Wide-Area Virtual Machine Migration as Resilience Mechanism

Andreas Fischer*, Ali Fessi†, Georg Carle† and Hermann de Meer*
*University of Passau
Email: {andreas.fischer,demeer}@uni-passau.de
†Technische Universität München
Email: {fessi,carle}@net.in.tum.de

Abstract—**The resilience of services in the Internet has become an important issue and is expected to become even more important in the future. Virtualization is one of the means which can be deployed for resilience purposes. In this paper we follow a systematic approach to the use of virtualization to increase the resilience of network services. First, we provide an analysis of the potential failures of services running within Virtual Machines (VM) and how VM migration or replication can be used to address these failures. Then, we address the problem of re-establishing connectivity between a service and its clients upon successful migration, by leveraging results from mobility research. A special focus is given to wide-area VM migration, since it is considered as the solution for some difficult failures, e.g., large-scale failures due to natural disasters.**

I. Introduction

With the global internet hosting more and more critical services like Web or DNS, which our society increasingly depends on, network resilience has gained increased attention. Prominent outages have been featured on the news, already[1]. A network disruption can become even life threatening (e.g. if emergency calls make use of Voice over IP services). It is, therefore, important to find ways to strengthen and increase the resilience of network services. Network resilience has been defined as "the ability of the network to provide and maintain an acceptable level of service in the face of various faults and challenges to normal operation" [1]. Virtualization can achieve this goal by reducing the dependency on specific hardware through hardware abstraction. This creates virtual resources that can be managed more dynamically than the underlying physical resources. Those virtual resources then comprise a Virtual Machine (VM), in which a service can be executed. VMs can then be migrated from one physical host to another – possibly even without the service noticing. This allows an operator to react more flexibly to challenges that are affecting the hardware environment of the service. What remains to be done is to re-route client requests to the new location of the service in the network and to ensure that the client correctly receives the responses of the server. This includes directing new client requests to the correct location, and keeping existing connections from clients alive. It is particularly challenging when virtual machines are migrated between different subnets, a process known as wide-area migration. This paper presents a systematic correlation between failure classes and migration strategies. Moreover, two concepts to facilitate network redirection after performing a wide-area migration are evaluated.

[1]E.g. http://news.cnet.com/8301-10784_3-9878655-7.html

The rest of this paper is organized as follows. Section II introduces important VM migration concepts. Section III discusses how to use migration as a resilience mechanism. In section IV, two concepts for networ redirection are evaluated. Section V presents related work. Finally, section VI concludes the paper and gives pointers to future work.

II. Background

This section gives an introduction to the VM migration concepts used in this paper. First, migration itself is presented. Then, the concept of *snapshots* is explained. Finally, wide-area migration and the subsequent network recovery are discussed.

A. VM Migration

A VM running on a first physical host can be migrated to another physical host. The migration itself encompasses transfer of the persistent state of the VM (i.e. its file system), transfer of the volatile state of the VM (i.e. RAM contents and CPU state), and the redirection of network traffic. Once the state transfer is complete, the VM continues to run in the new physical machine. VM migration can be performed either as a *cold migration*, or as a *live migration*. During a cold migration, the service is paused or terminated while the state of the VM is transferred. The advantage of this type of migration lies in the low complexity. However, it may incur a significant service downtime, since the VM state can comprise several Gigabytes of memory. An attractive alternative in terms of resilience is live migration. During live migration, the service continues to run on the source host. The state on the target host is then updated iteratively until the state difference is small. Subsequently, the VM is stopped, the rest of the state is copied, and the VM is resumed on the target host, resulting in a downtime of less than $0.2s$ [2].

B. Snapshots

A snapshot of a VM is the state of the VM at a certain point of time. This may be only a persistent state (i.e., the VMs file

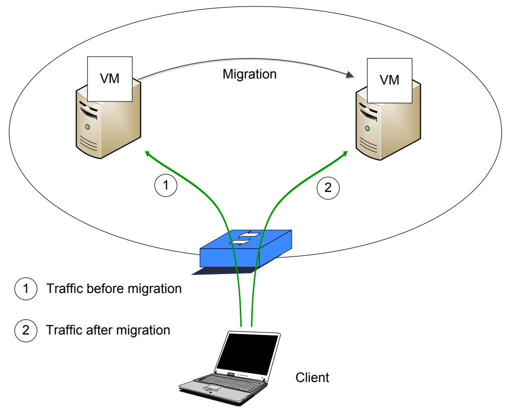

Fig. 1. Simple Network Recovery

system) if the VM was off when the snapshot was taken. Or it includes the memory and CPU state if the VM was running when the snapshot was taken. Snapshots can be stored in files and replicated in different locations in the network.

C. Wide-Area VM Migration

Wide-area migration describes the concept of migrating a VM across different subnets, resulting in a topological change of the network. In principle, wide-area migration uses the same concepts as local migrations. VMs can be migrated either cold or live. However, two factors add to the difficulty of wide-area migration. First, the bandwidth between source and destination site is likely to be severely limited in comparison to a local connection – i.e. the time needed for the state transfer will increase. This situation can be partly remediated by a proactive distribution of snapshots to potential target hosts. The second challenge lies in the fact that, due to the new location of the VM, with the implication of different addresses at the new location, the routing of network traffic for the VM has to be adapted – i.e. a network recovery has to be performed.

D. Network Recovery

In case of a cold migration with a VM reboot, the VM can acquire its new network configuration (IP and route) using legacy techniques like DHCP. In all other cases, the VM is resumed on another host with its old network configuration. However, the network configuration transferred from the source host may be not compatible with the target host. If the VM has been migrated within the same Local Area Network (LAN) segment, the target physical host forces an ARP update within the broadcast domain. Thus, from now on, all traffic addressed to the VM is sent to the target physical host, which itself forwards the traffic to the VM (see Fig. 1).

Given that this solution requires that the source and target physical hosts are in the same LAN segment, this limits the set of possible target hosts for a migration. In order to allow for a migration to a different LAN on the same site or to a different site, several solutions are possible. In Section III-C, we discuss

TABLE I
EXAMPLES OF FAILURE CAUSES CLASSIFIED ACCORDING TO [3]

	Failures			
	Crash	Omission	Timing	Byzantine
VM	Software buffer overflow	DoS attacks, high CPU or RAM usage		Conceptual software bugs
Host	Hard disk or CPU crash	High CPU or RAM usage by concurrent VMs		Bugs in hardware drivers
Network	Cable cut, routing failure	Network congestion, DoS attacks		Forged DNS or BGP messages

the solution space, present our solutions and compare them with related work.

III. MIGRATION AS A RESILIENCE MECHANISM

In this section, we analyze how virtualization, notably migration and replication of VMs, can help to address certain challenges, e.g., DoS attacks or natural disasters. First, we classify challenges (Section III-A). Then, we deduce the appropriate VM migration strategy (Section III-B). Finally, we explore the solution space for network recovery upon successful migration (Section III-C).

A. Failure Classes and Challenge Classification

According to Cristian [3], server failures can be classified into *crash*, *omission*, *timing* and *byzantine* failures. A crash failure occurs if a component does not respond at all. An omission failure occurs if a component does not respond to some input. A timing failure occurs if a component responds to input either too early or too late (also called *performance* failure). Finally, a byzantine failure causes the component to behave in a totally arbitrary manner outside of its specified behavior. According to this classification, crash failures are a subset of omission failures, omission failures are a subset of timing failures, and timing failures are a subset of byzantine failures. We use this classification as a starting point to identify in which cases virtualization is useful to address a certain challenge, and which parameters should be chosen to perform a VM migration, depending on the deployment scenario and the current challenge.

We further categorize failures into *VM failures*, *host failures*, and *network failures* (see Table I). VM failures occur at the VM itself. Host failures are all kind of failures (software and hardware) which may occur at the server hosting the VM. Network failures are all kind of failures that affect the network access of the server.

Moreover, we also classify failures into *predictable* and *non-predictable* failures. A non-predictable failure can be, e.g., due to an unknown software bug, or a cable cut due to construction work. A predictable failure can be, e.g., a power-supply failure where the infrastructure (i.e., network and servers) can still run on battery. This means that, unless power supply is recovered, the infrastructure will fail once the battery is drained. Another example of a predictable failure are natural disasters, like a huricane or a tsunami which is expected to hit a data center within the next few hours.

This distinction is particularly relevant for the feasibility of a VM migration. If the failure is predictable, then the remaining time until the failure actually occurs can be used to migrate the hot state of the VM. On the other hand, if, for example, an unforeseen failure causes the VM to be disconnected, then the hot state of the VM cannot be migrated in time, i.e. no live migration can be performed. Thus, the only option would be to fallback to a previous snapshot of the VM.

B. Selection of VM Migration Strategy

The type of challenge that a network service within a VM is facing has a direct implication on which migration or replication techniques are suitable. We assume that there are dedicated monitoring probes and failure detection mechanisms which can classify a failure into the appropriate category as given in Table I, taking into account whether the failure is predictable or not (which can be considered as a third dimension in addition to the two dimensions of Table I).

1) Recovering from VM Failures:

a) Crash Failures: Crash failures at the VM can be recovered by resetting the VM into a previously created snapshot. In this case, migration to another host is not required.

b) Omission and Timing Failures: In case of an omission or timing failure, if the failure can be traced back to a current overload situation at the VM, one can identify spare resources, either on the same host, or another host in the same site, or another site to start additional VM images and make them available. Note that a single omission or timing failure may not necessarily need to trigger any action. Thus, it has to be decided depending on the application requirements, and the number or frequency of omission of timing failures when a service migration or replication should be triggered. For this purpose, anomaly detection techniques are useful.

c) Byzantine Failures: Finally, as for byzantine failures at the VM, if the root cause of the failure can be unambiguously localized at the VM itself and not at the host or the network, a migration of the VM will most likely not solve the issue, since the failure will be migrated with the VM.

2) Recovering from Host Failures:

a) Crash Failures: If a crash failure at the host is predictable (e.g. a hard disk accumulating bad blocks, indicating its end of life), then a live migration can be performed as long as the host is functioning. If the crash failure is accidental, the VM can be recovered from a previous snapshot in the same site or on a different site.

b) Omission and Timing Failures: Omissions and timing failures at the host are an indication that the host does not have sufficient resources to meet the requirements of the service running within the VM. In this case, a VM migration or replication can be considered – to a different site, if resources are not available locally. Since the migration itself will consume additional resources, the approach of recovering VM state from previous snapshots, stored at a different location, in combination with recovery of the state difference can be considered. In case of VM replication to another host, a subset of the clients can be redirected to the new VM in order to release resources at the host where the failure occurs. This is related to network recovery (see Sec. II-D,III-C).

c) Byzantine Failures: Byzantine failures at the host can be addressed by a live migration if the state of the VM can still be migrated. Otherwise, a service recovery can be performed based on a previous VM snapshot on another host.

3) Recovering from Network Failures:

a) Crash Failures: Network crash failures result into service disconnectivity. In this case, a previous VM snapshot should be started at a different site, in combination with redirection of clients to the new VM.

b) Omission and Timing Failures: For omission and timing failures, VM live migration is a suboptimal solution, since the migration itself adds additional load to the network and the migration will last longer. Thus, the approach of starting a previously distributed VM snapshot at a different site should be considered. In this case, if possible, only the state difference of the VM has to be migrated.

c) Byzantine Failures: In case a Byzantine failure occurs in the network, a new VM image should be started at another site which is not affected by the failure. However, this approach has some limitations: In some cases of network failures, VM migration or replication may not be helpful to recover the service. For example, a DNS failure may make services running within the same domain unavailable. A routing failure may affect more than one site where the service is running, thus rendering it unavailable.

TABLE II
NETWORK RECOVERY SOLUTIONS

Layer	Through indirection	Through End-to-End
Application	Proxy	SIP, XMPP
Network	MIP with Home Agent, NAT, Layer 3 gateway	MIP Route Optimization
Data link	Layer 2 gateway	N2N

C. Wide-Area Network Recovery

Given the challenges explained above that can encounter network services, simple network recovery, as presented in Section II-D, is considered to be insufficient. In some cases it is necessary to migrate the service to a different geographic location, i.e. to perform wide-area migration. However, wide-area migration results into a *mobility* problem which may render the service unreachable unless network recovery is performed. In fact, several mobility solutions are applicable here. In Table II, we explore the solution space for network recovery after successful migration based on the mobility layer. Moreover, at each layer solutions can be classified into *mobility using an indirection point* and *end-to-end mobility*.

1) Network recovery with indirection: At the application layer, traffic is forwarded to the VM upon successful migration via an application-layer proxy remaining at the source network. Fig. 2 provides an overview of this setup.

After migration of the VM from site A to site B, traffic received by `proxy.a.example.com` is forwarded to `proxy.b.example.com`, which itself forwards the traffic to the migrated VM. This solution can be particularly attractive since many applications support the use of proxies.

At layer 3, a well known approach with an indirection point is Mobile IP (MIP) (RFC 3344, RFC 3775). In its simple variant, all traffic sent to the so-called Mobile Node (MN), in this case the VM, is sent to a gateway at the "home network" called Home Agent (HA). The HA then forwards the traffic tunneled to the MN in its new location.

Another approach to support wide-area VM migration is hiding the VM behind a Network Address Translation (NAT) router. In this case, the traffic from the client to the VM is sent to the external IP address of the NAT router which forwards the traffic to the VM. Thus, the VM migration is transparent to the client[2]. Another option at layer 3 can be a VPN gateway to which the VM connects after migration.

The same approach with a VPN gateway can be performed with a layer 2 gateway, or a large scale layer 2 switching fabric which may span different geographic locations [4].

A major drawback of solutions with an indirection point (application proxy, NAT, MIP HA, VPN gateway) is that the indirection point must remain available in the upcoming communication. This problem can be alleviated by giving the DNS record of a server running within the VM a short TTL (e.g. 60 seconds). After that time, all new incoming connections to the VM should arrive at the new location of the VM. Unfortunately, this does not solve the problem for two reasons. First, there are still clients which are currently connected to the server at the old location. Second, although the TTL is configured to be short, the DNS cache can still persist longer in several locations in the network, at the application, or the OS. These drawbacks are particularly important in cases in which the ongoing availability of the indirection point is not guaranteed. For example, in case of power failure or natural disaster, not only the hardware of the physical hosting where the VM was running may be affected, but also the hardware where the indirection point is running. An additional drawback, that is valid for MIP as well, is triangular routing, which introduces additional delay.

2) Network recovery with end-to-end notification: Given that wide-area migration with an indirection point does not cover certain resilience requirements, we also investigated end-to-end notification without indirection point. An existing approach at layer 3 which supports end-to-end notification is MIP route optimization. In this case, the communication party of the MN, called Corresponding Node (CN), i.e. the client in Fig. 2, is notified about the mobility event. Then, further communication between the MN and the CN (here: between the VM and its clients) takes place without involving the HA.

At the application layer, the application can support such a notification only if the application protocol allows for asynchronous notifications from the server to the client. This is the case, e.g., for SIP or XMPP.

[2]Some applications, like SIP or FTP, unfortunately transport IP addresses in their payload, making interaction with NAT routers complicated.

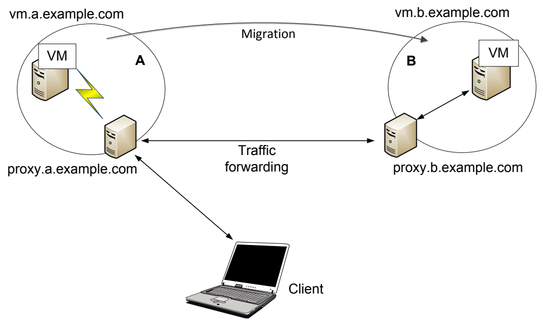

Fig. 2. Network recovery with an indirection point after wide-area migration

In layer 2, N2N [5] can be considered as an end-to-end mobility mechanism. N2N is a P2P layer 2 VPN which could be used to propagate the new location of the VM in the P2P network and re-direct the layer 2 traffic to the VM after successful migration. In this case, both the VM and the client must be part of the P2P network.

IV. Implementation

As discussed in Sec. III-C, there are failures which require wide-area migration as a remediation. In this section, two variants of network recovery after wide-area migration are presented. The two approaches differ in the way in which traffic redirection is achieved. In the first case, traffic redirection is performed on the link layer by building a tunnel to the target location of the virtual service. The second approach performs traffic redirection on the application layer by notifying clients about the new location of the service. The following experiments were tested on dedicated 2 GHz machines with 2-4 cores and 2-4 GB RAM, connected via Gigabit Ethernet.

A. Link Tunneling

If a failure does not affect the access router of a service, network redirection can be performed by creating a virtual link – i.e. a tunnel between the original access router of a service and its new location. In this scenario, the access router has to actively support wide-area migration by first setting up the tunnel, and then redirecting all incoming network traffic for the migrated service to its new location. Given that it is a layer 2 solution, it has the advantage of being transparent to the service, without any need for the VM to change its IP.

Fig. 3 shows the network setup we used for this kind of migration. A VM is to be migrated from Host A to Host B over an access router. Host A is connected to the access router via Network A, whereas Host B is connected via Network B. A client, which is connected to the access router via Network C, uses the service during migration. To facilitate the migration, several steps have to be performed. First, an additional virtual bridge, named *sshbr*, is created on Host B. Next, a link layer tunnel is created between *vmbr1* on the access router and the

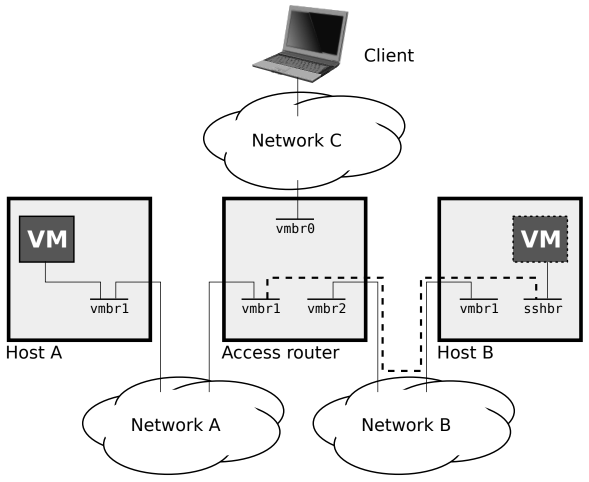

Fig. 3. Wide-area migration enabled by tunneling

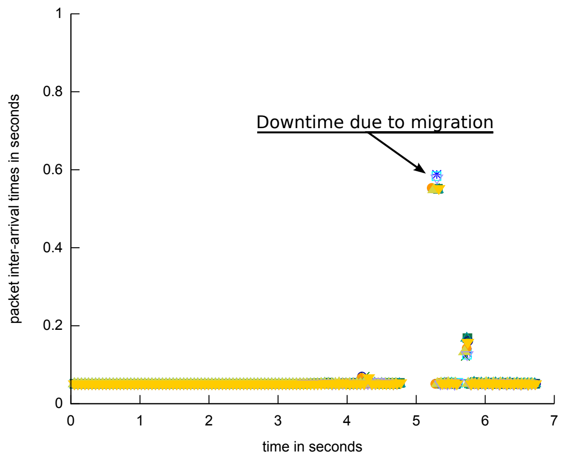

Fig. 4. Downtime of a virtual service during migration

newly created *sshbr* (indicated by the dashed line). This causes the virtual bridge *sshbr* to be logically in Network A. Finally, a standard live migration between Host A and Host B is initiated, where the VM in its new location is connected to *sshbr* instead of *vmbr1* on Host B. Using this technique, all network traffic can be transparently redirected to the VM. This is verified by the client, who accesses the VM during migration.

This scenario has been evaluated to measure the impact of wide-area migration on service downtime. The results are shown in Fig. 4. Connectivity of the virtual service is checked by sending UDP messages from the VM to Host A at a fixed rate of 20 packets per second. By measuring the inter-arrival times of packets at Host A we can deduce the downtime of the VM. Host B is equipped here with a pre-distributed snapshot of the VM. Migration starts about $4.2s$ into the experiment. At about $4.9s$ into the measurement, the live migration halts the original VM and network traffic is redirected to the new location. One can see that the resulting downtime is below $0.6s$. We have also measured latency increase due to the tunneling mechanism – however, the results were neglectible in our test-bed. This effect should be studied with a larger test-bed in order to reach meaningful results.

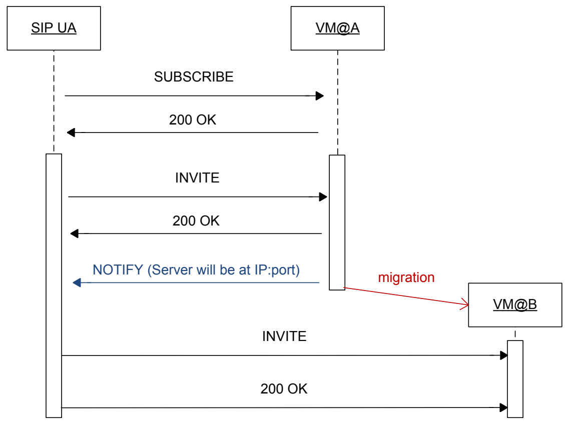

Fig. 5. Message flow: wide-area VM migration with end-to-end notification

B. SIP Signalling

The second approach implements VM wide-area migration with an IP address change. We use SIP-based end-to-end notification to propagate mobility events. Thus, according to Table II this is an application layer end-to-end approach. SIP User Agents (UA) initially connect to the SIP server hosted within the VM. Each message sent by a UA is sent to the SIP server and potentially forwarded to other SIP entities, e.g., other SIP UAs. SIP UAs initially find the SIP server with a DNS request according to RFC 3261. At this point comes our contribution: Upon successful registration, SIP UAs subscribe to the set of events that are relevant to keep the connectivity between them and the server. For this purpose, we use SIP Specific Event Notification (RFC 3265). The SIP server acts as a publish/subscribe system where server mobility events are propagated. Each UA which has subscribed to that kind of event receives a notification when such an event occurs.

For our implementation, we used and extended the publicly available implementation of the SIP stack in the Python programming language. It uses the event-based network programming framework twisted[3]. Fig. 5 shows a message flow between a SIP UA and the SIP server, which results from the collection of our Wireshark traces (For simplicity, we show only the relevant SIP messages). As we can see, the SIP UA can successfully establish a SIP session with a SIP INVITE message both before and after the VM migration.

Beyond this functional test to validate our implementation, we launched preliminary performance tests as well to estimate the service downtime. Despite the IP address and network configuration change of the VM, the downtime was the same order of magnitude as the results described in Sec. IV-A.

[3]http://twistedmatrix.com/

C. Comparison

The two investigated approaches each have their advantages and disadvantages. The link tunneling approach keeps the network configuration of the VM unchanged. This is in particular beneficial for legacy services that can not be changed to take topological mobility into account. However, the approach requires access to the access router of the VM at its original location. This can be a problem if the access router is affected by the network challenge itself (e.g. in the case of a natural disaster). In contrast, the SIP signaling approach indicates how network recovery on the application layer can mask an IP address change. This is beneficial, as traffic is routed more directly, and does not require a cooperative access router. However, the migration is not transparent to the application anymore. Both approaches achieve a downtime below one second – which is significantly better than no action and should be acceptable for most services.

V. Related Work

Live migration has been implemented multiple times already – see in particular [2] and [6]. However, they only addressed migration within a data center, not wide-area migration.

Replication of VMs has been previously discussed as a resilience mechanism (see [7], [8]). These approaches, however, focus on the duplication of a service, without discussing the implications of wide-area network traffic redirection.

Bienkowski et al. [9] discuss service migration in virtual networks. Their work investigates a simulated mobile scenario, where services are kept topologically close to their (mobile) clients. In contrast to the approach presented here, network recovery is not discussed in their work.

Travostino et al. [10] present a method for wide-area live migration of virtual machines. Similar to the first approach presented in this paper, a tunnel is used to redirect network traffic after the migration. However, their approach requires the (physical) source host to remain operable even after migration, in order to forward tunnel traffic.

Bradford et al. [11] present another approach of wide-area migration. They achieve traffic redirection by a combination of tunneling and Dynamic DNS updates. In contrast to our work, they do not consider failure of the source host explicitely. In their scenario, the source host only can be switched off after DNS updates have been propagated and all connections that were opened before the migration, have been closed.

In Section III-C, we mentioned MIP as an alternative for maintaining connectivity between a VM and its clients after migration. In fact, this has been implemented and evaluated by Silvera et al. [12].

An interesting concept is introduced in [13], where the authors propose live migration of routers. Unlike the paper presented here, their work focuses exclusively on routing as the virtual service.

VI. Conclusions and Future Work

This paper presents wide-area migration of VMs as a resilience mechanism. The constraints imposed by various network challenges on VM migration have been discussed. The need for wide-area migration has been stressed. Two approaches to facilitate network recovery after wide-area migration have been evaluated and compared. It has been shown that virtualized services can be migrated across different subnets with a downtime of less than one second.

The presented approach has been evaluated in a lab environment with controlled parameters. One step to extend this work is to use distributed testbeds in order to assess the impact of limited bandwidth, unreliable links, and similar restrictions.

Acknowledgment

The authors would like to thank Yahya Al-Hazmi for previous discussions on VM migration and network resilience and Philip Huppert, Jochen Ritzel, and Andreas Heider for implementation work.

The research leading to these results has received funding from the EC's Seventh Framework Programme ([FP7/2007-2013] [FP7/2007-2011]) in the context of the ResumeNet and EuroNF projects (grant agreement no. 224619 and 216366).

References

[1] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines," *Computer Networks: Special Issue on Resilient and Survivable Networks (COMNET)*, vol. 54, pp. 1245–1265, 2010.

[2] 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. 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, 2005, pp. 273–286.

[3] F. Cristian, "Understanding fault-tolerant distributed systems," *Commun. ACM*, vol. 34, pp. 56–78, 1991.

[4] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *Proc. ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 39–50.

[5] L. Deri and R. Andrews, "N2N: A Layer Two Peer-to-Peer VPN," in *Autonomous Infrastr., Management and Security*, 2008, pp. 53–64.

[6] M. Nelson, B.-H. Lim, and G. Hutchins, "Fast transparent migration for virtual machines," in *USENIX Annual Tech. Conf.*, 2005, pp. 25–25.

[7] H. A. Lagar-Cavilla, J. A. Whitney, R. Bryant, P. Patchin, M. Brudno, E. de Lara, S. M. Rumble, M. Satyanarayanan, and A. Scannell, "Snowflock: Virtual machine cloning as a first-class cloud primitive," *ACM Trans. Comput. Syst.*, vol. 29, pp. 2:1–2:45, 2011.

[8] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield, "Remus: high availability via asynchronous virtual machine replication," in *Proc. 5th USENIX Symposium on Networked Systems Design and Implementation*, 2008, pp. 161–174.

[9] M. Bienkowski, A. Feldmann, D. Jurca, W. Kellerer, G. Schaffrath, S. Schmid, and J. Widmer, "Competitive analysis for service migration in vnets," in *Proc. 2nd ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, 2010, pp. 17–24.

[10] 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 Gener. Comput. Syst.*, vol. 22, pp. 901–907, 2006.

[11] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg, "Live wide-area migration of virtual machines including local persistent state," in *VEE '07: Proc. 3rd international conference on Virtual Execution Environments*, 2007, pp. 169–179.

[12] E. Silvera, G. Sharaby, D. Lorenz, and I. Shapira, "IP mobility to support live migration of virtual machines across subnets," in *Proc. SYSTOR 2009: The Israeli Experimental Systems Conf.*, 2009, pp. 13:1–13:10.

[13] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: live router migration as a network-management primitive," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 231–242, 2008.