile IP approach proposed in [ToRe2014],

the approach used in this thesis adds no additional time to the total VM migration time and the downtime than the ordinary live VM migration.

The experiment environment setup shows how a VM migration can be experimented without having multiple physical hosts. The baseline performance of experiment setup has demonstrated similar results in terms of total VM migration time and VM downtime with other research that has been performed on different physical hosts [WeAn2013].

The performance of live VM migration has been studied for different values of available BW and RTT between the source and the destination cloudlets. The total live VM migration time, the VM downtime, the actual throughput and the RTT during migration have been considered as performance metrics. As the RTT increases with fixed available BW, the total VM migration time and VM downtime increase significantly, especially for high available BWs.

Even if a low initial RTT value between the source and the destination cloudlets is crucial to consider before the VM migration, the results from the experiment showed that having low initial RTT between the cloudlets does not guarantee for low RTT during the VM migration process between the MD and the migrating VM. The actual RTT during migration depends on both the available BW and the initial RTT.

5.1 Future Work

To make the live VM migration more beneficial, an algorithm is provided in Section 4.7. Implementing the algorithm is interesting topic worth further research in the future. Through more experiments, learning how to estimate the delta RTT and total VM migration time would also be helpful for other areas of live VM migration.

The other interesting topic is tracking the MD location to predict the next possible cloudlet. The benefit of knowing the next possible cloudlet is to start the VM migration before the MD is totally disconnected from the current cloudlet. If the VM migration is started earlier, the duration the MD access the VM instance through the WAN will be minimized or can be eliminated.

References

[BySu2011] Byung-Gon C.,Sunghwan I., Petros M., Mayur N., Ashwin P., "CloneCloud: Elastic Execution between Mobile Devices and Cloud", Proceedings of the 6th Conference on Computer Systems, 2011, pp 301-314.

[CaCh2013] Catalin N., Christoph P., Marcelo B., Costin R., "Evolving the Internet with Connection Acrobatics", Proceedings of the Workshop on Hot Topics in Middleboxes and Network Function Virtualization, 2013, pp 7-12.

[ChGr2012] Christoph P., Gregory D., Fabien D., Costin R., Olivier B.," Exploring Mobile/WiFi Handover with Multipath TCP", Proceeding of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design, 2012, pp 31-36.

[ChKe2005] Christopher C., Keir F., Steven H., Jacob G. H. Eric J., Christian L., Ian P., Andrew W. " Live Migration of Virtual Machines" Proceeding of the 2nd Conference on Symposium on Networked Systems Design & Implementation, Volume: 2, 2005, pp 273-286.

[DRDB] DRDB, http://www.drbd.org. Last accessed on Aug 09, 2014.

[EdAr2010] Eduardo C., Aruna B., Dae-ki C.,Alec W. ,Stefan S. ,Ranveer C.,Paramvir B., "Mobile Assistance Using Infrastructure (MAUI)," , Proceeding of the 8th International Conference on Mobile Systems, Applications, and Services, 2010, pp 49-62.

[FaMe2013] D. Farinacci, D. Meyer, D. Lewis "Locator/ID Separation Protocol (LISP)" IETF RFC 6830, 2013.

[HiYo2014] Hiroki K., Yoshiaki K., Tohru K., Tomohiko K., Yoshirou O., Ikuo N., Shunji A.,Shigetoshi Y.,Shinji S., Shimojo, S., "A Proposal and Evaluations

of a Distributed Storage System for a Widely Distributed Virtualization Infrastructure," IPSJ Journal, Volume: 55, No. 3, 2014,pp. 1140-1150.

[IPERF] Iperf, https://iperf.fr. Last accessed on Aug 09, 2014.

[Ja2012] Jason F. "Cyber Foraging: Bridging Mobile and Cloud Computing via Opportunistic Offload". Morgan & Claypool Publishers, 2012.

[JaTa2013] Jararweh, Y.; Tawalbeh, L.; Ababneh, F.; Dosari, F. "Resource Efficient Mobile Computing Using Cloudlet Infrastructure", Proceeding of the 9th International Conference on Mobile Ad-hoc and Sensor Networks (MSN), 2013, pp 373-377.

[JiKa2013] Jiwei L., Kai B., Xuan L., Bin X., "ENDA: Embracing Network Inconsistency for Dynamic Application Offloading in MCC", Proceedings of the 2nd ACM SIGCOMM Workshop on MCC, 2013, pp 39-44.

[JoTh2004] JOHN N. L., THOMAS P. K. "The Continuing Evolution of Pedestrian Walking Speed Assumptions", ITE Journal, Volume 74, 2004, pp 32-40.

[KiPa2013] Kiryong H., Padmanabhan P., Wolfgang R., Yoshihisa A., Mahadev S., "Just-in-time Provisioning for Cyber Foraging", Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services , 2013, pp 153-166.

[KoDe2014] Kommineni, S.; De A.; Alladi S.; Chilukuri S., "The Cloudlet with a Silver Lining", Proceeding of the 6th International Conference on Communication Systems and Networks, 2014, pp 1-4.

[KVMQEMU] Kernel-based Virtual Machine, www.linux-kvm.org. Last accessed on Aug 09, 2014.

[MaBa2009] Mahadev S.; Bahl, P.; Caceres, R.; Davies, N., "The Case for VM-Based Cloudlets in Mobile Computing", Pervasive Computing, IEEE Volume: 8, Issue: 4, 2009, pp 14-23.

[MiUm2009] Michael R. H., Umesh D. , Kartik G., “Post-copy Live Migration of Virtual Machines”, SIGOPS Operating Systems Review, Volume: 43, Issue: 3, 2009, pp 14-26.

[MPTCP] Multipath TCP, www.multipath-tcp.org. Last accessed on Aug 09, 2014.

[MuGr2007] Murray W., Greg F., Dorina C. P., "The Future of Software Performance Engineering", Proceedings of the 29th International Conference on Software Engineering; 2007, pp. 171-187.

[SoMu2012] Soyata, T.; Muraleedharan R.; Funai C.; Minseok K.; Heinzelman, W., “Cloud-Vision: Real-time Face Recognition using a Mobile-Cloudlet-Cloud Acceleration Architecture”, Proceeding of the Symposium on Computers and Communications (ISCC), 2012, pp 59-66.

[TC] Traffic control, lartc.org/manpages/tc.txt. Last accessed on Aug 09, 2014.

[TiRa2011] Timothy W., K. K. Ramakrishnan, Prashant S., Jacobus van der M., “CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines”, Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, Volume: 46, Issue: 7, 2011, pp 121-132.

[ToRe2014] Tohru K., Reiji A., Kazufumi S., Kaori M., “A Mobility Management System for the Global Live Migration of Virtual Machine across Multiple Sites”, Proceeding of the 38th Annual International Computers, Software and Applications Conference, 2014, pp 73-77.

[WeAn2013] Wenjin H., Andrew H., Long Z., Eli M. D., Vinay S., Hao J., Ronny B., Jeanna N. M.” A Quantitative Study of Virtual Machine Live Migration “, Proceedings of Cloud and Autonomic Computing Conference, 2013.