A Multimedia Document Retrieval System Using Partially-Ordered/Partially-Reliable Transport Service *

Phillip T. Conrad Edward Golden Paul D. Amer Rahmi Marasli

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

1 Introduction

In this paper we investigate the benefits of using a partially-ordered/partially-reliable(PO/PR) transport service for multimedia document retrieval by implementing a prototype system. For specification of prototype documents we introduce PMSL, a document specification language combining aspects of Little and Ghafoor's OCPN [7] with features for graceful degradation. PMSL is integrated with POCv2, a PO/PR transport protocol. POCv2 is designed to provide the following benefits over traditional transport protocols such as UDP and TCP:

- POCv2 provides mechanisms to achieve the reliability and synchronization requirements of individual multimedia objects as specified by the document author.
- POCv2 makes the application programmer's job easier by providing mechanisms for synchronization, reordering, and retransmission.
- POCv2 delivers a multimedia document with the reliability and synchronization specified by the author. When network conditions are poor, POCv2 provides for graceful degradation [4].

Our goals in building such a system are:

- to demonstrate and quantify improvements in performance when a partially-ordered/partially-reliable transport service is used instead of an ordered/reliable service or an unordered/unreliable service,
- to show how a PO/PR transport service facilitates coarse-grained synchronization of multimedia objects, and
- to demonstrate the mechanisms needed to implement a PO/PR transport protocol in practice.

The remainder of this extended abstract is structured as follows: Section 2 provides background concerning document specification and PO/PR service. Section 3.1 describes the architecture of a multimedia document retrieval system for PMSL documents built on POCv2. Section 3.2 describes the PMSL document model, comparing and contrasting it with OCPN. Section 4 describes the POCv2 service and the proposed architecture and implementation. Section 5 describes related work.

2 Background

2.1 Document Specification

Since multimedia documents have a temporal dimension, objects are played out according to some *temporal scenario*. Temporal scenarios may express exact timelines for object presentation, and/or temporal relationships (synchronization) between objects. Various *temporal specification schemes* have been proposed; [11] provides an overview of these. One such scheme is OCPN, which is based on timed Petri-nets [7]. A distinction can be made between *coarse-grained* synchronization (sometimes called *object* synchronization), and *fine-grained* or *stream* synchronization (e.g. lip sync)[13]. A multimedia workstation with sufficient capacity can present a document in compliance with its temporal specification provided that document objects are delivered to the display application ordered and error-free. However, if the document is stored on a remote

*This work 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).

document server, network errors and variable delays (as in a packet-switched network) may wreak havoc with attempts to present the document correctly.

In such situations, it is appropriate to provide for "graceful degradation" of the presentation of the multimedia document, because not all objects have equal importance or the same quality-of-service (QoS) requirements. Some objects are essential to document content, while others are "nice to have" but completely optional. We are developing a Prototype Multimedia Specification Language (PMSL) which combines some features of OCPN with features for graceful degradation. This language is still under development; the full paper will contain a precise specification. For purposes of this abstract we present only a brief sketch of the main ideas (see section 3.2.)

2.2 Partially-Ordered/Partially-Reliable Transport Service

Three fundamental QoS parameters for a transport-layer service are *loss*, *order*, and *duplication*. Some authors lump these together under "reliability," using "reliable" to refer to a transport service where no messages are lost, where all messages are delivered in the same order as submitted, and where no duplicates of messages are delivered.

To classify transport QoS with greater precision, we restrict "reliability" to refer only to whether messages may be lost, defining a *reliable* service as one that allows no loss whatsoever. In this way, reliability becomes orthogonal to the service features of order and duplication. "Order" refers to whether or not the sequence in which objects are sent is preserved in delivery. Thus, an *ordered* service delivers objects in the same total order as they were presented to that service. "Duplication" refers to whether or not multiple copies of the same object may be delivered. A *no-duplicates* service will detect and discard any duplicates.

A certain amount of loss and reordering is tolerable when retrieving and displaying multimedia documents over a network connection. Bandwidth reservation is one strategy that can be used to ensure that the network's QoS meets application needs. However, there may be situations where reservation mechanisms are not feasible or fail to provide the necessary QoS (e.g. a disaster situation or battlefield scenario involving intermittent jamming). In these cases the loss rate of the underlying network may still be higher than the application's tolerance for loss. In such a scenario, another strategy that can be employed to improve reliability is *retransmission* of lost packets. Although widely rejected as impractical, the retransmission of continuous media (e.g. audio) is shown to be feasible by Dempsey [5]. Retransmission-based protocols can offer greater reliability than the underlying network, in exchange for less desirable values for other QoS parameters (e.g. delay).

However, common applications must choose between classic transport services that provide either a ordered/reliable service (such as TCP) or an unordered/unreliable service (such as UDP). Neither of these transport services is, by itself, a good fit with the requirements of multimedia. Application designers must either (1) use UDP and accept the underlying loss and disorder of the network, (2) use TCP and accept any consequent delay increase and throughput reduction, or (3) use UDP, but incorporate reordering and loss recovery mechanisms into the application.

What is desirable is a standard mechanism (e.g. a library of routines) that applications could utilize to have flexible control over the ordering and reliability of individual objects. This would allow such applications to achieve a good tradeoff between {order, reliability} and {throughput, delay}. Such a mechanism is consistent with the Application Layer Framing (ALF) concept proposed by Clark and Tennenhouse [2]. In this extended abstract, we describe work underway to implement such a mechanism; we call this mechanism POCv2. POCv2 is a transport layer protocol that provides a *partially-ordered/partially-reliable* (PO/PR) service.

Partially-Reliable services are proposed in [5, 6]. In [1, 3] this is taken one step further by incorporating both partial order and partial reliability into a single transport protocol called POC (Partial Order Connection). An application using POC defines a partial order (PO) over the objects to be communicated and provides a representation of this PO to the transport layer. Objects are then submitted and delivered in a manner consistent with the PO. Simulation and analysis results based on models of POC have shown that applications using POC can make tradeoffs between {order, reliability} and {throughput, delay} [8, 9, 10].

In this paper, we describe the service, architecture and proposed implementation of version 2 of POC (POCv2.) While version 1 of POC was conceived as an abstract protocol to illustrate the concept of PO/PR service, POCv2 (described below) is being designed to make the benefits of partial order and partial reliability available to software developers, especially multimedia application developers.

3 Proposed Document Retrieval/Display System

3.1 Architecture

The architecture of the document retrieval and display system is shown in Figure 1.A. Documents are specified in PMSL, our prototype multimedia specification language. A PMSL specification is an human-readable ASCII representation of a directed graph which represents the multimedia document structure. The nodes in the directed graph represent multimedia presentation elements; examples of such elements include audio clips, video clips, fixed objects (images, text, geometric shapes), pauses, user interactions (e.g. entry fields, buttons), and erase events (which remove fixed objects from the screen). A PMSL document can be created via a text editor or (eventually) by a GUI-based authoring system.

Client processes open a POCv2 connection to a document server to request a list of available documents. The client then selects a document via a mouse click. The name of the document is transmitted to the server, which then begins transmitting the elements of the document. At the client side, the client can simply read and display the document elements as they are delivered by POCv2. The client can be required to explicitly signal the POCv2 connection immediately after the display of each presentation element "ends"—for example, at the moment an audio stream ends, the playout of a pause event expires, a button on the screen is pressed, or a fixed image has been successfully placed on (or erased from) the screen. Supplied with this knowledge and information from the PMSL specification, the POCv2 layer can provide a coarse-grained synchronization service that allows for graceful degradation. This topic will be fully developed in the full paper; the remainder of this extended abstract only sketches the main ideas.

3.2 The PMSL Document Model

The semantics of the PMSL model are similar to those of Object Composition Petri Net (OCPN) [7], but differ in several ways. In both OCPN and PMSL, objects can be composed in sequence and parallel to form a directed acyclic graph (DAG). The basic idea in both models is that no object should be presented until all of its predecessors have been presented. In PMSL, a partial order is used to represent this fact, while OCPN uses a petri-net representation. Both OCPN and PMSL objects have a duration, but in PMSL, this duration may be indeterminate. In addition, PMSL allows the document author to specify *handling instructions* for each object. A handling instruction is a specification of how the system transports, delivers, and presents an object. It has meaning to both the application layer client and server, as well as the underlying transport layer (POCv2). Formally, a handling instruction is a tuple $h = \langle rc, ot, dd, er \rangle \in \{Rc \times Ot \times \{true, false\} \times \{true, false\}\}$, where:

$rc \in Rc$ is a reliability class: $Rc = \{\texttt{RELIABLE}, \texttt{PARTIALLY\_RELIABLE}, \texttt{UNRELIABLE}\}$.

$ot \in Ot$ is an object type: $Ot = \{\texttt{BLOCK}, \texttt{STREAM}\}$.

dd is boolean value representing the "damaged delivery" flag. (This flag applies only to objects with ot = `BLOCK`, and rc =(`PARTIALLY_RELIABLE` or `UNRELIABLE`) as explained below.)

er is boolean value representing the "explicit release" (as explained in Section 4.1.)

At the document display level, the reliability class determines whether an object can be dropped from the presentation (e.g. due to a network error). `RELIABLE` objects are never dropped, while `UNRELIABLE` objects can be dropped at any time. At the transport level, POCv2 provides for retransmission of `RELIABLE` objects until they arrive, and never retransmits `UNRELIABLE` objects. At the document display layer, `PARTIALLY_RELIABLE` objects may be dropped if and only if certain conditions are met—conditions which will be explored in more detail in the full paper. POCv2 retransmits `PARTIALLY_RELIABLE` objects only up to the point where they are determined to no longer be useful to the application. Mechanisms are provided to allow the application and POCv2 transport layers to communicate information about the current usefulness of objects. In this way, the application has control over transport *policy* without having to be concerned with transport *mechanisms.*

`BLOCK` objects are treated as atomic; they cross the boundary between the application and transport layers as a single unit, and are either presented in their entirety, or not at all. The only exception to this is a `BLOCK` object with reliability type `UNRELIABLE` or `PARTIALLY_RELIABLE`, and dd=true. In this case, the `BLOCK` object may be delivered with some data replaced by zeros if that data is unavailable due to packet loss. (This may be used for delivery of a bit-mapped image when it is better to display it with damage instead of throwing it away, or retransmitting it.) `BLOCK` objects have a fixed size (in bytes) and a real duration. By contrast, a `STREAM` object is a ordered byte stream of indeterminate size and duration. `STREAM` objects are useful when length and/or duration is not known at the time of document authoring—for example, an object that is retrieved indirectly from some location[1] whose contents may change. `STREAM` objects may be submitted to the transport layer in arbitrary length pieces. At the client side, while the bytes of a particular `STREAM` object are ordered, they may be delivered in arbitrary length pieces and interleaved with the delivery of other objects. `STREAM` objects can be used to represent continuous media such as audio and video, where delivery should be interleaved, but it is inconvenient to represent each frame as an individual object.

Given an OCPN, we can construct an equivalent PMSL representation by constructing the partial order that corresponds to the petri-net, and setting $h = \langle \texttt{RELIABLE}, \texttt{BLOCK}, \mathit{false}, * \rangle$ (this will be shown formally in the full paper.) Therefore, by the results in [7], PMSL can represent all thirteen temporal intervals between any two elements, provided that their durations are known in advance. As an abstract model, PMSL augments the expressive power of OCPN by adding the capability to express indeterminacy and network QoS for the objects represented. As a language, PMSL is intended only as a prototype; our direction for PMSL is to

[1]i.e., a file, a URL, or a database record

investigate ways to express the concepts of the PMSL model within the framework of existing and emerging standards such as HTML and HyTime.

4 Overview of POCv2

4.1 Service provided by POCv2

For the most part, the service provided by POC is essentially defined by the partial order, and by the handling instructions outlined in Section 3.2. Each objects is submitted to the transport layer with a handling instruction that describes how it is to be delivered. At the interface between the transport receiver and the client application, the client calls a `read_poc_object()` function to retrieve data from the transport layer. The first objects that are delivered to the application are those than have no predecessors in the partial order. As such objects become available, the POC receiver will return them via the `read_poc_object()` function, along with an *object identifier* uniquely identifying the object delivered. The available objects are arranged in a directed graph that corresponds to the partial order; to order objects for delivery, the POCv2 receiver makes a breadth-first traversal of this graph, ensuring that `STREAM` objects are interleaved with `BLOCK` objects in an appropriate fashion. When an object (or piece of a `STREAM` object) is not available at the time it is "next in line" for delivery, POCv2 uses the handling instructions to determine whether to wait for the object to arrive or drop the object (or `STREAM` piece).

In addition, there is an "explicit release" mechanism that is useful for providing coarse-grained (object) synchronization. The explicit release mechanism is used to ensure that successor objects are not delivered by the transport layer until the *playout* of predecessor objects is complete. For example, suppose a fixed image is to appear immediately following the completion of an audio clip. When explicit release is used, the application explicitly signals the transport layer when the audio playout is complete, thereby releasing the successors of the audio clip for delivery.

The explicit release feature is controlled by the *er* flag in the handling instuction. If the *er* flag is true, the application must notify the transport layer when the object is finished with its playout, via a function `release_successors(object)`. As the name of this function implies, the purpose of this call is to tell the transport layer to release the precedence constraints on the immediate successors of `object`. This is analogous to the "unlocking" of a token in OCPN. Objects in the partial order with non-zero in-degree become deliverable when all of their immediate predecessors in the partial-order have been either delivered and "unlocked" in this way, or *declared lost* and "unlocked" in the case of `UNRELIABLE` or `PARTIALLY_RELIABLE` objects. The rules governing declaring `UNRELIABLE` and `PARTIALLY_RELIABLE` objects lost will be explored further in the full paper. If the *er* flag for some object is false, that object is implicitly "unlocked" when it is delivered to the application or declared lost.

4.2 Proposed POCv2 Architecture/Implementation

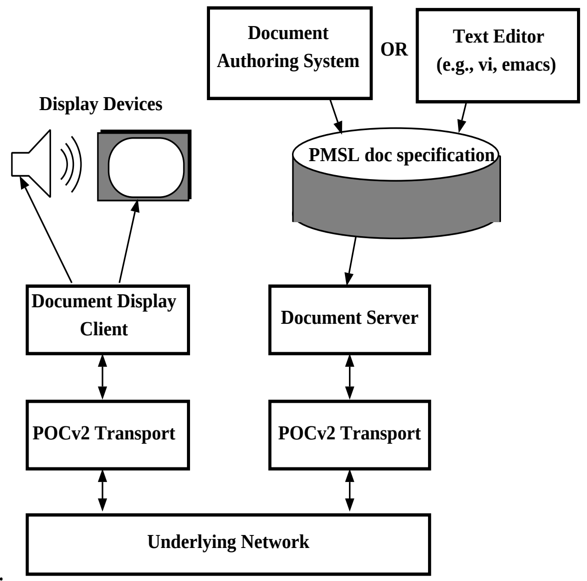
Unlike the monolithic structure of POC, the POCv2 architecture divides the transport layer into three sub-layers, two of which are proposed to be implemented at user layer as an application library, and one of which is proposed to be implemented in the Unix Kernel (see Figure 1.B.) The top layer provides the partial ordering mechanism. The middle layer provides for object segmentation and reassembly; this layer divides objects into segments based on the Path Maximum Transfer Unit (MTU) of the connection. The lowest layer is an unordered/partially-reliable segment delivery service. The target system for our prototype is Solaris 2.4, running on SPARCstation 5's and 20's.

5 Related Work

[12] describes the MEDIABASE/MEDIADOC project. This project has several features in common with our work. In both cases, (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 from their's in both focus and architecture. The focus of [12] is the design of "an advanced high-performance distributed multimedia information and communications system [DMIS] 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 partial order and partial reliability mechanisms in the transport layer; we use multimedia document retrieval 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 these ideas. Our hope is that by developing PO/PR mechanisms and demonstrating their utility for multimedia document retrieval, we can provide useful tools to the developers of advanced DMIS's such as MEDIABASE/MEDIADOC.

Summary

The authors have proposed and are in the process of constructing a prototype multimedia document retrieval system based on PMSL (a variant of the OCPN that provides for graceful degradation) and POCv2 (a partially-ordered/partially-reliable transport protocol.) The full paper will describe the results of this implementation experiment.

A. 

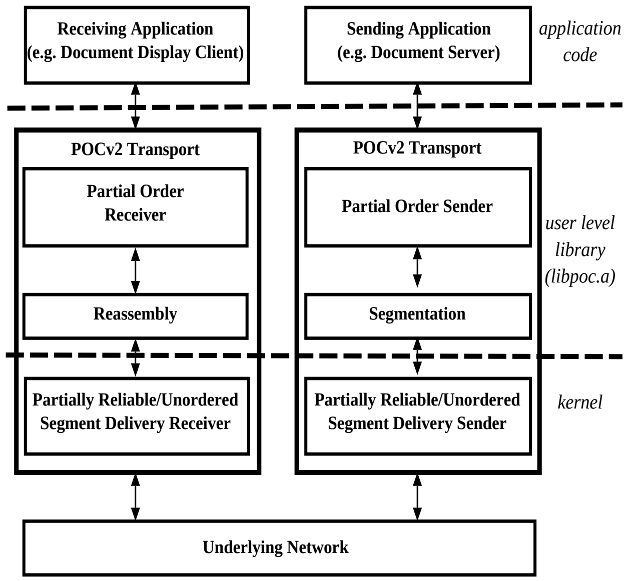
B. 
Receiving Application (e.g. Document Display Client)
Sending Application (e.g. Document Server)
application code
POCv2 Transport
Partial Order Receiver
Reassembly
Partially Reliable/Unordered Segment Delivery Receiver
POCv2 Transport
Partial Order Sender
Segmentation
Partially Reliable/Unordered Segment Delivery Sender
user level library (libpoc.a)
kernel
Underlying Network

Figure 1: (A) Doc Retrieval System Architecture (B) POCv2 Architecture

References

[1] P.D. Amer, C. Chassot, T. Connolly, M. Diaz, and P. Conrad. Partial order transport service for multimedia and other applications. *IEEE/ACM Trans on Networking*, 2(5):440–456, Oct 1994.

[2] D. Clark and D. Tennenhouse. Architectural considerations for a new generation of protocols. In *SIGCOM 90 Proceedings*. ACM, 1990.

[3] T. Connolly, P.D. Amer, and P. Conrad. Internet RFC1693: An Extension to TCP: Partial Order Service. Internet Request for Comments RFC1693, Nov 1994.

[4] Phillip T. Conrad, Paul. D. Amer, and Rahmi Marasli. Graceful degradation of multimedia documents via partial order and partial reliability transport protocols. In *IEEE Workshop on Multimedia Synchronization*, Tysons Corner, VA, May 1995. URL:`<http://spiderman.bu.edu/sync95/papers/conrad.ps>`.

[5] Bert J. Dempsey. *Retransmission-Based Error Control For Continuous Media Traffic In Packet-Swithced Networks*. PhD thesis, University of Virginia, 1994.

[6] F. Gong and G. Parulkar. An application-oriented error control scheme for high-speed networks. Technical Report WUCS-92-37, Department of Computer Science, Washington University in St. Louis, November 1992.

[7] T. Little and A. Ghafoor. Synchronization and storage models for multimedia objects. *IEEE J on Selected Areas in Comm*, 8(3):413–427, Apr 1990.

[8] R. Marasli, P. D. Amer, P. T. Conrad, and G. Burch. Partial order transport service: An analytic model. In *Ninth Annual IEEE Workshop on Computer Communications*, Marathon, Florida, October 1994.

[9] Rahmi Marasli. *Partially Ordered and Partially Reliable Transport Protocols: Performance Analysis*. PhD thesis, University of Delaware, (In progress).

[10] Rahmi Marasli, Paul D. Amer, and Phillip T. Conrad. An analytic study of partially reliable transport services. (Submitted for publication).

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

[12] J.A. Rody and A. Karmouch. A remote presentation agent for multimedia databases. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 223–230, Washington, DC, May 1995. IEEE.

[13] James Schnepf, Joseph A Konstan, and David Du. Doing FLIPS: Flexible interactive presentation synchronization. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 213–222, Washington, DC, May 1995. IEEE.