ReMDoR: Remote Multimedia Document Retrieval over Partial Order Transport

Phillip T. Conrad
Dept. of Computer and info. Sciences
Temple University
Philadelphia PA 19122
215 204 7910
conrad@acm.org

Armando Caro
Paul Amer
Dept. of Computer and Info. Sciences
University of Delaware
Newark DE 19716
302 831 1944
acaro@cis.udel.edu, amer@cis.udel.edu

ABSTRACT

This paper presents results from performance experiments that demonstrate and quantify performance improvements when a PO/R transport service is used instead of an ordered/reliable service (O/R e.g., TCP) or an unordered/unreliable service (e.g., UDP). We first describe the *Remote Multimedia Document Retrieval system (ReMDoR),* an experimental application developed by the authors to evaluate the performance of remote document retrieval over a variety of transport protocols. We then provide a detailed analysis of experiments comparing O/R service to PO/R service for retrieval of a multimedia document. Our results show that between 5% and 10% loss, user-perceivable improvements in progressive display are obtained when PO/R service is used. These results suggest that when packet losses occur in an underlying packet-switched network, transport services providing reliable delivery over independent streams (such the emerging Internet protocol SCTP) are beneficial for retrieval of streaming multimedia.

Categories and Subject Descriptors

C.2.1 **[Computer-Communication Networks]:** Network Architecture and Design - *Network communications;* **C.2.2 [Computer-Communication Networks]:** Network Protocols -*Applications;* H.5.1 **[Information Interfaces And Presentation]:** Hypertext/Hypermedia - *Architectures*

General Terms

Performance, Design, Reliability, Experimentation.

Keywords

Transport Protocols, Multimedia, Partial Order.

MM'01, Sept. 30-Oct. 5, 2001, Ottawa, Canada.

1. INTRODUCTION

1.1 Graceful degradation of multimedia documents over lossy networks.

Many systems now exist that allow authors to construct pre-orchestrated multimedia documents. One of the most popular commercial systems is Macromedia Director. The proceedings of the IEEE and ACM Multimedia conferences contain examples of research systems; examples include [14,15,16]. Multimedia documents consist of objects such as still images, text, audio clips and video clips, which are prearranged according to a temporal scenario. Various schemes exist for expressing temporal scenarios [1]. During the playback of such a document, object presentation proceeds according to this temporal scenario until some event occurs which stops or resets it--for example, a user interaction point is reached, or a user presses a pause button. Typically, a multimedia workstation with sufficient CPU, memory, and I/O capabilities can present a document in compliance with its temporal scenario, provided that the channel delivering the information is ordered and error-free.

However, suppose the document is stored on a remote tile server, and the channel delivering this information is the Internet. In this case, network errors and delays may wreak havoc with attempts to present the document correctly.

We propose that in such situations, it is appropriate to provide for *graceful degradation* of the multimedia document presentation. Graceful degradation is helpful in multimedia documents where not all objects have equal importance, or the same quality-of-service requirements; that is, some objects are essential to document content, while others are nice to have, but optional.

Graceful degradation is also helpful when some objects must be presented in a specific order, while other objects can be presented in an order different from their transmission order, with no loss of quality. For example, in a document describing a simple repair to a piece of equipment, "step 1" should be presented before "step 2". Now, suppose the same document also contains three images that should be presented roughly simultaneously. If two of them show up, and one has to be retransmitted, in many cases it is desirable to go ahead and present the images that arrived while waiting for the retransmission of the missing image.

Given that we want to provide for graceful degradation, what transport protocol should be used for multimedia objects? We

argue that classic transport services such as TCP and UDP are ill suited to this application, and investigate partial order/partial reliability transport service as an alternative.

1.2 Partial Order/Partial Reliability Transport Service

Partially Ordered, Reliable (PO/R) Transport Service bridges the gap between two traditional forms of transport service:

- Ordered/Reliable (O/R) service, such as the Internet's Transmission Control Protocol (TCP), and
- Unordered/Unreliable (U/U) service, such as the Internet's User Datagram Protocol (UDP)

An application using partially-ordered service defines a partial order, ***PO,*** over the objects to be communicated, and provides a representation of ***PO*** to the transport layer. Objects submitted by the sending application may then be delivered to the receiving application in any delivery order that is a ***linear extension, LE,*** of ***PO***. The ordered set ***LE*** is a linear extension of ***PO*** if it is a linear ordering of the elements in ***PO*** such that ($x \prec y$ in ***PO*** $\Rightarrow x \prec y$ in LE). The basic premise of partially-ordered service is that there are applications that have ***some*** message sequencing requirements, but can allow other messages to arrive in any of several orders. For these applications, partially ordered delivery may provide less delay, and use fewer memory resources than ordered delivery.

The premise of this work is that some applications-in particular, multimedia applications-require services that lie in between these two extremes. Unordered/Unreliable service is insufficient for these applications, yet Ordered/Reliable service is too restrictive and may cause the application to pay a performance penalty. The goal is to determine whether better Quality of Service (QoS) tradeoffs and/or performance improvements can be obtained by using a transport service in between these two extremes-one that is better matched to an application's needs.

Previous investigations of PO/R transport service used analytic and simulation modeling to investigate the performance of an abstract PO/R transport service called Partial Order Connection (POC) [2,3,4]. By contrast, this paper presents results of performance experiments designed to measure the extent to which Partially Ordered/Reliable transport service provides performance benefits for a real multimedia application.

Section 2 presents an overview two experimental systems used to evaluate PO/R transport service. Section 3 presents the results of performance experiments comparing PO/R and O/R service for retrieval of a multimedia document. Section 4 describes related work. Section 5 closes the paper with some remarks about the relevance of this work to the emerging Internet transport protocol SCTP (RFC2960) [5].

2. ReMDoR

The ReMDoR system developed by the authors provides users with the capability to create multimedia documents and place them on a server for remote retrieval and display via the Internet. The purposes of the ReMDoR system are:

- to show how a PO/R transport service facilitates coarse-grained synchronization of multimedia objects and graceful degradation during times of network stress,
- to demonstrate the mechanisms needed to implement a PO/R transport protocol in practice, and
- to demonstrate and quantify performance improvements when a PO/R transport service is used instead of an ordered/reliable service (e.g., TCP) or an unordered/unreliable service (e.g., UDP). [6,7,8].

2.1 Architecture

ReMDoR's basic model is similar to that of the World Wide Web; documents are available on a server and are retrieved via a browser. The user interface of the ReMDoR browser, is similar to that of familiar web browsers. However, unlike Web documents, ReMDoR documents ***are temporal-they*** have a time dimension requiring synchronization of elements such as audio, video, still-images, text, pauses, and interactions.

ReMDoR has capabilities that support experimentation with innovative protocols and data compression techniques, such as:

(1) the ability to select from a wide range of transport services and transport service features (via the UTL framework described in Section 2.2),

(2) the ability to record statistics about performance on an object-by-object basis,

(3) features to automate repeated performance experiments, and

(4) the ability to easily incorporate new image formats, such as formats required for network-conscious image compression research [9].

Figure 1 illustrates that the ReMDoR architecture has six components:

(1) The syntax and semantics of a language for specifying multimedia documents. We call the language used to specify ReMDoR documents ***Prototype Multimedia Specification*** Language (***PMSL***). PMSL was introduced in [6]..

(2, 3) A ***document compiler*** that takes a PMSL document as input, parses it, and produces as output a compiled multimedia document in the ***Prototype Multimedia File Format (PMFF).*** A PMFF file is an ASCII file containing information that the server can use to efficiently packetize the multimedia information, and send it via a partially-ordered transport protocol.

(4, 5) A ***server*** that can respond to requests for documents using a ***protocol.*** We call our protocol ***Prototype Multimedia Transfer Protocol (PMTP),*** by analogy with HTTP. Our server is similar in role to that of a Web server.

(6) A ***browser*** providing a GUI allowing the user to request from a server a PMFF file from a server via its URL. The browser parses a URL specification of a PMFF document, formats it as a PMTP request, sends the request to the server, interprets the data returned as multimedia elements: text, audio, vector graphics and bitmap graphics (images), and presents elements to the user.

2.2 The Universal Transport Library (UTL)

Early in the development of ReMDoR, we recognized that designing an application that can run over multiple transport protocols presents certain difficulties. Suppose we want to compare a PO/R transport service such as POC [6,10] to traditional transport services: e.g., ordered/reliable service, **and** unordered/unreliable service. We might imagine that we could compare POC to TCP and UDP. However, it turns out that TCP

and UDP differ in many ways other than order and reliability. Here are just three examples:

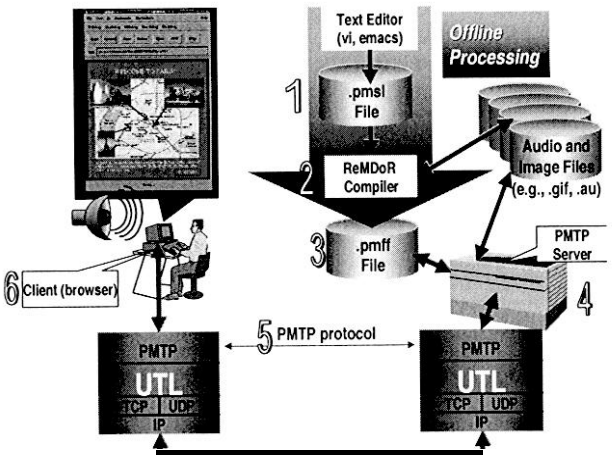

Figure 1: ReMDoR system architecture

- TCP is connection oriented, while UDP is connectionless. Connection-oriented communication requires a different set of system calls to set up communication, and clean up afterwards.
- TCP is byte-stream oriented, while UDP is message-oriented. TO achieve message-oriented communication over TCP, a considerable amount of extra code most be added to the application.
- TCP provides flow control and congestion control; UDP does not.

If we want to write an application that can operate over both TCP and UDP, as well as experimental protocols such as POC, we will likely have to write a great deal of special-case code. Special-case code that depends on specific transport layers is unappealing for several reasons. First it is time-consuming and error prone. Second, writing special-case code makes experimentation with additional transport services difficult, since with this approach, each time we want to add a new transport service to the experiment, the application most be modified. Finally, it opens the experiments up to criticism that the comparison is unfair, since the application code being executed depends significantly on the transport protocol.

To overcome these difficulties, we developed the Universal Transport Library (UTL). UTL is a library of transport layer software that can be linked in with an application, to provide a range of transport services through a single API. The transport services provided in UTL include simple wrappers for TCP and UDP, as well as a range of PO/R transport services. The transport layer functionality in UTL is implemented at user-level rather than in the kernel, and sits in between the application and the regular UDP and TCP services provided by the operating system.

UTL eliminates the need for special-case code in the application to handle particular transport protocols. The protocol is specified as a parameter to the function that listens for connections (in the case of a server) or the function that requests a connection (in the case of a client.)

UTL provides benefits both for developers of new transport layer services, and developers of applications that want to take advantage of various kinds of transport layer service. For developers of transport layer services and protocols, UTL provides a framework for rapid prototyping of transport layer implementations. For application writers, UTL provides a library of various transport services that can be accessed through a single API.

3. EXPERIMENTAL RESULTS

3.1 Experimental Design

In this section, we describe an experiment involving the retrieval of adocument called `parisScene1.pmsl`.

3.1.1 The parisScene1.pmsl document

In the `parisScene1.pmsl` document, the entire French national anthem is played in parallel with two image streams. The first image stream is a chain of maps zooming in to the city of Paris. In parallel with the maps, and with the audio, a series of eight postage-stamp sized scenes from Paris is presented. Images 1 through 4 are presented, by default, sequentially; this ordering is accomplished by specifying each image as a successor of a particular part of the linear audio stream. However, no precedence relationship exists among images l-4, which means that when partial order service is used, if an image is delayed by a retransmission, later images can be presented without delay. Images 5–8 follow images l-4, with 5 following1, 6 following 2, etc.

After the anthem, map sequence, and scenes have all been presented, an audio clip proclaims "Welcome to Paris", in parallel with the presentation of the final map of Paris. A "continue" button is then presented. The ReMDoR browser and experiment scripts allow the experimenter to simulate the pressing of continue buttons by a human user after a fixed, specified delay; for this experiment, the button is always pressed exactly one second after it appears.

3.1.2 Transport protocols compared

This experiment compares the performance of ReMDoR over two protocols from the UTL framework described in Section 2.2; namely, R3 and T3.

R3 provides a partially ordered/reliable (PO/R) service and, will therefore deliver messages out-of-order when permitted to do so by the document specification. By contrast, T3 enforces strict order-of-transmission delivery, similar to TCP.

However, apart from the difference in ordering constraints, R3 and T3 are designed to be as much alike as possible to eliminate factors other than ordering from affecting the outcome of the experiment. Both R3 and T3 use the same code for providing a reliable, flow-controlled, message delivery service on top of UDP. Both protocol implementations use the same routines to implement TCP-friendly congestion-control algorithms, and emulate important TCP features such as RTT estimation, fast-retransmit and fast-recovery (as in TCP Reno). R3 and T3 both use the partial order features of UTL to emulate Object Composition Petri-Net based multimedia synchronization as described in [6] and thus have equal packet header sizes.

3.1.3 Experiment parameters

Table 1 presents our experiment parameters. Given that there are 2 window sizes, 3 loss rates, and 2 transport services. there are

twelve experiments in all. For each of the twelve, we performed 40 runs, for a total of 480 repetitions.

Table 1. Experiment Parameters

	Experiment Number	
Parameter	1	2
Transport Services	PO/R vs. O/R	
Pkt Loss Rates	0%,5%,10%	
Network	Reflector	
Bit rate	512Kbps	
One-way delay	250ms	
Document	parisScene1.pmsl	
Sender Window Size (pkts)	256	
Receiver Window Size (bytes)	4096	8192

3.1.4 Number of runs performed

We discarded runs in which there was packet loss during the initial connection establishment. Our reason for doing is that given the initial RTO values used in most TCP implementations (which values we also use in UTL) cause a delay of several seconds if there is a packet timeout before the true RTT has been measured. Inclusion of these runs can distort the results in ways that are entirely unrelated to the use of ordered vs. unordered or partially ordered transport. We claim that discarding such runs:

(1) introduces no bias, since the initial connection establishment is equivalent regardless of the ordering used, and we applied the same discard criteria to all experiments regardless of the ordering being used, and

(2) provides a more accurate comparison among protocols, since it reduces variance unrelated to the aspect of protocol performance being studied.

Table 2 shows the number of observations on which each experiment is based. In no case do we report any numbers based on fewer than 26 repetitions.

Table 2. Number of runs

Exp.	Parameters	PO/R	O/R
1	LR00	40	40
	LR05	39	37
	LR10	37	32
2	LR00	40	40
	LR05	34	36
	LR10	34	26

3.1.5 Experiment Hypotheses

Our hypotheses for these experiments were as follows:

- **Hypothesis 1: No difference at 0% loss:** There will be no significant gain or penalty for using PO/R service vs. O/R service at 0% loss from the standpoint of (a) progressive display of bytes/pixels, (b) or in any of the audio metrics.
- **Hypothesis 2: Better graceful degradation of progressive display of bytes and pixels:** At loss rates greater than zero, progressive display of bytes and pixels will be better when using PO/R service rather than O/R service.
- **Hypothesis 3: Gain increases with loss rate:** In terms of the graceful degradation of the progressive display of bytes and pixels, there will be increasing gains from using PO/R service vs. O/R service at 10% loss vs 5% loss.
- **Hypothesis 4: Better graceful degradation of audio:** At loss rates greater than zero, all three audio metrics will degrade more slowly when PO/R service is used rather than O/R service.
- **Hypothesis 5: Throughput will improve with larger receive window sizes:** As the window size is increased from 4096 bytes to 8192 bytes, the throughput will improve for both PO/R and O/R service.

3.2 Results for Bytes and Pixels

In this section, we report results for the progressive delivery of bytes to the application, as well as the progressive display of pixels to the end user.

3.2.1 Observations for Bytes and Pixels

Figures 2 through 7 and Figure 21 show performance graphs from Experiments 1 and 2. We make the following observations concerning these graphs:

(1) At 0% loss there is little difference between the bytes and pixels graphs for PO/R service vs. O/R service.

Observation (1) is visible in Figures 2 and 5, where the lines representing PO/R and O/R service are directly on top of one another. This offers support for Hypothesis l(a). Observation (1) can also be seen in Figure 21 in the lines representing 0% loss. Each point in Figure 21 shows the difference between the average performance of PO/R and the average performance of O/R at each point in time. These points are plotted for all three loss rates, for both PO/R and O/R. We observe that the line in each graph that represents the performance at 0% loss remains close to the x-axis throughput the entire presentation of the document, showing that the performance of PO/R and O/R are nearly identical.

(2) The first derivative of the pixels graphs for the 0% loss case (Figure 5)varies over time, while the byte graph for the 0% loss case (Figure 2) is close to linear.

The near linear shape of the bytes graph reflects the fact that the flow control is effectively regulating the throughput; the application is consuming data at a steady rate, thereby opening up space for new packets to be submitted at a steady rate.

The curves in the pixels graph represent the fact that at different points in the linear extension, the fraction of the byte stream devoted to pixels vs. other data, most notably audio, changes over time. When present, audio is given preferential treatment in the linear extension selection algorithm. The linear extension used in this case was tuned so that audio would receive, on average, 50% of the bandwidth during periods where an audio element was available for transmission.

(3) As compared to O/R service, PO/R service offers significant gains in both the progressive display of bytes and pixels at two different window sizes, for both 5% and 10% loss, offering support for Hypothesis 2.

(4) The fact that there is a gain for PO/R service, and that the gain increases for 10% vs. 5% loss provides evidence to support Hypothesis 3

(5) Throughput increases with increased window size, regardless of the loss rate, providing evidence to support Hypothesis 5

(6) The advantage of PO/R over O/R service is reduced as the window size is increased from 4096 to 8192.

Observations (3) through (6) can be seen in Figures 3, 4, 5 and 7 where the performance gain is shown by the gap between the top line representing PO/R service, and the bottom line representing O/R service. The gain, relative to the entire size of the document, may appear small on these graphs, therefore Figure 21 is more useful in putting the absolute gain into perspective. In Figure 21, we see that the lines representing 5% and 10% loss show a gain that for bytes, starts at zero and increases in a near linear fashion, until near the end of the document. For the 8192 byte receive window, the gain tops out at 30–35KB, while for the 4096 byte receive window, the gain is even larger: a gain of 50-70KB. The drop in gain near the end can be explained by the fact that with out-of-sequence delivery, the end of the transmission is marked by a dramatic decrease in throughput, while the transport protocol retransmits the last few remaining packets. The decrease in throughput due to packet losses for the average performance of an ordered protocol is more evenly distributed over the entire transmission.

For pixels, the gain rises and falls with an interesting shape with three smaller peaks followed by a fourth larger peak. This shape is consistent across all four combinations for loss rate (5% or 10%) and window size (4096, 8 192). This shape is an artifact of the proportion of data in the document devoted to pixels vs. other data, and can be easily understood via an analogy. Consider a race between two runners, A and B, where A is faster than B on average, but both runners slow down and speed up from time to time. During periods where A is speeding up and B is slowing down, the distance between them will increase. During the periods where A is slowing down and B is increasing in speed, the distance between them will decrease.

The comparison between the progressive display of pixels for PO/R and O/R service is analogous to the distance between the runners. The "speeding up" and "slowing down" of the runners corresponds to the fact that the proportion of the bandwidth available to pixels is larger at certain parts of the document, and smaller at other parts. The user accessing a document via PO/R service arrives ***earlier*** at each of the points in the document where pixels are displayed rapidly, on average, than the user accessing the same document via O/R service. The gain for PO/R service "shoots up" when the PO/R user arrives at each of these points. The gain for PO/R service then falls when the O/R user "catches up" to the point where pixels are displayed more rapidly.

The exact shape of the curve is tied to the particular document content; other documents would have different curved shapes, as would the same document, if audio were scheduled with a different priority with respect to non-audio data.

Note that for 5% and 10% loss, the average gain over time is strictly positive, and increases steadily almost to the end of the document, and once established, for the bulk of the document, never falls below:

- 30,000 pixels for receive window of 4096, and
- 15,000 pixels for receive window of 8192.

3.2.2 Conclusions related to bytes/pixels

Overall, we conclude that we have found a set of parameters and a document where partial order delivery offers user-perceivable performance benefits in terms of progressive display of pixels and bytes. These results can provide a starting point for future investigations aimed at establishing the limits of the parameter space in which PO/R service can offer such perceptible improvements. Based on the observations above, along with those of all previous experiments, we conclude that this parameter space should be explored further along the dimensions of document size and structure, round-trip delay, bitrate, sender window size, and receiver window size, and loss rate.

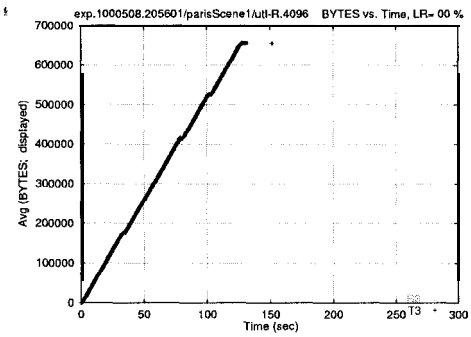

Figure 2: avg bytes received vs. time.
rcv window=4096 bytes, packet loss rate = 0%. Both PO/R and O/R give identical performance.

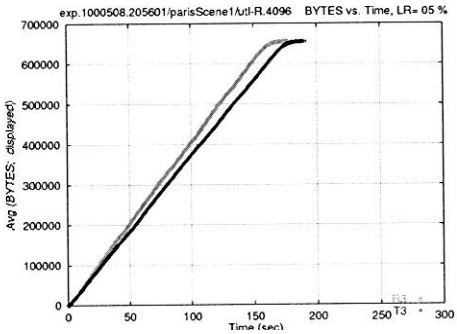

Figure 3: **avg** bytes received vs. time.
rcv window=4096 bytes, packet loss rate = 5%.
PO/R (top line) outperforms O/R (bottom line)

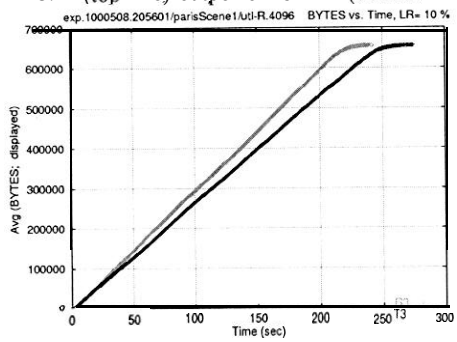

Figure 4: **avg** bytes received vs. time.
rcv window=4096 bytes, packet loss rate = 10%.
PO/R (top line) outperforms O/R (bottom line)

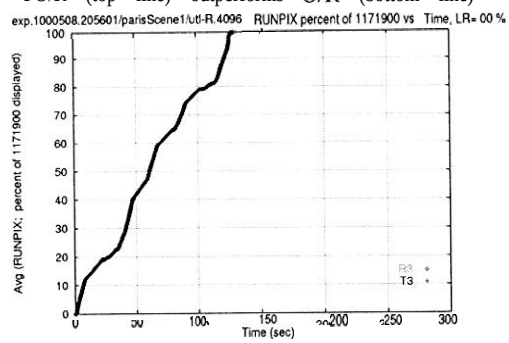

Figure 5: **avg** % of pixels displayed vs. time.
rcv window=4096 bytes, packet loss rate = 0%.
Both PO/R and O/R give identical performance.

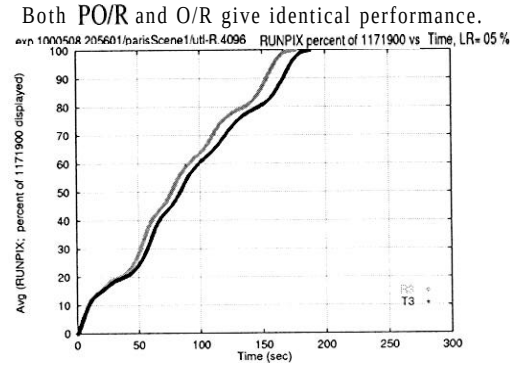

Figure 6: **avg** % of pixels displayed vs. time.
rcv window=4096 bytes, packet loss rate = 5%.
PO/R (top line) outperforms O/R (bottom line)

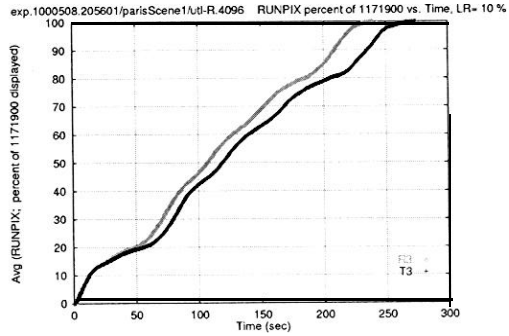

Figure **7: avg** % of pixels displayed **vs.** time.
rcv window=4096 bytes, packet loss rate = 10%.
PO/R (top line) outperforms OIR (bottom line)

3.3 Results for Audio Quality

In this section, we describe statistics that measure the smoothness of the audio presentation. The audio encoding used is the standard SUN. au format (8Khz p-law), which requires 64Kbps of throughput.

3.3.1 Three proposed metrics for audio performance

When audio is streamed over a network with a fully reliable service, the goal is to ensure that the audio device never underflow, because underflow introduce interruptions during playout. The method typically used is described in [11,12]. A small initial playout delay is introduced, during which a queue is allowed to accumulate packets. Once audio playout begins, this queue is drained at a constant rate—for example, at 64Kbps in the case of the 8Khz μ-law encoding used in the ReMDoR system. Since the service rate is constant, we can measure the queue length in seconds rather than in bits. The length of the queue in seconds determines how much time is available for the transport layer to achieve a retransmission of any missing packets.

If the queue length is too short, using a fully reliable service with audio has the potential to introduce defects in the form of *interruptions* when the audio queue underflow. Interruptions can be measured in two ways: the number of interruptions that occur, and the length of each interruption.

Suppose the network delay is constant, or has relatively low variance (say, a standard deviation of 5% of the mean value). In this case, a fixed playout delay of, say, twice the round-trip time should provide enough time for a single retransmission of a missing packet. However, several factors may cause a reliable service to underflow:

- The network delay may vary. causing the playout delay to be too small for even a single retransmission.
- The loss rate may be sufficiently high that multiple retransmissions are required.
- Retransmissions may rob bandwidth from original transmissions, causing the source to be unable to provide packets fast enough.

We would expect that for the case where the audio stream is transmitted in parallel with other streams (e.g., image data) that PO/R service would result in fewer underflow than O/R service. 'Ibis is because missing packets in the non-audio streams will impact the delivery of the audio stream for O/R service, while with PO/R service, only missing audio packets would cause

underflows. Therefore, we would like to measure the impact of underflows on audio quality, to assess the performance improvement offered by PO/R service.

There is a complicating factor, however. We can measure audio underflows in several ways, and it is unclear which way correlates best with perceived end-user audio quality. For example, suppose a user is listening to an excerpt from the French National Anthem lasting approximately 120 seconds. Which scenario would that user prefer?

(a) a single audio interruption of 3 seconds, occurring right in the middle of the piece

(b) 9 interruptions of one-third of a second each, uniformly distributed across the 120 seconds

(c) 3000 interruptions of 1 millisecond each, occurring every 40 ms (that is, between every single audio packet)?

The authors' anecdotal experience is that scenario (c) is perceived only as a slowing in the tempo of the music, and for some listeners may be the least objectionable defect. On the other hand, (a) is probably much less annoying than (b), since with (a) once the defect has passed, it is easily forgotten, while with (b), there is a constant reminder of noticeable problems.

This simple example illustrates that, when considered in isolation, neither the number of interruptions, nor the total duration of the interruptions (which is the same for all three cases above) nor the mean duration of the interruptions is necessarily a good indicator of the impact on quality.

Further, the impact on perceived quality of various kinds of defects will vary among listeners. The impact may also depend on the media content, and the media purpose. A user retrieving a clip solely for entertainment purposes may be intolerant of even slight defects, and may give up on the transmission altogether rather than listen to less than perfect playback. On the other hand, a student replaying a lecture the night before an exam, or a soldier retrieving useful intelligence information in a hostile environment may prefer a clip with fewer defects, but may nevertheless be grateful for any information at all.

Because of the subjective nature of perceived audio quality, a subjective metric called the Mean Opinion Score (MOS) has often been used. The MOS metric has frequently been applied to investigate defects in audio quality introduced by distortion resulting from A-D or D-A conversion, quantization, and lossy compression schemes. However, we are not aware of previous work that assigns Mean Opinion Scores to reliable playback of audio with interruptions. Such a study would be useful as future work.

Therefore, pending the outcome of such a study, [13] introduced three objective metrics for defects introduced by interruptions of reliable audio streams: The purpose of these metrics is to compare the difference between using O/R and PO/R service for documents containing audio.

1) INT (Absolute number of interruptions)

Zero interruptions represents perfect playback. The more interruptions there are, the worse the performance. While this is a useful metric, it does not capture all the information we might find useful. In particular, it would assign a much worse metric to a playout with 10 barely perceptible (or possibly imperceptible) interruptions of 1 millisecond, than it would to a playout with five interruptions of 3 seconds each, which might be more annoying. This fact motivates the next metric.

2) FRACPLAY (Fraction Playing).

Let P be the total playing time of an audio clip (if played without interruption) and let A be the sum o the duration of all audio interruptions. We define FRACPLAY = $P/(A+P)$, that is, the fraction of time during the playout of the audio that the user is actually hearing the audio playing, as opposed to hearing the silence of an interruption.

If there are no interruptions, then FRACPLAY = 1; this represents perfect playout. But if, for example, a 9-second clip is interrupted once for 1 second, than the metric would be 9/10, since the total playout time will now be 10 seconds.

The FRACPLAY metric makes a useful distinction between an 18 second clip interrupted twice for one second each, and an 18 second clip interrupted 10 times for .001 seconds each time, we assume that users will notice the former, and barely notice (or be altogether unaware of) the latter.

However, the FRACPLAY metric also fails to capture exactly what we might want. It does not distinguish between a 18 second clip interrupted once for 2 seconds, or the same 18 second clip interrupted 10 times for 0.2 seconds each time. We assume' that most users would find the second case more annoying.

The need to capture both the number of interruptions and the length of the interruptions motivates the third metric:

3) FRACPLAY^{INT}

To capture both the influence of the number of interruptions as well as the size of the interruptions, we propose the metric FRACPLAY^{INT} (signifying FRACPLAY raised to the INT power). The intuition behind this formula is that each time there is an interruption, there is a cumulative effect on the degree to which the user is annoyed; i.e., we suggest that ***annoyance multiplies.*** As with the INT and FRACPLAY metrics, we can assert that a value 1 represents perfect performance, and that interruptions will cause the value to tend towards zero.

3.3.2 Observations concerning audio metrics

Figures 8 through 20 show performance graphs for audio. Before comparing the performance of PO/R and O/R at 5% and 10% loss, we present several graphs for 0% loss to provide a baseline, and to familiarize the reader with the interpretation of the three proposed audio metrics. Figure 8 shows a histogram of the INT metric for Experiment 1 at 0% loss. As explained earlier, we eliminated runs where there was a packet loss during connection establishment, therefore the number of runs for each parameter value varied slightly (as seen in Table 2). To normalize our reporting of these metrics, rather than using histograms, the remainder of graphs will take the form of Figure 9, which shows the cumulative distribution of the same data as Figure 8.

Figure 9 shows that for both PO/R and O/R

- none of the runs had zero interruptions,
- around 80% of the runs had a value for the INT metric less than or equal to 1, and

[1] Testing the validity of this assumption is outside the scope of this paper; human subject research in this area is suggested as future work.

- 100% of the runs had a value for the INT metric less than or equal to 2.

The interrnptions in this case stem from the fact that at a receiver window size of 4096 bytes, there was insufficient throughput to prevent at least one underflow (and in some cases, two) from occuring. Contrast this with the similar graph for window size 8192 case. shown in Figure IO. Here we see what the cumluative distribution graph looks like in the case of near perfect performance. In this experiment, for both PO/R and O/R, exactly 39 out of 40 runs had zero interrnptions, and exactly 1 run had exactly 1 interruption.

Figures 1 1 through 14 present the graphs for 0% for the other two audio metrics for Experiments 1 and 2. By comparing Figures 9, 10 and 11, with figures 12, 13 and 14 respectively, we can make the following observations concerning these graphs:

(7) At 0% loss, there are some minor audio performance problems at a window sire of 4096 due to underflow, but virtually no problems at a window size of 8192.

(8) At 0% loss, the performance of PO/R vs. O/R service was virtually identical, even down to the distribution of interruptions occurring for the receiver window 4096 case, offering support for Hypothesis l(b).

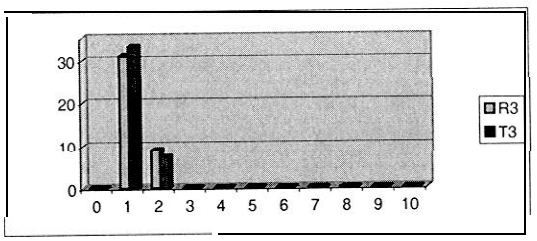

Figure 8: Audio Interruptions, win 4096, LR 0%
Histogram of INT metric; PO/R (left) O/R (right)

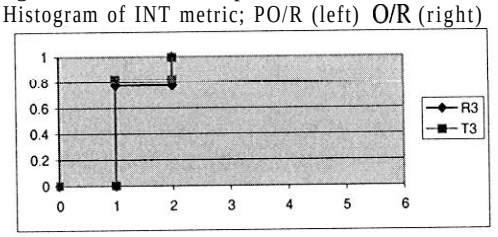

Figure 9: INT, win 4096, LR 0%
Cumulative Distribution of observed INT metric
PO/R (bottom) virtually identical to O/R (top)

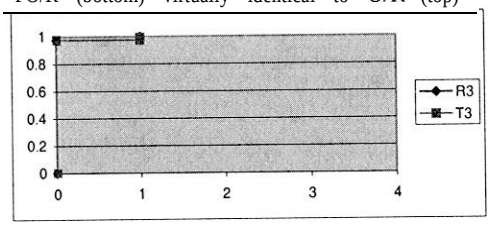

Figure 10: INT metric, win 8192, LR 0%
Cumulative Distribution of observed INT metric
PO/R (bottom) virtually identical to O/R (top)

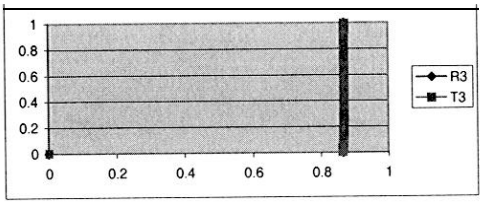

Figure 11: FRACPLAY, win 4096, LRO%
Cumulative Distribution of observed FRACPLAY
PO/R virtually identical to O/R

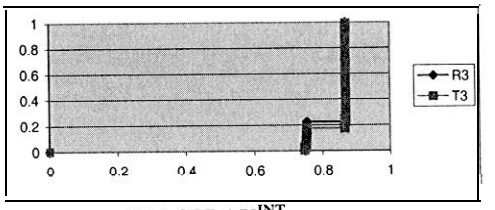

Figure 12: **FRACPLAY^{INT}** win 4096, LRO%
Cumulative Dist. of observed FRACPLAY^{INT}
PO/R (top) virtually identical to O/R (bottom)

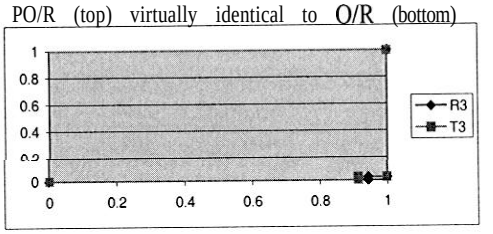

Figure 13: FRACPLAY, win 8192, **LR0%**
Cumulative Distribution of observed FRACPLAY
PO/R virtually identical to OIR

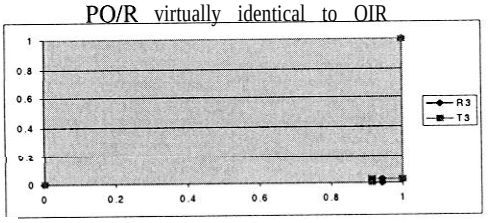

Figure 14: **FRACPLAY^{INT}** win 8192, **LR0%**
Cumulative Dist. of observed FRACPLAY^{INT}
PO/R (top) virtually identical to O/R (bottom)

Figures 15 through 17 present the graphs for 5% and 10% for all three audio metrics for Experiment 1. Figures 15 through 17 show that there is a measurable advantage to PO/R vs. O/R service for audio performance when the receiver window is limited to 409 6bytes. The advantage is somewhat modest at 5% loss: as Figure 15 shows, PO/R service nearly always experiences only 2 interruptions, while this is only true of O/R service about 314 of the time. However, at 10% loss the advantage is clearer. The average number of interruptions is only 2.46 for PO/R service, vs. 3.41 for O/R service. A more telling statistic is that for PO/R service, the number of interruptions is 3 or less 97% of the time. For O/R service, the number of interrnptions is 3 or less only 56% of the time. Tbe other metrics (FRACPLAY and FRACPLAY^{INT}) show similar trends: a slight advantage for PO/R vs. O/R at 5% loss, and a larger advantage at 10% loss.

In general, we can make the following observations concerning these graphs:

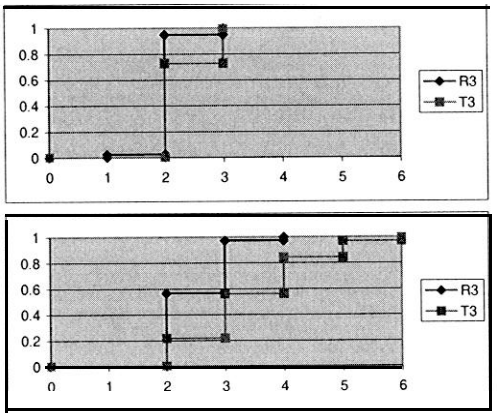

Figure 15: INT metric, win 4096,
cumulative distribution of observed INT metric
LR 5% (top graph),10% (bottom graph)
PO/R (top, left) outperforms O/R (bottom ,right)

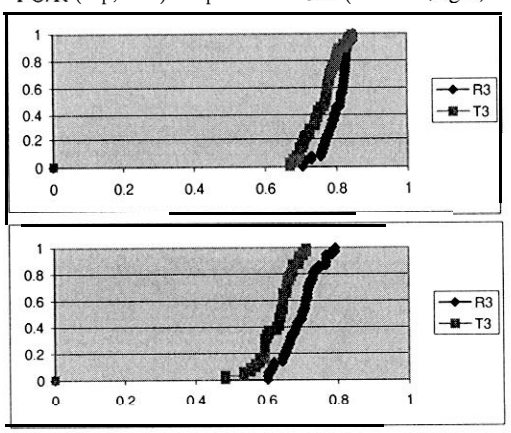

Figure 16: FRACPLAY metric, win 4096,
cumulative distribution of observed FRACPLAY
LR 5% (top graph),10% (bottom graph)
PO/R (bottom, right) outperforms O/R (top, left)

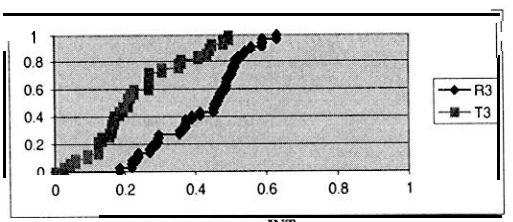

Figure 17: FRACPLAY^{INT} metric, win 4096,
"cum. dist. of observed FRACPLAY^{INT},
10% loss shown (5% omitted for space reasons)
PO/R (bottom, right) outperforms O/R (top, left)

(9) For a receive window size of 4096, all three metrics indicate that a user can expect better audio quality from PO/R service than O/R service at both 5% and 10% loss.

(IO) For a receive window size of 4096, the performance advantage of PO/R over O/R service is higher for 10% loss than for 5% loss.

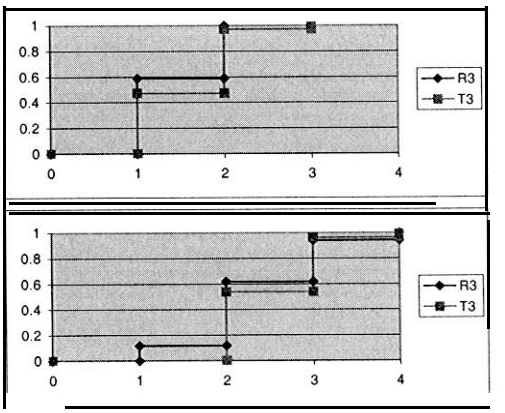

Figure 18: INT metric, win 8192,
cumulative distribution of observed INT metric
LR 5% (top graph),10% (bottom graph)
PO/R (top. left) outperforms O/R (bottom, right)

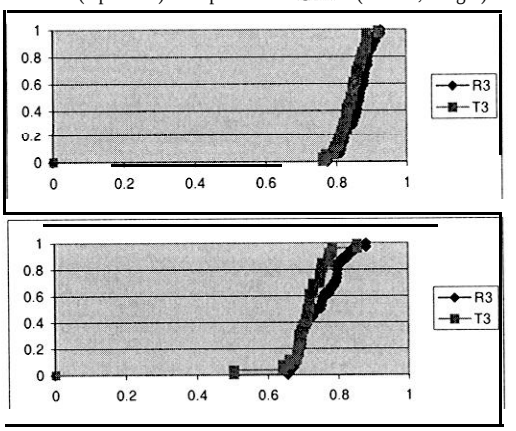

Figure 19: FRACPLAY metric, win 8192,
cumulative distribution of observed FRACPLAY
LR 5% (top graph),10% (bottom graph)
PO/R (bottom, right) outperforms O/R (top, left)

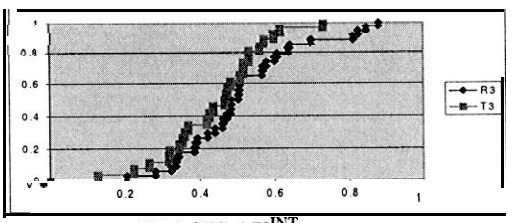

Figure 20: FRACPLAY^{INT} metric, win 8192,
cum. dist. of observed FRACPLAY^{INT},
10% loss shown (5% omitted for space reasons)
PO/R (bottom, right) outperforms O/R (top, left)

Figures 18 through 20 present the graphs for 5% and 10% for all three audio metrics for Experiment 2. We can make the following observations concerning these graphs:

(1 I) For a receive window size of 8192, the difference between PO/R and O/R service ranges from practically nothing, to only a slight advantage for PO/R service.

As with our results for bytes and pixels, the gains for PO/R service were reduced when the window size was increased from

4096 to 8192. As stated before, as the receiver window size increases, this effectively increases the playout delay that is available for retransmission of missing audio packets. When playout delay in increased, out-of-sequence delivery is less helpful in reducing audio interruptions.

3.3.3 Conclusions concerning audio metrics

Overall, with respect to audio, we conclude that PO/R service certainly does no harm with respect to audio, and may offer some help. It would be interesting to investigate what subjective score human subjects give to the audio performance at 4096 for both 5% and 10% loss, for both PO/R and O/R service. This would be useful in determining whether the gains seen in the metrics are perceived to be useful by end users.

3.4 The role of flow control in out-of-sequence delivery

The transport services used in this experiment (R3 and T3) provide a mechanism for end-to-end flow control similar to that provided by TCP. Specifically, the service user (the application) can specify a strict upper bound (in bytes) on the amount of data that may be buffered at the receiver waiting to be delivered to the application. The sending transport entity maintains a conservative estimate of the available buffer space at the receiver, and sends packets only when there is buffer space available in the window. Thus, the sending rate at the transport sender is regulated by the occupancy of the transport receiver's buffer.

The experiments reported in this paper, combined with those in [13] make it clear that flow control is an essential factor in evaluating out-of-sequence delivery. The dilemma regarding flow control and the benefits of out-of-sequence delivery can be summarized as follows:

> *If the receiving transport entity builds up a receive buffer that, in terms of document playout delay, is larger than the time required to do a retransmission, than out-of-sequence delivery cannot possibly be of any benefit to the application.*
>
> *On the other hand, if the receiving transport entity's buffer is too small, or the transmission speed is too slow to keep the buffer occupancy strictly greater than zero, then the application may frequently underflow regardless of whether ordered or partially-ordered service is used.*

We conclude that when assessing of the benefits of partial order delivery, the choice of window size is important, as is the accurate modeling the dynamics of the effective window, as it is shaped by congestion control algorithms, and flow control algorithms (including application backpressure).

4. RELATED WORK

4.1 Related work on partial order transport

[14] presents a more abstract comparison of unordered vs. ordered service in general. A limitation of this work is that it considers the benefits of unordered delivery only in terms of improvements in throughput, buffer utilization, and jitter, with the main emphasis decidedly on throughput. This viewpoint overlooks a key benefit of out-of-sequence delivery, namely the progressive display (or in the general case, the progressive processing) of information.

[14] makes the useful observation that a crucial factor in determining the *throughput* benefit of out-of-sequence delivery is the relationship among the round-trip delay, the bitrate, and the application's ADU processing time. This relationship can be best understood by considering what happens when there is a packet loss with an ordered service. A packet loss results in a *gap* in the sequence number space of bytes or packets. Until this gap is filled, data delivery is suspended, and packets that follow the gap must be buffered. The impact of this gap on throughput depends on the relationship between application processing time, and round-trip delay.

4.2 Related Multimedia Research

There are many examples of systems for creating multimedia documents (authoring systems) and client/server systems for making multimedia documents available over a network (multimedia document retrieval systems.) In this section we provide just two examples of these systems.

4.2.1 MEDIADOC/MEDIABASE

The notion of adding transport QoS to document specification schemes has been previously addressed in the MEDIADOC/MEDIABASE project [15] which has several features in common with our work. In both projects (1) a client/server system is used to serve multimedia documents over a network, (2) QoS (e.g., reliability) can be defined on a per object level, and (3) the transport layer assists with synchronization. Our work differs in both focus and architecture. The MEDIABASE/MEDIADOC project focuses on the design of "an advanced high-performance distributed multimedia information and communications system with a particular focus on document architectures, database models, communication and synchronization, and . . . storage." As such their architecture is quite sophisticated.

By contrast, our emphasis is on exploring the usefulness of PO/R transport protocols; we use multimedia document retrieval mainly as an example application to explore this concept. Hence we limit the scope of our document model and application architecture to the minimal framework necessary to test certain ideas. Our hope is that by developing PO/R mechanisms and demonstrating their utility for multimedia document retrieval, we can provide useful ideas to the developers of systems such as MEDIABASE/MEDIADOC.

4.2.2 Fiets, HyTime, DSSSL and SMIL

[16] presents several issues related to the mapping between the semantic relationships of objects in a multimedia presentation, and their eventual spatial and temporal layout. They argue that rather than nailing down the exact spatial and temporal layout of a document, it is better to represent the semantic relationships between objects, and allow the computer system to determine an appropriate spatial/temporal layout. They present their findings in the context of a system called Fiets (Foundation for Interactive Electronic Touring Systems.) Fiets is an example hypermedia application providing geographic and historic information about tourist attractions in the city of Amsterdam. Of particular interest is the fact that [16]. discusses the use of three standard SGML-based standard multimedia specification languages in the construction of the Fiets system: HyTime (ISO/IEC 10744 (1992)), DSSSL (ISO/IEC 10179 (1994)), and SMIL (W3C 1998). HyTime is a language for specifying Hypermedia

documents, including their spatial and temporal layout. DSSSL is a Scheme-like language for specifying transformations from one SGML-based representation of a document to another. SMIL is a more recent format for specifying coarse-grained synchronization for temporal media in Web documents; its recent incorporation into a commercial system from RealNetworks gives it a chance at more widespread acceptance than previous standards in this area have enjoyed. As such, SMIL, seems like a good candidate for forward progress with ReMDoR [17].

5. REMARKS

Until recently, partial order transport was mainly an idea of academic interest. However, in October 2000, the Internet Engineering Task Force (IETF) issued RFC2960, a standards-track specification of the *Stream Control Transmission Protocol (SCTP)*[5]. SCTP emerged from efforts in the telecommunications community to enable telecom switching protocols to run over IP networks. However, SCTP is also suitable for use as a general-purpose transport protocol.

Given the results in this paper, a particularly interesting aspect of SCTP is its ability to provide a limited form of partial order transport service: namely, reliable delivery of messages over multiple ordered streams. It is reasonable to expect that the performance gains for PO/R service demonstrated in this paper should also be obtainable by comparing SCTP to TCP. Such a comparison would be particularly compelling, since SCTP and TCP have been shown to compete "fairly" in terms of TCP-friendly congestion control [18].

6. ACKNOWLEDGMENTS

The authors thank Ed Golden, Sami Iren, and Mason Taube for their contributions to the software used in the experiments reported in this paper, and the anonymous referees for their helpful suggestions. This work was supported in part by grants from the US Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement number DAAL01-96-2-0002, and by the National Science Foundation (NCR-93 14056).

7. REFERENCES

[1] Perez-Luque and Little, 1996: M. Perez-Luque and T. Little. A Temporal Reference Framework for Multimedia Synchronization. IEEE Journal on Selected Areas in Communication. 14(1), 26-5 1, Jan. 1996

[2] R. Marasli, P. Amer, P. and P. Conrad. An analytic model of partially ordered transport. Computer Networks and ISDN Systems, 29(6). 675-699, May 1997.

[3] Marasli, 1997b: R. Marasli. Partially Ordered and Partially Reliable Transport Protocols: Performance Analysis, 1997, PhD Dissertation, CIS Dept., University of Delaware.

[4] Marasli et al., 1998: R. Marasli, P. Amer , and P. Conrad. Metrics for quantifying partially ordered transport services. Proc. 6th Int'l Conf on Telecommunication Systems, Nashville, Mar. 1998.

[5] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, V. Paxson. Stream control transmission protocol. RFC2960, October 2000.

[6] Conrad et al., 1996: P. Conrad, E. Golden, P. Amer, and R. Marasli. A multimedia document retrieval system using partially-ordered/partially reliable transport service. Proc. Multimedia Computing and Networking (MMCN'96). San Jose, CA, Jan. 1996.

[7] P. Conrad, P. Amer, M. Taube, G. Sezen. S. Iren, and A. Caro. Testing environment for innovative transport protocols. Proc. IEEE MILCOM '98, Boston, Oct. 1998

[8] A. Caro, ReMDoR 2.0:Remote Multimedia Document Retrieval Over Partially-Ordered, Partially-Reliable Transport Protocols. Honors Thesis for the degree Bachelor of Science with Distinction, University of Delaware, May1998

[9] P. Amer, S. Iren, G. Sezen, P. Conrad, M. Taube, and A. Caro. Network-conscious GIF image transmission over the Internet. Computer Networks, 31(7) , 693-708, Apr. 1999

[10] T. Connolly, P. Amer, and P. Conrad. An Extension to TCP: Partial Order Service. Internet RFC1693, Nov. 1994

[1 1] B. Dempsey. Retransmission-Based Error Control For Continuous Media Traffic In Packet-Swithced Networks, Ph. D. Dissertation, University of Virginia, 1994

[12] B. Dempsey, J. Liebeherr, and A.C. Weaver. On Retransmission-Based Error Control for Continuous Media Traffic in Packet-Switching Networks. Computer Networks and ISDN Systems, 28(5), 719-736

[13] P. Conrad. Partial Order and Partial Reliability Transport Service Innovations in a Multimedia Application Context. Ph.D. Dissertation, Department of Computer and Information Sciences, University of Delaware, 1999

[14] C. Diot, F. Gagnon. Impact of out-of-sequence processing on the performance of data transmission. Computer Networks, 31(5), 475-492, Mar. 1999

[15] A Remote Presentation Agent for Multimedia Databases. Proc. IEEE International Conference on Multimedia Computing and Systems (ICMCS'95), 223-230, Washington DC, May 1995

[16] L. Rutledge, L. Hardman, J. van Ossenbruggen and D. C. A. Bulterman. Structural distinctions between hypermedia storage and presentation Proceedings of the sixth ACM international conference on Multimedia September, 1998, Bristol, United Kingdom pp 145-150

[17] J. Urbano, A. Mendes, E. Monteiro, and P. Amer. Specification of order and reliability in SMIL documents. Proc WIAPP '99, IEEE Workshop on Internet Applications, San Jose, Jul. 1999.

[18] Jungmaier, A.; Schopp, M.; Tüxen, M.: Performance evaluation of the stream control transmission protocol. ATM 2000—Proceedings of the IEEE Conference on High Performance Switching and Routing, 2000. pp. 141–148.

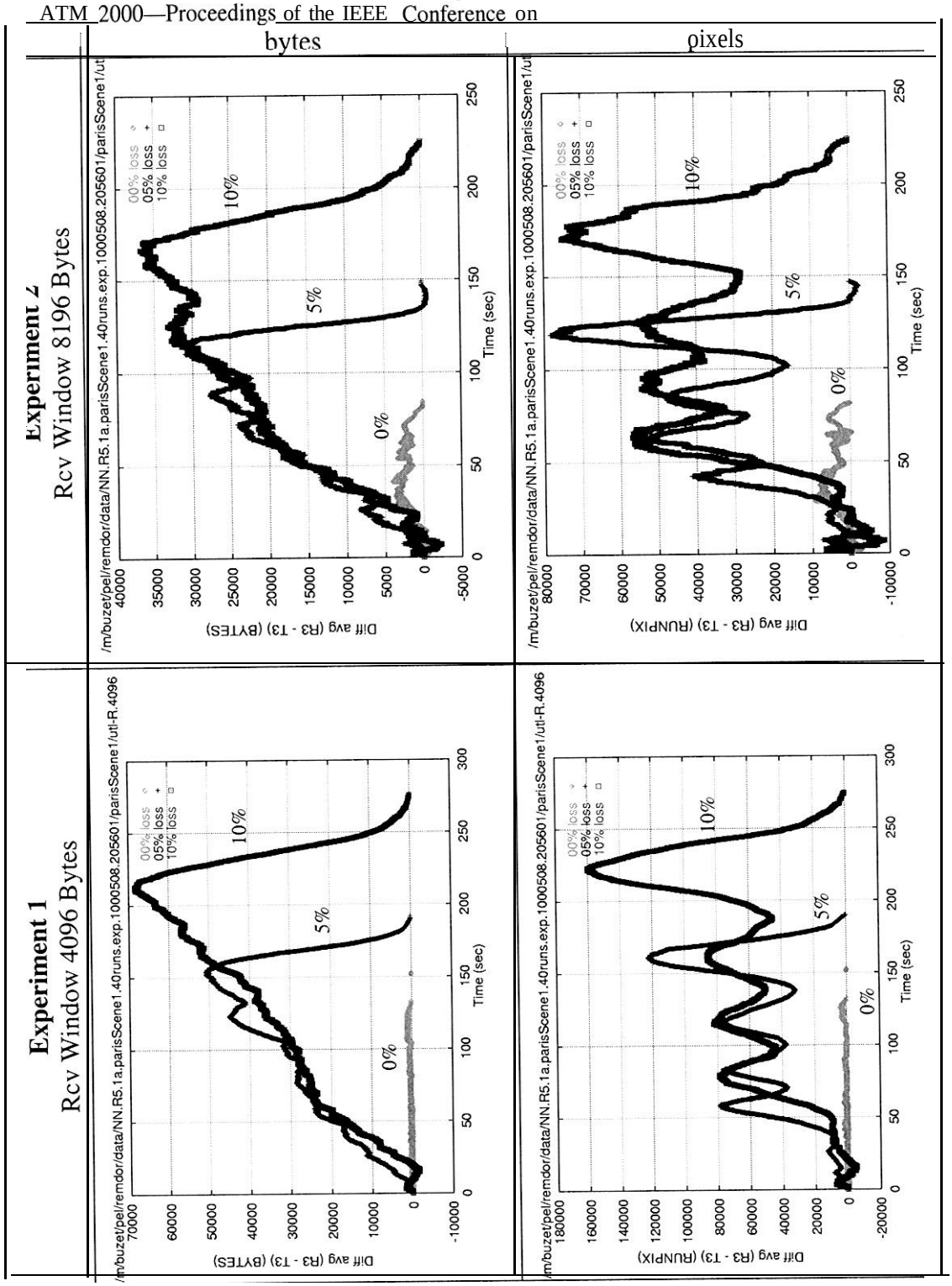

Figure 21: Advantage of **PO/R over O/R** for both **bytes and pixels**