A survey of software tools for the creation of networked testbeds

Christos Siaterlis
Inst. for the Protection and Security of the Citizen
Joint Research Centre
Via E. Fermi 2749, 21027 Ispra (VA) Italy
e-mail: christos.siaterlis@jrc.it

Marcelo Masera
Inst. for the Protection and Security of the Citizen
Joint Research Centre
Via E. Fermi 2749, 21027 Ispra (VA) Italy
e-mail: marcelo.masera@jrc.it

***Abstract*—The development of testbeds for networking research has been driven by the need for experimentation with complex systems, like the Internet, that simplistic simulators fail to reproduce. Recently, networked testbeds seem to head towards more advanced, flexible and automated experimental platforms mainly as the results of many projects and research initiatives in the field of Future Internet architectures. Although numerous publications can be found, most of them refer to prototypes and work in progress rather than to publicly available software that is ready to be widely used for the creation of such testbeds. The first contribution is the development of a framework that can be used to capture the main features of the available software. The second contribution is a literature review of state-of-the-art tools and their comparison under common criteria. This systematic analysis allows other researchers to make informed decisions about the usability of already available tools and decrease the initial cost of developing a new testbed, leading to an even wider use of such platforms. This paper provides the reader with a useful reference list of readily available software to choose from while designing or upgrading a research infrastructure, laboratory or experimentation facility.**

***Keywords*—distributed test bed; emulation ; network research;**

I. Introduction

The development of advanced testbeds for networking research and in particular research for Future Internet protocols and architectures is a recent trend that becomes evident after consideration of the amount of related projects and research initiatives [1]. Most notable are: the 'Global Environment for Network Innovation' or GENI [2] as the most important initiative with a multi million budget in the US, the ICT FIRE (Future Internet Research & Experimentation) initiatives [3] and the FEDERICA project [4] in Europe. A researcher that will try to familiarize himself with the topic of experimental platforms for networking research will face tons of acronyms and a huge list of relevant projects. The terms testbed and "experimental platforms" do not have clear definitions and are frequently used interchangeably although they might have different interpretations especially in size and sophistication. The lack of precise definitions has often introduced confusion in the related literature. In our point of view the fidelity or "level of realism" of an experiment often determines which description is most suitable for the specific setup: simulation, emulation or real testbed [5]. As this labeling might lead to confusions we will try to avoid it and characterize an *experimental platform as a combination of hardware and software based on an architectural design that enables the researcher to conduct experiments using components that provide different levels of realism or abstraction* (e.g., real or virtual hosts, simulators, traffic generators, mathematical models).

Recently, experimental platforms seem to gain wider use in other subfields of networking research as well e.g., in security research[6]. The driving force is the need not only for theoretical but also empirical security research that is based on more solid and compelling evidence and will produce useful results that can be promptly used to strengthen our Critical Information Infrastructures. The field of Internet security in particular, is often handled in a non-systematic way [7]. Furthermore:

- new developments in the field of security are often presented as "hacks", without detailed analysis of prerequisites and consequences;
- metrics about the security of a systems, system-of-systems or networks do not exist as a shared basis among researchers and practitioners [8];
- security-relevant data that can be used for research are scarce (mostly because of their sensitive nature);
- experimental platforms are often oversimplified and of limited scale and cannot accurately simulate real complex Internet environments;
- experiments are designed ad-hoc, without a methodology and a clearly stated approach for setting up testing campaigns, measuring significant variables and examining their outcome. The consequence is that the results are hardly reproducible by other researchers.

This strong need for networking research that follows rigorous scientific methods and produces provable results that are closely bound to reality by proving and disproving hypotheses drives us to the development of new experi-

mental platforms. This need is further analyzed by Neville and Li in [9]. For the time being, one of the barriers one has to overcome while designing, extending or developing a new testbed is the required effort to review the available approaches and software. Our paper aims to lower this initial barrier.

The excess of available information and the fact that most sources refer to prototypes and work in progress rather than published software that is ready for wider use, are two factors that magnify the need for a structured and practical review of available software. A survey of emulation related products has been published by Rimondini et al. [10], a second one about tools that can be used to develop, test or utilize routing protocols [11] was carried out by Oliver Bonaventure and finally the work of Volvnkin et al. [12] deals with general recommendations and architectural issues regarding the development of a testbed for information security experiments. To the authors's knowledge until today a study that compares under common criteria the software that is available for the development of distributed experimental platforms for networking research does not exist. Our work focuses on publicly available software that can be used for networking research and intentionally excludes platforms :

- that share computational resources (e.g., GRIDS);
- that focus only in simulation (like Simgrid [13]) ;
- that are specific to wireless or sensor networks (like signetLab [14]);
- that run on a single computer (like Marionnet [15], IMUNES [16] and Netkit [17]) and aren't a distributed testbed;
- that use custom hardware (like the Open Network Laboratory [18]) rather than using Commercial, off-the-shelf (COTS) components.

The paper is structured as follows. We begin in Section II with a presentation of an experiment's life cycle on an experimental platform. We continue in Section III with the proposed framework that can capture the features of a software suite for the creation of such platforms. In Section IV we provide the reader with a useful reference list of readily available software that are compared under common criteria. This list can be extremely useful while designing or upgrading a research infrastructure, laboratory or experimentation facility. In Section V, some significant ideas and approaches that are part of related projects will be shortly described. Finally in Section VI we summarize the main conclusions of our study.

II. Experiment lifecycle

In this paper an experimental platform is understood as a combination of hardware, software, architectural and operational policies that form a facility for conducting experiments with computer networks. In order to analyze further the features of experimental platforms it is useful to model the life of an experiment on top of an experimental platform. We define six phases of an experiment's lifecycle (Figure 1) - similar to the work of Guiller et al. [19] and Miyachi et al. [20].

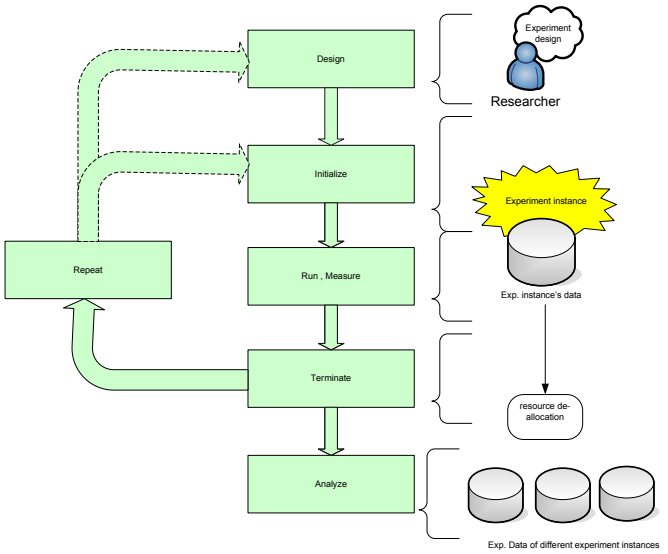

Figure 1. An experiment's lifecycle

1) **Design phase**. In this phase the researcher designs an experiment according to a scenario. The experiment has to be defined: a) structurally by using different building blocks such as hosts, network devices, links etc; b) functionally by defining the main and background processes that recreate the experiment environment as well as the variables of interest that have to be captured; c) procedurally by specifying the phases and steps of the experiment, and the conditions and criteria for launching and terminating each one of them. An important element for the procedural definition of an experiment is the notion of an experiment timeline (relative time).
2) **Initialization phase**. Before the experiment runs, all components that were defined in the design phase have to be initialized. The completion of this process involves the instantiation and configuration of all building blocks.
3) **Execution and Measurement phase**. During this phase the experiment is running and the platform is triggering events and actions according to a predefined schedule (experiment design). Each experiment that runs on the platforms should be identifiable with a unique ID that defines a specific instance/run of an experiment's design. A single experiment design can be instantiated and run many times in order to perform statistical validation of the outcome. During the experiment execution, measurement processes capture all data that will be valuable to the researcher during the analysis phase. The collected data should be stored and labeled with the specific experiment instance's unique ID into a repository.
4) **Termination phase**. When an experiment's in-

stance is meeting some termination criteria (e.g., amount of time that has passed) the instance is considered finished and a termination process is initiated. All components that are part of the experiment have to be brought into a clean state without retaining history (for example by shutting them down).

5) **Repetition phase**. An experiment might be repeated several times with or without changes in its design. Repeating an experiment without changes serves statistical validation of the outcome whereas the repetition of an experiment by changing one or more controlled variables (not only in a strict term e.g., a detection threshold but also in a wider sense e.g., the network topology or a host's configuration) can be used to conduct sensitivity analysis. Automation of this phase is very important so that the researcher can conduct repeating experiments efficiently without manual interaction.
6) **Analysis phase**. After all data has been collected the researcher will be able to analyze it without interaction with the experiment platform.

III. Features of software for experimental platforms

Ideally, an experimental platform for networking research would support the execution of complex, large scale and even disruptive experiments using rigorous scientific methods. The desired characteristics - features of such platforms are many and can be realized with different means; for example with the use of software, hardware or even organizational measures. It is obvious that every single feature can influence the overall usefulness and simultaneously implementing all of them is definitely non-trivial. One of the main reasons is that these features are not independent and design choices regarding one of them can influence the available implementation options of a different feature. By extending previous work of the DETER project [21] and Masera et al. [22], we first identify a set of the most important basic features and then present other more sophisticated characteristics that build upon the basic features and are called compound features. The features are labeled as Fx where $x \in N$ and discussed one by one in following Sections. We provide also a map of the dependencies between basic and compound features (Figure 2) that could serve as a "scorecard" for evaluating different approaches for the creation of a testbed. This framework can be used also to provide an overview of desired features and demonstrate the complexity of the development process of a new experimental platform.

A. Basic features

F1. Control of the experiment's environment is one of the most important attributes of the scientific approach and enables the researcher to analyze how a hypothesis is influenced from dependent and independent variables. This does not imply that we cannot use random or stochastic processes in an experiment as long as they are of a controlled nature (e.g., statistically modeled).

F2. An experiment clock is a necessary feature for every experimental platform as it can provide a solid reference point to characterize the occurrence of events (event scheduling) and the measurement of various variables. The synchronization of the internal clock of different devices with the master experiment clock is a non trivial and important task.

F3. Separation of control, measurement and experiment planes. Measurements should not interfere with the experiment because they might alter the experiment's outcome. Preferably control and measurement planes should be differentiated as well to maximize measurement accuracy.

F4. Storage facilities are needed to store the description of an experiment and the data that was collected during its execution (measurements). Storage facilities should provide secure access to the data and support backup and restore. An important aspect of choosing storage facilities are the supported data structures. Relational databases are not the only solution and specialized data structures like Round Robin Databases could be more efficient for specific measurements like time-series [23].

F5. The use of standard Application Programming Interfaces (APIs) by the software that supports an experimental platform is not only required as a good design and programming practice but because it is crucial for allowing researchers to extend its functionality in custom ways. The ability to automate tasks on an experimental platform is also dependent on the existence of APIs that are exposed to the user.

F6. Heterogeneity of technologies that can be included in an experiment (e.g., components that range from custom hardware to simulators) is an essential attribute of a platform in order to permit experiments with future technologies. The platform should not restrict the researcher to specific hardware and software vendors and should allow mixed configurations in order to resemble real world scenarios. This could be achieved in one extent with the use of virtualization technologies and standards.

F7. Clean reconfiguration means that the initial state of the experiment is build from the scratch without past experiments influencing it. This functionality is imperative for reliable repetitive experiments. No hidden state should be kept by the components of the experiment.

F8. Virtualization is important for scalability and can provide some independence from physical resources helping thus to automation, rapid reconfiguration and topology flexibility. Additionally it can lower operational and capital expenses. In this context virtualization is the ability to reuse a single physical resource. The extent of virtualization can differ significantly between two approaches. For example host virtualization can be implemented in a way where a single host running a specific Operating System (OS) can act as being multiple hosts with the same OS

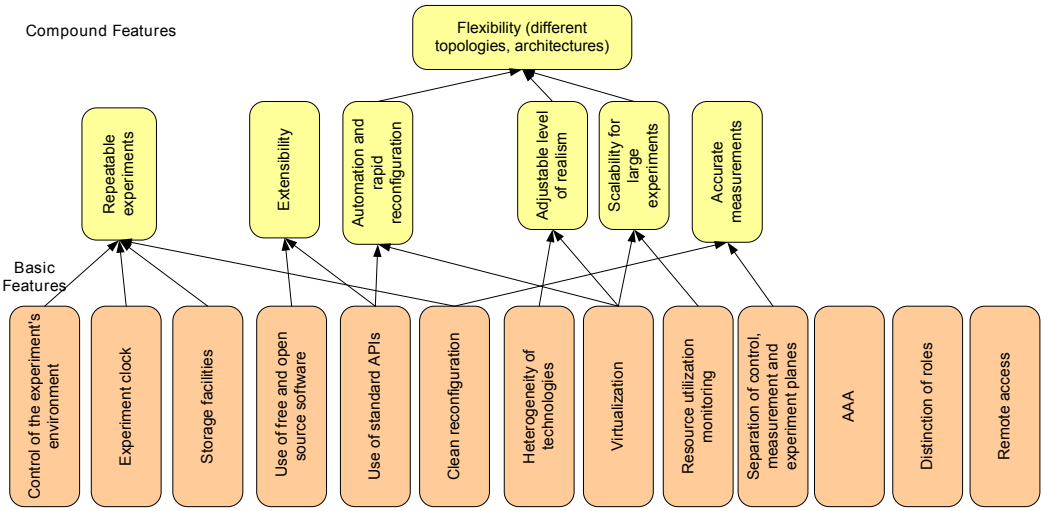

Figure 2. Dependencies of compound features from basic features.

(like in FreeBSD jails or Solaris zones) or can act as being multiple hosts with different OS (like VMware). Recently, besides host virtualization the concept of router virtualization emerged as a new approach; although software routers date back many years. Typical examples are logical routers by Cisco and Juniper and the Openflow switch [24].

F9. Resource utilization monitoring deals with the need of knowing the status of the available resources (in real time and historically). Such information is crucial for capacity planning and accounting purposes. The level of detail of the utilization information can differ significantly ranging from simple CPU utilization to per process details.

F10. Use of free and open source software is important because it can foster deeper understanding of underlying mechanisms and principles. Open source software can be reviewed and altered by other researchers that want to collaborate. Finally, academic researchers often consider that the free distribution of software helps to create a community and to reach critical mass, needed to support and extend a software suite.

F11. AAA - Authentication, Authorization and Accounting is essential if multiple users and especially from different organizations have access to an experimental platform. The importance of such security related functionality, e.g., isolating experiment resources and prohibiting users to view each other's data, is due to the fact that the experiments might involve the handling of sensitive data like network traffic captures. AAA functions are also related to the storage facilities as access to the stored data should be controlled.

F12. Distinction of roles. Access to an experimental platform could be restricted to private users but in some cases it might be beneficial to open up to a wider community. These issues can be defined in "Usage and Operation policies" that might differentiate the users and assign different roles and rights. For example requests to use an experimental platform that come from external users might have to get reviewed and prioritized before granted access.

F13. Remote access can form the basis of deeper and wider collaboration with researchers throughout the world. A first step is to support remote access to experimental data. An extension would involve the possibility to remotely control and monitor an experiment.

B. Compound features

On top of the basic features more advanced features and functionality can be built:

F14. Repeatable experiments require a controlled environment but to achieve them the researcher has to define clearly and in detail the experiment's initial and final state as well as all events in between these two states. These states and events form an experiment scenario. To reproduce a previously stored experiment scenario the researcher should be able to setup the experimental platform in the initial state and trigger all necessary events in the right order and time of occurrence.

F15. Extensibility can be viewed as the ability to adapt to future needs and requirements. To allow future extensions an experimental platform should have a modular design with clearly defined interfaces. In addition following standards and best practices (for measurements and setup) and the use of open source software improves extensibility.

F16. Automation and rapid reconfiguration aim to ease the researcher in his work and enable a more efficient use of the available resources. If 'rapid reconfiguration' is implemented in the form of a scripted experiment setup the benefits are multiple: experiments can be conducted without human interaction and thus minimum dependence from human errors, working hours and the limited speed of human actions. This can eventually lead to massively repetitive experiments and the possibility of statistical validation of the results.

F17. Adjustable level of realism means that we use only the required level of detail that is sufficient to test the experiment hypothesis. For example one experiment might need to reproduce a network at the very low level using real routers and computers with specific configurations (reproducing even the lower layers of the OSI model i.e., Layer 1 and 2). Of course reproducing all details is not easy (eg. reproducing a link with specific delay and loss). Another experiment might just need to reproduce reality with less detail and thus the use of router and traffic simulators might be sufficient (focusing for example at the application layer). The concept of adjustable level of realism is to have the option to use real hardware when it's really needed and simulators or other abstractions when not. Even if someone uses real hardware and software it is very hard to reproduce realistic network traffic. This is one of the greatest challenges for realistic experiments, rather than solely functional, and address it one might use special traffic generators or try to replay real traffic from the Internet.

F18. Scalability for large experiments implies that the platform can be extended to support a large number of experimental nodes (real or virtual) potentially distributed over several physical locations. An experiment might have a considerable size in order to serve as a meaningful abstraction of the complex structure and interaction phenomena of today's Internet. To achieve scalability an experimental platform should be designed to leverage on parallelism whenever it's possible (e.g., multi-process or multi-threaded software).

F19. Accurate measurements are the key for every scientific experiment. The separation of the measurement processes from the experiment might not be always achievable but the measurement process should have a minimum impact to the experiment result. Furthermore the researcher has to consider that the accuracy of measurements depends on many factors like the choice of sampling rate and sampling strategy. Capturing raw data from all possible and diverse sources (both nodes and links) is a good approach because the researcher will be able to extract measurements from the collected data at a later stage.

F20. Probably the most sophisticated feature is "Flexibility" i.e., the ability to reproduce different topologies and architectures of many different layers and with different levels of realism. In the sake of rapid reconfiguration and automation we might have to abandon the physical layer and focus above the data link layer up to the application layer.

IV. Review of existing software

Most experimental platforms aim to provide many of the above-mentioned features but each one has its one strengths and weaknesses. In the ideal case the software that is used to create an experimentation platform should help the researcher in all steps of an experiment's lifecycle: design, initialization, execution and measurement, termination, repetition and analysis, while minimizing the cost in terms of required human effort, financial cost and time commitment. During the review, we first uncovered the basic features of the reviewed software then inferred the higher level compound features and finally summarized them in a set of tables (Tables I-III). During this process we have taken also into account certain architectural and operational requirements that the specific approach/software implies. Furthermore two clarifications have to be made. First, it should be clear that the tools that we will present might evolve and get extended. This review doesn't serve as a definite guide about the possible uses of the software but rather as a first level overview that can quickly familiarize the reader with the different approaches. The second point is that practical details are often hidden from high level presentations and research papers making thus the process of understanding the practical use of some software very hard. Although a literature review will always miss some details, we have made an extensive effort to dig out the most important details from many different sources (e.g., mailing lists, configuration manuals).

In order to present the review of the available software tools in a systematic way we group them in two broad categories that were proposed by the NSF Report on Network Research Testbeds [1] [25]:

- **Overlay testbeds** are build on top of existing infrastructures and have been extensively used in the past to test and deploy new types of protocols, services etc in a large scale environment that cannot be recreated in a dedicated facility. By relying on an existing and widely used underlying infrastructure they can host experiments of great scale but on the other hand they are constrained by the limitations (e.g., bandwidth, delay) of the underlying infrastructure that connects the overlay nodes. In this sense these testbeds are not isolated from the production systems and networks. It is interesting to note that the early Internet was essentially an overlay on the telephone network.
- **Cluster testbeds** are typically experimental facilities that contain a large number of dedicated physical resources that can be used as components for network emulation e.g., links, routers and generic servers that are used as hosts, routers or even WAN emulators. The software used on cluster testbeds often assumes that all the resources are under a single administrative domain.

A. Software for overlay testbeds

The most famous software in this category is **Planetlab** [26], a term that is some times used to refer to the software suite (Table I) and other times to the infrastructure. Planetlab as an infrastructure consists of a set of PCs

[1] according to their naming convention we are covering multi-user, experimental facilities (MXF) that support research rather than proof of concept testbeds.

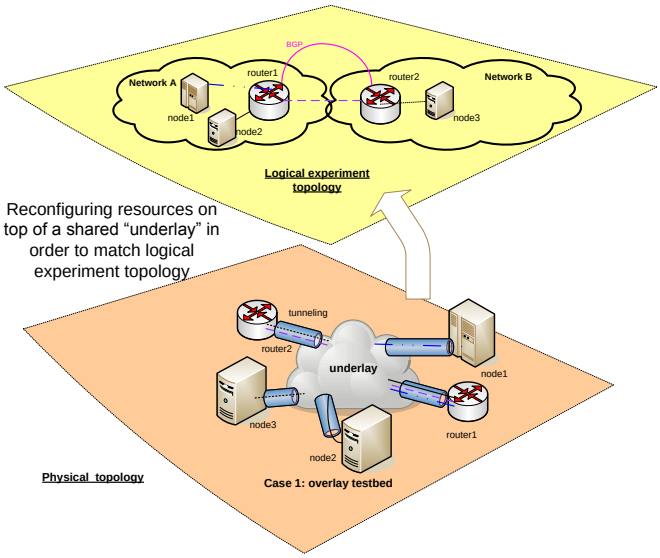

Figure 3. A rough illustration of an experiment instantiation over an overlay testbed.

(called nodes) connected to the Internet and forming an overlay network. The nodes are based on Linux and are divided into slices using virtualization techniques. A slice is a network of virtual machines, with a subset of a node's resources bound to each virtual machine. Users can deploy an experiment over specific slices of nodes and links of the overlay network. This approach is ideal for experiments that need dispersed points of presence (Internet-wide) e.g., realistic testing of new protocols. The Planetlab software allows the creation of "private Planetlabs" through a version called MyPLC. An example is Everlab [27] which is essentially a private Planetlab on high-end clusters spread over Europe (Evergrow). Planetlab's success has triggered the creation of various projects such as ONELAB which extends Planetlab to support wireless research and VINI [28] which extends the Planetlab software to support simultaneous experiments with arbitrary network topologies on a shared physical infrastructure. VINI is suitable for networking research because it exposes lower level interfaces (interfaces to virtual network devices e.g., TUN/TAP interfaces instead of sockets) to slices. The virtual network topologies are build using a clever combination of various open source software like Quagga and User Mode Linux (UML) and in a simplified manner they could be described as multiple UML instances connected by virtual point-to-point Ethernet links. VINI was deployed initially as a prototype on top of Planetlab (as an infrastructure) but the intension of its creators was to deploy it on a separate infrastructure.

In Figure 3 we present how an experiment would be implemented over an overlay testbed.

A similar concept for the creation of network-layer overlays over the Internet is used in **X-Bone** (Table I). X-Bone is a software tool that officially runs on FreeBSD and Linux and can be used to discover, configure, and monitor network resources to create overlays over a multicast enabled IP network. The X-Bone overlays isolate experiments from each other because they are constructed in such a way that packets sent to the virtual addresses of an overlay will go through the virtual links, while packets addressed to the base addresses will go through the base network (usually the Internet) instead of the virtual network. Of course this mechanism doesn't ensure proper isolation in cases of misconfiguration or error. Nodes communicate by sending and receiving packets to and from virtual addresses of the overlay. X-Bone provides a primitive mechanism to support application deployment on top of overlays and the applications have to be modified to use the overlay network.

B. Software for cluster testbeds

The need to reduce the required time to create the experiment environment e.g., vulnerable hosts and attackers, and allow the researchers to focus on developing novel technologies rather than rebuilding their test infrastructure is well known. An interesting approach dealing with this problem but without obvious continuation over the years was ViSe (Virtual Security Testbed) [32]. It proposed the use of VMware to create and store virtual machine images of different hosts that could represent attackers, detectors and victims in a cyber-attack scenario. The idea of building a virtual machine image repository is interesting and could help the collaboration between researchers but ViSe addresses only this issue and the flexible configuration of the network topology during an experiment is outside of its scope.

On the other hand another important approach that seems actively supported is the **Grid'5000** project [33]. The main purpose of this platform is to serve as an experimental testbed for research in Grid Computing and in all the layers between the network protocols up to the applications. Grid'5000 allows its users to deploy their own operating system on the resources they reserve for a limited number of hours. Furthermore its assumed that the resources are not a single cluster located in a single geographical position but distributed on different sites (allover France). The sites are connected through VLANS implemented by MPLS but are isolated from the Internet. Therefore the network interconnection of the resources is specific to the project. Nevertheless the project has released several tools that automate the tasks of reserving resources, deploying, configuring, monitoring and repeating experiments (Table II) like

- The HIPerNET tool is exploring the possibility for virtual network creation.
- Katapult automates some tasks for experiments e.g., deploying the nodes, re-deploying the nodes if too many of them failed, copying the user's SSH key to the node etc.

Table I
FEATURES OF SOFTWARE FOR OVERLAY TESTBEDS.

Feature	PlanetLab	X-Bone
F1.Control of the experiment's environment	In Planetlab there is no control of the underlying network as its designed to subject network services to real-world conditions and not to provide an isolated controlled environment. Using a private Planetlab installation on a dedicated infrastructure is nevertheless possible.	X-Bone was designed to create overlay networks and can be used either over the Internet or a dedicated controlled environment.
F2.Experiment clock and event scheduling mechanisms	Nodes are NTP synchronized but problems with clocks have been reported. No event triggering service.	Global clock is not provided but nodes can be NTP synchronized. No event triggering service.
F3.Separation of control, measurement and experiment planes	No separation between measurement and experiment planes. Isolation between experiments is provided in virtual hosts (on the OS level) and there are extensions like VIOLIN [29].	Control of the overlay is implemented with TCP connections over the underlying IP network. X-Bone keeps IP traffic on different overlays separated but additionally the researcher can enable IPSec and restrict each node to participate in one overlay at a time.
F4.Storage facilities	No centralized storage facility for user data but experiments are registered in central database.	No centralized storage facility. Custom overlays can be described in custom files and created through scripts.
F5.Use of programming interfaces	An XML-RPC programming interface exists (Planetlab Central API) through which users can access and update information in the database about users, nodes, sites, slices etc.	An X-Bone API protocol that serves as a programming interface exists.
F6.Heterogeneity of technologies (e.g., ability to use different OS)	Planetlab software runs only on Linux hosts. Custom hardware can get connected just like any other host in the Internet.	X-Bone is build by definition on top of hosts running FreeBSD or Linux. Unofficially Cisco routers are supported in overlay networks.
F7.Clean reconfiguration	Each sliver is independently created/configured. Secure ping of death can be used to reboot inaccessible nodes.	No support. It is left to the researcher.
F8.Virtualization of nodes and links	Virtual hosts can be used but with Linux OS. Virtual links exist e.g., connections between hosts in a slice but these are implemented in the OS (socket level).	Virtual hosts with different OS are not supported. Applications can access multiple overlays by using different IP address spaces.
F9.Resource utilization monitoring	Supported by software like CoMon , Plush etc.	No support. It is left to the researcher.
F10.Use of free and open source software	Yes, code is available via SVN.	Yes.
F11.Authentication and Authorization	Based on SSH keys.	Based on X.509 certificates.
F14.Repeatable experiments	No control of the underlying network can lead to repeatability problems.	No control of the underlying overlay network can lead to repeatability problems.
F15.Extensibility	Open source software and programming interfaces can provide extensibility.	Open source software and programming interfaces can support extensibility.
F16.Automation and rapid reconfiguration	Partially provided by other projects like Plush [30] where users describe their experiments in an XML document and Plush prepares the required resources.	The X-Bone API protocol could be used to automate the processes of requesting the deployment of an IP overlay and monitoring it.
F17.Adjustable level of realism	Researchers can use simulators in Linux OS and use any resource of the underlying infrastructure (e.g., Internet).	Researcher can use simulators but connecting with real hardware in custom ways is out of the scope of X-Bone's normal usage.
F18.Scalability for large experiments	Yes. Available nodes in 2009 are approximately 1000.	Yes, but underlying IP network has to support multicast.
F19.Accurate measurements (nodes and links)	Measurements during experiments are left to the researchers. For a discussion over the accuracy of the measurements depending on the measurement methods we refer to [31].	Measurements during experiments are left to the researchers.
F12.Distinction of roles	Yes. There exist roles for plain users and the people with the responsibility of overseeing their site's participation in Planetlab.	Roles could be defined in terms of Unix user accounts.
F13.Remote access	The testbed be used remotely.	Yes, overlays are remotely configurable.

- OAR is a reservation tool that allows the researcher to submit or reserve nodes either in an interactive or a batch mode.
- Kadeploy is an automatic deployment system that supports OS installation and configuration of nodes. Currently it deploys successfully Linux, *BSD, Windows, Solaris on x86 and 64 bits computers.
- Network eXperiment Engine is a tool that allows to simply script experiments involving hundreds of nodes. The scenarios are described through XML files where the topology, the configuration of and the interactions between the experimental nodes are documented.

Some of these tools are specific to Grid'5000 and the extent of the required modifications to run them on a different testbed is hard to estimate.

ModelNet [34] is a software that can emulate wide-area network conditions within a local area network. The ModelNet architecture comprises of 'edge nodes' and 'core routers' which are both hardware-wise normal servers. Users run the applications under test on the edge nodes using any OS and IP network stack and run unmodified application binaries. The edge nodes can even support multiple Virtual Machines as demonstrated in DieCast [35]. The core routers are FreeBSD servers that emulate the behavior of a WAN under the offered traffic load. The core routers upon receiving packets from the edge nodes route the traffic through an emulated network of pipes with specific characteristics such as queue length, bandwidth, latency and loss-rate. The emulation runs in real time, so packets traverse the emulated WAN with the same rates, delays and losses as the real network. ModelNet doesn't address issues like resource allocation and resource monitoring neither offers a measurement infrastructure for the nodes but nevertheless is a powerful tool to emulate WAN dynamics.

A more comprehensive approach has been followed by the StarBED [36] project in Japan. It consists of a facility with hundreds of generic PCs interconnected with Layer 2 switches that are shared between multiple concurrent experiments. In order to automate the resource allocation and the experiment execution processes, a software under the name **SpringOS** was designed and developed (Table III) . Each experiment is described in a file with the use of a custom script language. The system evaluates this description and through the interaction of several daemons automatically handles the following tasks:

- allocation of the required nodes according to the experiment description plus spares;
- initialization of experimental nodes using disk images;
- interconnection of experimental nodes by setting up VLANs in switches and IP addresses in nodes;
- synchronization of experimental nodes and message exchange;
- execution of an experiment scenario e.g., launching different programs;
- after the termination the resources are released;

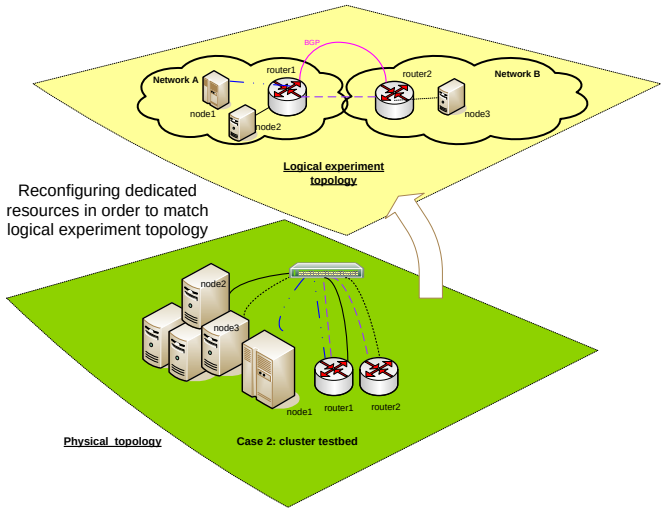

Figure 4. A rough illustration of an experiment instantiation over a cluster testbed.

SpringOS is bundled as a set of individual tools and does not cover all components that are required by the predefined architecture e.g., the dhcp and tftp daemons. This implies additional workload for the development of a complete and operational experimental platform. Furthermore the lack of a GUI for testbed manipulation might be seen as a shortcoming. Future plans of the StarBED community is to evolve into a testbed for ubiquitous networks and to provide an emulation environment.

One of the most advanced software suite for cluster testbeds is **Emulab** [37]. The name Emulab refers both to a facility at University of Utah and to a software. Nowadays the software is actively supported by multiple universities and there are many private installations throughout the world.

In Figure 4 we present how an experiment would be implemented over a cluster testbed.

From a technical point of view Emulab is quite sophisticated and feature rich and specifically:

- uses an extension of the NS [38] configuration language to describe an experiment;
- deals with the reservation and allocation of resources on the testbed;
- automates the installation of many operating systems on generic hosts i.e., experimental nodes;
- automates the deployment of custom disk images and also the creation of updated snapshots of disk images;
- supports the use of virtual hosts (FreeBSD jails) and virtual links (multiplexed links on FreeBSD);
- recreates a network topology by connecting nodes with a programmable switch using multiple VLANs;
- provides an event system that can launch arbitrary commands on hosts and modify link characteristics;
- allows the use of the simulation tool NS inside the

emulated network;
- supports packet capturing for monitoring purposes.

Finally, a software very similar to Emulab that can be used for emulating different network topologies between real or virtual hosts with the use of VLANs, is the Network Emulation Testbed (NET)[39] but the released software dates back to 2005 when the original publication was made.

V. Related work

The need for new experimental infrastructures for networking and distributed systems research as well as studies related to the design of the Internet of the Future has been the basis for several projects. Although some of them are frequently referenced we have not included them in our review because they do not provide software for the creation of testbeds (yet). Nevertheless for the sake of completeness we should mention the following:

- the 'Global Environment for Network Innovation' or GENI [2] as the most important initiative with a multi million budget in the US. GENI aims to become a wide-area shared platform that will be based on ideas from Planetlab, Emulab and Tycoon and act as a) a facility for controlled and repeatable experiments under safe conditions; b) a facility for precise, non-invasive observations of the behavior of existing networks and distributed systems under current network conditions; c)a field experimental station where new systems can be tested under actual network conditions.
- The ICT FIRE (Future Internet Research & Experimentation) initiative [3] that funds several projects towards Future Internet Research. The facility development efforts have been mainly towards overlay testbeds.
- The FEDERICA project [4](Federated E-infrastructure Devoted to European Researchers Innovating in Computing networking Architectures) aims to create a European wide experimental platform using the existing network infrastructures of National Research and Education Networks (NREN all over Europe and their interconnection network GÉANT). The proposed approach is to cut slices of the underlying resources (virtual hosts and routers) and allocate them for each experiment.
- DAS-3 (The Distributed ASCI Supercomputer 3) [40] is a project in Netherlands which provides a common computational infrastructure for researchers rather than a generic experimentation platform. It consists of a five-cluster wide-area distributed system, but its unique characteristic is that the clusters are connected by a optical network backbone which can be reconfigured on the fly (using optical routers with configurable wavelengths forming light paths). This hybrid optical network is called StarPlane and allows network users to partition the network resources and to create multiple overlay networks, each with a different logical topology.

VI. Conclusion

Networking research and particularly the assessment of the security of the current and future Internet should be based on solid and compelling evidence. In addition to a systematic analysis of incidents occurring to actual Internet systems, there is a strong need for testbeds for empirical security research. Current experimental platforms can be categorized in two types: cluster testbeds that use dedicated resources in isolation from production systems (or connected under strict control and monitoring safeguards) and overlay testbeds that use resources from an existing infrastructure and build an experimental overlay on top of them. The main advantages of cluster testbeds are the control and repeatability of experiments but their downside is that they offer artificial network conditions whereas overlay testbeds can offer real network conditions but less repeatability. Although the literature is rich of related projects and platforms, the publicly available software for the creation of a new testbed is limited. We provide an overview by presenting available tools according to a common set of basic and compound features. Emulab and Planetlab provide the most sophisticated software for each testbed type as well as documentation to support the development of private testbeds. In comparison with other tools they require the least customization and effort for creating a new private testbed. A promising approach that tries to combine the best from both approaches was recently proposed under the name "Flexlab" by Ricci et al. [41]. However further work in this direction as well as towards federated testbeds [42] is still needed.

References

[1] C. Siaterlis and M. Masera, "A review of available software for the creation of testbeds for internet security research," in *Advances in System Simulation, 2009. SIMUL '09. First International Conference on*, Sept. 2009, pp. 79–87.

[2] NSF, "Global environment for network innovations (GENI)," http://www.geni.net/ [Accessed January 14, 2010].

[3] "The european FIRE initiative future internet research and experimentation," overview of projects http://cordis.europa.eu/fp7/ict/fire/fire-fp7_en.html [Accessed January 14, 2010].

[4] "FEDERICA project," http://www.fp7-federica.eu/ [Accessed January 14, 2010].

[5] E. Göktürk, "A stance on emulation and testbeds, and a survey of network emulators and testbeds," in *21st EUROPEAN Conference on Modelling and Simulation ECMS*, 2007.

[6] DARPA, "National Cyber Range (NCR) Program," http://www.darpa.mil/sto/ia/ncr.html [Accessed January 14, 2010].

[7] R. Bajcsy, T. Benzel, M. Bishop, B. Braden, C. Brodley, S. Fahmy, S. Floyd, W. Hardaker, A. Joseph, G. Kesidis, K. Levitt, B. Lindell, P. Liu, D. Miller, R. Mundy, C. Neuman, R. Ostrenga, V. Paxson, P. Porras, C. Rosenberg, J. D. Tygar, S. Sastry, D. Sterne, and S. F. Wu, "Cyber defense technology networking and evaluation," *Commun. ACM*, vol. 47, no. 3, pp. 58–61, 2004.

[8] INFOSEC Research Council (IRC), "Hard problems list," 2005.

[9] S. W. Neville and K. F. Li, "The rational for developing larger-scale 1000+ machine emulation-based research test beds," *Advanced Information Networking and Applications Workshops, International Conference on*, vol. 0, pp. 1092–1099, 2009.

[10] M. Rimondini, "Emulation of computer networks with netkit," in *Technical Report RT-DIA-113-2007, Roma Tre University*, Jan 2007.

[11] O. Bonaventure, "Software tools for networking," in *IEEE Network Magazine, Volume 18, Issue 6*, Nov.-Dec. 2004, pp. 4 – 5.

[12] V. S. A. Volvnkin, "Large-scale reconfigurable virtual testbed for information security experiments," in *Proceedings of TridentCom*, May 2007, pp. 1–9.

[13] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: A generic framework for large-scale distributed experiments," in *Proceedings of UKSIM 2008*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 126–131.

[14] R. Crepaldi, S. Friso, A. F. Harris, III, A. Zanella, and M. Zorzi, "The design, deployment, and analysis of signetlab: a sensor network testbed and interactive management tool," in *Proceedings of WiNTECH 2006*. New York, NY, USA: ACM, 2006, pp. 93–94.

[15] J.-V. Loddo and L. Saiu, "Status report: marionnet or "how to implement a virtual network laboratory in six months and be happy"," in *ML '07*. New York, NY, USA: ACM, 2007, pp. 59–70.

[16] M. M. M. Zec, "Operating system support for integrated network emulation in imunes," in *Proceedings of the 1st Workshop on Operating System and Architectural Support for the on demand IT InfraStructure*, Oct. 2004.

[17] M. Pizzonia and M. Rimondini, "Netkit: easy emulation of complex networks on inexpensive hardware," in *Proceedings of TridentCom*, 2008, pp. 1–10.

[18] J. DeHart, F. Kuhns, J. Parwatikar, J. Turner, C. Wiseman, and K. Wong, "The open network laboratory," in *Proceedings of 37th SIGCSE*. New York, NY, USA: ACM, 2006, pp. 107–111.

[19] P. V.-B. P. R. Guillier, "Nxe: Network experiment engine: software to automate networking experiments in real testbeds."

[20] K.-i. C. Toshiyuki Miyachi, Shinsuke Miwa and Y. Shinoda, "On the nature of network experiments —issues to automate network experiments—," in *TESTCOM/FATES2008 Supplementary Proceedings*.

[21] T. Benzel, R. Braden, D. Kim, C. Neuman, A. D. Joseph, and K. Sklower, "Experience with DETER: A testbed for security research," in *TRIDENTCOM*, 2006.

[22] M. Masera and I. N. Fovino, "Methodology for experimental ict industrial and critical infrastructure security tests," in *International Conference on Availability, Reliability and Security (ARES) conference*, 2009.

[23] T. Oetiker, "About RRDtool," http://oss.oetiker.ch/rrdtool/ [Accessed January 14, 2010].

[24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.

[25] "Report of NSF workshop on network research testbeds," Oct 2002, http://www-net.cs.umass.edu/testbed_workshop/ [Accessed January 14, 2010].

[26] "Planetlab," http://www.planet-lab.org/biblio [Accessed January 14, 2010].

[27] E. Jaffe, D. Bickson, and S. Kirkpatrick, "Everlab: A production platform for research in network experimentation and computation," in *LISA*, 2007, pp. 203–213.

[28] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In vini veritas: realistic and controlled network experimentation," in *SIGCOMM*. New York, NY, USA: ACM, 2006, pp. 3–14.

[29] X. Jiang and D. Xu, "Violin: Virtual internetworking on overlay infrastructure," in *ISPA*, 2004, pp. 937–946.

[30] J. R. Albrecht, C. Tuttle, A. C. Snoeren, and A. Vahdat, "Planetlab application management using plush," *Operating Systems Review*, vol. 40, no. 1, pp. 33–40, 2006.

[31] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using planetlab for network research: myths, realities, and best practices," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 17–24, 2006.

[32] M. Richmond, "ViSe: A virtual security testbed," Jun 2005, master's Project Report. Department of Computer Science, University of California, Santa Barbara.

[33] "The grid'5000 project," https://www.grid5000.fr [Accessed January 14, 2010].

[34] K. Yocum, K. Walsh, A. Vahdat, P. Mahadevan, D. Kostic, J. Chase, and D. Becker, "Scalability and accuracy in a large-scale network emulator," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 28–28, 2002.

[35] D. Gupta, K. V. Vishwanath, and A. Vahdat, "Diecast: Testing distributed systems with an accurate scale model," in *NSDI*, 2008, pp. 407–422.

[36] T. Miyachi, K.-i. Chinen, and Y. Shinoda, "Starbed and springos: large-scale general purpose network testbed and supporting software," in *valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*. New York, NY, USA: ACM, 2006, p. 30.

[37] "Emulab ," http://www.emulab.net/ [Accessed January 14, 2010].

[38] ISI, "Network simulator ns-2," http://www.isi.edu/nsnam/ns/ [Accessed January 14, 2010].

[39] S. Maier, D. Herrscher, and K. Rothermel, "Experiences with node virtualization for scalable network emulation," *Computer Communications*, vol. 30, no. 5, pp. 943–956, 2007.

[40] "DAS-3 the distributed asci supercomputer 3 project," http://www.cs.vu.nl/das3/ [Accessed January 14, 2010].

[41] R. Ricci, J. Duerig, P. Sanaga, D. Gebhardt, M. Hibler, K. Atkinson, J. Zhang, S. K. Kasera, and J. Lepreau, "The flexlab approach to realistic evaluation of networked systems," in *NSDI*, 2007.

[42] E. F. Gouveia and S. Wahle, "Public deliverable d2.2 approach to technical infrastructure of the panlab project," 2008.

[43] F. C. Benjamin Quétier and Vincent Neri, "Selecting a virtualization system for grid/p2p large scale emulation," in *Proceedings of the EXPGRID*, 2006.

[44] P. V.-B. Primet, "INRIA, proposals for virtualization of the network in g5k."

[45] T. Miyachi, K.-i. Chinen, and Y. Shinoda, "Automatic configuration and execution of internet experiments on an actual node-based testbed," in *TRIDENTCOM '05: Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 274–282.

Table II
FEATURES OF SOFTWARE FOR CLUSTER TESTBEDS.

Feature	Grid'5000	ModelNet
F1.Control of the experiment's environment	Grid'5000 provides a controlled environment with hosts running on reserved resources. MPLS tunnels provide isolation and QoS guarantees. Different experiments can although interfere with each other.	ModelNet provides a controlled network environment which is implemented by the "core routers" that emulate a network model.
F2.Experiment clock and event scheduling mechanisms	Global clock not provided but applications like NXE can provide event scheduling functionality.	Global clock not provided. No event triggering service.
F3.Separation of control, measurement and experiment planes	Different VLANs are supported for control and experimentation.	Not defined. The experiment uses a different address space than the one used for controlling the hosts.
F4.Storage facilities	Storage is provided through NFS mounts. Experiments are registered centrally to a database for resource allocation purposes.	No storage facilities provided. The emulated network is described in XML files.
F5.Use of programming interfaces	Although OAR does not officially expose an API, a tool under development called CoRDAGe can provide a XML-RPC interface.	The emulated network topology is described in XML files. A remote API for controlling the "core routers" is not provided.
F6.Heterogeneity of technologies (e.g., ability to use different OS)	Hosts can run different OS, traffic generators, simulators etc. Custom hardware can get connected to the LAN.	The network emulator part of ModelNet ("core routers") has to run FreeBSD whereas the edge nodes can run different OS, traffic generators, simulators etc. Real hardware can get connected to the LAN. The use of virtualization inside nodes is supported on Linux, Solaris and FreeBSD.
F7.Clean reconfiguration	Hosts can be reinstalled with custom OS and rebooted.	No support. It is left to the researcher.
F8.Virtualization of nodes and links	As different OS images are supported common host virtualization software can be used. Recent work of Quétier et al. used Vserver [43] and relevant work in progress is HIPerNET [44].	Virtual hosts with different OS can act as edge nodes. Applications running on them can access the emulated network by using specific source and destination IP addresses.
F9.Resource utilization monitoring	Yes, the reservation of resources and the utilization of network links is monitored. Specialized tools like Kaspied provide more details.	No support. It is left to the researcher.
F10.Use of free and open source software	Yes but split into several tools without a central point for downloads.	Open source software available for download.
F11.Authentication and Authorization	Based on LDAP accounts.	No strict model. For automated deployment to edge nodes public key cryptography can be used for authentication (through use of daemons like authd).
F14.Repeatable experiments	Repeatable experiments are possible especially if a researcher uses an automated tool for experiment deployment.	Repeatable experiments are possible especially if the researcher uses an automated tool for experiment deployment.
F15.Extensibility	Extensibility is theoretically possible although there are concerns about programming interfaces.	Yes, the software can be extended as source code is available.
F16.Automation and rapid reconfiguration	Several tools are available for automated deployments Kadeploy, Katapult etc.	Under certain assumptions, tools for automated execution of an application on multiple virtual nodes with a single command are available.
F17.Adjustable level of realism	Yes, based on the ability to use different OS, traffic generators, simulators etc. on real or virtual hosts.	WAN traversal is always emulated by the "core routers" and does not include real routers (hardware or software). All traffic is routed along the shortest path, without emulation of routing protocols.
F18.Scalability for large experiments	Yes. Currently thousands of nodes are supported.	Yes, the emulated topologies can scale up to 1000 of nodes.
F19.Accurate measurements (nodes and links)	Measurements during experiments are left to the researchers.	Measurements during experiments are left to the researcher.
F12.Distinction of roles	A user has full privileges (root) on the reserved resources.	A user needs full privileges (root) on the testbed.
F13.Remote access	Yes, researchers can control the testbed remotely.	Yes, researchers can control the test bed remotely.

Table III
FEATURES OF SOFTWARE FOR CLUSTER TESTBEDS.

Feature	SpringOS	Emulab
F1.Control of the experiment's environment	SpringOS provides a controlled environment as it runs on top of a dedicated cluster with dedicated links.	Emulab provides a controlled environment as it runs on top of a dedicated cluster with dedicated links.
F2.Experiment clock and event scheduling mechanisms	The researcher has to configure NTP synchronization of nodes manually but event triggering services exist using a centralized or distributed model.	Global clock not provided but hosts are NTP synchronized. A powerful event launching system is provided with ability to schedule events statically and dynamically.
F3.Separation of control, measurement and experiment planes	Separation of control and experimental networks using VLANs. (nodes are required to have at least two NICs.	Separation of control and experimental networks using VLANs.
F4.Storage facilities	Is left to the researcher. An ftp server is needed for disk images. The descriptions of experiments are not stored in a central location but in plain files. The resources of the testbed are also stored in a file accessed by the resource management daemon.	User data stored centrally through a network file system (NFS). The descriptions of experiments are stored in a database.
F5.Use of programming interfaces	Work towards the creation of a standard API was referenced in [45] but the results remain unclear.	A XML-RPC interface is provided that supports creation, modification and termination of experiments as well as rebooting and configuring nodes and change of link characteristics.
F6.Heterogeneity of technologies (e.g., ability to use different OS)	Experimental nodes can run FreeBSD or Linux but not Windows. Therefore standard software such as traffic generators can be used in these nodes. The manager node must run Linux.	Hosts can run different OS, traffic generators, simulators etc. Real hardware can get connected to the LAN.
F7.Clean reconfiguration	Yes, nodes can be rebooted and reinstalled (through the snmpmine and ni tools).	Yes, nodes can be rebooted and reinstalled.
F8.Virtualization of nodes and links	Virtual nodes can be used but their complete control through SpringOS has to be developed further.	The use of virtualization inside nodes is supported out of the box on FreeBSD (although running virtual machines with different OS inside a host seems possible). Hosts can access also virtual links that are implemented on top of physical links.
F9.Resource utilization monitoring	Although the existence of a traffic and node state monitor is referenced the required software is not offered and deployment is left to the researcher (e.g., through snmp). The allocation of nodes to different experiments is accessible through the resource management daemon.	There is basic support for monitoring CPU usage, network output and ssh/serial port.
F10.Use of free and open source software	Yes, actively supported and maintained but English documentation is not complete.	Yes, actively supported and maintained.
F11.Authentication and Authorization	Based on username/password pairs.	Based on SSH keys and X.509 certificates.
F14.Repeatable experiments	Repeatable experiments are possible.	Repeatable experiments are possible.
F15.Extensibility	The availability of the source code provides some extensibility that is limited by the lack of documentation and programming interfaces.	The availability of the source code, detailed documentation and programming interfaces (XML-RPC API back-end) provides excellent extensibility.
F16.Automation and rapid reconfiguration	Several features for automated configuration, command execution are available.	Several features for automated configuration, command execution are available.
F17.Adjustable level of realism	A realistic environment for experimentation can be recreated, although it is limited by the lack of Windows OS support.	Based on the heterogeneity of the components that can be used obtaining different levels of realism seems possible.
F18.Scalability for large experiments	Reasonably scalable but the use of virtualization would permit scaling beyond the limits that are imposed by the availability of physical resources.	Yes, especially if virtual hosts and links are used.
F19.Accurate measurements (nodes and links)	No support. Is left to the researcher.	Emulab supports traffic monitoring by running tcpdump on intermediate nodes. Further measurements are left to the researchers.
F12.Distinction of roles	A user is associated to a project. The distinction between different roles and user's permissions remains unclear.	Several roles are defined (project leader, group leader, local root and user) and implemented using standard Unix users and groups.
F13.Remote access	Yes, researchers can control the testbed remotely.	Yes, researchers can control the testbed remotely.