Teka *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2016) 5:12
DOI 10.1186/s13677-016-0061-0

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Nearby live virtual machine migration using cloudlets and multipath TCP

Fikirte Teka, Chung-Horng Lung* and Samuel A. Ajila

Abstract

A nearby virtual machine (VM) based cloudlet is proposed for mobile cloud computing (MCC) to enhance the performance of real-time resource-intensive mobile applications. Generally, when a mobile device (MD) discovers a cloudlet in the vicinity, it takes time to set up a VM inside the cloudlet before data offloading from the MD to the VM starts. The time between the discovery of the cloudlet and actual offloading of data is considered as the service initiation time. When multiple cloudlets are present in a nearby geographical location, initiating a service with each cloudlet may be frustrating for cloudlet users that moving from one location to another. In order to eliminate the delay caused by the service initiation time after moving away from the source cloudlet, this paper proposes a seamless live VM migration between neighbouring cloudlets. A seamless live VM migration is achieved with the prior knowledge of the migrating VM IP address in the destination cloudlet and more importantly with multipath TCP (MPTCP). We have performed a number of experiments to validate the proposed approach using Linux KVM hypervisor. The experimental results demonstrate the feasibility of the proposed approach and also show performance improvement. Specifically, there is almost zero downtime at the destination cloudlet after the migration is completed.

Keyword: Cloudlets, Mobile cloud computing, Multipath TCP, Virtual machine migration, Fog computing

Abbreviations: AP, Access Point; DHCP, Dynamic Host Configuration Protocol; HoA, Home Address; LAN, Local Area Network; MCC, Mobile Cloud Computing; MD, Mobile Device; MoA, Mobile Address; MPTCP, Multipath TCP; QoE, Quality of Experience; RTO, Retransmission Time Out; TCP, Transport Control Protocol; VM, Virtual Machine; VPN, Virtual Private Network; WAN, Wide Area Network

Introduction

The high demand for mobile applications has encouraged software and mobile developers to bring the desktop level applications to mobile devices (MDs). Although the computation and storage capacities and battery life time of MDs have improved considerably in recent years, mobile devices still remain resource-poor devices because of battery power, bandwidth, and capacity to handle resource-hungry applications. Most of the applications executing on MDs are real-time and interactive applications. In addition to offloading, a real-time application to a distant remote cloud through the Internet increases response time due to Wide Area Network (WAN) latency [1]. Latency affects MD users in two ways. Firstly is the quality of experience (QoE). As latency increases, the QoE degrades. Secondly is the faster drain of energy on MDs [2, 3]. Energy efficiency is also an important performance parameter for MDs. Therefore, maximizing the benefit of offloading applications for MDs can be achieved by minimizing the latency between the MDs and the servers. In order to do this, researchers have proposed resource-rich nearby servers which are connected to or integrated with wireless access points (AP) as a solution for MDs offloading. Examples include CLONECLOUD [4], MOCHA [5], MAUI [2], Odessa [6], COMET [7], and virtual machine (VM) based cloudlet [1]. In this way, the need for improving QoE and energy efficiency can be accomplished by low latency, one-hop, high bandwidth (BW) wireless access to the servers.

There are two types of offloading mechanisms in the literature. One is by partitioning applications and sending only resource-intensive instructions to servers and executing the rest of instructions on the MD itself. The

* Correspondence: chlung@sce.carleton.ca
Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario K1S 5B6, Canada

second mechanism is to send all the instructions to the server. The second mechanism needs to create a VM instance on the server for each MD to serve as shared infrastructure and to provide strong isolation between computations from different MDs. VM-based cloudlets [1] adopted the second approach.

The authors in [1] define a cloudlet as "a trusted, resource-rich computer or cluster of computers that's well-connected to the Internet and available for use by nearby MDs." Cloudlets are a disruptive technique in mobile computing. Cloudlets can provide low delay, high bandwidth access to high-end computing devices that are within one wireless hop, which can bring substantial value to new emerging applications [8, 9]. The concept of cloudlets is consistent with the recent concept of Fog computing [10–12] which focuses on the edge for mobile users so that mobile users can access computation-intensive services via short-distance and low-delay local connections. Further, cloudlets can also be considered as "second-class data centers" which are much more flexible and easier to maintain or replace compared to the traditional data centers [9].

This paper focuses on the VM-based cloudlets. 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 one is the change of MD IP address, which will terminate the existing transport control protocol (TCP) connection between the source cloudlet and the MD.
- The second problem is that MD needs to wait for another service initiation time before the offloading starts, if there is available nearby cloudlet in the new location. Service initiation time is the time from discovering the cloudlet to the time the offloading starts. Service initiation time has negative impact on QoE.

To address these two problems, we initially proposed [13] to replace TCP with MPTCP [14] for the first problem. Further, our proposed solution to the second problem is to perform a VM migration from the source cloudlet to the destination cloudlet to eliminate or reduce the extra service initiation time.

This paper is an extension of [13]. The main contributions of the paper include: First, the paper demonstrates in detail the capability of MPTCP to support live server VM migration over a wide area network (WAN). Without any code modification to MPTCP, a server VM can be migrated live seamlessly without interrupting the application running on the VM. Second, this paper proposes two interfaces to be used by the VM and to configure the destination IP address information before the VM is migrated. This approach has two advantages: seamless connection migration and zero network VM downtime after the migration is completed. This paper also discusses location identifier, possible neighbor database to facilitate migration decision making, and various scenarios. We also have added an algorithm to a Virtual Private Network (VPN) and an algorithm for migration decision making.

The rest of the paper is organized as follows; Section II describes the related works. Section III discusses our proposed VM migration for cloudlets using MPTCP. Section IV describes our experiments and demonstrates the results. Section V discusses decision making for VM migration. Finally, Section VI is the conclusions and future directions.

Background and related work

Various technical areas are related to this paper. This section introduces the basic concepts of those related areas, specifically cloudlets, live VM migration, MPTCP and Fog computing.

Computational offloading and cloudlets

Mobile Cloud Computing (MCC) brings together cloud computing, mobile computing, wireless networks, and cloud services to provide MD users rich computational resources. MCC overcomes the resource limitation of wireless MDs by leveraging fixed infrastructure. Resources, such as CPU, RAM, data storage and battery energy, are limited resources in MDs. Resource intensive applications, such as speech recognition, natural language processing, computer vision and graphics, machine learning, augmented reality, planning, and decision-making become common in MDs. In order to enhance the performance of these resource-intensive applications, offloading from the MDs to resource-rich servers in the vicinity or to a distant server is a common solution for mobile computing.

Research efforts [1, 2, 4] have shown that offloading applications to a *remote server* is not always the optimal solution for MDs. The 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 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 latency between the cloudlet and the MD.

There are basically two types of offloading mechanism. 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 sever. The partitioning scheme has to adapt to the network condition and the availability of resources in real-time basis. For example, if a smartphone can access a remote server with a good wireless network connection, resource-intensive tasks can be executed remotely. If the energy cost to send the tasks remotely is higher due to poor wireless connection, the tasks will be executed locally on the smartphone. The following tools: CLONECLOUD [4], MOCHA [1], MAUI [2], Odessa [7], and COMET [7] adopt this approach.

The second offloading mechanism is to send the whole application to a VM instance identified 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. The authors in [1] proposed a VM-based cloudlet offloading approach.

There are two different methods to send the VM state to a cloudlet infrastructure. One is VM migration, in which an already executing VM is suspended, its processor, disk and memory states are transferred; and finally VM execution is resumed at the destination from the same point of suspension. The other approach is called dynamic VM synthesis which customizes a VM on demand. VM customization is time consuming, as the process includes the time to create VM disc space, install operating system and install a particular application. Delivering a service after all of the steps taken causes a great deal of delay. Minimizing VM customization time is the main objective of dynamic VM synthesis.

Fog computing

Fog computing, first proposed by Cisco [12], shares similar concept as cloudlets, as the emphasis is on the nearby cloudlets or edges. Fog computing extends the conventional cloud computing paradigm to the edge of the network that is close to the end users. Fog is also known as micro datacenter. Fog computing has received tremendous attention lately as the technology has the potential to bring the benefits of cloud computing closer to the users to leverage energy-rich, high-end computing, and low-latency.

Authors in [10–12] presented various scenarios for Fog computing. For instance, connected vehicles, smart grid, wireless sensor and actuator networks can benefit considerably from low-delay response using Fog computing [10]. Other potential scenarios include Internet of Things (IoT) and cyber-physical systems [11]. Fog computing can also facilitate augmented reality and real-time video analytics and mobile big data analytics that require intensive computations [12].

Fog computing is still in the early stage in terms of research and practical deployment. On the other hand, Fog computing addresses not only the technical aspects, but also the business concerns that exist for cloudlets. To effectively support Fog computing, a basic infrastructure for edge computing devices is essential and needs to be planned and managed by a network operator. The managed infrastructure can mitigate some of the issues that may occur for VM migration. For instance, the overlapping of cloudlets, and issues of handovers and IP address assignment can be effectively managed with pre-configured policies. Section III describes a simple pre-established network configuration policy that can help manage IP addresses for VM migration.

Live VM migration and network migration

VM migration, especially live VM migration, as mentioned in Section II.A, is useful for delay sensitive applications for MCC. Live VM migration, a key technology in virtualization, is the process of moving a running VM from one physical host to another with minimal downtime. For live VM migration, the service initiation time is a critical factor that causes delay. Service initiation time is the time from when the MD discovers a cloudlet to the time the MD starts offloading data. Four main steps are involved for the service initiation time: bind the MD with the cloudlet, transfer the VM overlay, decompress the VM overlay, and apply the VM overlay to the base VM to launch the VM. To minimize the service initiation time, Ha, et al. [15] applied four optimization techniques (Deduplication, Bridging the Semantic Gap, Pipelining, and Early Start) to minimize the delay significantly.

Live VM migration can be performed within a data center from one server to another or it can be performed across geographically distributed data centers. There are three key aspects to be considered in a live VM migration [16]: RAM state, storage and network migration. This paper focuses on the network migration aspect.

Network migration involves LAN migration and/or WAN migration. In LAN migration, the VM can retain its IP and MAC addresses after migration. The local switch needs to adjust the mapping for the VM's MAC address to its new switch port [17]. 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, e.g., CloudNet [18]. This allows open network connections to be seamlessly redirected to the VM's new location.

Mobile IP is another approach that has been used for connection redirection [18] 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). Every time the VM changes its location, the VM updates an agent of the new location. Traffic for the VM always goes through the HoA, and then HoA will redirect traffic to MoA. This causes additional delay for the traffic.

Multipath TCP (MPTCP)

To mitigate the VM migration delay and improve the robustness, another mechanism is to adopt MPTCP. MPTCP is a transport layer as opposed to a network layer and is a viable solution for live VM migration. The main idea is that even if the original IP address of a VM has changed, the connections remain established [19]. MPTCP is implemented for migration of a VM running a client application. Unlike the regular TCP/IP protocol, MPTCP is an IETF (Internet Engineering Task Force) protocol that allows *multiple interfaces* to be used *for a single application* and socket interface [14]. MPTCP is responsible for connection setup, transferring data between TCP connections, adding and removing subflows, and tearing down the session for one or more TCP connections. In MPTCP, a path is a sequence of links between a sender and a receiver and it is defined by 4-tuple source and destination address pair; a subflow is the same as TCP segments operating over an individual path; and an MPTCP connection is a one-to-one mapping between a connection and an application socket.

Other related work

Other approaches to cloudlet migration have been discussed in the literature. Li, et al. [20] discussed the lack of consistent network performance for mobile user while offloading; the optimal offloading decision becomes suboptimal due to MD user movement. The authors proposed a three-tier (Smartphone-cloudlet-cloud) architecture that tracks user locations using GPS and saves it in a centralized database found in a remote cloud. The location information helps to predict user's movement and to identify the energy efficient Wi-Fi AP. The real-time network performance between a smartphone and an AP and the server-side load are considered to make optimized offloading decisions. The approach considered multiple cloudlets and proposed a centralized cloudlet system which is different from the decentralized VM based cloudlet [1]. The research has focused only on energy saving. On the other hand, the paper does not mention how the state of the user's application could be transferred from one cloudlet to the other.

Kommineni, et al. [21] also considered more than one cloudlet in a nearby location. The authors 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. In addition, the wireless network latency between a smartphone and an AP and the BW of a fixed network from the cloudlet to the remote server were considered crucial metrics for cloudlet selection. The real-time network performance is not considered in this paper.

Jararweh, et al. [22, 23] studied how using cloudlet through Wi-Fi could save energy of smartphones than accessing a remote cloud through 3G/4G. Even though this paper describes transferring the state of the offloaded application from one cloudlet to another while the user is moving, the offloading mechanism and the networking collaboration between cloudlets is not mentioned.

Seamless live VM migration for cloudlets with multipath TCP

This section presents our seamless live VM migration approach between cloudlets which is triggered by the relocation of an MD. Our goal is to develop a nearby VM-based approach for cloudlets to improve MCC communications. Two major benefits of our approach are proximity of servers to MDs minimizes latency and offloading to a remote server through minimum latency not only supports QoE, but also saves the energy of the MDs. Dynamic VM synthesis [21, 24] customizes VM on demand for cloudlet users. After service initiation time, the user starts offloading data to enhance the applications performance. Since the cloudlet is one hop away from the user, only a minimum delay is experienced as long as the user remains in the communication range of the cloudlet.

The first problem with the existing dynamic VM migration is that the MD's IP address is changed after it moves from one cloudlet to another. A new IP address for the MD will terminate the established TCP connection with the VM instance, which compels the MD to start a new TCP connection. QoE will be significantly affected or may not be acceptable in such a solution. The second problem occurs when accessing the source cloudlet from the destination cloudlet during the migration process. This approach involves the network latency (could be WAN latency) which defeats the purpose of using a cloudlet to have a short delay.

To address the aforementioned two issues, the following two key points are considered for this research:

- To avoid re-establishment of the TCP connection, the proposed solution uses MPTCP. Hence, it is possible keep the connection established after the MD changes its IP address, which can significantly reduce the delay.
- To ensure that traffic is forwarded between two cloudlets, creating a trusted network collaboration using a VPN for geographically nearby cloudlets is a solution.

MPTCP for seemless live VM migration

The transport protocol plays a crucial role in migration of a VM instance. 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 type* to identify a connection between a server and a client. If any of the five tuples changes 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 with TCP.

Over the WAN, VM migration is performed from one network to another, which forces the IP address of the VM to be changed. This forces the VM to restart all the established connections, which causes delay and unacceptable QoE.

As stated before, this paper proposes the use of MPTCP in both the client and the server to reduce the delay caused by TCP re-establishment. MPTCP works only if the migrating VM runs as a client application, since only the client can send a TCP SYN packet with the new IP address acquired after migration. Even if a server supports MPTCP, the server does not synchronize with the client automatically after the IP address is changed. In addition, the existing MPTCP does not meet all the requirements for VM migration unless the kernel code is modified specifically for VM migration. To resolve this limitation without modification of the MPTCP kernel code, this paper proposes another solution to support a live VM migration without having to re-establish a new TCP connection. The following explains the additions to MPTCP.

Migrating a server VM with MPTCP

If a VM server has only one virtual interface and if it is restarted, all the connections to the server will be lost regardless of the transport layer protocol used. This is how both MPTCP and TCP are designed. With TCP, the server will not actively initiate a connection with a client. Modifying the kernel implementation to meet this requirement (e.g., server could initiate the connection) is a 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.

Therefore, this paper proposes two additional features on top of MPTCP. The first feature is for the VM instance to have two virtual interfaces and the second feature is to construct it in a way that the VM instance knows its future IP address via a pre-configured policy for IP addresses.

Prior knowledge of the next IP address

In VM migration, this feature can be supported with a pre-established network configuration policy. IP address assignment can be achieved without much difficulty if the cloudlets are managed by the same network operator or cooperative network operators. Emerging techniques, such as Fog computing [10], can be used to seamlessly support the feature.

One of the possible policies is explained with the following example:

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 new VM IP address is the next odd IP address of the *paired* MD IP address. For instance, assume that the LAN IP address of the source (C1) and destination (C2) cloudlets are 10.4.0.0/24 and 10.5.0.0/24, respectively. The service was originally initiated at C1 with *paired* IP addresses, i.e., 10.4.0.2/24 for the MD and 10.4.0.3/24 for the VM. After the user changes location, the MD is assigned a new IP address, say 10.5.0.8/24. As soon as the VM is accessed with this new 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.

The ADD_ADDR22 option from MPTCP [14] is used by the VM to inform the client that there is a new or additional IP address. To send the ADD_ADDR22 option, at least one active subflow is needed for the connection. Hence, this research proposes the VMs to have two virtual interfaces inside a cloudlet. One interface is to operate in the regular VM operation mode; the other works only after the VM is migrated.

Each VM inside of a cloudlet is configured with two virtual interfaces, e.g., Eth0 and Eth1. Usually, only Eth0 or Eth1 is used to communicate with the MD at a time. If Eth0 is up, then Eth1 is down. After a VM migration, the state of the two interfaces will exchange. Always, the interface with the DOWN state is intended to serve as the interface during migration. When a VM is accessed from a different IP address than the existing LAN IP address of the VM, the VM knows that the paired MD has changed the location. The VM then prepares the migration by turning on the interface that was in the DOWN state. The IP address for this new interface is determined by the paired MD IP address.

Although ADD_ADDR22 advertises the existence of the additional IP address, 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 after migration, the VM is unreachable through this interface. But the connection

Table 1 Operation phases and status of the server VM

Operation Phases 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

from the MD 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 previously 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 1 summarizes the operation mode of the VM and the interfaces status.

MD handover with MPTCP

A MD installed with MPTCP can be configured with three different operational modes based on requirements. Each operational mode identifies the handover mode as well. A device with one or more interfaces can select one of the handover modes based on the need.

Full-MPTCP mode

The full-MPTCP mode allows creating a TCP subflow with all active 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 throughput. If one of the interfaces goes down, the other active interface keeps working without any disruption.

Backup mode

In backup mode, MPTCP opens TCP subflows with all available interfaces just like the full-MPTCP mode. However, only a subset of interfaces is active for normal data transfer based on the priority of the interfaces. The MP_PRIO option (cf. Table 2) sent by the peer is used to identify the backup interface. As the other peer received the MP_PRIO option, MPTCP keeps the subflow open but never sends data unless the current active interface goes down.

Table 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

Single-path mode

Only a single subflow is active with this mode. MPTCP 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 disconnected from the Wi-Fi network. Compared to the backup mode, this mode waits for two more round-trip times before the new MPTCP subflow is established and data can be sent.

Handover scenarios for MDs

This paper considers two MD handover scenarios for VM migration: (i) from Wi-Fi to Wi-Fi, and (ii) from Wi-Fi to 3G/4G to Wi-Fi. The following presents an analysis for these scenarios. The objective is to show that a handover is performed before the MPTCP connection is closed between the VM and the MD. Both the source and the destination cloudlets are assumed to be in the same VPN. This allows the MD and the VM to communicate using their LAN IP address through the WAN even if they exist in different cloudlet networks.

From Wi-Fi to Wi-Fi

Two geographically close Wi-Fi networks may overlap or may have a gap. The distance between the two Wi-Fi network regions may enable the MD to handover without any interruption if the MPTCP protocol is installed both in the MD and the server the MD is connected to. The role of the RTO in the MPTCP is to give 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 if the application timeout is longer (applications are configured with their own response waiting time).

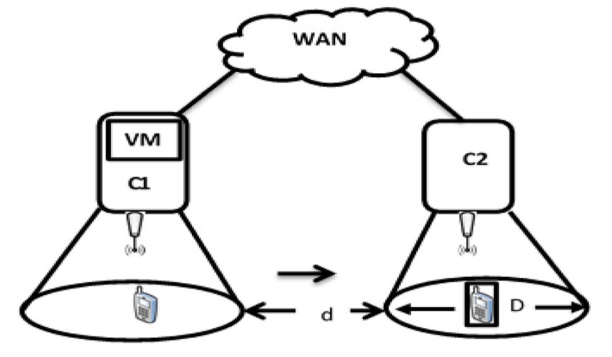

Fig. 1 Two Wi-Fi network regions separated with distance d

If two network regions overlap, for a MD moving from the source to the destination region, the time needed to create a new subflow with the MPTCP connection is the time for the MD to be assigned an IP address from the destination region.

In the case where there is no overlap between two cloudlet network regions, as shown in Fig. 1, the maximum distance d that a MD is allowed to catch up on the established MPTCP connection is calculated as

$$d = (RTO–RTT/2–T_{IP}) \times V_{hum} \quad (1)$$

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 access points (APs) through WAN, V_{hum} is the average human walking speed and T_{IP} is the time that the MD obtain a new IP address from the local Dynamic Host Configuration Protocol (DHCP). The RTO for the MPTCP protocol varies from 13 to 30 min [25]. The actual RTO value depends on the latency between the client and the server.

To guarantee that the new subflow is created before MPTCP connection timeouts, the minimum RTO value for MPTCP, i.e., 13 min (or 780 s) [25], is used to estimate the maximum distance between two cloudlet network regions before MPTCP retransmission timeout happens. This paper also assumes that the user walks with the average human walking speed of 1.2 m per second, i.e., $V_{hum} = 1.2$ m/s, based on research results in Civil Engineering [26]. Practically, when a user is using a MD, the walking speed generally is slower than the normal speed. In addition, to accommodate a wider range of variations or errors in estimating the maximum distance d, this paper discards the value of independent variables RTT and T_{IP} as listed in Eq. (1). Hence, $d <= RTO \times V_{hum} = 780 \times 1.2\ m/s = 936\ m.$ The estimated distance can be used as a reference for network operators that provide cloudlet services or Fog computing to avoid delay due to the MPTCP timeout mechanism. For instance, the distance could be configured shorter than the calculated distance to ensure that MPTCP timeout will not happen.

If D/2 (see Fig. 1) is the Wi-Fi AP communication range in meters, D + d will be the maximum distance between two cloudlets that allows a handover of MDs before the established MPTCP connection timeouts. This calculation for maximum distance gives the ideal distance based on the transport layer protocol. However, applications may 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 either full-MPTCP or single-path MPTCP. Both of these handover modes have the same performance, since only one subflow can be active at a time. Figure 2 shows the sequence diagram for the handover operation between the MD and the VM. Assume that the MD user was in the C1 network region, the left cloudlet as shown in Fig. 1. The VM instance is launched in the nearest cloudlet server (C1). As the user walks out of the region of C1 and is connected to cloudlet C2 on the right, MPTCP protocol handovers the established connection seamlessly. The sequence chart is explained as follows.

Steps 1–3 are the 3-way handshake of MPTCP between the VM and the MD with the MP_CAPABLE option. Duration A in Fig. 2 represents the period between MD and VM while MD remains in C1's coverage. T_t is the time that the MD spent without any network coverage which is less than the actual RTO of the connection.

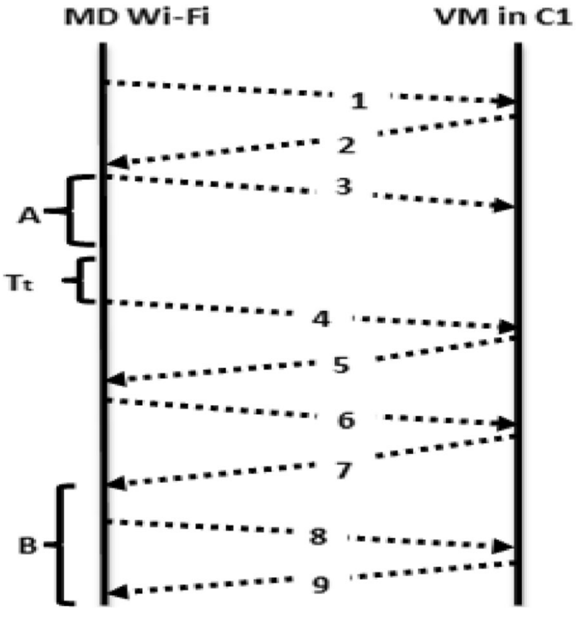

Fig. 2 Sequence chart for handover between a MD and a VM

After the MD is disconnected from C1 and connected with C2, C2 then assigns a new IP address to the MD. The new 3-way handshake starts at step 4 which is the TCP SYN packet with the JOIN option in MPTCP. Step 7 indicates an ACK for the JOIN subflow. Duration B represents data transfer using the new subflow. The REMOVE_ADDR option is sent from the MD to the VM in C1 to inform that its previous IP address is no longer available (step 8). Step 9 is an ACK message.

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

The ubiquitous nature of 3G/4G cellular network is one advantage over Wi-Fi networks. Devices with the 3G/4G capability can take the advantage of this feature to transfer data between Wi-Fi networks seamlessly with the help of MPTCP. The existence of a cellular network between two Wi-Fi networks, as shown in Fig. 3, does not change the maximum distance between two cloudlets for a potential live VM migration, see Eq. (1). The reason is that, for this scenario, some devices, e.g., tablets may only have Wi-Fi capability.

The handover mechanism is the same as that of the Wi-Fi to Wi-Fi network. 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 mode can be used. However, the 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 the backup handover mode is shown in Fig. 4 as an illustration. Steps 1–3 are the 3-way handshake of MPTCP 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 (duration *A* shows the data transfer period) while the JOIN option of MPTCP protocol is sent through 3G/4G interface (steps 4–7). 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.

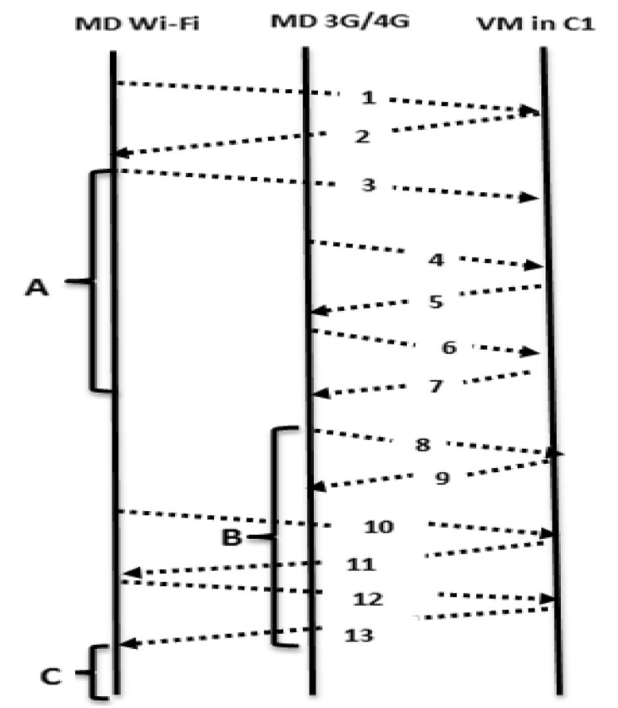

Fig. 4 Sequence chart for MPTCP backup handover mode for Wi-Fi to 3G/4G to Wi-Fi

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 duration). 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 ACK message is sent from the VM to the MD.

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 10–13). 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 period). Then, the 3G/4G subflow remains as a backup.

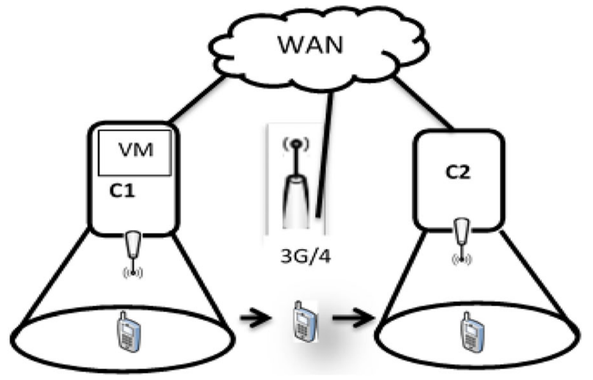

Fig. 3 A scenario with Wi-Fi and 3G/4G networks

Networking collaboration between cloudlets

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. For MCCs, Satyannarayanan, et al. [1] proposed cloudlet servers to be decentralized and

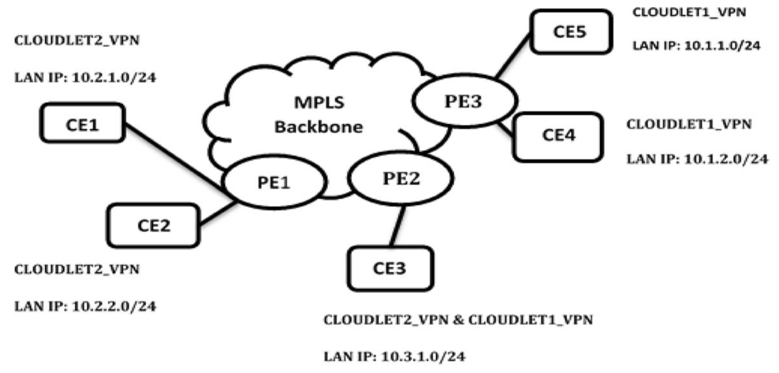

Fig. 5 VPN connections for cloudlets

managed only by the local businesses. In a scenario where multiple cloudlets are geographically close, it is impossible to make a seamless live VM migration from one cloudlet server to another if there is no collaboration between two cloudlets.

Hence, in this research, a VPN connection between cloudlets is proposed to allow a seamless live VM migration between two cloudlets. More detailed description is as follows.

VPN for cloudlets

Accessing a host remotely over the public network needs administrative permission and dedicated software that allows the remote user to access the host. In this research, the main objective is to migrate a VM between two cloudlets over the WAN. To meet this objective, each cloudlet must trust each other and there must be an administrative permission in each host 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 the popular peer model used in the field for VPN is proposed to meet the requirements.

As shown in Fig. 5, in a peer model VPN, there are two main types of routers, e.g., customer edge (CE) and provider edge (PE). Further, each CE is connected to a peer PE. To minimize the latency, we assume that cloudlets are capable of performing some functionalities of a CE. In other words, a cloudlet is connected directly to a PE. Hence, cloudlet and CE are used interchangeably in the context of VPNs in this paper.

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. 5, CLOUDLET1_VPN and CLOUDLET2_VPN are established when a proximity cloudlet servers are found. CE1 and CE2 belong to CLOUDLET2_VPN and CE4, and CE5 belong to CLOUDLET1_VPN, respectively. CE3 is a member of both CLOUDLET1_VPN and CLOUDLET2_VPN.

Adding a cloudlet to a VPN

A cloudlet is added to a VPN if and only if it is near at least one cloudlet of the VPN. The maximum distance a cloudlet can have with at least another cloudlet from a VPN can be estimated using Eq. (1). The network coverage of the wireless network in meters plus 936 m (calculated using Eq. (1)) is the maximum distance one cloudlet should have with at least another cloudlet from a VPN. The algorithm to add a cloudlet in a VPN is defined below. *Dmax* is the maximum distance between two cloudlets to make a seamless handover. The existing cloudlet VPNs are represented by *VPNc*. *C* stands for a new cloudlet to be added to a VPN. *Dck* is the distance between cloudlet *C* and cloudlet *K*.

A new cloudlet candidate can be added to the existing VPNs at any time. 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. 5 is an example.

```
Algorithm 1
Input: Dmax, VPNc, C
Output: VPNs where C is added as a new member
  for each VPN V ⤴ VPNc
    for each cloudlet K ⤴ V
      Add C to V, if Dck <= Dmax
      Update V
    end for
  end for
End Algorithm 1
```

Removing a cloudlet from a VPN will cause complication, if the cloudlet being removed is a neighbour for cloudlets far apart with a distance longer than *Dmax*. One VPN may be divided in two VPNs as a result of the removal of one cloudlet.

Location identifier

Packets coming to a cloudlet are filtered based on the source IP address. When a cloudlet is accessed from a different site, it is initiated to perform the VM migration. VM migration is triggered when a mobile IP address is changed based on the following two reasons:

- A cloudlet can be accessed with a different LAN IP address if it is in 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 latency is involved between the MD and the first cloudlet.
- Since cloudlets are assumed to be owned by local businesses or organizations, 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.

Possible neighbour database

Each cloudlet maintains a possible neighbour database which helps make decision on VM migration. While joining 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 the 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 computes the RTT with their neighbour cloudlets. The RTT value is computed and the database is updated periodically, e.g., every minute or minutes. The following are the two main reasons for making the RTT an important parameter in forming the neighbour database.

The first reason is QoE. After the MD changes a location, it 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. This research work considers VM migration between cloudlets only if the RTT between the cloudlets is less than 150 msec.

The second reason is that RTT determines the TCP throughput. Ideally the maximum TCP throughput achievable is: *TCP Throughput = Receiver window size / RTT.* Iperf [14] 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. 6.

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

The VM migration decision is based on the neighbour database. When a cloudlet receives a packet from a 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.

Other VPN functionalities

To fully support cloudlet services, more VPN-related functionalities are needed. VPN-related techniques for cloudlets are beyond the scope of this paper. A detailed description on those functionalities can be found in [18].

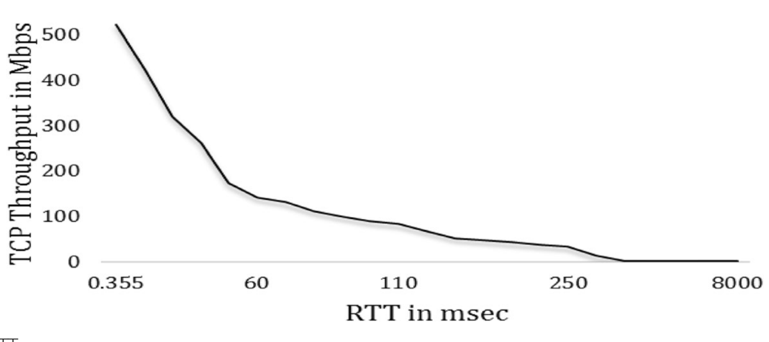

Fig. 6 TCP Throughput vs RTT

Table 3 VM specifications inside the Linux host

VM	Virtual CPU	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

Experimental setup and results

This section presents experiments and results of live VM migration with a change of IP address using MPTCP.

Experiment environment

Experiments have been conducted using a 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 [27] which uses a pre-copy RAM state migration mechanism [14].

Table 3 shows the specifications for the emulated components. Four VMs inside the Linux host are used for the experiment (see Table 3 and Fig. 7). VM1 and VM2 represent cloudlet 1 (C1) and cloudlet 2 (C2), respectively. The nested VM (VM3) inside C1 denotes the server VM instance used by the MD, and VM4 represents the MD.

Both wireless and wired transmission medium technologies are characterized by a VM bridged networking. The VMs networking in Fig. 7 emulates the cloudlets and the MD as shown in Fig. 1. In Fig. 7, 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, as shown in Table 1.

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 VM4's Eth0 to the DOWN state, then Eth1 to the UP state. Since most of the current Wi-Fi interface can connect to one AP at a time, the above mentioned method is suitable to represent a Wi-Fi interface for our objective.

MPTCP protocol is installed in all the VMs inside the Linux host in Fig. 7. A static routing protocol is used to forward traffic 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 the coverage. The following example explains the scenario. Assume that the MPTCP connection is initially established with the VM4's Eth0 interface and the

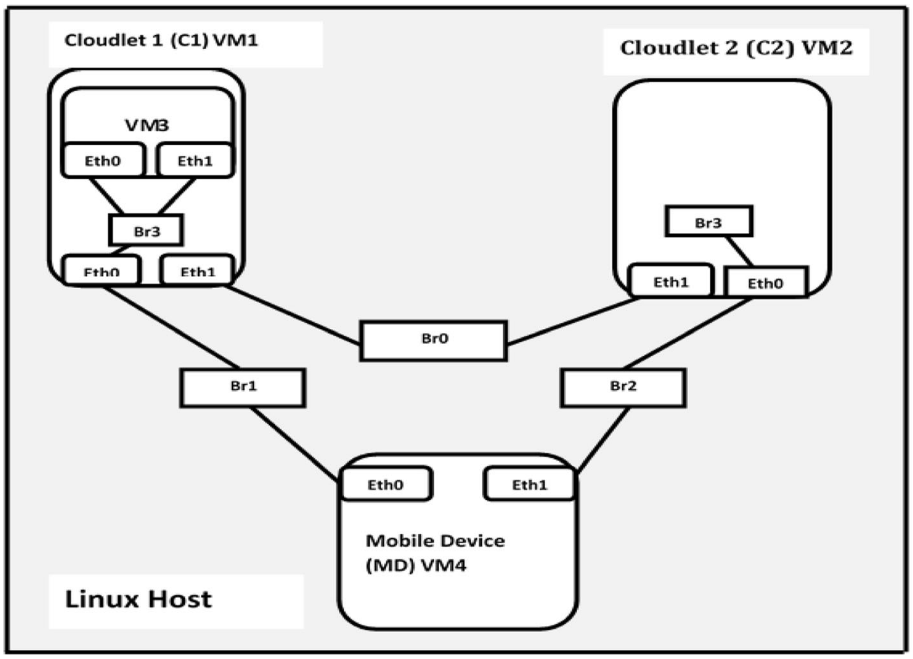

Fig. 7 Experiment using VMs and bridged networking inside a Linux host

VM3's Eth0 interface. The static routing forwards initial 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 emulate the movement of the MD from C1 to C2. 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 bandwidth (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 Linux traffic control (*tc*) command. Token Bucket Filter (*tbf*) queuing discipline provided by Linux is used to set the upload BW of an interface.

Performance metrics and measurements

QoE can be affected by various factors. This paper adopts four common metrics for performance evaluation.

- Throughput: Changing coverage from C1 to C2 exposed the MD to low throughput due to the WAN delay. If VM migration and the MD traffic share the same path, the MD's throughput will drop during the migration. Returning to the previous high throughput level for the MD is a performance measurement for the system. The *Iperf* [14] tool is used to measure throughput between the VM and the MD.
- RTT latency: As in throughput, the movement of the MD introduces a high RTT between the VM instance and the MD. Monitoring the RTT latency after the live VM migration is an important performance metrics. The *ping* command is used to measure the RTT with 1 s interval.
- Total VM migration time: The VM is migrated if the MD changes location and network coverage. The time duration the MD accesses the VM instance with high RTT latency and low throughput through WAN have to be as short as possible. This duration is tied to the VM migration process. The longer the VM migration time, the poorer the QoE. The total VM migration time has been collected. The VM migration is initiated using a Linux command, the time before and after the migration command completes is considered as the total VM migration time.
- 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, [16]), have a VM downtime. Unacceptable VM downtime may be experienced based on the network condition between source and destination.

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 (see Section III.A) at the same time. The VM downtime is the time the VM from the source is paused (no *ping* response) to the time the VM from the destination host starts running (*ping* response received). The downtime of the VM is measured with an interval of 10 msec. It is possible to minimize the interval to 1 msec but flooding of packets to a VM may influence the performance. For this reason, in this paper the VM downtime is known if only it is greater than 10 msec. Otherwise the down time is expressed as less than 10 msec.

Networking assumptions

This paper assumes that cloudlets are connected to high-speed Internet. The availability of high BW and low RTT latency between cloudlets in the vicinity are assumed throughout the experiment. The assumption of low RTT latency between cloudlets is supported by the geographical proximity between hosts for the cloudlet environment. In addition, low RTT is supported due to:

- Minimum Hops: As illustrated in Fig. 5, cloudlets are assumed to be capable of performing as CEs (customer edge), which make the cloudlets only one hop away from a PE (provider edge). The possibility of cloudlets in the vicinity that will be aggregated to one PE is high. In such cases, the total number of hops between two cloudlets is two, i.e., CE1 → PE1 → CE2 (see Fig. 5). This minimizes the total delay in each router.
- High data transmission rate: High-speed Internet connection to the cloudlet is also an assumption made in this paper. High-speed Internet services found from Internet Service Providers, such as Rogers in Canada, are taken as a reference. The service with download and upload speed of 350Mbps are high speed Internet connection products found in the market by the time the experiments were conducted, which represents low transmission delay.

Table 4 Baseline results for rtt and throughput

	RTT	Maximum Throughput
Between VM1 and VM2	0.382 msec	627 Mbps
Between VM3 and VM4	0.392 msec	160 Mbps

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, RTT results were collected for *Iperf* servers around the world, including 71 ms for California, 119 ms for (Brabant Wallon) Belgium, and 143 ms (Saint Petersburg) Russia. All those servers are > 15 hops away from the host. Hence, it is fair to assume that that the RTT between the cloudlets can be less than 150 ms. We also varied RTT from 20 ms to 150 ms in the evaluation.

The following sub-sections show some results. The first set of results represents baseline performance. No application was running on the migrating VM, i.e., the migrating VM has all the resources or there is no resource sharing for this scenario.

Baseline performance for VM migration

The baseline performance of MPTCP was evaluated with a single path between VM1 and VM2 (as shown in Fig. 7). MPTCP acts like a regular TCP and there was no application running on the migrating VM (VM3). The purpose is to minimize the effect of latency and BW usage between VM3 and other applications and to maximize the performance.

The average measured network RTT latency between the VMs and the maximum throughput achieved for the experiment setup is shown in Table 4. *Iperf* is used to measure the throughput and the *ping* command is used to measure the average RTT. The receiver window size (RWS) is set to the default TCP window size which is 64KByte.

Without any application running on VM1, VM2, and VM3, migrating VM3 from VM1 to VM2, (see Fig. 7), takes 10.67 s and VM downtime is 243.46 msec on average. The total VM migration and the VM downtime are similar to the result reported in other research [28]. Figure 8 depicts the effect of VM3 migration on the throughput between VM1 and VM2. The period T in Fig. 8 represents the total VM migration time. During T, the result illustrates that the maximum throughput decreases and fluctuates once the VM migration is started.

Performance of live VM migration with MPTCP

This section presents the results for a seamless live VM migration using MPTCP. The TCP connection remains open after both VM3 and VM4 change their original IP addresses.

As it is presented in Section III.A, 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 migration is achieved with the change of the VM IP address. In addition, the results from the experiment also prove the seamless handover of the MD from C1 to C2 network using MPTCP.

Figure 9 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. The maximum available BW between VM1 and VM2 through Br0 was configured as 350 Mbps and the network RTT was 20 ms in this experiment. In Fig. 9, the time is divided into slots A, B, C, D, and E. The following paragraphs describe the situations for each time slot.

Time slot A is the time where VM4 initially accesses VM3 through VM1. During period A, VM3 resides inside VM1 and VM4 is directly connected to VM1. The MPTCP connection is established with the UP state interfaces and the IP addresses as shown in Table 5 before VM4 migration. The traffic flow between VM4 and VM3 is as follows:

Eth0 of the VM4 → Br1 of the Linux Host → Br3 of VM1 → Eth0 of VM3.

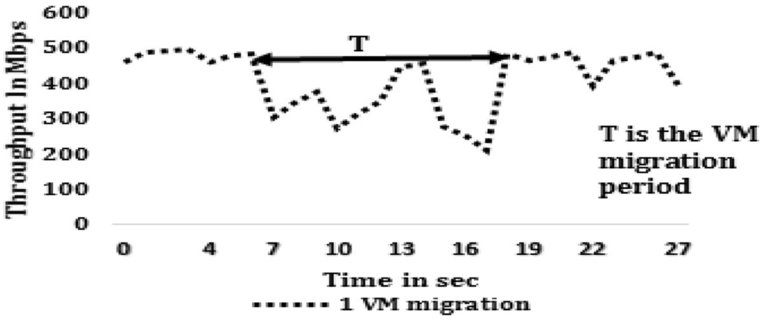

Fig. 8 Throughput between VM1 and VM2 when VM3 is migrating from VM1 to VM2

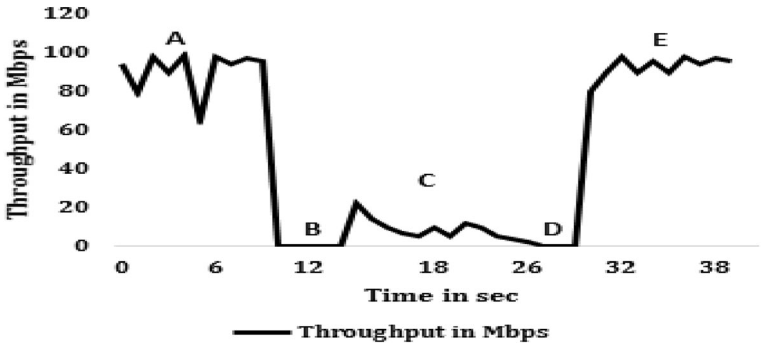

Fig. 9 Throughput between VM3 and VM4 before and after VM3 migration from VM1 to VM2

After some time, Eth0 of VM4 is taken down. Time slot B denotes the time where both the Eth0 and the Eth1 interfaces of VM4 are down (zero throughput for 5 s). This is to imitate the time a MD user walks from one Wi-Fi AP (access point) to another and the time the new Wi-Fi AP takes to connect and assign a new IP address for the real MD. If a real MD and Wi-Fi AP were used, the performance difference would be different in time slot B. The delay depends on the distance between the two WiFi network coverages and the walking speed of the user (See Fig. 1). For this experiment, the delay is configured to be 5 s. After 5 s 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 and an IP address for Eth1, as presented in Table 3 after VM4 migration.

As we can see from Fig. 9, during time slot C, the application keeps running even after VM4 IP address has 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, the throughput decreases because of the involvement of the WAN between VM3 and VM4 (emulated via Br0) to keep the application running.

Table 5 Initial Interfaces and IP Addresses for VM3 and VM4

	Interface	Before VM4 migration		After VM4 migration	
		Interface state	IP address	Interface state	IP address
VM3	Eth0	UP	10.1.1.15/24	UP	10.1.1.15/24
	Eth1	DOWN	NONE	DOWN	NONE
VM4	Eth0	UP	10.1.1.14/24	DOWN	NONE
	Eth1	DOWN	NONE	UP	10.1.2.18/24

As soon as VM3 is accessed with the new IP address of 10.1.2.18/24 for MD (VM4), VM3 recognizes that VM4 has changed its original location (interface). The moving of MD to a new location or VM4 changing the initial IP address triggers the migration of VM3 from VM1 to VM2. This is done to minimize the latency and maximize the throughput between VM3 and VM4 by avoiding the traffic flow through the WAN. As the approach mentioned in Section III.A, the Eth1 interface of VM3 will be changed to the UP state and assigned with the next odd IP address (i.e., 10.1.2.19/24) to pair with the current VM4 IP address (10.1.2.18/24).

VM3 migration process is completed after time slot D. Time slot D indicates that during the time period, VM3 is inaccessible at the application level. The total VM migration time is 14.00 s. VM3 application level downtime is 3 s whereas the network layer downtime is only 204.3 msec.

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

Eth1 of the VM4 → Br2 of the Linux Host → Br3 of VM2 → Eth1 of VM3.

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.

The following sub-sections demonstrate the effect of maximum available BW and RTT latency on the performance of VM migration between VM1 and VM2 for time slots C and D. The results were based on the KVM hypervisor which is known for its shorter VM downtime than that of Xen Server, Hyper V and VMware [28].

Table 6 Total VM Migration Time (*in Seconds*)

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

Table 7 Network Layer VM Downtime (*in Milliseconds*)

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

Total VM migration time and VM downtime

This experiment measures the total VM migration time (C + D) and VM network layer downtime by varying RTT and BW. The network RTT values used were 20 ms, 50 ms, 100 ms, and 150 m, and the BW values used for the experiments were 350Mbps, 175Mbps, and 60Mbps.

Results for total VM migration time and network layer VM downtime are shown in Tables 6 and 7, respectively. 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 that for the low BW, e.g., 60Mbps.

Figures 10 and 11 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, 175Mbps and 350Mbps than the low BW, 60Mbps. The following points explain the reasons for the results shown in Figs. 10 and 11.

- 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

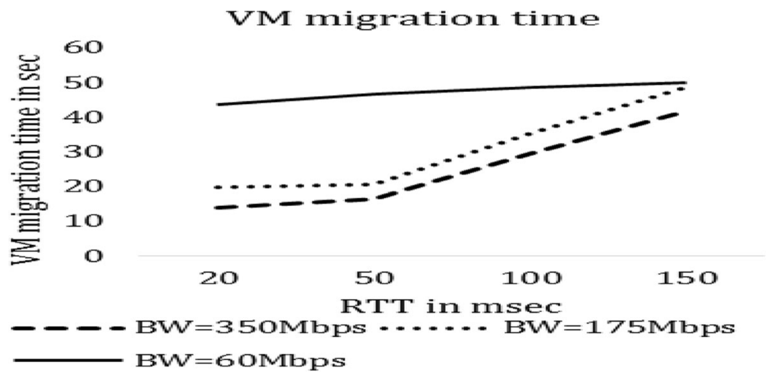

Fig. 10 Latency vs VM Migration Time

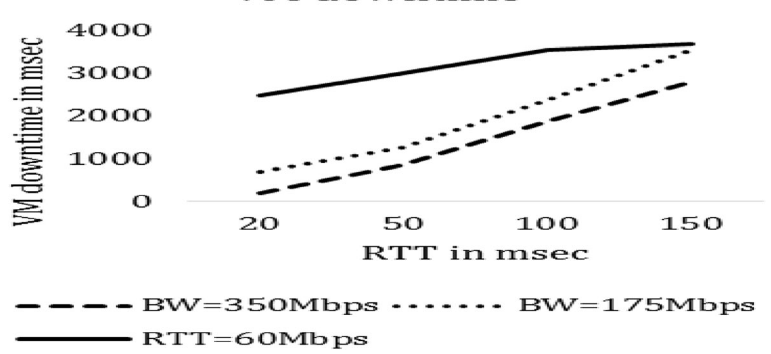

Fig. 11 Latency vs VM Downtime

memory and it has a way to abstract the free memory for migrating VM at the destination host. Recall from the baseline performance result, migrating more than 263MByte takes only 10.67 s. The estimated data transfer speed of KVM will be the used memory size divided by the migration time which is equal to 197.19Mbps. Note also that 197.19Mbps is the estimated speed, not the exact speed. To calculate the exact data transfer speed we need to know how many iterations 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 exact 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 increases the total VM migration time.
- The throughput for VM3 migration does not stay the same throughout the migration process. The TCP auto tuning is turned on, both on VM1 and VM2 and also the TCP stack on VM1 and VM2 is configured with minimum of 4Kbyte, maximum of 5394Kbyte, and 85Kbyte default receiver buffer 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 buffer size, TCP data transmission will not be efficient. To identify the scenarios which do not use the available BW efficiently, the average receiver window buffer size has to be known. TCP performance analysis is given in the next sub-section.

Performance analysis and TCP

Performance analysis related to TCP is crucial for migration decision making and design, as MPTCP is built upon TCP. The throughput for VM3 migration fluctuates during the migration process, as the migration is performed over the TCP protocol and the TCP auto tuning was turned on. Further, TCP performance is also closely related to the receiver window size (RWS). The TCP stack on VM1 and VM2 was configured with minimum of 4KByte, maximum of 5394KByte, and 85KByte default RWS. The efficiency of TCP is KAI determined by the BW delay product (BDP). If the BDP is greater than the RWS, TCP data transmission is not efficient. To identify the scenarios which do not use the available BW efficiently, the average RWS has to be known.

Table 8 The BDP results in KByte

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

Table 8 shows the BDP values in KByte for the corresponding RTT and BW values. From the BDP results, when BW is 350Mbps and RTT is 150 msec, the data transmission is inefficient even when the RWS is at its maximum value, i.e. 6409Kbyte ((350Mbps × 150 msec) / (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 RWS and the RTT instead of the available BW. If BDP is greater than the RWS, the ideal maximum TCP throughput can be calculated as:

$$Max.\ TCP\ Thoughput = RWS/RTT \tag{2}$$

Based on the total VM size and the VM migration time (as shown in Table 4), we can estimate the data transmission speed or throughput (e.g., VM size/VM migration time), which in turn can be used to calculate the RWS using Eq. (2). The shaded area in Table 6 indicates the scenarios where the transmission speed is inefficient. During those periods, data transmission speed is determined by RTT and RWS instead of the available BW. This is also the reason for a similar high total VM migration time values when RTT = 150 msec regardless of the available BW values (see Table 4).

Latency and performance analysis

This subsection describes the latency performance analysis during the VM migration process and the experiments show that the RTT between VM3 and VM4 increases from its original value. The goal here is to show the bahaviour of the RTT during migration. The RTT values used in this experiment are 20 msec, 50mesc, 100 msec, and 150 msec and the maximum available BW used in this experiment are 350Mbps and 60Mbps.

Before analyzing the results as shown in Figs 12, 13, 14, 15, 16, and 17, 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.

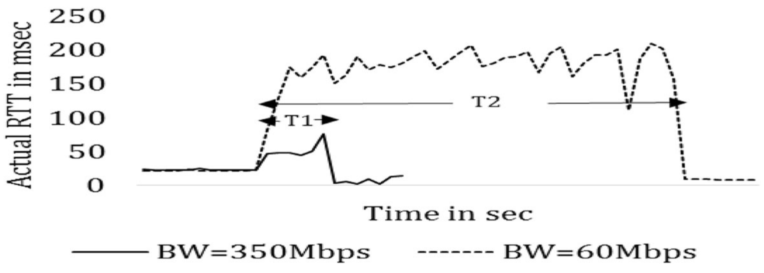

Fig. 12 Actual RTT during VM migration for initial RTT = 20 msec

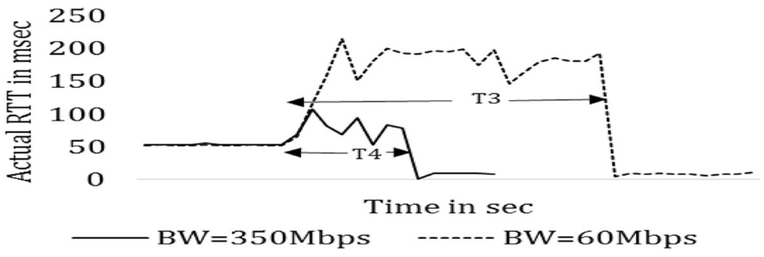

Fig. 13 Actual RTT during VM migration for initial RTT = 50 msec

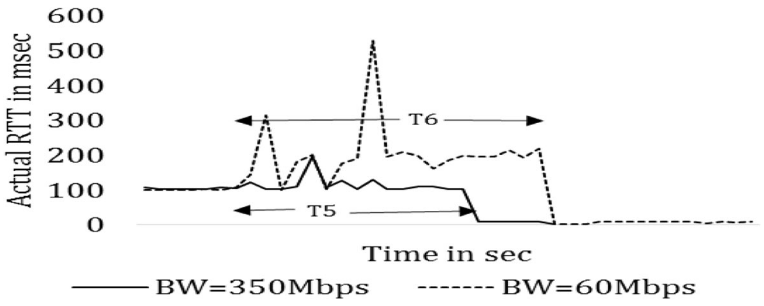

Fig. 14 Actual RTT during VM migration for initial RTT = 100 msec

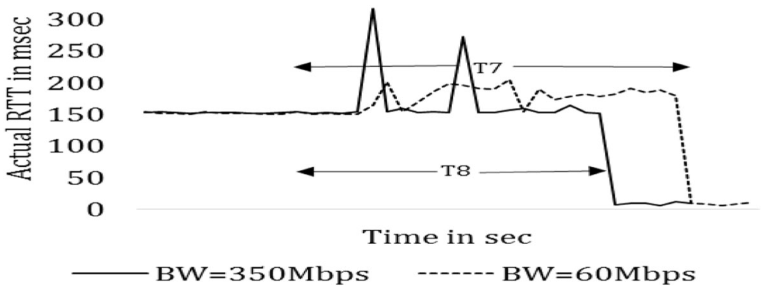

Fig. 15 Actual RTT during VM migration for initial RTT = 150 msec

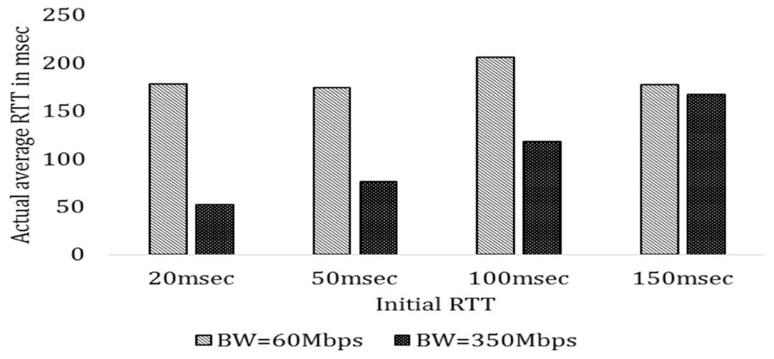

Fig. 16 Average RTT during migration process vs initial RTT

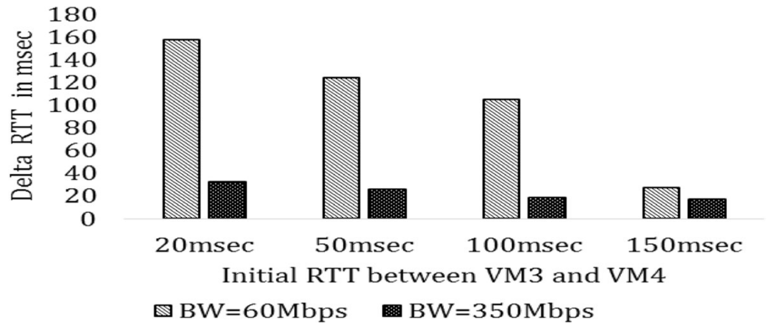

Fig. 17 Delta RTT vs the Initial RTT

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

Figures 12, 13, 14 and 15 show the actual RTT during VM migration for initial RTT of 20 msec, 50 msec, 100mesec and 150 msec respectively. The RTT was measured by using *ping* command with a 1 s interval. The objective is to show how much the latency increases when the VM starts to migrate. All the graphs show the results from the time when VM4 starts to access VM3 through the WAN. At the start, the low latency shows the RTT between VM4 and VM3 before VM3 migration process is started. The time durations are noted as T1, T2, T3, T4, T5, T6, T7 and T8 show the total VM migration time for different available BW and RTT values and the values for different Ti are given in Table 9. Figures 12, 13, 14 and 15 (i.e., T1, T4, T5, and T8) show that as the initial RTT increases, the duration of migration increases (cf. Table 9).

Table 9 Total VM migration time for the time duration shown in Figures 12, 13, 14 and 15

T1	T2	T3	T4	T5	T6	T7	T8
8.12 s	24.30s	24.0 s	9.91 s	15.62 s	25 s	26 s	23 s

Figure 16 summarizes the results in terms of the actual average RTT with respect to initial RTT of 20 msec, 50 msec, 100 msec, and 150 msec respectively. In the case of initial RTT of 150 msec cf. Figure 16), there is little or no difference in actual average RTT between the lower BW (i.e. 60Mbps) and the higher BW (i.e., 350 msec). For fixed latency, minimizing the available BW increases the VM migration time and the actual average RTT significantly. The significance is more noticeable for low latency. For low BW, 60Mbps, the actual average RTT is independent of the initial RTT. Regardless of the initial RTT, the actual average RTT reflects the same amount. For the high BW, 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.

Figure 17 illustrates 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. Hence, having a low initial RTT before VM migration is not a guarantee for low RTT for 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.

Throughput performance analysis

One main advantage of using a nearby cloudlet is to achieve a high throughput. The result shown in Fig. 18 is the throughput achieved between VM3 and MD (VM4). The figure shows the throughput only after VM4 is directly connected to VM2. In Fig. 18, the time before t = 10 is the time when VM4 is connected to VM3 through the WAN. At t = 10, the migration process is manually started. The completion of the VM migration for various RTT values is depicted in Fig. 18.

Lower RTT values result in higher throughput, as observed in Fig. 18. The throughput of VM4 drops at t = 10 when migration starts. Specially, for RTT = 20 msec, the throughput decreases significantly at t = 13. On the other hand, the throughput starts to increase earlier when RTT is lower. Figure 19 demonstrates a five times increase in RTT (from 20 msec to 100 msce), the average throughput only decreases by 26.4 % (from 7.2Mbps to 5.3 Mbps). With RTT = 20 msec, however, the performance is better than all the other RTT values.

If the proposed MPTCP and prior knowledge of the migrating IP address are not used, the throughput of the MD user would be zero before the VM is launched in the new cloudlet. If a real application is used, a notice7able delay is inevitable during the VM migration process.

VM migration decision algorithm

The results show that it is impossible to guarantee fast VM migration over the WAN using only the bare hypervisor. Optimization techniques for WAN VM migration mentioned in other research efforts, such as [18], has to be adopted to make the system more efficient. In addition, a smart VM migration decision maker is desirable for better QoE. As the main objective is eliminating the service re-initiation time for the MD, an efficient service has to be provided during migration. All the performance metrics need to be considered before the VM migration starts. If the VM migration is predicted to result in 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 determine the VM migration time. In this work, 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 (maximum of 15 % of VM3 CPU). Thus, 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 dirty memory pages frequently. A precopy migration works best only when memory pages can

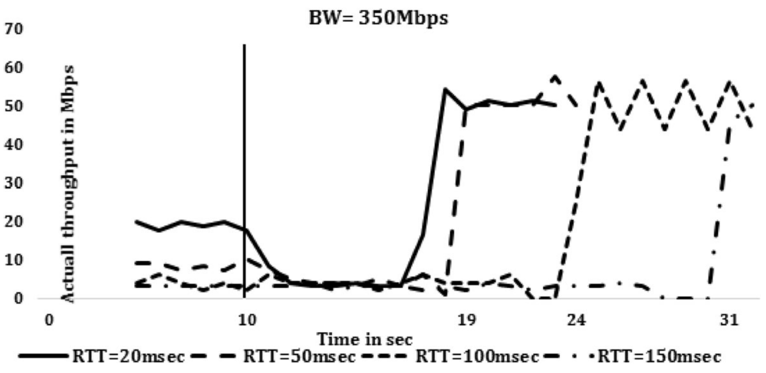

Fig. 18 Actual VM4 throughput during VM3 migration for 350Mbps

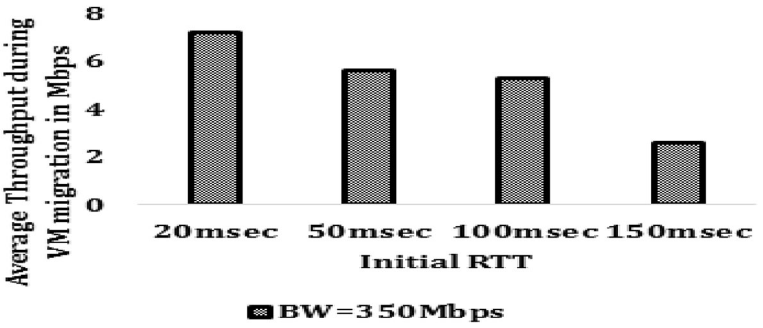

Fig. 19 Average throughput during VM3 migration

be copied to the destination host faster than they are dirtied. In addition, KVM will never finish live migration if the RAM page is dirtied with a speed of 32Mbps [28]. Therefore, for live migration to be beneficial, a decision algorithm plays an important role in determining if the live VM migration should proceed. The decision algorithm needs to consider factors such as RTT latency and available BW. The proposed VM migration decision algorithm is as follows:

```
Algorithm 2
Input: average RTT, initial RTT
Estimate Delta RTT  /* Delta RTT is the difference between the actual
                       average RTT and the initial RTT */
if (initial RTT + Delta RTT <= 150msec}
    then  migrate the VM
    else if ((150msec < Initial RTT + delta RTT <= 1sec} and
            (estimated VM migration time < service initiation time))
        then migrate the VM
        else
            ignore VM migration
            notify user to start a new service
            destroy the VM instance
        end if
end if
End Algorithm 2
```

The algorithm considers three main parameters in making decision on the VM migration; the service initiation time, the VM migration time and the RTT during VM migration. If the RTT during migration is less than 150 msec, it means that the user is having a good QoE even through the WAN. Regardless of the total estimated VM migration time, the VM migration will be performed. There is no need to trigger VM migration, if the actual RTT during VM migration is greater than 1 s since the service becomes useless for the user.

The algorithm should be implemented in each cloudlet. The estimation of the total VM migration time and delta RTT can be achieved through profiling. Applications and network resources profiler need to be deployed in to the system which keeps track of all VM migration results. Over time, the profiler provides precise information. Each cloudlet also can share their profiles to learn a wide variety of situations within short time.

Conclusions and future directions

Live VM migration over the WAN includes migration of the RAM state, network, and storage. This paper presented an approach for live server VM migration using MPTCP. To realize the approach, two main features have been proposed: (i) a scheme for IP address assignment to reduce TCP downtime and (ii) two virtual interfaces were proposed for the VM migration; one interface operates during normal operation and the other one operates when a VM migration is triggered.

A number of experiments using emulation have demonstrated that the proposed approach can be realized to support actual live VM migration. Further, the paper investigated the performance of VM migration, including total live migration time, VM downtime, throughput, RTT and TCP protocol. In addition, the effect of various parameters has been investigated for further advancement in this fast growing field.

The experiment does not consider storage migration. Storage migration is influenced by the network resources (BW and RTT) and the disc I/O speed. If the available BW is less than the reading speed, it may take a long time for VM migration. In general, migrating storage would not be effective.

The concept of cloudlets is similar to the emerging Fog computing technology. Our proposed live VM migration between cloudlets can also be applied in a Fog computing environment, where a service running in a Fog can be migrated from one edge device to another as the MD is moving from one area to another. In Fog computing, cloudlets are most likely managed by the same network operator, which can effectively facilitate IP address assignment and other migration issues, including the business concerns. VM migration for Fog computing is worth investigating.

Acknowledgements
We would like to express our gratitude to Jerry Buburuz who provided resources and resolved all the technical errors we faced during the lab environment setup.

Authors' contributions
The work is primarily based on FT's Master's research and thesis, which was co-supervised by CL and SA. All authors contributed to the technical areas and the writings. FT designed and implemented the experiments. All authors read and approved the final manuscript.

Competing interests
The authors declare that they have no competing interests.

Received: 12 January 2016 Accepted: 10 August 2016
Published online: 06 September 2016

References

1. Satyannarayanan M, Bahl P, Caceres R, Davies N (2009) The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput 8(4):14–23
2. Cuervo E et al (2010) MAUI: Making Smartphones Last Longer with Code Offload". Proc. of the 8th Int'l Conf. on Mobile Systems, Applications, and Services, ACM, New York, pp 49–62
3. Flinn J (2012) Cyber Foraging: Bridging Mobile and Cloud Computing via Opportunistic Offload, Morgan & Claypool Publishers
4. Chun B-G, Ihm S, Maniatis P, Naik M, Patti A (2011) CloneCloud: Elastic Execution between Mobile Devices and Cloud". Proc. of the 6th Conf. on Computer Systems, ACM, New York, pp 301–314
5. Soyata T et al (2012) Cloud-Vision: Real-time Face Recognition using a Mobile-Cloudlet-Cloud Acceleration Architecture. Proc. of Symp. on Computers and Communications, Cappadocia, pp 59–66
6. Ra M, Sheth A, Mummert L, Pillai P, Wetherall D, Govindan R (2011) Odessa: Enabling Interactive Perception Applications on Mobile Devices. Proc. of the

Int'l Conf. on Mobile Systems, Applications, and Services (MobiSys), ACM, New York, pp 43–56
7. Gordon MS, Jamshidi DA, Mahlke S, Mao ZM, Chen X (2012) COMET: Code Offload by Migrating Execution Transparently. Proc. of the 10th USENIX Symp. on Operating Systems Design and Implementation (OSDI), Hollywood, pp 93–106
8. Ha K et al (2013) The Impact of Mobile Multimedia Applications on Data Center Consolidation. Proc. of the IEEE Int'l Conf. on Cloud Engineering, Redwood City, pp 166–176
9. Satyanarayanan M (2014) A brief history of cloud offload. GetMobile 18(4):19–23
10. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog Computing and its Role in the Internet of Things. Proc. of the 1st Edition of the MCC Workshop on Mobile Cloud Computing, ACM, New York, pp 13–16
11. Yi S, Li C, Li Q (2015) A Survey of Fog Computing: Concepts, Applications, and Issues. Proc. of the Workshop on Mobile Big Data, ACM, New York, pp 37–42
12. Stojmenovic I, Wen S (2014) The Fog Computing Paradigm: Scenarios and Security Issues. Proc. of the Federated Conf. on Computer Science and Information Systems, Warsaw, pp 1–8
13. Teka FA, Lung C-H, Ajila S (2015) Seamless Live Virtual Machine Migration with Cloudlets and Multipath TCP. Proc. of the 39th IEEE Computer Software and Applications Conf. (COMPSAC), Taichung, pp 607–616
14. MPTCP, www.multipath-tcp.org, Last accessed in Nov, 2015
15. Ha K, Pillai P, Richter W, Abe Y, Satyanarayanna M (2013) Just-in-time Provisioning for Cyber Foraging". Proc. of the 11th Int'l Conf. on Mobile Systems, Applications, and Services, ACM, New York, pp 153–166
16. Hines M, Deshpande U, Gopalan K (2009) Post-copy live migration of virtual machines. SIGOPS Oper Syst Rev 43(3):14–26
17. Clark C, Fraser K, Hand S, Hansen JG (2005) Live Migration of Virtual Machines". Proc. of the 2nd Symp. on Networked Systems Design & Implementation, USENIX, Berkeley, pp 273–286
18. Wood T, Ramakrishnan KK, Shenoy P, van der Merwe J (2011) CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines. Proc. of the 7th ACM SIGPLAN/SIGOPS Int'l Conf. on Virtual Execution Environments, ACM, New York, pp 121–132.
19. Nicutar C, Paasch C, Bagnulo M, Raiciu C (2013) Evolving the Internet with Connection Acrobatics". Proc. of the Workshop on Hot Topics in Middleboxes and Network Function Virtualization, ACM, New York, pp 7–12
20. Li J, Bu K, Liu X, Xiao B (2013) ENDA: Embracing Network Inconsistency for Dynamic Application Offloading in MCC". Proc. of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing, ACM, New York, pp 39–44
21. Kommineni S, De A, Alladi S, Chilukuri S (2014) The Cloudlet with a Silver Lining". Proc. of 6th Int'l Conf. on Communication Systems & Networks, Bangalore, pp 1–4
22. Jararweh Y et al (2013) Resource Efficient Mobile Computing Using Cloudlet Infrastructure". Proc. of 9th Int'l Conf. on Mobile Ad-hoc and Sensor Networks, Dalian, pp 373–377
23. Jaraweh Y, Tawalbeh L, Ababneh F, Khreishah A, Dosari F (2014) Scalable Cloudlet-based Mobile Computing Model. Proc. of the 11th Int'l Conf. on Mobile Systems & Pervasive Computing, Niagara Falls, pp 434–441
24. Kondo T, Aibara R, Suga K, Maeda K (2014) A Mobility Management System for the Global Live Migration of Virtual Machine across Multiple Sites". Proc. of the 38th Int'l Computer Software and Applications Conf. Workshops (COMPSACW), Vasteras, pp 73–77
25. Paasch C et al (2012) Exploring Mobile/WiFi Handover with Multipath TCP. Proc. of the ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design, ACM, New York, pp 31–36
26. LaPlante JN, Kaeser TP (2004) The continuing evolution of pedestrian walking speed assumptions. ITE J 74:32–40
27. KVMQEMU, www.linux-kvm.org. Last accessed in Nov, 2015
28. Hu W et al (2013) A Quantitative Study of Virtual Machine Live Migration". Proc. of the 2013 ACM Cloud and Autonomic Computing Conf., Article No. 11, ACM, New York