Rethinking Access Networks with High Performance Virtual Software BRASes

Roberto Bifulco, Thomas Dietz, Felipe Huici, Mohamed Ahmed
Joao Martins, Saverio Niccolini, Hans-Joerg Kolbe
NEC Laboratories Europe
{firstname.lastname}@neclab.eu

***Abstract*—Broadband Remote Access Servers (BRASes) play a crucial role in today's networks, handling all traffic coming from access networks (e.g., DSL traffic), applying operator policies and providing the first IP point in the network. It is perhaps unsurprising then, that these are expensive, proprietary, difficult-to-upgrade boxes. They also represent a large, single point of failure, making operators even more reticent to deploy new functionality for fear it might seriously disrupt day-to-day operations.**

In order to remove some of these barriers to innovation, we advocate for turning BRASes from the monolithic hardware boxes they are today into flexible, virtualized, software-based devices running on inexpensive commodity hardware. As a proof-of-concept, we present the implementation and performance of a software BRAS based on ClickOS, a tiny Xen virtual machine designed specifically for network processing. Our software BRAS prototype can establish subscriber sessions at rates above 1,000 per second; requires only 1MB of memory per 1,000 established sessions; can boot in milliseconds; and can handle traffic at 10Gb/s for almost all packet sizes.

I. Introduction

One of the most crucial but lesser known components of today's networks is the Broadband Remote Access Server (BRAS), also referred to as a Broadband Network Gateway (BNG) [10]. BRASes are important: all access network traffic (e.g., DSL traffic) goes through them and they are in charge of a wide range of functionality critical to network operators. Among others, they take care of authenticating users, performing accounting and monitoring, handling sessions and tunnels, and shaping bandwidth, not to mention acting as the first IP point in the network.

Given their position in the network and all of the functionality they contain, it comes perhaps as no surprise that they are expensive, hardware-based proprietary boxes. Besides its price tag, a BRAS comes with a number of limitations. First, it presents a massive single-point of failure: one BRAS usually handles thousands of subscribers, something that can equate to a large number of customer support phone calls (and their related costs) should things go awry. This means that operators are loath to introduce new functionality or services in BRASes, lest they cause serious disruption to their operational networks.

Second, the fact that all session tunnels (e.g., PPPoE) are terminated at the BRAS means that any IP-based services such as multicast IPTV or access to CDN caches must sit behind the BRAS. This prevents operators from being able to place such services closer to end users (e.g., in the aggregation network), causing inefficiencies in the network and poorer service for users.

Finally, because they are hardware-based boxes, upgrading their functionality is less than trivial, often requiring a long wait, often years, until the next version is available; this prevents operators from deploying new customer services in a timely fashion. Also, BRASes are far from modular: operators must pay for all of the functionality even though they may only be interested in deploying some of it (e.g., PPPoE tunneling but not IGMP support).

Ideally, in order to tackle these problems, we would like to turn these rigid, monolithic hardware devices into software-based network functionality that can be easily decomposed and dynamically instantiated at different points in the network. This concept follows the existing trend captured by the NFV approach [9] which seeks to turn hardware devices like middleboxes (e.g., NATs, DPIs, proxies, firewalls) into virtualized software-based processing running on commodity hardware (i.e., inexpensive, off-the-rack x86 servers).

To the best of our knowledge, however, no-one in the research community has yet tackled the task of developing high performance virtualized software BRASes. In this paper we present a proof-of-concept implementation and performance evaluation of such a BRAS. Our software BRAS is built on top of ClickOS, a tiny Xen virtual machine (VM) based on the Click modular router software [6] and aimed at network processing. ClickOS VMs can be instantiated in tens of milliseconds, have a small memory footprint of 5MB when running and can process packets at 10Gb/s. In addition, we show that our software BRAS can establish PPPoE sessions at rates of 1000/s (the maximum that the IXIA tester we used could offer) while using a small amount of memory for the session state and the VM itself (about 20MB).

This software BRAS provides a number of advantages. Because it is software based, it is simple to upgrade, and Click's modularity means that adding functionality to it is also a quick affair (we will give an example of this in section III). This modularity, and that afforded by the tiny ClickOS VMs, allow for easy decomposition of the functionality contained in a traditional BRAS. As a result, we could envision deploying some functionality closer to end users, for instance to (1) break off tunnels at aggregation networks in order to provide

978-1-4799-2433-2/13 $31.00 © 2013 IEEE
DOI 10.1109/EWSDN.2013.8

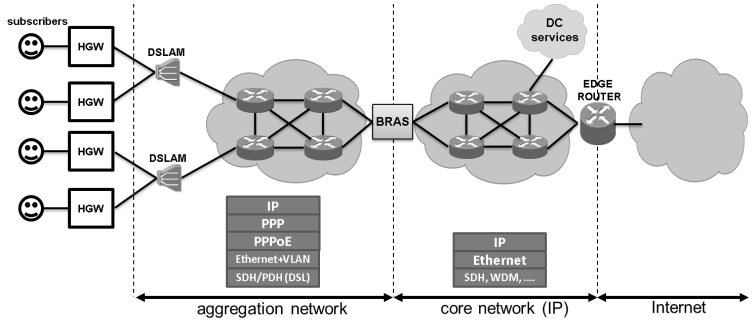

Fig. 1: Broadband access network architecture along with related protocol stacks.

services such as access to CDNs more efficiently; to (2) reduce tunnel overheads by allowing home gateways to send IP traffic without tunnels while introducing the necessary ACLs to control such traffic; or to (3) leverage the availability of large numbers of home gateways, many of which contain general-purpose processors, in order to scale processing.

In addition, the fact that the software BRAS is virtualized means that operators can test out new features on a small subset of subscribers without risking adverse effects on subscribers being handled by other VMs, perhaps removing a barrier to innovation. Finally, the software BRAS is relatively inexpensive: the server we ran our tests on costs in the region of $2,000.

The rest of the paper is organized as follows. In section II we introduce background information regarding access networks, protocols and BRASes that will help understand the details of our implementation. Section III describes our implementation and presents a performance evaluation of the software BRAS. Finally, section IV discusses, in greater detail, the scenarios that are enabled by a virtualized software BRAS and section VI concludes.

II. Background

In this section we give a brief background about broadband access networks and BRASes; this will make it easier to understand the implementation of our software BRAS in the coming section.

A. Broadband Access Networks

The most common technologies used by network operators in the last mile of the network are Digital Subscriber Line (DSL) and cable, with DSL being the most deployed technology, especially in Europe [11]. In a typical set-up, users' home gateways (e.g., xDSL routers) are aggregated at a Digital Subscriber Line Access Multiplexer (DSLAM). From there, the connections going through several DSLAMs are themselves combined using an aggregation network whose end point is the BRAS (see figure 1). The BRAS acts as the first IP point in the network, and connects to the operator's core IP network, which may itself be directly connected to data centers to provide localized services to its customers. Finally, one or more edge routers provide connectivity to the Internet.

A network operator provides connectivity to its subscribers by binding the subscriber to the concept of a session: the subscriber has to establish a session to get access to the network and its services. The session establishment usually involves procedures for the authentication of the subscriber and the application of network policies (e.g., bandwidth limits and quotas), and is initiated by the home gateway (HGW) and terminated at the BRAS. The most common protocol used for this purpose is the Point-to-Point Protocol (PPP), together with PPPoE, the variant used to work over Ethernet [10]. This effectively creates a tunnel where all the network packets sent or received by the HGW must be encapsulated in PPPoE/PPP headers and decapsulated by the BRAS. Figure 1 shows the relevant protocols stacks at each point of the access network.

B. Broadband Remote Access Server

As previously mentioned, the BRAS carries out a wide range of tasks. These include, but are not limited to:

- PPP/PPPoE session management and header decapsulation.
- Interface to AAA (Authorization, Authentication and Accounting) services.
- Traffic shaping, generally in the form of a hierarchical shaping policy that creates per subscriber and per service classes.
- Access control based on access control lists (ACLs), making sure that packets belong to a subscriber for whom a session has been already established.
- ARP proxy, replies to ARP requests coming from interfaces on the BRAS' IP side.
- IP forwarding to routers in the operator's core network.
- Assignment of IP addresses to subscribers.
- IGMP processing in order to support multicast.

In the next section we will describe our software BRAS implementation, including how its modularity allows us to decompose the large list of functionality contained within a traditional BRAS.

III. Proof-of-Concept

We will now describe our proof-of-concept software BRAS implementation. We begin by giving a small introduction to

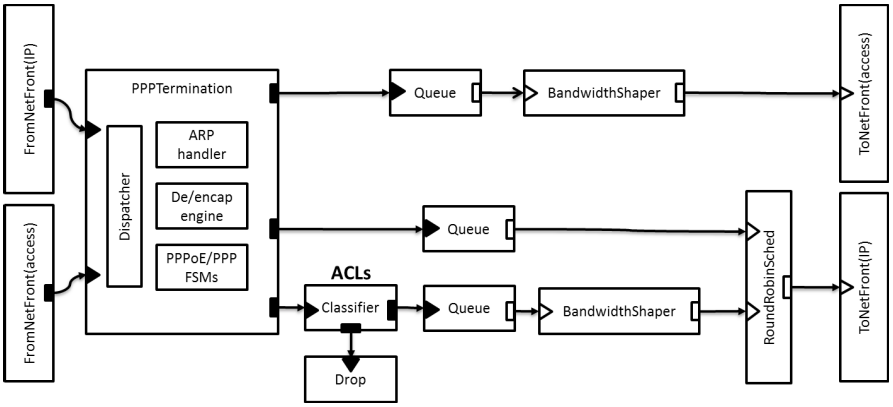

Fig. 2: Software BRAS design.

the Click modular software. We then discuss ClickOS, the virtualization system that our software BRAS is based on, followed by an explanation of the BRAS itself. We end the section by presenting results from a performance evaluation of it.

A. Mini Click Primer

Click is based around the concept of modules called *elements*, each performing a small amount of network-related packet processing such as decreasing the TTL or re-writing packet headers. An element also has read or write *handlers* that let users inspect or change the state of the element (e.g., the AverageCounter element has a read handler to get the number of packets seen so far and a write handler to reset that count). Elements can be connected to other elements, resulting in a directed graph called a *configuration*. Thanks to its modularity and the fact that it comes with hundreds of elements, it is quite simple to use Click to come up with configurations representing IP routers, NATs, firewalls, proxies, and many other kind of network devices. Click is also extensible, making it easy to connect existing elements to new, custom-designed ones; this is the case for our BRAS implementation described further below.

B. ClickOS

ClickOS consists of tiny, Xen-based virtual machines [1] (also known as guests or domUs). Xen has the concept of *paravirtualized* guests, whereby the operating systems of the VMs are slightly modified to hook into the Xen hypervisor and I/O subsystem; this is done in order to boost the performance of the guests. While mainstream operating systems such as Linux or FreeBSD have been paravirtualized and can thus run as guests in Xen, in this work we are only interested in doing *network* processing, for which a full fledged OS is overkill. What we would like is to have a paravirtualized, minimalistic OS to which we could add the necessary network functionality, and the Xen sources come with exactly the right thing: MiniOS.

MiniOS provides the basic facilities to run as a Xen guest with few overheads. For example, it has a very simple co-operative scheduler, so it does not incur context switch costs. In addition, it has a single address space, so there are no expensive system calls to speak of [1]. ClickOS, then, consists of combining MiniOS with the Click modular router software into a single Xen virtual machine.

What about networking? Xen implements a so-called split driver model, where a back-end driver called *netback* communicates with the NIC's device driver (usually via a software switch such as Open vSwitch [13]), and a front-end driver called *netfront* runs in the guest OS and talks to the netback driver using a ring API; one of the steps in paravirtualizing an OS is providing such a netfront driver. Xen networking does come with performance issues, so we had to heavily modify chunks of this subsystem in order to improve the performance of the software BRAS; details of these changes can be found in [7]. Finally, we added code to Click so that (1) it can interact with a netfront driver as opposed to a device driver and (2) it can run in a MiniOS environment which, among other things, does not provide a console to perform actions like installing Click configurations.

C. Software BRAS Design

The design of our software BRAS consists of a Click configuration centered around a new element we implemented called *PPPTermination* (see figure 2). The BRAS has two main interfaces, one facing the access network and another one connected to the IP network (four elements in total, one receive and one send element per interface).

Packets arriving from the access network go directly to the *PPPTermination* element which takes care of establishing sessions and handling tunnel decapsulation (more on this later). Packets for which a session exists are decapsulated and sent out of the *PPPTermination* element. From there they go on to a classifier that implements access control lists (ACLs) followed by a bandwidth shaper and finally out to the IP network.

On the reverse path, that is, packets coming from the IP network, the *PPPTermination* element handles ARP queries

[1]Note that these do not make ClickOS less secure: in general, the model is that each ClickOS VM is owned by a single entity, and entities are isolated by the fact that they use different VMs.

and replies. In addition, it takes care of forwarding IP packets onto subscribers by encapsulating them in PPP/PPPoE tunnels. From there packets go once again through a bandwidth shaper and then to the access network.

The *PPPTermination* element deserves a closer look (figure 3). The element implements the PPPoE and PPP protocols (RFC2516, RFC1661, RFC1332, RFC1877). Both the PPPoE and PPP RFCs define a Finite State Machine (FSM) whose transitions depend on events generated by both network packets and timers. The FSMs are mainly involved during the set-up and tear down phases of a subscriber session and use a common database to store session information. The same database is then used by the ARP handler and by the encap/decap component.

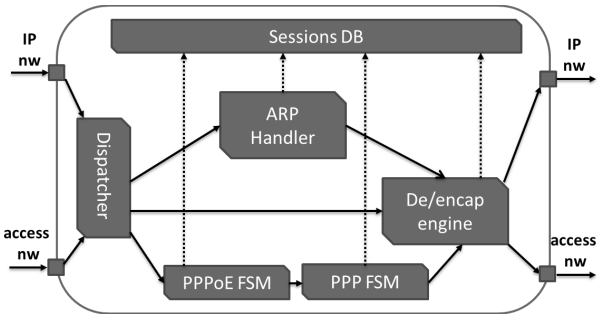

Fig. 3: PPPTermination element design.

In greater detail, packets coming into the *PPPTermination* element first hit a dispatcher which is in charge of splitting ARP packets from IP ones (for packets coming from the IP network) and packets from established sessions from those seeking to establish sessions (for packets coming from the access network). For established sessions, the dispatcher simply checks whether a session ID exists and if so sends them onto the encap/decap engine which, in turn, verifies that the ID is valid, ends the PPP/PPPoE tunnel, and sends the packet out onto the IP network (this is the BRAS' fast path). Packets which do not have a session ID are sent to the FSMs in order to establish a session.

D. Evaluation

This paragraph presents a performance evaluation of our software BRAS and the ClickOS system it is based on. The BRAS ran on an HP DL380G7 server equipped with two Intel L5640 (6-Core 2.26GHz) processors, 24GB of RDIMM PC3-10600R-9 memory and an Intel 10Gb card (82599 chipset). For traffic generation we used a similar server, and for PPPoE/PPP session establishment IXIA and Spirent testers [2]; all of these were connected to the software BRAS with a direct cable.

In addition, the Xen hypervisor was configured to assign four cores to the driver domain that hosts the device and netback drivers, while the remaining cores are used by the running ClickOS instances. Moreover, the driver domain is configured to use 1024 MB of system memory, with memory ballooning disabled.

[2]We used two different testers to ensure that our prototype was standards-compliant.

Instantiation Times On top of the modularity provided by the software BRAS, it would be ideal if we could instantiate its functionality in different parts of the network dynamically and as quickly as possible to adapt to changing network loads, to deploy new services, or to apply new policies to subsets of subscribers.

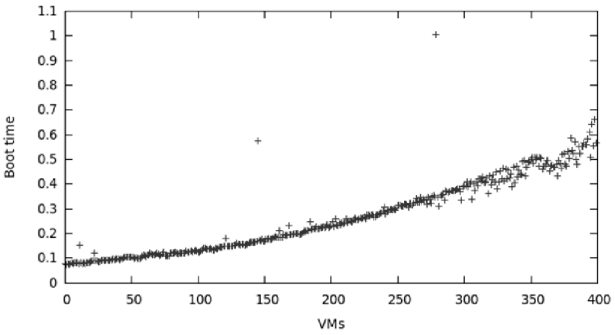

Fig. 4: ClickOS boot times, in milliseconds.

As it turns out, the BRAS, running on top of ClickOS, can be booted extremely quickly, in as little as 70 milliseconds (figure 4). The figure also shows that a large number of ClickOS virtual machines can be quickly booted on the same, inexpensive commodity server. Note that this does not mean that we would be able to run hundreds of concurrent software BRAS instances while handling traffic, but gives an idea of how long it would take to instantiate BRAS functionality on a server.

Session Establishment To measure the session establishment rate (in sessions per second), we used a single ClickOS virtual machine running the software BRAS configuration. We performed several tests, each time changing the number of sessions generated per second up to the maximum of 1,000 that the IXIA tester could handle. All the tests are performed without using external AAA services in order to focus only on the performance of the BRAS.

The results, plotted in figure 5, show that our prototype is able to achieve a session establishment rate almost equal to the session generation rate. Note that typical commercial BRASes are able to establish about 300 PPP sessions per second on average (albeit while handling data plane traffic, which we did not do in the test above). Further, we performed the same tests using up to 26 concurrent software BRAS virtual machines and obtained the same results.

One of the big advantages of using tiny ClickOS virtual machines is the small memory requirement: a compressed software BRAS virtual machine is less than 2MB in size. We performed a test to evaluate the amount of memory needed when establishing sessions. The results (figure 6) show that the memory consumption increases linearly with the number of sessions. More interestingly, only about 1MB of memory

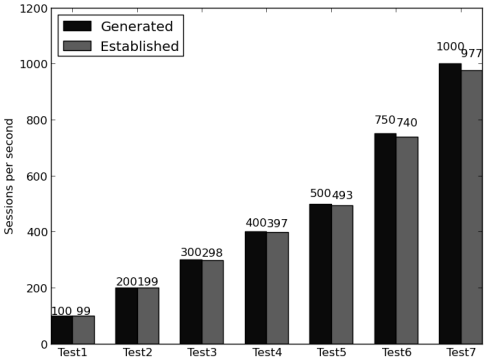

Fig. 5: Software BRAS session establishment rate.

is needed per 1,000 established sessions, with just 8MB extra for the virtual machine itself. According to this test, a single BRAS instance needs less then 100MB of memory to accommodate the maximum number of sessions supported by the PPPoE protocol (65,536).

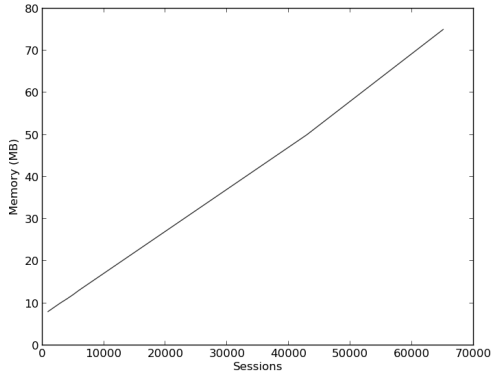

Fig. 6: Software BRAS memory consumption when establishing sessions.

Throughput So far we presented results for boot times and session establishment, but what about the performance when handling large quantities of traffic? To evaluate this, we instantiate an increasing number of concurrent ClickOS virtual machines on the same server, instructing each to send out large rates of traffic for different packet sizes. All packets then converge on a single 10Gb port, and we measure the *cumulative* rate at a separate box connected via a direct cable.

We plot the results in figure 7. The flat lines mean that the system is able to sustain the same rate for each packet size despite more and more virtual machines coming online. It is also worth pointing out that ClickOS can fill up a 10Gb pipe for almost all packet sizes. For loads with only 64-byte packets (an aberrant case that never happens in operational networks) it is still able to produce over 4.5 million packets per second. It is worth pointing out that the configuration used for these tests was a simple one rather than the software BRAS one. We are now in the process of performing similar tests when

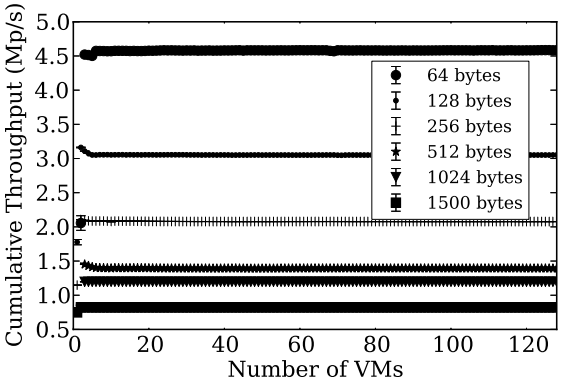

Fig. 7: ClickOS throughput performance while concurrently running different numbers of virtual machines.

running the BRAS configuration.

IV. Discussion

The early experimental results from our prototype are encouraging and make us optimistic about the applicability of a virtualized software BRAS in future broadband access networks. Our software BRAS solves many of the issues that current BRASes suffer from: it can be deployed on inexpensive general-purpose servers, it is non-proprietary, it allows for easy upgrades, and the fact that it is modular means that operators can pick and choose which functionality to actually deploy (and pay for).

In addition to these benefits, we believe that software BRASes like our prototype can enable a number of interesting scenarios that are difficult or impossible to achieve in current access networks. First, our BRAS allows for fast deployment of new functions by making it easier to conduct field tests: a pilot trial can be performed on a small subset of subscribers before deploying the solution on a larger scale, or rolled back to the operational implementation in case things fail. What's more, such a trial could be conducted on the same commodity server that is handling operational traffic without fear of disruption, since the trial would be running on a separate, isolated virtual machine.

Second, the ability to deploy different BRAS instances allows for the creation of targeted differentiated services aimed at users with different needs. For example, we could imagine a BRAS instance which supports virtual private networks for enterprise customers and another BRAS without such a function used for home subscribers, both running on the same inexpensive server. Indeed, the fact that our prototype is able to potentially run hundreds of instances on a single server means that we can create fine-granularity classes of subscribers, each with different policies.

Finally, the decomposition of the BRAS into its elementary functions, coupled with ClickOS' features like fast boot times and small memory requirements, allow for dynamic deployment of BRAS functionality on-demand and at different points in the access network, enabling further scenarios. For instance, the decapsulation/encapsulation function could be

moved closer to the user to allow early tunnel termination. In this way, operators could give access to cloud and CDN services deployed in the aggregation network, a gain in efficiency compared to today's traffic which has to traverse the aggregation network, go through the BRAS, and only then access the needed services.

In the extreme, we could envision putting some of the BRAS functionality in home gateways. Their large numbers could be leveraged to scale the performance of the network further, and since many of them also tend to use general-purpose processors (often ARM ones), it would be possible to extend our prototype to them. For example, we could have a software BRAS instance handling session management install ACLs directly on home gateways, removing the need to carry traffic in PPP/PPPoE tunnels and thus getting rid of overheads in the access network.

V. Related work

There is growing interest both in the research community and industry in turning traditional network processing as embodied by hardware devices like middleboxes into software-based functions running on cheap, commodity hardware. The authors in [8], for instance, advocate for turning middleboxes into software functions and deploying them as a service in the cloud in order to reduce management complexity and costs. Earlier research looked into scaling software router performance by constructing a single router out of several commodity servers [3]. In contrast, the work presented here does so in a virtualized environment, is able to fill up 10Gb pipes with a single inexpensive box, and focuses on implementing a BRAS instead of a router. The authors in [4] looked into measuring the performance of virtualized software routers, though that work was aimed at 1Gb interfaces and routing only.

On the industry side, a large number of major operators from around the world have recently formed the ETSI Network Functions Virtualization Industry Specification Group in order to define and try to solve the major challenges in turning hardware devices into software-based functions [5]. Regarding vendors, Cisco sells a virtualized router than can run on VMware ESXi or Citrix XenServer [2]. Vyatta [12] provides open-source software that implements middlebox functionality and can run on different virtualization platforms. Our work focuses on BRASes rather than middleboxes, uses tiny virtual machines that can boot quickly and have a small memory footprint, and provides a high-performance virtualized environment.

VI. Conclusions and Future Work

To the best of our knowledge, we have presented the first implementation and performance evaluation of a software BRAS in the literature. We have shown that our prototype can handle large session establishment rates and can instantiate BRAS functionality very quickly. These properties, along with the prototype's modularity, make it much easier for BRASes to be updated and new features tried out without fear of disrupting service to the majority of subscribers. In addition, we discussed a number of scenarios that our software BRAS enables, thanks to the fact that it can be easily decomposed and dynamically instantiated in different parts of the network.

As future work we are planning to test the performance of the ClickOS virtual machine's packet processing rate while running our software BRAS configuration. Further, we are working on mechanisms to be able to easily migrate session state between BRAS instances for reliability reasons or to to be able to dynamically scale out processing from an overloaded software BRAS. Finally, while the software BRAS configuration is modular, we are planning on doing performance tests where the BRAS functionality is carried out on different servers in order to improve scalability (e.g., one server handles session establishment, another one ACLs, and yet another one traffic shaping).

VII. Acknowledgements

The research leading to these results has received funding from the European Union's Seventh Framework Programme FP7/2007-2013 under the Trilogy 2 project, grant agreement 317756. We would also like to thank Xantaro, and in particular Carsten Michel, for their time and for letting us use their IXIA equipment.

References

[1] 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 *Proceedings of the nineteenth ACM symposium on Operating systems principles* (New York, NY, USA, 2003), SOSP '03, ACM, pp. 164–177.

[2] Cisco. Cisco Cloud Services Router 1000v Data Sheet. http://www.cisco.com/en/US/prod/collateral/routers/ps12558/ps12559/data_sheet_c78-705395.html, July 2012.

[3] Dobrescu, M., Egi, N., Argyraki, K., Chun, B.-G., Fall, K., Iannaccone, G., Knies, A., Manesh, M., and Ratnasamy, S. Routebricks: exploiting parallelism to scale software routers. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles* (New York, NY, USA, 2009), SOSP '09, ACM, pp. 15–28.

[4] Egi, N., Greenhalgh, A., Handley, M., Hoerdt, M., Huici, F., and Mathy, L. Towards high performance virtual routers on commodity hardware. In *Proceedings of the 2008 ACM CoNEXT Conference* (New York, NY, USA, 2008), CoNEXT '08, ACM, pp. 20:1–20:12.

[5] Institute, E. T. S. Industry specification group.

[6] Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. The click modular router. *ACM Trans. Comput. Syst. 18*, 3 (Aug. 2000), 263–297.

[7] Martins, J., Ahmed, M., Raiciu, C., and Huici, F. Enabling fast, dynamic network processing with clickos. *To appear in SIGCOMM HotSDN* (2013).

[8] Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., and Sekar, V. Making middleboxes someone else's problem: network processing as a cloud service. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication* (New York, NY, USA, 2012), SIGCOMM '12, ACM, pp. 13–24.

[9] AT&T, BT, CenturyLink, China Mobile, Colt, Deutusche Telekom, KDDI, NTT, Orange, Telefom Italia, Telefonica, Telstra, and Verizon. Network function virtualization - white paper.

[10] Broadband forum. Tr101v2 migration to ethernet-based dsl aggregation - issue 2, 2011.

[11] OECD. Broadband portal http://dx.doi.org/10.1787/888932398138.

[12] Vyatta. http://www.vyatta.com/.

[13] vSwitch, O. Production quality, multilayer open virtual switch.