Experiences Implementing Live VM Migration over the WAN with Multi-Path TCP

Franck Le
IBM T.J. Watson Research Center
Yorktown Heights, New York, USA
fle@us.ibm.com

Erich M. Nahum
IBM T.J. Watson Research Center
Yorktown Heights, New York, USA
nahum@us.ibm.com

***Abstract*—Live VM Migration allows a running virtual machine or service to be moved from one host to another without the need to be shut down. This critical process offers many benefits for Internet services, including load balancing and service availability especially during host maintenance. Nevertheless, VM migration has been limited to layer 2 environments where the VM's IP address can be migrated with the VM. This is because the IP address must remain reachable after the migration. This necessarily restricts the ability to migrate VMs, limiting their potential and utility.**

In this paper, we show how a new Internet standard, Multi-Path TCP, can be used to seamlessly migrate live VMs across WAN boundaries. This allows services to migrate closer to their clients while preserving active TCP connections, improving performance, responsiveness, and user engagement. We show this by designing and implementing LSM-MPTCP , a system for VM Migration over the WAN. We demonstrate how our approach can improve throughput and latency in a real cloud environment, achieving throughput improvements up to 6 times and reducing round-trip times by 99%. We also expose subtle networking issues related to migration that can heavily affect loss rates.

I. Introduction

Live Virtual Machine (VM) migration [1]–[3] allows a running VM to move from one physical machine to another without having to shut it down, a critical technology for maintenance and service robustness in today's data centers. This feature provides load balancing and service availability when machines need to be serviced or upgraded.

Extending VM migration over the Wide Area Network would enable a number of new use cases. For example, IBM's UStream video delivery service treats TCP connection resets as a KPI to be minimized, and some connections can last days [4]. WAN migration also allows "follow-the-sun" [5] support across the continent without breaking an established TCP connection. It even allows local cases in the data center where two different subnets are used, a typical case in incremental build out. For example, the IBM SoftLayer Dallas data center has several separate facilities with disjoint subnets [6].

While VM migration is well understood inside the data center, it is much more difficult to achieve it in the wide area. This is because migrating a VM from one physical host to another in the same subnet does not require any update to the VM's IP address, allowing existing TCP connections to be preserved. In contrast, in the WAN, migrating a VM from one physical host to another machine belonging to a different subnet requires the VM to obtain a new IP address in the new subnet. This requirement affects service reachability. In particular, the change of IP address makes existing TCP connections unreachable, causing timeouts and resets, degrading the overall service offered to clients.

To address the problem, several proposals have suggested extending the layer 2 network over the WAN [7]–[10]. However, such an approach can suffer significant network performance degradation, as all existing and future traffic must be tunneled back to the source data center. This triangular routing can significantly increase delay, and lower available bandwidth, especially in the wide area. Another approach consists of tunneling packets (at layer 3) from the original subnet to the new subnet while a new DNS entry is distributed over the wide area [11]. As with the previous solutions, this approach suffers from triangular routing. It also creates an additional point of failure, since traffic must continue to traverse the original host.

Recently, Nicutar *et al.* [12] proposed how to use MPTCP [13]–[15] in the guest VM to allow wide-area VM migration. MPTCP allows a VM to obtain an IP address in a new subnet and create new subflows for existing TCP connections. By establishing new subflows on the local subnet, MPTCP can shift traffic from subflows using the old IP address to subflows using the new one in a transparent application-unaware fashion. Once a new subflow is established, the old one can be closed. This elegant solution avoids triangle routing, persistent tunnels, guest modifications, or virtualizing the network by extending layer 2 over the WAN. They show an example migrating a VM from Romania to Germany. However, many details of this process are not explicit in their paper.

In this paper we present our experiences implementing WAN migration using MPTCP. Many problems were not immediately obvious and thus we share them here so that others need not encounter them. This paper makes the following contributions:

- We present the design and implementation of LSM-MPTCP (Live Service Migration using MPTCP), which uses KVM and Linux (§III). Our approach integrates the original design of Nicutar *et al.* [12] with the tunneling approach of [11]. A key design goal is to minimize service downtime.
- We evaluate our approach in the Wide-Area environment using IBM's SoftLayer cloud (§V-B). We compare LSM-

MPTCP with a tunneling-only approach using several data centers across North America, each with differing network bandwidths and round-trip times to the clients. We show throughput improvements up to 600% after migration and delays reduced by up to 99%.

• We show a number of unusual problems we encountered in the process and how we solved them, including subtle interactions with ARP, bridging, and routing (§IV). We show the importance of using the proper MTU size and its significant impact on performance (§V-C).

• We study upload performance, ignored in previous work. We show how the upload case incurs much more significant packet losses (§V-D). Uploads can experience loss rates of up to 35% in our environment during migration, compared to that of 1% for downloads. We show that this is because of packets arriving while the VM is suspended during migration, and not due to the network itself.

A key requirement for our system is that MPTCP must be deployed on the clients. However, we believe this will eventually be the case, as evidenced by current IETF standardization activities [13], [14], active research in MPTCP [15]–[19], commercial deployment in load balancers [20] and WAN optimizers [21], and most significantly by Apple supporting it in iOS 7.0 and above [22].

II. Background

In this Section, we give an overview of VM Migration and Multi-Path TCP.

A. Live VM Migration

The concept of migrating a running virtual machine was first introduced by Sapuntzakis *et al.* [3] in 2002. Implementations for VMWare [2] and Xen [1] arrived in 2005. At first, approaches to migration were limited to within the data center where the VM could maintain its IP address.

1) LAN Migration: Live migration within the data center proceeds in the following stages. As illustrated in Figure 1 (adapted from [1]):

1. The remote host is prepared for the new VM.
2. The running VM is copied to the remote host, at the granularity of pages. The host Virtual Memory Monitor (VMM) is used to track the state of pages. All pages are initially marked dirty. When a page is copied, it is marked clean, and thus does not need to be copied again. If the VM writes to that page during this stage, the VMM marks it dirty, indicating the page has changed and needs to be copied again. This stage iterates over the dirty pages until some cutoff threshold is reached (described in more details below).
3. The VM is suspended. All remaining dirty pages are copied to the remote host.
4. The VM is resumed on the remote host.
5. The original host reclaims the VM resources.

Migrating a VM requires choosing tradeoffs between multiple metrics, including: (1) *migration time* (i.e., the time between when the migration is started and completed), (2) *service downtime* (i.e., the time the service is unavailable to clients), (3) *application overhead* (i.e., the overhead the system incurs while migration is under way, due to the additional load created by migrating, which in turn affects application throughput and response time), and (4) *amount of data transferred* (i.e., the volume of data transferred over the network.)

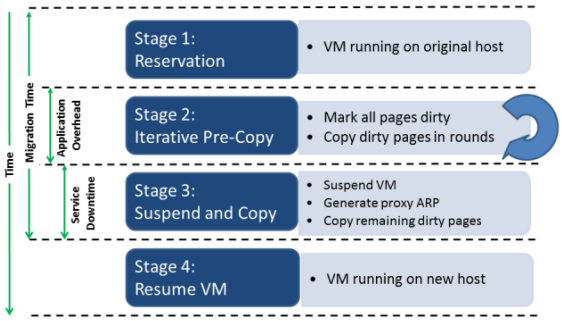


Fig. 1. VM Migration Stages

2) WAN Migration: WAN migration introduces a significant hurdle: How to route packets to the VM's IP address if the VM ends up in a different routable subnet? More recently, several approaches have proposed migrating VMs across large physical distances. However, these approaches all require extending the layer 2 network over the WAN [7]–[10]. This is problematic as all existing and future traffic must be tunneled back to the source data center. This triangular routing can add significant delay, on the order of dozens of milliseconds in the wide area, and reduce the bandwidth between the VM and its clients, as we will show in Section V.

Bradford *et al.* [11] propose an approach that does not require network virtualization, which is to tunnel packets from the original subnet to the new subnet while a new DNS entry is distributed over the wide area. This approach is illustrated in Figure 2. Similar to the previous solutions, this approach suffers from triangular routing. For example, in Figure 2, the client is in San Jose, and the VM has been migrated from Montreal to Dallas. Despite being physically closer to San Jose, the client now incurs longer RTTs and lower throughput to the service. Although the degradation in network performance (e.g., delay, bandwidth) in this case may be limited to TCP connections that were established before the migration, updating a DNS entry may take a long time (e.g., [23], [24]). Thus, new connections may still continue to be established after the migration to the old IP address, which in turn will suffer poor network performance. As their approach is the closest to ours, we compare our approach with theirs in Section V.

B. Multi-Path TCP

MPTCP is a recently standardized set of TCP options, which allow two endpoints to simultaneously use multiple paths – known as *subflows* – between them [13]. These subflows are defined logically by default by all end-to-end interface pairs.

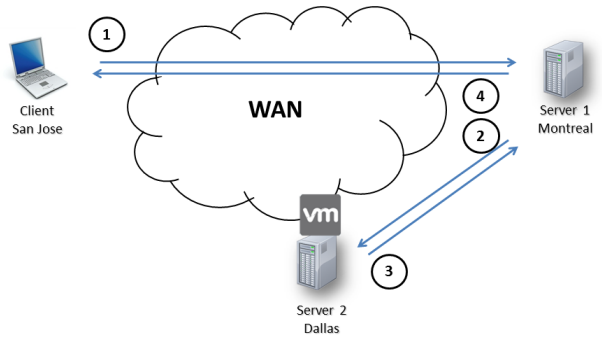


Fig. 2. Migration with Tunneling

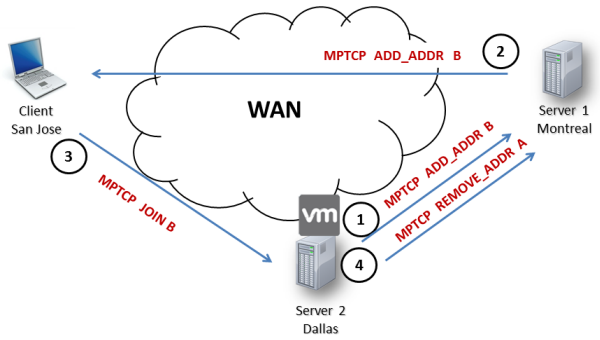


Fig. 3. Migration with MPTCP

For example, if each endpoint in a conversation has two interfaces, MPTCP will 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 naturally 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 from one path to another 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 [15]. 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 [14]. In MPTCP, the client is defined as the endpoint that sent the original SYN packet.

III. System Design

LSM-MPTCP allows the live migration of VMs and service across subnets, transparent to applications. LSM-MPTCP builds on current live VM migration procedures supported by existing virtualization technologies (e.g., VMWare, Xen, KVM) to copy the VM across host servers, and takes advantage of MPTCP to handle the network connectivity. LSM-MPTCP requires VMs to support MPTCP and have at least two virtual network interfaces.

Figure 2 demonstrates the state of the art in WAN Migration. After migrating from Montreal to Dallas, the requests and responses must be continuously tunneled through Montreal. This exhibits triangle routing, incurring extra latency and an additional point-of-failure throughout the connection.

Figure 3 illustrates very broadly our system. As before, initially a tunnel is established between the source and destination server hosts to preserve the existing network connectivity. Then, MPTCP naturally and automatically advertises a new address B for existing TCP connections between the migrated VM and its client, as shown by Arrows 1 and 2 in the figure. The client establishes a new connection using the advertised address B, as illustrated by Arrow 3. Traffic is subsequently shifted and sent through this new interface. Finally, the VM shuts down the old interface, causing MPTCP to close the old subflows, Arrow 4 in the Figure. The tunnel is terminated, and the source host server releases any related resources. This process happens over 5 phases, explained in more detail below.

Phase 1. *Tunnel creation and routing preparation:* Before the VM migration procedure is initiated, we create a tunnel between the source and destination hosts, and prepare the routing for both traffic incoming from and traffic outgoing to the VM at the destination host server. Configuring the routing at the destination host server for the VM minimizes the service down time, and is possible since no traffic to the VM is yet arriving at the destination host server. This configuration is therefore not affecting the ongoing traffic to and from the VM.

Phase 2. *VM migration:* We initiate the VM migration procedure. We note that each host server has two physical network interface cards: eth0 connected to a private network connecting all SoftLayer data centers, and eth1 connected to the public Internet. We take advantage of this property to separate the VM migration traffic from the VM traffic (e.g., to its clients). While the traffic to and from the VM goes through the eth1 interface of the hosts, we perform the VM migration over eth0 of the hosts (i.e., the highly

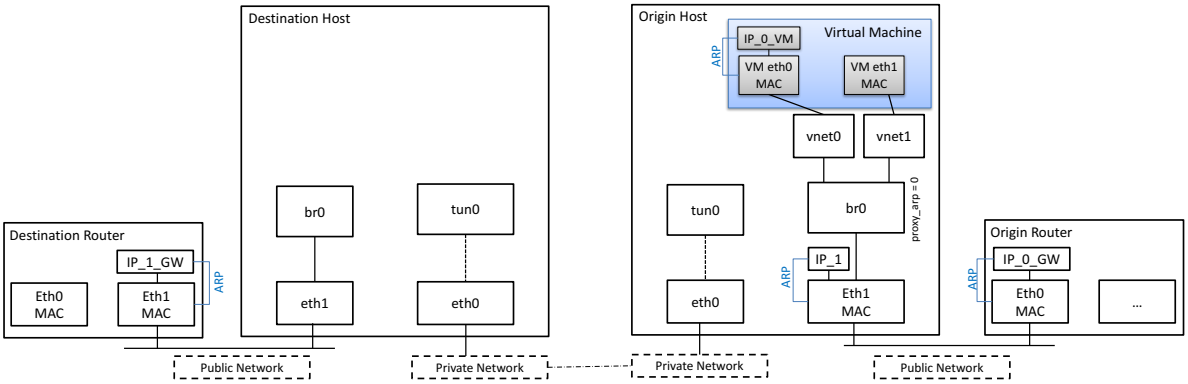


Fig. 4. Origin Host and Destination Host Before Migration

provisioned SoftLayer private network). This separation of traffic minimizes the network performance degradation for the VM.

Phase 3. *Tunnel Switchover:* As soon as the VM migration procedure has completed, we create routing entries at the source host server to forward traffic destined to the VM towards the tunnel. The time interval for Phase 3 is critical as the VM has no network connectivity during this time period. The preparation in Phase 1 minimizes the actions we need to take in Phase 3, as the routing at the destination host server is already completed. At the end of Phase 3, the VM has resumed its network connectivity, and existing TCP connections are routed through the tunnel via the source host server.

Phase 4. *New subflows:* After the VM has migrated to the destination host server, and resumed network connectivity, we bring up the second network interface (i.e., VM eth1) and configure routing so that both interfaces can be used simultaneously. Traffic to and from the new interface does not traverse the tunnel, but goes directly to the Internet. The VM then naturally advertises the newly acquired IP address in its existing MPTCP connections, and creates new subflows. Once the new subflows are successfully established, the VM shuts down the old subflows associated with eth0, and both the tunnel and resources at the source host server are released.

Phase 5. *Normal Operation:* The system is now back in its normal state, similar to where it was before Phase 1, except that it is now using a different local subnet. If the VM needs to migrate again, the same process is used, except that the roles of eth0 and eth1 are reversed.

IV. Implementation Challenges

Our implementation is based on KVM. Full software and hardware details are provided in Section V-A. We use bridged networking to support the VMs, since the IP addresses must be globally visible.

Several challenges need to be overcome, some that are unique to the wide-area VM migration case, and some because of requirements of the operational cloud network. We use the IBM SoftLayer Cloud, also described in more detail in Section V-A.

A. Before the Migration at the Source Host

Figure 4 shows the state of the source host before the guest VM is migrated (i.e., at the beginning of Phase 1). Each host server has two physical network interfaces: eth0 connected to a private network, and eth1 connected to the Public Internet. Note that *the subnet of the VM is different than that of the subnet of the host*. This is a characteristic of the operational SoftLayer cloud environment. Global IP addresses for VMs are allocated from specific IP address blocks that are called Secondary on VLAN blocks [25]. Those IP addresses do not belong to the host server's subnet, but have their own network prefix, prefix length, and gateway. The VM has two virtual interfaces (VM eth0, VM eth1), one of which (VM eth1) is not up. To allow VM connectivity to and from the Public Internet, we create a bridge device on the host, and add eth1, vnet0, and vnet1 to that bridge. Although vnet1 is down, adding it to the host's bridge will allow vnet1 to be globally reachable after we migrate the VM to the destination host server and bring vnet1 up.

B. After the Migration at the Source Host

After the VM is migrated away, we temporarily rely on the tunnel between the source and destination host server to maintain existing TCP connections (i.e., end of Phase 3). We add a route entry on the source host for packets destined to the VM IP address to be routed to the destination host server through the tunnel. However, despite this route entry, we experienced two problems.

Problem: VM MAC address goes away after Migration. After the VM is migrated to the new host, packets arriving at the origin bridge are dropped. This is because those IP packets are sent with a destination MAC address of the VM, and since that MAC address is no longer attached, the bridge drops them.

Since they do not reach the routing layer of the origin host server, they cannot be forwarded over the tunnel.

Solution: We add an ebtables redirect target rule on the source host, matching on IP packets destined for the VM, to rewrite the destination MAC address to that of the local bridge, which pushes the packets up to the IP layer to be routed.

Problem: ARP entry on the origin router expires. When the ARP entry for the VM at the origin router times out, the router sends an ARP request for the MAC address of that VM. However, because the VM is no longer attached to the bridge, there is no ARP reply generated. As a result, the origin router can no longer properly forward arriving packets destined to the origin host. Without any ARP entry for the VM, the origin router drops those packets.

Solution: We enable proxy_arp at the bridge of the origin host, allowing the bridge to reply to the ARP request with its MAC address and ensure that the gateway can forward packets destined to the VM to the bridge. At this point, the ebtables rule previously installed above is longer needed. We highlight that the proxy_arp feature is enabled during Phase 1, before the VM is migrated, to minimize downtime.

C. After Migration at the Destination Host

After the VM is migrated, while we still rely on the tunnel to route packets between the source and destination hosts (i.e., at the end of Phase 3), we experienced problems for packets both incoming to and outgoing from the VM.

Problem: Outgoing packets from the VM are dropped. When the VM resumes after migration, the default route next hop gateway in the VM is still the origin router, with a local ARP entry in the VM pointing to the origin gateway's MAC address. However, that MAC address is not reachable over the LAN of the destination host. Thus, when the VM sends its outgoing packets with a destination MAC address of the origin router, those packets end up being dropped at the local bridge. Eventually, the ARP entry expires, and the VM sends an ARP request over the local LAN for the origin router. However, since the origin router is not on the local LAN, no reply is sent back to the VM. Thus, the VM cannot send traffic.

Solution:. Our solution takes 3 steps. First, we add an ebtables (redirect target) rule on the destination host that matches on packets from the VM, and then re-writes the destination MAC address to that of the local bridge. This causes packets sent from the VM to be pushed up to the routing layer.

Second, we create a source route entry at the destination host to route packets from the VM's old IP address back over the tunnel to the source host. The first two steps solve the problem until the ARP entry expires.

Third, we configure the destination host bridge to use proxy ARP, so that when the ARP entry expires and a new MAC address is requested, the destination host provides the MAC address of the bridge. Once the ARP response has been generated, subsequent packets from the VM do not require MAC rewriting and the ebtables entry can be removed.

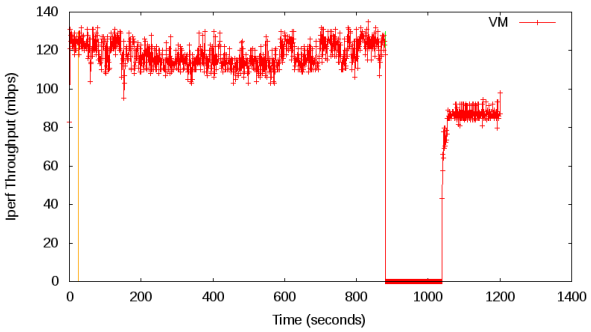


Fig. 5. Throughput Gap due to ARP Issue

While an unsolicited ARP broadcast would achieve the same end, it requires knowing when the VM is active, potentially creating a race condition between the host generating the ARP and the VM sending a packet. By using ebtables before the VM is even migrated, we avoid any delay that might be incurred by the unsolicited ARP.

Problem: Routing loop for packets sent to the VM. When a packet destined to the VM is routed through the tunnel, and decapsulated at the destination host, the destination host's routing table is consulted to determine the next action to be taken. Because the default route is the most specific route matching packets sent to the VM, those packets are then forwarded to the destination host's default gateway. In other words, those packets are forwarded back out onto the Internet. We observed a forwarding loop as packets are sent to the source host which then routes the packets through the tunnel back to the destination host.

Solution: We add a host route (/32) at the destination host to forward all packets destined to the VM to the local bridge interface, where the VM is connected to.

Problem: Incorrect ARP entry on the destination host. Despite the newly created IP route, packets were still not always properly forwarded to the VM. Figure 5 illustrates a problem we observed at times. This graph shows the throughput measured via iPerf from a VM that is migrated to another site. As can be seen, the throughput drops to zero at around 900 seconds and stays there until about 1050 seconds. This was because of an incorrect ARP entry for the VM's IP address at the destination host. Instead of pointing to the MAC address of the VM, the ARP entry sometimes pointed to the destination host's default gateway router MAC address. The incorrect ARP entry thus prevented packets from reaching the VM. The non-deterministic nature of the problem made the problem difficult to diagnose. The problem lasted until the local ARP entry at the VM for its default gateway expired. At that time, the VM initiates an ARP request, indirectly enabling the destination host to update its ARP entry for the VM.

Fig. 6. SoftLayer Data Centers (North America)

Location	San Jose	Dallas	Montreal
San Jose	-	43	71
Dallas	43	-	43
Montreal	71	43	-

TABLE I
DATA CENTER AVERAGE RTTS (MS)

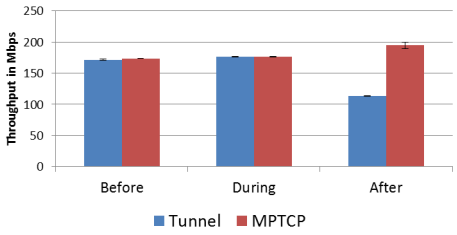


Fig. 7. Dallas Throughputs

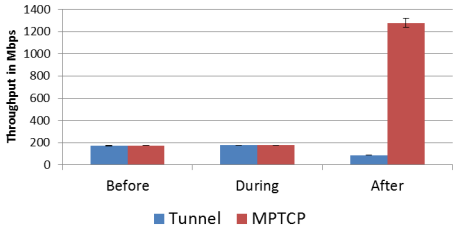


Fig. 8. San Jose Throughputs

Solution: Before the migration even begins, we place a permanent ARP entry on the destination host mapping the VM's old IP address to its old MAC address. The combination of the host route and the ARP entry ensures that incoming packets are properly forwarded to the migrated VM.

V. EVALUATION

This section presents some of our results to show the success of our prototype. We first describe our experimental environment, including cloud WAN, hardware, software, methodology, and metrics.

A. Experimental Environment

Our basic approach is to migrate a 15 GB VM over the wide area while running a network workload, using both our MPTCP-based solution and using tunneling [11] for comparison.

The SoftLayer Network. For our experiments, we rent several machines across North America from IBM's SoftLayer Cloud. We use bare-metal machines for the hypervisors, in San Jose, Dallas, and Montreal. We use a separate bare-metal machine in San Jose for our client, rather than a VM, in order to prevent any client-side performance effects from other VMs or applications. The client always runs in San Jose, and the VM starts in Montreal. We examine scenarios where the VM is migrated to Dallas and to San Jose. Table I shows the average round-trip times between the sites [26].

Hardware and Software. Our machines use 2.4 GHz Intel E5-2620 processors, with 2 sockets and 6 cores per socket. Each machine has two Intel 10-gigabit NICs. We use Ubuntu 14.04 with Linux kernel 3.13 with the MPTCP implementation 0.90 from `multipath-tcp.org` [27]. We use KVM/QEMU 2.0.0 and virsh 1.2.2.

Methodology. The internal private SoftLayer network between data centers is shared across customers. To minimize variability from both the SoftLayer private network, and the public Internet, we take two steps. First, for each metric we report the average of 10 runs and include the 90 percent confidence intervals. Second, we perform our runs at night, from roughly 9 PM to 7 AM. The resulting relatively tight confidence intervals (typically less than 1% of the measured values) give us confidence that our measurements are statistically significant.

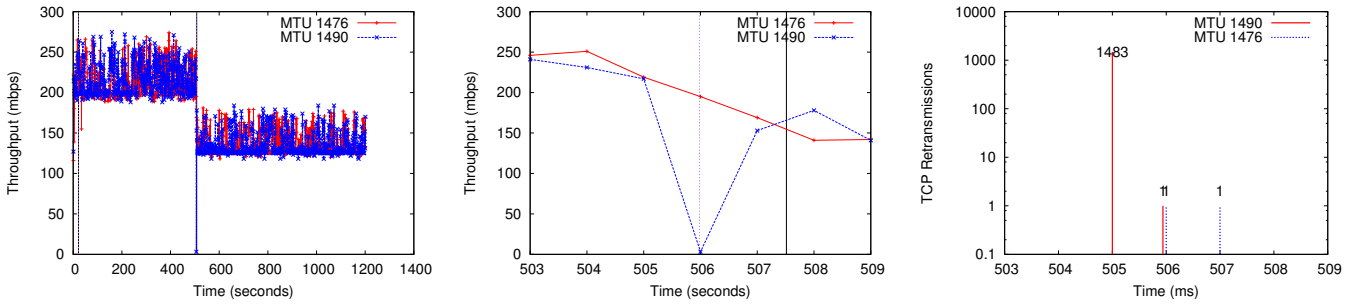
We report the following metrics:

- *Throughput* is measured using iPerf over a 1200-second interval. Our scripts timestamp the different phases (Section III) and use the timestamps to measure throughput before, during, and after the VM migration.
- *Loss Rate* is calculated by capturing packets at the sender and analyzing them using Tshark [28].

B. Baseline Results

This Section provides baseline results, comparing our approach using MPTCP with the tunneling solution from [11]. We look at two configurations: one where the VM is migrated from Montreal to Dallas, shown in Figure 7, and one where the VM is migrated from Montreal to San Jose, shown in Figure 8. Note the presence of the 90th percentile confidence intervals. The Figures show throughputs before, during, and after migration.

First, we observe that the throughput after migration using MPTCP significantly outperforms the tunnel-only solution. In the Dallas case, the tunnel approach throughput falls by 33 percent, whereas the MPTCP case improves throughput by 15 percent. In the San Jose case, the results are even starker: tunneling throughput after migration falls 50 percent, while increasing over 6-fold using MPTCP. This is because TCP throughput is very sensitive to round-trip times [29]. By moving the VMs closer to the client, the connections have lower RTT and thus higher throughput. Table II shows the



(a) Throughput over Time (b) Throughput over Time (Zoom) (c) Loss Counts over Time (log scale)

Fig. 9. Effects of MTU Size

Client	Dallas MPTCP	Dallas Tunnel	San Jose MPTCP	San Jose Tunnel
San Jose to:	40	110	0.5	142

TABLE II
ICMP PING TIMES (MS)

difference in measured RTTs for the two approaches over the two configurations, as seen by the San Jose client. As expected, the network delay is significantly reduced when MPTCP is used.

Second, throughput during migration is statistically the same as before migration, indicating the absence of any application degradation. This is because our machines are well-provisioned, both in terms of CPU power and network capacity. More importantly, we take advantage of the private SoftLayer network connecting the data centers in order to copy the VM. This prevents contention between the VM migration traffic, and the traffic between the VM and its clients sent over the public Internet. This result has not been achieved in earlier works.

We note that several metrics are statistically identical between the two approaches, including migration time and service downtime, not shown due to space limitations. This is to be expected, since the techniques are identical up until the point that the client opens a new MPTCP subflow with the VM using its new IP address. It is only after the migration completes that we see the performance benefits of using MPTCP.

C. MTU Performance Issues

Here we present our experience concerning packet MTU size, and how important it was to get this correct.

First, we noticed an 80% reduction in throughput when running iPerf from the VM compared to when running iPerf from the origin host server (with the iPerf server being on the Public Internet). We suspected the MTU size to be responsible for the difference in performance, and we therefore varied the MTU size from 1500 to 1462 (increments of 2). We found that the optimal MTU size for the VM is 1496 bytes. This might be due to the fact that the secondary on VLAN IP address assigned to VMs uses a VLAN header. Despite the extra 4 bytes for the 802.1Q tag, the MTU should usually remain as 1500 bytes with most switches. However, in this specific instance, we found that setting the MTU to 1496 bytes for the VM was optimal.

Second, we then expected that when packets are forwarded through the GRE tunnel during the migration, the optimal MTU size would be 1472 (1496 - 24 since the GRE header size is 24 bytes). However, the optimal MTU size in that case was instead 1476 bytes. We again varied the MTU size from 1500 to 1462 (increments of 2), and observed the following.

When the MTU size was larger than 1476 bytes, once the VM migrates, all traffic is tunneled via GRE, and the packets being too large – with the don't fragment (DF) bit set – are dropped. The tunnel endpoint sends ICMP messages and the sender adjusts the MSS accordingly. However, this process takes a number of milliseconds to filter back to the sending TCP. In the meantime, we have observed thousands of packets getting dropped before the window is adjusted.

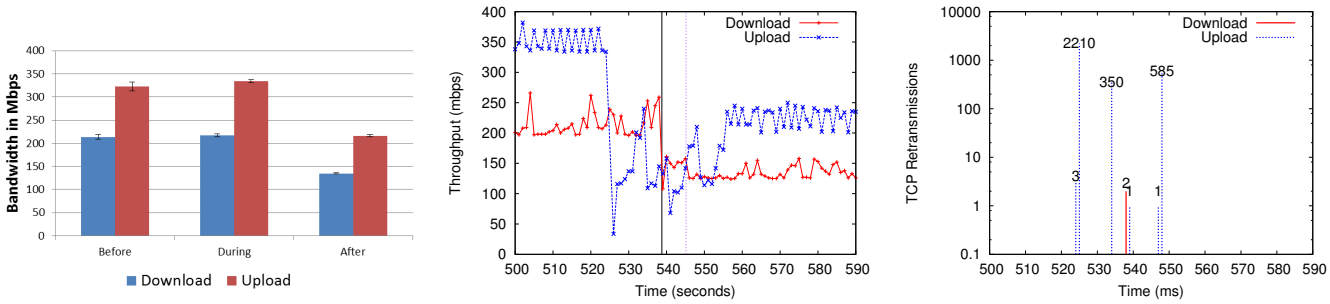
In contrast, when setting the MTU size 1476 bytes, we observe no disruption in throughput, and packet losses up to 3 orders of magnitude lower. Figure 9(a) shows the throughput over time, for the two MTU sizes. Note the blip at around 500 seconds. Figure 9(b) also shows the throughput over time, zoomed in to between 503 and 509 seconds. As can be seen, there is a large disruption in throughput for the 1490 byte MTU. In contrast, when the MTU in the VM NIC is set to 1476 bytes, there no disruption occurs. Figure 9(c) shows the packet losses for the two MTUs, on a log scale. Note the 1476 byte MTU to be 3 orders of magnitude lower.

The above examples illustrate that the optimal MTU size may vary depending on the adopted path, and although we fixed the problem by adjusting the MTU of the interface(s), this solution requires a number of experiments to determine the optimal MTU sizes *a priori*.

D. Upload vs. Download Performance

We consider migration performance when uploading. Upload traffic is growing in importance due to applications such as continuous backup, photo uploads, web cams, etc. Previous works have not considered upload performance.

Figure 10 presents a number of views of throughput comparing upload vs. download for migrating the VM to Dallas



(a) Throughput (b) Throughput over Time (Zoom) (c) Loss Counts over Time (log scale)

Fig. 10. Upload vs. Download Throughput

using the tunneling approach. Other configurations behaved similarly, not shown due to space limitations. We observe several interesting results.

First, upload bandwidth appears to be significantly higher (Figure 10 (a).) We do not believe this is fundamental to the upload case, but rather that there is more available bandwidth in our network, since downloads are still more common.

Second, both migration time and service downtime are statistically indistinguishable (not shown due to space limitations). The latter surprised us since we expected higher losses in the upload case, which we thought would affect service downtime. To study this more deeply, we examine a trace of throughput over the time of the run. We observe that the throughputs are consistent during most of the run, but at around time 500 both configurations take a substantial hit. This is when migration happens. Figure 10 (b) shows the time period from 500 to 590 seconds. While both transfers are affected by the migration, the upload transfer is clearly disrupted more.

We continue our investigation by measuring packet losses over 1 second intervals. We study the interval during migration, which incurs both the lowest throughput and the highest loss rates. We show the raw number of losses (TCP retransmissions) in Figure 10 (c) for the period from 500 to 590 seconds. Note the y-axis is in log scale. It is clear the upload case is incurring significant burst losses, at roughly 3 points, including 1 burst of over 2200 packets. The download case, in contrast, exhibits very few losses.

This is because, while the VM is suspended, large bursts of packets can arrive for the VM and thus get dropped. In the download case, the VM is the sender, and thus cannot send while suspended.

VI. Related Work

Sapuntzakis *et al.* [3] present the first work on migrating virtual machines. They define the running state of the machine as a capsule, and present a number of optimizations that reduce migration time by a factor of 18. Clark *et al.* [1] present VM migration using the Xen hypervisor. Nelson *et al.* [2] describe VMWare's VMotion VM migration product, which uses a very similar approach. Travastino *et al.* [8] present measurements of migrating Xen VMs over long distances. However, their approach is to use layer 1 and 2 circuits without any routed paths. This approach requires extending L2 notions over the wide-area and is not practical for accessible IP-based services. Our approach, on the other hand, works with any standard IP routing architecture. Wood *et al.* [10] present *Sandpiper*, a system for monitoring physical and virtual resources, that determines when and where to migrate VMs so as to minimize hotspots. Bradford *et al.* [11] demonstrate live VM migration over the wide-area by tunneling traffic from the old host in one subnet to the new host. While they show that connectivity is maintained, service downtime is 3 seconds in the LAN case and 68 seconds in the WAN case. Surie *et al.* [30] propose using opportunistic replay as a way of reducing the amount of state transferred during VM migration. Liu *et al.* [31] propose combining checkpoint/restart with trace/replay as a means to improve migration performance. They demonstrate a reduction of 74% in service downtime, 31% in migration time, and 95% in data transferred. Jin *et al.* [32] propose using compression to shrink the size of the VM data transferred between source and destination hosts. They show reductions of 27% in service downtime, 32% in migration time, and 68% in data transferred. Jin *et al.* [33] propose reducing the virtual CPU rate during migration as a way of preventing the VM from dirtying pages faster than they can be transferred over the network. Wood *et al.* [9] present CloudNet, a system for migrating VMs over the WAN. However, Cloudnet requires using network virtualization to extend the layer 2 network over the wide area. Nicutar *et al.* [12] are the first to propose using MPTCP for VM migration. They show an example of a VM migrating from Germany to Romania, taking a few seconds, reducing application level RTT from 50 to 4 milliseconds. However, during migration, there is a spike of application-level RTT by over a factor of 10, which we attribute to packet loss as the machine migrates. They do not discuss low-level details of the migration, and do not mention tunneling. Our work, in contrast, goes into significant detail about how to implement migration using MPTCP, and provides a detailed body of results. Mashtizadeh *et al.* [7] present the design of VMWare's XvMotion, which can migrate VMs over long distances. They demonstrate migrating a VM from to India. However, they

utilize network virtualization techniques to extend the layer 2 LAN across the two data centers. They experience a migration time of 374 seconds with a service downtime of 1.39 seconds. Our approach, again, requires no network virtualization and works with any standard IP routing architecture.

VII. Conclusions and Future Work

We shared our experience and described the issues we encountered while implementing LSM-MPTCP to enable live service migration of VMs across subnets, without using persistent tunneling or network virtualization techniques. By migrating VMs closer to the client, we can significantly improve the performance they experience.

We observed that during migration, TCP connections can experience high loss rates for various reasons. We plan to look at techniques to mitigate this problem.

VIII. acknowledgement

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, ser. NSDI'05. Berkeley, CA, USA: USENIX Association, 2005, pp. 273–286.

[2] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '05. Berkeley, CA, USA: USENIX Association, 2005, pp. 25–25.

[3] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum, "Optimizing the migration of virtual computers," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 377–390, Dec. 2002.

[4] I. U. Tamas Ecsedi, "Personal communication," April 2017.

[5] 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 *Proceedings of the Seventh ACM Symposium on Cloud Computing*, ser. SoCC '16. New York, NY, USA: ACM, 2016, pp. 141–154.

[6] IBM Softlayer, "Softlayer dallas data center," https://www.ibm.com/cloud-computing/bluemix/our-network.

[7] A. J. Mashtizadeh, M. Cai, G. Tarasuk-Levin, R. Koller, T. Garfinkel, and S. Setty, "Xvmotion: Unified virtual machine migration over long distance," in *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. Philadelphia, PA: USENIX Association, 2014, pp. 97–108.

[8] 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*, vol. 22, no. 8, pp. 901–907, 2006.

[9] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. van der Merwe, "Cloudnet: Dynamic pooling of cloud resources by live wan migration of virtual machines," in *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '11. New York, NY, USA: ACM, 2011, pp. 121–132.

[10] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*, ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 17–17.

[11] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *Proceedings of the 3rd International Conference on Virtual Execution Environments*, ser. VEE '07. New York, NY, USA: ACM, 2007, pp. 169–179.

[12] C. Nicutar, C. Paasch, M. Bagnulo, and C. Raiciu, "Evolving the internet with connection acrobatics," in *Proceedings of the 2013 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMiddlebox '13. New York, NY, USA: ACM, 2013, pp. 7–12.

[13] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," RFC 6182, Internet Engineering Task Force, Mar. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6182.txt

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

[15] 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 *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 29–29.

[16] A. Croitoru, D. Niculescu, and C. Raiciu, "Towards WiFi mobility without fast handover," in *Proc. of USENIX NSDI*, 2015.

[17] B. Han, F. Qian, S. Hao, and L. Ji, "An anatomy of mobile Web performance over Multipath TCP," in *Proc. of ACM CoNEXT*, 2015.

[18] B. Han, F. Qian, L. Ji, V. Gopalakrishnan, and N. Bedminster, "MP-DASH: Adaptive video streaming over preference-aware multipath," in *Proc. of ACM CoNEXT*, 2016, pp. 129–143.

[19] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend TCP?" in *Proc. of ACM IMC*, 2011.

[20] Citrix Corporation, "Maximize mobile user experience with NetScaler Multipath TCP," https://www.citrix.com/blogs/2013/05/28/maximize-mobile-user-experience-with-netscaler-multipath-tcp/.

[21] F5 Networks, "Overcome all application performance bottlenecks," http://www.f5.com/pdf/products/big-ip-application-acceleration-manager-datasheet.pdf.

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

[23] godaddy.com, "What factors affect DNS propagation time," https://www.godaddy.com/help/what-factors-affect-dns-propagation-time-1746.

[24] Managed.com, "DNS propagation and why it takes so long– explained," https://support.managed.com/kb/a604/dns-propagation-and-why-it-takes-so-long-explained.aspx.

[25] Softlayer, "Static and portable IP blocks," https://knowledgelayer.softlayer.com/articles/static-and-portable-ip-blocks.

[26] SoftLayer, "Softlayer IP backbone :: Looking Glass," http://lg.softlayer.com.

[27] C. Paasch and S. Barre, "Multipath TCP in the Linux kernel," http://www.multipath-tcp.org.

[28] Wireshark Project, "Tshark," https://www.wireshark.org/docs/man-pages/tshark.html.

[29] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. of ACM SIGCOMM*. ACM, 1998.

[30] A. Surie, H. A. Lagar-Cavilla, E. de Lara, and M. Satyanarayanan, "Low-bandwidth VM migration via opportunistic replay," in *Proceedings of the 9th workshop on Mobile computing systems and applications*. ACM, 2008, pp. 74–79.

[31] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," in *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing*, 2009.

[32] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan, "Live virtual machine migration with adaptive memory compression," in *IEEE International Conference on Cluster Computing*, 2009.

[33] H. Jin, W. Gao, S. Wu, X. Shi, X. Wu, and F. Zhou, "Optimizing the live migration of virtual machine by CPU scheduling," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1088–1096, 2010.