The Routerlab - Emulating Internet Characteristics in a Room

Amir Mehmood, Oliver Hohlfeld, Dan Levin, Andreas Wundsam, Florin Ciucu, Fabian Schneider, Anja Feldmann
Deutsche Telekom Laboratories, Technische Universität Berlin
{amir|oliver|dlevin|andi|florin|fabian|anja}@net.t-labs.tu-berlin.de

Ralf-Peter Braun
Deutsche Telekom Laboratories
{ralf-peter.braun}@telekom.de

Abstract

Designing infrastructures for automatically conducting controlled, reproducible Internet experiments poses substantial challenges. These include modelling the Internet structure in a useful way, and providing sufficciently Internet-like traffic.
In this paper we describe a multi-purpose experimental platform, called Routerlab, enableing complex Internet measurement experimentation. Departing from the approach taken by many popular testbeds, the Routerlab is a *router centric*, customizable testbed that reflects the macroscopic structure of the Internet, with different clouds and tier levels and simulation of differing access properties via network emulation. This, in combination with a flexible *traffic generation framework*, enables us to conduct Internet-level experiments on a small scale, within the borders of our server room.
Routerlab's management platform called *Labtool* supports heterogeneous hardware landscapes and user groups (e.g., labcourse student teams, individual researchers working on differing layers, developers). To accomodate the span of differing requirements of these user groups, it offers both virtualized and bare-metal, non-virtualized resources and can support versioned, controlled changes to the physical topology.

1 Introduction

Over the last three decades, the tremendous increase in the size of the Internet has necessitated the development of increasingly complex network protocols and configuration tools for its management. Accordingly, it is increasingly important to explore the effects of proposed design choices for the *Future Internet*, and simultaneously be able to predict the reaction of today's Internet to any changes. To this end, there exist many theoretical models, simulations, and testbed environments, each with particular strengths and applicabilities, which all depend on the ability to abstract away the Internet's complexity.

Theoretical models like network queueing theory are important for understanding fundamental properties of these networks, and simulations can be used to model and understand giant networks at scale and under complete control. However, both approaches cannot directly handle the full complexity of real world networks, and require modeling abstractions to make problems tractable. Accordingly, they require verification of the results with real hardware.

Testbeds offer solutions to deal with the limitations imposed by theoretical models and simulator environments. The main benefit which testbeds provide is to expose the experiment to the realities of existing networking hardware and software, including the effects of device implementation details and bugs, timing issues, and physical layer conditions (e.g., wireless connection quality), all of which are typically abstracted away by analytical models and simulators. Testbed experimentation is thus essential for validating the simplifications made by theoretical models and simulators, and to gain further insight into the behavior of real systems. Moreover, methodology and configurations from testbed experimentation are typically more easily transferable to real-world deployments, as identical or similar hardware is used.

However, due to resource constraints, testbeds typically must operate on a much smaller scale than the Internet. Accordingly, it remains a challenge to understand the Internet at large, with its multi-tier structure and decentralized organization in many different *autonomous systems* (AS) with millions of hosts from a multitude of vendors, running diverse software implementations and configurations. This translates into the necessity for testbeds to closely reproduce relevant network characteristics of interest, while respecting the limitations of available resources.

Along these lines, this paper describes a testbed called *Routerlab* [5]. The Routerlab is a customizable, *router centric* testbed, reflecting the macroscopic structure of the Internet. Like many other testbeds, it is composed of commodity switches, routers, load-generating servers, and network path emulators (e.g., for emulating DSL access line characteristics), all from multiple vendors. To reflect the Internet's structure on a miniature scale, its devices are organized into several, hierarchical and semi-isomorphic clouds that are connected using a variety of different electrical and optical connections, depicted in Figure 1. This allows an experiment to comprise emulated end-user access lines, edge level routers, AS'es, and a high speed forwarding core, using real hardware.

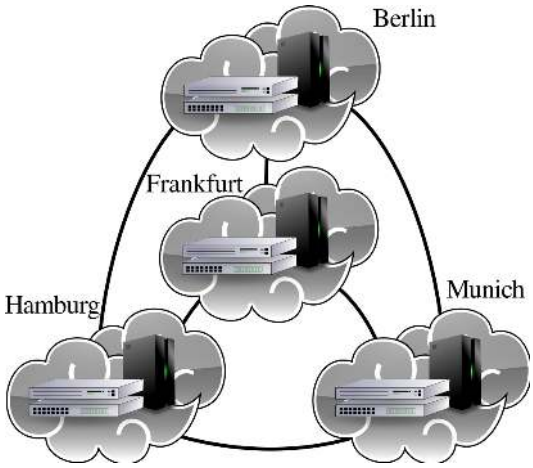

Figure 1: Routerlab Cloud Topology

A key component of *Routerlab* is its software management system, called *Labtool*, which allows multiple users to deploy their topologies, swap, and redeploy experiment configurations for easy reuse. It supports a heterogeneous hardware landscape and simultaneous, isolated experimentation by different user groups (e.g., labcourse student teams, individual researchers working on differing layers, developers). It further allows the use of multiple operating systems in real and virtualized environments. Other experimental features include Internet-like traffic generation, network emulation, and also tools for exact traffic measurements and infrastructure to monitor and collect experimental data. The experiment execution environment fully supports automation and experimental data analysis. Furthermore, the extensible design of the testbed allows the study of various different protocols, applications, and scenarios under different networking conditions without requiring deployment in the real Internet. When changes to the physical topology are necessitated, the Labtool tracks these in a versioned history.

This feature set differentiates *Routerlab* from other popular testbed platforms, which are largely switch centric and assume a fixed, unchanging physical topology. Several important challenges are associated with *Routerlab*'s router centric approach, including (a) interaction with the *heterogeneous hardware landscape*, including shareable and non-shareable, virtualizable and non-virtualizable resources, (b) accommodation of a large range of different user groups with widely *differing use requirements*, spanning user-space to kernel-space, and (c) ensuring isolation between experiments running in parallel, and thus providing reproducible results required for research.

Our paper continues with a presentation of several use-cases to motivate discussion of the design decisions of *Routerlab*. We provide an overview of existing testbeds, and compare them to our approach. We then detail the architecture and design of Routerlab, discussing associated challenges, and finally highlight experimental traffic generation conducted in the lab.

2 Use Cases

The presented testbed enables research in various fields. We are particularly interested in the following use cases, which are currently deployed in the Routerlab and used for active research.

HAIR [14] is a clean slate routing architecture that tackles the problem of routing table growth, restricts the visibility of routing updates, and inherently supports traffic engineering, mobility, and multipath. HAIR separates locators from identifiers. The routing and mapping system rely on a hierarchical scheme that leverages the structure of today's Internet. A prototype implementation was evaluated within the framework of the Routerlab, requiring a router-centric testbed design and making use of the tier-oriented architecture of the Routerlab.

While Future Internet ideas promise great value, deployment and adoption of new networking ideas to the Internet have proven to be problematic in recent years. **Virtualization**, and in particular **Virtual Networks** [18] are considered key enablers for deployment of Future Internet ideas [7]. They allow multiple network instances to co-exist on a common physical network infrastructure. In addition to the security properties offered by VPNs, virtual networks allow for completely isolated network instances, that can run custom network protocols and routing, may have bandwidth guarantees. The presented testbed is a key element in the evaluation of virtual networks and currently runs the prototype proposed in [18]. This experiment setup makes heavy use of the flexibility offered by the Routerlab, combining centralized storage and automatically managed network components, but also bare-metal resources managed by the experiment itself. These are used to run performance tests with customized virtualization techniques (e.g., based on XEN and KVM), while virtualized resources are used for development.

An emerging link virtualization technology, OpenFlow [15] introduces innovative flow switching possibilities into future networks. As in virtual networks according to [18], **OpenFlow** can be used to separate physical networks, allowing for experiments to be staged in productive networks. The Routerlab is partially OpenFlow enabled and used for ongoing research on OpenFlow.

Moreover, virtual networks allow innovative applications, e.g., concerning troubleshooting, debugging and safe upgrades to network components [23]. For instance, multiple versions of the same component or software can be run on **MirrorNets** in parallel virtual networks and a comparison of their behavior can shed light into possible bugs or performance implications. An experiment showcasing the key concepts, based on the mentioned virtualization framework and OpenFlow, was presented at SIGCOMM 2009 [24].

In recent years, a set of **Router Buffer Sizing** schemes have been proposed that challenge the prevalent rule-of-thumb applied in the dimensioning of Router buffer sizes, i.e., Router buffers being dimensioned according to the Bandwidth delay product. Given the tremendous growth of bandwidth on the Internet, the rule-of-thumb formula raises practical concerns in terms of memory usage. The current growth in bandwidth demand creates a dilemma: On one hand, high capacity links are only possible in the optical domain. On the other hand, memory requirements on the line cards adhering to the rule of thumb are not possible in the optical domain, due to lack of large hardware buffer implementation [11]. Also it is largely acknowledged that buffer resizing of linecards can have negative implications on the overall network performance, in particular on QoE factors. Therefore, sophisticated evaluations of existing and novel buffer sizing schemes are needed for their practical validation. In addition to these use cases, research areas such as **1–10Gbps Packet capturing** require dedicated hardware access [19]. For this purpose, there is a need for not only generating representative traffic at very high speed but also for infrastructure that can capture and monitor those packets.

As more and more Internet Service Providers integrate voice and video products into their product portfolio, the **Quality of Experience** describing the subjective service quality becomes a key element in the success of upcoming services. The Routerlab testbed presents an experimental platform allowing the study of influences of network parameters on the subjective service quality.

Moreover, we use the testbed as a **teaching** environment on a yearly basis. In this course, we invite students to take part in putting the important pieces of the Internet together. In this way, students get hands-on experience, and accesss to hardware that is typically only available at large network operators.

3 Testbed landscape

There are numerous frameworks available for building testbed networks. We now discuss their similarities and differences from our setup in the Routerlab.

Academic Testbeds Globally, **PlanetLab** [9] is probably the most successful and widely deployed testbed platform in existence. Led by a Princeton initiative, it consists of several thousand virtualized nodes connected via the Internet. Users can request groups of virtualized guest nodes (called slices).
VServer, a Linux kernel extension, is used as virtualization platform. As a container based single OS approach, the virtualization provided by VServer is light-weight and performant, but restricted to user level guests – there is no possibility for an experimenter to run custom operating systems or adapt the running kernel to their requirements. Performance isolation is limited: The performance of any given slice can be severely impacted if other slices cause heavy load.
The network is not virtualized at all in PlanetLab. Instead, every host has but one global IP address that is shared between all the clients. Connection quality varies according to the utilization of the underlying Internet connections. These properties make PlanetLab ideal for testing application level or overlay applications under Internet conditions, but less suited to reproducible experimentation, or for applications that require changes to the OS kernel. The PlanetLab framework assumes a pure PC based infrastructure, and does not support interaction with enterprise networking hardware such as routers and switches.
VINI [10] an extension to PlanetLab, adds network virtualization capabilities to PlanetLab, allowing the user to specify *virtual topologies* between their reserved nodes, that are set up via tunnels between the substrate nodes. First running as PlanetLab guest via User Mode Linux, the network virtualization was later moved to the substrate via Trellis [12]. The Official VINI nodes run in Internet2 locations with good connectivity, reducing cross-slice traffic congestion. It is still limited to userspace guests and does not support interaction with Enterprise networking hardware.
Emulab [21] is a testbed management framework geared towards local, tightly connected testbeds. It features experiment lifecycle management and can automatically provision and configure different kinds of networking hardware. Custom experimental topologies are supported via VLANs configured on the underlying physical network. Emulab does not currently support organized custom rewiring and assumes a switch centric network. Routerlab, in contrast, is a router centric network and management framework, mirroring the situation in the macroscopic Internet.
Orbit [16] is a wireless testbed at Rutgers University. The accompanying management framework is specifi-

cally geared towards the properties of wireless experiments. In the wireless domain, isolation is tricky due to the realities of the wireless transmission medium, forcing concurrent experiments to be separated in spectrum or physical space. It does not focus on virtualized networks or interaction with enterprise networking hardware.

The **OpenNetwork Laboratory** [22] is a testbed and management framework built around special purpose programmable network processors. It has GUI support for configuring and provisioning experiment topologies. It also has been extended to support management of other enterprise networking hardware. It assumes a switch-based backbone connecting the individual components, and does not support custom cabling at this point.

National/international testbed initiatives There is a growing awareness of the importance of large-scale testbeds in the research community. As such testbeds require substantial financial resources to build and maintain, they are typically hard to sustain by a single research group. Thus, many initiatives to build larger, federated testbeds have formed in recent years.

GLab [3], a German national testbed initiative, aims at building a large scale federated testbed for repeatable networking experiments. Its first prototype currently runs on PlanetLab technology, limiting its use to experiments supported by this platform.

GENI [2], an American initiative, aspires to build a large-scale comprehensive testbed infrastructure, in a spiral based development approach. The current spiral focuses on extending and federating existing platforms like PlanetLab, Emulab, and OpenFlow. Accordingly, there is not yet a consistent GENI architecture, but differing approaches by the individual research groups that are converging.

Summary Compared to the existing testbeds and frameworks, our approach provides a router-oriented, yet flexible testbed that is customizable down to the physical layer. Accordingly, we also require a custom solution that exposes this flexibility to the end users.

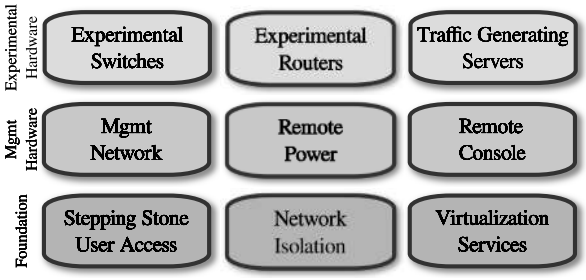

Figure 2: Logical Hardware Architecture

4 Hardware infrastructure

The Routerlab hardware infrastructure is designed to maximize device availability, utilization, and ease of expansion. In order to support a diverse network environment, our infrastructure is designed with vendor neutrality in mind. Toward these goals, the hardware of Routerlab is made up of devices for experimentation as well as dedicated management hardware to support and provide access to the experimentation hardware. The complete hardware infrastructure of the Routerlab can be organized into the three layers depicted by Figure 2 and realized in Figure 3.

4.1 Foundation

The foundation management hardware supports the most fundamental services for the Routerlab, including the means by which the rest of Routerlab is kept separate – but also accessible from the Internet. This separation is essential, as high volumes of experimental traffic which may be generated within the Routerlab must be kept from entering the public Internet. At the same time, traffic such as software package downloads from public repositories or data collected on experimental devices must be permitted to pass freely into and out of the Routerlab respectively. To this end, the foundation hardware utilizes a tightly controlled gateway server to filter and regulate these types of traffic. This server furthermore provides a virtual host acting as a stepping stone, through which users must authenticate before logging into the experimental devices from the public Internet. The gateway server provides a convenient platform for running other basic services within both the internal Routerlab network and the public Internet. These include *directory services*, a *web based information and monitoring platform*, and *network infrastructure services*, such as DNS, NFS, NTP, *storage resources*, and *OS virtualization* for isolation of user-accessible services from back-end services such as authentication, *monitoring of all infrastructure services*, a full *on-site data backup* solution, and other security-related systems.

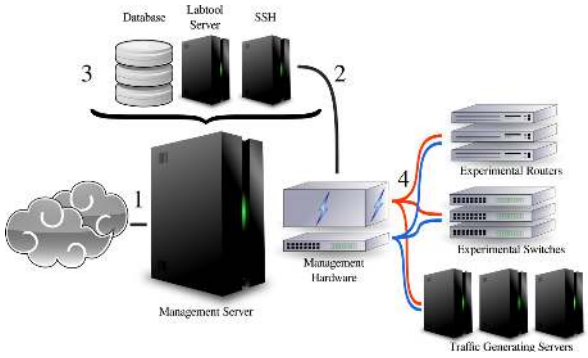

Figure 3: Physical Architecture

4.2 Management hardware

The middle architectural layer provides direct support to the experimental hardware within the isolated internal environment ensuring availability and uniformity of access. The Routerlab management hardware includes Gigabit Ethernet management switches, SNMP enabled power sockets, and serial console servers, providing network connectivity, remote power cycling ability, and console access to all experimental devices. These management devices ensure that all experimental devices can be reached for interaction, or rebooted by users in a consistent and familiar way, without the need for physical access to the experimental devices – even when the devices themselves do not feature integrated out-of-band management features.

4.3 Experimental hardware

In order to support the wide spectrum of requirements that results from the research work in the group, the Routerlab experimental hardware landscape consists of a multitude of different devices from several different classes. It contains commercial *routers*, *classical Ethernet switches*, *OpenFlow switches*, *programmable network hardware* based on NetFPGA, and *general purpose servers*, see Table 1. These devices are grouped into two different areas, an *Internet oriented, router-centric* area, and a *Switch centric, general research area*.

This heterogeneous hardware landscape poses significant challenges to the management platform. For instance, some of these devices can be safely shared between experimenters and used through controlled Interfaces. This includes virtualized servers, infrastructure services, such a storage, and some switches. Other devices, e.g., classical routers, can only be assigned exclusively, and need to be time-multiplexed between competing experimenters.

Each type of devices offers a unique set of in-band and out-of-band administration facilities. Some offer *lights-out management* capabilities to enable remote administration, while others don't. In these cases, we rely on external management hardware, e.g., SNMP controlled power sockets and terminal servers to provide control interfaces. All control interfaces are then abstracted and uniformly presented to the user by the management platform.

The links between switches, routers, and load-generating servers can be defined both through physical cabling but also logically via the VLAN configuration of the experimental switches – configurable via our software management system. This flexibility allows for the quick and repeatable deployment of complex, tiered topologies made up of modular elements as that pictured in Figure 4, many of which may further be software defined – especially in the case of topologies utilizing OpenFlow based routing.

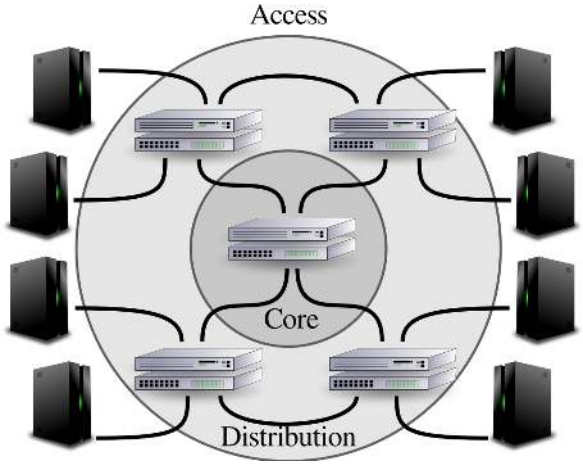

Figure 4: Constellation Experimental Topology

4.4 Resource sharing between testbeds

The vendor-agnostic management layer installed in the Routerlab increases our testbed flexibility by allowing us to easily integrate hardware from other testbed environments. The Routerlab shares a number of experimental devices with other testbeds, which introduces the challenge of integrating external experimental hardware into the Routerlab in a uniform fashion, yet still allowing for separation of resources when each testbed must function independently. Our management hardware in conjunction with our software management tool enables us to flexibly share devices, and to quickly reassign devices according to the current load on the testbeds.

5 Labtool software management

The Labtool software management system lies at the heart of the Routerlab, sitting atop the foundation hardware infrastructure, and serves as the primary interface through which users of the Routerlab reserve and simultaneously interact with their experimental hardware. The Labtool is experiment-centric, in that it organizes all of its functionality around the management, configuration, and repeatable deployment of experimental topologies and device configurations. The software architecture of the Labtool utilizes a three-layer client, server, and database structure, and is built to be extendible and scriptable with a client API in Ruby. In this section, we begin by stepping through the key features. Then we look a bit further into the software internals of the Labtool. Finally, we step through user experience of setting up and executing an experiment.

5.1 User features

Experiment life cycle management: The Labtool maintains an experiment-level view of all the actions it

Type	Vendors	Technologies	Shareable	Virtualized	OBM‡
Classical Switch	Cisco, HP, Linksys	IOS, 1Gb/s FX/TX	∗	no	∗
OpenFlow Switch	HP, Quanta, NEC	OpenFlow	no	no	no
Router	Cisco, Juniper	IOS, JUNOS, 1Gb/s, FX/TX	no	no	no
Server	Sun, Other	x86, Opteron, NetFPGA	∗	on demand	∗

Table 1: Routerlab experimental hardware (∗: partially, ‡: Out-of-Band Management)

performs. In concrete terms, this means that devices, the physical and virtual links connecting them, and their individual configurations are kept related in the underlying database schema. This allows for easier hibernation, migration, and restoration of any particular experimental setup.

Physical topology versioning: The Labtool keeps track of all custom cabling changes over time and across experiments. Versioned cabling enables Routerlab administrators to alter and reliably restore topology changes.

Boot-mode configuration with imager system: An experiment performed on one set of devices should be repeatable on another set of suitably similar devices. To this end, the Labtool allows experimental device configurations for a given device to be redeployed onto any sufficiently similar device. In the case of traffic-generating servers, individual disk partitions or full hard disk device images may be archived or restored by the Labtool imager system from any server to any server. The Labtool provides a collection suitably hardware-agnostic base operating system images which facilitate quick deployment of OS experimental environments.

5.2 Labtool software internals

In order to realize these features, the Labtool integrates a collection of modular components and presents them via a uniform client interface. These components are implemented largely in Ruby and Perl in a three-tiered server daemon, database back-end, and client architecture. The first software component handles device reservation management, calculates reservation timeslots, and prevents conflicts from arising between users running multiple experiments. The next component of the Labtool is the imager system, which encompasses a set of tools to automatically backup or restore a particular hard drive partition or full disk. The imager makes uses of a NFS-based storage system, from which a network-bootable Linux OS image can be deployed on any traffic-generating server. From this image, incremental file-system backups can be made to and restored from the NFS storage pool. Another core component implements the device interaction functions which provide power-management and diagnostic console access. These components also help lay the foundation for a separate experiment automation process we have under development, which supports further coordinated execution and collection of data distributed across multiple experimental devices.

5.3 End-user experience

The process of utilizing the Routerlab begins with the user authenticating against – and logging into the stepping-stone server as depicted by step 1 in Figure 3, from which the Labtool client can be invoked. The user may then invoke the Labtool to create a new experiment or reload a previously saved experiment. In the event that the user reloads a previously defined experiment, once the necessary devices become free, the user can immediately begin to log into and interact with the experimental devices via the management network depicted by step 2. Otherwise, experimental devices are picked out for the experiment and a timeslot is reserved by the user, often in advance, for their use. Each Labtool experiment encompasses a collection of specific experimental devices, the full physical cabling topology defined by the selected devices, and the complete device configuration for each device. All of these elements are represented in the relational database back-end of the Labtool system. The Labtool database is consulted to obtain the appropriate device configurations for each device in the active experiment, shown by step 3. As the user conducts the experiment, she may interact with the devices through the Labtool without needing to know any of the details of the underlying management network. For example, in the event that a user wants to powercycle a device, it suffices to issue the Labtool powercycle command which requires knowledge only of the device name. The appropriate SNMP directives are issued to the appropriate power source for the appropriate hardware device. Interactive console access and configuration management proceeds similarly over the course of the experiment, depicted by step 4. The experimental devices may all be driven and coordinated from the login server, upon which all the experimental data may be collected, once the actual experiment has finished.

6 Traffic generation

The biggest challenge of emulating the Internet is to feed the actual hardware of Routerlab with traffic that is qualitatively identical to that observed in the provider networks. In the real Internet, users send requests to

different application servers which are geographically placed at various different locations. This brings another challenge that the generated traffic must not only show the same flow-level behavior but also exhibit similar delays. For the purpose of traffic generation and network delay emulation we employ tools such as *Harpoon* [20] which is a flow-level traffic generator and a successor to *SURGE* [8]. The Harpoon traffic generation is based on five distributional parameters which can be easily extracted from the netflow data captured at any vantage point in the Internet. One of the key features of Harpoon which distinguishes it from its counterparts, is that it relies on the system's underlying native TCP stack. For emulating end-to-end performance characteristics of Internet we rely on emulation tools such as *Dummynet* [17] and *NistNet* [4]. These tools provide the ability to emulate asymmetric DSL link bandwidths.

A sample run of traffic generated with heavy-tail flow distribution and a RTT of 150ms for a duration of 120 seconds is shown in Figure 5. This plot indicates that the generated traffic complies not only with the configurable average throughput but also exhibits burstiness characteristics. This complementary usage of traffic generation and network emulation leverage Routerlab hardware for more realistic network experimentation.

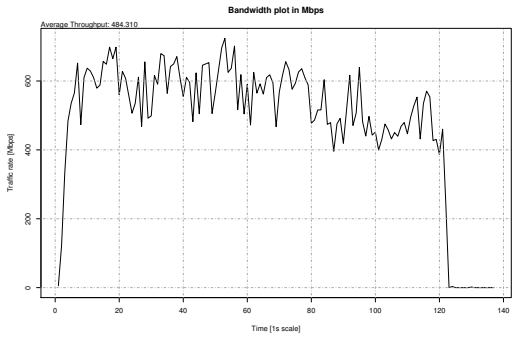

Figure 5: Generated Traffic (Mbps) – Time scale 1sec

6.1 Traffic monitoring and capturing

Monitoring the state of experiments is crucial for accurate and reliable measurements. To this end, Routerlab has enhanced capabilities for monitoring links via the span port in switches and optical splitters. Moreover, for experiment data capturing, Routerlab provides dedicated high volume storage devices. This data is then analysed and used for post experimentation analysis.

6.2 Traffic validation

It has been shown in the literature related to Internet traffic analysis that wavelet transformation [13] are the effective means for the analysis of Long Range Dependent – LRD traffic and for the calculation of Hurst Parameter [6]. In order to validate Long Rage Dependence we also rely on wavelet based energy estimation for different timescales. Since LRD stems from different ON-OFF sources, the wavelet energy spectrum remains high even over very large time scales. Our generated traffic is compliant with the previous reported results as shown in Figure 6.

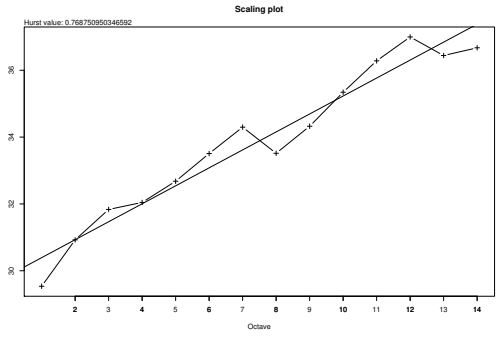

Figure 6: Scaling Plot - log log diagram

7 Summary and outlook

In this paper, we introduce the Routerlab, a *router centric* testbed that reflects macroscopic properties of the Internet on a miniature scale. This is complemented by a flexible traffic generation framework that offers customizable levels of synthetic and captured traffic to be introduced into the experiments, while ensuring they mirror important properties of real Internet backbone traffic.

Another important element in the design of the Routerlab is a vendor agnostic management platform, supporting a wide heterogeneous landscape of different enterprise and network devices. With the help of this framework, the Routerlab can be extensively customized to experiment requirements, from a free choice of utilized operating system images and virtualized and non-virtualized resources and virtual topologies down to versioned and controlled changes to the underlying physical topology, when unavoidable.

The Routerlab is under extensive use for a wide range of experiments. To further improve experimenter experience we plan to add the following features in the near future:

Experiment automation: Currently under development, an improved experiment automation process, will offer experimenters the opportunity to fully automatically, configure, provision, and run experiments in coordination with monitoring and capturing facilities, then automatically collect results and input them into user defined analysis frameworks.

Virtual Network integration: As Virtual Networks, realized by OpenFlow and other technologies, become an integral part of network architectures, we intend to

fully integrate VNet support into the Routerlab, allowing for higher degrees of freedom in topology definition and automatic combination of virtualized and non-virtualized testbeds.

Testbed federation:n the context of national and international projects [1, 2, 3], several approaches and architectures for automatic testbed federation are currently being discussed. We envision the Routerlab to be fully integrated in some of these federations in the future, opening its resources and management abilities to experimenter outside of our research group.

References

[1] 4ward project. `http://www.4ward.eu/`.

[2] Geni project. `http://www.geni.net/`.

[3] German lab. `http://www.german-lab.de/`.

[4] Nist net home page. `http://www-x.antd.nist.gov/nistnet/`.

[5] Routerlab. `http://net.t-labs.tu-berlin.de/research/routerlab`.

[6] P. Abry, P. Goncalves, and P. Flandrin. Wavelets, spectrum analysis and 1/f processes. In *Wavelets and Statistics, Lecture Notes in Statistics*.

[7] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the internet impasse through virtualization. *Computer*, 38(4), 2005.

[8] P. Barford and M. Crovella. Generating representative web workloads for network and server performance evaluation. In *SIGMETRICS*, pages 151–160, 1998.

[9] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating system support for planetary-scale network services. In *USENIX NSDI*, 2004.

[10] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In vini veritas: Realistic and controlled network experimentation. In *ACM Sigcomm*, 2006.

[11] N. Beheshti and Y. Ganjali. Packet scheduling in optical fifo buffers. In *High-Speed Networking Workshop (In Conjunction with IEEE INFOCOM)*, 2007.

[12] S. Bhatia, M. Motiwala, W. Mühlbauer, Y. Mundad, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford. Trellis: A platform for building flexible, fast virtual networks on commodity hardware. In *ACM ROADS Workshop*, 2008.

[13] A. Feldmann, A. C. Gilbert, W. Willinger, and T. Kurtz. The changing nature of network traffic: scaling phenomena. *ACM Sigcomm CCR*, 1998.

[14] A. Feldmann, L. Cittadini, W. Mühlbauer, R. Bush, and O. Maennel. Hair: Hierarchical architecture for internet routing. In *ACM ReArch Workshop*, 2009.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *ACM Sigcomm CCR*, 38(2), 2008.

[16] D. Raychaudhuri, M. Ott, and I. Secker. Orbit radio grid tested for evaluation of next-generation wireless network protocols. In *TRIDENTCOM*, 2005.

[17] L. Rizzo. A simple approach to the evaluation of network protocols. acm computer communications. *ACM Sigcomm CCR*, 2007.

[18] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy. Network virtualization architecture: Proposal and initial prototype. In *ACM VISA Workshop*, 2009.

[19] F. Schneider, J. Wallerich, and A. Feldmann. Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware. In *PAM*, 2007.

[20] J. Sommers and P. Barford. Self-configuring network traffic generation. In *ACM IMC*, 2004.

[21] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *USENIX OSDI*, 2002.

[22] C. Wiseman, J. Turner, and P. Crowley. The open network laboratory. In *ACM Sigcomm Demo Session*, 2009.

[23] A. Wundsam, A. Mehmood, A. Feldmann, and O. Maennel. Improving network troubleshooting using virtualization. Technical report, Technische Universitaet Berlin, Fakultät Elektrotechnik und Informatik, June 2009.

[24] A. Wundsam, A. Mehmood, A. Feldmann, and O. Maennel. Network troubleshooting with shadow vnets. In *ACM Sigcomm Demo Session*, 2009.