Live Replication of Virtual Machines

VIOLETA MEDINA
Universidad Michoacana de
San Nicolás de Hidalgo
División de Estudios de Posgrado
Facultad de Ingeniería Eléctrica
Francisco J. Múgica S/N,
Col. Felicitas del Río, 58030
Morelia, Mich.
MÉXICO
medinav@acm.org

JUAN MANUEL GARCÍA
Instituto Tecnológico de Morelia
Computer System Department
1500 Tecnológico Avenue,
Col. Lomas de Santiaguito, 58120
Morelia, Mich.
MÉXICO
jgarciagarcia@acm.org

Abstract: Replication has been identified as a useful mechanism in Virtual Machine *(VM)* management. Replication allows the network administrators to face some daily troubles like system faults, load balancing and scheduled/unscheduled maintenances. A desirable characteristic during this process is zero downtime, the replication process should cause a minimal service interruption if a Virtual Machine becomes unavailable. Replication has been identified as a tool to achieve high availability in a computational system. In this paper a live replication protocol to reach high availability in services for virtual machines is proposed. It consists on replicating every task that a virtual machine executes to other virtual machine. Each individual result that is obtained in a virtual machine called *original* is also obtained in a Virtual Machine called *replica*. The protocol is based on TCP/IP. This mechanism duplicates each command that the original virtual machine receives and sends it to the replica virtual machine.

Key–Words: Virtual machine, replication, high availability, virtualization, TCP/IP, network.

1 Introduction

In this paper a replication protocol for virtual machines is presented.

High availability[1] has been one of the goals of works developed for cluster environment and distributed systems[13]. The high availability systems have proposed mechanisms, such as migration and replication in order to achieve disponibility in computational systems[9]. It is considered that the migration process in virtual machines copies the state of a virtual machine of one physical machine to another physical machine (state in memory, hard disk and network connections), this process can completely stop the services that a virtual machine provides. On the other hand, the replication process can simultaneously copy the virtual machine state to more than one place and it is considered that the virtual machine can continue working without stopping services.

Both mechanisms have common objectives, such as allowing load balance, fault-tolerance and hardware upgrades. And both techniques along with the above mentioned characteristics provide High Availability.

Some Virtual Machine Monitors *(VMM)* have created their own migration technology and have incorporated it as part of their environment, such as Xen [5] and VMWare [18]. In this kind of mechanism a running virtual machine changes its physical ubication from one machine to another. This process should be transparent for the virtualized environment. The original VM should not stop its services while the machine state is transfered to the destination virtual machine and the downtime should be minimal when this process is running. As a result of this process, a functional virtual machine in a new location that keeps up the original virtual machine state is expected.

Replication is a widely used technique in distributed systems, such as file systems [13] or data bases [7], where the system is divided into different fragments and these fragments can be copied in two or more sites.

The protocol proposed in this paper is based on replication, that is, the complete state of a virtual machine is replicated from one VM to another VM. For this purpose, each operating system command that is executed in the original virtual machine is replicated to an image virtual machine. Virtual machines are connected in an Ethernet local network and the data transmission is realized via TCP/IP protocol [10, 22].

ISBN: 978-960-474-277-6

Every TCP/IP packet is captured and modified in order to be transfered to an image VM. This prototype was developed on base the replication of TCP/IP commands because TCP/IP is a protocol that guarantee the data reception through acknowlegments and delivery package in the order in which packets were sent. Filtering of TCP/IP traffic is made using the *libpcap*[15] functions. The packet injection was made by the use of *libnet*[14] library.

This prototype protocol was developed and tested using the facilities that Xen hypervisor provides in a system composed by heterogeneous computers.

2 Related Work

Several replication and migration methods to achieve fault-tolerance and high availability have been proposed. Some of them consider storage migration and others assume that a SAN or NAS strategy is used among VMs. In [4] a prototype hypervisor based in instruction set architecture replication was built. The hypervisor coordinates a primary VM with its backup, both execute the same sequence of instructions and produce the same results. Xen[3], one of the most well-known open source VMM integrates in [5] a live migration mechanism, where the memory migration of a virtual machine is based in *pre-copy*[17] approach. The memory pages are iteratively copied from a source VM to a destination VM and during this process the source VM continues running. It is considered to migrate the open network connections through an unsolicited ARP reply from the migrated host, notifying that the IP has a new location. Storage migration is not necessary because a shared storage form, like *NAS* is assumed.

VMWare developed a migration mechanism called *VMotion* in [18], where a memory migration similar to Xen is followed. A *pre-copy* memory scheme is used. First, the physical memory from the source VM is copied and marked as read-only, so any modification can be detected by the VMM. When this process is finished modified pages by the VM in execution may exist. These pages are copied to the new VM. This step is repeated until the remaining number of modified pages is small (16 megabytes) or there is a reduction in changed pages of less than 1 megabyte.

VMWare provides a Virtual Ethernet Network Card, *VNIC*, as part of its virtual platform, which has a unique MAC address within the local network. A VNIC can be associated to one or more physical network cards. Due to the VNIC has a MAC address independent of the physical MAC address, the VMs can be moved from one host to another without stopping services and keeping the active network connections. This is possible, only if the new VM is opened in the same subnet as the original machine.

In storage, VMWare assumes that the computers are connected to a server SAN o NAS.

Internet Suspend/Resume *(ISR)*[12, 19] presents a project where the complete state of a VM can be migrated. ISR is based on the idea of a VMM encapsulates the volatile execution state of a VM and a VMM transfers the state of their VMs to files in the local file system within the host machine. When a VM is suspended, the volatile state is transfered to files until a suspension point. These files, including the operating system are copied to the remote machine, where the VM can be reassumed. The downtime depends directly of the file size to be transfered.

In [6] an enhancement to the migration mechanism of Xen is presented. This mechanism, called Remus, was incorporated to Xen as a software layer that improves live migration of VMs. It reaches high availability by replicating frequent checkpoints of a whole active VM to a physical backup. On the backup, the VM image persist active in memory, as a receptacle that can become executable if a failure happens. All modifications in the VM are stored in a buffer in the backup host until the complete state is received and an acknowledgement is sent to the original VM when the checkpoint is complete. Then, the image in the backup is considered as consistent. This checkpoint process frequently occurs, up to forty times per second. The checkpoint includes the network and disk state. It is considered that more than one backup can be preserved at the same time. Even though, this mechanism has reached a high degree of efficiency in the migration process, this contribution proposes a replication protocol that can be independent of the installed VMM and does not requiere to modify the platform where is executed.

In [2] a detailed live migration analysis of Xen in an cluster environment is presented and a reduction in the iterations number in pre-copy phase is proposed in order to optimize a High Performance Computer environment.

A synchronization approach for VMs based in Xen is proposed in a project called *Kemari* [23]. In this project, when Kemari detects an event from the guest domain, this domain is paused, the dirty pages created since the last synchronization are located and sent to the image VM. When the synchronization is completed, Kemari unpauses the guest domain and the event is executed in domain 0.

In [16] a live and incremental migration of VMs has been proposed, it includes memory, CPU and disk state migration. It is used a Three-Phase Migration *(TPM)* algorithm (pre-copy, freeze-and-copy, post-copy), which is an expansion of live migration of

Xen where an Incremental Migration *(IM)* is used for storage. During the pre-copy phase, the storage data are iteratively pre-copied. In the first iteration, all the storage data should be copied to the destination VM. After, modified data, just during last iteration is sent to backup VM.

In the TCP packet transmission area, in [21] a performance analysis of packet path diversity is shown. The acknowledgements (ACKs) from client to server are duplicated and sent through two different networks paths. The comparison between the two arrival times to the receiver gives the best path and the data packets can switch to a better traffic route.

3 Migration vs. Replication

In this article, the migration and replication are treated as different concepts. Some advantages and drawbacks are cited in the following lines.

3.1 Migration

For migration mechanism some advantages can be mentioned as:

- Memory migration has reached a high degree of sophistication and efficiency, downtime can be reduced just to milliseconds [2, 5] so, this process does not affect the final user.
- The use of technologies of network storage like *NAS* or *SAN* or distributed file systems allow storage migration to be considered as not necessary, due to the fact that storage can be accessible from any point in the network.

Migration of virtual machines can show drawbacks like:

- Usually, migration is explicitly activated. If a VM undergoes modifications after a migration process, these changes will not be in the backup VM.
- The total migration time can be very high if a storage migration is considered. This time can be directly proportional to the storage size.

3.2 Replication

In replication, some advantages can be mentioned:

- The changes made to the original VM can be transfered almost immediately to the destination VM. There is no need for a explicit suspension of the VM; the replication process can be present during the continue execution of a virtual machine. If there is a failure in the system, there would not be loss of data.
- The data replication can be simultaneously made to more than one machine.
- The modifications in local storage of a virtual machine are considered as part of the replication process.
- An exact copy of a virtual machine can be reassumed if a failure occurs.
- This process does not require shared storage, then replicas could be stored in remote networks.

It can be mentioned that the most important difference between a migration and a replication process is that migration requires that the original VM remains available during most of the process, while the replication process allows that any replica VM becomes active and assumes the place of the original VM. So, the replication process could be executed from the new original VM due to all copies contain the same state.

On the other hand, some disadvantages can be listed:

- If the number of replicas grows up, this can cause traffic network, due to sending/receiving of many copies of the same data through the network.
- Complicated synchronization protocols must be implemented to maintain the data consistency.

4 Protocol

In this section a description of the proposed replication protocol, the experimental environment and the obtained results are presented.

4.1 Protocol description

This contribution proposes the design and implementation of a replication protocol of virtual machines. In the first part of this section, a general ideal of the protocol is described and in the second part, the specific implementation is explained.

4.1.1 Protocol idea

It is considered that a computer *A* exists, which is exchanging information with a computer called *B*. The machine *A* can modify the complete state of the machine *B* through remote commands. There is a third machine called C, which initially has the same state of B. The protocol will duplicate every action on B to C. The process of replication is transparent for *C* and it assumes that is working directly with *A*.

In a general way, the operation of the replication protocol is illustrated in Figure 1.

ISBN: 978-960-474-277-6

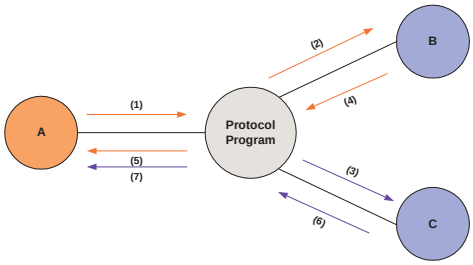

Figure 1: Replication Protocol

The protocol operates as follows:

1. A remote command is launched from the computer *A*, having as destination the computer *B*.
2. Package filtering is applied and if a package with B's destination is detected, this package is duplicated and sent to a waiting queue, where it will wait until can take its correct place within a communication conversation between *A* and *C*.
3. The caught package is extracted from the queue. Depending on the protocol package, the required fields are changed. And the modified package is sent to computer *C*.
4. Computer *B* returns an answer to *A*. This package does not suffer any changes.
5. The package continues its normal path and is received by *A*.
6. Computer *C* sends a response package.
7. The answer from *C* is captured and dropped in order to avoid that the conversation between *A* an *C* could be reseted.

This prototype protocol will only capture Local Area Network traffic, hence packets directed to B can always be intercepted. Within the LAN can exist more than one computer that assumes the role of computer *A*. The error recovery philosophy of the proposed protocol replication is based on TCP/IP mechanisms to establish a reliable communication between hosts. TCP[10] is able to recover corrupted data, data loss, duplicated information or packets out of order. This prototype as TCP assigns a unique sequence number to each packet that will be transmitted to the replica VM and requires an acknowledgement (ACK) from the receiving VM. If the ACK is not received within a timeout, the packet is retransmitted by TCP/IP protocol. On base to the sequence numbers, the receptor can control the correct order in segments that may be received out of order and eliminate duplicated packets. The proposed protocol implements a flow control mechanism through a "sliding window"[11]. The number of bytes sent and received by the client and server of the replica conversation are counted and according to this byte account some pointers within the structure of the sliding window are modified, these pointers save information about bytes sent and acknowledged, or bytes sent but not yet acknowledged for both the sender and the receiver. The sliding window indicates the next sequence number that should be sent and the next sequence number that should be received.

Damaged data is detected by adding a checksum to each segment transmitted. The checksum is verified by the receiver, and discarded if a error in this field is found.

4.1.2 Protocol Implementation

The protocol performs a copy of each remote command executed on the original virtual machine and replicates these operations on a virtual machine called *replica*.

The final objective of this protocol is to maintain a state of complete equality of a VM, including memory, local storage disk and network connections in other VM. TCP/IP based commands can be replicated with this protocol from a *original* VM to a *replica* VM.

In the implementation of this protocol two open source libraries were used, *Libnet*[14] and *Libpcap*[15]. Libnet includes functions and procedures of network that provide access to several protocols. It allows the injection and modification to network packages in protocols like IP, TCP, UDP, IGMP, etc. Libpcap is an open source library written in C language, it offers an interface able to capture packages in the network layer and package filtering in a similar way to the system operating command, as tcpdump does.

Figure 2 shows the actual prototype and the implementation of this protocol can be described with the following steps. The replication protocol program was installed in a real machine from where remote commands were launched. These instructions had as destination the virtual machine called *original*. The packages sent by the real machine to the *original* virtual machine were captured and some fields in IP header and others that belong to TCP header were modified in order to change the final destination of the package (*replica* VM). In the IP header, the original destination address of the packet was replaced by the IP address of the *replica* virtual machine.

The TCP header undergoes modifications in the fields sequence number and acknowledgement number, which were calculated with RFC 793[10]. The connection is established according to TCP the three-way handshake and the replica conversation follows

ISBN: 978-960-474-277-6

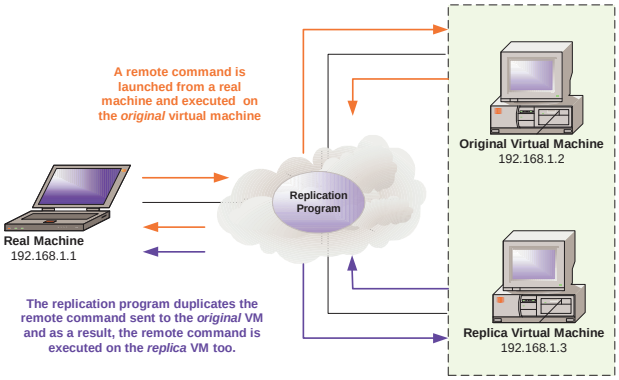

Figure 2: Replication scheme

the following steps:

1. A SYN package is sent from the real machine to the original VM. So, the replicated package has the same sequence number as the communication conversation between the real machine and the original machine. The acknowledgement number is initialized to zero.

2. As second step, the server *(replica VM)* replies with a SYN+ACK. The sequence number in the original and replica conversation are random numbers, which are different between them. The acknowledgement number in both conversations is set to one more than the sequence number in step 1.

3. In the third step, an ACK is sent by the client *(real machine)*. In both conversations, the sequence number is equal to acknowledgement number of the package in step 2 and the ACK number is set to the sequence number of step 2 plus 1.

During the replica conversation, the order in which the data is sent in the original information transfer is not modified, but taking as reference the original conversation, some ACK packages can be inserted or deleted in the replica conversation according to the data that must be acknowledged by the replica VM.

Due to the modifications in other fields, the checksum was recalculated with a TCP pseudoheader as is indicated in RFC 793. In TCP options field, the values, timestamp value *(TSval)* and timestamp echo replay *(TSecr)* were modified to be recognized as a valid package.

In the real machine, a rule for dropping packages from replica VM is added with *iptables*, in order to avoid that packages from replica could be reseted during the replication process.

A package filtering is applied to the traffic network and just the packages that belong to the TCP protocol and a conversation between the real machine and the *original* VM or between the real machine and the *replica* VM are caught.

In Figure 3 a flowchart of the general operation of the replication protocol program is shown.

5 Evaluation

In next section, the experimental environment is described and test cases of the proposed protocol are presented.

5.1 Experimental environment

Two machines were used for the tests to be reported next. A server with two Xeon quad-core 2.0 GHz, 3 GB RAM and a hard disk of 72GB was used. The base system operating was Debian 4.0, and the hypervisor Xen 3.1.3. The other computer was a laptop with Pentium M processor, 2 GB RAM, a hard disk of 80 GB, and base system operating Debian 4.0. Two virtual machines with the same characteristics were configured. Both have as guest system operating to Ubuntu 8.04, 512 MB of RAM memory and 2 GB in storage. Besides, the software environment in both VMs was identical.

5.2 Test Case 1. Consistency

The first test case was focused on verifying the replica consistency. For this purpose, the following test was made. In a Local Area Network, a real machine with IP address `192.168.1.1` launched remote commands, the final destination of these commands was a virtual machine with IP address `192.168.1.2` and the replication program caught and reproduced them, in a virtual machine with IP address `192.168.1.3`. Remote commands were executed with *rsh (Remote Shell)*. As requirement, the virtual machine to where the command is directed must have the rshd demon running. This demon uses the TCP port 514.

The real machine can execute remote commands like copy of files, change the access mode of a file, change the owner of a file, create or delete a file, etc.

For this case, an installation of the Web Server Apache [8] using remote commands on the original VM machine and its replication to the replica VM was realized.

The installation of Apache was achieved by a sequence of commands shown in Table 1.

Before the execution of this test, in each VM the open source application *Tripwire*[20] was installed. This tool is useful for monitoring integrity and data security, because it alerts about the changes in specific

ISBN: 978-960-474-277-6

Figure 3: Replication protocol flowchart program

ISBN: 978-960-474-277-6

Table 1: rsh commands

Remote commands
rsh 192.168.1.2 /root/configure
rsh 192.168.1.2 make -C /root
rsh 192.168.1.2 make install -C /root
rsh 192.168.1.2 /usr/local/apache2/bin/apachectl start

files inside the system. *Tripwire* works as a intrusion detection system, but instead of detecting intrusions in level of network layer, *tripwire* detects changes in the objects of the file system.

When tripwire initialize its database, the application scans the complete file system and stores information in the database about each file. After the database has been created, the registered files will be reviewed and the result obtained will be compared against the values saved in the database. If there are changes, it is reported to the user. Cryptographic hashes are used to detect modifications in a file without saving the whole content of a file in the database.

Thus, before the execution of the remote commands, the file system state was registered in the *tripwire* database of both VMs (original and replica). This first register was considered as a consistent state in both machines, that is the reason because the registers were initialized to zero. After the remote commands execution, the integrity was reviewed again with *tripwire*.

Since the replication program was running, the remote command was executed in the image VM too.

Two registers from tripwire in the original and replica VMs were obtained. From the comparison of both, it can be observed that after the execution of the remote commands on the *original* VM, there were reported changes in the Root configuration. There was detected the addition and modification of files due to the installation of Apache. These modifications were also shown in the *replica* VM. Table 2 presents some lines taken from the integrity check report of *tripwire* in the replica VM.

Table 2: Changes in configuration files of the replica VM according to *tripwire* reports

Rule name	Added	Removed	Modified
Root config files(initial)	0	0	2
Root config files(final)	779	0	213

These rows reveal some changes detected according to the tripwire policy rule for root configuration *(*Root config files)*. It can be observed that the *replica* VM suffered several changes in comparison with the initial report, however these changes were also reported in the original virtual machine. Therefore, the states of both VMs are consistent. The changes that took place in the *original* VM were reflected in the *replica* virtual machine.

5.3 Test Case 2. Performance

The same installation of Apache has been replied to more than one VM. In Table 3 the average time that every command that is part of the installation is illustred.

Table 3: Average times in Apache installation with several replicas (sec.)

#Rep	configure	make	make install	apachectl
0	41.68	122.26	10.79	0.03
1	42.38	122.80	11.48	0.07
2	43.59	123.80	12.22	0.12
3	44.93	124.46	12.14	0.17
4	53.33	126.92	17.23	0.47

In Table 4 the latency in relation to the results without replicas is shown. It can be observed that in the most critical case the latency can increase until 97.11% in relation to the spent time in the installation without replicas, but this command is executed in hundredths of a second without replicas and the time increases to almost a half of a second, which can be considered acceptable.

Table 4: Latency

#Rep	configure	make	make install	apachectl
1	1.65%	0.44%	6.00%	56.82%
2	4.37%	1.25%	11.72%	74.59%
3	7.23%	1.77%	11.14%	81.91%
4	21.85%	3.68%	37.38%	93.40%

6 Conclusion and future work

A prototype of a replication protocol able to duplicate commands based on TCP/IP protocol has been implemented. The protocol replicates commands under the *rsh (Remote Shell)* protocol in a backup VM without interrupting the services in the active VMs. The protocol synchronizes sending/receiving of packages in both TCP/IP conversations and the communication can be done without loss of information. Modification in local storage in VMs can be replicated with this protocol eliminating the necessity of a kind of network storage. This prototype can be applied to produce more than one replica VM, where network

ISBN: 978-960-474-277-6

latency by queuing and processing delays can be observed, but in the best case can be noticed that latency increases just 3.68% with 4 replicas.

As future work, tests in a Wide Area Network are considered and because the latency is expected to grow, the use of techniques of parallel programming, like multithreading to reduce latency are contemplated. The addition of encrypted protocols, such as ssh (Secure shell) is considered as an immediate improvement. As a future work the construction of a ssh tunnel between virtual machines, in order to perform secure transfers, will be incorporated.

References:

[1] T. Adelstein, F. Timme, and B. Lubanovic. *Linux System Administration*. O'Reilly Media, Inc., 2007.

[2] M. Atif and P. Strazdins. Optimizing live migration of virtual machines in smp clusters for hpc applications. *Network and Parallel Computing Workshops, IFIP International Conference on*, 0:51–58, 2009.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM.

[4] T. C. Bressoud and F. B. Schneider. Hypervisor-based fault tolerance. *ACM Trans. Comput. Syst.*, 14(1):80–107, 1996.

[5] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[6] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: high availability via asynchronous virtual machine replication. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 161–174, Berkeley, CA, USA, 2008. USENIX Association.

[7] A. Downing, I. Greenberg, and T. Lunt. Issues in distributed database security. In *Computer Security Applications Conference, 1989., Fifth Annual*, pages 196–203, Dec 1989.

[8] T. A. S. Foundation. The apache software foundation. http://www.apache.org/, 2010.

[9] R. Hurley and S. A. Yeap. File migration and file replication: a symbiotic relationship. *Parallel and Distributed Systems, IEEE Transactions on*, 7(6):578–586, Jun 1996.

[10] I. S. Institute. RFC 793, sep 1981. Edited by Jon Postel. Available at http://www.rfc-es.org/rfc/rfc0793-es.txt.

[11] V. Jacobson, R. Braden, and D. Borman. Tcp extensions for high performance, 1992.

[12] M. Kozuch and M. Satyanarayanan. Internet suspend/resume. In *WMCSA '02: Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, page 40, Washington, DC, USA, 2002. IEEE Computer Society.

[13] E. Levy and A. Silberschatz. Distributed file systems: concepts and examples. *ACM Comput. Surv.*, 22(4):321–374, 1990.

[14] Libnet. Libnet home page. http://libnet.sourceforge.net/libnet.html, 1999.

[15] Libpcap. Tcpdump libpcap public repository. http://www.tcpdump.org/, 2009.

[16] Y. Luo, B. Zhang, X. Wang, Z. Wang, Y. Sun, and H. Chen. Live and incremental whole-system migration of virtual machines using block-bitmap. *Cluster Computing, 2008 IEEE International Conference on*, pages 99 –106, 29 2008-Oct. 1 2008.

[17] D. S. Milojičić, F. Douglis, Y. Paindaveine, R. Wheeler, and S. Zhou. Process migration. *ACM Comput. Surv.*, 32(3):241–299, 2000.

[18] M. Nelson, B.-H. Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 25–25, Berkeley, CA, USA, 2005. USENIX Association.

[19] M. Satyanarayanan, B. Gilbert, M. Toups, N. Tolia, D. O'Hallaron, A. Surie, A. Wolbach, J. Harkes, A. Perrig, D. Farber, M. Kozuch, C. Helfrich, P. Nath, and H. Lagar-Cavilla. Pervasive personal computing in an internet suspend/resume system. *Internet Computing, IEEE*, 11(2):16–25, March-April 2007.

ISBN: 978-960-474-277-6

[20] SourceForge. Open source tripwire. http://sourceforge.net/projects/tripwire/, 2010.

[21] E. G. Steinbach, Y. J. Liang, and B. Girod. A simulation study of packet path diversity for tcp file transfer and media transport on the internet. In *in Tyrrhenian International Workshop on Digital Communication (IWDC*, pages 67–70, 2002.

[22] W. R. Stevens. *TCP/IP illustrated (vol. 1): the protocols*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[23] Y. Tamura. Kemari: Virtual machine synchronization for fault tolerance using domt. *Xen Summit*, June 2008.

ISBN: 978-960-474-277-6