Performance of a Distributed Object-Based Internet Collaboratory *

G. Robert Malan, Farnam Jahanian,
Craig Rasmussen, Peter Knoop
University of Michigan
Department of Electrical Engineering and Computer Science
1301 Beal Ave.
Ann Arbor, Michigan 48109-2122

Phone: (313) 936-2974
Fax: (313) 763-1503

E-mail: {rmalan,farnam}@eecs.umich.edu
{rasmussn,knoop}@umich.edu

Abstract

This paper describes an experimental assessment of the Upper Atmospheric Research Collaboratory (UARC), an object-based distributed system. For the past three years, the UARC software system has enabled space scientists to collaborate on atmospheric experiments in real-time over the Internet. The UARC system provides a means for its users to view data from remote atmospheric instruments, share annotations of that data, discuss experimental results in a chat window, and simultaneously edit text manuscripts. However, UARC's distribution of atmospheric data to this geographically dispersed group of scientists is its primary mechanism for effecting their collaboration. This paper investigates the impact of UARC's implementation as a large distributed object-based software system as a means for supporting wide-area collaboratories. Specifically, it focuses on the communication performance and scalability of its object-based data distribution mechanism. First, Internet microbenchmarks are presented which characterize the UARC topology; then the results of application-level experiments are described that investigate UARC's use of NeXTSTEP's Distributed Object method invocations as a communication primitive. Finally, an analysis and discussion of the UARC system's object-based implementation concludes the paper.

Keywords: Distributed Systems, Internet, Network Performance

1 Introduction

The Upper Atmospheric Research Collaboratory (UARC) project is a multi-institution research effort, whose focus is the creation of an experimental testbed for wide-area scientific collaboratory work [4]. This testbed is implemented as a large object-based distributed system on the Internet. The UARC system provides a collaboratory environment in which a geographically dispersed community of space scientists perform real-time experiments at a remote facility in Greenland. Essentially, the UARC project enables this group to conduct team science without ever leaving their home institutions. This community of space scientists has extensively used the UARC system for over three years, and has expressed a high degree of satisfaction with its mechanisms for remote collaboration. During the winter months, a UARC campaign takes place almost

*This work is supported by the National Science Foundation under the cooperative agreement IRI-92-16848.

every day; the scientists use the term *campaign* to denote one of their experiments. This community has grown to include regular users from such geographically diverse sites as SRI International in Menlo Park, California; the Danish Meteorological Institute; the Universities of Alaska, Maryland, and Michigan; and the Lockheed Palo Alto Research Laboratory.

The UARC system provides a variety of services to its users including shared window data displays, multiparty text chat windows, a shared annotation database, and a distributed text editor. However, the primary mechanism for collaboration is the distribution of atmospheric data to the experiment's participants. This data is collected at a remote site in Kangerlussuaq, Greenland, and is distributed over the Internet to the scientific collaboratory. The types of data distributed include radar, Fabry-Perot interferometers, Allsky-imagers, and magnetometers. This data is then displayed to the scientists in real-time during which additional collaboration can take place, such as annotation of the data, discussion of the results, or the remote direction of the instrument operators in Greenland. All of this occurs over the Internet.

Over the last several years, during the exponential growth of the Internet, the UARC scientists have noticed a marked decrease in the system's performance. This paper investigates the impact of UARC's implementation as a large distributed object-based software system on its communication performance. Specifically, it focuses on the performance and scalability of its object-based data distribution mechanism; for additional information on the UARC system's tools and architecture, refer to [9, 12, 18, 19, 21, 24, 26, 27]. First the results of a series of Internet UDP experiments are presented that describe the UARC topology's end-to-end loss and latency characteristics. After which, the results from a series of UARC application-level experiments are presented. These experiments identify the system's application-level characteristics; and quantify the effects of implementing its data distribution mechanism with NeXTSTEP's Distributed Object subsystem. Specifically, the paper shows that by failing to control its communication policy and by relying on the relatively inflexible policy set by the underlying Distributed Object subsystem, UARC's performance and scalability suffers. The remainder of the paper is organized as follows: Section 2 describes the overall UARC architecture; Section 3 outlines the experimental apparatus; Section 4 details the Internet UDP experiments; Section 5 contains the results of the application-level UARC experiments; and finally Section 6 provides a discussion and conclusion.

2 UARC Architecture

The UARC 5.0 software system uses three separate components to distribute atmospheric data: UARC Data Suppliers, the UARC Server, and UARC Clients. Figure 1 shows how these components work together. The Data Suppliers are located near the atmospheric instruments in Greenland. They package the raw data and send it over the Internet to a UARC server at the University of Michigan. This server then distributes the data to all the connected clients. Generally, each atmospheric instrument is associated with its own Data Supplier during a UARC campaign. The total number of UARC Clients connected to the Server varies, depending on the level of interest in the experiment. To support the availability of data, a backup server takes over the data distribution service if the primary UARC Server fails.

NeXTSTEP and Objective C [22], were chosen as the UARC system's platform and implementation language because they provided a rich software development environment. This environment facilitated the creation, maintenance, and extension of the wide variety of collaboratory tools the UARC system provides. Their support of objects at the system level made the development of familiar user interface idioms such as the ability to drag-and-drop data objects between UARC's collaboratory tools simple. It was natural to combine the three UARC components using NeXTSTEP's Distributed Object subsystem [23] to form a single distributed system. Thus, in UARC 5.0, Distributed Object method invocations are used to transparently send atmospheric data between the components over the Internet. These method invocations are implemented as reliable TCP streams. As such, the UARC system effectively provides a multicast data distribution service using multiple point-to-point network connections. This is similar to the graph-based `NNTP` protocol [10]; data distribution over the World Wide Web using `HTTP` [1]; and the Unidata weather distribution system [8].

The topology covered by the UARC system represents a broad cross section of the Internet's current

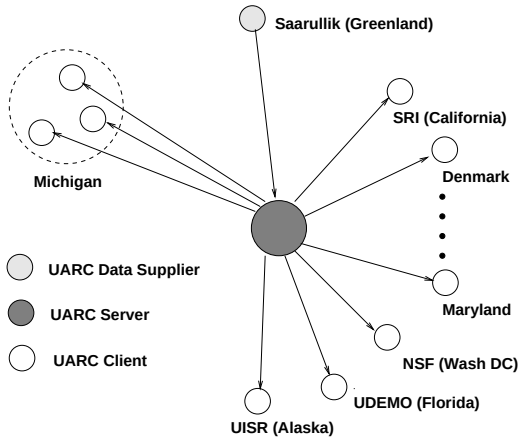

Figure 1: UARC 5.0 Architecture

physical network layer. The Data Suppliers in Greenland communicate with the rest of the Internet over a slow serial line interface (56 Kbaud) to a high bandwidth, high latency satellite connection to Goddard Air Force Base. The UARC Server, several space scientists, the UARC software development team, and administrative support are all located at the University of Michigan, a large network composed of multiple 10Mbps ethernet LANs connected by a 100Mbps FDDI ring. The remainder of the UARC community is distributed throughout North America and Europe, with high concentrations of space scientists in Denmark, California, and Florida. They are all connected by varying degree to the high speed, low latency Internet backbone.

3 Experimental Setup

A broad range of experiments were conducted to determine the UARC system's performance characteristics. These experiments fell into two distinct categories: those constructed to determine the characteristics of the underlying UARC network topology, and experiments designed to evaluate the application-level performance of UARC under various network conditions. The first group of experiments consisted of tests that measured the packet loss and latency over the UARC Internet toplogy, as well an experiment that determined this topology's routing stability. The second group of experiments measured the UARC system's overall message latencies and loss statistics. These experiments characterized the overhead of NeXTSTEP's Distributed Object subsystem and TCP in the UARC system's Data Suppliers, Server, and Clients.

The machines used in our experiments also fall into two distinct categories: machines local to the University of Michigan, and those distributed over the internet. We make this distinction because the UARC server is located at the University of Michigan. These machines and their locations are listed in Table 1.

4 Underlying Network Characterization

Before we began our application-level UARC experiments, we wanted to quantify the Internet's congestion by measuring the end-to-end loss and latency characteristics of UARC's distributed system topology. We knew that the UARC system performed significantly worse during the day (Eastern Daylight Time) than at night. However, we felt that quantifying these statistics were important to a full understanding of UARC's Distributed Object and TCP behavior. Three long-term experiments were conducted to measure these characteristics: an experiment that measured the packet loss between the UARC Server and Client hosts; an experiment that measured the packet latency between these hosts; and finally, an experiment which determined the stability of UARC's Internet routing topology.

Name	Location	Processor	Memory
ariadne	U Michigan (EECS)	Intel Pentium	16M
jupiter	U Michigan (EECS)	HPPA (7100LC)	32M
oosik	U Michigan (SPRL)	MC68040	16M
pluto	U Michigan (EECS)	MC68040	16M
pollux	U Michigan (CREW)	MC68040	32M
saturn	U Michigan (EECS)	MC68040	16M
uranus	U Michigan (EECS)	MC68040	16M
viera	U Michigan (SILS)	Intel 486	24M
allsky	Lockheed	MC68040	16M
dmistp	Denmark	MC68040	16M
oberon	SRI (California)	MC68040	32M
saarullik	Greenland	MC68040	16M
uiris	U Maryland	MC68040	16M
ugeo	NSF (Washington DC)	Intel 486	24M
uisr	U Alaska	Intel 486	24M
udemo	Florida	MC68040	16M

Table 1: Machine characteristics used in experiments.

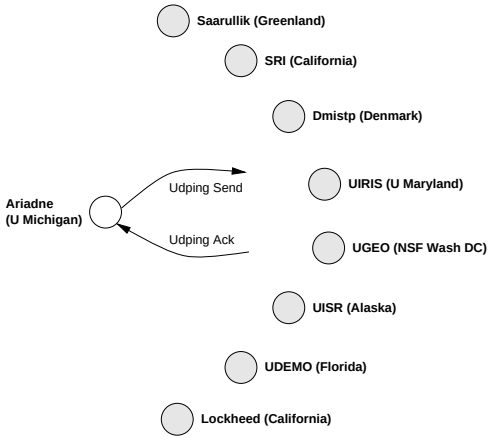

Figure 2: Setup for Udping test

4.1 Characterization of Packet Loss in the Topology

A characterization of the raw IP packet loss in a time-dependent fashion was the focus of this experiment. It was necessary to understand the raw packet loss in the UARC's Internet topology to adequately explain the behavior of the system's Distributed Object implementation. At first we used `ping` to measure this loss, however there was some concern that congested routers would deal differently with ICMP packets than IP data packets and thereby inflate the loss statistics. We then built and deployed a UDP-based ping program, `udping` which is similar to previous measurement programs [2, 28]. The general `udping` experimental setup is shown in Figure 2.

The experiment consisted of sending a `udping` packet from the UARC server to several UARC Clients and Data Suppliers every ten seconds. This interval was chosen so that the probability that a packet was lost would be relatively independent of the other packets in the experiment. The experiment was conducted over a period of approximately one month. Table 2 lists the machines that participated in the experiment.

Name	Location
allsky	Lockheed
ariadne	U Michigan (EECS)
dmistp	Denmark
oberon	SRI (California)
pollux	U Michigan (CREW)
saarullik	Greenland
umd	U Maryland
ugeo	NSF (Washington DC)
uisr	U Alaska
udemo	Florida

Table 2: Participants in udping packet loss and latency experiments

The graphs in Figure 3 summarize the round trip packet losses to a representative subset of the participants. Each point in the figure represents the packet loss experienced over a ten minute interval. The gaps in the graphs indicate periods during which the test was stopped, either for UARC campaigns, or a due to a crash of the server machine. It is clear, from an examination of the graphs, that packet loss is dependent on the time of day, and the day of the week. About 11:00am to 10:00pm EDT seems to be the most congested period of the day, corresponding to the general office hours of the continental United States. This would seem to confirm one space scientist's opinion that, "Night time is a good time for science!" Weekends appear much less congested than the weekdays. Keep in mind that a lost packet in Figure 3 represents a UDP datagram packet that did not make it *both* directions, and does not represent a unidirectional loss statistic.

An interesting feature of Figure 3 are the vertical bars which indicate a failure along the route. During these intervals, neither the `udping` server or client machines have crashed, yet for ten minutes at least 60 consecutive `udping` packets are lost between the hosts. By comparing the times of these occurrences between the four graphs, one can tell which parts of the internet are reachable from Michigan, and which are disconnected. A detailed analysis of these results are beyond the scope of this paper, but are nonetheless interesting. Finally, the data suggest that sustained round-trip packet losses of approximately 5% to 20% occur each weekday during the United States' afternoon and early evening, a troubling measurement of the Internet's load. For our purposes, it is clear that there are two main times to run application-level experiments: during the busy daytime hours, and during the relative calm of the late night and early morning.

4.2 Characterization of Packet Latency in the Topology

This experiment characterized the raw IP packet latency over the UARC Internet topology in a time-dependent fashion just as the previous test characterized the packet loss. Again, this raw packet latency is helpful in understanding the higher-level behavior of distributed objects. In this experiment, ten `udping` packets were sent 300ms apart every five minutes from the UARC server to the Clients and Data Suppliers listed in Table 2. The interval of 300ms was chosen so that it would be short enough to keep the remote `udping` server in memory, and long enough to not overflow intermediate router buffers. The round-trip time was computed by sending a timestamp in the `udping` packet, and then comparing it to a second timestamp taken upon receipt of the echo. A sample of the results from one such experiment is shown in Figure 4. This figure summarizes the round trip UDP datagram latency for a representative subset of the experiment's participants. As in the previous figure of packet loss, the gaps in these graphs are due to a stoppage of the experiment. Note that the latency scale for `dmistp`, the host in Denmark, is an order of magnitude greater than those in North America. This reflects the observations of the space scientists in Denmark, who consistently experience high latencies in their collaboratory actions. Again, there seems to be a correlation between latency and the time and day of the week. However, there are some anomalies, such as the behavior

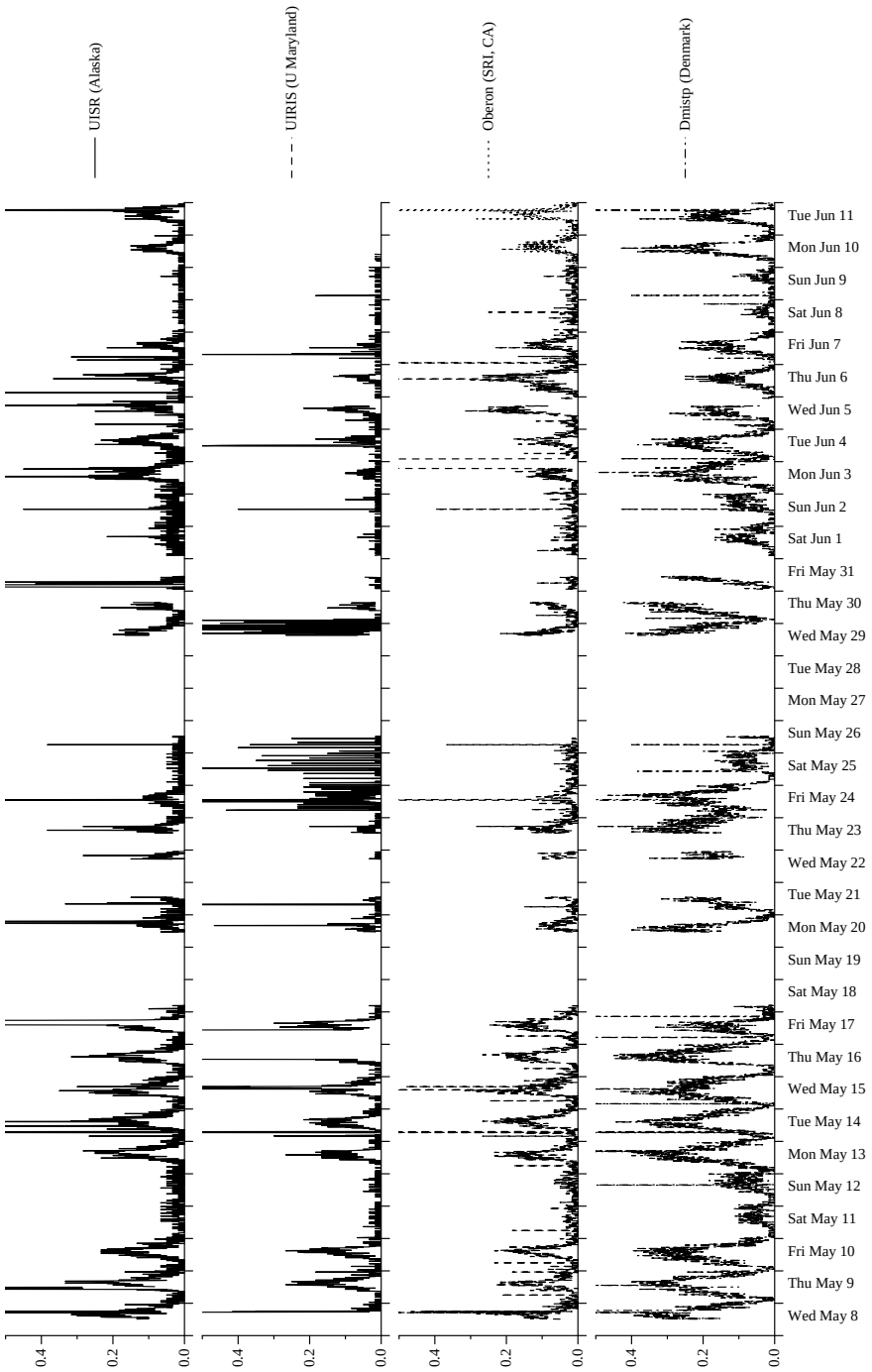

Figure 3: Measured round-trip UDP datagram packet loss as a percentage of ten minute aggregation units between a UARC Server host at the University of Michigan (`ariadne`) and several UARC Client hosts during May and June 1996.

of `oberon` on Thursday June 6, that we cannot explain. These provide anecdotal evidence for some of the seemingly random behavior observed in our distributed system by the space scientists.

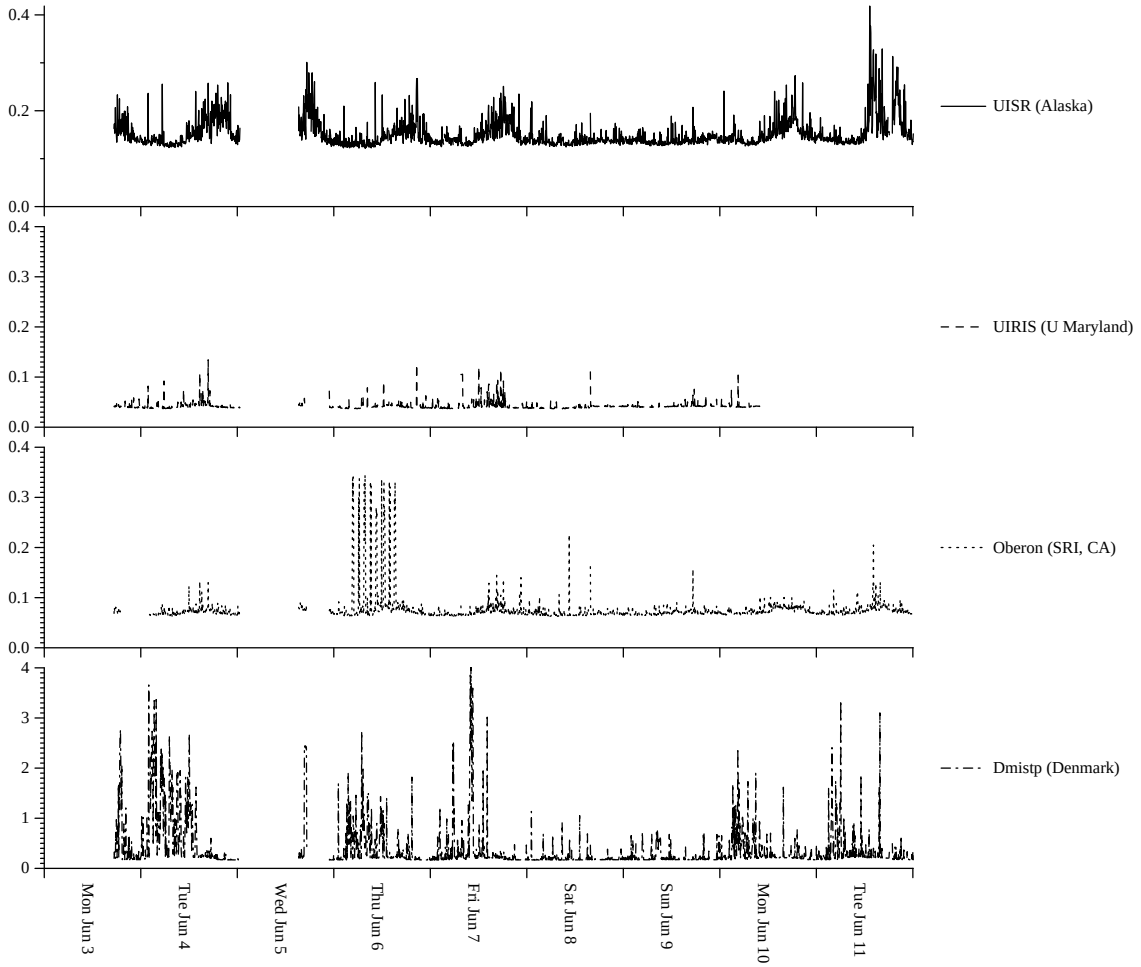

Figure 4: Measured UDP datagram round trip latency, in seconds, between the University of Michigan (`ariadne`) and several UARC Client hosts during June 1996. using `udping`.

4.3 Topological stability

Although the results of the previous two experiments showed fairly consistent loss and latency characteristics, we felt that our intuition about UARC's underlying Internet topology would be strengthened by some knowledge of its routing stability. To this end, we conducted an experiment using the `traceroute` facility. At periodic intervals during the experiment (typically 2-4 hours), simultaneous bidirectional traceroutes were started between the UARC Server and a Client. The results of this experiment showed that for almost all of the Server to Client routes in the topology, a stable, symmetric, route exists with symmetric latencies. That is, there is a single dominant route between the two points that did not change for the duration of our experiment, and that packets sent from the Server to a UARC Client, would return over roughly the same route. The only exception to this was the route from the University of Michigan to the Client host in Denmark (`dmistp`), which for a period of about a day took a longer symmetric route than at any other time. This experiment confirmed that most of the variation in packet loss and latency from the previous two experiments were primarily due to congestion, and not to unstable routes. The goals for this experiment were to look only at the stability of UARC's portion of the Internet, and were not meant to provide a general characterization of Internet routing stability as described by Paxson [25].

5 UARC 5.0 Experiments

With a solid understanding of the UARC topology's Internet characteristics, application-level UARC experiments focused on investigating its Distributed Object implementation were undertaken. The goals of these experiments were: to determine the consequences of using a high-level language construct, namely object method invocations, as a mechanism for supporting asynchronous message passing; to quantify the effects that slow, poorly connected clients have on fast, well-connected ones; to measure the system's performance under the current space science campaign parameters; and finally to determine the scalability of UARC's data dissemination service through a series of stress tests. As was discussed in Section 2, the UARC system consists of Data Suppliers, a Server which acts as a distribution point, and any number of Clients. With these components, the significant parameters in the UARC data distribution experiments were:

- The *Number of UARC clients* connected to the server.
- The *Placement of UARC clients* at either hosts local or remote to the UARC Server.
- The *Placement of UARC Server* at various hosts. For all of our experiments, the UARC Server was local to Michigan due to administrative constraints.
- The *Number of Suppliers* which supply test data.
- The *Bandwidth of a Supplier*, varied by setting the amount of data to supply, and the period over which it sends.
- The *Placement of Suppliers* at either local or remote hosts.
- The *Underlying network conditions* for a given experiment.

The primary statistics collected during our UARC experiments were:

- The *Message Latency* between points in the distribution tree: Suppliers, Server, and Clients.
- and *Message Loss* between these points.

The use of the term *message* in these statistics comes from the fact that the Distributed Object method invocations are only used to send messages between the UARC components. The term *message* is easier to use, and will represent the action of invoking one of these methods in the remainder of the paper.

Figure 5 shows the general setup for the application-level experiments. The Data Suppliers are either in Greenland (`saarullik`), or local with respect to the UARC Server in Michigan (`pollux` or `saturn`). The UARC Server feeds the data messages to a varied number of Clients which are scattered both locally and globally. A timestamp is written into the data message's contents as it flows down the distribution tree. Each message's timestamps are written to a log file upon receipt at the UARC Server and Clients. By analyzing these log files offline, the experiment's message loss and latency statistics can be extracted.

Since the analysis of data in these experiments relies on timestamp information, a method for synchronizing the clocks on the system's hosts must be applied. All of the machines in these experiments were running `xntpd` which could not be disabled due to administrative constraints. In order to compensate for skewed clocks, at periodic intervals during the testing, a snapshot of the clock skew was taken. These snapshots were taken using a probabilistic clock synchronization technique, similar to the protocols developed by Cristian [3]. Specifically, the `udping` client and servers were used to exchange timestamp information between two hosts at one second intervals for two minutes. From this information the mean round trip time between the hosts along with its standard deviation was computed. Assuming the routes were symmetric, which the traceroute data indicated, the clock skew can be determined by taking the difference in timestamps between the two hosts and subtracting half of the average roundtrip time. This time skew between hosts was then used to normalize the timestamp logs to a single clock. Multiple snapshots were taken during the experiments to verify that the clocks didn't change significantly.

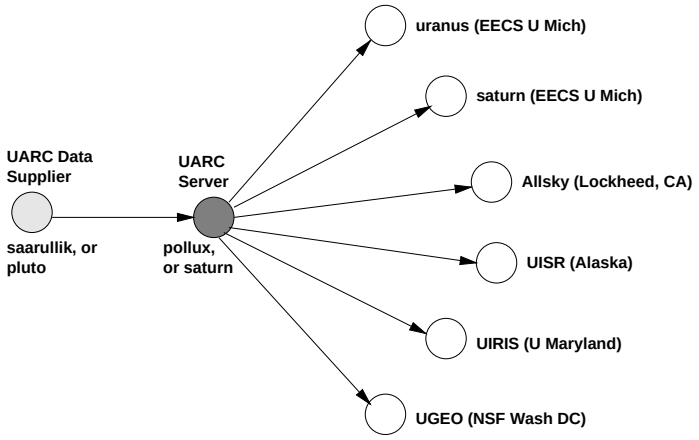

Figure 5: UARC 5.0 experimental setup

5.1 Effects of Distributed Objects

This first set of experiments is used to show the immediate consequences of using Distributed Object method invocations as the UARC system's message passing primitive. Again, the reason for this choice of mechanism grew out of the broader system implementation as an object-based distributed system. This choice made the development of high-level collaboratory tools, such as the shared window support, data augmentation, and user interface idioms such as drag-and-drop, much easier. Not coincidentally, it made the development of the UARC data distribution mechanism simpler as well. This experiment, along with the following sets of experiments, quantifies the costs of that design choice.

A more detailed understanding of UARC's Distributed Object implementation is helpful in interpreting the results of our experiments. In the UARC system, instrument data is represented by an Objective C object. To send this data to a remote machine, a Distributed Object method invocation is used with the instrument data object as a parameter. Both the Supplier and Server use this method invocation mechanism to send UARC data down the distribution tree. To increase performance, these methods are defined as asynchronous invocations with no responses, or `oneway void` methods in Distributed Object parlance. That means that the method invocation returns immediately upon delivery to the Distributed Objects subsystem, and does not wait for a return value. This is equivalent to sending an asynchronous message. This `oneway void` construct is one of the few ways the UARC application code is able to control its communication policy. In this case it was able to chose between synchronous and asynchronous message passing mechanisms.

In general, the UARC system has no control over the communication policies that Distributed Objects uses for message transport. The Distributed Objects transport mechanism is derived from the Mach Netmsg server [17, 29], which in turn uses TCP for interhost communication. When a local object invokes a remote method, the Distributed Object subsystem creates a persistent TCP connection between the two hosts. This connection is used for all communication between the two objects, including subsequent method invocations. This avoids unnecessary TCP setup and teardown costs for repeated method invocations. When a remote method's activation record is delivered to the Distributed Objects subsystem, it is eventually placed on a queue that is bound to the persistent TCP port. These method invocations are delivered in a FIFO order to the remote application's respective object. It is in this way that the UARC data is disseminated: first a Distributed Object method invocation copies the atmospheric data object from the Supplier to the Server; then the server makes another method invocation that copies the data object to the Client. Since the Distributed Object layer uses TCP for its transport, the natural question arises: how and where are data messages lost? To answer this, several experiments were conducted.

The first experiment overloads the bandwidth on the link between the Supplier and Server to determine its loss characteristics. Specifically, a UARC Data Supplier was started on `UISR` in Alaska that supplied a Server and Client on `saturn` at the University of Michigan. The TCP throughput between `UISR` and `saturn`

was informally measured to be about 25.36 KB per second at the time of the experiment using FTP. The Supplier fed 40000 bytes of data once per second, which was roughly twice the link's capacity. Figures 6 and 7 show the results from the first four minutes of this experiment.

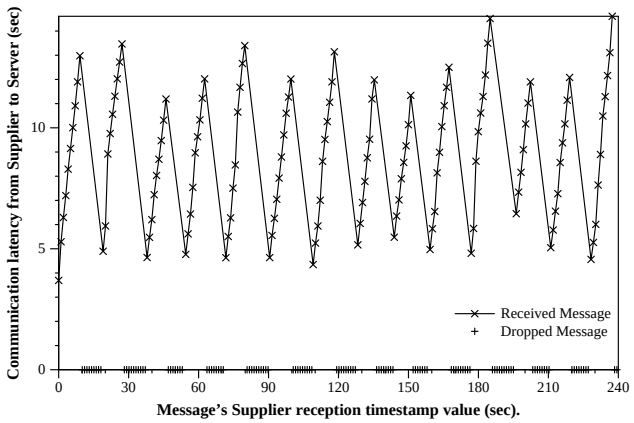

Figure 6: The effects on communication latency of overloading the network bandwidth between a UARC Supplier and the Server as seen from the application layer.

The points in Figure 6 represent messages that are successfully transferred from the Supplier to the Server; or in object parlance, the Supplier's remote method invocations that are executed on the Server. The horizontal axis represents the timestamp given to the message by the Supplier. The vertical axis represents the message's latency between the Supplier and Server (the difference in their timestamps). If every message got through, the graph would show successful messages at one second intervals; it does not. Along the horizontal axis are crosses which indicate points at which a one second interval occurred for which the server saw no corresponding message (a data message was lost). The sawtooth pattern indicates that a buffer is being filled either in the Distributed Object subsystem or the NeXT Mach kernel. What is surprising, is that once the buffer fills, no other messages are queued until the buffer drains. It turns out that the type of method invocation the UARC Supplier makes, uses a flag that blocks the invocation when the buffer space is full. It is therefore only an asynchronous method invocation when the message can be stored in the buffer. We surmise that the Distributed Object subsystem is only reactivated by the NeXT Mach kernel after some low watermark is reached, at which point the application regains control. In effect, the missing messages are never sent, since the single threaded Supplier is suspended from execution.

The graph in Figure 7 demonstrates that even though the Supplier periodically pauses during its execution as shown in Figure 6, the Server is continually receiving packets in an uninterrupted manner from the Supplier's transport layer. The points in the graph represent successfully delivered messages. The horizontal axis represents the messages' Supplier timestamp values. Notice the periodic gaps in the horizontal direction. These gaps correspond to those from figure 6. The vertical axis represents the timestamp at the Server when the message is received from the Supplier. Notice that, no significant gaps are present here, and in fact the inter-reception spacing is uniform, implying a uniform network bandwidth used by the Supplier. This shows that Distributed Object's transport mechanism manages the available network bandwidth evenly.

The second experiment focused on the link between the Server and the Client. In this experiment, the Supplier and Server ran locally on `saturn`, while the single Client ran remotely on `UISR` in Alaska. Again, the link to Alaska was overloaded with a data feed of one 40000 byte message per second. Figures 8 and 9 show the results from the first four minutes of the experiment.

Figure 8 represents information similar to that shown in Figure 6, where the points indicate successful messages. However, the the horizontal axis in Figure 8 represents the timestamp value given to the message by the Server, and the vertical axis represents the latency from the Server to the Client. The crosses that lie along the horizontal axis represent unsuccessful method invocations (messages) that are dropped at the

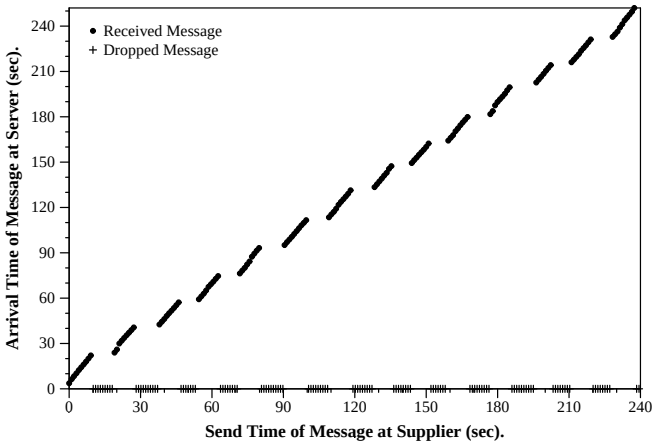

Figure 7: Arrival times versus send times for messages on an overloaded UARC Supplier to Server link.

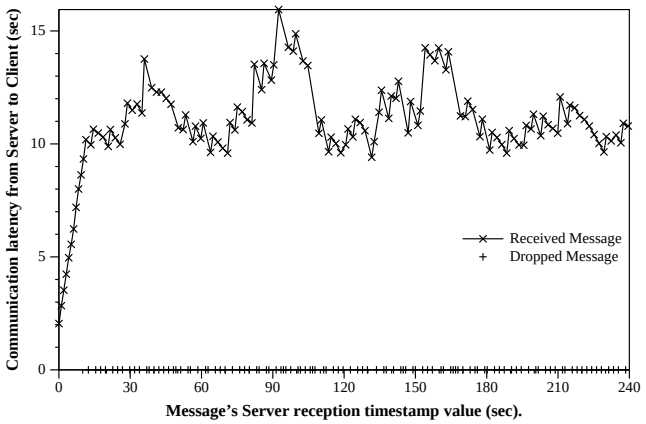

Figure 8: The effects on communication latency of overloading the network bandwidth between a UARC Server and a single Client as seen from the application layer.

Server. This figure displays a more intuitive result than the previous experiment. In this experiment, Distributed Object invocations initially fill a buffer; after which, as messages are delivered and buffer space is reclaimed, subsequent packets are added to the buffer. The difference between these results and those of the previous experiment are due to the multithreaded implementation of the UARC Server in which separate threads handle the incoming data messages. Instead of blocking when the buffer is full, the Server's method invocations are flagged to return immediately with an exception. This exception is ignored, and the message is discarded, which gives the appearance of a truly asynchronous message facility. This produces an interleaving of successful and failed messages, unlike the previous experiment where a clumping of successful messages is followed by a block of presumed failures. Figure 9 supports this result by displaying an absence of significant gaps in either the Server or Client timestamps.

These results show another attempt by the UARC server to control its communication policies; which meets with only limited success. The Distributed Object subsystem extends very little power to the UARC application for the management of its buffers. In this case the only options are whether to block and hold it until there is space, or discard the latest message. No provisions for urgent messages or alternative drop mechanisms are present.

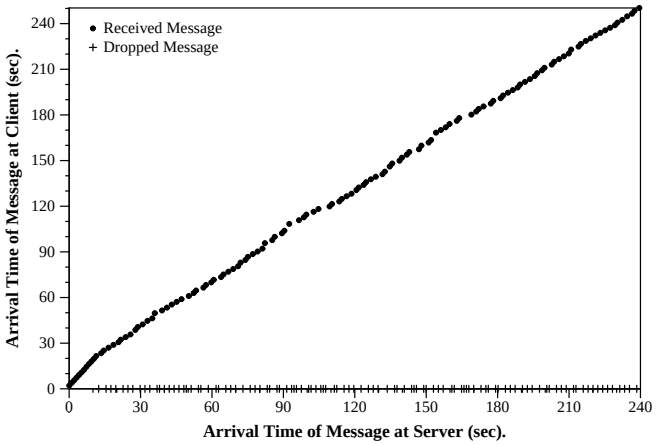

Figure 9: Arrival times versus send times for messages on an overloaded UARC Server to Client link.

5.2 The Adverse Effects of Slow Clients

During the scientific campaigns, it was observed that certain UARC Clients couldn't keep up with the Supplier's data rate; thereby losing a significant amount of atmospheric data. Furthermore it was speculated that these slow clients were somehow affecting the performance of the rest of the system due to interdependencies in the Distributed Object subsystem. This experiment attempts to quantify the effects, if any, that a slow UARC Client has on a fast one. Two Clients are connected to the UARC Server in this experiment: the first by an underutilized link, the other by a link loaded beyond its capacity. Specifically, the Supplier and Server for the experiment are located on `saturn`; the fast Client, `uranus`, is only three routing hops away on our local campus network; and the slow Client is located on `UISR` in Alaska. A Supplier feed rate of one 40000 byte message every second was used. The experiment took place on Monday, June 10, 1996 starting at approximately 11:26am EDT. Figures 10 and 11 show the results from the first two minutes of the experiment. These results are consistently obtained during other experiments using different Clients.

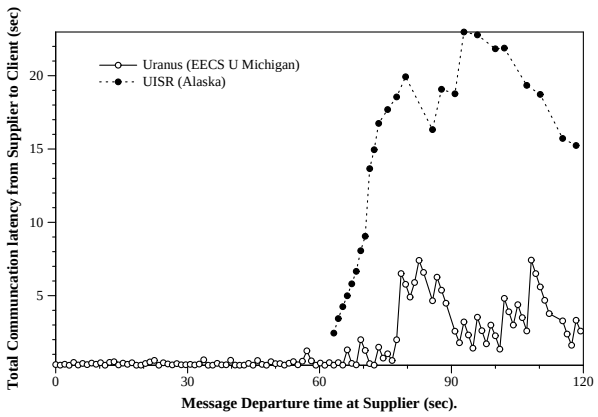

Figure 10: Slow UARC Clients adversely affect Fast UARC Clients

The information represented in Figure 10 is similar to that given in Figure 6, where points in the graph represent successful messages, and the horizontal axis represents the initial timestamp given to the message by the Supplier. It differs, in that the vertical axis represents the overall message latency from the Supplier to the Client. At the beginning of the test, only the fast Client, `uranus`, is connected to the Server. It is clear that the latency for messages during this time is very small, only a fraction of a second. However,

about a minute into the experiment, the second Client at Alaska, `UISR`, is added. It is apparent that the latency to the local Client increases dramatically at this point, being adversely affected by the addition of the second client. The explanation lies with the interaction of the UARC code and the Distributed Object subsystem. When the second Client is added, the Server's single Distributed Object buffer begins to fill with the overloaded link's messages. The Server's code is written so that it makes a separate method invocation to each Client for every message. This is equivalent to sending separate point-to-point messages to each Client. When the Distributed Object buffer is full, there is contention between the fast and slow Clients for space. It is this contention for resources which increase the latency. To further explain this interaction, a machine on the same ethernet as the Server ran `tcpdump` [15] to catch all of the packets exchanged between the UARC Server and the two Clients. Figure 11 shows these results.

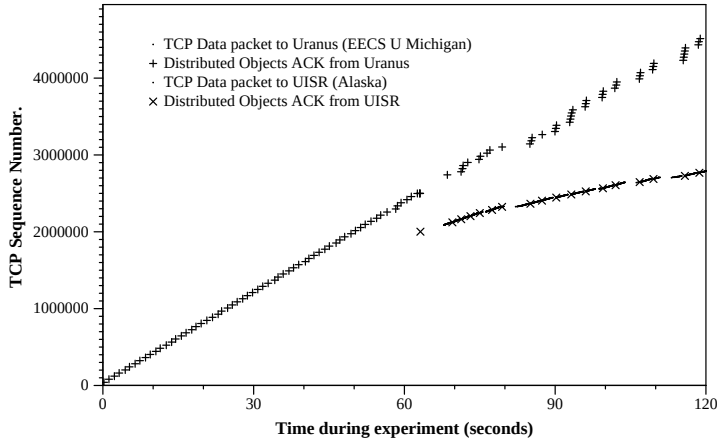

Figure 11: TCP packets between the UARC Server and both the well-connected and slow UARC Clients.

The two lines in Figure 11 represent the TCP traffic between the UARC Server `saturn` and the two hosts, `uranus` and `UISR`. The horizontal axis represents the same time as Figure 10. The vertical axis represents the TCP sequence number for a given point. A dot represents a point where `tcpdump` saw the Server send a 512 byte data packet to a Client. The cross and rotated cross represent points at which the Clients sent the Server twenty bytes of data piggybacked onto a TCP acknowledgment packet. These twenty byte acknowledgements occur every 40000 bytes into the data stream. We surmise that these are object-level acknowledgments sent by the Client's Distributed Object layer to the Server. Note that during the first minute while only the fast Client was connected, there are no gaps in its TCP service. However, once the slow client is added, the fast client's TCP packets are clumped together in bursts, thereby increasing the latency of their distribution. In contract, during this same time period, the Server almost continually transmits data to the slow Client. It is also interesting to note that during this portion of the experiment the Server only dropped about 3% of the messages it tried to send to the fast client, while it dropped 79% of the messages it tried to send to Alaska. It is hard to determine the cause for these phenomena without a more detailed knowledge of the NeXT Operating System and Distributed Object subsystem. Some possibilities include the scheduling of the NeXT Netmsg service; the presence of some feedback mechanism in the Distributed Object subsystem that somehow gives priority to slower clients; or just an artifact of the NeXT Operating System's implementation of TCP/IP. Regardless of the causes, it is clear that the addition of slow UARC Clients adversely affect the entire UARC system, confirming our initial observations.

5.3 UARC Campaign Experiments

The third set of UARC experiments measured its performance under campaign conditions. Of particular interest, was the performance of the Internet route between Greenland and Michigan, which includes the low-bandwidth, high-latency NASA satellite link. During a typical campaign, the following four types of atmospheric data are supplied from a host, `saarullik`, in Greenland:

- A *Fabry-Perot* interferometer, that generates a 250 byte data message every three minutes.
- An *IRIS magnetometer*, that generates 1500 bytes of data every minute.
- The *Sondrestrom Radar*, that produces 5300 bytes of data every five seconds.
- and the *Allsky Camera*, that generates 20000 bytes every minute.

The network conditions were varied by running the experiments twice on Thursday, May 30: once at 4:00pm EDT on a busy network; and again starting at 10:51pm EDT. The host in Greenland supplied data to the server `pollux` at Michigan. The number of Clients connected to the Server was increased from one to six over the length of the experiment to show the effects of loading the Server. These clients were all remote and included: `UISR` (Alaska), `UGEO` (NSF Washington DC), `UIRIS` (University of Maryland), `Allsky` (Lockheed, California), and `Dmistp` (Denmark).

The results shown in Figure 12 detail the amount of data lost between the four Suppliers in Greenland and the UARC Server at Michigan. The horizontal axis represents the number of Clients connected to the Server during the experiment; while the vertical axis represents the percentage of data lost between the Suppliers and the Server. The graphs show that during busy time, roughly a third of the messages are lost, while during the night, almost all of the Supplier's data safely arrive at the Server. Since the only traffic on the Greenland satellite link during both experiments is generated by the UARC Suppliers, this marked change in loss must be due to the congestion on the rest of the Internet. The random losses injected into the Greenland Supplier's TCP streams by background Internet congestion severely degrades the utilization of the satellite link's bandwidth by forcing TCP retransmissions to occur. Not shown in the graph are the number of messages that are subsequently lost between the Server and Clients. During the night these numbers are negligable, while during the day message loss ranges from 30% to 40% in the Six-Client test. It's fortunate that the scientists don't often run experiments during the daylight hours, when the chance of receiving data from Greenland is about one in two.

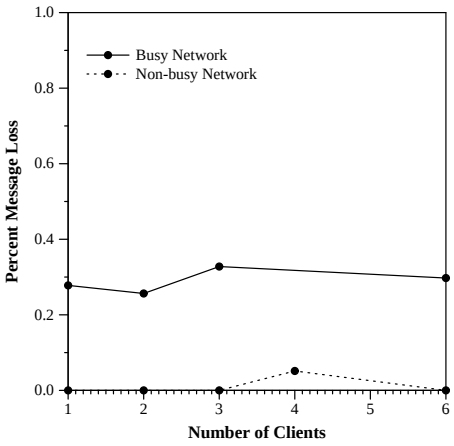

Figure 12: Message Loss from UARC data supplier in Greenland (`saarullik`) to Server at U Michigan (pollux)

Another interesting result shown in Figure 12, is that during both experiments the amount of data lost between the Supplier and Server is fairly constant, regardless of the number of Clients connected. That the losses do not seem to vary with the number of clients, is a little surprising since at its peak of six Clients, the experiment is pushing around 5600 bytes per second (since roughly a third of the supplied 1420 bytes were lost between the Supplier to the Server). At this point during the Six-Client busy-network experiment, even the fastest Client, `UGEO`, lost about 40% of the messages sent to it by the Server. In contrast, in the Two-Client busy-network test (`UGEO` and `UISR`), both hosts experienced no message loss from the Server. These results when added to those of the previous section imply that the outbound buffer is overflowing during the

Six-Client busy-network experiment. The fact that the Server has no additional loss between these two tests indicates that the incoming Distributed Object buffer is somewhat isolated from the overflowing outgoing buffer.

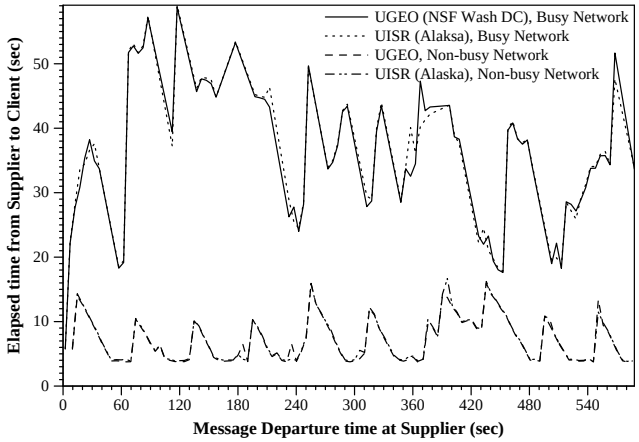

Figure 13: Comparison of message latency between busy and non-busy networks

Figure 13 shows a comparison between day and night latencies. Specifically, this figure displays the latencies for the first ten minutes of both a day and night experiment. In these experiments two Clients are connected to the Server: `UISR` (Alaska), and `UGEO` (NSF Washington DC). Although the latencies differ between experiments, there is almost no message loss between the Server and these Clients. All of the messages from Greenland arrive at both Clients at night. During the day, only two thirds of the messages arrive, since a third of the messages are lost between Greenland and the Server. Thus, a UARC system with even a lightly loaded Server and a congested feed experiences a significant degradation in both application-level message loss and latency during the day.

5.4 UARC Stress Tests

The purpose of this final set of experiments was to determine the scalability of UARC's data dissemination service. Specifically, the tests were designed to determine whether the system's bottleneck is its inherent point-to-point distribution mechanism, or its implementation as a distributed object application. To find the bottleneck, the amount of data fed by the Supplier was varied; whereas the number of clients were fixed at six, two local and four remote. The Supplier and Server were on different machines at Michigan, `pluto` and `pollux`. This was to obviate their contention for outbound buffer space, which would have occurred had they been placed on the same machine. The Client machines were: `saturn` (Michigan), `jupiter` (Michigan), `allsky` (Lockheed, CA), `UIRIS` (U Maryland), `UISR` (Alaska), and `UGEO` (NSF Wash DC). The experiments were run twice, once on a busy network and once late at night. To vary the load on the Server, the data feed's message size was increased from 5000 to 40000 bytes during the experiments. These data messages were sent from the Supplier at one second intervals, which corresponded to a total server throughput that ranged from 30 to 240 Kbytes per second.

Figure 14 represents results from experiments run on Tuesday, June 4 starting at 1:31pm EDT; and Saturday, June 1 starting at 12:47am EDT. They show two of the six clients' loss statistics under varying throughput loads for both busy and non-busy network conditions. The graphs for the two Clients shown, `saturn` and `UISR`, are representative of the results seen by the other UARC Clients. Similarly, the graphs displayed in Figure 15 show the mean latency and standard deviation for these two hosts during the same experiments. These results provide strong evidence that UARC's data distribution service does not scale well. It is intuitive that a point-to-point implementation of multicast would not scale; however these measurements bring to light a more interesting point. Notice that during the 180 Kbps non-busy experiment, the UARC

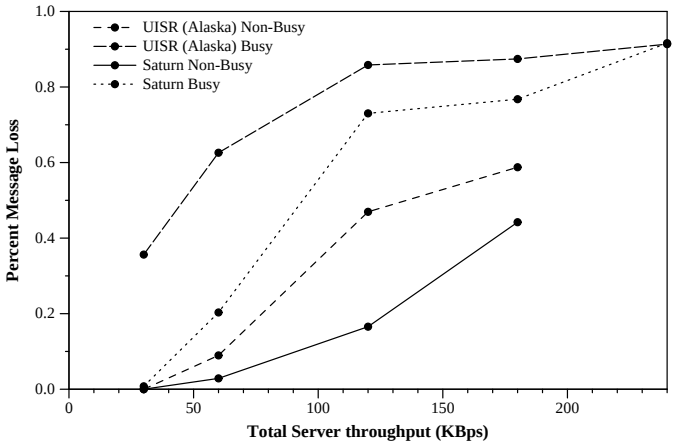

Figure 14: Time of Day effect on Message Loss. Each point represents the fraction of messages lost by a specific client during a UARC stress experiment. During these tests the UARC Server was connected to a total of six UARC clients (two local, four remote). The two clients shown in this graph are representatives of their respective categories (`saturn` is local to the server, and `UISR` is remote).

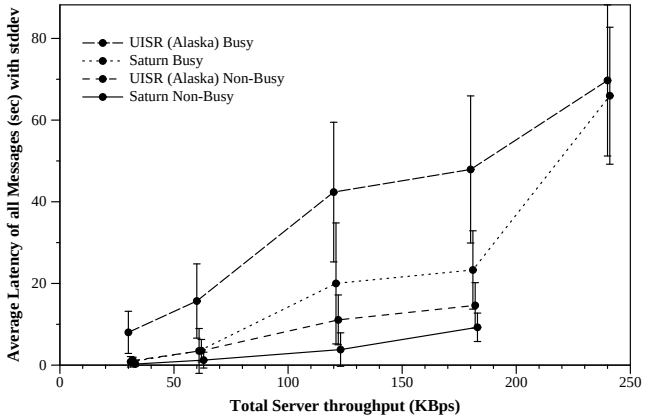

Figure 15: Time of Day effect on Message Latency. Each point represents the mean latency, accompanied by its standard deviation, for a specific client during a UARC stress experiment. During these tests six Clients were connected to the Server.

Server delivers barely half of the data messages to the local Client. We see this performance during the middle of the night on a lightly loaded campus network! The load on the Server's local ethernet is roughly 210 Kbytes per second, well below the point where half of the data should be lost. The same experiment during the day loses almost 75% of the data between `pollux` and `saturn`. This is well below the daytime FTP throughput between these hosts, which is consistently measured at approximately 230 Kbytes per second.

The data displayed in Figure 16 represent the message latency during the first six minutes of the 60 Kbps busy-network experiment exhibited in Figures 14 and 15. It shows the latencies for all six of the connected UARC clients during this interval. Note the spike at time 240. This feature, where all of the connected UARC Clients simultaneously experience a dramatic rise in their message latencies, is observed in most of the UARC experiments. It often is observed when the link bandwidth between the Server and one or more of the Clients is overloaded. These spikes are a graphic representation of the limited scalability of the UARC data distribution mechanism; they only rose in frequency and magnitude as the message size increased during the experiment. Again, the explanation lies in our implementation of UARC using the Distributed

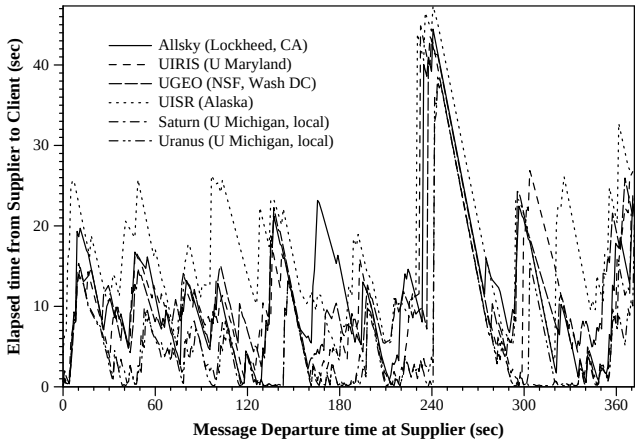

Figure 16: All six UARC Client's end-to-end latencies from Supplier to Client during the 60 Kbps busy-network experiment. The data from `saturn` and `UISR` in this figure are expressed as two points in Figure 15.

Objects subsystem. Both the loss and latency characteristics in these experiments are determined by the interaction among the various method invocation's TCP streams in the Distributed Object subsystem. It is our conjecture that slow Clients periodically monopolize most of the Server's outbound buffer space, thereby forcing the rest of the Clients to wait until some slow Client receives an object-level acknowledgement. At this point the Distributed Object subsystem frees the contentious resource and allows the faster Clients to continue.

From the data in these experiments, one can conclude that the network is not the UARC data distribution mechanism's bottleneck; but instead, that its implementation as a Distributed Object application is the limiting factor in its scalability.

6 Discussion and Conclusions

The UARC object-based collaboratory has been heavily used over the past three years by a distributed collection of space scientists who perform and evaluate atmospheric experiments in real-time over the Internet. This paper described a broad set of experiments which measured the UARC system's communication performance. The results of both Internet microbenchmarks, and application-level experiments were presented; thereby generating a holistic picture of the system's behavior. The results of the Internet `udping` experiments in Section 4 showed that both packet loss and latency are dependent on the time and day of the week; and that sustained round-trip packet losses as high as 5% to 20% are experienced everyday on UARC's portion of the Internet. The `traceroute` experiment determined that this part of the Internet's topology is stable and symmetric. The application-level experiments in Section 5.1 described how UARC's distribution mechanism loses data messages; and how a minor variation in the object invocation semantics produce profoundly different message loss characteristics. The experiments in Section 5.2 demonstrated how a slow Client can adversely affect the performance of a fast well-connected Client due to outbound buffer contention. The results of Section 5.3 showed that congestion between Michigan and the satellite uplink to Greenland, overwhelmed the otherwise unused satellite bandwidth with retransmissions, thereby causing 30% message loss from the Supplier to the Server. It also demonstrated that Distributed Object's inbound and outbound buffers are isolated from each other. Finally the experiments in Section 5.4 exposed the Distributed Object subsystem as the bottleneck in UARC's scalability, and showed how the interactions among the various Client data streams periodically cause dramatic increases in message latency. From these experiments one can conclude that UARC's data distribution mechanism does not scale well and is insufficient for numbers of clients as small as six when distributing a moderate rate of data.

The UARC data distribution mechanism is a relatively small component in the overall UARC software

system; a system that is the composition of many interrelated collaboratory tools [12, 26] developed on the NeXTSTEP platform. NeXTSTEP's Distributed Object package is a mature language-level tool for creating a seamless distributed system from separate software components. The choice for using Distributed Object method invocations as UARC's communication primitive was made as a natural extension of the overall object-oriented system design. The consequences of this choice are made apparent by our experimental results. These results echo one of Lampson's hints: *Don't Hide Power* [13]. By relying on a fairly rigid language-level primitive, the UARC system gave up the ability to control its network management policies. Specifically, it allows slow clients to adversely affect fast ones, to the point where contention for shared resources brings the system to its knees. The Distributed Object subsystem allowed the UARC application very little control over its communication policies, and unwittingly allocated its resources in a manner that caused severe performance penalties. We feel that the an important avenue for further work lies in providing this policy control throughout the various protocol layers.

Clearly, any data distribution service implemented as multiple point-to-point data streams does not scale well. Scalability could be improved by an order of magnitude by utilizing an Internet multicast mechanism [5, 6]. The current integration between reliable multicast mechanisms [7, 11, 14] and language-level primitives, such as object method invocations, is nonexistent; providing an excellent opportunity for further study. In fact several organizations have just begun projects where Internet multicast is used to support communication between groups of objects.

A solution for distributing data to a mixture of both fast and slow hosts will always be needed. As the Internet evolves, slow hosts will not disappear, but will only increase; Fiber-To-The-Curb (FTTC) [30] alone assures a long-term split between well connected business or educational hosts, and an entire class of slower residential hosts. Although people have started looking at this problem (see McCanne [20]), it is far from solved.

The computer industry's collective shift towards object-oriented software development, is in many respects a good thing. It allows the development of extensible, maintainable software systems; however particular attention should be paid to the costs that accrue due to the use of language-level abstractions. Careful attention should be paid during the implementation of these abstractions to allow for the use of the underlying system's full power and flexibility. Otherwise these abstractions give up most of the policy control that is essential to the construction of scalable distributed systems.

Acknowledgements

A profound debt of gratitude is owed to all the cooperative UARC sites around the world. Specifically, the space scientists' patience and understanding is greatly appreciated. Wu-Chi Feng and Robert Hall offered constructive criticism and helpful advice during these experiments. David Thaler graciously proofread earlier drafts of this paper.

References

[1] Berners-Lee, T., Fielding, R., and Frystyk, H. "Hypertext Transfer Protocol (HTTP) 1.0." RFC 1945, May 1996.

[2] Bolot, J. "End-to-End Packet Delay and Loss Behavior in the Internet." In *Proceedings of ACM SigComm '93*, pp. 289-298, September 1993.

[3] Cristian, F. "Probabilistic Clock Synchronization," *Distributed Computing*, (3) pp. 146-158, 1989.

[4] Clauer, C.R., Kelly, J.D., Rosenberg, T.J., Rasmussen, C.E., Stauning, E., Friis-Christensen, E., Niciejewski, R.J., Killeen, T.L., Mende, S.B., Zambre, Y., Weymouth, T.E., Prakash, A., Olson, G.M., McDaniel, S.E., Finholt, T.A., and Atkins, D.E. "A New Project to Support Scientific Collaboration Electronically." In *EOS Transactions on American Geophysical Union*, June 28, 1994 (75).

[5] Deering, S., Cheriton, D. "Multicast Routing in Datagram Internetworks and Extended LANs." In *ACM Transactions on Computer Systems*, pp. 85-110, May 1990.

[6] Eriksson, H. "Mbone: The multicast backbone." *Communications of the ACM*, 37, 8, pp. 54-60, 1994.

[7] Floyd, S., et. al. "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing." In *Proceedings of ACM SigComm '95*, pp. 342-356, October 1995.

[8] David, G.P, Rew, R.K. "The Unidata LDM: Programs and Protocols for Flexible Processing of Data Products." In *Proceedings of the Tenth International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology of the American Meteorological Society*, January 1994.

[9] Hall, R.W., Mathur, A., Jahanian, F., Prakash, A., Rasmussen, C. "Corona: A Communication Service for Scalable, Reliable Group Collaboration Systems." To appear in *Proceedings of the ACM Conf. on Computer Supported Cooperative Work (CSCW '96)*, Boston, MA, Nov. 1996.

[10] Kantor, B., Lapsley, P. "Network News Transfer Protocol (NNTP)." RFC 977, February 1986.

[11] Koifman, A., Zabele, S. "RAMP: A Reliable Adaptive Multicast Protocol." In *Proceedings of IEEE Infocom '96*, (3) pp. 1442-1451, March 1996.

[12] Knister, M., Prakash, A. "Issues in the Design of a Toolkit for Supporting Multiple Group Editors." *Computer Systems - The Journal of the Usenix Association*, 6(2), pp. 135-166, Spring 1993.

[13] Lampson, B.W. "Hints for Computer Systems Design." In *ACM Operating Systems Review*, 21(5) Special Issue, pp. 33-48, 1983.

[14] Lin, J.C., Paul, S. "RMTP: A Reliable Multicast Transport Protocol." In *Proceedings of IEEE Infocom '96*, (3) pp. 1414-1423, March 1996.

[15] Jacobson, V., Leres, C., and McCanne, S. "The Tcpdump Manual Page." Lawrence Berkeley Laboratory, Berkeley, CA, June 1989.

[16] Jacobson, V. "The Traceroute Manual Page." Lawrence Berkeley Laboratory, Berkeley, CA, December 1988.

[17] Julin, D. P and Sansom, R. D. "Issues with the Efficient Implementation of Network IPC and RPC in the Mach Environment.", Unpublished, September 1989.

[18] Manohar, N.R., Prakash, A. "Asynchronous Collaboration via The Record and Replay of Temporal Multimedia Streams." in *ACM SIGOIS Bulletin*, (15)2, pp. 32-34, December 1994.

[19] Mathur, A.G., Prakash, A. "Protocols for Integrated Audio and Shared Windows in Collaborative Systems." In *Proceedings of ACM Multimedia 94*, pp. 318-388, October 1994.

[20] McCanne, S., Jacobson, V., Vetterli, M. "Receiver-driven Layered Multicast." To appear in *Proceedings of 1996 SIGCOMM*, August 1996.

[21] McDaniel, S.E., Olson, G.M., Olson, J.S. "Methods in Search of Methodology – Combining HCI and Object Orientation." in *Proceedings of ACM CHI '94*, pp. 145-151, 1994.

[22] NeXT Computer, Inc. "NeXTSTEP General Reference." (1) Chapters 1-3, 1995.

[23] NeXT Computer, Inc. "NeXTSTEP General Reference.", (2) Chapter 6, 1995.

[24] Olson, G.M., Olson, J.S., Carter, M., Storrxsten, M. "Small Group Design Meetings: An Analysis of Collaboration." *Human Computer Interaction*, 7, pp. 347-374, 1992.

[25] Paxson, V. "End-to-end Routing Behavior in the Internet." To appear in *Proceedings of 1996 SIGCOMM*, August 1996.

[26] Prakash, A., Shim, H. "DistView: Support for Building Efficient Collaborative Applications using Replicated Objects." In *Proceedings of the 5th Conf. on Computer Supported Cooperative Work*, pp. 153-164, October 1994.

[27] Prakash, A., Knister, M. "A Framework for Undoing Actions in Collaborative Systems." in *ACM Transactions on Computer-Human Interaction*. In press.

[28] Sanghi, D., Agrawala, A.K., Gunmusndson, O., Jain, B.N. "Experiment Assessment of End-toend Behaviour on Internet." In *Proceedings of IEEE Infocom '93*, pp. 867-874, March 1993.

[29] Sansom, R. D. "Building a Secure Distributed System." PhD dissertation, Carnegie Mellon University, May 1988.

[30] Tannenbaum, A.S. *Computer Networks, Third Edition*, Prentice-Hall, Chapter 2, pp. 102-138, 1996.

Contents

List of Figures

List of Tables