NemC: A Network Emulator for Cluster-of-Clusters*

Hyun-Wook Jin† Sundeep Narravula‡ Karthikeyan Vaidyanathan‡ Dhabaleswar K. Panda‡

†Computer Engineering Department
Konkuk University
Seoul, 143-701 Korea
jinh@konkuk.ac.kr

‡Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{narravul, vaidyana, panda}@cse.ohio-state.edu

Abstract

A large number of clusters are being used in all different organizations such as universities, laboratories, etc. These clusters are, however, usually independent from each other even in the same organization or building. To provide a single image of such clusters to users and utilize them in an integrated manner, cluster-of-clusters has been suggested. However, since research groups usually do not have the actual backbone networks for cluster-of-clusters, which can be reconfigured with respect to delay, packet loss, etc. as needed, it is not feasible to carry out practical research over realistic environments. Accordingly, the demand for an efficient way to emulate the backbone networks for cluster-of-clusters is overreaching. In this paper, we suggest a novel design for emulating the backbone networks of cluster-of-clusters. The emulator named NemC can support the fine-grained network delay resolution minimizing the additional overheads. The experimental results show that NemC can emulate the low delay and high bandwidth backbone networks more accurately than existing emulators such as NISTNet and NetEm. We also present a case study showing the performance of MPI applications over cluster-of-clusters environment using NemC.

Keywords: *Network Emulator, Cluster-of-Clusters, High-Speed Backbone Networks, and MPI*

1. Introduction

Cluster systems are becoming more popular for a wide range of applications owing to their cost-effectiveness. A large number of such clusters are being used in all different organizations such as universities, laboratories, etc. These clusters are, however, usually independent from each other even in the same organization, i.e., applications (e.g., scientific parallel applications) can run only on a single cluster and cannot utilize the idle resources of other clusters. Thus it is highly desired that the clusters in the same organization provide a single image to users and are utilized in an integrated manner. To cater to such needs, researchers have suggested Cluster-of-Clusters [2, 17], which aims to construct a cluster combining few or many clusters with high-speed backbone networks. Though this term can be also referred as Grid, in this paper, we consider it as a cluster of clusters that is geographically distributed in a *small area* (i.e., the same campus or building), which is more tightly coupled system than Grid. This computing environment will be beneficial to the organizations that want to fully utilize their clusters providing a single image without exposing the system to the out side.

The cluster-of-clusters environment poses several research challenges including performance, compatibility, security, authentication, etc. However, before addressing such research challenges, one of the foremost critical issues is how to construct the experimental environment of cluster-of-clusters. Since research groups usually do not have the actual backbone networks for cluster-of-clusters, which can be reconfigured with respect to delay, packet loss, etc. as needed, it is hard to carry out practical research over realistic environments. Accordingly, the demand for an efficient way to emulate the backbone networks for cluster-of-clusters is overreaching. Approaches involving simulations and modeling are widely accepted [7, 3, 5]; however, these approaches have the limitations that they cannot run actual software (i.e., applications, middleware, and system software). On the other hand, if we can emulate only the backbone networks running actual clusters, it will provide very close environments to the real-world systems but also give flexibility to change the system parameters, such as network delay, packet loss, etc. For the emulation, a workstation can be configured as a router with multiple Network Interface Cards (NICs), of which each is connected to a cluster. By running a network emulation software that generates artificial network delay, packet loss, etc. on the workstation-based router we can emulate the backbone networks for cluster-of-clusters while running actual software over the

* This research is supported in part by the Faculty Research Fund of Konkuk University, Department of Energy's Grant #DE-FC02-01ER25506, National Science Foundation's grants #CNS-0403342, and #CNS-0509452; and equipment donations from Ammasso, Inc.

clusters in a transparent manner.

Though there are several existing network emulators [6, 11, 14, 15], they are focusing on large scale Wide Area Networks (WANs) such as Internet. However, there are many prominently different characteristics between such WANs and the backbone networks for cluster-of-clusters. For example, the backbone networks usually have a much lower delay than typical WAN environments though the backbone networks have a higher delay than the intra-cluster LAN environments. The emulators that can emulate a millisecond network delay resolution may not be enough to emulate the high-speed backbone networks. In addition, the bandwidth provided by the backbone networks for cluster-of-clusters is higher than the WAN case. Hence the emulator should be able to emulate higher bandwidth networks.

In this paper, we suggest a novel design for emulating the backbone networks of cluster-of-clusters. The emulator named *NemC* (Network Emulator for Cluster-of-Clusters) can support the fine-grained network delay resolution minimizing the additional overheads. We design a new packet scheduling mechanism that performs on-demand scheduling, which is independent on any system timers. Also we minimize the additional overhead by designing it at the kernel-level to emulate high bandwidth networks. In addition to the network delay emulation, current implementation of NemC can emulate packet losses and out-of-order packets. To the best of our knowledge, no research has focused on the network emulation for cluster-of-cluster environments and NemC is the first emulator to address this.

The experimental results show that NemC can emulate the low delay and high bandwidth backbone networks more accurately than existing emulators such as NISTNet [6] and NetEm [11]. We also present the performance evaluation results of MPI [9] applications such as NAS [1] and Gromacs [4] over cluster-of-clusters environment using NemC as a case study.

Rest of this paper is organized as follows: Section 2 suggests a new network emulator for cluster-of-clusters and details its design. The experimental evaluation of the emulator and example of its use are presented in Section 3. The related work is discussed in Section 4. Finally, this paper concludes in Section 5.

2. Design and Implementation of NemC

In this section, we detail the design and implementation of our network emulator for cluster-of-clusters named NemC. NemC is implemented using the `netfilter` hooks provided by Linux, which can be dynamically inserted to the kernel's chain of packet processing. A run-time loadable kernel module which runs on Linux-based routers is used to perform all operations. Its design does not require any kernel modifications. The current implementation can insert network delay with fine-grained resolution, packet drops, and out-of-order packets.

Figure 1 shows the overall design of NemC. As shown in the figure, NemC consists of four components: (i) NemC netfilter, (ii) NemC scheduling demon, (iii) NemC kernel module and (iv) NemC user applications. The NemC netfilter intercepts the packets arrived at the router node after the IP routing decision. Based on the parameters set by the user applications, the NemC netfilter can drop packets, generate out-of-order packets, or introduce network delays. These parameters can be controlled at run-time by using the NemC user applications. The NemC scheduling daemon is a user-level process, which requests the netfilter to search the packets that has been sufficiently delayed and reinject them into the network. The kernel module takes care of insertion of the netfilter in the initialization phase but also provides access to the internal data structures and parameters of the NemC netfilter to the scheduling daemon and the user applications.

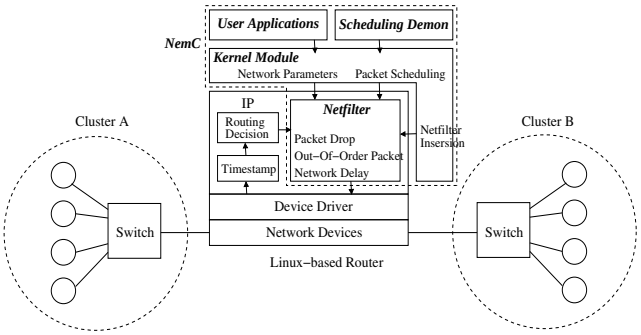


Figure 1. Overall Design of NemC

2.1. Packet Scheduling for Fine-Grained Delay Resolution

The backbone networks for cluster-of-clusters have low network delay compared to general WANs such as Internet. To emulate such networks, the emulator is required to support fine-grained delay resolution. The delay resolution of a network emulator is mainly decided by the triggering mechanism of packet scheduling. The packets delayed more than the given time, net_delay, at the router node are reinjected into the network by the packet scheduling routine. The most widely used mechanism to trigger the packet scheduling is to invoke the scheduling routine for every timer interrupt. This mechanism is simple to design and implement; however, since it depends on the system timer resolution, it may not be able to support fine-grained delay resolution. For example, if the network emulator uses Linux timer then it can support only 10ms (with kernel version 2.4) or 1ms (with kernel version 2.6) delay resolution, which is too coarse-grained to emulate the backbone networks for cluster-of-clusters. On the other hand, if the network emulator directly uses a hardware timer in the system,

the interrupt can be generated very high frequency and can delay the actual packet processing.

To overcome these limitations of the timer based mechanism, we suggest the on-demand packet scheduling mechanism. In this mechanism, the packet scheduling routine is triggered by either incoming packet or scheduling daemon. That is, whenever there is a new packet arrived at the router node, it triggers the packet scheduling routine, while the user-level scheduling demon continually tries to invoke the packet scheduling routine if there are no packets waiting to be processed in the protocol stacks and the system is idle. It is to be noted that the user-level scheduling daemon has lower priority than the kernel-level packet processing context. Thus, if packets arrive at the router node in a bursty manner the scheduling routine will be invoked very frequently by those packets. On the other hand, if packets arrive intermittently then the user-level daemon will continuously trigger the packet scheduling. In this manner, we can trigger the scheduling routine as much as possible (i.e., in a fine-grained mode) without any effect on the actual packet processing of the protocol stacks. In this mechanism, since both newly arrived packets and the user-level daemon invoke the scheduling routine, which accesses the same data structures in the NemC netfilter, we guarantee that only one can access the data structures at a time by locking. We use the time stamp in the `sk_buff` data structure of the Linux kernel to calculate the total time duration spent by the packet in the router node.

2.2. Low Overhead Emulation for High Bandwidth Support

Another important characteristic of the backbone networks for cluster-of-clusters is high bandwidth. To emulate the high bandwidth networks, we need to address two critical issues: i) delay cascading and ii) emulation overhead.

If an emulator holds a packet for a given time to add a delay without yielding the CPU resource, this delay will be cascaded to the next packets that have been already arrived at the router node. For example, if an emulator is implemented as a high priority kernel-level process and polls the timer occupying the CPU resource, the delay can be cascaded on subsequent packets. To avoid this delay cascading problem, we place the packets that need to be delayed into a queue and immediately return the context to the original routine. The packets queued are re-injected by the packet scheduling mechanism described in Section 2.1.

On the other hand, higher emulation overheads can reduce the effective bandwidth between the clusters in the experimental systems, which is a undesired side effect. Broadly, the emulator can be implemented at the user-level or the kernel-level. The user-level emulation requires two data copies between user and kernel buffers for each packet. This copy operation is a well-known bottleneck of packet processing. Hence, our network emulator is designed at the kernel-level to prevent any additional data copy.

2.3. Packet Drop and Out-of-Order Packet Generation

Since the backbone networks for cluster-of-clusters can use store-and-forward networks there can be packet drops because of network congestion. To emulate such case, we generate packet drops based on the packet drop rate value, $drop\_rate$, given by a NemC user application. NemC chooses a packet randomly for every $drop\_rate$ packets and simply drops this packet freeing all the resources occupied by this packet.

Out-of-order packets can occur in cluster-of-clusters due to multi-path and adaptive routing. To emulate such case, we generate out-of-order packets using a given out-of-order packet generation rate, ooo_rate, and a delay for out-of-order packets, ooo_delay. These values are set by a NemC user application. It is guaranteed that the value of ooo_delay is always larger than that of net_delay. NemC chooses a packet randomly for every ooo_rate packets and delays this packet as much as ooo_delay. Since this packet has been delayed more than other packets it becomes an out-of-order packet if the packet interval between this packet and the next is smaller than ooo_delay.

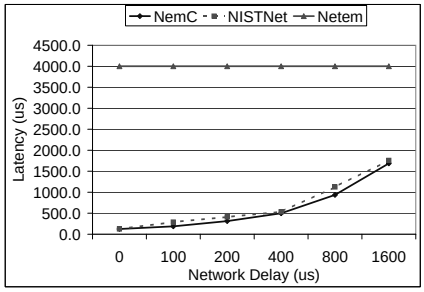
3. Experimental Evaluation of NemC

In this section, we describe our experimental methodology. In Section 3.1, we compare our network emulator, NemC, with popular existing network emulators and evaluates the benefits of NemC. In Section 3.2, we outline the overall usage of NemC and demonstrate the main uses of NemC. For all our experiments we used two clusters whose descriptions are as follows:

Cluster A: A cluster system consisting of 4 nodes built around SuperMicro SUPER P4DL6 motherboards which include 64-bit 133 MHz PCI-X interfaces. Each node has two Intel Xeon 2.4 GHz processors and 512 MB of main memory. We used the RedHat 9.0 Linux distribution.

Cluster B: A cluster system consisting of 4 nodes built around SuperMicro SUPER X5DL8-GG motherboards which include 64-bit 133 MHz PCI-X interfaces. Each node has two Intel Xeon 3.0 GHz processors and 512 MB of main memory. We used the RedHat 9.0 Linux distribution.

The nodes are connected with Ammasso Gigabit Ethernet interface cards. The software (SDK) version used is 1.2-ga2. These cluster nodes are internally connected with Netgear GS524T Gigabit Switches. As shown in Figure 1 these switches are connected each other through the workstation-based router that is similar in configuration to Cluster B nodes'. This router node runs emulators to emulate the backbone networks.



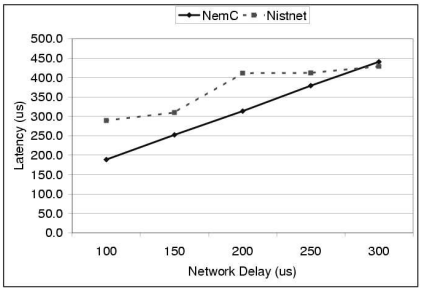
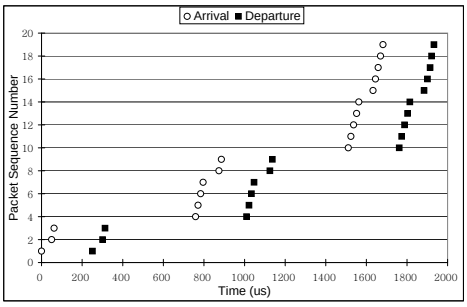


Figure 2. Latency: (a) Comparison with Varying Network Delay (b) Fine-Grained Network Delay



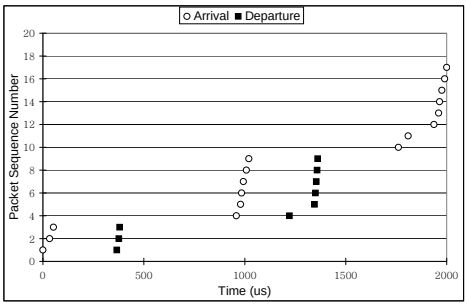


Figure 3. TCP Flow Analysis for (a) NemC and (b) NISTNet

3.1. Microbenchmarks

In this section, we compare NemC with existing emulators such as NISTNet [6] and NetEm [11]. We measure the latency and the bandwidth with `nttcp` (version 1.47) varying the emulated network delay. Since the focus of the paper is the fine-grained network delay emulation, we do not include the experimental results for packet drop and out-of-order packet cases.

Figure 2(a) shows the 512B message latency between two nodes in different clusters, Clusters A and B, while emulating the network delay using NemC, NISTNet, and NetEm. We vary the network delay from 0μs to 1600μs. As we can see in the figure, NetEm shows almost constant latency regardless of the expected network delay. Also this latency is much higher than the given delay. It is because the packet scheduling of NetEm uses the Linux system timer, which has milliseconds timer resolution. Hence, it cannot generate the fine-grained network delay. On the other hand, NemC and NISTNet generate the network delay very close to the given delay value.

To closely look at NemC and NISTNet, we measure the 512B message latency again with finer network delay values. Figure 2(b) presents the results. We can observe that NISTNet shows interestingly the almost same latency for 100μs and 150μs delay values. We can observe the same trend with 200μs, 250μs, and 300μs delay values. It is due to the fact that NISTNet uses the MC146818 real-time clock for the packet scheduling of which the tick resolution is approximately 122μs. Therefore, it cannot support finer network delay resolution than 122μs. On the other hand, as we can see in the figure, NemC is emulating the given delay values very accurately. The reason why we see around 200μs latency with 100μs network delay is because of the default latency between two nodes, which is roughly 100μs.

Table 1. Comparisons with Real Network Latency

Message Size (Bytes)	Real Network (μs)	Emulation Error (%)		
		NemC	NISTNet	NetEm
512	336.6	6.8	22.3	1088.4
16384	2206.5	3.3	30.3	225.4

In order to see how much NemC can emulate accurately the real network, we measure the latency of a real intra-building network and compare it with the emulation results. Table 1 shows the 512B and 16KB message latency of the real network and the emulation errors of each emulator. As we can see NemC shows very few errors compared to other emulators.

We also measure the 512B message bandwidth between two nodes in different clusters. Similar to the latency results, we have observed that NetEm shows almost the same bandwidth no matter what delay value has been given. Since NetEm adds too much delay for small delay val-

ues the bandwidth is also very low. More importantly, we have observed that NemC can achieve higher bandwidth (up to 37%) than NISTNet for small network delay values. To compare NemC and NISTNet in detail, we execute `tcpdump` on the router node, in which the emulator is running, and observe the behavior of each emulator while performing the bandwidth test. For this experiment, we have set the network delay into 250μs. The message size is 512B. Figures 3(a) and (b) present a snapshot for the first 2ms of NemC and NISTNet, respectively. The graphs show when each packet has been arrived at the router node (indicated with empty circles in the figures) and when it has left (plotted with filled rectangles in the figures). With these figures, we can clearly see how long the packets have been delayed in the router node by the emulator. As we can observe in the figure, NemC emulates 250μs network delay very accurately while NISTNet adds more than 350μs, which accounts for 40% error. This explains why we see better bandwidth with NemC than NISTNet.

3.2. Case Study: MPI Applications over Cluster-of-Clusters

In this section, we evaluate and analyze the performance of MPI applications over cluster-of-clusters using our network emulator, NemC. We choose the following applications as a representative set for evaluation: (i) NAS version 2.3 (Class B) - EP, IS and MG and (ii) Gromacs version 3.3 - d.villin. We have used the MPI library, MPICH Version 1.2.7p1 [10].

Each evaluation is divided into two parts: (i) Execution on a single cluster (represented by 4x1 in the graphs) and (ii) execution on a cluster-of-clusters with varying emulated network delay (represented by the corresponding network delay in the graphs). Each single cluster contains 4 nodes. The cluster-of-clusters experiments utilize the nodes of both the clusters.

Figure 4(a) shows the performance of EP, IS, and MG. The single cluster execution of EP takes 82.5s. Further, we notice that the execution times of EP over a cluster-of-clusters do not depend largely on the network delay. Performance and execution time of applications like EP can hence be improved immensely by utilizing the nodes of cluster-of-clusters. In addition, since the network delay does not affect the execution time of EP executed on cluster-of-clusters significantly, this application can benefit even from cluster-of-clusters formed by widely separated clusters with high network delay. In the figure, we also observe that the network delay shows a fair impact on the execution times of IS and MG. The execution times of these applications executed on single clusters (50.3s and 24.4s respectively) are higher than the their execution times with well-connected cluster-of-clusters (about 30.0s and 16.4s respectively for network delay of 100μs). This shows that these applications can perform up to 40% and 33% faster using the cluster-of-clusters setup. Further, we notice that the benefit of using cluster-of-clusters diminishes with increasing network delay. Hence these applications can benefit from running on multiple clusters for network delays up to some extent.

Performance of Gromacs - d.villin shown in Figure 4(b) shows that the single cluster execution of this application performs significantly better than its execution on cluster-of-clusters (with all delays). It is to be noted that the y-axis of this graph is *Simulations/Day*. Since this application is highly communication intensive, the execution on more number of nodes spread over the high delay networks increase its communication overheads heavily. Hence applications like these can rarely benefit from cluster-of-clusters systems.

In aggregate, we have demonstrated that our network emulator NemC can accurately answer the following questions: (i) Can a given application execute faster on a cluster-of-clusters? (ii) What is the maximum network delay that can sustain this benefit? and (iii) What is the measure of the extent of benefit possible? The capability of NemC to emulate fine-grained delay enables us to evaluate and predict these trends accurately.

4. Related Work

There has been a significant research on network emulation. NISTNet [6] and NetEm [11] are the widely employed network emulators running on Linux systems. However, these emulators are focusing on how to emulate general WANs. Hence fine-grained network delay resolution was not an important factor to these emulators as shown in Section 3.1. On the other hand, NemC has been designed carefully to deal with low delay and high bandwidth network characteristics of cluster-of-clusters. For FreeBSD based systems, Dummynet [14] and ModelNet [15] have been suggested. Again, these emulators target large scale WANs rather than cluster-of-clusters environment.

There are also well designed Grid emulators. For example, MicroGrid [13] provides a virtual Grid environment for Grid applications. Netbed [16] provides integrated access to simulated, emulated, and wide-area network testbeds. These emulators are beneficial to develop and evaluate applications over large scale Grid environments.

We also have introduced a simple delay generator for network emulation in one of our previous works to evaluate RDMA over IP [12]. This emulator, however, does not consider all the design issues discussed in this paper and has only limited features.

5. Conclusions and Future Work

In this paper, we suggest a novel design for emulating the backbone networks of cluster-of-clusters. The emulator named NemC can support the fine-grained network delay resolution minimizing the additional overheads. To reflect the low delay and high bandwidth characteristics of

Figure 4. Performance of MPI Applications -*Single Cluster* vs. *Cluster-of-Clusters* (a) NAS - EP, IS and MG and (b) Gromacs - d.villin

the backbone networks, we design a new packet scheduling mechanism that performs on-demand scheduling. In addition to the network delay emulation, current NemC implementation can emulate packet losses and out-of-order packets.

The experimental results clearly show that NemC can emulate the low delay and high bandwidth backbone networks more accurately than existing emulators such as NISTNet and NetEm. We also present the performance evaluation results of MPI applications such as NAS and Gromacs over cluster-of-clusters environment using NemC as a case study. On the whole, we demonstrate the ability of NemC to accurately evaluate the possible benefits (or lack thereof) for applications executing over cluster-of-clusters environments with varying network characteristics.

As future work, we plan to add more features such as statistical generation of delay. In addition, we intend to evaluate NemC with 10 Gigabit Ethernet [8] and its scalability. We try to see how much the multi-processor system can improve the scalability and whether the Linux kernel needs to be optimized further for better scalability. We plan to evaluate the applications over a larger system size. We also like to extend our target emulation system to even Grid environment.

References

[1] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, D. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS Parallel Benchmarks. *The International Journal of Supercomputer Applications*, 5(3):63–73, 1991.

[2] M. Barreto, R. Avila, and P. Navaux. The MultiCluster Model to the Integrated Use of Multiple Workstation Clusters. In *Proceedings of PC-NOW*, pages 71–80, 2000.

[3] W. Bell, D. Cameron, L. Capozza, A. Millar, K. Stockinger, and F. Zini. OptorSim - A Grid Simulator for Studying Dynamic Data Replication Strategies. *International Journal of High Performance Computing Applications*, 17(4):403–416, 2003.

[4] H. Berendsen, D. van der Spoel, and R. van Drunen. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications*, 91(1), 1995.

[5] J. Cao. ARMSim: a Modeling and Simulation Environment for Agent-based Grid Computing. *Simulation*, 80(4), 2004.

[6] M. Carson and D. Santay. NIST Net: A Linux-based Network Emulation Tool. *Computer Communication Review*, 33(3):111–126, June 2003.

[7] K. Fall and K. Varadhan. The NS Manual (Formerly NS Notes and Documentation), February 2006.

[8] W. Feng, P. Balaji, C. Baron, L. N. Bhuyan, and D. K. Panda. Performance Characterization of a 10-Gigabit Ethernet TOE. In *Proceedings of HotI*, August 2005.

[9] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, Version 1.1, June 1995.

[10] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. High-performance, portable implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, 22(6):789–828, 1996.

[11] S. Hemminger. Network Emulation with NetEm. In *Proceedings of LCA*, April 2005.

[12] H.-W. Jin, S. Narravula, G. Brown, K. Vaidyanathan, P. Balaji, and D. K. Panda. Performance Evaluation of RDMA over IP: A Case Study with the Ammasso Gigabit Ethernet NIC. In *Proceedings of HPI-DC*, pages 41–48, July 2005.

[13] X. Liu, H. Xia, and A. Chien. Network Emulation Tools for Modeling Grid Behavior. In *Proceedings of CCGrid*, May 2003.

[14] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *Computer Communication Review*, 27(1):31–41, January 1997.

[15] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of OSDI*, December 2002.

[16] 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 *Proceedings of OSDI*, pages 255–270, December 2002.

[17] M. Xu. Effective Metacomputing using LSF MultiCluster. In *Proceedings of CCGrid*, pages 100–105, 2001.