Fast Transparent Virtual Machine Migration in Distributed Edge Clouds

Lucas Chaufournier[1], Prateek Sharma[1], Franck Le[2], Erich Nahum[2], Prashant Shenoy[1], Don Towsley[1]
[1]University of Massachusetts Amherst, [2]IBM Thomas J. Watson Research Center

ABSTRACT

Edge clouds are emerging as a popular paradigm of computation. In edge clouds, computation and storage can be distributed across a large number of locations, allowing applications to be hosted at the edge of the network close to the end-users. Virtual machine live migration is a key mechanism which enables applications to be nimble and nomadic as they respond to changing user locations and workload.

However, VM live migration in edge clouds poses a number of challenges. Migrating VMs between geographically separate locations over slow wide-area network links results in large migration times and high unavailability of the application. This is due to network reconfiguration delays as user traffic is redirected to the newly migrated location. In this paper, we propose the use of multi-path TCP to both improve VM migration time and network transparency of applications.

We evaluate our approach in a commercial public cloud environment and an emulated lab based edge cloud testbed using a variety of network conditions and show that our approach can reduce migration times by up to 2X while virtually eliminating downtimes for most applications.

Keywords

Multipath TCP, WAN Migration, Edge Computing, Virtual Machine Live Migration

CCS Concepts

•Networks → Network performance analysis; Cloud computing; Data center networks; •Software and its engineering → Cloud computing; •General and reference → *Empirical studies;*

SEC '17, October 12–14, 2017, San Jose / Silicon Valley, CA, USA

DOI: https://doi.org/10.1145/3132211.3134445

1. INTRODUCTION

Cloud computing has emerged as a popular paradigm for hosting a variety of online applications and services. Today's commercial cloud computing platforms employ a quasi-centralized architecture, where cloud resources such as servers and storage are hosted in a few large global data centers. While doing so provides economies of scale and statistical multiplexing of resources across a large customer base, it usually implies hosting a cloud application at a cloud site that may be "distant" from its end-users.

Recently a new architecture for cloud computing has emerged that is based on the principle of *edge computing*—where cloud resources such as servers and storage are distributed in small clusters at a large number of edge cloud locations. Applications can be hosted at the edge of the network closer to their end-users to minimize latency. Cloudlets [39] and others [7, 11, 35] have advocated such a distributed edge cloud model. Edge clouds can also be viewed as a natural evolution of content delivery networks (CDNs) that rely on servers and storage at edge locations to deliver content such as video. In the case of edge clouds, edge locations host full-fledged applications rather than just data.

The growing popularity of mobile phones and Internet of Things (IoT) devices has made edge clouds an attractive choice for running emerging mobile and IoT services. Since these devices are often resource constrained, they can rely on computation and storage at nearby edge cloud servers to implement their services (e.g. computational offloading [5, 14, 21]). Furthermore, edge clouds can provide low-latencies (due to their proximity to mobile or IoT Devices) enabling applications such as mobile multiplayer games or IoT applications with actuation and tight control loops where low latency is a key requirement.

Edge clouds are poised to enable a slew of novel applications with new requirements. Consider a mobile phone that runs mobile applications (e.g. Kinect style gesture recognition) that offload resource-intensive computations to a nearby edge cloud. As the user travels from home to work and back, the nearest edge cloud to the current phone location will change over the course of the day. To provide the lowest latency, the edge cloud VMs that support computational offload will need to "follow the user" and transparently migrate to a new edge cloud location whenever the user changes location. Similarly, cloud-based virtual desktops— cloud-based VMs that provide a desktop PC accessed through thin clients on mobile devices [17] will also need to be migrated to different edge cloud sites to continue providing low-latency access in the presence of user nomadicity. Researchers have also advocated "follow the sun" [36, 40] applications where a global team of users collaboratively work on a task using a cloud-based application—as the sun rises on each continent, the cloud application is migrated to a new edge location to provide low latency access to team members on that continent. Finally, multi-user applications such as online multiplayer games may be replicated at multiple edge sites and replicas may be migrated within each region to optimize performance as the user base changes over time.

The ability to transparently move the running state of an edge cloud application from one edge cloud location to another is key to enabling the above scenarios. Assuming edge cloud servers are virtualized, a simple approach may be to live migrate the VM [13] and its associated state from one location to another. However, live VM migration, which was developed for LAN migration *within* a data center, suffers from two key limitations in the WAN scenario. First, the VM network IP addresses will change across WAN edge cloud sites, which is no longer transparent to end devices (phones, tablets, and IoT devices) that may be actively accessing the VMs services. Second, bandwidth between edge cloud sites is likely to be more constrained than LAN bandwidth, which increases migration costs. There has been some limited work on WAN VM migration [23, 36, 40] but these systems use network tunneling or require explicit support from the network, which limit broader deployment.

Our approach relies on multi-path TCP [32], which is now widely available in server [33] and mobile OS platforms (e.g. Linux [30], Android, Apple iOS [3]). Multi-path TCP relies on the presence of multiple network interfaces on clients and servers to send and receive TCP data in parallel over multiple network paths. We use MPTCP to speedup VM migrations and to mask the effect of IP address changes due to WAN VM migration.

Our approach also tackles many of the practical challenges presented in edge-cloud scenarios that are not addressed by existing work on VM migration. A key feature of our approach is that it relies solely on the end-hosts, and does not depend on support from the underlying network. Past work on maintaining network transparency during VM migrations relies on using tunneling and Software Defined Networks [36, 40]. In edge clouds that can span multiple independent heterogeneous networks, relying on the underlying network to provide such functionality may not be practical. The second challenge posed by edge clouds is the variance in the parameters of WAN links such as bandwidth, round-trip-time, and congestion. We study the effect of this variance, and show that our approach is robust in the face of network heterogeneity and variations.

In developing a multi-path approach to wide-area network VM live migration, we make the following contributions:

MPTCP-Based Virtual Machine Migration We show how multi-path TCP can improve live migration times by allowing the hypervisor to take advantage of the increased aggregate bandwidth. This enables our system to speed up the transfer of disk and memory state and complete migrations in about half the time of a regular TCP migration.

Network Transparency for VM Clients Traditional WAN migration processes rely on techniques such as network tunneling that increase the downtime and latency that end users perceive during VM migrations. This is not suited to edge cloud environments, where the goal is to migrate VMs to decrease client perceived latency. Our system lowers client perceived latency and increases after-migration throughput by using MPTCP to notify clients of the migrated VMs new network address and switching them to the new address while still maintaining active connections.

Wan-Optimized Pre-Copy Algorithm Edge cloud VM migrations for non-well provisioned sites may take on the order of 10's of minutes rather than the seconds of traditional LAN environments. As a result of longer migration times, traditional optimizations for pre-copy algorithms may not be suitable. We propose *hot-skip*, a new pre-copy algorithm optimized for edge cloud WAN environments that groups memory pages by historical write frequency, and reduces migration time by 20% compared to existing KVM live migration.

Prototype and Experimentation Across Public and Emulated Clouds

We implement our migration system as a prototype wrapper script for KVM hypervisor. We evaluate the performance of our prototype on the IBM SoftLayer network as well as in a lab based edge cloud emulator testbed for a wide variety of edge cloud conditions. We show that our system decreases migration times by 50% and improves after-migration throughput anywhere from 12% to 60% depending on client distance to edge cloud locations.

Thus, our approach provides a WAN VM migration that offers easy deployability, bandwidth aggregation, end-point transparency, and a WAN optimized pre-copy algorithm.

2. BACKGROUND

In this section, we present background on edge clouds, virtualization and migration, and multipath transport protocols.

2.1 Edge Clouds

An edge cloud provides cloud resources such as computing and storage from multiple locations at the edge of the network, specifically to provide low-latency and high bandwidth access to end users. As noted in Section 1, edge clouds are particularly well suited for mobile users and IOT devices and others (e.g. Cloudlets [39], fog computing [7], computational offloading [5, 14, 11]) have argued for such architectures.

In this paper, we assume an edge cloud architecture where clusters of storage and computation are deployed at hundreds or thousands of locations distributed across a large geographic region (Fig 1). We further assume that edge cloud platforms offer a similar abstraction to traditional centralized cloud platforms—a customer may request compute or

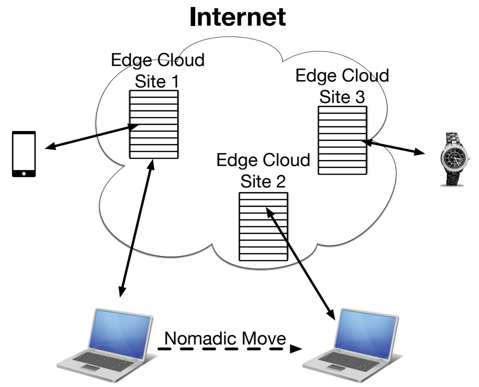

Figure 1: Computation and storage resources are located at multiple sites close to the users.

storage resources at one or more locations and the cloud platform will provide virtualized resources to the application at the chosen location(s). Unlike CDNs [29] that host and deliver content, an edge cloud can host any arbitrary application inside virtual machines and provide online services using computational and storage resources at the edge.

2.2 Virtualization in Edge Clouds

As shown in Figure 1, each edge cloud server is assumed to be virtualized and can host one or more virtual machines that run end-user applications. We assume that the "control plane" of the VMs extends across the edge cloud locations. That is, identity management, access control, security groups etc., can all be either migrated along with the VM, or are already available across all edge cloud locations via some consistent distributed storage. Given the large number of edge locations, initial placement of a customer's VM depends on the location of the end-users— typically VM(s) are placed at a nearby edge cloud site to reduce latency and maximize bandwidth from end-devices to the cloud-hosted application.

Edge cloud workloads, however, are expected to be more dynamic than traditional cloud workloads due to their focus on mobile users and devices. Mobile users are nomadic and thus change locations frequently. Similarly, IOT workloads may see skews in when and where interesting events may be observed, requiring computational resources to be dynamically provisioned at those locations.

An edge cloud can react to these dynamics by continuously adjusting the locations where application VMs are placed. This can be achieved by dynamically instantiating application replicas at new edge locations or by migrating an application's VM(s) from one edge location to another.

Provisioning additional replicas at new locations can be done similarly to current cloud platforms. However, migrating an application between edge cloud locations poses many challenges (and is not supported by today's cloud platforms). First, doing so involves migrating the VM's memory and disk state from one edge cloud location to another. While VM migration on a LAN (i.e. from one server to another at the *same* cloud location) is well understood [13], WAN VM migrations are not. Live VM migration in a LAN assumes that the IP address of the VM remains unchanged after the migration and thus all active network socket connections between client devices and the VM are unaffected. In WAN migration, the VM IP changes to that of the new cluster at the new site, which makes it challenging to maintain *network transparency* to end-clients. In an edge cloud where minimizing access latency is critical, client perceived downtimes during migrations also need to be minimized.

Bandwidth between edge cloud locations may be more constrained than LAN bandwidths and can vary widely depending on how network links between edge cloud clusters are provisioned. Lower bandwidths increase migration latency and can also increase data transfer overheads (since data that is actively changing may need to be re-sent multiple times). The problem is exacerbated in the WAN case since migration latencies are much larger.

Efficient support for VM migration between edge cloud locations needs to address these challenges.

2.3 Multipath Transport Protocols

Todays servers have multiple network interfaces. Similarly, end-devices such as mobile phones have multiple interfaces (e.g. WiFi, cellular, etc.) for communication. A multipath transport protocol exploits the presence of multiple network paths between two servers, or a server and a client. In this case, network packets between socket endpoints are sent over multiple network interfaces (rather than a single interface in a traditional socket) and the data can traverse over multiple paths to the destination (Figure 2).

Using multiple paths for a single connection has many advantages. The first is improved reliability, since network problems over one interface or path can be masked by other interfaces and paths. Multi-path transport protocols are also able to dynamically adapt to congestion and other path-health metrics, and send data on multiple paths based on their suitability. The second advantage is that multi-pathing provides link-aggregation and the ability to send data on the multiple paths. This increases the effective bandwidth available for data flows. Thus, applications can speed up data transfers by utilizing the bandwidths offered by multiple paths.

An implementation of multi-path network connections already exists in the form of MPTCP [33]—a multi-path version of TCP. MPTCP is a proposed [15] protocol extension to TCP. Many widely used operating systems such as Linux and Android now support MPTCP with a kernel patch, and Apple's iOS is deployed with built-in support for MPTCP. MPTCP is implemented as a set of TCP options, which allow two endpoints to simultaneously use multiple paths—also known as *subflows* between them [15]. These subflows are defined logically, by default, by all end-to-end interface pairs. For example, if each endpoint in a conversation has two interfaces, MPTCP can create up to four subflows. MPTCP has many benefits: First, it is transparent to user applications. The MPTCP layer is hidden from user applications by providing a standard TCP socket. Existing TCP applications need not be modified to take advantage of MPTCP. Second, by utilizing multiple subflows, MPTCP can natu-

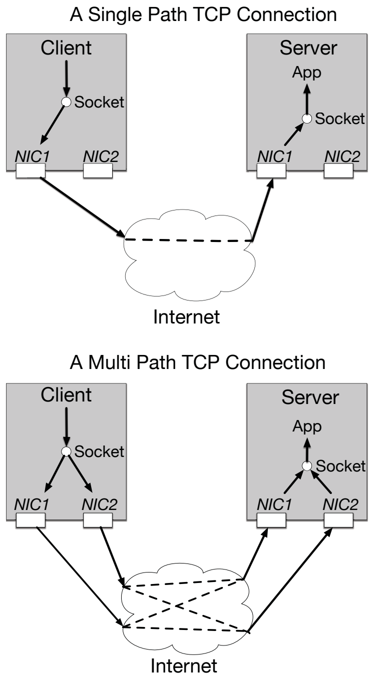

Figure 2: Standard TCP vs. Multi-Path TCP network connections. MPTCP allows a single socket connection to be multiplexed over multiple network interfaces, and allows a single connection over multiple paths.

rally achieve throughput equivalent to the best individual subflow, and sometimes even higher bandwidths. Finally, resilience is improved since no one path is a single point of failure. MPTCP can dynamically migrate traffic between paths in response to changing network conditions.

MPTCP communicates control information through new TCP header extensions, which announce multipath capability, available interfaces, and set up and tear down subflows. Once an MPTCP connection is initiated, each end host knows at least one of its peer's IP addresses. If a client wishes to create an additional subflow over another interface that it has (e.g., a cellular NIC), it can send another SYN packet with a *JOIN* option containing the IP of the NIC to the server's known IP address. This subflow becomes associated with the currently established MPTCP connection.

As many clients are behind Network Address Translators (NATs), it is difficult for the server to send a *JOIN* packet to the mobile client as NATs usually filter out unidentified packets [33]. Instead, if a server also has an additional interface, the server sends an *Add Address* option to inform the client of the available address. Once the client receives it, it can send another SYN packet with the *JOIN* option to the server's newly announced IP address, creating a new subflow [16]. In MPTCP, the client is defined as the endpoint that sent the original SYN packet. Our approach exploits MPTCP to address WAN migration challenges of network transparency and bandwidth constraints in edge clouds.

3. VM MIGRATION IN EDGE CLOUDS

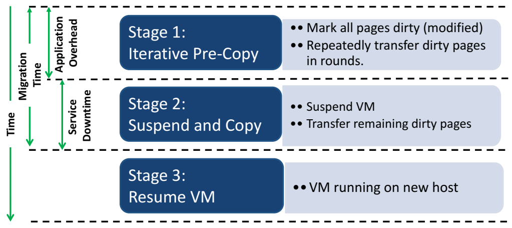
Time
Migration Time
Application Overhead
Service Downtime
Stage 1: Iterative Pre-Copy
•• Mark all pages dirty (modified)
•• Repeatedly transfer dirty pages in rounds.
Stage 2: Suspend and Copy
•• Suspend VM
•• Transfer remaining dirty pages
Stage 3: Resume VM
•• VM running on new host

Figure 3: VM Migration Stages

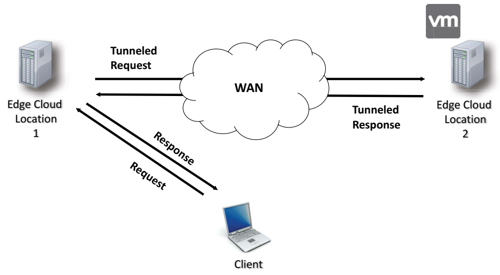

Figure 4: VM Migration using Tunneling

We now explain the inadequacies of current migration techniques for edge clouds and then present the intuition behind our approach, followed by our migration algorithm.

3.1 Why are existing VM migration techniques inadequate?

We now provide an overview of the state of the art in LAN and WAN VM migration and argue that these methods are inadequate for edge clouds.

LAN Migration. Figure 3 depicts the steps involved in LAN-based live VM migration. Live migration of memory is one of the key challenges in LAN migration [13, 27, 34], since in many scenarios the disk state of the VM is accessible at both the source and destination machines (e.g. via shared storage). The first stage of the migration is called iterative pre-copy, which iteratively copies the memory pages of the VM from the source to destination servers. Since the application continues to run uninterrupted during this stage, memory pages may get dirtied after they are copied. Hence, after each round, all newly dirtied pages are re-sent in the next round. The iterative process continues until the residual dirty state is "small"—at this point, the source VM is paused and the residual state is sent over. The destination VM can resume execution from where the source VM paused.

WAN Migration. In the WAN case, the IP address of the VM can not be retained, since the destination machine is part of a different network subnet; maintaining network transparency is therefore more challenging. Current approaches [23, 38, 40] all require extending the layer 2 network over the WAN. As shown in Fig 4, existing and future clients continue to send network traffic to the old IP address at the source machine, which forwards it over a network tunnel to the VM's new IP address at the destination. This triangle routing can add significant additional delay between the client and the VM, and reduces available bandwidth. This

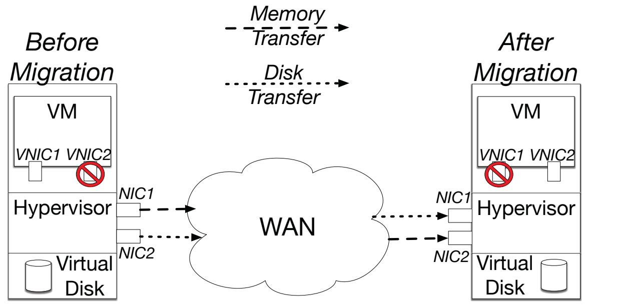

Figure 5: Edge Cloud Locations with multiple interfaces

is particularly detrimental in the edge cloud scenario since VM's may migrate to a new edge cloud location to reduce latency to end-users, while the triangle routing actually worsens it further.

Bradford et. al. [8] propose an approach that uses tunneling only for current clients and updates DNS records for the VM to allow future clients to directly connect to the VM's new IP address. However, DNS updates can be slow to propagate and existing long-lived network connections will still suffer from triangle routing and tunneling delays. CloudNet [40] uses VPN technology to keep the IP address of the VM unchanged at the destination, which eliminates these drawbacks. But it requires router support to implement its migration method, which limits general use.

Thus, current WAN migration methods either degrade client-edge cloud latency or require router support to maintain network transparency. Further, current approaches only focus on network transparency and have not considered the impact of lower WAN bandwidth (and larger migration latency). Larger latencies may increase the amount of dirty data and negatively impact the pre-copy stage of the migration. Finally, disk state of the VM may also need to be migrated in addition to memory state when shared storage is not available across edge cloud sites. These limitations motivate a new approach for VM migration that is tailored for edge clouds.

3.2 MPTCP-based Migration Approach

Our approach exploits multipath transport to overcome both the network transparency and bandwidth limitations of WAN VM migrations for edge clouds. Since MPTCP is now available on major server (e.g. Linux) and mobile (e.g. iOS) operating system platforms, our approach is easy to implement and deploy using current systems.

We assume that each edge cloud server has at least two network interfaces and cloud VMs hosted by these servers have two or more logical network interfaces. This is a reasonable assumption since most commodity server configurations ship with dual or quad port ethernet cards and VMs on commercial clouds have at least two logical network interfaces (e.g. public and private IPs for Amazon EC2 cloud VMs). We further assume client devices use MPTCP.

To address the bandwidth constraints, our VM migration techniques simply uses MTCP between the sender and the destination machines as shown in Fig 5. The use of multiple paths to migrate the memory and disk state *parallelizes* the data transfer and increases the bandwidth available during migration and reduces network latency. As shown in our experiments, MPTCP often doubles the available bandwidth during the migration.

However, even with the use of MPTCP, the WAN bandwidth available on the network paths between edge cloud sites may be more constrained than LAN bandwidth. Hence, we present a new pre-copy algorithm for migration that is tailored for WAN scenarios. Our algorithm tracks the dirty rates of memory pages (or disk blocks) across iterations and only sends the coldest pages. This decreases the length of iterations and reduces the chance that a page that was already sent was dirtied, thus reducing overall migration time. To further reduce data transfer overhead, especially for disk transfers, our algorithm can proactively copy disk snapshots to possible candidate locations and only send deltas in case the VM moves to those machines. In cases where a VM may migrate back to an old location, retaining previous disk state and sending a delta also saves migration overhead. Together, our WAN-optimized data copying algorithm and multipath transfers address bandwidth issues in edge cloud migrations.

We also use MPTCP to address network transparency after a WAN migration. Our technique uses two logical NICs on a VM, one active and the other inactive. All network connections use MPTCP and are configured to use both NICs– since only one logical NIC is active, only that one is actually used. After a migration, the other logical NIC is activated with a new IP address and the first NIC is then deactivated. Doing so causes MPTCP sockets at all client devices to seamlessly switch to using the new interfaces and stop sending over the old interface. Thus, all TCP connections remain active *despite an IP address change at the VM*.

3.3 Edge Cloud Migration Algorithm

We now present our migration algorithm. We assume that both the VM and the host machines have at least two NIC's. To avoid confusion we refer to the physical NICs on hosts as *NIC1* and *NIC2* and the logical NICs of the VM as *vNIC1* and *vNIC2*. Our edge cloud migration involves six phases:

Phase 1: *Tunnel creation and routing preparation*
As a first step, we create a tunnel between the source and destination hosts and prepare the routing tables for both incoming and outgoing traffic of the VM at the destination. Although this tunnel is not used until phase 4, we pre-create this network state to reduce client-perceived downtime when switching the VM from source to destination, which is important for maintaining network transparency. We specifically enable *proxy-arp*, create route entries to and from the VM and install `ebtable` rules for the tunnel and routing setup.

Phase 2: *Migrate disk state of the VM*
Next, we migrate the disk state (i.e. virtual disk) of the VM. This phase is necessary only if there is no shared storage between the two edge cloud sites. If shared disk storage (or a distributed file system) is available, the disk state can simply be shared with the new machine and need not be migrated.

The disk state, which can be substantial, is migrated by performing a block by block copy over a MPTCP connection between the source and destination hosts. Since the VM continues to execute, data blocks may be modified after they have been sent, and iterative copying will be necessary to resend modified disk blocks. The disk migration over slow WAN links can be optimized using three mechanisms. First, MPTCP itself parallelizes the transfers over multiple network paths and increases the bandwidth available for the migration. Second, by sorting disk blocks by their write frequency and sending blocks in increasing order of write frequency, we can reduce the data that needs to be retransmitted to the destination. Alternatively, blocks may be sent based on their dirty rate, with higher frequency blocks not being sent until the end. The write frequency or age of a block is derived from that of the file to which it belongs. Third, data deduplication [2, 18] techniques can be used to reduce the data transfer overheads.

Phase 3: *Migrate the Memory State of the VM*
Migration of memory state of the VM begins when most of the disk state has been migrated. [1] Our memory migration uses an iterative pre-copy approach that is tailored for WAN-based edge cloud environments by skipping the hottest pages to reduce the duration of iterations (Section 3.3.2).

Phase 4: *VM Switchover*
At the end of phase 3, both disk and memory state of the VM have been migrated. The VM at the destination then takes over and resumes execution from where the source VM was paused. The tunnel from the source to destination is activated.

At this point, clients continue to send network traffic to the VM's old IP address that is associated with the source machine and this traffic is forwarded over the tunnel to the VM at the destination. Thus, all network connections stay active, albeit with a larger round trip latency due to tunneling.

Phase 5: *MPTCP-based Network Switchover*
Next, the inactive second interface of the VM (*vNIC2*) is activated and given a new IP address belonging to the subnet at the destination edge cloud. Routing tables are also configured so both interfaces can be used simultaneously. Traffic to and from the second interface (*vNIC2*) does not traverse the tunnel. MPTCP naturally advertises the newly acquired IP address in the existing connections and creates new subflows. Once the new subflows are established, the other original interface, *vNIC1*, is deactivated and the tunnel is torn down. This causes all end-clients to send MPTCP traffic directly to the new IP address at *vNIC2*. Doing so optimizes latency from end clients to the VM's new edge cloud location.

Phase 6: *Normal Operation*
The VM is now back in a normal state, similar to where it was prior to the migration, except that it is now at a new edge cloud location and in a different subnet. If the VM needs to migrate again, the same process is used, except that roles of the two logical NIC's *vNIC1* and *vNIC2* are reversed.

3.3.1 Optimizing Disk Migration Overheads

Since a VM's disk state can be quite large, migration of the disk state can dominate the total migration latency. Phase 2 of our migration algorithm employs several optimizations to speed up the disk migration.

In scenarios where it is possible to predict the future locations of an edge cloud VMs or when the VM "hops" between the same set of locations, additional optimizations are possible. If the future location of a VM can be predicted, a snapshot of the VMs disk can be transferred to one or more candidate locations well in advance of the migration. In this case, only a delta of the disk state that was modified after the snapshot needs to be migrated in phase 2. Since only a small portion of disk files are modified over the course of hours or days, this can drastically reduce the migration of disk state.

Similarly, if the VM hops between known locations (e.g. follow the user, where a user moves from home to work and back or follow the sun scenario) then the disk state at the *source* machine is not deleted after migration. When the VM migrates back to this location, only a delta is copied back.

3.3.2 WAN-optimized Pre-copy Algorithm

The iterative pre-copy approach used in most hypervisors such as KVM is suited for LAN environments where the network bandwidths are high enough that the state of VMs can be copied over in under a minute. WAN bandwidth can be several orders of magnitude lower than inter-data center bandwidth. This reduction in bandwidth is particularly detrimental to the migration process.

In the pre-copy migration approach, the hypervisor iteratively identifies and sends the remaining dirty pages over the network. The page dirty rate is only bound by the memory bandwidth, which can be several hundred Gbps and thus several orders of magnitude higher than network bandwidth. To be able to completely migrate a VM, pre-copy approaches assume that the writable working set of a VM—the pages that are part of its working set that see frequent writes, is small.

The reduced WAN bandwidth thus increases the time required for migration, as the hypervisor takes longer to send dirty pages, thereby increasing the time between successive iterations. This increased iteration time has a compounding effect—with longer iterations, *more* pages are likely to be dirtied in the successive iteration. Thus the reduced bandwidth not only increases migration time due to slow network transfer rates, but also increases the amount of data (dirty pages) that is to be transferred.

To address the steep increase in migration time due to low bandwidths, we propose a modified pre-copy algorithm which reduces the pages sent (and hence the total migration time) in WAN environments. The key idea in our algorithm is to reduce the number of pages sent in an iteration, by not sending the "hot" pages. Hot pages are those that see a lot of write traffic, and are highly likely to be dirtied in the next iteration. Thus sending hot pages in initial iterations only increases the iteration time and the compounding effect of the increased number of dirtied pages in subsequent iterations. During each iteration, we identify the dirty pages, and divide them into two classes, hot and cold, and we only send the

[1] Disk and memory state is migrated simultaneously.

cold pages. This keeps iterations short and reduces total migration time. We call our approach the "hot-skip" approach because it skips and does not send the hot pages. To ensure that the migration process converges, we ensure that the number of skipped hot pages is monotonically decreasing. Hot-pages are identified either using a thresholding scheme (top 10% most frequently written pages are hot), or by using other memory-profiling techniques [41] which allows the threshold to be determined based on the application behavior.

We identify hot and cold pages based on their *long-term* write frequency, which we record across all iterations based on the dirty bitmap. Our approach is similar to the skip-list approach used in Xen's live migration [13, 25] that skips twice-dirtied pages in an iteration. However, our approach is different in the sense that we consider long-term history when classifying pages as hot and cold. In contrast, skip-lists "forget" a page's history after the current iteration.

3.4 System Implementation

We have implemented a prototype of our transparent WAN migration system in the KVM hypervisor. Our implementation is split into three main parts: the hypervisor, a migration manager, and the guest operating system in the VM.

Hypervisor. Our migration approach relies on using MPTCP for migrating VM state. Since the hypervisor (KVM version 2.0 in our case) performs the actual migration of the VM, it must have the ability to use MPTCP-backed sockets. Migrations in KVM are performed by the userspace QEMU layer, and open a standard TCP socket. QEMU runs on top of Linux, and we use an MPTCP enabled Linux kernel (version 3.13) by using kernel patches for MPTCP version 0.91 [30]. This allows the QEMU migration process to use MPTCP and to be able to send the VM state over multiple network interfaces and increase the effective bandwidth.

We observed that MPTCP throughput is highly dependent on the TCP window sizes, much more so than standard TCP [32]. We tune the kernel TCP window buffers, `net.ipv4.tcp_rmem` and `net.ipv4.tcp_wmem`, according to RFC 6824 [16] with a value of 60 MB which is roughly twice the sum of the available bandwidths multiplied by the highest RTT value. We do not use *MPTCP* checksumming so as to reduce its CPU overhead [33]—this does not affect checksumming of the *TCP* subflows, which are still checksummed. We use the default MPTCP coupled congestion control algorithm, LIA (Linked Increases Algorithm). LIA is based on TCP Reno, and we use Reno for TCP congestion control to provide a fair comparison.

Migration Manager. The migration manager is a user-space software component that runs on both the source and destination hosts. It interacts with the QEMU migration thread and is primarily responsible for updating the network configuration at both ends to ensure network transparency.

The migration manager is implemented as a wrapper perl script around the regular migration process. Our script reads a configuration file with the VM and networking details. The configuration file includes the following details: the source and destination host IP addresses, the destination hostname, the virtual machine name, the VM's current networking details (IP address, gateway, interface name), and the VM's final networking details (IP address, IP prefix, interface name). We then proceed through the steps outlined in Section 3.3. We use the VM networking details to establish `ebtable` rules for routing packets once the VM is at the destination server. During stage 2, we use the destination host IP address to establish the disk state at the destination by precopying a snapshot. In stage 3, we use the VM name and destination IP to migrate the VM using virsh. In stage 4, a time critical step, we use the VMs network configuration with ebtables [1] to change the network routes so that packets are forwarded through our pre-established tunnel. Finally in stage 5, we use the specified network interfaces to decide which interface to activate and deactivate after migration completes.

Virtual Machine Guest OS. Finally, our approach also requires that the guest operating system use MPTCP, and for that we again use an MPTCP patched Linux kernel. This ensures that all applications running inside the VM use multiple vNICs and MPTCP. Other than an MPTCP-enabled kernel, our approach requires no other application modification.

Our approach requires that both the hypervisor and the VM have multiple network interfaces. The hypervisor can easily create multiple vNICs for each VM. Since we use MPTCP in the VM for endpoint-transparency, we note that the VM does not actively use the multiple interfaces during regular operation. Instead, we use MPTCP's "backup mode" so that the traffic of the TCP connections falls back to the alternate interfaces during the migration.

Our implementation was successfully tested on EC2, Amazon, the IBM SoftLayer cloud, as well as our lab based emulated edge cloud testbed.

4. EVALUATION

In this section, we present our results when MPTCP is used by the hypervisor and the VM as the default transport protocol for network activity. We first describe our experimental environment including the commercial distributed cloud, emulated edge cloud testbed, hardware, software and metrics. We then present our evaluation of MPTCP. Our experiments are designed to evaluate the following aspects of VM State migration over MPTCP for edge clouds:

- How quickly can we migrate memory and disk state?
- How do varying bandwidth and RTTs affect migration?
- What are the benefits of our proposed optimizations for reducing memory and disk copy overheads?
- What is the effectiveness of our MPTCP-based switchover while maintaining network transparency?

4.1 Environment and Methodology

We shall use the following metrics :

- *Application throughput:* For network intensive applications, we are interested in seeing what effect our migration approach has on their throughput.
- *Migration time:* The total time the migration process takes. This dictates how quickly VMs can completely migrate to

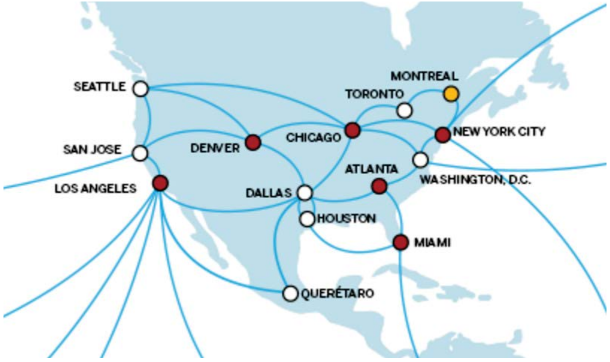

Figure 6: IBM SoftLayer cloud data center locations in North America. We migrate VMs from Montreal to either San Jose or Dallas.

different edge cloud locations.

• *Application downtime:* Since migration can cause a temporary loss in network connectivity, it can cause the application to face downtimes. We evaluate the downtime by using packet traces during the migration obtained using tcpdump. We define the start of the downtime as the time we observe the last packet sent by the VM that does not traverse the tunnel, and the end of the downtime as the time we observe the first packet sent from the VM on the tunnel.

• *Loss Rate:* In addition to downtime, we also measure the packet loss rate during migration to quantify the disruptive effects of migration.

Workloads. We use two approaches to evaluate our system's performance at the host and guest levels. To evaluate throughput performance, we migrate a 30 GB VM with 16 GB RAM over a WAN while running a memory workload. To evaluate network transparency, we migrate a 15 GB VM over the wide area while running a network workload.

To evaluate migration performance in our emulated edge cloud, we run a workload that allocates a block of memory about the size of the VM's virtual memory (for a 16 GB VM, we use 15 GB) which dirties a significant amount of memory pages to force the hypervisor to have to transfer the entirety of RAM over the network. Unless otherwise stated, we run 10 trials each with MPTCP and TCP and present averages. For our experiments evaluating network transparency, we run an iperf client in San Jose while the VM hosts the iperf server. We run experiments using both our approach and the tunneling approach used by most existing WAN migration systems. To minimize the effect of congestion we perform our experiments at night from 9:00 PM to 7:00 AM. We report the average of 10 runs and 90 percent confidence intervals.

Commercial Distributed Public Cloud. We provision several bare-metal machines from IBM's SoftLayer cloud, pictured in Figure 6, and deploy our MPTCP-enabled hypervisor and migration manager. While the SoftLayer cloud is not a full-fledged edge cloud, it is nevertheless a distributed commercial cloud offering multiple sites in North America from which we picked locations in San Jose, Dallas, Toronto, and Montreal to function as our edge cloud locations. We use a separate bare metal machine in San Jose as our client and the VM always starts in Montreal. We examine scenarios where the VM is migrated to Dallas and San Jose. Our SoftLayer edge clouds have two 2.4 GHz Intel E5-2620 processors with hyperthreading enabled. The systems have 64 GB RAM and 1 TB SCSI disk, and two Intel 10-GigE NICs.

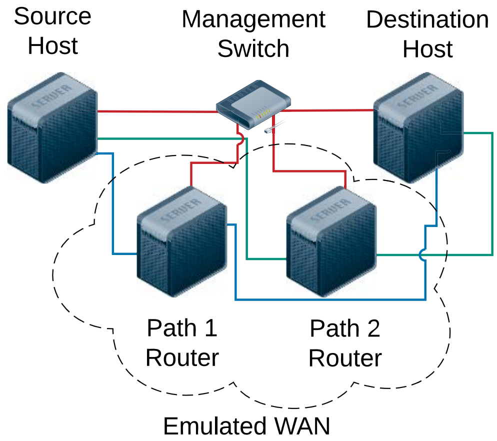

Figure 7: Lab-based Edge Cloud Testbed. A WAN is emulated by connecting two servers to two routers running dummynet.

Lab-based Edge Cloud Testbed. Since network bandwidth and delay as well as VM allocation are hard to control in a real production environment, we set up two controlled edge sites connected by a WAN emulated by Dummynet [10]. This allows us to control the bandwidth, loss, and delay while evaluating our system in a controlled environment.

The emulated Edge Cloud environment consists of four PowerEdge R430 servers with 2.10GHz Intel E5-2620 v4 processors with two sockets and eight cores with hyperthreading enabled. Each system has 64 GB RAM and one TB SATA disks. We use two one GigE interfaces on each machine. Our network topology is illustrated in Figure 7.

We emulate WAN routers on two servers running FreeBSD 11.0 - STABLE and the Dummynet network emulator with queue sizes of 8000 and 12000 KBytes. Our edge cloud locations plug directly into the routers, which forward all traffic to their destinations. As packets flow through the routers, dummynet applies preconfigured rules to alter the bandwidth and delay for the tcp connections. We use these rules to configure our desired network conditions.

4.2 Migrations in Distributed Public Cloud

Migrating a VM over MPTCP increases the effective bandwidth available for migration, since it enables the transfer of VM memory and disk state over multiple interfaces. Thus, the choice of transport protocol used by the hypervisor for VM migration can impact VM migration times.

We evaluate the migration time in our distributed public cloud environment in Figure 8, which shows the migration times when a VM of size 16 GB is moved between the Montreal and San Jose locations. We performed a total of 195 migrations over a three day period. Each VM migration has a choice of two paths (via the public and the private net-

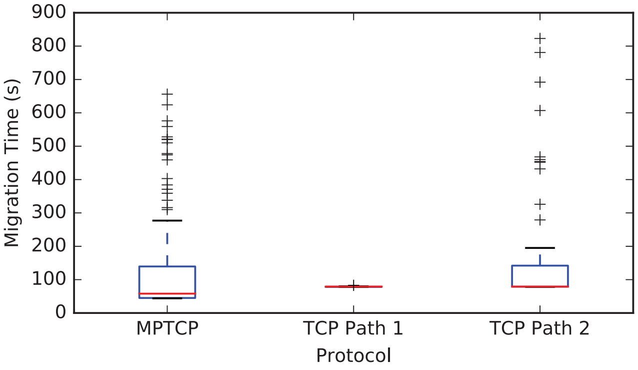

Figure 8: Migration times in the distributed public cloud show high variance due to variable network bandwidths between geographically separated data centers.

work interface respectively). MPTCP is able to use *both* these paths, and is also able to reduce the mean migration time by almost 30% compared to using TCP alone.

We see that TCP on a single path (TCP Path 1 in Figure 8) can sometimes outperform MPTCP if the path was not congested, and sending data on this uncongested path using TCP overshadows some of the link-aggregation benefits of MPTCP. We emphasize the adaptive nature of MPTCP—while TCP on the manually selected "best" path may be sometimes competitive with MPTCP, selecting this best path may not be practical in edge clouds that see a high variance in network characteristics. MPTCP is adapative in nature and requires no manual path selection since it adaptively sends more data on uncongested paths—we can see this from the reduction in worst-case migration times compared to the congested path (TCP Path 2) in Figure 8.

4.3 Migrations in Lab-based Edge Cloud

Since network bandwidth and latencies in a globally distributed edge cloud can vary over a wide range, we also evaluate our migration approach in a lab-based edge cloud testbed which emulates a wide range of network behavior. Our setup allows us to run VM live migrations over an emulated wide-area network similar to what you would see in an edge cloud setup. We emulate network conditions of a WAN in our local network by using the dummynet [10] emulator to control bandwidth and latency.

Network Latency. We first analyze the effect of network latency on migration times. Network latency can vary significantly in an edge cloud since edge cloud locations can be globally distributed, and the network latency is proportional to speed-of-light delays between locations.

Figure 9 shows the migration time over a wide range of round trip times (RTTs). We see that migration time with MPTCP is roughly half of that with TCP—this is because MPTCP utilizes both the physical network interfaces and effectively double the available bandwidth. The migration time with MPTCP is around 150 seconds compared to almost 300 seconds with TCP. For both MPTCP and TCP, the effect of RTT on migration time is negligible. Migration time depends on network throughput, since it involves bulk data transfers—the hypervisor sends large batches of 4 KB pages on every iteration. Since both transport layer protocols (MPTCP and TCP) are able to achieve the same network throughput regardless of the round trip times, the migration times are unaffected by the RTTs.

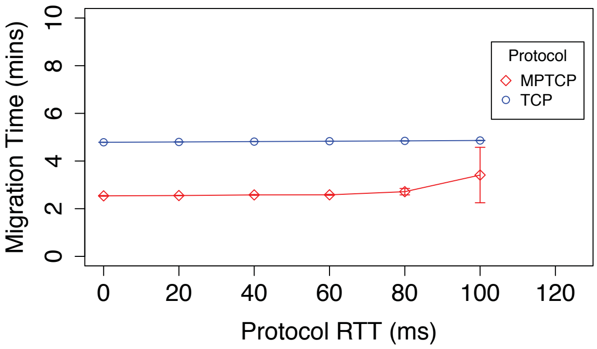

Figure 9: Round trip times have no significant effect on migration times. MPTCP is able to utilize both the network interfaces and results in lower migration times.

Network Bandwidth. Network bandwidth between different locations of an edge cloud can also vary significantly. We evaluate the effect of available bandwidth on the migration times in Figure 10, with RTT kept constant at 60ms in all cases. At a maximum one Gbps bandwidth, VM migrations are about 1.2X faster with MPTCP. However, MPTCP shows a pronounced reduction in migration times as the available bandwidth declines. At 100 Mbps, migrations take about 15 minutes compared to 28 minutes with TCP—a significant reduction of almost 2X. We emphasize that network bandwidths can be quite low in edge clouds because of geographically dispersed locations and inter-continental bandwidth is expensive, thus minimizing bandwidth requirements is a primary optimization goal when designing applications and systems for edge clouds. In bandwidth constrained environments, MPTCP allows VMs to be migrated in half the time that existing TCP-based approaches allow.

VM Size. Finally, the memory size of a VM can vary due to the varying memory requirements of different applications. We evaluate the impact of VM memory size on the migration time in Figure 11. MPTCP allows 2x faster migrations for VMs larger than 8 GB or more, when compared to using TCP. We evaluate TCP on both interfaces individually, and our results in Figure 11 show no noticeable difference in the migration times due to the choice of interface. This indicates that in our emulated network setup, MPTCP outperforms TCP even over the *best* path.

4.4 WAN Optimized Pre-copy Algorithm

We next evaluate our WAN-optimized pre-copy algorithm described in Section 3.3.2. To evaluate the viability of our algorithm, we implement a model driven [25] virtual machine

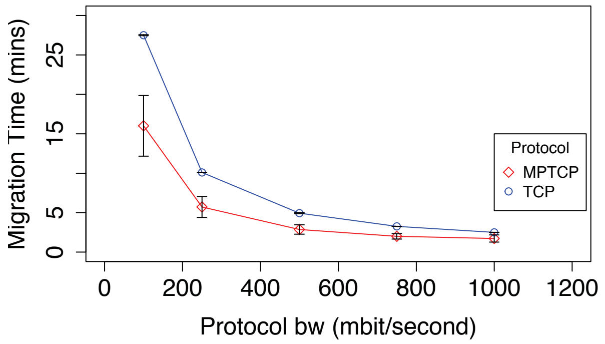

Figure 10: Migration time is roughly inversely proportional to network bandwidth. MPTCP yields between 1.2-2X reduction in migration times.

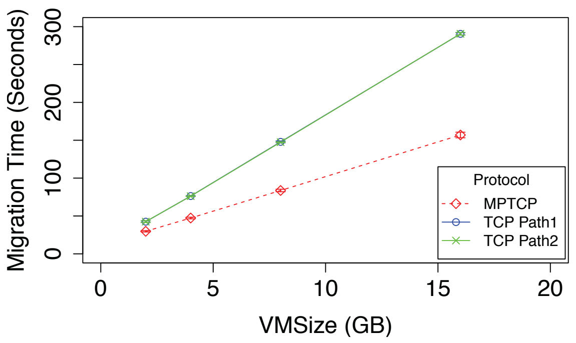

Figure 11: Migration Time vs VM Size. MPTCP allows 2x faster migrations for all large VM sizes.

migration simulator in python. The simulator evaluates both our *hot-skip* approach and the default KVM sequential algorithm by calculating the fraction of dirty and non-dirtied pages of each migration round. The simulator then computes the length of the iteration by taking the number of pages to transfer over the available bandwidth. It performs these iterations until a given stop and copy threshold is reached.

Figure 12 shows the migration times for different available network bandwidths for our wan-optimized hot-skip pre-copy algorithm. Compared to the default QEMU migration algorithm which sends all dirty pages in an iteration, our approach of skipping hot pages reduces the total migration time by more than 20%. We also observe from Figure 12 that the reduction in migration time with our hot-skip approach is more pronounced in low bandwidth scenarios.

4.5 Disk Migration

So far we have looked at the migration times of the memory state of the VMs, and assumed that the virtual disk of the VM is shared between the source and destination edge cloud locations. Sharing disk state is common in environments where a virtual disk is network mounted on both locations. However, there are scenarios where the disk contents (along with the memory contents) of a VM also need to be migrated. Once the virtual disk is migrated, the migrated VM can access it on the faster local disk on the destination, instead of incurring a network round-trip for every disk access. Thus disk migration is important in edge clouds to minimize latency.

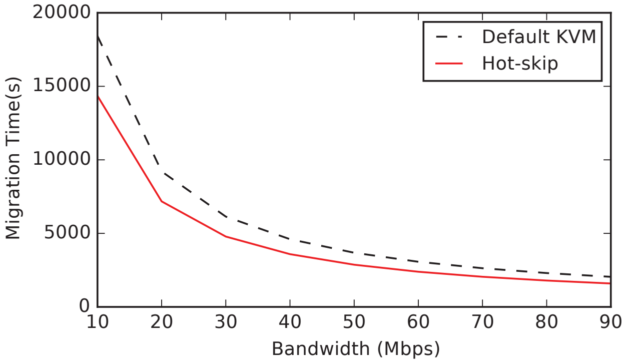

Figure 12: Our hot-skip pre-copy algorithm allows migrations to finish more than 20% faster, especially in low-bandwidth environments.

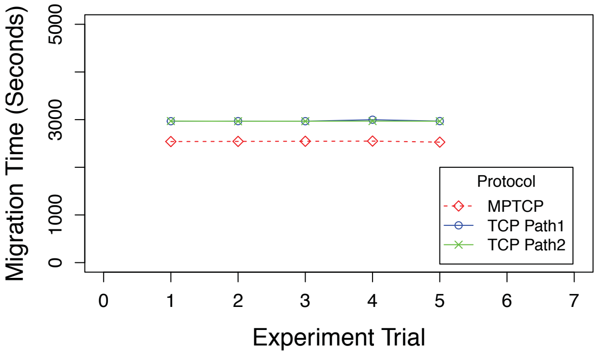

Figure 13: Disk migration time with full-disk-copy.

We evaluate disk migration by copying the entire virtual disk of the VM along with memory state. In this *full-disk-copy* approach, the migration times can significantly increase due to the large amount of data transfer, since the size of virtual disks can be in tens (or even hundreds) of GBs. We observe that using MPTCP consistently lowers migration times by 15% or *almost 10 minutes*. We note that disk migration is bounded by the disk bandwidth which can be significantly lower than the network bandwidth, especially in a WAN environment. While MPTCP improves effective network bandwidth, its effectiveness is reduced because of the bottleneck caused by the low disk bandwidth.

We note that a further optimization can be made by using a technique known as *incremental-disk-copy* where snapshots are migrated prior to live migration and used as a base image. This allows the live migration process to transfer only the delta's between the current image and the snapshot, thus reducing the amount of data being sent over the network.

4.6 Network Transparency

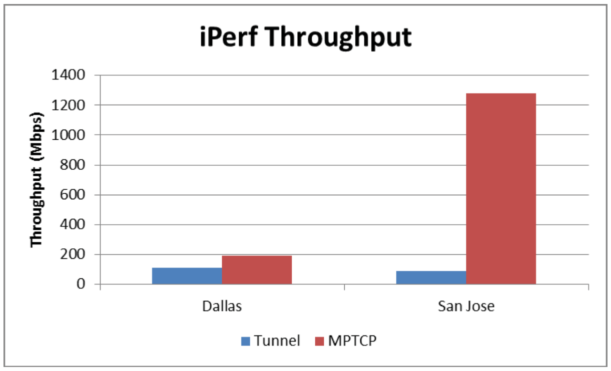

Figure 14: Iperf throughput when migrating from Montreal to Dallas and San Jose. MPTCP achieves almost 2x throughput to Dallas and 10x throughput to San Jose after migration compared to tunneling.

We now look at how our approach handles network transparency and the effect it has on throughput. We evaluate migration performance when the VM is migrated from Montreal to two different locations (San Jose and Dallas). In both scenarios, the client is located in San Jose while the VM is running the iperf server.

Figure 14 show our results when the VMs are migrated to the Dallas and San Jose edge sites respectively. We observe that after migration, application throughput when using MPTCP is larger than what is achieved using a tunnel-only solution. Both TCP and MPTCP yield approximately the same throughput before and during the migration process. However after the VM has been migrated to the destination, there is a stark contrast in the throughput. When migrated to Dallas, throughput with MPTCP is 1.8x that of TCP. Similarly, after migrating to a different location (San Jose), the difference in throughput is more than 13x.

We note that migration time, service downtime, and throughput before and during migration are statistically identical under both approaches. This is expected since both techniques are identical until the point that the client opens a new MPTCP subflow with the VM using its new IP address. We only see the performance and transparency benefits of our MPTCP system after migration completes.

5. RELATED WORK

Our work is based off of and combines prior work on VM live migration, multi-path TCP, and edge clouds.

Live Migration. Modern virtual machine live migration was first presented in [13], which described Xen's iterative pre-copy migration algorithm that also forms the basis of our approach. Live migration is implemented in almost all hypervisors, including VMWare ESX [27].

WAN Migration. Migrating virtual machines over a wide area network imposes additional challenges due to increased network latencies and reduced bandwidths compared to LAN environments. We also show that highly variable WAN bandwidth presents additional challenges in migration. Wide-area live migration of virtual machines is described in [40], which also developed various optimizations such as page and block compression to reduce network data transfer. Wide-area live migration has also been used to move VMs between cloud providers [36] to lower costs. Network transparency in wide-area migration is maintained by using techniques such as tunneling and software-defined networks. VMWare's Xv-Motion can also migrate VMs over long distances, and uses network virtualization techniques to extend the layer 2 LAN across two data centers. Tunneling and network virtualization impose additional network latency since client requests are first routed to the original source location of the VM. Moving VMs closer to the clients to reduce latency is an important feature of edge-clouds, and tunneling *increases* latency since it adds an additional wide-area hop between the client and the new destination. In contrast, our approach relies on using MPTCP without requiring any tunneling, and thus it can *reduce* client latency by moving VMs closer to the clients.

Migration with MPTCP. Some prior work has proposed using MPTCP for VM migrations. [28] shows how to use MPTCP to change the endpoints of the connection in the context of middleboxes, and also proposes WAN VM migration as a *potential* use-case. Their approach relies on changing the MPTCP stack—unlike our approach which requires no protocol changes.

Previous work [24] looks at combining hierarchical token bucket scheduling with multi-path TCP and SDNs while looking at VM migration. Their emphasis is solely on migrations within a LAN which does not apply to the WAN context of edge clouds since it can not provide transparency when the VM's IP address changes across sites. Our approach is tailored for WAN migrations between edge cloud locations—we provide network transparency, optimize the VM pre-copy migration for WAN environments, and also exploit MPTCP for virtual machine disk transfers. [37] has looked at VM migrations with MPTCP and pre-assigning IP addresses to lessen downtimes. They rely on a VPN for migrations among non collaborative edge sites where as our approach functions over the public internet between edge cloud locations. We enable MPTCP in both the VM and the hypervisor to enable faster migration times, provide more network transparency and to optimize migrations for WAN environments.

Migration Optimizations. Virtual machine live migration has two important metrics: the total migration time and the downtime. A comprehensive empirical model of these metrics is provided in [25], which also identifies factors, such as application page dirty rates, that affect migration performance. The performance benefits of migration optimizations such as page deduplication and compression are evaluated in [26]. Recently, migration optimizations have been proposed that use application hints (such as Java garbage collection) to reduce the pages sent during iterative pre-copy [9, 20]. The overhead of storage migration can be reduced by scheduling the transfer of storage blocks and taking advantage of spatial and temporal locality [42].

Edge Cloud Migrations Live migration has found use in many optimization tasks in edge clouds, and is one of the fundamental mechanisms for managing applications in edge clouds. [19] examines using live migrations to perform seamless handoffs between edge cloud locations as users move. This approach relies on leveraging VM state at the destination as well as reducing overall data sent across the network through techniques like deduplication, compression and VM synthesis. Our system with MPTCP does not rely on existing state at the destination host and as such can be used to lower migration time when migrating to new destinations. VM Handoff can be viewed as complementary to our system and may result in even lower migration times when used in combination.

Multi-Path TCP. Multi-path TCP is proposed in [16], and MPTCP has been used and evaluated in many contexts. MPTCP in data center networks has been evaluated in [33, 32]. MPTCP has also found use in mobile environments, where it allows multiple interfaces such as WiFi and cellular network to be used [31, 12, 22]. Our use of multi-path TCP is novel in the sense that it *combines* both the throughput and the network transparency advantages that MPTCP provides. That is, we reduce the migration time because of the improved bandwidth available under MPTCP as well as reduce downtimes by instantly switching over to the other interface.

Some prior work also leverages MPTCP to achieve endpoint transparency. [6] proposes a MPTCP enabled proxy and middlebox for address transparency. We rely only on end-host support and do not require middleboxes. The use of Software Defined Networks has been proposed to provide some of the address-transparency aspects of MPTCP [4]. However, our MPTCP-based approach does not depend on the underlying network for address transparency.

6. CONCLUSION

In order to provide user mobility and nomadicity, edge clouds must rely on VM live migration that is optimized for WAN scenarios. Our migration approach that is tailored to edge cloud environments, takes advantage of MPTCP to speed up migrations and provide increased network transparency for clients. We utilize the aggregate throughput provided by MPTCP to parallelize migration, thus reducing migration time. We also lower latency after migration by relying on MPTCP's resilience properties to automatically switch active network connections to the VM's new address, thus removing the need for a persistent tunnel. Our system introduces a new iterative pre-copy algorithm for disk and memory, optimized for the lower bandwidths of edge clouds.

We have demonstrated our prototype's performance on both a distributed public cloud and a lab based edge cloud. Our experiments show that our prototype reduces migration time by up to 50% and in some scenarios increases clients after-migration throughput by almost 6x compared to typical tunneling approaches.

Acknowledgements. We thank all the reviewers for their insightful comments, which improved the quality of this paper. This work is supported in part by NSF grants #1422245 and #1229059, by Amazon Web Services (AWS) and Microsoft Azure Cloud Credits as well as in part by a gift from Huawei and by the US Department of Energy under Contract DE-AC02-06CH11357. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

7. REFERENCES

[1] Linux Ethernet Bridge Firewalling. http://ebtables.netfilter.org/, December 2016.

[2] A. Anand, V. Sekar, and A. Akella. Smartre: an architecture for coordinated network-wide redundancy elimination. In *SIGCOMM*. ACM, 2009.

[3] Apple Corporation. iOS: Multipath TCP support in iOS 7. https://support.apple.com/en-us/HT201373.

[4] D. Banfi, O. Mehani, G. Jourjon, L. Schwaighofer, and R. Holz. Endpoint-transparent multipath transport with software-defined networks. In *Local Computer Networks (LCN), 2016 IEEE 41st Conference on*, pages 307–315. IEEE, 2016.

[5] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. In *INFOCOM*. IEEE, 2013.

[6] Y. Benchaïb, S. Secci, and C.-D. Phung. Transparent cloud access performance augmentation via an mptcp-lisp connection proxy. In *Architectures for Networking and Communications Systems (ANCS)*, pages 201–202. IEEE, 2015.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the Internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[8] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *Virtual Execution Environments (VEE)*. ACM, 2007.

[9] R. Bruno and P. Ferreira. Alma: Gc-assisted JVM live migration for Java server applications. In *Middleware Conference*, pages 19–20. ACM, 2016.

[10] M. Carbone and L. Rizzo. Dummynet revisited. *SIGCOMM*, pages 12–20.

[11] A. Chandra, J. Weissman, and B. Heintz. Decentralized edge clouds. *IEEE Internet Computing*, 17(5):70–73, 2013.

[12] Y.-C. Chen, Y.-S. Lim, R. J. Gibbens, E. Nahum, R. Khalili, and D. Towsley. A measurement-based study of multipath TCP performance in wireless networks. In *Proc. of ACM IMC*, pages 455–468, Nov 2013.

[13] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. of USENIX NSDI*, 2005.

[14] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making smartphones last longer with code offload. In *International conference on Mobile systems, applications, and services*. ACM, 2010.

[15] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar. Architectural guidelines for multipath TCP development. RFC 6182, Mar. 2011.

[16] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP extensions for multipath operation with multiple addresses. RFC 6824, 2013.

[17] T. Guo, V. Gopalakrishnan, K. Ramakrishnan, P. Shenoy, A. Venkataramani, and S. Lee. Vmshadow: optimizing the performance of latency-sensitive virtual desktops in distributed clouds. In *MMSYS*. ACM, 2014.

[18] D. Gupta, S. Lee, M. Vrable, S. Savage, A. C. Snoeren, G. Varghese, G. M. Voelker, and A. Vahdat. Difference engine: Harnessing memory redundancy in virtual machines. *Communications of the ACM*, 53(10):85–93, 2010.

[19] K. Ha, Y. Abe, Z. Chen, W. Hu, B. Amos, P. Pillai, and M. Satyanarayanan. Adaptive vm handoff across cloudlets. 2015.

[20] K.-Y. Hou, K. G. Shin, and J.-L. Sung. Application-assisted live migration of virtual machines with Java applications. In *EuroSys*. ACM, 2015.

[21] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang. Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In *INFOCOM*. IEEE, 2012.

[22] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, R. J. Gibbens, and E. Cecchet. Design, implementation, and evaluation of energy-aware multi-path TCP. *CoNEXT15*, 2015.

[23] A. J. Mashtizadeh, M. Cai, G. Tarasuk-Levin, R. Koller, T. Garfinkel, and S. Setty. XvMotion: Unified virtual machine migration over long distance. In *Proc. of USENIX Annual Technical Conference*, 2014.

[24] R. Nasim and A. J. Kassler. Network-centric performance improvement for live vm migration. In *Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on*, pages 106–113. IEEE, 2015.

[25] S. Nathan, U. Bellur, and P. Kulkarni. Towards a comprehensive performance model of virtual machine live migration. In *SoCC*. ACM, 2015.

[26] S. Nathan, U. Bellur, and P. Kulkarni. On selecting the right optimizations for virtual machine migration. In *VEE*, 2016.

[27] M. Nelson, B.-H. Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *Proc. of USENIX Annual Technical Conference*, 2005.

[28] C. Nicutar, C. Paasch, M. Bagnulo, and C. Raiciu. Evolving the Internet with connection acrobatics. In *Proceedings of the SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization*. ACM, 2013.

[29] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai network: A platform for high-performance Internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[30] C. Paasch and S. Barre. Multipath tcp in the linux kernel, available from http://www.multipath-tcp.org.

[31] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring mobile/WiFi handover with multipath TCP. In *Proc. of ACM Cellnet*, pages 31–36, 2012.

[32] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath TCP. In *Proc. of ACM SIGCOMM*, 2011.

[33] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley. How hard can it be? Designing and implementing a deployable multipath TCP. In *Proc. of USENIX NSDI*, 2012.

[34] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum. Optimizing the migration of virtual computers. In *Proc. of USENIX OSDI*, 2002.

[35] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[36] Z. Shen, Q. Jia, G.-E. Sela, B. Rainero, W. Song, R. van Renesse, and H. Weatherspoon. Follow the sun through the clouds: Application migration for geographically shifting workloads. In *SoCC*. ACM, 2016.

[37] F. Teka, C.-H. Lung, and S. A. Ajila. Nearby live virtual machine migration using cloudlets and multipath tcp. *Journal of Cloud Computing*, 5(1):12, 2016.

[38] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. De Laat, J. Mambretti, I. Monga, B. Van Oudenaarde, S. Raghunath, and P. Y. Wang. Seamless live migration of virtual machines over the MAN/WAN. *Future Generation Computer Systems*, 22(8):901–907, 2006.

[39] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt. Cloudlets: Bringing the cloud to the mobile user. In *Workshop on Mobile cloud computing and services*. ACM, 2012.

[40] T. Wood, K. Ramakrishnan, P. Shenoy, and J. Van der Merwe. CloudNet: Dynamic pooling of cloud resources by live WAN migration of virtual machines. In *VEE*. ACM, 2011.

[41] W. Zhao, Z. Wang, and Y. Luo. Dynamic memory balancing for virtual machines. *ACM SIGOPS Operating Systems Review*, 43(3):37–47, 2009.

[42] J. Zheng, T. S. E. Ng, and K. Sripanidkulchai. Workload-aware live storage migration for clouds. In *VEE*. ACM, 2011.