WANCE: A Wide Area Network Communication Emulation System*

Yongguang Zhang Bharat Bhargava
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

Abstract

Experimentation in wide area computer networks such as the Internet is difficult. One difficulty is the lack of experimental hosts for individual researches or research groups. We have developed a new approach to overcome this problem. Our approach is to emulate a remote site using one local machine, and to emulate a true Internet communication environment in a local area network, by routing the inter-site communication through actual Internet hosts. This approach provides a way to conduct experiments with large scale geographically dispersed distributed systems without any designated experimental sites except for the local ones. We have studied this approach and observed that it is feasible and reliable. We have realized this approach in a communication package called the WANCE tool, which can be used to experiment distributed software in this new approach. We have also evaluated this approach by comparing it with real experiments, using a distributed database application. The relative errors we measured are within 3%.

1 Introduction

The availability of a variety of communication and networking technologies has created a great deal of interest in large scale distributed systems. Such systems are geographically wide spread and are connected by long-haul wide area networks (WAN), such as the Internet. The Internet is the only convenient WAN testbed available to the research community. The difficulties with experimentation include the lack of resources for individual researches or research groups, lack of controls in the public shared long-haul networks, the increasing chores when multiple administration units are involved, and the unpredictable dynamic nature of the Internet.

One application of large scale distributed systems that we are interested in is the transaction processing in WANs. We want to study the performance of various concurrent control, replication control, and commitment algorithms in the WAN environment and to improve them. The experiments that we plan to conduct are to run benchmark transactions in a replicated database system with widely distributed sites. The goal of this research is to develop software facilities to help such experiments, and more generally, experiments of large scale distributed systems that involves the WANs.

2 Problem Statement

We observed that the availability of large number of sites is often a problem faced by a researcher. Testing an experimental distributed system in the real-life Internet requires one to install the software in the actual sites, to run them simultaneously, and to establish communication among them. For statistics significance the experiments should be repeated using different sites.

Unfortunely, setting up an experimental site is not easy. The Internet is built bottom-up with many autonomous networks. Few people have accounts on computers of more than one different administrative domains, let alone having access to large number of computer all over the world necessitated to complete the experiment. Even some who do have access to more than one sites, privileges often differ from site to site as do their resources. Moreover, having experimental sites wide spread makes it more difficult to monitor the experiment process.

In the subsequent sections we will present a novel experimental method that will solve this problem. It is based on an *emulation* approach.

3 An Emulation Approach

An experiment that uses emulation is similar to a real experiment, but the software does not run on the remote sites. Instead, it runs in a laboratory or a local area network (LAN) environment. This kind of experiments can be set up in any LAN that has connection to the global WANs. We now illustrate how it works using an example.

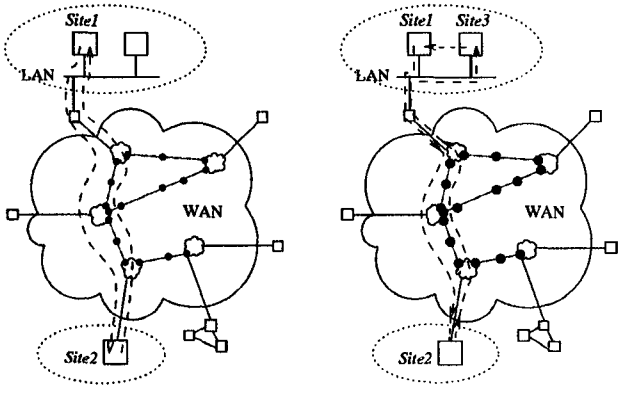
Assume that we have a two-site distributed system. To carry a task, one site will send messages to the other site; the second site will send reply messages to the first site. Suppose we want to set up an experiment with the following two sites: `raid10.cs.purdue.edu` (*Site1*) in Purdue University (West Lafayette, Indiana) and `cs.helsinki.fi` (*Site2*) in Helsinki University (Helsinki, Finland). Usually, we need to login to the both machines using `telnet` or `rlogin`,[1] run the software, and set up the experiment. In the emulation model, we run the system on the Purdue site as usual. However, we don't run the other peer in

*This research is in part supported by Army Research Lab (Software Technology Branch).

[1] Both are network utility programs that establish interactive terminal sessions with a remote host in the Internet.

0-8186-5250-0/93 $3.00 © 1993 IEEE

the Helsinki site. Instead, we can simply pick another machine at Purdue, say `raid11.cs.purdue.edu` (*Site3*), which is in the same LAN as *Site1*. A specially designed communication software for the distributed system is used so that all the communication packets from *Site1* to *Site3* are routed through *Site2* (see Figure 1). Therefore, even though both sites are close to each other and are directly connected by a LAN, the packets travel across the Atlantic Ocean to northern Europe and back. This setup makes the processing of the *Site1–Site3* system nearly identical to that of the *Site1–Site2* system.



(a) Site1-Site2 configuration (b) Site3 emulates Site2

Figure 1: Configuration of an emulation experiment

If we use this emulation approach, we can conduct distributed system experiments without having control over remote hosts. The rationale behind this approach is based on the observations about the isomorphic behaviors of the two configurations, a distributed system that spans over WANs and the same system that runs in a LAN. The difference between them is not due to the individual computers of the distributed system (as long as they are comparable models), nor on where these testbed computers are located. It is the communication performance such as the delays between the testbed computers that matters. Since they both have the same communication path over the real WANs, we can project conclusions from one system that runs in a LAN (the emulation result), to the other that runs over the real WANs (the target configuration).

Two-phase emulation technique

Using the above configuration alone is not always sufficient. In many cases request/reply messages might not have the same size, or sometimes a request message may not result in a reply message. Therefore, we need to repeat the above emulation experiment in a "*mirrored*" configuration, that is, with the pathes of request and reply messages swapped. We called this a **two-phase** emulation technique. (See Figure 2 for an example.)

In a two-phase emulation experimentation, we repeat the same experiments under two configurations. In the first configuration, all messages from *Site1* to *Site3* will be routed through *Site2*; all messages from *Site3* to *Site1* will go directly (as described above). This is the first phase. In the second configuration, all messages from *Site1* to *Site3* will travel directly in the LAN, while all messages from *Site3* to *Site1* will be routed through *Site2*. This is the second phase. We should make measurements in both phases of experiments and take the averages of the results.

4 Discussions

4.1 Comparison with other methods

There are many attempts to conduct experiments on large scale distributed processing systems in the existing WANs and studying the complexity of the Internet [9, 7, 10, 12, 11, 5, 1]. In addition to the approach presented in this paper, researchers usually conduct *real experiments* that uses the actual sites in the existing Internet. For example, research groups in Vrije University have conducted wide-area experiment using Amoeba between Cornell in the United States and Amsterdam in the Netherlands to study the transparent computing [11]. A team at Columbia University investigated Internet performance measurement methodology through experimental studies [10]. Two experiments were performed, one was to run a distributed processing facility called Camelot between Carnegie-Mellon University (CMU) and Columbia University, and the other to do Webster dictionary lookup between Columbia and University of Washington in Seattle. Long et al [9] of University of California at Santa Cruz have been studying the performance of the Internet and the distributed processing using the Internet. Parameters such as mean-time-to-failure (MTTF) and availability of the Internet sites were measured and used to develop better distributed processing software.

Real experiments and testing with the actual sites in the existing Internet are usually of small scale, due to the geographical distance, scarcity of resources, and administrative barriers. They must be conducted on the designated sites. The experiment itself cannot be scaled arbitrarily or be easily repeated on other sites. When the number of sites becomes large, more autonomous units are involved, or the selection of sites needs to be wider, or the experiments need to be repeated many times using different sites, real experiments simply become unrealistic.

Comparatively the emulation approach is easy to control and easy to implement because the processing of the distributed systems can be controlled inside a laboratory environment. It requires no designated site, since the only requirement for the emulated site is that it can bounce back the messages. No software is require to set up or run on that site. Therefore we can do experiment with virtually *any* hosts in the Internet. This enable us to obtain a large amount of data points under different configurations for statistical purposes, or to carry thorough testing using many different sites. Both would be otherwise impractical in real experiments.

Besides real experiments, *simulation* is another way to study the behavior of large scale distributed sys-

tems. In many cases, we however want to test our distributed software under real conditions, not just simulated ones. The emulation approach provides us more genuine results, and yet an inexpensive alternative to the real experimentation.

4.2 Applicability and limitation

This approach has its limitations. First, the communication time for a message from one site to another in emulation configuration does not "equal" to that in the real experiment. From Figure 1 we can see that there is overhead of a local message from *Site3* to *Site1*. However, compared with the message round trip time through the long haul WAN, the delivery time between two local sites is negligible. In fact, we will see in Section 6 that the overhead is really insignificant.

Furthermore, this scheme works only when the distributed system is not based on the clock synchronization among different sites, because of the asymmetric communication time, (the time to deliver the reply message was actually taken to return the request message). Since there has not been a practical solution to the clocks drift problem in the Internet, many distributed systems are indeed based on asynchronous coordination among sites.

Another limitation about the emulation approach is the limited horizontal scale. Since the computational part of the distributed system is not emulated, the number of sites that can participate in the experiment is bounded by the number of computers available in the LAN. Although it is possible to have a computer emulating multiple sites, the parameters will be more complicated.

5 Implementation

We have implemented a special communication software system that is used to support the emulation experiments in the Internet. It is called the WANCE (Wide Area Network Communication Emulation) tool. It is a general-purpose communication package for testing distributed software.

The functionality of the WANCE tool is to capture the communication messages and route them through a pre-defined remote site, according to the message's source and destination and the user-supplied emulation configuration. It also ensures that each message arrives at the appropriate destination after reaching the intermediate echo site.

There are many ways to implement a WANCE tool for the Internet, such as the echo service at the protocol level or the packet source routing, depending on the strategy used. A criterion for a good implementation is that it should make the emulation experiment as close to the real experiment as possible.

5.1 Routing the messages

In the emulation model, for each message originated from *Site1* and destined to *Site3*, the WANCE tool has to send it to *Site2* over the Internet and it must be bounced back to *Site3*. Although there is no generally available high-level message forwarding service in the Internet, we can use two kinds of standard Internet services to solve the problem. They are the *loose source routing* option of IP (Internet Protocol) datagram delivery, and the *echo services* in each protocol layer [6].

IP source routing Internet is a packet delivery network. The path of an IP datagram traveling from the sender host to the receiver host in the Internet is usually determined by the gateways, according to network topology and the current connectivity condition. The *loose source routing* option of the IP packet also provides a way for the sender to dictate a path of the packet through the Internet. It is a sequence of Internet addresses specifying that the packet must follow the sequence of Internet addresses before reaching the destination. This feature of Internet delivery can be used to redirect messages in the WANCE tool. For example, if we want to route a message from source *Site1* through *Site2* to destination *Site3*, we can specify the loose source route option "*Site1*, *Site2*, *Site3*" in every packet of the message before sending it out from *Site1*.

One limitation of the IP source routing technique is its availability. Although source routing is a standard option of IP delivery that every gateway in the Internet is supposed to conform, some implementations of Internet gateways choose to ignore it for the sake of performance. Some even drop those packet that carries the source routing option. Many others, although doing source routing, put packets that carry the source routing option into queues with lower priorities. Such packets may travel slower than others.

Echo services We can also use the echo services in each layer of the Internet protocol, IP (*Internet Protocol*, the fundamental layer), UDP (*User Datagram Protocol*), and TCP (*Transmission Control Protocol*). IP includes ICMP (*Internet Control Message Protocol*) as an integral part that handles error and control messages. The echo service of IP level is implemented by the ICMP echo request/reply messages. Any gateway or host that receives an echo request addressed to it will formulate an echo reply and return it to whichever machine sent the request. UDP adds a port number to the IP layer to distinguish multiple applications running in a same machine. Port number 7 is reserved for the UDP echo services. Whenever a packet arrives at that port, the receiving host sends the identical packet back to the sender. The same service is also implemented in the TCP layer. A program can get back what it writes to a remote host using TCP as long as it connects to the port 7 of the remote host. These echo services returns a message that is identical to what one sends[2], only with a delay of the round-trip communication time across the Internet, from the sender to the echo host.

All these echo services in the three layers can be used in the emulation system to redirect messages. For example, to route a message from source *Site1* to destination *Site3* through *Site2* before reaching to the

[2]Some implementations of UDP echo service restrict the size of the packet. For example, the maximum size of a UDP echo packet allowed in SunOS is 1024 bytes.

destination, *Site1* can simply send the message to *Site2* using IP, or UDP to *Site2*:7/UDP, or *Site2*:7/TCP, depends on the communication protocol used in the distributed system.

Usually the message will get back to the sender (*Site1*). To make it reach the destination (*Site3*) instead, the WANCE tool in *Site1* simply picks up this returned message and directly passes it to *Site3* (which is in the same LAN). Although extra overhead is involved here, the time for the message to travel over the Internet dominates in the total time from *Site1*, to *Site2*, back to *Site1*, then to *Site3*. The overhead for the last leg is a small constant and negligible.

An alternative implementation is to set up a special forwarding program running in the gateway of the LAN. All messages are sent to the gateway first. The gateway does the redirection by sending them to *Site2* and upon receiving the echo messages, passes them to the receiver.

5.2 WANCE tool

We implemented the WANCE tool using echo services instead of source routing since the latter is not always available in the gateway of our Purdue network to the Internet. We developed the WANCE tool on SunOS. It is based on Unix interprocess communication primitives and consists of a library of C routines and a set of data collection and analysis programs. For emulation experiments, each module or server of the distributed software must be linked against the WANCE library first. The library functions as a communication subsystem that sits in between the distributed software modules and the communication primitives provided by Unix. It provides the same interface for the upper layers but replaces the original communication primitives with special message redirection semantics. A variety of distributed system may be linked to the tool with little or no change to the code.

A configuration file is supplied and is read by servers when they start up. It defines the mapping from a local site in the LAN (the emulating site) to the corresponding real host in the Internet (the site being emulated). The WANCE library routines in the server will route the messages of certain destinations according to the configuration file. The configuration file contains lines like

site1 site3 host2

where site1, site3 are logical addresses of the servers, and host2 is a physical address of the computer that is emulated by the site where it runs. For the example in Section 3, site1 is the one that runs on `raid10`, site3 on `raid11`, and the emulated host2 is `cs.helsinki.fi`.

With the WANCE tool, we can easily set up experiments for our experimental distributed software. We have linked the Raid system[3] with WANCE and have been able to conduct experiments on the Internet. The integration was smooth. Not a single line of the Raid server code needs to be changed.

6 Evaluation

Finally, before we rely on the emulation results and draw conclusions for the real WANs, we should conduct error estimation on the emulation method first. The question is how close is this emulation method to the real experiment.

To find out the the relative error involved in the emulation method, we have conducted a set of estimation experiments. We ran a real experiment and an emulation experiment side-by-side and compared the two measurements. The same benchmark and input parameters were used for both experiments.

Procedure We used a two-site Raid[3] distributed database system as the application in the experiments. We used a banking transaction benchmark called DebitCredit (aka TP1, ET1)[8]. This is a simple yet realistic and well-accepted. More detail of the application and benchmark can be found elsewhere [4, 8].

The database is fully replicated at both sites. We used Read One Write All protocol in the replication control, two phase locking protocol in the concurrency control, and two phase commit protocol in the atomicity control. We set the concurrency level to be one, i.e. one transaction at a time, since we liked to distinguish the cause of transaction failures due to conflict or due to the unreliability of the Internet.

We used three Sun workstations in the experiments. Two of them were located in our laboratory. We refer them as `raid4` and `raid11` here. `raid4` is a SPARC IPC and `raid11` is a SPARCstation-1. The third workstation, also a SPARC IPC, was physically located in Hong Kong[4][5]. We refer this machine as `ust`. All these machines were properly equipped for distributed database application. Both `raid4` and `ust` had the same configuration and ran the same version of SunOS 4.1.1, for the purpose of fair comparison between emulation experiments and real experiments.

Figure 2 shows the configurations of the real experiment and the emulation experiment. In the real experiments, site 1 of the two-site database system runs in `raid11` and site 2 runs in `ust`. In the emulation experiments, site 2 runs in `raid4` instead. Since Raid uses UDP as its transport protocol, we put `7/UDP` as the echo service. In phase I of the emulation experiment, messages from site 1 to site 2 are redirected to `ust`, while in phase II, messages from site 2 to site 1 are redirected. This is to study how much bias will be introduced if request and reply messages are not of the same length.

[3] Raid is a distributed database system developed in Purdue. We have been using it as an excellent experimental infrastructure for our researches in communication, adaptability, reliability, transaction processing over several years [3, 2]. The interactions between Raid servers are in form of request/reply messages following the "client/server" paradigm, handled by the Raid communication subsystem.[4]

[4] It is the courtesy of the Department of Computer Science, the Hong Kong University of Science & Technology for allowing us to use their facility for collaborative research.

[5] As a side note, the geographical distance between the two sites in our laboratory and the one in Hong Kong is about 8 thousand miles. The Internet connection between them are through Chicago, San Francisco, and then across Pacific Ocean to Hong Kong.

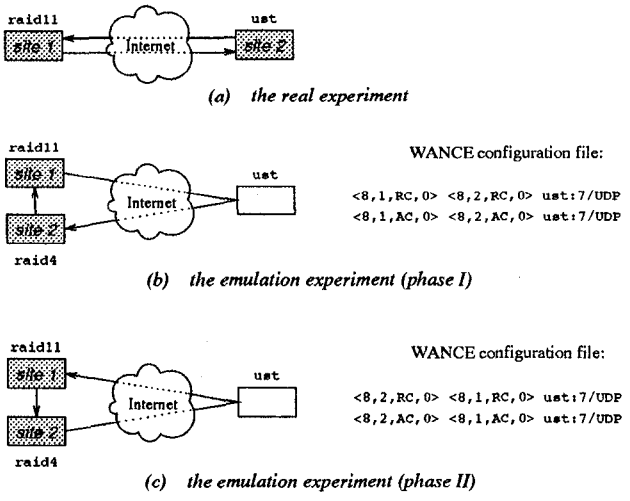


Figure 2: Three configurations of the experiments for error estimation

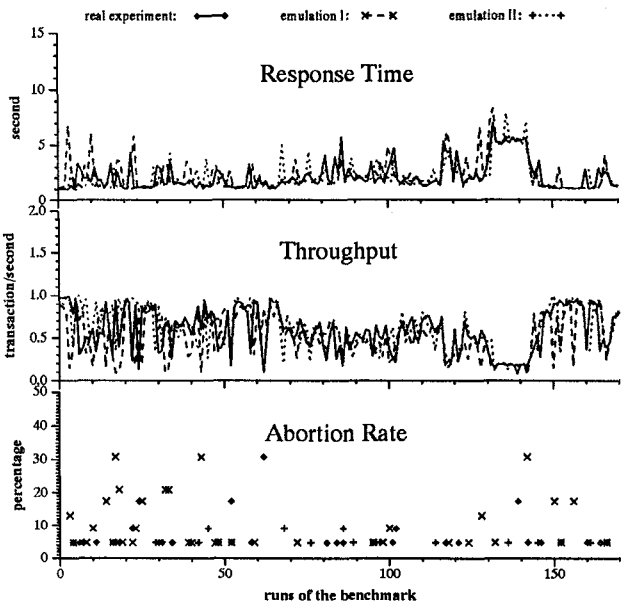


Figure 3: Three configurations for error estimation

We ran the experiments in a weekend when the network load was low. In each run of the experiment we fed the stream of 20 transactions generated from the benchmark into site 1. We recorded the response time for each transaction, and calculate the average response time of the 20 transactions. We set the timeout to be 2 second for each operation, and the maximum number of restarts of a failed transaction to be three. The experiment was repeated 171 times for each of the three configurations: a real experiment, followed by phase I of the emulation one, and the phase II. They were interleaved to ensure that they were under the comparable Internet communication conditions.

Data We measured three important performance metrics about distributed transaction processing, the estimated mean response time, throughput, and transaction abortion rate. The response time is the elapse time between submission of a transaction to the system and the result (or failure) returned by the system. The throughput is the average number of transactions completed by the system in a second. The transaction abortion rate means the percentage of transaction that cannot be finished due to the failure of the message delivery in a specific time in the Internet. Figure 3 shows the samples of the response time (RT), throughput (TP), and the abortion rate (AR) for each run of the benchmark transactions.

From Figure 3, we can see that in the samples acquired in both phases of the emulation experiments are very close to that acquired in the real experiments. We calculated the estimated mean times of RT, TP, and AR and the standard deviations. (see Table 1). By taking the average of the numbers acquired in both phases of the emulation experiments we got "combined samples" for the emulation experiment. These data were compared to the data acquired from the real experiments and the relative errors[6] were also computed. Table 1 shows such estimated errors.

unit: RT: second, TP: per second, AR: %

methods		real	emulation experiment		
		expr	phase I	phase II	combined
RT	(1)	2.145	2.331	2.076	2.204
	(2)	-	8.7%	-3.2%	2.7%
	(3)	1.294	1.486	1.322	1.255
TP	(1)	0.581	0.534	0.617	0.576
	(2)	-	-8.2%	6.1%	-1.0%
	(3)	0.250	0.255	0.250	0.218
AR	(1)	1.176	2.153	0.688	1.420
	(2)	-	83.1%	-41.5%	20.8%
	(3)	0.036	0.057	0.019	0.031

(1) mean, (2) relative error, (3) standard deviation

Table 1: The estimated errors

Results From Figure 3 and Table 1 we can see that the emulation method introduces small relative errors to the experimental data. For both response time and throughput data, the relative errors are within ±3%. The standard deviations are also very close to each other.

We noticed that neither phase I nor phase II of the emulation experiment alone is sufficient due to the

[6] $(d-r)/r$, where d is the datum from the emulation experiment and r is the datum from the real experiment.

unbalanced message traffic in either direction. We examined the message log between the two Raid sites. For one benchmark transactions stream, 36 messages, a total of 7504 bytes (including message headers, 36 bytes per message, ditto below), were sent from site 1 to site 2. Another 36 messages, a total of 2880 bytes, were sent from site 2 to site 1, which were only one-third the size of the other direction. This is because all the transactions were injected into site 1, read data were fetched locally, but all writes had to go to site 2 as well. In a real experiment, a total of 10384 bytes traveled across the Internet. But in emulation experiment phase I 15008 bytes and in phase II only 5760 bytes were transmitted. Since small messages travel slightly faster than large messages, the transaction response times measured in the real experiment were shorter than those measured in the emulation phase I but longer than those in phase II. Similar situation applies to the throughput measurements. The average of both phases was however closer to the real data since the average number of messages and the average size of each message are the same as in the real experiments.

The transaction abortion rate measurements in emulation experiments were much less accurate. This is because transaction abortion is one kind of independent failures, which will be much harder to measure than the response time or the throughput. Message loss in Internet is often transient.

The experiments clearly show the high fidelity of the emulation method on the measurements of distributed transaction processing performance, including metrics like response time and throughput. Although it is unrealistic to achieve 100% accuracy, we have shown that the emulation experimental method is feasible and reliable.

7 Conclusion

We have shown that the emulation approach can ease the task of experimentation with the help of the WANCE tool. The major advantage of the experimental approach we presented is that no designated experimental site is required in an experiment. Virtually one can do experiments with any host in the Internet by utilizing the builtin routing or echoing facilities. Although it still "steals" a few CPU cycles of the emulated host, it uses the minimum resource required for experimentation with true communication, compared with the otherwise heavy load in a remote host in the real experiment.

We have also discussed our implementation of this approach in the WANCE tool, and shown that it could provide reliable results. The use of this tool can ease the tasks of experimental setups and free us from many tedious chores in the experimentation so that we can concentrate more on the observations and data analysis.

With the access of the Internet and the emulation tool, we are able to conduct more experimental studies on large scale distributed transaction processing in WANs. We believe that we will have better observations and better understanding of the natures of distributed processing in WANs, with the help of the experimental infrastructure described in this paper.

Authors' e-mail addresses:

ygz@cs.purdue.edu, bb@cs.purdue.edu

References

[1] Ashok K. Agrawala and Dheeraj Sanghi. Network dynamics: An experimental study of the Internet. In *Proceedings of GLOBECOM'92*, Orlando, FL, December 1992.

[2] Bharat Bhargava, Karl Friesen, Abdelsalam Helal, and John Riedl. Adaptability experiments in the Raid distributed database system. In *Proc of the 9th IEEE Symposium on Reliability in Distributed Systems*, Huntsville, Alabama, October 1990.

[3] Bharat Bhargava and John Riedl. The Raid distributed database system. *IEEE Transactions on Software Engineering*, 15(6), June 1989.

[4] Bharat Bhargava, Yongguang Zhang, and Enrique Mafla. Evolution of communication system for distributed transaction processing in Raid. *Computing Systems*, 4(3):277–313, Summer 1991.

[5] Ramón Cácceres, Peter B. Danzig, Sugih Jamin, and Danny J. Mitzel. Characteristics of wide-area TCP/IP conversations. In *Proc. of ACM SIGCOMM'91 Conf*, pages 101–112, Zurich, Switzerland, September 1991. in Computer Communication Review 21(4).

[6] Douglas E. Comer. *Internetworking with TCP/IP*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[7] Richard Golding and Darrel D. E. Long. Accessing replicated data in a large-scale distributed system. Technical Report UCSC-CRL-91-01, University of California at Santa Cruz, January 1991.

[8] Jim Gray, editor. *The Benchmark Handbook for Database and Transcation Processing Systems*. Morgan Kaufmann, San Mateo, CA, 1991.

[9] D. D. E. Long, J. L. Carroll, and C. J. Park. A study of the reliability of Internet sites. In *Proceedigs of the 10th Symposium on Reliable Distributed Systems*, pages 177–186, Pisa, Italy, September 1991. IEEE.

[10] Calton Pu, Frederick Korz, and Robert C. Lehman. A methodology for measuring applications over the Internet. Technical Report CUCS-044-90, Columbia University, September 1990.

[11] Robbert van Renesse and Hans van Staveren. Wide-area communication under Amoeba. Technical Report IR-117, Dept of Mathemactics and Computer Science, Vrije Universiteit, December 1986.

[12] Robbert van Renesse, Hans van Staveren, and Andrew S. Tanenbaum. Performance of the world's fastest distributed operating system. *Operating System Reviews*, 22(4):25–34, October 1988.