ARTICLE IN PRESS

Computer Networks xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Future Internet research and experimentation: The G-Lab approach

Dennis Schwerdel [a], Bernd Reuther [a], Thomas Zinner [b], Paul Müller [a,*], Phouc Tran-Gia [b]

[a] Integrated Communication Systems Lab, University of Kaiserslautern, Germany
[b] Chair of Communication Systems, University of Würzburg, Germany

ARTICLE INFO

Article history:
Received 16 August 2013
Received in revised form 20 December 2013
Accepted 24 December 2013
Available online xxxx

Keywords:
Future Internet
Research
Experimentation
Simulation
Experimental facility
Virtualization

ABSTRACT

The German Lab (G-Lab) project aims to investigate architectural concepts and technologies for a new inter-networking architecture as an integrated approach between theoretic and experimental studies. Thus G-Lab consists of two major fields of activities: research studies of future network components and the design and setup of experimental facilities. Both are controlled by the same community to ensure that the experimental facility meets the demands of the researchers. Researchers gain access to virtualized resources or may gain exclusive access to resources if necessary. We present the current setup of the experimental facility, describing the available hardware, management of the platform, the utilization of the PlanetLab software and the user management. Moreover, a new approach to setup and deploy virtual network topologies will be described.

1. Introduction

The G-Lab project[1] started in 2008 as a distributed joint research and experimentation project for Future Internet studies and development. This BMBF project[2] was initially distributed across six universities in Germany: Würzburg, Kaiserslautern, Berlin, München, Karlsruhe, and Darmstadt. The G-Lab project itself is divided in two parts, Future Internet research, and the experimental facility (ExpFac), with the overall goal to foster experimentally driven research.

The research aspects of the G-Lab project are not limited to theoretical studies and novel ideas, but also includes experimental verification of the derived theoretic results. Investigation of new approaches concerning for example routing, addressing, monitoring and management, content distribution, and their interaction is such an intricate task that it could not be validated with only analytic research methods. Consequently, in G-Lab, theoretic research and the ExpFac are closely intertwined. This means that theoretic work leads to practical experimentation in the ExpFac, which leads to modification of the algorithms and models used. In the end, this spiral process can converge into a new inter-networking architecture as depicted in Fig. 1.

Within this process the ExpFac adapts to the demands of the actual research work. Thus it is important that the ExpFac is flexible enough to adapt to the needs of the experiments and ultimately became a research field itself. With this G-Lab aims to avoid the situation that platform providers offer services that nobody uses.

G-Lab is organized in two phases. Phase one involves six universities, each one focusing on different theoretical work, every university also operates and hosts parts of the distributed ExpFac. The first phase is structured into eight activities (ASPs) as shown in Fig. 2: project coordination, architecture [1–3], routing [4,5], wireless and mobility [6],

* Corresponding author. Tel.: +49 6312052263; fax: +49 6312053056.
E-mail addresses: schwerdel@informatik.uni-kl.de (D. Schwerdel), reuther@informatik.uni-kl.de (B. Reuther), zinner@informatik.uni-wuerzburg.de (T. Zinner), pmueller@informatik.uni-kl.de (P. Müller), trangia@informatik.uni-wuerzburg.de (P. Tran-Gia).

[1] German-Lab project website: http://www.german-lab.de.

[2] Funded by Federal Ministry of Education and Research (BMBF) 2008–2012.

1389-1286/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.bjp.2013.12.023

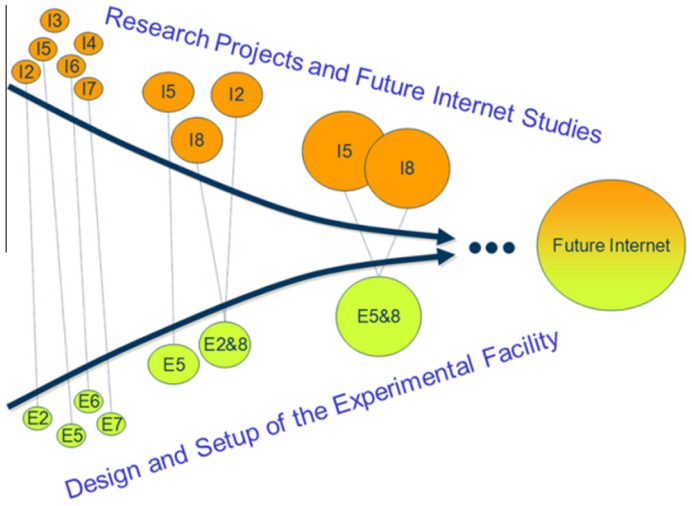

Fig. 1. German-Lab philosophy: theoretic ideas (I) and practical experiments (E) influence each other and help close the gap between theory and reality, leading to an actual Future Internet prototype.

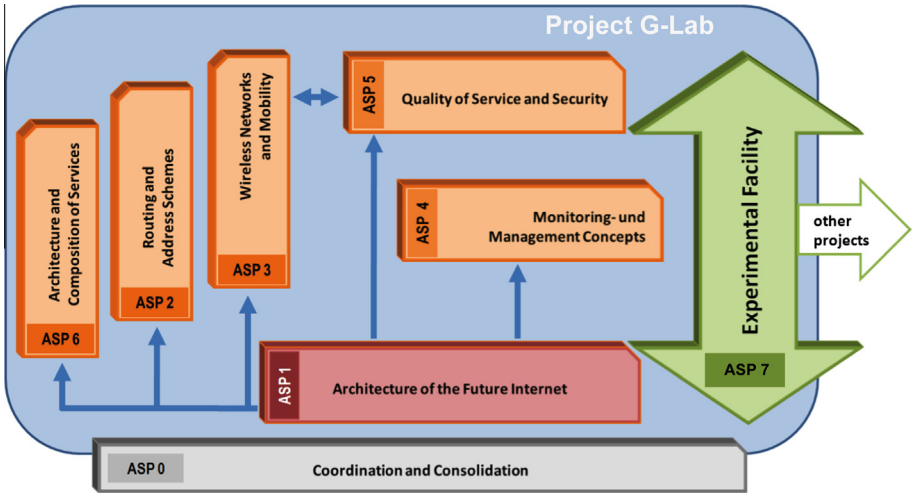

Fig. 2. German-Lab project structure.

monitoring, QoS and security [7], service composition [8], and the ExpFac [9]. The last activity is responsible for the operation, adaption, and enhancement of the ExpFac. In the second phase, G-Lab has been extended by industrial partners.[3] Most of the projects in phase two focused on deepening, extending, or applying the work being done in the first phase. Moreover, some partners of phase two enhanced the ExpFac with special equipment like sensor networks.

The remainder of this paper is organized as follows. In Section 2 we give a high-level overview of the hardware in the G-Lab ExpFac with examples of its use. In Section 3 we describe a new approach for designing and deploying virtual network topologies. In Section 4 we describe two experiments in the context of German-Lab. In Section 5 we describe the first federation efforts of the G-Lab ExpFac. In Section 6 we draw some conclusions and highlight key areas for future work.

2. German-Lab Experimental Facility (ExpFac)

The ExpFac consists of wired and wireless hardware with over 170 nodes, which are fully controllable by the G-Lab partners. The platform is distributed into clusters at six different locations within Germany, with Kaiserslautern as the main site [9].

A major challenge of the ExpFac is fulfilling the changing requirements of the research projects and sometimes even integrating intermediate research results. This leads to a continuously changing infrastructure.

Since the ExpFac and the research projects that use it are part of the German-Lab project, they start at the same time. Therefore it is a special challenge for the ExpFac to be ready for the first experiments as fast as possible. Thus, mostly existing software was used in the beginning and

[3] German-Lab phase two partners and projects: http://www.german-lab.de/phase-2.

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

Table 1
Phase 1 node hardware.

Node type	CPU	RAM	Disk	Network interfaces
Head node	2× Xeon Quad E5450 3.0 GHz	16 GB	16× 146 GB SAS	4× 1 Gb/s
Network node	2× Xeon Quad L5420 2.5 GHz	16 GB	4× 146 GB SAS	8× 1 Gb/s
Normal node	2× Xeon Quad L5420 2.5 GHz	16 GB	4× 146 GB SAS	4× 1 Gb/s

Table 2
Phase 1 node counts.

Site	Head nodes	Network nodes	Normal nodes
University of Kaiserslautern	1	2	56
University of Würzburg	1	2	22
Karlsruhe Institute of Technology	1	2	22
University of Munich	1	2	22
University of Darmstadt	1	2	22
University of Berlin	1	2	12

that software was substituted over time by software developed for specific purposes.

2.1. Hardware equipment

The core of the ExpFac consists of nodes located at 6 sites. Table 1 describes the different node types:

Normal Node: This is the standard type of node, which can be used to run networking tests, computations and simulations.
Network Node: The second node type is designated for special networking tests requiring more network interfaces.
Head Node: The last type acts as a head node for the local site. It has the task of managing the local site.

All the nodes include a dedicated service processor allowing remote control and monitoring the hardware through a special management network interface. Each site has one head node, two network nodes and a variable number of normal nodes as shown in Table 2. In phase 2 of the German-Lab project, a few additional nodes in Stuttgart, Hamburg, Hannover and Passau were fully integrated into the ExpFac.

The networking equipment consists of one layer-3 switch from Cisco Systems (Catalyst 4500 E Series) at each site. To enable experiments with OpenFlow, the Würzburg nodes are also equipped with OpenFlow switches. Additionally, some nodes run OpenVSwitch to provide OpenFlow functionality.

As part of phase 2, some projects also brought specialized hardware such as mobile testbeds and sensor nodes into the ExpFac. As this hardware is custom to specific projects it is not fully integrated into the ExpFac but all project members have access privileges to it.

2.2. Network setup

Each site has two logical networks: a main network and a management network. The main networks are connected to the Internet via the campus networks, and interconnected using the DFN[4] on layer 3. All of the nodes are connected to their main network with as many interfaces as possible.

The management network connects all the management interfaces of the nodes (which should not be exposed to the Internet due to security considerations). These two networks are separated logically using VLANs and only the main network has a connection to the campus network. To save some ports on the main switch, 100 Mb/s switches are used to collect the management interfaces at most of the sites. The head node has one of its normal network interfaces inside the management network and serves as a gateway into that network. Authorized users can get access to the management network using VPN technology. This structure is depicted in Fig. 3.

In general, all nodes have one public IPv4 address that is assigned via static DHCP to their primary interface. Additional addresses may be obtained either from the DHCP pool or from a defined range of reserved addresses. IPv6 is available at most sites and is assigned either by stateless auto-configuration or by DHCP. The interfaces in the management network have private IP addresses that are routed via VPN. The domain german-lab.de, which is used for all nodes and services within the facility, is managed at the Kaiserslautern site. Site-specific zones are delegated to each site to allow decentralized DNS management.

2.3. Head nodes

Each site has a head node that is used to manage that site and provide infrastructure services. Originally this head node had an operating system running directly on the hardware but that became too inflexible and hard to manage. Hence the physical head node is now virtualized using Proxmox [10] virtualization software. This has a number of major advantages:

- Different functionalities can be separated into different virtual machines. Those virtual machines can run different operating systems. This solved a major problem: The PlanetLab software is RPM[5]-based, but most of the administrators were used administrating Debian systems.

[4] DFN: German research network, http://www.dfn.de.

[5] RPM Packet Manager.

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

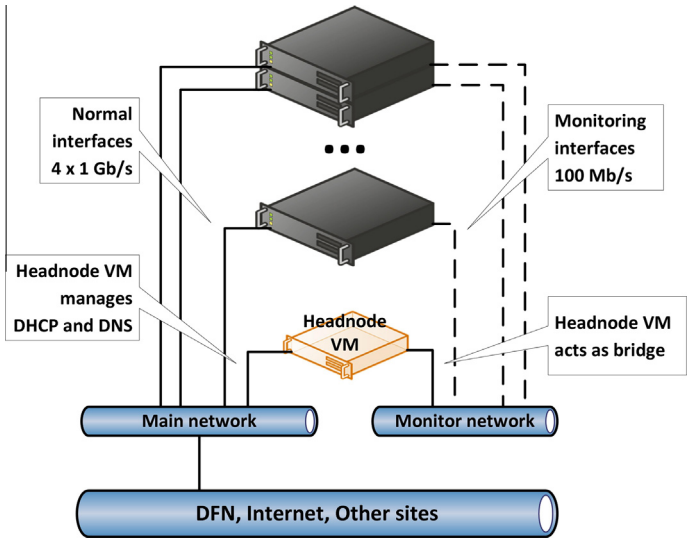

Fig. 3. The network structure consists of two networks, the main network that is connected to to the other sites via the Internet and the private management network.

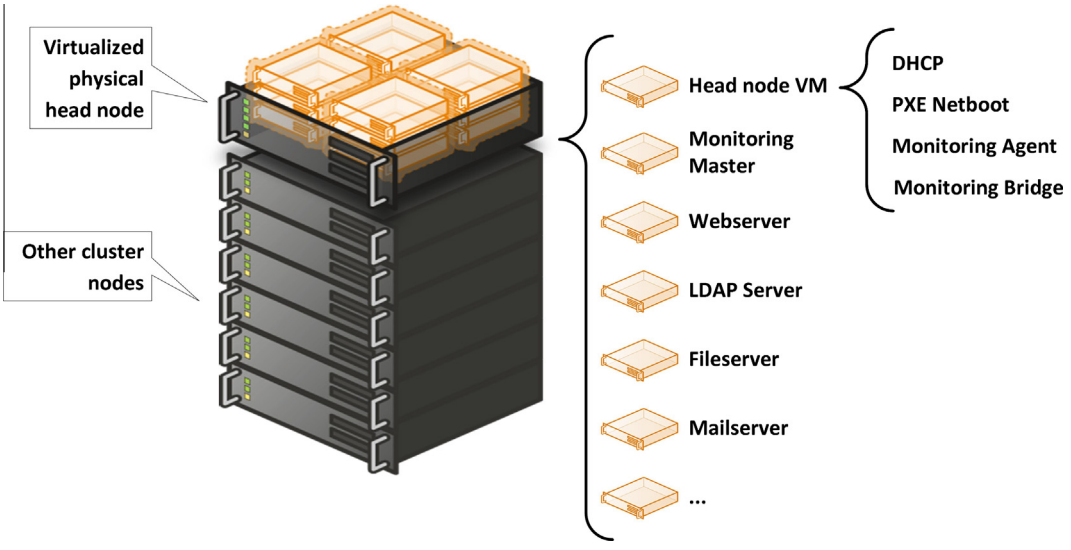

Fig. 4. The physical head node hosts several virtual machines that provide services for each site.

- In the case of a system failure, virtual machines are much easier to handle. The host system provides access to the console of the guest system even when the guest network is not running. Broken systems can also be restored from backups and be up and running in a few minutes.
- Virtual machines can be cloned, which enables backup and testing. This has been used in the beginning of the project to deploy the head node software to the other sites. However, it has been problematic to preserve site-specific modifications on updates, so head node deployment has been substituted by a system using packages and puppet [11] configurations.

The physical head node hosts several virtual machines providing services to the specific site or to the whole project. Depending on what is needed, a variable number of virtual machines reside on the physical head node as shown in Fig. 4. One virtual machine is specifically designed to control the local site, the virtual head node. It has the following functionality:

- It provides a DHCP server that assigns IP addresses to the nodes primary interfaces using a static mapping and offers a pool of dynamic addresses. At some sites this functionality is disabled as DHCP is provided by the campus network.

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

- The head node acts as a gateway to the management network and provides access to the management interfaces using VPN technology.
- All of the nodes are configured to boot via PXE netboot. The head node controls this by adding a field to its DHCP replies and by providing the boot configurations and boot images for the nodes via TFTP. An agent allows a remote service to manage the boot configurations and images, and to power cycle the nodes using their management interfaces. Section 2.7 explains this in more detail.
- The head node also has an agent service that can be controlled by remote monitoring software. Via this monitoring agent, the head node exposes not only its own state, but also the states of the nodes that can be obtained from the management interfaces. Section 2.4 explains this in more detail.

2.4. Monitoring

The German-Lab facility uses the monitoring framework Icinga[6] to monitor the entire infrastructure. A dedicated virtual server in Kaiserslautern is used for the monitoring infrastructure. It hosts the monitoring daemon as well as the graphical web interface and also notifies administrators by email when problems are detected.

Nagios Remote Plugin Executor (NRPE)[7] software is used to collect monitoring data from remote hosts. NRPE is a small agent that runs on the remote systems and allows authorized monitoring servers to execute pre-configured commands to obtain the internal state of this host. In German-Lab this functionality has been extended: NRPE runs on all head nodes and collects monitoring information from the management interfaces of the nodes via SNMP.

Using this configuration, the following monitoring data can be obtained:

- Availability of all nodes and service processors.
- Resource usage (CPU, memory, disk, etc.) on all virtual machines (via direct NRPE).
- Hardware health of all nodes (using the management interfaces via NRPE).

Additionally, existing monitoring systems of the network provider DFN and the local campus network operators are provided to the administrators so they can also monitor the network. Overall network monitoring shows only minimal one-way-delays (OWD) between the six main G-Lab sites (see Fig. 5). In order to create more realistic network behavior, it is necessary to create artificial link characteristics without affecting the real infrastructure. This can be done with the Topology Management Tool (ToMaTo) that allows link emulation to configure all relevant QoS parameters as described in Section 3.2.

The G-Lab monitoring architecture has proven very valuable as it helped to detect and solve problems quickly. Problems that can be fixed without hardware change have frequently been solved within a few hours.

[6] Icinga is a modern fork of the Nagios monitoring software, http://www.icinga.org.

[7] NRPE: http://docs.icinga.org/latest/de/nrpe.html.

2.5. Identity management

User management is an important part of an ExpFac and the organization of user identities and access privileges are critical issues. In the G-Lab project user management is necessary in two different areas: infrastructure services, and the testbed platform.

Infrastructure services consist of the internal and external project documentation area, mailing lists, help desk, and software management. The testbed can be divided into management and experimenter views. The experimenter requires access to the nodes and testbed resources on several layers.

Administration of users and system resources is done by a distributed administration team organized as a subproject of the G-Lab project. This approach distributes the responsibility for the users of a specific site to a representative of that site. This procedure requires additional role and access rights assignments for an extended group of identities. For example, the headnodes, node management and monitoring, and the local PlanetLab node administration are typical tasks, which are delegated to site representatives. Site representatives also have to organize the experiments and the resource usage of their sites.

Fig. 6 shows an architectural overview of G-Lab's identity and role management. In general, a central LDAP server stores the user identities in a tree, which is organized in subtrees containing the users of specific sites. An identity is not associated with any access rights, this is organized in a separate tree, the group tree. Each service is represented by a unique group, which grants its members access to that service. A third separate subtree organizes virtual identities on a machine level, such that each site has its own system level access user. This enables a fine grained and easily manageable environment at the site level, even when changes are needed. For services such as the private PlanetLab installation, accounts are synchronized with the central LDAP database serving as the master. This can easily be extended to future services, if required. The management of the central database is done by a set of scripts, which respect defined default roles for specific tasks. These scripts also verify the integrity of the stored data.

2.6. Infrastructure services

The German-Lab ExpFac not only provides experimental resources but also provides infrastructure services for the whole project. These services should help the users with their research as well as build a community and inspire joint research. The German-Lab ExpFac offers the following services:

- The main project website which also hosts sub-project websites.
- The LDAP server that stores all user identification. This service can be accessed by projects that want to reuse the information to authenticate users. This way, users only have one account in the whole German-Lab and all associated services.
- The monitoring system of the facility can be used by researchers to monitor their long-running experiments

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

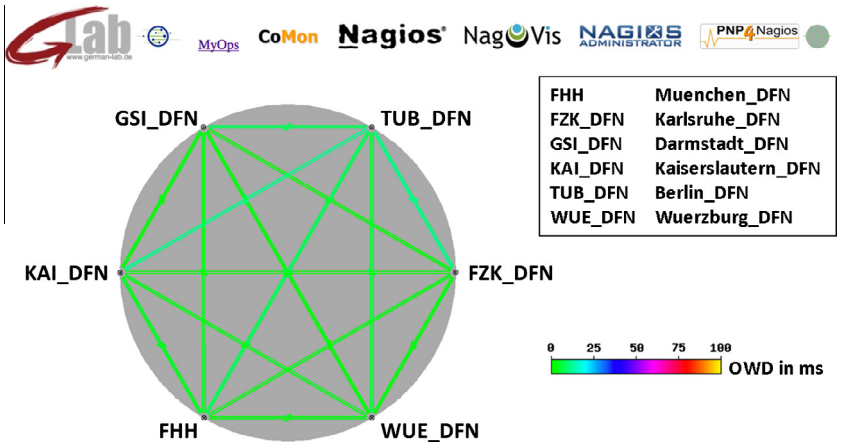

Fig. 5. Monitoring the delay between G-Lab Sites.

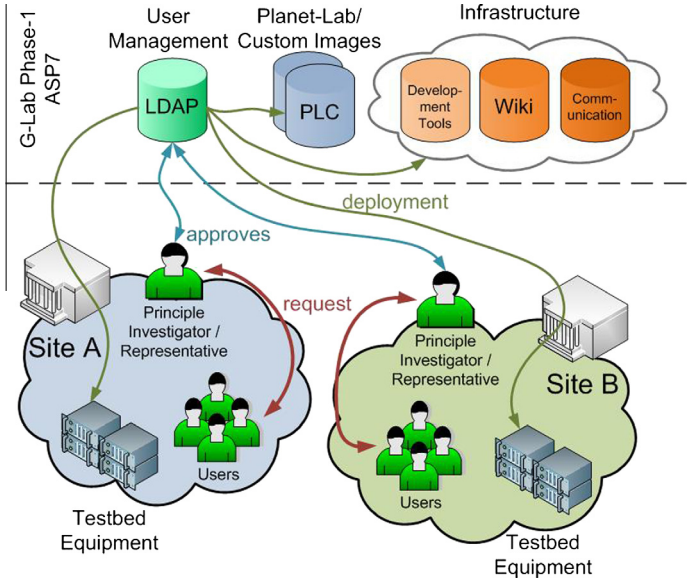

Fig. 6. Identity management.

or record environmental conditions during their experiment.

- Mailing lists and a Wiki system allow users to communicate with each other and share information. These systems are key enablers for the German-Lab community and have been used to initiate several cross-site collaborations.
- A fileserver provides Webdav access to a global shared storage as well as group folders. The fileserver also provides the versioning repositories Git and SVN so that users can store their documents and software.
- An OTRS[8]-based ticket system allows administrators to build a support infrastructure and work on open issues concurrently.

2.7. Control frameworks

One of the key concepts of the German-Lab ExpFac is its flexibility. While testbeds in general restrict the possible uses either by software or by policy, in German-Lab the ExpFac is open to all possible experiments, even if they impact the infrastructure of the ExpFac.

In reality this means that German-Lab has multiple control frameworks that users can choose from, and that individual nodes are not bound to a fixed control framework. Section 2.7.1 describes the system that is used in German-Lab to boot nodes with specific boot images.

The hardware setup can also be changed as required by experiments, especially the network nodes and their cabling. This includes manual modifications such as different wiring or more disk capacity. To reduce the amount of manual work, virtualization has been introduced also on

[8] Open-source Ticket Request System, http://www.orts.com.

the level of network topologies, resulting in the Topology Management Tool described in Section 3.

There is a clear trade-off between access for more users and more privileges for users. While software like PlanetLab gives all users access to all nodes and therefore enables huge experiments using many nodes, that access is limited to a high level of abstraction. In PlanetLab, users do not have access to kernel modules, raw sockets or network configuration. At the other extreme, custom boot images offer full access to the hardware with complete control over the kernel and the network interfaces. This comes at the cost of limiting access to one experiment at a time. The German-Lab facility offers both to its users and controls the share of globally accessible nodes with policies.

2.7.1. Boot image management

The Boot Image Management (BIM) system manages and controls the process of assigning boot images to nodes. It has been developed to reduce the manual work. BIM consists of a central server, a fileserver for distributing the boot images, and agents embedded in the head node software.

The central server provides a web-based user interface to administrators that allows them to upload and manage boot images, manage the entries representing the nodes, and assign boot images to those nodes. The BIM agents on the head nodes are responsible for managing the PXE boot configuration and for controlling the power state of the nodes via the management interface of the respective nodes.

When a new boot image is assigned to a node, BIM executes the following steps (Fig. 7):

1. The administrator uses the web frontend of the BIM master to request a boot image change.
2. The BIM master calls the BIM agent and updates the boot configuration of the node in the TFTP server. The agent then changes a file that corresponds to the MAC address of the node to make that node run the new boot image.
3. The BIM master issues a remote call to the BIM agent on the local head node at the site of the node to power-cycle that node.
4. Upon reboot, the node fetches the new PXE boot configuration from the BIM agent and then boots the image.

There is a special boot image entry that makes the node boot from the local hard drive. This can be used to boot an already installed operating system.

Boot images can also specify preparation boot images that must be booted once before that image can be booted. In that case the steps mentioned above are executed first for the preparation boot image that runs for a specific time and then those steps are executed again for the configured boot image. Using this two-step approach, BIM is able to execute preparation tasks like wiping the disks or simply run an installer once and then boot from the local disk.

2.7.2. PlanetLab

At the beginning of the project, the ExpFac needed to be constructed quickly because the first researchers already wanted to run experiments. As these experiments were mostly from the field of peer-to-peer software, a private PlanetLab [12] testbed was installed and used as the main boot image. The PlanetLab management software, MyPLC, was installed on the head node and later moved into a virtual machine. In a cooperation with the GENI PlanetLab testbed, the private German-Lab installation got access to the PlanetLab-specific monitoring and control tools.

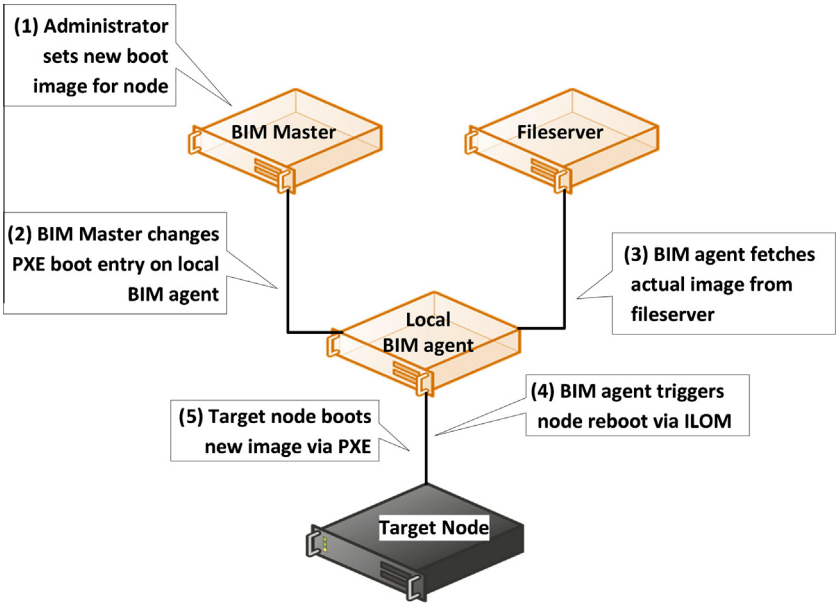

Fig. 7. The boot image management tool configures boot images using PXE.

Using this software, the ExpFac could provide all its users access to all available nodes and allow huge networking experiments. As the project progressed, other experiments had requirements that PlanetLab could not fulfill. The need for control over the network interfaces that PlanetLab shares with all the users made it necessary to offer alternative boot images.

2.7.3. Custom boot images

Custom boot images provide the users with the option to run an image of their choice on the nodes. This image can include any operating system and can be pre-equipped with the experiment software of the user. This image gives the user full control over the hardware, especially over the running kernel and the network interfaces.

The user can decide who gets access the nodes running his custom boot image by creating and sharing accounts on those machines. As custom images, like all other images, need to be configured by an administrator in BIM, this makes it easy to enforce some policies and make sure that the number of nodes booted with custom images is limited.

In general, custom images are very work intensive for the administrators as analyzing and fixing boot errors requires administrative access to the nodes. Sometimes custom boot images also need a custom hardware setup that must be configured manually, e.g. by interconnecting the interfaces in a special way. Virtualization has now taken a huge share of earlier custom image usage, but that option still exists for users that really need it.

2.7.4. Virtualization

As an alternative to boot images, virtualization has been introduced in the German-Lab ExpFac. The Proxmox virtualization software runs on several nodes and therefore all users can get access to virtual machines running their custom software. This is a good compromise between custom images and PlanetLab: I.e. users can run their own software and have full access to virtual machines while keeping the physical hosts available to all users.

As the project advances, more and more resources are virtualized and even PlanetLab nodes are now running in virtual machines. This way the number of nodes in PlanetLab can stay constant or even increase without limiting the use of other control frameworks.

Even with virtualization, some experiments require manual administrative effort, e.g., to inter-connect the networking interfaces in a special way or to capture packets at host level. The Topology Management Tool was developed to automate even those tasks by providing completely virtual topologies.

3. The Topology Management Tool (ToMaTo)

Over time more and more requirements from research projects were beyond the capabilities of the standard boot image (the PlanetLab image) because of their need to access to the lower network layers and to define the network topology. To solve this, an ExpFac software called Topology Management Tool (ToMaTo) [13,14] has been developed in the German-Lab project.[9] ToMaTo runs on top of the existing virtualized nodes of the ExpFac and enriches them with topology virtualization and additional functionality like packet capturing and link emulation.

ToMaTo allows researchers to create virtual network topologies populated by virtual nodes running the experiment's software. In this section we describe the ToMaTo software and compare it to other ExpFac software. The Topology Management Tool is a testbed specifically designed for networking experiments. In contrast to most other networking testbeds, ToMaTo is topology-oriented. As such, each experiment in ToMaTo consists of a topology of virtual components. Fig. 8 shows an example topology containing several components.

3.1. ToMaTo concepts

One of the main concepts of ToMaTo is that connections between components, or to external networks, are explicitly declared in the topology. This allows users to define reachability, routing, and link attributes in their topology as needed. The example in Fig. 8 shows that the links between components and elements like switches and the bridge to the Internet are explicitly declared in the topology.

Another important concept of ToMaTo is the flexibility in technology. The software architecture allows topology elements of different technologies. ToMaTo currently features virtual machines using KVM[10] and OpenVZ[11] technology, scripted elements based on Repy [15], switches and hubs based on Tinc[12] VPN technology, and tunnel endpoints using different solutions. Plans exist to include VLAN technology as well as LXC and VirtualBox virtual machines.

3.2. Key features

3.2.1. Editor

ToMaTo features an easy-to-use editor (Fig. 9) to create, manage, and control networking topologies. With this editor, users can configure their topology components and control them in the same intuitive interface. The editor also gives users direct access to their virtual machines using web-based VNC technology.

3.2.2. Link emulation

ToMaTo exposes the link emulation functionality of TC/NetEm [16] to its users, allowing them to configure latency, jitter, bandwidth limitation, and packet loss ratio for each direction of each link in their topology. These settings can be modified during a running experiment and become effective immediately.

3.2.3. Packet capturing

ToMaTo users can choose to capture and filter network packets on any link in their topology and later view them

[9] ToMaTo is online available: http://dswd.github.io/ToMaTo.
[10] KVM: Kernel-based Virtual Machine, http://www.linux-kvm.org.
[11] OpenVZ is a container-based virtualization technology, http://www.openvz.org.
[12] Tinc is a peer-to-peer virtual private network software, http://www.tinc-vpn.org.

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

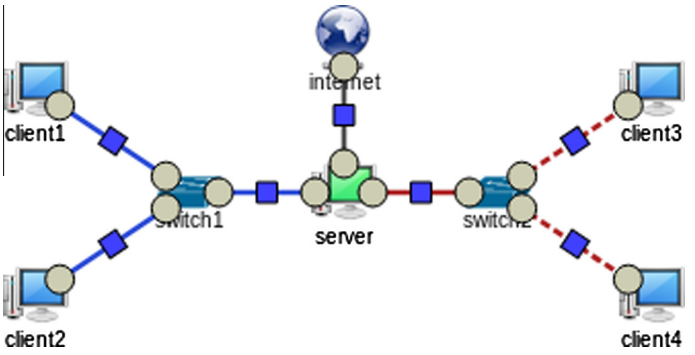

Fig. 8. Example topology in ToMaTo that shows multiple topology elements like virtual machines and networking devices.

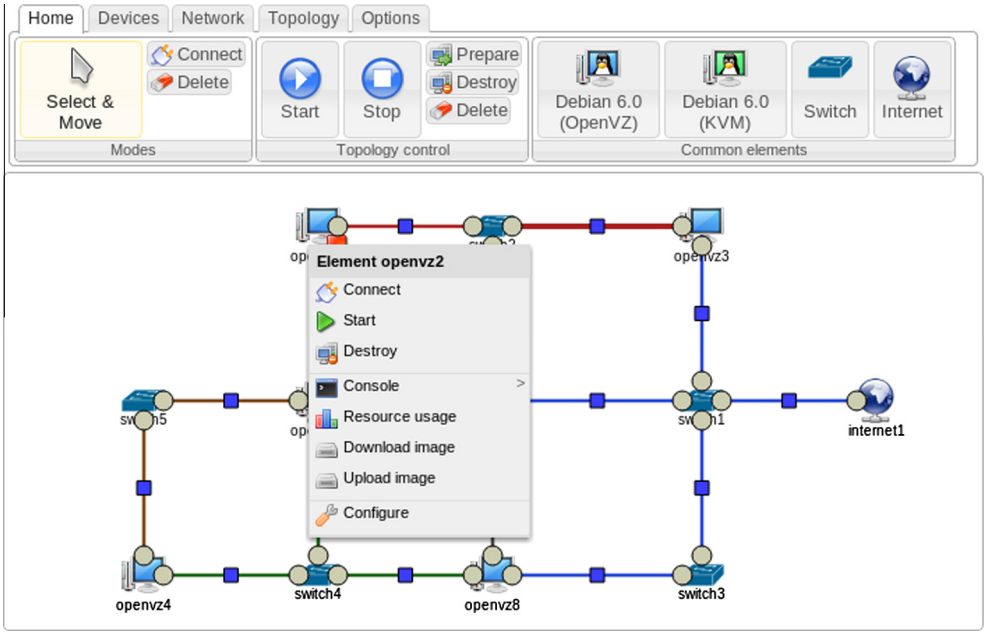

Fig. 9. The topology view of the ToMaTo editor. This web-based user interface allows users to edit, control, and use their topologies.

in packet analysis tools like Wireshark[13] or CloudShark.[14] Alternatively, ToMaTo also provides live-capture where copies of packets are sent directly to a running Wireshark instance where the user can see them without delay. As this functionality is independent of the operating system running inside the virtual machines, packet capturing can also capture packets that the operating system tries to hide.

3.2.4. Templates

ToMaTo provides several pre-installed templates for virtual machine based topology elements. This includes all common operating systems as well as some templates equipped with special software for network experiments, like a virtual switch (OpenVSwitch). Users can either choose these templates for their topology elements or provide their own images.

[13] Wireshark is a packet analyzer tool, http://www.wireshark.org.
[14] Cloudshark is an online packet analyzer tool, http://www.cloudshark.org.

3.3. Architecture

The architecture of ToMaTo consists of three major components as shown in Fig. 10.

3.3.1. Hostmanager

This component resides on each host in the ToMaTo topology cloud and controls its resources. It offers the resources and all of the supported technologies (e.g., KVM, OpenVZ), to authorized backends. It also monitors the topology elements and offers the monitoring data of each element to the backend that owns that specific element.

3.3.2. Backend

This central component manages all experiment topologies. It creates new elements on the hosts and forwards control commands to them. The backend is also responsible for setting up VPNs and tunnel endpoints on the hosts so that topology elements on different hosts can be

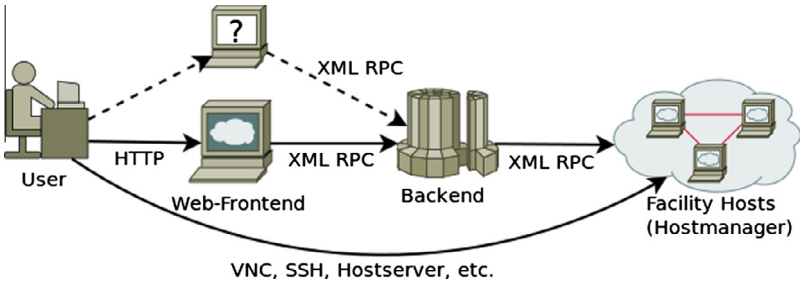

Fig. 10. The architecture of ToMaTo consists of the hostmanager cloud, the backend, and the frontends.

connected directly. The backend provides access for authorized users via an XML-RPC interface.

3.3.3. Frontends

The Frontends use the XML-RPC interface of the backend to provide a frontend to the users. Currently two types of frontend exist: the web-based frontend and the Python-based command-line frontend. Additional frontends are possible as the backend provides a generic interface and the frontend uses the authentication information of the user.

To allow an improved resource allocation algorithm, users are not allowed to select hosts for their components directly, instead they can only choose a site as is common in other cloud systems like EC2.[15] Final host selection is done by an algorithm in the backend that aims for load balancing as well as minimizing cross-site traffic.

The architecture allows hosts to be located at different sites without the need of a direct connection on layer 2. Virtual networks are used to span the layer 2 gaps and provide consistent layer 2 networks for each topology. All data-intensive operations have been carefully designed to work with direct connections between the hostmanager and the frontend, lifting as much load from the backend as possible. This way the testbed scales well and can work with enough hosts to handle topologies with thousands of components.

Resource efficiency is supported by the ability to use multiple virtualization technologies in one experiment. Users can choose whether they want full-featured but resource-hungry KVM machines or lightweight OpenVZ machines. With the scripted elements based on Repy technology, users can have topologies with hundreds of components that use close to no resources.

3.4. Monitoring

ToMaTo features a monitoring system that records resource usage and availability for all hosts in the topology cloud as well as individual topology elements. This information is used to identify problems on hosts and to assign components to hosts on a load balancing basis.

ToMaTo users can obtain usage records for their topologies as well as individual components as shown in Fig. 11. Platform providers might utilize the information for accounting and billing of resource consumption.

[15] Amazon elastic compute cloud, http://aws.amazon.com/ec2.

3.5. Evaluation

Evaluating a software design is a complex task. One approach is to compare the design goals or requirements with the actual capabilities of the resulting software. In the case of ToMaTo, the design goal was to support experiments and help researchers carry out their experiments. To evaluate ToMaTo's fulfillment of this goal, Section 3.5.1 first develops a classification of experiments in the German-Lab project, and then Section 3.5.2 outlines how ToMaTo supports these types of experiments. Section 3.5.3 takes a quick look at the efficiency and scalability of ToMaTo.

3.5.1. Types of experiments

The following experiment types have been identified in the German-Lab project.

3.5.1.1. Access layer experiments. Access layer experiments consider the lower networking layers and examine the usage of hardware for networking. Examples of this experiment class are mobile handover protocols. These experiments need access to real hardware, they often need to run custom operating systems (e.g. those with real-time support) and they need heterogeneous access technologies (3G, IEEE 802.11, Fiber, etc.). In most cases these requirements can only be fulfilled with custom testbeds, thus supporting this kind of experiment was not a design goal for ToMaTo.

3.5.1.2. Network layer experiments. This type of experiment considers the TCP/IP suite and its networking layers. Examples for this class are experiments with IPv6 extensions and TCP substitutes. This kind of experiment needs to run modified kernels. The resources that a single experiment needs are normally limited to a few devices but these devices have to be connected in complex network topologies with link emulation.

3.5.1.3. Protocol/Algorithm experiments. Experiments with protocols or algorithms work on top of the network layer and usually consider new approaches for lager networks. Nearly all peer-to-peer experiments fall in this category. These experiments need a high number of nodes but usually no hardware or kernel access. They only need simple network topologies with link emulation.

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

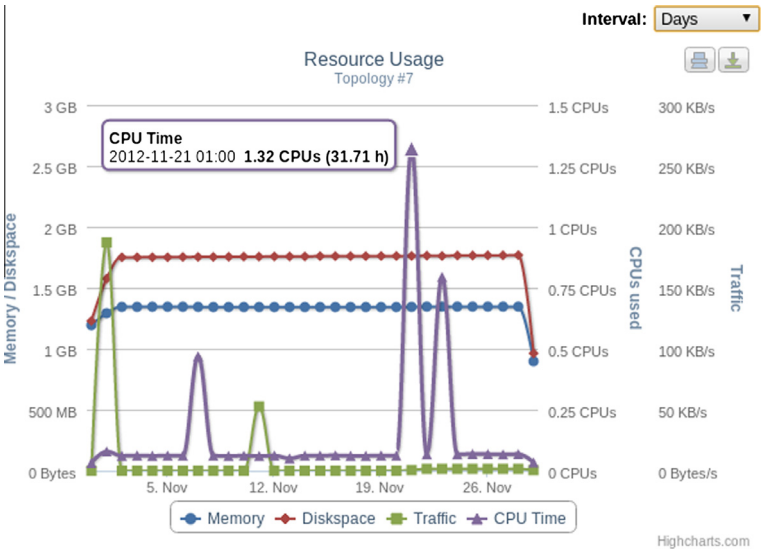

Fig. 11. An example view of usage statistics that are recorded by ToMaTo for every topology and all their elements.

3.5.1.4. Proprietary application experiments. Experiments using proprietary software cannot be modeled because of unspecified or unpublished behavior. Examples of this are Skype and Windows. Experiments with this software often need special operating system environments including Internet access and link emulation. In turn, these experiments normally do not need big or complex network topologies.

Experiences of the German-Lab experimental facility show that most experiments can be categorized fairly well with this scheme. A few experiments have two experiment classes, and thus have requirements of both classes. The resource requirements of the classes are very heterogeneous but in general, trade-off between more resource access and access to more resources, i.e. in general experiments either need a high number of (lightweight) resources with restricted access or a small number of resources with full access.

3.5.2. Experiment support in ToMaTo

ToMaTo has been designed to support all experiment classes identified in Section 3.5.1 except for access layer experiments because these experiments need a specialized ExpFac depending on the access technology. Wisebed [17] and DES testbed [18], for example are specialized experimental facilities for sensor and wireless networks.

Network layer experiments can be done easily in ToMaTo using KVM devices and virtual switches. KVM devices offer the flexibility in kernel choice and modification required by this experiment class. Switched networks provide connectivity on layer 2 so that any TCP/IP modification or substitute can be examined. Even very complex topologies can be easily designed with the commandline frontend. Capturing and downloading network traffic can be very handy for this kind of experiment.

Protocol/Algorithm experiments are supported in ToMaTo using OpenVZ devices. Since OpenVZ devices are very lightweight, a high number of devices can be used in topologies. Using an Internet connector, external resources like PlanetLab nodes can be included in the experiment. Using the upload/download image feature, users can prepare a device image once and upload it to all of their devices. Network traffic capturing and link emulation can be used to debug the protocols.

ToMaTo also supports proprietary application experiments using KVM devices and Internet connectors. KVM devices can run nearly all x86 operating systems including Windows and BSD, therefore users can build custom environments for their legacy applications. The proprietary application can communicate with external services using the Internet connector. Traffic of the proprietary application can be captured and analyzed using specialized tools without any operating system support.

3.5.3. Efficiency and scalability

With ToMaTo, users can choose between OpenVZ and KVM virtual machines. This way, users can get the level of access that is needed for their experiments and still use as few resources as possible. A single cluster node can handle up to 250 OpenVZ devices and up to 50 KVM devices, depending on device usage. The networking components only pose a very small overhead and can handle connections with over 100 Mb/s without influencing them.

ToMaTo hosts are based on an existing operating system and only need small changes that have been bundled as a software package. This has the advantage that operating system support and security updates are available from the original sources and do not have to be provided by the ExpFac administrators. As the ToMaTo backend only controls the hosts and only contacts them when users change their topologies, the back-end can handle many host nodes making the solution very scalable.

ToMaTo can be used to create experimental facilities with distributed hosts. Limitations in network emulation apply since the resulting link characteristics are a combination of real and emulated link properties. ToMaTo offers

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

long-term link statistics so that users can plan their experiments accordingly.

ToMaTo allows its users to design, manage and control networking topologies for use in network research. ToMaTo fits for a wide range of experiments identified as common in the context of the German-Lab project. The design of the ExpFac software offers efficiency and scalability. ToMaTo is not bound to German-Lab and can easily be used to build similar experimental facilities.

In the German-Lab ExpFac currently 35 of 182 hosts are ToMaTo-enabled. The goal is to increase this number by integrating nodes from international projects and thereby increase the usability of the testbed. Early plans exist to integrate support for OpenFlow hardware and software to allow even more complex network topologies.

4. Experiments in German-Lab

The following sections describe two German-Lab experiments that used ToMaTo.

4.1. Experiment: attack mitigation in VoIP systems

ToMaTo has been successfully used for an experiment in the context of German-Lab Deep[16] that focuses on attack mitigation in VoIP systems. In this scenario, an attacker initiates malicious calls that must be detected by the distributed VoIP system and blocked at the nearest gateway using cross-layer components.

Fig. 12 shows the topology used for this experiment. It contains several end systems and smart middle boxes using KVM technology. In this scenario, it was very important to explicitly define the links in the topology and to be able to route traffic over programmable middle boxes.

This experiment was central to the whole project and brought together different software components. A demonstration at the Euroview conference successfully showed the interoperability of these components using ToMaTo [19].

4.2. Experiment: malware analysis

A live demonstration at Euroview 2011 [20] showed the unique features of ToMaTo that allow users to analyze the network behavior of malicious software. In this demonstration, the network behavior of an older Internet worm was analyzed using ToMaTo.

Fig. 14 shows the topology that has been used in the demonstration. It clearly shows that the Internet in the upper left corner is not connected to the rest of the topology, so all the other components of the topology had no access to the Internet and the worm could safely be analyzed without the risk of escape. In most other networking testbeds, this would not be possible, because there is an implicit network link to the Internet or to core components of the testbed.

To analyze the malware, a virtual machine using KVM and running Windows XP was first configured. A backup of that machine was downloaded to be able to replay the infection later. Using the packet capturing feature on the outgoing link and scripted network components it was possible to analyze the network behavior of that software without relying on tools that run inside the infected machine.

As a result of the analysis as shown in Fig. 13, the control server was identified as a probably hacked name server. The protocol used by the control server was identified as IRC and even the channel name was captured. With this information it would be possible to contact the hoster of the control server to shut it down, or to try a man-in-the-middle attack on the control protocol for further analysis.

5. International federation

Early in the G-Lab project it became clear that its success would depend on its international embedding into related projects. Therefore from it's early beginning, G-Lab had close relationships with the FIRE initiative in Europe and the GENI projects in the US and others.[17] The current international connectivity of G-Lab in the international network infrastructure can be seen in Fig. 15 where the G-Lab site at Kaiserslautern is connected by a 1 Gb/s L2 tunnel to the Starlight facility with a path to the International Center for Advanced Internet Research (iCAIR) of Northwestern University, Chicago.

As one of the first efforts in this direction, a node for the GpENI project [21] has been installed at Kaiserslautern. GpENI offers an international programmable network testbed and is built upon a multi-wavelength fiber interconnection between the four GpENI universities within the GPN (Great Plains Network), with direct connection to the Internet 2 backbone and L2TPv3 tunnels to Europe and Asia. GpENI was not directly federated with G-Lab, but researchers from G-Lab could get access to this infrastructure upon request.

Another federation was organized with the GENIcloud and TRANScloud projects [22] that considers a high-performance cloud architecture across multiple administrative domains. Here the flexibility of the G-Lab infrastructure could be used in the sense that it was easy to set up hosts with a custom boot image (as described in Section 2.7.3) where the slice-based federation architecture (SFA) layer was implemented over OpenStack and Eucalyptus. Based on this infrastructure, federation was demonstrated at the GENI engineering conference GEC10 in Puerto Rico in 2011, and subsequently at other conferences around the world. In this federation the HP Palo Alto site runs OpenStack under the SFA; UCSD, Northwestern and TU Kaiserslautern run PlanetLab under the SFA using the MyPLC cluster controller.

In July 2012 G-Lab demonstrated another federation effort with the "slice around the world" consortium at GEC14 in Boston[18] which showed the potential of designing and implementing worldwide environments consisting of global computational clouds based on programmable networks.

[16] "Deepening G-Lab for Cross-Layer Composition", http://www.g-lab-deep.de.

[17] http://www.german-lab.de/collaborations/.

[18] http://groups.geni.net/geni/wiki/GEC14Agenda/TuesdayPlenary.

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

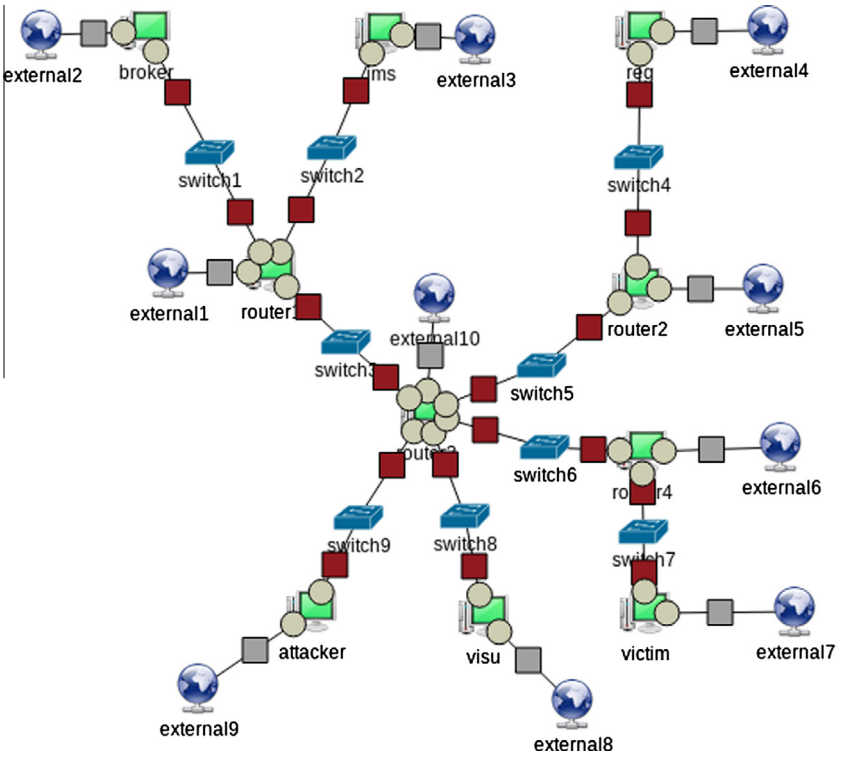

Fig. 12. The topology used by a DeepG experiment of attack and defense mechanisms in distributed VoIP scenario. All components are connected by smart routers.

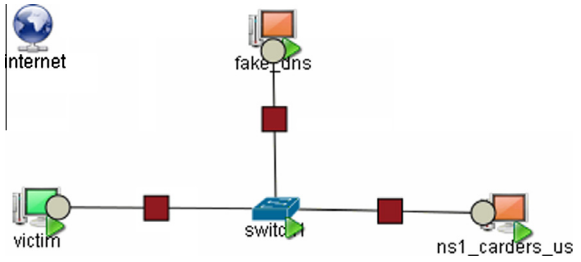

Fig. 13. The results of the malware analysis reveal that a hacked nameserver is used as a master server that controls the victims using IRC.

There were two different application cases: The first created a scientific visualization for nanotechnology by using a virtual viewing scope for invisible objects. Here researchers from nanotechnology could customize the infrastructure specific to their application, in other words the person at the edge, dynamically creates a customized network and changes it in real time as he works with that application. Another application was "the greenest city of the world". This application, conducted by University of Victoria, Canada, calculates the greenness of cities based on pictures from Landsat satellites in the visible, near-infrared, short wave, and thermal infrared regions of the electromagnetic spectrum.

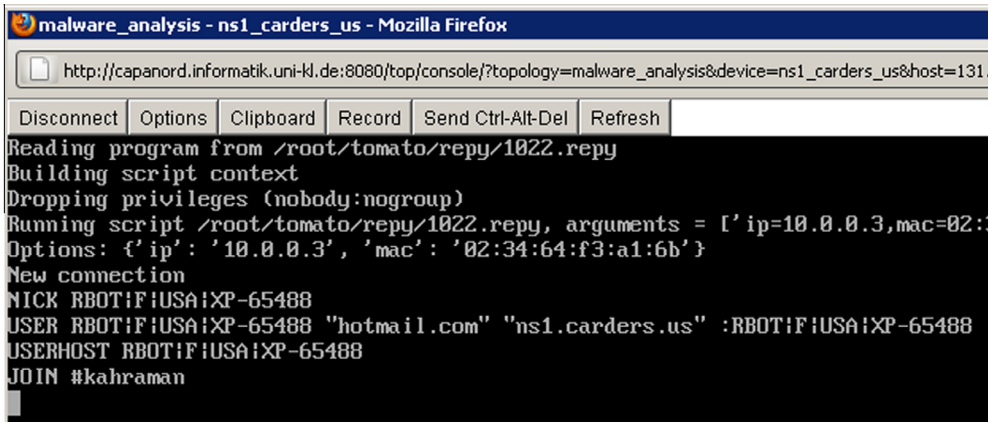

Fig. 14. The topology used for a malware analysis experiment. This topology is not connected to the Internet for safety reasons.

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

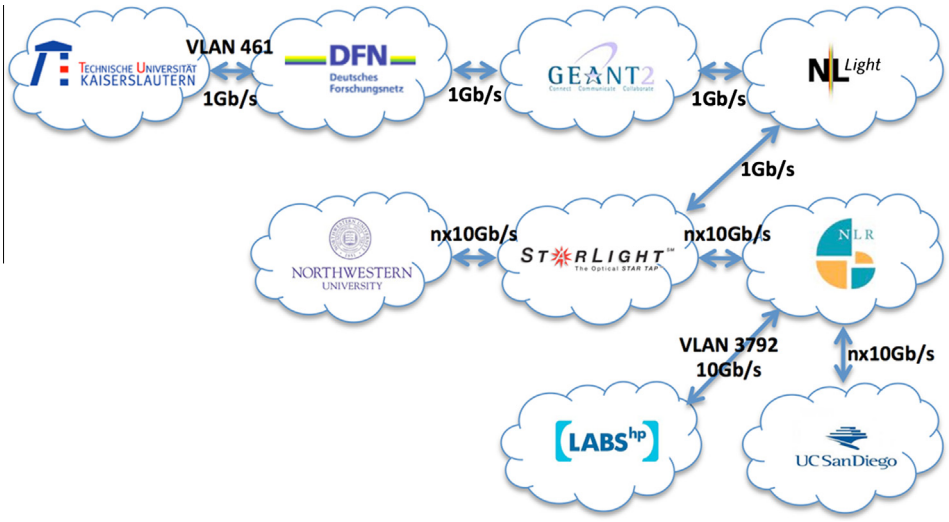

Fig. 15. International network connectivity in which G-Lab at Kaiserslautern is embedded.

These satellites provide digital pictures of almost all larger cities, and based on the different radiation intensity of the plants there, the greenest city of the world can be calculated. This calculation can be done on the fly by distributing the data sets in the cloud.

6. Conclusion and outlook

In this contribution we gave a brief introduction of the G-Lab project as a whole, with a special focus on G-Lab's ExpFac and its control frameworks with respect to its close relation to the functionality required by the research projects. Moreover, we presented the control framework ToMaTo (Topology Management Tool) that was built to fulfill requirements of various research projects and became the most popular control framework within G-Lab.

The German-Lab platform has been developed for nearly five years and is still operating. At this point, an evaluation of some decisions can be done. One lesson learned is that virtualization is a key technology. When choosing virtualization techniques one faces the trade-off between level of abstraction and level of control. In G-Lab we had very few requests to access real hardware, but there were needs to run modified kernels as well as running many (>100) systems in parallel for one experiment. The ToMaTo approach integrates different virtualization technologies and levels of resource access. ToMaTo offers three virtualization technologies that work on different levels of abstraction: script sandboxes (Repy), container virtualization (OpenVZ), and full virtualization (KVM). This provides maximal flexibility for experimenters and thus increases the usage of the platform. Another observation is that monitoring is very helpful for the management and operation of such a facility. For experimenters it is important to have information about the load of the platform and administrators need tools to identify misbehaving experiments. ToMaTo can do load balancing when deploying a new VM automatically based on monitored system load. As such, monitoring should not be an additional component, but it must be integrated into the architecture and be developed as early as possible. Furthermore we have seen that the ability to control the characteristics of network links has been frequently used by experiments. Explicit degradation of link capabilities in addition to monitoring real network links seamed to be a practical approach and sufficient for experiments conducted in G-Lab.

The different sites of the G-Lab ExpFac are connected by using the Internet connectivity of the participating universities. Within five years we have observed two incidents of abnormal high bandwidth usage due to failures in experiments. If the G-Lab Platform was opened to a much broader group of users then dedicated links should be used or at least explicit bandwidth limitations should be implemented. For some scenarios there was the request to integrate specific real hardware (e.g., an open flow switch) into a ToMaTo topology. Currently this is only possible with manual configuration but we plan to support integration of real hardware by ToMaTo in near future. This will be useful for example for teaching. It is also considered to enable the definition of link profiles (including scripts) in order to emulate the real behavior of wireless and mobile links.

The G-Lab ExpFac and the ToMaTo control framework were developed during the five years of the G-Lab project and reached a mature status. Although the overall project ended in late 2012, the ExpFac will be operating at least for two more years, as was agreed between the six sites and the funding agency BMBF. Therefore the ExpFac is available for further research projects. A first project which request the usage of G-Lab's ExpFac is the MAKI project from University of Darmstadt that was started early 2013 as a collaborative research center funded by the German Research Foundation (DFG). Within this project there is some funding for extending the G-Lab ExpFac especially in the direction of software defined networking (SDN) and increasing the numbers of available nodes.

Currently there is a discussion about building an international collaboration to simplify federated interoperation among different experimental facilities and provide experimenters (beyond computer science) access to system and

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023

connectivity resources for use in multi-continent experiment designs.

References

[1] R. Khondoker, B. Reuther, D. Schwerdel, A. Siddiqui, P. Müller, Describing and selecting communication services in a service oriented network architecture, in: Kaleidoscope: Beyond the Internet? – Innovations for Future Networks and Services, ITU-T, 2010, pp. 1–8.

[2] R. Khondoker, E. Veith, P. Müller, A description language for communication services of future network architectures, in: 2011 International Conference on the Network of the Future (NOF), 2011, pp. 68–75, http://dx.doi.org/10.1109/NOF.2011.6126685.

[3] H. Wippel, T. Gamer, C. Faller, M. Zitterbart, Hierarchical node management in the future internet, in: 2011 IEEE International Conference on Communications Workshops (ICC), 2011, pp. 1–5, http://dx.doi.org/10.1109/iccw.2011.5963590.

[4] A. Feldmann, L. Cittadini, W. Mühlbauer, R. Bush, O. Maennel, Hair: hierarchical architecture for internet routing, in: Proceedings of the 2009 Workshop on Re-architecting the Internet, ReArch '09, ACM, New York, NY, USA, 2009, pp. 43–48, http://dx.doi.org/10.1145/1658978.1658990.

[5] O. Hanka, G. Kunzmann, C. Spleiss, J. Eberspacher, A. Bauer, Hiimap: hierarchical internet mapping architecture, in: First International Conference on Future Information Networks, ICFIN, 2009, pp. 17–24, http://dx.doi.org/10.1109/ICFIN.2009.5339608.

[6] H. Schotten, Wireless access optimization for future networks, in: Proceedings of 9th Würzburg Workshop on IP: Visions of Future Generation Networks (EuroView2009), 2009.

[7] W. Fritz, O. Hanka, Smart card based security in locator/identifier-split architectures, in: 2010 Ninth International Conference on Networks (ICN), 2010, pp. 194–200, http://dx.doi.org/10.1109/ICN.2010.39.

[8] D. Schwerdel, A. Siddiqui, B. Reuther, P. Müller, Composition of self descriptive protocols for future network architectures, in: 35th Euromicro Conference on Software Engineering and Advanced Applications. SEAA '09, 2009, pp. 571–577, http://dx.doi.org/10.1109/SEAA.2009.75.

[9] D. Schwerdel, D. Günther, R. Henjes, B. Reuther, P. Müller, German-lab experimental facility, Future Internet-FIS 2010 (2010) 1–10.

[10] A. Kovári, P. Dukan, Kvm & openvz virtualization based iaas open source cloud virtualization platforms: Opennode, proxmox ve, in: 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics (SISY), IEEE, 2012, pp. 335–339.

[11] J. Loope, Managing Infrastructure with Puppet, O'Reilly, 2011.

[12] L.L. Peterson, T. Roscoe, The design principles of planetlab, Oper. Syst. Rev. 40 (1) (2006) 11–16.

[13] D. Schwerdel, D. Hock, D. Günther, B. Reuther, P. Müller, P. Tran-Gia, ToMaTo – a network experimentation tool, in: 7th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2011), Shanghai, China, 2011.

[14] D. Schwerdel, J. Götze, B. Reuther, P. Müller, Testing mobile apps in the tomato testbed, in: Proceedings of 11th Würzburg Workshop on IP: Visions of Future Generation Networks (EuroView2011), 2011.

[15] J. Cappos, A. Dadgar, J. Rasley, J. Samuel, I. Beschastnikh, C. Barsan, A. Krishnamurthy, T. Anderson, Retaining sandbox containment despite bugs in privileged memory-safe code, in: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS '10, ACM, New York, NY, USA, 2010, pp. 212–223, http://dx.doi.org/10.1145/1866307.1866332.

[16] S. Hemminger et al., Network emulation with netem, in: Linux Conf Au, Citeseer, 2005, pp. 18–23.

[17] T. Baumgartner, I. Chatzigiannakis, M. Danckwardt, C. Koninis, A. Kröller, G. Mylonas, D. Pfisterer, B. Porter, Virtualising testbeds to support large-scale reconfigurable experimental facilities, in: Proceedings of EWSN – 7th European Conference of Wireless Sensor Networks, 2010, pp. 210–223.

[18] DES-Testbed A Wireless Multi-Hop Network Testbed for Future Mobile Networks.

[19] M. Kleis, C. Varas, A. Siddiqui, P. Müller, I. Simsek, M. Becke, D. Hoffstadt, A. Marold, E. Rathgeb, C. Henke, J. Müller, T. Magedanz, Cross-layer security and functional composition for a future internet, in: Proceedings of 11th Würzburg Workshop on IP: Visions of Future Generation Networks (EuroView2011), 2011.

[20] D. Schwerdel, B. Reuther, P. Müller, Malware analysis in the tomato testbed, in: Proceedings of 11th Würzburg Workshop on IP: Visions of Future Generation Networks (EuroView2011), 2011.

[21] J.P. Sterbenz, D. Medhi, B. Ramamurthy, C. Scoglio, D. Hutchison, B. Plattner, T. Anjali, A. Scott, C. Buffington, G.E. Monaco, et al., The great plains environment for network innovation (gpeni): a programmable testbed for future internet architecture research, in: Testbeds and Research Infrastructures. Development of Networks and Communities, Springer, 2011, pp. 428–441.

[22] A. Bavier, Y. Coady, T. Mack, C. Matthews, J. Mambretti, R. McGeer, P. Müller, A. Snoeren, M. Yuen, Genicloud and transcloud, in: Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit, ACM, New York, NY, USA, 2012, http://dx.doi.org/10.1145/2378975.2378980.

Dennis Schwerdel is a researcher at the University of Kaiserslautern. He received his Diploma in computer science in 2008 and joined the "Integrated Communication Systems Lab. (ICSY)" in 2009. His current research interests focus on distributed systems, peer-to-peer technology, future internetworking architectures, and network testbeds. He is the main developer of the Topology Management Tool and leader of the administrator team of the German-Lab experimental facility.

Bernd Reuther is head of the network department in the regional computing center at University Kaiserslautern. He received his PhD in computer science 2011 at the University of Kaiserslautern. Up to 2011, he was leading several research projects in the "Integrated Communications Systems Lab. (ICSY)". His current research interests include distributed systems, SDN, and future internetworking architectures.

Thomas Zinner is heading the NGN research group "Next Generation Networks" at the Chair of Communication Networks in Würzburg. He finished his PhD thesis on "Performance Modeling of QoE-Aware Multipath Video Transmission in the Future Internet" in 2012. His main research interests cover the performance assessment of novel networking technologies, in particular software-defined networking and network function virtualization, as well as network-application interaction.

Paul Müller is professor for computer science and director of the regional computing center at the University of Kaiserslautern. He received his PhD from the faculty of mathematics at the University of Ulm in the field of statistics. Thereafter, he was responsible for various research projects and the development of a statewide computer network in Germany. His current research interests are mainly focused on distributed systems, future internetworking architectures and service-oriented architectures. His research group "Integrated Communications Systems Lab. (ICSY)" within the department of computer science at the University of Kaiserslautern is aiming at the development of services to implement integrated communication within heterogeneous environments especially in the context of the emerging

discussion about Future Internet. This is achieved by using concepts from service-oriented architectures (SOA), Grid technology, and communication middleware within a variety of application scenarios ranging from personal communication (multimedia) to ubiquitous computing.

Phuoc Tran-Gia is professor and director of the Chair of Communication Networks, University of Würzburg, Germany. He is also Member of the Advisory Board of Infosim (Germany) specialized in IP network management products and services. He is also cofounder and board member of Weblabcenter Inc. (Dallas, Texas), specialized in Crowdsourcing technologies.
Previously he was at academia in Stuttgart, Siegen (Germany) as well as at industries at Alcatel (SEL) and IBM Zurich Research Laboratory. He is active in several EU framework projects and COST actions. He was coordinator of the German-wide G-Lab Project "National Platform for Future Internet Studies" aiming to foster experimentally driven research to exploit future internet technologies.
His research activities focus on performance analysis of the following major topics: Future Internet & Smartphone Applications; QoE Modeling & Resource Management; Software Defined Networking & Cloud Networks; Network Dynamics & Control.

Please cite this article in press as: D. Schwerdel et al., Future Internet research and experimentation: The G-Lab approach, Comput. Netw. (2014), http://dx.doi.org/10.1016/j.bjp.2013.12.023