Expanding Home Services with Advanced Gateways

John Whiteaker, Fabian Schneider, Renata Teixeira, Christophe Diot,
Augustin Soule, Fabio Picconi, Martin May

Expanding Home Services with Advanced Gateways

Jon Whiteaker* jbw@berkeley.edu
Fabian Schneider† fabian@ieee.org
Renata Teixeira‡ renata.teixeira@lip6.fr
Christophe Diot* christophe.diot@technicolor.com
Augustin Soule* augustin.soule@technicolor.com
Fabio Picconi* fabio.picconi@technicolor.com
Martin May* martin.may@technicolor.com
*Technicolor †NEC Laboratories Europe ‡UPMC Sorbonne Universités and CNRS

ABSTRACT

The success of over-the-top (OTT) services reflects users' demand for personalization of digital services at home. ISPs propose fulfilling this demand with a cloud delivery model, which would simplify the management of the service portfolio and bring them additional revenue streams. We argue that this approach has many limitations that can be fixed by turning the home gateway into a flexible execution platform. We define requirements for such a "service-hosting gateway" and build a proof of concept prototype using a virtualized Intel Groveland system-on-a-chip platform. We discuss remaining challenges such as service distribution, security and privacy, management, and home integration.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Distributed networks; D.4.8 [**Performance**]: Reliability, availability, and serviceability

General Terms

Design, Management, Performance

Keywords

Gateway, Virtualization, Sevice, Cloud

1. INTRODUCTION

The fast spread of high-speed broadband access has enabled home users to consume a number of services over the Internet, ranging from legacy communication services such as TV and telephony to video-on-demand (VoD), online gaming, and home automation. At home, Internet access is typically commercialized in the form of a service bundle (e. g., tripleplay bundles IPTV, VoIP, and Internet), or direct access to the Internet (e. g., DSL or Cable). In both cases, users access the Internet and its services via a *home gateway*—which often combines the functionality of a modem, a WiFi access point, and a router—that manages all network connectivity in the home. TVs and other devices without native network support are typically connected through a *set-top box* (STB), which is responsible for transcoding the content so that it can be properly displayed in the end-device. STBs can either be provided by the ISP or acquired off-the-shelf by the user. In the latter case, STBs are qualified as over-the-top (OTT) as they are not managed by the ISP. Media centers sold by Western Digital or Roku, as well as Google TV and Apple TV, are popular OTT boxes.

For reasons of security, digital rights managements, or simply marketing, most STBs are closed platforms which limit the set of services that can be run on them. ISPs generally provide one STB with a small number of services and users have to acquire additional OTT boxes to expand their service portfolio. This is frustrating for the user who has to deal with multiple physical devices, as well as for the ISP who sees a lost opportunity to increase its revenues in these external services. The solution that ISPs have put in place to solve both problems consists of hosting services in the cloud and delivering them through a simple (and cheap) STB. This approach has two main advantages for the ISP: (1) it allows service personalization without any intervention in the home, and (2) it simplifies the management of the services. On the customer side, it simplifies installation, configuration, and utilization of the service portfolio.

We argue that this cloud-based approach has critical limitations. The cloud adds potentially large network delays and creates a dependency on the availability of network connectivity. It does not exploit local storage and caching at home. When data is stored in the cloud (and potentially exploited for profiling and advertisement), people consider that as a threat to their privacy. More importantly, it is unlikely that a single cloud will offer all services a user might want, or that a user will accept contacting her ISP (possibly for a fee) anytime she wants to launch a new service. Only an ISP managed cloud can control the complete network path to the user. Therefore, accessing multiple clouds comes at the risk of poor service quality.

We propose leveraging the home gateway as a proxy to the cloud to address the limitations identified above. We call this gateway a service-hosting gateway (*SHG*). The *SHG* acts as a flexible, always-on, secure execution platform located inside the home. It can be used to improve interaction, cache data, and allow the user to deploy new services locally. These services can be made available to users through an application store model. The *SHG* can also act as a home hub and coordinate devices such as smartphones, tablets, and home sensors, providing a unified interface to the user. Moreover, the gateway lies at the border between the Internet and the home network. This unique position allows the *SHG* to perform advanced network resource management and troubleshooting, increasing the quality of experience for the user.

On the negative side, the *SHG* represents additional up-

front hardware cost and management complexity for the ISP. However, we believe the benefits to the ISP and its customers outweigh the drawbacks. In the following sections, we identify the functional requirements of *SHG*s and discuss enabling technologies. We then evaluate the feasibility of *SHG* using a virtualized Intel Atom-based system-on-a-ship (SoC) platform. Our results show that it is possible to simultaneously run a number of popular media services while forwarding traffic between the home and the Internet. We conclude with a discussion on the research challenges that need to be addressed before *SHG*s can be deployed.

2. REQUIREMENTS

We define six requirements for *SHG*s: *(i)* flexibility to deploy and stop services, *(ii)* sufficient system performance to guarantee an excellent experience to the users, *(iii)* isolation among services and fine-grained resource allocation *(iv)* support for services management and remote troubleshooting, *(v)* transparency for the user; all this at a *(vi)* reasonable cost per gateway.

The *SHG* implements a model where users select the services running at their homes at any time from a variety of services providers. Users have the flexibility to add and remove any service following the model of an application store. In this model, we expect that users pay per service, either through their ISP, or directly with the service provider. In addition, the *SHG* offers the ISP a way to charge the service or application provider to control the service quality experienced by the user. Such control involves the capability to sandbox each service, providing fine-grained, per-service *system performance guarantees* in terms of CPU, memory, I/O, and storage. Competing services should indeed not be able to monitor one another. Providers of copyrighted content today only offer their services on platforms that provide such levels of isolation and security.

The *SHG* adds complexity to current home gateways, while the cloud approach reduces (at least in theory) management in the home to its minimum. The *SHG* will have to instantiate services on demand in a secure fashion. Users may want to run more services than the system can handle. In this case some of the services cannot be admitted. Therefore, the *SHG* must provide top-of-class and transparent management and troubleshooting (e. g., inform the user about the congested resource and allow her to prioritize services). This is probably the main challenge of the proposed model.

The flexibility and revenue opportunities enabled by the *SHG* require more powerful hardware, which will add cost. The *SHGs* must be dimensioned carefully to support all features described above at the minimum price. Note though that most resource consuming tasks (such as processing video streams) can be performed in the cloud, or even on a tablet or any other home device with a general purpose processor. The *SHG* will end up performing tasks such as management, data aggregation, or caching.

We discuss the feasibility of these requirements in the next sections.

3. ENABLING TECHNOLOGIES

This section discusses technologies in operating system (OS) virtualization, hardware design, and management, which can be used to implement crucial components of the *SHG*.

3.1 Operating System Virtualization

Virtualizing the OS on the *SHG* fulfills the majority of the requirements mentioned above.

Service and performance isolation Virtualization prevents buggy services from affecting other services running on the same gateway. While standard process isolation ensures this property, the sand-boxing implemented by virtualization offers stricter guarantees of separation [8], and provides better performance isolation and fine-grained resource allocation. Virtualization also provides a stronger isolation of data stored in a virtual machine.

Faster development cycles Gateways and STBs must be robust and require as little human intervention as possible once deployed in homes. Content owners require the firmware to be validated by external parties, and new software features can take months before they are deployed and reach the customers. The isolation properties of virtualized operating system allow for faster deployment of services, as an individual service failure cannot compromise the overall service bundling performance.

With virtualization and its isolation properties, bugs in a particular service do not compromise the stability of the platform. Thus, software validation is limited to the modules that require it. As a result, service providers can deploy beta versions much faster than with non-virtualized devices, and simply update the software as needed.

Flexible service deployment Virtualization runs each service in a separate virtual machine (VM). Launching a new service is as simple as transferring the VM image to the gateway and starting it. Migrating a user's services to another gateway is also straightforward. Any operation on VMs rebalances the resources controlled by the virtualized OS.

Simplicity of management Offering a complete view of each service in a virtual machine enables simplified monitoring, accountability, and policy enforcement. Furthermore, virtualization allows fine-grained networking policies to be applied to services. Resource reservation is built in to most virtualization solutions [2, 11]. Applying additional rules (e. g., priorities) is straightforward since each service corresponds to a single VM whose resource allocation and usage can be easily monitored and controlled.

Virtualization of home gateways has already been suggested. Royon et al. [10] propose using OSGi to run multiple services on a gateway. Given the limitations of OSGi and the increasing availability of faster CPUs, recently papers suggest full operating system virtualization for embedded systems [7, 9] like we do.

3.2 Hardware Platform

The current platforms for embedded boxes comprise a general purpose CPU, memory and storage. Additional dedicated hardware may be present depending on the features of the box. For example a dedicated network processor may be added to facilitate packet routing, or a hardware video encoder/decoder may support efficient handling of high quality video. The *SHG* mostly requires a general purpose CPU and memory. The presence of video decoding hardware, although not strictly necessary for a *SHG*, is in practice desirable if the box must support multiple video streams.

Performance Most gateways and STBs use a low performance embedded hardware platform. These platforms are optimized to minimize the cost while delivering a specific and predetermined set of services such as IPTV, VoIP, or network connectivity. These platforms are generally so well dimensioned for a specific usage that they cannot be upgraded and need to be replaced when the service bundle changes, or when the type of video supported requires more resources. Therefore, service providers have recently moved toward more advanced System-on-Chip (SoC) platforms. SoCs are flexible by design; they include a fast CPU, significant amount of memory and high speed *I/O* connectivity such as PCI express or SATA.

Using a SoC, it is possible, at a reasonable cost, to design a *SHG* with 2GB of memory, a large SSD or HDD, wireless capabilities, video transcoding, and a DRM converter on a single chip. Every SoC vendor has versions of their chips that support hardware virtualization instructions. The number of concurrent services that can run on a single box is mainly limited by the available RAM.

Cost Popular OTT boxes such as the D-Link Boxee Box[1] or the Logitech Revue[2] Google TV, as well as high-end STBs[3] use a SoC from Intel which is compatible with the x86 instruction set. As this platform offers an acceptable cost/performance ratio for these service providers, we select similare hardware for our *SHG*.

3.3 Management

The complexity of *SHGs* must be balanced by a sophisticated management infrastructure both within the *SHG* and among the devices utilizing the *SHG*.

Service quality Managing local and network resources to ensure quality of experience is a problem that has been widely studied since packet networking was invented (see the plethora of papers on QoS and queuing theory [1]). Many resource management techniques can be reused in this context.

On the home side, we expect to combine resource control mechanisms with protocols such as Universal Plug and Play (UPnP) and other technologies supported by the Digital Living Network Alliance (DLNA) to control the service quality experienced by the users. However, these may require augmentation as many devices do not comply fully to the standards [4].

On the network side, resource management techniques can also be used. Recently, Yiakoumis et al. [14] proposed isolating home services and offering per service bandwidth and latency guarantees using an OpenFlow switch at home. Although they envision cloud-based services controlled solely from the network, we can build upon their network isolation and performance management, move the controller to the gateway, and add the extra capabilities needed to host services.

Simplicity of management The automatic identification and management of connected home devices is essential. UPnP and other DLNA technologies offer some device communication primitives, but they are not always available and can be somewhat limited across devices [4].

Dixon et al. [5] propose a "home operating system" (HomeOS) to control the heterogeneity of home devices. The *SHG* can rely on HomeOS to interact with heterogenous devices including discovering and associating with devices. HomeOS also has mechanisms to specify policies to control access to devices and mediate conflicting accesses. For instance, what happens when two services want to display content on the same TV or control the temperature in the same room? Our future work will study how to integrate HomeOS with *SHGs*.

4. PROOF OF CONCEPT

In this section, we describe a proof of concept implementation of the *SHG*. We do not implement the full set of enabling technologies. Instead, we show the feasibility of the gateway-centric approach we advocate in this work. We assess the CPU and memory consumption footprint of multiple services and standard gateway tasks.

4.1 Test Platform

We based our hardware and virtualization selection on preliminary experimentation, which is documented in a tech report [12]. This tech report *(i)* analyzes the performance of different hardware platforms, ranging from typical home gateways to multicore Atom based PC systems, *(ii)* compares different virtualization solutions including Xen, VMware and VServer, and *(iii)* discusses cost/performance trade-offs.

We choose an Intel media processor platform (a pre-release board with a single core CE4235 processor operating at 1.2 GHz with 2 GBytes of RAM), which is based on low-power Intel Atom processors. This SoC includes specialized hardware for video de/encoding and rendering. We run the Meego[4] Linux operating system, because it already supports the specific hardware components of the SoC. This SoC is similar (in cost and performance) to the ones used in STBs and OTT boxes described in Section 3.2.

There are a number of virtualization techniques available, with different performance overhead and levels of flexibility [13]. For our prototype, we choose LXC[5] because it reaches near-native performance, incurs negligible RAM and CPU overhead, and is part of the main Linux kernel development.

4.2 Use Cases

We want to show that our *SHG* prototype can perform standard gateway functions in parallel to delivering popular home services with good quality. Therefore, we set up the *SHG* to perform NAT and forward background traffic (which we generate with `iperf`).

In addition to standard gateway functionality, we evaluate the *SHG*'s ability to support four representative services. We select these four services/applications because each of them stresses different components of the system: *video streaming and decoding*, as it is a crucial application of home networking; *P2P downloading*, since it is a popular way to obtain media content; *VPN*, because it is a popular application for working remotely; and last, we install a *web server*. P2P and VPN are services typically run on home computers, but moving them to the gateway allows these

[1] `http://www.dlink.com/boxee`
[2] `http://www.logitech.com/smarttv/revue`
[3] `http://newsroom.intel.com/servlet/JiveServlet/download/38-3338/Bouygues_Telecom_Intel.pdf`
[4] `http://www.meego.com`
[5] `http://lxc.sourceforge.net/`

Table 1: CPU usage and quality of service use cases

Service use case	virtualized		Service Quality
	no	yes	
Forwarding traffic (`iperf`)			
20 Mbps UDP	0.5 %	—	no loss
100 Mbps UDP	2 %	—	no loss
850 Mbps TCP	10 %	—	no loss
VoD: Decoding a video stream			
w/ HW support	25 %	30 %	clear video
w/o HW support	no video is displayed		
P2P: File-sharing with BitTorrent			
10 Mbps	8 %	8 %	
20 Mbps	14 %	14 %	
VPN: Crypto throughput (`ssh -L`)			
10 Mbps	21 %	22 %	no loss
20 Mbps	33 %	33 %	no loss
80 Mbps (max)	102 %	102 %	<2 % loss
Web server benchmark (`ab`)			
1 conc. request	18 %	18 %	460 req/s
1000 conc. requests	68 %	73 %	1100 req/s

services to run persistently in the background and eliminates the need to keep the PC on or re-authenticate.

We run each service inside its own virtual machine as well as without virtualization. By comparing the two we can determine the impact of virtualization on service performance. We first run each service individually, and then measure the performance of the *SHG* when all services are run simultaneously.

4.3 Performance discussion

We run the use cases described above on our test platform while measuring the CPU and memory utilization. Furthermore we look at the time it takes to start a virtual machine (VM), and we discuss storage requirements.

CPU utilization and service quality Table 1 shows the CPU utilization for non-virtualized and virtualized executions. Since our platform supports hyper-threading, the total reported CPU utilization can exceed 100 %. In addition to CPU usage, we report the service quality in terms of packet losses, video quality, or requests served per second depending on the use case.

We look first at the overhead of virtualization. We only observe a noticeable difference in CPU utilization for I/O intensive services; both video decoding and requesting 1000 objects concurrently from the web server cause a large number of interactions between the processor and the graphics/network card. For all other use cases the overhead is less than 1 %. Running multiple services in parallel neither increases CPU consumption of each service, nor causes an increase in the total CPU utilization due to VM scheduling.

Turning to the CPU usage, forwarding traffic consumes minimal CPU resources, even for line-rate TCP connections. The other services consume 10–22 % for a network throughput of 10 Mbps, and 15–33 % for 20 Mbps. This shows that our test platform can easily handle service load at typical broadband speeds. In our experiments encryption is the most CPU intensive task, and limits throughput to 80 Mbps using all available CPU resources.

Thanks to its dedicated hardware, our prototype can decode and display an HD video stream with very good quality, using only around 30 % of the CPU. Intel specs state that the SoC supports two HD or five SD streams concurrently, which is in line with our findings. GPU assistance is valuable here, as without it the platform would not be able to decode video and perform additional CPU-consuming tasks.

These measurements suggests that our platform can support the services that a large household (5-6 people) would run in parallel with excellent quality of experience. While monitoring and management tasks may have a slight impact on CPU utilization, we believe this should be small and should not be noticeable on the overall performance of the system.

Resource requirements In terms of memory, none of the use cases perceptibly add to the baseline memory utilization. Moreover, instantiating a new VM (i. e., Linux container) does not cause additional memory to be allocated by the OS (other than that reserved by the application). This shows that the memory overhead of container virtualization is negligible.

In our experiments, starting a virtual machine takes less than a second. We also test instantiating VMs that are stored on a network share (NFS) and do not observe any perceptible overhead. These low-overhead results are specific to LXC.

Other virtualization techniques (such as Xen HVM and KVM) can support multiple guest OSes, but at the cost of increased memory and CPU overhead. Nested virtualization limits the memory overhead, but still incurs significant overhead for I/O intensive tasks [3]. The study of such alternative virtualization techniques is left for future work.

We also find that our prototype design is able to host multiple services when run in parallel. All services get a fair share of the CPU and provide acceptable performance. However, there may be drops in service quality due to insufficient resources, particularly when too many services are running in parallel. In this case the user can decide which services to prioritize. We refer the reader to our tech report [12] for a detailed performance analysis when running multiple services in parallel.

5. REMAINING CHALLENGES

Before *SHGs* can be commercialized, we need to address a number of research and implementation challenges.

On demand service instantiation In our prototype we deploy services manually. For a production *SHG*, an automated solution is necessary. The Apple App-Store and the Android Market can serve as a model for the *SHG*. These solutions act as a searchable central repository for applications, where those selected for installation are pushed to the device.

Security model When users activate and deactivate services, software developed by different entities will be installed and un-installed on the gateway. This dramatically changes the security requirements compared to today's model, where the software running on the gateway is static and developed by a single entity. In particular, competing services want to ensure that other services are not able to

spy on them. Virtualization provides some isolation here, but services may desire stronger protection (e.g., trusted computing).

Virtualization of hardware While the CPU is partitioned in virtualization, not all hardware and software supports secure virtualized access. For example, to enable efficient virtualization of the GPU in our prototype, the isolation of inter-process communication (IPC) for guests in LXC had to be broken. This limitation comes from the particular video driver that assumes full and privileged access to the memory. A new version of the driver, compatible with virtualization, is currently being developed. This new driver also introduces the concept of virtual screen that allows a "screen" scheduler to decide when and where an application should be displayed on the screen.

Additionally, we observed that forwarding traffic in the host domain at rates close to the link capacity starves competing guest domain traffic. For a large-scale deployment, this issue needs to be addressed either in software in the host domain, or by performing forwarding in a guest domain, which may add latency.

Finally, the current version of the virtualization instruction set does not include the instructions for directed I/O such as VT-d[6]. This component is essential to offer fast and secure I/O transfers between peripherals controlled by different VM.

Until all specialized hardware is properly virtualized and scheduled, a CPU scheduler based on aggregate resource consumption, such as Gupta et al.'s scheduler for Xen [6], would help share hardware resources in a more fair way.

Management of heterogeneous home devices As the number of connected devices in the home increases, new technologies for controlling and managing these devices need to be designed. Ideally, the *SHG* needs to know the specifications of each connected device, and which devices interact with which services. A more comprehensive, higher-level protocol would provide the *SHG* with more control over the home network as well as better troubleshooting capabilities. Consequently, sources of performance degradation could be pinpointed and acted upon by the *SHG* to improve the situation. We believe that UPnP and HomeOS can be leveraged in this context.

User privacy The *SHG* is a point for aggregation and management of user data in the home. As sensors in the home become more common (e. g., smart power meters, personal health monitoring), the *SHG* can store and control access to personal data collected from these sensors. This same model could be applied to other types of personal data or user credentials, e. g., for information stored on social networks today or user profiles that inform targeted advertisement. In this context, protecting users privacy becomes an important priority. In that sense the *SHG* offers an opportunity to better preserve and control user privacy at home. In particular, users can control access to sensitive information at a single, in-home location potentially offering better privacy protection than when personal data is scattered over multiple cloud services. We acknowledge, however, that additional research is necessary to develop appropriate mechanisms to protect user's privacy and give users more control over their own data.

[6]`http://www.intel.com/technology/itj/2006/v10i3/2-io/7-conclusion.htm`

Dimensioning gateway hardware The fact that STB and media center manufacturers already plan on using similar hardware for high-end devices endorses our claim that the cost for such hardware is reasonable. To further reduce costs, the gateway architecture could be modular, extending its functionalities through the addition of hardware modules.

Support for cloud-based services

Cloud-based applications and services have become commonplace. Cloud computing allows service providers to run application logic on huge datacenters, where CPU, memory and storage resources are orders of magnitude larger than those present in mobile devices or in the home network. The downside is that devices must exchange significant traffic with servers located in remote datacenters to leverage cloud-based services. This adds network latencies and bandwidth limitations compared to applications that operate locally.

The *SHG* can be especially useful for cloud-based services. Such services can maintain their datacenter-based architecture, but add a local *Cloud Proxy* as service running on the *SHG*. Such a service can be application-specific (to implement the specific protocols and logic of the application), and perform tasks such as content buffering, caching and prefetching, video transcoding, replicating part of a remote database, and many more.

Consider the case where a user of a cloud-based email service (e.g., Gmail) wants to send an email with a very large attachment. The send operation might take a long time, especially if the user has an ADSL connection with a slow uplink. Gmail could provide a helper service located on the *SHG*. When the user sends a bulky email, the browser quickly uploads the attachment to the gateway (through the user's high-speed LAN). The user can then turn off his computer or leave the house, while the gateway continues uploading the email to Gmail's servers in the background.

We believe that Cloud Proxy services can greatly improve the quality of experience of users of cloud-based applications, and are therefore a significant incentive for ISPs to deploy *SHGs* that support them.

6. CONCLUSION

We propose to place intelligence at the home gateway in order to help deliver high quality services and offer complete freedom to configure a home digital service portfolio. The *SHG* is a natural way to allow user generated content and services to be delivered to the home, while offering a monetization opportunity to the ISP, and an increase in Quality-of-Experience for service providers that currently rely on over-the-top (OTT) solutions. However, adding such complexity to the home gateway will make it more difficult to manage and more likely to fail. Therefore, it is essential to make *SHGs* robust and reliable. We are currently addressing these and the other challenges identified in the previous section in a more complete *SHG* prototype (using the same hardware) that will offer management capabilities and improved virtualization.

Acknowledgements

This work was supported by the European Community's Seventh Framework Programme (FP7/2007-2013) no. 223850 (Nano Data Centers) and no. 258378 (FIGARO).

7. REFERENCES

[1] Aurrecoechea, C., Campbell, A. T., and Hauw, L. A survey of qos architectures. *Multimedia Systems 6* (1998), 138–151.

[2] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles* (2003), pp. 164–177.

[3] Ben-Yehuda, M., Day, M. D., Dubitzky, Z., Factor, M., Har'El, N., Gordon, A., Ligouri, A., and Wasseman, O. The turtles project, design and implementation of nested virtualization. In *Proceedings of Operating Systems Design and Implementation* (2010).

[4] DiCioccio, L., Teixeira, R., May, M., and Kreibich, C. Probe and pray: Using upnp for home network measurements. In *Proceedings of the 13th International Conference on Passive and Active Measurement* (2012), PAM '12, pp. 96–105.

[5] Dixon, C., Mahajan, R., Agarwal, S., Brush, A., Lee, B., Saroiu, S., and Bahl, V. An operating system for the home. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation* (2012), NSDI '12.

[6] Gupta, D., Cherkasova, L., Gardner, R., and Vahdat, A. Enforcing performance isolation across virtual machines in xen. In *Middleware '06: Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware* (2006), pp. 342–362.

[7] Ibanez, M., Madrid, N., and Seepold, R. Security management with virtual gateway platforms. In *Proceedings of the third Interernationan Conference on Emerging Security Information, Systems and Technologies (SECURWARE'09)* (June 2009), pp. 70–75.

[8] Kroeker, K. L. The evolution of virtualization. *Commun. ACM 52* (March 2009), 18–20.

[9] Laoutaris, N., Rodriguez, P., and Massoulie, L. ECHOS: Edge capacity hosting overlays of nano data centers. *ACM SIGCOMM Computer Communication Review* (Jan. 2008).

[10] Royon, Y., and Frenot, S. Multiservice home gateways: business model, execution environment, management infrastructure. *Communications Magazine, IEEE 45*, 10 (Oct. 2007), 122–128.

[11] Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. *SIGOPS Oper. Syst. Rev. 41*, 3 (2007), 275–287.

[12] Whiteaker, J., Schneider, F., Soule, A., Teixeira, R., May, M., Picconi, F., and Diot, C. Designing a service-hosting gateway for the home. Tech. Rep. CR-PRL-CR-PRL-2012-08-0001, Technicolor Technical Report, Aug. 2012.

[13] Whiteaker, J., Schneider, F., and Teixeira, R. Explaining packet delays under virtualization. *SIGCOMM Comput. Commun. Rev. 41*, 1 (Jan. 2011), 38–44.

[14] Yiakoumis, Y., Yap, K.-K., Katti, S., Parulkar, G., and McKeown, N. Slicing home networks. In *Proceedings of the 2011 ACM SIGCOMM workshop on Home networks* (2010), HomeNets '11.