The Longcut Wide Area Network Emulator: Design and Evaluation

Lars Ailo Bongo
Department of Computer Science
University of Tromsø
Norway
Email: larsab@cs.uit.no

***Abstract*— Experiments run on a Grid, consisting of clusters administered by multiple organizations connected by shared wide area networks (WANs), may not be reproducible. First, traffic on the WAN cannot be controlled. Second, allocating the same resources for subsequent experiments can be difficult. Longcut solves both problems by splitting a single cluster into several parts, and for each part having one node emulating a WAN link by delaying messages sent through it. The delay is calculated using latency and bandwidth measurements collected using the Network Weather Service and a parallel application monitor. We evaluate the precision, usability for WAN collective operation research, and scalability of Longcut.**

I. Introduction

A Grid consists of clusters administered by multiple organizations connected by shared wide area networks. Two factors make system research difficult in such an environment. First, experiments may not be reproducible since the traffic on shared WANs cannot be controlled [8]. Second, allocating exclusive access, at the same time, to several clusters is usually not supported by Grid middleware. To avoid these problems a large cluster (or several small cluster) at a single site can be used with emulated WAN links.

As input to the emulator we use latency and bandwidth traces of real WAN links collected using the Network Weather Service (NWS) [16] and the EventSpace parallel application monitor [2].

The reaming of this paper proceeds as follows. Section II describes related work. Section III describes the design and implementation of the Longcut WAN emulator. The trace collection tools are described, and the collected traces evaluated, in section IV. Longcut is evaluated in section V, by doing experiments measuring the precision, scalability, and usability for WAN collective operation research of the collected traces. Finally section VI concludes.

II. Related Work

The design of Longcut is inspired by the Panda WAN emulator [8]. Both use sub-cluster gateway nodes to run WAN emulation code. Also, both are closely integrated with the communication system. Our experiments differ from the distributed work queue experiments in [8], in that we use applications with higher communication frequency.

Other emulators are Netbed [15]. Dummynet [13], nse [7], Trace Modulation [10] and ModelNet [14]. Most use low-level rerouting which requires adding a module to the operating system. Longcut runs unmodified applications on unmodified operating systems.

Alternatives to emulation are simulation [5], [17] and live-network experimentation. Simulation provides a controlled, easy to change, and repeatable environment. However, higher level abstraction must be used due to the scale of the system; thus accuracy is lost. Live-network experimentation using environments such as PlanetLab [11] is most realistic, but often these are not designed for performance experiments. For example PlanetLab uses virtualization to share resources and protect services from each other, which makes it difficult to control the load on resources.

There are several network monitoring tools [6], [9], [16] that can be used to collect the traces used by Longcut. However, most of the existing traces do not have the high sample rate required for our experiments.

III. Design and Implementation

In many clusters a gateway node provides the single entry point to the compute nodes, to the benefit of cluster users and administrators.

The design of Longcut is similar to the Panda WAN emulator [8]. A cluster is split into several sub-clusters. For each sub-cluster we select one node to act as a gateway. All communication to the sub-cluster is routed through its gateway, which delays messages to emulate the higher latency of WAN connections.

To implement Longcut, we need to change the communication paths used by applications, such that messages are routed through the gateway where the emulation code is run. Being a research tool, Longcut should be extensible such that users can add their own emulation code, and configurable such that the emulated topology can be easily changed. Our communication system, PATHS [1], supports all this.

PATHS provide configurable *paths* though the communication system. A path consists of several *wrappers* that can run arbitrary code. Figure 1 shows how we reconfigure a path between two nodes to include a gateway node which runs the emulation code in form of a wrapper. Extending Longcut with other emulation approaches requires writing a new wrapper (consisting of 3 functions).

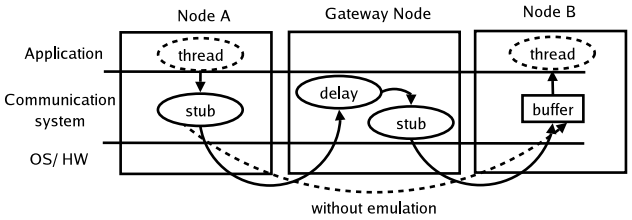


Fig. 1. Communication path with WAN emulation wrapper.

All communication paths used by an application are specified in a *pathmap* [2], which is created using three data structures: cluster topology, application communication information, and a mapping of application threads and communication buffers to the clusters. To re-route messages the cluster topology is changed. To add emulation wrappers, scripts are run that reconfigure the pathmap.

We have implemented two types of WAN emulation where the delay is calculated using: (i) constant WAN latency-, and bandwidth, and (ii) latency and bandwidth time series read from trace files. The first type is useful for simple experiments where different topologies are evaluated. The tools used to collect the traces are described in the following section.

On the gateway there is one thread per TCP/IP connection. In our initial implementation the threads waited either by blocking (by calling *usleep*) or spinning. Spinning had to be used since it was not possible to sleep for less than 30 ms. This approach does not scale well, since gateways emulating many WAN links can have many threads spinning at the same time causing loss of accuracy (as reported in [4]).

To make sure only one thread spins at a time, we reimplemented the delay code as shown in figure 2. Threads block if there is already one thread spinning. Threads are unblocked when the currently spinning thread exists, or when they are done waiting. The scalability of Longcut is evaluated in section V.

IV. Trace Collection

Five cluster gateways were monitored:

vvgw.cs.uit.no	: Pentium 4 3.2 GHz in Tromsø, Norway.
psgw.cs.uit.no	: dual Pentium II 300 MHz in Tromsø, Norway.
clustis.idi.ntnu.no	: dual Athlon MP 1.6 GHz in Trondheim, Norway.
roadrunner.imada.sdu.dk	: Pentium III 1.4 GHz in Odense, Denmark.
benedict.aau.dk	: dual-CPU Pentium III 733 MHz in Aalborg, Denmark.

The topology, ping latency and link bandwidth of the WANs between the monitored nodes are shown in figure 3.

A. Monitoring Tools

1) Network Weather Service: A widely used network monitoring tools is the Network Weather Service (NWS) [16].

```
done_time = current_time() + wait_time;

if (somebody spinning)
  // signaled by spinning thread
  condition_wait();

while (1) {
  current_time = timestamp();
  if (current_time() > done_time) {
    if (thread blocked)
      // unblocked thread will do the
      // spinning
      condition_signal();
    break;
  }

  for (each blocked thread)
    if (current_time() > thread done_time)
      // unblocked thread will exit
      condition_signal();

  // allow others to run
  yield();
}
```

Fig. 2. Pseudo code for the delay function.

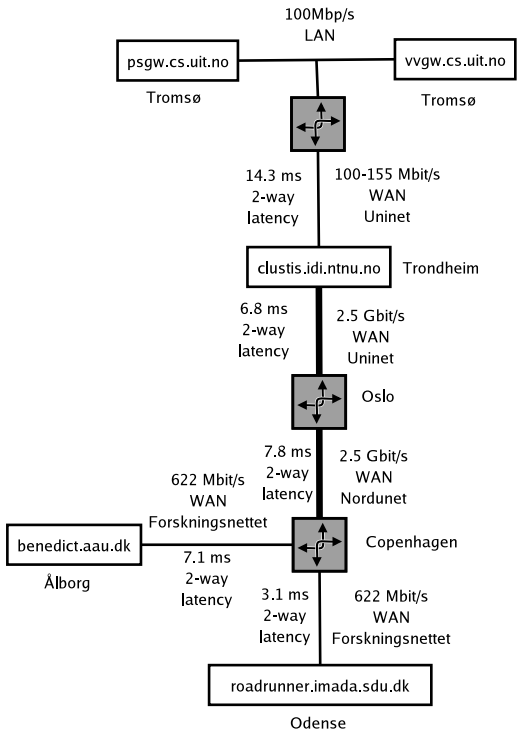


Fig. 3. The monitored topology (all intermediate routers are not shown).

TABLE I

NWS TWO-WAY LATENCY IN MILLISECONDS.

	benedict	clustis	psgw	roadrunner	vvgw
benedict		22.43	36.70	7.11	36.52
clustis	22.14		17.88	18.81	14.88
psgw	36.71	15.03		34.84	0.31
roadrunner	7.03	18.83	34.00		32.95
vvgw	36.29	14.52	0.54	32.82	

TABLE II

NWS MEAN BANDWIDTH IN MBITS/SEC.

	benedict	clustis	psgw	roadrunner	vvgw
benedict		3.15	2.58	8.60	2.32
clustis	3.50		5.57	3.96	5.63
psgw	2.28	4.85		2.58	81.73
roadrunner	6.27	2.70	1.67		1.87
vvgw	2.28	4.78	79.64	2.56	

NWS has low monitoring overhead, and has been shows to provide measurements accurate enough to predict future TCP/IP latencies and bandwidth [16]. It is easy to install and use, but three ports need to be opened on firewalls.

Latency is measured by sending a four byte message. Bandwidth is measured by sending four 16 Kbytes messages using a socket buffer size of 32 Kbytes each 60th second. We tried using a shorter sample period (1 second), but the rate was too high for the monitored WAN connections.

2) EventSpace: Using the EventSpace monitoring tool [2] we can trace the latencies of TCP/IP connections as used by a communication system for WANs. EventSpace allows low-overhead monitoring of the actual communication rate of the applications we are interested in. However, installing EventSpace can be difficult due to a large number of libraries used (e.g. Python). Also, firewalls need to be opened for the PATHS server ports.

We collected traces for two benchmarks. The first was collected for a collective operation micro-benchmark run on a multi-cluster (the experiment is described in [3]). As only small message were used, we do not report bandwidth results. In the second experiment, we used a benchmark designed for latency and bandwidth measurements. For each iteration it sends an eight byte message, followed by two 32 Kbytes messages. Sends were blocking, hence one must complete before a new one can be initiated.

We did one experiment were the Nagle algorithm was disabled on all TCP/IP connections, to ensure that even small messages are sent immediately, but it did not significantly reduce the latency.

B. Collected Traces

Tables I, III and IV shows the mean two-way TCP/IP latency measured for the different links (in both directions). The NWS trace has smaller mean latencies for small latency links than the EventSpace traces. Tables II and V shows the TCP/IP throughput. The EventSpace trace has higher bandwidth than the NWS trace.

TABLE III

EVENTSPACE COLLECTIVE OPERATION TRACE TWO-WAY LATENCY (MILLISECONDS).

	benedict	psgw	roadrunner
benedict		35.76	9.16
psgw			32.49
roadrunner		32.35	

Increasing the sample rate lowers the observed variation both in bandwidth and latency. Also, the bandwidth differs in two directions, while the latency usually does not. Conclusions should not be drawn from the above results since we have only collected one trace for each link.

V. EXPERIMENTS

For the experiments we use a cluster with 44 nodes, each with a single-CPU Pentium 4 3.2 GHz with Hyper-threading (2-way SMT) enabled. The nodes are connected using Gigabit Ethernet, and all run Linux with kernel version 2.4.21. We use NPTL threads for the experiments. On all TCP/IP connections the Nagle algorithm was disabled and default socket sizes were used. The delay is implemented with the single-thread-spinning approach described above.

A. Precision

To investigate the precision of Longcut, we measured application level ping-pong latency and bandwidth between a cluster in Tromsø and Trondheim using PingPong from the Pallas Microbenchmark suite (PMB) [12] (ported to PATHS). We also traced the link by using EventSpace to monitor the latency-bandwidth micro-benchmark, and used the captured trace to emulate the link on our cluster. Each experiment was repeated twice. For most message sizes the real and emulated links have similar latency and bandwidth (figure 4).

We also measured how different traces influence the latency and bandwidth of PingPong. Two traces were used; the NWS and EventSpace latency-bandwidth microbenchmark traces presented in section IV. Also we did one experiment with constant latency and bandwidth values (means from the EventSpace trace).

Figure 5 shows the difference in latency and bandwidth for the WAN link between Odense and Aalborg. The PingPong results differ for the NWS and EventSpace traces since the NWS trace has lower latency and lower latency than the EventSpace trace. However, using constant values does not differ significantly from using the EventSpace trace, even if it has smaller variation. We have similar results for other links.

We also measured how the different traces influence the performance of collective communication using Allreduce from PMB. The cluster was split into four parts with 10 nodes in each part in addition to the node selected as gateway. The four clusters were emulated to be in Tromsø (behind vvgw), Trondheim, Odense and Aalborg. The difference between the constant value trace and the EventSpace trace is smaller than for PingPong (figure 6). However, the difference in NWS

TABLE IV

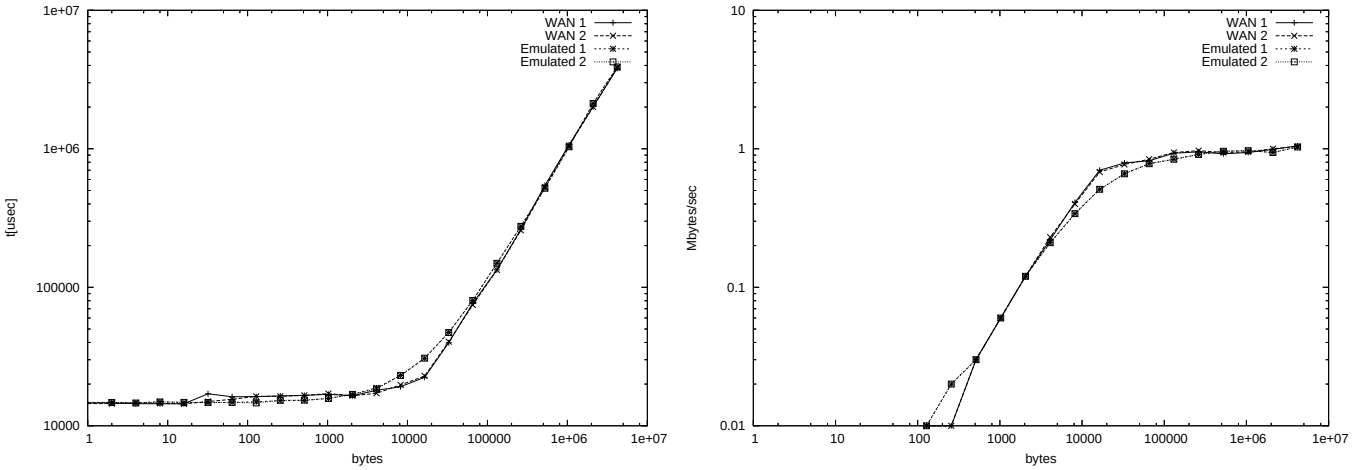
EVENTSPACE LATENCY-BANDWIDTH MICROBENCHMARK TRACE LATENCY (MILLISECONDS). STANDARD DEVIATION IN PARENTHESIS.

	benedict	clustis	psgw	roadrunner	vvgw
benedict		23.18 (3.31)	37.19 (1.66)	12.70 (27.03)	36.98 (1.48)
clustis	22.82 (3.41)		14.80 (2.47)	23.87 (30.67)	14.91 (3.84)
psgw	37.13 (1.57)	15.00 (1.17)		36.90 (28.51)	1.81 (1.52)
roadrunner	9.85 (4.02)	20.43 (3.12)	33.93 (3.75)		34.02 (4.38)
vvgw	36.95 (1.46)	15.00 (1.19)	1.84 (1.34)	36.49 (26.84)	

TABLE V

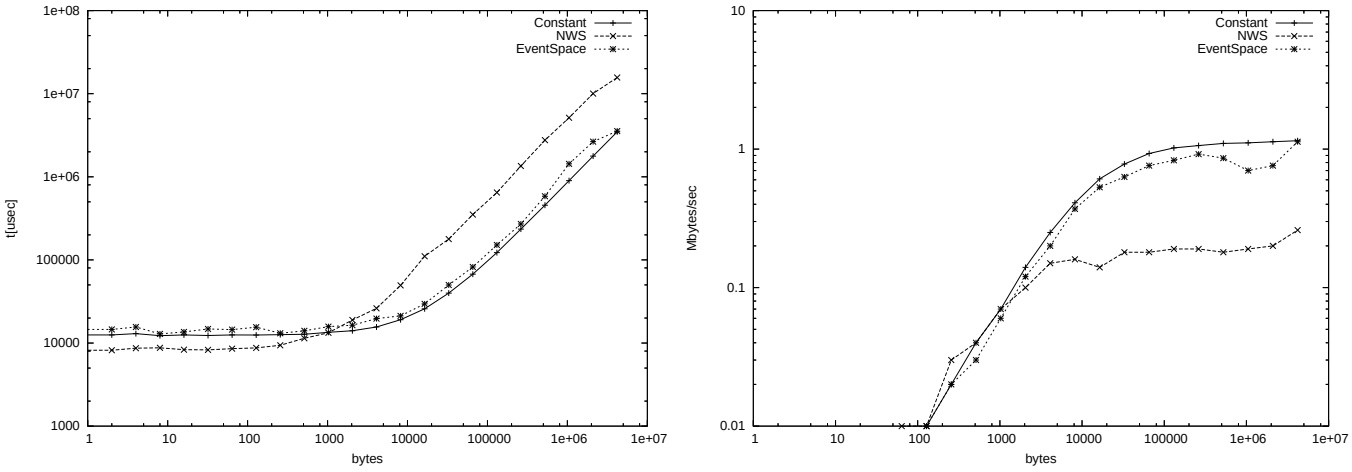
EVENTSPACE LATENCY-BANDWIDTH MICROBENCHMARK TRACE BANDWIDTH (MBITS/SEC). STANDARD DEVIATION IN PARENTHESIS.

	benedict	clustis	psgw	roadrunner	vvgw
benedict		8.80 (0.45)	5.75 (0.66)	10.60 (3.22)	3.19 (0.08)
clustis	4.97 (0.50)		9.74 (3.01)	6.76 (2.35)	7.78 (0.69)
psgw	3.18 (0.10)	12.73 (0.75)		4.86 (1.64)	44.49 (10.99)
roadrunner	10.08 (1.28)	8.18 (2.30)	5.49 (1.48)		3.46 (0.26)
vvgw	3.19 (0.09)	12.79 (0.76)	51.50 (9.68)	4.75 (1.63)	



(a) Latency (b) Bandwidth

Fig. 4. Measured and emulated PingPong latency and bandwidth between nodes in Tromsø and Trondheim.



(a) Latency (b) Bandwidth

Fig. 5. Emulated PingPong latency and performance using different traces for Odense and Aalborg link.

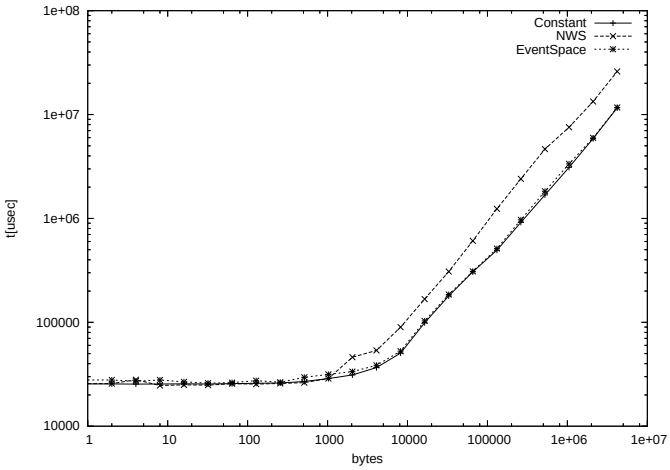


Fig. 6. Emulated Allreduce latency using different traces.

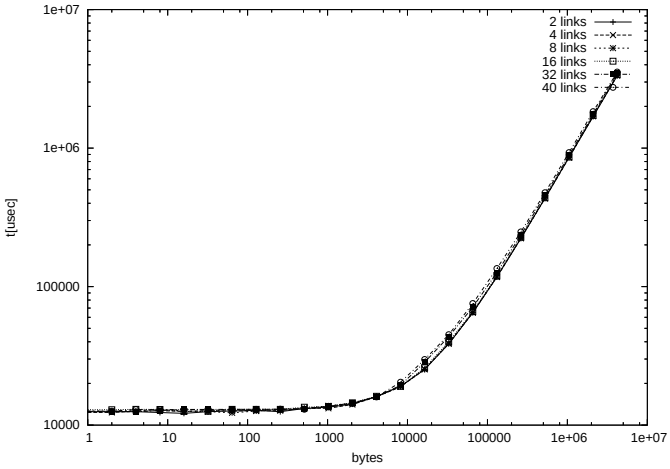


Fig. 7. Longcut scalability measured using PingPong with increasing number of emulated connections per gateway.

and EventSpace latency and bandwidth do influence Allreduce performance.

B. Scalability

To evaluate the scalability of Longcut, we measured the number of TCP/IP connections each gateway can emulate without loss in precision. The cluster was divided into two parts with 20 nodes in each part, and we run several instances of PingPong, all communicating over emulated WAN links (Aalborg–Odense). For each instance of PingPong each gateway handles two TCP/IP connections. Figure 7 shows the maximum latency observed for each experiment. PingPong latency does not differ when emulating 2 and 40 connections.

C. Usability

In our final experiment, we measure the execution time of an application kernel. The kernel is Successive Over-Relaxation (SOR). We use a Red-Black checker pointing version of SOR, with a matrix size of 48000×48000. The cluster was divided into four parts as described above. Each worker-process is assigned 1200 rows, and each updates all its red points and then exchanges red border point values by sending a 19800 bytes message to each neighbor. Then black points are updated and the communication is repeated. At the end of each iteration the global change in the system is calculated using allreduce (with and 8 byte message). For the problem size chosen about 70% of the execution time is spent communicating when using the NWS trace. Table VI shows the execution time, and the slowdown compared to the constant value trace.

TABLE VI
SOR PERFORMANCE WITH DIFFERENT TRACES.

Trace	Exec. time	Slowdown
Constant	383.8 sec.	
EventSpace	390.3 sec.	2%
NWS	461.9 sec.	17%

VI. CONCLUSION

We have described the design and implementation of the Longcut WAN emulator, shown the emulation precision using traces collected by different tools, and evaluated the scalability of Longcut.

We learned the following lessons:

- For most traces, bandwidth differs in two directions, while latency does not.
- Traces with finer granularity have higher latency.
- The difference for point-to-point communication performance does not significantly differ when using constant and traced latency and bandwidth values.
- For synchronizing collective communication, such as allreduce, there are small differences between using latency-bandwidth traces and constant values.

The collected traces are available at http://www.cs.uit.no/∼larsab/longcut/.

ACKNOWLEDGMENTS

Thanks to Brian Vinter for providing us access to the clusters in Denmark. Also thanks to Josva Kleist and Gerd Behrmann for allowing us to use the cluster in Aalborg, and Anne C. Elster for allowing us to use the cluster in Trondheim. Thanks to Otto J. Anshus, John Markus Bjørndalen and Espen S. Johnsen for discussions, and to the MNF-8000 students who were referees for this paper.

REFERENCES

[1] BJØRNDALEN, J. M. *Improving the Speedup of Parallel and Distributed Applications on Clusters and Multi-Clusters*. PhD thesis, Department of Computer Science, University of Tromsø, 2003.

[2] BONGO, L. A., ANSHUS, O., AND BJØRNDALEN, J. M. EventSpace - Exposing and observing communication behavior of parallel cluster applications. In *Euro-Par* (2003), vol. 2790 of *Lecture Notes in Computer Science*, Springer, pp. 47–56.

[3] BONGO, L. A., ANSHUS, O., BJØRNDALEN, J. M., AND LARSEN, T. Extending collective operations with application semantics for improving multi-cluster performance. In *ISPDC/HeteroPar* (July 2004), IEEE Computer Society, pp. 320–327.

[4] BONGO, L. A., ANSHUS, O. J., AND BJØRNDALEN, J. M. Low overhead high performance runtime monitoring of collective communication, 2005. To appear in Proc. of ICPP'05.

[5] BRAKMO, L. S., AND PETERSON, L. L. Experiences with network simulation. In *SIGMETRICS '96: Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* (1996), ACM Press, pp. 80–90.

[6] DINDA, P., GROSS, T., KARRER, R., LOWEKAMP, B., MILLER, N., STEENKISTE, P., AND SUTHERLAND, D. The architecture of the Remos system. In *Proc. 10th IEEE Symp. on High Performance Distributed Computing* (2001).

[7] FALL, K. Network emulation in the vint/ns simulator. In *In Proc. IEEE ISCC '99* (1999).

[8] KIELMANN, T., BAL, H. E., MAASSEN, J., VAN NIEUWPOORT, R., EYRAUD, L., HOFMAN, R., AND VERSTOEP, K. Programming environments for high-performance grid computing: the Albatross project. *Future Generation Computer Systems 18*, 8 (2002), 1113–1125.

[9] Network performance tools. http://www.caida.org/tools/taxonomy/.

[10] NOBLE, B. D., SATYANARAYANAN, M., NGUYEN, G. T., AND KATZ, R. H. Trace-based mobile network emulation. In *SIGCOMM '97: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication* (1997), ACM Press, pp. 51–61.

[11] PETERSON, L., CULLER, D., ANDERSON, T., AND ROSCOE, T. A blueprint for introducing disruptive technology into the internet, 2002.

[12] PMB - Pallas MPI Benchmarks, http://www.pallas.com/e/products/pmb/.

[13] RIZZO, L. Dummynet and forward error correction. In *In Proc. of the 1998 USENIX Annual Technical Conf* (June 1998).

[14] VAHDAT, A., YOCUM, K., WALSH, K., MAHADEVAN, P., KOSTIC, D., CHASE, J., AND BECKER, D. Scalability and accuracy in a large-scale network emulator. In *In Proc. 5th OSDI* (Dec 2002).

[15] WHITE, B., LEPREAU, J., STOLLER, L., RICCI, R., GURUPRASAD, S., NEWBOLD, M., HIBLER, M., BARB, C., AND JOGLEKAR, A. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation* (December 2002), pp. 255–270.

[16] WOLSKI, R., SPRING, N. T., AND HAYES, J. The network weather service: a distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems 15*, 5–6 (1999).

[17] ZENG, X., BAGRODIA, R., AND GERLA, M. Glomosim: a library for parallel simulation of large-scale wireless networks. In *PADS '98: Proceedings of the twelfth workshop on Parallel and distributed simulation* (1998), IEEE Computer Society, pp. 154–161.