Live Replication of Virtualized VoIP Servers

Jiri Hlavacek, Robert Bestak
Department of Telecommunications
Faculty of Electrical Engineering, CVUT Prague
Prague, Czech Republic
e-mail: hlavaji1@fel.cvut.cz, robert.bestak@fel.cvut.cz

Abstract— **Virtualization technology enables resource sharing and it can also improve system availability thanks to the continuous real-time migration mechanism. Highly available Internet Protocol communication systems are still expensive as there is no standard solution and customized software development is lengthy and expensive. In this paper, we analyze the continuous live replication mechanism of Xen hypervisor in the context of virtual machine hosting a Voice over Internet Protocol server processing both signalization and user data. We demonstrate that, nowadays, replication mechanism is unsuitable for soft real-time applications. The main drawbacks are jitter and packet bursts generated by the replication mechanism output buffering. To eliminate this, we propose to classify output packets and to let packets with real-time data bypass the buffering. The results obtained confirm that the soft real-time application's availability can be significantly improved by using virtual machine's live replication.**

Keywords-VoIP; system availability; virtualization; live replication.

I. INTRODUCTION

Voice over Internet Protocol (VoIP) technology [1] is becoming omnipresent. Its rich set of services, low cost for operators and ease of integration with other technologies are the main reasons for its success. However, one of the drawbacks is lower availability compared to legacy systems. IP networks are designed to be fault tolerant for general use cases, for example client-server applications. Telephony is a soft real-time application requiring availability equal to 99.999%, i.e., 5.26 minutes downtime per year. In order to achieve such requirements, specific network configuration and application design are needed. Replication enabled applications are complex and, therefore, expensive; thus, other solutions are under research.

Virtualization is widely used for its advantages such as better resource usage, flexibility and scalability. Moreover, it can also be employed to improve system availability using live migration techniques. The main advantage of virtualization consists in the fact that the migration is completely transparent for applications. Live migration mechanisms are still in development. The main challenges include scheduling of virtual machines [2] and network bandwidth optimization [3]. Other issues are latency, jitter and packet bursts introduced by the replication process itself.

In this paper, we propose a modification of the network buffer used by the replication mechanism. This buffer enables saving of all outbound data between two successive replication steps to ensure data consistency in case of failure. Our proposition consists in network packet inspection and classification, where only Transmission Control Protocol (TCP) packets [4] are buffered. User Datagram Protocol (UDP) and Real-time Transport Protocol (RTP) packets [5][6] are forwarded, bypassing the live replication buffering. This way, the latency and jitter for RTP packets are minimized and the voice quality of calls is improved. Our modification minimizes virtualization's impacts on real-time data flows.

The rest of this paper is organized as follows. Section II provides an overview of related work. Section III presents the main virtualization principles followed by an analysis of virtualization's impact on voice quality. Section IV describes our proposition and outlines a potential implementation. In Section V, we present the testbed used and the results obtained. Finally, Section VI concludes our work.

II. RELATED WORK

The problem of VoIP system's availability is addressed in many proprietary or standardized approaches but none of them really fits all real scenarios. In this work, we focus on the VoIP server availability in the standard server/client architecture; peer-to-peer networks are not taken into account in our study.

A common point of the solutions described below is the use of the Internet Protocol (IP) failover mechanism [7]; the active server uses a virtual IP address that is taken by the backup server when the active one fails. Thanks to that, the process is entirely transparent to the clients.

A. Application Dependent Solutions

Application level replication is described by A. Gorti [8]. The proposed solution requires a specific software development, which is expensive and long. Replication enabled applications based on software frameworks for high availability (e.g., Terracota) are hard to configure and maintain. Furthermore, there are several requirements on the software architecture such as thread safety [9]. G. Kambourakis et al. propose a database-based state sharing mechanism [10]. Contexts are saved in a database and the replication is done by the database engine. This solution is relatively easy to implement as only a database connector

 ISBN: 978-1-61208-283-7

needs to be developed instead of a complex replicated system preserving data consistency. Nevertheless, the architecture remains complex and hard to integrate with existing applications.

A common problem to all application dependent solutions is the breakdown of TCP and Transport Layer Security (TLS) connections when taking over the IP address. These connections can't be migrated without specialized operating systems or replication aware clients.

B. Application-Transparent Solutions

Application transparent solutions usually require more resources, but it is compensated by easier development and maintenance of applications. One of the recommended methods to ensure service continuity in case of server failure is the use of the Domain Name System Service Record (DNS SRV) mechanism [11][12]. In this method, a single point of failure is the DNS server itself. A secondary DNS server is usually present but the timeout and takeover adds some seconds to the call establishment. Furthermore, the context replication issue is not addressed and thus this solution is suitable only for stateless servers.

Virtualization techniques are nowadays very popular, mainly because of their better hardware resources usage. These techniques are also employed to facilitate hardware maintenance and fault tolerance. A special use case is continuous live migration of running Virtual Machine (VM) used for high available systems. The principle of the continuous live migration is to replicate the primary machine state to the backup machine continuously and to start the backup machine if the primary one fails. However, this technique is resource demanding. It also introduces a periodic short interruption necessary for synchronization to the primary machine execution. Two ways of machine's replication exist. The first way is based on replay of non-deterministic events received by the primary machine on the backup one [13]. This technique is used for example by VMware [14]. The deterministic replay is not adapted to the Symmetric Multi Processing (SMP) environments since ordered memory access is needed [15]. The second machine's replication approach is based on periodic replication of checkpoints, i.e., processor and memory states, with a high frequency [16]. This replication method is appropriate for SMP environments, but requires more bandwidth for the replication process.

Virtualization also reduces virtual machine performances. A detailed analysis of XEN hypervisor's scheduler identifying bottlenecks for media applications is presented by M. Lee et al. [17]. Modifications aiming at a better adaptation of the XEN scheduler for media applications are presented by M. Lee et al. [2]. These two works show that virtualization supports performance requirements of VoIP servers. Adaptation of virtualized environment for high available VoIP servers is proposed by D. Patnaik et al. [18], where authors show that real-time replication without network buffering performs well enough to run a VoIP server. Nevertheless, data consistency isn't preserved without network buffering; primary and backup machine states can become desynchronized. The main virtualization's advantage is its transparency at the application level, i.e., any VoIP server implementation can be used.

In order to support complex VoIP services, servers are composed of many different modules. Therefore, the application level replication approach becomes too complex and security requirements are hard to meet [9]. On the other hand, the migration of virtualized machines is transparent at the application level and preserves TCP/TLS connections.

III. PROBLEM ANALYSIS

Virtualization [19] is a new challenging mechanism and its use for real-time and multimedia applications is being actively investigated by researchers. In our study, we focused on network metrics such as jitter and burst generation, as these aspects are strengthened by the replication mechanism itself.

A. Virtualization Basics

Virtualization allows one or more virtual machines to run simultaneously on one hardware platform. Hardware sharing is ensured by a specific layer between the hardware and virtualized machines called the hypervisor. Each machine disposes of several resources that are attributed by the hypervisor and the machine can be unaware of being virtualized. Hypervisors often allow migration of virtual machines between different hardware servers in a shutdown state. Current hypervisor implementations improved the migration mechanisms and enable a real-time migration without virtual machine shutdown. This approach is called live migration. The live migration is useful to change the server hosting the VM to obtain more resources or to enable hardware maintenance. Another usage is to build high availability systems. The live migration process can be described as follows. The virtual machine execution is paused by the hypervisor; the virtual machine's state (including memory content and processor state) is transferred to the backup server where the virtual machine is resumed.

B. Live Migration

To overcome hardware faults, live migration has been improved to support continuous live migration. This type of migration allows immediate resuming of a backup virtual machine when the primary server goes down or becomes unreachable. The execution of the primary virtual machine is divided into small periods of time and the state of the primary machine is entirely replicated on the secondary server at the end of each period. These periods are called epochs. The backup machine on the secondary server waits in a paused state. The replication period depends on the machine purpose and typically ranges from 40ms to 200ms [16]. In case of failure, the backup machine on the secondary server is resumed at the last completed checkpoint. This means that a packet loss will be observed from the user's perspective.

C. Data Consistency

The challenge of live migration mechanism is to ensure data consistency in case of primary server failure. To

 ISBN: 978-1-61208-283-7

minimize the interruption and to preserve performances, the replication is being done continuously. A synchronization checkpoint ensuring the consistency is introduced at the end of each replication cycle. Synchronization checkpoint consists of stopping the execution, synchronization and consistency verification and resuming the execution [16].

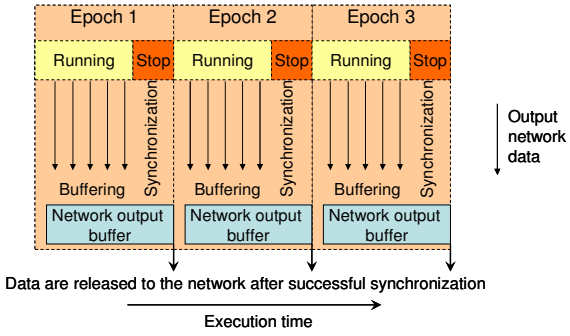

Figure 1. Network buffering during continuous live migration.

To integrate such a system into the real environment, network outputs need to be saved in a buffer between each checkpoint and released once the state of virtual machine is successfully replicated on the backup server. This buffering process introduces jitter and packets burst. This mechanism is depicted in Fig. 1. Although these events can be tolerated in interactive client/server applications, it is a significant issue for real-time applications such as VoIP services. In certain cases, the buffering impact can be accepted for signalization, but needs to be definitively eliminated for user data like RTP flows.

IV. PROPOSED SOLUTION

This section details the proposed solution to the problem described above, including a possible way of its implementation.

A. Principle

Current replication implementation buffers all outgoing packets (signaling, data). The buffering of real-time transport protocol packets has a negative impact on real-time applications. Thus, the replication mechanism needs to be enhanced by inspecting packets and classifying them in two categories: i) packets that need to be buffered and ii) packets that can be sent without being buffered. As we focus on the VoIP service, we simplify this classification as follows. All TCP packets are buffered in order to ensure TCP state consistency and to avoid signaling packet loss when using TCP or TLS protocol. Non-TCP packets are sent to the network without any processing by hypervisor. This approach minimizes a negative impact on network forwarding such as latency increase, jitter introduction or burst generation. The classification needs to be refined to match only packets with real-time data.

B. Implementation

In our testbed, we use the latest XEN hypervisor version 4.2-unstable [20]. A server with XEN hypervisor includes the following components:

- XEN hypervisor.
- Domain 0 – the XEN's terminology for privileged domain with direct hardware access run by the hypervisor to manage the server and control other virtual machines.
- Domain U – the XEN's terminology for unprivileged domains hosting virtualized machines.

The XEN hypervisor live migration implementation is provided by Remus project [16]. Remus provides fault detection, virtual machine state (memory and processor state) replication and network buffering. The network buffer implementation is based on Linux traffic control together with Intermediate Functional Block [21]. This component allows ingress traffic queuing. The traffic sent by virtual machines is buffered at the Domain 0. Standard Linux traffic control tools are used to redirect all incoming traffic from the replicated machine to the buffer. We analyze all passing packets and redirect to the buffer only TCP packets. The traffic classification can be done directly by the Linux kernel. Thus, the kernel code doesn't need to be modified. Our implementation proposal consists of the following 3 steps:

- Packet classification using Linux packet inspection.
- Introduction of redirection rules to buffer only non-real time packets.
- Real-time packets are transmitted without any buffering.

V. OBTAINED RESULTS

This section presents the results of the measurements performed.

A. Testbed Description

The testbed platform with low performance machines and 100 Mbit/s network switch is considered. One server includes a double core Intel Pentium-M processor running at 1.8 GHz (3 GB RAM), whereas the second server is built on a single core AMD Duron processor running at 900 MHz (1 GB RAM). Both servers are running XEN hypervisor, version 4.2-unstable and Linux Ubuntu 12.04 [22] with 3.2.0-24-pae kernel in Domain 0 and the same kernel in Domain U. Fig. 2 illustrates the testbed network configuration. Additionally, Freeswitch [23] version 1.0.7 as a SIP (Session Initiation Protocol) server is used together with one SIP VoIP phone and one SIP softphone. The voice signal processing is done using G.711 A-law codec with 20 ms packetization time (ptime).

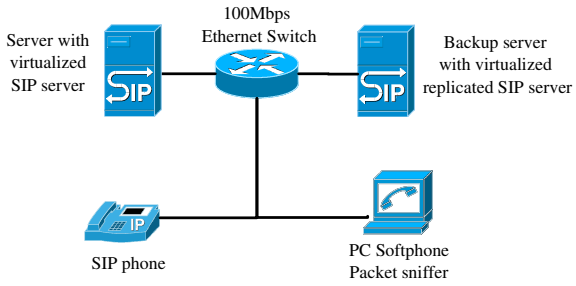

Figure 2. Testbed configuration.

B. *Jitter calculation*

As there is no packet loss except during the effective migration, we consider jitter values to evaluate the impact of virtualization. Jitter calculation is implemented as described in [24]. It is a first-order estimator with noise reduction using a gain parameter.

C. *Virtualization Impact*

Figs. 2 to 6 show the jitter observed under different conditions. The horizontal axes represent time and vertical axes jitter. As shown in Fig. 3, when running a SIP server without virtualization, the jitter is about 100µs with a minimal variation. Performances of Freeswitch running on a virtualized machine (Fig. 4) are not as good as without the virtualization, but still lower than 500µs, which is usually the limit in the Service Level Agreement among operators.

D. *Live Replication Impact*

Fig. 5 illustrates the impact of continuous live migration process on the machine performances. Without any modifications, network buffering introduces an important jitter and packet bursts at each checkpoint. A peak can be observed approximately every 400 ms, which equals to the checkpoint interval.

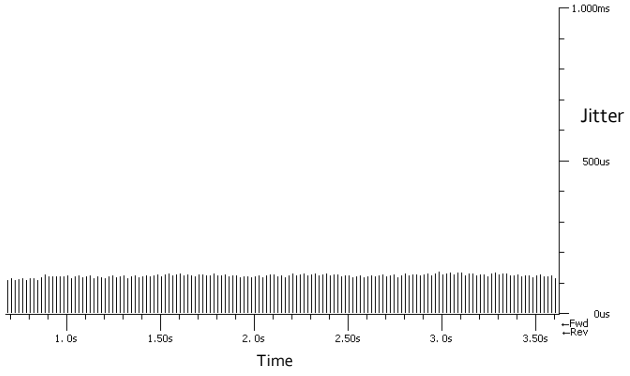

Figure 3. Jitter without virtualization.

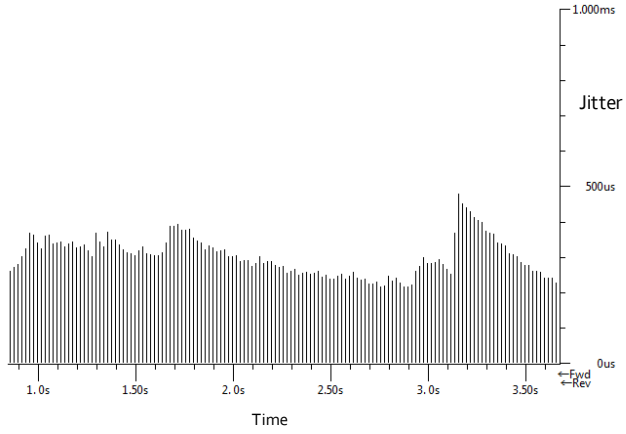

Figure 4. Jitter measured calling via a virtualized Freeswitch server.

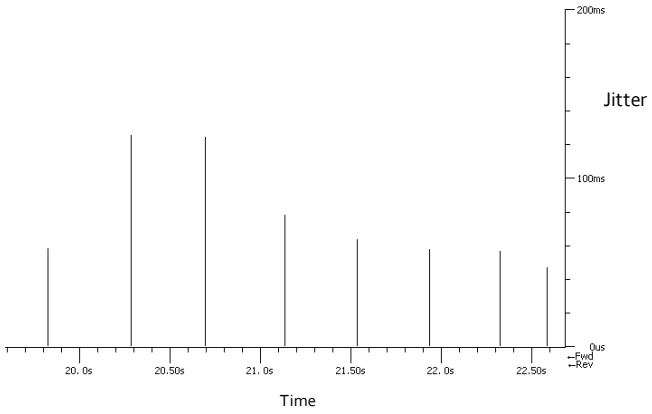

Figure 5. Jitter using unmodified replication mechanism.

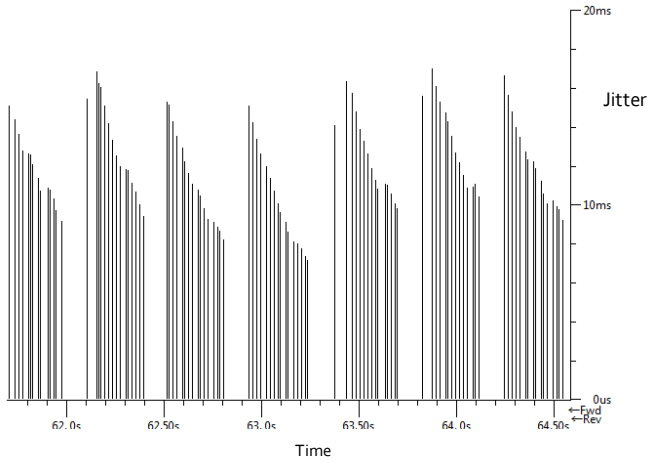

Figure 6. Jitter using modified replication mechanism.

These peaks correspond to a burst of data released from the buffer. With a 20 ms packetization time, each peak represents abound 20 RTP packets by call and direction. The observed degradation of network characteristics deteriorates quality of VoIP calls. Such packet bursts can overload network equipments and cause packet loss. The impact of the continuous live replication is studied in more detail in [25].

Our proposal's impact is shown in Fig. 6. Contrary to the unmodified replication mechanism, RTP packets are forwarded as long as the virtual machine runs. The observed jitter is more important with continuous live replication because of performance impact generated by the continuous VM state replication. This effect is emphasized by the low-performance CPU. Execution interruptions can be observed every 300ms, which corresponds to the configured checkpointing interval.

The considered testbed is composed of relatively low performance machines and a single 100 Mbit/s network, while a 1 Gbit/s network dedicated to replication is recommended. Used low performance testbed allows us to verify the behavior of the proposed modification with limited resources.

 ISBN: 978-1-61208-283-7

Measured jitter and interruption length are therefore quite high, but fully rectified by jitter buffer and the impact on the call quality is unnoticeable from the user's point of view. Note that the calculation method defined in [24] considers the previous packet jitter; therefore we can observe a descending trend following each interruption.

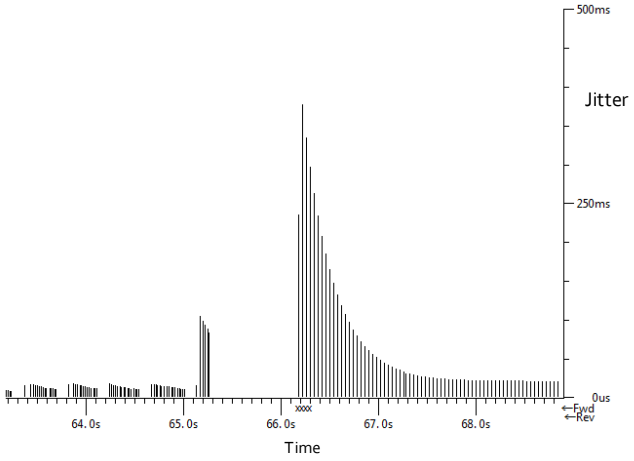

Figure 7. Jitter and packet loss during the migration following a failure.

The objective of the continuous live migration is to maintain established calls and connections in case of hardware or network failure. Once a fault is detected, the replicated machine resumes on the backup server without call interruption. To announce the new location of the virtual IP address, a gratuitous Address Resolution Protocol (ARP) request is used. The jitter observed during the migration is depicted in Fig. 7. On the left side we observe the system in a stable state, continuous VM replication running. The failure of the primary server is represented by a period without packets, which lasts about 1s. The detection of the primary server's failure, generation of the gratuitous ARP and its processing by network components is time consuming. In our configuration it's almost 1s, but the time strongly depends on network hardware. Users of the VoIP service perceive the failover as a short interruption. As the second virtual machine runs without continuous live replication, the jitter observed after the migration is stable without interruption. The descending trend is due to the used calculation method discussed above.

VI. Conclusion and Future Work

Recent VoIP systems are complex and often interconnected with external services. The possibility of implementing a high availability system with no impact on the application is therefore a very challenging task. In this article, we study the jitter as a major drawback of virtual machine's live migration. We propose a modification to improve networking properties of this mechanism without impact on data consistency. The proposition is easy to implement and its impact on the system performance is negligible compared to the unmodified system. Our measurements demonstrate that the modification is beneficial for VoIP and other soft real-time applications. Contrary to the conventional implementation, our implementation does not introduce any jitter to the real-time packet flow except the jitter caused by the interruption required to synchronize replicated virtual machines.

The remaining jitter is caused by the interruption necessary for checkpointing. In our future work, we will focus on reducing the necessary interruption time. The optimal length of checkpointing interval is another point that is worth to be investigated as well as better packet classification.

Acknowledgment

This research work was supported by the Grant Agency of the Czech Technical University in Prague, grant no. SGS13/199/OHK3/3T/13.

References

[1] P. Mehta and S. Udani, "Voice over IP", IEEE Potentials, vol. 20, issue 4, November 2001, pp. 36-40.

[2] M. Lee, A. S. Krishnakumar, P. Krishnan, N. Singh, and S. Yajnik, "Supporting soft real-time tasks in the Xen hypervisor," VEE 2010, Pittsburgh, PA, March, 2010, pp. 97-108.

[3] Ch. Clark et al., "Live migration of virtual machines," In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, vol. 2, USENIX Association, Berkeley, CA, USA, 2005, pp. 273-286.

[4] J. Postel, "Transmission Control Protocol", RFC 793, Internet Engineering Task Force, September 1981.

[5] J. Postel, "User Datagram Protocol", RFC 768, Internet Engineering Task Force, August 1980.

[6] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, Internet Engineering Task Force, July 2003.

[7] K. Singh and H. Schulzrinne, "Failover, load sharing and server architecture in SIP telephony," Comput. Commun. 30, 5, March 2007, pp. 927-942.

[8] A. Gorti, "A fault tolerant VoIP implementation based on open standards," In Proceedings of the IEEE Dependable Computing Conference, Dependable Computing Conference, 2006, pp. 35–38.

[9] J. Hlavacek and R. Bestak, "Improvements in the availability of SIP networks," In Proceedings of the 2010 Networking and Electronic Commerce Research Conference, Dallas, TX: American Telecommunications Systems Management Association Inc., 2010, pp. 109-117.

[10] G. Kambourakis et al., "High availability for SIP: Solutions and real-time measurement performance evaluation," International Journal of Disaster Recovery and Business Continuity, vol. 1, no. 1, 2010, pp. 11-30.

[11] J. Rosenberg and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP servers," RFC3263, Internet Engineering Task Force, June 2002.

[12] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," RFC2782, Internet Engineering Task Force, February 2000.

[13] D. J. Scales, M. Nelson, and G. Venkitachalam, "The design of a practical system for fault-tolerant virtual machines," SIGOPS Operating Systems Review, vol. 44, issue 4, December 2010, pp. 30-39.

[14] VMware, VMware Incorporation, http://www.vmware.com, [retrieved: March, 2013].

 ISBN: 978-1-61208-283-7

[15] G. Kambourakis et al., “Hardware and Software Approaches for Deterministic Multi-Processor Replay of Concurrent Programs,” In Intel Technology Journal, vol. 13, issue 4, 2009, pp. 20–41.

[16] B. Cully et al., “Remus: High availability via asynchronous virtual machine replication,” In Proc. Symp. Network Systems Design and Implementation (NSDI), 2008, pp. 161–174.

[17] M. Lee, A. S. Krishnakumar, P. Krishnan, N. Singh, and S. Yajnik, “XenTune: Detecting Xen Scheduling Bottlenecks for Media Applications,” IEEE Globecom 2010 (Communications Software, Services and Multimedia Applications Symposium), Miami, FL, Dec 6-10, 2010, pp. 1-6.

[18] D. Patnaik, A. Bijlani, and V. K. Singh, “Towards high-availability for IP telephony using virtual machines,” IEEE 4th International Conference on Internet Multimedia Services Architecture and Application (IMSAA), December 2010, pp. 1-6.

[19] S. Crosby and D. Brown, “The Virtualization Reality”, Queue, vol.4, issue 10, December 2006, pp. 34-41.

[20] XEN, Xen hypervisor, http://xen.org, [retrieved: March, 2013].

[21] IFB, The Intermediate Functional Block device, http://www.linuxfoundation.org/collaborate/workgroups/networking/ifb, [retrieved: March, 2013].

[22] Ubuntu, Open Source Linux operating system, http://www.ubuntu.com, [retrieved: April, 2013].

[23] Freeswitch, Open Source Multi-Protocol Soft Switch, http://www.freeswitch.org, [retrieved: April, 2013].

[24] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 3550, Internet Engineering Task Force, July 2003.

[25] J. Hlavacek and R. Bestak, “Configuration of Live Migration for VoIP Applications,” In Proceedings of 15th Mechatronika 2012, Praha, Czech Technical University in Prague, 2012, pp. 187-190.

 ISBN: 978-1-61208-283-7