System-level virtualization and Mobile IP to support service mobility

Vittorio Manetti, Roberto Canonico,
Giorgio Ventre
Dipartimento di Informatica e Sistemistica
Universitá di Napoli Federico II
via Claudio 21, 80125 Napoli, ITALY
Email: {vittorio.manetti, roberto.canonico}@unina.it,
giorgio.ventre@unina.it

Ioannis Stavrakakis
Department of Informatics and Telecomunications
National and Kapodistrian University of Athens,
Athens, Greece
Email: ioannis@di.uoa.gr

***Abstract*—In many areas of ICT the use of virtualization techniques has decoupled the binding between physical resources and functional roles assigned to them. While the use of virtualization in computing systems is already extremely popular, similar forms of virtualization have been proposed for the network infrastructure as well. In such a context, we propose a new paradigm that aims at extending the concept of virtualization to network services, by decoupling service execution environments and their physical location. We call this paradigm *Service Switching*. In a Service Switching environment, service instances may be dynamically migrated across geographically dispersed data centers to pursue better usage of both network and computing resources.**

I. Introduction

The current Internet has been designed according to a simple model comprising two different kinds of entities: end-systems and intermediate devices. In such a model, applications and services run in end-systems, located at the network edges, while intermediate entities are responsible of packet handling operations, such as routing, filtering, address translation. In recent years, such a clear distinction between services and traffic management is becoming blurred, and boundaries between network infrastructures and computing data centers are progressively vanishing as well.

Following such trend, the use of virtualization techniques is becoming extremely popular in many areas of ICT. Virtualization is used in computing systems with the main goal to decouple the binding between physical resources and functional roles assigned to them, but recently, similar technique have been proposed for the network infrastructure as well [1].

In this paper we propose the *Service Switching*, a new paradigm that aims at extending the concept of virtualization to network services, by decoupling service execution environments and their physical location. Service instances in a Service Switching environment may be dynamically migrated across geographically dispersed data centers, with the goal to pursue better usage of both network and computing resources.

Service mobility is a key feature for new generation networks. In the ubiquitous computing context, for instance, supporting the mobility of services may help in providing better quality of experience to mobile users, while they wander across heterogeneous environments. In distributed service hosting environments, requirements like efficient management of available resources, the ability of implementing computational load balancing strategies, and the ability of detecting and reacting to critical conditions to guarantee service continuity are of the uttermost importance.

The optimal placement and selection of services [2] is a key issue for efficient service provisioning in large scale networks. An example of large-scale bandwidth service could be the real-time distribution of software updates and patches, or of virus definition files, etc. Similar kind of services must cope with the typically voluminous and bursty demand, both in terms of overall load and geographical distribution of the sources of demand, due to flash-crowd phenomena. To effectively deploy such services, decisions must be made on one hand on the location, and on the other hand on the number of nodes used to deliver the service. A scalable approach [3] consists in implementing a scheme in which an initial set of service facilities are allowed to migrate adaptively to the best network locations, and optionally to increase/decrease in number so as to best service the current demand.

Migration of complete service instances is a solution for most of the above mentioned requirements. While migration of executables (*code migration*) is somewhat easy to solve, *data migration* is less trivial, but this latter solution does not suffer of the problem that is knowed as *residual dependency*. The Service Switching paradigm offers to *Application Service Providers* (ASPs) the possibility of deploying network services on a geographically distributed infrastructure, and implements mechanisms for transparent service migration.

The rest of this paper is organized as follows. Section II presents some details about the system-level virtualization technologies and the Mobile IP classical model, the background we acquired before defining the proposed paradigm and architecture, that are introduced respectively in section III and IV. In section V we present related work concerning the use of virtualization in the networking context, and the service mobility problem. Finally, section VI concludes the paper.

II. Technologies employed

A. System-level Virtualization mechanisms

Virtualization is a widely used technique in which a software layer multiplexes lower-level resources among higher-level software programs and systems. *Virtuality* differs from *reality* only in the formal world, while possessing a similar essence or effect. A virtual environment is perceived the same as that of a real environment by application programs and the rest of the world, though the underlying mechanisms are formally different. The virtual environment presents a misleading image of a resource that has more or less capability compared to the physical resource underneath.

A typical computer system already uses virtualization; conceptually, a Virtual Machine (VM in short) represents an operating environment for a set of user-level applications. Each VM is used to be an instance of the physical resource that gave users an illusion of accessing the physical resource directly. It was an elegant and transparent way to enable time-sharing and resource-sharing on the highly expensive hardware. There can be several levels of abstraction where virtualization can take place: instruction set level, hardware abstraction layer (HAL), operating system level (system call interface), user-level library interface, or in the application level. Whatever may be the level of abstraction, the general mechanism still remains the same: it partitions the lower-level resources using some novel techniques to map to multiple higher level VMs transparently.

Modern computers are sufficiently powerful to use virtualization to present the illusion of many smaller VMs each running a separate operating system instance. Accordingly to this consideration, the main use for VMs is to enable execution of a range of applications originally targeted for different hardware and operating systems on a given machine. When applied to a complete computing system, the technique is referred as *System-level Virtualization.* System-level Virtualization is the faithful reproduction of an entire architecture in software which provides the illusion of a real machine to all software running above it. A virtualized system includes a new layer of software, the *Virtual Machine Monitor* (VMM in short).

Successful partitioning of a machine to support the concurrent execution of multiple operating systems poses several challenges. Firstly, VMs are isolated from one another: it is not acceptable for the execution of one to adversely affect the performance of another. This is particularly true when VMs are owned by mutually untrusting users. Secondly, allowing to support a variety of different operating systems accommodates the heterogeneity of popular applications. Thirdly, the performance overhead introduced by virtualization should be small.

A VMM manages the creation, destruction and control of one or more VM on a computer, and it is responsible for controlling access to the resource of the real hardware as well as multiplexing the execution of multiple VMs fairly. The VMM provides a virtual processor and other virtualized versions of system devices such as I/O devices, storage,

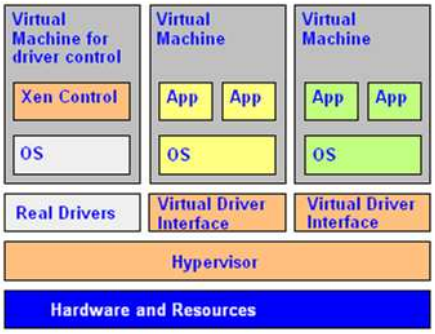

Fig. 1. System-level virtualization

memory, etc, and it also provides isolation between the hosted VMs, so that problems in one cannot effect another. VMs do not access the system's real resources directly, but through the VMM.

In a traditional VMM, the virtual hardware exposed is functionally identical to the underlying machine, so that operating system and software may run on the virtual hardware exactly as they would on the original hardware. Although, the so called *Full Virtualization*, has the obvious benefit of allowing unmodified operating systems to be hosted, it also has a number of drawbacks. This is particularly true for the prevalent x86 architecture, because of support for full virtualization was never part of such kind of architecture. A more recent solution is *Paravirtualization*, a technique that avoids the drawbacks of full virtualization by presenting a VM abstraction that is similar but not identical to the underlying hardware. This promises improved performance, although it does require modifications to the guest operating system. Paravirtualization, however, does not require changes to the application binary interface (ABI), and hence no modifications are required to guest applications. Paravirtualization system have the potential for improved scalability and performance over prior VMM implementations.

The implementation of the model we are going to introduce is based on Xen [4], a Virtual Machine Monitor developed by the University of Cambridge. Xen provides a VM monitor for x86 processors that supports execution of multiple guest operating systems at the same time. In Xen, a guest Operating System is identified as XenoLinux, each of which exports an ABI that is identical to a non-virtualized kinux kernel. Xen's approach to paravirtualizing the x-86 architecture can be summarized as follows:

1. Each VM is given read only access to the *hardware page tables*, updates are queued and processed by the VMM.
2. Xen runs guest VMs at a lower hardware priority level than the VMM.
3. Guest operating systems must register them with the Xen VMM.
4. Guest operating systems can install their system call

handlers directly with the processor, bypassing the context switches necessary for the VMM to process the trap.
5. Hardware interrupts are replaced by a lightweight event system.
6. Xen only provides to the guest VM a small set of fast, generic devices.

B. IP Mobility Support

IP version 4 assumes that the IP address of a node uniquely identifies its point of attachment to the Internet. A node must be located on the network indicated by its IP address in order to receive datagrams which are destined to it. If a node needs to change its point of attachment without losing the ability to communicate, one of the two following mechanisms can be employed:

a) The node has to change its IP address whenever it changes its point of attachment to the Internet.
b) Host-specific routes have to be propagated throughout much of the Internet routing fabric.

Both of these alternatives are often unacceptable. The first makes it impossible for a node to maintain transport and higher-layer connections when the node changes location. The second has obvious and severe scaling problems.

IP Mobility Support (or Mobile IP) provides a mechanism which enables nodes to change their point of attachment to the Internet without changing their IP address. In Mobile IP, a mobile node is considered as a node that can communicate with other nodes after changing its link-layer point of attachment to the Internet, even though such nodes do not implement these mobility functions. Mobile IP enables nodes to move from one IP subnet to another, and supports mobility across both homogeneous and heterogeneous media.

Mobile IP introduces the following new functional entities:

1) Mobile Node: A host or router that changes its point of attachment from one network or subnetwork to another. A mobile node may change its location without changing its IP address; it may continue to communicate with other Internet nodes at any location using its (constant) IP address, assuming link-layer connectivity to a point of attachment is available.

2) Home Agent: A router on a mobile node's home network which tunnels datagrams for delivery to the mobile node when it is away from home, and maintains current location information for the mobile node.

3) Foreign Agent: A router on a mobile node's visited network which provides routing services to the mobile node while registered. The foreign agent detunnels and delivers datagrams to the mobile node that were tunneled by the mobile node's home agent. For datagrams sent by a mobile node, the foreign agent may serve as a default router for registered mobile nodes.

A mobile node is given a long-term IP address on a home network. This home address is administered in the same way as a permanent IP address provided to a stationary host. When away from its home network, a *care-of address* is associated with the mobile node and reflects the mobile node's current point of attachment. The mobile node uses its home address as the source address of all IP datagrams that it sends, except for datagrams sent for certain mobility management functions.

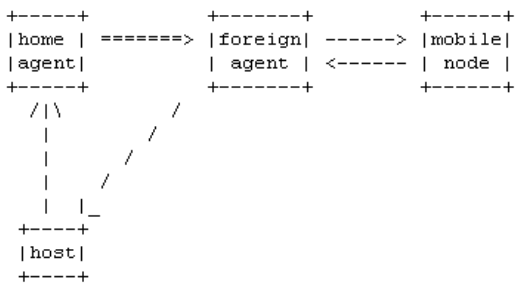

Fig. 2. IP Mobility Support

When away from home, Mobile IP uses protocol tunneling (as shown in figure 2) to hide a mobile node's home address from intervening routers between its home network and its current location; such tunnel terminates at the mobile node's care-of address. The care-of address has to be an address to which datagrams can be delivered via conventional IP routing. At the care-of address, the original datagram is removed from the tunnel and delivered to the mobile node.

Aiming to support the aformentioned facilities, Mobile IP imposes the following essential rules:

1) A home agent *must* be able to attract and intercept datagrams that are destined to the home address of any of its registered mobile nodes.
2) A mobile node and a prospective or current foreign agent *must* be able to exchange datagrams without relying on standard IP routing mechanisms; that is, those mechanisms which make forwarding decisions based upon the network-prefix of the destination address in the IP header.

III. The Service Switching Paradigm

For several classes of applications and services, the traditional paradigm based on a server instantiated at a fixed location at the edge of the network, has proven to be not completely satisfactory. Some services could benefit if they were implemented in intermediate network devices rather than at the edges, where they could intercept and aggregate service requests. Other services present very stringent requirements in terms of resiliency, scalability and security, and they could benefit from the availability of network-provided migration and replication mechanisms.

The Service Switching paradigm allows creation and management of *Service Execution Environments* across different data-centers located within a single network domain with minimal impact on service continuity.

The architectural implementation of the Service Switching paradigm is centered around a main component, that we call *Service Switch.* Such a component is a network node that, in addition to the plain packet and/or flow switching capabilities, has more advanced features, such as the ability to:

a) host and manage customizable Service Execution Environments;
b) provide guarantees to existing Service Execution Environments, in terms of available computing and communication resources, isolation and security;
c) manage service location and migration;
d) detect and react to critical conditions;
e) coordinate with other Service Switches for efficient management of resources at a global level.

A Service Switch could be located at the edge of an Autonomous System Domain in order to monitor and manage services, and it could well be located also in the core part in order to provide fast reconfiguration and migration of services and to re-allocate service load in critical conditions.

Some examples of service that could benefit of the capabilities offered by the model we propose: (i) Multimedia Transcoding, (ii) Content Distribution, (iii) Intrusion Detection, (iv) services based on P2P technologies, (v) Remote Authentication, (vi) VoIP.

In the following section we present the architectural design of an infrastructure supporting the Service Switching paradigm.

IV. Service Switching Architecture

System-level virtualization techniques (e.g. those based on the paravirtualization model) allow execution of Virtual Machines (VMs in short) at levels of performance very much close to those achievable without virtualization [4]. Modern virtualization techniques, adopting the so-called eagerly approach, also support fast and efficient VM migration with no *residual dependency* [5]. These considerations led us to the conclusion that Service Execution Environments can be implemented as VMs. In the context of Service Switching, a VM hosts one service, or a set of logically dependent services. Service Switching allows services to be deployed at different geographic locations, each of which hosts a cluster of physical machines. All the VMs hosted in the same physical cluster are collectively identified as a *Virtual Cluster*. A physical cluster is connected to the Internet through a special router, that we call *Edge Service Switch*. A Service Switching enabled infrastructure consists of several Virtual Clusters located at different sites. The main purpose of the infrastructure consists in creating and managing the VM lifecycle, allowing transparent VM migration from one Virtual Cluster to another.

When a VM is first created, it is associated to one of the available Virtual Clusters. The IP address associated to a VM at creation time will be kept for the entire VM lifecycle, even in case of VM migration. Such IP address is referred to as the VM's *Home Address*. The IP subnet to which the VM's Home Address belongs to is the VM's *Home Network*. The Edge Service Switch located at the edge of the VM's Home Network will be referred to as the VM's *Home Service Switch*. An Edge Service Switch not only behaves as a normal IP edge router, forwarding incoming packets to the VMs hosted in the cluster and outgoing packets to a next hop router according to its current routing table, but also implements specific traffic flow

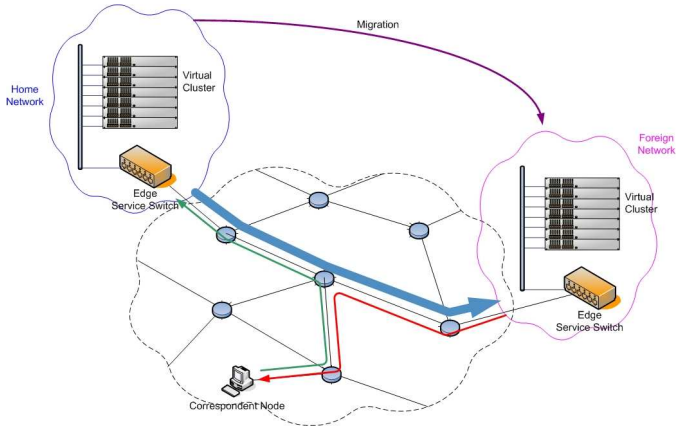

Fig. 3. Tunneling mechanism implemented on the edge

readdressing mechanisms to support service migration. Such mechanisms have been derived as extensions of the classical Mobile IP model [6]. A generic end user terminal accessing a service will be referred to as *Correspondent Node*.

In order to access a given service, a Correspondent Node sends packets to the VM on which that service is running, according to a specific application layer protocol, using the VM's Home Address as IP Destination Address. Incoming packets will be processed by the VM's Home Service Switch. By querying a *Mobility Binding Table* (MBT in short), the Home Service Switch checks if the VM to which packets are addressed is residing in its Home Network or not.

In accordance with the classical Mobile IP model, the MBT has an entry for each migrated VM, and keeps the association between the VM's Home Address and the corresponding *Care-of Address*. Such Care-of Address is the IP address of the Edge Service Switch associated to the Virtual Cluster hosting the migrated VM, that we may call the VM's *Foreign Service Switch*. The IP subnet associated to the hosting Virtual Cluster may be referred to as the migrated VM's *Foreign Network*. After a VM has been migrated, since migration must be transparent to VMs, it keeps using its own Home Address as IP source address for outgoing packets. Service migration is performed through a procedure called *registration*, consisting in updating the Home Network's MBT, by creating an entry for the migrated VM. Since VM migration is transparent to Correspondent Nodes as well, they keep sending packets by using the VM's Home Address as IP Destination Address. Once these packets reach the Home Service Switch, this latter forwards them to the Foreign Service Switch, by encapsulating such packets in a point-to-point tunnel (figure 3). The Foreign Service Switch, in turn, de-tunnels the incoming packets and delivers them to the migrated VM. As it happens in the Mobile IP scheme, reverse traffic is sent by the migrated VM directly to the Correspondent Nodes.

The Service Switching architecture can also be extended with Service Switch nodes deployed in the core of the network, that we may call *Core Service Switches*. Deploying such Core Service Switches it is possible to optimize the packet forward-

Home Address	Protocol	Source Port Number	Private IP Address	Target Port Number	Care-of Address
143.225.229.254	TCP	80	192.168.10.10	80	143.225.172.254
143.225.229.254	TCP	8080	192.168.10.14	80	143.225.81.254
143.225.229.254	UDP	5060	192.168.10.12	5060	192.167.9.254
...	...	...	...	...	...

TABLE I
EXTENDED MOBILITY BINDING TABLE

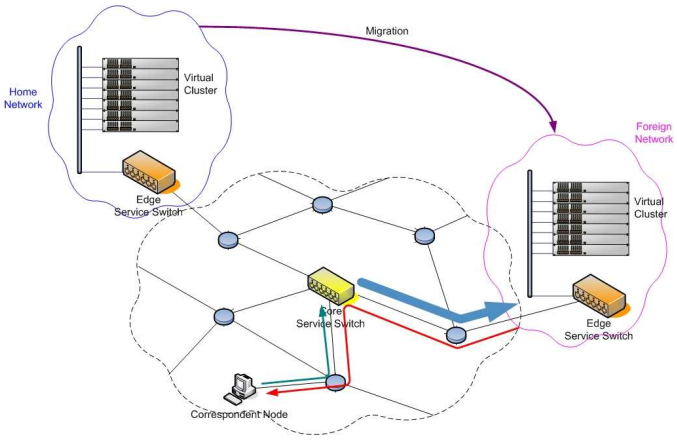

Fig. 4. Tunneling mechanism implemented in the core

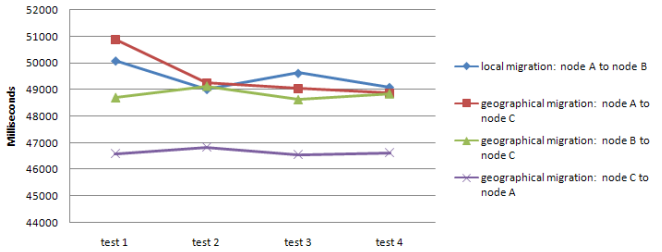

Fig. 5. Tests on local and geographical VM migration

ing mechanism for packets addressed to migrated VMs, by creating shorter paths from Correspondent Nodes to the VM's Foreign Service Switch. When a VM has been migrated to a different site, the Core Service Switch is able to readdress packets directed to the VM by establishing a point-to-point tunnel with the Foreign Service Switch, on behalf of the VM's Home Service Switch (figure 4).

A Service Switching enabled infrastructure is capable to manage the service life cycle in a completely transparent and automatic way, referring in particular to the activation and migration procedures. Moreover, some nodes composing the architecture have the task to detect critical conditions, and eventually reorganize and reallocate the computational load on the architecture in order to solve such conditions.

We realized a port forwording mechanism implemented on the Service Switch nodes, aiming to allow the use of private IP addresses on a Service Switching enabled infrastructure. Table I shows an extended version of the Mobility Binding Table which is accessed during the port forwarding process.

A. Preliminary experimental results

Our first prototype implementation of the Service Switching architecture relies on Linux-based PCs and Xen. Such prototype has been used to implement a flexible and dynamically reconfigurable Content Delivery Network (CDN). Our first tests (the results are reported in figure 5) have been targeted at measuring the migration time of the CDN service facilities, and its impact on the CDN service downtime. These tests have shown that migration time of a Service Execution Environment does not depend on the particular kind of application, but it depends on the amount of RAM associated to the VM and, of course, on the available bandwidth between source and destination sites.

V. RELATED WORK

In the past few years virtualization has re-emerged as a hot topic in operating systems research, and it seems natural to wish to extend these benefits to network routing. If a single hardware platform can simultaneously perform the roles of multiple independent routers, then many practical applications are enabled. For example, a telecommunications provider might install one router in an office building, and support many separate small business customers on the same hardware platform, while allowing each to operate and configure their own virtual router. Sustaining that virtual routers can be much more flexible, Egi et al. [7] focused on the case of single commodity hardware platform using operating system virtualization to perform the roles of multiple independent edge routers. They evaluate the performance of a software IP router realized inside the Xen virtual machine monitor environment, with the goal to identify design issues in Virtual Routers.

Wang et al. [8], [9] propose VROOM (Virtual ROuters On the Move), a new network architecture where virtual routers can freely move from one physical router to another. In VROOM, the physical routers merely serve as the carrier substrate on which the actual virtual routers operate. VROOM can migrate a virtual router to a different physical router without disrupting the flow of traffic or changing the logical topology, obviating the need to reconfigure the virtual routers while also avoiding routing-protocol convergence delays. For example, if a physical router must undergo planned maintenance, the virtual routers could move (in advance) to another physical router in the same Point-of-Presence (PoP). In addition, edge routers can move from one location to another by virtually re-homing the edge links that connect to neighboring domains. Virtual router migration implemented with the VROOM framework is

possibleby leveraging and combining the following technical innovations: (i) Virtual routers, (ii) Virtual machine migration, (iii)Programmable transport layers, (iv) Packet-aware access networks.

In this paper we propose the Service Switching paradigm as a solution for service mobility, and we introduce a new network node architecture (the Service Switch) implementing functionalities aimed at providing support for such a mechanism. Some other models tackle the problem of seamless mobility without changes to the Internet architecture for the case when a user does not have a permanent IP address. Such solutions leverages the fact that most users do not care about ubiquitous reachability, but they use solutions like dynamic DNS. Therefore, the mobility problem reduces to maintaining ones workspace, including all existing network connections, when moving between networks with minimal overhead. Feldman et al. [10] propose a solution to such problem; the key idea is to allow any new connection to use the current IP address, taking advantage of the heavytailed nature of connections. With the majority of sessions being short-lived, only a small number of connections need to be retained after a move. They implemented these ideas within the Seamless InternetMobility System (SIMS). SIMS enables mobility even for users that do not have a permanent IP address and therefore cannot rely on a Mobile IP home agent. SIMS imposes no overhead for applications initiating network traffic in the current network, and it preserves sessions that started in any previously visited network location. SIMS is robust, scalable, and easily deployable in the current Internet, and it addresses economic issues of roaming between different providers.

Exploiting system-level virtualization technologies, the Internet Suspend/Resume model [11], [12] cuts the tight binding between PC state and PC hardware. The ISR system emulates the suspend/resume capability of laptop hardware, allowing to use hardware transiently at any location. ISR builds on two technologies: virtual machine and distributed storage. Each VM encapsulates a distinct execution and user-customization state called parcel. The distributed storage layer transports a parcel across space (from suspend site to resume site) and time (from suspend instant to resume instant). Users can own multiple parcels, just as they can own multiple machines with different operating systems or application suites.

VI. Conclusion and Future Work

Virtualization techniques have found widespread adoption in many areas of ICT. Forms of virtualization have been recently proposed as the basis to create more flexible and extensible network infrastructures. In this paper we present *Service Switching*, a new paradigm that aims at extending the concept of virtualization to network services, by decoupling service execution environments and their physical location. We also present an implementation of this model, relying on the combination of Xen and Mobile IP, and supporting transparent migration of service instances across geographically dispersed data centers to pursue better usage of both network and computing resources. In this paper we also present a smooth transition for the introduction of Service Switching into the network. While the paradigm may be implemented in a minimal form only at the network edges, it can fully exhibit its potential with support of what we call *Core Service Switches*, whose role is to optimize packet routes towards migrated VMs. Our next efforts are aimed at defining distributed management procedures that may be used to efficiently manage both the computing and communications resources of a Service Switching infrastructure.

VII. Acknowledgements

This work has been supported by the European Union under the IST Content (FP6-2006-IST-507295) project. The CONTENT Network of Excellence targets Content Delivery Networks for Home Users, as an integral part of Networked Audio-Visual Systems and Home Platforms.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[2] R. Cohen and G. Nakibly, "A Traffic Engineering Approach for Placement and Selection of Network Services," in *Proceedings of the 26th IEEE International Conference on Computer and Communications, INFOCOM 2007*, 2007, pp. 1793–1801.

[3] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros, "Distributed placement of service facilities in large-scale networks," *CS Department, Boston University, Tech. Rep. BUCS-TR-2006-018*, 2006.

[4] 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," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[5] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines."

[6] C. Perkins *et al.*, "IP mobility support," 1996.

[7] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, L. Mathy, and T. Schooley, "Evaluating xen for router virtualization," in *Computer Communications and Networks, 2007. ICCCN 2007. Proceedings of 16th International Conference on*, 2007, pp. 1256–1261.

[8] Y. Wang, J. van der Merwe, and J. Rexford, "VROOM: Virtual routers on the move," in *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Networking*, 2007.

[9] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: Live router migration as a network-management primitive," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 231–242, 2008.

[10] A. Feldmann, G. Maier, W. Muhlbauer, and Y. Rogoza, "Enabling Seamless Internet Mobility," in *Proceedings of the 16th IEEE Workshop on Local and Metropolitan Area Networks*, 2008, pp. 67–72.

[11] M. Kozuch, M. Satyanarayanan, I. Res, and P. Pittsburgh, "Internet suspend/resume," in *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications*, 2002, pp. 40–46.

[12] M. Satyanarayanan, B. Gilbert, M. Toups, N. Tolia, D. O Hallaron, A. Surie, A. Wolbach, J. Harkes, A. Perrig, D. Farber, *et al.*, "Pervasive personal computing in an internet suspend/resume system," *IEEE Internet Computing*, vol. 11, no. 2, p. 16, 2007.