# PARTIALLY-ORDERED, PARTIALLY-RELIABLE TRANSPORT SERVICE FOR MULTIMEDIA APPLICATIONS

**Paul D. Amer**
**Phillip T. Conrad**
**Edward Golden**
**Sami Iren**
**Armando Caro**

Computer and Information Science Department
University of Delaware, Newark, DE 19716 USA
Email: {amer,pconrad,golden,iren,acaro}@cis.udel.edu

## ABSTRACT

*We demonstrate UD's implementation of transport protocols providing partially-ordered, partially-reliable (PO/PR) service. We are developing two example client/server applications: (1) compressed wavelet-encoded image transmission, and (2) remote multimedia document retrieval.*

*By using these examples, we show why PO/PR service is more appropriate than protocols such as TCP or UDP, particularly in lossy packet-switched battlefield network environments. In (1), we show how the system can assist in transmitting wavelet-encoded images of wounded soldiers and/or enemy locations. In (2), we retrieve a multimedia document that explains (for example) how to repair a piece of equipment damaged in the field. We create loss via a dynamically adjustable 'lossy' IP router located at UD one hop away from the server.* [a]

## 1 INTRODUCTION

Work is currently underway at the University of Delaware on various software components that will allow us to demonstrate the benefits of partially-ordered and partially-reliable (PO/PR) transport service. These components include:

- A prototype implementation of $k$-XP, a protocol providing an unordered, partially-reliable transport service.
- A system for retrieving compressed images across a lossy network, allowing experimentation with various combinations of image compression techniques (including wavelet-based techniques) and transport protocols.
- A prototype implementation of *Partial Order Connection, version 2* (POCv2), which provides a *partially-ordered/partially-reliable* (PO/PR) transport service, as well as incorporating features for coarse-grained multimedia synchronization.
- A multimedia document retrieval system, which uses POCv2's coarse-grained synchronization features.

In the sections that follow, we will first provide a brief sketch of what a visitor to our demo might expect to see. This will be followed by a brief overview of the main ideas behind each of the software components, including pointers to more detailed presentations of these ideas.

## 2 BRIEF OVERVIEW

The first demonstration will consist of image retrieval over a lossy network. This demonstration is intended to simulate a system that might be used in military communications to transmit images such as wounded soldier images for telemedicine or images of equipment such as tanks, airplanes, etc. for intelligence gathering. In this demo, there are two stations, a "sender" and a "receiver", separated by an IP-based network. The sender wishes to transmit a single still image to the receiver. However, the intervening packet-switched network may lose or reorder packets in unpredictable ways. Therefore, some retransmission may be necessary if any image quality is to be assured.

[a] Prepared through collaborative participation in the Advanced Telecommunications/Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002. This work also supported, in part, by the National Science Foundation (NCR-9314056) and by the US Army Research Office (ARO) (DAAL03-92-G-0070, DAAH04-94-G-0093).

In order to simulate the loss that may be present in a battlefield network, a route is chosen so that a special "lossy router" lies in between the two stations. This lossy router allows the experimenter to control the level of packet loss between the two stations (see Section 6 for more details.)

Various aspects of the transmission are under the control of the user, including the image compression technique and the transport protocol used. In this way, we can experiment with various combinations of image compression techniques and transport protocols, and demonstrate the advantages of particular combinations. This software will be used to test our hypothesis that a combination of wavelet-based compression with PO/PR transport service offers benefits over other approaches.

The second demonstration involves retrieval of multimedia documents from a server. The basic model is similar to that of the World Wide Web; documents are available on a server, and are retrieved via a browser. Unlike Web documents, however, these documents are "temporal"—that is, they play out over time, and may incorporate audio, video, pauses and interactions.

We will demonstrate the current state of a multimedia document retrieval system based on partially-ordered/partially-reliable service. As with the still-image transmissions system described above, the eventual goal is to be able to demonstrate such a system over a wide-variety of transport services, so as to compare our protocols (e.g. $k$-XP and POCv2) with traditional protocols (e.g. TCP and UDP).

## 3 INTRODUCTION TO PARTIALLY-ORDERED AND PARTIALLY/RELIABLE TRANSPORT SERVICE

In a packet-switched network, the transport layer is the lowest layer responsible for end-to-end quality-of-service (QoS). Typical functions of the transport layer include (1) recovery from data loss, (2) detecting and removing duplicates, (3) resequencing out-of-order data, and (4) flow control/congestion avoidance. The level of service provided by each of these functions is a QoS parameter of the transport layer.

Commonly, transport protocol selection is a choice between extremes. For example, in the Internet protocol suite we note that the service provided by *Transmission Control Protocol* (TCP)[10] is ordered, reliable (no-loss), discards duplicates and is flow-controlled, while the service provided by *User Datagram Protocol* (UDP)[11] is unordered, unreliable (lossy), may deliver duplicates, and is not flow-controlled. The fact that these protocols provide service at the extremes of each of these four QoS axes creates a dilemma for the designer of an application whose needs reside in the middle. When designing an application that will run over Internet protocols, today's implementer typically has three choices: (1) use TCP, (2) use UDP, or (3) build the needed transport functionality as part of the application development effort.

Choosing either TCP or UDP when neither is appropriate has negative consequences. If TCP is chosen for an application that does not require total order and/or full reliability, the result may be unnecessary delays in information delivery. If UDP is used to transmit vital information, important data may be lost or misordered unless the application incorporates the complexity of handling these cases. (Reference [7] provides a mathematical analysis of the performance penalty when a transport service does not support the ideal reliability level for an application.) However, correctly designing and implementing an application specific transport protocol that correctly handles retransmission and flow-control may be a larger effort than designing and implementing the application that sits on top of it!

What is desirable is a standardized transport service (e.g. a library of routines) that applications could utilize to gain flexible control over the ordering and reliability of individual objects. This would allow such applications to achieve an appropriate balance among various QoS parameters without having to "reinvent the transport-layer wheel" with every application. Such an approach is consistent with Application Level Framing as proposed by Clark and Tennenhouse [1].

We are engaged in the task of constructing such libraries of transport layer functionality, based on the concept of *partially-ordered partially-reliable (PO/PR) transport service.* In this demo, we focus on two particular services: $k$-XP, and POCv2. Both $k$-XP and POCv2 provide services that provide a middle ground between UDP and TCP.

$k$-XP and POCv2 have some features in common. Both are connection-oriented; in this respect, they are like TCP and differ from UDP. Both are message-oriented in the sense that they preserve message boundaries; in this way, they differ from TCP and resemble UDP. Both provide a partially-reliable service in the sense that individual messages can be designated with different levels of reliability: each message can be designated as fully reliable, i.e., no-loss (in which case the protocol provides for as many retransmissions as is necessary to transmit the message,) or as unreliable, i.e., "best-effort" (in which case no retransmissions are done.)

Both $k$-XP and POCv2 also allow individual messages to be designated as "partially-reliable", although the meaning of this is slightly different for each of these protocols. In $k$-XP, a "partially-reliable" message is one for which the sender specifies a maximum number of retransmissions. In POCv2, a "partially-reliable" message is one which should be retransmitted as long as it does not delay the delivery of other messages; this notion is defined more formally in [2].

The main difference between $k$-XP and POCv2 is the way order is handled. $k$-XP provides an unordered service; in this respect, it resembles UDP. Therefore, $k$-XP is appropriate for applications where some level of reliability is required, but where messages can be delivered in any order.

By contrast, for applications where some messages must be delivered in a specific order, but some messages can be delivered in different orders, POCv2 provides a "partially-ordered" service. When using POCv2, an application that wants to send a group of messages first specifies a partial order over those messages (in the form of a directed graph). This partial order is then used by the sender and receiver to control the delivery order for those messages. In this way, the receiver does not have to buffer out-of-order messages (as it would for an ordered protocol such as TCP) unless there is a specific order constraint that would be violated. This gives the receiver more flexibility to deliver out-of-order messages sooner in cases where the application can make use of them, and avoid unnecessary buffering delays.

For purposes of testing and demonstrating these protocols, we are developing applications based on these libraries for the purpose of demonstrating and measuring the benefits of PO/PR services and protocols. The next two sections discuss two of these applications.

## 4 WAVELET-ENCODED IMAGE TRANSMISSION

One example application we have implemented for testing different transport protocols is wavelet-encoded image transmission over a low-bandwidth network that loses and reorders packets. Recently, wavelet-based image and video coding has become quite popular; many wavelet-based encoding schemes have been developed which outperform the DCT-based algorithms in terms of both objective criteria (bit rate versus distortion) and subjective criteria [3]. One of the most important features of wavelet-based algorithms is the fact that, at low bit rates, they do not suffer from the *blocking artifacts,* which are the DCT-based algorithms' most notorious defect. This feature makes wavelet-based algorithms a better choice for low bit rate image and video coding applications.

The basic procedure in a wavelet-based image compression scheme is as follows: First, the image is decomposed into wavelet coefficients in different resolution layers to form a multiresolution representation of the image [6]. Then, insignificant wavelet coefficients are discarded (thresholded). The remaining coefficients are quantized and encoded to give the compressed image file. Multiresolution representation has become quite an important issue especially for layered multicast [8] and adaptive multimedia applications [4]. The fact that wavelet-based images provide multiresolution representation, and thus provide layering of information, makes them an excellent choice for today's heterogeneous networks as well as low-bandwidth networks that lose and reorder packets.

Our application consists of two components: An image sender and an image receiver. The image sender is designed flexibly to allow the user to control the image quality and size and transport service being used to obtain the optimum performance from the available resources for that specific application. The image quality and size are controlled by user-adjustable parameters such as thresholding level (i.e., percentage of wavelet coefficients that are set to zero), quantization level (i.e., number of bits used for quantization), and the encoding method. The image quality can be measured both subjectively (by visualizing the image) and objectively (by PSNR). The image receiver simply waits for image data and progressively displays it as it arrives.

Three different encoding methods have been implemented: run-length encoding on *maximum transmission unit* (MTU) sized blocks, LZW (Modified Lempel-Ziv, as in GIF images) on MTU-sized blocks, and LZW on the whole image. The (MTU) is another user-defined parameter provided to test the effects of segmentation/reassembly on image data. In the first two encoding methods, each TPDU contains enough information about how to decode the data and where to put it at the receiving end. This feature, namely *Application Level Framing* [1, 5], makes these encoding methods more suitable especially for lossy networks.

Once the image quality and size have been decided, the user has the option to use one of two different transport protocols: either TX or $k$-XP. TX is a transport protocol based on TCP that provides reliable (no-loss), no-duplicates, ordered delivery; it is essentially nothing more than a wrapper to make TCP a message-oriented service. $k$-XP on the other hand, provides partially-reliable, maybe-duplicates, unordered delivery on top of UDP or IP's datagram service. Clearly, TX is more suitable for an encoding scheme where no loss or reordering is tolerable such as the third encoding method mentioned above which is LZW on the whole image. $k$-XP can be used with encoding schemes that can tolerate some degree of loss, and any degree or reordering. The first two encoding methods mentioned above are well-suited for $k$-XP.

One of our main objectives in developing this application is to test different transport protocols developed by our research group. We plan to run experiments to obtain performance measurements for those protocols in a lossy network environment. Another objective is to study the relationship between image compression algorithms and the transport protocols or the network. We believe that compression algorithms should be designed with different networking conditions in mind. More precisely, compression algorithms should be able to tolerate certain degree of loss, and they should be able to adapt to heterogeneous networks.

We believe that the concept of *application level framing* is well-suited for image compression algorithms. Application layer framing refers to an approach to protocol design where packet boundaries are set by the application in such a way that each packet can be individually decoded and displayed at the receiving end with no dependencies on other packets [1]. In this method, each packet is a so-called "Application Data Unit" (ADU) that can be delivered out-of-order, thus providing faster response in progressive images. In addition, each ADU can have a different QoS. For example, our application allows the following scenario: Since wavelets provide layering of information, each layer is given a different loss probability. The most important layer is transmitted fully reliably and the least important layer is transmitted unreliably. The layers in between are transmitted partially reliably. For purposes of comparison, we have implemented encoding methods that take advantage of application level framing and methods that do not; this will allow us to compare the two approaches.

## 5 MULTIMEDIA DOCUMENT RETRIEVAL

Many systems now exist that allow authors to construct pre-orchestrated multimedia documents. These documents consist of objects such as still images, text, audio clips and video clips, which are pre-arranged according to some temporal scenario. Various schemes exist for expressing these temporal scenarios[9]. During 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 processing capacity (e.g., CPU, memory) 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 file server, and the channel delivering this information is a network connection. In this case, network errors 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. This recognizes that in many multimedia documents, not all objects have equal importance, or the same quality-of-service requirements; some objects are essential to document content, while others are "nice to have," but completely optional.

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 suboptimal for this application. We propose PO/PR service as an alternative. We are developing a prototype system [2] for authoring multimedia documents, and retrieving and displaying them across the Internet. Our goals in building this system are: (1) to show how a PO/PR transport service facilitates coarse-grained synchronization of multimedia objects and graceful degradation during times of network stress, (2) to demonstrate the mechanisms needed to implement a PO/PR transport protocol in practice, and (3) to demonstrate and quantify performance improvements when a PO/PR transport service is used instead of an ordered/reliable service (e.g., TCP) or an unordered/unreliable service (e.g., UDP).

## 6 LOSSY ROUTER

The successful development of communications protocols requires a testing environment to be used for proving designs and gathering performance benchmarks. The testing environment needs to be as close an approximation as possible to the real environment in which the protocol is to operate. In the particular case of communications protocols, this means simulation of various propagation delay, throughput, and error conditions.

The Lossy Router (LR) is a program which was created to meet these needs. A reasonably simple piece of software, the LR is intended to run on a machine serving as an IP gateway. All of the normal routing functions of that host are replaced by this software. This way, any communications between two hosts which route their messages through the LR gateway may be affected. In the current implementation, the LR allows a developer to specify a basic rate of packet loss. The forwarding of each IP datagram is viewed as a simple Bernoulli trial, where the specified percentage of packets are deliberately dropped by this software.

Other factors can also be added along the route between two communicating hosts, such as a PPP-link which affects throughput and propagation delay. By introducing this LR gateway into a properly configured testing environment, particular conditions can be simulated.

Future implementations of the LR will permit developers to specify different models of loss. This would allow the choice of a Markovian loss model, or a Normal distribution instead of a linear distribution. Also, the LR could be extended to introduce delay and limit aggregate throughput.

## 7 DISCLAIMER

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied of the Army Research Laboratory or the U.S. Government.

## 8 REFERENCES


[1] David D. Clark and David L. Tennenhouse. Architectural considerations for a new generation of protocols. In *SIGCOMM '90*, pages 200–208, Philadelphia, Pennsylvania, September 1990. ACM. Computer Communications Review, Vol. 20(4), Sept. 1990.

[2] Phillip T. Conrad, Edward Golden, Paul. D. Amer, and Rahmi Marasli. A multimedia document retrieval system using partially-ordered/partially-reliable transport service. In *Multimedia Computing and Networking 1996 (MMCN96; sponsored by SPIE/IS&T)*, San Jose, CA, USA, Jan 1996. URL:`<http://www.eecis.udel.edu/~amer/PEL/poc/postscript/mmcn96full.ps>`.

[3] Charles D. Creusere. Image coding using parallel implementations of the embedded zerotree wavelet algorithm. In *IS&T/SPIE Symposium on Electronic Imaging*, volume 2668, San Jose, CA, 1996.

[4] C. Diot. Adaptive applications and QoS guaranties. In *IEEE Multimedia Networking*, Aizu, Japan, September 1995. IEEE.

[5] C. Diot, I. Chrisment, and A. Richards. Automated implementation of distributed applications. In *IFIP High Performance Networking*, Palma, Spain, September 1995.

[6] S.G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 11:674–693, July 1990.

[7] Rahmi Marasli, Paul Amer, and Phillip Conrad. Retransmission-based partially reliable services: An analytic model. In *IEEE INFOCOM*, San Fransisco, California, March 1996. URL: `http://www.eecis.udel.edu/~amer/PEL/poc/postscript/infocom.ps`.

[8] S. McCane, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *ACM SIGCOMM*, pages 117–130, Stanford, CA, September 1995. ACM.

[9] M. Pérez-Luque and T. Little. A temporal reference framework for multimedia syncrhonization. In *IEEE Workshop on Multimedia Synchronization*, Tysons Corner, VA, May 1995. URL:`<http://hulk.bu.edu/sync95/papers/mjpl_posit.ps>`.

[10] J.B. Postel. Transmission Control Protocol. Internet Request for Comments RFC793, Sept 1981.

[11] J.B. Postel. User Datagram Protocol. Internet Request for Comments RFC768, Aug 1981.