Network Emulation on Globus-based Grids: mechanisms and challenges

Roberto Canonico, Pasquale Di Gennaro,
Vittorio Manetti, and Giorgio Ventre

Dipartimento di Informatica e Sistemistica
Università di Napoli Federico II
via Claudio 21, 80125 Napoli, ITALY
{roberto.canonico, pasquale.digennaro,
vittorio.manetti, giorgio}@unina.it

Summary. In the last few years, many emulation systems have been developed to help researchers evaluate the effectiveness of new protocols and applications in realistic network scenarios. NEPTUNE (*Network Emulation for Protocol TUNing and Evaluation*) is a flexible and scalable system developed at University of Napoli for the emulation of different network scenarios by means of a cluster of workstations. Setting up an emulation experiment in a cluster-based system requires, firstly, the ability to map virtual resources requested by an experimenter onto available physical resources and, secondly, the ability to exert a precise control over the allocated physical resources. These two requirements have much in common with resource management issues already addressed by the Grid computing community. Hence, we decided to exploit the *Virtual Workspace* concept at the foundation of the design of the NEPTUNE architecture. In this paper, we illustrate the peculiar virtualization requirements of a cluster-based emulation system and discuss how a Globus Virtual Workspace based on Xen virtual machines can be used as the basis for implementing a distributed network emulation system.

1 Introduction

Design validation and performance evaluation of new distributed applications and protocols is usually performed either by means of real testbeds or through simulation. Simulation presents unquestionable merits [1], such as the possibility to simulate large scale networks in a controlled and repeatable environment; yet, it has some drawbacks, such as the use of ad-hoc implementations of network protocols and applications and the difficulty of modelling low-level implementation details that may affect a system performance (e.g. [2]). An alternative approach to simulation consists in reproducing the system under test in a real testbed: sets of computers, communication devices, and other resources that either reproduce real-world networks or are overlaid upon the real

Internet. Large scale geographic testbeds, like PlanetLab [3], have been built in the last few years, thanks to cooperative effort of several research institutions. Nowadays, they represent a valuable tool for testing new planet-scale services, like Content Distribution Networks. Nonetheless, the testbed approach still presents some limitations. The main limits of experimental testbeds are 1) limited scalability, 2) limited software reconfigurability, 3) difficulty to reproduce the heterogeneity of the real Internet, both in terms of terminal capabilities and networking technologies (e.g. PlanetLab hosts are quite homogenous in terms of hardware configuration, and they have more powerful CPUs and faster Internet connections than a typical end-user system). Thus, results obtained on an experimental testbed, even a large scale one like PlanetLab, may not necessarily be representative of the behavior of a real-world system. To overcome these limits, in the last few years several emulation systems have been developed to evaluate the effectiveness of new protocols and applications in more heterogeneous, controllable and realistic network scenarios.

The basic idea behind network emulation is that simulated network elements should be capable of interacting with real network components and applications. A fundamental difference between simulation and emulation is that while the former runs in a virtual simulated time, the latter must run in real time to allow coexistence of both simulated and real network elements [4, 5]. To support network emulation, several approaches have been pursued.

For simple networking experiments (e.g. testing new multimedia applications under variable network conditions) the *dummynet* tool is a practical and simple solution [6]. Dummynet works by simulating the effects of finite queues, bandwidth limitations and communication delays in a single workstation.

Another approach consists in extending a packet-based network simulator with some emulation facilities, so that it can be fed in real-time with real traffic traces directly extracted from a real network. The well-known ns2 network simulator, for instance, has been extended to support emulation [7].

The new frontier of network emulation is cluster-based network emulation. These systems bring together a large number of network components (links, switches, PCs that are interchangeable as hosts, routers, or WAN emulators), in a common facility that can be remotely accessed by users through a web interface. They support flexible configuration, so that users may perform experiments on a variety of distinct network topologies with programmable delays that emulate the latencies of wide-area networks. Through virtualization and space-sharing that fully isolates simultaneous experiments, they efficiently use cluster resources.

Maybe the most complex cluster-based emulation system developed so far is Emulab [4]. Emulab is a free-for-use, Web-accessible, time- and space-shared, reconfigurable network testbed, providing integrated access to a wide range of experimental environments.

NEPTUNE is a cluster-based emulation system developed at University of Napoli. Even though many design assumptions made for NEPTUNE were borrowed by Emulab, since from the early stages of design, NEPTUNE has

assumed virtualization as a key technology for realizing complex networking scenario. The goal of this paper is to illustrate the importance of virtualization for network emulation and the role of GRID execution management mechanisms in NEPTUNE.

The rest of the paper is organized as follows. In Section 2 we briefly describe different approaches for resource virtualization, and, with more details, the Xen para-virtualization system. In Section 3 we describe the general architecture of NEPTUNE. In Section 4 we describe the importance of virtualization for cluster-based Network Emulation systems. In particular, we describe two different kinds of resource multiplexing techniques: one for computational resources (that we call *node multiplexing*) and another for communication resources (that we call *link multiplexing*). In this section we also describe the virtualization techniques we adopted in NEPTUNE to solve the node multiplexing and the link multiplexing problems. In Section 5, we describe the use of Grid Virtual Workspaces as a new instrument for dynamically deploying Xen Virtual Machines in the NEPTUNE network emulation system. Finally, we describe the current state of the implementation and present some concluding remarks.

2 Resource Virtualization techniques

The main purpose of virtualization techniques is to hide the physical characteristics of computing resources from the upper layers of a computing system. One of the reasons for virtualization is making a single physical resource appear to function as multiple logical (or "virtual") resources.

There are many approaches to virtualization. Virtualization can be applied to single physical resources of a computing system (e.g a single device) or to a complete computing system. When applied in this latter sense, (a.k.a. *Platform Virtualization*), virtualization allows the coexistence of multiple "Virtual Machines" in the same computing "host".

Platform virtualization, is implemented by means of an additional software layer, called Virtual Machine Monitor (VMM) (or *Hypervisor*), that acts as an intermediary between the system hardware resources and the Operating System. The so called *full virtualization* approach implements in software a full virtual replica of the emulated system's hardware, so that the operating system and user applications may run on the virtual hardware exactly as they would in the original system. An alternative, more recent, approach is called *paravirtualization* and consists in implementing a software interface that is similar but not identical to the underlying hardware. Such an approach requires the operating systems to be explicitly ported to run on top of the VMM.

Xen [8] is a paravirtualization system developed at the University of Cambridge. Xen provides a Virtual Machine Monitor for x86 processors that supports execution of multiple guest operating systems at the same time.

3 The NEPTUNE emulation system

NEPTUNE is a cluster-based network emulation system developed at University of Napoli [9] that can be used to assess new networking technologies and protocols (e.g. to test new QoS routing protocols and Traffic Engineering schemes in MPLS-based networks), as well as new distributed applications (e.g. multimedia peer-to-peer applications).

At the time of this writing, the NEPTUNE emulation system runs on a cluster of workstations consisting of 28 biprocessor nodes ProLiant DL380, each equipped with two Intel Pentium IV Xeon 2.8 GHz CPUs, 5 GB of PC-2100 RAM, one 100 Mbps Ethernet NIC and one Gigabit Ethernet NIC. Each node is equipped with a 34.6 GB SCSI disk. A 700GB centralized disk array is also available to the whole cluster. The cluster nodes are connected each other through a set of 100/1000 Ethernet switches.

One of the cluster nodes, the NeptuneManager, provides the fundamental services (like dhcp, dns, tftp, nfs, and so on) needed to properly configure at boot-time the physical cluster nodes and the virtual machines participating to the emulation experiments. A web-based system is used to manage and configure the whole system.

Setting up an emulation experiment in NEPTUNE consists primarily in defining a "virtual topology" made of emulated intermediate network nodes (routers) and end-system nodes (user terminals). A testbed mapping module (much like the one used in Emulab [10]) is responsible of mapping the "virtual" topology onto the cluster physical resources. Virtual network nodes are implemented in NEPTUNE as Xen virtual machines. The main advantage of the use of virtualization techniques to instantiate virtual network nodes is the significant reduction in equipment and management costs. Virtual machines allow the creation of customized execution environments, where customization consists in selecting the operating system, installed software packages and user access policies. Furthermore, virtual machines can be paused or shut down at any time, and later resumed, even at a different physical location (migration). Finally, virtual machines support fine-grained mechanism for resource usage control, allowing to define (and even change at run-time) precise limits to the the amount of usable RAM and disk space.

In NEPTUNE, a complex networked system is reproduced by allocating multiple "virtual" network nodes (both routers and end systems) on each of the cluster physical nodes.

4 Use of virtualization in emulation systems

In a typical cluster-based network emulation system, users submit to the system an experiment request. An experiment request contains a "virtual" network description to be reproduced with the available cluster resources. An experiment description usually contains at least the following information:

- the "virtual" network topology, including a list of relevant parameters for each of the nodes and links;
- operating systems and software packages to be loaded in each of the emulated nodes.

When the experiment is activated (swap-in), a system module maps the logical topology of the experiment onto actual testbed hardware, loads the requested operating system and software onto the allocated devices and establishes the network links.

Traditional network emulation systems use conservative resource allocation and map the emulated "virtual" nodes onto dedicated PCs and emulated links onto switched ethernet links, thus severely limiting scalability. Nowadays, with increasing computational power made available at low-cost, it is possible to exploit virtualization techniques to map multiple "virtual" nodes on a single CPU. There are many good reasons for doing that. For example many applications need to be evaluated on large topologies, yet they are not resource hungry. Moreover, multiplexing provides a more efficient use of communication resources as the bandwidth of the emulated geographic links is usually much less than that available in the local interconnect used in a modern cluster [11]. These reasons motivate the use of small-scale clusters to emulate medium/large size topologies in a inexpensive manner [12].

4.1 Node Multiplexing

Node multiplexing is the problem of emulating more than a network node on the same physical cluster node. This problem is inherently a problem of machine virtualization, as it has been described in section 2. Hence, it can be solved with one of the many available virtualization technics. Aspects to be taken into account to select the proper one for a cluster-based emulation system are efficiency, scalability, flexibility, isolation, and operating system customization.

In Emulab, node multiplexing is implemented by means of a modified version of FreeBSD Jail. In NEPTUNE we decided to implement node multiplexing by means of Xen. We decided that Xen suits our needs because:

- it is highly scalable;
- it potentially supports different kinds of Operating Systems;
- it is transparent to the applications running in the virtual machines;
- it provides good isolation among different virtual machines running concurrently;
- it supports virtual machine migration, allowing dynamic re-allocation of experiments on the cluster nodes;
- it implements different optimization techniques in the communication mechanisms, allowing good communication performance among virtual machines implemented within the same physical node.

4.2 Link Multiplexing

The nodes of a cluster are connected by means of one or more switched Ethernet LANs. Each cluster node may be equipped with one or more (Giga or Fast) Ethernet NIC. These NICs, in turn, may be connected to the same switch or to different switches. In theory, it would be possible to connect the cluster nodes in pairs, by means of crossed Ethernet cables, so to physically reproduce the desired topology. However, such a solution is not viable, for at least two reasons. Firstly, because changing the network topology would be extremely impractical, time-consuming and error-prone. Secondly, because this would make it impossible to emulate network topologies with a number of links greater than half of the number of Ethernet NICs. Hence, practical solutions require to emulate multiple point-to-point connections on top of one or more shared Ethernet LANs. This is usually performed by means of Virtual LANs (VLANs) [11], [13]. Such a solution is implemented by properly configuring the Ethernet switches and does not require any configuration and processing in the cluster nodes. This makes, however, the system configuration software extremely dependent on the characteristics of the network switches.

For the above reasons, we decided not to use VLANs in NEPTUNE and we adopted two network device independent solutions for link multiplexing:

- IP-aliasing and destination MAC address filtering
- Virtual NICs

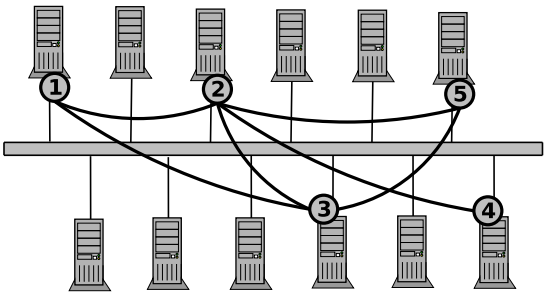

Fig. 1. A virtual network with IP aliasing and destination MAC filtering

The first technique consists in activating multiple "aliased" logical network interfaces for each of the node's NICs. Figure 1 shows a virtual network composed by 5 nodes connected by means of 6 "virtual links". Each of these links is associated to a point-to-point IP subnet. The IP-level network topology is implemented by setting static IP routes. To emulate links of assigned capacity, outgoing packets are handled by the CBQ scheduler implemented in the Linux Traffic Control kernel module. Outgoing packets are assigned to

the corresponding queue by the *ebtables* kernel module, that marks packets according to the MAC address of the destination node.

The second technique is implemented by exploiting the virtualization mechanisms implemented in Xen. Every time a new virtual machine (*domU* in Xen terminology) is instantiated, Xen creates new "connected virtual ethernet interfaces", with one end of each pair within the domU and the other end within the Hypervisor (*dom0* in Xen terminology). Virtualised network interfaces are given different Ethernet MAC addresses. These addresses may be either assigned at random in the range of unicast "locally assigned" MAC addresses (i.e. addresses of the form XY:XX:XX:XX:XX:XX, where X is any hexadecimal digit, and Y is one of 2, 6, A or E) or they can be assigned a given value at domU creation time. When using such an approach, it is not necessary to share a single interface in order to emulate several links, since each virtual network interface is assigned to one of the two ends of an emulated link in the virtual topology. Figure 2 shows a virtual network in which link multiplexing is implemented by means of this latter technique. The figure shows how a network of six nodes has been emulated by activating three virtual machines in Physical Host A and three virtual machines in Physical Host B. Six virtual links have been created, by properly instantiating one, two or three virtual ethernet interfaces in each of the virtual machines and configuring their IP addresses so to create six different point-to-point IP subnets.

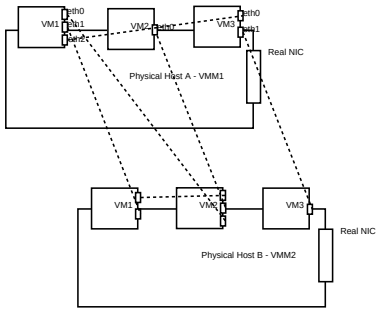

Fig. 2. A virtual network implemented by means of virtual machines

In order to assign a given capacity value to each of the emulated links, a queuing discipline and a traffic shaper are associated to both ends of an emulated link. We refer to this technique as the *one link per interface* technique. Figure 3 shows with more details the network configuration of a host, in which two Virtual Machines (domU_1 and domU_2) have been setup and five different links have been emulated, three of them connected to node domU_1 and the remaining two of them to node domU_2.

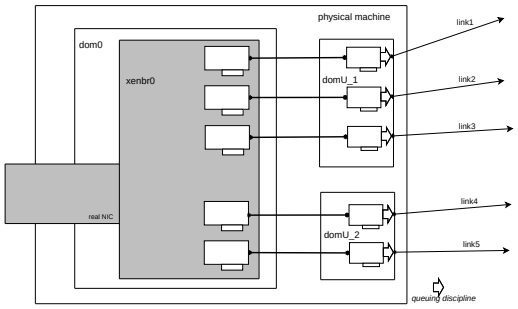

Fig. 3. *one link per interface* technique

5 Network Emulation as an on-demand Grid application

Resource allocation is a key aspect of shared testbed infrastructures such as PlanetLab and Emulab [14]. While designing our system, we recognized that many resource management issues have already been addressed by the GRID community.

Traditional cluster-based Network Emulation systems have based their design on very specific and customized hardware, in order to reproduce the behaviour of real-life communications equipments.

Grid computing, on the other hand, is evolving on a different path, i.e. it aims at building a reliable and efficient infrastructure to satisfy different user requirements by means of shared powerful general-purpose computing facilities. In a Grid system, a middleware like Globus has the role of managing users requests and of customizing the computational resources available in a cluster of general purpose computers in order to match the users demands. Such an approach gives the system a higher degree of flexibility and manageability.

To emulate large-scale heterogeneous networking scenarios, the above mentioned features of a Grid infrastructure are also desirable for a cluster-based Network Emulation system. In the following, we describe what strategies we are currently pursuing in order to bring some of the technologies developed for Globus-based Grids into the next version of NEPTUNE.

5.1 Virtual Workspaces in the Grid

Nowadays several production-quality Grids exists, which offer powerful execution infrastructures to help solve many science specific problems. Much progress has been achieved with the deployment of Grid-based applications, but the preparation of remote execution environment and the enforcement of Quality of Service (QoS) while still providing manageability is yet a hot research topic. One problem with most Grid platforms, today, is the lack of performance isolation and resource usage enforcement: activities relative to one user or virtual organization [15] can unpredictably influence the performance of other processes executing on the same platform. Another issue is the

impossibility to dynamically customize the execution environments according to the users requirements, thus compromising their *Quality of Life* (QoL) [16] in interactions with Grid software.

To satisfy such a requirement, the *Virtual Workspace* concept has been recently introduced in the Grid computing model. A Virtual Workspace (VW) is an abstraction of an execution environment that can be made dynamically available to authorized clients by using well-defined protocols [17]. The abstraction captures resource quota assigned to such execution environment on deployment (such as CPU, memory share, disk size etc) as well as software configuration aspects of the environment (such as operating system installation or provided services). This environment abstraction can be implemented by using various technologies. The Globus Toolkit supports at least two different techniques: one based on dynamically created user accounts (a.k.a. *Dynamic Accounts*) and one based on dynamic deployment of Virtual Machines. Virtual workspaces implemented as Virtual Machines offer a high degree of isolation, fine-grained enforcement and configuration: VM image configuration, in fact, can reflect a workspace's software requirements, while a hypervisor, such as Xen, can ensure the enforcement of hardware properties.

In Globus, the so-called Workspace Service allows a Grid client to dynamically create and manage workspaces. The Workspace Service implementation based on Virtual Machines takes as input a VM image wrapped in meta-data providing critical deployment information and deploys the VM on one of the physical hosts administered by it.

An atomic workspace, implementing a single execution environment, contains both the data (VM image) and some metadata describing deployment information and prerequisites. Atomic workspaces can be combined to form aggregate workspaces such as virtual clusters [18].

5.2 NEPTUNE as a Virtual Workspace

We are currently working at a next release of NEPTUNE in which node participating to a network emulation experiment are part of a virtual cluster implemented according to the Virtual Workspace concept. This allows the cluster physical resources to be used concurrently with other Grid applications. In our very first implementation we are interested in demonstrating the feasibility of our proposal: a portable cluster-based network emulator easily deployable as a Grid application. Our goal is to not make use of any hardware-dependent solution, so that our system may be replicated on any cluster.

The NEPTUNE cluster emulator can be seen as composed of two workspace sets: the first set contains the head node (*NeptuneManager*), while the second set contains worker nodes. NeptuneManager, the head-node of the virtual cluster, is instantiated as a virtual workspace by itself. The virtual network deployment is made by NeptuneManager.

The NEPTUNE worker nodes may be deployed on-demand by authorized users. Each worker node could have a specialized configuration, but for sim-

plicity's sake we can assume in our experiments that all worker nodes share same virtual workspace image, which can be easily cloned.

The service node of the physical cluster runs a Xen 3.0 virtual machine monitor, Globus Toolkit 4 (GT4) and the Workspace Service. We are currently using the TP1.2.1 version (*Technology Preview*) of the Virtual Workspace Service. The workspace infrastructure implemented in Globus Toolkit 4 uses the *Web Services Resource Framework*(WSRF) model and is composed of:

- the *Workspace Factory Service* that allows a Grid client to deploy a Xen-based workspace according to a deployment request specifying resource allocation and length of deployment;
- the *Workspace Service* that allows a Grid client to manage a workspace by starting, stopping, pausing or destroying it.

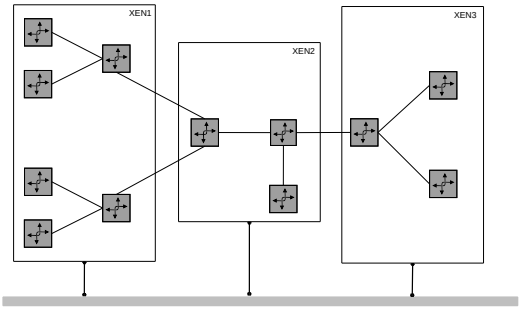

Fig. 4. An emulated network implemented by means of virtual machines

Fig. 4 shows an overlay network created by means of virtual machines. After the head node (NeptuneManager) has been instantiated by the workspace service, we have performed the following steps:

- definition of the network topology to emulate using the NeptuneManager services;
- configuration of the virtual node's features like RAM and virtual interfaces;
- definition of the software image to be loaded into the virtual machines;
- selection of the physical cluster's worker nodes (equipped by Xen 3);
- deployment of the virtual machines.

As soon as the virtual nodes are available, NeptuneManager configures the overlay network and activates any services. The link-multiplexing shown in fig. 4 was realized by the *one link per interface* technique described above. In order to test the implemented configuration, we enable OSPF routing by GNU/quagga on the emulated network; finally, we generate and trace several traffic flows between the virtual nodes.

6 Conclusion and Future Works

Synthetic creation, configuration, deployment and management of medium-large scale virtual networks by means of a cluster-based system requires fine-grained control of resource utilization and manageability of the instantiated execution environments. These requirements have much in common with the challenges addressed in the context of Grid computing: a single virtual-node participating to an emulation experiment, its computational resources and its software configuration, can be seen like a Grid's Virtual Workspace. Recently, in the Grid community, the use of virtualization techniques to support on-demand instantiation of virtual workspaces has been proposed. In particular, para-virtualization techniques appear as the most promising, due to the high degree isolation between the execution environments obtained with low performance overhead. In the next future, paravirtualization will also benefit from hardware support implemented by both Intel and AMD in their latest CPUs, allowing OSes to run natively (i.e. unmodified) and with no overhead penalties on these CPUs.

In this paper, we have shown how Xen can be used to realize the virtualization of both computational and network resources, justifying the use of this instrument in the network emulation context. We have also shown how network emulation can be implemented as an on-demand application for a Grid computing environment. Finally, we have described the strategies we have pursued in order to bring some of the technologies developed for Globus-based Grids into the next version of NEPTUNE.

In the future, our main goal is to realize a further integration between our emulation environment (NEPTUNE) and the Globus virtual workspace.

7 Acknowledgements

This work has been partially funded by the FP6 IST-2005-034819 project "OneLab: an open networking laboratory supporting communication network research across heterogeneous environments". Pasquale Di Gennaro has been funded from the Italian Ministry of University and Research (MIUR), in the form of a grant in the framework of the S.Co.P.E. Project.

References

1. L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.
2. Ramaswamy Ramaswamy, Ning Weng, and Tilman Wolf. Considering processing cost in network simulations. In *Proc. of Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools) in conjunction with ACM SIGCOMM*, pages 47–56, Karlsruhe, Germany, August 2003.

3. L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In *Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I), Princeton, New Jersey, USA*, Oct. 2002.
4. B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, December 2002. USENIX Association.
5. Daniel Herrscher and Kurt Rothermel. A Dynamic Network Scenario Emulation Tool. In *Procs. of the 11th International Conference on Computer Communications and Networks (ICCCN 2002)*, pages 262–267, Miami, October 2002.
6. Luigi Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *SIGCOMM Comput. Commun. Rev.*, 27(1):31–41, 1997.
7. K. Fall. Network emulation in the VINT/NS simulator. In *Proceedings of the fourth IEEE Symposium on Computers and Communications*, 1999.
8. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Procs. of the 19th ACM Symposium on Operating Systems Principles, SOSP'03*, pages 164–177. ACM Press, 2003.
9. NEPTUNE network emulation system (`http://neptune.comics.unina.it`). Università di Napoli Federico II.
10. Robert Ricci, Chris Alfeld, and Jay Lepreau. A solver for the network testbed mapping problem. *SIGCOMM Comput. Commun. Rev.*, 33(2):65–81, 2003.
11. Mike Hibler, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, and Jay Lepreau. Feedback-directed virtualization techniques for scalable network experimentation. Technical report, University of Utah, Flux Group Technical Note 2004-02, May 2004.
12. Shashi Guruprasad, Robert Ricci, and Jay Lepreau. Integrated network experimentation using simulation and emulation. In *Proceedings of TridentCom*. IEEE, February 2005.
13. Steffen Maier, Daniel Herrscher, and Kurt Rothermel. On node virtualization for scalable network emulation. In *Procs. of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 05), Philadelphia, PA, July 24-28, 2005*, pages 917–928, July 2005.
14. Robert Ricci, David Oppenheimer, Jay Lepreau, and Amin Vahdat. Lessons from resource allocators for large-scale multiuser testbeds. *SIGOPS Oper. Syst. Rev.*, 40(1):25–32, 2006.
15. Ian T. Foster. The anatomy of the grid: Enabling scalable virtual organizations. In *CCGRID*, pages 6–7. IEEE Computer Society, 2001.
16. K. Keahey, I. Foster, T. Freeman, and X. Zhang. Virtual workspaces: Achieving quality of service and quality of life in the grid. *Scientific Programming Journal*, 13(4):265–276, Feb 2005.
17. K. Keahey, I. T. Foster, T. Freeman, X. Zhang, and D. Galron. Virtual workspaces in the grid. In *Proceedings of Euro-Par 2005, Lisbon, Portugal, Aug.30 - Sept. 2, 2005*, volume 3648 of *Lecture Notes in Computer Science*, pages 421–431. Springer, 2005.
18. Zhang X., K. Keahey, I. Foster, and T. Freeman. Virtual clusters workspaces for grid applications. In *CCGRID*, pages 513–520. IEEE Computer Society, 2006.