# Virtual Software Routers: A Performance and Migration Study

Zdravko Bozakov

zdravko.bozakov@ikt.uni-hannover.de

**Abstract:** It is expected, that the application of the virtualization paradigm to network resources can provide the basis for a Future Internet architecture and stimulate the introduction of novel protocols, services, and business cases. As a starting point for further research, we analyze the routing performance of a software router executed within different open source virtualization solutions, and evaluate the applicability of existing live-migration mechanisms to virtual routers.

## 1 Introduction

Network virtualization is an emerging technology, which has the potential to address the ossification of the current Internet architecture and become a fundamental building block for the Future Internet. In addition to allowing the operation of independent, parallel virtual networks, e.g. for Infrastructure as a Service (Iaas), virtual routers can be employed for the gradual introduction of new network protocols into existing infrastructures. In the network management domain, a major advantage of virtualized network resources, is the capability to transparently transfer logical router instances between physical routers. ISPs can utilize this feature, to adapt their infrastructure to changing traffic conditions, customer requirements or for energy conservation during off-peak hours.

While proprietary router virtualization solutions, such as logical routers, Virtual Device Contexts (VDC) or Virtual Routing and Forwarding (VRF), are available today, they are too limited in terms of interoperability and flexibility to serve as a basis for a virtual network architecture. Neither router migration, nor the execution of unsupported routing processes is currently possible. Due to the closed nature of commercial offerings, it is likely that research into router virtualization will be confined to open software platforms, at least for the near future.

To this end, we evaluate open source system virtualization solutions with respect to their routing performance, and their potential to serve as a starting point for further research in the network virtualization domain. Furthermore, we analyze the feasibility of existing virtualization technology as a basis for live router migration, and investigate the impact of router mobility on network traffic.

## 2 Related Work

Currently, several virtualization platforms are freely available under open-source licenses, including Xen, OpenVZ, KVM and VirtualBox. The performance level of all solutions has improved steadily, enabling the execution of virtual servers at near native speeds. As a result, today virtualization plays a major role for resource consolidation in data centers. However, I/O performance and the virtualization of network devices in particular, constitute a bottleneck and remain an open research field [AMN06, EGH+07]. An evaluation of Xen for network virtualization has been conducted in [MCZ06]. The concept of router migration as a management primitive was first advocated in [WKB+08], where the authors outline an architecture for control plane migration based on virtual machines (VM). Building upon these findings, we quantify the effects of migration mechanisms on the network, examine the network performance of Xen, KVM and OpenVZ, taking recent developments of the software suites into account.

### 2.1 System Virtualization Approaches

Paravirtualization, full virtualization and container-based virtualization are the three currently prevalent system virtualization approaches.

The Xen [BDF+03] virtual machine monitor (VMM), enables the execution of multiple guest operating systems on a single host machine by providing an abstraction of the underlying hardware. The Xen architecture is comprised of layers, known as domains, with the VMM (or hypervisor) running in the lowest and most privileged domain. The VMM is responsible for instantiating guest domains and managing system resources. Guest hosts require a kernel specifically modified to utilize the interfaces offered by the VMM. A major advantage of paravirtualization approach used by Xen, is the strict VM isolation, as well as the low performance penalty in guest domains which results from the adapted guest kernels. Additionally, the use of virtualization aware network drivers leads to significantly higher network performance. Furthermore, recent versions of Xen also support the virtualization of unmodified guests on CPUs providing hardware virtualization support.

The Kernel-based Virtual Machine (KVM) [KKL+07] project is a full virtualization implementation for Linux. In contrast to Xen, KVM is implemented as a loadable kernel module, which relies exclusively on hardware virtualization extensions, such as Intel VT-x or AMD-V. As a result, no guest-side modifications are necessary: each VM can be regarded as a Linux process executing in a special guest mode. Scheduling is performed using standard Linux mechanisms. On the downside, all guest I/O requests are trapped and emulated in user space by a dedicated process (QEMU), resulting in poor performance for I/O intensive operations. Recently, the availability of paravirtualized network drivers [Rus08] have led to significant improvements in KVM network and disk performance. In contrast to Xen, KVM is included in the main-line Linux Kernel, and can therefore benefit from short development cycles and frequent kernel optimizations.

A more lightweight virtualization strategy is implemented in OpenVZ [ope10]. In con-

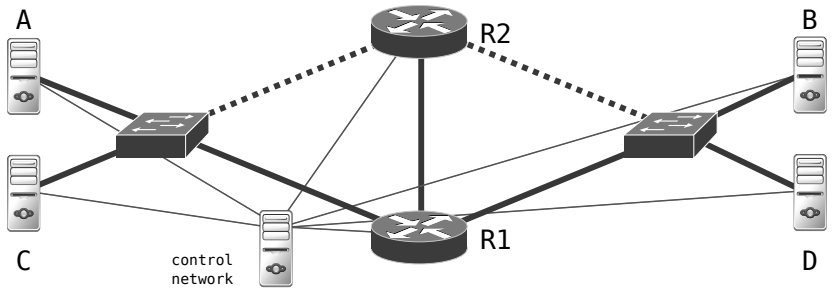



Abbildung 1: Experimental setup

trast to Xen and KVM, OpenVZ provides virtual environments (VE) which share the host machine's kernel. Guest system calls are passed through to the host kernel, creating the illusion of a standalone Linux system, with isolated resources. OpenVZ offers quotas, which can be used to control the allocation of physical resources. Even though VEs are not a true virtualization solution, they can be checkpointed and migrated across physical machines. The reduction of processing overheads, is the main motivation behind this approach.

## 3 Applications of Virtualization in Routers

From a network operator's viewpoint, two main advantages of network virtualization are the ability to support multiple concurrent networks, each potentially running different protocols, as well as increased flexibility resulting from the decoupling of physical and logical network entities.

Depending on the use-case requirements, different elements of a router architecture can be virtualized.

A prerequisite for live router migration, is the strict separation of control and forwarding planes. A router control plane running within a virtual environment (VE) can hence be transparently migrated while the data plane continues to forward traffic, as outlined in [WKB+08]. In order to ensure that network operation is not disrupted, for instance by dropped route update messages, it is vital to minimize the control plane downtime during the migration process.

Moreover, virtualization mechanisms can be employed to increase router availability. A possible approach consists of using control plane VE snapshots to ensure a quick router recovery e.g. after a crash. Additionally, it is conceivable that fallback control plane instances can be operated concurrently on separate hardware platforms, by suitably modifying existing migration mechanisms. The use of virtualized routers for testing alternative router configurations has also been proposed in [AWY08].

To operate isolated, concurrent networks on the same hardware substrate, the forwarding plane design must be adapted to support virtualization. High performance and efficient

sharing of hardware resources are key requirements in this case. At the same time, a virtual forwarding plane should be expandable, in order to support a wide range of protocols - a level of flexibility similar to system virtualization is desirable. The recently proposed OpenFlow [MAB+08] framework, is a highly promising approach for data plane virtualization. A different architecture has been proposed in [AF09].

Finally, a straightforward but highly demanding approach consists of executing an entire router within a virtualized environment. It can be expected, that due to the performance penalty associated with virtualization, the real-world use of fully virtualized routers will be limited in the foreseeable future. Nevertheless, we believe such a setup serves as a means to identify existing technologies which can be adopted for future use in specialized network virtualization solutions. Therefore, this option is evaluated in this paper.

### 3.1 Live Migration

The most basic VM migration strategy involves creating a VM snapshot (checkpointing), stopping VM execution, transferring the VM snapshot to a destination host, and resuming execution there. This approach, commonly referred to as *stop-and-copy*, is employed by OpenVZ [MKK08]. The downside of this mechanism is that it introduces a significant network downtime, approximately equivalent to the time needed to transfer the allocated VM memory over a link with a given capacity.

Xen employs a so-called *iterative pre-copy* migration mechanism followed by a *stop-and-copy* step [CFH+05]. When a migration is initialized, the guest's entire memory content is transfered to the destination host, and logging of modified (dirty) memory pages is enabled on the source VM. While the source VM continues to execute, pages modified since the beginning of the migration are iteratively copied to the destination VM until the number of dirty pages falls below a pre-defined threshold or a time limit is reached. Xen dynamically adjusts the migration transfer rate in order to minimize the disruption of the network. In the final step, the source VM is stopped, and the remaining dirty pages are copied to the destination VM at maximum speed. Ideally, this final working set should make up a fraction of the entire memory content, resulting in minimal network downtime. In this work we aim to quantify this disruption.

A similar approach is used in KVM. However, as shown in the results section, the implementation details are different.

For the remainder of this paper, we focus on the migration mechanisms of Xen and KVM.

## 4 Experimental Setup

We used the setup depicted in Fig. 1 to evaluate the routing performance of virtualized software routers and the effects of live migration on network traffic. Four nodes were connected by Gigabit Ethernet links, and additionally attached to a separate control net-

work. Nodes A and B act as a UDP traffic source and sink respectively. As it is not possible to saturate a Gigabit link using commodity hardware and operating systems, we utilized NetFPGA cards as traffic generators. The NetFPGA packet generator [CGLM09] is capable of transmitting 64 byte packets at line rate. A XORP software router [HHK03] was started within a VM/VE on node R1 and was subsequently migrated to R2 over a dedicated link. We believe, that the assumption of dedicated resources for router migration is realistic for network operation centers. Each VM was assigned 256 MB RAM and a single CPU core. R1 and R2 were booted from a live-CD, eliminating the need for a shared file-system. In addition, we measured the round-trip time (RTT) between nodes C and D before and during the migration process.

We used Xen version 3.2.1, KVM version 84 and OpenVZ version $1.2133\_FC5.026$. Dell Optiplex 760 with Q8400 Core 2 Quad CPUs and 4GB RAM were used for nodes R1 and R2. The nodes were equipped with Quad Port Intel PRO/1000 network cards. Each network interface was bound to a dedicated Linux bridge.

All measurements were automated using [BB08] and repeated 25 times. $95\%$ confidence intervals are included in all plots.

# 5 Results

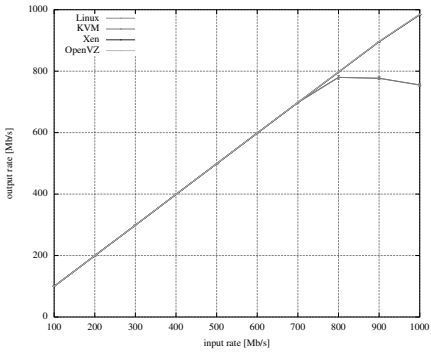
## 5.1 Virtual Machine Forwarding

We measured the performance penalty associated with routing packets within a VM, by comparing input and output packet rates. We used constant bit rate, UDP cross-traffic consisting of maximum and minimum sized Ethernet packets (1500B and 64B respectively). As a baseline the forwarding performance of a bare Linux system was also measured.

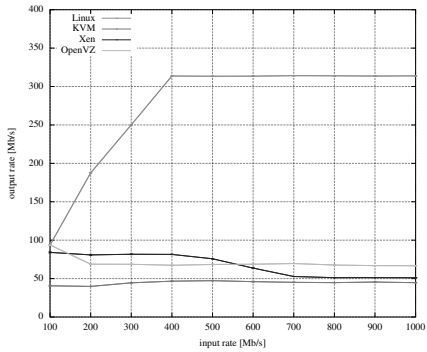
All evaluated virtualization approaches rely on Linux bridging to provide network connectivity to VMs. KVM and OpenVZ utilize user space virtual network devices (TAP) to the enable access to the network hardware, while the Xen hypervisor provides dedicated kernel space interfaces.

With large cross-traffic packets the forwarding performance of Linux, Xen and OpenVZ, was identical and sufficient to fully saturate the Gigabit link. Using the *virtio* network driver, KVM was able to fill almost $80\%$ of the link. In contrast, the fully virtualized Intel *e1000e* driver, KVM performance was extremely poor at $\sim 5$Mbps.

While Linux forwarding was able to achieve slightly over $300$ Mbps using small packet cross-traffic, all virtualization approaches achieved significantly lower throughput speeds. It is interesting to note that the forwarding performance of Xen and OpenVZ deteriorates after $500$Mbps and $100$Mbps respectively. KVM throughput was constant at only $\sim 5\%$ of the link capacity. Analysis of the traces showed that in all cases, the discrepancy between the Linux and VM throughput values is mainly due to packet loss between the physical input interface and the bridged virtual interface, rather than loss withing the VM. This implies, that the Linux bridging system might be the cause for significant packet loss.

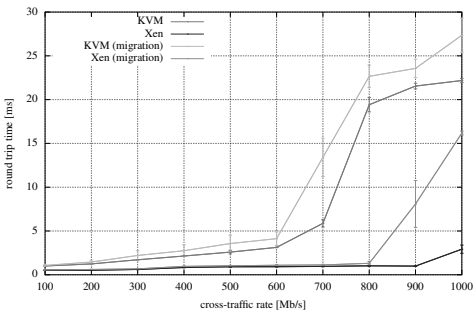



(a) 1500B packet cross-traffic

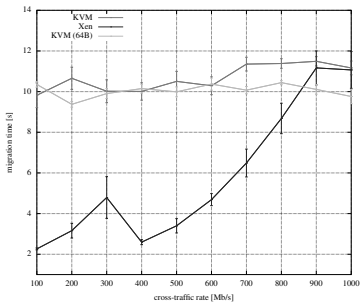



(b) 64B packet cross-traffic

Abbildung 2: Forwarding performance




(a) Packet round trip time



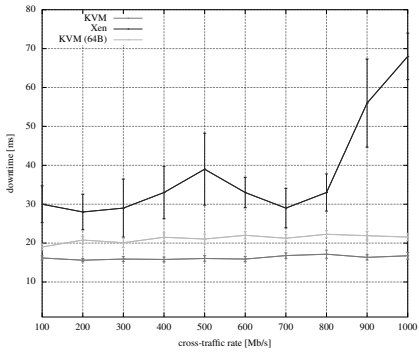

(b) Migration duration

In addition, we measured the performance of Xen's PCI pass-through capability, which allows for network devices to be assigned exclusively to a virtual machine. Results were identical to the pure Linux forwarding case and are not shown here.

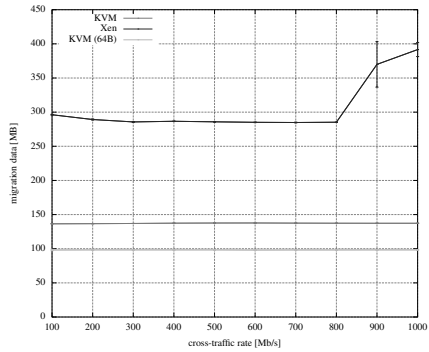
The forwarding results, are depicted in Fig. 2. Additionally, Fig. 3a shows the round trip time dependency on the cross-traffic.

### 5.2 Live-Migration Effects

A dedicated link between R1 and R2 was used for the migration traffic, in order to examine the influence of the migration process on the network. Migration while forwarding small cross-traffic packets proved a challenging task for both Xen and KVM. The Xen migration process failed repeatedly, even leading to reproducible lockups of the virtual machine. This poses a significant problem, limiting the usability of Xen migration in real-world scenarios.




(a) Migration downtime




(b) Migration traffic volume

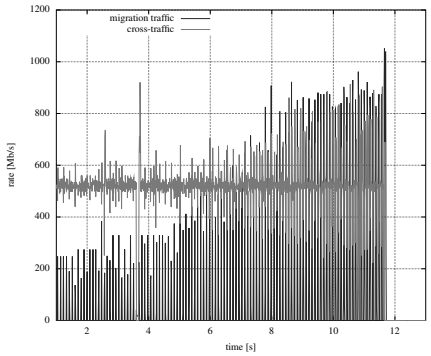
In the following, we only present results for large packet cross-traffic under Xen.

The migration process at R1 for KVM and Xen for 500Mbps and 1000Mbps cross-traffic respectively are exemplified in Fig. 3. The migration is initiated at time 0, with the stop-and-copy peak visible in the right end of each migration traffic curve. Traffic data was collected using the Linux *proc* packet counters, with 10ms resolution. It is notable, that the total length of the migration process appears reasonably constant for KVM. The migration is performed as a series of bursts, of increasing transfer rates, each followed by a inactivity period. As the migration progresses, the migration rate increase, adversely affecting the cross-traffic rate. In contrast, using Xen the migration traffic rate correlates with the intensity of the cross traffic, resulting in shorter migration times at low cross-traffic rates.

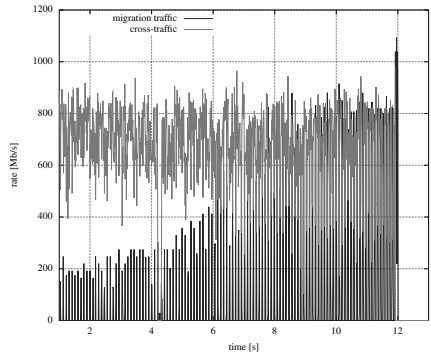
These notions are confirmed by our series of automated experiments. For both virtualization approaches, we measured the total duration of the migration process, the network downtime caused by the stop and copy phase, and the bandwidth generated by the migration. Using Xen, the total duration time of the migration showed a dependency on the cross traffic intensity, ranging from 2s up to 12s. An unexpected increase in the migration time can be observed at a cross-traffic rate of 300Mbps. For the KVM migration implementation, no significant cross-traffic dependency was apparent. The results are depicted in Fig. 3b.

The network downtime introduced by the final stop and copy migration phase was independent of the cross-traffic at $\approx 0.17$s for KVM and highly variable for Xen ranging from 3s to 7s. For small packet cross traffic the KVM downtime is slightly higher than for large packets. The relationship is shown in Fig. 3a.

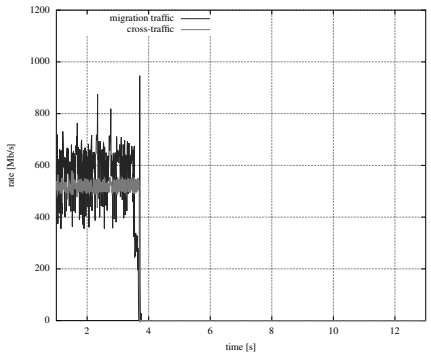
Fig. 3b shows the total amount of data transferred during the VM migration. Using Xen, the migration traffic is slightly higher than the allocated memory for the VM. At 900Mbps cross-traffic, the amount of data increases significantly. It is evident, that the migration traffic volume can not fully account for the increase in migration time. Hence, the behaviour visible in Fig. 3b must be due to Xen's migration rate adaptation mechanism.



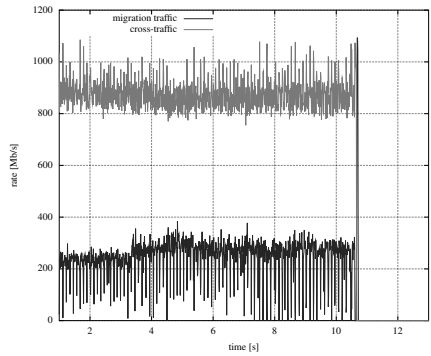

(a) KVM: 500Mbps cross-traffic




(b) KVM: 1000Mbps cross-traffic




(c) Xen: 500Mbps cross-traffic




(d) Xen: 1000Mbps cross-traffic

Abbildung 3: Migration process samples

For KVM no notable increase in the amount of transferred data is evident. It is noteworthy, that the constant data volume of $\approx 140$MB is less than the size of assigned VM memory.

Additionally, the round trip increase during the migration process is plotted in Fig. 3a. As expected, the RTTs increase as the cross-traffic intensity rises. For Xen, an exceptionally large increase occurs at cross-traffic speeds larger than $800$Mbps.

## 6 Conclusion

Our evaluation showed, that the performance of the Linux network stack running on commodity hardware is insufficient to serve a full Gigabit network link using minimum sized Ethernet packets. Not surprisingly, the virtualization layers generate additional packet pro-

cessing overhead, leading to further performance deterioration. Regardless of the utilized virtualization approach, the penalty on network throughput for small packet sizes remains extremely high. Moreover, under such conditions, the Xen migration process did not complete reliably in a significant number of cases. This constitutes a major obstacle for the utilization of Xen based migration of software routers in real-world scenarios.

Nevertheless, examining traffic with larger packet sizes showed that the forwarding performance was acceptable for Linux as well as all virtualization approaches. Among the evaluated solutions, Xen delivers the best forwarding performance for small Ethernet packets. KVM's full virtualization architecture, combined with paravirtualized network drivers achieves results almost comparable to Xen. Surprisingly, in terms of network performance, OpenVZ's lightweight, container-based architecture does not achieve significant improvements and is on par with Xen. Further work is required to verify indications that the Linux bridging system represents a bottleneck in the system, as all evaluated approaches rely on it to provide connectivity between physical and virtual network devices.

We confirmed that the simple migration mechanism employed in OpenVZ leads to significant downtimes, and that extremely short network disruptions can be achieved using the iterative stop and copy migration mechanism. Assuming migration stability issues are addressed in future software versions, the migration functionality of both Xen and KVM can be applied for applications with moderate throughput requirements. The migration of the control plane of a software router represents a feasible scenario. However, we expect that dedicated hardware support is essential for the implementation of a high performance forwarding-plane in virtual routers. In future work, we aim to present a migratable virtual router platform, with an OpenFlow based data plane, where the controller responsible for computing routing tables is executed within a virtual environment.

## Literatur


[AF09] Muhammad Bilal Anwer und Nick Feamster. Building a fast, virtualized data plane with programmable hardware. In *VISA '09: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 1–8, New York, NY, USA, 2009. ACM.

[AMN06] P. Apparao, S. Makineni und D. Newell. Characterization of network processing overheads in Xen. *Virtualization Technology in Distributed Computing, 2006. VTDC 2006. First International Workshop on*, pages 2–2, Nov. 2006.

[AWY08] Richard Alimi, Ye Wang und Y. Richard Yang. Shadow configuration as a network management primitive. *SIGCOMM Comput. Commun. Rev.*, 38(4):111–122, 2008.

[BB08] Zdravko Bozakov und Michael Bredel. SSHLauncher - A Tool for Experiment Automation. Technical report, TU-Darmstadt, 2008.

[BDF+03] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt und Andrew 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 Press.

[CFH+05] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt und Andrew 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.

[CGLM09] G.A. Covington, G. Gibb, J.W. Lockwood und N. Mckeown. A Packet Generator on the NetFPGA Platform. In *Field Programmable Custom Computing Machines, 2009. FCCM '09. 17th IEEE Symposium on*, pages 235–238, April 2009.

[EGH+07] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy und T. Schooley. Evaluating Xen for Router Virtualization. *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, pages 1256–1261, Aug. 2007.

[HHK03] Mark Handley, Orion Hodson und Eddie Kohler. XORP: an open platform for network research. *SIGCOMM Comput. Commun. Rev.*, 33(1):53–57, 2003.

[KKL+07] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin und Anthony Liguori. kvm: the Linux Virtual Machine Monitor. In *Proceedings of the Linux Symposium*, June 27th–30th 2007.

[MAB+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker und Jonathan Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, 2008.

[MCZ06] Aravind Menon, Alan L. Cox und Willy Zwaenepoel. Optimizing Network Virtualization in Xen. In *In Proc. USENIX Annual Technical Conference (USENIX 2006)*, pages 15–28, 2006.

[MKK08] Andrey Mirkin, Alexey Kuznetsov und Kir Kolyshkin. Containers checkpointing and live migration. In *Proceedings of the Linux Symposium*, July 23rd–26th 2008.

[ope10] OpenVZ. http://www.openvz.org, January 2010.

[Rus08] Rusty Russell. virtio: towards a de-facto standard for virtual I/O devices. *SIGOPS Oper. Syst. Rev.*, 42(5):95–103, 2008.

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