Seamless Migration of Virtual Machines Across Networks

Umar Kalim,*† Mark K. Gardner,† Eric J. Brown,† and Wu-chun Feng*
Office of IT†, Department of Computer Science*
Virginia Tech, Blacksburg, VA 24060
{umar, mkg, brownej, wfeng}@vt.edu

***Abstract*—Current technologies that support live migration require that the virtual machine (VM) retain its IP network address. As a consequence, VM migration is oftentimes restricted to movement within an IP subnet or entails interrupted network connectivity to allow the VM to migrate. Thus, migrating VMs beyond subnets becomes a significant challenge for the purposes of load balancing, moving computation close to data sources, or connectivity recovery during natural disasters. Conventional approaches use tunneling, routing, and layer-2 expansion methods to extend the network to geographically disparate locations, thereby transforming the problem of migration between subnets to migration within a subnet. These approaches, however, increase complexity and involve considerable human involvement.**

The contribution of our paper is to address the aforementioned shortcomings by enabling VM migration across subnets and doing so with uninterrupted network connectivity. We make the case that decoupling IP addresses from the notion of transport endpoints is the key to solving a host of problems, including seamless VM migration and mobility. We demonstrate that VMs can be migrated seamlessly between different subnets — without losing network state — by presenting a backward-compatible prototype implementation and a case study.

***Index Terms*—migration, virtual machine, wide-area network, subnet, connectivity, TCP/IP**

I. Introduction

Virtual machine (VM) migration has served as a technology to enhance resource allocation and utilization. In turn, it has ushered in the cloud computing era, whether for load-balancing purposes to eliminate hotspots [1], moving computation close to data sources [2], or failover planning. Seamless VM migration across networks opens additional doors for opportunities. Consider, for example, the possibility of a live migration of financial services hosted in a data center on the east coast of the United States. In an impending disaster, these services may be migrated to a data center on the west coast without interrupting existing network connections, which may have originated from within or outside the data center. While the concerns of live VM migration within a data center have been successfully resolved to a large extent, the issues of live VM migration *beyond* a subnet are yet to be addressed.

The seamless migration of VMs involves a host of challenges such as transferring VM images [1], [3], managing storage [2], copying intermediate state [4], maintaining network connections [3], [5], addressing security considerations [6], meeting performance goals [1], facilitating operations management, and so forth. In this paper, we focus on the issue of *maintaining network connection state following a live VM migration beyond a subnet*.

Contemporary hypervisors rely on the Reverse Address Resolution Protocol (RARP) to maintain network connection state following a migration *within* a subnet. However, migration *beyond* a subnet is a challenge as the IP addresses of the network interface normally change. This change results in a disruption in continuity as existing connections timeout because the endpoint with the old IP address does not exist anymore. Unless applications implement reconnection, migration results in disconnections. This precludes the straightforward migration of VMs to geographically disparate locations (e.g., between data centers in different regions).

We leverage our prior research in developing an *isolation boundary* [7], [8], as a backward-compatible extension for TCP/IP, which decouples the naming of endpoints from the naming of the flows. The decoupling ensures that a change of IP address does *not* impact the connection state because the connection will be identified by a label independent of the IP address (rather than by the traditional 4-tuple of IP addresses and ports). Such independence not only enables VM migration, but it also facilitates features such as mobility and reliable connectivity [7].

Conventional solutions, such as tunneling [3], modified routing [5], and layer-2 expansion [9]–[11], work around the problem of connection loss due to a change of IP address. Specifically, in these cases, the layer-2 network is extended to geographically disparate locations. This approach transforms the problem of migration of VMs between subnets to VM migration within a subnet, which is already well understood.

However, extending a subnet to geographically disparate data centers is complex and requires considerable human involvement [3], [5]. Methods of expanding the layer-2 network [9]–[11] may not perform well at large scale (e.g., the scale of the Amazon cloud), where the different subnets are to form parts of a single layer-2 domain. With tunneling [3], routing [5], and some layer-2 expansion methods (e.g., [11]), a coupling exists between the source subnet and the migrated VM, which is an undesirable constraint.

Moreover, network architects typically partition networks within a data center into multiple subnets, particularly for large-scale data centers and clouds (e.g., Google or Amazon). Whether for flexibility of design, partitioning services, managing reliability or failover, balancing load, managing operations

(e.g., maintenance), or maintaining security, partitioning allows the architect to have more degrees of freedom. Therefore, by enabling layer-3 migration (vs. layer-2 migration), we can avoid the migration constraints that cloud services may have due to coupling with the hardware (or layer-2 domain), particularly when data centers are partitioned into subnets.

Thus, rather than continuing to address the symptoms of the problem as above, we focus on the *source* of the problem — the use of IP addresses in the 4-tuple to identify a transport connection — in order to enable seamless layer-3 migration of VMs across networks.

In Section II, we discuss the state-of-the-art methods used for VM migration across networks. Section III explains the details of the challenges faced in maintaining network state and why state-of-the-art technologies cannot address the problem adequately. Section IV presents our proposed approach and its realization in FreeBSD v8.1. We also present a case study of VM migration across networks. Section V discusses and evaluates our proposed approach and the case study. To conclude, we summarize our work and identify future directions of research in Section VI.

II. Related Work

Following a migration within a subnet, hypervisors currently rely on the Reverse Address Resolution Protocol (RARP) protocol to enable continued use of existing TCP connections. After migration, the VM continues to use the IP addresses that it was configured with in the source subnet. On the other hand, the physical MAC addresses of the new host will be different. RARP renews the mapping between the VM's IP addresses and the host's MAC addresses. As a result, the migration process appears seamless. Here we assume that the downtime during the migration is such that the network connection does not timeout.

Apart from the issues of transferring VM images and managing its state and storage, the challenge of migrating a VM beyond a subnet (in contrast to within a subnet) is maintaining appropriate network (TCP) connection state. The methods used to address the challenge of live migration while maintaining network state are different manifestations of tunneling [3], routing [5] and layer-2 expansion [9]–[11].

In their tunneling-based approach, Bradford et al. [3] use IP tunneling [12] to redirect network traffic from the source to the destination subnet and therefore avoid network disruption. Upon migration, an IP tunnel is setup, and traffic is tunneled to the destination subnet after migration is complete. As soon as the VM is initialized at the destination subnet, it acquires a new IP address for its interface and can respond to incoming traffic meant for the new address as well as the old address — the interface is setup to respond to both addresses. Dynamic DNS is used to update the IP address of the services hosted in the VM. The tunnel is terminated after all connections using the old address are shut down. The above approach, however, requires cooperation from the source server. In addition, until the old connections terminate, a coupling exists between the migrated VM and the source subnet.

Erickson et al. [5] demonstrate that OpenFlow allows applications/VMs to continue to use their old IP addresses even when they are migrated to different subnets. The migration tools and hypervisors deal with the issues of transferring the VM, while OpenFlow is used to direct traffic to the destination subnet. If the setup is automated, it may not take long to configure the forwarding tables.

The VXLAN [9] framework creates an overlay network over layer-3 by encapsulating the entire layer-2 frame. VXLAN Tunnel End Points (VTEPs) may be used to expand a layer-2 network to geographically disparate locations. For example, two geographically disparate data centers could be made to be one layer-2 network using a VTEP at each data center. The VTEPs would be responsible for encapsulating traffic with a VXLAN header, forwarding it, and subsequently decapsulating the layer-2 frame when transferring it to the destination host. Note that as with OpenFlow, the VXLAN solution requires additional complexity in the form of VTEPs.

Another manifestation of the approach of having a single layer-2 network is to have a large layer-2 domain. As the Spanning Tree Protocol (STP) has stability issues when the layer-2 domain grows too large, protocols such as TRILL [10] enable large layer-2 domains by replacing STP. In essence, TRILL applies the Intermediate System to Intermediate System (ISIS) protocol to route Ethernet frames. Though vendors support such solutions, the debate is still open whether TRILL would benefit data center implementations or result in poor data center designs [13].

Similarly, a virtual private LAN service (VPLS) [11] over IP/MPLS has been used as a method to expand the layer-2 domain. Typically, LAN segments are brought together by virtualizing a switch across a link — making multiple switches appear as a single virtual switch extending over geographical distances. Here too, the layer-2 domain is allowed to grow to large sizes by managing multiple but localized spanning trees.

While the above approaches are sound, they are fundamentally ad-hoc solutions because they treat the symptoms of the problem rather than the problem itself. In contrast, Salz et al. [14] and Snoeren et al. [15] have suggested decoupling flow labels from IP addresses. However, their approaches have not been adopted because they are not backward compatible and require a "flag day" for deployment.

Furthermore, the primary purpose of the above approaches is not to support VM migration; instead their contribution is elsewhere. For example, with VXLANs, the primary motivation is to expand the VLAN address space. VXLANs can go well beyond limit of 4094 logical networks that can be setup with VLANs — with the 24-bit segment ID, 16-million layer-2 VXLAN networks can exist in a common layer-3 subnet. Similarly, TRILL replaces STP to allow much larger layer-2 domains as well as better link utilization; layer-2 links, which may have been ignored to avoid loops, may be used for better load distribution and thus bandwidth utilization.

By leveraging our research on TCP extensions, we demonstrate a new approach that decouples flow labels from IP addresses and enables the continued use of existing TCP

connections following a migration. Unlike Salz's solution, our approach is backward compatible with legacy TCP stacks. (However, such stacks will not gain the benefit of the extended features.)

III. Background

In this section, we discuss the challenges for VM migration and our previous research on an isolation boundary.

A. Challenges for VM Migration

Current approaches to VM migration are limited because they use IP addresses and ports to identify the TCP connection. However, IP addresses are meant to identify the network interface of the host. Therefore, overloading the use of an IP address to also identify a TCP connection binds the connection, *for its lifetime*, to that IP address alone. If the IP address were to change (for reasons of mobility or migration), the transport connection which uses the old IP address would break. This is because the interface would have a new IP address and the transport connection labeled with the old IP address is not valid. The hypervisor (or application) would have to setup a new connection with its peer, using a new socket, to continue communication.

This limitation does not allow seamless VM migration because migrating to a different subnet requires the acquisition of new IP addresses. This results in the termination of existing connections and requires the set-up of new connections. As mentioned earlier, there are ad-hoc methods that allow continued use of old addresses. In contrast, we propose a clean and practical approach that eliminates the dependence on IP addresses for labeling transport connections.

B. Isolation Boundary

In our prior research, we developed a mechanism called the *isolation boundary* [7], [8], which enables rich extensions to TCP. Here we leverage the capability of the isolation boundary to decouple transport endpoint identifiers from IP addresses and thereby address the issue that a change in the IP address breaks TCP connections. With the isolation boundary, we create a notion of an abstract flow, which we refer to as a transport-independent flow (TI flow) to emphasize that it is different than a TCP flow. The TI flow is identified by a transport-independent flow identifier (TIFID). As the TIFID is independent of the underlying network addresses, a change in the IP addresses does not invalidate the connection. Instead, the mapping of the TIFID to the (new) IP address is updated.

TIFIDs are options that are exchanged during the connection setup phase; we refer to such options as *isolation boundary options*. A mapping between the abstract flow's sequence space to that of the TCP connection is created and synchronized at key stages during state transition.

Figure 1 shows an illustration of how the isolation boundary enables the mapping of TI flow to TCP connection. It also identifies when synchronization of state occurs following a disruption.

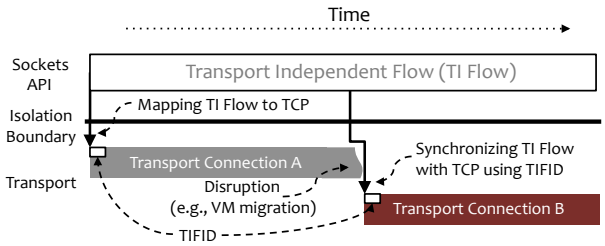

Fig. 1: An illustration of the transport-independent flow mapping to TCP connections.

IV. Methodology

Here we discuss the required functionality and mechanics for enabling layer-3 migration, along with our associated prototype. We then present our experimental setup and case study for migration between subnets.

A. Required Functionality

There are three fundamental functionalities that we need to enable layer-3 migration:

1) The connection between the client and the service needs to be independent of the underlying transport. The notion of a logical flow enables layer-3 migration, and changes in IP address will not disrupt existing TCP connections.
2) Following a migration, a mechanism is needed for the VM to realize that network configuration needs to change. This would happen when the VM is migrated and resumed. Ideally, this would be implemented as part of the hypervisor — perhaps as part of the virtual driver. (This may also be viewed as a need for cross-layer communication.)
3) Once the VM that hosts the server has migrated to a different subnet (with help via the hypervisor), the network interface gets a new IP address. The abstract flow to TCP connection mapping can be updated at the VM. However, the client would not be aware of the server's change of IP address, and thus, there is a need to update the client's mapping of the abstract flow to the TCP connection. Once done, the client continues operation.

B. Mechanics

To understand the mechanics of layer-3 migration using the isolation boundary, we explain the processes involved before, during, and after the migration. Figure 2 illustrates the steps involved in the migration.

1) *Connection Setup:* The client contacts the server that is hosted in a VM. During connection setup (i.e., three-way handshake), the isolation boundary options are exchanged.
2) *Suspension:* Upon the decision to migrate, the hypervisor hosting the VM suspends the VM's activity and records ephemeral state. The VM image and intermediate state are copied over to the destination.[1]

[1]This process may be optimized to achieve live migration [4].

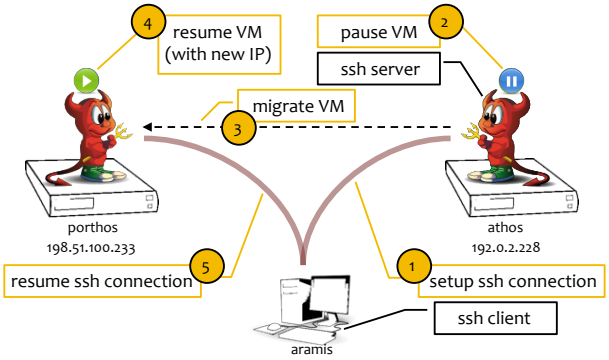

Fig. 2: An illustration of the steps involved and where they take place during a VM Migration.

3) *Resumption:* Once the VM is migrated with its image and ephemeral state, it is resumed. At this point, the VM still has the old network configuration.
4) *Synchronization:* Once the VM migration has taken place, the server contacts the client with a SYN message. Here, the TCP SYN request would be with the new IP address. The client recognizes the TI flow because of the accompanying TIFID in the SYN message and allows connection setup instead of replying with a reset or ignoring the request. The isolation boundary synchronizes the logical sequence space over the new TCP connection. In the scenario where the VM hosting the server migrates, which is explained in Section IV-E, the hypervisors of the VM triggers synchronization with the client. However, if the client moves, the client would trigger synchronization. If both move, additional bookkeeping is required.
5) *Continued Operation:* The application continues interacting with the service. Seamless migration from the perspective of the client and server is complete.

C. Implementation

To realize the functionalities identified above, we create a prototype with the following components: (1) isolation boundary, (2) link-status daemon, and (3) synchronization agent. (Other implementations are possible.)

A prototype implementation of the isolation boundary is available for the FreeBSD v8.1 kernel; details of the implementation are discussed in our prior work [7], [8]. To update network configuration of a VM following a migration, we chose an expedient method of a daemon that monitors for changes in link status when the VM is resumed and generates an interrupt-like notification. To synchronize state information, an agent at the client listens for a new TCP connection with a complete TIFID. The synchronization updates the mapping of the TIFID to TCP connection (with the new IP address) for both the service and the client.

As an artifact of our implementation, the stack in the client role listens on the same port through which it has an established connection. The listening port is available only for SYN messages with complete TIFIDs (not the partial TIFIDs typical of SYN segments establishing the initial connection) so that existing TI flows may updated with a mapping to the new addresses. An ideal approach would be to implement such functionality as part of the hypervisor.

D. Experimental Setup

To study correctness and establish a proof of concept, we setup FreeBSD v8.1 images with isolation-boundary-enabled kernels. We use VMware Player as the hypervisor and Ubuntu v12.04.1 LTS as the host OS. An SSH server acts as the service/application and a client `aramis` connects over the network to the SSH server. We chose SSH as the application to validate the correctness of the TI flow to TCP connection mapping, both before and after the migration, as any misaligned or lost byte in the encrypted bitstream would break the SSH-over-TCP connection.

As illustrated in Figure 2, the SSH server is hosted in a VM deployed at the host `athos` in the subnet 192.0.2.0/24. The VM migrates to the host `porthos`, which is setup in the subnet 198.51.100.0/24. These subnets represent networks in different buildings on the Virginia Tech campus.

E. Case Study

We study the scenario where we migrate a VM between buildings, which contain different subnets. In this setup, the client `aramis` connects to the VM hosted at `athos` over SSH and executes different jobs for test purposes. The VM at `athos` is then suspended.

Implementing live migration requires involvement of hypervisors, which we did not choose to do for our prototype demonstration as we used a proprietary hypervisor to demonstrate the general applicability of the approach. Instead, to emulate live migration, we copy the static image hosted at `athos` to `porthos` in advance. After suspension, we use `rsync` to copy the intermediate state saved by the hypervisor.

Following the transfer of intermediate state, the VM is resumed at `porthos`. Upon resumption, the link-state daemon notices the change in link states and reconfigures the network interface. The daemon is setup to generate a DHCP request to acquire network configuration parameters whenever the interface comes up. Subsequently, the synchronization agent takes action to synchronize the existing TI flow by setting up a TCP connection with the client using the new IP address.

After synchronization, the client at `aramis` continues interaction with the server (migrated to `porthos`), oblivious to the fact that the server has migrated to a different subnet. Thus, SSH connectivity was not interrupted.

V. Discussion & Evaluation

Here we compare and contrast our proposed approach with existing solutions and discuss our case study.

A. Layer-3 vs. Layer-2 Migration

Whether migrating VMs *within* a data center or *between* data centers, the challenge remains the same. In either case, the possibility of layer-3 migration adds to the design flexibility available to system and network architects as it allows for

a clean separation of concerns. Such flexibility applies to both communication scenarios: when communication originates from within the data center and outside the data center.

Whether we use tunneling, routing, or layer-2 expansion, the intention is to convert the problem of migrating VMs between subnets to the problem of migrating within a subnet. In other words, the problem of layer-3 migration is converted into a problem of layer-2 migration. As discussed in Section II, layer-2 expansion proposals are not meant to address VM migration. Therefore by using a method that was developed for a different purpose, we are not addressing the problem that limits migration beyond the subnet, instead we are working around the problem.

Such an approach puts a constraint on the design of the network as well. As discussed in Section I, partitioning the network into subnets adds to the design flexibility. Therefore, if transport connections are independent of network labels we would have the best of both worlds, that is, enabling seamless live migration beyond networks as well as flexibility in the design of networks.

With the isolation boundary, we tackle the source of the problem, which is the coupling of naming abstractions. In effect, the solution does not require dealing with layer-2, but instead enables layer-3 migration. TIFIDs present a label that is independent of the network address. Subsequently, a change in the network address has no impact on connectivity.

B. Downtime and Latency

Minimizing the time for live migration is desirable. Automating the process of configuring forwarding tables using OpenFlow incurs a negligible increase in the downtime for a live migration. Similarly with IP tunneling, setting up the tunnel incurs minimal overhead, which may be minimized further by optimizations. Layer-2 expansion methods (i.e., VXLAN, TRILL, VPLS/MPLS) also do not incur any more downtime than what is necessary to transfer intermediate state.

In these cases, if all communication destined for a service hosted by the migrated VM originates within the subnet, then the downtime may not be more than what is necessary for transfer of intermediate state. However, if communication happens with elements outside the subnet, then migrating a VM to a geographically disparate location requires traffic to be routed through the source subnet. This would incur additional latency for the existing network connections (whose state was maintained following the migration). It is not be feasible to advertise new routes to the outside world for the portion of the subnet that has been migrated to the new location.

With a hypervisor implementing the isolation boundary, the network interface would need to acquire a new IP address for the migrated VM. Optimizations may be applied to minimize this overhead; for example, as part of live migration the hypervisor may acquire an address for the interface before the intermediate state is transferred to the destination subnet. Unless such optimizations are applied, the time required to acquire a new IP address (e.g., with a DHCP request) would be greater than that of an RARP request to update the IP-to-MAC address mapping.

On the other hand, the latency between the client and a hypervisor implementing the isolation boundary does not incur any overhead. There is no increase in latency between the client and migrated server as the new IP address assigned to the network interface is owned by the destination subnet. Thus, communication is direct, unlike the other methods where traffic from the client is routed through the source subnet before it gets to the server in the destination subnet. This is what we demonstrate in the case study.

C. VM's Coupling with Previous State (/Subnet)

As noted above, tunneling, routing, and layer-2 expansion methods may be applied to extend a subnet to a different geographical location, but these methods create unnecessary coupling between the source and destination subnets. Such solutions do not work well where disaster recovery or failover management is the concern (i.e., when migrating VMs between sites).

If VMs are to be migrated to a different site for disaster management, we cannot assume that the forwarding elements at the source subnet would continue to assist after migration is complete. With tunneling methods, communication from outside the subnet continues to arrive at the source subnet, which is then forwarded through the tunnel to the destination subnet. Herein lies the assumption that the source subnet would continue to assist even after the migration is complete.

There may be some optimizations, where the migrated VM's network interface card (NIC) is assigned a new IP address while it maintains its old IP address until the old TCP connections are active. However, if the source subnet were unable to assist, such a solution would not work, at least for the network connections setup before migration.

Similarly, if an OpenFlow-based approach to the traffic is used, the controller configures the forwarding elements so that traffic is sent to the destination subnet. While the VM may be hosted in the destination subnet, the forwarding element at the source subnet continues to participate in the communication. Such behavior would not be acceptable when dealing with VM migration for disaster management. The same is the case with layer-2 expansion methods.

Due to the use of the isolation boundary, there is no requirement that the source subnet participate after the migration. This is because the service, after migration, uses an IP address that belongs to the subnet where the VM migrated.

D. Correctness

Using SSH enables us to validate correctness of the migration. With SSH any misalignment of bytes, lost segments or incorrect ordering would break the encrypted stream stream. As we are able to successfully use the SSH client application following a migration, we establish the fundamental correctness of the method and implementation. Unfettered network access is enabled, even after a change of IP address, because

we were able to decouple the IP address (network label) from the socket (transport label).

As seen in Figures 3, 4, and 5, the network configuration changes such that the VM is connected via different subnets before and after migration. The transport connection state also shows that, after migration, the old connection does not exist and a different port at the server is used to interact with the same port that was used earlier on the client — showing that the logical connection has been resumed over a new port at the server. We see the same configuration from the client's perspective; the local setup remains the same, but the server port changes for the same TCP connection after migration.

In spite of these changes, the application continues to operate without a hitch. Indeed SSH continues to show the old address. In Figure 6, we see that the `SSH_CONNECTION` environment variable is set when the connection was setup but later the application is oblivious to the change in network configuration. It appears that the SSH application does not make use of the information stored in the environment variable.

E. Pause-and-Copy Migration

Interruption in connectivity can also be a concern for pause-and-copy migrations. This is because, if the VM migrates to a different subnet and is required to acquire new IP addresses, the existing connections that were paused would be discontinued. Solutions such as Dynamic DNS that update domain name to IP mapping cannot help as the services are paused, not stopped, for migration. Therefore, pause-and-copy migrations effectively involve the same procedures. It is just that the time scales at which they happen are much larger as compared to live migration.

As we demonstrate in the case study, by using the isolation boundary, we can avoid all issues of coupling between transport and network labels and thereby enable a seamless pause-and-copy migration.

F. Backward Compatibility

In case a client does not support the isolation boundary, it is able to interact with the VM initially as the isolation boundary extension is backward compatible. However, if the VM migrated in such a case, the client would not be able to resume connectivity with the VM as it would not recognize the transport connection with a new IP address.

G. Deployment

As the isolation boundary is backward compatible, there is no requirement that all participating network elements be aware of the functionality. Those network stacks that implement the isolation boundary would benefit from the functionality, all others would fall back to legacy support. However, entire subnets can benefit from the features by deploying isolation-boundary-aware gateways (e.g., NATs, load balancers). The nodes would only benefit from the support provided by the isolation-boundary-aware gateway during the time they remain within the scope of the gateway. Nevertheless, such gateways may facilitate incremental adoption.

```
em0, before migration:
  ether 08:00:27:49:75:00
  inet 192.0.2.228    broadcast 192.0.2.255
em0, after migration:
  ether 08:00:27:49:75:00
  inet 198.51.100.233 broadcast 198.51.100.255
```

Fig. 3: Network configuration at the server, before and after VM migration.

```
connection state before migration:
  Proto  Local Address    Foreign Address    (state)
  tcp4   e4.dhcp.v.ssh    d8.dhcp.v.41270    ESTAB.
  tcp4   *.ssh            *.*                LISTEN
connection state after migration:
  Proto  Local Address    Foreign Address    (state)
  tcp4   e9.dhcp.v.48472  d8.dhcp.v.41270    ESTAB.
  tcp4   *.ssh            *.*                LISTEN
```

Fig. 4: TCP connection state at the server, before and after migration.

```
connection state before migration:
  Proto  Local Address    Foreign Address    (state)
  tcp4   aramis.41270     *.*                LISTEN
  tcp4   aramis.41270     e4.dhcp.v.ssh      ESTAB.
connection state after migration:
  Proto  Local Address    Foreign Address    (state)
  tcp4   aramis.41270     *.*                LISTEN
  tcp4   aramis.41270     e9.dhcp.v.48472    ESTAB.
```

Fig. 5: TCP connection state at the client, before and after migration. Note the listening socket on the same port is an artifact of our implementation.

```
application state before migration:
     > echo $SSH_CONNECTION
     192.0.2.216 41270 192.0.2.228 22
application state after migration:
     > echo $SSH_CONNECTION
     192.0.2.216 41270 192.0.2.228 22
```

Fig. 6: Application state at the server, when logged from the client, before and after VM migration. Note that the environment variable is set at the time of connection setup and is oblivious to change in configuration.

H. Compatibility with Live Migration

Our proposed approach does not make any assumptions about procedures involved in migration. There is no expectation from either the service hosted at the VM or the client application. This allows live migration procedures to exist and operate independent of how the isolation boundary operates. To these processes, the isolation boundary is an extension. For our case study, we did not consider a live migration. However, the case of saving VM's ephemeral state and resuming it after migration such that network connections remain valid involve the same technical challenges (process migration, storage management, etc.) except for the duration of the time the VM is inactive. Though live migration is not the focus of

our study, with the straight forward optimization of copying the VM image in advance and then using `rsync` to copy the intermediate state we were able to reduce the inactive time to tens of seconds. Adopting an implementation approach similar to Clark et al. [4] would reduce inactive times to milliseconds.

I. Middleboxes and TCP Options

Honda et al., in their paper [16] state that middleboxes today tend to either strip custom TCP options if they are part of the data stream or drop the packets altogether. However, if the custom options are used during the connection setup phase alone, then middleboxes tend to allow most of the traffic through. This finding is favorable to our approach where the isolation boundary options are only exchanged during connection setup phase (i.e., the 3-way handshake).

With our case study, we validate the hypothesis that if the the isolation boundary options are exchanged successfully, we will be able to enable uninterrupted communication following a VM migration beyond a network. However, if the options are removed, the network stack would fall back to legacy behavior and communication would take place until the VM is migrated to a different subnet. At this point, communication would stop and a new connection would have to be setup between the client and the server, with the server's new address. Unless the applications implement reliability such a discontinuation may require the application to reset.

J. Network Performance & Scalability

As the isolation boundary options only participate during connection setup or synchronization of state and not during the rest of the communication, there is no impact on the performance. Performance evaluation of the isolation boundary is presented in our prior work [7], [8].

For the network stack to scale in terms of managing large number of connections, the implementation needs to be thread safe. The fact that the proposed method is only active during the 3-way handshake — for connection setup or synchronization — significantly reduces the demand for locks, which could otherwise inhibited performance.

K. Security Considerations

Our proposal does not introduce any security threat greater than that to which TCP is already exposed. A flow can only be hijacked if the TIFIDs can be guessed correctly along with the sequence numbers. As we use the same methods of initializing TI sequence numbers as is done with TCP's sequence numbers, we do not introduce any risk greater than TCP. The response to an invalid request to synchronize a TI flow over a new TCP connection, with an invalid TIFID or TIAck, is a reset.

VI. Summary and Future Work

In this paper, we highlight that the migration of a virtual machine beyond a subnet is of significant importance. Until now, this goal has only been realized through ad-hoc means.

We make a case that the use of IP addresses as part of flow labels — to identify the transport connection — overloads the notion of network naming and that this is the fundamental reason that inhibits a clean approach for VM migration beyond a subnet. We suggest that decoupling the transport end point naming from IP addresses is not only possible, but is also efficient. We establish this claim by demonstrating seamless VM migrations between different subnets such that the application are oblivious to the migration.

We also note that there are occasions where there is a need to discover the context in which the communication is taking place (e.g., link status, network interface name). Subsequently, there is a need to act if there is a change in context. We intend to explore how existing network architectures may be extended to incorporate automated discovery, appreciation, and reasoning of the context in which communication takes place.

Acknowledgements

This research was funded in part by Virginia Tech and Juniper Networks.

References

[1] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," in *NSDI*. USENIX, 2007.

[2] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom *et al.*, "Application-Aware Virtual Machine Migration in Data Centers," in *IEEE INFOCOM*, 2011.

[3] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live Wide-Area Migration of Virtual Machines Including Local Persistent State," in *SIGPLAN VEE*. ACM, 2007.

[4] C. Clark, K. Fraser, S. Hand, J. G. Hansen *et al.*, "Live Migration of Virtual Machines," in *NSDI*. USENIX, 2005.

[5] D. Erickson, G. Gibb, B. Heller, D. Underhill *et al.*, "A Demonstration of Virtual Machine Mobility in an OpenFlow Network," in *SIGCOMM (Demo)*. ACM, 2008. [Online]. Available: http://conferences.sigcomm.org/sigcomm/2008/papers/p513-ericksonA.pdf

[6] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," in *CloudCom*. Springer-Verlag, 2009.

[7] U. Kalim, E. Brown, M. Gardner, and W. Feng, "Enabling Renewed Innovation in TCP by Establishing an Isolation Boundary," in *8th PFLDNeT*, 2010. [Online]. Available: http://pfld.net/2010/technical.php

[8] E. Brown, M. Gardner, U. Kalim, and W. Feng, "Restoring End-to-End Resilience in the Presence of Middleboxes," in *IEEE ICCCN*, 2011.

[9] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal *et al.*, "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," Internet Draft (Work in Progress), 2013. [Online]. Available: https://tools.ietf.org/html/draft-mahalingam-dutt-dcops-vxlan-02

[10] D. Eastlake, A. Banerjee, D. Dutt, R. Perlman *et al.*, "Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS," RFC 6326 (Proposed Standard), 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6326.txt

[11] K. Kompella and Y. Rekhter, "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling," RFC 4761 (Proposed Standard), 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4761.txt

[12] C. Perkins, "IP Encapsulation within IP," RFC 2003 (Proposed Standard), Internet Engineering Task Force, Oct. 1996, updated by RFC 3168. [Online]. Available: http://www.ietf.org/rfc/rfc2003.txt

[13] "TRILL in the Data Center: Look Before You Leap," White Paper, Juniper Networks, 2012. [Online]. Available: http://www.juniper.net/us/en/local/pdf/whitepapers/2000408-en.pdf

[14] J. Salz, A. C. Snoeren, and H. Balakrishnan, "TESLA: A Transparent, Extensible Session-Layer Architecture for End-to-end Network Services," in *USITS*. USENIX, 2003.

[15] A. C. Snoeren and H. Balakrishnan, "An End-to-End Approach to Host Mobility," in *MobiCom*. ACM, 2000.

[16] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh *et al.*, "Is it Still Possible to Extend TCP?" in *Internet Measurement Conference*. ACM SIGCOMM, 2011.