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: {marashi,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 $x < y$ 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 $R = (r_1, r_2, \ldots, r_N)$ 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 $S = (N, PO, R, L)$ called the service profile. The fourth component, $L = \{l_1, l_2, \ldots, l_N\}$, 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 (DAAL03-94-G-0086, DAAL03-92-G-0070).
each \( \ell_i \) 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 \( N = \{1 \prec 2 \prec 3 \prec \ldots \prec N - 1 \prec N \} \), and \( R = \{0, 0, \ldots, 0\} \). Similarly, unreliable, unordered service is specified by the antichain \( \bar{N} \), 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 LEs 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 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.
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.
### Table 1: System Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{pack}}$</td>
<td>packet transmission time</td>
</tr>
<tr>
<td>$t_{\text{ack}}$</td>
<td>acknowledgment transmission time</td>
</tr>
<tr>
<td>$t_{\text{prop}}$</td>
<td>propagation delay</td>
</tr>
<tr>
<td>$t_{\text{wait}}$</td>
<td>timeout period for retransmissions</td>
</tr>
<tr>
<td>$p$</td>
<td>probability of losing a packet within network layer</td>
</tr>
<tr>
<td>$q$</td>
<td>probability of losing an acknowledgment within network layer</td>
</tr>
<tr>
<td>$RT$</td>
<td>round trip delay, defined as $t_{\text{pack}} + t_{\text{ack}} + 2t_{\text{prop}}$</td>
</tr>
<tr>
<td>$Buf_S$</td>
<td>number of buffers at POC Transport Sender</td>
</tr>
<tr>
<td>$Buf_R$</td>
<td>number of buffers at POC Transport Receiver</td>
</tr>
<tr>
<td>$p_{\text{succ}}$</td>
<td>probability of a successful packet transmission for POC Transport Sender, defined as $(1-p) \times (1-q)$</td>
</tr>
</tbody>
</table>

We will refer to this system as $NET$. Hence $NET = \{t_{\text{pack}}, t_{\text{ack}}, t_{\text{prop}}, t_{\text{wait}}, p, q, RT, Buf_S, Buf_R, p_{\text{succ}}, A\}$, where $t_{\text{pack}}$ through $p_{\text{succ}}$ 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.

### Table 2: Assumptions

<table>
<thead>
<tr>
<th>Assumption</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$p$ and $q$ are fixed and independent for each packet and acknowledgment transmission</td>
</tr>
<tr>
<td>2</td>
<td>$t_{\text{wait}} = RT$, and $t_{\text{wait}}$, $t_{\text{prop}}$, $t_{\text{ack}}$, and $t_{\text{pack}}$ are constants</td>
</tr>
<tr>
<td>3</td>
<td>$t_{\text{ack}} \leq t_{\text{pack}}$</td>
</tr>
<tr>
<td>4</td>
<td>$t_{\text{wait}}$ is an integral multiple of $t_{\text{pack}}$</td>
</tr>
<tr>
<td>5</td>
<td>Processing time of a packet or an acknowledgment at each side is negligible</td>
</tr>
<tr>
<td>6</td>
<td>$Buf_S = \frac{t_{\text{wait}}}{t_{\text{pack}}}$ and $Buf_R = \infty$</td>
</tr>
<tr>
<td>7</td>
<td>Only selective acknowledgments are used (i.e., no piggybacking or cumulative acknowledgments)</td>
</tr>
<tr>
<td>8</td>
<td>All packets in $Untransmit Set$ are ready at User Sender or, equivalently, a packet arrives at User Sender at every $t_{\text{pack}}$ time</td>
</tr>
<tr>
<td>9</td>
<td>Packets in $Retransmit Set$ have priority over the packets in $Untransmit Set$ for transmission</td>
</tr>
<tr>
<td>10</td>
<td>User Sender submits packets to POC Transport Sender by respecting the given $PO$</td>
</tr>
</tbody>
</table>

### 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 $t_{\text{pack}}$ implies fixed packet size. Constant $t_{\text{prop}}$ 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 $t_{\text{pack}}$.
- 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.
De/\text{initions}

Transport

total number of packets

Receiver, respectively

a linear extension

Receiv er defined as E(Pac k et Receiver before Pac k et Sender), respectively

time that Pac k et is deliv ered to User Receiver

reglo Retran a,b

number of retransmissions (of any pac k et) between s_{a{\ell}} and s_{b{\ell}}

N

a full number of pac k ets

PO a partial order

LE a linear extension

Dist a,b (LE) distance between Pac k et a and Pac k et b in the linear extension LE; defined as “seq[b] - seq[a]” where seq[x] returns the assigned sequence number for pac k et x

Table 3: Important Variables

<table>
<thead>
<tr>
<th>Variable Names</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retran a,b (LE)</td>
<td>E(time between retransmissions (of any pac k et) between s_{a{\ell}} and s_{b{\ell}})</td>
</tr>
<tr>
<td>Buf a,b (PO, LE)</td>
<td>E(time between arrival times of Pac k et a and b at POC Transport Receiver when a arrives after) b defined as E[r_{a} - r_{b} when r_{a} &gt; r_{b}] if a &lt; b; 0 otherwise</td>
</tr>
<tr>
<td>Buf a (PO, LE)</td>
<td>E(time that Pac k et a is buffered at POC Transport Receiver) defined as E[d_{a} - r_{a}]</td>
</tr>
<tr>
<td>Buf a,b (PO, LE)</td>
<td>P(Pac k et a arrives at Pac k et b) defined as P(d_{a} &gt; r_{b}) if a &lt; b; 0 otherwise</td>
</tr>
<tr>
<td>Buf a (PO, LE)</td>
<td>P(Pac k et a is buffered at POC Transport Receiver) defined as P(d_{a} &gt; r_{a})</td>
</tr>
<tr>
<td>t_{end-to-end} (PO, LE)</td>
<td>E(end-to-end delay for Pac k et a) defined as E[d_{a} - s_{a{\ell}}]</td>
</tr>
<tr>
<td>t_{end-to-end} (PO, LE)</td>
<td>E(end-to-end message delay) defined as E[d_{a{\ell}} - s_{a{\ell}}]</td>
</tr>
<tr>
<td>t_{a,b} (PO, LE)</td>
<td>E(time between first transm issions of Pac k et a and Pac k et b) defined as E[s_{b{\ell}} - s_{a{\ell}}]</td>
</tr>
<tr>
<td>T_{a,b} (PO, LE)</td>
<td>E(number of full timeout periods that will occur between s_{a{\ell}} and s_{b{\ell}}) defined as E[T_{a,b} (PO, LE) / t_{total}]</td>
</tr>
<tr>
<td>R (PO, LE)</td>
<td>E(memory utilization at POC Transport Receiver)</td>
</tr>
<tr>
<td>\eta (PO, LE)</td>
<td>efficiency of the system, a number between 0 and 1, defined as: \eta = \frac{\text{time spent on successful transmissions}}{\text{total time}}</td>
</tr>
</tbody>
</table>

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, pac k et a should be deliv ered to User Receiver before pac k et b (that is, in the partial order PO, a \prec b). Pac k ets a and b (and possibly other pac k ets) get lost several times before reaching the POC Transport Receiver. Pac k et a succeeds at its \text{ith} attempt while b succeeds at its \text{jth} attempt. Since the first successful arrival of pac k et b occurs before the first successful arrival of pac k et a, pac k et b is stored in a buffer at the POC Transport Receiver, “waiting” for pac k et a to arrive. After pac k et a’s arrival, pac k et b waits for the other depending pac k ets to arrive. Then the deliv ery of pac k et 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 Retran a,b (LE), expected number of retransmissions (of any pac k et) between the first transm issions of pac k ets a and b.

- Next, Retran a,b (LE) is used to compute F a,b (PO, LE), the expected number of full timeout periods between the initial transm issions of pac k ets a and b.
• Then, after computing buffering probabilities and times, i.e., \( p_{Buf_a,b}(PO,LE) \), \( Buf_a,b(PO,LE) \), \( p_{Buf}(PO,LE) \), and \( Buf(PO,LE) \), end-to-end packet delay is computed by using \( Buf(PO,LE) \).

• 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 \( p_{Buf_a,b}(PO,LE) \) 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: \( \text{retran}_{a,b} \)

\( \text{retran}_{a,b} \), the number of retransmissions (of any packet) in time period \([s_{a_1}, s_{b_1}]\), will be computed for \((a < b)\). 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 \( \text{Dist}_{a,b}(LE) + \text{retran}_{a,b} - 1 \). Out of \( \text{Dist}_{a,b}(LE) + \text{retran}_{a,b} - 1 \) transmissions, \( \text{retran}_{a,b} \) of them will result in a retransmission. Therefore:

\[
P[\text{retran}_{a,b} = i] = \binom{\text{Dist}_{a,b}(LE) + i - 1}{i} \times [p_{succ} \text{Dist}_{a,b}(LE) \times (1 - p_{succ})^{i - 1}] \quad (1)
\]

This is a negative binomial distribution with mean and variance:

\[
\text{retran}_{a,b}(LE) = \frac{1 - p_{succ}}{p_{succ}} \times \text{Dist}_{a,b}(LE) \quad \text{var}(\text{retran}_{a,b}(LE)) = \frac{1 - p_{succ}}{[p_{succ}]^2} \times \text{Dist}_{a,b}(LE) \quad (2)
\]

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 \( a \).

\[
F_{a,b}(PO,LE) = \frac{[\text{retrans}_{a,b}(LE) + \text{Dist}_{a,b}(LE)] \times t_{\text{pack}}}{t_{\text{cut}}} \\
= \frac{\text{Dist}_{a,b}(LE)}{B_{\text{Buf}} \times P_{\text{mcc}}}
\]

(3)

(4)

\( F_{a,b}(PO,LE) \) increases step-wise with \( \text{Dist}_{a,b}(LE) \), i.e., as \( \text{Dist}_{a,b}(LE) \) 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: \( p\text{Buf}_{a,b}(PO,LE) \)

\( p\text{Buf}_{a,b}(PO,LE) \) 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., \( (a < b) \). (If there is no dependency relation between \( a \) and \( b \), this value is zero by definition.)

\[
p\text{Buf}_{a,b}(PO,LE) = P(r_a > r_b) \]

(5)

\[
p\text{Buf}_{a,b}(PO,LE) = 1 - P(r_a \leq r_b) \]

(6)

\[
p\text{Buf}_{a,b}(PO,LE) = 1 - \sum_{j=1}^{\infty} (1 - p) \times p^{j-1} \times P(r_a \leq r_b | b \text{ succeeds at } j^{th \text{ attempt}}) \]

(7)

\[
p\text{Buf}_{a,b}(PO,LE) = \frac{p}{1 + p} \times p\text{Buf}_{a,1}(PO,LE) \]

(8)

![Figure 5: Relationship between \( p\text{Buf}_{a,b}(PO,LE) \) and \( \text{Dist}_{a,b}(LE) \)]

Thus, \( p\text{Buf}_{a,b}(PO,LE) \) decreases exponentially as \( F_{a,b}(PO,LE) \) increases. Since \( F_{a,b}(PO,LE) \) increases step-wise with \( \text{Dist}_{a,b}(LE) \), \( p\text{Buf}_{a,b}(PO,LE) \) decreases step-wise exponentially as \( \text{Dist}_{a,b}(LE) \) 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: \( B_{a,b}(PO,LE) \)

The value \( B_{a,b}(PO,LE) \), the expected time that packet \( b \) is buffered waiting for packet \( a \) to arrive, will be computed for \( (a < b) \). 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 \( s_a \) and \( s_b \), 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 \( b \).

\[
B_{a,b}(PO,LE) = E(r_a - r_b \text{ when } r_a > r_b)
\]
$$= \sum_{i=2}^{\infty} (1-p) \ast p^{i-1} \ast E(r_a - r_b \mid a \text{ succeeds at } i^{th} \text{ attempt})$$

$$= \left( \frac{[F_{a,b}(PO,LE) + 1] \ast p + F_{a,b}(PO,LE) \ast p^2 - 2 \ast F_{a,b}(PO,LE) \ast p^3}{1 \ast p^2} \right) \ast p^{F_{a,b}(PO,LE)} \ast t_{out}$$

$$+ \left( \frac{p^{F_{a,b}(PO,LE)+3}}{1 \ast p^2} - p^{F_{a,b}(PO,LE)+1} \right) \ast \left( \text{packet length } + \text{Dist}_{a,b}(LE) \right) \ast t_{pack,b}$$

(9) is a complicated expression. But if network is lossy and/or the \((\text{transmission time} / \text{propagation delay})\) ratio is large, then $F_{a,b}(PO,LE) \approx \frac{\text{Dist}_{a,b}(LE)}{\text{Buf}_a \ast \text{proc}}$ can be approximated with \(\frac{\text{Dist}_{a,b}(LE)}{\text{Buf}_a \ast \text{proc}}\), and expression (9) can be simplified to:

$$\text{Buf}_{a,b}(PO,LE) = \frac{p}{1 \ast p^2} \ast p^{F_{a,b}(PO,LE)} \ast t_{out} = p \text{Buf}_{a,b}(PO,LE) \ast \frac{t_{out}}{1 \ast p}$$

Figure 6: Relationship between $\text{Buf}_{a,b}(PO,LE)$ and $\text{Dist}_{a,b}(LE)$ when $F_{a,b}(PO,LE) \approx \frac{\text{Dist}_{a,b}(LE)}{\text{Buf}_a \ast \text{proc}}$.

$\text{Buf}_{a,b}(PO,LE)$ decreases with larger $\text{Dist}_{a,b}(LE)$ values (see Figure 6). When $F_{a,b}(PO,LE) \approx \frac{\text{Dist}_{a,b}(LE)}{\text{Buf}_a \ast \text{proc}}$, $\text{Buf}_{a,b}(PO,LE)$ decreases exponentially as $\text{Dist}_{a,b}(LE)$ increases. This means, in this special case, even a small increase in $\text{Dist}_{a,b}(LE)$ 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: $p\text{Buf}_a(PO,LE)$

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

- **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 $p\text{Buf}_a(PO,LE)$ is $P(a \text{ will be buffered at } \text{POC}_{\text{Transport}}_{\text{Receiver}})$ which is equivalent to $1 - P(a \text{ will not be buffered at } \text{POC}_{\text{Transport}}_{\text{Receiver}})$. Then:

$$p\text{Buf}_a(PO,LE) = 1 - \sum_{i=1}^{\infty} (1-p) \ast p^{i-1} \ast P(a \text{ will not be buffered at } \text{POC}_{\text{Transport}}_{\text{Receiver}} \mid a \text{ succeeds at } i^{th} \text{ attempt})$$

$$= 1 - \sum_{i=1}^{\infty} (1-p) \ast p^{i-1} \ast P(a \text{ arrives after all packets in } \text{before}_a(PO) \mid a \text{ succeeds at } i^{th} \text{ attempt}) \quad \text{[10]}$$

$p\text{Buf}_a(PO,LE)$ will be equivalent to the following values for the given cases:

- when $\text{before}_a(PO) = \{b\}$, then

---

2These sets are defined slightly differently in [5] with names $\downarrow a$ and $\uparrow a$: $\downarrow a = \text{before}_a(PO) \cup \{a\}$ and $\uparrow a = \text{after}_a(PO) \cup \{a\}$.
\[ p_{Buf_a}(PO,LE) = \frac{p}{1+p} \times p^{F_{1,a}(PO,LE)} = p_{Buf_{b,a}}(PO,LE) \]

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

\[ p_{Buf_{b,a}}(PO,LE) = \frac{p}{1+p} \times (p^{F_{1,a}(PO,LE) + F_{2,a}(PO,LE)}) - P(a \text{ arrives before } b_1 \text{ and } b_2) \]

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:

\[
p_{Buf_a}(PO,LE) \approx \sum_{\forall \; b \; \text{in} \; \text{before}_a(PO)} P[a \text{ would overtake } b] \tag{11}
\]

\[
\approx \sum_{\forall \; b \; \text{in} \; \text{before}_a(PO)} p_{Buf_{b,a}}(PO,LE) \tag{12}
\]

If the density of \(PO\) is small\(^3\) (i.e., relatively few ordering constraints), then \(|\text{before}_a(PO)|\) will tend to be small, leading to fewer terms in the expression for \(p_{Buf_{b,a}}(PO,LE)\) (see expression 12). \(\text{Dist}_{b,a}(LE)\) 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 \(p_{Buf_a}(PO,LE)\) values will tend to be smaller.

### 2.4.6 Expected buffering time for a packet: \(Bu_{fa}(PO,LE)\)

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

\[
Bu_{fa}(PO,LE) = E(\text{time that } a \text{ is buffered at POC_Transport_Receiver})
\]

\[
= \sum_{\forall \; b \; \text{in} \; \text{before}_a(PO)} E(r_b - r_a \mid b \text{ arrives after all packets in } \{a\} \cup \text{before}_a(PO) - \{b\})*
\]

\[
P[b \text{ arrives after all packets in } \{a\} \cup \text{before}_a(PO) - \{b\}] = \sum_{\forall \; b \; \text{in} \; \text{before}_a(PO)} \sum_{j=2}^{\infty} P(b \text{ succeeds at } j^{th} \text{ attempt}) *
\]

\[
E(r_b - r_a \mid b \text{ arrives after all packets in } \{a\} \cup \text{before}_a(PO) - \{b\} \text{ and } b \text{ succeeds at } j^{th} \text{ attempt})
\]

It is obvious that there is a dependency between \(Bu_{fa}(PO,LE)\) values and \(PO,LE\). 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 \(E(d_a - s_a)\). Therefore:

\[
T_{end-end_a}(PO,LE) = E(r_a - s_a) + d_a - r_a \tag{13}
\]

\[
= E(r_a - s_a) + Bu_{fa}(PO,LE) \tag{14}
\]

\[
= t_{pack} + t_{prop} + \frac{p}{1-p} * t_{out} + Bu_{fa}(PO,LE) \tag{15}
\]

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

---

\(^3\)The \textit{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 \(a < b\) 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 $E(d_{a.d} - s_{1,})$. Let us assume that packet $a$ is the last one to be delivered to User_Receiver. Then:

$$T_{msg}(PO, LE) = E(d_{a.d} - s_{1,}) + E(s_{1,} - s_{1,})$$

$$= (\text{Retran}_{1,a}(LE) + \text{Dist}_{1,a}(LE)) \ast t_{pack} + T_{end-end_d}(PO, LE)$$

$$= (\text{Retran}_{1,a}(LE) + \text{Dist}_{1,a}(LE)) \ast t_{pack} + t_{prop} + \frac{p}{1-p} \ast t_{out} + Buf_a(PO, LE)$$

We would like to use this expression to gain insight into the relationship between $T_{msg}(PO, LE)$ and $\text{Dist}_{1,a}(LE)$. However, it appears that mere inspection will not suffice. If $\text{Dist}_{1,a}(LE)$ increases, then the first term of the expression increases; however, the last term, $Buf_a(PO, LE)$, might decrease. The case where $\text{Dist}_{1,a}(LE)$ 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 \text{POC}_\text{Transport}_\text{Sender} is always $Buf_s$, and the expected time that a packet spends in a buffer of \text{POC}_\text{Transport}_\text{Sender} is $t_{out}/p_{succ}$. Therefore, by Little’s theorem:

$$Buf_s = \frac{\eta(PO, LE) \ast t_{out}}{t_{pack} \ast p_{succ}}$$

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

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

Interestingly, if $Buf_R = \infty$, then the system efficiency does not depend on $(PO, LE)$. From now on, we will refer to this efficiency value as $\eta_{\text{NET}}$, i.e., $\eta_{\text{NET}} = p_{\text{succ}} = (1-p) \ast (1-q)$. $\eta_{\text{NET}}$ 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:

$$T_{msg}(PO, LE) \simeq N \ast \frac{t_{pack}}{\eta_{\text{NET}}} + t_{prop}$$

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

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

$$\overline{R\_Buffered}(PO, LE) = \frac{\eta_{\text{NET}} \ast \sum_{i=1}^{N} Buf_i(PO, LE)}{t_{pack} \ast N}$$

This result shows that if $Buf_R = \infty$, then the dependency relationship between $\overline{R\_Buffered}(PO, LE)$ and $(PO, LE)$ is exactly same as that between $Buf_a(PO, LE)$ values and $(PO, LE)$. That is, the $(PO, LE)$ 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 $(PO, LE)$ 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 $(PO, LE)$ 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) transmission time/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


