Electronic Communications of the EASST
Volume 17 (2009)

Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009 (WowKiVS 2009)

Using System Virtualization to Create Virtualized Networks

Andreas Berl, Andreas Fischer, and Hermann de Meer

12 pages

Guest Editors: M. Wagner, D. Hogrefe, K. Geihs, K. David
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Using System Virtualization to Create Virtualized Networks

Andreas Berl, Andreas Fischer, and Hermann de Meer

University of Passau
Faculty of Computer Science and Mathematics
94032 Passau, Germany
berl, fischean, demeer @fim.uni-passau.de

Abstract: The method of system virtualization is very popular for the use in data centers and desktop virtualization today. In this work, system virtualization is applied to core network elements (routers and links) in order to create a virtualized network. The selection of this virtualization method crucially determines the emerging network model. The network model consists of virtual networks, virtual routers, and virtual links that form overlays on top of the physical network. The properties, features, and limitations of this network model are analyzed and described in this paper. Additionally, a proof of concept implementation using currently available technology and infrastructure is presented. Finally the dynamic configurability of virtual resources in such a system virtualization based virtualized network is evaluated.

Keywords: System Virtualization, Virtual Network, Overlay, Virtual Router, Virtual Link, XEN

1 Introduction

System virtualization is currently a very popular virtualization method that is used to virtualize desktops and servers. Virtualized resources are easy to manage and are not bound to specific physical hardware. Especially in data centers system virtualization is used to provide multiple services in parallel and independent from each other on the same hardware, mainly to increase the utilization of resources. In this paper the method of system virtualization is investigated as a solution towards the creation of virtualized networks. Its high popularity, its extensive usage in productive environments, and the current rapid development of management solutions for system virtualization are the main reasons to investigate it in the context of network virtualization. When system virtualization is applied to the core network (consisting of routers and links), a determined kind of virtualized network model emerges from it. The features and limitations of this model are analyzed, discussed, and illustrated in this paper. This compilation of the possibilities that such a network model offers can be used as basis for the development of an advanced management solution of virtual networks (e.g. to create resilient networks, networks with mobility management, or service supporting networks). The main elements of a virtualized network infrastructure that is based on system virtualization are multiple virtual networks running in parallel, consisting of virtual routers and virtual links. Such virtual networks form overlay structures that are not directly related to the underlying physical network. A virtual network has all ordinary properties of a physical network, but it also gains additional features inherited from system virtualization.

The remainder of this paper is structured as follows. Section 2 clarifies the method of system virtualization. In Section 3 virtualized networks are investigated and described as they emerge when network resources are virtualized using the method of system virtualization. In Section 4 a proof of concept implementation of virtualized networks is described and the dynamic reconfigurability of such a virtualized network is evaluated. In Section 5 related work is described that deals with the virtualization of network resources. Section 6 concludes the paper and outlines open issues for future work.

2 System Virtualization Background

In order to discuss the method of *system virtualization*, first the method of *process virtualization* has to be explained. Process virtualization has been in use for *Operating Systems (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 resources (e.g. CPU, memory, hard disk, etc.) and is not aware of the fact that it is sharing these 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 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. System virtualization on the other hand takes the concept of resource virtualization one step further. A *Virtual Machine (VM)* is created in system virtualization, i.e. a full machine is virtualized, consisting of virtual CPUs,

virtual memory, virtual hard disk, virtual Network Interface Card (NIC), 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 resources from physically available resources the term *Real Machine (RM)* is used to refer to physical hardware [PG74]. The software that provides VMs is usually called *Virtual Machine Monitor (VMM)* and 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 [BDF+03] or VMWare ESX Server [vmw]). 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 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 - it is called *Guest OS*.

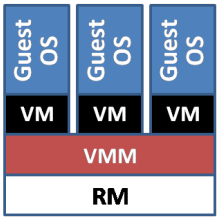

Figure 1: Full system virtualization

Figure 1 illustrates the method of full system virtualization. On top of a RM, a VMM virtualizes the RM's 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 [dmt07]. It is even possible to have a *live migration* [CFH+05]. This means that a service in a VM can be moved to another RM without being interrupted.

3 Virtualized Networks based on System Virtualization

In contrast to other approaches towards the creation of virtualized networks, this approach is based on the method of system virtualization (cf. Section 2). The physical network that is virtualized consists of routers and links between them (core network). Link layer equipment or end systems are not yet considered in this work. When system virtualization is selected as method to virtualize such a kind of network, the created virtualized network is bound to the limitations that are induced by system virtualization. Even more, the decision to use system virtualization crucially determines the model of the emerging network.

*A **virtual router (VR)** in the context of system virtualization is an operating system with routing functionality, encapsulated in a virtual machine that is managed by a virtual machine monitor.*

The element that is considered first in this section is the router. It is the only core network element to which system virtualization can be applied directly. In terms of system virtualization,

the RM is the physical router (hardware). It is virtualized by a VMM to provide one or more VMs on top of it. Within every VM an OS with routing functionality (*Router OS*) is installed to get a VR.

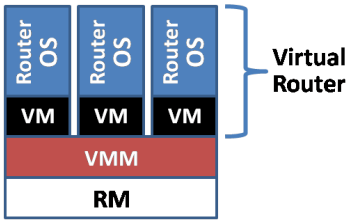

Figure 2: System virtualization based virtual router

This approach is illustrated in Figure 2. It can be seen that a VMM provides in this example three VMs, each with a Router OS installed in it. VRs appear on first glance to be usual routers that have properties similar to physical routers. VRs can have different configurations and features (e.g. different protocol stacks like IPv4 or IPv6 or they can support features like IPSec). However, additionally to common routing properties they inherit properties of the VM in which they are encapsulated. First, there are the basic primitives of VM management applicable to VRs (cf. Section 2). Second, VRs are not necessarily aware of the fact that they are running within VMs. They are also not necessarily aware of the fact that they are sharing physical resources with other VRs which might run in parallel on the same RM. Third, VRs are like VMs reconfigurable in terms of allocated virtual resources (CPU cycles, number of CPUs, memory, disk space, number of NICs, etc.), because the VMM virtualizes CPU, memory and also peripheral devices. One of the main benefits from router virtualization is that several VRs can run in parallel on a single hardware, without being aware of each other. This provides an easy way to test new protocols in networks or to drive different protocols at the same time. The difference to physical routers lies mainly in the provided performance. Hardware routers are usually implementing most of their functions by using support of specialized hardware. However, taking a certain performance penalty into account, all of these functions can also be implemented in software. VRs provide an abstraction of this specialized hardware. They provide standardized virtual hardware, independent on which hardware they are actually running. This abstraction makes it easy to deploy new routing solutions.

*A **virtual link (VL)** in the context of system virtualization is a logical interconnection of two virtual routers, appearing to them as a direct physical link with dynamically changing properties.*

The virtualization of links results directly from the virtualization of routers. The VMM virtualizes peripheral devices, especially NICs for its VMs. Virtual NICs are not necessarily identical to real NICs. VLs appear at first glance to be common links that have properties similar to physical links. They are accessed in an ordinary way and the VRs experience common properties (a certain jitter, bandwidth, throughput, etc.). But also VLs inherit properties from virtualization. VLs appear as a single hop link to a VR, however, a VL can either be based on a physical link that exists between two RMs, or it can be a tunnel, including several physical hops. Theoretically the VMM is able to emulate (with limitations) a different network technology (e.g. token ring) than it is provided by the physical NIC. It can also aggregate several physical links to a single

VL (a technique known as *channel bonding*). The quality of service of a VL might change over time, due to reconfigurations of the virtual resources done by the VMM (e.g. the VMM changes the bandwidth limitations of a VL). VLs can also appear or disappear for a VR, when the VMM adds or removes virtual NICs.

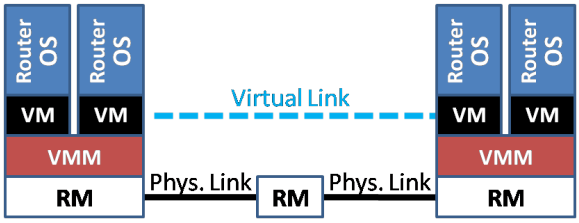

Figure 3: Virtual link and physical link

In Figure 3 a VL and its physical representation are illustrated. It can be observed that the VL interconnects two of the four VRs; the other VRs are not connected with each other in this example. The physical representation of the VL consists of more than one hop. The interconnected VRs are not aware of the physical topology they are actually using. The main benefit of this approach is that the management of VLs is possible without modifying routing configurations of the Router OS. In contrast, VRs can be considered as black boxes that are managed by modifying properties of the VMs they are based on. VLs may be adapted to current events within the physical network or to changes in the virtualized network. The VRs are experiencing a change of reality and are reacting conforming to their configuration without being aware of the underlying physical network. Another advantage is that VLs are not bound to a specific kind of physical link; they might run on top of very different physical links, e.g. fiber, coax, or wireless links. The VRs may experience a different quality of service of the VL, but not necessarily a different VL.

*A **virtual network (VN)** in the context of system virtualization consists of virtual routers and virtual links that form an undirected connected graph. This undirected connected graph is called the **overlay**, in contrast to the underlay that refers to the physical network topology. Several independent virtual networks forming different overlays can exist in parallel on top of a physical network.*

When two VRs are directly or indirectly interconnected via VLs they belong to the same VN. VNs appear at first glance to be usual networks that can be used the same way as if they were not virtualized (e.g. setting up an IPv4 network by installing an appropriate Router OS in VRs). But VNs also acquire properties from being virtualized. A VN forms a virtual topology, the overlay, which is not necessarily identical to the graph structure of the physical network, called underlay. The mapping of VRs in the overlay to RMs in the underlay is single-valued, because a single VR can be placed on only a single RM, limited by the approach of system virtualization (Grid like approaches are not supported by system virtualization, yet). The mapping of VRs in the overlay to RMs in the underlay is not injective, because a single RM is able to host several VRs. The mapping of VLs in the overlay to real links in the underlay is neither injective, nor single-valued, because the VMM is theoretically able to aggregate several physical links to a single VL and also to run several VLs over a single physical link. In practice this means

that several VNs with different overlays can be set up in parallel on the same underlay, e.g. a second IPv6 network can be driven in parallel to the IPv4 network. The two VNs will share physical resources, but not interact otherwise. Another property coming from virtualization is the dynamicity of such VNs. VRs and VLs can be added or they can be removed from the VN. The mapping of overlay to underlay is even more flexible, because VRs and VL are able to change their physical representation without necessarily changing their topological position in the overlay. VRs can move to other RMs and VLs can change their path in the underlay.

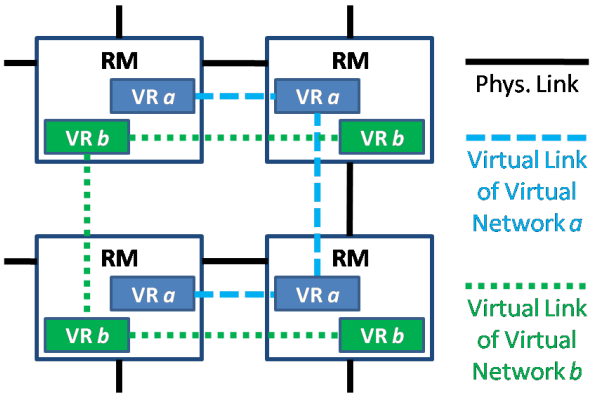

Figure 4: Two independent virtual networks

In Figure 4 two parallel VNs (VN *a* and VN *b*) with different overlays are illustrated. Four RMs can be observed, each hosting two different VRs. It can be seen that the overlay is not identical to the underlay. The VLs of VN *a* are a subset of the physical links, but the VLs of VN *b* are not. There are a number of benefits resulting from virtualizing networks with system virtualization. Virtual resources are not as static as physical resources. Overlays can change in reaction to unforeseen conditions in the physical network by manipulating VRs or VLs. VNs can be created on demand or be destroyed if not needed anymore. Also pausing of VNs is possible by pausing VRs and freeing their resources. Paused VNs are not consuming resources but can be powered on immediately, if needed. Additionally, VNs are providing an easy to use abstraction from the physical network. VRs are easy to deploy on the well known virtual hardware of VMs. There is no need to deal with specialized hardware which is hard to handle for Router OSs. If only aspects of graph theory were considered in the creation of VNs, they could have arbitrary complexity. Multiple highly complex VNs could be created on a single RM. However, there are hard limitations in the creation of VNs due to limited performance of software and hardware. On current hardware it is only feasible to host a few VMs in parallel, depending for example on the number of CPUs and the available memory (cf. Section 5). Additionally, the VMM causes notable overhead when virtualizing the resources of a RM. Such performance issues limit the practical usage of a system virtualization based VN at least with currently available software and hardware.

4 Evaluation

In this section it is evaluated if system virtualization based virtualized networks can be set up on current hardware. Furthermore it is evaluated if VNs are reconfigurable by changing virtual resources for VRs, without reconfiguring the Router OS itself (black box principle).

A simplified model of a system virtualization based virtualized network was implemented, using currently available hardware and software, equipped with additional JAVA-based management software, and evaluated.

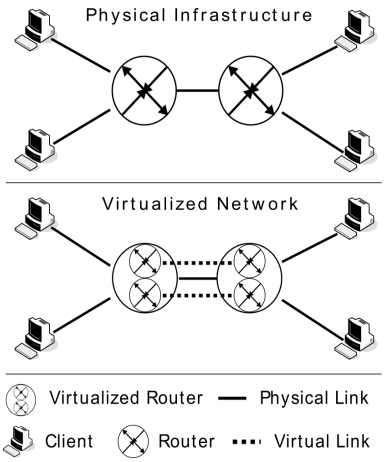

Figure 5: From the physical infrastructure to a virtualized network

The model consists of a minimal IPv4 network with two client sites interconnected via two physical routers over a single network path (cf. Figure 5). Two VNs were set up upon this infrastructure. In these VNs the properties of the VLs were dynamically modified during their operation. The VRs themselves did not have to be modified in order to change properties of the VLs between them. This scenario allows examination of several aspects of VNs. First, it can be shown that it is possible to migrate the physical network entirely into a VN. Second, it is possible to run several VNs in parallel, sporting different attributes (e.g. different link properties for the respective VLs). Third, changes within one network can be achieved dynamic during run-time, without any disruption of service in any VN.

This scenario was realized by outfitting the physical machines with an appropriate hypervisor. XEN [BDF+03] was chosen as system virtualization solution, since it provides a very fast approach called *paravirtualization*. Paravirtualization trades implementation purity for speed, making the Guest OS partly aware of being virtualized and cooperating in the virtualization process. The demonstrated test setup uses two physical machines representing the two routers of the described model. These machines are virtualized with XEN and populated with two VRs each, running LINUX as a Router OS. The VRs are interconnected and form two parallel, independent VNs. The two client sites are represented with two clients on each side of the network (four clients in total), connected to the respective VRs. The VLs are realized in XEN by creating virtual NICs which are connected to the physical NIC via bridge management tools. After startup of the VNs, the clients begin to communicate with each other. To demonstrate the feasibility of

dynamic modification of link properties, one of the VLs in the scenario is modified. This was achieved by applying traffic shaping to the virtual NICs. A bandwidth guarantee is given to the VL, causing traffic between clients using that link to acquire the entire guaranteed bandwidth.

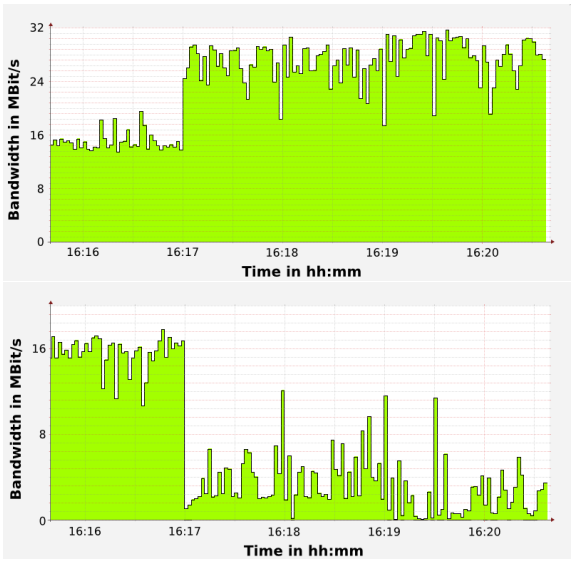

Figure 6: One VL is given a bandwidth guarantee, the other one consumes the spare bandwidth

The results of this experiment can be seen in Figure 6. The physical link offers a bandwidth of 32 MBit/s, which is shared between the two VLs. The value of 32 MBit/s has been chosen to avoid technical difficulties with XEN when handling a high amount of network traffic. Specifically, the combination of traffic shaping and system virtualization produced unexpected bandwidth artefacts when operating above a bandwidth of 32 MBit/s. The source of these problems will have to be investigated in future work.

At first, both VLs share the available bandwidth equally with each VL consuming about 16 MBit/s of bandwidth. After a few minutes, one of the VL is given a bandwidth guarantee of 30 MBit/s. It can be seen, that at that time the consumed bandwidth of the first VL increases to about 30 MBit/s, whereas the second VL at the same time drops to about 2 MBit/s.

The inverse scenario was also examined: limiting a VL in its available bandwidth, causing the clients using that link to experience a loss of bandwidth available to them. Limiting a VL in its available bandwidth is not shown explicitly here – the results were very similar (just inverted) to the ones shown above. In both cases it was possible to modify a VL in its bandwidth without making changes to the involved VRs and without any noticeable consequences for involved VNs, apart from changes in the available bandwidth. This illustrates that the approach of virtualizing networks using system virtualization as described in Section 3 can be realized with current technology. It allows to virtualize an existing network infrastructure, to start at least two VNs in parallel, and to dynamically modify properties of one network without disruption of service in the other VNs (apart from the available bandwidth). While here only a modification of bandwidth was demonstrated, other modifications to the network infrastructure are also possible - e.g. adding or removing Virtual Links, or modifying computing capabilities of a Virtual Router.

5 Related Work

The idea of using virtualization in networks is not new and there are several approaches in this research area [CB08, FGR07]. Two forms of virtualized networks are widely used today: *Virtual Local Area Networks (VLANs)* and *Virtual Private Networks (VPNs)*. VLANs like IEEE 802.1Q [vla03] operate mainly on the link layer, subdividing a switched *Local Area Network (LAN)* 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 [KS05], 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. Such virtualization approaches are focussing on the virtualization of links whereas the approach described in this paper deals with the virtualization of a whole network infrastructure.

Overlays and *Peer-to-Peer (P2P)* networks [SW05] are a widely used approach to get an abstraction of the physical topology of networks. In this approach logical links are defined on top of the physical infrastructure. A single logical hop in the overlay can be mapped to several physical hops in the network. This approach can also be considered as a virtualization of the underlying network. PlanetLab [CCR+03] envisions an open distributed platform for distributed end-to-end applications. Resources of end-hosts located all over the world are virtualized. A user is provided a *slice* that consists of hundreds of shells, one for each end-host. Comparable approaches towards end-to-end virtualization are done by other projects (e.g. GENI [gen] or VINI [vin, BFH+06]). Cloud computing approaches try to offer computing power independent of actual hardware location. They are however not concerned with the virtualization of entire networks. Within the research area of future generation networks virtual home networks are investigated. In [HHW+08, GBH+08] a *Virtual Home Environment (VHE)* is proposed, where end-hosts in home networks are virtualized in order to share resources with other end-hosts while reducing the overall energy consumption. In contrast to the described overlay, P2P, and end-to-end virtualization approaches, the approach in this paper deals with virtualization of the core network infrastructure.

Virtualized networks have also been examined in the context of programmable networks [CMK+99]. These approaches however take a different route, leaving out the abstraction provided by system virtualization.

Virtualization of routers has been investigated in different contexts and is already available in commercial products [cis]. The application of system virtualization to routers has been investigated in [EGH+07, MCZ06]. Performance challenges were identified that have to be tackled when virtual routers are based on the popular XEN hypervisor [BDF+03]. In contrast to these related virtualization investigations, this work analyzes the network model emerging when the method of system virtualization is applied to the core network infrastructure.

Wang et al. [WKB+08] discussed the prospect of having virtual routers that are movable on physical hardware. They propose an architecture supporting dynamic migration of virtual routers to decouple logical and physical configuration of a router. While they focus on a single network management primitive, this work considers the broader impact of system virtualization on the core network.

6 Conclusions and Future Work

System virtualization can be used to create a virtualized network infrastructure. If it is applied to the current core network infrastructure a predetermined network model arises. This network model does not only have properties of a physical network, it additionally inherits properties from the applied virtualization method. The virtualized network consists of virtual routers and virtual links that create overlays on top of the physical infrastructure. Several virtual networks with different properties can exist in parallel on the same physical network, without being aware of each other. In this paper the properties, features, and limitations of such virtualized networks are described and illustrated in order to ease the design of sophisticated network management solutions on top of virtualized networks (e.g. resilient networks, networks with mobility management, or service supporting networks). The network model was prototypically implemented by using currently available virtualization tools and network infrastructure. Two virtual networks were set up in parallel. It has been shown, that such networks are dynamically reconfigurable in terms of link properties and are able to operate in parallel, using the same physical infrastructure. The most interesting feature of such networks is however, that the virtual networks can be managed by modifying virtual resources. The routers themselves can be considered as black boxes and do not necessarily have to be changed. Instead, the routers experience a change of their reality and adapt to it, according to their configuration. In future work the application of further virtualization methods to networks will be analyzed (e.g. the grid-based approach).

Acknowledgements: Parts of the work in this paper were done in the context of the EU-funded project AutoI [aut] and the Network of Excellence EuroNF [eur07]. This paper was also partly funded by the German Research Foundation (Deutsche Forschungsgemeinschaft - DFG), contract number ME 1703/4-2

Bibliography

[aut] Autonomic Internet Project (AUTOI), STREP, FP7, grant no. ICT-2007-1-216404. http://www.ist-autoi.eu. Accessed 17 Dec 2008.

[BDF+03] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.* 37(5):164–177, 2003.

[BFH+06] A. Bavier, N. Feamster, M. Huang, L. Peterson, J. Rexford. In VINI veritas: realistic and controlled network experimentation. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*. Pp. 3–14. ACM, New York, NY, USA, 2006.

[CB08] N. M. K. Chowdhury, R. Boutaba. A Survey of Network Virtualization. Technical report, University of Waterloo, Oct. 2008.

[CCR+03] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman. PlanetLab: an overlay testbed for broad-coverage services. *SIGCOMM Comput. Commun. Rev.* 33(3):3–12, 2003.

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

[cis] Logical Routers Commands on Cisco IOS XR Software. http://www.cisco.com /en/US/docs/ios_xr_sw/iosxr_r3.2/interfaces/command/reference/hr32lr.html. Accessed 17 Dec 2008.

[CMK+99] A. T. Campbell, H. G. D. Meer, M. E. Kounavis, K. Miki, J. B. Vicente, D. Villela. A survey of programmable networks. *SIGCOMM Comput. Commun. Rev.* 29(2):7–23, 1999.

[dmt07] DMTF Standard DSP 1057: Virtual System Profile. http://www.dmtf.org/standards/published_documents/DSP1057.pdf, May 2007. Accessed 17 Dec 2008.

[EGH+07] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, T. Schooley. Evaluating XEN for Router Virtualization. In *16th Int. Conf. on Comp. Commun. and Networks - ICCCN 2007*. Pp. 1256–1261. Aug. 2007.

[eur07] European Network of the Future (EuroNF), NoE, FP7, grant 216366. http://euronf.enst.fr, 2007. Accessed 17 Dec 2008.

[FGR07] N. Feamster, L. Gao, J. Rexford. How to lease the internet in your spare time. *SIGCOMM Comput. Commun. Rev.* 37(1):61–64, 2007.

[GBH+08] A. E. García, A. Berl, K. A. Hummel, R. Weidlich, A. Houyou, K. D. Hackbarth, H. de Meer, H. Hlavacs. An Economical Cost Model for Fair Resource Sharing in Virtual Home Environments. In *Next Generation Internet Networks, 2008 (NGI 2008)*. Pp. 153–160. 2008.

[gen] GENI - Global Environment for Networking Innovations. http://www.geni.net. Accessed 17 Dec 2008.

[HHW+08] H. Hlavacs, K. A. Hummel, R. Weidlich, A. Houyou, A. Berl, H. de Meer. Distributed Energy Efficiency in Future Home Environments. *Annals of Telecommunication (Special Issue on Home Networking)*, Aug. 2008.

[KS05] S. Kent, K. Seo. IETF RFC 4301: Security Architecture for the Internet Protocol. http://tools.ietf.org/html/rfc4301, Dec. 2005. Accessed 17 Dec 2008.

[MCZ06] A. Menon, A. L. Cox, W. Zwaenepoel. Optimizing Network Virtualization in Xen. In *USENIX Annual Technical Conference*. Pp. 15–28. May 2006.

[PG74] G. J. Popek, R. P. Goldberg. Formal Requirements for Virtualizable third Generation Architectures. *Commun. ACM* 17(7):412–421, 1974.

[SW05] R. Steinmetz, K. Wehrle. *Peer-to-Peer Systems and Applications (Lecture Notes in Computer Science)*. Springer-Verlag New York, Secaucus, NJ, USA, 2005.

[vin] VINI - A virtual network infrastructure. http://www.vini-veritas.net/. Accessed 17 Dec 2008.

[vla03] IEEE Standard 802.1Q: Virtual Bridged Local Area Networks. http://standards.ieee.org/getieee802/download/802.1Q-2003.pdf, May 2003. Accessed 17 Dec 2008.

[vmw] VMWare ESX - Bare-Metal Hypervisor for Virtual Machines. http://www.vmware.com/products/vi/esx/. Accessed 17 Dec 2008.

[WKB+08] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, J. Rexford. Virtual routers on the move: live router migration as a network-management primitive. *SIGCOMM Comput. Commun. Rev.* 38(4):231–242, 2008.