Partial Order Transport Service: An Analytic Model *

Rahmi Marasli Paul D. Amer Phillip T. Conrad Greg Burch

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

Abstract

Traditional transport service is either *(1) reliable and ordered* (e.g., TCP) or *(2) unreliable and unordered* (e.g., UDP). Some new computer applications such as *multimedia* do not fit perfectly with either of these services. Partial Order Connection (POC), a new transport-layer protocol, tries to fill the gap between (1) and (2). With POC, an application specifies a partial order (PO) for valid orderings of the packets. Packets can be transmitted or delivered by respecting any linear extension (LE) of the given PO. POC provides more generality than traditional transport protocols, which allow a user to specify the order only as a chain or an anti-chain. In POC, the reliability requirements are similarly generalized: rather than a complete guarantee as with TCP, or no guarantee as with UDP, in POC, a user specifies controlled levels of loss. Hence, POC provides *partial-order* and *partial-reliability* service. This paper presents an analytic model for POC. Results show that POC provides performance improvements over existing transport services when network service is lossy, the ratio of transmission time to propagation delay is large, or when the PO has few precedence constraints.

1 Introduction and Motivation

Current applications that need to communicate objects (i.e., images, video frames, files, sound samples) over packet-switched networks choose between classic transport services that provide either a reliable, ordered service or one that does not guarantee any ordering or reliability. However, many applications require services between these two extremes, i.e. *partial-order* service and/or *partial-reliability* service.

For example, multimedia traffic is often characterized by periodic, synchronized parallel streams of continuous-bit-rate information (e.g., audio and/or video), and/or structured image streams (e.g., displays of multiple overlapping and non-overlapping windows). Neither of the two transport services described above is a good fit with the requirements of multimedia. Audio and video streams may require more reliability than an "unreliable" service provides, but not as much as a "reliable" service. Objects destined for the same window on a remote display may have an ordering requirement, while objects destined for different windows may arrive in many different orders.

Many other possibilities exist for order and reliability besides these "all or nothing" services. Suppose we want to communicate N objects. We can represent the various possibilities for order using a partial order PO over the set $[N] = \{1, 2, \ldots, N\}$, where $x \prec y$ in PO signifies that object x must be delivered to the receiving application prior to object y. We can represent the various requirements for reliability, by using a reliability vector $R = \langle r_1, r_2, \ldots, r_N \rangle$ where each r_i indicates the level of reliability required for object i. Each value of r_i indicates how hard the transport protocol should work to guarantee object i's delivery within quality of service time constraints. For example, in a simple binary reliability approach, we may define that $r_i = 1$ signifies that object i may be lost, and $r_i = 0$ signifies that object i may not be lost.

A *partial-order, partial-reliability transport protocol* is a transport protocol that allows the user to provide a partial order PO and reliability vector R to specify the precise level of service required. *Partial Order Connection* (*POC*) is an example of such a protocol [2, 3]. Except when needed for emphasis, we usually drop the adjective *partial-reliability* and assume that a partial-order service (connection, protocol) also provides partial-reliability unless otherwise specified.

The number of objects N, the partial order PO, and reliability vector R are three components of a tuple $\mathcal{S} = \langle N, PO, R, L \rangle$ called the *service profile*. The fourth component, $L = \{\ell_1, \ell_2, \ldots, \ell_N\}$, is the *length vector*, where

*This work supported, in part, by the National Science Foundation (NCR-9314056), the US Army Communication Electronics Command (CECOM), Ft. Monmouth, and the US Army Research Office (DAAL03-91-G-0086, DAAL03-92-G-0070).

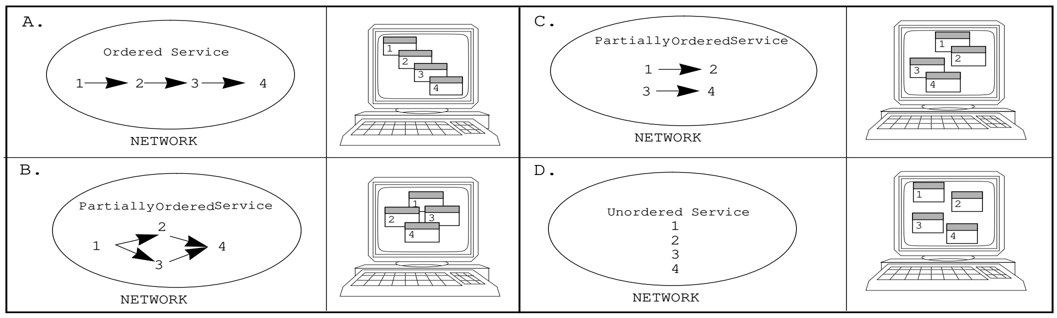

Figure 1: Screen Refresh

each ℓ_i is the length of object i in octets. The service profile contains all information needed to specify a given level of partial-order transport service. This information must be communicated from the application to both the sending and receiving transport layers in order to specify the level of order and reliability that is required.

Note reliable, ordered service is specified by letting PO be the chain $\mathbf{N} = \{1 \prec 2 \prec 3 \prec \ldots \prec N-1 \prec N\}$, and $R = \{0, 0, \ldots, 0\}$. Similarly, unreliable, unordered service is specified by the antichain $\bar{\mathbf{N}}$, and $R = \{1, 1, \ldots, 1\}$. Since PO may be any partial order of size N, and R may assume 2^N different values when defined over $\{0, 1\}$, the number of different potential service profiles for a given N is large (although finite).

1.1 Example Applications

Reference [1] reviews the development and motivation for a *partial order* protocol/service including several examples. The material is briefly summarized here. Essentially, a partial-order service can be employed and is motivated whenever a total order on the delivery of objects is not mandatory. When two objects can be delivered to a transport service user in either order, there is no need to use an ordered service that delays delivery of the second one transmitted until the first arrives. Partial reliability can be employed whenever some level of loss can be tolerated, or where objects have "temporal value"; that is, they are valuable for some period of time, and then they become worthless. Such objects are termed *loseable*. An object representing a subtitle in a film is an example of a loseable object; if the part of the movie that goes with the subtitle is already over, then the subtitle is no longer useful and may be considered lost. Other examples include a single frame in a moving picture. An example of a non-loseable object might be a still image or a window full of text.

1.1.1 A simple application for POC: Screen Refresh

Consider an application that must do a "screen refresh" on a workstation screen/display containing multiple windows (see Figure 1). In refreshing the screen from a remote source, objects (icons, still or video images) that overlap one another should be refreshed from bottom to top for optimal redisplay efficiency. However, objects that do not overlap may be refreshed in any order. Therefore, the way in which the windows overlap induces a partial order, as shown in the figure.

Consider the four cases in Figure 1. A sender wishes to refresh a remote display that contains four active windows (objects) named $\{1\ 2\ 3\ 4\}$. Assume the windows are transmitted in numerical order and receiving application refreshes windows as soon as the transport service delivers them. If the windows are configured as seen in Figure 1.A, an ordered service (sometimes referred as a FIFO channel) is required. In this case, only one ordering is permitted at the destination. If window 2 is received before window 1, the transport service must buffer window 2 in a buffer and deliver it only after window 1 arrives and is delivered.

At the other extreme, if the windows are configured as in Figure 1.D, an unordered service would suffice. Here any of $4! = 24$ delivery orderings would satisfy the application since the four windows can be refreshed in any order. As notation, four ordered objects are written $1 \prec 2 \prec 3 \prec 4$ and unordered objects are written using a parallel operator: $1||2||3||4$ ($x||y$ means there is no dependency relation between objects x and y). Figure 1.B and Figure 1.C demonstrates two (of many) window configurations that call for a partial order delivery service. In these cases, two and six orderings, respectively, are permitted at the destination.

1.1.2 A good application for POC: DEMON

The DEMON system developed by Bellcore [7, 8, 9] involves the interactive display and retrieval of multimedia documents from a remote server. Documents processed by DEMON are "temporal" in the sense that they "display themselves in real time without additional action from a viewer (unlike for example, a multimedia magazine with pages to be turned.)" Applications for such documents include: "electronic "yellow-pages", "infotainment" documents such as travelogs or documentaries, interactive maps, interactive training and educational presentations, music videos, board games, and interactive role-playing games as some examples [7]".

Consider a travelog document. In such a document, there is a sound track. At various points, still images representing scenic attractions of the area are displayed on the screen. At other times, images are removed. Occasionally, text will appear. Occasionally, a button may be displayed, allowing the user to take a "side-trip" through another set of images. Finally, towards the end of the presentation, a small window may appear with a short moving picture sequence.

The innovation of DEMON is to allow such documents to be presented over low-bandwidth channels; specifically channels in the range from 128 kbps to 1.5Mbps, which is within the capabilities of copper access facilities of the public telephone network. It is assumed that the receiving station has some local storage and processing capability, but not enough to store an entire document in advance. Therefore, to present information in real time which may require more bandwidth than that which is available, a greedy scheduling algorithm is used to deliver information in advance during periods of low bandwidth utilization—but not too far in advance, since the buffer capacity of the receiver is limited. For example, during the periods where still images are being presented (which require less bandwidth), the beginning of the motion picture sequence is being downloaded and stored. (It is for this reason, that in the scenario described above, the motion-picture portion of the document appears towards the end of the travelog, rather than at the beginning.)

Several features of DEMON make it a good candidate for POC: (1) the inclusion of overlapping and non-overlapping objects (which induces a partial order), (2) its document orientation (which allows the partial order to be known in advance), and (3) its scheduling aspect, which allows for the possibility of retransmissions.

2 Analytic Modeling of POC

Two important questions must be asked about any new protocol. The first one is "Can the protocol be implemented efficiently?" For POC, Conrad et. al. [4] addresses this question suitably and gives reasonable time bounds for the necessary operations.

The second one is "What, if any, performance improvement does the protocol provide over existing protocols and under what circumstances?" The analytic modeling presented in this paper not only addresses this question for POC, but also provides a basis for studying the effects of different *LE*s of the same *PO* on system performance. It is shown that in conditions of lossy and slow networks, POC provides performance improvements over the existing protocols.

The model introduced in this paper emphasizes the partial order aspect of POC, and assumes that reliable service is required. Further simplifying assumptions are discussed in the next section.

2.1 Introduction to Model

In a summarization of the OSI model, we use a three layer architecture which includes only the network layer, the transport layer, and the user application layer (see Figure 2). In the network layer, the loss of a packet or an ack is determined by a Bernoulli process and a constant end-to-end network delay is assumed.

POC_Transport_Sender takes a packet from User_Sender, transmits the packet over the network, then sets a timer and buffers it. If the corresponding ack does not arrive within its timeout period, that packet is retransmitted. By assumption, there is no problem with running out of buffer space at the POC_Transport_Receiver. When a packet is received, if it is deliverable (i.e., if all the packets that this packet depends on are already delivered), then it is delivered to User_Receiver; otherwise it is buffered. After delivering a packet, POC_Transport_Receiver checks its buffers for any packets that have become deliverable as a result of the most recent delivery; these packets are also then delivered.

It is assumed that User_Sender submits packets to POC_Transport_Sender by respecting the given *PO*. It is also assumed that all packets are available to be given to POC_Transport_Sender at User_Sender. [1] User_Receiver

[1] This may not be the case for a multimedia application, because there may be times when the application is not yet ready to send something.

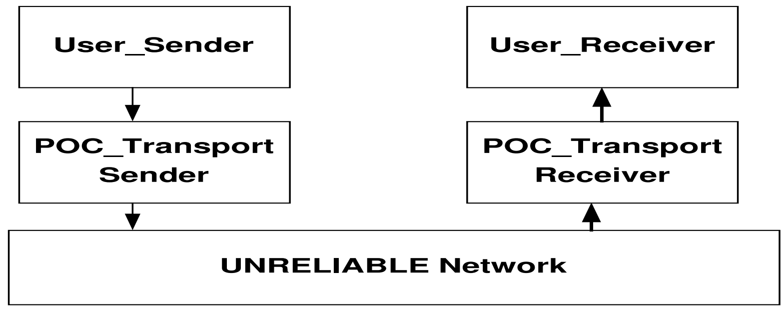

Figure 2: Model

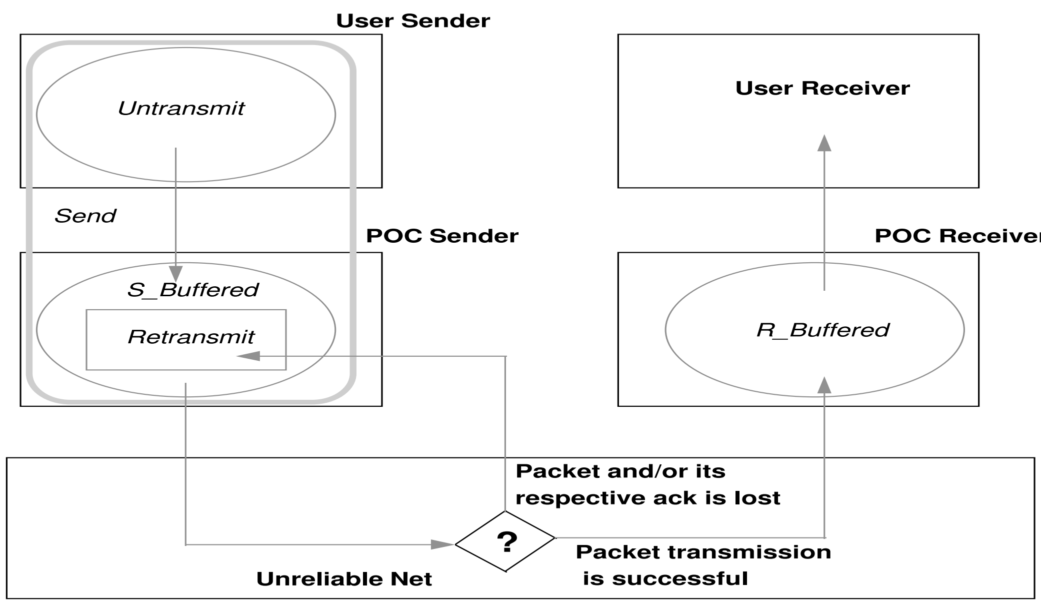

Figure 3: Important Sets

just accepts packets from POC_Transport_Receiver.

Now some important sets are defined to help understand the model (see Figure 3).

- *Untransmit Set:* Packets not yet given to the POC_Transport_Sender by the User_Sender for transmission.
- *S_Buffered Set:* Packets at POC_Transport_Sender that have been transmitted and are awaiting acknowledgment.
- *Retransmit Set:* Subset of packets in S_Buffered Set eventually needing retransmission by POC_Transport_Sender (The elements of this set are determined by a stochastic process).
- *Send Set:* Union of Untransmit and S_Buffered sets.
- *R_Buffered Set:* Packets received by POC_Transport_Receiver but not yet delivered to User_Receiver.

2.2 System Description

Table 1 includes the definitions for system variables. The important assumptions are in Table 2.

System Variables	Definitions
t_{pack}	packet transmission time
t_{ack}	ack transmission time
t_{prop}	propagation delay
t_{out}	timeout period for retransmissions
p	probability of losing a packet within network layer
q	probability of losing an ack within network layer
RT	round trip delay, defined as $t_{pack} + t_{ack} + 2t_{prop}$
Buf_S	number of buffers at POC_Transport_Sender
Buf_R	number of buffers at POC_Transport_Receiver
p_{succ}	probability of a successful packet transmission for POC_Transport_Sender, defined as $(1-p)*(1-q)$

Table 1: System Variables

	ASSUMPTIONS
1	p and q are fixed and independent for each packet and ack transmission
2	$t_{out} = RT$, and t_{out}, t_{prop}, t_{ack} and t_{pack} are constants
3	$t_{ack} \leq t_{pack}$
4	t_{out} is an integral multiple of t_{pack}
5	Processing time of a packet or an ack at each side is negligible
6	$Buf_S = \frac{t_{out}}{t_{pack}}$ and $Buf_R = \infty$
7	Only selective acks are used (i.e. no piggy-backing or cumulative acks)
8	All packets in *Untransmit Set* are ready at User_Sender or, equivalently, a packet arrives at User_Sender at every t_{pack} time
9	Packets in *Retransmit Set* have priority over the packets in *Untransmit Set* for transmission
10	User_Sender submits packets to POC_Transport_Sender by respecting the given PO

Table 2: Assumptions

We will refer to this system as *NET*. Hence $NET = \langle t_{pack}, t_{ack}, t_{prop}, t_{out}, p, q, RT, Buf_S, Buf_R, p_{succ}, A\rangle$, where t_{pack} through p_{succ} represent the system variables and A stands for the assumptions given in Table 2. All subsequent values and computations in this paper will refer to this given fixed *NET* unless otherwise stated.

2.2.1 Explanation of Assumptions

Table 2's assumptions simplify the problem and help in better understanding the model. With regard to these assumptions, the following points are worth noting:

- In assumption 2, constant t_{pack} implies fixed packet size. Constant t_{prop} reflects the fixed end-to-end network delay.
- As a result of assumptions 2, 5, 6 and 8, the transmitter at POC_Transport_Sender is never idle.
- As a result of assumptions 2, 4, 5, and 8, time can be viewed as slotted in slots of size t_{pack}.
- As a result of assumption 9, a new packet from *Untransmit Set* is accepted for transmission only if no timeout of any previously transmitted packet expires at the beginning of the current time slot.

2.3 Definitions

In this section, the important variables and target values are defined (See figure 4). Table 3 defines the important variables, and the target values to be computed for the system NET are defined in Table 4.

Of the variables in Table 3, only N, PO, and LE are independent variables. All others depend on N, PO, LE, and the system NET. $Dist_{a,b}(LE)$ is most important since it is used to express the LE information in the target values.

Variable Names	Definitions
s_{a_i}, s_{b_i}	time that the "i^{th}" (re)transmission of packets a, b starts at POC_Transport_Sender, respectively
r_a, r_b	time that packets a, b are received by POC_Transport_Receiver, respectively
d_a, d_b	time that packets a, b are delivered to User_Receiver, respectively
d_{last}	time that the last packet is delivered to User_Receiver
$retran_{a,b}$	number of retransmissions (of any packet) between s_{a_1} and s_{b_1}
N	total number of packets
PO	a partial order
LE	a linear extension
$Dist_{a,b}(LE)$	distance between packet a and packet b in the linear extension LE; defined as "seq(b)-seq(a)" where seq(x) returns the assigned sequence number for packet x

Table 3: Important Variables

Target Values	Definitions
$\overline{retran}_{a,b}(LE)$	E(number of retransmissions (of any packet) between s_{a_1} and s_{b_1})
$Buf_{a,b}(PO,LE)$	E(time between the arrival times of packets a and b at POC_Transport_Receiver when a arrives after) b defined as E($r_a - r_b$ when $r_a > r_b$) if $a \prec b$; 0 otherwise
$Buf_a(PO,LE)$	E(time that packet a is buffered at POC_Transport_Receiver) defined as E(d_a - r_a)
$pBuf_{a,b}(PO,LE)$	P(packet a arrives after packet b) defined as P($r_a > r_b$) if $a \prec b$; 0 otherwise
$pBuf_a(PO,LE)$	P(packet a is buffered at POC_Transport_Receiver) defined as P($d_a > r_a$)
$T_{end-end_a}(PO,LE)$	E(end-to-end delay for packet a) defined as E(d_a - s_{a_1})
$T_{msg}(PO,LE)$	E(end-to-end message delay) defined as E(d_{last} - s_{1_1})
$T_{a,b}(PO,LE)$	E(time between first transmissions of packet a and packet b) defined as E(s_{b_1} - s_{a_1})
$F_{a,b}(PO,LE)$	E(number of full timeout periods that will occur between s_{a_1} and s_{b_1}) defined as $E\lfloor T_{a,b}(PO,LE)/t_{out}\rfloor$
$\overline{R\_Buffered}(PO,LE)$	E(memory utilization at POC_Transport_Receiver)
$\eta(PO,LE)$	efficiency of the system, a number between 0 and 1, defined as: $$\eta = \frac{\text{time spent on successful transmissions}}{\text{total time}}$$

Table 4: Target Values

All target values depend on the system NET, LE and/or PO. We will study the effects of PO and LE on system performance through these values.

In Figure 4, packet a should be delivered to User_Receiver before packet b (that is, in the partial order PO, $a \prec b$). Packets a and b (and possibly other packets) get lost several times before reaching the POC_Transport_Receiver. Packet a succeeds at its i^{th} attempt while b succeeds at its j^{th} attempt. Since the first successful arrival of packet b occurs before the first successful arrival of packet a, packet b is stored in a buffer at the POC_Transport_Receiver, "waiting" for packet a to arrive. After packet a's arrival, packet b waits for the other depending packets to arrive. Then the delivery of packet b to User_Receiver happens at time d_b.

2.4 Computation of Target Values

Our goal is to investigate the buffering, delay, and efficiency characteristics of the system for a given PO and LE. The analysis will proceed as follows:

- First, we derive a formula for $\overline{retran}_{a,b}(LE)$, expected number of retransmissions (of any packet) between the first transmissions of packets a and b.
- Next, $\overline{retran}_{a,b}(LE)$ is used to compute $F_{a,b}(PO,LE)$, the expected number of full timeout periods between the initial transmissions of packets a and b.

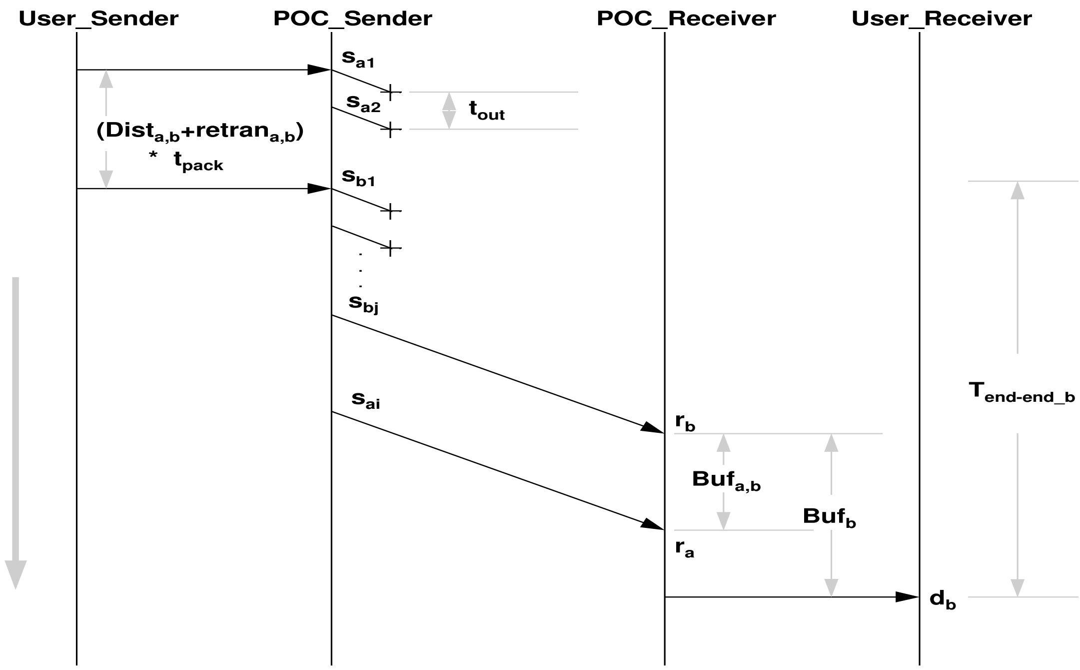

Figure 4: Target Values

- Then, after computing buffering probabilities and times, i.e., $pBuf_{a,b}(PO, LE)$, $Buf_{a,b}(PO, LE)$, $pBuf_a(PO, LE)$, and $Buf_a(PO, LE)$, end-to-end packet delay is computed by using $Buf_a(PO, LE)$.
- Finally, the formula for $\eta(PO, LE)$, the efficiency of the system, is derived by Little's theorem. Then $\eta(PO, LE)$ is used to compute end-to-end message delay and the expected memory utilization at POC_Transport_Receiver.

In addition, it turns out that $Buf_{a,b}(PO, LE)$ and $pBuf_{a,b}(PO, LE)$ have nice closed-form formulae and are useful for studying the effects of linear extensions. By studying these formulae closely, we can determine the best conditions for using Protocol POC.

2.4.1 Number of retransmissions (of any packet) between initial transmissions of a specific pair of packets: $retran_{a,b}$

$retran_{a,b}$, the number of retransmissions (of any packet) in time period $[s_{a_1}, s_{b_1}]$, will be computed for $(a \prec b)$. Any unsuccessful transmission that occurs in time period $[s_{a_1} - t_{out}, s_{b_1} - t_{out}]$ will be repeated in time period $[s_{a_1}, s_{b_1}]$. Notice that the packet transmission that starts at time $s_{a_1} - t_{out}$ should be successful since at time s_{a_1}, the first transmission of packet a starts (which is not a retransmission). Therefore, the number of transmissions that can lead to a retransmission in time period $[s_{a_1} - t_{out}, s_{b_1} - t_{out}]$ is $Dist_{a,b}(LE) + retran_{a,b} - 1$. Out of $Dist_{a,b}(LE) + retran_{a,b} - 1$ transmissions, $retran_{a,b}$ of them will result in a retransmission. Therefore;

$$P(retran_{a,b} = i) = \binom{Dist_{a,b}(LE) + i - 1}{i} * (p_{succ})^{Dist_{a,b}(LE)} * (1 - p_{succ})^i \tag{1}$$

This is a negative binomial distribution with mean and variance :

$$\overline{retran}_{a,b}(LE) = \frac{1 - p_{succ}}{p_{succ}} * Dist_{a,b}(LE) \qquad var(retran_{a,b}(LE)) = \frac{1 - p_{succ}}{(p_{succ})^2} * Dist_{a,b}(LE) \tag{2}$$

2.4.2 Expected number of timeout periods between initial transmissions: $F_{a,b}(PO, LE)$

$F_{a,b}(PO, LE)$ is the expected number of full timeout periods that will occur between s_{a_1} and s_{b_1}. If packet a is successfully transmitted at any time during these periods, then packet a will have no effect on whether or not packet b is buffered. Other packets that b depends upon may cause b to be buffered, but not a.

$$F_{a,b}(PO, LE) = \lfloor \frac{(\overline{retran}_{a,b}(LE) + Dist_{a,b}(LE)) * t_{pack}}{t_{out}} \rfloor \quad (3)$$

$$= \lfloor \frac{Dist_{a,b}(LE)}{Buf_S * p_{succ}} \rfloor \quad (4)$$

$F_{a,b}(PO, LE)$ increases step-wise with $Dist_{a,b}(LE)$, i.e., as $Dist_{a,b}(LE)$ gets larger, $F_{a,b}(PO, LE)$ also gets larger or stays the same. While not a surprise, this result is encouraging, because it allows the user to judiciously space the sending order of dependent packets by some distance, thereby reducing their effect on one another. This implies that the sender may influence the overall system performance by wise decisions in packet sending order.

2.4.3 Buffering probability for a specific pair of packets: $pBuf_{a,b}(PO, LE)$

$pBuf_{a,b}(PO, LE)$ is the probability that packet a will be received after packet b (thus resulting in buffering packet b). This value will be computed when b's delivery depends on a's previous delivery, i.e., $(a \prec b)$. (If there is no dependency relation between a and b, this value is zero by definition.)

$$pBuf_{a,b}(PO, LE) = P(r_a > r_b) \quad (5)$$

$$= 1 - P(r_a \leq r_b) \quad (6)$$

$$= 1 - \sum_{j=1}^{\infty} (1-p) * p^{j-1} * P(r_a \leq r_b \mid b \text{ succeeds at } j^{th} \text{ attempt}) \quad (7)$$

$$= \frac{p}{1+p} * p^{F_{a,b}(PO,LE)} \quad (8)$$

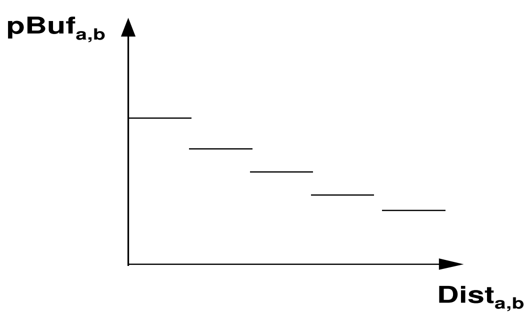

Figure 5: Relationship between $pBuf_{a,b}(PO, LE)$ and $Dist_{a,b}(LE)$

Thus, $pBuf_{a,b}(PO, LE)$ decreases exponentially as $F_{a,b}(PO, LE)$ increases. Since $F_{a,b}(PO, LE)$ increases step-wise with $Dist_{a,b}(LE)$, $pBuf_{a,b}(PO, LE)$ decreases step-wise exponentially as $Dist_{a,b}(LE)$ increases (see Figure 5). This is again an encouraging result since by putting some distance between two dependent packets, the sender side can decrease the buffering probability of one of those packets at the receiving side.

2.4.4 Buffering time for a specific pair of packets: $Buf_{a,b}(PO, LE)$

The value $Buf_{a,b}(PO, LE)$, the expected time that packet b is buffered waiting for packet a to arrive, will be computed for $(a \prec b)$. This value is zero by definition when there is no dependency between a and b; in that case b does not have to wait for a in the buffers of the POC_Transport_Receiver.

Notice that r_a can be greater than r_b if and only if packet a fails at least $1 + F_{a,b}(PO, LE)$ times. This can be seen as follows: there will be $F_{a,b}(PO, LE)$ timeout periods between s_{a_1} and s_{b_1} and thus $F_{a,b}(PO, LE)$ retransmissions of a, if necessary, before b is transmitted for its first time. Additionally, a will have its first transmission which, if successful, will not affect packet b. Hence packet a must have at least a total of $1 + F_{a,b}(PO, LE)$ failures in order to have a chance to influence what happens to packet b.

$$Buf_{a,b}(PO, LE) = E(r_a - r_b \text{ when } r_a > r_b)$$

$$
\begin{aligned}
&= \sum_{i=2+F_{a,b}(PO,LE)}^{\infty} (1-p)*p^{i-1}*E(r_a - r_b \mid a \text{ succeeds at } i^{th} \text{ attempt}) \\
&= \left(\frac{(F_{a,b}(PO,LE)+1)*p + F_{a,b}(PO,LE)*p^4 - 2*F_{a,b}(PO,LE)*p^3}{1-p^2}\right)*p^{F_{a,b}(PO,LE)}*t_{out} \\
&\quad + \left(\frac{p^{F_{a,b}(PO,LE)+3}}{1+p} - p^{F_{a,b}(PO,LE)+1}\right)*\left(\overline{retran}_{a,b}(LE) + Dist_{a,b}(LE)\right)*t_{pack} \qquad (9)
\end{aligned}
$$

(9) is a complicated expression. But if network is lossy and/or the $(\frac{\text{transmission time}}{\text{propagation delay}})$ ratio is large, then $F_{a,b}(PO,LE) = \lfloor\frac{Dist_{a,b}(LE)}{Buf_S * p_{succ}}\rfloor$ can be approximated with $\frac{Dist_{a,b}(LE)}{Buf_S * p_{succ}}$, and expression (9) can be simplified to:

$$
Buf_{a,b}(PO,LE) = \frac{p}{1-p^2}*p^{F_{a,b}(PO,LE)}*t_{out} = pBuf_{a,b}(PO,LE)\,\frac{t_{out}}{1-p}
$$

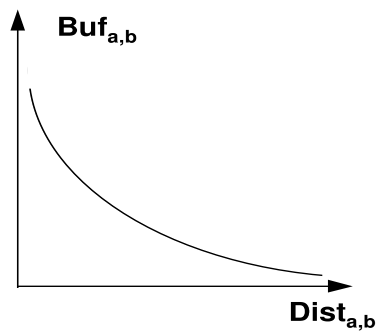

Figure 6: Relationship between $Buf_{a,b}(PO,LE)$ and $Dist_{a,b}(LE)$ when $F_{a,b}(PO,LE) \simeq \frac{Dist_{a,b}(LE)}{Buf_S * p_{succ}}$

$Buf_{a,b}(PO,LE)$ decreases with larger $Dist_{a,b}(LE)$ values (see Figure 6). When $F_{a,b}(PO,LE) \simeq \frac{Dist_{a,b}(LE)}{Buf_S * p_{succ}}$, $Buf_{a,b}(PO,LE)$ decreases exponentially as $Dist_{a,b}(LE)$ increases. This means, in this special case, even a small increase in $Dist_{a,b}(LE)$ values can make a large difference in reducing the buffering times of packet b at the receiver.

2.4.5 Overall buffering probability for a packet: $pBuf_a(PO,LE)$

Before computing the overall probability of having to buffer a given packet, two more set definitions are introduced:[2]

- $before_a(PO)$ *Set:* Packets that must be delivered to User_Receiver before packet a.
- $after_a(PO)$ *Set* : Packets that must be delivered to User_Receiver after packet a.

PO information is expressed in the target values through these sets. Having these sets for every packet completely defines a PO. The definition of $pBuf_a(PO,LE)$ is $P(a$ will be buffered at POC_Transport_Receiver) which is equivalent to $1 - P(a$ will not be buffered at POC_Transport_Receiver). Then:

$$
\begin{aligned}
pBuf_a(PO,LE) &= 1 - \sum_{i=1}^{\infty}(1-p)*p^{i-1}*P(a \text{ will not be buffered at POC\_Transport\_Receiver} \mid a \text{ succeeds at } i^{th} \text{ attempt}) \\
&= 1 - \sum_{i=1}^{\infty}(1-p)*p^{i-1}*P(a \text{ arrives after all packets in } before_a(PO) \mid a \text{ succeeds at } i^{th} \text{ attempt}) \qquad (10)
\end{aligned}
$$

$pBuf_a(PO,LE)$ will be equivalent to the following values for the given cases:

- when $before_a(PO) = \{b\}$, then

[2] These sets are defined slightly differently in [5] with names $\downarrow a$ and $\uparrow a$: $\downarrow a = before_a(PO) \cup \{a\}$ and $\uparrow a = after_a(PO) \cup \{a\}$.

$$pBuf_a(PO,LE) = \frac{p}{1+p} * p^{F_{b,a}(PO,LE)} = pBuf_{b,a}(PO,LE)$$

- when $before_a(PO) = \{b_1, b_2\}$, then

$$pBuf_a(PO,LE) = \frac{p}{1+p} * \left(p^{F_{b_1,a}(PO,LE)} + p^{F_{b_2,a}(PO,LE)}\right) - P(a \text{ arrives before } b_1 \text{ and } b_2)$$

If the packet loss probability, p, is small, then it is unlikely that packet a would overtake two preceding packets. Hence, when p is small, buffering probability for packet a can be approximated as:

$$pBuf_a(PO,LE) \simeq \sum_{\forall\, b\ in\ before_a(PO)} P(a \text{ would overtake } b) \quad (11)$$

$$\simeq \sum_{\forall\, b\ in\ before_a(PO)} pBuf_{b,a}(PO,LE) \quad (12)$$

If the density of PO is small [3] (i.e., relatively few ordering constraints), then $|before_a(PO)|$ will tend to be small, leading to fewer terms in the expression for $pBuf_{b,a}(PO,LE)$ (see expression 12). $Dist_{b,a}(LE)$ values between dependent packets can in general be made larger with LEs of low density POs. Combining these two observations, we can say that if PO has low density, then $pBuf_a(PO,LE)$ values will tend to be smaller.

2.4.6 Expected buffering time for a packet: $Buf_a(PO,LE)$

This is the expected buffering time for packet a at POC_Transport_Receiver.

$$\begin{aligned}
Buf_a(PO,LE) &= E(\text{time that } a \text{ is buffered at POC_Transport_Receiver}) \\
&= \sum_{\forall\, b\ in\ before_a(PO)} E(r_b - r_a \mid b \text{ arrives after all packets in } \{a\} \cup before_a(PO) - \{b\}) * \\
&\quad P(b \text{ arrives after all packets in } \{a\} \cup before_a(PO) - \{b\}) \\
&= \sum_{\forall\, b\ in\ before_a(PO)} P(b \text{ arrives after all packets in } \{a\} \cup before_a(PO) - \{b\}) * \\
&\quad \Big(\sum_{j=2+F_{b,a}(PO,LE)}^{\infty} P(b \text{ succeeds at } j^{th} \text{ attempt}) * \\
&\quad E(r_b - r_a \mid b \text{ arrives after all packets in } \{a\} \cup before_a(PO) - \{b\} \text{ and } b \text{ succeeds at } j^{th} \text{ attempt})\Big)
\end{aligned}$$

It is obvious that there is a dependency between $Buf_a(PO,LE)$ values and $\langle PO, LE\rangle$. The authors are continuing to pursue a simpler form for this expression.

2.4.7 End-to-end packet delay: $T_{end-end_a}(PO,LE)$

$T_{end-end_a}(PO,LE)$, the expected end-to-end packet delay, is by definition, equivalent to $E(d_a - s_{a_1})$. Therefore:

$$\begin{aligned}
T_{end-end_a}(PO,LE) &= E(r_a - s_{a_1} + d_a - r_a) && (13)\\
&= E(r_a - s_{a_1}) + Buf_a(PO,LE) && (14)\\
&= t_{pack} + t_{prop} + \frac{p}{1-p} * t_{out} + Buf_a(PO,LE) && (15)
\end{aligned}$$

Notice that "$t_{pack} + t_{prop} + \frac{p}{1-p} * t_{out}$" is constant and independent of $\langle PO, LE\rangle$. Hence, the relationship between $\langle PO, LE\rangle$ and $T_{end-end_a}(PO,LE)$ is exactly the same as that between $Buf_a(PO,LE)$ and $\langle PO, LE\rangle$. The $\langle PO, LE\rangle$ that makes the expected buffering time for packet a minimal also minimizes the expected end-to-end packet delay for a.

[3] The *density* of a partial order is a measurement defined as follows [6]. Let D represent the cardinality of the set of all ordered pairs (a,b) such that $a \prec b$ in PO. The maximum value for D is $n(n-1)/2$, therefore the density, d, is defined by the ratio $d = 2D/[n(n-1)]$. For a chain, d=1; for an antichain, d=0.

2.4.8 End-to-end message delay: $T_{msg}(PO, LE)$

$T_{msg}(PO, LE)$, the expected end-to-end message delay, is equivalent to $E(d_{last} - s_{1_1})$. Let us assume that packet a is the last one to be delivered to User_Receiver. Then:

$$\begin{aligned}
T_{msg}(PO, LE) &= E(d_a - s_{a_1}) + E(s_{a_1} - s_{1_1}) && (16)\\
&= (Retran_{1,a}(LE) + Dist_{1,a}(LE)) * t_{pack} + T_{end-end_a}(PO, LE) && (17)\\
&= (Retran_{1,a}(LE) + Dist_{1,a}(LE)) * t_{pack} + t_{pack} + t_{prop} + \frac{p}{1-p} * t_{out} + Buf_a(PO, LE) && (18)
\end{aligned}$$

We would like to use this expression to gain insight into the relationship between $T_{msg}(PO, LE)$ and $Dist_{1,a}(LE)$. However, it appears that mere inspection will not suffice. If $Dist_{1,a}(LE)$ increases, then the first term of the expression increases; however the last term, $Buf_a(PO, LE)$, might decrease. The case where $Dist_{1,a}(LE)$ decreases is similarly ambiguous. Therefore, we will pursue this question by deriving $\eta(PO, LE)$, the efficiency of the system.

2.4.9 Efficiency of The System

With the assumptions 1, 2, 4, 5, 6, and 8, the number of packets at POC_Transport_Sender is always Buf_S, and the expected time that a packet spends in a buffer of POC_Transport_Sender is t_{out}/p_{succ}. Therefore, by Little's theorem:

$$Buf_S = \frac{\eta(PO, LE)}{t_{pack}} * \frac{t_{out}}{p_{succ}} \tag{19}$$

Since $Buf_S = t_{out}/t_{pack}$,

$$\eta(PO, LE) = p_{succ} = (1-p) * (1-q) \tag{20}$$

Interestingly, if $Buf_R = \infty$, then the system efficiency does not depend on $\langle PO, LE \rangle$.[4] From now on, we will refer to this efficiency value as η_{NET_∞}, i.e., $\eta_{NET_\infty} = p_{succ} = (1-p) * (1-q)$. η_{NET_∞} is the optimal efficiency value that can be achieved in a system NET.

Now we will compute end-to-end message delay and the expected memory utilization at the receiver through the system efficiency:

- If N, the total number of packets, is large, then the transient period will be small as compared to total time. Hence for this situation:

$$T_{msg}(PO, LE) \simeq N * \frac{t_{pack}}{\eta_{NET_\infty}} + t_{prop} \tag{21}$$

  As before, if $Buf_R = \infty$, then $T_{msg}(PO, LE)$ does not depend on $\langle PO, LE \rangle$.

- The expected memory utilization at the receiver, $\overline{R\_Buffered}(PO, LE)$, can be computed by Little's theorem as follows:

$$\overline{R\_Buffered}(PO, LE) = \frac{\eta_{NET_\infty}}{t_{pack}} * \frac{\sum_{i=1}^{N} Buf_i(PO, LE)}{N} \tag{22}$$

This result shows that if $Buf_R = \infty$, then the dependency relationship between $\overline{R\_Buffered}(PO, LE)$ and $\langle PO, LE \rangle$ is exactly same as that between $Buf_a(PO, LE)$ values and $\langle PO, LE \rangle$. That is, the $\langle PO, LE \rangle$ that minimizes the overall average buffering times will also minimize the expected memory utilization at the receiver. Notice that the dependency between $\overline{R\_Buffered}(PO, LE)$ and $\langle PO, LE \rangle$ exists even when $Buf_R = \infty$. Hence the sender has some control over the expected memory utilization at POC_Transport_Receiver by its ordering of packets transmitted.

3 Conclusions and Future Work

One can conclude that the Protocol POC provides performance benefits under the following conditions: (1) the network service is lossy (as in mobile networks, or the networks in times of high stress, i.e., networks in disaster

[4] There is a dependency between $\langle PO, LE\langle$ and the system efficiency in the presence of flow-control and/or finite buffers at the receiver. Due to page limitations, we could not include the discussion on the effects of flow-control and finite buffers at the receiver on system performance in this paper.

areas), (2) $\frac{\text{transmission time}}{\text{propagation delay}}$ ratio is large (as in low-speed networks). The results also suggest that the gain from this protocol can increase as the density of PO decreases.

Similarly, we have shown that the sending choice of LE for a given PO affects system performance. Specifically, the following performance metrics can be improved by protocol POC:

1. Buffering probabilities at POC_Transport_Receiver
2. Buffering times at POC_Transport_Receiver
3. Efficiency of the system
4. Expected memory utilization at POC_Transport_Receiver
5. End-to-end message delay
6. End-to-end packet delay

This paper has presented an analysis of a model which assumes constant end-to-end propagation delay at the network layer, constant packet/ack size, selective acks, and no flow control. Additionally, we have studied the effects of flow control, and/or finite buffers at the receiver on performance metrics of POC, but due to size limitations, that analysis does not appear in this paper.

Future analytic study of POC may include modeling the effects of variable packet sizes, variable propagation delay at the network layer, different arrival processes, and different acknowledgment schemes (e.g. cumulative acks) and the incorporation of partial reliability into the model. In addition to analytic modeling, the authors are also pursuing simulation and experimentation as a means to study POC performance; we are designing a simulation model, and an IP-based implementation of POC that will run over the Internet.

References

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

[2] Paul D. Amer, C. Chassot, Thomas J. Connolly, and M. Diaz. Partial order transport service for multimedia applications: Reliable service. In *Proc 2nd High Performance Dist'd Computing Conf (HPDC)*, pages 272–280, Spokane, Washington, July 1993.

[3] Paul D. Amer, C. Chassot, Thomas J. Connolly, and M. Diaz. Partial order transport service for multimedia applications: Unreliable service. In *Proc 3nd International Networking Conf (INET)*, pages BFA 1–10, San Francisco, CA, Aug 1993.

[4] Phillip T. Conrad, Paul D. Amer, and Thomas J. Connolly. Improving performance in transport-layer communications protocols by using partial orders and partial reliability. Technical Report 95-03, CIS department, University of Delaware, 1994.

[5] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 1990.

[6] William V. Gehrlein. On methods for generating random partial orders. *Operations Research Letters*, 5(6):285–291, December 1986.

[7] Darren New, Jonathan Rosenberg, Gil Cruz, and Thomas Judd. Requirements for network delivery of stored interactive multimedia. In *Network and Operating System Support for Digital Audio and Video: Proceedings of the Third International Workshop*, pages 157–163, La Jolla, CA, November 1992. Springer-Verlag. number 712 in Lecture Notes in Computer Science.

[8] Jonathan Rosenberg, Gil Cruz, and Thomas Judd. Presenting multimedia documents over a digital network. *Computer Communications*, pages 15(6) 374–380, August 1992.

[9] Jonathan Rosenberg, Robert E. Kraut, Louis Gomez, and C. Alan Buzzard. Multimedia communications for users. *IEEE Communications Magazine*, pages 20–36, May 1992.