Virtualization in Energy-Efficient Future Home Environments

Andreas Berl and Hermann de Meer
Computer Networks and Communications
University of Passau, Germany
Email: {berl,demeer}@uni-passau.de

Helmut Hlavacs and Thomas Treutner
Distributed and Multimedia Systems
University of Vienna, Austria
Email: {helmut.hlavacs, thomas.treutner}@univie.ac.at

***Abstract*—Home environments promise high potential in terms of resource sharing and energy saving. More and more home computers are running on an always-on basis (e.g. media-centers or file-sharing clients). Such home environments have not been sufficiently analyzed regarding the possibility of aggregating home user resources in an energy efficient way. This article describes a future home environment in which available hardware resources (e.g. CPU cycles, disk space, or network capacity) are shared energy efficiently and balanced among end-users. Furthermore, the article provides an overview of different virtualization methods that are needed in future home environments to enable cooperation of home networks. Virtualization-related requirements are discussed in detail and virtualization methods and concepts are compared to each other with respect to their usability in the architecture.**

I. Introduction

More and more end-users have home networks that consist of several computers (e.g. personal digital assistants, laptops, desktops, or home servers). Such home networks provide a huge pool of hardware resources that has not sufficiently been analyzed yet. The idea of Future Home Environments (FHE) is to enable sharing home network resources with users of other homes, similar to Grid computing approaches. Unlike the centralized management of Grids, FHE is managed in a decentralized way (e.g. peer-to-peer inspired), without the need of a central management instance.

The advantages for end-users are manifold: Users get access to more resources than they have available locally. In addition, users gain increased availability of computational resources. Their tasks can be processed, even when some of their own equipment is down. Another main advantage of the envisioned kind of resource sharing is that it can be done in an energy efficient way. Increased costs of energy and the desire to reduce green-house gases make energy-efficient computing an increasingly important topic. End-devices in home networks are contributing to a large portion of the electricity consumption growth according to a survey [1], commissioned by the EU in 2006.

Section II presents a FHE architecture that enables balanced resource and energy sharing among home networks. The FHE architecture follows resource-sharing and energy-saving concepts of data centers. Energy wastage in data centers is mainly caused by underutilized hardware. To increase energy-efficiency, services are virtualized and consolidated (several services run on the same hardware). The FHE architecture realizes this kind of consolidation in home networks. Hardware resources of end-hosts (e.g. CPU cycles, disk space, or network capacity) are virtualized and shared among users. *Always-on* services of users (e.g. media-servers or file-sharing applications) are consolidated on end-hosts and unused computers are turned off (or hibernated) to save energy. Although this concept seems to be very similar to energy-saving concepts of data centers, there are severe differences. Services in data centers are located in controlled closed environments with high bandwidth networks. Distributed home environments have no central management and require different virtualization and management methods to share hardware resources and energy in a balanced way.

To enable resource sharing in home networks, the following virtualization-related requirements have to be fulfilled:

- *Resource availability*: Provision of idle hardware resources in separate runtime environments;
- *Home network interconnection*: Addressing and locating of participating home networks;
- *Resource mediation*: Addressing and locating of idle hardware resources;
- *Resource allocation*: Distributed management of state and resource information;

Sections III and IV discuss these virtualization-related requirements of the architecture in detail and give an overview on *host virtualization* (virtualization of hardware resources of a host) and *network virtualization* approaches that can be used to meet these requirements. Section V describes a prototype implementation of the architecture and discusses scalability issues. Section VI concludes the article and identifies future work.

II. A Future Home Environment Architecture

This section introduces a FHE architecture [2], [3], [4], [9] that enables the energy-efficient sharing of hardware resources amongst home networks. The two main goals of the architecture are 1) to enable hardware resource sharing among home users and 2) to achieve energy efficiency by the consolidation of load (e.g. in terms of bandwidth consumption, CPU usage or disk space).

To enable hardware resource sharing, resources on home computers are virtualized, using virtualization methods (as

discussed in Sections III and IV. To achieve energy efficiency (similar to approaches in data centers) in the FHE architecture, the overall load is shifted to a small number of computers in order to relieve others. Unloaded computers can be hibernated (or turned off) to save energy. The consolidation of load is envisioned as follows: A user starts a task (e.g. a file-sharing client) locally on his computer. The FHE environment discovers the potential of energy savings. It moves the task to another computer (probably to another home network) that is already running and has enough hardware resources left to process the task. The local computer can be turned off to save energy. When the task is finished, the local computer is restarted and the result is sent back. In this way only a small portion of always-on computers is needed.

A home network usually consists of a number of computers that are connected via a *gateway* (standard home router) to the Internet. The interconnected home networks form a distributed pool of virtual resources and the FHE uses a distributed management to allocate resources to home networks dynamically. To simplify energy savings, a distinction is made between *active* and *passive* home networks (contributing and non-contributing home networks). A home network is called active if it contains at least one computer which is turned on and can share resources. In a passive home network only the gateway is online and other hardware is hibernated (or turned off).

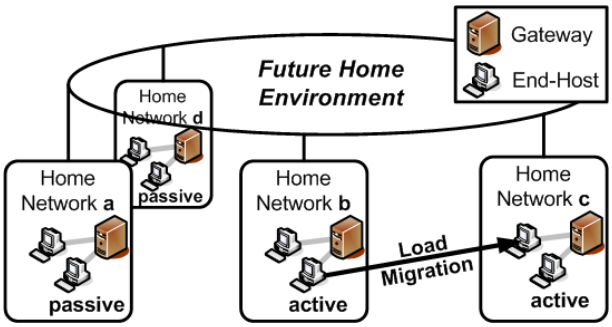

Fig. 1. FHE architecture

The FHE architecture is illustrated in Figure 1. In this example four home networks are interconnected by the FHE overlay, two active and two passive homes. In the figure load is migrated from an end-host in the active home network *b* to an end-host in the active home network *c*. The end-host in home network *b* can be hibernated or turned off after the migration process. If no further computer is turned on in home home network *b*, it can change its status to passive.

III. Host Virtualization Methods in FHE

To enable hardware resource sharing (e.g. CPU cycles, memory, or disk space) among end-hosts in home networks it is necessary to make idle resources available for processes (guest applications) of other end-hosts (*resource availability*, see Section I). A runtime environment has to be established, where guest applications of other users can be processed. This runtime environment has to be flexible enough to enable the processing of a wide variety of guest applications. The guest application might come from a different *Operating System* (OS) (e.g. Windows, MAC, or Linux) or from a different computer architecture (e.g. x86 or PowerPC). This flexible runtime environment also has to deal with privacy and security issues. On the one hand, guest applications are sent to unknown hosts within the FHE architecture. The guest applications might come together with private data (e.g. a movie that needs to be encoded). The owner of the guest application wants it to be separated as much as possible from the user environment of the host. On the other hand, a user that hosts a guest application wants his machine to be separated as far as possible from the guest application. A guest application might be buggy or even malicious and try to attack the host it runs on. The security mechanism to execute untrusted programs from unverified third-parties is often called *sandbox*. A sandbox typically couples a tightly-controlled set of hardware resources within a runtime environment to execute guest applications. The ability of guest applications to inspect the host system or to read from input devices are usually disallowed or restricted. The protection of guests against malicious hosts is more difficult and is not discussed in this article.

The virtualization of a host's hardware resources provides solutions for such requirements of the FHE architecture. *Process virtualization* is a host virtualization method that has been in use in OS for a long time to enable multiprogramming. It allows several different processes to run in parallel while being isolated from each other. Each process experiences full access to all available hardware resources (e.g. CPU, memory, hard disk, etc.) and is not aware of the fact that it is sharing these hardware resources with other processes. In fact, it only owns some time slices of the CPU, a part of the memory (virtual memory), and also shares the peripheral devices with all other processes. The OS is allocating the hardware resources to the processes as needed, following a resource allocation algorithm. Another well known example of process virtualization is the Java Virtual Machine, where processes are executed in sandboxes independent from each other.

OS-Level virtualization has been proposed by the Linux-VServer [5], for instance, and is a Linux kernel based virtualization method. A *jail* provides a set of resource limits that is imposed on programs by the OS kernel. It can include I/O bandwidth caps, disk quotas, network access restrictions and a restricted filesystem namespace.

System virtualization takes the concept of resource virtualization one step further. A *Virtual Machine* (VM) is created in system virtualization, i.e., the complete hardware of a computer is virtualized (consisting of virtual CPUs, virtual memory, virtual hard disk, virtual Network Interface Card, etc). A VM is a perfect recreation of a real machine in such a way that an OS can be installed on it without being aware of the resource virtualization. To distinguish virtualized hardware resources from physically available hardware resources the term *Real Machine* (RM) is used to refer to physical hardware. The software that provides VMs is usually called *Virtual*

Machine Monitor (VMM) and is either located directly on top of the RM (called full virtualization) or on top of an OS (called hosted virtualization). The VMM in the full virtualization approach is also called hypervisor (classical hypervisors are e.g. XEN [6] or VMWare ESX Server[1]). A VMM can host several VMs that are operating in parallel on a single RM. The VMs are independent from each other and are not necessarily aware of the existence of other VMs, running on the same RM. The number of provided VMs is limited by the available hardware resources on the RM. It is important to see that the virtualization layer causes overhead, thus not all of the available resources can be provided to VMs. An OS can be installed within a VM and is called *Guest OS*.

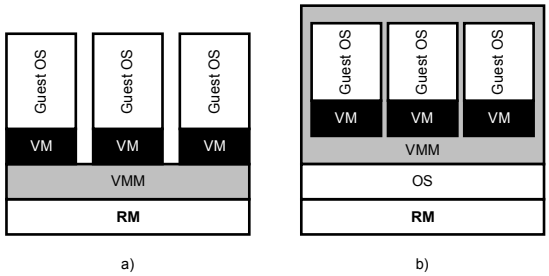

Fig. 2. System virtualization: a) full b) hosted

Figure 2 a) illustrates the method of full system virtualization. On top of a RM, a VMM virtualizes the RM's hardware resources and hosts three VMs in this example. In each VM a Guest OS is installed. Full system virtualization is mainly used to virtualize services in data centers today. There are several basic primitives of management functions for VMs available: create VM, destroy VM, start VM, stop VM, move VM, copy VM and pause VM. It is even possible to have a *live migration*. This means that a virtualized server can be moved to another RM without interruption of the service.

QEMU [7] is an emulator, that does not only virtualize a full system (QEMU is a *hosted* VMM, see Figure 2 b)), but additionally emulates a CPU based on dynamic binary translation. This kind of emulation makes QEMU a very flexible VMM: the virtual machines can be run on different architectures. For Linux, lately, a modified QEMU being part of the Kernel-based Virtual Machine (KVM) kernel module[2] has become increasingly popular, due to its ease of use, and its good performance.

All of these host virtualization methods can be used to virtualize hardware resources in the FHE architecture. All of them provide the envisioned separated runtime environment for guest applications. However, among these concepts system virtualization seems to be the most flexible choice. Guest applications can be enclosed within a Guest OS which is freely configurable by the owner of the guest application. In contrast to process virtualization approaches, guest applications can be standard programs (in arbitrary languages) that can run within the Guest OS. Among the system-virtualization approaches, QEMU (and perhaps KVM under Linux) seems to be the most appropriate one for the FHE architecture. QEMU is a very flexible sandbox solution (Guest OSs can be migrated between machines of different architectures) and has been used as an initial host virtualization method within FHE [4].

IV. Network Virtualization Methods in FHE

Host virtualization methods alone are not sufficient to solve the energy-efficient resource sharing in home environments. In addition, three further virtualization-related requirements have to be achieved: *Home network interconnection*, *resource mediation*, and *resource allocation* (see Section I).

Home networks that are participating in the FHE architecture have to be interconnected in some way. In the FHE architecture no central FHE provider is intended. Therefore, the addressing of home networks has to be solved in a distributed way, to make participating home networks locatable. Host virtualization makes idle resources on hosts available in separated runtime environments, however they are not yet accessible to other users. A mediation of available hardware resources has to be established. Idle resources have to be discovered within the FHE network and they have to be made addressable to enable the allocation of idle resources to other participants. Another important requirement of the FHE architecture is the distributed management of resource allocation. No central architectural element is available in FHE that manages the balanced cooperation of home networks and the access to available resources. This has to be achieved in a distributed way. Energy-efficient resource sharing implies a number of constraints that have to be met. Examples are the balanced distribution of energy consumption or the provision of a sufficient quality-of-service to users. A cost model to target such constraints in FHE is discussed in [2]. The distributed management has to be aware of the different states of the home networks (active, passive), the hardware resources that are available at a certain point of time, and the special needs of guest applications. To achieve this management, statistics have to be gathered and managed within the FHE.

Such requirements of the FHE architecture can be met by network virtualization methods [8], [9]. Two kinds of virtualized networks are widely used today: *Virtual Local Area Networks* (VLANs) and *Virtual Private Networks* (VPNs). VLANs like IEEE 802.1Q[3] operate mainly on the link layer, subdividing a switched Local Area Network into several distinct groups, either by assigning the different ports of a switch to different VLANs, or by tagging link layer frames with VLAN identifiers and then routing accordingly. VPNs like IPSec[4], on the other hand, establish a network layer tunnel to either connect two networks (site-to-site), one network and a host (site-to-end) or two hosts (end-to-end) with an encrypted and/or authenticated channel over the Internet. However, these kinds of virtualization methods target mainly the sharing of links among users and are not sufficient for the FHE approach.

[1] http://www.vmware.com/products/vi/esx
[2] http://www.linux-kvm.org/
[3] http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf
[4] http://tools.ietf.org/html/rfc4301

Besides the virtualization of links, also the virtualization of routers has been investigated [8]. In [10] system virtualization is applied to routers to create virtualized networks with special features. In [11] performance challenges are identified that have to be tackled when virtual routers are based on the XEN VMM. Other forms of router virtualization are already available in commercial products (e.g. CISCO Logical Routers). However, virtualization of routers does not solve the network virtualization issues of the FHE architecture. Such solutions mainly allow the concurrent usage of network infrastructure. In the FHE, a mediation of available hardware resources and their distributed management is needed.

A further approach towards network virtualization are *peer-to-peer* (P2P) overlays, (e.g. Chord, Pastry, eDonkey, or Skype) [12]. In this approach logical links are defined on top of a physical infrastructure. A single logical hop in the overlay can be mapped to several physical hops in the network. P2P networks are mainly classified according to their architecture and their algorithmic features. In pure P2P overlays (e.g. Chord) all peers are assumed to be equal. In hybrid P2P overlays some peers are distinguished from other peers, i.e. some peers have different capabilities compared to others (e.g. eDonkey). P2P overlays are denoted to be unstructured if the algorithms establish overlay links which do not follow a regular connectivity pattern. In contrast, P2P overlays are said to be structured if a generic but predefined organization scheme (e.g. a ring) of the overlay exists. In contrast to the previously mentioned network virtualization approaches, P2P overlays do not only virtualize links and nodes, but they solve three main FHE issues: *Home network interconnection*, *resource mediation*, and *resource allocation*. P2P networks establish an addressing scheme within the overlay that enables the addressing of peers as well as the addressing of available resources. In P2P file-sharing networks (e.g. eDonkey), for instance, files can be discovered and addressed, whereas in P2P VoIP applications like Skype users are locatable and addressable. This solves the problem of home network interconnection as well as the resource mediation problem. Also a management with respect to resource allocation is provided in some P2P overlays. File-sharing protocols like eDonkey, e.g., establish complex tit-for-tat principles to enable a balanced resource sharing among peers and allow for the distributed download of files from different sources concurrently.

There are several P2P overlays available that can be used within the FHE architecture. Solutions have to be scalable (with a high number of home networks) and they need to be lightweight to operate on the mentioned always-on gateways (see Section II).

An appropriate solution for the FHE is the unstructured and hybrid eDonkey P2P overlay.

eDonkey is a very popular file-sharing P2P network with a high amount of users (and traffic) that has practically proven to be very scalable and is resistant to high churn-rates[5]. It has the capability to solve all of the mentioned network virtualization-related requirements, it interconnects the home networks, mediates contents, and manages the access to resources. In addition, the hybrid eDonkey structure is very similar to the FHE structure (end-hosts and gateways).

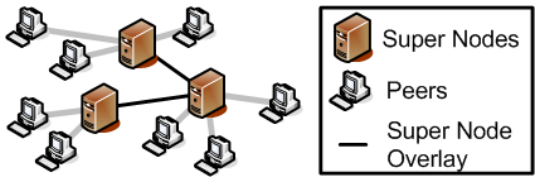

Fig. 3. eDonkey

The eDonkey approach is illustrated in Figure 3. Two kinds of participants can be seen in the network: Peers and super nodes (Index servers). Peers are providing and consuming resources, similar to the end-hosts in the FHE architecture. Super nodes form a separate overlay to share information. A peer can report his available resources to the super node and request the location of hardware resources from the super node. The location of a gateway within the FHE architecture is very similar to the location of a super node — the gateway is physically the first node of each home network. This location makes the gateway the natural place for gathering the statistics about the home network that are needed to enable a balanced and energy-efficient resource sharing among home networks. In addition, the gateway is supposed to be always-on, which enables it to manage and distribute information among other gateways. This makes the gateway an ideal super node of an eDonkey like network.

In [4] Pastry, a structured pure P2P overlay has been used as network virtualization technology for a first test. Pastry has been proven to be scalable and is available as open source platform. Pastry was successfully used in a prototype that is described in Section V. It solves the home network interconnection and the resource mediation problems, as mentioned before. However resource allocation is not addressed in Pastry and has to be developed separately. In general, structured overlays like Pastry are more vulnerable to high churn-rates than unstructured networks [12]. However, end-hosts might show a very dynamic behavior in the FHE architecture, concerning on-line and off-line times. In future work, eDonkey-based network virtualization will be implemented and evaluated for the FHE architecture, as alternative to the Pastry approach.

V. A Pastry- and QEMU- Based FHE Prototype

Task virtualization (the virtualization approach of the prototype) is an application of system virtualization, where a minimal OS is installed within a VM, to enable it to carry out a certain task. Each given task (e.g., the download of a specified file), is encapsulated in a separate VM.

In the prototype, VMs consist of a minimal Linux kernel image, a minimal root filesystem built by OpenEmbedded[6] and

[5]In P2P parlance, the term churn denotes the stochastic process of peer turnover as occurring when peers join or leave the system.

[6]http://wiki.openembedded.net/

a task filesystem, which is put on top of the read-only root filesystem by UnionFS[7]. The VM's task filesystem includes a shell script that is executed after the VM has started and diskspace (to install required packages and store resulting data).

The following tasks have been implemented with the prototype: performance tests, converting MP3 files to OGG Vorbis, a Personal Stream Recorder (PSR) capturing N seconds of a predefined radio webstream and saving it as MP3, stress tests for stability analysis, downloading a specified file using BitTorrent, and downloading a specified file using the command *wget*. Though all related VMs contain a fully bootable Linux machine, sizes of compressed versions vary between only 3.8 and 4.7 MB (excluding the data, e.g., an MP3 file).

vPastry (based on the open source library *FreePastry*) has been used as a first network virtualization approach for the prototype. On the first start the user is prompted to provide some basic information (CPU cores, size of main memory, system energy efficiency, access bandwidth, etc.) about his system. For finding appropriate hosts to carry out a task, a user would select the required performance characteristics and click on the search button. vPastry then uses the publish/subscribe messaging system offered by FreePastry to perform the operation. When a node receives an invitation for outsourcing a task, the user is prompted to select a VM (which contains the task), while a file transfer channel is established in the background. When the transfer is finished, the VM is decompressed into a temporary folder and started by a runtime execution using the *Kernel-based Virtual Machine* (KVM) driver and QEMU. Once a task is finished (which basically means that the respective VM has exited), the caculated data (on the task filesystem) is compressed and sent back to the task owner.

To assess the resource consumption of concurrently executing numerous VMs, the following experiment for a download sharing scenario was done: A PC (AMD Phenom 9550 Quad-Core Processor (2.20 GHz) with 8 GB main memory) was setup to carry out tasks. An instance of vPastry sent $1 \leq k \leq 30$ VMs (each containing a download task) to this host, which executed the VMs. The task was to download a file via wget at 300 KByte/s from an FTP server connected via GigabitEthernet. The total upstream bandwidth of the FTP server was limited to 50 Mbit/s to simulate future VDSL access networks with an according downstream bandwidth. The upstream bandwidth of a single FTP connection was limited to 300 KByte/s to simulate a download that does not fully utilize the available total downstream bandwidth.

The resource demands of such a scenario are illustrated in Figure 4. It shows the utilization of the host's CPU, the main memory and the network card. The utilization is depicted as a function of the number of concurrently executed VMs, where each VM carried out a download task, as specified above. It can be seen that the resource utilization depends linearly on the number of hosted VMs. Memory and CPU utilization grow slowly, due to the little resource demands of the minimalistic Linux OS. This indicates that the bottleneck is the network, whereas CPU and memory have capacities left to host other types of tasks. Furthermore, it is clearly visible that the network gets saturated with 20 VMs ($20 \cdot 300\texttt{KByte/s} \cdot 8 = 48\texttt{Mbps}$) and the slope of CPU and memory utilization decreases. At this point, there is no use in adding more VMs containing download tasks, as the resource utilization rises with no further benefit.

[7]http://www.filesystems.org/project-unionfs.html

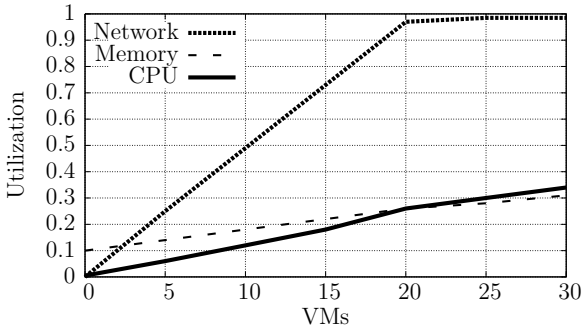

Fig. 4. Resource demands of the download sharing scenario

VI. Conclusions and Future Work

The vision of Future Home Environments is to enable resource sharing for possibly millions of homes, thus enabling to access vast amounts of otherwise unused resources in a reliable and energy-efficient way. In this work we present an architecture for FHE and identify several constraints that have to be met. We further analyze existing techniques for virtualization and peer-to-peer overlays, and identify suitable candidates for constructing a FHE. Finally we present a first prototype that enables task virtualization by encapsulating tasks into minimal VMs, and sending them to other homes connected by a Pastry overlay. In the future we will concentrate on creating new minimal VMs for different tasks, understanding under which conditions energy-efficiency can be achieved, and turn the prototype into a tool usable by casual users.

References

[1] P. Bertoldi and B. Atanasiu, "Electricity consumption and efficiency trends in the enlarged European Union," *IES–JRC. European Union*, 2007.

[2] A. E. García, A. Berl, K. A. Hummel, R. Weidlich, A. Houyou, K. D. Hackbarth, H. de Meer, and H. Hlavacs, "An Economical Cost Model for fair resource sharing in Virtual Home Environments," in *Proceedings of Next Generation Internet Networks (NGI 2008), Krakow, Poland, 28-30 April, 2008*. IEEE, April 2008, pp. 153–160.

[3] H. Hlavacs, K. A. Hummel, R. Weidlich, A. Houyou, A. Berl, and H. de Meer, "Distributed Energy Efficiency in Future Home Environments," *Annals of Telecommunication: Next Generation Network and Service Management*, vol. 63, no. 9-10, pp. 473–485, October 2008.

[4] H. Hlavacs, R. Weidlich, and T. Treutner, "Energy Saving in Future Home Environments," in *2nd Home Networking Conference at IFIP Wireless Days*, Dubai, United Arab Emirates, 11 2008.

[5] B. des Ligneris, "Virtualization of linux based computers: The linux-vserver project," *High Performance Computing Systems and Applications, Annual International Symposium on*, vol. 0, pp. 340–346, 2005.

[6] 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," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.

[7] F. Bellard, "QEMU, a fast and portable dynamic translator," in *Proceedings of the USENIX Annual Technical Conference, FREENIX Track*, 2005, pp. 41–46.

[8] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, 2007.

[9] A. Berl, H. Hlavacs, R. Weidlich, M. Schrank, and H. de Meer, "Network Virtualization in Future Home Environments," in *LNCS 5841: IFIP, Proceedings of Int. Workshop on Distributed Systems: Operations and Management (DSOM09), Venice, Italy, October 27-28, 2009*. Springer, Berlin (Germany), October 2009, pp. 177–190.

[10] A. Berl, A. Fischer, and H. de Meer, "Using System Virtualization to Create Virtualized Networks," in *Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen (WowKiVS2009), Kassel, Germany, March 2-6, 2009*, ser. Electronic Communications of the EASST, vol. 17. EASST, March 2009.

[11] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley, "Evaluating xen for router virtualization," in *16th Int. Conf. on Comp. Commun. and Networks - ICCCN 2007*, Aug. 2007, pp. 1256–1261.

[12] R. Steinmetz and K. Wehrle, *Peer-to-Peer Systems and Applications (Lecture Notes in Computer Science)*. Secaucus, NJ, USA: Springer-Verlag New York, 2005.